UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Automatización de la señalización vial en México

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Ricardo Angeles Estrada

DIRECTOR DE TESIS

Ing. Orlando Zaldívar Zamorategui

Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado AUTOMATIZACION DE LA SEÑALIZACION VIAL EN MEXICO que presenté para obtener el titulo de INGENIERO EN COMPUTACIÓN es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

RICARDO ANGELES ESTRADA
Número de cuenta: 317014187

índice

1. Introducción

Este trabajo de tesis tiene como objetivo el desarrollo de una aplicación de software que ayude a los ingenieros civiles en México en el diseño y la elaboración de propuestas de señalización vial de los diferentes tramos carreteros del país. Las carreteras representan la forma de viajar más importante dentro de un país, y en México no es la excepción, siendo la Red Nacional de Caminos (RNC) un elemento fundamental en la infraestructura del país. En el año 2022, la RNC abarcaba 810,129.97 kilómetros de carreteras, caminos rurales, veredas y vialidades [3], cifra que aumentó a 836,603 kilómetros para el año 2023 [2]. Este sistema de carreteras y vías de comunicación es vital para la economía mexicana, facilitando el transporte de bienes y personas, promoviendo el turismo y mejorando la comunicación entre diferentes regiones del país.

La infraestructura vial es un elemento estratégico para el desarrollo de cualquier nación, no solo por su importancia en la movilidad de personas y bienes, sino también por su impacto en la economía, la seguridad y el ordenamiento territorial [6].

La expansión y mantenimiento de la infraestructura vial en México representan un desafío constante para las autoridades y los profesionales encargados del desarrollo de las propuestas. El adecuado diseño y señalización de estas vías son indispensables para garantizar la seguridad y la eficiencia del transporte. Sin embargo, el proceso de diseño y cálculo de señalización vial puede ser complejo y está sujeto a diversas regulaciones y normativas, lo que requiere tener cierta comprensión y conocimiento para una correcta aplicación de las normativas establecidas.

Con esto en mente, se planeó el desarrollo una aplicación de software que ayude a simplificar y agilizar el proceso de diseño y cálculo de la señalización vial en México, brindando a los ingenieros civiles una herramienta práctica y precisa para llevar a cabo estas tareas. Por ello, este proyecto se enfoca en la creación de una solución que permita realizar cálculos precisos y adecuados conforme a lo establecido en el "Manual de Señalización y Dispositivos para el Control del Tránsito en Calles y Carreteras" de la Dirección General de Servicios Técnicos, Ciudad de México, edición del año 2023, este manual es usado y aprobado por la Secretaría de Infraestructura, Comunicaciones y Transportes (SICT). Con lo anterior se busca estar contribuyendo a mejorar la seguridad y eficiencia en las vías de comunicación terrestre del país.

Los cálculos que serán abordados por la aplicación se dividen en dos categorías principales: cálculos horizontales y cálculos verticales. Entre los cálculos horizontales se incluyen la cuantificación de líneas de pintura, botones, rayas logarítmicas, flechas sobre pavimento, reductores de velocidad, pintura en pasos peatonales y boyas. En cuanto a los cálculos verticales, se abordan aspectos como la cuantificación individual y/o total de señales, delineadores, defensa metálica y delineadores en bifurcaciones, entre otros.

La aplicación es una herramienta de apoyo para los ingenieros civiles, permitiéndoles ingresar información específica centrada en los diferentes elementos requeridos para realizar cada cálculo, mientras que la aplicación se encarga de procesar estos datos y generar resultados precisos de acuerdo con los lineamientos del manual mencionado anteriormente.

Esta tesis busca contribuir a la ingeniería civil en México mediante el desarrollo de una herramienta informática que simplifique y agilice el proceso de cuantificación y cálculo de señalización vial, con el objetivo de mejorar la seguridad y eficiencia en las vías de comunicación terrestre del país. Para desarrollar esta aplicación, se utilizó el lenguaje de programación de Java. Java ofrece una amplia gama de bibliotecas y frameworks que pueden facilitar el desarrollo de la aplicación, así como una sólida portabilidad que permite que la aplicación pueda ejecutarse en diferentes sistemas operativos sin necesidad de reescribir o modificar el código. Además, Java se basa en el paradigma de programación orientada a objetos (POO), lo que permitió estructurar el código de manera modular y reutilizable, facilitando su mantenimiento y futuras expansiones o actualizaciones. [5]

Problemática

El desarrollo de proyectos de señalización vial en México requiere realizar múltiples cálculos, cuantificaciones y procesos de organización de información conforme a los lineamientos establecidos en el *Manual de Señalización y Dispositivos para el Control del Tránsito en Calles y Carreteras* de la Secretaría de Infraestructura, Comunicaciones y Transportes (SICT). Estos procesos suelen llevarse a cabo de manera manual o mediante herramientas genéricas como hojas de cálculo, lo que puede provocar retrasos, errores de captura, inconsistencias en los resultados y dificultades para mantener organizada la información del proyecto.

Además, la generación de tablas y cuantificaciones para elementos como señales horizontales, señales verticales, defensas metálicas, botones y dispositivos de seguridad implica repetir procesos operativos que consumen tiempo y requieren una revisión constante por parte del ingeniero responsable. Además, hacer estas tablas en otras aplicaciones o por otros métodos, puede representar otra problemática, debido a que al momento de importar las tablas a AutoCAD estas tablas pierden el formato de líneas, anchos, tipo de letra, entre otros aspectos visuales. Esta situación aumenta la complejidad del trabajo y dificulta la elaboración eficiente de propuestas de señalización vial.

Otro aspecto importante es que muchas herramientas existentes se enfocan únicamente en el dibujo técnico dentro de plataformas CAD, sin integrar funciones específicas para la automatización de cálculos, almacenamiento estructurado de información y generación de tablas conforme a normativas mexicanas. Como consecuencia, el ingeniero debe dividir el trabajo entre diferentes programas y procesos manuales.

Propuesta de solución

Para atender esta problemática, se propone el desarrollo de una aplicación de software orientada al apoyo de ingenieros civiles en la cuantificación y generación de propuestas de señalización vial. La aplicación busca automatizar procesos relacionados con el cálculo de elementos de señalización horizontal y vertical, así como facilitar el almacenamiento, organización y exportación de información técnica.

La solución desarrollada permite:

- Registrar proyectos de señalización vial.
- Capturar información técnica mediante módulos especializados.
- Automatizar cálculos de cuantificación.
- Generar tablas organizadas conforme a lineamientos oficiales.
- Exportar resultados hacia AutoCAD.
- Mantener la información estructurada mediante una base de datos relacional.

El sistema fue desarrollado utilizando Java y una arquitectura modular, permitiendo integrar formularios, tablas dinámicas y procesos automatizados que facilitan el flujo de trabajo del usuario.

Con esta propuesta se busca reducir tiempos de trabajo, minimizar errores humanos y mejorar la organización de la información relacionada con proyectos de señalización vial.

2. Señalización Vial en México.

Antecedentes Internacionales

La evolución de la señalización vial en México ha sido influenciada por una combinación de antecedentes internacionales y desarrollos nacionales. A nivel internacional, los Congresos Panamericanos y de Turismo subrayaron la importancia de estandarizar la señalización vial en todo el continente, reconociendo la necesidad de normas comunes para mejorar la seguridad y la movilidad entre países. Estos congresos sirvieron como foro para discutir y promover la adopción de prácticas uniformes en la señalización vial, con el objetivo de reducir accidentes y facilitar el flujo de tráfico en las diferentes naciones. [4].

En la Conferencia de Transporte Vial de las Naciones Unidas en 1949, se abordó la unificación del sistema de señalización vial a nivel global, buscando establecer normas y símbolos reconocibles internacionalmente para mejorar la eficiencia en las carreteras. Los representantes de varios países colaboraron para identificar áreas de cooperación y desarrollar un marco común que pudiera ser implementado en diferentes contextos geográficos y culturales. Esto estableció los cimientos para futuras convenciones y acuerdos internacionales relacionados con la señalización vial. La Convención sobre Circulación Vial en 1968 representó un avance significativo al reconciliar los sistemas de señalización americano y europeo, estableciendo estándares que permitieron una mayor armonización y comprensión de las señales viales en diferentes regiones del mundo. Esta convención fue el resultado de años de negociaciones y cooperación entre los países participantes, y marcó un hito en el desarrollo de una señalización vial globalmente coherente y efectiva. [4]

Además, es importante reconocer que el desarrollo del transporte y de la infraestructura vial a nivel mundial también influyó indirectamente en la evolución de la señalización. Desde la aparición de las primeras carretas en América en el siglo XVI, hasta la consolidación del ferrocarril en el siglo XIX y el crecimiento acelerado del tránsito motorizado en el siglo XX, las crecientes necesidades de movilidad impulsaron la necesidad de ordenar y regular el flujo vehicular mediante señales más claras y visibles. Con el surgimiento de los Sistemas Inteligentes de Transporte en las últimas

décadas, se han incorporado tecnologías avanzadas como el reconocimiento automático de placas, la gestión de tráfico en tiempo real y los sistemas satelitales de guiado, todos los cuales requieren una señalización estandarizada para funcionar adecuadamente [6].

Desarrollo Nacional

A nivel nacional, México adoptó el sistema propuesto por la ONU desde 1957, aunque con ciertas adaptaciones para reflejar las particularidades del país, como el tráfico local y las características geográficas. La adopción de un sistema de señalización vial internacionalmente reconocido permitió a México alinearse con las prácticas y estándares internacionales, facilitando la navegación de conductores nacionales y extranjeros en las carreteras mexicanas.

Se formaron comités encargados de crear manuales de señalización vial específicos para México, proporcionando directrices claras sobre el diseño, la colocación y el significado de las señales viales. Estos manuales surgieron como respuesta a la necesidad de contar con estándares nacionales y facilitaron la implementación de una señalización coherente en todo el país. Además, se establecieron procesos de revisión periódica para mantener actualizados los manuales y adaptar la señalización a los cambios en la infraestructura vial, los avances tecnológicos y las tendencias en seguridad vial. Esta revisión constante aseguró que la señalización vial en México estuviera siempre en línea con las mejores prácticas y necesidades locales, contribuyendo a la seguridad y eficiencia del sistema de transporte del país.

Cabe mencionar que el crecimiento del parque vehicular en el país, especialmente a lo largo del siglo XX, exigió una mayor profesionalización en la planeación del tránsito y en la implementación de medidas de control como la señalización, las cuales se vieron fortalecidas con la adopción de tecnologías inteligentes y estrategias de gestión más modernas. [6]
En conjunto, la cooperación internacional y los esfuerzos nacionales han contribuido significativamente a la evolución de la señalización vial en México, mejorando la seguridad y la eficiencia en las carreteras del país. [1][6].

2.1. Manual de Señalización Vial

La elaboración del manual se llevó a cabo en cumplimiento con las normativas oficiales, lo que representa un esfuerzo coordinado entre diferentes instituciones gubernamentales y organismos

relacionados con el tránsito y la seguridad vial. Estas instituciones deben garantizar que el manual esté respaldado por la autoridad correspondiente y que cumpla con los estándares establecidos a nivel nacional. Además, se ha ampliado los criterios de aplicación del manual para incluir calles urbanas. Esta expansión responde a la creciente importancia de la seguridad y accesibilidad en entornos urbanos, donde la interacción entre peatones y vehículos es más intensa. A continuación, se explican los diversos capítulos que contiene el manual de señalización y que es lo que se explica en cada uno.

El manual comienza con una sección de generalidades de la señalización y uso de dispositivos, donde se proporciona una visión general de los elementos de señalización y dispositivos viales. Aquí se incluyen normativas, criterios y procedimientos para su instalación y mantenimiento, destacando la importancia de la capacitación del personal involucrado en el manejo del tránsito y la disponibilidad del Banco Digital de Señalización Vial como herramienta de apoyo.

Después se explica la señalización horizontal detallando las diferentes marcas y aplicaciones en función de guiar y regular el flujo de tráfico en las vías. La señalización vertical es otro componente fundamental descrito en el manual. Este capítulo clasifica y describe los diversos tipos de señales verticales utilizadas en la vialidad, incluyendo señales preventivas, restrictivas, informativas, turísticas y de servicios. Además, se aborda la estructura de soporte para estas señales y las restricciones relacionadas con la publicidad. En la sección de dispositivos diversos, se presentan los dispositivos destinados a la protección y canalización del tránsito, como indicadores de alineamiento, balizas, reductores de velocidad y barreras de protección, entre otros.

Otro aspecto cubierto en el manual es la señalización en zonas de obras viales. Se describen los elementos temporales utilizados en obras viales para garantizar la seguridad de los usuarios y del personal de construcción, cubriendo aspectos como la señalización horizontal y vertical, así como los dispositivos de canalización. En cuanto a los semáforos y dispositivos electrónicos, en este capítulo se explora soluciones tecnológicas para mejorar la seguridad vial, incluyendo la instalación y operación de semáforos y otros dispositivos electrónicos complementarios, que contribuyen a regular el tránsito de manera eficiente.

El manual también dedica una sección a la señalización para orientación peatonal y ciclista, proporcionando orientación sobre los dispositivos destinados a mejorar la seguridad y comodidad de los peatones y ciclistas, facilitando su movilidad en el entorno urbano y vial. Además, se abordan aplicaciones específicas, proporcionando ejemplos concretos de situaciones donde se aplican ciertas señales y dispositivos especiales para garantizar la seguridad y fluidez del tránsito.

Finalmente, el manual incluye apéndices que contienen definiciones, instructivos y detalles técnicos adicionales, así como información sobre el Banco Digital de Señalización Vial y la Tipografía de México utilizada en la señalización vial mexicana. Estos recursos complementarios facilitan la comprensión y aplicación de las normativas establecidas en el manual. En conjunto, este manual representa una herramienta integral y actualizada que contribuye a la mejora de la seguridad y eficiencia del sistema de transporte vial en México.

2.2. Revisión de planos Carreteros

La revisión corresponde al ingeniero designado y autorizado por la Secretaría de Comunicaciones y Transportes (Dirección Coordinadora de Proyectos Técnicos), en conformidad con el Manual de Señalización y Dispositivos para el Control de Tránsito en Calles y Carreteras 2023, así como las normativas NOM-037-SCT2-2020 referente a barreras de protección en carreteras y vialidades urbanas, NOM-008-SCT2-2020 sobre amortiguadores de impacto en carreteras y vialidades urbanas, y la norma para la elaboración de señalamiento y dispositivos para protección en zonas de obras viales, NOM-086-SCT2-2023. Además, se considerará la NOM-034-SCT2/SEDATU-2022 sobre Señalización y Dispositivos Viales para calles y carreteras.

La revisión implica verificar que el señalamiento desarrollado esté en consonancia con las especificaciones de las bases de licitación y las últimas normativas autorizadas por la SICT mencionadas anteriormente. Para ello, se examinará que el señalamiento horizontal cumpla con las especificaciones del manual, incluyendo el tipo de línea correspondiente al camino, sus dimensiones y colores específicos, así como la correcta disposición de botones con sus respectivas características, todo ajustado al tipo de vía. Es esencial que el plano presentado incluya un cuadro resumen detallando los tipos de líneas utilizadas, con información sobre el tipo de línea, color, dimensión, cantidad y observaciones pertinentes.

Además, se requerirá la presentación de cuadros que indiquen las cantidades totales de dispositivos, como botones, con detalles tales como clave, color, rayado, ubicación, características y cantidad correspondiente. Se adjuntarán cuadros previamente autorizados y revisados por la misma dependencia para su debida referencia y validación.

SEÑALAMIENTO HORIZONTAL					
NUEVA CLAVE	COLOR	DIMENSION	CANT.		OBSERVACIONES
M-2.1	BLANCO	150mm. DISC.	592	ml.	RAYA SEPARADORA DE CARRILES, CONTINUA SENCILLA
M-2.3	BLANCO	150mm. DISC.	850	ml.	RAYA SEPARADORA DE CARRILES, DISCONTINUA
M-3.1	BLANCO	150mm. CONT.	1,140	ml.	RAYA EN LA ORILLA DERECHA, CONTINUA
M-3.2	BLANCO	150mm. CONT.	1,029	ml.	RAYA EN LA ORILLA DERECHA, DISCONTINUA
M-3.3	AMARILLO	150mm. CONT.	3,680	ml.	RAYA EN LA ORILLA IZQUIERDA
M-9	BLANCO	600 mm. CONT.	640	ml.	RAYA LOGARITMICA
M-11.1	BLANCO		10	pzas.	FLECHAS SOBRE PAVIMENTO
M-15	AMARILLO/NEGRO		2	pzas.	REDUCTOR DE VELOCIDAD

	TIPO DE DISPOSITIVOS		
DISPOSITIVOS	CLAVE	CANTIDAD	UNIDAD
MENSULA REFLEAJANTE SOBRE BARRERA METALICA	DH-3	396	pzas.
MENSULA REFLEAJANTE SOBRE BARRERA DE CONCRETO (TIPO NEW YERSEY)	DH-3	360	pzas.
MALLA ANTIDESLUMBRANTE	OD-12	1,800	ml.

TIPO DE DISPOSITIVOS					
BOTONES					
CLAVE	COLOR	RAYA	UBICACIÓN	CARACTERISTICAS	CANTIDAD
DH-1.9	BLANCO	M-2.3	A CADA 30 m.	UNA CARA	85 pzas.
DH-1.11	BLANCO	M-3.1	A CADA 30 m.	UNA CARA	38 pzas.
DH-1.11	BLANCO	M-3.2	A CADA 32 m.	UNA CARA	64 pzas.
DH-1.14	AMARILLO	M-3.3	A CADA 30 m.	UNA CARA	120 pzas.
DH- (totales)	BLANCO	M-9		UNA CARA	384 pzas.

Figura 2.1. Cuadros autorizados y revisados

La revisión también constará del señalamiento vertical, donde tendrá las especificaciones de fabricación y materiales de las señales.

Su revisión consta de observar que todo se cumpla conforme a las normas, teniendo los cuadros que contenga la cantidad, ubicación, el tipo de señal, dimensión y descripción de la señal, así también tendrá un cuadro de resumen total, donde este tendrá el tipo de señal, dimensión cantidad y descripción de las señales.

CARRETERA: TAMAZUNCHALE - CD VALLES				
CANT	UBICACIÓN	SEÑAL	DIMENSION	DESCRIPCION
2	326+610	SR-9	117X117	VELOCIDAD
2	326+480	SR-13	117X117	CAMIONES CARGA CARRIL DERECHO
2	326+400	SP-6	117X117	CURVA
2	326+170	SP-6	117X117	CURVA
2	326+090	SP-41	117X117	REDUCTOR DE VELOCIDAD (T.A. 35X152)
2	326+020	SP-32	117X117	PEATONAL
2	326+000	SII-15	30X76	KILOMETRAJE DE RUTA
2	325+940	SR-9	117X117	VELOCIDAD
1	325+800	SID-9	56X300	SEÑAL BAJA 2 TABLEROS
1	325+725	SIR-	56X300	SEÑAL INFORMACION RECOMENDACION
1	325+670	SID-13	122X366	BANDERA
1	325+670	OD-5	30X122	SEÑAL OBSTACULO
2	325+650	SP-6	117X117	CURVA
1	325+560	SID-11	56X300	SEÑAL CONFIRMATIVA 1 TABLERO
2	325+260	SR-34	117X117	UTILICE SU CINTURON DE SEGURIDAD

CUANTIFICACION TOTAL CARRETERA: TAMAZUNCHALE - CD VALLES			
SEÑAL	DIMENSION	CANT.	DESCRIPCION
SP-6	117X117	12	CURVA
SP-8	117X117	2	CURVA INVERSA
SP-32	117X117	4	PEATONAL
SP-41	117X117	6	REDUCTOR DE VELOCIDAD (T.A. 35X152)
SR-9	117X117	10	VELOCIDAD
SR-13	117X117	2	CAMIONES CARGA CARRIL DERECHO
SR-34	117X117	4	UTILICE SU CINTURON DE SEGURIDAD
SII-15	30X76	4	KILOMETRAJE DE RUTA
SIR-	56X300	2	SEÑAL INFORMACION RECOMENDACION
SID-9	56X300	2	SEÑAL BAJA 2 TABLEROS
SID-11	56X300	2	SEÑAL CONFIRMATIVA 1 TABLERO
SID-13	122X366	2	BANDERA
OD-5	30X122	2	SEÑAL OBSTACULO
OD-6	VER DETALLE	16	DELINEADORES TOTALES
OD-11	76X90	39	INDICADOR DE CURVA CERRADA

Figura 2.2. Cuadros autorizados para revisión

También se contará con tablas para diversos dispositivos, conforme a la norma de dispositivos de protección, como barrera metálica, barrera central de concreto, amortiguadores de impacto, secciones extremas de amortiguamiento (NOM-037-SCT2-2020, NOM-008-SCT2-2020).

TIPO DE DISPOSITIVOS			
BARRERAS			
CLAVE	POSICION	CANTIDAD	CARACTERISTICAS
OD-4.1.2 (NC-3)		1,980 ml.	BARRERA METALICA 3 CRESTAS
OD-4.1.3 (NC-3)		1,800 ml.	BARRERA CONCRETO NEW JERSEY
OD-4.4.1 (NC-3)		11 pzas.	SECCION EXTREMA AMORTIGUAMIENTO
OD-4.4.1 (NC-3)		2 pzas.	AMORTIGUADOR DE IMPACTO
OD-4.4.2/S (NC-3)		10 pzas.	SECCION TERMINAL COLA DE PATO

Figura 2.3. Cuadros autorizados para dispositivos

El proyecto también constara con el plano de protección de obra, esta revisión constara de la última norma NOM-086-SCT2-2023 dispositivos para protección en zonas de obras viales.

El plano constará con las señales autorizadas de la misma norma autorizada, contendrá los cuadros y los detalles contando con la clave, el tipo de señal, color, altura, dimensión y cantidad.

Así también contara con un cuadro sobre su cálculo sobre el área de protección, transición, y zona de información, así también contara con el cuadro de fabricación y materiales que se utilizara para el desarrollo de las señales utilizadas.

CANTIDADES DE OBRA

CLAVE	SEÑAL	COLOR	ALTURA TIPOGRAFICA	DIM.	CANT.
SIP-7		NARANJA	17	71X178	3
SIP-8		NARANJA	15	71X178	2
SIP-9		NARANJA	15	71X178	1

Figura 2.4. Cuadros autorizados para Señales

La secretaria dará el visto bueno, si el proyecto cumple con la norma, así la empresa después de autorizar el proyecto, la empresa, debe de imprimir al tipo de papel autorizado y a la escala permitida.

2.3. Funcionamiento de la Aplicación de Señalización Vial

En base a todo lo anterior con la aplicación debe permitir al usuario crear un nuevo proyecto o ver uno anterior. Cada vez que se inicia un nuevo proyecto, se despliega una ventana donde se solicitan datos básicos del proyecto, como la velocidad del proyecto, el nombre del proyecto y el tipo de camino. Estos datos son fijos para todo el proyecto y son necesarios para realizar ciertos cálculos. Una vez ingresada esta información, se muestran botones para que el usuario pueda seleccionar el cálculo a realizar, ya sea cuantificar las señales horizontales, señales verticales, el cálculo de la defensa o de dispositivos. Al seleccionar una de estas opciones, se despliega una ventana donde se muestran botones que permiten realizar todos los cálculos correspondientes a cada elemento, por ejemplo, para la señalización horizontal la ventana emergente muestra botones para los 20 tipos de

marcas existentes a la señalización horizontal, cada botones abre una ventana que permite realizar los cálculos de la marca seleccionada. Una vez completados todos estos cálculos, la aplicación permite generar un archivo de texto que puede ser leído por AutoCAD, por medio de un archivo .LIPS, generando una tabla editable de los cuadros de cuantificación y totales de las señales calculadas en el programa.

Dentro del manual de señalización, se tienen dos apartados importantes en los que se dividen la señalización vial, Señales Horizontales y Señales Verticales. En los siguientes apartados, se explicarán a detalle las señales que se resuelven dentro de la aplicación, así como las fórmulas utilizadas para realizar los cálculos usados en la aplicación, toda la información que se muestra a continuación proviene del manual de señalización vial [1]. En el caso de las fórmulas, aunque no necesariamente están todas incluidas explícitamente en el manual, se han desarrollado basándose en la interpretación de las explicaciones y de las tablas expuestas dentro del manual.

2.3.1. Señalización Horizontal

Corresponde al conjunto de marcas y dispositivos que se pintan o colocan sobre el pavimento, guarniciones y estructuras con el propósito de delimitar las características geométricas de las calles y carreteras. Sirve también para describir todos aquellos elementos estructurales que estén instalados dentro del derecho de vía, para regular y canalizar el tránsito de peatones y vehículos, así como proporcionar información a los usuarios. Estas marcas y dispositivos son: rayas, símbolos, leyendas, botones reflejantes o delimitadores.

Las marcas y dispositivos para la señalización horizontal, según su uso, se clasifican como se muestra en la Tabla 1.

Clasificación	**Nombre**
M-1	Raya separadora de sentidos de circulación
M-2	Raya separadora de carriles
M-3	Raya en la orilla del arroyo vial
M-4	Rayas de trayectorias en intersecciones
M-5	Rayas canalizadoras

M-6	Raya de alto
M-7	Rayas para cruce de peatones
M-8	Marcas para cruce de ferrocarril
M-9	Rayas con espaciamiento logarítmico
M-10	Marcas para estacionamiento
M-11	Rayas, símbolos y leyendas para regular el uso de carriles
M-12	Marcas en guarniciones
M-13	Marcas en estructuras y objetos adyacentes a la superficie de rodadura
M-14	Marca de emergencia para frenado
M-15	Marcas para vías ciclistas
M-16	Marcas temporales
M-17	Marca de área de espera para vehículos no motorizados y motocicletas
M-18	Marca de ceda el paso
M-19	Marcas para indicar prohibiciones
M-20	Marcas para identificar reductores de velocidad

Tabla 1. Clasificación de las marcas para la señalización horizontal

2.3.1.1. Rayas De Pintura

Como se visualiza en la Tabla 1 para las rayas tenemos diversas clasificaciones de Marcas sobre pavimento (M) y estas a su vez tienen subdivisiones según el tipo de raya que se va a utilizar, entonces para el cálculo de pintura de cada tipo de raya tenemos

- ***Raya separadora de sentidos de circulación (M-1)***

  Se usa para separar los sentidos de circulación vehicular en calles y carreteras de dos sentidos, tanto en tramos rectos como en curvas. Debe ser amarilla y reflectante, y se complementa con botones reflectantes. El ancho de la raya debe ser el especificado dependiendo del tipo de calle o carretera.

  - Raya continua sencilla (M-1.1)

    La raya continua simple (M-1.1) se utiliza en tramos donde la visibilidad para el adelantamiento en ambas direcciones es insuficiente o en tramos donde adelantar esté prohibido por razones de seguridad.

    El cálculo sería:

$$Raya\ M-1.1[m^2] = D\ [m]\ x\ A\_L\ [m]$$

Donde:

- D: Es la distancia total en metros que va recorrer la raya.
- A_L: Es el ancho de la línea que se le da según la siguiente tabla.

Tipo de calle o carretera	Ancho de la raya [cm]
Carretera de dos o más carriles por sentido de circulación	15
Carretera con un carril por sentido de circulación	10
Calle	10
Vía ciclista	10

Tabla 2. Ancho de camino

- Raya discontinua sencilla (M-1.2)

Se emplea como se muestra en aquellos tramos donde, para ambos sentidos de circulación, la distancia de visibilidad es igual o mayor que la necesaria para el rebase.

El cálculo sería:

$$Raya\ M-1.2[m^2] = \frac{D\ [m]}{L\_S\ \ [m] + S\_S\ [m]}\ *L\_S[m]\ *\ A\_L[m]$$

Donde:

- **D:** Es la distancia total en metros que va recorrer la raya
- **A_L:** Es el ancho de la línea, tomar valor de Tabla 2
- **L_S:** Es la longitud de los segmentos [m]
- **S_S:** Es la separación de los segmentos [m]

Tanto para L_S y S_S se obtienen según el tipo de vía a utilizar, según la siguiente tabla:

Tipo de vía	Longitud de los segmentos(L_S)	Separación entre segmentos(S_S)
Carretera	5	10
Calle con velocidad mayor a 50 km/h	5	10
Calle con velocidades de hasta 50 km/h	2.5	5
Vía ciclista	1	2

Tabla 3. Longitud y separación de los segmentos

- Raya continua-discontinua (M-1.3)

  Se utiliza en tramos donde la visibilidad disponible permite que un automóvil rebase, pero solo desde uno de los sentidos. La línea del lado del carril donde se permite el rebase debe ser discontinua. La separación transversal entre la raya continua y la discontinua es igual al ancho de línea según la Tabla 2.

  El cálculo sería:

$$Raya\ M-1.3[m^2] = \left(\frac{D[m]}{L\_S[m]+S\_S[m]} * L\_S[m] * A\_L[m]\right) + (D\ [m]x\ A\_L[m])$$

  Donde:

  - **D:** Es la distancia total en metros que va recorrer la raya
  - **A_L:** Es el ancho de la línea, tomar valor de Tabla 2.
  - **L_S:** Es la longitud de los segmentos [m]
  - **S_S:** Es la separación de los segmentos [m]

- ***Raya separadora de carriles (M-2)***

  Se utiliza para delimitar los carriles del mismo sentido de circulación en calles y carreteras de dos o más carriles por sentido, así como para delimitar carriles especiales para giros y carriles exclusivos para ciertos tipos de vehículos. Debe ser blanca y reflectante, dependiendo del tipo de vía. Esta raya debe complementarse con botones reflectantes. Puede

ser discontinua si se permite cruzarla, o continua o continua doble cuando está prohibido su cruce

- Raya separadora de carriles, discontinua (M-2.3)

  Cuando se permita cambiar de carril, la línea debe ser discontinua. Para carreteras o calles con velocidad superior a 50 km/h, los segmentos deben ser de 5 m y estar separados entre sí por 10 m. En calles con velocidad permitida de hasta 50 km/h, los segmentos deben ser de 2,5 m y estar separados por 5 m. Para vías ciclistas, los segmentos deben ser de 1 m y estar separados por 2 m. como se muestra en la Tabla 3.

  El cálculo sería:

$$Raya\ M-2.3[m^2] = \frac{D[m]}{L\_S[m] + S\_S[m]} * L\_S\ [m] *\ A\_L[m]$$

  Donde:

  - **D:** Es la distancia total en metros que va recorrer la raya.
  - **A_L:** Es el ancho de la línea, tomar valor de Tabla 2.
  - **L_S:** Es la longitud de los segmentos, tomar valor de Tabla 3.
  - **S_S:** Es la separación de los segmentos, tomar valor de Tabla 3.

  Y si tenemos velocidad menor o igual a 50 km/h quedaría:

$$Raya\ M-2.3[m^2] = \frac{D[m]}{7.5} * 2.5\ *\ A\_L[m]$$

  Y si tenemos velocidad mayor a 50 km/h quedaría:

$$Raya\ M-2.3[m^2] = \frac{D[m]}{15} * 5\ *\ A\_L[m]$$

- ***Raya de trayectorias en intersecciones (M-4)***

  Se utiliza para delimitar la zona de transición entre los carriles de tránsito directo y el de cambio de velocidad en las entradas y salidas, o para conectar los extremos de los enlaces, así como para indicar las trayectorias dentro de una intersección para vehículos en general y para marcar las trayectorias de los carriles exclusivos de transporte público en intersecciones de calles y accesos a predios. La utilización de las rayas en las intersecciones está condicionada al uso de otro tipo de rayas, como las rayas para prohibición de parar en intersección.

- Raya para entradas y salidas (M-4.1)

  Se alinea a la raya de la orilla del arroyo vial cuando se genera un carril de aceleración o desaceleración.

  El cálculo sería:

$$Raya\ M-4.1[m^2] = \frac{D[m]}{L\_S[m] + S\_S[m]} * L\_S\ [m] * \ A\_L[m]$$

  Donde:

  - **D:** Es la distancia total en Km que va recorrer la raya
  - **A_L:** Es el ancho de la línea, tomar valor de Tabla 2.
  - **L_S:** Es la longitud de los segmentos
  - **S_S:** Es la separación de los segmentos

  En este caso para L_S y S_S se obtienen según el tipo de vía a utilizar, según la siguiente tabla:

Tipo de raya	Tipo de vía	Longitud del segmento [m]	Separación entre segmentos [m]
Raya para entradas y salidas	Carretera	2	4
Raya para entradas y salidas	Calle de circulación continua	2	4
Raya para trayectorias dentro de una intersección	Calle	1	2
Raya para trayectoria de transporte público de pasajeros	Calle	1	2

Tabla 4. Longitud y separación de los segmentos en intersecciones

2.3.1.2. Rayas Logarítmicas

Se utilizan en calles y carreteras, generalmente antes de cruces peatonales, cruces con vías férreas, zonas escolares, o donde se necesite reducir la velocidad de los vehículos, creando la ilusión óptica y auditiva de que el vehículo se está acelerando al conductor.

Deben ser blancas y reflectantes, con un ancho de 60 cm, colocadas transversalmente al eje de la vía en el sentido del tráfico, abarcando todo el ancho de la carretera, incluyendo los acotamientos. Las rayas deben tener una altura resaltada de 3 a 5 milímetros. En calles, estas rayas también pueden ser complementadas con Botones alertadores (BT).

La longitud total de la zona a marcar, el número de rayas y su separación deben determinarse, según la diferencia entre la velocidad requerida para la restricción y la velocidad proyectada en el caso de una nueva calle o carretera, o la velocidad de operación en una calle o carretera existente.

Para ello se utiliza la siguiente tabla mostrada en la siguiente figura:

Separación entre rayas (m)	Diferencia de velocidades (km/h) / Número de líneas requeridas						
	20 / 13	30 / 20	40 / 26	50 / 32	60 / 38	70 / 44	80 / 51
	15.25	15.25	15.25	15.25	15.25	15.25	15.25
	11.75	12.55	13.10	13.50	13.70	13.90	14.05
	9.55	10.70	11.50	12.05	12.50	12.80	13.05
	8.05	9.30	10.25	10.90	11.45	11.85	12.15
	6.95	8.25	9.25	10.00	10.60	11.05	11.40
	6.10	7.40	8.40	9.20	9.80	10.30	10.70
	5.50	6.70	7.70	8.50	9.15	9.70	10.10
	4.95	6.10	7.15	7.95	8.60	9.15	9.60
	4.50	5.65	6.60	7.40	8.10	8.65	9.10
	4.15	5.25	6.20	7.00	7.65	8.20	8.65
	3.85	4.85	5.80	6.60	7.25	7.80	8.25
	3.55	4.55	5.45	6.25	6.90	7.45	7.90
		4.30	5.15	5.90	6.55	7.10	7.55
		4.05	4.90	5.60	6.25	6.80	7.25
		3.85	4.65	5.35	6.00	6.55	7.00
		3.65	4.45	5.10	5.75	6.30	6.75
		3.45	4.25	4.90	5.50	6.05	6.50
		3.30	4.05	4.70	5.30	5.80	6.25
		3.15	3.90	4.50	5.10	5.60	6.05
			3.75	4.35	4.90	5.40	5.85
			3.60	4.20	4.75	5.25	5.65
			3.45	4.05	4.60	5.10	5.50
			3.30	3.90	4.45	4.95	5.35
			3.20	3.75	4.30	4.80	5.20
			3.10	3.65	4.20	4.65	5.05
				3.55	4.10	4.50	4.90
				3.45	4.00	4.35	4.75
				3.35	3.90	4.25	4.65
				3.25	3.80	4.15	4.55
				3.15	3.70	4.05	4.45
				3.10	3.60	3.95	4.35
					3.50	3.85	4.25
					3.40	3.75	4.15
					3.30	3.65	4.05
					3.20	3.55	3.95
					3.10	3.45	3.85
					3.05	3.35	3.75
						3.30	3.65
						3.25	3.55
						3.20	3.45
						3.15	3.40
						3.10	3.35
						3.05	3.30
							3.25
							3.20
							3.15
							3.10
							3.05
							3.00
							2.95
Σ1	84.15	122.30	158.40	194.40	231.25	266.35	304.20
Σ2	91.95	134.30	174.00	213.60	254.05	292.75	334.80

∑1 = Longitud de espaciamiento
∑2 = Longitud total (espaciamiento + anchura de la raya)

Figura 2.5.1. Separación entre rayas con espaciamiento logarítmico

2.3.1.3. Flechas Sobre Pavimento

En intersecciones, las flechas reflejantes guían los movimientos permitidos desde carriles específicos y regulan el tráfico. Estas señales deben repetirse a intervalos adecuados antes de la intersección para que los conductores elijan su carril con anticipación.

Las flechas deben alargarse en la dirección del tráfico para una mejor percepción, adaptándose a la velocidad de la vía, según la Figura 2.3.1.2. Al acercarse a una intersección, las flechas se pintan 2 m antes de la línea de alto, de ceda el paso, del área de espera para no motorizados o de las indicaciones para peatones, orientadas en la dirección del tráfico. Después de la intersección, se pintan 2 m antes del cruce peatonal, también en la dirección del tráfico.

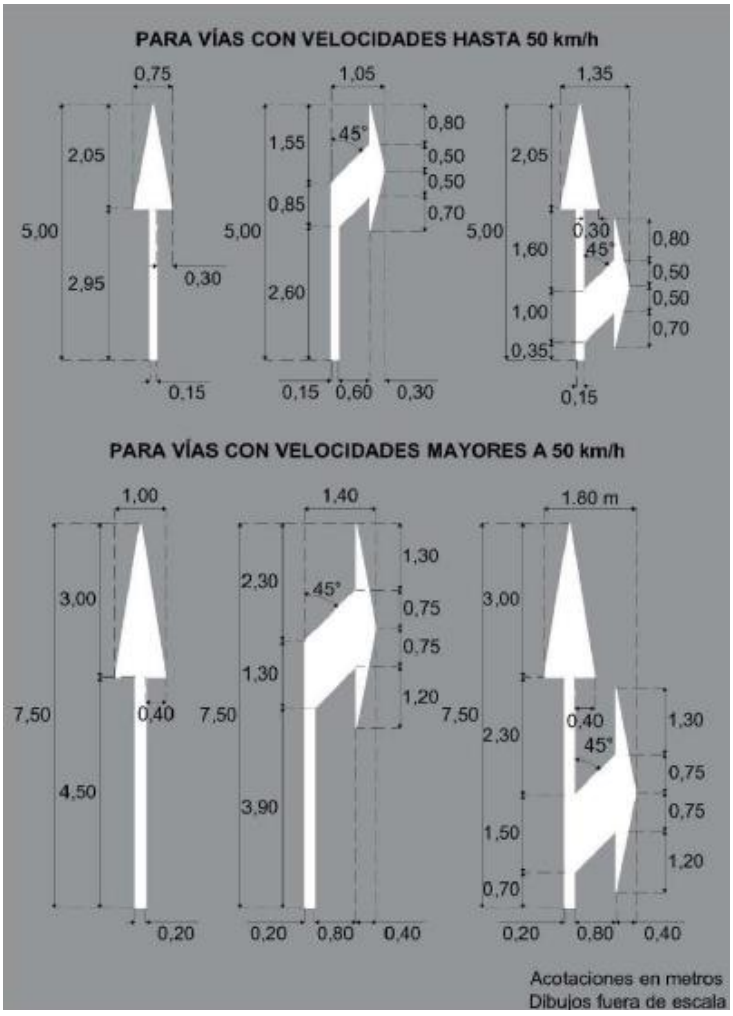


Figura 2.3.1.2.6. Flechas sobre carriles

Basándonos en la Figura 2.3.1.2, en orden de izquierda derecha, tenemos Flecha Frente, Flecha Vuelta y Flecha Frente Vuelta, y tomando en cuenta las medidas por flecha según la velocidad del camino, entonces el cálculo de pintura por flecha sería:

$$Pintura = A\_F[m^2] \times Cantidad$$

Donde:

- A_F: Es el área de la Flecha, que se obtiene de calcular el área por flecha según las medias de la Figura 2.3.1.2
- Cantidad: es el número de flechas utilizadas en el plano

Entonces los cálculos por tipo de flecha serian:

Flechas sobre Pavimento

- Flechas para velocidad menor de 60 km/h:
    - Frente: $1.21[m^2] \times cantidad$
    - Vuelta: $1.44[m^2] \times cantidad$
    - Frente vuelta: $2.19[m^2] \times cantidad$
- Flechas para velocidad mayor de 60 km/h:
    - Frente: $2.08[m^2] \times cantidad$
    - Vuelta: $3.05[m^2] \times cantidad$
    - Frente vuelta: $4.08[m^2] \times cantidad$

2.3.1.4. Reductor De Velocidad

Son dispositivos que se construyen sobresaliendo del pavimento en todo el ancho del arroyo vial, incluyendo en su caso los acotamientos, sólo en casos excepcionales en los que se requiera obligar al conductor a reducir la velocidad del vehículo para que se detenga inmediatamente antes del inicio de un área de conflicto, como un cruce de peatones, una zona urbana, una intersección a nivel con otra carretera o vialidad más importante y las estaciones de cuerpos de emergencia, como bomberos y ambulancias, entre otros. Son rayas diagonales de 40 cm de ancho separadas entre sí 40 cm, en el

caso de calles, y de 60 cm de ancho separadas 60 cm en el caso de carreteras, con una inclinación de 45° y deben ser antecedidas por las señales preventivas SP-41 Reductor de velocidad y restrictiva SR-9 Velocidad.

Para el cálculo se tiene que considera que el ancho de reductor es de 4.5[m] y que el largo es, preferentemente el mismo que el ancho del camino, como se muestra en la Figura 2.3.1.3.

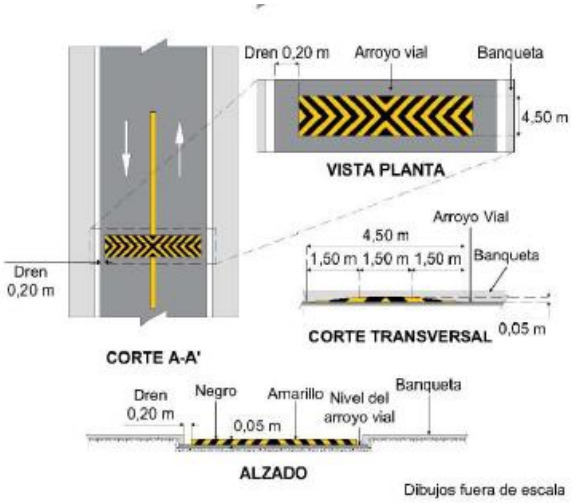


Figura 2.3.1.2.7. Reductor de velocidad en carreteras

El cálculo sería:

$$4.5[m] * A\_C[m] = A\_R[m^2]$$

Donde:

- A_C: Es el ancho de camino
- A_R: Es área del reductor

Y para la pintura, un cálculo aproximado seria:

- Para Calles:

$$P\_R[m^2] = \frac{A\_R[m^2]}{0.4}$$

- Para Carreteras

$$P\_R\ [m^2] = \frac{A\_R[m^2]}{0.6}$$

Donde:

- P_R: Es la pintura del reductor
- A_R: Es área del reductor

2.3.1.5. Pintura Paso Peatonales.

Rayas para cruce de peatones (M-7):

Se emplean para marcar las áreas de cruce de peatones en las intersecciones de calles y carreteras correspondientes. Deben ser de color blanco reflejante y trazarse a lo ancho de la vía, como se muestra en la Figura 2.3.1.2.8. Estas rayas consisten en una secuencia de líneas de 40 cm de ancho, separadas por 40 cm, alineadas con la dirección del tráfico y con una longitud mínima, pero nunca menor que el ancho de las aceras que conectan.

Cuando la geometría de la intersección o el alto flujo de peatones no permitan una trayectoria de cruce claramente definida, la disposición de las rayas puede ser diagonal, sujeta a un estudio de ingeniería de tráfico para su determinación.

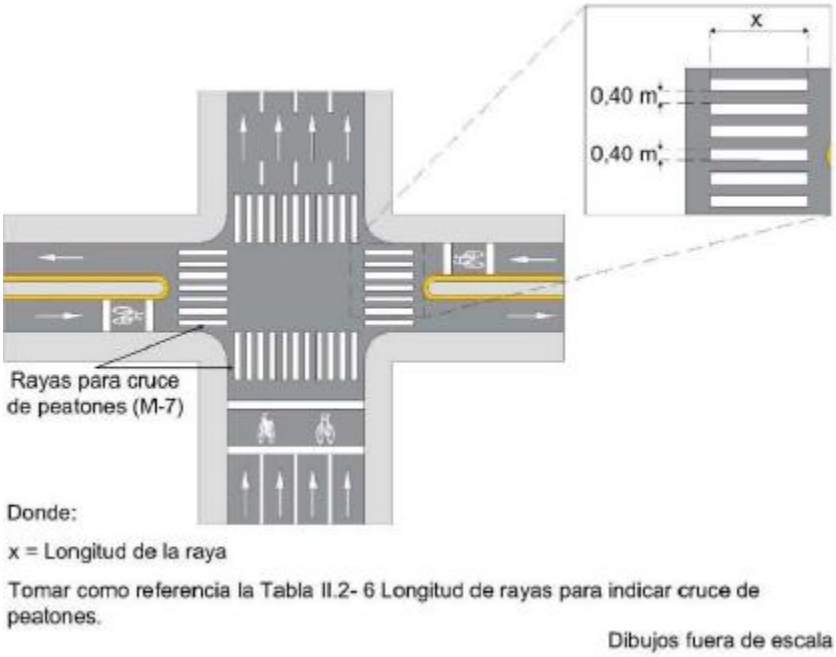


Figura 2.3.1.2.9. Rayas para cruce de peatones (M-7)

Basado en esto el cálculo de pintura quedaría:

$$P\_C\_P = \frac{A\_C[m]}{0.4 * 2} * L\_R\ [m]$$

Donde:

- P_C_P: Es la pintura del cruce peatonal en [m²]

- A_C: Es el Ancho del camino
- L_P: Es Longitud de Raya

Y L_P se obtiene de la siguiente tabla:

Tipo de vía	Longitud de la raya
Carretera	2
Calle primaria	6
Calles secundaria y terciaria	4
Vía ciclista	3

Tabla 5. Longitud de rayas para cruce de peatones

2.3.1.6. Botones

BOTONES (BRM):

Se utilizarán para complementar las marcas sobre el pavimento, su estructura deberá ser lisa y se fijarán en anclas o adhesivos, no debiendo sobresalir más de 2 cm del nivel del pavimento.

TIPO DE DISPOSITIVOS					
BOTONES					
CLAVE	COLOR	RAYA	UBICACIÓN	CARACTERISTICAS	CANTIDAD
DH-1.9	BLANCO	M-2.3	A CADA 30 m.	UNA CARA	85 pzas.
DH-1.11	BLANCO	M-3.1	A CADA 30 m.	UNA CARA	38 pzas.
DH-1.11	BLANCO	M-3.2	A CADA 32 m.	UNA CARA	64 pzas.
DH-1.14	AMARILLO	M-3.3	A CADA 30 m.	UNA CARA	120 pzas.
DH- (totales)	BLANCO	M-9		UNA CARA	384 pzas.

Figura 2.3.1.2.10. Cuadros de Botones Autorizado

Botones utilizado en las Rayas separadora de sentidos de circulación:

- Línea M-1.1, M-1.2, M-1.3, M-1.4 en curvas: $\frac{distancia\ [m]}{15}$
- Línea M-1.1, M-1.2, M-1.3, M-1.4 en tangente: $\frac{distancia[m]}{30}$

Raya separadora de carriles: (en tangente y curva)

- Línea M-2.1 sencilla continua: $\frac{distancia[m]}{30}$

- Línea M-2.2 continua doble: $\frac{distancia[m]}{30}$
- Línea M-2.3 discontinua: $\frac{distancia[m]}{30}$

Raya en la orilla del arroyo (en tangente y curva):

- Línea 3.1 derecha continua: $\frac{distancia[m]}{30}$
- Línea 3.3 izquierda continua: $\frac{distancia[m]}{30}$

Raya canalizadora:

- Línea M-5: $\frac{distancia[m]}{2}$

Sobre rampa de emergencia

- Línea M-14: $\frac{\boldsymbol{distancia}[m]}{\mathbf{15}}$ en curva
- Línea M-14: $\frac{distancia[m]}{30}$ en tangente

Raya logarítmica:

- Total de rayas: $A\_C[m] * C\_R[m]$

Donde:

- A_C: Es el Ancho del camino
- C_R: Es la cantidad de rayas

2.3.2. Señalización Vertical

La señalización vertical se utiliza para advertir sobre posibles peligros en la carretera, indicar restricciones de uso y guiar a los conductores con información útil. Esto incluye nombres de lugares, números de ruta y lugares de interés. También sirve para transmitir mensajes de seguridad.

Las señales verticales se dividen en varios tipos según su función, como preventivas, restrictivas, informativas. También pueden identificarse por su forma y tamaño, como señales bajas o elevadas. Las señales elevadas se usan en carreteras principales o donde la visibilidad es limitada, mientras

que las señales bajas son más comunes en áreas urbanas y calles estrechas. Las señales verticales, según su función, se clasifican como se indica en siguiente tabla.

Clasificación	Tipos de señales
SP	Señales preventivas
SR	Señales restrictivas
SI	Señales informativas
SII	Señales informativas de identificación
	● De nomenclatura
	● De ruta
	● De kilometraje
	● De salida
SID	Señales informativas de destino
	● Previas
	● Diagramáticas
	● Decisivas
	● Confirmativas
SIR	Señales informativas de recomendación
SIG	Señales de información general
SIT	● Señales turísticas
SIS	● Señales de servicios
OD	Señales adicionales
OD-5	Indicadores de obstáculos
OD-8	Reglas para vados y zonas inundables
OD-12	Indicadores de curvas cerradas

Tabla 6. Clasificación funcional de la señalización vertical

2.3.2.1. Cuantificación de Señales

Como se mencionó anteriormente, en la revisión también se incluye el señalamiento vertical, donde se detallan las especificaciones ubicación y dimensiones de cada señal. Esta revisión implica

asegurarse de que todo cumpla con las normas establecidas, utilizando cuadros que contengan la cantidad, ubicación, tipo de señal y su descripción.

Para realizar la cuantificación de señales, se recopilan todos los datos en una tabla detallada. Esta tabla debe incluir columnas para la cantidad de señales, su ubicación exacta, el tipo de señal (por ejemplo, SR-9) y una descripción clara de la función de cada señal (por ejemplo, "Velocidad"), como se muestra en la Figura 2.11. Luego, se suman todas las señales del mismo tipo para generar el cuadro de resumen total. Este cuadro de resumen proporciona una visión global del señalamiento vertical requerido para el proyecto, mostrando claramente el tipo de señal y su función, así como la cantidad total. Este enfoque asegura una cuantificación precisa y una organización clara para la implementación del señalamiento vial.

2.3.2.2. Tipos de Señales

SEÑALES PREVENTIVAS (SP):

Tableros fijados en postes con símbolos que tienen por objeto prevenir a los conductores de vehículos sobre la existencia de algún peligro en el camino y su naturaleza.

SEÑALES RESTRICTIVAS (SR):

Tableros fijados en postes con símbolos y/o leyendas que tienen por objeto indicar al usuario, lento en zona rural como urbano, la existencia de limitaciones físicos o prohibiciones reglamentarias que regulan el tránsito.

SEÑALES INFORMATIVAS (SI):

Tableros fijados en postes con leyendas y/o símbolos que tienen por objeto guiar al usuario a lo largo de su itinerario por calles y carreteras e informarle sobre nombres y ubicación de poblaciones, lugares de interés, servicios, kilometrajes y ciertas recomendaciones que conviene observar.

SEÑALES INFORMATIVAS DE DESTINO (SID):

Se usarán para informar a los usuarios sobre el nombre y la ubicación de cada uno de los destinos que se presentan a lo largo de su recorrido; podrán ser señales bajas, diagramáticas y elevadas.

SEÑALES INFORMATIVAS DE RECOMENDACIÓN (SIR):

Se utilizarán con fines educativos para recordar a los usuarios determinadas disposiciones o recomendaciones de seguridad que conviene observar durante su recorrido por calles y carreteras.

SEÑALES DE INFORMACIÓN GENERAL (SIG):

Se utilizarán para proporcionar a los usuarios información general de carácter poblacional y geográfico, así como para indicar nombres de obras importantes en el camino, límites políticos, ubicación de casetas de cobro, puntos de inspección y sentido de circulación del tránsito, entre otros.

SEÑALES INFORMATIVAS DE SERVICIOS Y TURÍSTICAS (SIST):

Se utilizarán para informar a los usuarios sobre la existencia de un servicio o de un lugar de interés turístico y/o recreativo. En algunos casos, estas señales podrán usarse combinadas con una informativa de destino en un mismo tablero.

2.3.2.3. Delineadores

INDICADORES DE ALINEAMIENTO (OD-6):

Se emplearán para delinear la orilla de una vía de circulación, en cambios de alineamiento horizontal, para señalar los extremos de muros de cabeza de alcantarillas y para marcar estrechamientos de una vía de circulación.

- $\frac{distancia}{40} = total$
- $total \times 2\ (Ambos\ lados\ del\ camino) = total$

2.3.2.4. Barrera Metálica

DEFENSAS (DD-):

Se emplearán para evitar en lo posible que los vehículos salgan del camino o invadan el carril contrario, podrán ser de lámina galvanizada, concreto u otro material resistente apoyados en postes adecuados al tipo de material.

3. Enfoque de Desarrollo de Proyecto

3.1. Objetivo

El objetivo de este proyecto es desarrollar una aplicación de software que ayude a los ingenieros civiles en México en el diseño y la elaboración de propuestas de señalización para el sistema vial del país. Esta aplicación busca facilitar el proceso de cálculo y diseño de señalización horizontal y vertical, conforme a los lineamientos establecidos por el *Manual de Señalización y Dispositivos para el Control del Tránsito en Calles y Carreteras* (2023), permitiendo generar tablas de la cuantificación de pintura y de cuantificación de señales, así como de defensa, además de la cuantificación de algunos dispositivos.

Para ello se desarrollará una interfaz que sea amigable al usuario, donde se le permita hacer los cálculos correspondientes y esos cálculos se acomoden en una tabla que sirva para visualizar los totales de los cálculos, y que después estas tablas se colocan dentro de los planos que se mandan a la SICT, que se encargan de aprobar o rechazar los planos recibidos.

3.2. Estado del Arte

Existen diversas herramientas y aplicaciones orientadas al diseño, cuantificación y gestión de información relacionada con proyectos de infraestructura vial. Sin embargo, gran parte de estas soluciones se enfocan únicamente en el dibujo técnico dentro de plataformas CAD, dejando de lado procesos específicos de automatización de cuantificaciones y la generación de tablas conforme a las normativas.

Actualmente, herramientas como AutoCAD Civil 3D permiten desarrollar planos y modelos relacionados con la señalización vial, sin embargo, muchas de las funciones de cuantificación y organización de señalamiento se realizan de forma manual o mediante hojas de cálculo externas. Esto genera procesos repetitivos, incremento en los tiempos de trabajo y mayor posibilidad de errores humanos.

Asimismo, existen soluciones desarrolladas mediante hojas de cálculo o scripts personalizados que automatizan parcialmente ciertos cálculos de señalización. Sin embargo, estas alternativas suelen tener ciertas problemáticas o limitaciones, como por ejempló:

- No hay una integración directa con AutoCAD, y AutoCAD permite solo hacer tareas específicas, que después se tienen que organizar en las tablas .
- La exportación de la tabla no es editable o se pierde el formato.
- Se requiere hacer instalaciones extra o pagar una suscripción.
- Interfaces gráficas especializadas para hacer varias tareas no solo lo relacionado con señalización.
- Los cálculos, que se puede hacer en otras plataformas, después se tienen que organizar o capturar los resultados en las tablas que el usuario tiene que presentar.

Debido a estas limitaciones, el presente proyecto propone una aplicación enfocada específicamente en la gestión y cuantificación de elementos de señalización vial, integrando almacenamiento estructurado, generación automática de tablas y exportación compatible con AutoCAD.

3.3. Planificación de Proyecto

Para la planificación del proyecto se identificó las necesidades de un ingeniero civil encargado de la señalización, para ello se le cuestionó sobre los problemas que pasa al desarrollar su trabajo y formas en que se podría optimizar su trabajo. A partir de esto, se definieron las etapas clave del desarrollo: análisis de requerimientos, diseño del sistema, implementación, pruebas y validación, así como su posible expansión futura o actualizaciones.

La planificación se estructuró en seis fases:

1. **Análisis de requerimientos**: Estudio detallado del manual y necesidades del usuario.
2. **Diseño del sistema**: Definición de interfaz gráfica, estructuras de datos y lógica de cálculo.
3. **Desarrollo**: Implementación del código en Java.
4. **Pruebas**: Verificación de cálculos y validación de normas.
5. **Implementación y capacitación**: Aplicación en proyectos reales y entrenamiento de usuarios.
6. **Mantenimiento y actualización**: Adaptación a futuros cambios normativos.

Estas etapas son las que se consideran esenciales para la realización del programa.

3.4. Metodología

Para el desarrollo del proyecto se adoptó el **Modelo Espiral**, una metodología que permite incorporar cambios que vayan surgiendo durante el desarrollo, lo cual resulta adecuado cuando los requerimientos proporcionados son modificados y/o corregidos, por el usuario, de manera progresiva conforme se vaya avanzando en el proyecto.

En este proyecto, la interacción principal se desarrolla entre un ingeniero civil, quien funge como fuente de requerimientos, y el desarrollador. Durante cada ciclo o espira del modelo, el ingeniero civil aporta nuevos requerimientos o ajustes sobre las funcionalidades implementadas del sistema, y el desarrollador se encarga tanto de analizarlos como de incorporarlos en la siguiente iteración. De este modo, cada ciclo del modelo espiral permite modificar objetivos, evaluar riesgos, planificar actividades y desarrollar prototipos o módulos funcionales que posteriormente son revisados por el usuario.

Aunque el proyecto es desarrollado por una sola persona, el modelo espiral permite organizar distintas responsabilidades en cada etapa del ciclo:

- **Definición de objetivos y Requerimientos:** el ingeniero civil establece las necesidades reales del proyecto y el desarrollador las traduce en características concretas a implementar.
- **Análisis de riesgos y planificación:** el desarrollador evalúa posibles problemas técnicos, identifica dependencias, estima tiempos y decide la priorización de funciones.
- **Desarrollo, pruebas y documentación:** toda la labor técnica recae en el desarrollador, quien construye cada módulo, realiza pruebas funcionales y documenta el avance.
- **Revisión y retroalimentación:** al finalizar cada iteración, el módulo construido se presenta al ingeniero civil para validar su funcionamiento y recopilar observaciones que alimentarán la siguiente espira del modelo.

Este enfoque proporciona varias ventajas dentro de un contexto donde se cuenta con requerimientos cambiantes:

- Permite mantener una visión clara del progreso, ya que cada ciclo produce un avance verificable.
- Favorece una estructura modular, donde cada funcionalidad se desarrolla e integra de forma progresiva.

- Asegura una adaptación continua a nuevos requerimientos, sin necesidad de reiniciar procesos o afectar significativamente el calendario del proyecto.
- Facilita una gestión ordenada del tiempo y de las prioridades, al analizar los riesgos y planificar de manera iterativa.

En cuanto a las herramientas de desarrollo, se eligió el lenguaje **Java**, ya que ofrece características alineadas con un enfoque iterativo y modular. Entre sus principales ventajas destacan:

- **Portabilidad:** gracias a la Máquina Virtual de Java (JVM), el programa puede ejecutarse en distintos sistemas operativos sin modificar el código.
- **Programación Orientada a Objetos (POO):** facilita la creación de componentes modulares, escalables y reutilizables, coherentes con la naturaleza iterativa del modelo espiral.
- **Amplio soporte de bibliotecas:** lo cual resulta fundamental para el desarrollo de interfaces gráficas, manipulación de archivos y la exportación de información hacia formatos como Excel o AutoCAD.

En conjunto, la aplicación del Modelo Espiral y el uso de Java permitieron dividir el proyecto en módulos funcionales que se integran de manera progresiva, ofreciendo una arquitectura clara, manteniéndose adaptable a los cambios solicitados por el usuario y acelerando el desarrollo global del sistema.

3.5. Análisis de Requerimientos

Durante el desarrollo del proyecto se empleó el Modelo Espiral, por lo que la recopilación, análisis y validación de los requerimientos se realizó de forma iterativa. En cada ciclo del modelo se sostuvo comunicación con el ingeniero civil, quien funge como usuario principal del sistema, para identificar nuevas necesidades, resolver dudas técnicas y ajustar especificaciones conforme se avanzaban en las funcionalidades del software.

Como resultado de este proceso se definieron los siguientes grupos de requerimientos funcionales:

Señalización Horizontal:

- El programa debe realizar la cuantificación de pintura para las marcas de señalización horizontal más utilizadas, en este caso se consideran que hay 20 marcas de pintura.

- La medición se realizará en metros lineales, por lo que no es necesario considerar el ancho de la raya.
- Esta cuantificación se debe de presentar en una tabla que cumpla con un formato utilizado.

SEÑALAMIENTO HORIZONTAL					
NUEVA CLAVE	COLOR	DIMENSION	CANT.		OBSERVACIONES
M-2.1	BLANCO	150mm. DISC.	592	ml.	RAYA SEPARADORA DE CARRILES, CONTINUA SENCILLA
M-2.3	BLANCO	150mm. DISC.	850	ml.	RAYA SEPARADORA DE CARRILES, DISCONTINUA
M-3.1	BLANCO	150mm. CONT.	1,140	ml.	RAYA EN LA ORILLA DERECHA, CONTINUA
M-3.2	BLANCO	150mm. CONT.	1,029	ml.	RAYA EN LA ORILLA DERECHA, DISCONTINUA
M-3.3	AMARILLO	150mm. CONT.	3,680	ml.	RAYA EN LA ORILLA IZQUIERDA
M-9	BLANCO	600 mm. CONT.	640	ml.	RAYA LOGARITMICA
M-11.1	BLANCO		10	pzas.	FLECHAS SOBRE PAVIMENTO
M-15	AMARILLO/NEGRO		2	pzas.	REDUCTOR DE VELOCIDAD

Figura 3.1. Tabla Señalización Horizontal

- La tabla generada debe ser exportable a AutoCAD como tabla real, no como imagen, para permitir edición posterior.

Señalamiento Vertical:

- El programa debe hacer la cuantificación de señales verticales obteniendo el nombre, ubicación y dimensión de cada señal.
- Esta cuantificación se debe de presentar en una tabla que cumpla con un formato utilizado.

CARRETERA: TAMAZUNCHALE - CD VALLES				
CANT	UBICACIÓN	SEÑAL	DIMENSION	DESCRIPCION
2	326+610	SR-9	117X117	VELOCIDAD
2	326+480	SR-13	117X117	CAMIONES CARGA CARRIL DERECHO
2	326+400	SP-6	117X117	CURVA
2	326+170	SP-6	117X117	CURVA
2	326+090	SP-41	117X117	REDUCTOR DE VELOCIDAD (T.A. 35X152)
2	326+020	SP-32	117X117	PEATONAL
2	326+000	SII-15	30X76	KILOMETRAJE DE RUTA
2	325+940	SR-9	117X117	VELOCIDAD
1	325+800	SID-9	56X300	SEÑAL BAJA 2 TABLEROS
1	325+725	SIR-	56X300	SEÑAL INFORMACION RECOMENDACION
1	325+670	SID-13	122X366	BANDERA
1	325+670	OD-5	30X122	SEÑAL OBSTACULO
2	325+650	SP-6	117X117	CURVA
1	325+560	SID-11	56X300	SEÑAL CONFIRMATIVA 1 TABLERO
2	325+260	SR-34	117X117	UTILICE SU CINTURON DE SEGURIDAD

Figura 3.2. Señalización Vertical

- Hay que considerar que hay dos tipos de proyectos, Entronque y Camino Abierto, en el entronque a parte de los datos anteriores también hay que agregar un apartado para un código, este código sirve como una forma de identificación de la señal.

- El sistema debe generar una tabla que cuantifiqué el total de cada señal utilizada en el plano

CUANTIFICACION TOTAL CARRETERA: TAMAZUNCHALE - CD VALLES			
SEÑAL	DIMENSION	CANT.	DESCRIPCION
SP-6	117X117	12	CURVA
SP-8	117X117	2	CURVA INVERSA
SP-32	117X117	4	PEATONAL
SP-41	117X117	6	REDUCTOR DE VELOCIDAD (T.A. 35X152)
SR-9	117X117	10	VELOCIDAD
SR-13	117X117	2	CAMIONES CARGA CARRIL DERECHO
SR-34	117X117	4	UTILICE SU CINTURON DE SEGURIDAD
SII-15	30X76	4	KILOMETRAJE DE RUTA
SIR-	56X300	2	SEÑAL INFORMACION RECOMENDACION
SID-9	56X300	2	SEÑAL BAJA 2 TABLEROS
SID-11	56X300	2	SEÑAL CONFIRMATIVA 1 TABLERO
SID-13	122X366	2	BANDERA
OD-5	30X122	2	SEÑAL OBSTACULO
OD-6	VER DETALLE	16	DELINEADORES TOTALES
OD-11	76X90	39	INDICADOR DE CURVA CERRADA

Figura 3.3. Cuantificación total de señales Verticales

- La tabla debe poder exportarse a AutoCAD como tabla editable.

Defensa:

- El programa debe registrar el kilometraje en donde inicia y en donde termina la defensa en el plano y también indicar en qué lado de camino se está señalando.
- Independiente del lado de la defensa siempre se debe acomodar en orden creciente al kilometraje de inicio de la defensa al momento de Exportar.
- Esta cuantificación se debe de presentar en una tabla que cumpla con un formato utilizado

COLOCACION DE BARRERA METALICA 3 CRESTAS					
INICIO	FINAL	POSICION	SECC. EXTREMA AMORTIGUAMIENTO		LONGITUD ml.
315+020	315+060	LADO DERECHO	1	pzas.	40
315+290	315+330	LADO DERECHO	1	pzas.	40
315+510	315+550	LADO DERECHO	1	pzas.	40

Figura 3.4. Tabla de Defensa.

- La tabla debe poder exportarse a AutoCAD como tabla editable.

Barreras:

- Hay 5 elementos que esta debe poder cuantificar, el total de Barrera Metálica(Defensa), la barrera central tipo New Jersey, Sección Transición Rigidez/Puente, Sección Extrema Amortiguamiento, Amortiguador De Impacto y Sección Terminal Cola De Pato.

- Esta cuantificación se debe de presentarse en una tabla que cumpla con un formato utilizado

TIPO DE DISPOSITIVOS			
BARRERAS			
CLAVE	POSICION	CANTIDAD	CARACTERISTICAS
OD-4.1.2 (NC-3)		120 ml.	BARRERA METALICA 3 CRESTAS
OD-4.4.1 (NC-3)		3 pzas.	SECCION EXTREMA AMORTIGUAMIENTO
OD-4.4.2/S (NC-3)		3 pzas.	SECCION TERMINAL COLA DE PATO
OD-4.2.3 (NC-3)		3 pzas.	BARRERA TIPO NEW JERSSEY
OD-4.3 (NC-3)		3 pzas.	SECCION TRANSICION DE RIGIDEZ/PUENTE
OD-4.4.1 (NC-3)		3 pzas.	AMORTIGUADOR DE IMPACTO
OD-4.4.2/A (NC-3)		3 pzas.	SECCION TERMINALES ATERRIZADAS

Figura 3.5. Tabla de Barreras.

- La tabla debe poder exportarse a AutoCAD como tabla editable.

Dispositivos:

- Dentro de los dispositivos a cuantificar son: indicadores de alineamiento, balizas, delimitadores, dispositivos antideslumbrantes y botón reflejante, estos dispositivos se debe cuantificar la cantidad total.
- Esta cuantificación se debe de poner en una tabla que cumpla con un formato utilizado

	TIPO DE DISPOSITIVOS		
DISPOSITIVOS	CLAVE	CANTIDAD	UNIDAD
MENSULA REFLEAJANTE SOBRE BARRERA METALICA	DH-3	396	pzas.
MENSULA REFLEAJANTE SOBRE BARRERA DE CONCRETO (TIPO NEW YERSEY)	DH-3	360	pzas.
MALLA ANTIDESLUMBRANTE	OD-12	1,800	ml.

Figura 3.6. Tabla de Dispositivos.

- Esta tabla debe de poder usase en AutoCAD como tabla, no como simple imagen

Botones:

- Los botones deben de poder calcularse a partir de las marcas horizontales que ya se haya agregado al programa y partir de esas calcular el total de botones, pero solo para las marcas que deben llevar botones.
- Esta cuantificación se debe de poner en una tabla que cumpla con un formato utilizado

TIPO DE DISPOSITIVOS					
BOTONES					
CLAVE	COLOR	RAYA	UBICACIÓN	CARACTERISTICAS	CANTIDAD
DH-1.9	BLANCO	M-2.3	A CADA 30 m.	UNA CARA	85 pzas.
DH-1.11	BLANCO	M-3.1	A CADA 30 m.	UNA CARA	38 pzas.
DH-1.11	BLANCO	M-3.2	A CADA 32 m.	UNA CARA	64 pzas.
DH-1.14	AMARILLO	M-3.3	A CADA 30 m.	UNA CARA	120 pzas.
DH- (totales)	BLANCO	M-9		UNA CARA	384 pzas.

Figura 3.7. Tabla de Botones.

3.6. Gestión de Riesgos

Se contemplaron riesgos como:

- **Cambios en la normativa oficial**: que salgan nuevas normativas o actualizaciones de las mismas que hagan hacer cambios.
- **Errores de cálculo o interpretación del manual**: esto puede representar problemas al usuario.
- **Limitaciones técnicas (exportación, compatibilidad)**: que no se usen modulaciones correctas o que se creen clases innecesarias al momento resolver una función.
- **Falta de validación de usuarios reales**: ya que se planea crear el programa para un usuario en específico, es posible el problema de uno no sea el de otro.

4. Diseño de Proyecto

4.1. Lenguajes de Programación

El desarrollo del programa, como se mencionó anteriormente, se basa principalmente en el lenguaje de programación Java, seleccionado por su portabilidad, robustez y compatibilidad con medios de desarrollo orientados a objetos. Java permite implementar una estructura modular, reutilizable y escalable, que se adapta adecuadamente a la complejidad de una herramienta de asistencia técnica para señalización vial.

Java es una tecnología que surgió como respuesta a las limitaciones de los lenguajes tradicionales como C y C++, particularmente en lo referente al desarrollo de aplicaciones complejas, portables y modulares. Su diseño orientado a objetos permite dividir los sistemas en componentes reutilizables y autónomos, lo que mejora significativamente el mantenimiento y la escalabilidad del software desarrollado [7].

A diferencia de C, donde la complejidad de los programas crece de manera poco controlada a medida que aumenta su tamaño, Java adopta de forma nativa el paradigma de programación orientada a objetos, facilitando la creación de estructuras independientes con interfaces bien definidas que ocultan la complejidad interna del sistema [7]. Esto permite, por ejemplo, encapsular toda la lógica relacionada con un elemento de señalización en una clase específica, como la de señales verticales, horizontales, etc., sin que el resto del sistema deba conocer los detalles de su funcionamiento.

Además, Java se caracteriza por ser multiplataforma, ya que el código desarrollado se ejecuta sobre la Máquina Virtual de Java (JVM), lo cual garantiza que la aplicación pueda correr en diferentes sistemas operativos sin modificaciones adicionales, o cambios significativos en el código. Esta portabilidad es fundamental para una aplicación que se busca que sirva como una herramienta que utilizada por profesionales en distintos entornos de trabajo.

Otro aspecto clave es su amplia cobertura de bibliotecas y herramientas, que facilitan tareas comunes como el desarrollo de interfaces gráficas, el manejo de bases de datos, o la integración con otros sistemas como AutoCAD.

Para el desarrollo del sistema se evaluaron distintas alternativas tecnológicas considerando aspectos como integración con AutoCAD, facilidad de construcción de interfaces gráficas, manejo de bases de datos y mantenimiento del sistema.

Aunque lenguajes como Python ofrecen algunas ventajas, especialmente en rapidez de desarrollo y disponibilidad de bibliotecas, Java resultó más conveniente para este proyecto debido a la estructura modular requerida y a la facilidad para construir aplicaciones de escritorio robustas con múltiples ventanas y componentes interactivos.

Además, el uso de Java permitió mantener una arquitectura más organizada para integrar formularios, tablas dinámicas, validaciones y procesos de exportación.

Comparación con otras alternativas

Python

Python representa una alternativa viable gracias a su sintaxis sencilla y gran cantidad de bibliotecas disponibles. Sin embargo, para este proyecto se identificaron algunas limitaciones:

- Las interfaces gráficas de escritorio pueden requerir bibliotecas externas adicionales.

- La distribución de aplicaciones ejecutables puede resultar menos práctica.
- Algunas integraciones con otras herramientas como CAD pueden depender de configuraciones externas.

A pesar de ello, Python destaca en automatización, análisis de datos y prototipado rápido.

C#

C# también fue considerado debido a su capacidad para desarrollar aplicaciones de escritorio mediante Windows Forms o WPF. Sin embargo, su dependencia más directa con el sistema Windows y herramientas específicas de Microsoft hizo que Java representara una opción más flexible para el desarrollo académico del proyecto.

Hojas de cálculo y macros

El uso de Excel y macros VBA puede resolver parcialmente tareas de cuantificación; sin embargo:

- Presenta menor escalabilidad.
- Complica el manejo relacional de datos.
- Dificulta el mantenimiento del sistema.
- Limita la integración estructurada con múltiples módulos.

A parte de utilizar el lenguaje de Java, se planea el uso de AutoLISP para la integración con AutoCAD, permitiendo automatizar procesos como la inserción de tablas generadas por la aplicación directamente en los planos de señalización. AutoLISP es el lenguaje nativo de AutoCAD para scripts y personalización de sus archivos de trabajo, lo que asegura una compatibilidad con los objetos del dibujo (como textos, líneas y bloques), facilitando así la obtención de los datos de los objetos de dibujo.

4.2. Productos del Diseño

El diseño del sistema se desarrolló mediante ciclos iterativos siguiendo el Modelo Espiral, lo que permitió definir de manera progresiva los componentes, interfaces, estructuras y procesos internos a medida que se validaban funcionalidades con el usuario. Como resultado, se generaron los siguientes productos de diseño**:** detalle procedimental, estructura de datos, arquitectura del sistema e interfaces gráficas, cuidando criterios de alta cohesión y bajo acoplamiento para garantizar un sistema modular, mantenible y escalable.

4.2.1. Detalle Procedimental

Este producto del diseño describe los pasos lógicos que seguirá el sistema para ejecutar cada una de sus funciones principales.

Bajo el modelo espiral, estos procedimientos fueron refinados iterativamente conforme surgieron nuevos requerimientos, ajustes del usuario y validación de riesgos, especialmente en la comunicación entre Java y AutoCAD mediante archivos LISP.

Debido a que el usuario trabaja directamente en AutoCAD, y la información geométrica se captura desde allí, los procedimientos incluyen el flujo:

Java → LISP → AutoCAD → LISP → Java → BD.

Procedimiento para crear o cargar un proyecto

- Mostrar el menú inicial en la aplicación Java.
- Permitir elegir entre:
  - Crear un proyecto nuevo.
  - Cargar un proyecto existente desde la base de datos.
- En caso de un Nuevo Proyecto, Permitir validar los datos del proyecto:
  - Nombre.
  - Tipo de vía.
  - Velocidad de operación.
- Guardar el proyecto en la BD (si es nuevo).
- Redirigir al menú principal donde se seleccionará la cuantificación a realizar.

Procedimiento para cuantificar señalización horizontal

- El usuario selecciona la opción “Señalización Horizontal” en la aplicación.
- La aplicación muestra una lista con todas las marcas que se pueden agregar.
- Según la selección se muestra una ventana con botones para agregar la marca correspondiente.
- Cada botón muestra las instrucciones correspondientes, también indica el comando a usar en AutoCAD.
- En AutoCAD, el usuario ejecuta el comando correspondiente.
- El archivo LISP:
  - Según la tarea a realizar se muestran una serie de instrucciones a realizar.

- Guarda los datos en un archivo de Texto.

- El usuario regresa a la aplicación, se asegura que los datos sean correctos.
- La aplicación:
  - Lee el archivo de texto.
  - Hace cálculos con los datos obtenidos, según la marca correspondiente.
  - Crea un renglón con las columnas correspondientes a los datos a guardar en la base de datos por marca.
  - Inserta registros en la base de datos asociados al proyecto.
- Se confirma para subir los datos a la base de datos.
- La tabla temporal de resultados se actualiza para su futura exportación.

Procedimiento para cuantificar señalamiento vertical.

- El usuario selecciona "Señalización Vertical" en la aplicación.
- El archivo LISP realiza las siguientes tareas:
- Solicita al usuario seleccionar el texto correspondiente a una señal en el plano.
- El usuario debe elegir los datos de la señal en siguiente orden:
  - Ubicación (Kilometraje).
  - Nombre de la Señal.
  - Dimensión.
- Se pueden seleccionar varias señales, considerando que por señal se deben seleccionar los 3 textos.
- Almacena la información recopilada en un archivo de texto.
- La aplicación importa los datos almacenados:
  - Lee el tipo de señal, su ubicación y dimensiones.
  - La aplicación obtiene la descripción correspondiente al nombre de la señal.
  - Genera una tabla acumulada de señales verticales.
- Registra los datos en la base de datos correspondiente.

Procedimiento para cuantificar defensa.

- El usuario selecciona en la aplicación la opción relacionada con la cuantificación de defensa.
- El sistema genera un archivo LISP genérico que permite capturar información directamente desde AutoCAD.

- El usuario carga el archivo LISP y ejecuta el comando indicado.
- El programa LISP:
  - Permite seleccionar un tramo representado por una línea o polilínea.
  - Obtiene los puntos inicial y final de la defensa, obteniendo los kilometrajes de cada punto.
  - Guarda todos los kilometrajes en un archivo de texto.
- La aplicación Java importa los datos:
  - Ordena los datos conforme se seleccionaron los datos en AutoCAD.
  - Calcula longitudes totales conforme a los Kilometrajes, considerando que seleccionaron en el orden de kilometraje inicial y kilometraje final.
  - Genera la tabla en el formato requerido para el proyecto.
  - Inserta la información procesada en la base de datos.

Procedimiento para cuantificar Barreras.

- Dentro de la ventana de la Defensa, habrá un botón que redirigirá para la cuantificación de Barreras.
- La aplicación leerá los datos de la defensa total que se hayan agregado en la ventana anterior.
- El usuario solo debe de indicar la cuantificación de todos los elementos que componen la tabla .
- El usuario indica el nivel de contención.
- La aplicación permite actualizar los datos de la base datos para este proyecto.

Procedimiento para cuantificar dispositivos y botones

Para dispositivos

- Dependiendo el dispositivo se utiliza el LISP correspondiente para hacer los cálculos.
- La aplicación hace los cálculos y en algunos casos leerá los datos ya almacenados en la tabla de Defensa.
- Java importa el archivo de texto y llena la tabla correspondiente.

Para botones

- El cálculo se hace internamente en la aplicación Java.
- El sistema analiza las señales horizontales ya capturadas y determina cuáles requieren botones.

- Calcula automáticamente el total.
- Genera su propia tabla.

Procedimiento para exportar tablas a AutoCAD

- El usuario selecciona la opción “Exportar”, este botón estará en todas las ventanas de cada procedimiento mencionados anteriormente.
- Java convierte la información de la tabla en un archivo de texto, la cual cada fila de datos de la tabla se delimitará por comas.
- Se tendrán archivos LISP para generar la tabla correspondiente a cada procedimiento.
- Se muestra una tabla real editable en AutoCAD.

4.2.2. Estructura de Datos

Se plantea la creación de una base de datos relacional, cuyo objetivo principal será almacenar toda la información vinculada a los diferentes elementos de señalización (horizontal, vertical, defensas, botones, dispositivos). La tabla principal es la de proyectos, y a partir de esta se derivarán todas las demás entidades relacionadas.

Cada entidad se diseñará con base en los requerimientos del manual de señalización y en las necesidades funcionales del sistema, permitiendo una organización completa entre los datos de cada elemento vial y su aplicación dentro de un proyecto determinado. A continuación, se describen las tablas que se tienen contempladas:

- **Proyectos**: Representará cada obra o tramo vial registrado en el sistema. Contendrá campos como el nombre del proyecto, la velocidad permitida y el tipo de vía. Será la tabla central desde la cual se relacionarán todos los elementos del proyecto.
- **Señales horizontales (bajas)**: Esta tabla almacenará la información referente a las marcas viales pintadas sobre el pavimento (rayas continuas, discontinuas, doble línea, etc.). Se incluirán campos como tipo de raya, clasificación, marca, lado, distancia y cantidad. Cada señal estará asociada a un proyecto.
- **Señales verticales**: Se incluirán datos como el nombre de la señal dimensiones, ubicación, código (para Entronques) y cantidad. Esta información permitirá generar una cuantificación detallada por proyecto.

- **Defensas metálicas**: Se planea una tabla que registre tramos de defensa, con datos como kilómetros inicial y final, posición de instalación conforme al camino, longitud y cantidad de sección de amortiguadores, asociada también a cada proyecto.
- **Botones**: Esta entidad contendrá información sobre los botones instalados en el pavimento, incluyendo clave, color, tipo de raya en la que se colocan, ubicación y características. Se considera la posibilidad de vincularlos directamente a una señal horizontal si es necesario.
- **Dispositivos**: Esta tabla estará destinada a registrar dispositivos complementarios como los indicadores de alineamiento, balizas, dispositivos antirreflejantes, etc. almacenado la clave, posición, unidad de medida y cantidad.
- **Barreras y elementos de contención**: Se prevé una tabla para registrar datos sobre elementos como barreras metálicas, tipo New Jersey, secciones amortiguadoras, secciones de cola de pato y niveles de contención, vinculados también al proyecto correspondiente.

Relaciones entre entidades

El diseño prevé establecer claves foráneas que relacionen cada tabla secundaria con la tabla principal de proyectos. De esta forma, se garantizará que todos los elementos estén correctamente asociados a un único proyecto, facilitando su consulta, modificación y eliminación en conjunto. Además, algunas tablas podrán tener relaciones cruzadas adicionales, por ejemplo, asociar un botón a una raya específica.

Estas relaciones permitirán mantener la integridad de los datos, evitando inconsistencias, duplicados o elementos huérfanos. Asimismo, se considerarán reglas como la eliminación en cascada (ON DELETE CASCADE) para asegurar que, al borrar un proyecto, se eliminen automáticamente todos los elementos asociados.

Diseño orientado a objetos

El sistema se desarrollará en Java bajo el paradigma de programación orientada a objetos (POO). Cada tabla de la base de datos estará representada por una clase Java con atributos equivalentes a sus campos, lo que permitirá un mapeo directo entre los datos almacenados y los objetos manipulados en el programa.

Este enfoque modular permitirá implementar funcionalidades específicas para cada tipo de señal o elemento vial, como cálculos automáticos, validaciones, generación de tablas y exportación a

formatos de Textos. Además, al tener clases separadas, se facilitará el mantenimiento del sistema, su escalabilidad y la incorporación de nuevas funcionalidades en el futuro.

4.2.3. Arquitectura de la Aplicación

La arquitectura de la aplicación se fundamenta en el patrón de diseño **Modelo-Vista-Controlador (MVC)**, una estructura ampliamente adoptada en el desarrollo de software que permite separar de manera clara la lógica de negocio, la presentación de la información y la interacción con el usuario [8]. Esta división de responsabilidades no solo mejora la organización del código, sino que también facilita su mantenimiento, escalabilidad y pruebas.

Modelo

El Modelo representa el núcleo funcional del sistema, encargado de gestionar los datos, las reglas de negocio y la lógica del dominio [8]. En este proyecto, cada entidad del sistema (por ejemplo, señalamiento horizontal, vertical, defensas o dispositivos) se implementa como una clase Java que actúa como modelo, incluyendo atributos y métodos para almacenar y manipular la información técnica de cada elemento vial.

Estas clases están directamente vinculadas a la base de datos mediante estructuras relacionales, permitiendo un mapeo eficaz de datos desde el entorno visual hacia el sistema de almacenamiento persistente.

Vista

La Vista corresponde a la interfaz gráfica que permite al usuario interactuar con el sistema de manera sencilla y visualmente clara. Esta interfaz se desarrolla utilizando bibliotecas gráficas de Java como Swing, y presenta al usuario formularios de entrada, botones de acción, tablas de resultados y ventanas de navegación, adaptadas a cada módulo de trabajo (por ejemplo, cuantificación de señalamiento horizontal o vertical, cuantificación de defensas o botones).

El diseño de la vista se enfoca en mostrar los datos requeridos en cada etapa del diseño vial, como distancias, cantidades, ubicaciones y clasificaciones de los elementos según sea el caso. Además, permite visualizar las tablas finales de resultados antes de su exportación directa a AutoCAD, con formato compatible y conforme a los requerimientos establecidos por la SICT.

Controlador

El Controlador gestiona los eventos y acciones realizadas por el usuario dentro de la aplicación. Su función principal es unir la vista con el modelo, es decir, tomar las entradas que el usuario introduce

en las ventanas o selecciona desde AutoCAD, con ayuda de LISP, procesa los datos, ejecuta los cálculos cuando sea necesario, y actualiza tanto los datos como la visualización en pantalla.

En el contexto de este sistema, el controlador también se encarga de operaciones como:

- Validar que los datos ingresados cumplan con el formato y los rangos establecidos por el manual de señalización.
- Realizar los cálculos de distancias y cantidades por tipo de señal o dispositivo.
- Registrar los datos en la base de datos del proyecto.
- Generar y actualizar las tablas para exportarlas a AutoCAD.
- Controlar la navegación entre las diferentes ventanas del programa.
- Obtener y modificar archivos de texto, que son los que se leen con los LISP

Esta organización modular permite integrar fácilmente nuevos módulos en el futuro, como la inclusión de normativas nuevas, dispositivos adicionales o compatibilidad con otros formatos de salida, manteniendo una estructura clara y funcional en el código.

Componentes principales

Modelo

Incluye todas las clases que representan las tablas del sistema:

- Proyecto
- SeñalesHorizontales
- SeñalesVerticales
- Defensa
- Barrera
- Dispositivos
- BarreraTotal
- Botones

Estas clases encapsulan la información y operaciones básicas (alta cohesión).

Controladores

Se encargan de:

- Gestionar la comunicación entre interfaz y modelo
- Controlar procesos de cálculo
- Validar entradas

- Coordinar la exportación a AutoCAD

Separar estos procesos reduce el acoplamiento con las vistas.

Vistas

Ventanas independientes para:

- Menú principal
- Cuantificación de señalamiento horizontal
- Cuantificación de señalamiento vertical
- Cuantificación de defensas
- Cuantificación de barreras
- Cuantificación de dispositivos
- Cuantificación de botones
- Vista preliminar de las tablas finales

El modularidad mantiene las vistas altamente cohesionadas y evita sobrecarga de funciones.

Beneficios de la arquitectura

El uso de MVC en este sistema permite una alta cohesión por capas, ya que cada componente cumple una sola responsabilidad y permite un desarrollo más organizado, donde los componentes se desarrollan y prueban de forma independiente. Esto es especialmente importante dado que el sistema se compone de varios módulos con funcionalidades distintas (cuantificación de señales horizontales, verticales, defensa, dispositivos, etc.), los cuales pueden integrarse de manera progresiva y permite un bajo acoplamiento siendo que las capas se comunican mediante interfaces mínimas.

Además, esta arquitectura facilita futuras actualizaciones, como la incorporación de nuevas normas de señalización o la extensión del sistema para cubrir nuevos tipos de dispositivos, agregar otros tipos de cálculos o cuantificación. Al mantener las capas desacopladas, cualquier cambio en la lógica en la interfaz puede abordarse sin alterar el resto del sistema.

4.2.4. Diseño de la Interfaz

El propósito de este sistema es ofrecer una experiencia clara y funcional para los ingenieros responsables de la señalización vial, priorizando la facilidad de uso y el acceso rápido a las herramientas de cuantificación y generación de tablas. Se utiliza Java Swing como biblioteca principal para la creación de las ventanas y componentes gráficos, lo que permite construir una interfaz flexible y personalizable.

Estructura general

La interfaz se organiza en ventanas y módulos independientes, que responden a las necesidades específicas de cada etapa del proceso de señalización:

- **Pantalla de inicio:** Permite al usuario crear un nuevo proyecto o cargar uno previamente creado. Incluye campos para definir parámetros iniciales como el nombre del proyecto, velocidad de diseño y tipo de vía.
- **Menú principal del proyecto:** Desde aquí el usuario puede acceder a los diferentes módulos: señalización horizontal, señalización vertical, defensas y barreras, dispositivos y botones.
- **Módulos de cuantificación:** Cada módulo cuenta con una tabla en donde el usuario puede agregar renglones a las tablas correspondientes, además este módulo cuenta con botones que cuentan con formularios especializados para capturar datos técnicos (por ejemplo, tipo de raya, dimensiones de la señal o kilometraje de defensas, cuantificación de pintura), según el módulo que estemos trabajando.
- **Vista previa de tablas:** Un espacio donde el usuario puede revisar el formato final de las tablas generadas antes de exportarlas a AutoCAD, garantizando su compatibilidad y el cumplimiento con los lineamientos oficiales.

Componentes clave

- **Formularios dinámicos:** Diseñados para capturar información de manera estructurada, con validaciones que previenen errores en los datos.
- **Tablas interactivas:** Presentan los resultados de las cuantificaciones con la posibilidad de ordenar, filtrar y editar valores antes de su confirmación final y pasarlo a la base de datos.
- **Botones de acción:** Permiten al usuario ejecutar cálculos, guardar información, navegar entre ventanas y exportar datos sin necesidad de interacciones complejas.
- **Mensajes de validación y confirmación:** Guían al usuario durante el proceso, alertando sobre posibles inconsistencias o confirmando la correcta ejecución de las operaciones.

Flujo de la aplicación

La Figura 4.2.1 muestra el flujo general de navegación del sistema. Este flujo describe la secuencia lógica de interacción del usuario:

1. Inicio del sistema
2. Creación o carga de proyecto
3. Acceso al menú principal
4. Selección de módulo (señalización, defensas, dispositivos, etc.)
5. Captura y edición de datos
6. Generación de tablas
7. Vista previa
8. Exportación a AutoCAD

Este flujo garantiza que el usuario siga un proceso estructurado, evitando omisiones y reduciendo errores durante la captura de información.

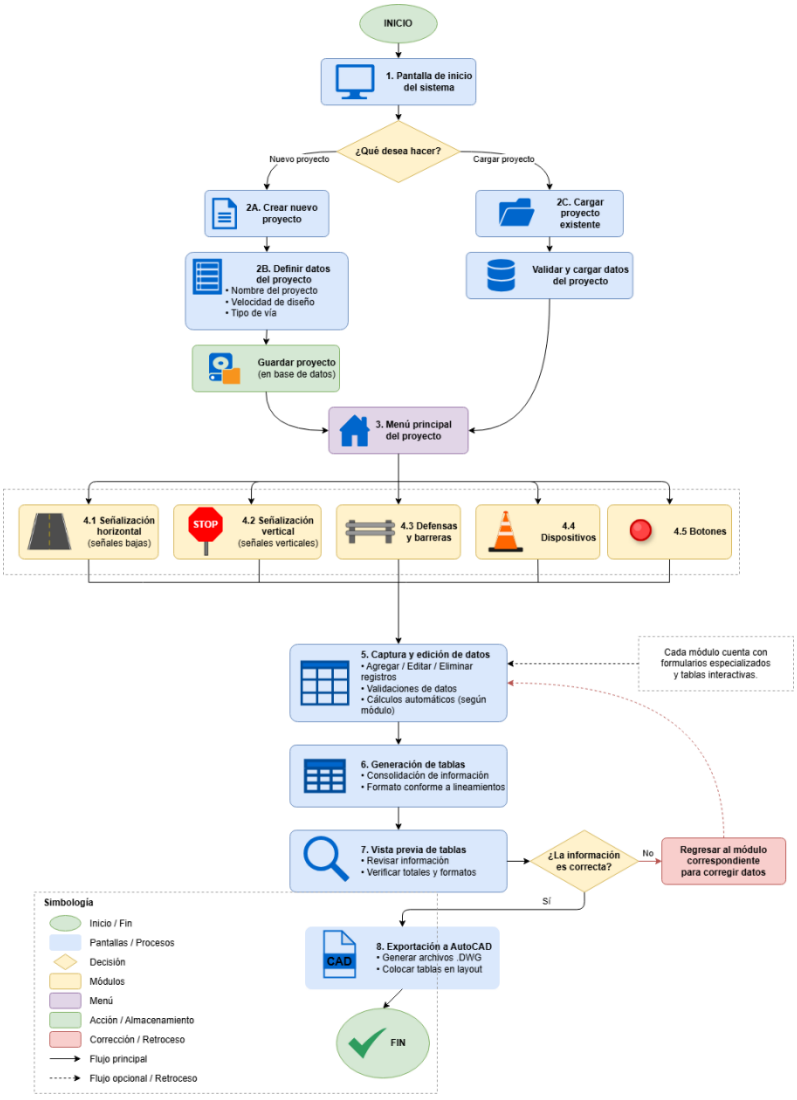


Figura 4.2.1. Diagrama del flujo de trabajo de la aplicación

De este modo, el usuario avanza de manera lógica y ordenada por el proceso, reduciendo tiempos de operación y minimizando errores.

Este enfoque modular y centrado en el usuario no solo facilita el uso inicial del sistema, sino que también permite escalar la aplicación en el futuro, agregando nuevas funcionalidades o adaptándola a actualizaciones normativas sin comprometer su usabilidad.

Criterios de evaluación de la experiencia de usuario.

Para asegurar la calidad de la interfaz, se definieron los siguientes indicadores:

- **Facilidad de uso:** El usuario puede completar tareas sin capacitación extensa.
- **Tiempo de ejecución:** Tiempo necesario para capturar y generar una tabla.
- **Tasa de error:** Número de errores cometidos durante la captura de datos.
- **Consistencia:** Uniformidad en botones, formularios y navegación.
- **Satisfacción del usuario:** Evaluada mediante retroalimentación directa.

Estos criterios permiten medir el desempeño de la interfaz y detectar áreas de mejora.

Prototipos de la interfaz (mockups)

Durante el desarrollo del sistema no se elaboraron prototipos formales o mockups iniciales previos a la implementación. Esto se debe a que el proyecto se desarrolló bajo un enfoque progresivo, en el cual la interfaz y las funcionalidades fueron construyéndose y ajustándose conforme surgían nuevas necesidades por parte del usuario y del flujo de trabajo requerido para la elaboración de proyectos de señalización vial.

El desarrollo de la interfaz se realizó priorizando en una primera etapa la funcionalidad del sistema antes que la apariencia visual. Inicialmente se implementaron las tablas, formularios y botones necesarios para garantizar el correcto funcionamiento de los módulos de captura, cálculo y generación de información. Posteriormente, conforme las funciones principales quedaron operativas, se trabajó en la organización visual de los componentes, la distribución de ventanas y la mejora estética de la interfaz.

Este enfoque permitió adaptar el diseño de manera flexible, ya que el usuario proporcionó libertad creativa durante el desarrollo y las modificaciones se realizaron conforme se identificaban mejoras en la experiencia de uso. De esta manera, la ubicación de botones, tablas, formularios y elementos

de navegación fue refinándose progresivamente para facilitar el flujo de trabajo del usuario y reducir la complejidad operativa del sistema.

4.3. Especificaciones Técnicas

Para el desarrollo del sistema se consideró los requerimientos técnicos necesarios para asegurar un desempeño eficiente, compatibilidad con las herramientas existentes y actuales, y escalabilidad a futuro. A continuación, se presentan las especificaciones clave:

Lenguajes y herramientas

- **Lenguaje principal:** Para este caso se usará Java SE 21, por su robustez, portabilidad y compatibilidad además de ser la última versión de soporte a largo plazo (LTS).
- **Bibliotecas utilizadas:**
    - **Interfaz gráfica (Swing y AWT):**

      javax.swing.* y java.awt.* para la construcción de formularios, botones, tablas y paneles.

      Ayudan a la Personalización de componentes gráficos mediante clases como Graphics2D, RenderingHints, GlyphVector para mejorar la presentación de textos y formas.
    - **Gestión de eventos:**

      ActionListener, MouseAdapter y TableModelListener para la captura de acciones del usuario y actualización dinámica de tablas.
    - **Conexión a base de datos:**

      java.sql.* (Connection, PreparedStatement, ResultSet) para la interacción con la base de datos relacional.
    - **Manejo de archivos:**

      java.io.* (FileReader, BufferedReader, FileWriter, PrintWriter) para la lectura y escritura de datos en archivos de texto.

      javax.swing.JFileChooser para la selección de archivos y rutas desde la interfaz gráfica.
    - **Colecciones y estructuras de datos:**

      java.util.* (ArrayList, HashMap, Map) para el almacenamiento y manipulación de datos en memoria.

Base de datos

El sistema utiliza una **base de datos relacional** para almacenar los proyectos y los elementos cuantificados (señalamiento horizontal, vertical, defensas, botones y dispositivos).

Para su implementación se empleará XAMPP como entorno de desarrollo, aprovechando su servidor MySQL para gestionar los datos de manera eficiente. Esta elección permite:

- Pruebas locales rápidas sin necesidad de un servidor externo.
- Compatibilidad con entornos de producción, ya que MySQL es ampliamente usado en servidores reales.
- Escalabilidad, permitiendo migrar la base de datos a un servidor.

Cada proyecto que se realice con la aplicación estará centralizado en una tabla principal (**proyectos**) y vinculado con tablas secundarias que almacenan los distintos elementos de señalización, mediante relaciones con claves foráneas.

Formatos de exportación

La integración con AutoCAD no se realiza de manera directa, sino mediante un intercambio controlado de archivos de texto y el uso de scripts AutoLISP personalizados.

El flujo de trabajo es el siguiente:

1. **Carga de archivos en AutoCAD:** Desde la aplicación Java, el usuario puede abrir un archivo DWG. Al hacerlo, el sistema ejecuta automáticamente un script AutoLISP que habilita comandos personalizados dentro del entorno de AutoCAD.
2. **Obtención de datos desde AutoCAD:** Cuando el usuario realiza operaciones en AutoCAD (por ejemplo, medir distancias de líneas o cuantificar elementos), el script AutoLISP guarda los resultados en archivos de texto (.txt). Posteriormente, la aplicación Java lee estos archivos, interpreta los datos y los integra a sus tablas.
3. **Envío de datos hacia AutoCAD:** La aplicación Java puede generar archivos de texto en formato CSV, que contienen tablas cuantificadas (señalización, defensas, dispositivos, etc.). Estos archivos son leídos por los scripts AutoLISP para cargar las tablas directamente dentro del plano, manteniendo el formato requerido por la SICT.

Este esquema permite una interacción flexible entre AutoCAD y el sistema, evitando dependencias complejas y asegurando que los datos puedan actualizarse de manera bidireccional.

Escalabilidad

La aplicación sigue un esquema modular, lo que permite incorporar nuevas funcionalidades como:

- Nuevos tipos de elementos viales.
- Ajustes por cambios en el Manual de Señalización y Dispositivos para el Control del Tránsito (2023).
- Agregar más funcionalidades que el usuario en el futuro considere necesarias.

Con estas especificaciones, el sistema mantiene un flujo de trabajo eficiente entre el cálculo, la cuantificación y la representación gráfica de los datos en AutoCAD.

5. Desarrollo

En este capítulo se describe el proceso de implementación del sistema, abordando la estructura interna del software, los módulos que lo conforman y la integración con AutoCAD. El proyecto se desarrolló siguiendo la programación orientada a objetos utilizando el patrón Modelo-Vista-Controlador (MVC) y el Modelo de Desarrollo en Espiral, lo que permitió una construcción iterativa, con retroalimentación continua y control de riesgos, esto facilito la comunicación con el usuario y así poder hacer cambios o correcciones de cada modelo en su momento.

Cada ciclo del modelo en espiral permitió planificar e integrar nuevas funciones del sistema, especialmente aquellas que dependían de interacción directa con AutoCAD o de validaciones técnicas proporcionadas por el ingeniero civil encargado de los requerimientos. Esto posibilitó refinar gradualmente los cálculos, formatos y flujos de trabajo hasta cumplir con los criterios técnicos solicitados.

Estructura general del desarrollo

El sistema se organizó en tres capas principales:

- **Modelo**

  Incluye las clases que representan los diferentes elementos viales (señales horizontales, señales verticales, defensas, dispositivos y botones), así como la lógica para administrar proyectos y realizar cálculos. En este nivel también se encuentran los procedimientos para la conexión y manipulación de la base de datos en MySQL, operando sobre el entorno de desarrollo local con XAMPP.

Durante los ciclos del modelo en espiral, este módulo se fue ajustando para soportar nuevos tipos de datos, cambios de estructura y reglas técnicas proporcionadas durante la validación de Requerimientos.

- **Vista**

  Corresponde a las interfaces creadas con Swing, diseñadas para facilitar la captura de información, el control de proyectos y la visualización de cálculos y tablas.
  La presentación gráfica también evolucionó conforme avanzaron las iteraciones del modelo en espiral, incorporando mejoras solicitadas para hacer más claro el flujo de trabajo y permitir acciones como importar datos desde AutoCAD o exportar tablas.

- **Controlador**

  Administra la comunicación entre la vista y el modelo. Procesa eventos de usuario (botones, menús, formularios), ejecuta cálculos, valida datos y coordina la transferencia de información entre la aplicación y AutoCAD. En este componente se integraron progresivamente las rutinas de importación y exportación, así como los mecanismos para generar y leer archivos producidos por los scripts que se ejecutan en AutoCAD.

Integración con AutoCAD

De forma paralela al desarrollo del sistema, se construyeron scripts en AutoLISP para permitir la captura automatizada de información desde los planos y la inserción de tablas calculadas por la aplicación.

Este proceso también siguió el modelo en espiral: cada comando se probó en condiciones reales, se ajustaron los parámetros y posteriormente se integró con la interfaz Java.

Los mecanismos implementados incluyen:

- **Exportación desde AutoCAD hacia Java**

  Los scripts AutoLISP ejecutados por el usuario en AutoCAD permiten obtener información directamente del dibujo (como ubicaciones, longitudes, cantidades o características del elemento).

  Estos datos se almacenan temporalmente en archivos simples (TXT), que son luego interpretados por la aplicación Java.

- **Exportación desde Java hacia AutoCAD**

  La aplicación genera archivos estructurados (en TXT con estructura de CSV) que AutoCAD utiliza para construir tablas con el formato establecido.

  El diseño se mantuvo genérico durante el desarrollo, debido a que los formatos definitivos dependen de especificaciones técnicas que se definen según los requerimientos del proyecto.

Enfoque iterativo aplicado

El desarrollo del sistema se realizó siguiendo el Modelo en Espiral, el cual permitió avanzar mediante ciclos que integran análisis, construcción y revisión continua. Cada iteración del proyecto se estructuró en los siguientes cuatro pasos:

1. **Identificación de objetivos y Requerimientos**

   En esta fase se definían las funciones que debía incluir cada módulo, tomando como base los requerimientos proporcionados por el ingeniero civil. Aquí se establecían los alcances de cada iteración, como capturar señales, calcular longitudes, generar tablas o integrar comandos con AutoCAD.

2. **Análisis y evaluación de riesgos**

   Antes de programar, se identificaban posibles dificultades técnicas, tales como compatibilidad entre versiones de AutoCAD, estructura de los bloques, interpretación de coordenadas o manejo de archivos temporales. Esto permitió anticipar problemas y planificar soluciones viables para evitar retrabajos.

3. **Desarrollo del prototipo y construcción**

   Con los riesgos evaluados, se implementaban prototipos funcionales que permitían validar rápidamente la lógica del módulo: lectura de datos desde AutoCAD, cálculos en Java, generación de tablas o visualizaciones en la interfaz. A partir del prototipo validado, se procedía a construir la versión estable del componente.

4. **Pruebas y retroalimentación**

   Después de construir cada parte del sistema, se realizaban pruebas directas en planos reales o ejemplos proporcionados. Con ello se obtenía retroalimentación para ajustar cálculos, mejorar la interfaz, reorganizar catálogos o perfeccionar los comandos AutoLISP. Los resultados de estas pruebas alimentaban el siguiente ciclo de la espiral.

Un ejemplo de la aplicación del modelo en espiral ocurrió durante el desarrollo del módulo de señales verticales y generación de tablas para AutoCAD.

En una primera iteración, se desarrolló un prototipo capaz de capturar información básica desde AutoCAD, usando un archivo LIPS para poder seleccionar 3 textos (Nombre, Dimensión y ubicación) correspondientes a las señales verticales, y guardar los textos en un archivo TXT, para después organizar los textos dentro de la tabla con los datos de cantidad, ubicación, código, señal, dimensión y descripción. Esta primera versión permitía validar la comunicación entre AutoCAD y la aplicación Java, así como comprobar la viabilidad de las asignaciones correspondientes, por ejemplo, si se recibía un "SP-6" entonces en la descripción se le asignaba "CURVA".

Posteriormente, el ingeniero civil encargado de validar los requerimientos realizó observaciones sobre el funcionamiento del módulo. Entre la retroalimentación proporcionada se indicó que:

- Algunas clasificaciones de señales verticales no coincidían con la nomenclatura utilizada en proyectos reales, en este caso el uso de minúsculas y mayúsculas correspondientes a la descripción asignada.
- Era necesario considerar que debe haber una tabla para cada sentido de circulación (máximo 2), pero que debe haber una tabla total de ambos sentidos.
- Se requería incorporar condiciones especiales, como cuando se tienen un camino con dos sentidos de circulación, hay veces que una señal está en la misma ubicación en ambos sentidos, entonces se debe considerar que al guardar en la base de datos no se considere que la señal esta repetida y por ello ya no se guarde en la base.

A partir de esta retroalimentación se inició una nueva iteración del modelo en espiral, donde se modificó la estructura de datos, para hacer las asignaciones correctas de la descripción conforme al señal recopilada de AutoCAD, se ajustaron las interfaces gráficas. Además, se realizaron cambios en el guardado en la base para poder tener 2 señales iguales, pero preguntándole antes al usuario si la señal realmente debe de estar repetida o no, y en el formato de las tablas generadas para alinearlos con los criterios técnicos solicitados.

Finalmente, la nueva versión fue nuevamente probada sobre planos reales, permitiendo validar que los cambios realizados mejoraban tanto la precisión de los resultados como el flujo de trabajo del usuario. Este proceso de construcción, evaluación y ajuste continuo se repitió en otros módulos del sistema, como señales horizontales, defensas, barrearas, botones y dispositivos de seguridad vial.

5.1. Estructura de paquetes y clases

La estructura del proyecto se diseñó con el objetivo de mantener una organización clara entre los distintos módulos y facilitar el mantenimiento del código. Los archivos están distribuidos principalmente en paquetes que agrupan las clases según su función dentro del sistema la cual tiene la siguiente distribución:

```
DiseñosCarreteros/
└── src/ (Modelo)
    ├── AutoCAD/
    │   └── CargarAutoCAD.java
    ├── BD/
    │   └── Conexion.java
    ├── Clases/
    │   ├── BarreraTotal.java
    │   ├── Botones.java
    │   ├── Defensa.java
    │   ├── Dispositivos.java
    │   ├── PanelConFondo.java
    │   ├── Proyecto.java
    │   ├── SeñalesHorizontales.java
    │   └── SeñalesVerticales.java
    ├── Exportacion/
    │   ├── FrmExportarTablaBarreras.java
    │   ├── FrmExportarTablaDefensa.java
    │   ├── FrmExportarTablaDispYBtn.java
    │   ├── FrmExportarTablasSH.java
    │   └── FrmExportarTablasSV.java
    ├── Home/
    │   ├── FrmCargarProyecto.java
    │   ├── FrmEleccionSeñalamiento.java
    │   ├── FrmHome.java
    │   └── FrmNuevoProyecto.java
    ├── Imagenes/
    ├── Lisp/
    ├── TXT/
```

```
└── Vistas/
      ├── FrmBarreraTotal.java
      ├── FrmBase.java
      ├── FrmDefensa.java
      ├── FrmDispositivos.java
      ├── FrmEleccionSH.java
      ├── FrmMarcasSH1.java
      ├── FrmMarcasSH2.java
      ├── FrmMarcasSH3.java
      ├── FrmMarcasSH4.java
      └── FrmSeñalesVerticales.java
```

En primer lugar, se encuentra el paquete Clases, que contiene componentes personalizados utilizados en la interfaz gráfica. Un ejemplo de ello es la clase PanelConFondo, que extiende JPanel para permitir la inclusión de imágenes de fondo en las ventanas, aportando un diseño visual más atractivo, al igual que están los módulos que permitirán la comunicación entra las vistas y la base de datos.

En los paquetes de Home, Exportación y Vistas se encuentran las clases responsables de la construcción de las ventanas y formularios que interactúan con el usuario. Estas clases, en su mayoría, heredan de JFrame y hacen uso de componentes de Swing como JTable, JButton, JLabel y JFileChooser. Además, se incluyen escuchadores de eventos (ActionListener y MouseAdapter) que permiten gestionar las acciones del usuario, como el clic en botones o la interacción con tablas, y personalizaciones visuales mediante el uso de Graphics2D y RenderingHints para mejorar la calidad del renderizado de textos e imágenes.

Por otro lado, el proyecto incorpora la conexión a bases de datos utilizando la API de conectividad de base de datos Java (JDBC), con clases que gestionan la apertura de conexiones (Connection), la preparación y ejecución de consultas (PreparedStatement) y la lectura de resultados (ResultSet). Este módulo permite que los datos puedan ser almacenados y consultados dinámicamente desde el sistema.

Asimismo, se incluye un conjunto de clases y métodos para la lectura y escritura de archivos. Se emplean clases como BufferedReader, FileReader, BufferedWriter y PrintWriter para importar y exportar información, lo que servirá para la comunicación entre AutoCAD y el programa.

En conjunto, esta organización de paquetes y clases permite que el proyecto mantenga una separación lógica entre sus diferentes componentes: la interfaz gráfica, las operaciones de base de datos y el manejo de archivos. Esto no solo mejora la claridad del código, sino que también facilita futuras modificaciones o ampliaciones del sistema.

5.2. Implementación del modelo

Analizando los datos que requerimos recopilar para la creación de las tablas solicitadas por el usuario, se crearon las siguientes clases para recabar la información necesaria. En cada una de ellas se aplicó el modelo en espiral, lo que permitió refinar atributos, funciones y riesgos técnicos conforme se avanzaba en las iteraciones del proyecto.

Para el señalamiento horizontal o pintura se usa la clase ***SeñalesHorizontales*** y tiene la siguiente estructura:

```
package Clases;

import BD.Conexion;

public class SeñalesHorizontales {
	int id;
	int distancia;
	String tipo_raya;
	String clasificacion;
	int total_pintura;
	int proyecto_id;
	String marca;
	String lado;
	int cantidad;
	Conexion Uri = new Conexion();

	public int getCantidad() {
		return cantidad;
	}

	public void setCantidad(int cantidad) {
```

```
            this.cantidad = cantidad;
        }

        public String getLado() {
            return lado;
        }

        public void setLado(String lado) {
            this.lado = lado;
        }

        public String getMarca() {
            return marca;
        }

        public void setMarca(String marc) {
            marca = marc;
        }

        public int getId() {
            return id;
        }

        public void setId(int id) {
            this.id = id;
        }

        public int getDistancia() {
            return distancia;
        }

        public void setDistancia(int distancia) {
            this.distancia = distancia;
        }

        public String getTipo_raya() {
            return tipo_raya;
        }
```

```
	public void setTipo_raya(String tipo_raya) {
		this.tipo_raya = tipo_raya;
	}

	public String getClasificacion() {
		return clasificacion;
	}

	public void setClasificacion(String clasificacion) {
		this.clasificacion = clasificacion;
	}

	public int getTotal_pintura() {
		return total_pintura;
	}

	public void setTotal_pintura(int total_pintura) {
		this.total_pintura = total_pintura;
	}

	public int getProyecto_id() {
		return proyecto_id;
	}

	public void setProyecto_id(int proyecto_id) {
		this.proyecto_id = proyecto_id;
	}

	//Funcion para agregar SH en la BD
	public boolean AgregarSeñalBaja() {
		if (Uri.AgregarSeñalBaja(this)) {
			return true;
		} else {
			return false;
		}
	}

	//Funcion para eliminar SH en la BD
	public boolean EliminarSeñalBaja() {
```

```
                if (Uri.EliminarSeñalBaja(this)) {
                        return true;
                } else {
                        return false;
                }
        }

        //Funcion para regresar todas las SH que corresponda a un tipo especifico de SH en la
BD
        public SeñalesHorizontales[] RegresarTabla(Proyecto proy, String marca) {
                SeñalesHorizontales[] resultado = Uri.RegresarTabla(proy, marca);
                if (resultado != null && resultado.length > 0) {
                        return resultado;
                } else {
                        return null;
                }
        }

        //Funcion para regresar todas las SH en la BD
        public SeñalesHorizontales[] RegresarTablaCom(Proyecto proy) {
                SeñalesHorizontales[] resultado = Uri.RegresarTablaCom(proy);
                if (resultado != null && resultado.length > 0) {
                        return resultado;
                } else {
                        return null;
                }
        }

}
```

Lo que se hace en esta clase es recabar información que se considera importante almacenar de las señales horizontales, las cuales serían id, distancia, tipo_raya, clasificación, total_pintura, proyecto_id, marca, lado y cantidad. Cada una sirve como dato para los cálculos correspondientes en las vistas.

La variable “Uri” gestiona la comunicación con la base de datos mediante las funciones RegresarTablaCom, RegresarTabla, AgregarSeñalBaja y EliminarSeñalBaja.

Esta clase se desarrolló en una espira enfocada en la cuantificación de señalamiento horizontal**,** donde se evaluaron riesgos relacionados con lectura de distancias, estandarización de tipos de raya y compatibilidad con las entradas de AutoCAD.

Y se realiza algo parecido para las demás clases, en la clase ***Botones*** se administra la información relacionada con los botones que deben colocarse sobre las marcas de pintura o el señalamiento horizontal en este caso se incluyen atributos como id, clave, color, raya, ubicación, características, cantidad, proyecto_id e id_señal_baja. Estos valores permiten identificar y organizar los botones asociados a las señales horizontales, y para su gestión se emplean las funciones AgregarBotones, EliminarBotones y RegresarTablaBotones, que se encargan de insertar, eliminar y obtener registros desde la base de datos mediante la conexión establecida con la variable "Uri".
Esta clase se desarrolló en una espira centrada en el cálculo de botones, donde se analizaron riesgos como el tipo de raya que requiere botones y la correcta asociación con las señales horizontales.

De manera similar, la clase ***SeñalesVerticales*** concentra los datos necesarios para trabajar con señales verticales, como id, ubicación, señal, dimensión, descripción, código, cantidad y proyecto_id. Con ellos se logra un control detallado de cada elemento de señalización en un proyecto, y a través de la conexión con la base de datos se implementan funciones como AgregarSeñalVertical, EliminarSeñalVertical y RegresarTablaSeñVert, que aseguran la correcta administración de esta información.
Esta clase se desarrolló en una espira dedicada al señalamiento vertical, evaluando riesgos como variación de tipos de señales, dimensiones y la diferencia entre proyectos de Camino Abierto y Entronque.

La clase ***Defensa*** se orienta a almacenar y manejar datos de las defensas utilizadas en un tramo de carretera, registrando atributos como id, km_inicial, km_final, posición, cantidad de amortiguadores, longitud y proyecto_id. Estos datos son clave para definir la ubicación y características de cada defensa colocada sobre un poryecto. Gracias a la variable "Uri", se pueden ejecutar las funciones AgregarDefensa, EliminarDefensa y RegresarTablaDefensa, que cumplen con la tarea de gestionar estas estructuras en la base de datos.

Esta clase se desarrolló en una espira enfocada en la cuantificación de defensas, evaluando riesgos como el cálculo correcto de kilometrajes, detección del lado del camino y ordenamiento de tramos.

En cuanto a la clase ***Dispositivos***, esta se enfoca en los elementos adicionales del proyecto, tales como dispositivos de señalización o apoyo. Sus atributos son id, nombre del dispositivo, clave, posición, cantidad, unidad y proyecto_id. Con esta información se logra identificar y cuantificar los dispositivos implementados en el proyecto, mientras que las funciones AgregarDispositivos, EliminarDispositivos y RegresarTablaDispositivos permiten mantener actualizada la información en la base de datos a través de la conexión "Uri".
Esta clase se desarrolló en una espira enfocada en los dispositivos auxiliares, revisando riesgos como la variabilidad de tipos, claves y cantidades capturadas desde AutoCAD.

Por su parte, la clase ***Proyecto*** cumple un papel central, ya que en ella se definen los datos generales del proyecto en sí: id, nombre, velocidad de diseño, tipo de camino y la lista de proyectos registrados. A través de esta clase se pueden agregar proyectos, obtener el identificador de uno en curso y consultar la lista de proyectos almacenados. Las funciones AgregarProyecto, ObtenerIdProyecto y RegresarProyectos, junto con la variable "Uri", hacen posible la interacción directa con la base de datos para el control de la información global.
Esta clase se desarrolló en una espira inicial destinada a definir la estructura del proyecto, donde se evaluaron riesgos de identificación, persistencia y manejo de múltiples proyectos.

Finalmente, la clase ***BarreraTotal*** concentra los datos referentes a los sistemas de barreras de contención. Entre sus atributos se encuentran id, nivel de contención, número de barreras metálicas, barreras tipo New Jersey, secciones de rigidez y amortiguación, amortiguadores, secciones tipo cola de pato e id_proyecto. Estos valores permiten caracterizar de manera precisa las barreras en un proyecto. Las funciones principales son ActualizarBarreras y RegresarTablaBarreras, que posibilitan la modificación y consulta de los datos en la base de datos, siempre a través de la conexión que provee la variable "Uri".

Esta clase se desarrolló en una espira dirigida a la cuantificación total de barreras, donde se analizaron riesgos como la clasificación normativa de barreras, determinar el nivel de contención, combinaciones de elementos y validación de totales.

PanelConFondo

Otra clase que usaremos será la clase de ***PanelConFondo*** la cual la usaremos para poner un fondo a todas las vistas y que esta imagen se auto redimensione según el tamaño de la ventana.

Esta clase tiene una función visual dentro de la interfaz gráfica del sistema, ya que permite personalizar los paneles con una imagen de fondo. Sus atributos y métodos están orientados al manejo de imágenes dentro de un JPanel. En esta clase, se define el atributo imagen, que almacena la referencia de la imagen a utilizar, y se implementan métodos como el constructor, que inicializa el panel con la ruta de la imagen a mostrar, el método setFondo, que permite cambiar dinámicamente la imagen de fondo y redibujar el panel, y finalmente el método sobrescrito paintComponent, encargado de renderizar la imagen escalada a las dimensiones actuales del panel. Con ello se logra una mejor experiencia del usuario dentro de la aplicación.

5.3. Implementación de la base de datos

El sistema requiere de un mecanismo que permita almacenar de manera organizada la información correspondiente a los proyectos y a los distintos elementos de señalización. Para ello, se diseñó una base de datos que establece la estructura necesaria para mantener la integridad de los datos y garantizar la relación entre cada proyecto y los elementos asociados a este.

En primer lugar, se encuentra la tabla de Proyectos, que funciona como entidad principal y contiene los datos generales como nombre, velocidad de diseño y tipo de vía. A partir de esta tabla se derivan las relaciones con las demás tablas, de modo que cada registro en las tablas consecuentes queda vinculado mediante la clave foránea proyecto_id.

Como parte del modelo en espiral, la estructura de cada tabla fue diseñada y refinada en distintas espiras, validando riesgos como campos necesarios, relaciones, dependencias y compatibilidad con los módulos de captura y AutoCAD.

Posteriormente, se definieron tablas específicas para cada módulo del sistema:

- **señales_bajas**, destinada a almacenar la información de las rayas de señalamiento horizontal, incluyendo distancia, clasificación, marca, lado, cantidad y el total de pintura requerido.
- **señales_verticales:** En la que se registran los datos relacionados con las señales físicas como ubicación, tipo de señal, dimensiones, código, descripción y cantidad.
- **defensa**: Que permite guardar información de los tramos con defensa metálica, indicando los kilómetros de inicio y fin, posición, cantidad de amortiguadores y longitud total.
- **botones:** Orientada al registro de botones que se relacionan con las señales bajas, se guarda datos como clave, color, tipo de raya, ubicación, características y cantidad.
- **dispositivos:** Que gestiona la información de dispositivos adicionales, incluyendo clave, posición, cantidad y unidad de medida.
- **barrera:** Enfocada en el registro de barreras metálicas y de concreto, así como secciones de rigidez, amortiguamiento y elementos especiales como amortiguadores y colas de pato.

Cada una de estas tablas está vinculada a la tabla de proyectos mediante relaciones foráneas, asegurando la integridad referencial de los datos. Además, se implementaron restricciones ON DELETE CASCADE y ON UPDATE CASCADE para evitar inconsistencias o errores, de manera que al eliminar o actualizar un proyecto, las entidades relacionadas se modifiquen en consecuencia. En conjunto, esta implementación asegura que el sistema cuente con una base para el almacenamiento y consulta de información, con un diseño que facilita tanto la ampliación como el mantenimiento de la base de datos.

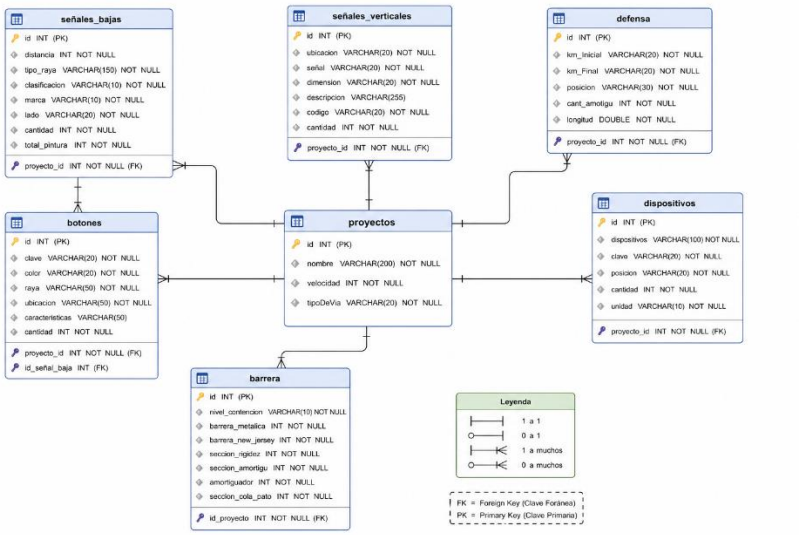


Figura 5.3.1. Diagrama entidad–relación de la base de datos

La estructura descrita anteriormente puede observarse de manera gráfica en la Figura 5.3.1, donde se presenta el diagrama entidad–relación de la base de datos implementada. En este diagrama, la tabla proyectos se muestra como la entidad central del sistema, debido a que concentra la información principal de cada proyecto y establece relaciones con los demás módulos de la base de datos. Cada una de las tablas secundarias se conecta mediante claves foráneas, lo que permite mantener una relación organizada entre los distintos elementos de señalización y el proyecto al que pertenecen.

La figura también permite identificar las relaciones de cardinalidad existentes entre las entidades. La relación es de uno a muchos, lo que significa que un proyecto puede contener múltiples registros de señales bajas, señales verticales, defensas, dispositivos, botones y barreras. Asimismo, se observa la relación entre las tablas señales_bajas y botones, donde una señal baja puede asociarse con varios botones, garantizando una estructura coherente para el almacenamiento de elementos complementarios del señalamiento horizontal.

De igual manera, el diagrama muestra las claves primarias (PK) y claves foráneas (FK) utilizadas para asegurar la integridad referencial de la información. Esto permite que cada registro permanezca correctamente vinculado dentro del sistema y evita inconsistencias durante operaciones de actualización o eliminación. Además, la representación visual facilita la comprensión de la distribución de datos y la interacción entre los módulos del sistema, contribuyendo tanto al mantenimiento como a futuras ampliaciones de la base de datos.

5.4. Desarrollo de la interfaz gráfica (Vista)

Para la parte visual se crearon varias ventanas donde se muestran los datos de las tablas que se guardan en la base datos y también ventanas que muestra el resultado final antes de mandar a AutoCAD.

Consideraciones sobre las Rutas Temporales

Para la interacción entre AutoCAD y el programa que se desarrolló, se utilizan archivos de texto que almacenan resultados generados por los scripts (LIPS) (como distancias, ángulos, áreas o textos seleccionados). Estos archivos permiten que el programa en Java procese la información de manera independiente al entorno de AutoCAD.

Para que todos los archivos se guardan en una carpeta específica dentro de Documentos del usuario:

```
protected String rutaTemp = System.getProperty("user.home") + "\\Documents\\MisLisp\\";
```

De esta forma, sin importar el equipo donde se ejecute la aplicación, el sistema localizará automáticamente la carpeta del usuario y dentro de ella generará el directorio MisLisp, que funcionará como repositorio temporal.

FrmHome

Primeramente, se muestra la ventana de inicio, la que contine el Método principal, la cual inicializa el programa, en ella solo se busca que el usuario elija si quiere iniciar un nuevo proyecto o cargar uno nuevo, entonces en este caso solo tiene dos botones los cuales nos redirigen cuando el usuario quiera iniciar un nuevo proyecto o cuando quiera cargar una ya guardado en la base de datos, esta clase se llama ***FrmHome***, esta clase extiende de JFrame y utiliza el panel personalizado de la clase ***PanelConFondo*** para mostrar una imagen de fondo. En ella se definen dos botones principales: “Cargar Proyecto” y “Nuevo Proyecto”, que permiten al usuario abrir un proyecto existente o crear uno nuevo respectivamente.

Un aspecto importante dentro de esta clase es el método estilizarBoton, el cual se encarga de personalizar la apariencia y comportamiento de los botones cuando el mouse pasa sobre ellos. Esta función recibe como parámetros el botón que se desea modificar y el color que tendrá como fondo principal. Entre sus funciones más relevantes se encuentran:

- **Propiedades básicas del botón:** Se desactiva el pintado por defecto de bordes y área de contenido (setBorderPainted(false), setContentAreaFilled(false)), lo que permite darle una apariencia personalizada, cambiando de igual forma la fuente (setFont) y el color del texto (setForeground).
- **Cambio de cursor:** Se establece el cursor en forma de mano (Cursor.HAND_CURSOR), lo cual permite al usuario saber que el botón es interactivo.
- **Efectos al pasar el mouse:** Se implementa un MouseListener que detecta cuándo el ratón entra o sale del área del botón.
    - Al **entrar**, el color de fondo se oscurece ligeramente (colorFondo.darker()), se activa el relleno del área de contenido y se vuelve opaco, logrando un efecto de resaltado.
    - Al **salir**, el botón vuelve a su estado inicial, transparente y con su color original, simulando un efecto dinámico de "hover".
- **Texto con borde (contorno):** Se redefine la interfaz de usuario del botón (setUI) para sobrescribir la forma en que se pinta el texto. Aquí se utilizan objetos como Graphics2D, FontMetrics y GlyphVector para:
    - Calcular la posición exacta del texto en el centro del botón.
    - Generar la forma vectorial de cada carácter.
    - Dibujar un contorno negro alrededor del texto mediante un bucle que desplaza la forma en todas las direcciones, simulando un borde grueso.
    - Finalmente, rellenar el texto principal con el color original del botón, logrando un contraste fuerte y mayor legibilidad sobre cualquier fondo.

Gracias al método estilizarBoton, los botones adquieren un aspecto diferente al que viene por defecto de la función JBotton, este método se usara en todas las clases para cambiar la apariencia de los botones, facilitando la navegación dentro de la aplicación.

FrmNuevoProyecto

Si el usuario eligió un Nuevo Proyecto entonces se nos redirige a la clase de ***FrmNuevoProyecto*** la cual tiene la función de guardar los datos necesarios para un proyecto esto a través de un breve

cuestionario donde se le solita al usuario que ingrese el nombre, el tipo y la velocidad del proyecto que se va estar trabajando.

La clase ***FrmNuevoProyecto*** permite al usuario registrar un nuevo proyecto dentro del sistema. Para ello se extiende de JFrame y construye una ventana con un con un fondo personalizado, sobre el cual se colocan los elementos gráficos. Que serían los campos para que el usuario agregue la información solicitada, además de una lista desplegable que permite seleccionar el tipo de camino entre las opciones de carretera, calle o vía ciclista. También hay un botón de Aceptar, que valida que los campos no estén vacíos, verifica que la velocidad se pueda convertir a un número entero y guarda estos datos en una clase ***Proyecto*** para, después, guardar la información en la base de datos. En caso de éxito, se muestra un mensaje de confirmación y se abre la ventana ***FrmEleccionSeñalamiento***, mientras que si ocurre algún error se notificará al usuario mediante cuadros de diálogo. Al igual que en otras partes de la aplicación, los botones son estilizados con un método especial que mejora su presentación, aplicando colores, efectos al pasar el ratón y un contorno en el texto para hacerlo más legible, lo que contribuye a mantener una interfaz uniforme y visualmente agradable.

FrmCargarProyecto

Para cuando el usuario elige Cargar Proyecto esta se muestra la venta de la clase ***FrmCargarProyecto*** permite al usuario visualizar y seleccionar proyectos ya existentes dentro del sistema. Crea una ventana con un panel de fondo personalizado sobre el cual se muestra una tabla (JTable) que contiene los proyectos registrados en la base de datos, mostrando columnas para el ID, nombre, velocidad y tipo de camino. Esta tabla está configurada con un efecto de transparencia, logrado principalmente mediante el uso de setOpaque(false) en el JTable y en el JScrollPane (este efecto se repetirá para las demás tablas que se muestran en la aplicación). El usuario puede elegir un proyecto de la tabla y confirmarlo con el botón Aceptar, lo cual toma los datos seleccionados, los asigna a un objeto de la clase Proyecto y abre la ventana ***FrmEleccionSeñalamiento*** para continuar con el flujo del programa. En caso de no seleccionar ninguna fila, se muestra un mensaje de advertencia. Al igual que en otras partes de la aplicación, los botones son estilizados mediante un método que les otorga un mejor diseño visual.

FrmEleccionSeñalamiento

En el caso de ***FrmEleccionSeñalamiento*** permite al usuario seleccionar que tabla de señalización quiere agregar al proyecto que selecciono o que creo.

Esto sirve como menú principal para decidir qué tipo de elementos viales se van a agregar a un proyecto. Es una extensión de JFrame y genera una ventana con un fondo personalizado sobre el cual se colocan botones que permiten al usuario acceder a distintas opciones: Señales Horizontales, Señales Verticales, Defensas y Dispositivos. Cada uno de estos botones cierra la ventana y abre la interfaz correspondiente, pasando como referencia el proyecto en el que se está trabajando. Además, se incluye un apartado para abrir un archivo DWG de AutoCAD, donde al presionar el botón *Abrir* se muestra un mensaje informativo y luego se abre un explorador de archivos para poder abrir el archivo de AutoCAD para realizar diferentes tareas con LISP, esto ayuda de la clase ***CargarAutoCAD***. En la parte superior se presenta el nombre del proyecto con un doble efecto de texto (blanco con un contorno negro), lo que mejora su visibilidad sobre el fondo.

FrmEleccionSH

Si el usuario va a implementar la tabla de Señales Horizontales esta lo manda a la clase ***FrmEleccionSH*** esta clase permite al usuario elegir entre las diferentes marcas que se pueden agregar, para se le mostrara una serie de botones que redirigen a una venta para agregar la marca correspondiente que se seleccionó. Como se menciona en el capítulo 2.3.1, sobre la señalización Horizontal, hay en total 20 marcas que van de M-1 a M-20, aparte de eso cada marca puede tener submarcas como M-1.1, M-1.2, etc. entonces a través de una extensión de JFrame, se genera una ventana con un fondo personalizado sobre el cual se colocan botones que permiten al usuario acceder a cada uno de los tipos de marcas.

Además, se agrega un botón de Exportar el cual te muestra una ventana donde se muestra la table de Señalamiento Horizontal con las marcas que el usuario haya agradado ya a la base de datos.

Las clases **FrmEleccionSH**, **FrmEleccionSeñalamiento** y **FrmHome** se consideró como parte de la misma espira del modelo en espiral, como ventanas de transición donde los botones funcionan para la navegación entre modelos, donde se considera que como posible error haya una mala vinculación entre ventanas y botones. En el caso de las clases **FrmNuevoProyecto** y

FrmCargarProyecto ambas tienen como función principal la comunicación con la base de datos con la tabla de **proyectos**, para ello se consideró que como posibles errores haya una mala comunicación con la base de datos, además de que se tiene que estar revisando los datos que se mandan desde la aplicación y así como los que se reciben de la base datos al momento en que se muestran en la aplicación.

Ahora bien, para las siguientes vistas del usuario contaran con una estructura similar ya que se muestra una tabla similar a la que se usó en ***FrmCargarProyecto***, con la trasparecía, la forma en que se obtiene los datos y en la asignación del fondo; debido a esto para las siguientes clases solo se mostrara las partes de código que requieran explicación.

FrmBase

Para realizar la tabla de Señalamiento Horizontal, considerando que hay 20 marcas de pintura que se pueden analizar, se creó una clase que sirva como base para las demás clases, es decir se creó una clase plantilla que las demás clases (M1 – M20).

Como parte de la aplicación del modelo en espiral, esta clase se desarrolló a través de varias espiras sucesivas en las que se identificaron riesgos relativos a la estandarización de las 20 marcas de pintura y la necesidad de evitar código duplicado, además de representar correctamente el nombre y su respectiva nomenclatura de manera correcta, se planificó una estructura común adaptable a todas las marcas a trabajar, se construyó un prototipo funcional de la plantilla de la vista mostrar, para validar su comportamiento y reutilización y se evaluó con el usuario para ajustar campos, validaciones y formato de tabla antes de generar el resto de las clases derivadas, como errores a considerar son en su gran parte que los datos recopilados tengan su formato correcto y que cuando estos se hagan cálculos no hay inconsistencias o cambios repentinos entre los tipo de variables.

La clase ***FrmBase*** funciona como una plantilla para las demás ventanas del sistema, ya que concentra la lógica común y define métodos abstractos que luego se implementan en las subclases, evitando así la repetición de código. En ella se definen tanto la estructura general de la tabla como las acciones básicas que estarán presentes en todas las vistas de cada una de las 20 marcas.

Dentro de esta ventana encontramos el botón Aceptar, que tiene la tarea de validar y guardar la información que el usuario introduce en la tabla. Cuando se presiona, primero revisa que las celdas no estén vacías y que los valores tengan el formato correcto, por ejemplo, que la velocidad se pueda

convertir a un número. Si los datos son válidos, se guardan en la base de datos y se notifica al usuario mediante un mensaje de confirmación. En caso de que se presente algún error, como un dato incorrecto o un fallo en la conexión, se muestra un cuadro de diálogo informando el problema y no se procede al guardado.

```
// Exporta datos a la base de datos
            JButton btnAceptar = new JButton("Aceptar");
            btnAceptar.setBounds(636, 420, 88, 30);
            estilizarBoton(btnAceptar, new Color(40, 167, 69));
            btnAceptar.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        if (!cambiosRealizados) {
                              JOptionPane.showMessageDialog(FrmBase.this, "No hay
cambios para guardar.", "Información",
                                          JOptionPane.INFORMATION_MESSAGE);
                              return;
                        }
                        int confirm = JOptionPane.showConfirmDialog(null, "¿Deseas
guardar los cambios?", "Confirmar",
                                    JOptionPane.YES_NO_OPTION);

                        // Verificar la opción seleccionada por el usuario
                        if (confirm == JOptionPane.YES_OPTION) {
                              int exitosos = 0;
                              int fallidos = 0;
                              for (int i = 0; i < tableModel.getRowCount(); i++) {
                                    SeñalesHorizontales mar = new
SeñalesHorizontales();
                                    try {
                                          // Para kilometraje_inicial y
kilometraje_final que son Integer

      mar.setDistancia(Integer.parseInt(tableModel.getValueAt(i, 0).toString()));

      mar.setTipo_raya(tableModel.getValueAt(i, 1).toString());
```

```
        mar.setClasificacion(tableModel.getValueAt(i, 2).toString());

        mar.setTotal_pintura(Integer.parseInt(tableModel.getValueAt(i,
3).toString()));
                                                mar.setProyecto_id(proyecto.getId());
                                                mar.setMarca(marca);
                                                if (marca.equals("M3")) {

        mar.setLado(tableModel.getValueAt(i, 4).toString());

        mar.setCantidad(Integer.parseInt(tableModel.getValueAt(i, 5).toString()));
                                                } else {
                                                        mar.setLado(" ");

        mar.setCantidad(Integer.parseInt(tableModel.getValueAt(i, 4).toString()));
                                                }

                                                // Guarda la nueva raya en la base de
datos
                                                if (mar.AgregarSeñalBaja()) {
                                                        exitosos++;
                                                } else {
                                                        fallidos++;
                                                }

                                        } catch (NumberFormatException ex) {

        JOptionPane.showMessageDialog(FrmBase.this,
                                                                "Error en el formato de
los datos en la fila " + (i + 1), "Error",

        JOptionPane.ERROR_MESSAGE);
                                        }
                                }
```

```
                                JOptionPane.showMessageDialog(FrmBase.this,
                                                "Guardado finalizado.\nFilas guardadas
exitosamente: " + exitosos + "\nFilas con error: "
                                                                + fallidos,
                                                "Resumen", exitosos > 0 ?
JOptionPane.INFORMATION_MESSAGE : JOptionPane.ERROR_MESSAGE);
                        } else if (confirm == JOptionPane.NO_OPTION) {
                                System.out.println("No se guardarán los cambios.");
                        }

                }
        });
        panelFondo.add(btnAceptar);
```

Por otra parte, el botón Eliminar permite borrar filas de la tabla, pero únicamente si el usuario ha seleccionado una previamente. Una vez verificado esto, la fila se elimina del modelo y se actualizan los cálculos correspondientes, garantizando que los totales se mantengan correctos. Si el usuario intenta presionar el botón sin haber seleccionado nada, se le mostrará un mensaje de advertencia solicitando que primero elija una fila.

```
//Elimina los datos seleccionados
        JButton btnEliminar = new JButton("Eliminar Fila");
        btnEliminar.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        int selectedRow = datosRaya.getSelectedRow(); // Obtener
la fila seleccionada del JTable

                        int confirm = JOptionPane.showConfirmDialog(null, "¿Seguro
que deseas eliminar la fila?", "Confirmar",
                                        JOptionPane.YES_NO_OPTION);

                        // Verificar la opción seleccionada por el usuario
                        if (confirm == JOptionPane.YES_OPTION) {
                                if (selectedRow != -1) {
```

```
                                    // Obtener los valores de cada columna
                                    String distancia =
datosRaya.getValueAt(selectedRow, 0).toString();
                                    String tipoRaya =
datosRaya.getValueAt(selectedRow, 1).toString();
                                    String clasificacion =
datosRaya.getValueAt(selectedRow, 2).toString();
                                    String totalPintura =
datosRaya.getValueAt(selectedRow, 3).toString();
                                    String lado;
                                    String cantidad;

                                    if (marca.equals("M3")) {

                                          lado =
datosRaya.getValueAt(selectedRow, 4).toString();
                                          cantidad =
datosRaya.getValueAt(selectedRow, 5).toString();
                                          JOptionPane.showMessageDialog(null,
                                                      "Datos a eleminar:\n" +
"Distancia: " + distancia + "\n" + "Tipo de Raya: "
                                                                  + tipoRaya +
"\n" + "Clasificación: " + clasificacion + "\n"
                                                                  + "Total
Pintura: " + totalPintura + "\n" + "Lado: " + lado + "\n"
                                                                  + "Cantidad:
" + cantidad + "\n",
                                                      "Información",
JOptionPane.INFORMATION_MESSAGE);

                                    } else {
                                          lado = " ";
                                          cantidad =
datosRaya.getValueAt(selectedRow, 4).toString();
                                          JOptionPane.showMessageDialog(null,
```

```
                                                        "Datos a eleminar:\n" +
"Distancia: " + distancia + "\n" + "Tipo de Raya: "
                                                                + tipoRaya +
"\n" + "Clasificación: " + clasificacion + "\n"
                                                                + "Total
Pintura: " + totalPintura + "\n" + "Cantidad: " + cantidad + "\n",
                                                        "Información",
JOptionPane.INFORMATION_MESSAGE);

                                        }

                                        SeñalesHorizontales señal = new
SeñalesHorizontales();

      señal.setDistancia(Integer.parseInt(distancia));
                                        señal.setTipo_raya(tipoRaya);
                                        señal.setClasificacion(clasificacion);
                                        señal.setMarca(marca);

      señal.setTotal_pintura(Integer.parseInt(totalPintura));

      señal.setCantidad(Integer.parseInt(cantidad));
                                        señal.setLado(lado);
                                        señal.setProyecto_id(idProyect);
                                        if (señal.EliminarSeñalBaja()) {
                                              //si se elimina se quita de la lista de
las pinturas originales

      JOptionPane.showMessageDialog(FrmBase.this, "Datos eliminados exitosamente.",
"Éxito",

      JOptionPane.INFORMATION_MESSAGE);
                                              pinturaOriginal.remove(selectedRow);

                                              tableModel.removeRow(selectedRow);
```

```
                                        Map<Integer, Integer> nuevoMapa = new
HashMap<>();
                                        for (int i = 0; i <
tableModel.getRowCount(); i++) {
                                              Object valorCelda =
tableModel.getValueAt(i, 3);
                                              try {
                                                    int pintura =
Integer.parseInt(valorCelda.toString());
                                                    nuevoMapa.put(i, pintura);
                                              } catch (NumberFormatException |
NullPointerException ex) {
                                                    nuevoMapa.put(i, 1);
                                              }
                                        }
                                        pinturaOriginal = nuevoMapa;

                                        cambiosRealizados = true;
                                  } else {

      JOptionPane.showMessageDialog(FrmBase.this, "Error al eliminar los datos ",
"Error",

      JOptionPane.ERROR_MESSAGE);
                                  }
                            } else {
                                  JOptionPane.showMessageDialog(FrmBase.this,
"Por favor, selecciona una fila para eliminar.",
                                        "Advertencia",
JOptionPane.WARNING_MESSAGE);
                            }
                      }
                }
            });
```

```
            btnEliminar.setBounds(575, 361, 150, 30);
            estilizarBoton(btnEliminar, new Color(220, 53, 69));
            panelFondo.add(btnEliminar);
```

Otro aspecto importante es el bloque de código encargado de verificar los cambios en la tabla. Mediante un TableModelListener, la aplicación detecta cada vez que el usuario modifica una celda. En ese momento identifica qué fila y columna fueron editadas y, si la marca no corresponde a “M3”, recalcula de manera automática el total de pintura multiplicando por la nueva cantidad ingresada. Esto permite que los resultados se mantengan actualizados sin necesidad de que el usuario realice operaciones adicionales, reduciendo la posibilidad de errores.

```
//Verifica cambios en la cantidad para multiplicar con el Total de Pintura
            tableModel.addTableModelListener(new TableModelListener() {
                  @Override
                  public void tableChanged(TableModelEvent e) {
                        int row = e.getFirstRow();
                        int column = e.getColumn();
                        if (!"M3".equals(marca)) {

                              if (column == 4) {
                                    int cantidad =
Integer.parseInt(tableModel.getValueAt(row, 4).toString());

                                    if (pinturaOriginal.containsKey(row)) {
                                          try {
                                                int pint =
pinturaOriginal.get(row);
                                                int nuevoTotal = cantidad *
pint;

      tableModel.setValueAt(nuevoTotal, row, 3);
                                          } catch (NumberFormatException ex) {
```

```
      tableModel.setValueAt(pinturaOriginal.get(row), row, 3);
                                          }
                                    } else {

                                          System.out.println("Fila " + row + " no
está en pinturaOriginal aún");
                                    }
                              }
                        }else {
                              if (column == 5) {
                                    int cantidad =
Integer.parseInt(tableModel.getValueAt(row, 5).toString());

                                    if (pinturaOriginal.containsKey(row)) {
                                          try {
                                                int pint =
pinturaOriginal.get(row);
                                                int nuevoTotal = cantidad *
pint;

      tableModel.setValueAt(nuevoTotal, row, 3);
                                          } catch (NumberFormatException ex) {

      tableModel.setValueAt(pinturaOriginal.get(row), row, 3);
                                          }
                                    } else {
                                          System.out.println("Fila " + row + " no
está en pinturaOriginal aún");
                                    }
                              }
                        }
                  }
            });
```

Finalmente, ya que en esta clase sirve como base, aquí se definen los métodos comunes y se establecen las funciones que deben implementarse en cada clase hija, como CargarDatos, manejarBtnAgregar, agregarRaya, leerArchivo, los cuales permiten que cada subclase tenga sus propias particularidades sin perder coherencia con el resto del sistema.

El método cargarDatos tiene la tarea de consultar la base de datos y traer la información relacionada con el proyecto en curso, en particular las rayas que ya se habían guardado anteriormente. Una vez obtenidos los registros, cada fila se agrega al DefaultTableModel que alimenta la tabla de la interfaz, de modo que el usuario puede visualizar, modificar o eliminar los datos. Esta función garantiza que al abrir nuevamente una vista no se pierda la información previa, manteniendo la continuidad del trabajo.

El método manejarBtnAgregar se activa cuando el usuario presiona el botón "Agregar". Su función es validar que los campos necesarios tengan información correcta antes de proceder con la inserción. Una vez verificados, este método llama a agregarRaya para que el nuevo dato se integre tanto en la tabla como en la base de datos. De esta manera, manejarBtnAgregar actúa como el intermediario que gestiona la interacción del usuario con la tabla y asegura que el flujo de agregado de información se realice de forma ordenada, este método se usa en la mayoría de las clases que usamos ya nos manda una ventana con las instrucciones y componentes para obtener los datos que se requieren para poder hacer los cálculos correspondientes.

El método agregarRaya es el encargado de añadir un nuevo registro de raya al proyecto. Para ello, primero recibe los valores proporcionados (clasificación, dimensiones, cantidad, entre otros) y los incorpora en una nueva fila dentro del modelo de la tabla. Posteriormente, también guarda esa información en la base de datos asociada al proyecto, asegurando que no se trate de un cambio temporal, sino que quede almacenado de manera permanente. Con esta función se mantiene sincronizada la vista con la información en memoria y con los datos persistentes.

El método leerArchivo permite cargar en la tabla datos que provienen de un archivo externo. Este proceso se utiliza, por ejemplo, cuando la aplicación obtiene información de AutoCAD a través de un archivo generado previamente. La función abre el archivo, interpreta el contenido y lo organiza en filas dentro del modelo de la tabla para que el usuario pueda visualizarlo y, si es necesario, realizar

modificaciones. Gracias a leerArchivo, se facilita la integración de información calculada automáticamente por AutoCAD con la interfaz gráfica de la aplicación, y este método se usará en la mayoría de las clases que se crearon para poder hacer una comunicación los datos obtenidos de AutoCAD.

Como se trata de una clase abstracta, define tres métodos (agregarBotonesEspecificos, getClasificacion y getTotalPintura) que deben ser implementados en cada clase hija según el tipo de raya que se esté trabajando. Esto proporciona flexibilidad y al mismo tiempo mantiene la estructura común en toda la aplicación.

```
//definimos funciones abstractas se se tienen que definir en las clases Hijas
	protected abstract void agregarBotonesEspecificos();

	protected abstract String getClasificacion(String tipoDeRaya);

	protected abstract double getTotalPintura(String clasificacion, double
distancia, int velocidad, String tipoCamino);
```

FrmMarcasSH

Para las clases de M1 a M20, se distribuyeron en 4 archivos que son ***FrmMarcasSH1***, ***FrmMarcasSH2***, ***FrmMarcasSH3*** y ***FrmMarcasSH4***, esto para facilitar su organización de esta manera tenemos que en ***FrmMarcasSH1*** tenemos de M1 a M5, en ***FrmMarcasSH2*** tenemos de M6 a M10, en ***FrmMarcasSH3*** tenemos de M11 a M15 y en ***FrmMarcasSH4*** tenemos de M16 a M20. Dentro de estos archivos se crearon las clases de ***FrmM1*** a ***FrmM20*** donde cada uno sirve para hacer los cálculos correspondientes a cada marca de señalización vial.

En este proceso se siguió el modelo en espiral, avanzando de forma iterativa a través de la **definición de objetivos**, donde se determinó que debía hacer cada clase y cómo debía interactuar con el resto del sistema; posteriormente se realizó el **análisis de riesgos**, identificando posibles errores en la comunicación entre formularios, manejo de datos y cálculos; después se avanzó al **desarrollo**, implementando las clases y comportamientos específicos de cada marca; y finalmente se llevó a

cabo la **planificación** del siguiente ciclo, organizando qué módulos continuarían con su implementación o ajuste para mantener una construcción progresiva y controlada del sistema.

La clase ***FrmM1*** es una implementación hija de la clase base ***FrmBase***, lo que significa que hereda toda la estructura común que ya se definió previamente, pero al mismo tiempo la especializa para trabajar exclusivamente con el señalamiento horizontal M-1, que corresponde a las rayas separadoras de sentidos de circulación. En esta ventana, además de mostrar la tabla que gestiona los datos del proyecto, se definen los botones y comportamientos específicos para los subtipos M-1.1 a M-1.4, así como el cálculo de pintura que requiere cada uno de ellos, y su estructura es la siguiente:

```
package Vistas;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.DefaultCellEditor;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import Clases.Proyecto;

//Definimos todas las clases Hijas de FrmBase (M1 a M5)
class FrmM1 extends FrmBase {
	private static final long serialVersionUID = 1L;
	private static String mar = "M1";
	private JButton btnAgregarM11;
	private JButton btnAgregarM12;
	private JButton btnAgregarM13;
	private JButton btnAgregarM14;
```

```
	// Tipos de raya disponibles para selección en la tabla
	private String[] tiposDeRaya = {
			"Raya continua sencilla",
			"Raya discontinua sencilla",
			"Raya continua-discontinua",
			"Raya continua doble"
	};

	public FrmM1(Proyecto proy) {
			super(proy, mar);
			inicializacion();
	}

	// Configura los elementos iniciales de la ventana
	private void inicializacion() {
			JComboBox<String> comboBox = new JComboBox<>(tiposDeRaya);
			DefaultCellEditor comboBoxEditor = new DefaultCellEditor(comboBox);
			datosRaya.getColumnModel().getColumn(1).setCellEditor(comboBoxEditor);

			JLabel lblNewLabel = new JLabel("Raya Separadora de Sentidos de
Circulación (M-1)");
			lblNewLabel.setFont(new Font("Century Schoolbook", Font.BOLD, 13));
			lblNewLabel.setBounds(10, 10, 579, 31);
			lblNewLabel.setForeground(Color.WHITE);
			panelFondo.add(lblNewLabel);

			agregarBotonesEspecificos();
	}

	// Calcula el total de pintura para M-1.4 considerando la separación entre
rayas
	protected int calcularTotal(double distancia, int sepRayas) {
			double resultado;
			if (sepRayas < 50) {
```

```
                resultado = distancia * 2;
            } else {
                resultado = (distancia * 2) + ((distancia / (2 * sepRayas)) *
sepRayas);
            }
            return (int) Math.round(resultado);
      }

      // Coloca y configura los botones de selección para cada subtipo M-1.x
      @Override
      protected void agregarBotonesEspecificos() {
            btnAgregarM11 = new JButton("Agregar M-1.1");
            btnAgregarM12 = new JButton("Agregar M-1.2");
            btnAgregarM13 = new JButton("Agregar M-1.3");
            btnAgregarM14 = new JButton("Agregar M-1.4");

            btnAgregarM11.setBounds(575, 52, 150, 30);
            btnAgregarM12.setBounds(575, 93, 150, 30);
            btnAgregarM13.setBounds(575, 134, 150, 30);
            btnAgregarM14.setBounds(575, 175, 150, 30);

            estilizarBoton(btnAgregarM11, new Color(0, 123, 255));
            estilizarBoton(btnAgregarM12, new Color(0, 123, 255));
            estilizarBoton(btnAgregarM13, new Color(0, 123, 255));
            estilizarBoton(btnAgregarM14, new Color(0, 123, 255));

            // Botón para M-1.1
            btnAgregarM11.addActionListener(e -> {
                  String mensaje = "<html>Instrucciones:<br/>"
                              + "1. Vaya al proyecto de AutoCAD con el LISP
cargado (sino vaya a la ventana anterior para cargar el LISP).<br/>"
                              + "2. Escriba el comando 'GuardarLongitudes' y
seleccione todas las líneas que sean M-1.1 (Raya continua sencilla).<br/>"
                              + "3. Una vez realizado lo anterior presione OK en
esta ventana, si no solo de en cancelar.</html>";
```

```
                agregarRaya(tiposDeRaya[0], getClasificacion(tiposDeRaya[0]),
mensaje, proyecto, 1);
            });

            // Botón para M-1.2
            btnAgregarM12.addActionListener(e -> {
                String mensaje = "<html>Instrucciones:<br/>"
                            + "1. Vaya al proyecto de AutoCAD con el LISP
cargado (sino vaya a la ventana anterior para cargar el LISP).<br/>"
                            + "2. Escriba el comando 'GuardarLongitudes' y
seleccione todas las líneas que sean M-1.2 (Raya discontinua sencilla).<br/>"
                            + "3. Una vez realizado lo anterior presione OK en
esta ventana, si no solo de en cancelar.</html>";

                agregarRaya(tiposDeRaya[1], getClasificacion(tiposDeRaya[1]),
mensaje, proyecto, 1);
            });

            // Botón para M-1.3
            btnAgregarM13.addActionListener(e -> {
                String instrucciones = "<html>Instrucciones:<br/>"
                            + "1. Vaya al proyecto de AutoCAD con el LISP
cargado (sino vaya a la ventana anterior para cargar el LISP).<br/>"
                            + "2. Escriba el comando 'GuardarLongitudes' y
seleccione todas las líneas que sean M-1.3 (Raya continua-discontinua).<br/>"
                            + "3. Una vez realizado lo anterior presione OK en
esta ventana, si no solo de en cancelar.<html>";
                agregarRaya(tiposDeRaya[2], getClasificacion(tiposDeRaya[2]),
instrucciones, proyecto, 1);
            });

            // Botón para M-1.4 con cálculo extra por separación
            btnAgregarM14.addActionListener(e -> {
```

```
                    String instrucciones = "<html>" +
"<p><strong>Instrucciones:</strong></p>" + "<ol>"
                                + "<li>Vaya al proyecto de AutoCAD con el LISP
cargado (o regrese a la ventana anterior para cargar el LISP).</li>"
                                + "<li>Escriba el comando 'GuardarLongitudes' y
seleccione solo una línea de cada par que forme las M-1.4 (Raya continua
doble).</li>"
                                + "<li>Introduzca la separación entre rayas (en cm)
y presione OK para continuar, o Cancelar para salir.</li>"
                                + "</ol>" + "</html>";

                    JLabel txtSepara = new JLabel("Separación entre rayas (cm):");
                    JTextField textField = new JTextField(10);

                    Component[] componentesPanel = new Component[] { txtSepara,
textField };

                    // Procesa el cálculo cuando el usuario confirma los datos
                    manejarBtnAgregar(instrucciones, componentesPanel, () -> {
                          String valorIngresado = textField.getText().trim();
                          if (valorIngresado.isEmpty()) {
                                JOptionPane.showMessageDialog(null, "La separación
no puede estar vacía.", "Error",
                                            JOptionPane.ERROR_MESSAGE);
                                return;
                          }

                          try {
                                int sepRayas = Integer.parseInt(valorIngresado);
                                double distancia = 0.0;

                                // Suma de distancias desde el archivo generado por
AutoCAD
                                String[] lineas = leerArchivo(rutaArchivo);
                                for (String linea : lineas) {
```

```
                                    distancia += Double.parseDouble(linea);
                                }

                                // Calcula el total y lo agrega a la tabla
                                int total = calcularTotal(distancia, sepRayas);

                                tableModel.addRow(new Object[] {
String.valueOf((int) Math.round(distancia)), tiposDeRaya[3],
                                        getClasificacion(tiposDeRaya[3]),
String.valueOf(total), 1 });
                                int nuevaFila = tableModel.getRowCount() - 1;
                                pinturaOriginal.put(nuevaFila, total);

                            } catch (NumberFormatException ex) {
                                JOptionPane.showMessageDialog(null, "Error en el
formato de la separación.", "Error",
                                        JOptionPane.ERROR_MESSAGE);
                            }
                        });
                });

                panelFondo.add(btnAgregarM11);
                panelFondo.add(btnAgregarM12);
                panelFondo.add(btnAgregarM13);
                panelFondo.add(btnAgregarM14);
        }

        // Devuelve la clasificación según el tipo de raya
        @Override
        protected String getClasificacion(String tipoDeRaya) {
                switch (tipoDeRaya) {
                case "Raya continua sencilla":
                        return "M-1.1";
                case "Raya discontinua sencilla":
                        return "M-1.2";
```

```
            case "Raya continua-discontinua":
                  return "M-1.3";
            case "Raya continua doble":
                  return "M-1.4";
            default:
                  return "";
            }
      }

      // Calcula la cantidad de pintura necesaria según clasificación, distancia,
velocidad y tipo de camino
      protected double getTotalPintura(String clasificacion, double distancia, int
velocidad, String tipoCamino) {
            switch (clasificacion) {
            case "M-1.1":
                  return distancia;
            case "M-1.2":
                  if (tipoCamino.equals("Carretera")) {
                        return (distancia / 15) * 5;
                  } else if (tipoCamino == "Calle") {
                        if (velocidad > 50) {
                              return (distancia / 15) * 5;
                        } else {
                              return (distancia / 7.5) * 2.5;
                        }
                  } else if (tipoCamino.equals("Vía ciclista")) {
                        return distancia / 3;
                  }
            case "M-1.3":
                  if (tipoCamino.equals("Carretera")) {
                        return ((distancia / 15) * 5) + distancia;
                  } else if (tipoCamino.equals("Calle")) {
                        if (velocidad > 50) {
                              return ((distancia / 15) * 5) + distancia;
                        } else {
```

```
                        return ((distancia / 7.5) * 2.5) + distancia;
                    }
                } else if (tipoCamino.equals("Vía ciclista")) {
                    return (distancia / 3) + distancia;
                }
            default:
                return 0;
            }
        }
}
```

Durante la inicialización se agrega un título en la parte superior de la ventana que indica el señalamiento con el que se está trabajando, en este caso “Raya Separadora de Sentidos de Circulación (M-1)”, manteniendo la claridad en la interfaz. Este módulo también se desarrolló siguiendo el modelo en espiral, avanzando primero con la definición de objetivos, donde se estableció la información que debía mostrar la ventana y las interacciones que debía permitir; luego se realizó un análisis de riesgos, evaluando posibles errores en la captura de longitudes, validación de datos y comunicación con el LISP, posteriormente se procedió al desarrollo de cada componente gráfico y funcionalidad interna. Y finalmente se continuó con la planificación del siguiente ciclo, ajustando los procedimientos y preparando las mejoras o extensiones necesarias para los subtipos restantes.

La función agregarBotonesEspecificos se encarga de colocar y configurar los botones “Agregar M-1.1”, “Agregar M-1.2”, “Agregar M-1.3” y “Agregar M-1.4”. Cada uno abre un cuadro de instrucciones que guía al usuario en AutoCAD para capturar las longitudes de las líneas que correspondan al subtipo seleccionado, mediante el comando GuardarLongitudes del LISP cargado previamente. Una vez que el usuario confirma, la función agregarRaya procesa esos datos y los agrega a la tabla junto con su clasificación. Cabe resaltar el caso particular de M-1.4, que requiere un paso adicional: antes de agregar la información, el usuario debe indicar la separación entre las rayas dobles. Con ese valor se realiza un cálculo más complejo en el método calcularTotal, donde se ajusta la cantidad de pintura en función de la distancia y la separación ingresada, reflejando la diferencia constructiva de este subtipo respecto a los demás.

El método getClasificacion se utiliza para convertir el nombre descriptivo de la raya en su clave oficial, de modo que “Raya continua sencilla” corresponde a “M-1.1”, “Raya discontinua sencilla” a “M-1.2”, y así sucesivamente. Esta clasificación se guarda en la tabla y sirve como referencia para la base de datos.

Por último, la función getTotalPintura define el cálculo de la pintura necesaria en cada subtipo según la clasificación, la distancia de las líneas medidas, la velocidad de la vía y el tipo de camino seleccionado. Aquí se observa que para M-1.1 la pintura equivale directamente a la distancia, mientras que en el caso de M-1.2 y M-1.3 se aplican fórmulas diferentes dependiendo de si el camino es carretera, calle o vía ciclista, así como de la velocidad permitida. De esta forma se refleja la normativa y las variaciones de diseño que debe seguir la señalización en función del contexto vial.

Las clases ***FrmM2***, ***FrmM3*** y ***FrmM4*** mantienen la misma estructura básica que ***FrmM1***, ya que todas heredan de ***FrmBase*** y trabajan con el mismo esquema de tabla, botones y cálculo de cantidades de pintura. En cada caso, la diferencia principal se encuentra en los datos que se cargan, los botones disponibles y las fórmulas específicas que se aplican en el cálculo final. La implementación de estas clases también siguió el modelo en espiral: primero con la definición de objetivos de cada subtipo, después con el análisis de riesgos de su captura en AutoCAD, seguido del desarrollo particular de los botones y métodos de cálculo, y finalmente la planificación del siguiente módulo.

En el caso de ***FrmM2***, la ventana se centra en el manejo de la señal M-2, que corresponde a la raya separadora de carriles. Su funcionamiento es prácticamente igual al de ***FrmM1***, pero con una diferencia importante: cuenta con dos botones, uno que permite registrar el dato de la raya separadora normal y otro que corresponde a la raya auxiliar. El resto del procedimiento es idéntico, desde la validación de datos hasta la inserción en la tabla y el cálculo de pintura, donde la fórmula utilizada es la que corresponde a esta señal en particular.

Por su parte, ***FrmM3*** gestiona la señal M-3, que corresponde a las rayas de aproximación a islas canalizadoras. La estructura de la clase sigue el mismo esquema ya visto, pero en este caso se dispone de tres botones diferentes, cada uno asociado a un tipo de raya: M-3.1, M-3.2 y M-3.3. Estos permiten capturar longitudes de acuerdo con la variante seleccionada, y la función de cálculo ajusta la fórmula dependiendo del subtipo registrado. De este modo, aunque se mantiene el procedimiento

general de validación, inserción y cálculo, la clase contempla más de una opción para reflejar las particularidades de esta señal.

La clase ***FrmM4*** trabaja con la señal M-4, que representa rayas de trayectorias en intersecciones. Aquí también se agregan tres variantes diferentes: M-4.1, M-4.2 y M-4.3. El manejo de botones es similar al de FrmM3, aunque con una validación adicional: el botón que corresponde a la M-4.2 incluye una restricción que impide registrar datos si el tipo de camino no es una "Calle". En cuanto al cálculo de pintura, este también se diversifica según el subtipo: para la M-4.1 se multiplica la longitud cada 6 metros por dos, en la M-4.2 se duplica directamente y en la M-4.3 se toma solo la tercera parte de la longitud.

La clase ***FrmM5*** conserva la organización general de tabla, botones y validaciones, pero introduce un cambio importante: se manejan dos variantes que corresponden a la señal M-5, las rayas canalizadoras. Estas son las que limitan la zona neutral (M-5.1) y las que se dibujan dentro de esa zona neutral (M-5.2).

La interfaz muestra un título con el nombre de la señal y después se invoca el método agregarBotonesEspecificos, donde se define un único botón para procesar las M-5. Al presionarlo, se muestran instrucciones que indican cómo se deben seleccionar las líneas en AutoCAD utilizando los comandos GuardarLongitudes y GuardarAngulos. Esta secuencia es necesaria porque la señal involucra tanto medidas lineales como ángulos, y el programa debe leer ambos datos desde los archivos de texto que genera el LISP.

El evento del botón está diseñado para verificar que efectivamente se hayan seleccionado dos líneas, ya que estas definen los límites de la zona neutral, de igual manera se le solicita al usuario que elija el color de la zona neutral, ya sea Blanco o amarillo. A partir de ahí se calculan dos registros diferentes en la tabla. El primero corresponde a las M-5.1, donde se suman las longitudes de ambas líneas para obtener la distancia total y se registra junto con su clasificación. El segundo corresponde a las M-5.2, donde el cálculo cambia: se obtiene el área triangular delimitada por esas dos líneas y el ángulo entre ellas, y con esa información se calcula además la longitud de las rayas inclinadas que se dibujan dentro de la zona neutral. Para ello, la clase incluye el método getTotal, que hace uso de trigonometría (ley de cosenos, bisectriz y ángulo transversal de 45°) para estimar cuántas líneas caben en la zona y cuál sería su longitud total.

También contiene la función getClasificacion devuelve la clave adecuada dependiendo del tipo de raya seleccionado y getTotalPintura en este caso se limita a devolver la distancia en el caso de la M-5.1, ya que la M-5.2 tiene su propio cálculo independiente. En conjunto, esta clase amplía la lógica de las anteriores, pues no solo registra longitudes lineales, sino que incorpora un cálculo geométrico más complejo para representar correctamente las zonas canalizadoras.
Durante el desarrollo de esta clase también se aplicó el modelo en espiral, iniciando con la definición de objetivos para determinar cómo debían capturarse las líneas y los ángulos, seguido del análisis de riesgos relacionado con errores en la selección de elementos y lecturas incorrectas del LISP; luego se procedió al desarrollo del cálculo geométrico y de la interfaz correspondiente; y finalmente se realizó la planificación del siguiente ciclo, donde se ajustaron los métodos en función de las pruebas realizadas con zonas canalizadoras reales.

La clase ***FrmM6*** hace refiere a la M-6 (Rayas de alto), la estructura se mantiene muy parecida a las anteriores (M1 – M4), pero aquí solo existe un tipo de raya: "Rayas de alto".
La interfaz incluye un botón principal Agregar M-6, que al presionarlo despliega las instrucciones para ir a AutoCAD, ejecutar el comando GuardarLongitudes y seleccionar todas las líneas que correspondan a esta señal. Posteriormente, se registran los datos en la tabla.
En el método getClasificacion, cualquier selección se clasifica directamente como M-6, y en el cálculo (getTotalPintura), la cantidad de pintura corresponde simplemente a la distancia total medida.
El desarrollo de FrmM6 siguió igualmente el modelo en espiral, comenzando con la definición de objetivos para determinar el único subtipo M-6, continuando con el análisis de riesgos respecto a la captura múltiple de líneas en AutoCAD, posteriormente realizando el desarrollo del botón, validaciones y cálculo, y finalizando con la planificación del ciclo siguiente para ajustar la interacción con el LISP y mejorar la precisión de la medición.

La clase ***FrmM7***, M-7 (Rayas para cruce de peatones), se mantiene la misma lógica de interacción con la tabla y los botones, pero aquí aparece una particularidad importante: en lugar de depender de

datos obtenidos desde AutoCAD, el usuario debe introducir manualmente la longitud de la raya y el ancho del camino mediante un cuadro de diálogo personalizado con campos de texto.

Cuando el usuario presiona el botón Agregar M-7, se muestran las instrucciones y los dos campos a llenar. Tras validar que los valores no estén vacíos y que sean numéricos, el sistema inserta una nueva fila en la tabla y asigna: la distancia calculada como el producto de la longitud y el ancho, la clasificación fija como M-7, y el total de pintura estimado mediante la función calcularTotal. Esta función aplica una fórmula específica que considera la relación entre el ancho del camino, la separación de franjas (0.8 m) y el ancho de cada raya (0.4 m), resultando en un valor aproximado del material necesario.

Esta clase también fue construida mediante el modelo en espiral, iniciando con la definición de objetivos para establecer los datos que el usuario debía introducir manualmente, continuando con el análisis de riesgos por posibles errores de captura o inconsistencias numéricas, luego avanzando con el desarrollo del cuadro de diálogo y las validaciones correspondientes, y finalizando con la planificación del siguiente ciclo para mejorar el cálculo y ajustar la fórmula según las condiciones de distintos anchos de camino.

En la clase ***FrmM8*** (Marcas para cruce de ferrocarril), el flujo de trabajo sigue la misma lógica que las clases anteriores como M1, M2, M3, etc. El botón Agregar M-8 presenta al usuario un mensaje con instrucciones precisas: asegurarse de que el LISP esté cargado en AutoCAD, ejecutar el comando GuardarLongitudes y seleccionar únicamente las líneas correspondientes a las marcas M-8.

Una vez que el usuario confirma, se llama al método agregarRaya, que registra en la tabla los datos de la selección, asigna la clasificación M-8 y calcula la cantidad de pintura usando getTotalPintura, que en este caso simplemente devuelve la distancia total seleccionada sin realizar transformaciones adicionales.

En esta clase también se aplicó el modelo en espiral, iniciando con la definición de objetivos para establecer el método de captura desde AutoCAD, continuando con el análisis de riesgos respecto a errores en la selección de líneas o incompatibilidades con el LISP, luego realizando el desarrollo del botón, el método agregarRaya y el cálculo correspondiente, y finalizando con la planificación del siguiente ciclo para ajustar la interacción con AutoCAD y validar los resultados con pruebas reales.

En la clase ***FrmM9***, se define la señalización de tipo M-9, que corresponde a rayas con espaciamiento logarítmico. La clase inicia añadiendo una etiqueta de título en la interfaz que indica que se trata de rayas M-9. Se crea un botón Agregar M-9, que al presionarlo indica al usuario que debe ingresar el número de rayas logarítmicas que desea agregar y el ancho del camino en metros.
Después se validan los campos para asegurarse de que no estén vacíos. Luego, los valores ingresados se convierten a números y se calcula el total de pintura necesario mediante la fórmula:

$$totalPintura = totalLineas * anchoCamino$$

Se agrega una nueva fila a la tabla con los siguientes datos: cantidad (número de líneas), tipo de raya, clasificación M-9, total de pintura calculado y cantidad rayas logarítmicas que hay en el camino con el mismo número de rayas. Además, el total de pintura se guarda en el mapa pinturaOriginal para mantener un registro del valor original.
Los métodos getClasificacion y getTotalPintura retornan respectivamente la clasificación M-9 según el tipo de raya y el total de pintura calculado a partir de la distancia ingresada.
El desarrollo de esta clase siguió igualmente el modelo en espiral, comenzando con la definición de objetivos para determinar qué datos debían ser introducidos manualmente, seguido del análisis de riesgos como valores inconsistentes o fórmulas incorrectas, avanzando con el desarrollo del cálculo logarítmico y del registro en la tabla, y concluyendo con la planificación del ciclo siguiente para refinar las validaciones y ajustar la fórmula según los anchos de camino más comunes.

La clase **FrmM10** corresponde a las marcas para estacionamiento (M-10) y abarca tres subtipos: M-10.1, M-10.2 y M-10.3, que representan respectivamente las marcas para estacionamiento de vehículos motorizados, para zonas de pago y para servicios especiales. En la inicialización se añade una etiqueta de título indicando que se trata de marcas de estacionamiento (M-10). Se definen tres botones. El botón M-10.1 muestra instrucciones para que el usuario ejecute el comando AutoCAD GuardarLongitudes y seleccione las líneas continuas y punteadas correspondientes a M-10.1, solicitando cuántas líneas continuas seleccionó mediante un JTextField. Luego se leen las distancias guardadas en un archivo, separando las continuas y punteadas, y se calcula el total de pintura sumando la distancia de líneas continuas y punteadas mediante getTotalPintura y calcularTotal.

Finalmente se agrega una fila en la tabla con la distancia total, el tipo de raya, la clasificación, el total de pintura y la cantidad de tramos, y se guarda en pinturaOriginal.

El botón M-10.2 funciona de manera similar, pero para la subclase M-10.2, leyendo las distancias de líneas continuas y punteadas, calculando los totales y actualizando la tabla y pinturaOriginal.

El botón M-10.3, además de indicar líneas continuas, solicita el largo del cajón y el ancho de la franja de circulación, que se usan para calcular pintura adicional según la fórmula:

$$pasoPeat = ((largoCajon / 0.8) * 0.4) * anchoFranjaCirculacion$$

La fila de la tabla se llena con el total de pintura incluyendo esta cantidad adicional.

Durante la creación de esta clase también se siguió el modelo en espiral, empezando con la definición de objetivos para organizar los tres subtipos de estacionamiento y sus requerimientos, continuando con el análisis de riesgos asociado a la mezcla de líneas continuas, punteadas y cálculos adicionales, luego desarrollando (desarrollo) los botones, lecturas de archivos y fórmulas específicas, y finalmente realizando la planificación del siguiente ciclo para ajustar los cálculos y mejorar la interacción en AutoCAD y en la interfaz.

La clase **FrmM11** corresponde a las rayas, símbolos y leyendas para regular el uso de carriles (M-11), y contempla tres subtipos: M-11.1 para flechas y leyendas en carriles, M-11.2 para flechas y leyendas que indican un carril exclusivo, y M-11.3 para establecer lugares de parada. Durante la inicialización se añade una etiqueta de título que identifica el propósito de la ventana. Se define un botón principal btnAgregarM111 para agregar flechas y símbolos, al que se le aplica un estilo visual específico.

La clase **FrmM11** también fue construida siguiendo el modelo en espiral, iniciando con la definición de objetivos para determinar cómo gestionar los diferentes subtipos y sus símbolos, siguiendo con el análisis de riesgos relacionado con la lectura de distancias y la correcta representación de flechas y leyendas, continuando con el desarrollo de los botones, cálculos y estructura interna de los registros, y finalizando con la planificación del siguiente ciclo para adaptar la lógica según las pruebas en planos reales y la validación con el ingeniero responsable.

Al presionar el botón, se despliega un panel con instrucciones para seleccionar el tipo de flecha a agregar y un JComboBox con las opciones: “Flecha Recta”, “Flecha Curva” y “Flecha Recta y Curva”. Una vez que el usuario hace su selección y confirma, se añade una nueva fila en la tabla

donde se registra el tipo de raya, la subclasificación correspondiente (obtenida mediante getClasificacion), la cantidad calculada de pintura según getTotalPintura y un valor fijo de 1 en la columna de cantidad de flechas. El método getTotalPintura determina la cantidad de pintura necesaria en función del tipo de flecha y la velocidad del proyecto; por ejemplo, para una flecha recta la pintura es 0.76875 si la velocidad es menor o igual a 50 km/h, y 2.4 si es mayor. Para flechas curvas o combinadas, se aplican valores diferentes siguiendo el mismo criterio. El método calcularTotal aplica un factor de 0.5 según la subclasificación (M-11.1, M-11.2 o M-11.3) para obtener la pintura correspondiente a otras operaciones de la clase. En conjunto, FrmM11 permite agregar de manera precisa y controlada símbolos y flechas en carriles, calculando automáticamente la pintura necesaria y registrando los datos en la tabla de forma organizada.

En ***FrmM12*** Marcas en guarniciones (M-12), los cuatro botones (btnAgregarM121 a btnAgregarM124) están diseñados para manejar la entrada de datos de AutoCAD y calcular la pintura necesaria según el tipo de raya M-12. Cada botón primero muestra instrucciones al usuario sobre cómo usar el comando GuardarLongitudes del LISP y qué líneas deben seleccionarse, ya sean continuas, punteadas o ambas. En el caso de M-12.2 y M-12.4, además se solicita al usuario que indique cuántas líneas continuas se seleccionaron mediante un campo de texto.

Luego, se leen las distancias de las líneas seleccionadas usando leerArchivo(rutaArchivo) y se suman para obtener la distancia total. La pintura se calcula de forma distinta según el botón: para M-12.1 y M-12.3, se usa directamente la distancia total multiplicada por los factores de getTotalPintura y calcularTotal; para M-12.2 y M-12.4, la distancia de las líneas continuas se suma normalmente y la de las punteadas se multiplica por 0.5 antes de sumarse al total.

Después de calcular la pintura, se agrega una nueva fila a la tabla con la información de distancia, tipo de raya, clasificación, pintura total y número de tramos, y el valor de pintura calculado se guarda en pinturaOriginal.

En esta clase también se aplicó el modelo en espiral, comenzando con la definición de objetivos para determinar cómo capturar y procesar las diferentes combinaciones de líneas continuas y punteadas, seguido del análisis de riesgos relacionado con la correcta separación de distancias y la validación del número de líneas, avanzando con el desarrollo de los botones, cálculos y registros en

la tabla, y finalizando con la planificación del siguiente ciclo para ajustar las fórmulas y la interacción con los datos obtenidos desde AutoCAD.

En ***FrmM13*** Marcas en estructuras y objetos adyacentes a la superficie de rodadura (M-13), el botón btnAgregarM131 se encarga de capturar las dimensiones necesarias para calcular la pintura de marcas en estructuras y objetos adyacentes a la superficie de rodadura. Al presionarlo, se muestran instrucciones al usuario y un panel con campos de texto para ingresar el ancho de columna, la altura libre de la estructura (gálibo), el alto de trabe y el ancho de camino.
Dependiendo del valor del alto de la estructura, se valida que todos los campos obligatorios estén completos; si la estructura supera 4.5 metros, el cálculo del alto de trabe se ajusta automáticamente a 3 metros y se omite la altura de la estructura para la validación.
Luego, los valores se convierten a números, se verifica que el ancho de columna y el alto de trabe no excedan 1 metro, se calcula la pintura necesaria con la función calcularTotal, y se agrega una nueva fila a la tabla con la información de distancia, tipo de raya, clasificación, pintura total y número de tramos, guardando también el valor de pintura en pinturaOriginal. Este botón centraliza todo el proceso de captura de datos, validación, cálculo y registro en la tabla para el tipo de raya M-13.1.
El desarrollo de esta clase siguió igualmente el modelo en espiral, empezando con la definición de objetivos para establecer los parámetros que el usuario debía ingresar manualmente, continuando con el análisis de riesgos por posibles errores en los valores de altura, gálibo o restricciones normativas, avanzando con el desarrollo del panel de captura, validaciones y fórmula de cálculo, y concluyendo con la planificación de mejoras posteriores basadas en pruebas con distintos tipos de estructuras.

En la clase ***FrmM14*** Raya de emergencia para frenado (M-14), se manejan cuatro botones que agregan los distintos tipos de raya de emergencia para frenado y marcas asociadas.
El primer botón, M-14.1, permite al usuario seleccionar en AutoCAD las líneas discontinuas correspondientes a esta raya; al presionar OK, se leen las longitudes desde un archivo, se calcula la pintura necesaria usando la función getTotalPintura y se agrega una fila a la tabla con la información de distancia, tipo de raya, clasificación, total de pintura y cantidad.

El botón M-14.2 funciona de manera similar, pero para las líneas continuas de la raya de emergencia para frenado continua, siguiendo el mismo flujo de lectura, cálculo y actualización de la tabla.
Los botones M-14.3 R y M-14.3 B agregan áreas coloreadas (rojo o blanco) correspondientes a la marca de acceso a la rampa de emergencia; en este caso se utiliza el comando GuardarAreas en AutoCAD y se leen las superficies desde un archivo, calculando el total de pintura y agregando una fila a la tabla, diferenciando la clasificación con el sufijo "R" o "B". La función getClasificacion traduce el tipo de raya a su código correspondiente y getTotalPintura define la forma en que se calcula la pintura según el tipo de raya, con reglas específicas como dividir la distancia entre 15 y multiplicar por 5 para la M-14.1 y usar la distancia directa para las demás.
En esta clase se aplicó nuevamente el modelo en espiral, comenzando con la definición de objetivos para gestionar tanto líneas como áreas de color, continuando con el análisis de riesgos al manejar distintos comandos de AutoCAD y la separación entre colores y subtipos, desarrollando (desarrollo) las funciones de lectura, cálculo y clasificación, y finalizando con la planificación del ciclo siguiente para ajustar los cálculos de áreas y mejorar la precisión en la interacción con AutoCAD.

En el ***FrmM15*** Marcas para vías ciclistas (M-15), se implementa la gestión de rayas para cruce de ciclistas, específicamente la M-15.5. Se crea un botón que permite al usuario agregar esta raya a la tabla de pintura vial. Al presionarlo, se muestra un panel con instrucciones detalladas donde se solicita la extensión del cruce y el ancho del camino. Los valores ingresados se validan para asegurarse de que no estén vacíos y se convierten a tipo numérico; si hay errores en el formato, se muestra un mensaje de advertencia.
Una vez validados los datos, se agrega una nueva fila a la tabla con la información de distancia, tipo de raya, clasificación, total de pintura y cantidad. El cálculo del total de pintura se realiza mediante el método calcularTotal, que aplica una fórmula basada en la extensión de la raya y el ancho del camino, convirtiendo el resultado a entero. La función getClasificacion traduce el tipo de raya a su código correspondiente "M-15.5", y getTotalPintura devuelve el valor de pintura según la distancia indicada.
También aquí se siguió el modelo en espiral, empezando con la definición de objetivos para determinar qué datos serían capturados manualmente, luego el análisis de riesgos sobre valores incorrectos o formatos no numéricos, continuando con el desarrollo del panel, validaciones y

fórmula de cálculo, y cerrando con la planificación de ajustes futuros basados en pruebas con cruces ciclistas de diferentes dimensiones.

En ***FrmM16*** se maneja la funcionalidad de marcas temporales. Se crea un botón que permite al usuario agregar estas marcas a la tabla de pintura vial. Al presionarlo, se muestra un diálogo con instrucciones detalladas sobre cómo usar AutoCAD con el LISP cargado, indicando que debe ejecutar el comando GuardarLongitudes y seleccionar únicamente las líneas correspondientes a M-16, además de elegir el color de la marca. Se proporciona un JComboBox para seleccionar el color (Azul o Naranja).
El sistema lee desde un archivo generado por AutoCAD/LISP todas las longitudes seleccionadas, las suma y redondea el total para luego calcular la pintura requerida mediante el método getTotalPintura. Se agrega una nueva fila a la tabla con la distancia total, el tipo de raya, la clasificación basada en el color elegido, el total de pintura y la cantidad. El valor de pintura original se guarda en un mapa pinturaOriginal. La función getClasificacion traduce el tipo de raya al código "M-16", mientras que getTotalPintura devuelve la distancia como el total de pintura requerido.
Esta clase también empleó el modelo en espiral, empezando con la definición de objetivos para gestionar marcas temporales con variaciones de color, siguiendo con el análisis de riesgos por posibles errores en la lectura de longitudes o selección de color, avanzando con el desarrollo del botón, diálogo y fórmula, y finalizando con la planificación de mejoras para optimizar el proceso de captura y el manejo de colores en futuras versiones.

En ***FrmM17*** se gestiona la marca de área de espera para vehículos no motorizados y motocicletas (M-17). Se define un botón que permite al usuario agregar estas marcas a la tabla de pintura vial. Al presionarlo, se muestra un mensaje con instrucciones detalladas para usar AutoCAD con el LISP cargado, indicando que se debe ejecutar el comando GuardarLongitudes y seleccionar todas las líneas correspondientes a M-17.
El método agregarRaya se encarga de procesar los datos, agregando la información a la tabla: se registra el tipo de raya, su clasificación “M-17”, la distancia total obtenida desde AutoCAD y la cantidad. La función getClasificacion convierte el tipo de raya en su código correspondiente, y

getTotalPintura retorna la distancia como total de pintura requerida, manteniendo la lógica consistente con las demás clases hijas de FrmBase.
Durante el desarrollo de esta clase se aplicó nuevamente el modelo en espiral, iniciando con la definición de objetivos para establecer cómo capturar las áreas de espera, continuando con el análisis de riesgos ante posibles errores en la selección de líneas en AutoCAD, desarrollando (desarrollo) el botón, el método agregarRaya y la lógica de clasificación, y finalizando con la planificación de nuevos ciclos para validar el funcionamiento mediante pruebas en planos reales.

La clase ***FrmM18*** se encarga de gestionar la marca de ceda el paso (M-18), y sigue la misma estructura que las demás clases hijas de FrmBase. En primer lugar, se definen las variables necesarias, destacando el botón btnAgregarM18 que permitirá al usuario agregar esta marca, y el arreglo tiposDeRaya que contiene únicamente el tipo de raya correspondiente. Durante la inicialización se agrega un JLabel al panel de fondo con el título "Marca de ceda el paso (M-18)" con un formato destacado para mayor claridad visual.
Dentro de agregarBotonesEspecificos, se crea y configura btnAgregarM18, aplicándole estilo visual y ubicándolo en la interfaz. El usuario debe ejecutar el comando GuardarAreas y seleccionar todos los triángulos que representen la marca M-18. Cuando se confirma la operación mediante el método manejarBtnAgregar, se leen los valores desde el archivo generado por LISP, se suman las áreas seleccionadas para obtener la distancia total y, a partir de ella, se calcula la cantidad de pintura necesaria.
La construcción de esta clase también siguió el modelo en espiral, comenzando con la definición de objetivos para el manejo de triángulos de ceda el paso, continuando con el análisis de riesgos asociado a la lectura de áreas y su conversión a pintura, avanzando con el desarrollo del botón, lectura de archivos y fórmula, y concluyendo con la planificación de ciclos posteriores para optimizar el cálculo y la interacción con AutoCAD.

La clase **FrmM19** se encarga de manejar las marcas de prohibición M-19, que incluyen tres subtipos: M-19.1 para "Prohibido estacionar", M-19.2 para "Prohibido parar" y M-19.3 para "Prohibido parar en intersección". Para ello, se definen tres botones específicos, uno por cada subtipo de raya, y un arreglo tiposDeRaya que contiene los nombres correspondientes. Durante la

inicialización se agrega un JLabel al panel de fondo con el título “Marcas para indicar prohibiciones (M-19)” con un formato visual destacado, asegurando que el usuario identifique claramente la sección correspondiente. Después se llama al método agregarBotonesEspecificos() para configurar los botones.

El usuario debe ejecutar el comando GuardarLongitudes y seleccionar las líneas correspondientes a cada tipo de marca. Posteriormente, se llama al método agregarRaya con la información pertinente, de modo que los datos se incorporen a la tabla. La clase también incluye el método calcularTotal, que calcula la cantidad de pintura requerida en función de la distancia y la separación de las rayas; si la separación es menor a 50, la fórmula es distinta a la de cuando la separación es mayor, ajustando así el cálculo según la densidad de las marcas.

En esta clase también se utilizó el modelo en espiral, empezando con la definición de objetivos para gestionar tres subtipos con reglas diferentes, continuando con el análisis de riesgos asociado a las variaciones de espaciamiento y separación, desarrollando (desarrollo) los botones, cálculos y registros en tabla, y finalizando con la planificación del siguiente ciclo para mejorar las fórmulas y validar resultados con condiciones reales.

La clase ***FrmM20*** gestiona las marcas M-20, correspondientes a los reductores de velocidad, y permite al usuario calcular la pintura necesaria según las características del camino y la velocidad permitida. Durante la inicialización, se agrega un JLabel al panel de fondo con el título “Marcas para identificar reductores de velocidad (M-20)” para identificar claramente la sección. La clase define un método calcularTotal que determina la cantidad de pintura requerida considerando el tipo de camino y la velocidad; para carreteras y calles, se calcula la longitud diagonal del reductor, el número de líneas y el área de cada línea, multiplicando estos valores según la velocidad para obtener el total de pintura necesaria.

El método agregarBotonesEspecificos crea un botón que, al presionarse, despliega un panel donde el usuario ingresa el ancho del camino y la velocidad. Una vez confirmados los datos, se agrega una nueva fila a la tabla con la distancia, el tipo de raya y la clasificación M-20. El total de pintura se calcula multiplicando por cuatro, considerando que cada proyecto incluye cuatro reductores. El valor calculado se guarda además en pinturaOriginal para seguimiento de cambios.

Esta clase también aplicó el modelo en espiral, iniciando con la definición de objetivos para determinar cómo influirían el ancho del camino y la velocidad en el cálculo, siguiendo con el análisis de riesgos por posibles errores en los valores ingresados y variaciones en las normas, avanzando con el desarrollo del panel, cálculos y estructura de registro, y concluyendo con la planificación de iteraciones futuras para mejorar la precisión del cálculo y la experiencia del usuario.

FrmSeñalesVerticales

La clase ***FrmSeñalesVerticales*** es una ventana que permite agregar señales verticales al proyecto vial que el usuario este trabajando, mostrando, agregando y eliminando registros en una tabla asociada al proyecto. La tabla tiene columnas para cantidad, ubicación, código, señal, dimensión y descripción, y se configura con transparencia y anchos específicos para cada columna.

Se definió la necesidad de una interfaz que permitiera registrar señales verticales dentro del proyecto vial, administrando información proveniente de AutoCAD y organizándola en una tabla con las columnas descritas anteriormente.

Se consideró el riesgo de inconsistencias entre los datos exportados desde AutoCAD (los textos seleccionados sean distintos, códigos incompletos, textos faltantes) y la estructura interna requerida por el sistema. También se identificaron riesgos de duplicidad de señales y errores en la descripción automática.

Los botones principales son: btnAgregarMVC y btnAgregarMVE, que permiten agregar nuevas señales a la tabla leyendo datos de un archivo generado por AutoCAD; btnAgregarMVC agrega señales con señal, dimensión y ubicación siendo para las señales de camino abierto, mientras que btnAgregarMVE agrega además el código, que se usa para los entronques; ambos usan el método manejarBtnAgregar para mostrar instrucciones en un JOptionPane y ejecutar la lógica de lectura de archivo, procesamiento de datos en bloques, obtención de la descripción con obtenerDescripcion y adición de filas al tableModel.

El botón btnEliminar elimina la fila seleccionada en la tabla mostrando primero un mensaje con los datos y solicitando confirmación; si el usuario confirma, crea un objeto SeñalesVerticales y llama a EliminarSeñalBaja para borrar la señal de la base de datos, eliminando la fila del tableModel en caso de éxito.

El botón btnAceptar guarda los cambios realizados, revisa duplicados en la combinación señal + dimensión + ubicación, solicita confirmación si se encuentra alguno y finalmente llama a AgregarSeñalVertical para guardar cada fila, mostrando un resumen de filas guardadas y fallidas. Además, en caso de que haya una señal que ya se haya guardado en la base datos con los mismos datos se manda un mensaje para preguntar al usuario si quiere guarda la señal en caso de aceptar a esa señal se le agrega un "A" para diferenciar con la que ya está en la base de datos.

El botón btnExportar cierra la ventana y abre FrmExportarTablasSV para exportar la tabla a AutoCAD.

Entre los métodos importantes, cargarDatos() obtiene los datos de señales del proyecto y los muestra en la tabla, gestionando duplicados y eliminando de las señales la "A" que se agregó para diferenciar las señales iguales que estén en ambos lados del camino.

FrmDispositivos

La clase ***FrmDispositivos*** incorpora elementos propios para la gestión de dispositivos relacionados con la señalización vial. Como en el resto de las clases, se definen y configuran los componentes de la interfaz, organizando los botones, tablas y paneles que el usuario utilizará.

la clase inicializa los elementos principales y aplica estilos mediante funciones auxiliares ya mencionadas, como estilizarBoton, que garantiza que todos los botones tengan la misma apariencia y comportamiento dentro del sistema. A este método le agregamos una nueva función: estilizarToggle, la cual se encarga de aplicar un formato visual parecido a los de estilizarBoton, pero aplicado a los botones de tipo alternable (JToggleButton). Este botón permite representar estados activados o desactivados, y gracias a esta función podemos mostrar dos tablas dentro de la misma tabla, que serían la tabla de Dispositivos y la Tabla de Botones , para ello estilizarToggle ajusta los colores, tipografía y dimensiones definidos en el proyecto.

En esta clase, el desarrollo siguió una espira del modelo en espiral. En la determinación de objetivos, se definió la necesidad de gestionar dispositivos dentro de la señalización vial, así como integrar un mecanismo visual que permitiera alternar de manera clara entre la tabla de dispositivos y la tabla de botones mediante controles tipo JToggleButton. En la identificación y análisis de riesgos, se consideraron posibles inconsistencias en la visualización cuando ambas tablas compartieran el mismo espacio, además del riesgo de que los botones alterables no mantuvieran uniformidad estética

con los demás componentes de la interfaz. Durante la fase de desarrollo y verificación, se implementó la función estilizarToggle, diseñada para aplicar un formato visual coherente con el resto del sistema, ajustando colores, tipografía y dimensiones para garantizar una correcta interpretación visual del estado activado o desactivado del botón; posteriormente se verificó su funcionamiento alternando entre ambas tablas. Finalmente, en la planificación de la siguiente iteración, se proyectó mejorar la gestión dinámica de dispositivos y habilitar futuras expansiones para permitir filtros adicionales, edición avanzada o integración con otros módulos del sistema.

Se creo un JToggleButton el cual permitirá hacer cambios de visualización de lo que se muestra en la ventana. Lo que hace es mostrar los botones según lo que se esté haciendo en el caso de los botones solo se muestra el botones de Agregar Botones y el de Agregar Curvas que solo importantes para el cálculo de los botones, en el caso de los dispositivos lo que se muestra son los botones para calcular la Indicadores de Alineamiento, las Balizas, los dispositivos Antideslumbrantes, los Delimitadores, y los Botones Reflejantes que van sobre la defensa metálica.

Primero tenemos el botón btnAgregarBotones se encarga de agregar los botones sobre las señales horizontales cargadas en el proyecto. La visibilidad del botón se controla según la lógica del flujo de trabajo del proyecto.

Al activarse, el botón invoca la función manejarBtnAgregar(), la cual despliega instrucciones para guiar al usuario en el proceso de agregación de botones, para ello primero en las instrucciones se la indica al usuario que para agregar botones primero debe hacer los pasos que inicia el Botón de "Agregar Curvas", ya que esto nos sire para identificar cuanta distancia es curva del tramo que se está trabajando en el proyecto y cuanto es tangente, lo cual es importante para los botones porque según sea el caso entonces la distancia entre botones puede ser de 30 o de 15 m. para tangente y para curva respectivamente, y el usuario ya ha hecho lo de indicar la distancia total de curva, entonces, ejecuta un conjunto de procedimientos para obtener los datos de las señales horizontales mediante la función RegresarTablaCom() de la clase ***SeñalesHorizontales*** y verifica que existan registros disponibles. En caso contrario, se notifica al usuario para que agregue primero señalamiento horizontal.

La función continúa evaluando casos especiales de clasificación, como M-1.4 y M-5, que requieren un manejo especial. Para cada señal, se verifica si ya existe en la tabla para evitar duplicación y se acumulan las distancias totales correspondientes cuando es necesario unificar botones de una misma

marca dentro de un mismo renglón. A continuación, se calcula la cantidad de botones a agregar utilizando parámetros como la distancia de la línea, la ubicación de las curvas y el tipo de camino del proyecto. En estos cálculos, se invoca obtenerClave() y obtenerDescripcion() para complementar la información de la tabla con la clave de botón y sus características correspondientes.

Para los casos especiales que requieren una interacción adicional por parte del usuario, como M-9 o M-5, se despliegan paneles con campos de entrada y selección, esto permite agregar información adicional correspondiente a el ancho del camino o el color de la marca, para M-9 y para M-5 respectivamente integrando los resultados de forma consistente en la tabla de botones.

Finalmente, una vez procesadas todas las señales, se ejecuta la agregación de botones para las marcas unificadas de M-1.4 y M-5, aplicando los cálculos pertinentes y registrando los datos en la tabla. La función garantiza que todos los botones requeridos se agreguen de manera organizada y conforme a la normativa del proyecto, integrando las funcionalidades de verificación de existencia previa, cálculo de distancias y presentación de datos en la interfaz.

Y las funciones obtenerClave() y obtenerDescripcion() que se usaron son las siguientes:

```
public static String obtenerClave(String raya) {
	Map<String, String> descripciones = new HashMap<>();

	descripciones.put("M-1.1", "BRM-A2");
	descripciones.put("M-1.2", "BRM-A2");
	descripciones.put("M-1.3", "BRM-A2");
	descripciones.put("M-1.4", "BRM-A2");
	descripciones.put("M-2.1", "BRM-B1");
	descripciones.put("M-2.3", "BRM-B1");
	descripciones.put("M-3.1", "BRM-B2");
	descripciones.put("M-3.3", "BRM-A1");
	descripciones.put("M-5.1", "BRM-B1");
	descripciones.put("M-5.2", "BRM-B1");
	descripciones.put("M-9", "BT");
	descripciones.put("M-14.1", "BRM-R1");
	descripciones.put("M-14.2", "BRM-R1");
	descripciones.put("M-17", "BRM-B1");

	// Devuelve la descripción si existe, si no, un mensaje por defecto
```

```
        return descripciones.getOrDefault(raya, "Clave no encontrada");
}

public static String[] obtenerDescripcion(String clave) {
        Map<String, String[]> descripciones = new HashMap<>();

        descripciones.put("BRM-A1", new String[] { "AMARILLO", "UNA CARA" });
        descripciones.put("BRM-A2", new String[] { "AMARILLO", "DOS CARAS" });
        descripciones.put("BRM-B1", new String[] { "BLANCO", "UNA CARA" });
        descripciones.put("BRM-B2", new String[] { "BLANCO", "DOS CARAS" });
        descripciones.put("BRM-R1", new String[] { "ROJO", "UNA CARA" });
        descripciones.put("BT", new String[] { "BLANCO", "UNA CARA" });

        return descripciones.getOrDefault(clave,
                        new String[] { "Color no encontrado", "Característica no
encontrada" });
}
```

El botón btnAgregarIndiAlin está diseñado para la gestión de indicadores de alineamiento, tanto en tangentes como en curvas. Al activarse, despliega primero un cuadro de confirmación para determinar si existen tangentes con indicadores de alineamiento. En caso afirmativo, se presentan instrucciones que guían al usuario en la interacción con AutoCAD a través del comando GuardarLongitudes. Posteriormente, se leen las longitudes desde el archivo correspondiente, se calcula la distancia total y, con base en una separación de 40 metros, se determina el número de indicadores necesarios. Estos datos se registran en la tabla de dispositivos (tableModelDisp) bajo la clasificación “DD-1”.

De forma similar, se solicita confirmación respecto a la existencia de curvas con indicadores de alineamiento. Si se confirman curvas, el programa solicita al usuario ingresar el grado de curvatura para cada una. Dicho parámetro se utiliza para calcular la separación entre indicadores (24/gradCurv), lo que permite estimar el total a instalar a lo largo de la curva. Los resultados se integran también en la tabla de dispositivos bajo la clave “DD-1”.

El botón btnAgregarBaliza gestiona la incorporación de balizas en función de las marcas horizontales tipo M-5 que deben estar previamente cargadas en el proyecto. Una vez activado, despliega instrucciones que indican la necesidad de contar con dichas marcas antes de continuar. Posteriormente, se recuperan los datos de señalamiento horizontal mediante la función RegresarTablaCom(), verificando la existencia de registros válidos. Si se identifican marcas de tipo M-5.1, se calcula la cantidad de balizas a partir de la distancia total registrada y la separación normativa (2 m para caminos convencionales y 0.5 m para vías ciclistas). Los resultados se agregan a la tabla bajo la clasificación "DD-2", señalando explícitamente la distancia de separación aplicada. El botón btnAgregarDelimita regula la inserción de delimitadores de confinamiento a partir de las marcas horizontales M-2.2 o M-1.4. Al igual que en los casos anteriores, se solicita confirmación al usuario y se revisan los datos cargados en el proyecto. Si se detecta alguna de las clasificaciones válidas, se aplica una separación estándar de 3.8 metros para calcular la cantidad total de dispositivos requeridos. El resultado se registra en la tabla con la clave "DCM-A1-R1", señalando en la columna de posición la referencia "A CADA 2 m.", conforme a la normatividad vigente.

El botón btnAgregarDispAnti permite la integración de dispositivos antideslumbrantes (malla o valla) asociados a la existencia de barrera central en el proyecto. Para ello, solicita al usuario seleccionar entre las opciones disponibles y recupera las distancias desde la clase ***BarreraTotal***. Si no existe barrera central cargada, se emite un mensaje de advertencia. En caso contrario, se calcula la cantidad de dispositivos según su tipología: en el caso de la valla, una unidad por cada 0.7 metros, mientras que, en el caso de la malla, se equipara directamente a la distancia total. Adicionalmente, se calcula la cantidad de botones reflejantes necesarios para la barrera central, estimando un dispositivo cada 10 metros en ambos lados, con clave "BRE-A1".

El botón btnAgregarBtnRefle se centra en la instalación de botones reflejantes sobre defensas metálicas y barreras centrales. Tras la confirmación, se recuperan los datos de longitudes desde la clase ***BarreraTotal***, sumando la distancia de defensa metálica y de barrera New Jersey. El cálculo considera un dispositivo cada 10 metros, ajustando la cantidad al doble cuando la barrera corresponde a un "Nivel de Contención 3". Los datos obtenidos se registran en la tabla con la clave "BRE-A1", especificando como posición "AMBOS LADOS".

En la clase también se incluyen los métodos para carga la información, cuya estructura es similar a la que ya se ha descrito previamente en otras secciones del proyecto. El método cargarDatos() se encarga de recuperar, a través de la clase ***Dispositivos***, los registros almacenados en la base de datos y mostrarlos directamente en la tabla de dispositivos (tableModelDisp). Si no existen datos asociados al proyecto actual, se despliega un mensaje de advertencia.

Por su parte, el método cargarDatosBotones() recupera los elementos clasificados como botones. Para ello, utiliza la clase ***Botones*** como puente con la base de datos y, en caso de obtener resultados, los organiza en un modelo de tabla específico (tableModelBtn) que contempla atributos como clave, color, tipo de raya, ubicación, características y cantidad. Adicionalmente, este procedimiento registra los identificadores de las marcas agregadas en la lista idMarcasAgre, lo que facilita un seguimiento posterior en la gestión del proyecto.

FrmDefensa

La clase ***FrmDefensa*** permite gestionar la información de defensas viales dentro del proyecto. Presenta una tabla con columnas para el kilometraje inicial, final, la posición (lado derecho o izquierdo), el número de piezas de amortiguamiento y la longitud de la defensa.

Los botones btnAgregarDefeLD y btnAgregarDefeLI permiten agregar registros de defensas, indicando automáticamente si corresponden al lado derecho o al lado izquierdo. Por medio del método agregarFilaDefensa Estos botones muestran al usuario los paso seguir en AutoCAD y, tras leer los datos del archivo de texto, calculan la longitud de la defensa.

El botón btnEliminar Se comporta de igual forma que en clases anteriores al elegir una fila y esta es la se elimina de la base de datos. El botón btnAceptar guarda los registros actuales de la tabla en la base de datos. Recorre cada fila, crea un objeto Defensa y lo intenta registrar con AgregarDefensa.

FrmBarreraTotal

La clase ***FrmBarreraTotal*** permite gestionar y calcular diferentes tipos de barreras viales dentro del proyecto. Está diseñada de forma que muestra una tabla con información de cada tipo de barrera y permite actualizar sus valores según lo que el usuario indique o seleccione. También se encarga de guardar estos datos en la base de datos.

En esta clase, el desarrollo se realizó siguiendo una espira del modelo en espiral. En la determinación de objetivos, se estableció que la ventana debía permitir gestionar de forma clara los diferentes tipos de barreras viales, mostrando en una tabla su información esencial y permitiendo al usuario modificarla según el nivel de contención o el tipo de elemento seleccionado. En el análisis de riesgos, se consideraron problemas como inconsistencias en los valores modificados por el usuario, errores en el cálculo automático de parámetros asociados a cada tipo de barrera y la necesidad de evitar que datos incompletos fueran guardados en la base de datos. Durante la fase de desarrollo y verificación, se implementaron los controles principales la tabla, el JComboBox para elegir el nivel de contención y los botones para actualizar los valores— verificando que cada modificación se reflejara correctamente en la interfaz y que los datos resultantes pudieran ser almacenados sin errores.

La ventana se compone principalmente de una tabla se muestran las barreras, un JComboBox que permite elegir el nivel de contención, y una serie de **botones** que actualizan los valores según el tipo de barrera.

Los botones que se crearon sirven para registrar los valores las cantidades de los siguientes elementos:

- **Barrera Metálica**: al presionarse abre una ventana de instrucciones. Después, obtiene los datos de defensa metálica que ya haya sido guardada en el proyecto, asignándolo en la tabla, la defensa debe ser calculada antes de hacer estos cálculos.
- **Barrera Central**: al presionarse se indica al usuario que debe usar el comando 'DistanciaCurva' en AutoCAD. Luego se leen los datos de un archivo de texto y se calcula la distancia entre puntos seleccionados. Ese valor se coloca en la tabla.
- **Sección de Transición de Rigidez**: despliega un cuadro con un JSpinner para que el usuario indique el número de transiciones. Ese valor se inserta directamente en la tabla.
- **Sección de Amortiguamiento**: funciona de manera similar a la barrera metálica, pero en este caso calcula el total de las secciones extremas de amortiguamiento a partir de los datos de defensa, por ello la defensa ya debe de estar cargada antes de calcular los amortiguadores.
- **Amortiguador de Impacto**: abre una ventana con un JSpinner donde el usuario indica cuántos amortiguadores existen, y se guarda el valor en la tabla.

- **Cola de Pato**: al igual que los anteriores, abre un JSpinner para que el usuario indique la cantidad de secciones terminales tipo “Cola de Pato”. El número se guarda en la tabla.
- **Actualizar**: guarda todos los valores en la base de datos, creando o actualizando un objeto BarreraTotal. Valida si hubo cambios antes de proceder.

Y como las demás clases carga los datos de la base de datos, personaliza los botones a utilizar, lee los archivos de texto correspondientes y utiliza el método manejarBtnAgregar para mostrar al usuario las instrucciones y componentes necesarios para hacer los cálculos.

FrmExportarTabla

Las siguientes clases corresponden a vistas previas de tablas de datos, desarrolladas a partir de las clases que realizan los calculo y guardan en la base de datos. Estas versiones permiten al usuario revisar y confirmar los datos, guardados en la base de datos, antes de enviarlos a AutoCAD, mostrando la información organizada en las tablas como se mostrarán en AutoCAD.

Cada clase incluye una tabla que refleja los datos pertinentes al tipo de señal o elemento correspondiente, y se mantiene la estructura de columnas y estilos visuales ya definidos. Permitiendo Exportar la tabla de totales, si corresponde, tras confirmar los datos agregados o modificados por el usuario.

En este conjunto de clases de exportación (FrmExportarTablaBarreras, FrmExportarTablaDefensa, FrmExportarTablaDispYBtn, FrmExportarTablasSH y FrmExportarTablasSV) se refleja nuevamente la aplicación del modelo en espiral dentro del desarrollo del sistema, pues cada una de ellas constituye una iteración enfocada en cubrir una necesidad específica de salida de información hacia AutoCAD.

En la etapa de definición de objetivos, se determinó que cada módulo debía tomar los datos previamente calculados y almacenados en la base de datos y prepararlos con el formato exacto requerido por los comandos LISP correspondientes. Después, en la identificación y análisis de riesgos, se evaluó la posibilidad de inconsistencias en los datos, errores de lectura o escritura de archivos y diferencias en la estructura de cada una de las tablas, ajustando el diseño para asegurar un manejo uniforme mediante métodos comunes como escribirArchivo. Posteriormente, en la fase de desarrollo y construcción, se implementaron las clases reutilizando la lógica ya existente, pero

especializándolas según el tipo de señal o dispositivo, asegurando que cada exportación incluyera los datos correctos y las unidades adecuadas según el elemento vial. Finalmente, en la etapa de planificación y revisión, cada clase fue verificada para asegurar que los datos exportados coincidieran con la estructura requerida por AutoCAD y que los usuarios pudieran revisar visualmente la tabla antes de confirmar la exportación. Así, estas clases representan un ciclo adicional dentro del modelo en espiral, consolidando la información generada en etapas previas y garantizando un flujo seguro y confiable hacia el sistema de dibujo asistido.

Dado que la mayoría de las configuraciones de las tablas y estilos se mantienen con las clases de los cálculos, la descripción detallada de estas se omite en cada clase, enfocándose únicamente en la lógica específica de exportación y manejo de datos en el botón "Aceptar" y también al momento de cargar los datos.

Además, en estas clases se usa el método escribirArchivo se encarga de guardar todas las cadenas de un ArrayList<String> en un archivo especificado por rutaArchivo. Para cada elemento de la lista, escribe la línea en el archivo y agrega un salto de línea. Maneja posibles errores mediante try-catch y asegura que el recurso BufferedWriter se cierre correctamente al finalizar la operación, evitando pérdidas de información o bloqueo del archivo.

La clase ***FrmExportarTablaBarreras*** se construye con un fondo personalizado, una tabla con formato transparente, etiquetas de encabezado y botones personalizados.

El botón Aceptar su funcionamiento es el siguiente: al presionarlo, primero se muestra una ventana de confirmación preguntando si realmente se desea exportar la tabla. Si el usuario acepta, el programa recorre cada fila de la tabla y construye una línea de texto con los datos de la barrera, agregando automáticamente la unidad correspondiente: metros lineales para las barreras metálicas y centrales, o piezas para los demás elementos. Todas esas líneas se guardan en un archivo de texto en la carpeta temporal del usuario. Al finalizar, se notifica mediante un mensaje que la exportación terminó y se indican las instrucciones para cargar la tabla en AutoCAD usando el comando CrearTablaBarrera.

La función cargarDatos() obtiene los datos del proyecto mediante RegresarTablaBarreras y verifica que exista un nivel de contención definido; si no hay datos, se muestra un mensaje de advertencia. Para cada tipo de barrera o sección registrada (barrera metálica, New Jersey, secciones de rigidez y

amortiguamiento, amortiguador y cola de pato), se agregan filas a la tabla solo si su cantidad es diferente de cero, asignando etiquetas de identificación según el nivel de contención y construyendo la descripción correspondiente.

La clase ***FrmExportarTablaDefensa*** se construye con un fondo personalizado, una tabla con formato transparente, etiquetas de encabezado y botones personalizados.
El botón Aceptar es el encargado de ejecutar la exportación de datos. Primero, al hacer clic, aparece un cuadro de confirmación preguntando si se desea exportar la tabla. Si el usuario acepta, se recorre fila por fila la tabla, extrayendo los valores de inicio, final, posición, sección de amortiguamiento y longitud. Luego, esos datos se organizan en una cadena separada por comas y se guardan en un archivo de texto en la ruta establecida (DatosTablaDefensa.txt). Finalmente, se muestra un mensaje indicando que la exportación ha sido completada y explicando cómo usar el archivo en AutoCAD mediante el comando CrearTablaDefensa.
La función cargarDatos() se encarga de obtener los registros de defensa a partir del proyecto activo mediante RegresarTablaDefensa, verificando primero si existen datos; en caso contrario, se muestra un mensaje de advertencia. Los datos se agregan a un ArrayList, se ordenan por kilómetro inicial considerando la parte antes y después del símbolo '+', y finalmente se insertan en la tabla, haciendo un redondeo de la longitud a valores enteros.
La clase ***FrmExportarTablaDispYBtn*** se construye con un fondo personalizado, una tabla con formato transparente, etiquetas de encabezado y botones personalizados.
El botón Aceptar gestiona la exportación de las tablas a archivos de texto, primero se solicita confirmación para exportar la tabla de dispositivos; si el usuario acepta, se recorre cada fila de tablaDispositivos y se construye una cadena separada por comas con sus valores, agregando estas líneas a un ArrayList que luego se escribe en DatosTablaDisp.txt mediante el método escribirArchivo. Después, se solicita confirmación para la tabla de botones y, si se confirma, se realiza un proceso similar, cuidando que los valores que contengan comas se coloquen entre comillas, y se guarda el resultado en DatosTablaBtn.txt. Tras cada exportación, se muestra un mensaje indicando al usuario el comando que debe usar en AutoCAD para crear la tabla correspondiente.

En cuanto a la carga de datos, la clase implementa cargarDatos para dispositivos y cargarDatosBotones para los botones. En dispositivos, se agrupan los elementos por clave sumando cantidades y se ordenan considerando los números dentro de la clave, generando una lista limpia y organizada que se muestra en la tabla. Para los botones, se manejan casos especiales como las rayas M-1.1 a M-1.4 que en caso de existir un M-1.4, todos los botones que se agrupan en M-1 total y las M-5.1 y M-5.2 que se agrupan en M-5, sumando las cantidades y ordenando por los números de la raya. Esto asegura que los datos mostrados en la vista previa reflejen exactamente la estructura que se exportará a AutoCAD.

La clase ***FrmExportarTablasSH*** se construye con un fondo personalizado, una tabla con formato transparente, etiquetas de encabezado y botones personalizados.
El botón Aceptar se solicita confirmación para exportar los datos, si el usuario acepta, se recorre cada fila de la tabla y se construye una línea de texto separada por comas. En caso de que el campo de observaciones contenga comas, se encierra entre comillas para mantener el formato correcto. Todas las líneas se almacenan en un ArrayList y luego se escriben en el archivo mediante el método escribirArchivo. Finalmente, se muestra un mensaje indicando al usuario que, en AutoCAD, puede usar el comando CrearTablaSB para visualizar la tabla con los datos exportados.

```
JButton botonAceptar = new JButton("Aceptar");
botonAceptar.setBounds(535, 427, 89, 23);
estilizarBoton(botonAceptar, new Color(40, 167, 69));
botonAceptar.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		int confirm = JOptionPane.showConfirmDialog(null, "¿Deseas Exportar
esta tabla?", "Confirmar",
				JOptionPane.YES_NO_OPTION);
		ArrayList<String> lineas = new ArrayList<>();
		if (confirm == JOptionPane.YES_OPTION) {
			for (int i = 0; i < modeloTabla.getRowCount(); i++) {
				String clave = modeloTabla.getValueAt(i, 0).toString();
				String color = modeloTabla.getValueAt(i, 1).toString();
				String dimension = modeloTabla.getValueAt(i,
2).toString();
```

```
                        String cantidad = modeloTabla.getValueAt(i, 3).toString();
                        String observaciones = modeloTabla.getValueAt(i,
4).toString();
                        String coma;
                        String cantidadMl = "ml.";
                        String linea;
                        if (observaciones.contains(",")) {
                              coma = "\""+observaciones+"\"";
                              linea = String.join(",", clave, color, dimension,
cantidad, cantidadMl, coma);
                        }else {
                              linea = String.join(",", clave, color, dimension,
cantidad, cantidadMl, observaciones);
                        }

                        lineas.add(linea);
                  }
                  escribirArchivo(rutaArchivo, lineas);
                  JOptionPane.showMessageDialog(null, "Expotacion Finalizada:\n"
                              + "Vaya al proyecto de AutoCAD con el LISP cargado
(o regrese a la ventana anterior para cargar el LISP)"
                              + "\n" + "Escriba el comando 'CrearTablaSB' para
mostrar la tabla" + "\n", "Información",
                              JOptionPane.INFORMATION_MESSAGE);
            }
      }
});
panelFondo.add(botonAceptar);
```

El cargarDatos obtiene las señales horizontales del proyecto a través de la clase SeñalesHorizontales. Primero se agrupan las señales por clasificación, sumando las cantidades de pintura de señales con la misma clasificación. Después, se ordena la lista considerando tanto los números como las letras de la clasificación, de manera que la tabla quede organizada de manera lógica. Para la columna de

dimensión se configura un JComboBox con opciones para que el usuario pueda modificar la dimensión según se pida en el proyecto.

La clase ***FrmExportarTablasSV*** se construye con un fondo personalizado, una tabla con formato transparente, etiquetas de encabezado y botones personalizados.

Aquí también se creó el botón Agregar OD-12, visible únicamente al cambiar a la vista de totales, al presionarlo, se muestran instrucciones para que el usuario seleccione las curvas en AutoCAD y capture los kilómetros iniciales y finales. Luego, se solicita el grado de curvatura para cada curva, se calcula la separación adecuada según este grado y se determina el número total de señales OD-12 a agregar (totalOD12), además se le pide al usuario que indique si es doble vista o no en caso afirmativo entonces la señales se multiplicaran por 2. Finalmente, se agrega una nueva fila a tablaSVTotal con la cantidad calculada, el tipo de señal OD-12 y la dimensión inicial seleccionada mediante un JComboBox, asegurando que la tabla de totales refleje correctamente las modificaciones antes de la exportación. Esto se agregó como después de la

El botón Aceptar permite exportar los datos mostrados en la tabla actual, primero preguntando al usuario si desea exportar únicamente las filas seleccionadas; si no hay selección, se notifica al usuario. Para las filas seleccionadas se construye un arreglo con los valores concatenados de cada columna y se guarda en un archivo de texto mediante escribirArchivo. Posteriormente, se ofrece la opción de exportar la tabla de totales; si se realizaron cambios con OD-12, se solicita confirmar su guardado, y la última fila se inserta en la base de datos antes de generar el archivo de totales. Tras completar la exportación, puede usar el comando CrearTablaSVTotal en AutoCAD para importar los datos.

En cargarDatos() se consultan las señales verticales del proyecto usando la clase SeñalesVerticales, filtrando aquellas de tipo OD-12 y eliminando los sufijos "A", que se agregaron en caso de dos señales iguales en ambos lados del camino, los registros se cargan directamente en el modelo de la tabla principal tablaSeñalesVerticales, asegurando que la tabla refleje correctamente la información relevante para la vista previa normal. Por su parte, cargarDatosTotales() agrupa los datos de todas las señales por tipo y clasificación, sumando cantidades y ordenando según un prefijo definido, construyendo así el modelo tablaSVTotal que se utilizará cuando el usuario active la vista de tabla de totales.

5.5. Implementación del controlador

El controlador se implementó mediante el manejo de eventos asociados a los botones y de los componentes principales de cada formulario, coordinando el flujo de datos entre el modelo y las vistas. En este esquema, la vista se representa por las interfaces gráficas (JFrame y JTable), mientras que el modelo está constituido por las clases que encapsulan la información de proyecto (como SeñalesVerticales, SeñalesHorizontales, Proyectos, etc.). El controlador, implementado en cada ventana, responde a las acciones del usuario y ejecuta procesos de carga, validación y exportación de la información.

En este proceso también se refleja claramente la aplicación del modelo en espiral, ya que cada nueva funcionalidad implicó recorrer nuevamente las cuatro etapas del ciclo. En la definición de objetivos, cada vez que se añadía un nuevo botón en la vista o una nueva acción del usuario, se establecía qué datos debían almacenarse, modificarse o recuperarse de la base de datos. Posteriormente, durante el análisis de riesgos, se evaluaban posibles inconsistencias como errores al insertar campos no válidos, fallos en la conexión, estructuras incompletas en las tablas o duplicidad de identificadores, ajustando tanto la lógica del modelo como las validaciones en la vista. En la fase de desarrollo y construcción, primero se implementaba la clase del modelo correspondiente, después se añadía el método específico en la clase Conexion y finalmente se integraba en el evento del botón dentro de la vista, lo cual permitía cerrar el ciclo funcional de cada nueva característica. Finalmente, en la fase de planificación y revisión, se verificaba que la interacción entre modelo, vista y controlador siguiera un flujo correcto de ida y vuelta en los datos, comprobando que los registros se insertaran o consultaran adecuadamente, y se realizaban ajustes para mejorar la robustez y consistencia del sistema. De esta manera, el controlador y la lógica de base de datos evolucionaron de forma incremental, ampliándose con cada vuelta del modelo en espiral conforme se agregaban nuevas herramientas al proyecto.

La gestión de eventos se concentra principalmente en los botones Aceptar, encargados de iniciar el proceso de exportación, que permiten introducir información complementaria. En el caso de Aceptar, el controlador recupera los datos del modelo cargados en las tablas de la vista, valida las selecciones realizadas por el usuario y ya sea que lo guarda en la base de datos o genera archivos de texto con los registros procesados según la tarea que el usuario este realizando, los cuales posteriormente se integrarán en AutoCAD mediante comandos definidos.

Además, se desarrolló la clase ***Conexion***, la cual actúa como intermediaria entre la aplicación y la base de datos MySQL. En esta clase se guarda toda la lógica de conexión y organiza el acceso a los datos del resto de la aplicación. Para gestionar la conexión con la base de datos se utilizó un archivo externo denominado config.properties, en el cual se definen los parámetros necesarios para establecer la comunicación con el servidor MySQL. Este archivo contiene propiedades como la URL de conexión, el usuario y la contraseña, lo que permite modificar la configuración sin alterar el código fuente.

El método conectar() devuelve un objeto Connection listo para ejecutar consultas, manejando errores y mostrando mensajes de advertencia en caso de fallos de conexión.

Sobre esta base, se implementaron métodos especializados para cada tabla definida en la base de datos. Por ejemplo, para la tabla proyectos, el método AgregarProyecto() inserta un nuevo registro calculando de manera dinámica el identificador mediante la consulta MAX(id) y sumando uno al último valor encontrado.

De igual manera, RegresarProyectos() devuelve un arreglo de objetos Proyecto con todos los registros disponibles, y ObtenerIdProyecto() permite obtener la clave primaria de un proyecto a partir de su nombre y tipo de vía.

En el caso de señales_bajas, se desarrollaron operaciones como AgregarSeñalBaja() que validan si un registro ya existe antes de insertarlo, EliminarSeñalBaja() que aplica validaciones de si existe el registro y entonces lo elimina en caso afirmativo y métodos de consulta (RegresarTabla, RegresarTablaCom) que devuelven los registros asociados a un proyecto en particular.

Este mismo patrón de métodos se replica para el resto de las tablas (señales_verticales, defensa, botones, dispositivos, barrera), garantizando uniformidad en las operaciones CRUD y manteniendo la consistencia de los datos.

De esta forma, Conexion centraliza todo el acceso a la base de datos, funcionando como capa DAO (Data Access Object) dentro de la arquitectura de la aplicación, y permitiendo que los controladores y vistas trabajen únicamente con objetos de negocio (Proyecto, SeñalesHorizontales, etc.) sin preocuparse por la lógica interna de consultas SQL ni la gestión de recursos JDBC.

5.6. Comunicación con AutoCAD

Se creó la clase ***CargarAutoCAD*** la cual cumple la función de puente entre la aplicación y el entorno de AutoCAD, automatizando la apertura de archivos de dibujo DWG y la carga de scripts de AutoLISP necesarios para la ejecución de cálculos y generación de tablas.

Desde la perspectiva del modelo en espiral, se realiza la identificación de objetivos, verificando la existencia de una instalación válida de AutoCAD para asegurar que el entorno es adecuado y que los scripts estén creados de manera correcta con su funcionalidad correspondiente. En segundo lugar, se aplica la análisis y evaluación de riesgos, deteniendo el proceso si AutoCAD no se encuentra o si existe algún riesgo que impida continuar. Después, en la fase de desarrollo y construcción, se generan la carpeta MisLisp, los archivos LISP necesarios, verificando que todos los scripts que se crearon se le carguen correctamente al DWG seleccionado por el usuario. Finalmente, en la etapa de prueba y validación, AutoCAD se ejecuta con el archivo DWG seleccionado y el script generado, confirmando que el archivo queda listo para el usuario y que todos los módulos funcionan correctamente.

Su lógica principal se concentra en el método AbrirArchivo(), el cual inicia con la detección de la ruta de instalación de AutoCAD a través del registro de Windows. En caso de no localizar la aplicación, el proceso se interrumpe, asegurando que la ejecución continúe únicamente si existe un entorno válido de AutoCAD.

Una vez validada la ruta, el método muestra un cuadro de diálogo mediante JFileChooser, permitiendo al usuario seleccionar el archivo DWG que se abrirá en AutoCAD. Posteriormente, la clase genera de manera dinámica la carpeta MisLisp en el directorio de documentos del usuario, donde copia los scripts LISP y archivos TXT auxiliares que se encuentran empaquetados como recursos de la aplicación. Esto garantiza que el entorno de trabajo cuente siempre con los ficheros actualizados y disponibles, reemplazando versiones previas si existen.

Con los archivos preparados, la clase construye un script SCR en el directorio de documentos, cuyo contenido establece la carpeta de trabajo y carga secuencialmente cada uno de los archivos LISP definidos en la lista. Este archivo SCR funciona como un archivo de instrucciones que AutoCAD interpreta en el arranque, asegurando que todos los módulos de cálculo (longitudes, ángulos, distancias, textos, tablas de señalamiento) se encuentren disponibles al abrir el dibujo.

Finalmente, la ejecución de AutoCAD se realiza mediante un ProcessBuilder, que abre la aplicación con el archivo DWG seleccionado e incluye el script SCR generado. Con ello, el usuario accede a un entorno de AutoCAD listo para trabajar con las rutinas definidas, sin necesidad de cargar manualmente los LISP.

```
package AutoCAD;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.List;
import javax.swing.JFileChooser;

public class CargarAutoCAD {

    // Abre un archivo DWG en AutoCAD y carga los scripts Lisp
    public static void AbrirArchivo() {
       String rutaAutoCAD = obtenerRutaAutoCAD();
       if (rutaAutoCAD == null) {
           System.err.println("No se encontró AutoCAD instalado.");
           return;
       }

         JFileChooser selectorArchivos = new JFileChooser();
         selectorArchivos.setDialogTitle("Selecciona un archivo DWG para abrir en
AutoCAD");
         selectorArchivos.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter("Archivos DWG", "dwg"));

         int resultado = selectorArchivos.showOpenDialog(null);
```

```
        if (resultado == JFileChooser.APPROVE_OPTION) {
            File archivoSeleccionado = selectorArchivos.getSelectedFile();
            String rutaArchivoDWG = archivoSeleccionado.getAbsolutePath();

            String carpetaMisLisp = System.getProperty("user.home") +
"\\Documents\\MisLisp\\";
            new File(carpetaMisLisp).mkdirs();

            // Lista de scripts Lisp
            List<String> listaLisp = Arrays.asList(
                "seleccionarLineas.lsp", "calcularAngulo.lsp", "crearTablaSB.lsp",
"seleccionarAreas.lsp",
                "seleccionarTextos.lsp", "seleccionarTextosEntr.lsp",
"crearTablaSV.lsp", "crearTablaSVTotales.lsp",
                "KmEntrePuntos.lsp", "crearTablaDefensa.lsp",
"crearTablaBotones.lsp", "DistanciaCurva.lsp",
                "crearTablaDispositivos.lsp", "crearTablaBarreras.lsp"
            );
            // Lista de archivos de texto
            List<String> listaTxt = Arrays.asList(
                "Angulo.txt", "Areas.txt", "DatosTablaBarrera.txt",
"DatosTablaBtn.txt",
                "DatosTablaDefensa.txt", "DatosTablaDisp.txt", "DatosTablaSB.txt",
"DatosTablaSV.txt",
                "DatosTablaSVTot.txt", "Distancias.txt", "KmCurva.txt",
"KmDistancia.txt",
                "TextosSeleccionados.txt", "TextosSeleccionadosEnt.txt"
            );

            // Copiar archivos Lisp a la carpeta destino
            for (String nombre : listaLisp) {
                try (InputStream is =
CargarAutoCAD.class.getResourceAsStream("/Lisp/" + nombre)) {
                    if (is == null) {
```

```
                        System.err.println("No se encontró el recurso: " + nombre);
                        continue;
                    }
                    Files.copy(is, new File(carpetaMisLisp + nombre).toPath(),
java.nio.file.StandardCopyOption.REPLACE_EXISTING);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            // Copiar archivos TXT a la misma carpeta
            for (String nombre : listaTxt) {
                try (InputStream is =
CargarAutoCAD.class.getResourceAsStream("/TXT/" + nombre)) {
                    if (is == null) {
                        System.err.println("No se encontró el recurso: " + nombre);
                        continue;
                    }
                    Files.copy(is, new File(carpetaMisLisp + nombre).toPath(),
java.nio.file.StandardCopyOption.REPLACE_EXISTING);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            // Crear archivo .scr que cargará los Lisp en AutoCAD
            String rutaScript = System.getProperty("user.home") +
"\\Documents\\cargarLisp.scr";

            try (PrintWriter escritor = new PrintWriter(rutaScript)) {
                escritor.println("(vl-file-syst-write-dir \"" +
carpetaMisLisp.replace("\\", "\\\\") + "\")");
                for (String nombre : listaLisp) {
                    String rutaLispAbs = carpetaMisLisp + nombre;
```

```
                    escritor.println("(load \"" + rutaLispAbs.replace("\\", "\\\\")
+ "\")");
                }
                escritor.println("(princ)");
            } catch (IOException e) {
                e.printStackTrace();
                System.out.println("Error al crear el archivo SCR.");
                return;
            }

            // Ejecutar AutoCAD con el archivo seleccionado y el script
            ProcessBuilder constructorProceso = new ProcessBuilder(rutaAutoCAD,
rutaArchivoDWG, "/b", rutaScript);

            try {
             @SuppressWarnings("unused")
                Process proceso = constructorProceso.start();
                System.out.println("AutoCAD se está iniciando con el archivo: " +
rutaArchivoDWG);
            } catch (IOException e) {
                e.printStackTrace();
                System.out.println("Error al abrir AutoCAD.");
            }
        } else {
            System.out.println("No se seleccionó ningún archivo.");
        }
    }

    // Obtiene la ruta de instalación de AutoCAD desde el registro de Windows
    private static String obtenerRutaAutoCAD() {
        try {
            @SuppressWarnings("deprecation")
                    Process proceso = Runtime.getRuntime().exec("reg query
\"HKLM\\SOFTWARE\\Autodesk\\AutoCAD\" /s /v AcadLocation");
```

```
            BufferedReader lector = new BufferedReader(new
InputStreamReader(proceso.getInputStream()));
            String linea;
            while ((linea = lector.readLine()) != null) {
                linea = linea.trim();
                if (linea.startsWith("AcadLocation")) {
                    String[] partes = linea.split("\\s+", 3);
                    if (partes.length == 3) {
                        String ruta = partes[2];
                        if (!ruta.endsWith("\\")) ruta += "\\";
                        return ruta + "acad.exe";
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Ahora bien, con ***CargarAutoCAD*** se habilita la carga automática de una serie de procedimientos desarrolladas en AutoLISP, las cuales cumplen un papel importante en la automatización de cálculos y en la extracción de información geométrica directamente desde los dibujos de AutoCAD. Estos scripts fueron diseñados para interactuar con los elementos del modelo (líneas, polilíneas, áreas, textos y bloques), procesar sus propiedades y exportar los resultados a archivos de texto. Dichos archivos se integran posteriormente en la aplicación Java, permitiendo que los datos trabajados en AutoCAD se sincronicen con las vistas y tablas de la interfaz gráfica.

CrearTablaDispositivos

El script CrearTablaDispositivos.lsp, tiene como finalidad generar en AutoCAD una tabla dinámica a partir de los datos previamente exportados desde la aplicación Java al archivo DatosTablaDisp.txt.

Para lograrlo, en primer lugar, se implementa la función auxiliar parse-line, encargada de interpretar cada línea del archivo de texto, separando sus valores con base en un delimitador (en este caso de una coma) y respetando posibles valores que contengan comillas. Con ello se garantiza que los registros puedan ser reconstruidos en forma de lista, lista que posteriormente servirá como entrada para llenar las celdas de la tabla.

El comando principal definido en la función c:CrearTablaDispositivos inicia obteniendo la ruta estándar de trabajo (MisLisp en la carpeta de Documentos del usuario) y localizando el archivo DatosTablaDisp.txt, que contiene la información procesada en la capa Java. Dicho archivo es leído línea por línea y cada registro se transforma en una estructura de lista mediante la función parse-line. En caso de no encontrarse o no poder abrirse, el programa emite un mensaje en la línea de comandos y termina su ejecución.

Posteriormente, se configuran los parámetros de la tabla a insertar en AutoCAD. Se definen los nombres de columna (DISPOSITIVOS, CLAVE, POSICIÓN, CANTIDAD, UNIDAD) y se establecen los anchos específicos de cada una, lo que permite un formato controlado de visualización. También se determinan las dimensiones totales de la tabla, donde el número de filas se calcula dinámicamente en función de la cantidad de registros más las filas de encabezado. Para construir el objeto gráfico, se utiliza el método vla-AddTable, que inserta la tabla en el espacio modelo en una posición temporal inicial (0,0).

Una vez creado el objeto, se ajusta su estética: alineación centrada en los tres niveles (título, encabezado y datos), tipografía basada en el estilo estándar, alturas de texto diferenciadas (5 para el título, 3 para los encabezados y 2.7 para los datos), color uniforme y tipo de línea continua. Seguidamente se asigna el título de la tabla (“TIPOS DE DISPOSITIVOS”), se colocan los encabezados de cada columna y, de forma iterativa, se llenan las filas con los valores extraídos del archivo de datos.

Finalmente, el script solicita al usuario un punto de inserción en el dibujo mediante getpoint, de manera que la tabla pueda colocarse directamente en la posición deseada dentro del modelo. El objeto tabla se desplaza desde su punto base hasta el punto indicado, simulando así el comportamiento de un “pegar con cursor”. Con esta estructura, el script asegura que cualquier conjunto de datos exportados por la aplicación puede visualizarse de forma estructurada y estandarizada en AutoCAD, manteniendo coherencia con el resto de tablas del sistema.

Entonces usando el comando CrearTablaDispositivos dentro de AutoCAD obtenemos:

Figura 5.6.1. Comando: CrearTablaDispositivos

CrearTablaDefensa

El script CrearTablaDefensa.lsp tiene como objetivo generar en AutoCAD una tabla que documenta la colocación de barreras metálicas de dos crestas, usando como fuente el archivo DatosTablaDefensa.txt.

Al igual que en CrearTablaDispositivos.lsp, se utiliza la función parse-line para separar cada línea del archivo en campos, de modo que los datos puedan ser insertados en la tabla. La tabla se crea nuevamente en el espacio modelo y se aplican configuraciones de filas, columnas y estilos similares a las descritas previamente.

Se establece un título principal y los nombres de las columnas correspondientes a la información de inicio, final, posición, sección de amortiguamiento y longitud. Los datos del archivo se cargan de manera secuencial, siguiendo la misma lógica que se explicó en el LISP anterior.

Finalmente, se solicita al usuario un punto de inserción para colocar la tabla, replicando el mecanismo de “pegar con cursor” usado en el script anterior. Esto permite ubicar la tabla en la posición deseada dentro del plano sin necesidad de moverla manualmente después.

Entonces usando el comando CrearTablaDefensa dentro de AutoCAD obtenemos:

Figura 5.6.2. Comando: CrearTablaDefensa

CrearTablaBotones.

El script CrearTablaBotones.lsp genera una tabla en AutoCAD que documenta la información de los botones de señalización, tomando como fuente el archivo DatosTablaBtn.txt.

Se reutiliza la función parse-line ya explicada en CrearTablaDispositivos para separar las líneas en campos, y la lógica de creación de tabla, configuración de filas y columnas, así como la estética y estilos, es la misma que en los scripts anteriores.

El título y encabezados se adaptan a esta tabla: se crea un encabezado general “TIPOS DE DISPOSITIVOS” y un subencabezado específico “BOTONES”. Luego se cargan los nombres de columnas correspondientes a clave, color, raya, ubicación, características y cantidad, y los datos se insertan fila por fila.

Al igual que en los LISP previos, se solicita al usuario un punto de inserción para colocar la tabla en el plano, usando el mismo mecanismo de “pegar con cursor”.

Entonces usando el comando CrearTablaBotones dentro de AutoCAD obtenemos:

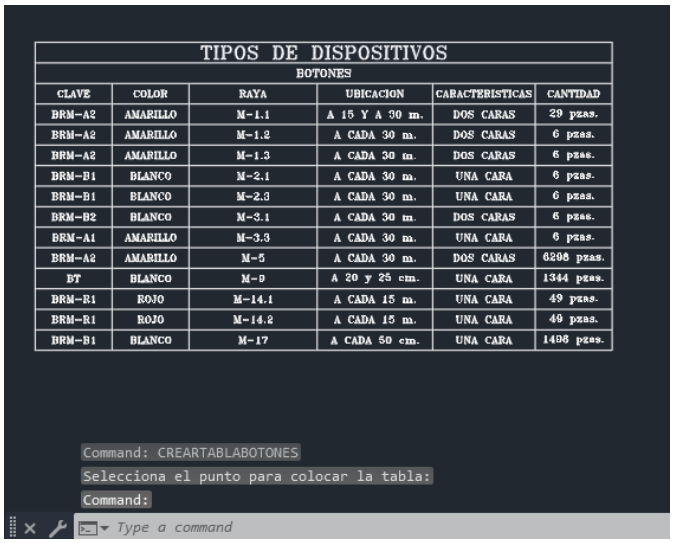


Figura 5.6.3. Comando: CrearTablaBotones

CrearTablaSB

El script CrearTablaSB tiene como objetivo generar una tabla en AutoCAD para el señalamiento horizontal, tomando los datos del archivo DatosTablaSB.txt ubicado en la carpeta MisLisp del usuario. Al igual que en los LISP anteriores, se utiliza la función parse-line para separar cada línea del archivo en campos según el delimitador coma.

La tabla se configura con las columnas "CLAVE", "COLOR", "DIMENSION", "CANT.", " " y "OBSERVACIONES". Se coloca el encabezado "Señalamiento Horizontal", aplicando merge en algunas celdas superiores para centrarlo visualmente.

Posteriormente, se cargan los datos en las celdas de la tabla, manteniendo el estilo de texto, alineaciones y color igual que en los LISP previos. Finalmente, la tabla se inserta en el dibujo mediante selección de punto por parte del usuario, trasladando la tabla desde la posición (0,0) al punto elegido, de manera idéntica a lo implementado en los LISP anteriores.

Entonces usando el comando CrearTablaSB dentro de AutoCAD obtenemos:

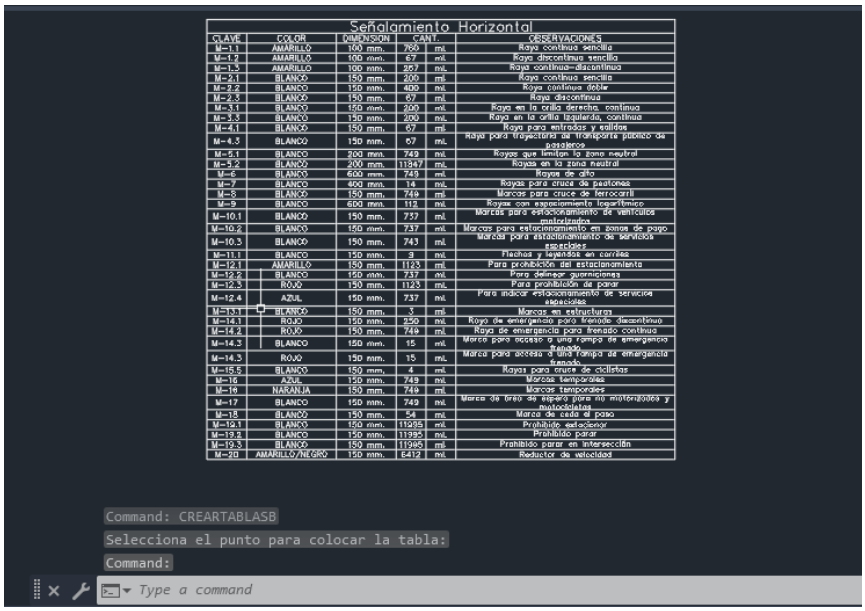


Figura 5.6.4. Comando: CrearTablaSB

CrearTablaSV

El script CrearTablaSV genera una tabla en AutoCAD con información de señalamiento vertical, tomando los datos del archivo DatosTablaSV.txt dentro de la carpeta MisLisp del usuario.

Como en los LISP anteriores, se reutiliza la función parse-line para dividir cada línea del archivo según comas y obtener los distintos campos.

La tabla se configura con las columnas "CANT", "UBICACIÓN", "CÓDIGO", "SEÑAL", "DIMENSIÓN" y "DESCRIPCIÓN". El encabezado se coloca en la primera fila con el texto "CARRETERA:" esto para que el usuario ya dentro de AutoCAD le ponga el nombre de la carretera para cumplir con el diseño solicitado, mientras que la segunda fila contiene los nombres de las columnas.

La carga de datos en las celdas sigue el mismo procedimiento que los LISP previos, añadiendo un pequeño margen a la última columna para mejorar la presentación visual. Finalmente, la tabla se inserta en el dibujo mediante selección de punto por parte del usuario, trasladando la tabla desde la posición (0,0) al punto elegido, manteniendo la estética y alineación uniforme de textos y colores, de la misma manera que en los LISP anteriores.

Entonces usando el comando CrearTablaSV dentro de AutoCAD obtenemos:

Figura 5.6.5. Comando: CrearTablaSV

CrearTablaSVTotal

El script CrearTablaSVTotal sigue la misma estructura que el LISP anterior CrearTablaSV, por lo que gran parte de su funcionamiento ya ha sido descrito previamente.

Su propósito es generar una tabla en AutoCAD con los totales de señalamiento vertical pero esta vez de los totales, tomando los datos del archivo DatosTablaSVTot.txt ubicado en la carpeta MisLisp del usuario. Se vuelve a usar la función parse-line para dividir cada línea en campos separados por comas. La tabla cuenta con las columnas "CANT", "CÓDIGO", "SEÑAL", "DIMENSIÓN" y "DESCRIPCIÓN". Se coloca un encabezado en la primera fila con el texto "SEÑALAMIENTO VERTICAL TOTALES", y la segunda fila contiene los nombres de las columnas.

La carga de datos aplica un margen al último campo para mejorar la presentación, y la inserción de la tabla se realiza mediante selección de punto por el usuario, trasladando la tabla desde (0,0) al punto elegido.

Entonces usando el comando CrearTablaSVTotal dentro de AutoCAD obtenemos:

Figura 5.6.6. Comando: CrearTablaSVTotal

CrearTablaBarrera

El script CrearTablaBarrera mantiene la misma lógica que los LISP anteriores para crear tablas en AutoCAD, reutilizando nuevamente la función parse-line para dividir cada línea de datos en campos separados por comas. Esta rutina se enfoca en generar una tabla con información sobre barreras, tomando los datos del archivo DatosTablaBarrera.txt ubicado en la carpeta MisLisp del usuario.

La tabla contiene las columnas "CLAVE", "POSICION", "CANTIDAD" y "CARACTERISTICAS". El encabezado de la tabla indica "TIPOS DE DISPOSITIVOS", mientras que la segunda fila específica "BARRERAS", uniendo sus celdas para abarcar todas las columnas. La inserción de datos comienza en la tercera fila y, al igual que en los LISP previos, se coloca un margen. Finalmente, la tabla se “pega con cursor”, permitiendo al usuario seleccionar el punto de inserción, trasladando la tabla desde (0,0) al punto elegido. En resumen, este LISP sigue el mismo patrón de los anteriores, ajustando únicamente los nombres de columnas, anchos y encabezado al contexto de las barreras.

Entonces usando el comando CrearTablaBarrera dentro de AutoCAD obtenemos:

Figura 5.6.7. Comando: CrearTablaBarrera

GuardarAngulos

El script GuardarAngulos permite calcular y registrar el ángulo entre dos líneas o polilíneas seleccionadas por el usuario, reutilizando conceptos previos de interacción con AutoCAD, pero con un enfoque en cálculos geométricos.

Primero, se define la función mi-acos para obtener el arco coseno de un valor, dado que se necesita calcular el ángulo entre vectores. Al ejecutar el LISP, se muestra un mensaje de instrucciones y se abre el archivo Angulo.txt en la carpeta MisLisp para almacenar el resultado.

El usuario debe seleccionar exactamente dos líneas o polilíneas, a partir de estas, se obtienen los puntos inicial y final de cada línea, se calculan los vectores correspondientes y se determina su producto punto. Con las magnitudes de los vectores, se calcula el coseno del ángulo y se obtiene el ángulo en radianes mediante mi-acos, convirtiéndolo posteriormente a grados.

Finalmente, el ángulo absoluto se guarda en el archivo y se notifica al usuario en la línea de comando. Esta rutina sigue la misma filosofía de automatización y registro de datos que los LISP anteriores, adaptada a cálculos de ángulos entre elementos del dibujo.

Entonces usando el comando GuardarAngulos dentro de AutoCAD obtenemos:

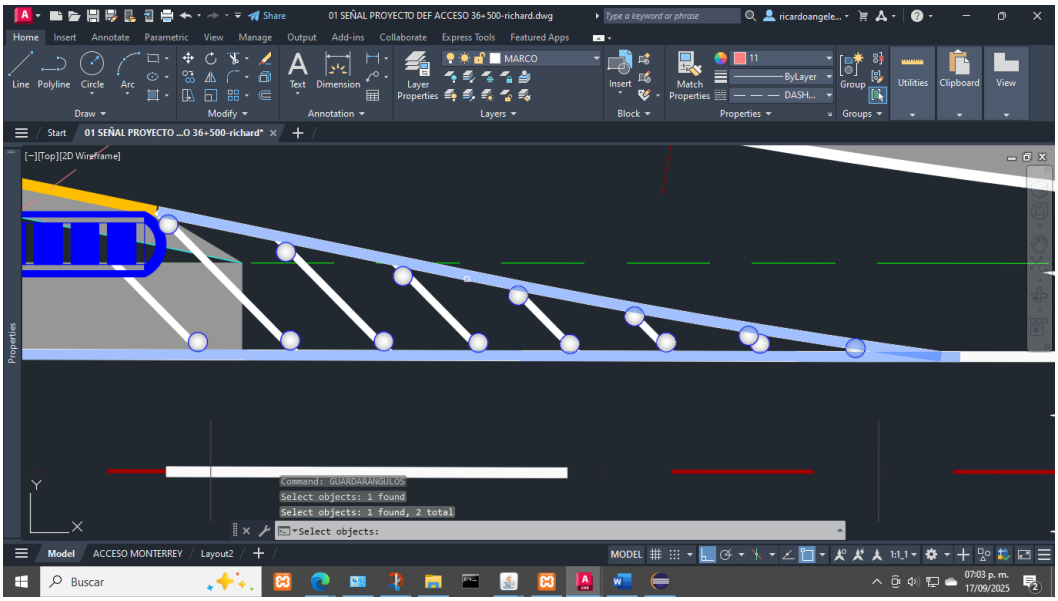


Figura 5.6 8. Comando: GuardarAngulos

SeleccionarTextos

El script SeleccionarTextos permite al usuario seleccionar cualquier texto, mtext o atributo dentro del dibujo y guardar su contenido en un archivo de texto, siguiendo la misma lógica de automatización y registro de información que los LISP anteriores.

La rutina define la ruta del archivo TextosSeleccionados.txt en la carpeta MisLisp del usuario, luego solicita la selección de entidades tipo TEXT, MTEXT o ATTRIB. Si se selecciona al menos un elemento, abre el archivo en modo escritura y recorre cada entidad, extrayendo el valor textual con entget y escribiéndolo línea por línea.

Al finalizar, cierra el archivo y notifica al usuario la ubicación donde se guardaron los textos. En caso de que no se haya seleccionado nada, muestra un mensaje indicando que no se detectaron textos. Esta función reutiliza la lógica de manejo de archivos y selección de entidades ya vista en los LISP previos, simplificando la extracción de información textual del dibujo.

```
(defun c:SeleccionarTextos (/ ss i ent texto archivo ruta)
  (setq carpeta (strcat (getenv "USERPROFILE") "\\Documents\\MisLisp\\"))
  (setq ruta (strcat carpeta "TextosSeleccionados.txt"))

  (setq ss (ssget '((0 . "TEXT,MTEXT,ATTRIB")))) ; selecciona textos y atributos
  (if ss
```

```
    (progn
      (setq archivo (open ruta "w")) ; abre archivo para escribir
      (setq i 0)
      (repeat (sslength ss)
        (setq ent (ssname ss i))
        (setq texto (cdr (assoc 1 (entget ent))))
        (write-line texto archivo)
        (setq i (1+ i))
      )
      (close archivo)
      (princ (strcat "\nTextos guardados en: " ruta))
    )
    (princ "\nNo se seleccionó ningún texto.")
  )

  (princ)
)
```

Entonces usando el comando SeleccionarTextos dentro de AutoCAD obtenemos:

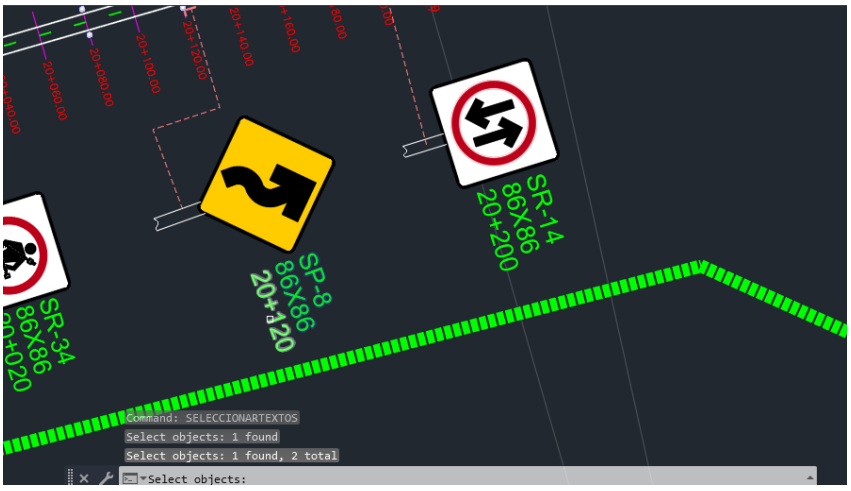


Figura 5.6.9. Comando: SeleccionarTextos

GuardarLongitudes

El script GuardarLongitudes sigue la misma dinámica que los scripts anteriores, enfocándose ahora en la extracción de longitudes de líneas y polilíneas dentro del dibujo. La rutina define la ruta del archivo Distancias.txt dentro de la carpeta MisLisp del usuario, y muestra un mensaje de instrucciones para guiar al usuario en la selección de entidades.

Una vez seleccionadas, recorre cada objeto y, utilizando las funciones de AutoLISP y ActiveX, obtiene la longitud de cada línea o polilínea mediante vlax-curve-getDistAtParam. Cada longitud se escribe en el archivo de texto en formato decimal con dos cifras significativas, de manera similar a cómo se registraron los textos o ángulos en LISP previos. Al finalizar, cierra el archivo y notifica al usuario la ubicación del registro.

```
(defun c:GuardarLongitudes ()
  (setq carpeta (strcat (getenv "USERPROFILE") "\\Documents\\MisLisp\\"))
  (setq archivo (strcat carpeta "Distancias.txt"))

  ;; Muestra el mensaje de cómo usar el LISP
  (alert "Instrucciones:\n1. Selecciona las líneas y polilíneas.\n2. La longitud
será calculada y guardada en un archivo de texto.")

  ;; Abre el archivo para escribir
  (setq archivo-handle (open archivo "w"))

  ;; Selecciona las líneas y polilíneas
  (setq seleccion (ssget '((0 . "LINE,POLYLINE,LWPOLYLINE"))))

  ;; Si hay objetos seleccionados
  (if seleccion
    (progn
      (setq i 0
            cantidad (sslength seleccion))

      ;; Itera a través de cada objeto seleccionado
      (while (< i cantidad)
```

```
        (setq entidad (ssname seleccion i))
        (setq datos (entget entidad))

        ;; Obtiene la longitud de la entidad
        (cond
          ((or (= (cdr (assoc 0 datos)) "LINE") (= (cdr (assoc 0 datos))
"POLYLINE") (= (cdr (assoc 0 datos)) "LWPOLYLINE"))
           (setq longitud (vlax-curve-getDistAtParam entidad (vlax-curve-
getEndParam entidad)))
           ;; Escribe la longitud en el archivo
           (write-line (strcat (rtos longitud 2 2)) archivo-handle)
           )
        )

        (setq i (1+ i))
      )

      ;; Muestra un mensaje en la línea de comando
      (princ (strcat "\nLongitudes guardadas en " archivo))
    )
    (princ "\nNo se seleccionaron líneas o polilíneas.")
  )

  ;; Cierra el archivo
  (close archivo-handle)
  (princ)
)
```

Entonces usando el comando GuardarLongitudes dentro de AutoCAD obtenemos:

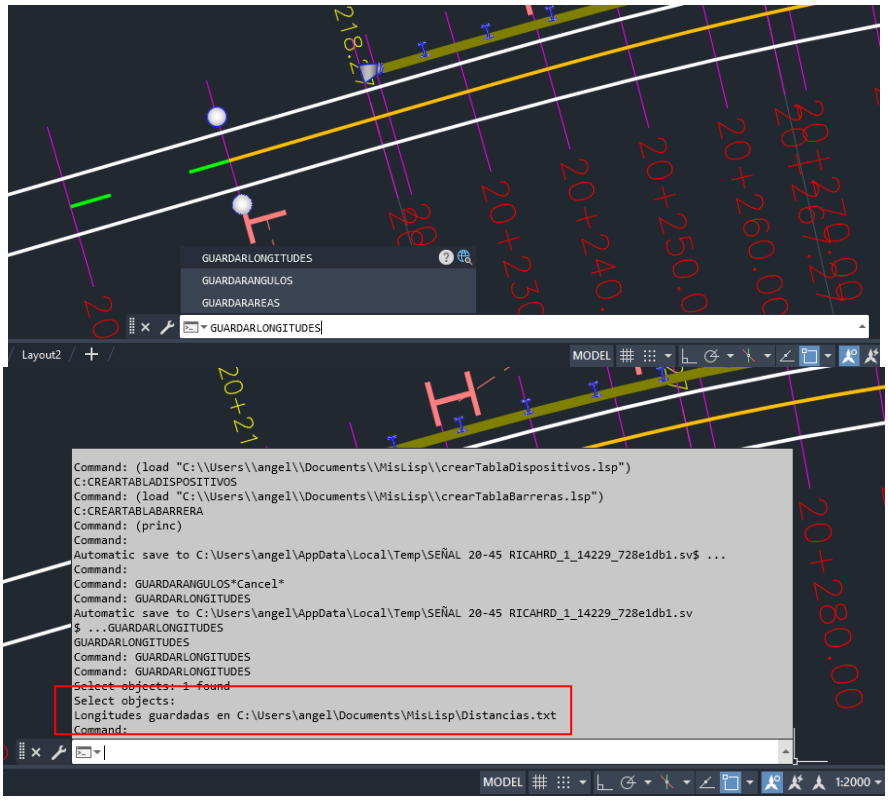


Figura 5.6.10. Comando: GuardarLongitudes

DistanciaEntrePuntos

El script DistanciaEntrePuntos permite calcular y registrar el kilometraje de dos puntos, un inicio y un fin, seleccionados sobre una línea o polilínea dentro del dibujo.

Primero define la función auxiliar round para redondear valores y formatear-km para convertir distancias a un formato de kilometraje "km+metros" con metros redondeados al múltiplo de 10 más cercano. El usuario ingresa un kilometraje inicial y, posteriormente, puede seleccionar múltiples pares de puntos sobre la línea o polilínea; para cada par, el LISP calcula la distancia acumulada desde el inicio del eje de la línea usando vlax-curve-getdistatpoint y registra los valores en el archivo KmDistancia.txt dentro de la carpeta MisLisp del usuario.

El proceso se repite hasta que el usuario decide no continuar, manteniendo la misma estructura de apertura, escritura y cierre de archivo utilizada en rutinas anteriores.

Entonces usando el comando DistanciaEntrePuntos dentro de AutoCAD obtenemos:

Figura 5.6.11. Comando: DistanciaEntrePuntos

GuardarAreas

El script GuardarAreas permite calcular el área total de los hachurados seleccionados en el dibujo y guardarla en un archivo de texto. Primero define la ruta del archivo Areas.txt dentro de la carpeta MisLisp del usuario y muestra un mensaje de instrucciones para indicar que se deben seleccionar los hachurados.

Luego, abre el archivo para escritura y obtiene todas las entidades tipo HATCH mediante ssget. Si se seleccionan objetos, recorre cada hachurado, obtiene su área con vla-get-area y acumula el valor en area-total. Una vez sumadas todas las áreas, escribe el resultado en el archivo y muestra un mensaje confirmando que los datos se guardaron correctamente. El LISP mantiene la misma estructura de apertura, escritura y cierre de archivo utilizada en rutinas previas.

Entonces usando el comando GuardarAreas dentro de AutoCAD obtenemos:

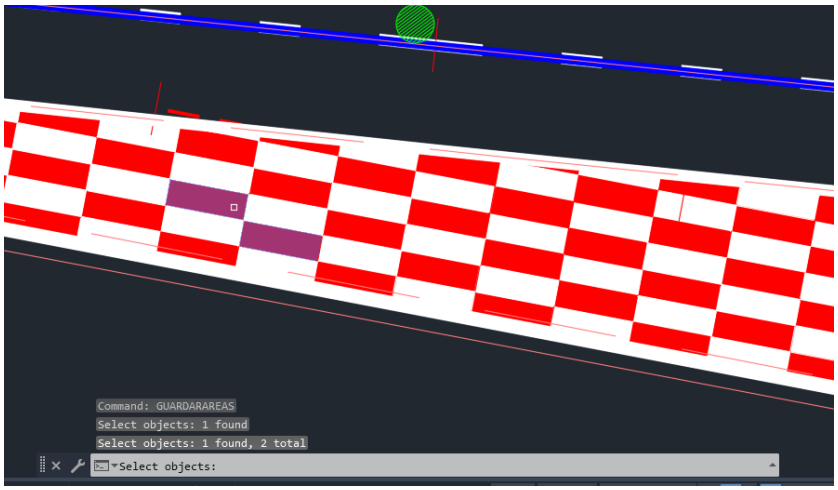


Figura 5.6.12. Comando: GuardarAreas

DistanciaCurva

El script DistanciaCurva permite calcular y guardar el kilometraje de segmentos de línea o polilínea seleccionados sobre una carretera o trazo curvo, este es similar a DistanciaEntrePuntos, la diferencia es que este LISP no hace un redonde de 10 en los metros, porque este da una distancia más precisa. Primero se define la función round para redondear correctamente valores positivos y negativos, y formatear-km para construir cadenas en formato "km+metros", asegurando que kilómetros y metros se muestren con ceros a la izquierda cuando sea necesario.

Al ejecutar el comando, el programa solicita al usuario seleccionar una línea o polilínea y verifica que la selección sea válida. Luego pide el kilometraje inicial o con el que desea comenzar, que servirá como referencia para los cálculos. El archivo KmCurva.txt se crea automáticamente en la carpeta MisLisp del usuario, centralizando los resultados.

Posteriormente, mediante un bucle, se permite al usuario seleccionar pares de puntos sobre la entidad, se calculan las distancias parciales desde el inicio de la línea, se suman al kilometraje inicial, y los resultados se formatean y guardan en el archivo. Finalmente, se pregunta si se desea continuar con otro par de puntos hasta que el usuario decida finalizar, y al cerrar el archivo, se muestra un mensaje indicando la ubicación de los datos guardados.

Entonces usando el comando DistanciaCurva dentro de AutoCAD obtenemos:

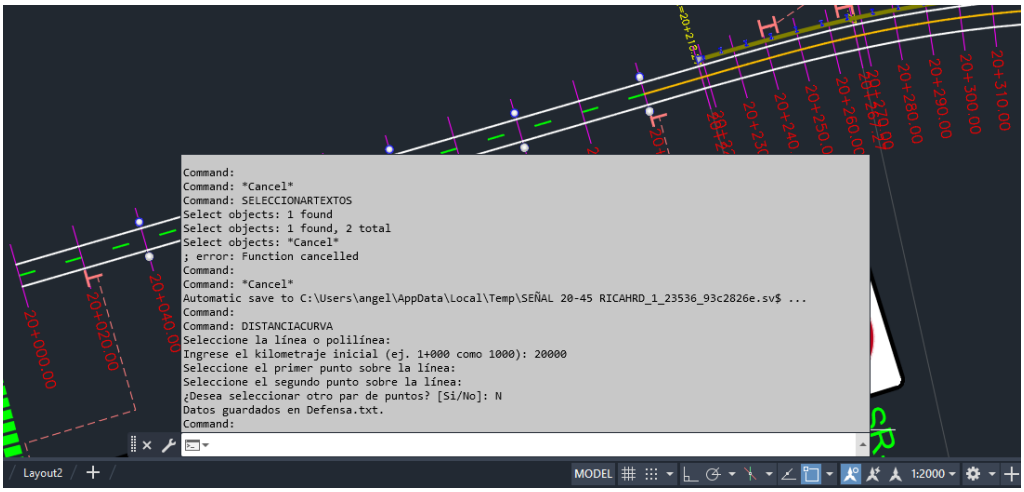


Figura 5.6.13. Comando: DistanciaCurva

5.7 Pruebas

Durante el desarrollo del sistema se realizaron principalmente pruebas funcionales y pruebas de usabilidad básicas, enfocadas en verificar el correcto funcionamiento de cada módulo y en validar que el flujo de trabajo fuera práctico para el usuario final. Debido a que el sistema fue desarrollado bajo un enfoque progresivo, las pruebas se llevaron a cabo de manera continua conforme se implementaban nuevas funcionalidades.

En las primeras iteraciones, las pruebas estuvieron orientadas a validar el funcionamiento interno de los módulos de captura de información. En esta etapa se verificó principalmente:

- El registro correcto de datos en la base de datos.
- La comunicación entre formularios y tablas.
- La generación adecuada de cálculos y cuantificaciones.
- La integración entre Java y AutoCAD para la exportación de tablas.
- Que los Scripts creados para AutoCAD guardaron y leyeran correctamente lo datos de los archivos de TXT.

Como resultado de estas pruebas iniciales, surgieron modificaciones relacionadas con:

- Corrección de errores en validaciones de datos.
- Ajustes en nombres de campos y relaciones de la base de datos.

- Reorganización de botones y formularios para mejorar la experiencia del usuario.
- Mejora en el manejo de errores y mensajes de validación.
- Ajustes a los Scripts de AutoCAD para mejora visibilidad de las tablas creadas.

Posteriormente, en iteraciones más avanzadas, las pruebas se enfocaron en la experiencia de uso y en el flujo de navegación del sistema. En esta fase se evaluó:

- La facilidad para ingresar información.
- La claridad de los formularios.
- La rapidez para generar tablas.
- La comprensión de los mensajes y opciones disponibles.

A partir de estas pruebas se realizaron cambios como:

- Reubicación de botones de acción.
- Simplificación de formularios.
- Incorporación de mensajes de confirmación y advertencia.
- Ajustes visuales en tablas y ventanas para mejorar la legibilidad.

Finalmente, se realizaron pruebas integrales del flujo completo del sistema, desde la creación del proyecto hasta la exportación de tablas a AutoCAD. Estas pruebas permitieron comprobar la estabilidad general del sistema y verificar que la información se mantuviera consistente entre módulos y base de datos.

En conjunto, las pruebas realizadas durante cada iteración permitieron la verificación progresiva tanto las funcionalidades como la interfaz del sistema, siguiendo el enfoque del modelo en espiral y permitiendo adaptar la aplicación conforme surgían nuevas necesidades o mejoras detectadas durante el desarrollo.

6. Documentación

6.1. Manual de Usuario

Aplicación: Diseños Carreteros

Versión: 1.0.0

Autor: RAE

Este documento corresponde al Manual de Usuario de la aplicación Diseños Carreteros, desarrollado en lenguaje de programación Java y diseñado con el propósito de asistir a los usuarios en el proceso de instalación, configuración y uso del sistema.
Incluye información sobre la estructura general de la aplicación, los requerimientos técnicos necesarios y los pasos básicos para comenzar a utilizar sus principales funciones.

Introducción

La aplicación Diseños Carreteros ha sido desarrollada con el objetivo de facilitar la gestión de señalización vial y el diseño de proyectos relacionados con carreteras, cumpliendo normativas de la Secretaría de Infraestructura, Comunicaciones y Transportes (SICT), conforme a su **Manual** edición 2023.
Su estructura se encuentra organizada en paquetes que separan de manera lógica los distintos módulos, abarcando desde la interfaz gráfica con el usuario hasta la conexión con bases de datos y la comunicación con AutoCAD mediante scripts en Lisp y el manejo de archivos TXT.
Entre las principales características del sistema se encuentran:

- Construcción de interfaces gráficas que le permiten realizar diferentes cálculos de la señalización vial, como, por ejemplo: Calculo de pintura para las marcas M1 a la M20, cuantificación de la señalización vertical, la cuantificación de Defensa, así como de Barreras y los diferentes dispositivos (Balizas, Delimitadores, Botones, etc.).
- Comunicación con la base de datos por medio de tablas que muestra los datos almacenados.
- Visualización previa de cómo se encuentran las tablas y la información en la base de datos, antes de exportar a AutoCAD.
- Comunicación directa con AutoCAD para la selección, lectura y almacenamiento de datos mediante scripts Lisp.
- Funcionalidades de exportación a archivos de texto, permitiendo guardar resultados y facilitar su integración con otros módulos del sistema.
- Instalación automatizada con Inno Setup, que incluye la configuración de dependencias necesarias como JDK 21 y XAMPP para el manejo de bases de datos.

El presente manual tiene como propósito proporcionar al usuario las instrucciones necesarias para instalar, configurar y comenzar a trabajar con la aplicación de forma rápida y eficiente.

Requisitos del sistema

Para asegurar un funcionamiento correcto de la aplicación, se recomienda contar con las siguientes especificaciones:

Requisitos de hardware

- **Procesador:** Intel Core i3 o equivalente como mínimo (se recomienda i5 o superior).
- **Memoria RAM:** 4 GB mínimo.
- **Espacio en disco:** 2 GB disponibles para instalación de la aplicación y dependencias.

Requisitos de software

- **Sistema operativo:** Windows 10 o superior (64 bits).
- **Java Development Kit (JDK 21):** instalado automáticamente por el instalador si no está presente.
- **Servidor local XAMPP (versión 8.0.30):** configurado automáticamente durante la instalación.
- **MySQL y phpMyAdmin:** incluidos en el paquete de XAMPP para la gestión de la base de datos.
- **AutoCAD:** AutoCAD 2016 en adelante es una versión compatible que se requiere instalar en el equipo para la ejecución de los scripts Lisp.

Instalación

Al usuario se le proporcionara un ejecutable llamado InstalarDiseñosCarreteros.exe, para su instalación es necesario otorgar permisos de administrador, una vez que se ejecute se mostrar la siguiente ventana:

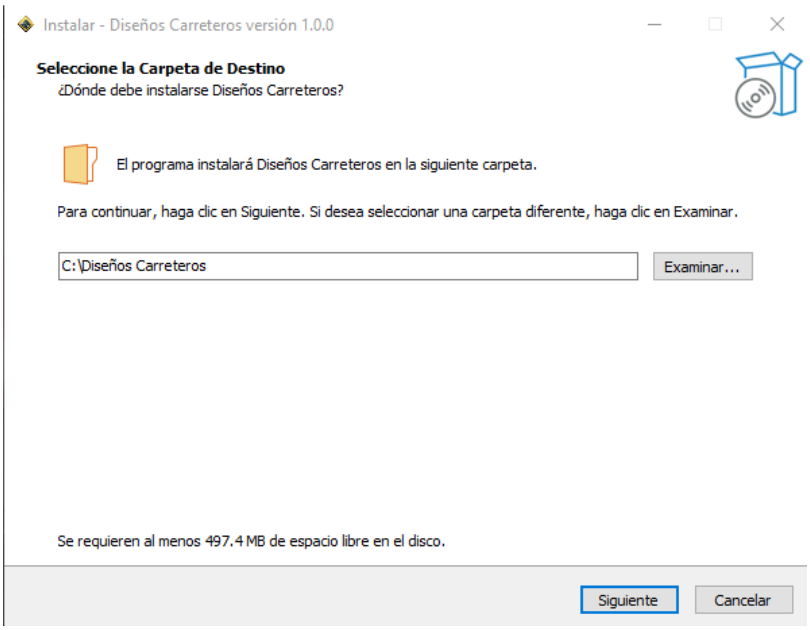


Figura 6.1. Instalación: Ruta

El usuario pude elegir en que ruta desea instalar la aplicación en este caso se instala en la ruta "*C:\Diseños Carreteros*", una vez elegida la ruta se presiona en Siguiente y se mostrara la siguiente ventana:

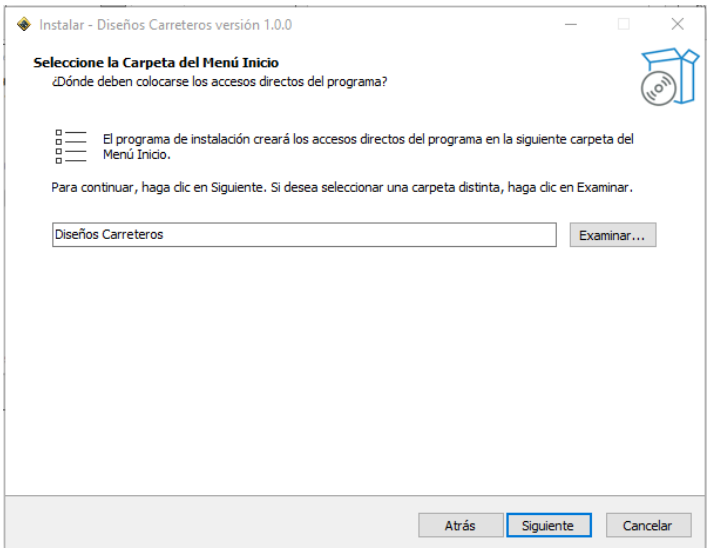


Figura 6.2. Instalación: Nombre Carpeta

Donde se le pide que indique el nombre de la carpeta donde estará los accesos directos de la aplicación, los ejecutables de la aplicación y los de desinstalación. A continuación, se le presiona en siguiente y se mostrara la siguiente ventana:

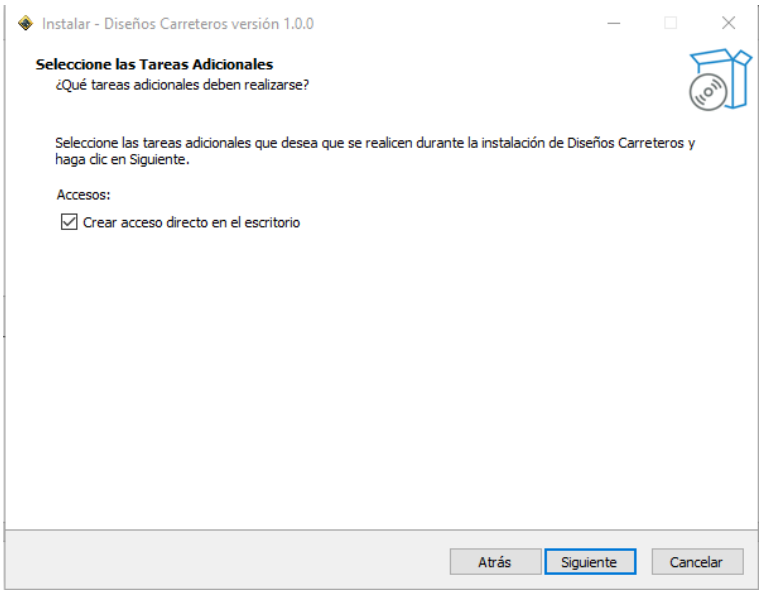


Figura 6.3. Instalación: Acceso Directo

Se le solicita al usuario si requiere de un acceso Directo en el Escritorio, se marca la casilla según las necesidades del usuario, Después se presiona en siguiente y comenzara la instalación como se muestra en la siguiente ventana:

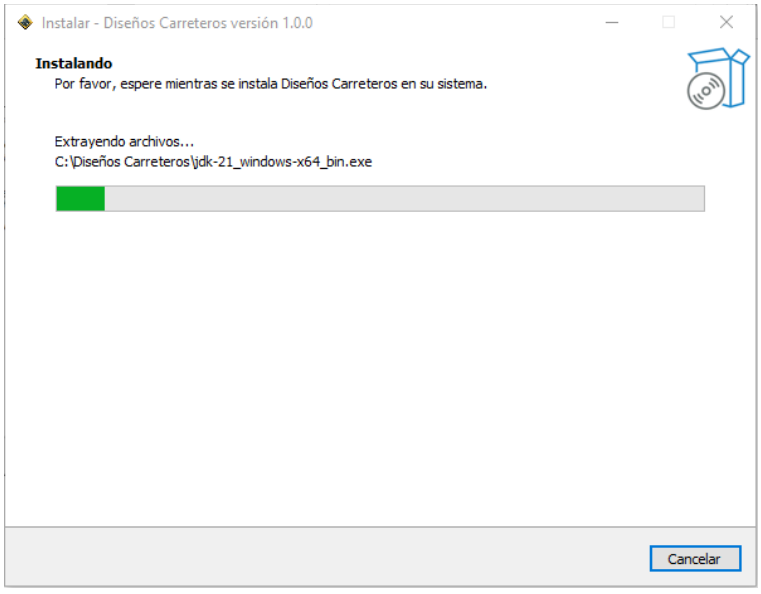


Figura 6.4. Instalación: Comienzo

Después de instalar la aplicación, comenzará la instalación del JDK 21 de Oracle y de la aplicación de XAMPP la cual abrirá una nueva ventana que mostrar el proceso de instalación de XAMPP, como se muestra a continuación:

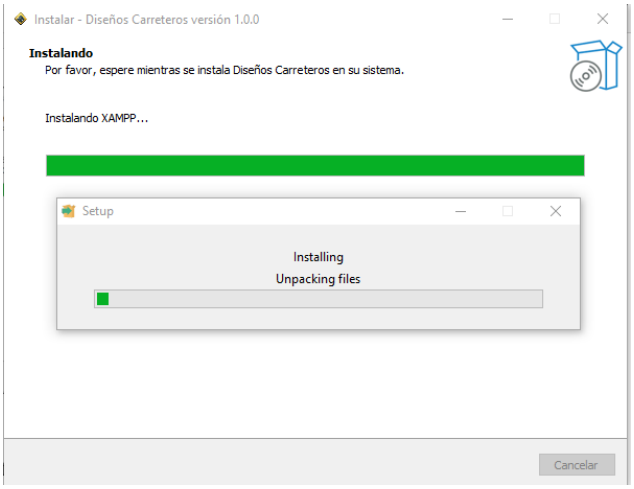


Figura 6.5. Instalación: XAMPP

Una vez terminado el proceso se mostrará ventana de finalización donde se la indica si quiere iniciar el programa, se le recomienda al usuario desmarcar la casilla, porque aún se debe configurar la Base de datos.

Figura 6.6. Instalación: Finalización

Una vez que Finalizo la instalación buscamos la aplicación de XAMPP y la ejecutamos, y se te mostrara la siguiente ventana:

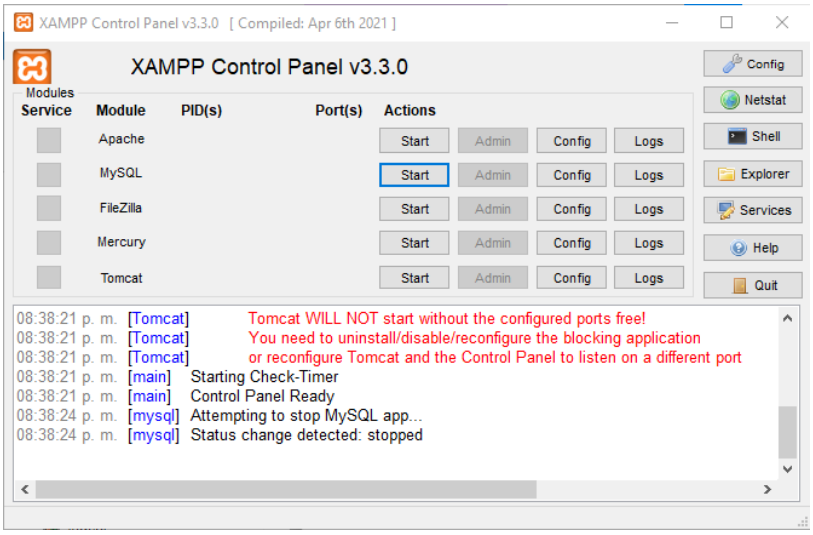


Figura 6.7. XAMPP: Inicio

Por única vez de iniciar el servicio tanto de Apache como de MySQL como se muestra a continuación:

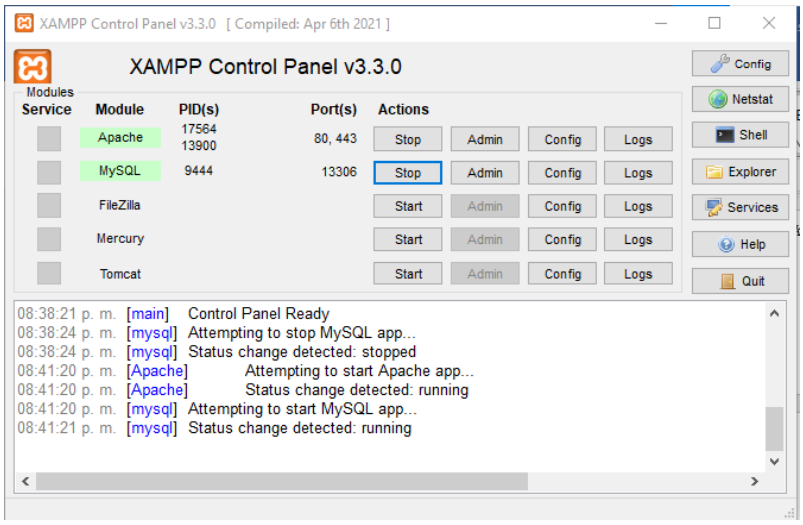


Figura 6.8. XAMPP: Inicio Apache y MySQL

Una vez iniciados ambos servicios, en la parte de MySQL se presiona el botón de Admin, el cual abrirá el navegador predeterminado del equipo del usuario, y mostrará lo siguiente:

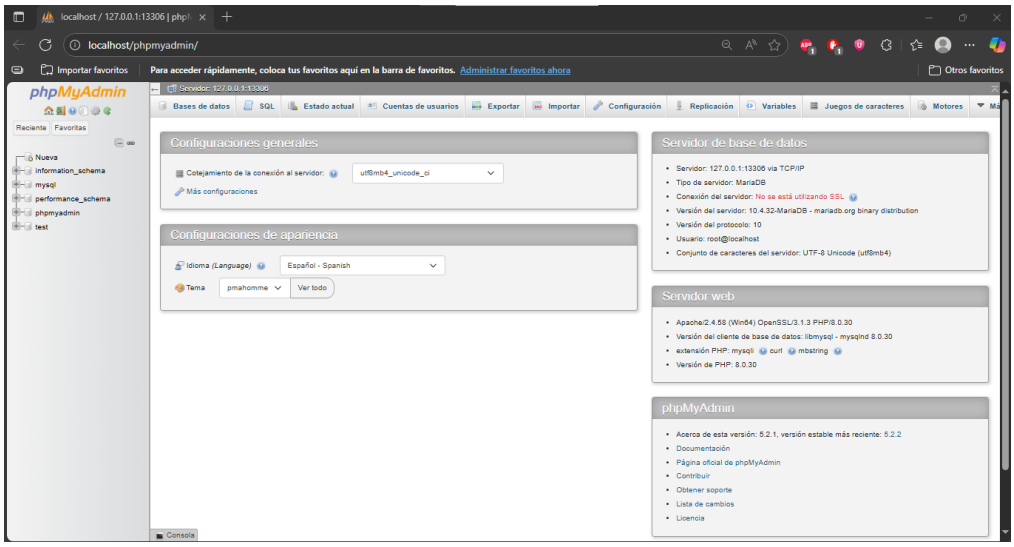


Figura 6.9. XAMPP: phpMyAdmin

En esa ventana se le presiona al apartado de “Importar” y se mostrar la siguiente ventana:

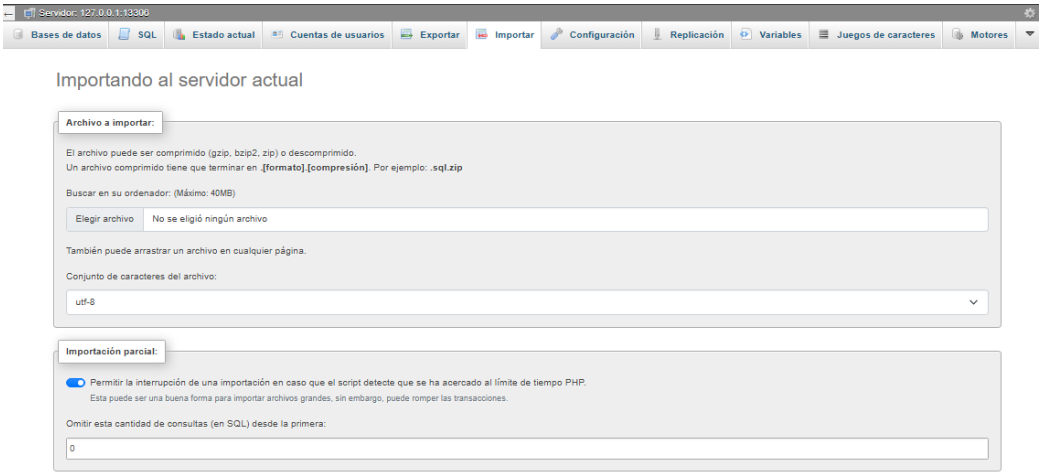


Figura 6.10. XAMPP: Abrir Archivo

En esta ventana se selecciona el botón “Elegir Archivo” aquí se abrirá un explorador de archivos y buscamos la carpeta donde se instaló el programa de Diseños Carreteros, en este caso se instaló en “C:\Diseños Carreteros” y ahí abrimos la carpeta db y se selecciona el archivo diseño_carreteras.sql y le damos en abrir.

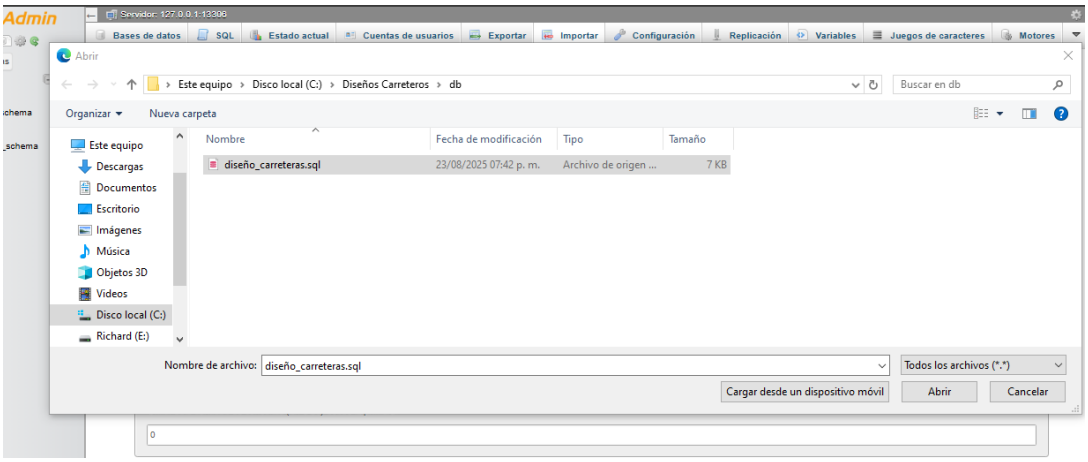


Figura 6.11. XAMPP: SQL

Entonces se le da en importar y debe salir algo parecido a lo siguiente:

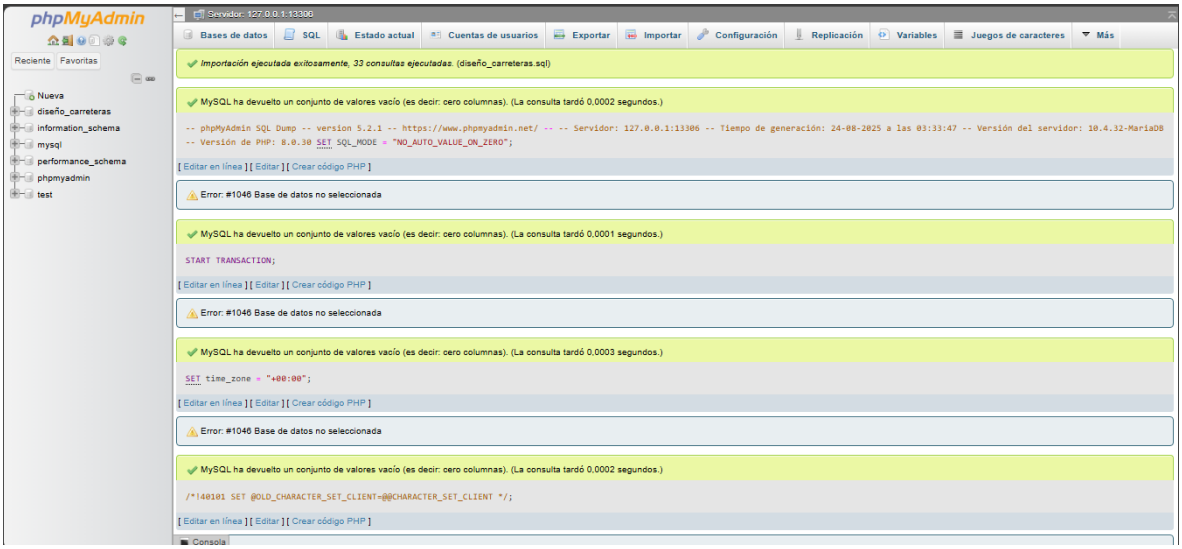


Figura 6.12. XAMPP: Importación exitosa

Una vez que se puede observar que, en la lista del lado izquierdo, que son todas las bases de datos que tienes en este servidor, el nombre de la base de datos entonces ya se terminó de configurar la base de datos y ya se puede usar la aplicación.

Funcionalidades

Para ejecutar la aplicación, primeramente, se le pide al usuario que abra XAMPP e inicie el servicio de MySQL como se muestra a continuación:

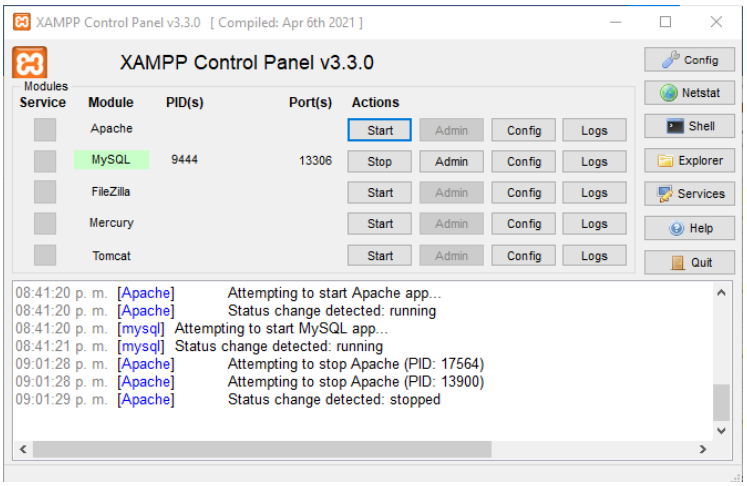


Figura 6.13. Aplicación: Iniciar XAMPP

Lo anterior para poder comunicar la aplicación con la base de datos.

➢ *Ventana Home*

Una vez activado se ejecuta la aplicación de Diseños Carreteros mostrando la siguiente vista:

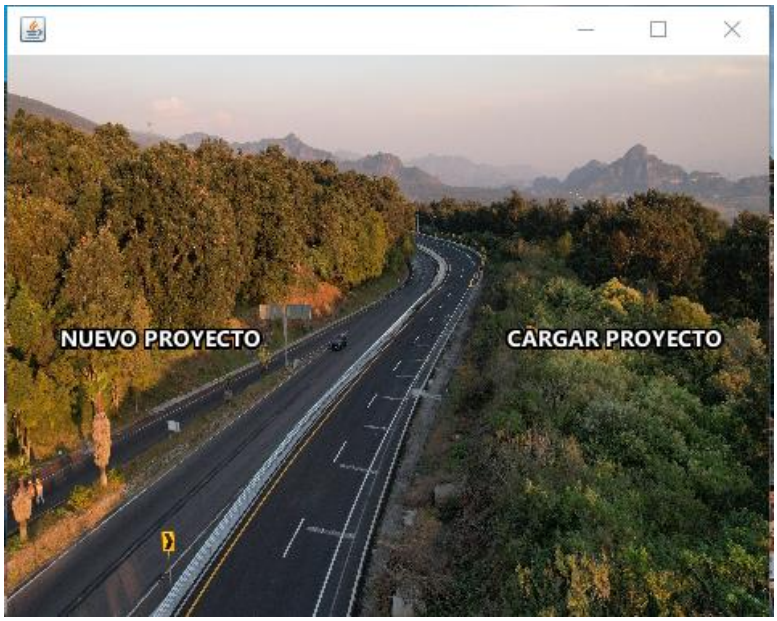


Figura 6.14. Aplicación: Vista Inicial

En esta ventana el usuario tiene 2 opciones la primera es “Nuevo Proyecto” y la segunda es “Cargar Proyecto”.

❖ Nuevo Proyecto

Si se selecciona “Nuevo Proyecto” entonces se abrirá una ventana donde se le solicitará al usuario los datos del proyecto a iniciar, se le solicitara el nombre del proyecto la velocidad y que tipo de camino es, para el tipo de camino se tiene 3 opciones a elegir si es Carretera, Calle o Vía Ciclista, como ejemplo se pondrá los siguientes datos: nombre: México - Cuernavaca (15 - 20), Velocidad: 80 y Tipo de camino: Carretera.

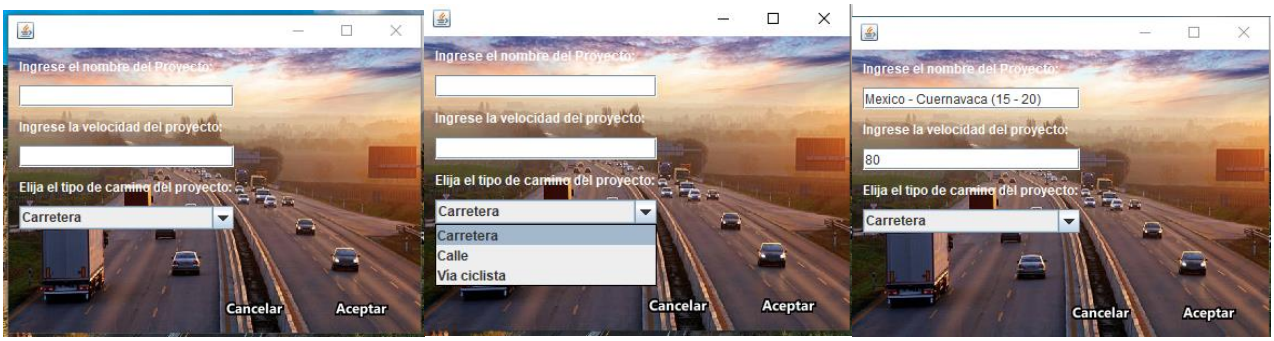


Figura 6.15. Aplicación: Nuevo Proyecto

Una vez colocados los datos y se presione en aceptar, se abre la ventana donde el usuario puede elegir que datos agregar al proyecto, si Señales Horizontales, señales Verticales, Defensa o Dispositivos, de igual forma hay un apartado para abrir un proyecto de AutoCAD, al cual se le cargaran los LIPS para poder hacer las diferentes tareas del programa.

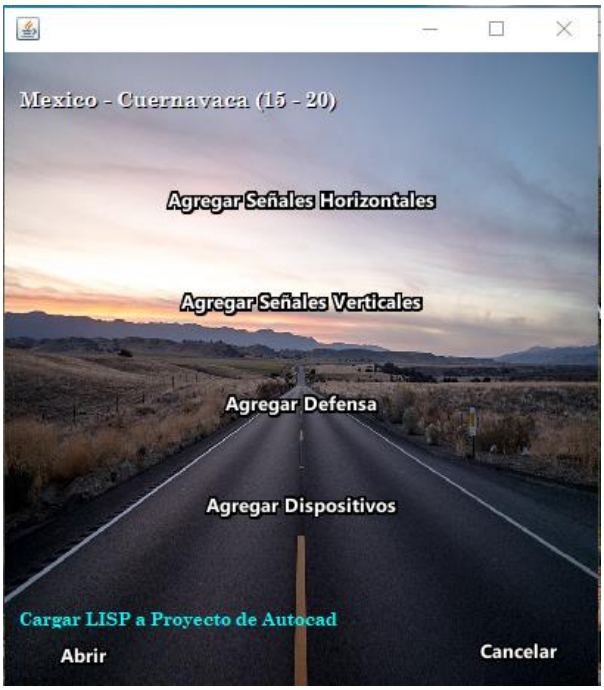


Figura 6.16. Aplicación: Elección de datos

❖ Cargar Proyecto

Si el usuario elige Cargar Proyecto aquí se mostrar una ventana con una tabla que mostrara todos los proyectos que se encuentren guardados en la base datos, el usuario puede elegir qué proyecto abrir para ello debe seleccionar el dato y des pues en Aceptar.

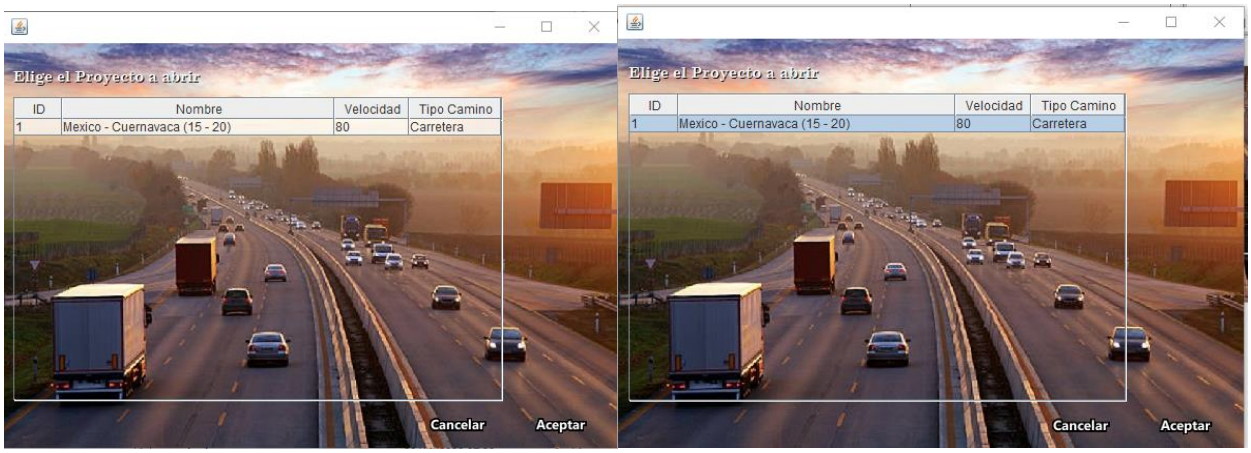


Figura 6.17. Aplicación: Cargar Proyecto

Entonces se abre la ventana donde el usuario puede elegir que datos agregar al proyecto, si Señales Horizontales, señales Verticales, Defensa o Dispositivos, de igual forma hay un apartado para abrir

un proyecto de AutoCAD, al cual se le cargaran los LIPS para poder hacer las diferentes tareas del programa.

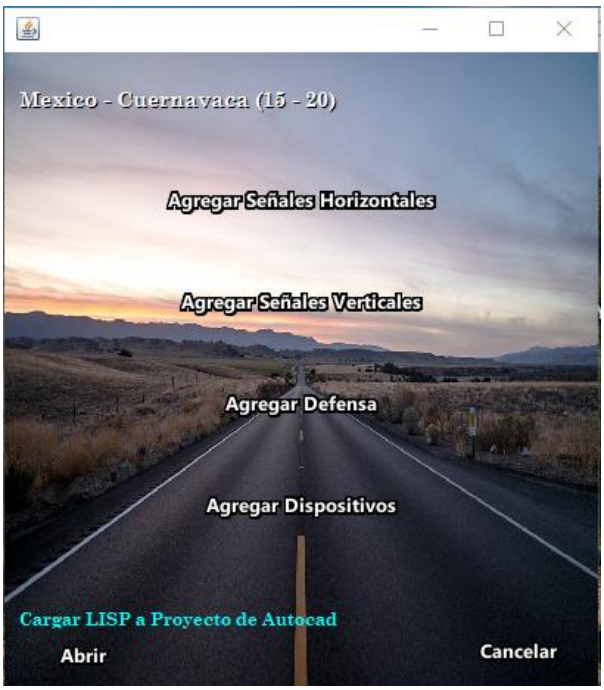


Figura 6.18.

- *Ventana Elección de Señalamiento*

  - Abrir

En esta ventana, si es la primera vez que se inicia un Proyecto entonces se recomienda primero usar el botón “Abrir” para poder cargar los LISP al Proyecto de AutoCAD del proyecto que se está trabajando, se abrirá un explorador de archivos donde el usuario buscar el archivo DWG del proyecto que se va a trabajar.

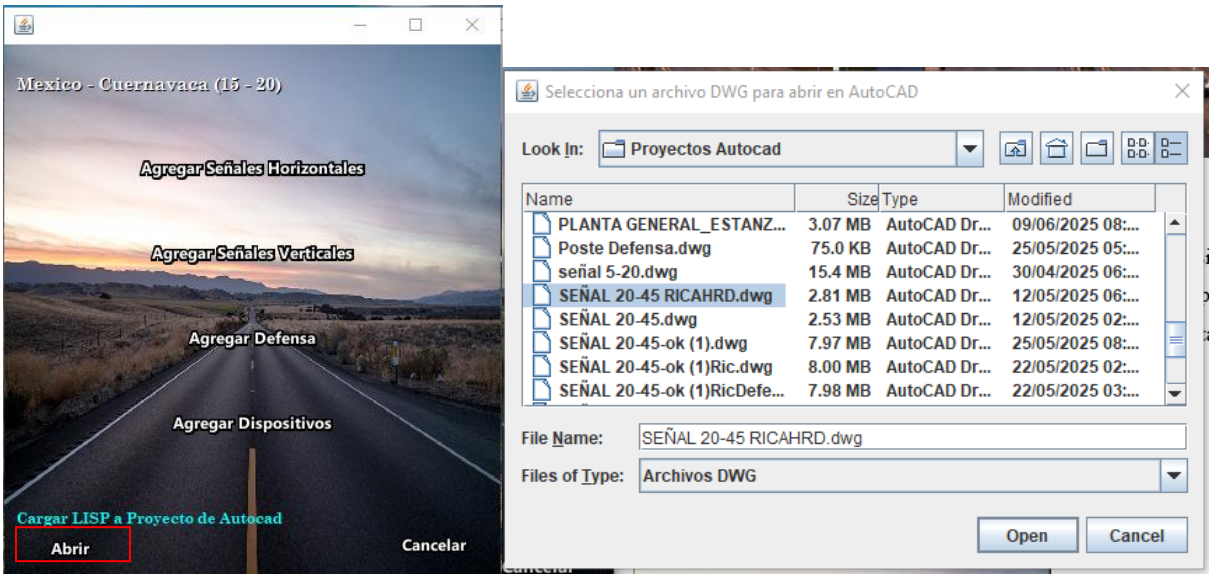


Figura 6.19. Aplicación: Abrir Archivo

Una vez que se selecciona y se le dé en Open, entonces se abrirá automáticamente el AutoCAD que se encuentre instalado en la computadora, y al terminar en la consola del AutoCAD se mostrara que los LIPS fueron cargados en este proyecto.

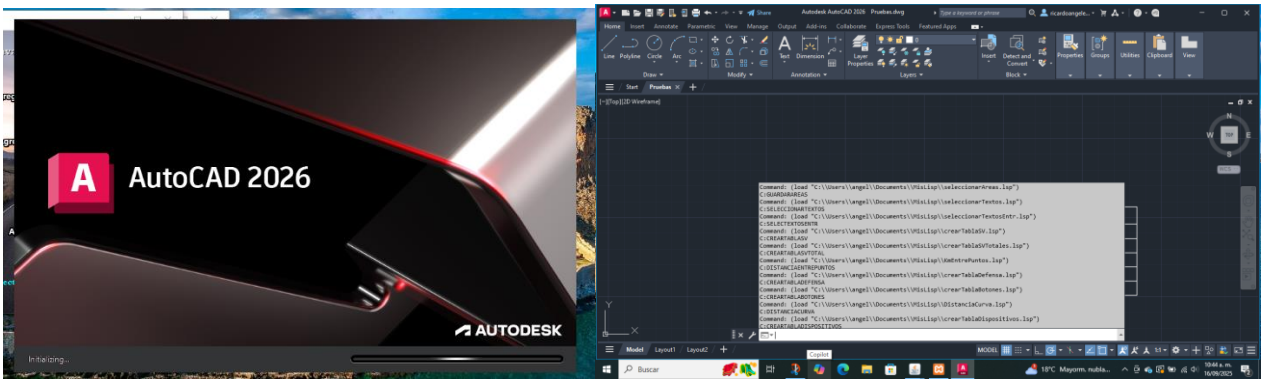


Figura 6.20. Aplicación: Cargar LIPS a AutoCAD

Entonces una vez cargado el proyecto entonces el usuario puede regresar a la ventana de Elección de Señalamiento.

❖ Agregar Señales Horizontales

Si el usuario elige “Agregar Señales Horizontales” entonces se abrirá una ventana donde se encuentran Botones para seleccionar la marca del señalamiento Horizontal a agregar.

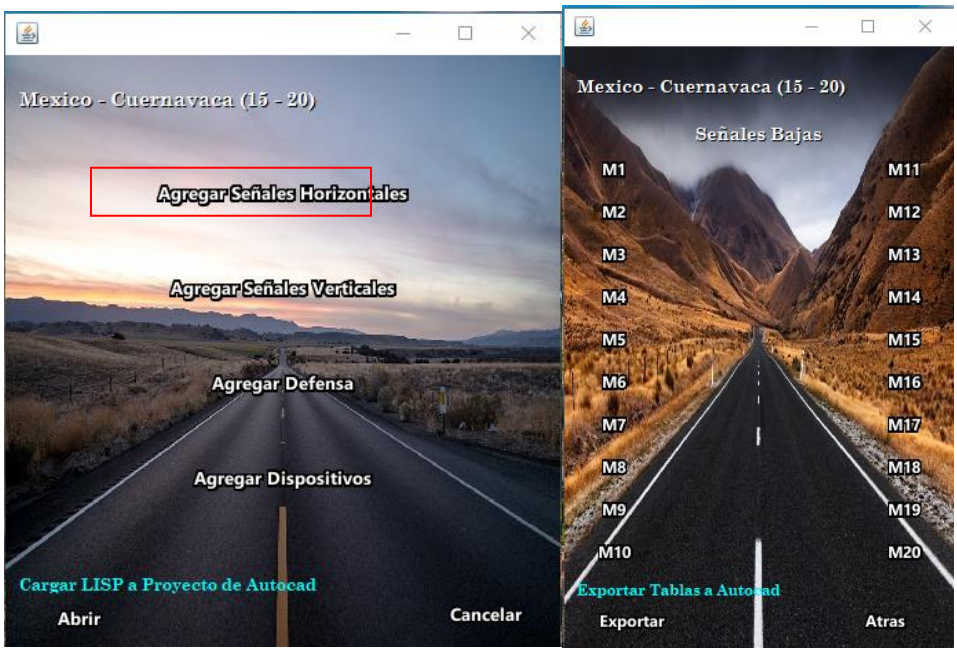


Figura 6.21. Aplicación: Señalamiento Horizontal (Elección)

En esta venta puedes seleccionar que marcas se quieren agregar al proyecto de la M-1 a la M-20, para cualquier caso solo presiona el botón correspondiente a la marca que se quiere agregar.

- **M1**

Si se selecciona “M1” entonces se abrirá una ventana correspondiente a las marcas M-1 (Raya Separadora de Sentidos de Circulación) aquí se podrá agregar la submarca de M-1.1 a M-1.4, para ello hay un botón correspondiente a cada uno.

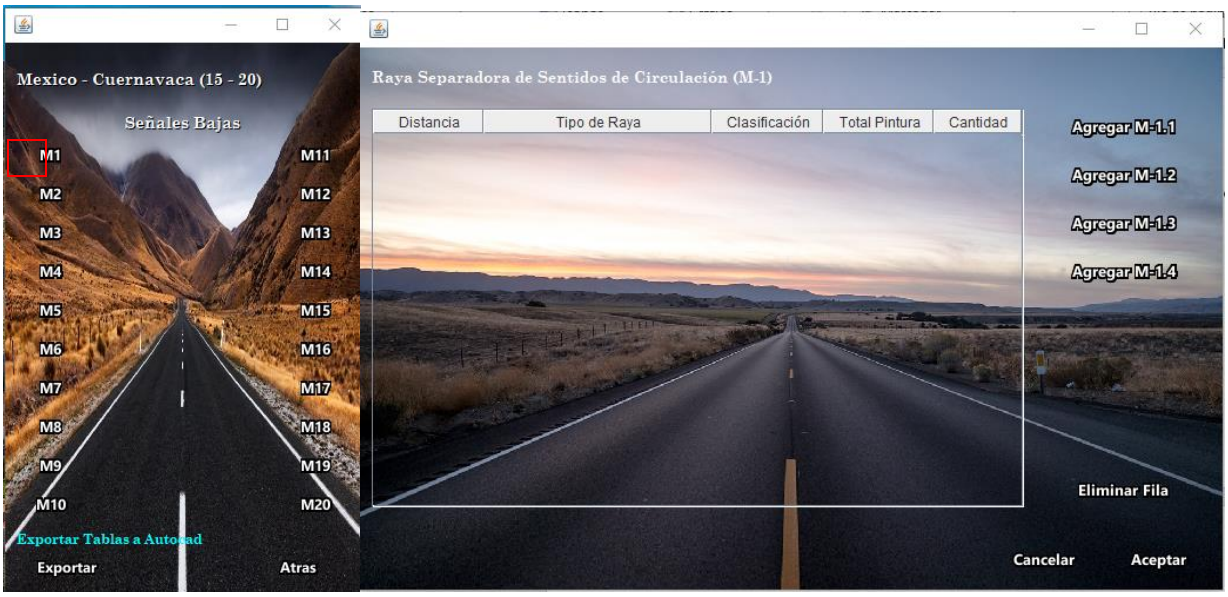


Figura 6.22. Aplicación: Marca M-1

Si seleccionamos Agregar M-1.1, M-1.2 o M-1.3 el mensaje que aparecerá será casi el mismo, solo cambiara la parte que dice:

"Escriba el comando 'GuardarLongitudes' y seleccione todas las líneas que sean …"

Aquí el mensaje cambia según la marca seleccionada. Lo importante es el comando GuardarLongitudes, que para las tres marcas el comando es el mismo.

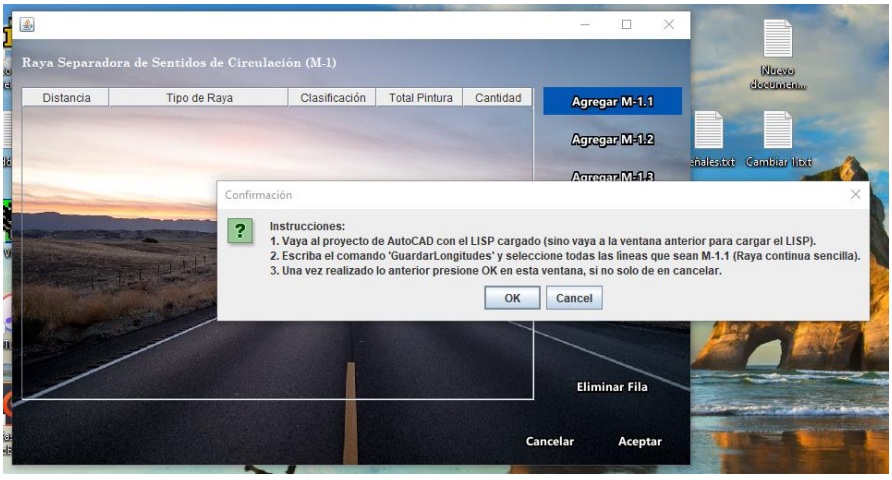


Figura 6.23. Aplicación: Agregar Marca M-1

Entonces cuando se presiona un botón para agregar alguna marca y aparece el mensaje de confirmación, el usuario tiene que ir a AutoCAD para usar el comando solicitado.

IMPOTANTE: NO DEBE CERRAR EL MENSAJE DE CONFIRMACIÓN, HASTA QUE PRIMERO SE REALICEN LAS INTRUCCIONES QUE SE MUESTRAN EN EL MENSAJE.

Por lo tanto, sin confirmar el mensaje, el usuario se dirige a AutoCAD y escribe el comando GuardarLongitudes y selecciona las líneas correspondientes al botón que el usuario haya presionado.

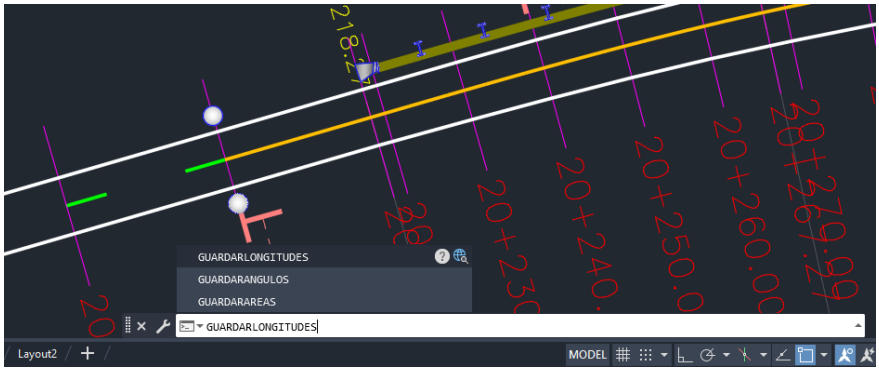


Figura 6.24. Aplicación: Guardar Longitudes AutoCAD

Se selecciona con el clic izquierdo TODAS las líneas correspondientes a la marca del botón seleccionado, y con el clic derecho, una vez terminado de seleccionar, se confirma que ya se terminó el proceso y entonces aparece el siguiente mensaje:

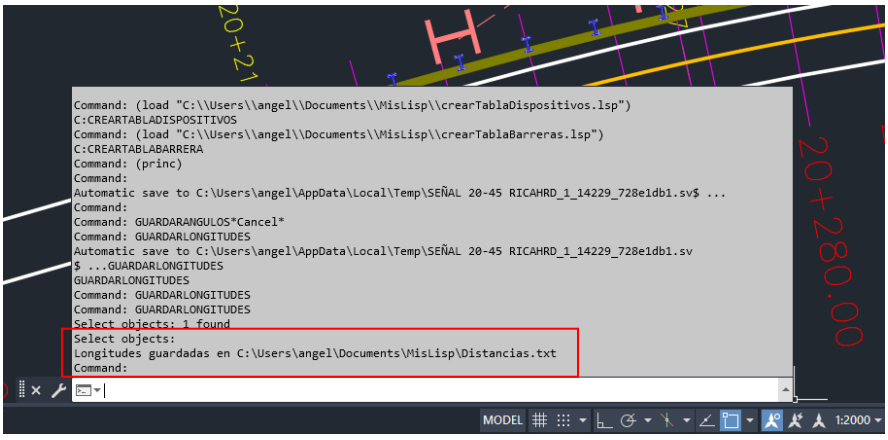


Figura 6.25. Aplicación: Confirmación AutoCAD

Una vez hecho lo anterior, ahora si el usuario puede cerrar el mensaje de confirmación en la aplicación entonces se agregará en la tabla una fila con los datos de la marca agregada, en la columna de Distancia se muestra la distancia total de las líneas seleccionadas y en la columna de Total Pintura

aparece el cálculo correspondiente de la pintura. Por ejemplo, para M-1.2 la línea es discontinua, por ello a la distancia se le tiene que quitar los espacios que hay entre las rayas que si son pintura.

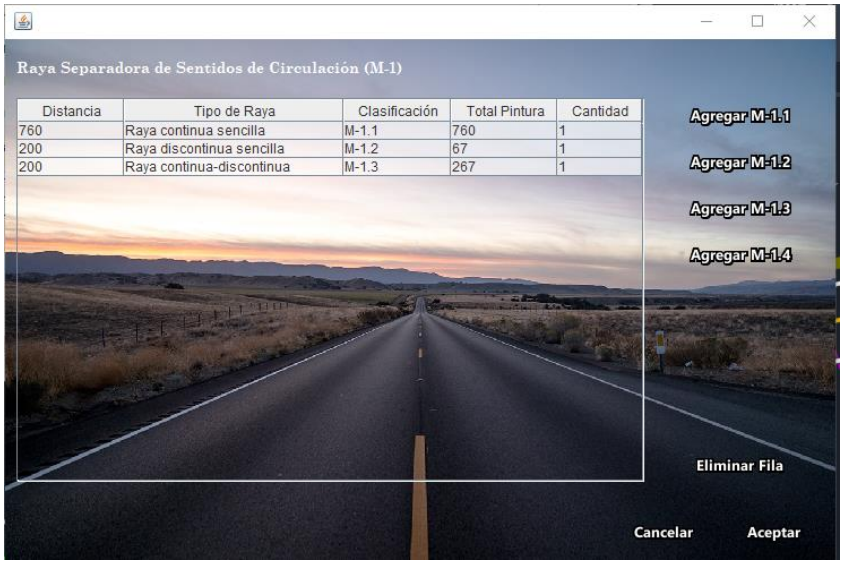


Figura 6.26. Aplicación: Agregando marcas M-1.1, M-1.2 y M-1.3

Para el caso de M-1.4 de igual forma usamos el comando de GuardarLongitudes y que solo se seleccione una de las dos líneas de M-1.4, solo que en el mensaje de confirmación al usuario se le solicita que también indique cual es la distancia de separación en cm de las líneas M-1.4.

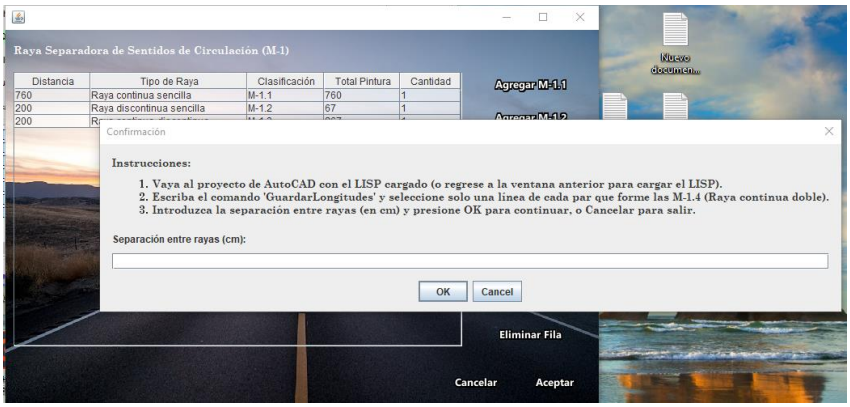


Figura 6.27. Aplicación: Agregando marca M-1.4

Si es mayor a 50 cm. la separación entre líneas se agregará al cálculo de pintura rayas que estarán a cada 2 m. entre las dos líneas de pintura, entonces si la separación es menor a 50 cm. entonces el cálculo solo será de dos veces la distancia de la línea seleccionada en AutoCAD.

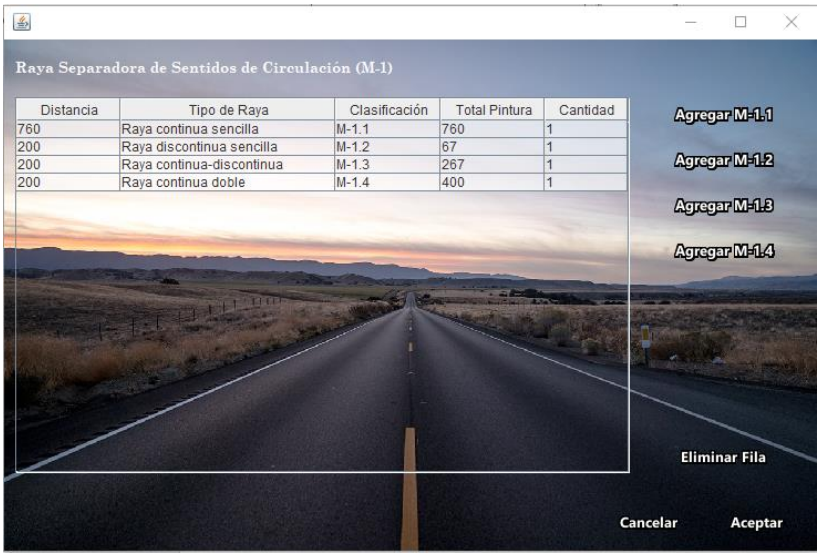


Figura 6.28. Aplicación: Marca M-1.4 Agregada.

Una vez terminado de agregar las marcas que se requieran, se presiona el botón de Aceptar para guardar los datos en la base de datos, para ello aparece un mensaje de confirmación y después de aceptar aparece un mensaje confirmación de si se guardaron los datos de forma correcta o no, en este mensaje hay una fila que dice "Filas con error", esto se refiere a que en la base datos ya tiene datos iguales guardados, para esos casos se tiene la columna de Cantidad, la cual en caso de que se requiera, se puede cambiar la cantidad por 2 o más, y el cálculo de pintura se multiplicará por la cantidad seleccionada.

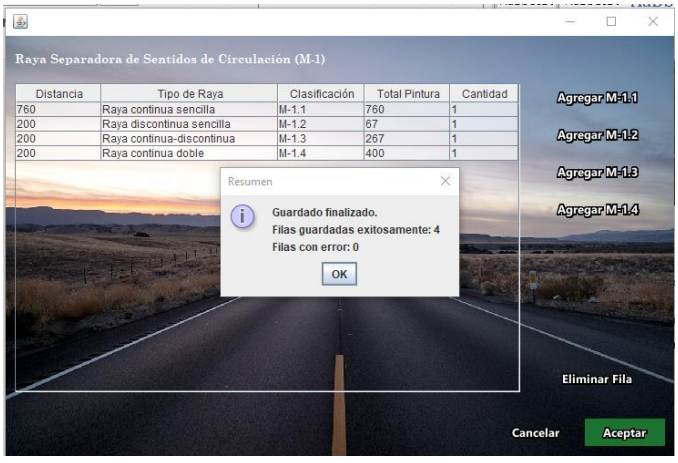


Figura 6.29. Aplicación: Guardado en Base de datos

Para el botón de Eliminar solo se debe seleccionar la fila a eliminar y después presionar el botón, entonces aparecerá un mensaje de confirmación y si se confirma aparece un mensaje con los datos

que se eliminaron. NOTA: Solo sirve para cuando los datos ya se guardaron en la base de datos, si se llega a usar antes de guardar en la base, solo marcar un error de que no se pudo eliminar la fila seleccionada

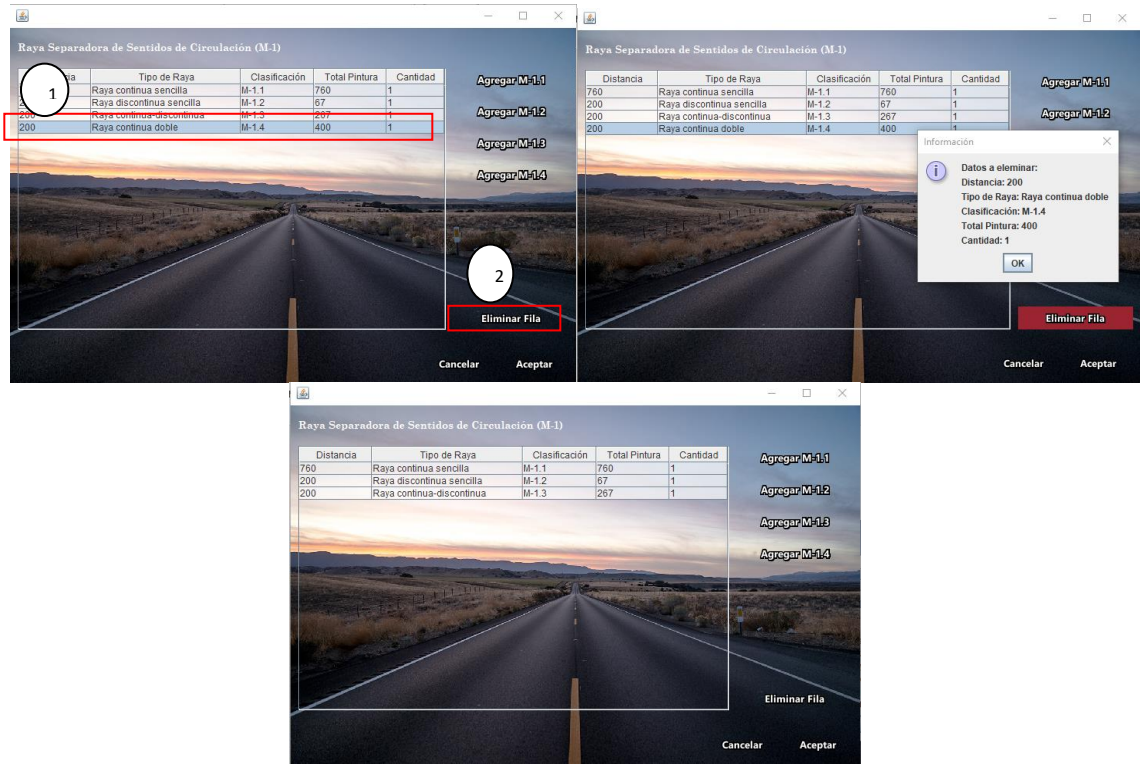


Figura 6.30. Aplicación: Eliminando una Fila

- **M2**

Para M2 se tiene casi lo mismo que para M1 solo que aquí se tienen tres botones en lugar de cuatro, pero la funcionalidad de estos es parecida, donde al presionarlos aparece el mensaje de confirmación, de igual forma NO se debe confirmar este mensaje hasta después hacer los paso en AutoCAD, y se pide que use el comando GuardarLongitudes para seleccionar las líneas del botón seleccionada correspondiente a las marcas M-2 (Raya separadora de carriles).

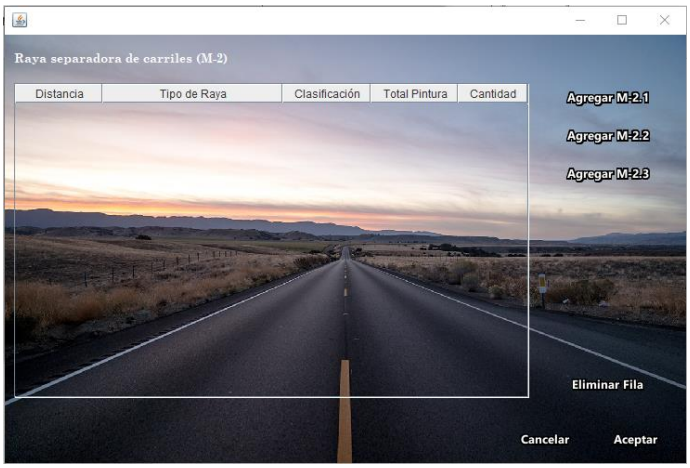


Figura 6.31. Aplicación: M-2

Entonces los botones funcionan casi igual al momento de agregar, después de hacer los pasos en AutoCAD se confirma el mensaje en la aplicación y se agregarán las marcas del botón seleccionado, y después se presiona el botón de aceptar para que así se pueda guardar en la base datos.

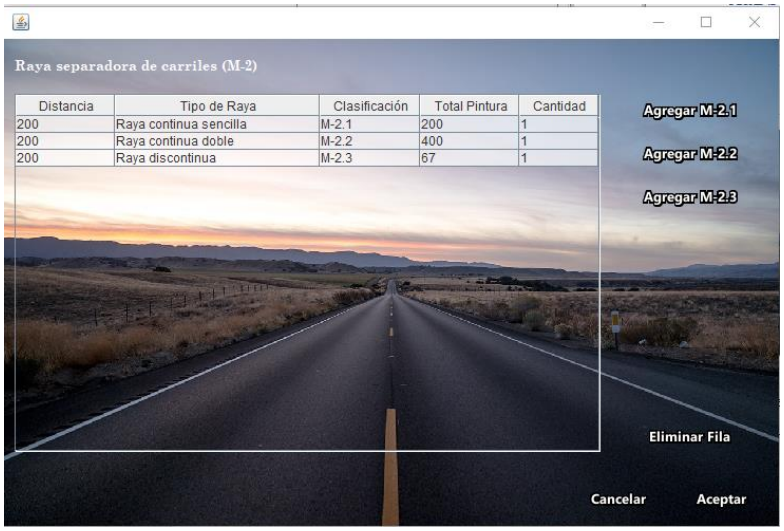


Figura 6.32. Aplicación: Agregando marcas M-2

- **M3**

Para M3 se tienen dos botones, con la funcionalidad igual a las anteriores, donde al presionarlos aparece el mensaje de confirmación, de igual forma NO se debe confirmar este mensaje hasta después hacer los pasos en AutoCAD, y se pide que use el comando GuardarLongitudes para seleccionar las líneas del botón seleccionada correspondiente a las marcas M-3 (Raya en la orilla del arroyo vial).

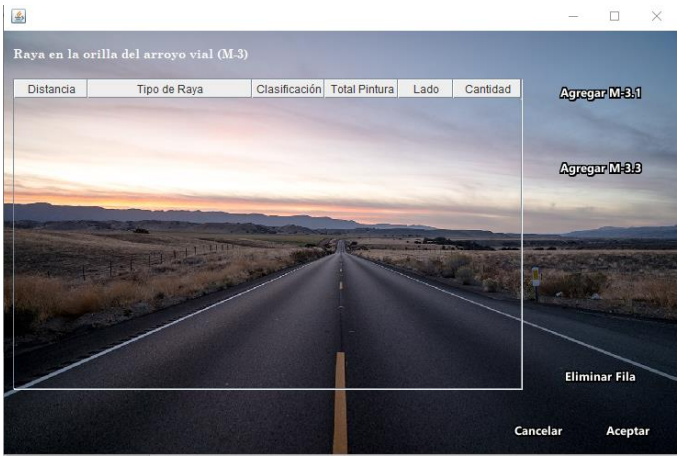


Figura 6.33. Aplicación: M-3

Entonces los botones funcionan casi igual al momento de agregar, después de hacer los pasos en AutoCAD se confirma el mensaje en la aplicación y se agregarán las marcas del botón seleccionado, y después se presiona el botón de aceptar para que así se pueda guardar en la base datos.

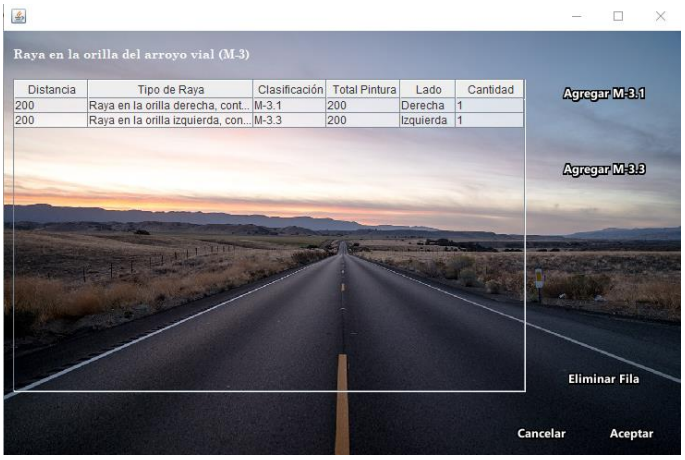


Figura 6.34. Aplicación: Agregando marcas M-3

- **M4**

Para M4 se tienen tres botones, con la funcionalidad igual a las anteriores, donde al presionarlos aparece el mensaje de confirmación, de igual forma NO se debe confirmar este mensaje hasta después hacer los pasos en AutoCAD, y se pide que use el comando GuardarLongitudes para seleccionar las líneas del botón seleccionada correspondiente a las marcas M-4 (Raya en la orilla del arroyo vial).

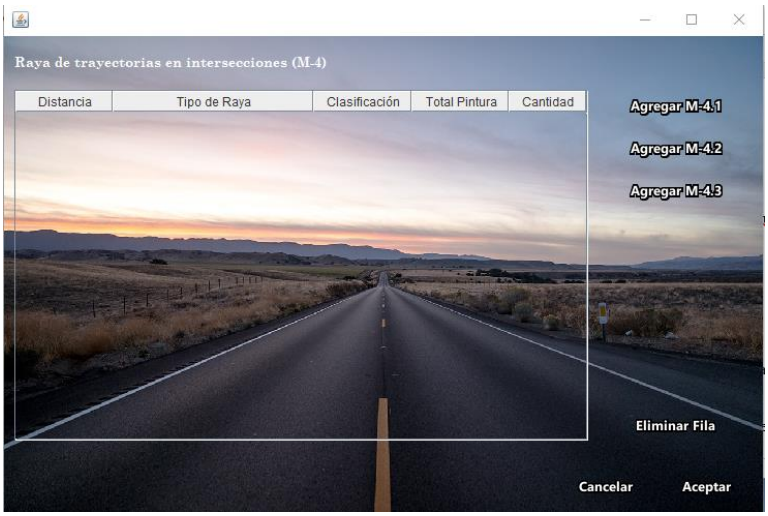


Figura 6.35. Aplicación: M-4

Aquí hay un caso especial en el caso del botón para agregar M-4.2, esta marca solo se puede agregar cuando el proyecto es de tipo Calle, entonces en este ejemplo el tipo de camino es carretera, y aparece un mensaje de advertencia de que M-4.2 se usa en Calles.

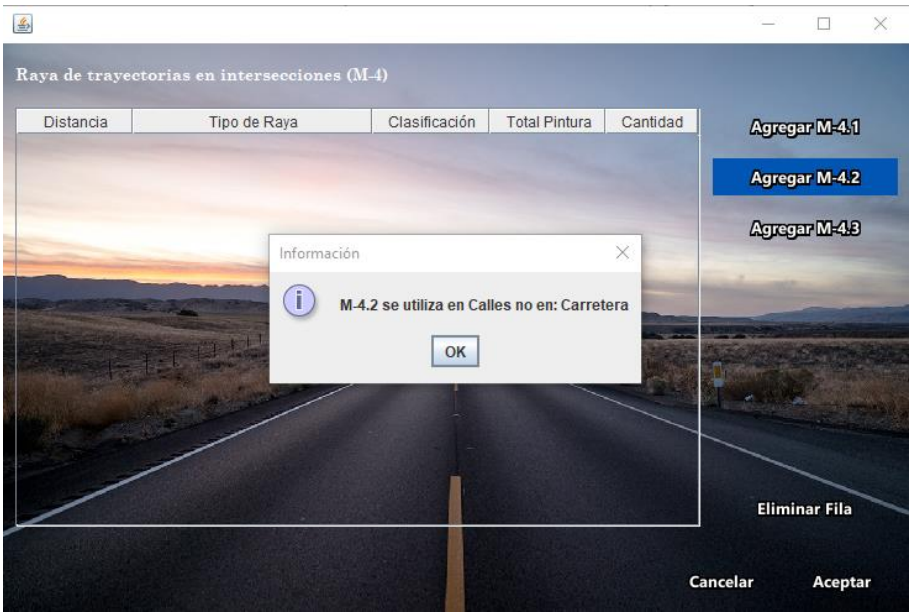


Figura 6.36. Aplicación: Advertencia M-4.2

Entonces los otros botones funcionan casi igual al momento de agregar, después de hacer los pasos en AutoCAD se confirma el mensaje en la aplicación y se agregarán las marcas del botón

seleccionado, y después se presiona el botón de aceptar para que así se pueda guardar en la base datos.

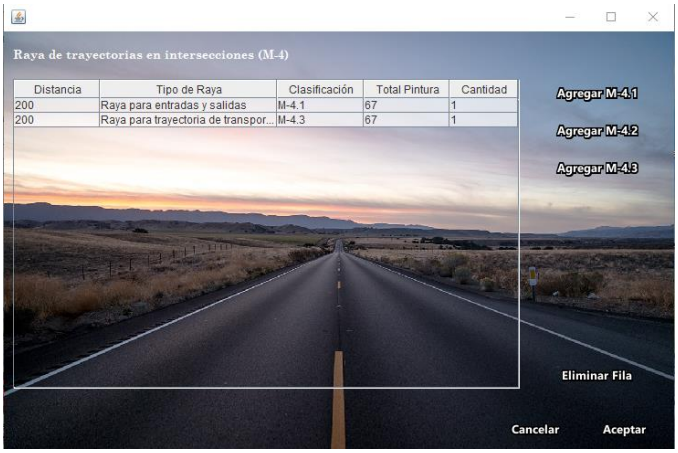


Figura 6.37. Aplicación: Agregando marcas M-4

- **M5**

En el caso de M5 se tiene un solo botón, pero este permite procesar dos tipos de marcas: M-5.1 (Rayas que limitan la zona neutral) y M-5.2 (Rayas en la zona neutral). La dinámica de uso sigue la misma lógica de las anteriores: al presionar el botón aparece un mensaje de confirmación con las instrucciones, y NO se debe confirmar este mensaje hasta realizar primero los pasos en AutoCAD.

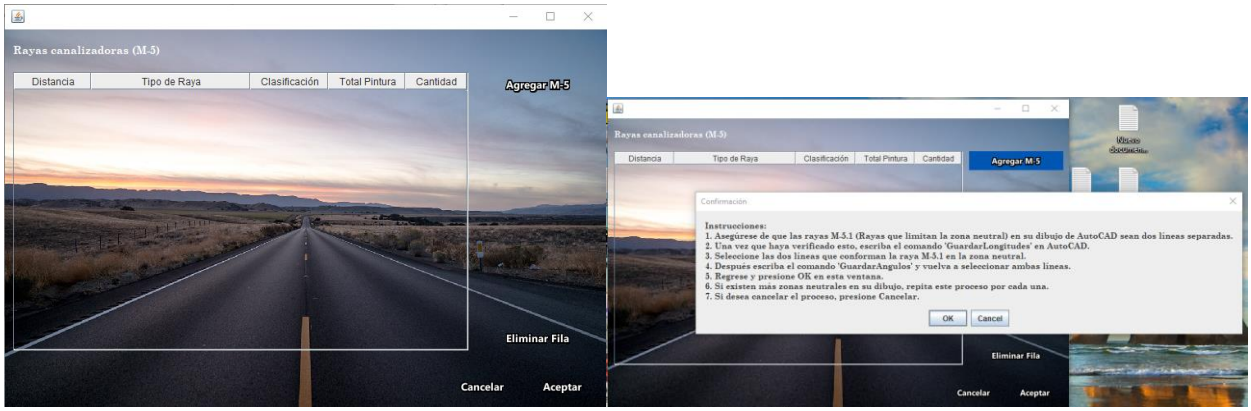


Figura 6.38. Aplicación: M-5

Para M-5.1 se solicita al usuario que, en AutoCAD, seleccione las dos líneas que delimitan la zona neutral, usando el comando GuardarLongitudes. Posteriormente, se debe ejecutar el comando GuardarAngulos para registrar el ángulo entre estas líneas, es importan que ambas líneas estén conectadas o que se forme un ángulo con ellas. Una vez realizado este procedimiento, se confirma

el mensaje en la aplicación, y los datos se agregan automáticamente a la tabla, incluyendo la distancia total y la clasificación de la raya.

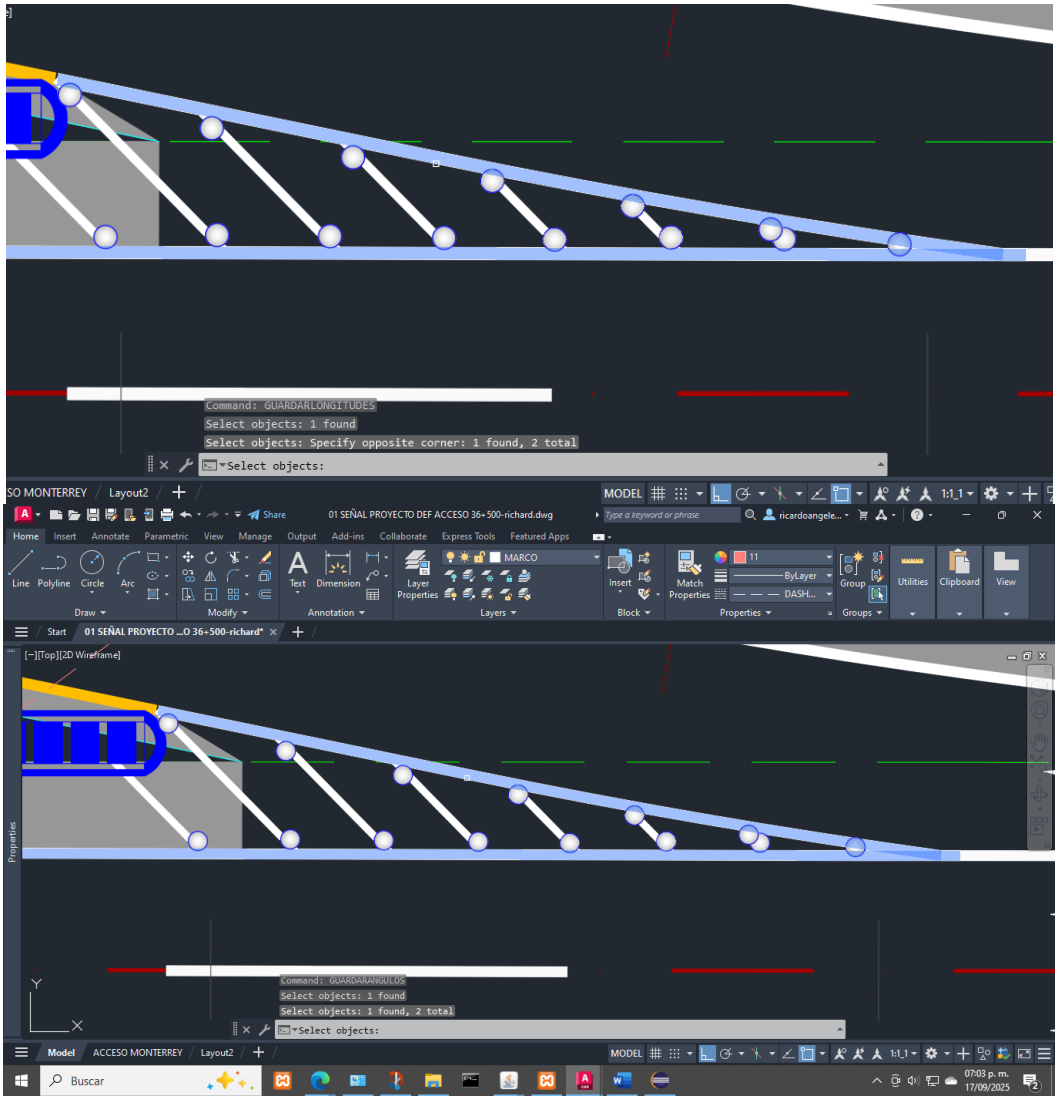


Figura 6.39. Aplicación: Usando GuardarLongitudes y GuardarAngulos

En el caso de M-5.2, el sistema calcula automáticamente el área triangular formada por las líneas seleccionadas y, a partir de esta información, determina la cantidad de rayas canalizadoras que deben colocarse en la zona neutral. El cálculo también considera el tipo de camino (por ejemplo, en vías ciclistas la separación entre rayas es menor). Una vez agregados los valores en la tabla, se confirma la información presionando el botón de aceptar, de manera que quede registrada en la base de datos.

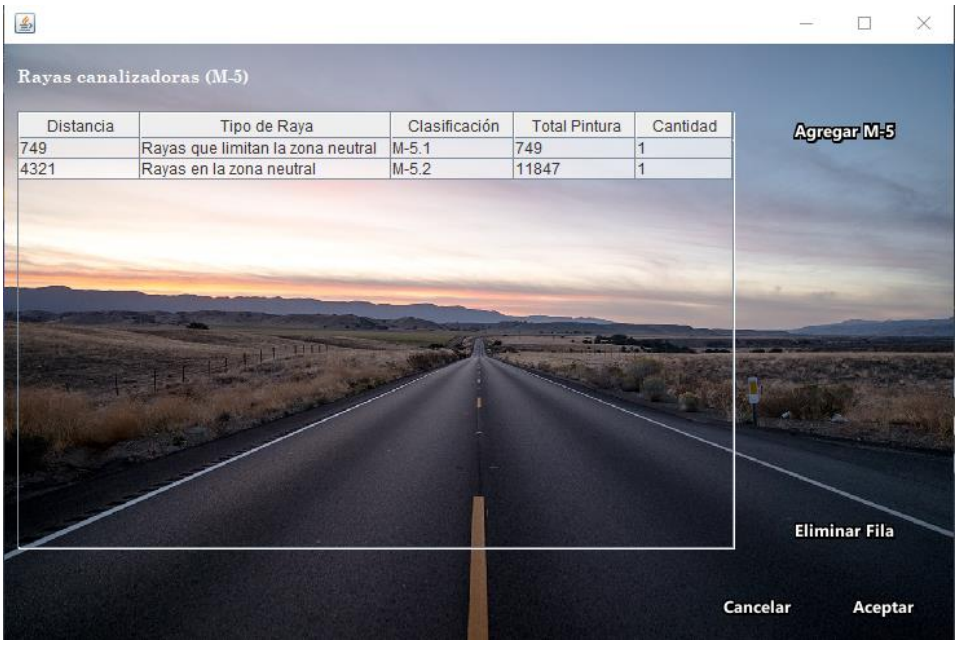


Figura 6.40. Aplicación: Agregando marcas M-5.2

- **M6**

En el caso de M6 se tiene un solo botón para agregar este tipo de marca. Su funcionamiento es muy similar a los anteriores: al presionar el botón aparece un mensaje de confirmación con las instrucciones, y NO se debe confirmar hasta realizar primero los pasos en AutoCAD.

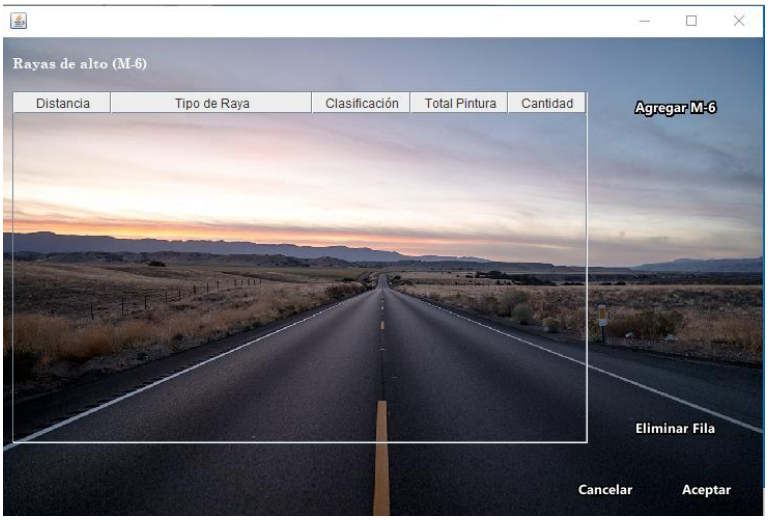


Figura 6.41. Aplicación: M-6

De igual forma, en AutoCAD se debe utilizar el comando GuardarLongitudes y seleccionar todas las líneas correspondientes a las rayas de alto (M-6). Una vez que la selección se ha completado, se

regresa a la aplicación y se confirma el mensaje. En ese momento, los valores de longitud total se registran en la tabla junto con su clasificación y el cálculo de pintura requerido. Y después el usuario debe presionar el botón de aceptar en la aplicación para que la información quede guardada en la base de datos.

Figura 6.42. Aplicación: Agregando marcas M-6

- **M7**

Para M7 se cuenta con un solo botón, cuyo propósito es agregar las marcas M-7 (rayas para cruce de peatones). A diferencia de otras marcas, aquí el usuario debe introducir manualmente dos datos: la longitud de la raya y el ancho del camino. Al presionar el botón, se despliega un mensaje de confirmación con las instrucciones y un formulario para capturar estos valores.

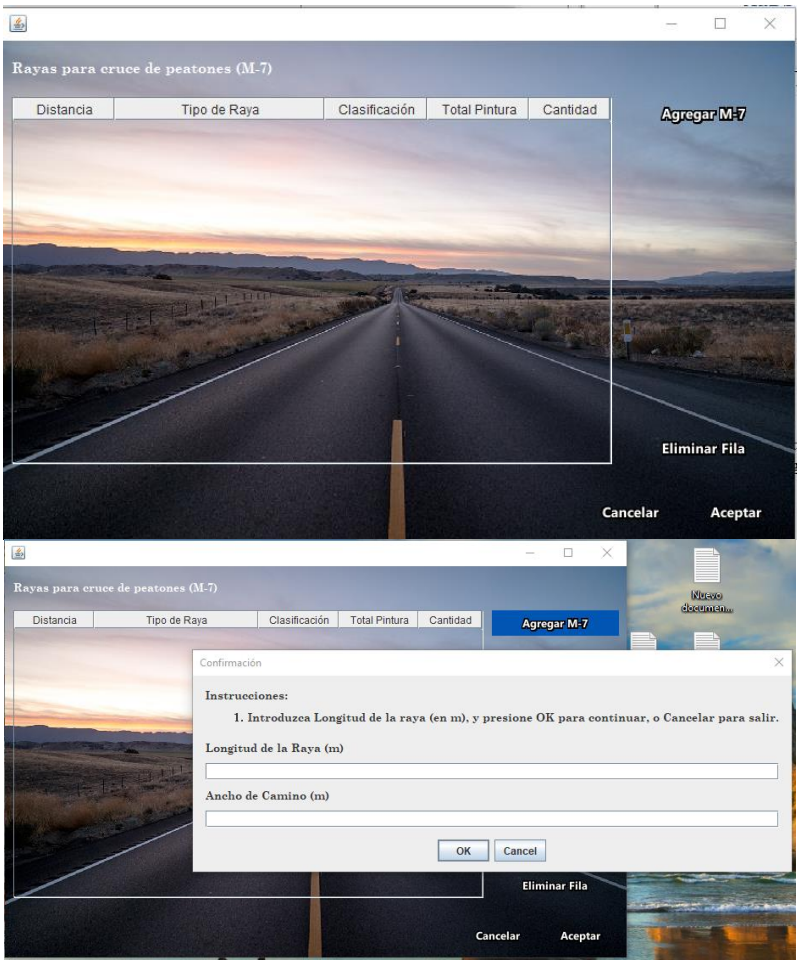


Figura 6.43. Aplicación: M-7

Una vez introducida la información y confirmada en la ventana, la aplicación realiza los cálculos correspondientes: primero determina el área total del cruce y posteriormente calcula la cantidad de pintura necesaria en función de la separación entre franjas. Estos resultados se registran automáticamente en la tabla, junto con la clasificación de la raya. Y después el usuario debe presionar el botón de aceptar para que los datos queden guardados en la base de datos.

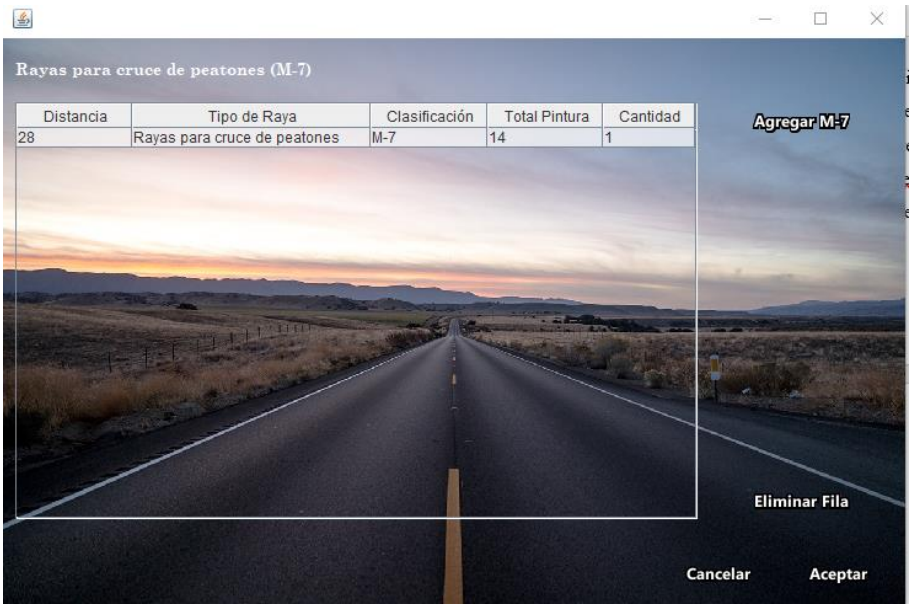


Figura 6.44. Aplicación: Agregando marcas M-7

- **M8**

Para M8 se dispone igualmente de un solo botón para agregar las marcas M-8 (Rayas de cruce de ferrocarril). La lógica de funcionamiento es similar a la de otras marcas: al presionar el botón, aparece un mensaje de confirmación con las instrucciones, y NO se debe confirmar hasta haber realizado primero los pasos en AutoCAD.

Figura 6.45. Aplicación: M-8

En AutoCAD, el usuario debe ejecutar el comando GuardarLongitudes y seleccionar todas las líneas correspondientes a M-8. Una vez realizado el procedimiento, se regresa a la aplicación, se confirma el mensaje y los datos se agregan a la tabla, incluyendo la clasificación y el cálculo de pintura. Y después se presiona el botón de aceptar para que la información quede registrada en la base de datos.

Figura 6.46. Aplicación: Agregando marcas M-8

- **M9**

En el caso de M9 se tiene un botón que permite agregar las marcas M-9 (rayas con espaciamiento logarítmico). Al presionarlo, se despliega una ventana de confirmación con instrucciones y campos para ingresar la cantidad de líneas logarítmicas y el ancho del camino.

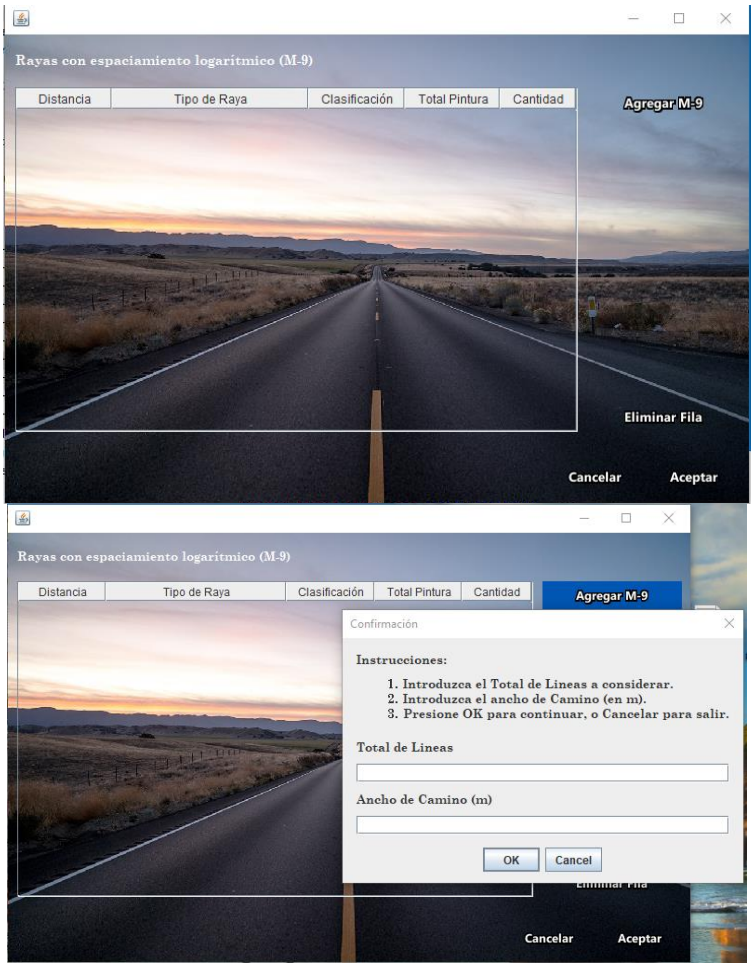


Figura 6.47. Aplicación: M-9

Con estos valores, la aplicación calcula automáticamente la cantidad de pintura requerida multiplicando el total de líneas por el ancho ingresado. Los resultados se registran en la tabla, junto con la clasificación de la raya. Y después se presiona el botón de aceptar para que la información quede registrada en la base de datos.

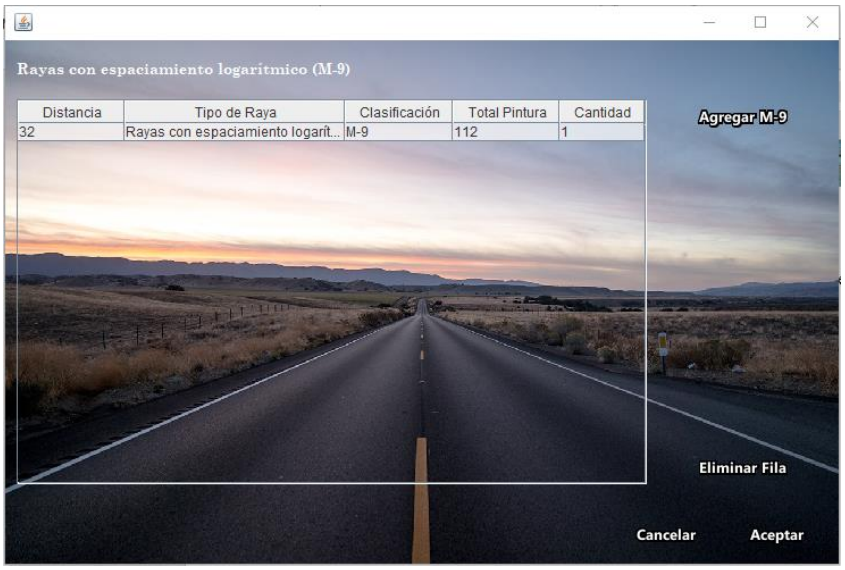


Figura 6.48. Aplicación: Agregando marcas M-9

- **M10**

En el caso de M10 se tienen tres botones, cada uno correspondiente a un subtipo diferente: M-10.1 (Marcas para estacionamiento de vehículos motorizados), M-10.2 (Marcas para estacionamiento en zonas de pago) y M-10.3 (Marcas para estacionamiento de servicios especiales). En todos los casos, al presionar el botón aparece un mensaje de confirmación con instrucciones, y NO se debe confirmar hasta haber realizado primero los pasos en AutoCAD.

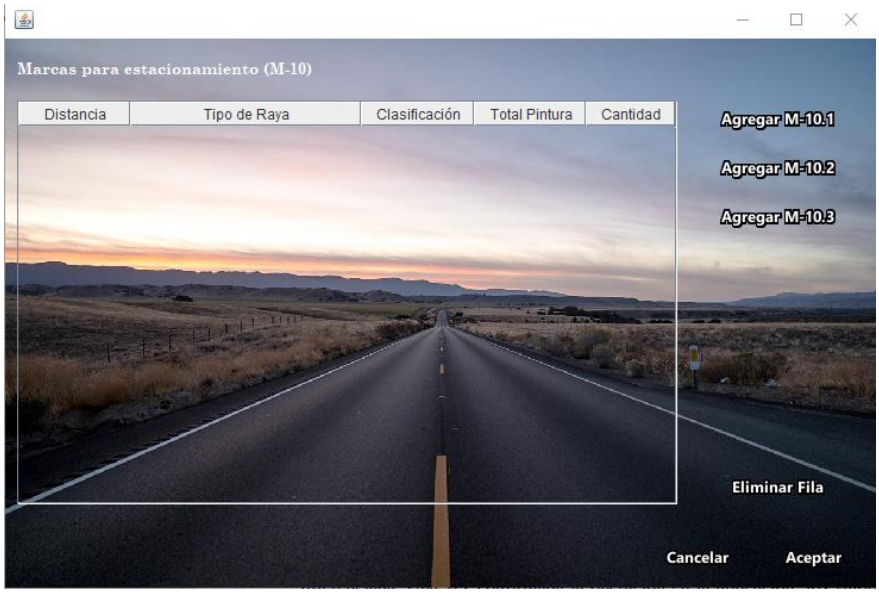


Figura 6.49. Aplicación: M-10

Para M-10.1, el procedimiento consiste en ir a AutoCAD, ejecutar el comando *GuardarLongitudes* y seleccionar las líneas continuas y punteadas que representen las marcas de estacionamiento de vehículos motorizados, se pide que primero se seleccionen todas las continuas y después todas las punteadas. Después en la aplicación solicita al usuario indicar cuántas de estas líneas son continuas, lo que permite separar las longitudes por tipo y realizar el cálculo de pintura de manera diferenciada. Una vez confirmada la operación en la aplicación, los valores se agregan a la tabla junto con la clasificación correspondiente.

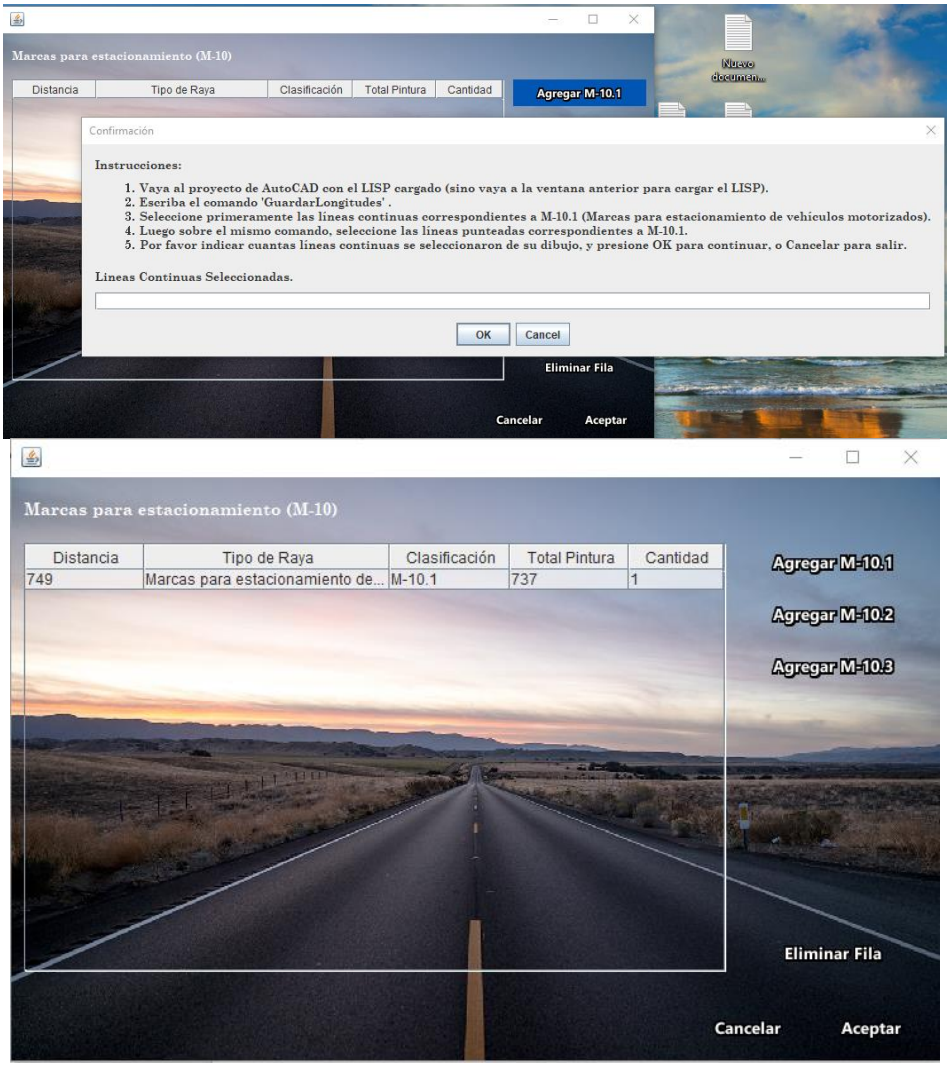


Figura 6.50. Aplicación: Agregando marcas M-10.1

En el caso de M-10.2, la lógica es similar, solo que corresponde a las marcas en zonas de pago. Nuevamente, el usuario debe seleccionar en AutoCAD las líneas continuas y punteadas con el

comando *GuardarLongitudes*, indicar en la aplicación el número de líneas continúas seleccionadas y confirmar. El sistema realiza el mismo procesamiento, calculando automáticamente la cantidad de pintura y registrando los resultados en la tabla.

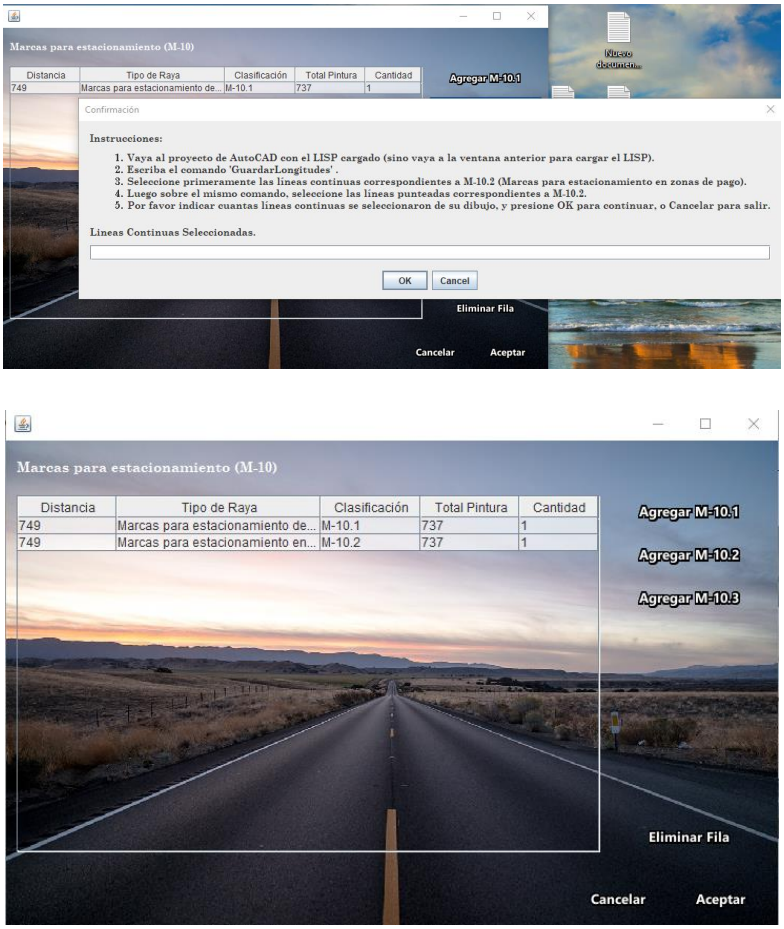


Figura 6.51. Aplicación: Agregando marcas M-10.2

Finalmente, para M-10.3 se solicitan datos adicionales además de las longitudes: el largo del cajón de estacionamiento y el ancho de la franja de circulación. Con estos valores, la aplicación calcula un área complementaria destinada al paso peatonal, la cual se suma al cálculo de pintura base. De esta forma se obtiene un resultado más preciso que también se registra en la tabla con la clasificación respectiva.

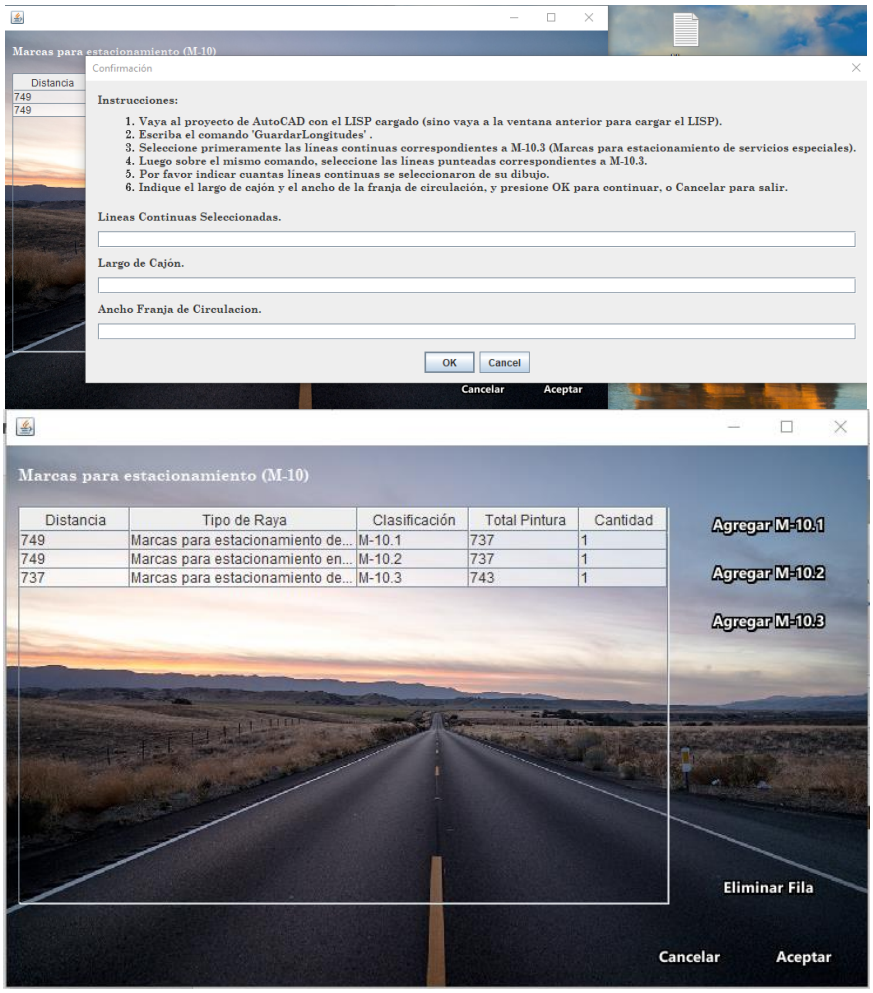


Figura 6.52. Aplicación: Agregando marcas M-10.3

En todos los subtipos de M-10, una vez agregados los resultados en la tabla, el usuario debe presionar el botón de aceptar para que la información quede guardada en la base de datos.

- **M11**

En el caso de M11 se utiliza un botón principal denominado “Agregar Flechas”, que permite trabajar con tres subtipos de clasificación: M-11.1 (Flechas y leyendas en carriles), M-11.2 (Flechas y leyendas para indicar un carril exclusivo) y M-11.3 (Para establecer lugares de parada). Al presionar el botón se despliega un cuadro con instrucciones, y además se habilita un campo para seleccionar el tipo de flecha que se desea agregar (Recta, Curva o Recta y Curva).

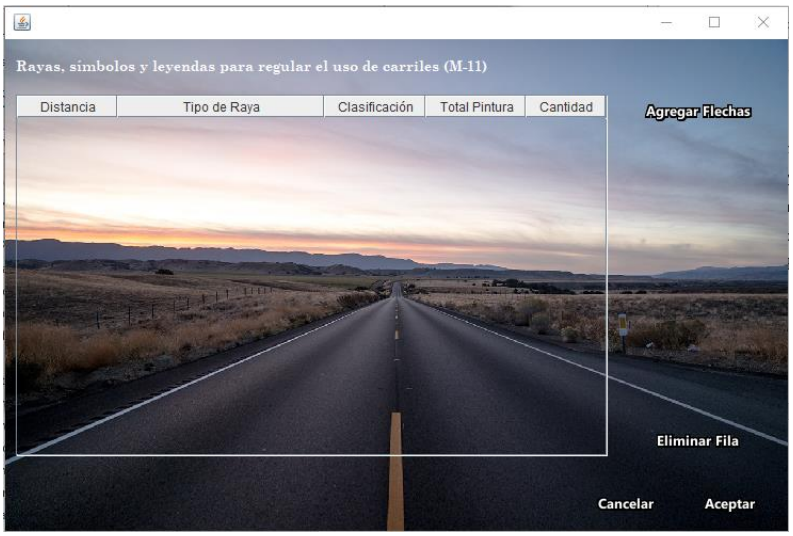


Figura 6.53. Aplicación: M-11

El procedimiento consiste en que el usuario indique el subtipo correspondiente. Dependiendo de la elección, el sistema calcula automáticamente la cantidad de pintura necesaria. Este cálculo varía según la velocidad del camino, de modo que para velocidades menores o iguales a 50 km/h se emplean dimensiones reducidas, mientras que para velocidades superiores se consideran medidas mayores.

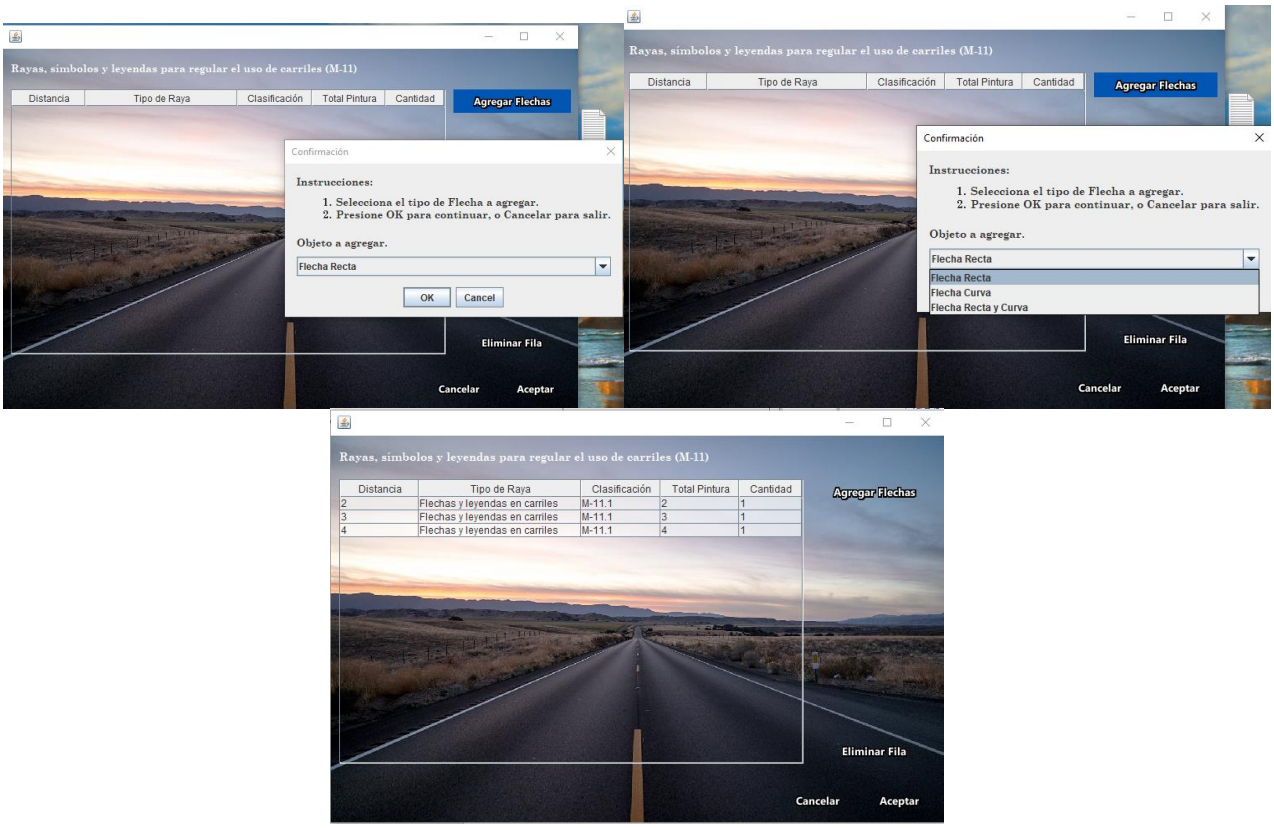


Figura 6.54. Aplicación: Agregando flechas M-11

Una vez confirmada la operación, los valores calculados se agregan a la tabla junto con la clasificación y el tipo de flecha seleccionado. Y después el usuario debe aceptar los cambios para que los datos queden guardados en la base de datos.

- **M12**

En el caso de M12 se tienen cuatro botones específicos, cada uno correspondiente a un subtipo distinto: M-12.1 (Para prohibición del estacionamiento), M-12.2 (Para delinear guarniciones), M-12.3 (Para prohibición de parar) y M-12.4 (Para indicar estacionamiento de servicios especiales). En todos los casos, al presionar el botón se muestra un cuadro con instrucciones que deben seguirse en AutoCAD, utilizando el comando *GuardarLongitudes* para seleccionar las líneas correspondientes.

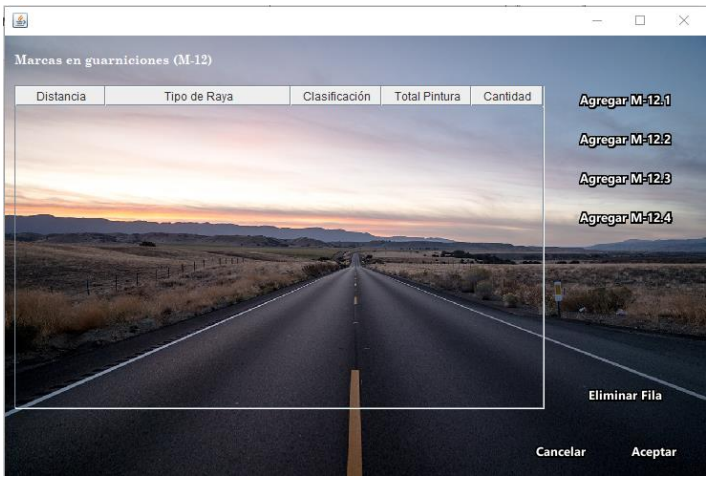


Figura 6.55. Aplicación: M-12

Para M-12.1 y M-12.3, el procedimiento es directo: el usuario selecciona las líneas continuas que según corresponda a cada caso, la cual suma la longitud y calcula la cantidad de pintura a registrar en la tabla.

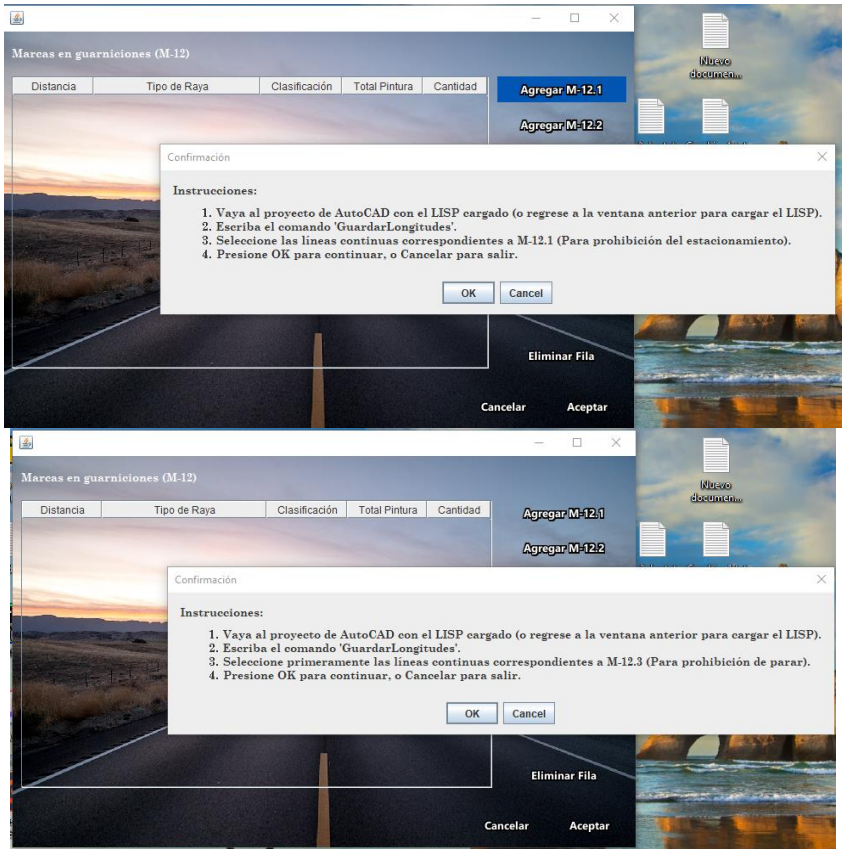


Figura 6.56. Aplicación: Agregando marcas M-12.1 y M-12.3

En el caso de M-12.2 y M-12.4, se requiere información adicional ya que se combinan líneas continuas y punteadas. El usuario debe indicar cuántas líneas continuas fueron seleccionadas en AutoCAD, para ello deben ser las primeras en seleccionarse, y la aplicación separa automáticamente ambas longitudes para calcular la pintura de manera diferenciada.

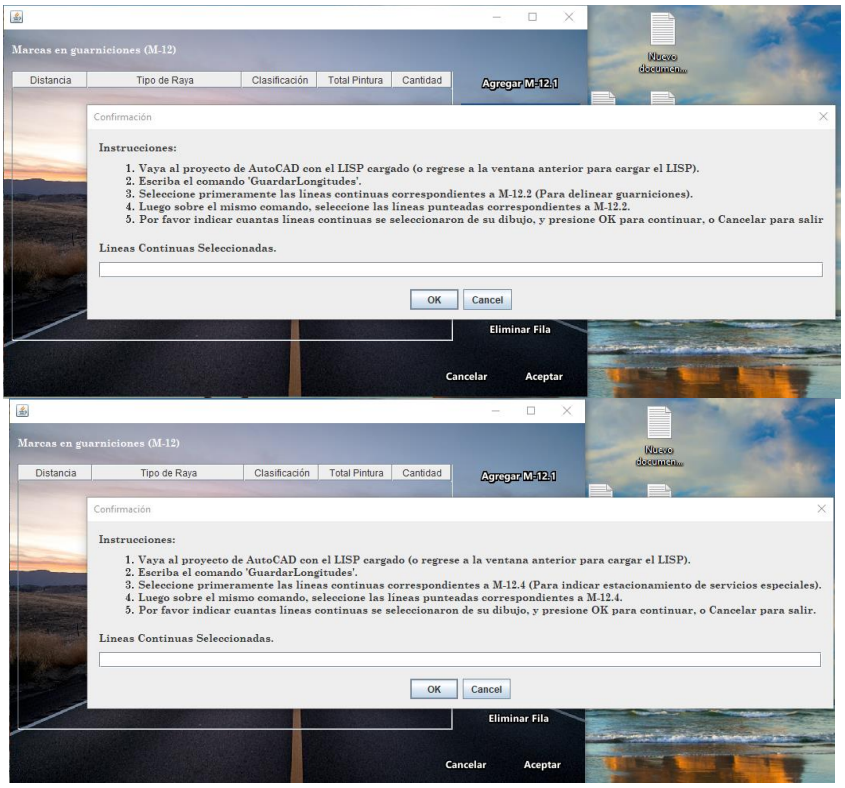


Figura 6.57. Aplicación: Agregando marcas M-12.2 y M-12.4

En todos los casos, después de agregar los resultados en la tabla, el usuario debe presionar el botón de aceptar para que la información quede almacenada en la base de datos.

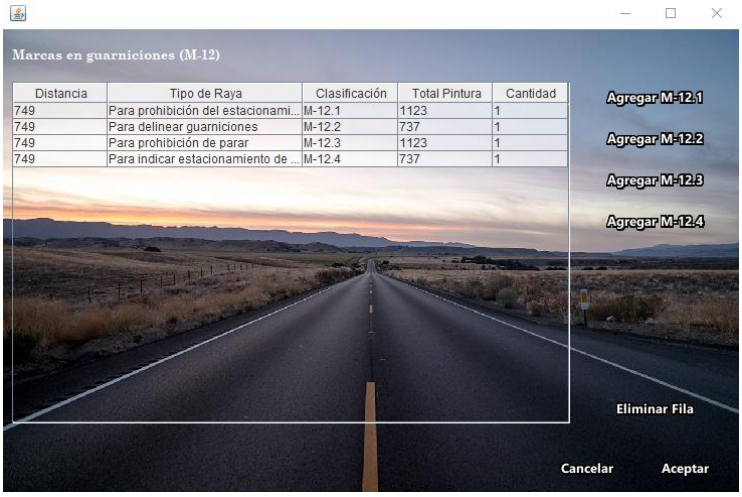


Figura 6.58. Aplicación: Agregando marcas M-12

- **M13**

En el caso de M13 se dispone de un único botón denominado “Agregar M-13.1”, correspondiente a las marcas M-13 (marcas en estructuras y objetos adyacentes a la superficie de rodadura).

Figura 6.59. Aplicación: M-13

Al presionar este botón, se despliega un mensaje de confirmación con las instrucciones y un formulario para capturar los siguientes datos:

- Ancho de columna.
- Altura libre de la estructura (Gálibo).

- Alto de trabe.
- Ancho de camino.

Con base en estos valores, la aplicación valida que los campos no estén vacíos y que los parámetros ingresados respeten los límites permitidos (por ejemplo, el ancho de columna y el alto de trabe no pueden ser mayores a 1).

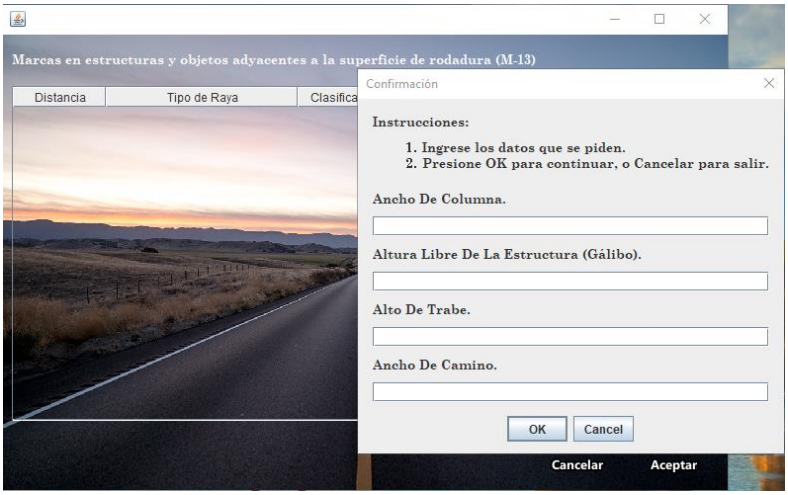


Figura 6.60. Aplicación: Agregando marcas M-13.1

Posteriormente, la aplicación realiza el cálculo automático de pintura:

- Si la altura de la estructura es mayor a 4.5 m, se considera un valor fijo de 3.0 m para el alto de trabe, y el resultado se obtiene multiplicando este valor por el ancho de columna.
- Si la altura de la estructura es menor a 4.5 m, el cálculo considera tanto las columnas como la trabe

El valor final se redondea y se agrega a la tabla junto con la clasificación M-13.1, el tipo de raya y la cantidad de pintura calculada. Finalmente, el usuario debe presionar el botón de aceptar para confirmar y registrar la información en la base de datos.

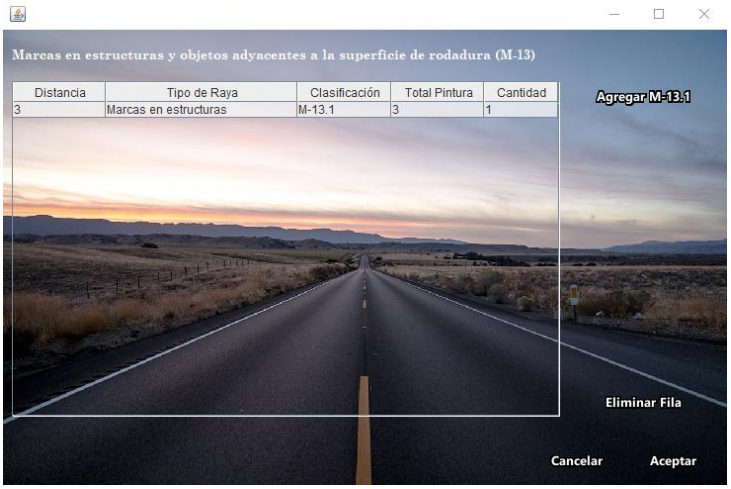


Figura 6.61. Aplicación: Marca M-13.1 agregada

- **M14**

En el caso de M14 se tienen cuatro botones que permiten agregar distintos subtipos de marcas: M-14.1 (Raya de emergencia para frenado discontinua), M-14.2 (Raya de emergencia para frenado continua), M-14.3R (Marca para acceso a una rampa de emergencia, áreas en color rojo) y M-14.3B (Marca para acceso a una rampa de emergencia, áreas en color blanco).

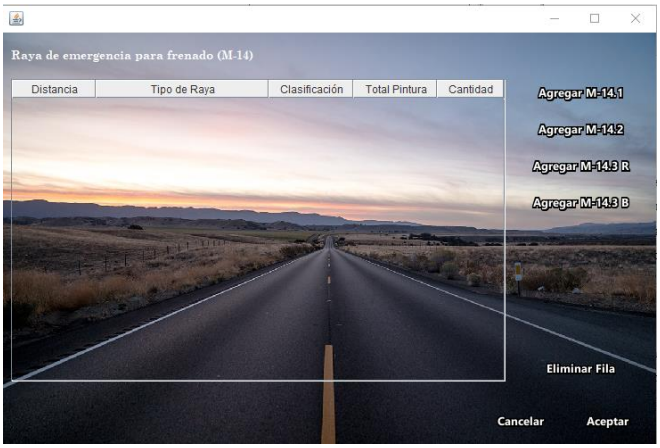


Figura 6.62. Aplicación: M-14

Para M-14.1, el procedimiento consiste en ejecutar en AutoCAD el comando *GuardarLongitudes* y seleccionar todas las líneas discontinuas correspondientes. Posteriormente, en la aplicación se confirma la operación y se realiza el cálculo de pintura aplicando la fórmula establecida para este subtipo.

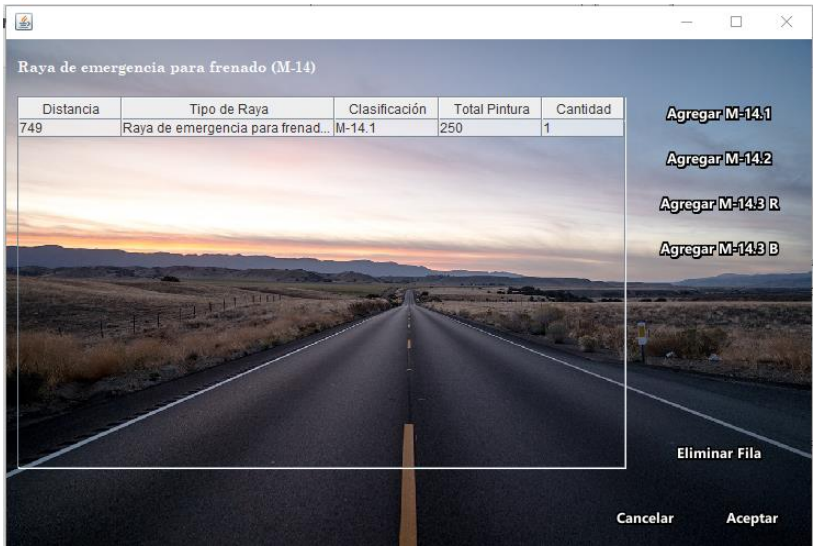


Figura 6.63. Aplicación: Agregando marcas M-14.1

En el caso de M-14.2, la lógica es similar, solo que se seleccionan líneas continuas. El cálculo de pintura en este caso corresponde directamente a la longitud total registrada.

Figura 6.64. Aplicación: Agregando marcas M-14.2

Para M-14.3, se distinguen dos variantes: M-14.3R para áreas en color rojo y M-14.3B para áreas en color blanco. En ambos casos, el usuario debe ejecutar el comando *GuardarAreas* en AutoCAD y seleccionar las áreas correspondientes al color indicado. La aplicación suma todas las áreas seleccionadas y con base en ese valor calcula la cantidad de pintura a registrar. La diferencia radica únicamente en el color, lo que permite clasificar los resultados en la tabla como M-14.3R o M-14.3B.

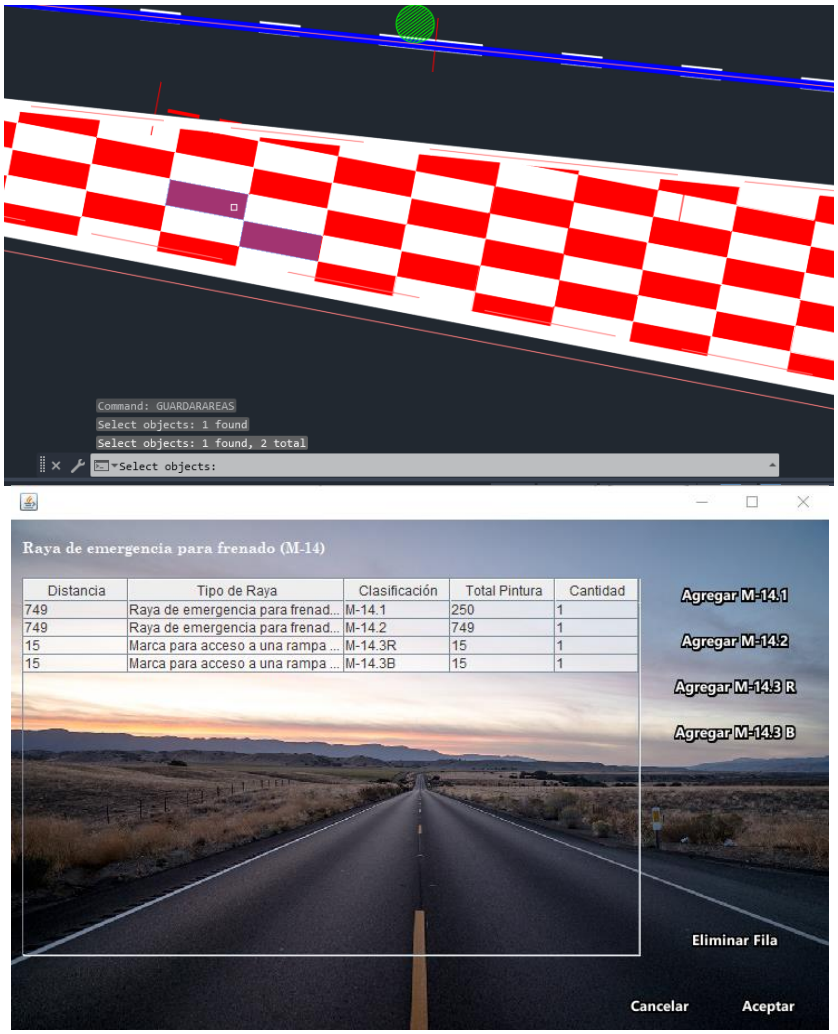


Figura 6.65. Aplicación: Agregando áreas M-14.3

Una vez agregados los valores en la tabla, el usuario debe confirmar presionando el botón de aceptar para que los datos se almacenen en la base de datos.

- **M15**

En el caso de M15 se dispone de un único botón para las marcas M-15 (Marcas para vías ciclistas), y en este caso solo se agregará las correspondientes a M-15-5 (Rayas para cruce de ciclistas)

Figura 6.66. Aplicación: M-15

Al presionarlo, se despliega una ventana en la que se solicita al usuario ingresar dos datos: la extensión del cruce y el ancho del camino. Con esta información, la aplicación realiza el cálculo automático de la cantidad de pintura.

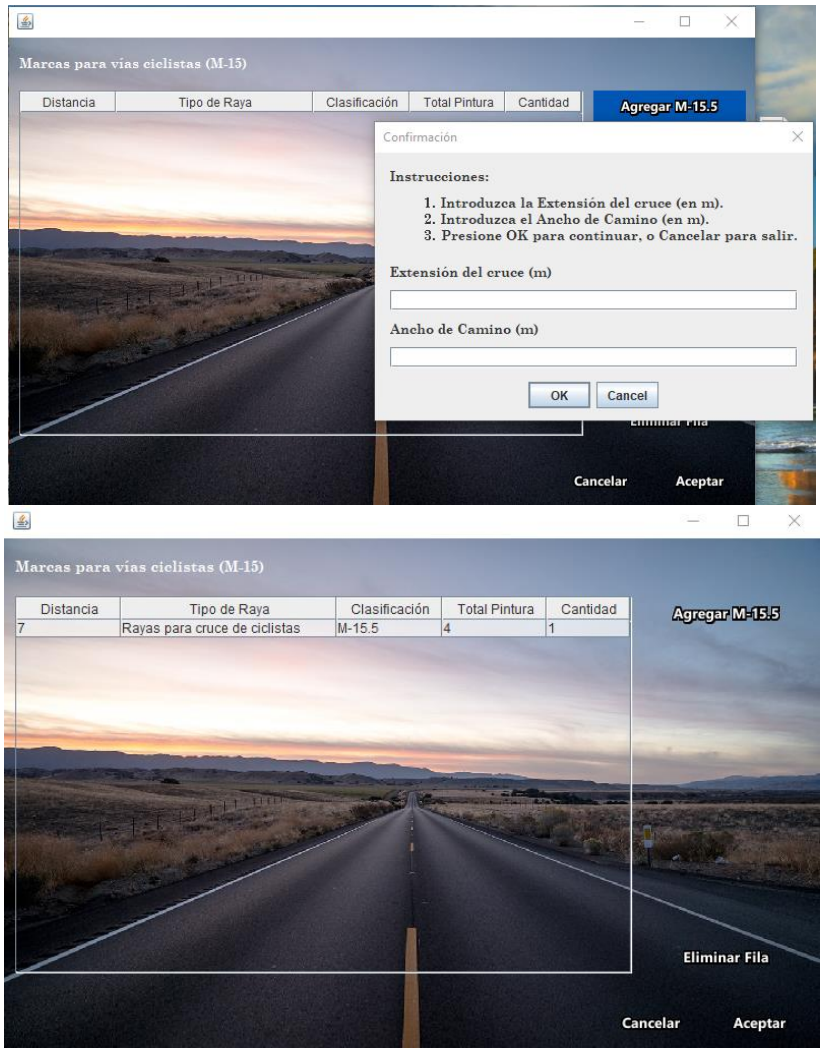


Figura 6.67. Aplicación: Agregando marcas M-15.5

El resultado del cálculo se agrega a la tabla junto con la clasificación M-15.5. Y después, el usuario debe presionar el botón de aceptar para confirmar y registrar los datos en la base de datos.

- **M16**

En el caso de M16 se dispone de un botón principal denominado *"Agregar Marcas"*, correspondiente a las marcas temporales.

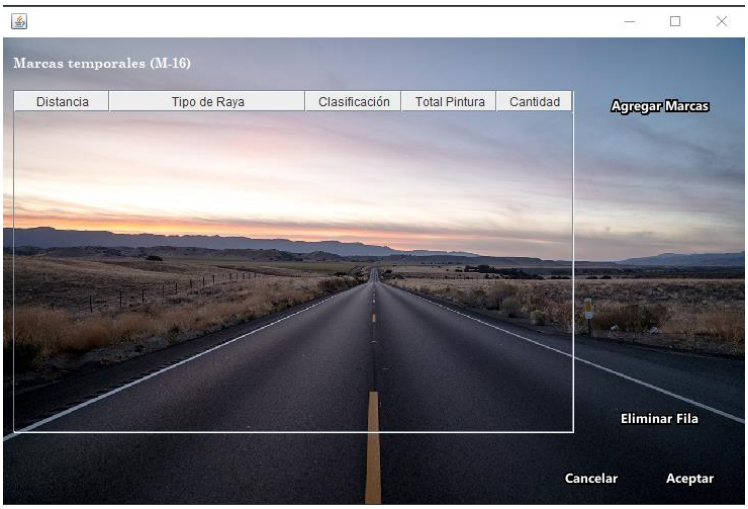


Figura 6.68. Aplicación: M-16

Al presionarlo, se despliega un cuadro con instrucciones para trabajar en AutoCAD, donde el usuario debe ejecutar el comando GuardarLongitudes y seleccionar todas las líneas que correspondan a las marcas temporales. Además, se habilita un campo adicional en el que se solicita seleccionar el color de la raya, que puede ser Azul o Naranja.

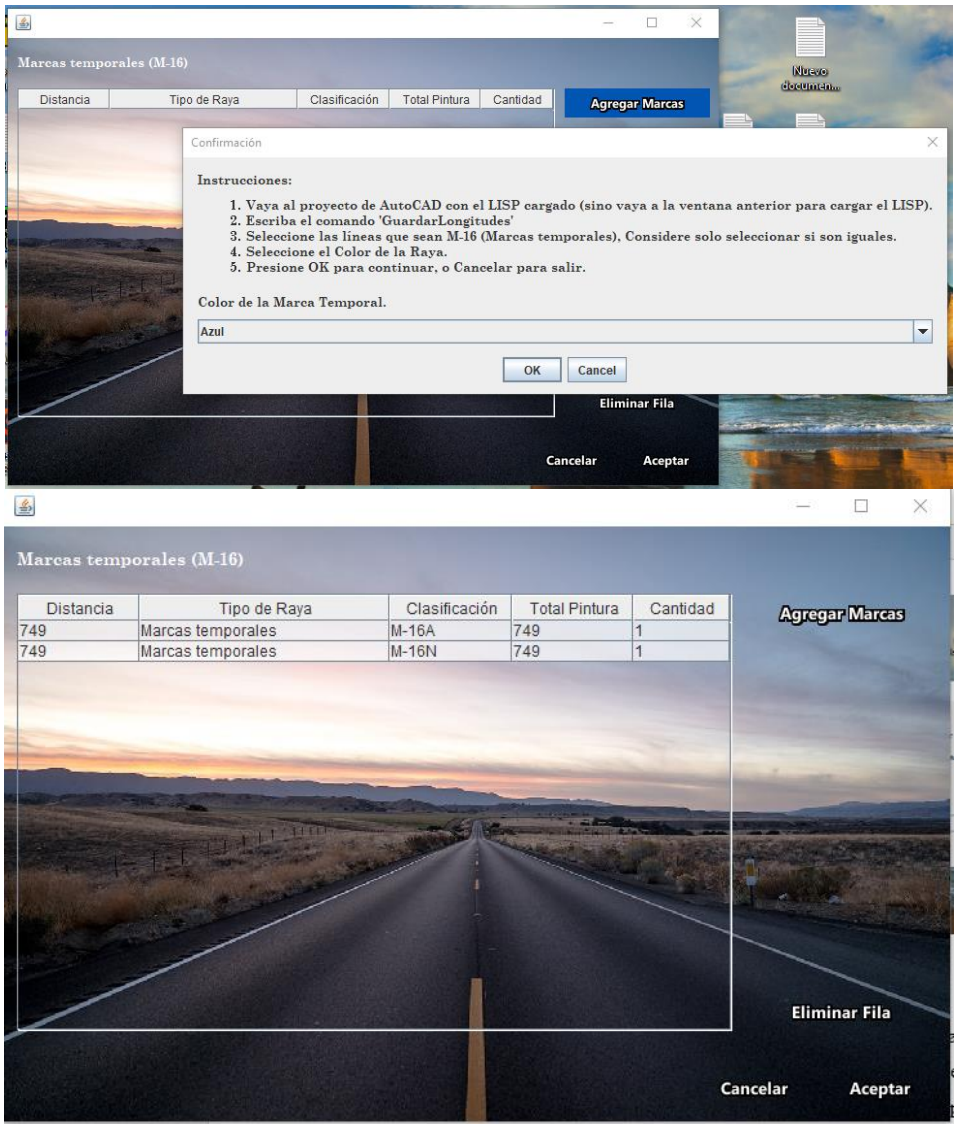


Figura 6.69. Aplicación: Agregando marcas M-16

El procedimiento consiste en sumar automáticamente la longitud de todas las líneas seleccionadas. Posteriormente, se agrega la información a la tabla, clasificando el resultado como M-16A (Azul) o M-16N (Naranja) según la elección del usuario. Y después, se debe presionar el botón de aceptar para confirmar y guardar los datos en la base de datos.

- **M17**

En el caso de M17 se utiliza un único botón denominado *"Agregar M-17"*, correspondiente a la marca de área de espera para vehículos no motorizados y motocicletas.

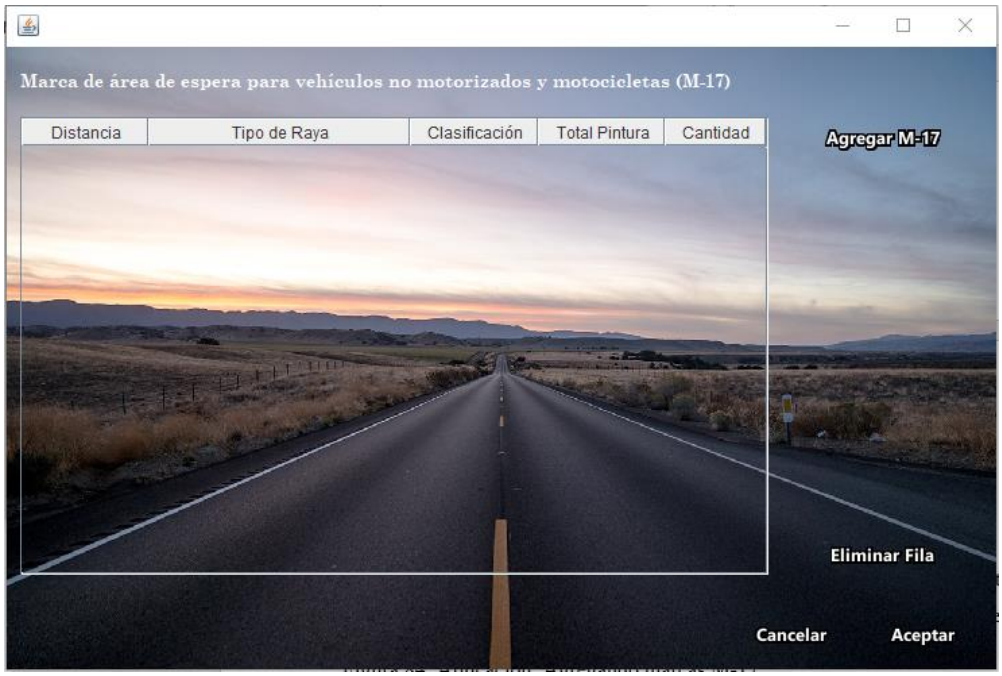


Figura 6.70. Aplicación: M-17

Al presionar este botón, se muestran las instrucciones que indican que en AutoCAD se debe ejecutar el comando GuardarLongitudes y seleccionar las líneas correspondientes a esta marca. Y una vez agregada a la tabla, usuario debe confirmar los cambios presionando Aceptar para almacenar los datos en la base de datos.

Figura 6.71. Aplicación: Agregando marcas M-17

- **M18**

En el caso de M18 se dispone de un botón denominado *"Agregar M-18"*, correspondiente a la marca de ceda el paso.

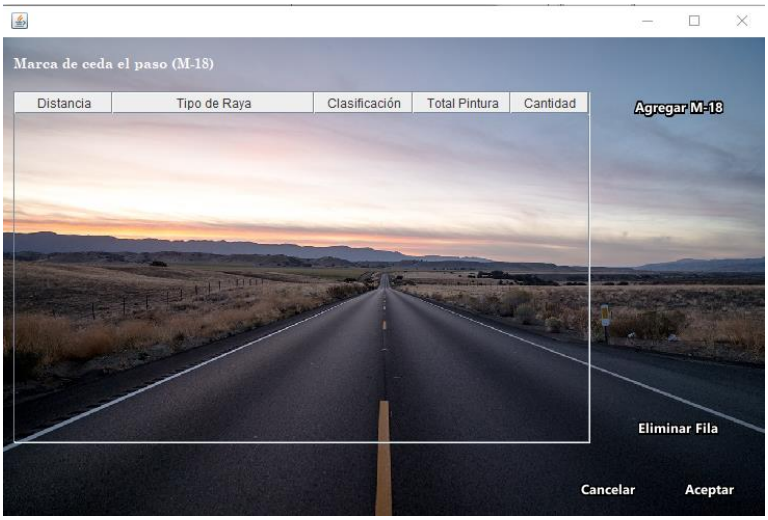


Figura 6.72. Aplicación: M-18

Al presionarlo, la aplicación muestra un cuadro de instrucciones donde se solicita al usuario abrir AutoCAD, ejecutar el comando GuardarAreas y seleccionar los triángulos que representan las marcas de ceda el paso.

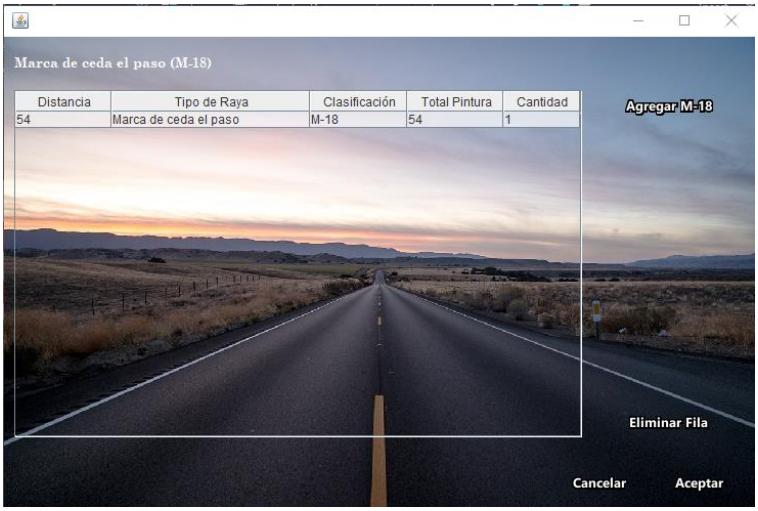


Figura 6.73. Aplicación: Agregando marcas M-18

La aplicación lee las áreas seleccionadas, las suma y con base en el resultado calcula automáticamente la cantidad de pintura necesaria. Los valores obtenidos se agregan a la tabla junto

con la clasificación M-18. Y después, se debe presionar aceptar para guardar la información en la base de datos.

- **M19**

En el caso de M19 se incluyen tres botones distintos, cada uno correspondiente a un subtipo de marca:

- M-19.1: Prohibido estacionar.
- M-19.2: Prohibido parar.
- M-19.3: Prohibido parar en intersección.

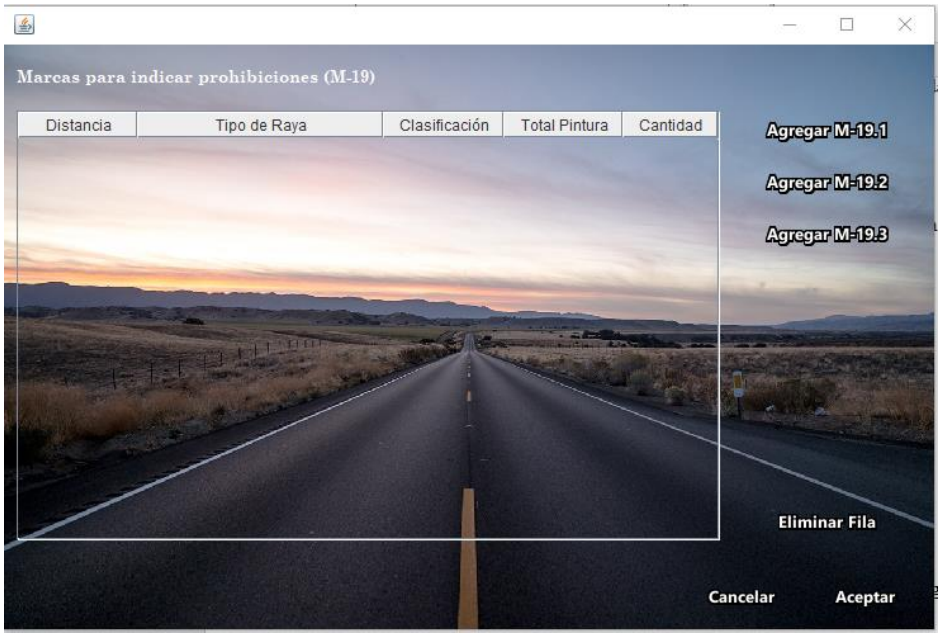


Figura 6.74. Aplicación: M-19

En todos los casos, el procedimiento consiste en abrir AutoCAD, ejecutar el comando GuardarLongitudes y seleccionar las líneas que correspondan al subtipo elegido.

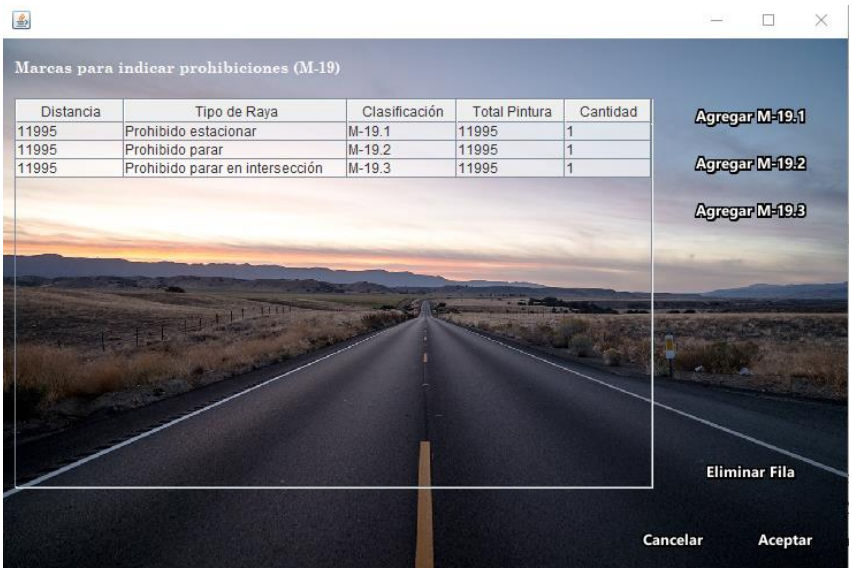


Figura 88: Aplicación: Agregando marcas M-19

Una vez confirmada la operación, la aplicación suma la longitud total registrada y calcula la pintura necesaria de acuerdo con la clasificación seleccionada (M-19.1, M-19.2 o M-19.3). Y después, los resultados que se agregan en la tabla y el usuario debe de presionar Aceptar para que la información quede almacenada en la base de datos.

- **M20**

En el caso de M20 se dispone de un botón denominado *"Agregar M-20"*, correspondiente a las marcas de reductores de velocidad.

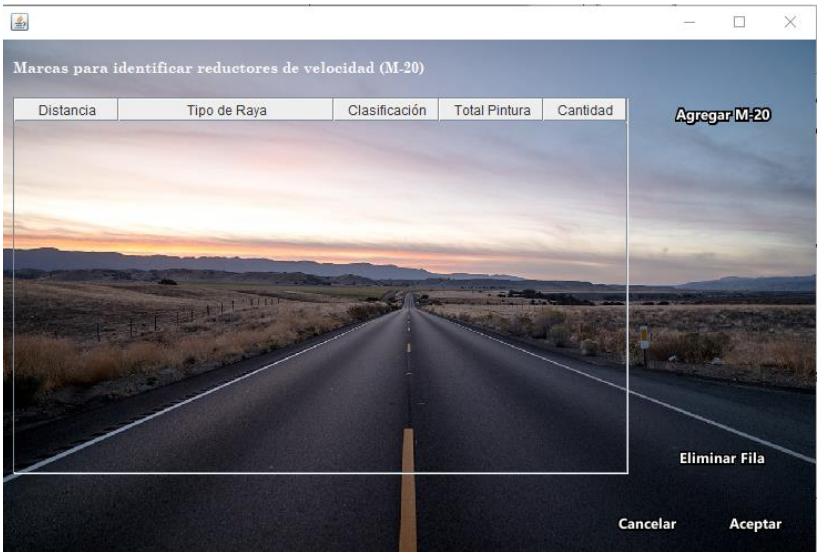


Figura 6.75. Aplicación: M-20

Al presionarlo, se despliega un mensaje en el que se solicita al usuario ingresar la velocidad del camino, el ancho de la vía.

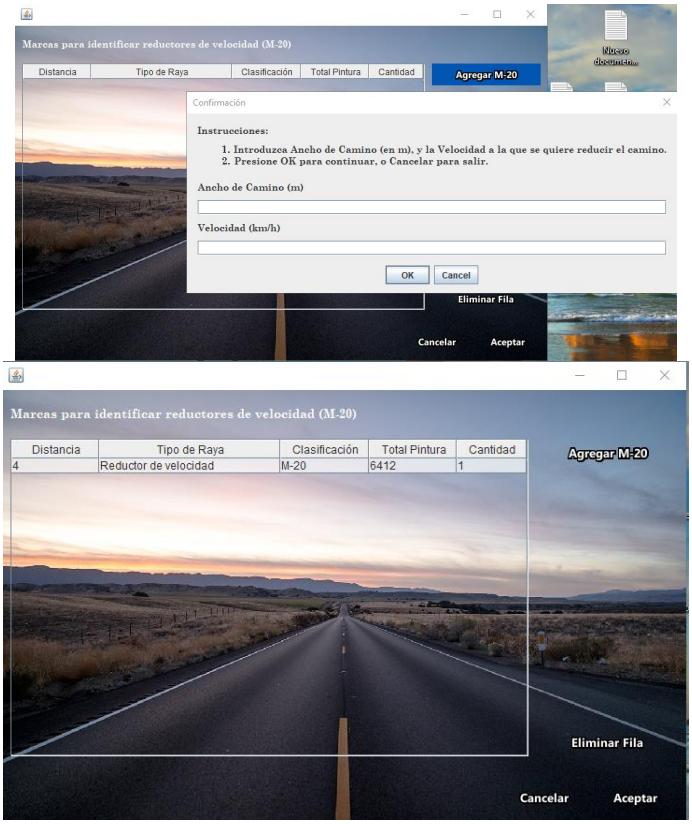


Figura 6.76 Aplicación: Agregando marcas M-20

Con estos datos, la aplicación realiza el cálculo automático del área de pintura aplicando la fórmula correspondiente a cada rango de velocidad. Y después, se debe presionar el botón de aceptar para confirmar y registrar los valores en la base de datos.

- **Exportar**

Una vez que el usuario haya terminado de agregar todas las Señales Horizontales que requiera su proyecto, puede presionar Exportar para visualizar una tabla con todas las señales Horizontales que haya guardado en la base de datos, esta tabla sirve como vista previa antes de mandar la tabla a AutoCAD, aquí el usuario puede cambiar algún valor antes de pasarlo a AutoCAD, también como aquí se les asigna una dimensión al momento de presionar alguna celda en la columna de Dimensión se desplegará una lista de posibles dimensiones según sea el caso.

Figura 6.77. Aplicación: Exportar tabla SH

Al confirmar que todos los datos son correctos se presiona aceptar para poder mandar los datos al archivo TXT que AutoCAD va a leer con ayuda del comando CrearTablasSB, el cual al usarlo permite al usuario colocar la tabla en algún punto deseado dentro del plano

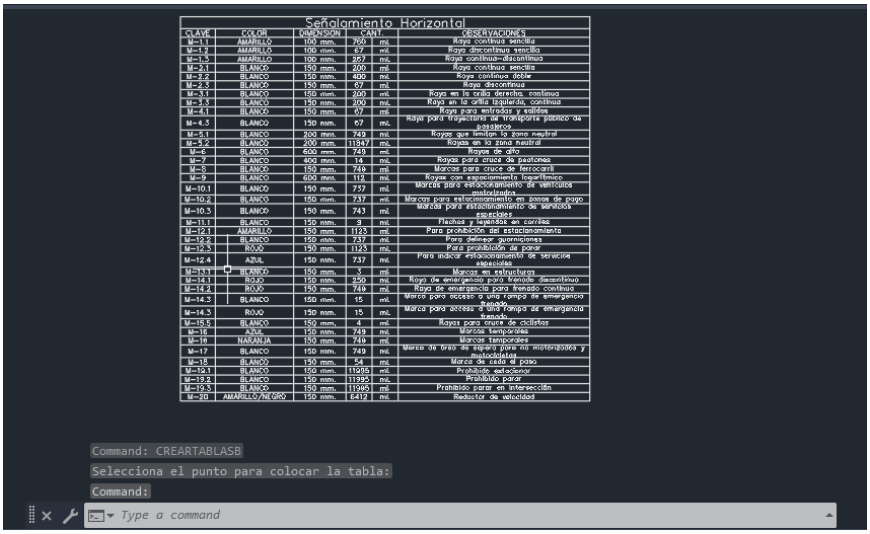


Figura 6.78. Aplicación: Tabla SB en AutoCAD

❖ Agregar Señales Verticales

Si el usuario elige "Agregar Señales Verticales" entonces se abrirá una ventana donde se encuentra una tabla y botones que permiten agregar señales Verticales en Camino abierto y en Entronques.

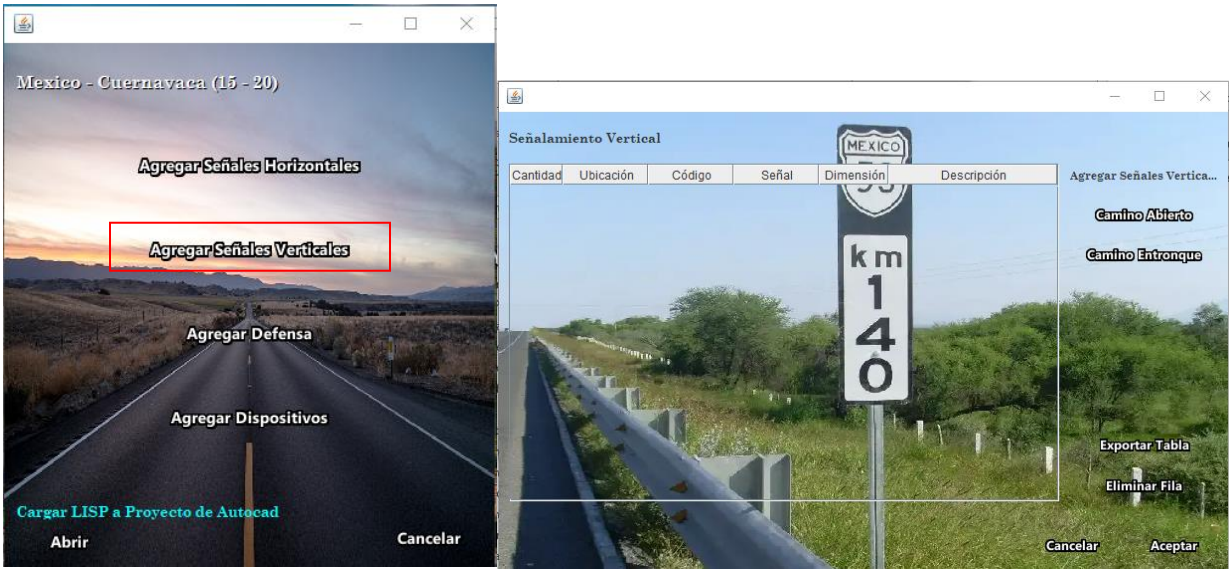


Figura 6.79. Aplicación: Señalamiento Vertical (Elección)

- **Señales Verticales en Camino Abierto**

Si el usuario elige "Camino Abierto", este permite agregar a la tabla las señales verticales correspondientes a tramos abiertos de camino. Al presionar este botón, la aplicación despliega un cuadro de instrucciones que indican al usuario ir a AutoCAD, ejecutar el comando SeleccionarTextos y seleccionar en el siguiente orden los textos que identifican a cada señal: Señal (nombre), Dimensión y Ubicación. Como ejemplo se tiene la siguiente señal:

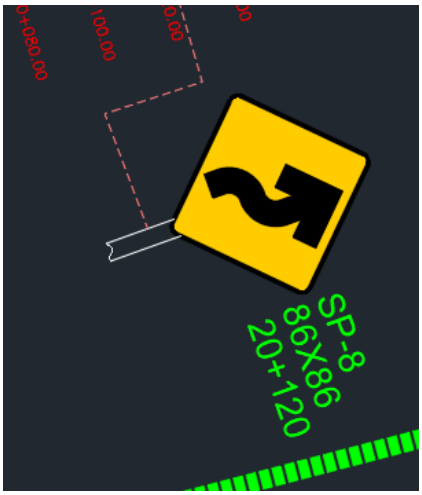


Figura 6.80. Aplicación: Ejemplo Señal Vertical Camino Abierto

Siguiendo este ejemplo el usuario usando el comando SeleccionarTextos debería seleccionar en este orden: SP-18, 86X86, 20+120

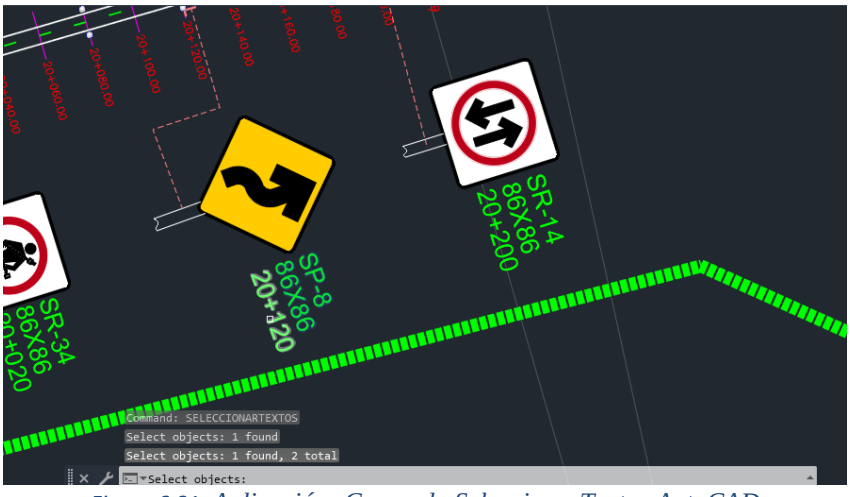


Figura 6.81. Aplicación: Comando SeleccionarTextos AutoCAD

Una vez realizada la selección, la aplicación, siguiendo la función de las ventanas anteriores, donde el usuario NO debe de confirmar el cuadro de información hasta realizar primero los pasos en AutoCAD, y una vez hechos los pasos, en la aplicación se confirma el cuadro y entonces se lee automáticamente la información y se organiza automáticamente los textos capturados y los organiza en grupos de tres elementos. Por cada grupo, se genera una fila en la tabla con la siguiente estructura:

- Cantidad (por defecto "1").
- Ubicación.
- Código (se deja vacío en este caso).
- Señal.
- Dimensión.
- Descripción (obtenida de forma automática a partir de la señal).

Figura 6.82. Aplicación: Agregando Señal Vertical de Camino Abierto

Finalmente, el usuario debe presionar el botón de aceptar para que la información quede registrada en la base de datos.

- **Señales Verticales en Entronque**

Si el usuario elige "Camino Entronque", este permite agregar a la tabla las señales verticales que corresponden a entronques. Al presionarlo, la aplicación muestra un cuadro de instrucciones donde

se indica que en AutoCAD se debe ejecutar el comando SelecTextosEntr y seleccionar los textos en el orden siguiente: Señal, Dimensión, Ubicación y Código. Como ejemplo se tiene la siguiente señal:

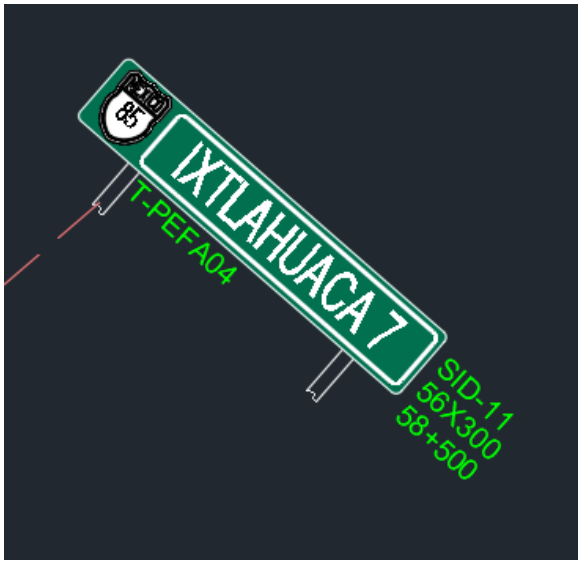


Figura 6.83. Aplicación: Ejemplo Señal Vertical en Entronque

Siguiendo este ejemplo el usuario usando el comando SelecTextosEntr debería seleccionar en este orden: SID-11, 56X300, 58+500, T-PEFA04

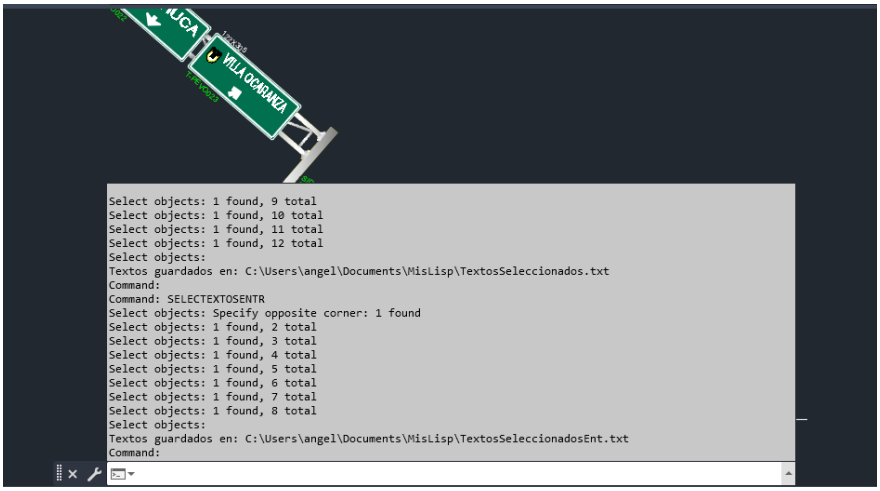


Figura 6.84. Aplicación: Comando SelecTextosEntr AutoCAD

En la aplicación, siguiendo la función de las ventanas anteriores, donde el usuario NO debe de confirmar el cuadro de información hasta realizar primero los pasos en AutoCAD, y una vez hechos los pasos, en la aplicación se confirma el cuadro y entonces se lee automáticamente la información

y se organiza automáticamente los textos en grupos de cuatro y genera en la tabla una nueva fila por cada conjunto de datos, con la siguiente estructura:

- Cantidad (por defecto “1”).
- Ubicación.
- Código.
- Señal.
- Dimensión.
- Descripción (calculada automáticamente con base en la señal).

Figura 6.85. Aplicación: Agregando Señal Vertical de Camino Abierto

Al finalizar, los datos quedan listos en la tabla y el usuario debe presionar el botón de aceptar para confirmar y guardar la información en la base de datos.

- **Eliminar**

En el caso del botón eliminar, funciona de igual que para las Señales Horizontales donde el usuario tiene que elegir una fila de la tabla (Los datos ya debieron haberse subido a la base de datos), y después presionar el botón de eliminar lo que hace es mostrar un mensaje de confirmación y después muestra los datos que se eliminaron.

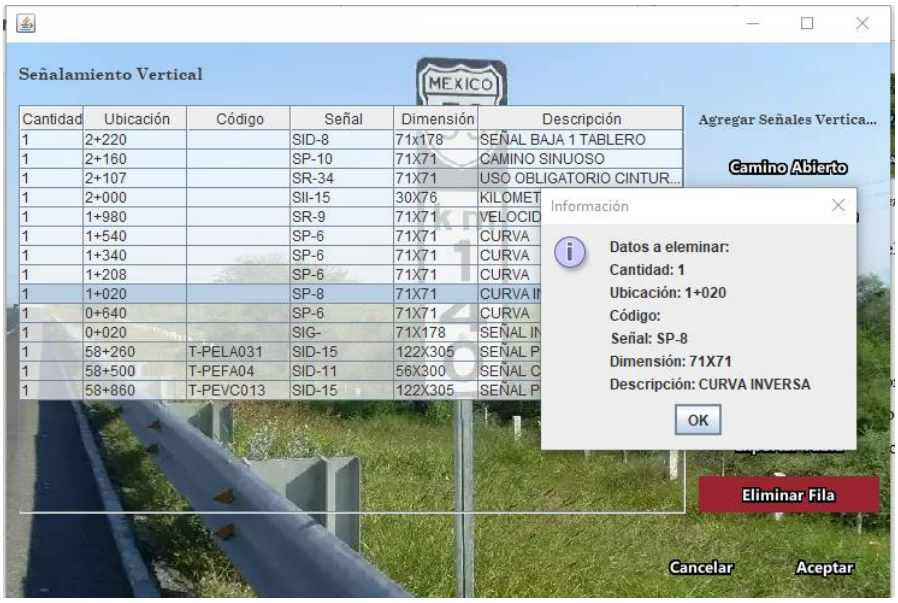


Figura 6.86. Aplicación: Eliminado Señal

- **Exportar**

Una vez que se este seguro que los datos ya están completos y se quieran pasar a AutoCAD, entonces se presiona el Botón de Exportar, el cual abrirá una nueva ventana donde se mostrará una tabla con los datos que ya se guardaron en la base de datos.

Figura 6.87. Aplicación: Exportación de tabla de Señales Verticales

En esta ventana hay un botón llamado “Mostrar Tabla Totales”, el cual permite cambiar de tabla. Inicialmente al abrirse la ventana se muestran todas las señales Verticales en el orden que se subieron

a la base de datos, al presionar “Mostrar Tabla Totales” la tabla que se muestra es una de totales donde salen las señales que se agregaron y el número que veces que se agregaron para ese proyecto

Figura 6.88. Aplicación: Exportación de tabla Totales de Señales Verticales

En esta misma ventana aparece un nuevo botón llamado “Agregar OD-12”, Si el usuario selecciona este botón, la aplicación despliega un cuadro de instrucciones donde se indica que se debe ir a AutoCAD y ejecutar el comando DistanciaCurva, siguiendo la función de las ventanas anteriores, donde el usuario NO debe de confirmar el cuadro de información hasta realizar primero los pasos en AutoCAD.

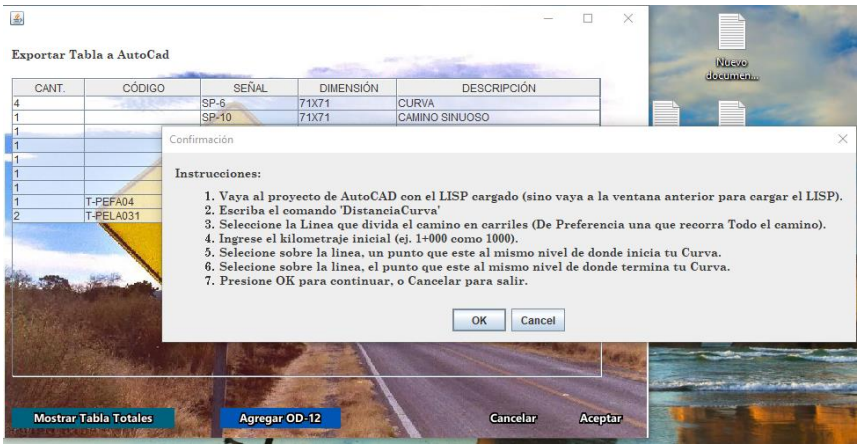


Figura 6.89. Aplicación: Botón Agregar OD-12

Las instrucciones que el usuario debe de seguir son:

1. Seleccionar la línea que divida el camino en carriles (preferiblemente una que recorra todo el tramo que se esté trabajando).
2. Ingresar el kilometraje con el que inicia el tramo considerando, por ejemplo: si el kilometraje inicial es 1+000 entonces ingresarlo como 1000.
3. Indicar, sobre la línea, el punto que corresponde al inicio de la curva donde quiera colocar OD - 12.
4. Indicar, sobre la línea, el punto que corresponde al final de la curva.
5. Confirmar la selección presionando OK en la ventana de la aplicación

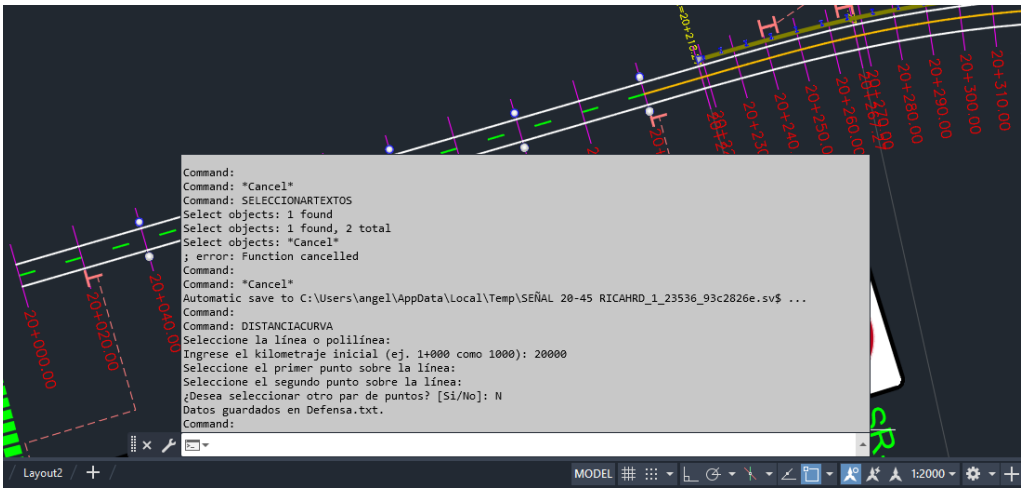


Figura 6.90. Aplicación: Ejemplo selección de curva en AutoCAD

Una vez confirmada la operación en la aplicación, se leen los valores registrados en AutoCAD. Para cada curva seleccionada, el sistema muestra un nuevo cuadro solicitando el grado de curvatura correspondiente. El usuario debe ingresar este valor en el campo habilitado; en caso de no hacerlo, se muestra un mensaje de error indicando que los campos no pueden quedar vacíos.

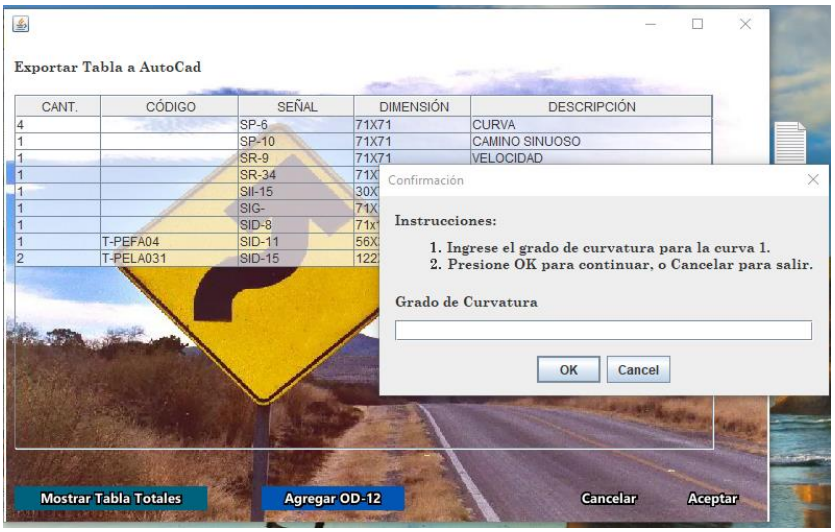


Figura 6.91. Aplicación: Ingreso de grado de curvatura

El sistema valida que el dato ingresado sea un número entero. A partir del grado de curvatura se calcula la separación de los dispositivos OD-12 a colocar, y con base en los kilometrajes inicial y final de la curva se obtiene la longitud total. Con estos valores, la aplicación determina la cantidad de OD-12 necesarios para la curva.

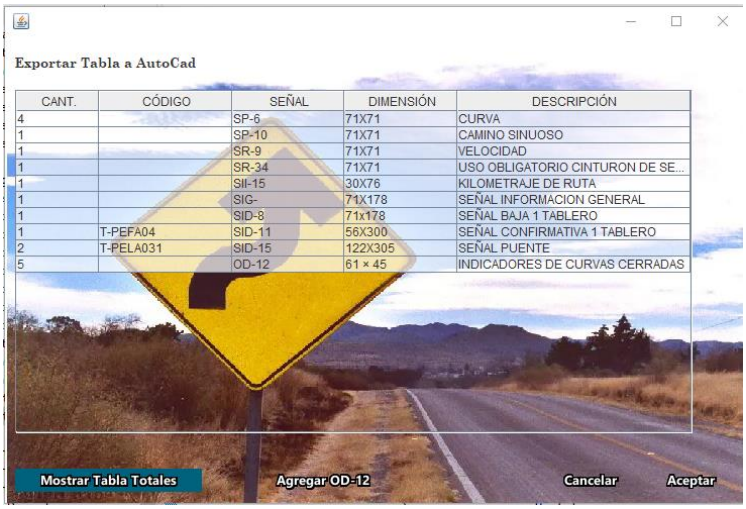


Figura 6.92. Aplicación: Agregando OD-12

Al terminar el proceso para todas las curvas, el sistema actualiza automáticamente la tabla de Totales agregando una fila de OD-12 y solo esta tabla se ve afectada.

Una vez que se haya verificado que ambas tablas están bien y completas, el usuario debe de usar el botón “Aceptar”, este botón permite exportar la información de las señales verticales que ya fueron

cargadas en la tabla. La acción de este botón se divide en dos procesos: exportación de las filas seleccionadas y exportación de la tabla de totales.

Para exportar las filas seleccionadas el usuario debe de primero, en la tabla donde esta toda la lista de las Señales Verticales, seleccionar todas señales que el usuario quiera exportar, esto es útil ya que hay casos en donde el usuario tiene un tramo con dos sentidos de circulación, y lo que normalmente se solicita para entregar es una tabla con las señales del lado derecho, otra con las del lado izquierdo y una tabla con los totales del tramo, es decir ambos lados, entonces lo que se hace es permitir al usuario que primero pueda agregar todas las señales correspondientes al tramo, pero al permitir al usuario que pueda seleccionar las filas a exportar, hacemos que pueda crear una tabla para las señales que corresponden al lado derecho y otra para las del lado izquierdo, además de una tabla los totales.

Al presionar este botón, la aplicación primero muestra un cuadro de confirmación donde se pregunta al usuario si desea exportar únicamente las filas seleccionadas de la tabla.

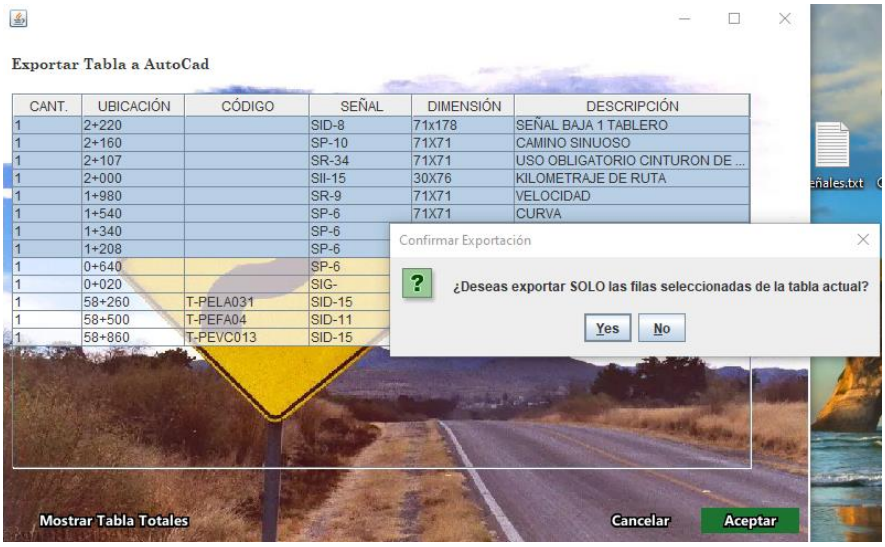


Figura 6.93. Aplicación: Exportando filas seleccionadas

- Si no se seleccionó ninguna fila, aparece un mensaje de error indicando que es necesario elegir al menos una.

- Si existen filas seleccionadas, el usuario debe confirmar la ventana y la aplicación toma automáticamente los valores de cada columna de las filas elegidas y los organiza en una estructura de texto separada por comas en un archivo TXT.
- Cada fila se convierte en una línea dentro de un archivo, el cual posteriormente será utilizado en AutoCAD.

Una vez finalizada la exportación, el sistema informa al usuario que debe ir a AutoCAD, cargar el LISP y escribir el comando CrearTablaSV para que los datos exportados se conviertan en una tabla dentro del proyecto.

Figura 6.94. Aplicación: Exportando filas seleccionadas en AutoCAD

Posteriormente, la aplicación consulta al usuario si desea exportar también la tabla de Totales. En caso afirmativo, si durante la sesión se agregaron nuevas señales (OD-12), la aplicación muestra un cuadro de confirmación adicional preguntando si se desea guardar esas señales antes de exportar, si el usuario confirma, se toman automáticamente los valores de la última fila de la tabla de totales y se almacenan en la base de datos como un nuevo registro. El sistema notifica si el guardado fue exitoso o si ocurrió algún error en el proceso.

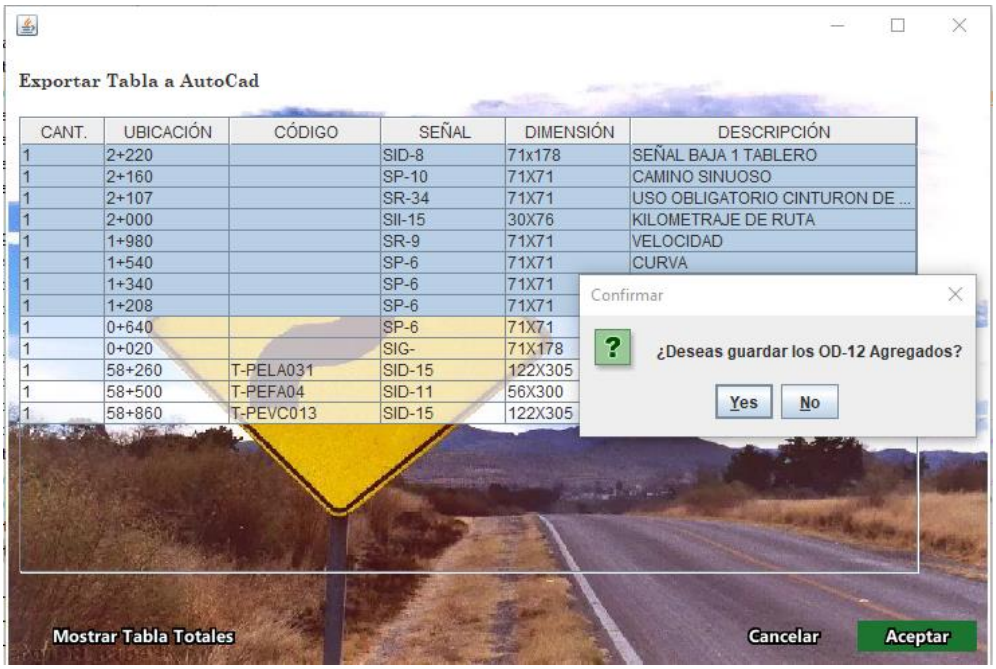


Figura 6.95. Aplicación: Exportando filas seleccionadas en AutoCAD

Después de esto, se genera el archivo con la información consolidada de la tabla de Totales, donde se incluyen todas las señales y el número de veces que fueron agregadas en el proyecto.

Para concluir, el sistema muestra un mensaje indicando que, en AutoCAD, tras cargar el LISP, se debe escribir el comando CrearTablaSVTotal para desplegar la tabla de totales.

Figura 6.96. Aplicación: Exportación de tabla de Totales en AutoCAD

❖ Agregar Defensa

Si el usuario elige “Agregar Defensa” entonces se abrirá una ventana donde se encuentra una tabla y botones que permiten agregar Defensa del lado Derecho y del lado Izquierdo.

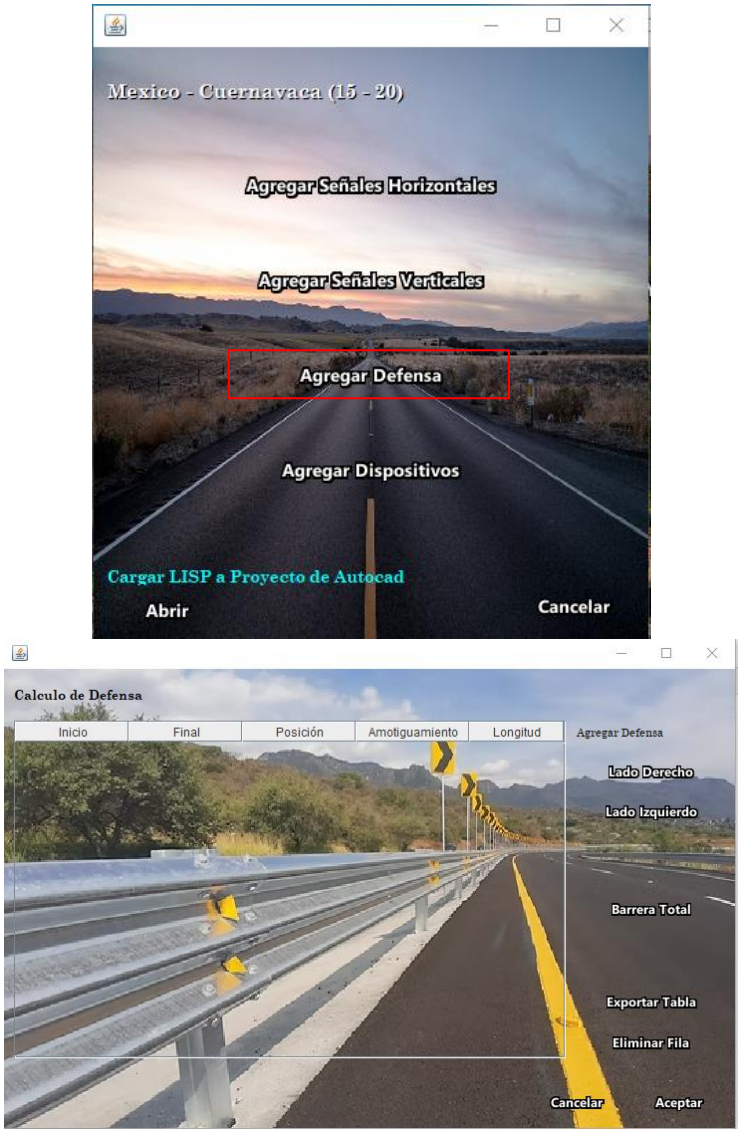


Figura 6.97. Aplicación: Defensa (Elección)

En esta sección la aplicación cuenta con dos botones principales: “Lado Derecho” y “Lado Izquierdo”, los cuales permiten registrar las defensas ubicadas a cada costado de la vía.

Al presionar cualquiera de estos botones, se despliega un mensaje con instrucciones que guían al usuario en el proceso dentro de AutoCAD. Se le indica que debe de usar el comando

DistanciaEntrePuntos, el cual le solicitará: Seleccionar una línea que recorra todo el tramo, preferentemente, ingresar el kilometraje con el que inicia el tramo considerando, por ejemplo: si el kilometraje inicial es 1+000 entonces ingresarlo como 1000, seleccionar sobre la línea el punto correspondiente al inicio de la defensa (lado derecho o izquierdo, según el botón presionado), seleccionar el punto que corresponda al final de la defensa.

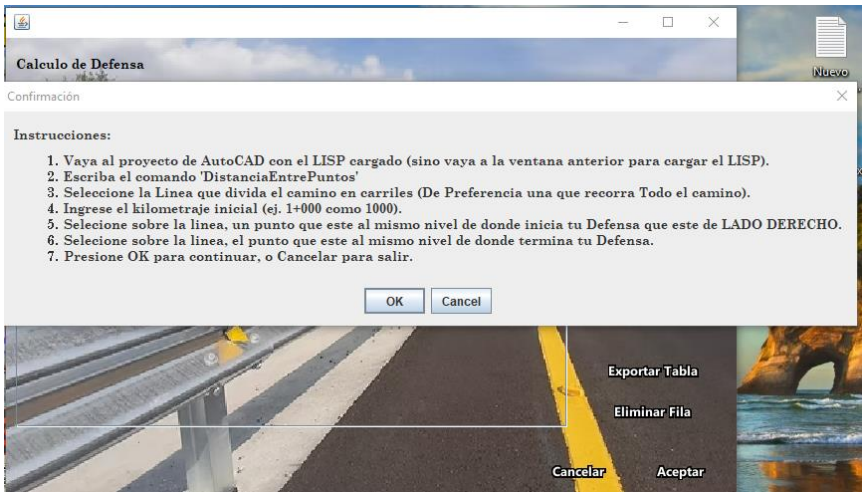


Figura 6.98. Aplicación: Instrucciones para agregar Defensa

Una vez completado este procedimiento, la aplicación lee los datos registrados en el archivo de AutoCAD y los procesa, de este modo, el sistema permite organizar y llevar un control detallado de las defensas laterales de cada tramo del proyecto.

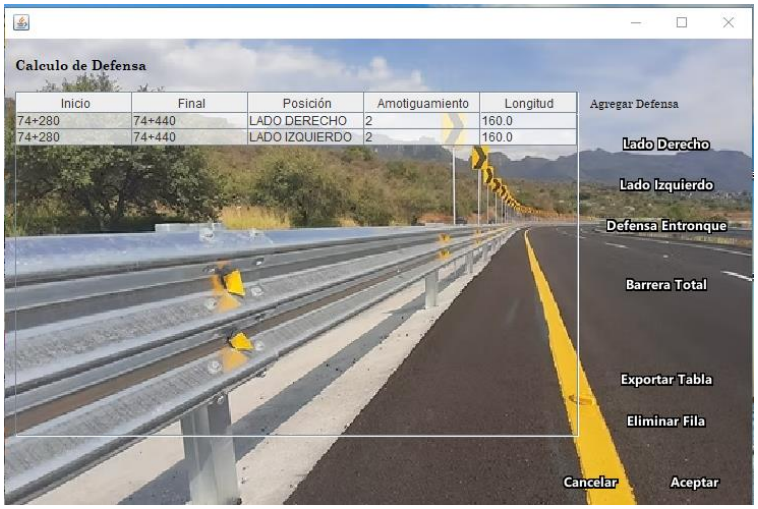


Figura 6.99. Aplicación: Agregar datos Defensa Ambos lados

Al finalizar, los datos quedan listos en la tabla y el usuario debe presionar el botón de Aceptar para confirmar y guardar la información en la base de datos.

- **Eliminar**

En el caso del botón eliminar, funciona de igual que Las Ventanas anteriores, donde el usuario tiene que elegir una fila de la tabla (Los datos ya debieron haberse subido a la base de datos), y después presionar el botón de eliminar lo que hace es mostrar un mensaje de confirmación y después muestra los datos que se eliminaron.

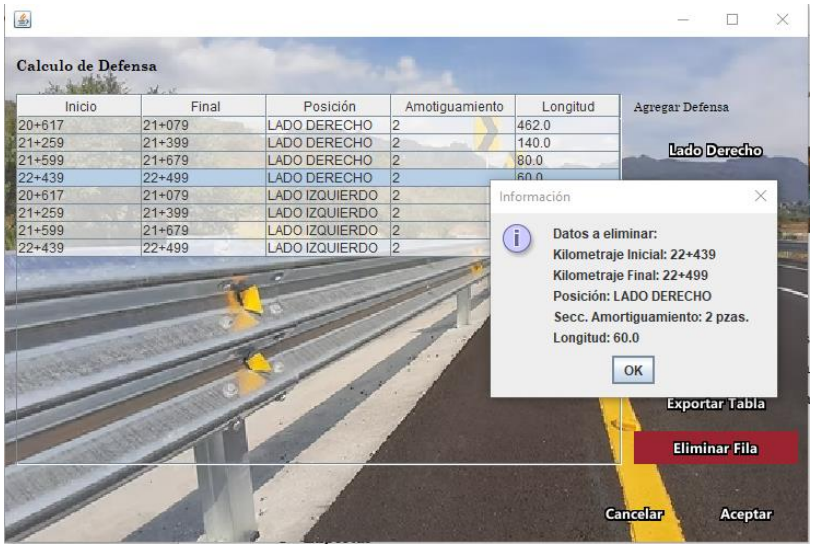


Figura 6.100. Aplicación: Eliminado Defensa

- **Exportar**

Una vez que se esté seguro que los datos ya están completos y se quieran pasar a AutoCAD, entonces se presiona el Botón de Exportar, el cual abrirá una nueva ventana donde se mostrara una tabla con los datos que ya se guardaron en la base de datos.

Figura 6.101. Aplicación: Exportación de tabla de Defensa

En esta ventana se muestran los datos acomodados de forma que da una vista previa de como estarán los datos en AutoCAD. Una vez que se comprueben los datos se presiona Aceptar para Exportar los datos a AutoCAD, indicando usar el comando CrearTablaDefensa.

Figura 6.102. Aplicación: Exportación de Tabla de Defensa

- Barrera Total

Si el usuario presiona “Barrera Total” entonces se abrirá una ventana donde se encuentra una tabla donde se muestran seis categorías principales: barrera metálica de tres crestas, barrera central tipo New Jersey, secciones de transición de rigidez, secciones extremas de amortiguamiento, amortiguadores de impacto y sección terminal tipo cola de pato.

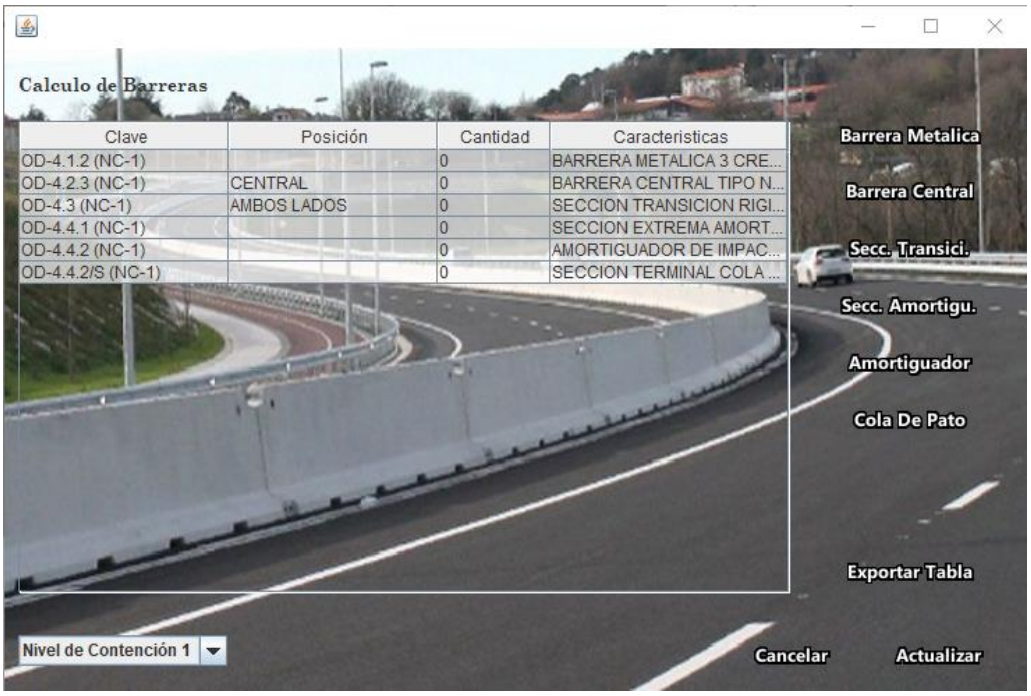


Figura 6.103. Aplicación: Barrera Total

Se cuentan con botones que permiten agregar información a cada uno de los apartados. Estos botones despliegan instrucciones específicas que el usuario debe seguir antes de confirmar la operación, ya que algunas requieren obtener datos directamente desde AutoCAD con apoyo de los scripts Lisp. Por ejemplo, en el caso de la "Barrera Central", se utiliza el comando DistanciaCurva para medir la longitud del tramo en donde se colocará este tipo de defensa. En otros apartados, como el de la sección de transición, los amortiguadores de impacto o cola de pato, el usuario simplemente debe indicar la cantidad total de elementos a través de un cuadro de selección numérica. En caso de la barrera metálica y la sección de Amortiguamiento, estos datos se leen automáticamente de la base de datos de la tabla de Defensa, lo que hace es sumar todas las longitudes de la defensa y el total de la sección de amortiguamiento que se crearon en la tabla de Defensa, por ello antes de hacer la cuantificación de las barreras, es necesario hacer la cuantificación de la defensa en la ventana de Cálculo de Defensa, vista anteriormente.

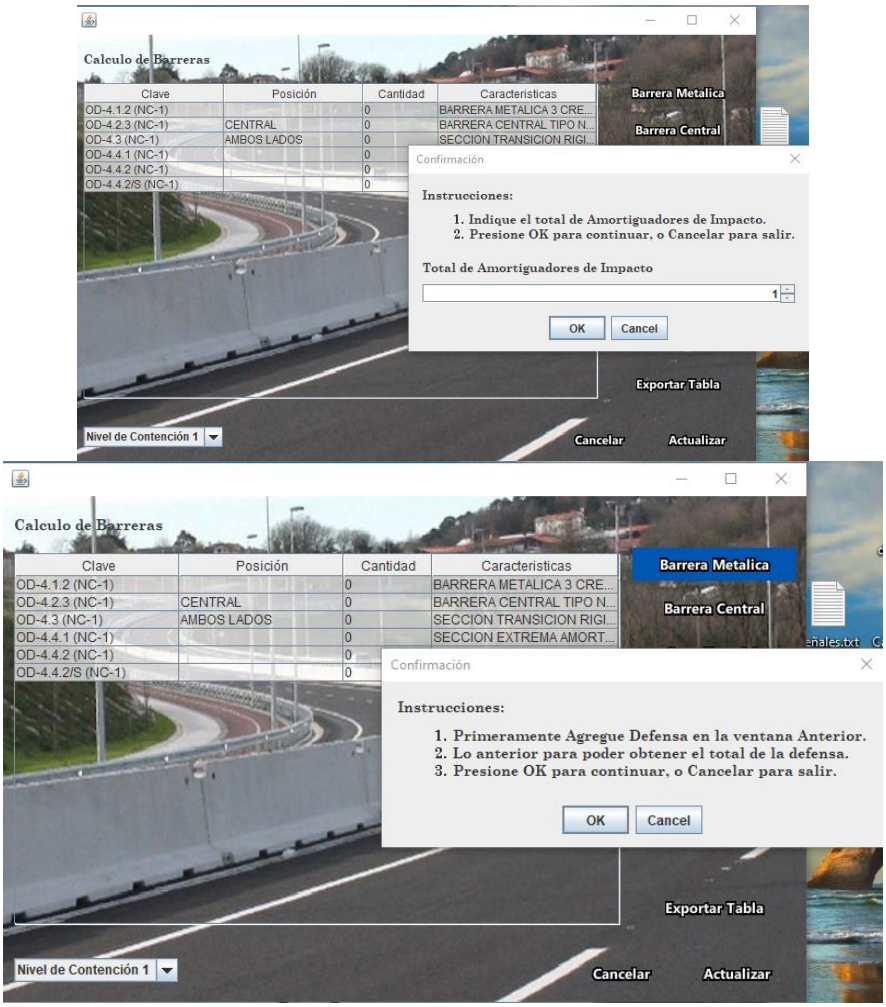


Figura 6.104. Aplicación: Agregando datos Amortiguador de Impacto y Barrera Metálica

Además, la ventana cuenta con un selector de nivel de contención (NC-1, NC-2 o NC-3), el cual actualiza las claves normativas mostradas en la primera columna de la tabla para mantener la referencia correcta según la clasificación establecida.

Figura 6.105. Aplicación: Elección de Nivel de Contención

Una vez que el usuario agregue los datos requeridos, se presione el botón Actualizar, para guardar la información en la base de datos.

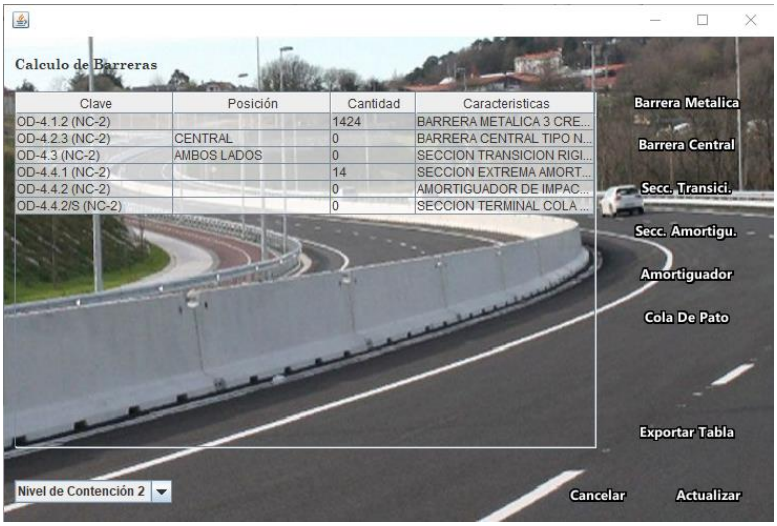


Figura 6.106. Aplicación: Actualizando datos Barrera Total

- **Exportar Tabla**

Una vez que se este seguro que los datos ya están completos y se quieran pasar a AutoCAD, entonces se presiona el Botón de Exportar Tabla, el cual abrirá una nueva ventana donde se mostrara una tabla con los datos que ya se guardaron en la base de datos.

Figura 6.107. Aplicación: Exportación de tabla Barrera Total

En esta ventana se muestra los datos acomodados de forma que da una vista previa de como estarán los datos acomodados en AutoCAD, si hay datos que tengan valor de 0 entonces estas filas no se mostrarán. Una vez que se comprueben los datos se presiona Aceptar para Exportar los datos a AutoCAD, indicando usar el comando CrearTablaBarrera.

Figura 6.108. Aplicación: Exportación de Tabla de Barrera en AutoCAD

❖ Agregar Dispositivos

Si el usuario elige "Agregar Dispositivos" entonces se abrirá una ventana donde se encuentra una tabla y botones que permiten agregar Diversos dispositivos como: Indicadores de alineamiento, Balizas, Delimitadores, Dispositivos Antideslumbrantes y Botones Reflejantes, además que también hay un botón llamando "Mostrar Tabla Botones", el cual permite que se pueda pasar a la tabla de botones y agregarlos.

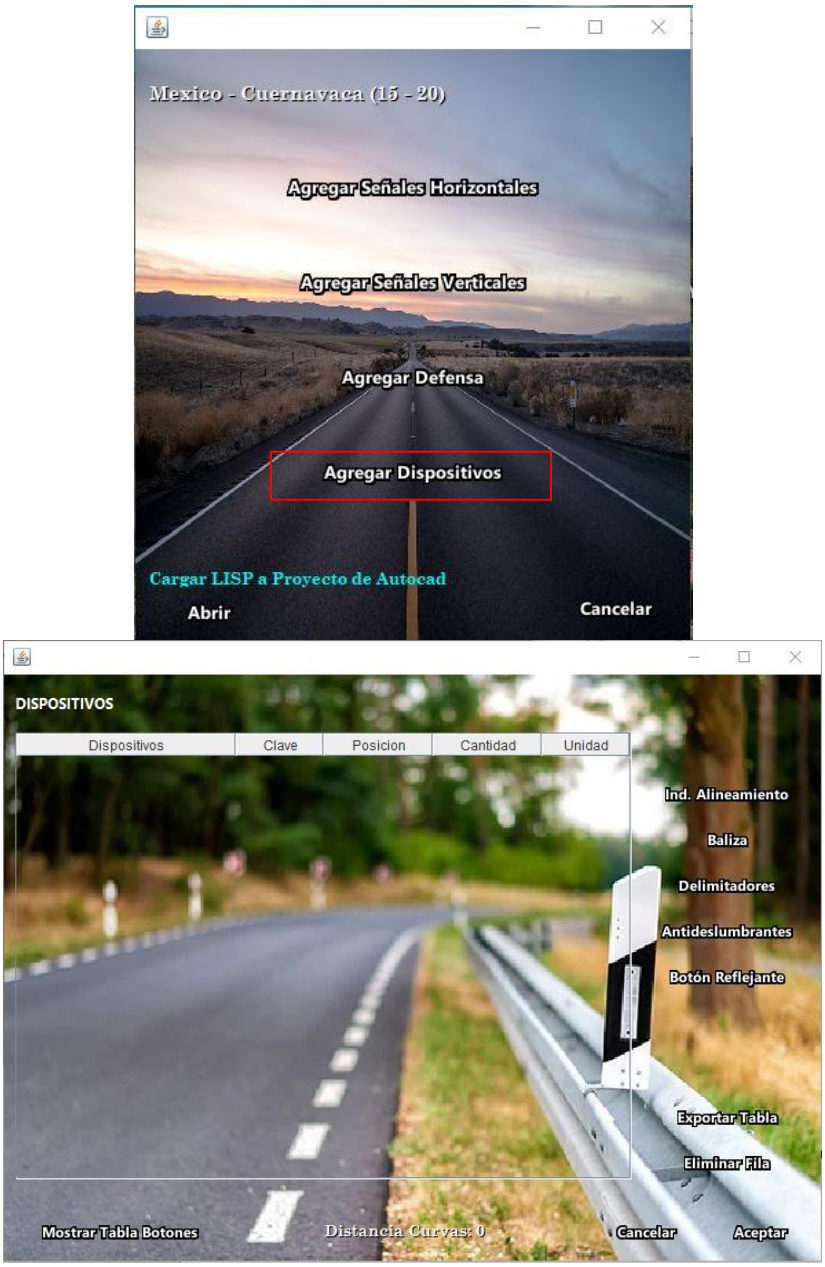


Figura 6.109. Aplicación: Dispositivos

- **Indicadores de Alineamiento**

La aplicación incluye un botón para agregar indicadores de alineamiento en dos situaciones diferentes: tangentes y curvas.

- **Tangentes**: al presionar el botón aparece un mensaje de confirmación donde se solicita verificar si existen tangentes con indicadores. En caso afirmativo, se muestran las

instrucciones para usar el comando GuardarLongitudes en AutoCAD y seleccionar la línea correspondiente. Con la distancia obtenida, la aplicación calcula automáticamente el número de indicadores necesarios, considerando una separación de 40 m, y agrega el resultado en la tabla de dispositivos.

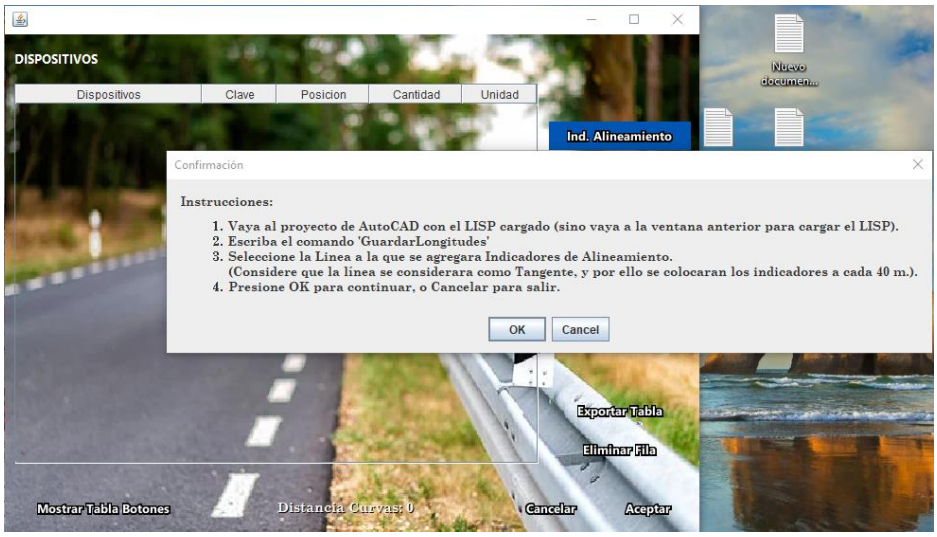


Figura 6.110. Aplicación: Indicadores de Alineamiento Tangentes

- **Curvas**: si el usuario confirma que existen curvas con indicadores, se despliega un conjunto de pasos que guían en el uso del comando DistanciaCurva en AutoCAD. Posteriormente, para cada curva seleccionada, la aplicación solicita el grado de curvatura y realiza el cálculo de los indicadores necesarios según la fórmula de separación establecida. Los resultados se registran de manera organizada en la tabla.

Figura 6.111. Aplicación: Agregando indicadores de alineamiento Curva

- **Balizas**

El botón de balizas permite relacionar estos dispositivos con las marcas de pintura M-5, que previamente deben de estar cargadas en el señalamiento horizontal.

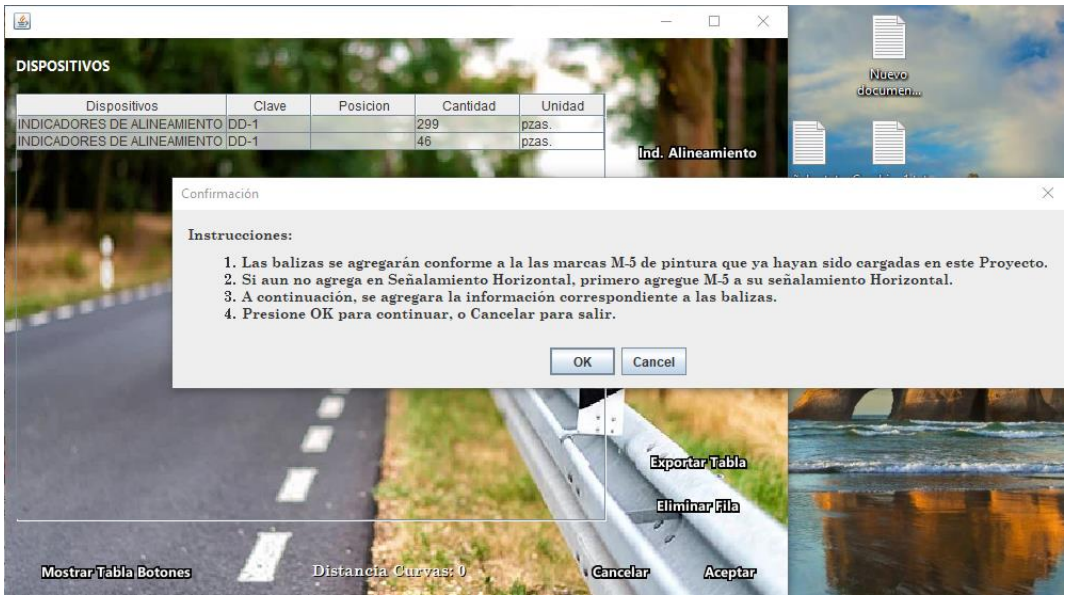


Figura 6.112. Aplicación: Balizas

El sistema revisa si existen registros de M-5; en caso de encontrarlos, calcula la cantidad de balizas necesarias con base en la separación correspondiente, que varía según el tipo de camino (por ejemplo, 2 m en carreteras y 0.5 m en vías ciclistas). Si no se detectan marcas M-5, se muestra una advertencia para que el usuario las agregue antes de continuar.

Figura 6.113. Aplicación: Agregando balizas

- **Delimitadores**

En el caso de los delimitadores para confinamiento, la aplicación depende de la existencia de marcas de pintura M-2.2 o M-1.4.

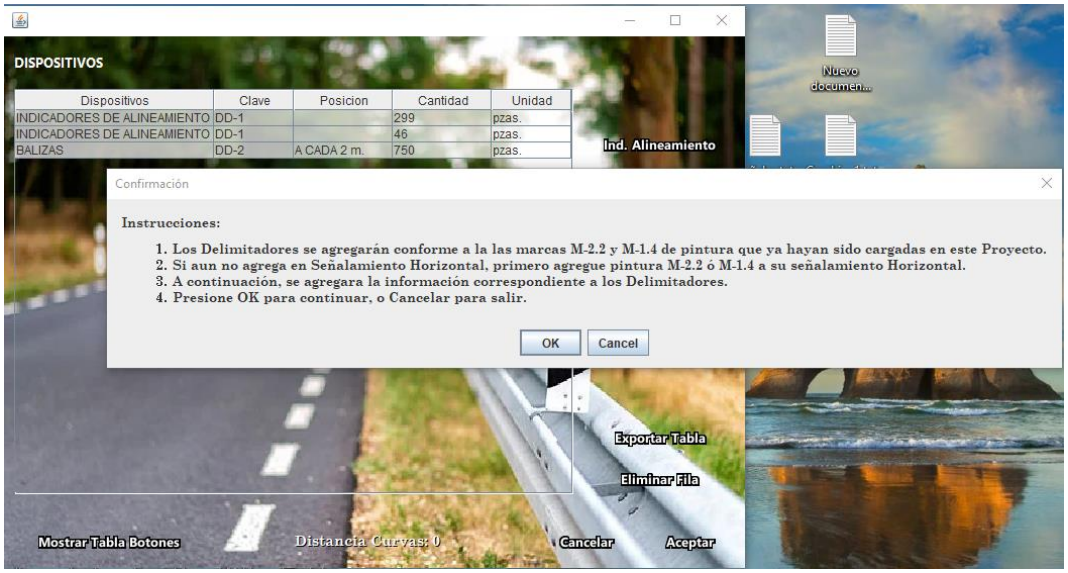


Figura 6.114. Aplicación: Delimitadores

Una vez localizadas estas marcas en el proyecto, se calcula el total de delimitadores con base en una separación de 3.8 m. El resultado se registra en la tabla con su clasificación normativa correspondiente. Si no se encuentran las marcas requeridas, se notifica al usuario mediante un mensaje de advertencia.

Figura 6.115. Aplicación: Agregando delimitadores

- **Dispositivos Antideslumbrantes**

Este botón permite la colocación de malla o valla antideslumbrante sobre las barreras centrales.

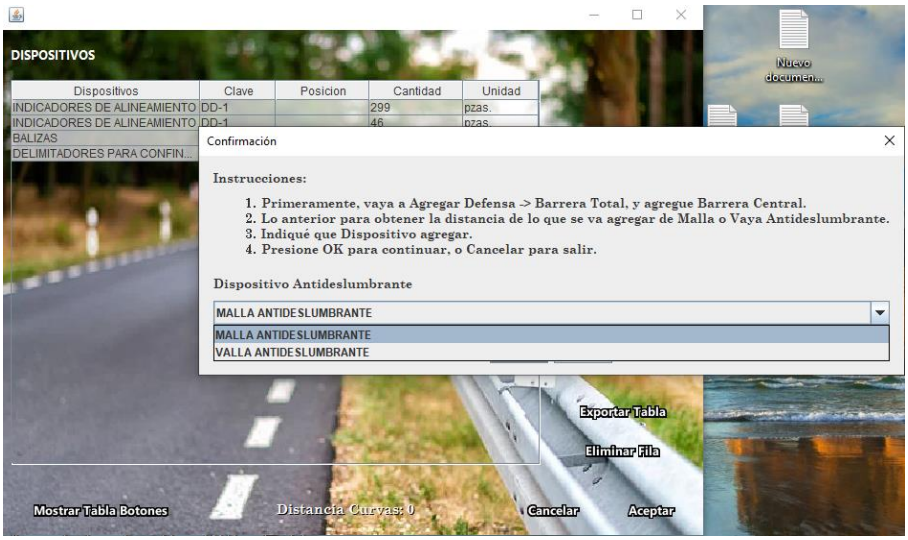


Figura 6.116. Aplicación: Dispositivos Antideslumbrantes

Las instrucciones iniciales indican que primero debe existir una barrera central tipo New Jersey registrada en el apartado de Barreras. Una vez confirmada, el usuario selecciona el tipo de dispositivo antideslumbrante. La aplicación calcula la cantidad total, ya sea en piezas (valla) o

metros lineales (malla) y adicionalmente agrega de forma automática los botones reflejantes para la barrera central, con una separación de 10 m a ambos lados.

Figura 6.117. Aplicación: Agregando dispositivos antideslumbrantes

- **Botón Reflejante**

El botón destinado a los botones reflejantes calcula este dispositivo en función de la longitud total de defensa metálica previamente cargadas en la Defensa.

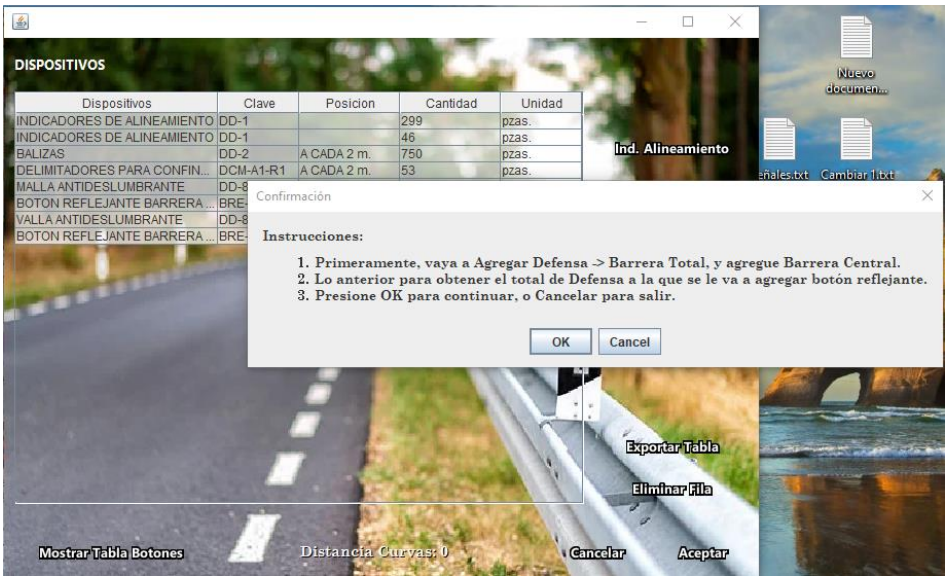


Figura 6.118. Aplicación: Botón Reflejante

La aplicación divide la suma de las distancias entre 10 para obtener el número total de botones requeridos. En caso de que el nivel de contención registrado sea NC-3, la cantidad se duplica automáticamente, ya que este nivel exige una mayor densidad de dispositivos. El resultado se refleja en la tabla con la clave normativa correspondiente.

Figura 6.119. Aplicación: Agregando botón reflejante

- **Mostrar Tabla Botones**

La aplicación ofrece la posibilidad de alternar entre dos tablas distintas: una destinada a los dispositivos y otra a los botones. Para ello, se dispone de un Botón que, al activarse o desactivarse, modifica el contenido mostrado en pantalla.

- **Vista de Botones**: al activar el interruptor, la tabla central adopta el modelo los botones se ajustan automáticamente los anchos de las columnas para esta configuración. El título de la ventana se actualiza a **"**BOTONES**"**, y se habilitan únicamente los botones relacionados con esta categoría, como *Agregar Botones* y *Agregar Curvas*. Asimismo, se actualiza la imagen de fondo para reforzar visualmente el cambio de contexto.

- **Vista de Dispositivos**: al desactivar el interruptor, que como esta al entra a la ventana, la tabla regresa a su modelo de Dispositivos. El título cambia nuevamente a **"**DISPOSITIVOS**"**, y se muestran los botones vinculados a esta categoría, tales como *Indicadores de Alineamiento*, *Balizas*, *Delimitadores*, *Dispositivos Antideslumbrantes* y *Botón Reflejante*. De igual forma, se actualiza la imagen de fondo para representar esta sección.

Figura 6.120. Aplicación: Vista de Botones

- Agregar Curvas

En la sección de botones, la aplicación incluye la opción **Agregar Curvas**, la cual permite registrar la distancia acumulada de los tramos curvos de un proyecto.

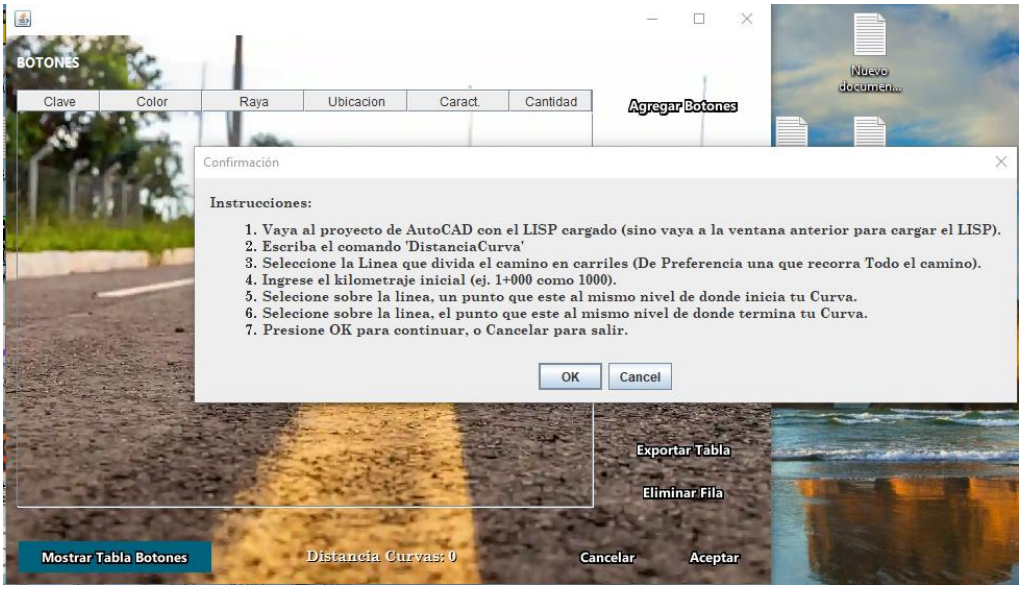


Figura 6.121. Aplicación: Agregar Curvas

Al presionarlo, se despliega un cuadro con las instrucciones que indican cómo obtener la información directamente desde AutoCAD utilizando el comando DistanciaCurva. El usuario debe seleccionar la línea que divide los carriles e indicar los puntos de inicio y fin de cada curva para generar los datos necesarios.

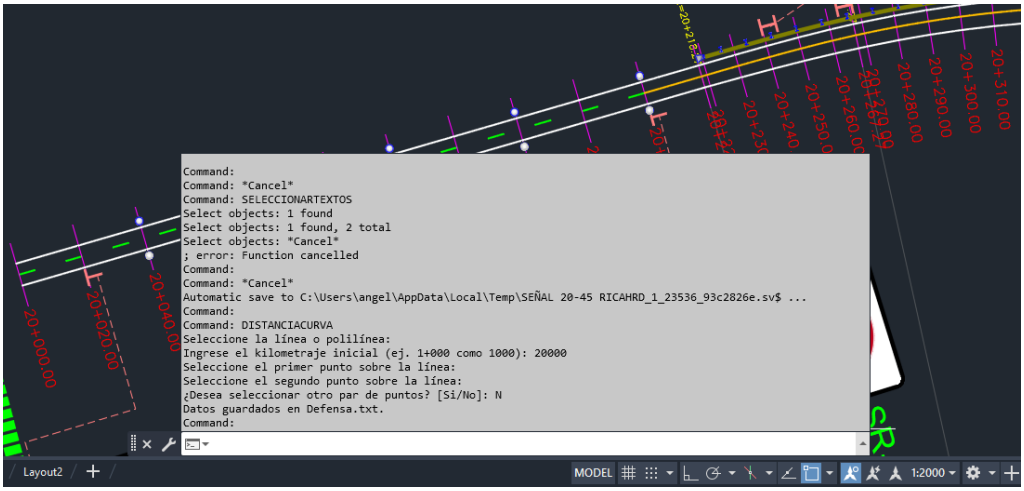


Figura 6.122. Aplicación: DistanciaCurva

Finalmente, la información obtenida se refleja automáticamente en la interfaz mediante la etiqueta "DistanciaCurva" donde muestra la distancia correspondiente a curvas del tramo que se está trabajando, permitiendo al usuario verificar en todo momento el valor actualizado conforme se vayan agregando más Botones.

- **Agregar Botones**

El botón "Agregar Botones" está diseñado para calcular y registrar la cantidad de botones reflejantes necesarios en el proyecto, tomando como base las marcas de pintura del señalamiento Horizontal previamente cargadas.

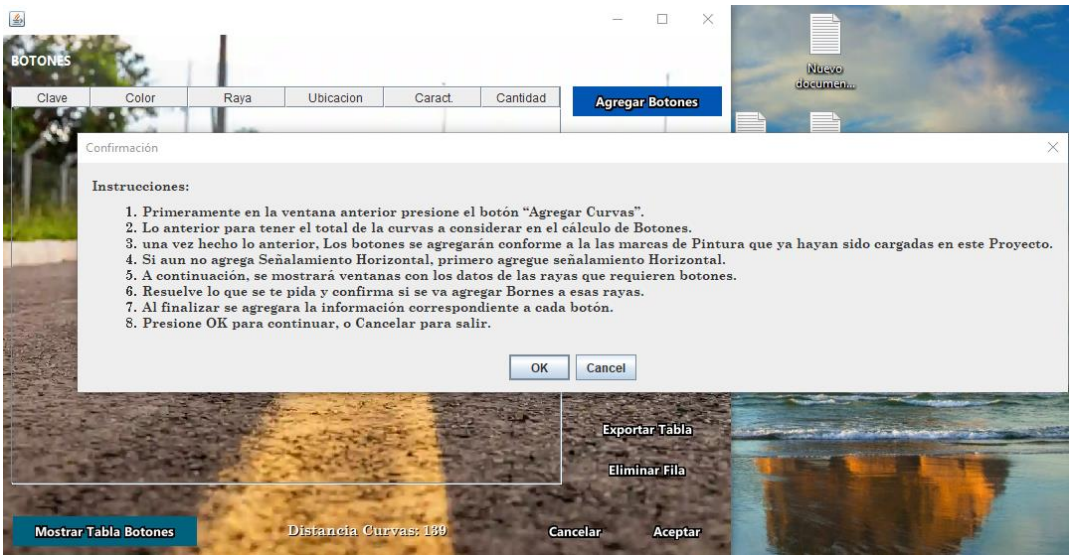


Figura 6.123. Aplicación: Agregar Botones

Para su uso, el sistema guía al usuario mediante una serie de pasos interactivos.

- Antes de usar esta función, es necesario haber utilizado el botón "Agregar Curvas", ya que el cálculo de botones toma en cuenta la distancia de los tramos con curvas y los tramos rectos.
- El sistema validará que existan marcas de señalamiento horizontal cargadas en el proyecto. En caso contrario, se mostrará una advertencia indicando que primero debe registrar dichas marcas.
- Una vez confirmados los datos, aparecerán ventanas de verificación para cada marca de pintura que requiera la instalación de botones.
- El usuario deberá confirmar en cada caso si desea aplicar botones a la señal correspondiente.

- En determinadas clasificaciones, el sistema solicitará información adicional (por ejemplo, ancho del camino o color de la raya) para calcular de manera precisa la cantidad de botones a colocar.
- Finalmente, los resultados se incorporarán automáticamente a la tabla de botones, mostrando la clave, el color, la clasificación, la ubicación, las características y la cantidad calculada para cada marca.

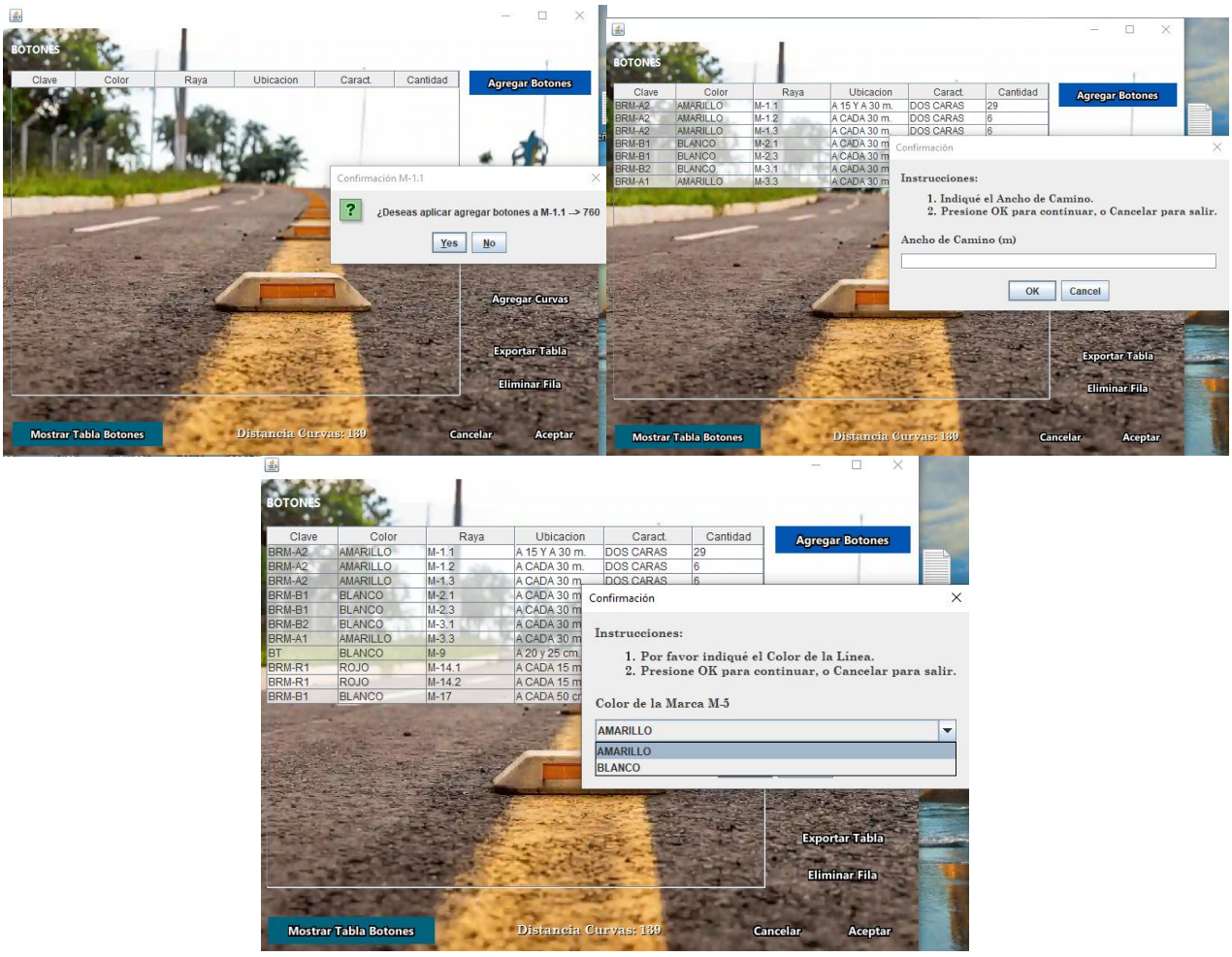


Figura 6.124. Aplicación: Confirmación de Botones

Una vez que el usuario confirme que tanto para la tabla de Dispositivos y para la tabla de Botones ya todos se han agregado, entonces se presión Aceptar para guardar los datos en la base de datos.

Exportar Tablas de Dispositivos y Botones

Una vez que se este seguro que los datos ya están completos y se quieran pasar a AutoCAD, entonces se presiona el Botón de Exportar Tabla, el cual abrirá una nueva ventana donde se mostrará una tabla con los datos que ya se guardaron en la base de datos.

Figura 6.125. Aplicación: Exportar Tablas

Por defecto, se muestra la tabla de dispositivos (con columnas como dispositivo, clave, posición, cantidad y unidad).

Mediante el botón Mostrar Tabla Botones, el usuario puede alternar la visualización para consultar los botones reflejantes (con sus datos de clave, color, raya, ubicación, características y cantidad).

Al presionar Aceptar, el sistema solicita confirmación para exportar primero la tabla de dispositivos y luego la de botones. Una vez completada la exportación, el sistema indica al usuario el comando que debe ejecutar en AutoCAD tras cargar el LISP: CrearTablaDispositivos para dispositivos y CrearTablaBotones para botones.

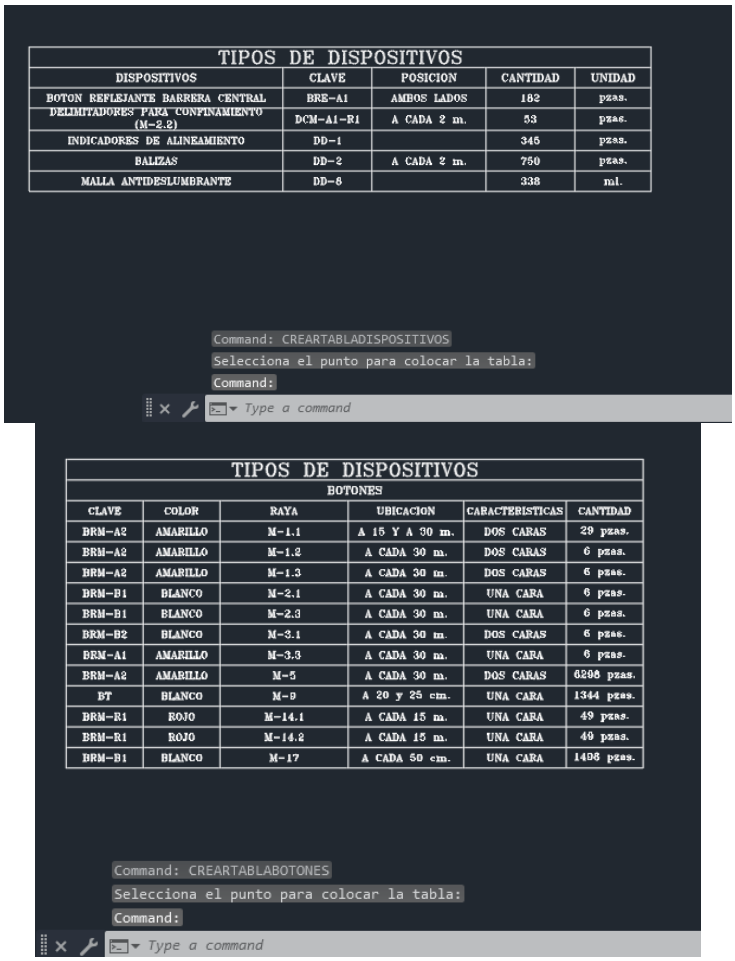


Figura 6.126. Aplicación: Exportación Tablas Dispositivos y Botones AutoCAD

7. Resultados, Impacto y Conclusiones

El desarrollo de la aplicación permitió cumplir con el objetivo principal de esta tesis: proporcionar a los ingenieros civiles en México una herramienta informática que facilite el diseño y cálculo de señalización vial conforme a las directrices del *Manual de Señalización y Dispositivos para el Control del Tránsito en Calles y Carreteras* (edición 2023).

7.1. Resultados obtenidos

1) Señalización Horizontal

- **Objetivo:** Verificar que el sistema cuantifica 20 marcas de pintura en metros lineales y exporta tabla editable a AutoCAD.
- **Pasos de verificación:**
    - Que se pude acceder a las 20 marcas de pintura y que los botones dentro de cada ventana muestren instrucciones personalizadas a la marca.
    - Que dentro de AutoCAD se puedan usar los comandos solicitados por la aplicación.
    - Revisar tabla y que se agreguen filas correctamente en la aplicación y que se pueda ver la tabla antes de exportar a AutoCAD.
    - En AutoCAD, abrir la tabla exportada y verificar que los campos son editables (no es una imagen).
- **Capturas:**

  Primeramente, tenemos 20 marcas de señalamiento Horizontal para ello tenemos una ventana con 20 posibles elecciones para el usuario pueda elegir la marca agregar

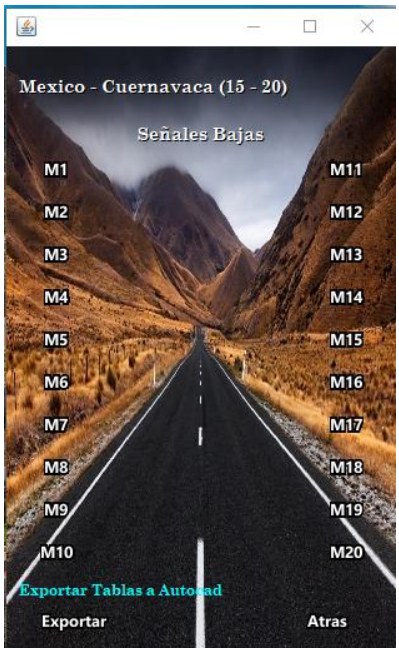


Figura 7.1. Señalamiento Horizontal (Elección)

Para cada botón de la ventana anterior se abre una ventana correspondiente a la marca seleccionada donde podremos agregar las submarcas correspondientes.

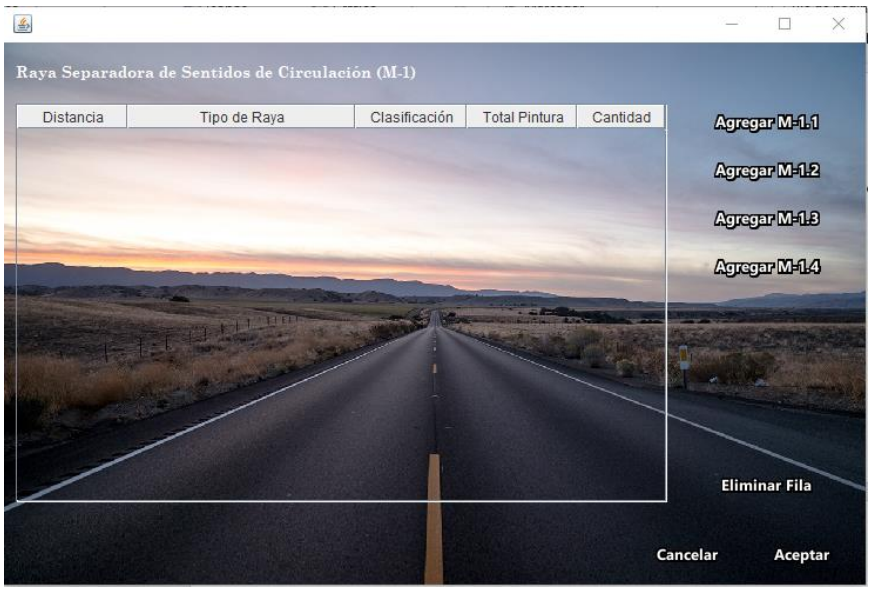


Figura 7.2. Marca M-1

Cada botón de las submarcas tiene instrucciones que le indican al usuario que hacer para agregar la marca a la tabla, para ello se pide que use un comando en AutoCAD, se comprobó que los comandos solicitados funcionen dentro de AutoCAD y que sea se guarden los datos y que esto datos los lea la aplicación.

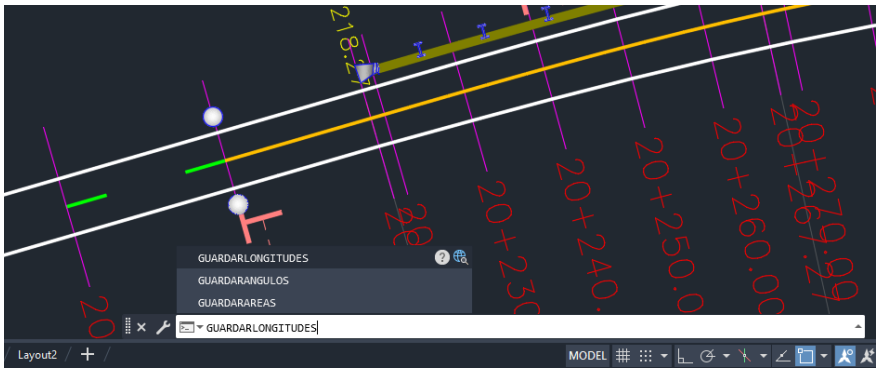


Figura 7.3. Guardar Longitudes AutoCAD

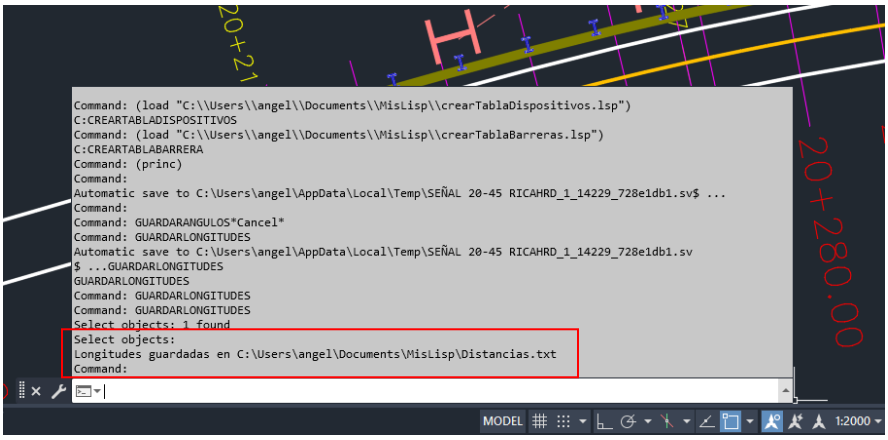


Figura 7.4. Confirmación AutoCAD

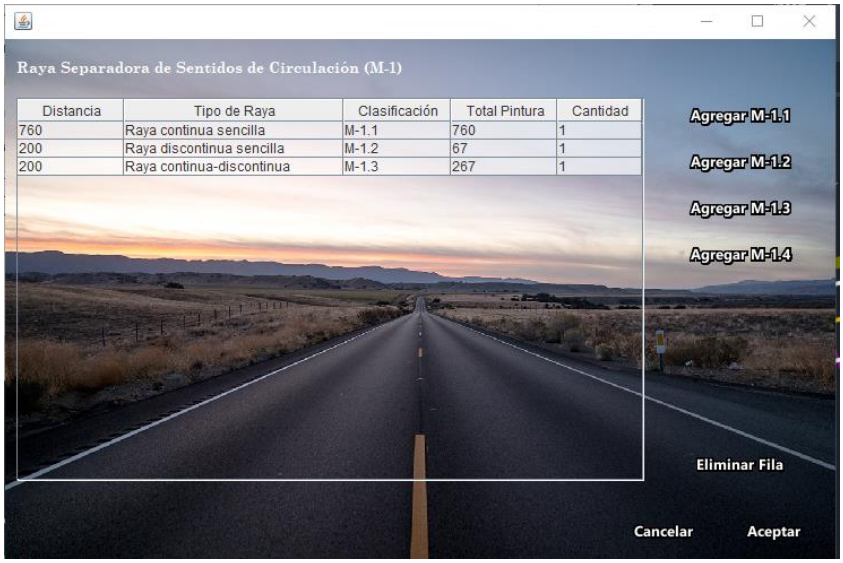


Figura 7.5. Agregando marcas M-1.1, M-1.2 y M-1.3

Cada botón debe de realizar su cálculo correspondiente y mostrar su total de pintura en la tabla. Entonces tenemos una ventana para cada marca.

- **M2**

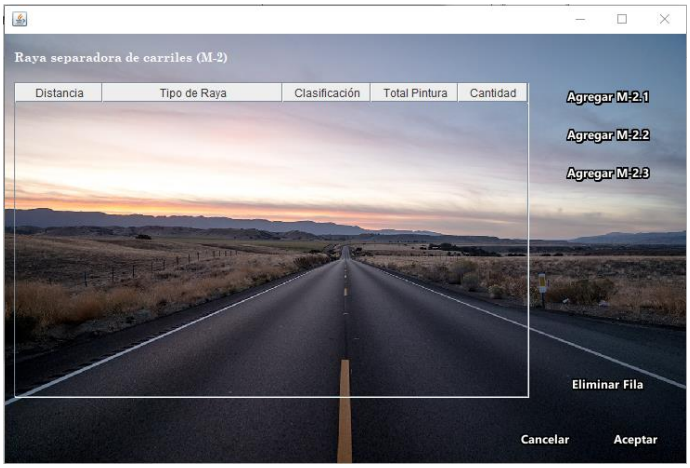


Figura 7.6. M-2

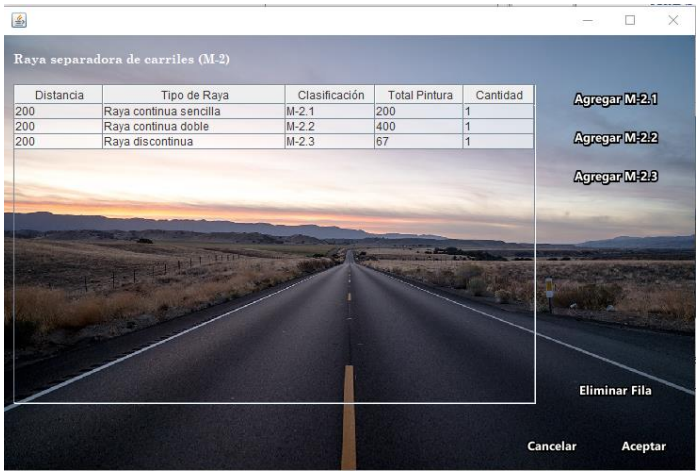
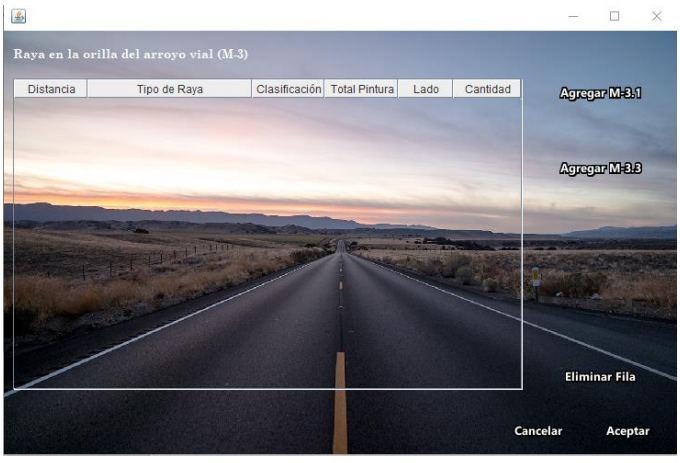


Figura 7.7. Agregando marcas M-2

- **M3**



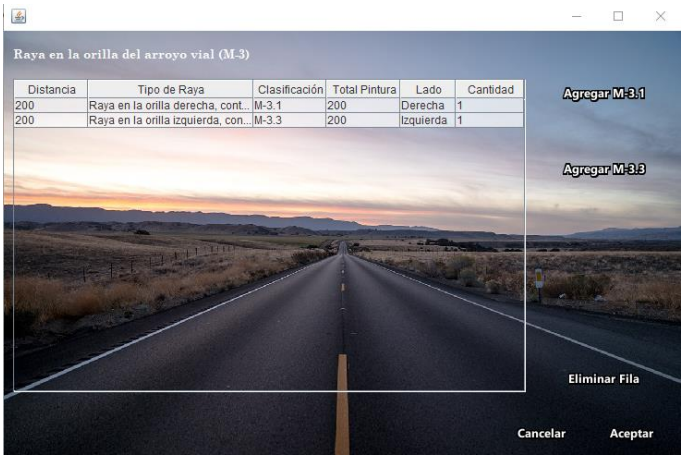


Figura 7.8. Aplicación: Agregando marcas M-3

- **M4**

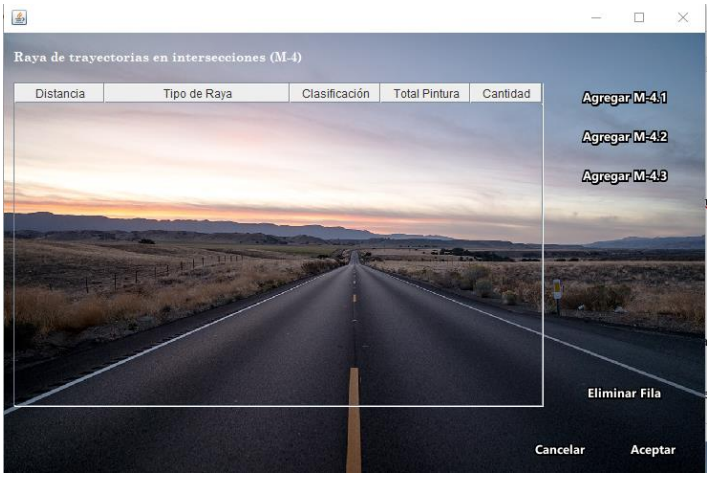


Figura 7.9. M-4

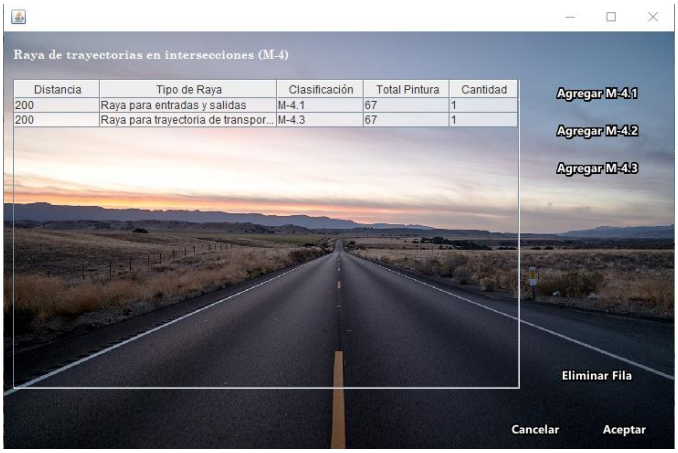


Figura 7.10. Aplicación: Agregando marcas M-4

- **M5**

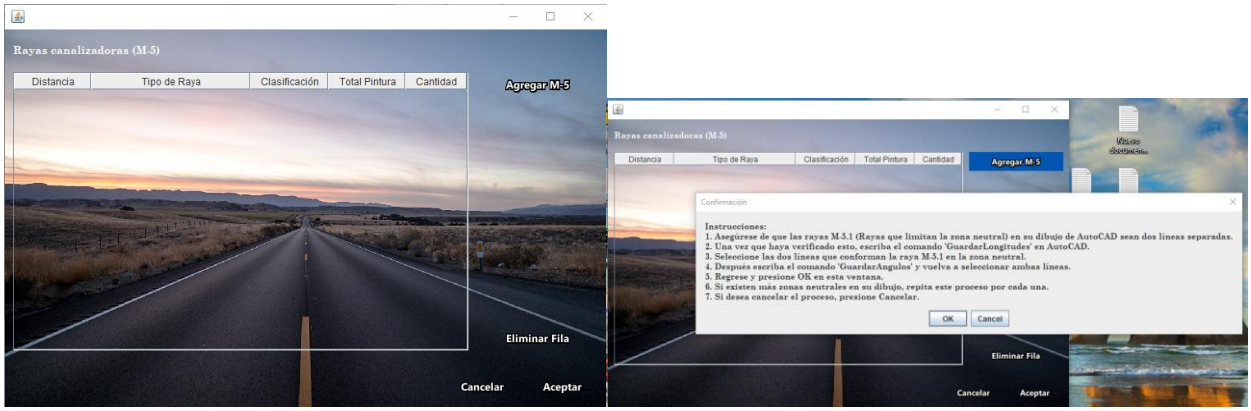


Figura 7.11. M-5

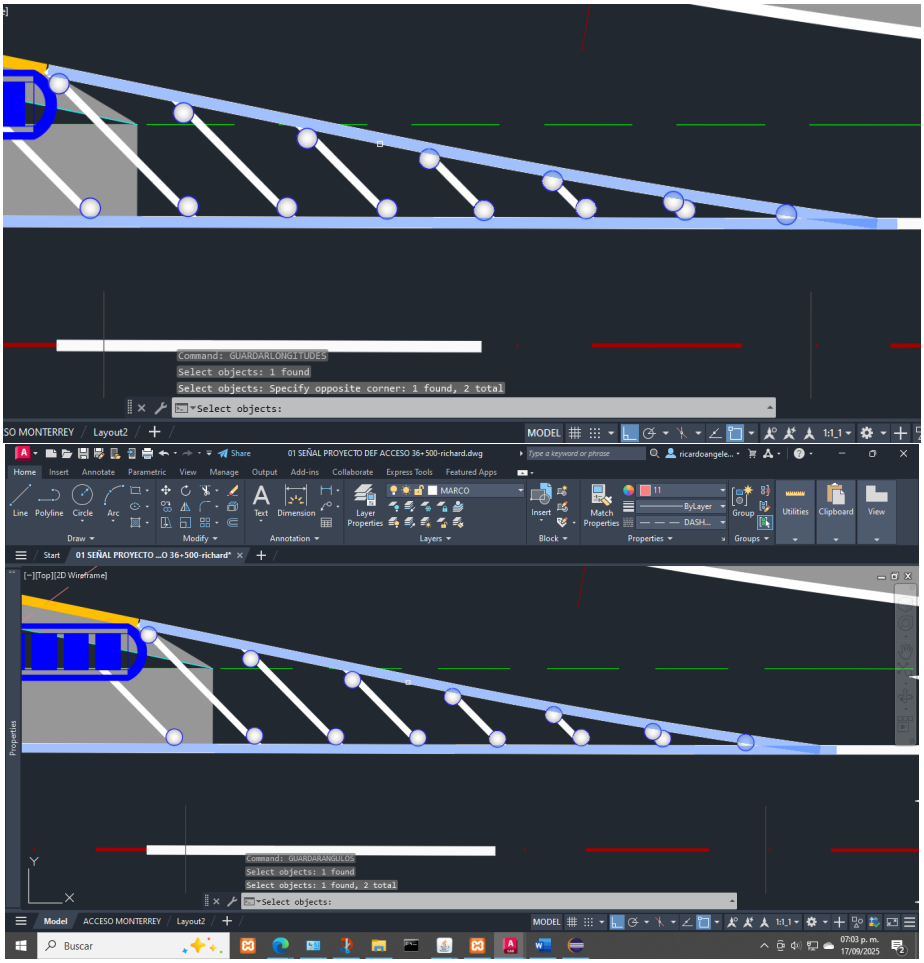


Figura 7.12. Usando GuardarLongitudes y GuardarAngulos

- **M6**

Figura 7.13. M-6

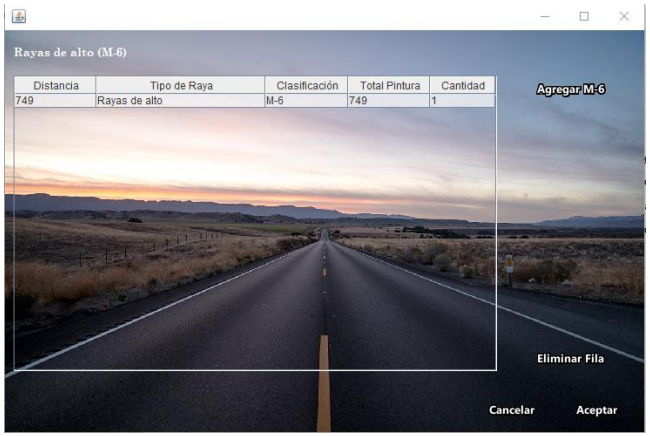


Figura 7.14. Agregando marcas M-6

- **M7**

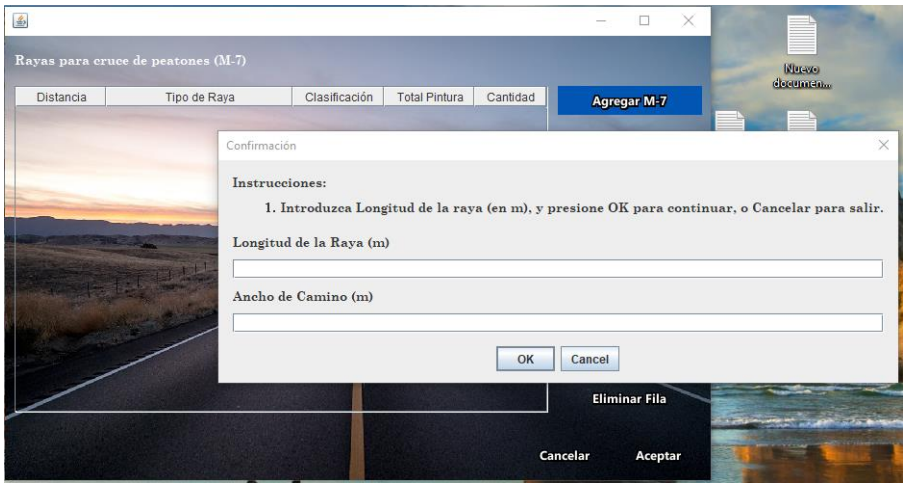


Figura 7.15. M-7

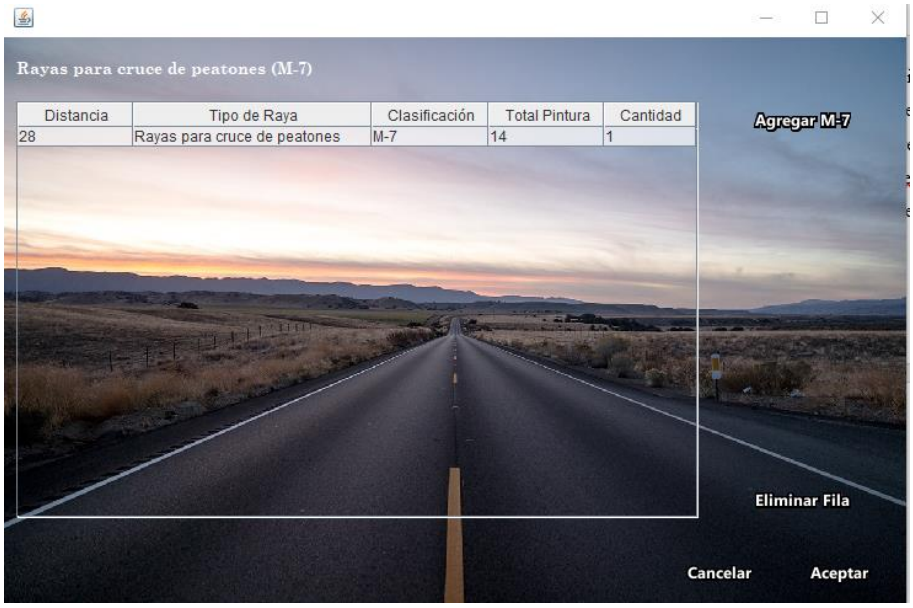


Figura 7.16. Agregando marcas M-7

- **M8**

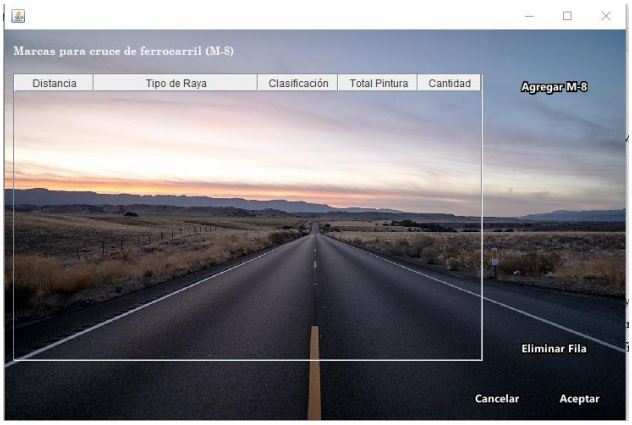


Figura 7.17. M-8

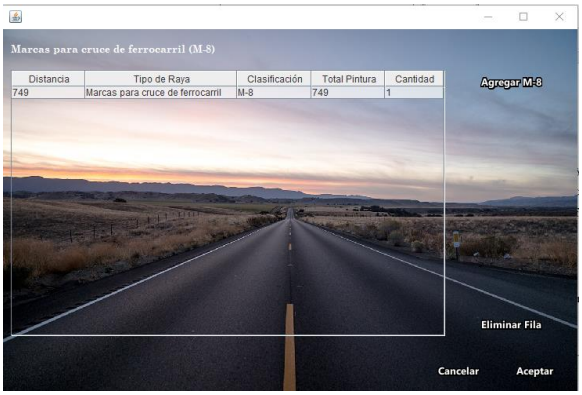
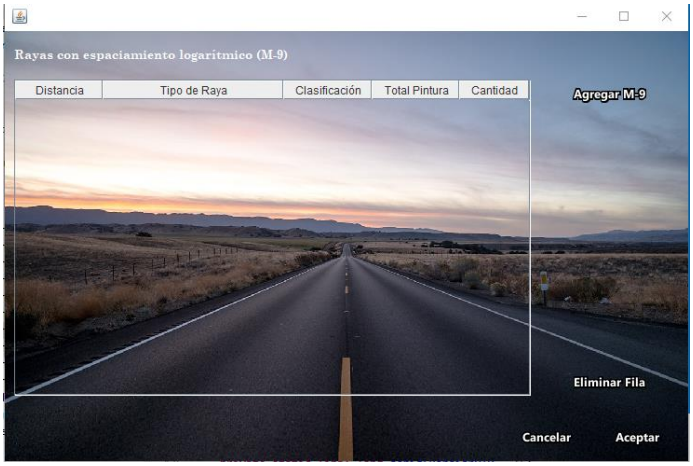


Figura 7.18. Agregando marcas M-8

- **M9**



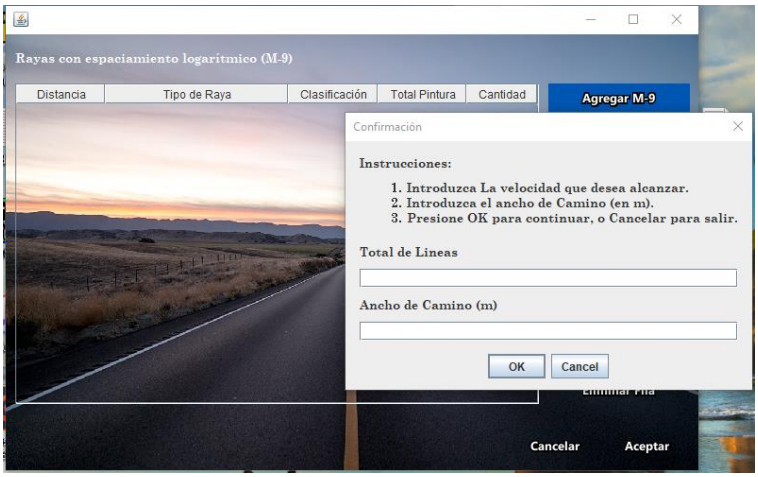


Figura 7.19. M-9

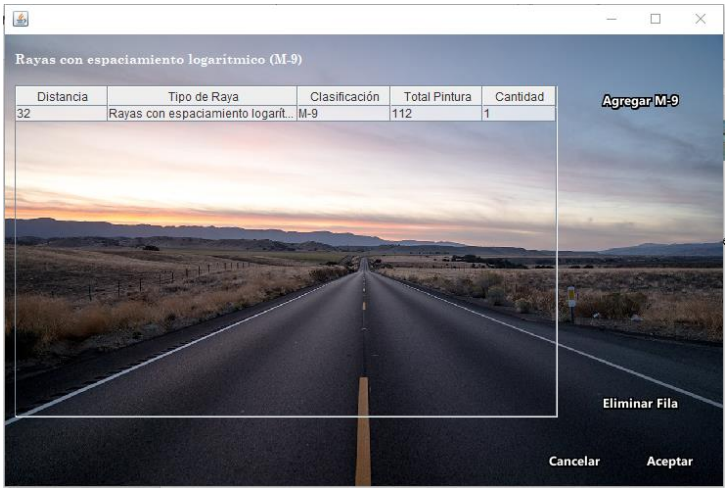


Figura 7.20. Agregando marcas M-9

- **M10**

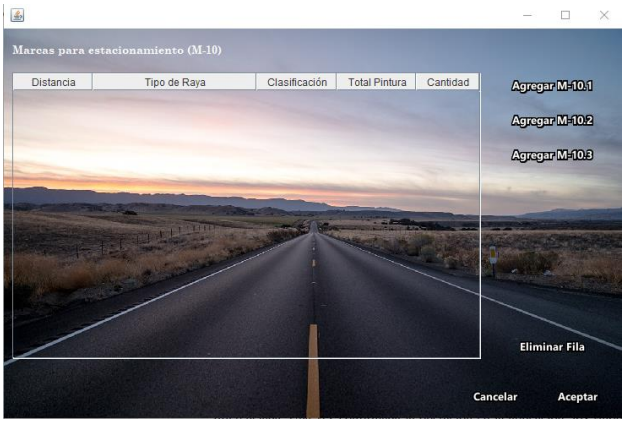


Figura 7.21. M-10

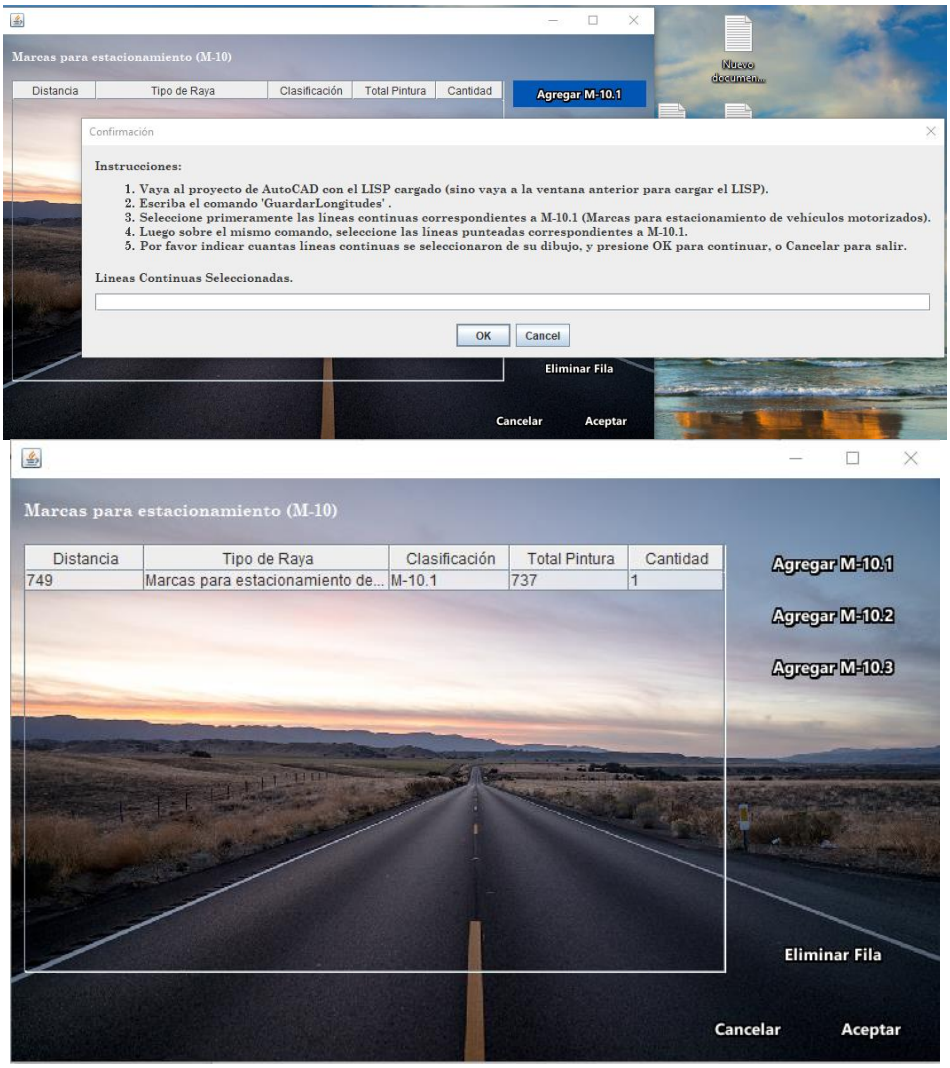
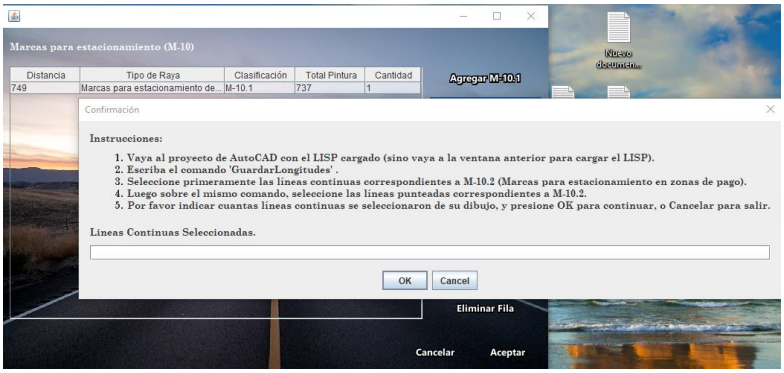


Figura 7.22. Agregando marcas M-10.1



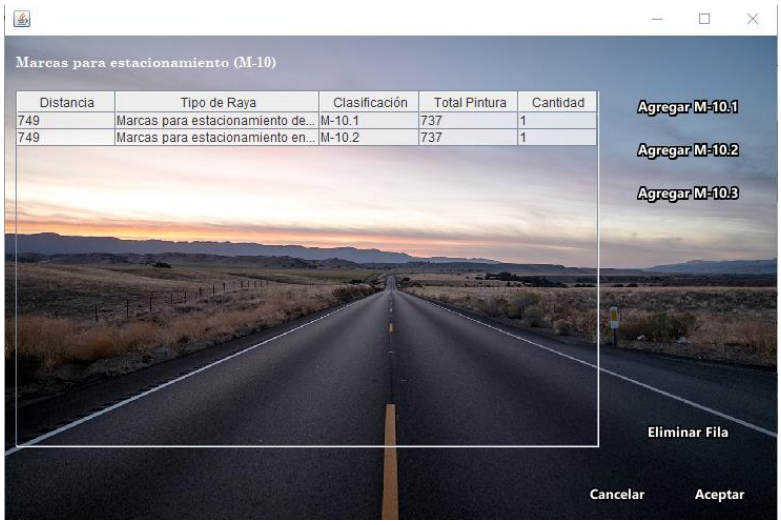


Figura 7.23. Agregando marcas M-10.2

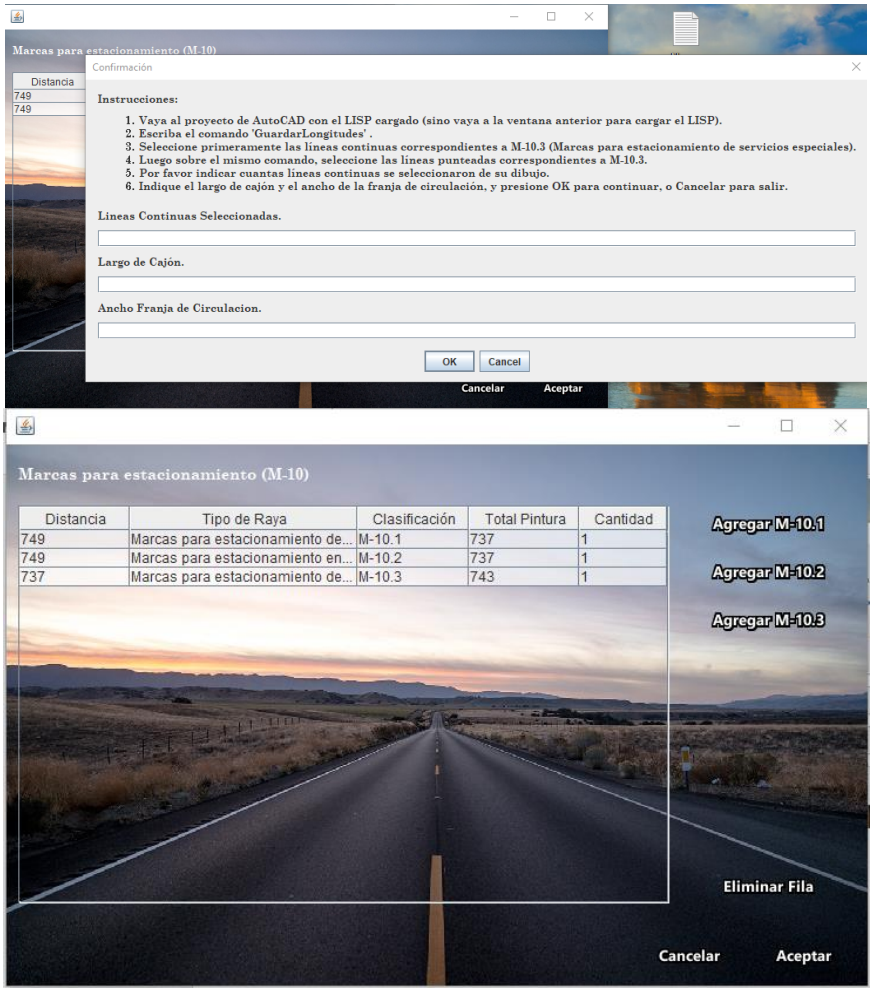


Figura 7.24. Agregando marcas M-10.3

- **M11**

Figura 7.25. M-11

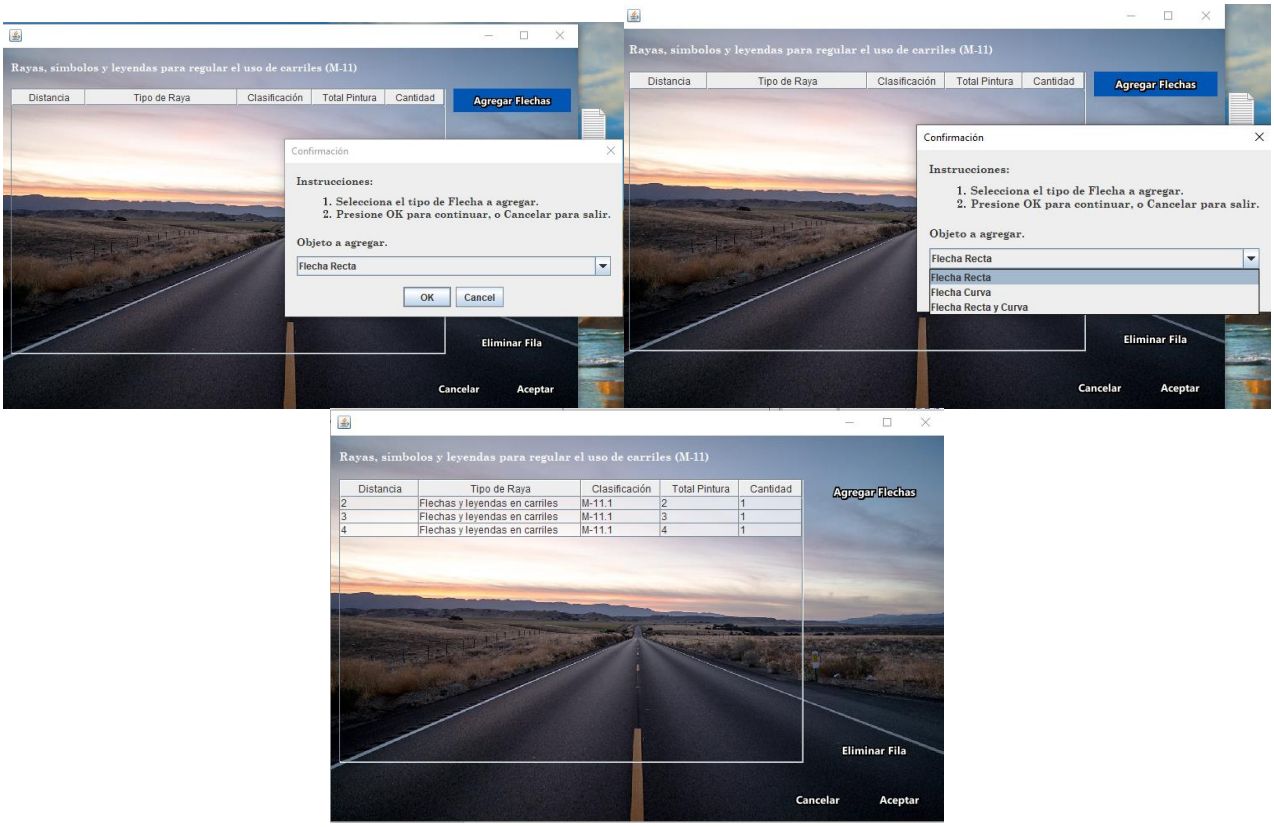


Figura 7.26. Agregando flechas M-11

- **M12**

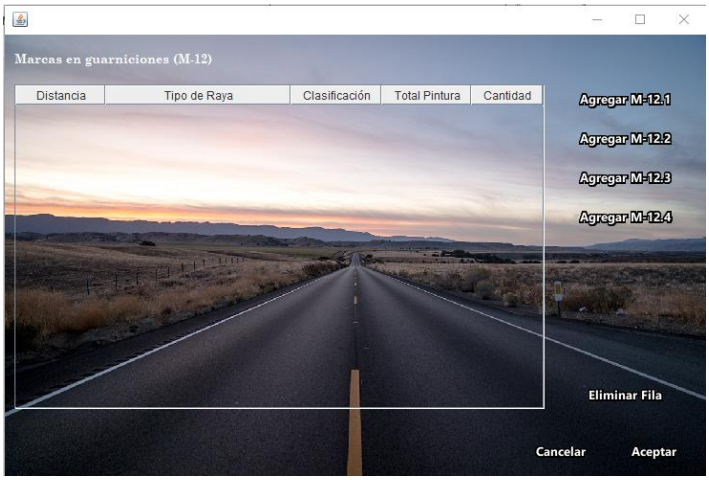


Figura 7.27. M-12

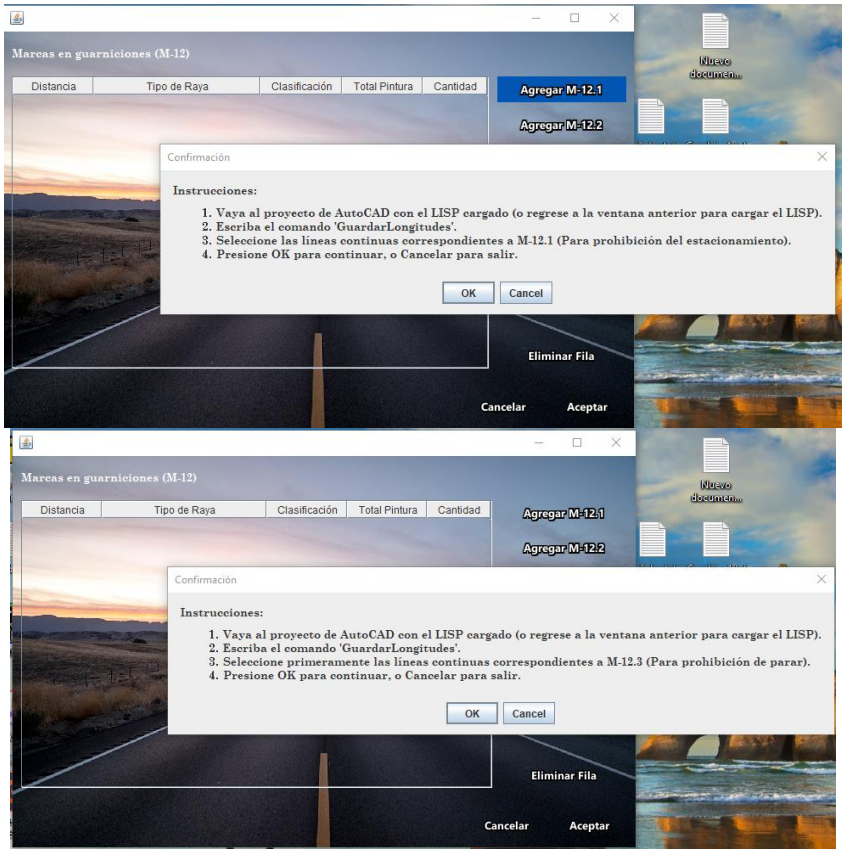


Figura 7.28. Agregando marcas M-12.1 y M-12.3

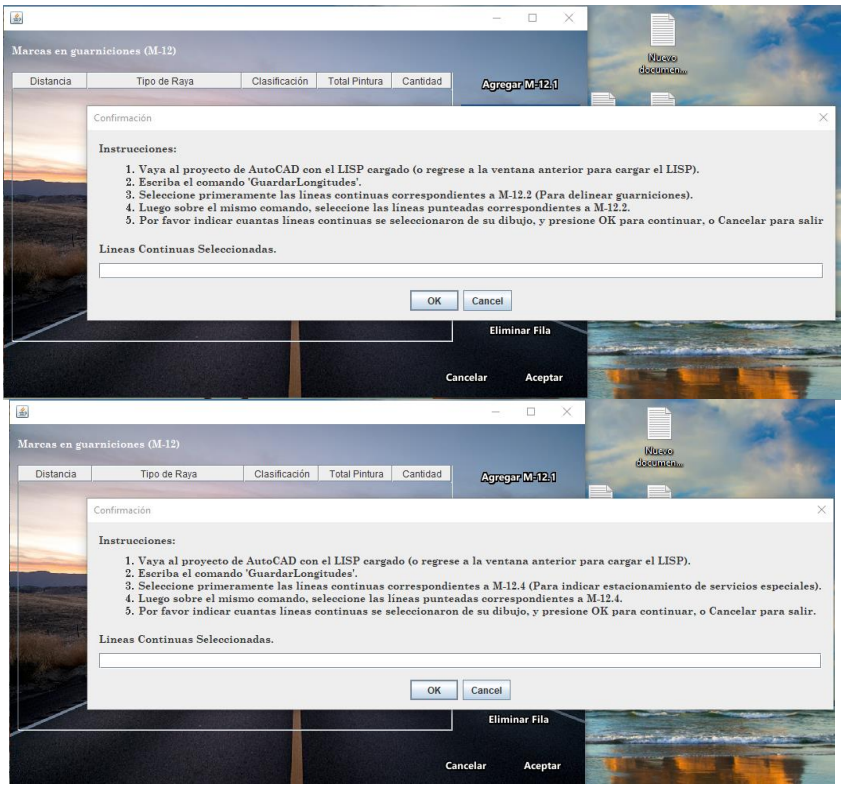


Figura 7.29. Agregando marcas M-12.2 y M-12.4

Figura 7.30. Agregando marcas M-12

- **M13**

Figura 7.31. M-13

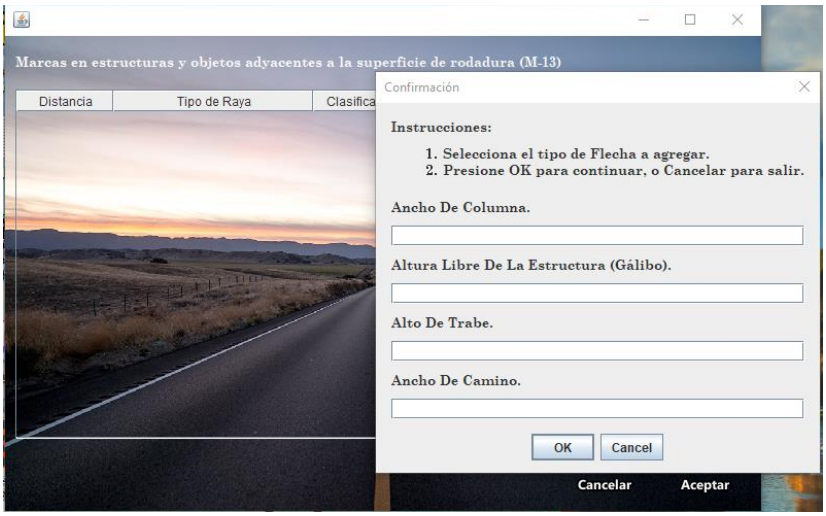


Figura 7.32. Agregando marcas M-13.1

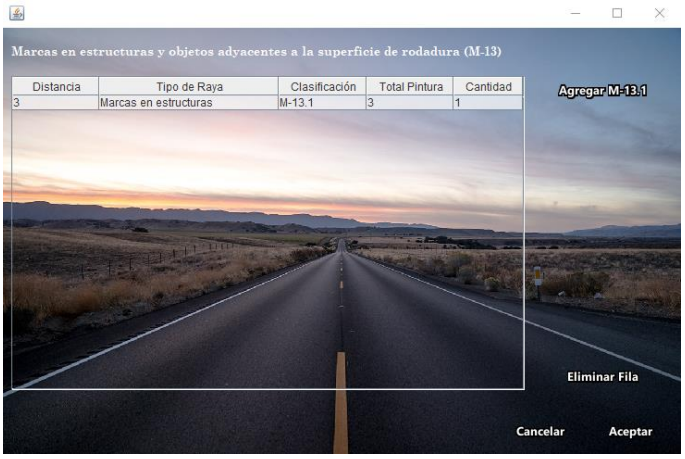


Figura 7.33. Marca M-13.1 agregada

- **M14**

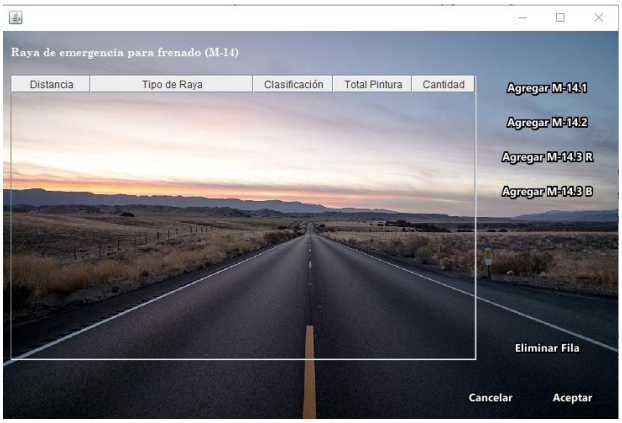


Figura 7.34. M-14

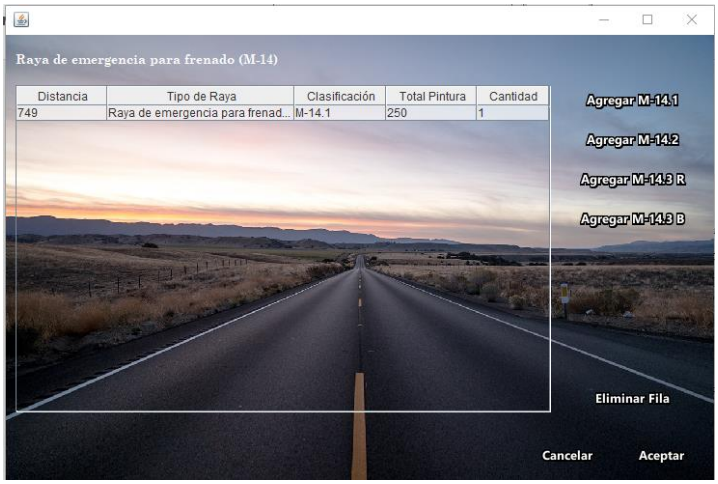


Figura 7.35. Agregando marcas M-14.1

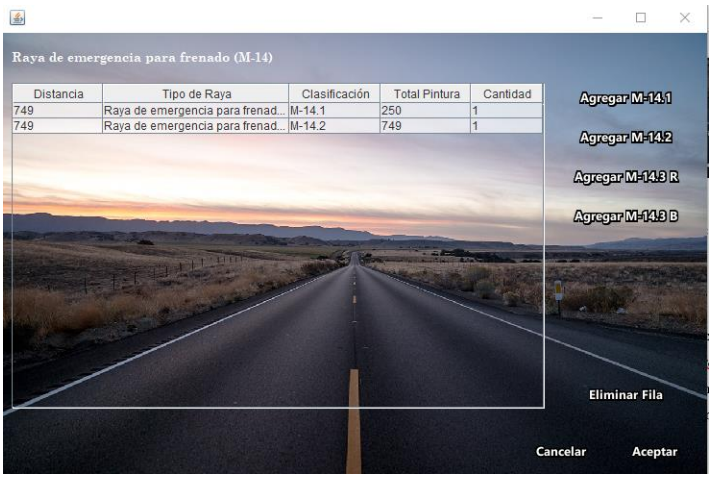


Figura 7.36. Agregando marcas M-14.2

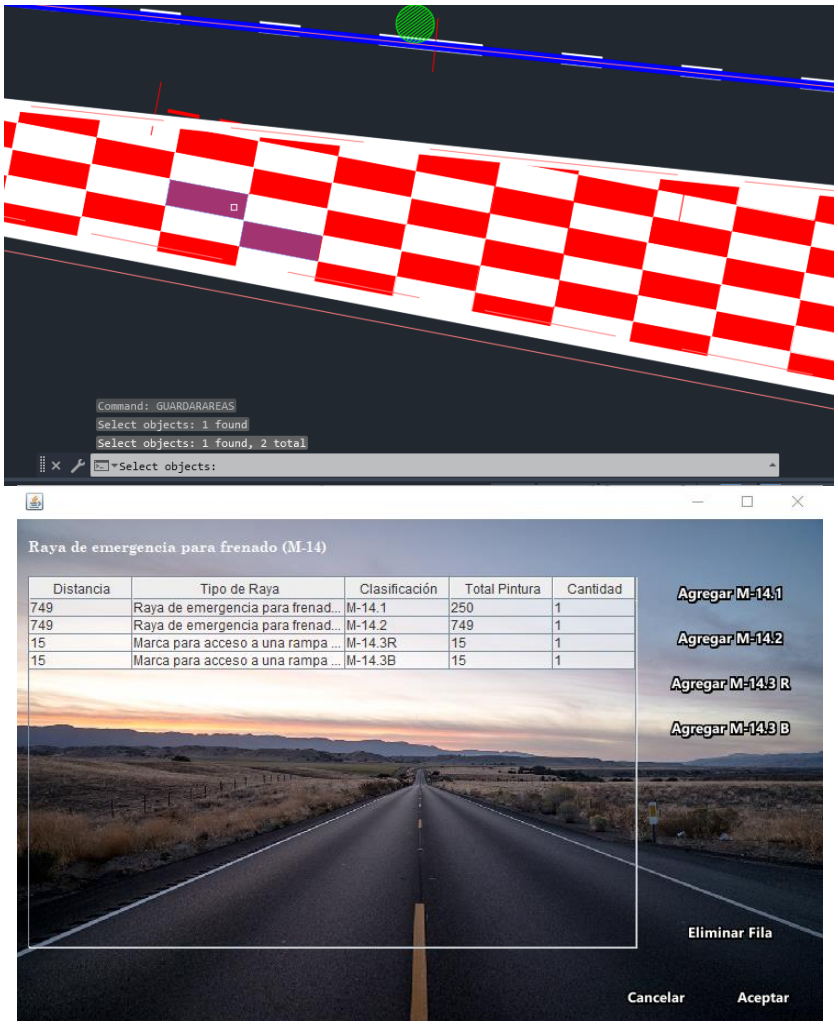


Figura 7.37. Agregando áreas M-14.3

- **M15**

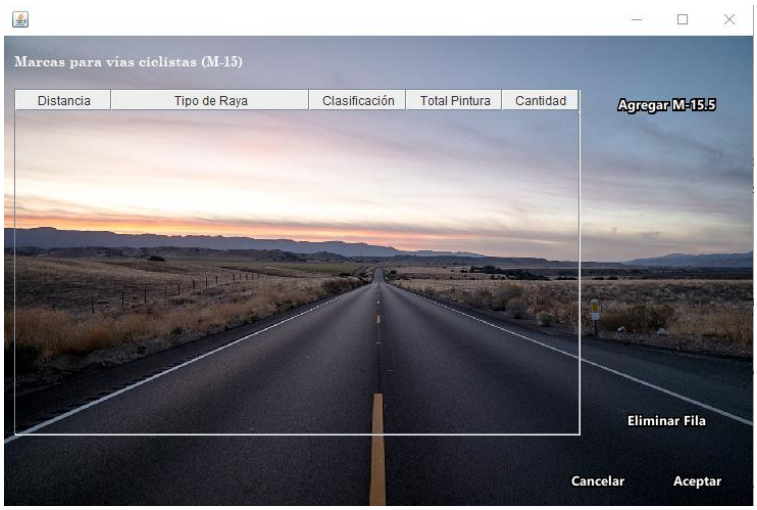


Figura 7.38. M-15

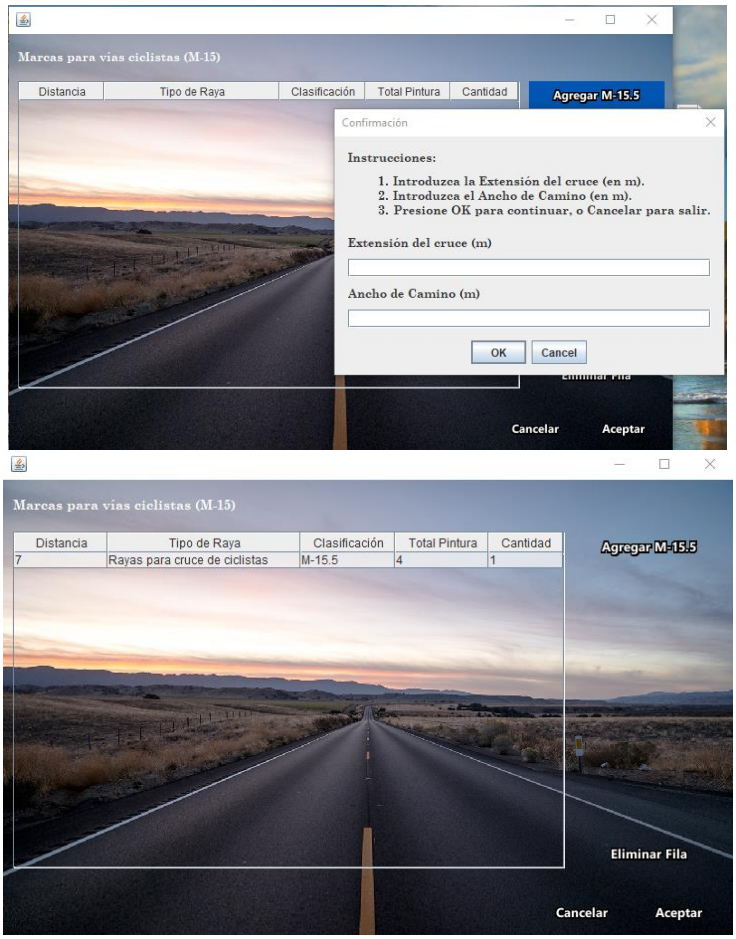


Figura 7.39. Agregando marcas M-15.5

- **M16**

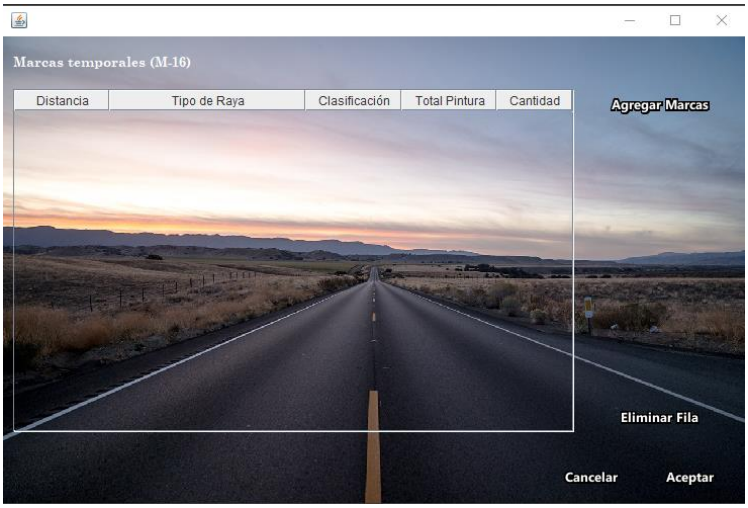


Figura 7.40. M-16

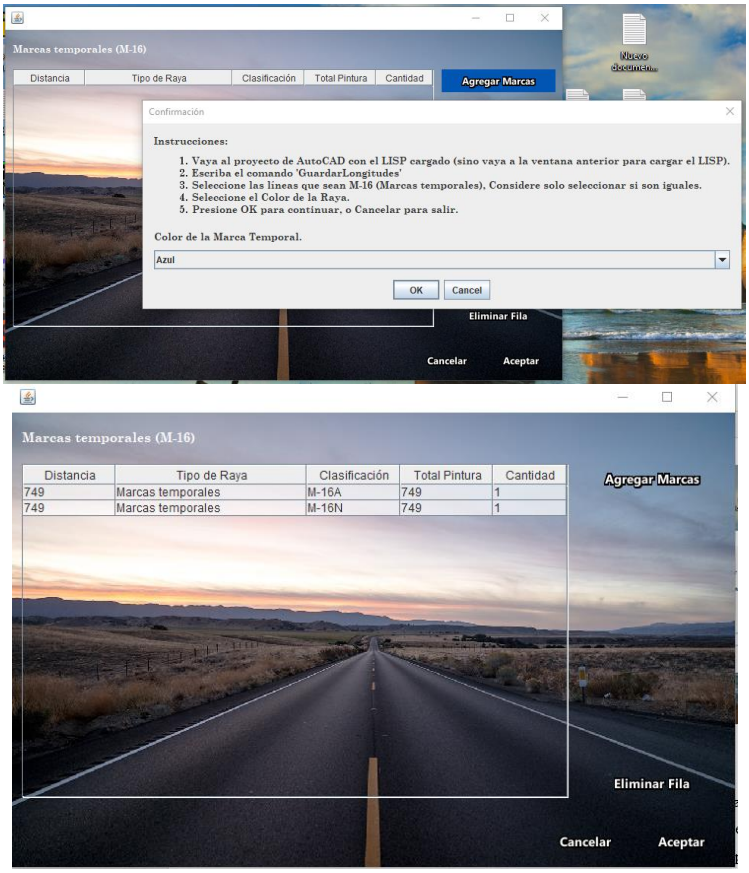


Figura 7.41. Agregando marcas M-16

- **M17**

Figura 7.42. M-17

Figura 7.43. Agregando marcas M-17

- **M18**

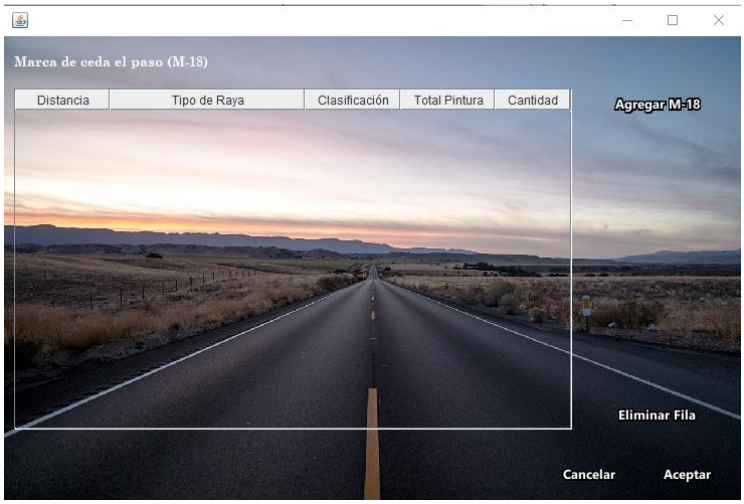


Figura 7.44. M-18

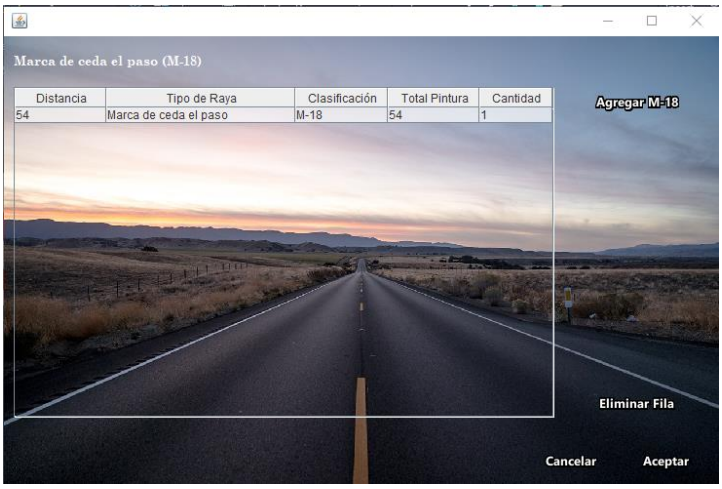


Figura 7.45. Agregando marcas M-18

- **M19**

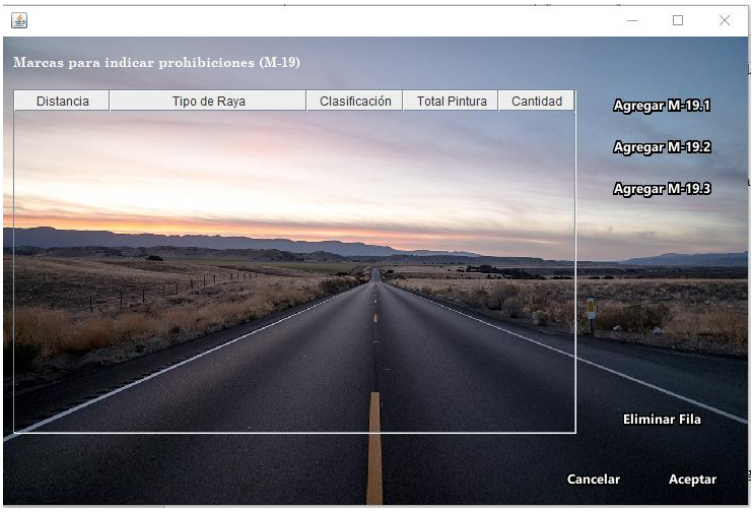


Figura 7.46. M-19

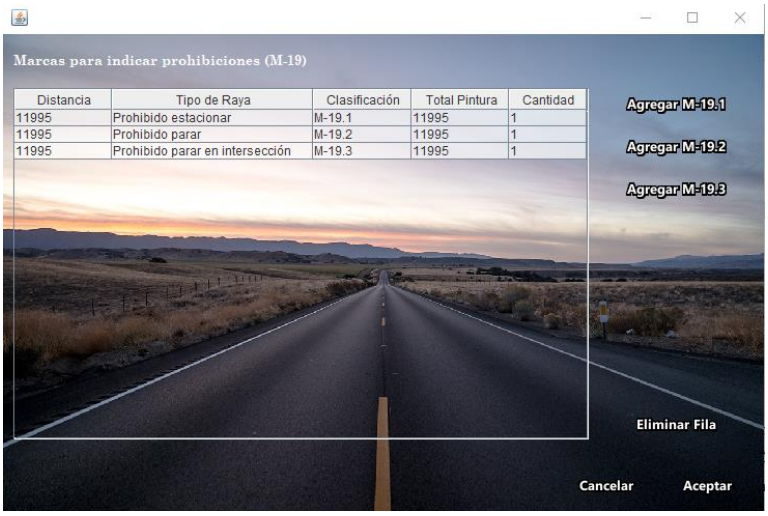


Figura 7.47. Agregando marcas M-19

- **M20**

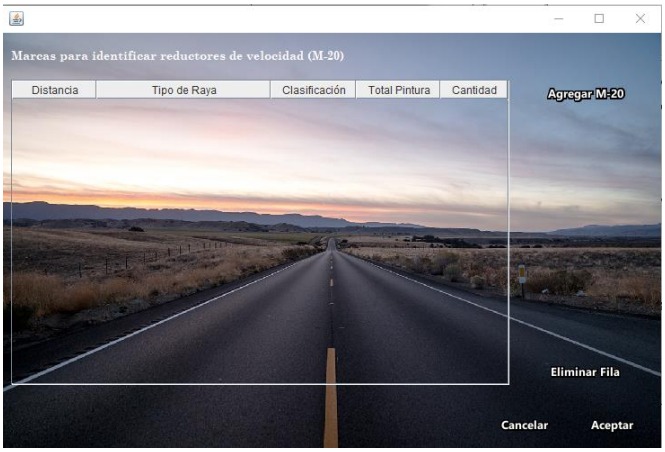


Figura 7.48. M-20

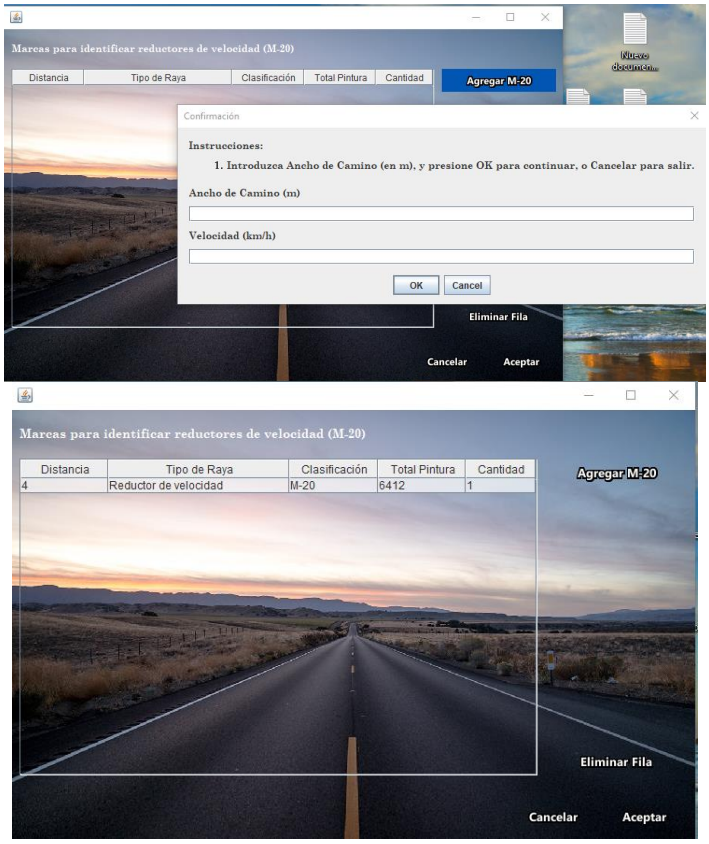


Figura 7.49. Agregando marcas M-20

Cada marca tiene su ventana y se cumple que se muestra las instrucciones y que los comandos solicitados al usuario para usar en AutoCAD funcionan de forma correcta en el archivo que el usuario haya seleccionado.

También se cuenta con la opción de que al final el usuario pueda verificar la tabla que se exportara a AutoCAD y aquí puede modificar y cambiar valores como Color y Dimensión en caso de ser necesario.

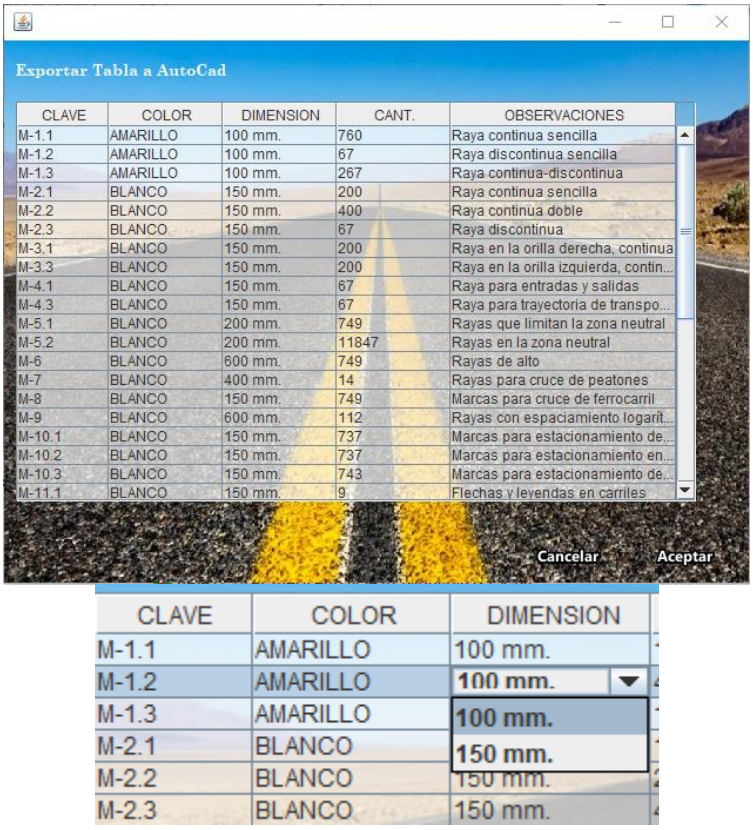


Figura 7.50. Exportar tabla SH

La tabla al exportarla a AutoCAD se comprueba que no se manda como imagen sino como una tabla de AutoCAD Editable

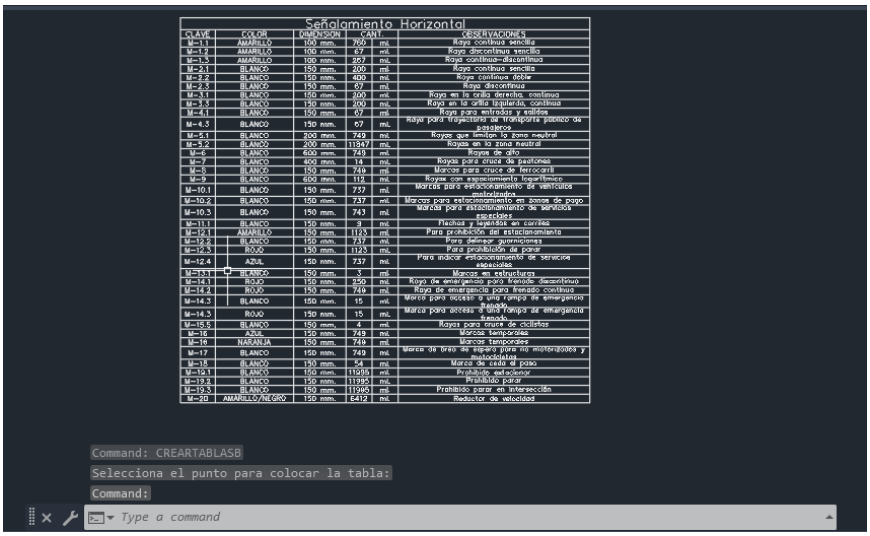


Figura 7.51. Tabla SB en AutoCAD

- **Resultado observado:** Todas las 20 marcas fueron cuantificadas y la tabla importada en AutoCAD permite modificar los valores en AutoCAD y el diseño de la misma para modificaciones simples o cambios posteriores.

2) Señalamiento Vertical

- **Objetivos:**

    1. Verificar extracción de nombre, ubicación y dimensión; en Entronque también incluir código.
    2. Generar tabla total de señales.
    3. Exportar tabla editable.

- **Pasos de verificación:**

    1. Permite usar los comandos creados para AutoCAD, para realizar la cuantificación de las señales verticales
    2. El comando permite seleccionar textos en AutoCAD y estos textos se deben de organizar en las columnas correspondientes dentro de la aplicación.

3. Verificar que al usar el comando correspondiente a Entronque permite elegir aparte de tres textos (nombre, ubicación y dimensión), podemos elegir 4 (nombre, ubicación, dimensión y código).

4. Generar cuantificación total por señal y exportar a AutoCAD.

- **Criterios de aceptación:** Todas las señales listadas con nombre/ubicación/dimensión; en Entronque aparece Código; tabla total coincide con conteo.

- **Capturas:**

  Dentro de la ventana de Señalamiento Verticales el usuario pude usar dos botones uno para camino abierto y otro para entronques.

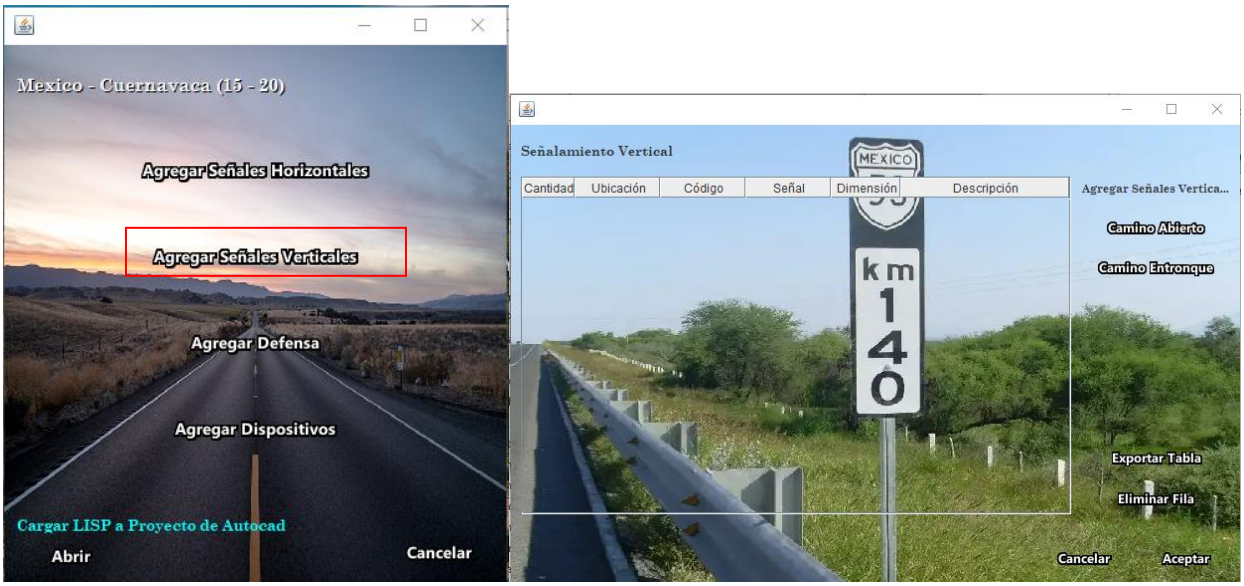


Figura 7.52. Señalamiento Vertical (Elección)

Con cualquiera que el usuario seleccione se debe de poder hacer una selección de textos correspondientes a los 3 o 4 elementos, según sea el caso, correspondientes a las señales verticales y después estos elementos se deben de acomodarse en las ventas correspondientes.

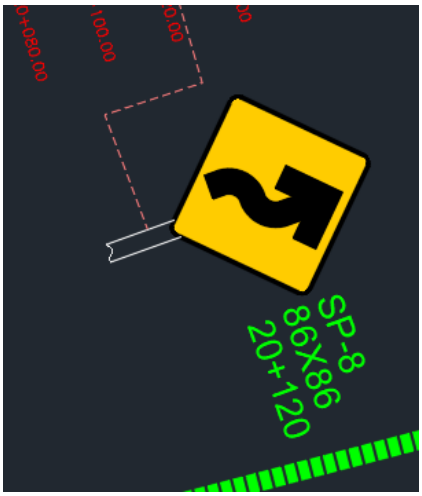


Figura 7.53. Ejemplo Señal Vertical Camino Abierto

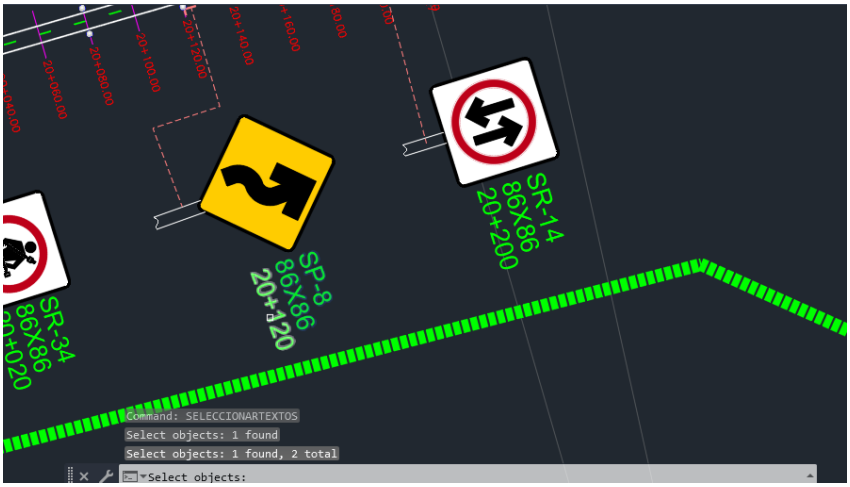


Figura 7.54. Comando SeleccionarTextos AutoCAD

Figura 7.55. Agregando Señal Vertical de Camino Abierto

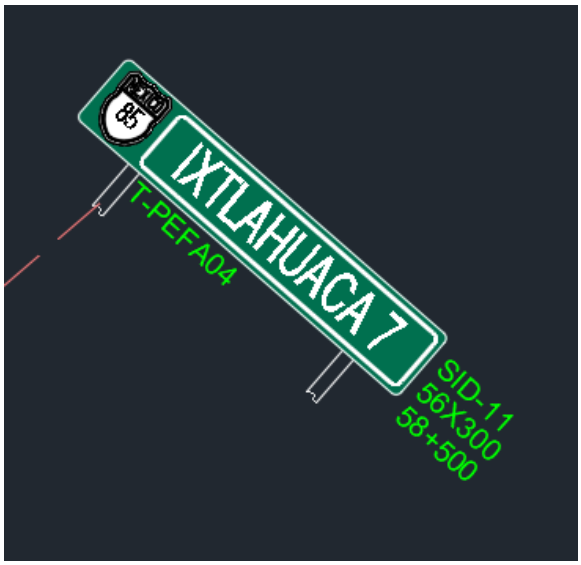


Figura 7.56. Ejemplo Señal Vertical en Entronque

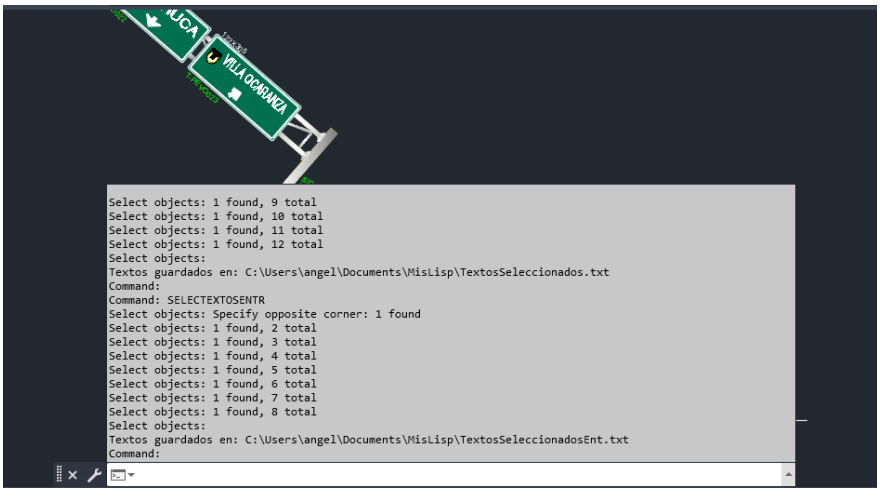


Figura 7.57. Comando SelecTextosEntr AutoCAD

Figura 7.58. Agregando Señal Vertical de Camino Abierto

Al momento de exportar se visualiza la tabla para su revisión final antes de exportar y además se permite ver una tabla con los totales de cada señal considerando que el nombre y la dimensión se a la misma para así poder hacer la suma de señales.

Figura 7.59. Exportación de tabla de Señales Verticales

Figura 7.60. Exportación de tabla Totales de Señales Verticales

En la tabla de totales se puede agregar la señal OD-12 y esta se pude agregara a la base de datos y se muestra únicamente en la tabla de totales.

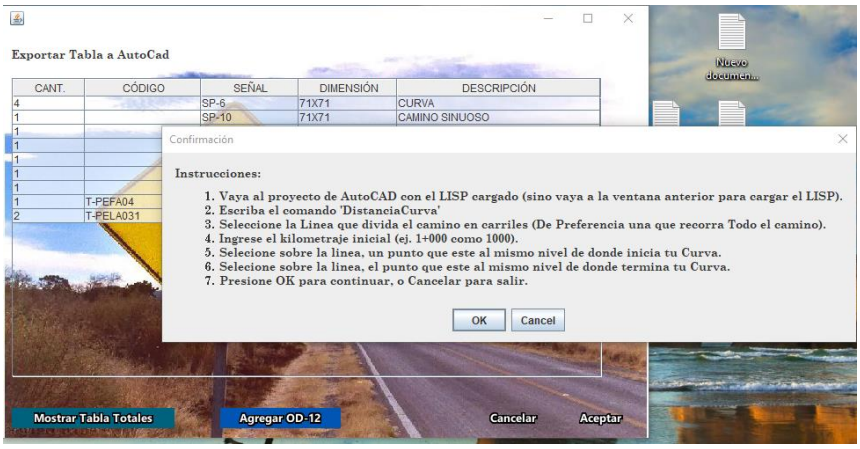


Figura 7.61. Botón Agregar OD-12

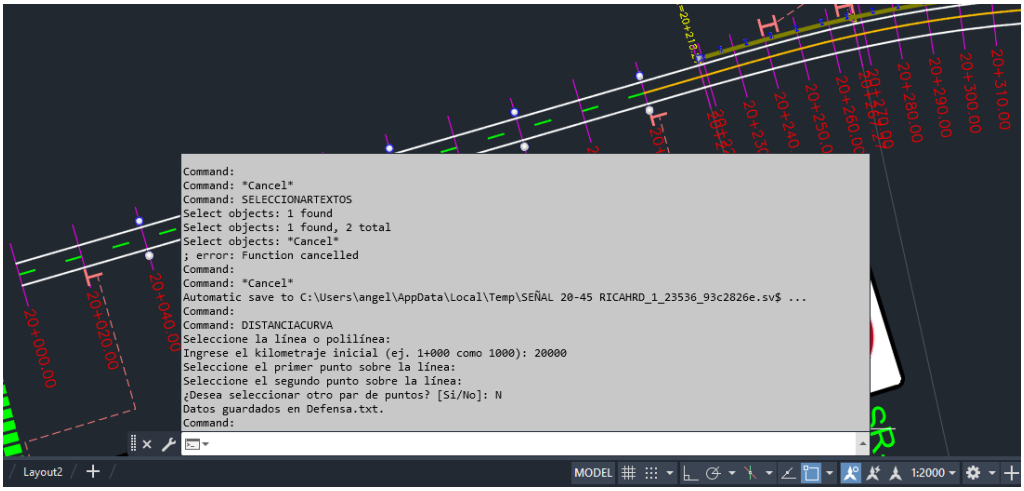


Figura 7.62. Ejemplo selección de curva en AutoCAD

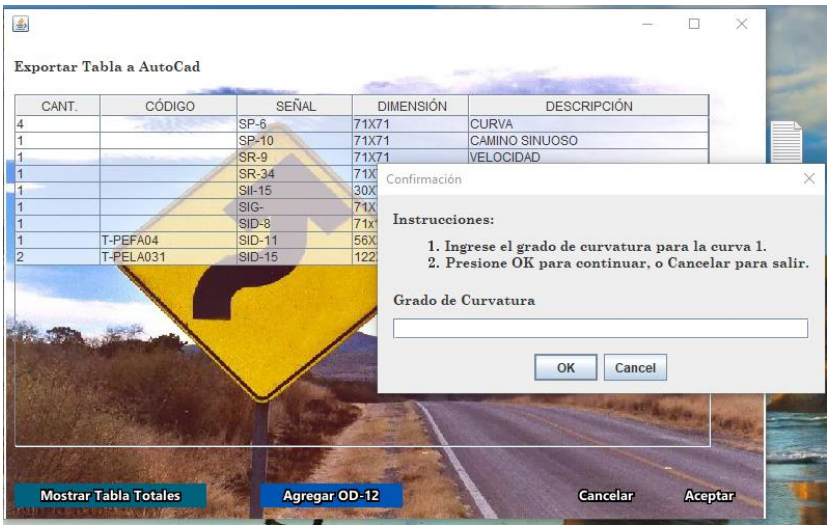


Figura 7.63. Ingreso de grado de curvatura

Figura 7.64. Agregando OD-12

Al momento de exportarse puede hacer la selección de datos que se desean exportar en la tabla con todas las señales y si desea exportar la tabla de Totales

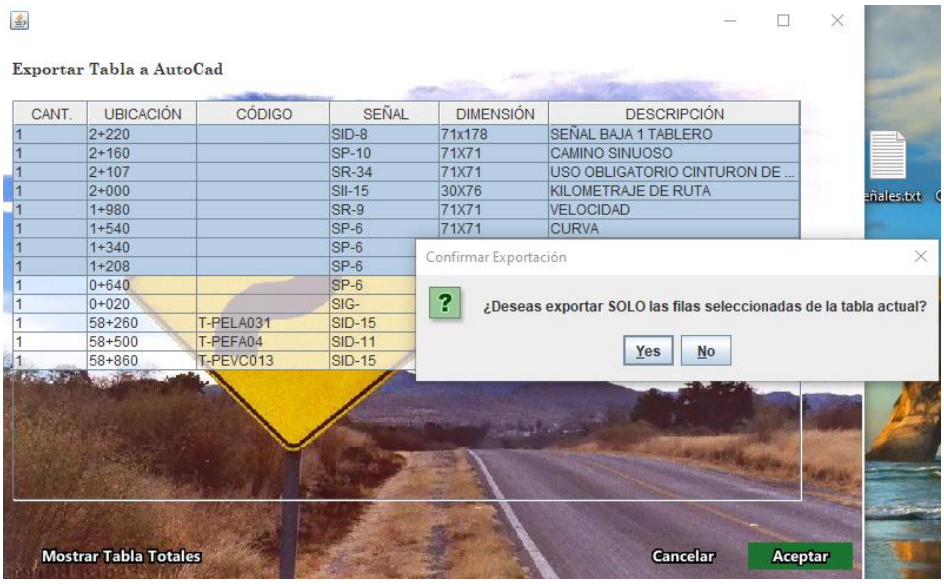


Figura 7.65. Exportando filas seleccionadas

CARRETERA:

CANT	UBICACIÓN	CÓDIGO	SEÑAL	DIMENSIÓN	DESCRIPCIÓN
1	2+220		SID-8	71x178	SEÑAL BAJA 1 TABLERO
1	2+160		SP-10	71X71	CAMINO SINUOSO
1	2+107		SR-34	71X71	USO OBLIGATORIO CINTURON DE SEGURIDAD
1	2+000		SII-15	30X76	KILOMETRAJE DE RUTA
1	1+980		SR-9	71X71	VELOCIDAD
1	1+540		SP-6	71X71	CURVA
1	1+340		SP-6	71X71	CURVA
1	1+208		SP-6	71X71	CURVA
1	0+640		SP-6	71X71	CURVA
1	0+020		SIG-	71X178	SEÑAL INFORMACION GENERAL
1	58+260	T-PELA031	SID-15	122X305	SEÑAL PUENTE
1	58+500	T-PEFA04	SID-11	56X300	SEÑAL CONFIRMATIVA 1 TABLERO
1	58+860	T-PEVC013	SID-15	122X305	SEÑAL PUENTE

Figura 7.66. Exportando filas seleccionadas en AutoCAD

SEÑALAMIENTO VERTICAL TOTALES

CANT	CÓDIGO	SEÑAL	DIMENSIÓN	DESCRIPCIÓN
4		SP-6	71X71	CURVA
1		SP-10	71X71	CAMINO SINUOSO
1		SR-9	71X71	VELOCIDAD
1		SR-34	71X71	USO OBLIGATORIO CINTURON DE SEGURIDAD
1		SII-15	30X76	KILOMETRAJE DE RUTA
1		SIG-	71X178	SEÑAL INFORMACION GENERAL
1		SID-8	71x178	SEÑAL BAJA 1 TABLERO
1	T-PEFA04	SID-11	56X300	SEÑAL CONFIRMATIVA 1 TABLERO
2	T-PELA031	SID-15	122X305	SEÑAL PUENTE
5		OD-12	61 × 45	INDICADORES DE CURVAS CERRADAS

Figura 7.67. Exportación de tabla de Totales en AutoCAD

- **Resultado observado:** Todos los datos que se seleccionan en el orden señal, dimensión y ubicación en AutoCAD se acomendan en su columna correspondiente en la tabla, además tenemos la opción de agregar las señales OD-12 y se pueden exportar dos tablas una con la ubicación y otra con el total se señales por tramo.

3) Defensa

- **Objetivo:**

    1. Obtener los kilometrajes de inicio/fin de cada sección de defensa, con esto obtener la longitud de cada defensa y cuantificar la sección de amortiguamiento.

    2. Tabla editable dentro de AutoCAD.

- **Pasos de verificación:**

    1. Seleccionar una línea en el tramo y por medio de esta poder seleccionar los puntos de inicio y final de cada defensa.

    2. Se genera la tabla indicando el lado de la Defensa y además con los kilometrajes de los puntos de inicio y final se calcula la distancia total de cada sección de defensa.

    3. Verificar que, en el archivo exportado, las filas están ordenadas por kilometraje inicial ascendente.

- **Capturas:**

    Al momento de entrar a la ventana para Agregar Defensa, se pude tener botones que permitan seleccionar lado derecho o lado izquierdo

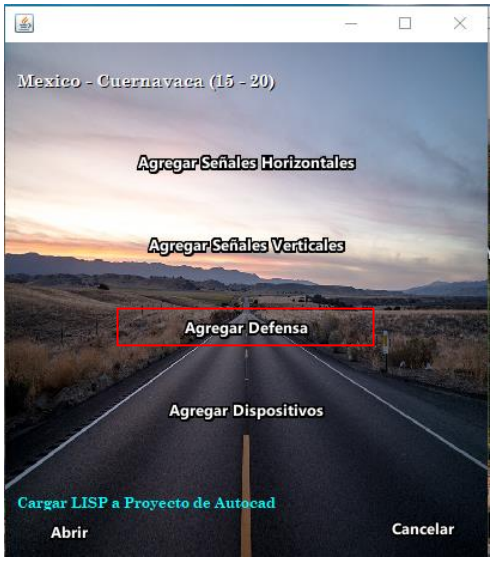


Figura 7.68. Defensa (Elección)

Dependiendo de la elección se sale un mensaje para cada lado.

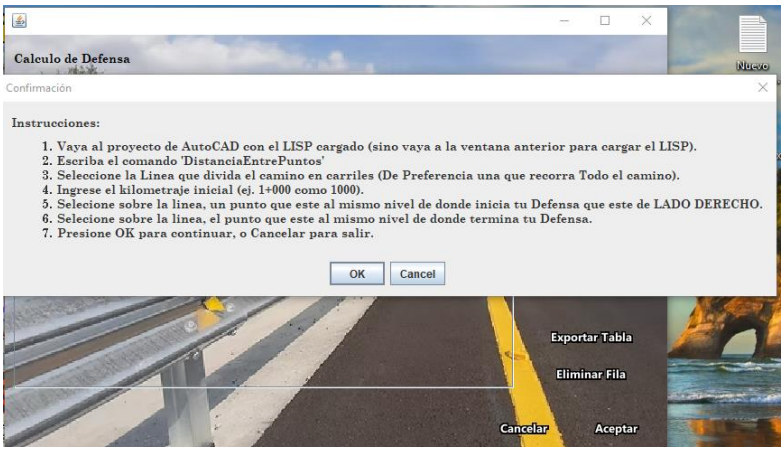


Figura 7.69. Instrucciones para agregar Defensa

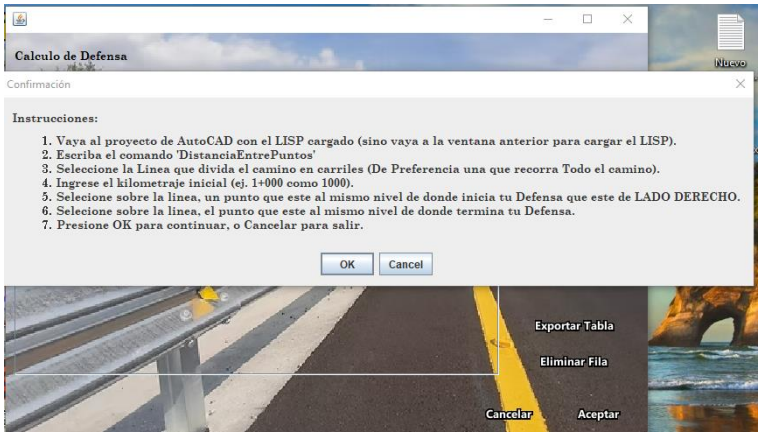


Figura 7.70. Agregar datos Defensa Ambos lados

Se pude hacer que dependiendo de los puntos seleccionados en AutoCAD estos se acomodan en Inicio y Final y se calcula la longitud de la defensa de manera automática y si quiere el usuario pude hacer ajuste en los kilometrajes que el programa tiene como puntos iniciales y finales.

La tabla que se genera es una tabla de AutoCAD la cual permite modificaciones dentro de AutoCAD ayudando con la realización de cambios y modificaciones.

Figura 7.71. Vista previa de tabla de Defensa

Figura 7.72 Exportación de Tabla de Defensa

- **Resultado observado:** se permite modificar los Kilometrajes de inicio y final que se obtienen de AutoCAD, y se realiza el cálculo correspondiente también permite en la sección de amortiguamiento para que el usuario agregue la cantidad correspondiente.

  La tabla generada es una tabla de AutoCAD permitiendo modificaciones.

4) Barreras

- **Objetivo:** Verificar cuantificación de los 5 elementos especificados de las barreras y exportación en una tabla editable dentro de AutoCAD.
- **Pasos de verificación:**

  1. Se debe de hacer la cuantificación total de longitud de Defensa y de la Sección de Amortiguamiento, para ello debe leer los datos ya guardados en la tabla de Defensa.
  2. Se debe de poder modificar los datos que aparecen en la tabla conforme al Nivel de Contención.
  3. Los demás datos solo se requieren que se haga la cuantificación en caso de que haya.
  4. Antes de exportar solo se debe de mostrar los elementos de la tabla donde se tenga algún valor o cantidad, si el valor es cero entonces se renglón del elemento correspondiente no se exporta.
  5. Exportar tabla a AutoCAD y verificar edición.

- **Capturas:**

  Se tiene los 5 Elementos a considerar de la Barreras cada una se pude hacer la cuantificación correspondiente, además que para la Barrera Metálica y la sección de amortiguamiento se leen los valores de la tabla de defensa que ya este almacenada en la base de datos.

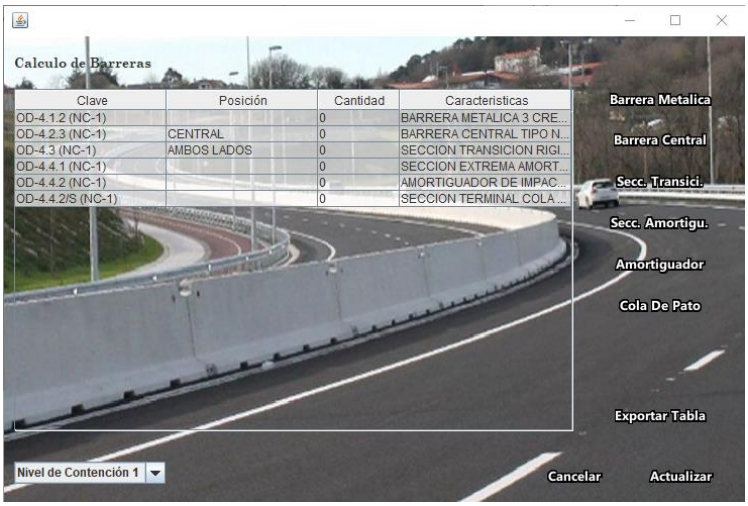


Figura 7.73. Barrera Total

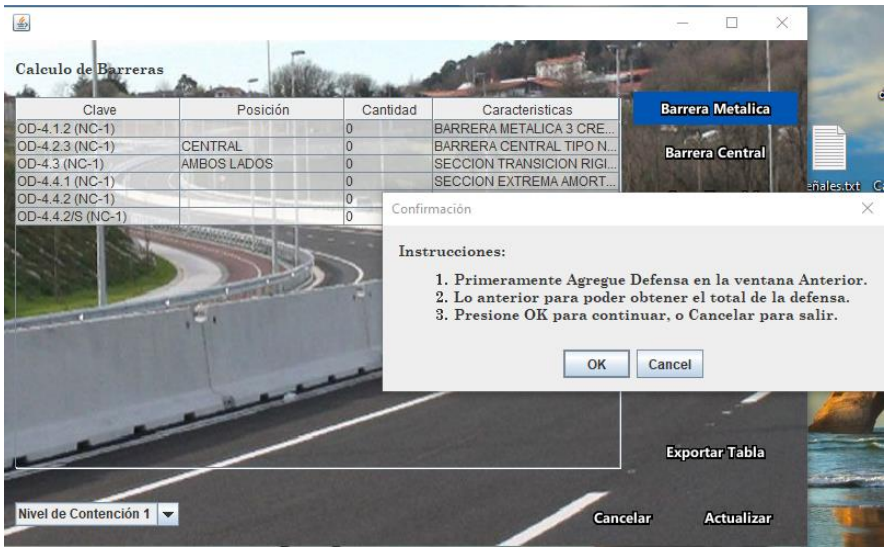


Figura 7.74. Agregando datos Amortiguador de Impacto y Barrera Metálica

Se cambian los valores de la tabla conforme a la elección de Nivel de Contención que el usuario haga

Figura 7.75. Elección de Nivel de Contención

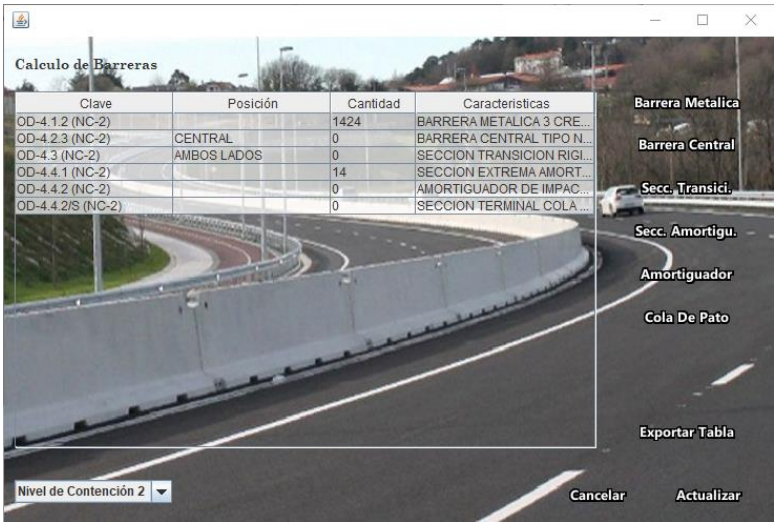


Figura 7.76. Actualizando datos Barrera Total

Una vez que se vaya a exportar la tabla solo muestra los elementos que fueron modificados y que contienen algún valor los cuales serán los que se exportaran.

Figura 7.77. Exportación de tabla Barrera Total

Figura 7.78. Exportación de Tabla de Barrera en AutoCAD

- **Resultado observado:** La Tabla lee la cuantificación de la defensa, al momento de exportar solo muestra los elementos que contienen un valor y tabla editable en AutoCAD.

5) Dispositivos y Botones

- **Objetivo:** Tener una tabla tanto para dispositivos como para botones. En los botones hace el cálculo con la distancia de las marcas Horizontales que ya estén en la base de datos. Para los dispositivos hay 5 dispositivos que se debe de obtener su cuantificación según sea el caso correspondiente.

- **Pasos de verificación:**

    1. Hacer la cuantificación de los dispositivos, en el caso de los botones reflejantes tiene que leer la distancia de Barreras totales para la Defensa Metálica.
    2. Cuantificación de Botones, donde se debe de leer las marcas de pintura del señalamiento horizontal que ya deben de estar en la base de datos, y se debe de aplicar solo a las que tiene que deben de llevar botones.
    3. Al exportar deben de mostrarse ambas tablas.
    4. Exportar y abrir en AutoCAD como una tabla editable.

- **Capturas sugeridas:**

    En el apartado de Agregar Dispositivos podemos ver la tabla de Dispositivos donde cada botón muestra la instrucción correspondiente al dispositivo de agregar

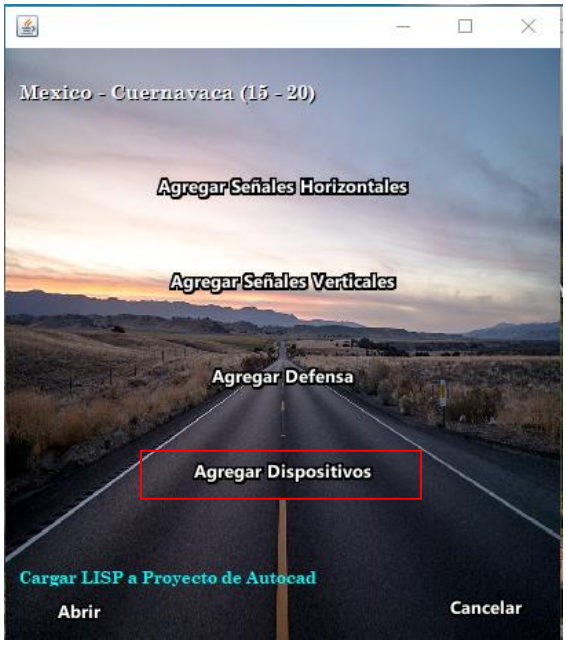


Figura 7.79. Dispositivos

Tenemos que para cada uno de los 5 dispositivos que se considerando realizar hace el cálculo correspondiente y le muestra al usuario que hacer para hacer el calculo

Figura 7.80. Agregando indicadores de alineamiento Curva

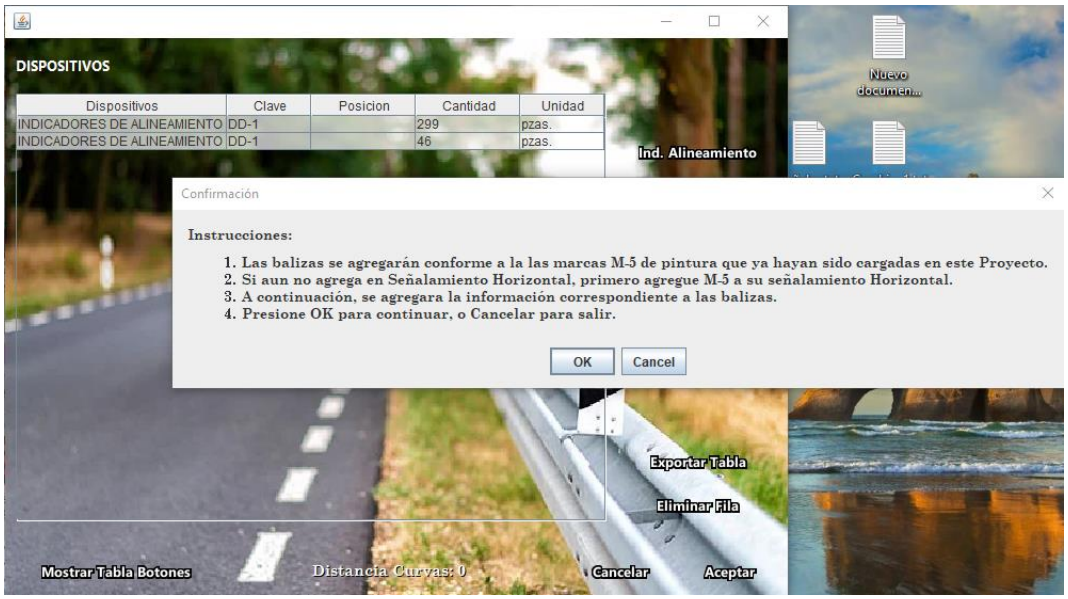


Figura 7.81. Balizas

Figura 7.82. Agregando balizas

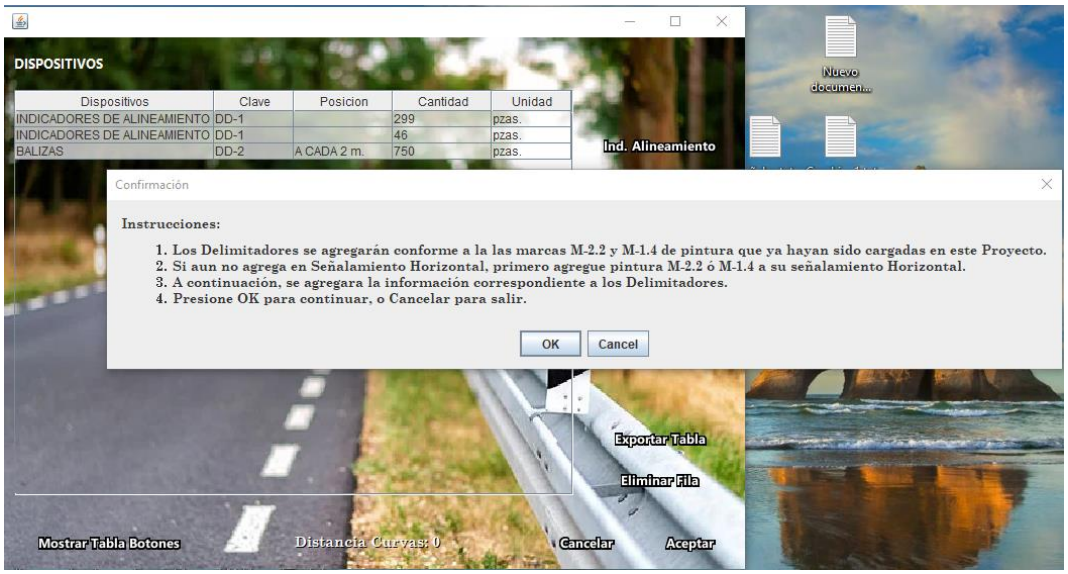


Figura 7.83. Delimitadores

Figura 7.84. Agregando delimitadores

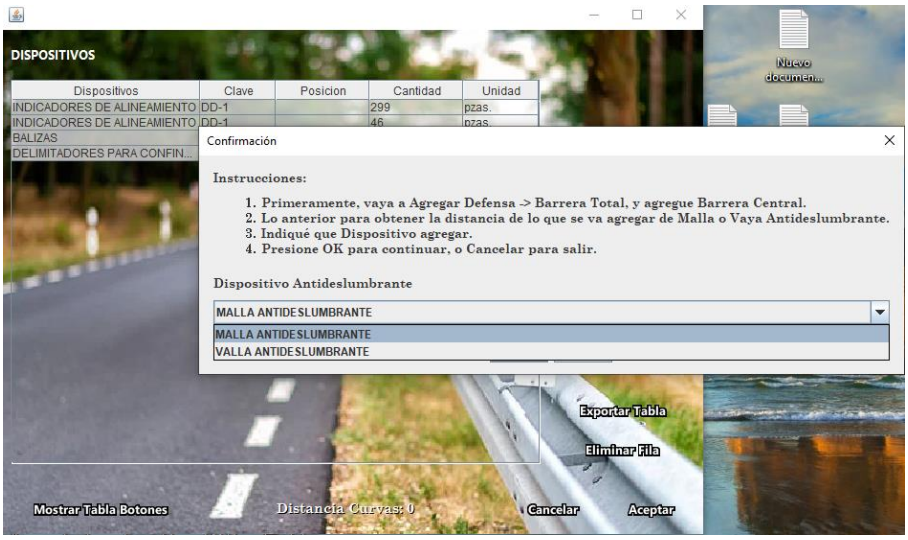


Figura 7.85. Dispositivos Antideslumbrantes

Figura 7.86. Agregando dispositivos antideslumbrantes

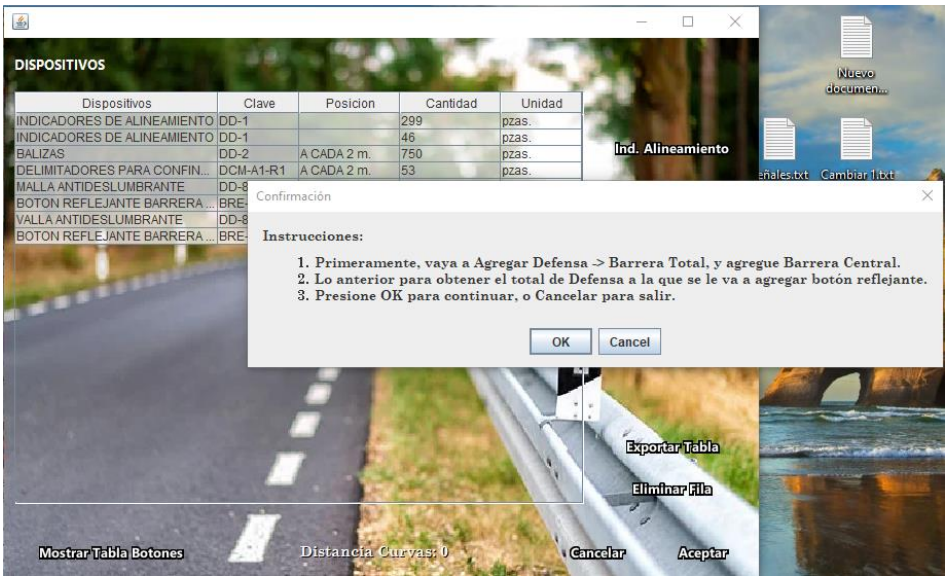


Figura 7.87. Botón Reflejante

Figura 7.88. Agregando botón reflejante

En la misma ventana tenemos la cuantificación de botones aquí podemos acceder a cualquiera de las 2 tablas, aquí se comprueba que se debe de considerar que cada que se va agregar botones hay que tener la cuantificación de las curvas que hay en el tramo, con esto hace el cálculo correspondiste a las curvas y tangentes. También se realiza una validación para confirma si se va agregar botones a una marca ya guardada en la base de datos.

Figura 7.89. Vista de Botones

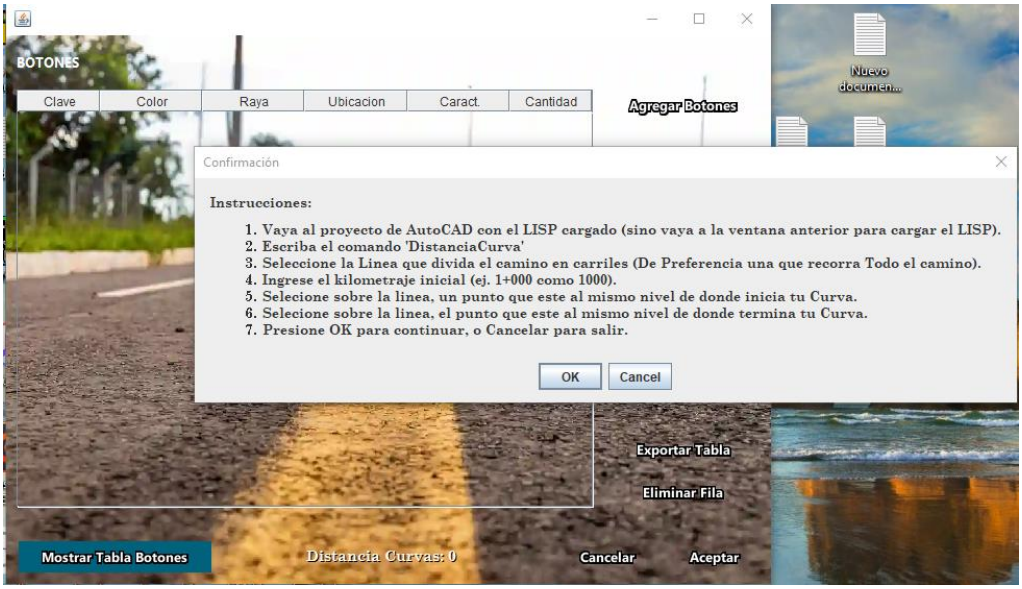


Figura 7.90. Agregar Curvas

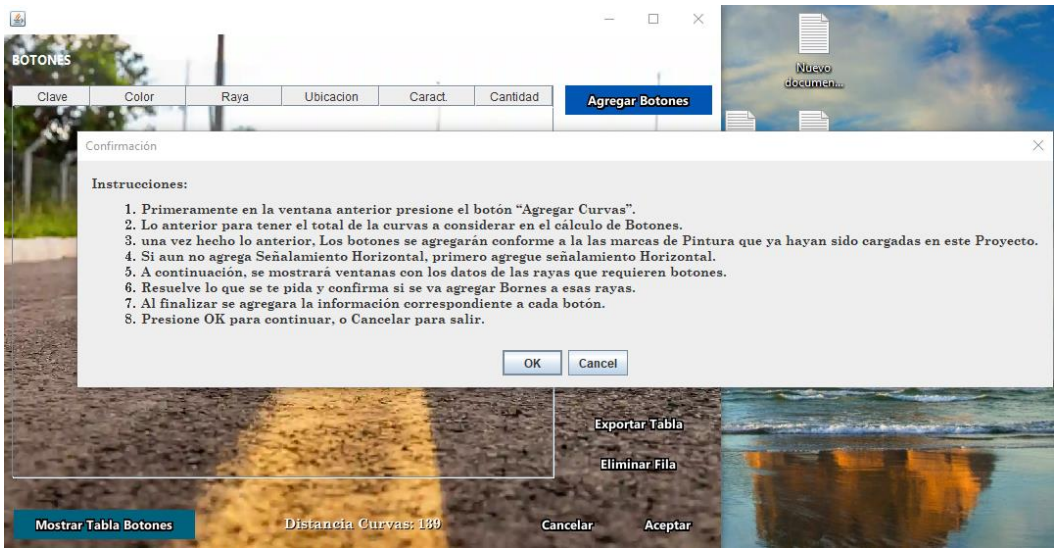


Figura 7.91. Agregar Botones

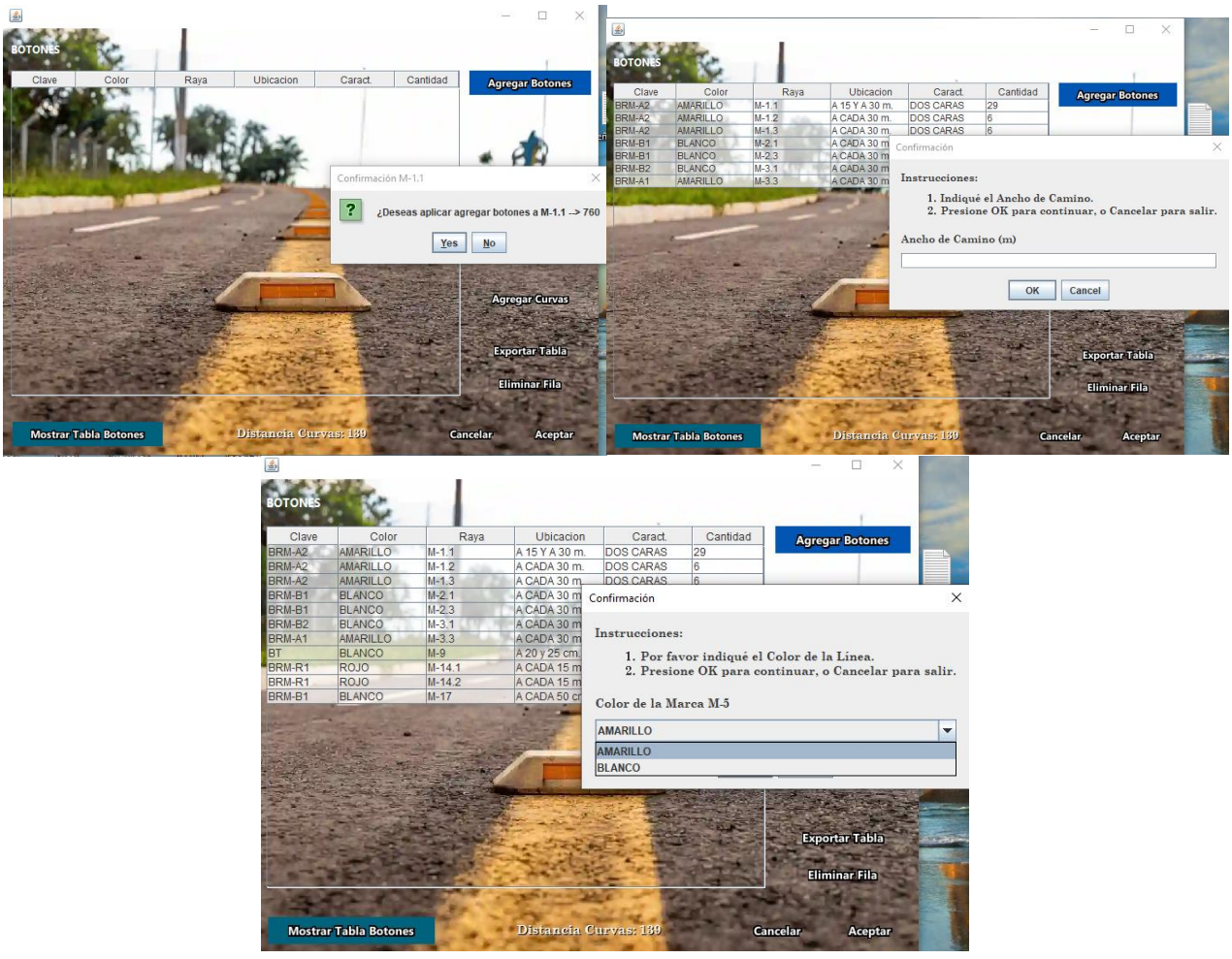


Figura 7.92. Confirmación de Botones

Al momento de exportar se pude ver la tabla tanto de dispositivos como la de botones, las tablas que se exportan de cada una son editable dentro de AutoCAD.

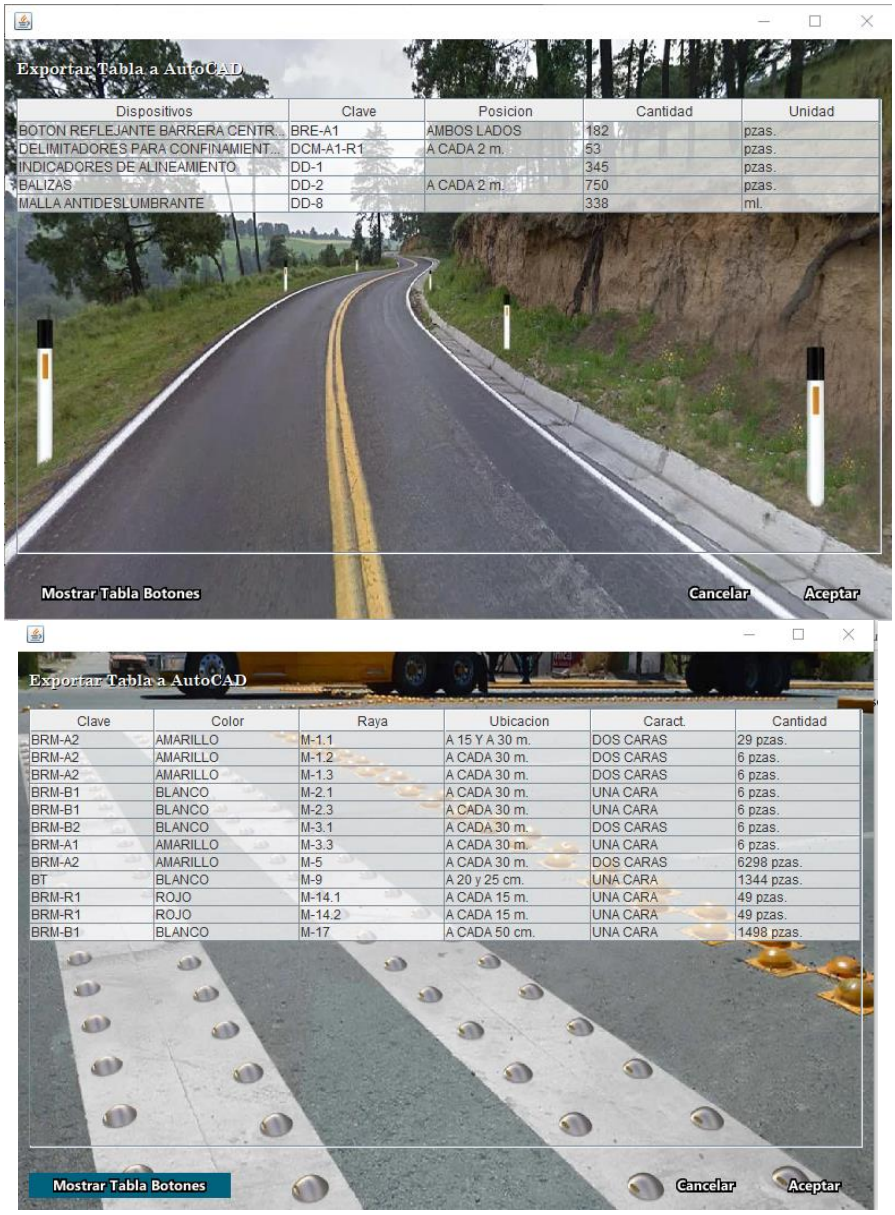
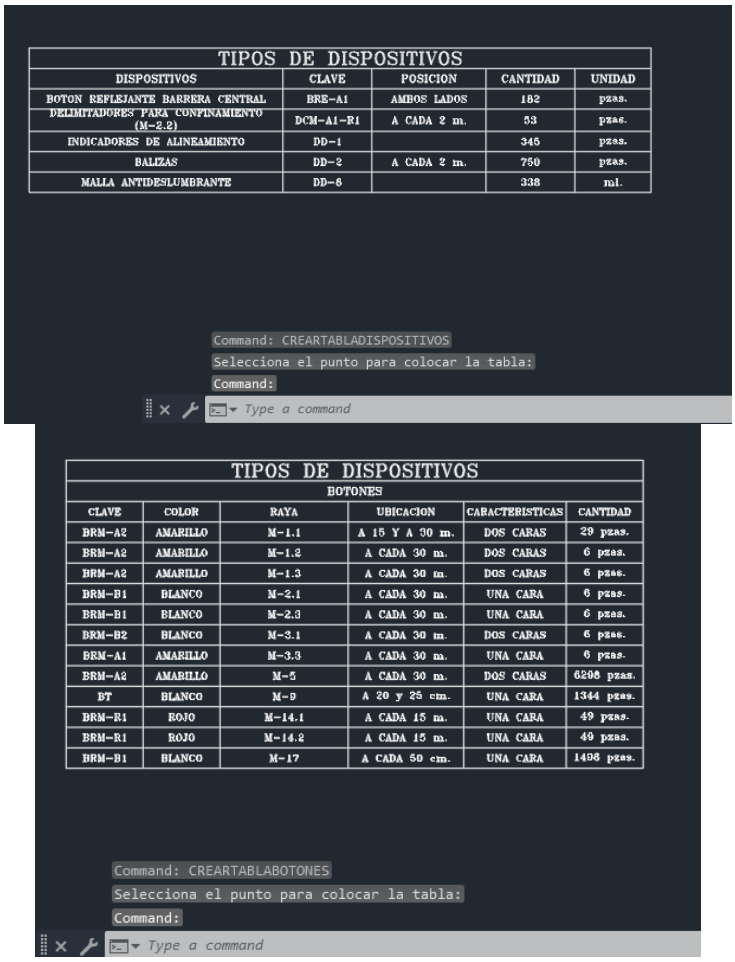


Figura 7.93. Exportar Tablas



TIPOS DE DISPOSITIVOS

DISPOSITIVOS	CLAVE	POSICION	CANTIDAD	UNIDAD
BOTON REFLEJANTE BARRERA CENTRAL	BRE-A1	AMBOS LADOS	182	pzas.
DELIMITADORES PARA CONFINAMIENTO (M-2.2)	DCM-A1-R1	A CADA 2 m.	53	pzas.
INDICADORES DE ALINEAMIENTO	DD-1		345	pzas.
BALIZAS	DD-2	A CADA 2 m.	750	pzas.
MALLA ANTIDESLUMBRANTE	DD-6		338	ml.

Command: CREARTABLADISPOSITIVOS
Selecciona el punto para colocar la tabla:
Command:
Type a command

TIPOS DE DISPOSITIVOS

BOTONES

CLAVE	COLOR	RAYA	UBICACION	CARACTERISTICAS	CANTIDAD
BRM-A2	AMARILLO	M-1.1	A 15 Y A 30 m.	DOS CARAS	29 pzas.
BRM-A2	AMARILLO	M-1.2	A CADA 30 m.	DOS CARAS	6 pzas.
BRM-A2	AMARILLO	M-1.3	A CADA 30 m.	DOS CARAS	6 pzas.
BRM-B1	BLANCO	M-2.1	A CADA 30 m.	UNA CARA	6 pzas.
BRM-B1	BLANCO	M-2.3	A CADA 30 m.	UNA CARA	6 pzas.
BRM-B2	BLANCO	M-3.1	A CADA 30 m.	DOS CARAS	6 pzas.
BRM-A1	AMARILLO	M-3.3	A CADA 30 m.	UNA CARA	6 pzas.
BRM-A2	AMARILLO	M-5	A CADA 30 m.	DOS CARAS	6298 pzas.
BT	BLANCO	M-9	A 20 y 25 cm.	UNA CARA	1344 pzas.
BRM-R1	ROJO	M-14.1	A CADA 15 m.	UNA CARA	49 pzas.
BRM-R1	ROJO	M-14.2	A CADA 15 m.	UNA CARA	49 pzas.
BRM-B1	BLANCO	M-17	A CADA 50 cm.	UNA CARA	1498 pzas.

Command: CREARTABLABOTONES
Selecciona el punto para colocar la tabla:
Command:
Type a command

Figura 7.94. Exportación Tablas Dispositivos y Botones AutoCAD

- **Resultado observado:** Se considera una lectura a las marcas ya guardadas en la base de datos, en el caso de los botones, y se hace el cálculo correspondiente a cada dispositivo agregado. También una tabla editable en AutoCAD.

7.2. Impacto

La implementación de este sistema genera un impacto directo y altamente significativo en el trabajo de elaboración de proyectos viales. Los beneficios no solo son medibles, sino tan notorios que transforman completamente la manera en la que se realiza la cuantificación dentro del plano:

- **Ahorro de tiempo:** La automatización elimina prácticamente todo el trabajo manual que antes llevaba horas o incluso días. Ahora la cuantificación se realiza de manera más rápida. Esto reduce la carga de trabajo y evita tareas repetitivas.

- **Menos errores y más precisión:** Al generar automáticamente las tablas y medidas, el sistema elimina errores humanos de captura, conteo o transcripción. La información queda ordenada y limpia desde el primer intento.
- **Tablas profesionales y listas para usarse:** Las tablas exportadas se generan con formato uniforme y completamente editable dentro de AutoCAD. Esto significa que ya no se pierde tiempo acomodando, corrigiendo o reescribiendo datos; todo sale listo y en el formato solicitado.
- **Mejor organización del proyecto:** Gracias al manejo de kilometrajes, clasificaciones, tipos de señal y códigos (cuando corresponda), el sistema proporciona una estructura clara que facilita el control, el análisis y la revisión del proyecto.
- **Compatibilidad total con AutoCAD:** Al exportar como tabla real y no como imagen, se evita cualquier reproceso. La información se integra directamente al plano sin pasos intermedios.
- **Mayor calidad en la entrega:** Las tablas generadas cumplen con los formatos utilizados en ingeniería vial, lo que mejora la presentación del proyecto y deja un trabajo más profesional y fácil de revisar.

En conjunto, el sistema no solo cumple los requerimientos: mejora, agiliza y estandariza completamente el proceso de cuantificación**,** reduciendo tiempos, minimizando errores y aumentando la calidad de los resultados finales.

7.3. Conclusiones

La aplicación desarrollada constituye una herramienta eficaz para asistir a los ingenieros civiles en el cálculo y creación de tablas de señalización vial en México, aportando precisión, rapidez y confiabilidad en los resultados.

La propuesta de solución planteada al inicio de este trabajo se desarrolló mediante la creación de una aplicación modular capaz de automatizar procesos que anteriormente se realizaban manualmente o mediante otras herramientas. A través de los diferentes módulos implementados, el sistema permitió centralizar la captura de información, la cuantificación de los elementos de señalización y la generación de tablas compatibles con AutoCAD.

Durante las pruebas y el uso del sistema se observó una mejora en la organización de la información y en la reducción de pasos necesarios para realizar las cuantificaciones. Asimismo, el sistema ayudó a disminuir errores derivados de capturas manuales y cálculos repetitivos, gracias a las validaciones implementadas y a la automatización de operaciones dentro de cada módulo.

La integración con AutoCAD mediante archivos temporales demostró ser una solución viable y práctica, ya que permitió aprovechar las capacidades gráficas de AutoCAD sin perder la independencia del software principal.

El uso de Java como lenguaje de programación fue apropiado debido a su portabilidad, robustez y facilidad para estructurar el código de manera modular, lo que facilitó tanto el mantenimiento como la identificación y corrección de errores durante las pruebas realizadas en cada iteración del desarrollo.

El enfoque en la separación de cálculos horizontales y verticales facilitó la organización interna del software, reflejándose en una interfaz intuitiva, debido al uso de elementos conocidos por el usuario dentro de un proyecto de AutoCAD, ya que se usaron elementos como las líneas, los textos, tablas y demás elementos, los cuales el usuario ya conoce y sabe identificarlos, respondiendo así a las necesidades reales de los usuarios al momento de cuantificar los elementos usados dentro de un proyecto.

La aplicación no solo reduce tiempos en el proceso de diseño de señalización vial, sino que también contribuye a mejorar la seguridad y eficiencia en las vías de comunicación, al proporcionar resultados ajustados a las normativas vigentes.

En conclusión, el proyecto representa un avance significativo en la digitalización de procesos de diseño vial en México, constituyendo una base sólida para futuras mejoras e integraciones que fortalezcan aún más la labor de los ingenieros civiles en este ámbito.

8. Referencias.

[1] Secretaría de Desarrollo Agrario, Territorial y Urbano, *"Manual de señalización y dispositivos para el control del tránsito en calles y carreteras"*, feb. 2024.

[2] Instituto Mexicano del Transporte, "Red Nacional de Caminos", gob.mx, 2023. [En línea]. Disponible en: https://www.gob.mx/imt/acciones-y-programas/red-nacional-de-caminos.

[3] INEGI, *"RED NACIONAL DE CAMINOS (RNC) ACTUALIZACIÓN 2022"*, Org.mx, 2022. [En línea]. Disponible en: https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2022/RNC/RNC2022.pdf.

[4] *"Breve Historia Del Automóvil En México",* Cherris.mx, 2020. [En línea]. Disponible en: https://cherris.mx/teoria-sin-costo/.

[5] *"¿Qué es Java?"*, Amazon.com. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/java/.

[6] Cal y Mayor, Rafael y Cárdenas, James. *"Ingeniería de Tránsito: Fundamentos y Aplicaciones"*, 9ª ed., México: Alfaomega Grupo Editor, abr. 2018.

[7] Holzner, Steven. Java 2 Black Book, 2.ª ed., Scottsdale, Arizona: Coriolis Group Books, 2001.

[8] MDN Web Docs. "MVC", *MDN Web Docs*, Fundación Mozilla. Disponible en: https://developer.mozilla.org/es/docs/Glossary/MVC