



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un Sistema de Inventario

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de
Ingeniero en Computación

P R E S E N T A

José Adrián Martínez Salinas

ASESOR DE INFORME

Dr. Ismael Everardo Bárcenas Patiño



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 DESARROLLO DE UN SISTEMA DE INVENTARIO que presenté para obtener el título 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.

JOSE ADRIAN MARTINEZ SALINAS
Número de cuenta: 313067930

*A **Efrén Rey**, espero **estés** orgulloso de mí;
al fin pude cumplir mi promesa.*

*Para **Guadalupe**, me enseñaste más de lo que crees
y me ayudaste más de lo que recuerdo.*

*Para **Ángeles**, hiciste lo mejor que pudiste
y gracias a ello estoy aquí.*

*Para **Alejandro**, mi hermano y mi mejor amigo,
gracias por creer en mí.*

*Para **Lázaro**, por apoyarme tantos años
y creer en mí; no lo hubiera logrado sin ti.*

*Para **Aglæ**, por darme ánimos cuando más lo necesité,
por enseñarme a esforzarme y perseguir mis sueños.*

*A todos aquellos que me apoyaron y estuvieron ahí
incondicionalmente en este camino. ¡Gracias!*

Resumen

El presente informe de actividades profesionales detalla el diseño, desarrollo e implementación de un sistema de inventario digital centralizado para una empresa del sector tecnológico. La problemática inicial se centraba en una gestión fragmentada de activos, basada en métodos manuales y herramientas no especializadas, lo que derivaba en falta de trazabilidad y riesgos de pérdida de información.

Para resolver esta necesidad, se desarrolló una plataforma robusta utilizando el *framework* **Django** y el lenguaje **Python**, integrada con una base de datos relacional en **MySQL**. La metodología de trabajo incluyó el levantamiento de reglas de negocio, el modelado conceptual mediante diagramas Entidad-Relación y la construcción de un modelo lógico optimizado. Asimismo, se implementó un sistema de seguridad basado en jerarquías de usuario (*Role-Based Access Control*) y protocolos de cifrado para garantizar el acceso remoto seguro.

Los resultados demuestran que la transición a un sistema digital permitió la centralización de datos, la automatización de reportes en **PDF** y una mejora significativa en la eficiencia operativa. El sistema no solo cumple con los requerimientos actuales de escalabilidad y disponibilidad, sino que establece una base tecnológica modular capaz de evolucionar ante futuras necesidades de la organización.

Palabras clave: Django, MySQL, Inventario, Gestión de Activos, Desarrollo Web, Ingeniería en Computación.

Índice general

Resumen	II
Índice de figuras	VI
Índice de tablas	VII
Índice de fragmentos de código	VIII
1 Introducción	1
1.1. Objetivos	2
1.1.1. Objetivo General	2
1.1.2. Objetivos Específicos	2
1.2. Índice Desglosado	2
2 Historia Laboral	4
2.1. Empresa de Licitaciones Gubernamentales y Privadas	4
2.1.1. Historia	4
2.1.2. Misión y Visión	5
2.1.3. Divisiones	6
2.2. Historia laboral en la empresa	6
2.2.1. Administración y Mantenimiento del Sitio Web	7
2.2.2. Desarrollo y Modelado de la Base de Datos	7
2.2.3. Desarrollo de la Aplicación Web e Integración	8
2.2.4. Despliegue y Puesta en Operación en Infraestructura Corporativa	8
3 Desarrollo de un Sistema de Inventario	9
3.1. Antecedentes	9
3.1.1. Gestión de Inventario	9
3.1.2. Tipos de Bases de Datos	10

3.1.3.	Bases de Datos SQL	12
3.1.4.	Interfaces de Usuario - Web	13
3.1.5.	Conexión entre bases de datos y aplicaciones web	14
3.1.6.	Servidor Web Empresarial (Hosting Propio)	16
3.2.	Definición del Problema	17
3.3.	Análisis de la Solución	18
4	Implementación	19
4.1.	Metodología para el Desarrollo de la Base de Datos	19
4.1.1.	Reglas de negocio	19
4.1.2.	Diagrama Entidad-Relación (ER)	22
4.1.3.	Diagrama Relacional	23
4.1.4.	Modelo Físico	23
4.1.5.	Pruebas	24
4.2.	Metodología para el Desarrollo de la Interfaz Web	25
4.2.1.	Creación del Proyecto y las Aplicaciones en Django	25
4.2.2.	Conexión entre Django y MySQL	26
4.2.3.	Frontend y Backend	27
4.2.4.	Autenticación	28
4.2.5.	Jerarquías y Permisos	28
4.2.6.	Pruebas	29
4.3.	Metodología para el Acceso Remoto	30
4.3.1.	Cargar el Proyecto en los Servidores de la Empresa	30
4.3.2.	Pruebas de Conexión	30
4.3.3.	Pruebas de Funcionamiento	31
4.3.4.	Despliegue e Integración en la Infraestructura Corporativa	31
4.4.	Resultados	33
4.4.1.	Validación y Cumplimiento de Objetivos	33
4.4.2.	Pruebas Funcionales y de Rendimiento	33
4.4.3.	Beneficios Obtenidos del Sistema de Inventario	34
4.4.4.	Evidencias de Operación del Sistema	35
5	Conclusiones	41
5.1.	Logros Técnicos y Aportaciones de Ingeniería	41
5.2.	Limitaciones Identificadas	42
5.3.	Aplicaciones y Trabajo a Futuro	42
5.3.1.	Ámbitos de Aplicación	42
5.3.2.	Líneas de Trabajo Futuro	43
	Glosario	44

Referencias	46
Anexo 1. Diagrama Entidad-Relación (ER) del sistema de inventario	48
Anexo 2. Diagrama Relacional del sistema de inventario	51
Anexo 3. Código SQL del modelo físico del sistema de inventario	53
Tablas	54
Triggers	56
Índices	57
Anexo 4. Código representativo del sistema en Django	58
inventario\models.py	59
inventario\views.py	60
admin_inventory\urls.py	61

Índice de figuras

3.1. Tipos de Bases de Datos	11
3.2. Funcionamiento de SQL	12
3.3. User Interface vs. User Experience	14
3.4. Conexión a la base de datos	15
3.5. ¿Cómo funciona el alojamiento web?	16
4.1. Diagrama de flujo de acceso y arquitectura de despliegue	32
4.2. Acceso al sistema mediante DNS local	35
4.3. Formulario de inicio de sesión	36
4.4. Módulo de consulta general y listado de equipos	36
4.5. Formulario de captura de empleado	36
4.6. Gestión de usuarios y asignación de jerarquías	37
4.7. Formulario de creación de usuario	37
4.8. Vista de detalle de proyecto	38
4.9. Reporte de entrega de equipos por proyecto en PDF	38
4.10. Vista de detalle de asignación de equipo	39
4.11. Documento de resguardo individual en PDF	39
4.12. Módulo de auditoría e historial de cambios	40
1. Diagrama ER, parte 1.	49
2. Diagrama ER, parte 2.	50
3. Diagrama Relacional.	52

Índice de tablas

4.1. Tabla de Jerarquías.	29
4.2. Matriz de trazabilidad y cumplimiento de objetivos.	33
4.3. Resultados del protocolo de pruebas funcionales y de rendimiento.	34

Índice de fragmentos de código

4.1. Ejemplo de definición de tabla Almacen.	24
5.1. Ejemplo de definición de tablas en SQL. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.	54
5.2. Ejemplos de triggers utilizados en el sistema de inventario. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.	56
5.3. Índices definidos para optimizar consultas frecuentes. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.	57
5.4. Definición del modelo Almacen en models.py. Fuente: Elaboración propia en Django. . .	59
5.5. Ejemplo de vistas para listado y creación de almacenes en views.py. Fuente: Elaboración propia en Django.	60
5.6. Configuración de rutas principales en urls.py. Fuente: Elaboración propia en Django. . .	61

Introducción

En la actualidad, la gestión eficiente de inventarios se ha convertido en un factor clave para el éxito de las empresas. Un control adecuado de los bienes y materiales no solo optimiza los procesos operativos, sino que también contribuye a la reducción de costos y mejora la toma de decisiones. En este contexto, el desarrollo de sistemas automatizados que integran bases de datos con interfaces *web* se presenta como una solución eficaz para la gestión de inventarios.

El presente informe tiene como propósito detallar el trabajo realizado durante el periodo laboral en una empresa de licitaciones gubernamentales y privadas, específicamente el desarrollo de un sistema de inventario que permita administrar eficientemente el flujo de entrada y salida de equipos tecnológicos en la organización.

El sistema está diseñado para ofrecer una interfaz amigable y útil al usuario mediante una aplicación *web*. Asimismo, se encuentra alojado dentro de la infraestructura interna de la empresa, configurado en un entorno virtualizado dedicado provisto por la propia organización.

Para el desarrollo del sistema se identificaron cuatro pilares esenciales: la base de datos relacional, la interfaz de usuario, la conexión lógica entre ambos componentes y el despliegue en la infraestructura local. Cada aspecto del proyecto se implementó utilizando herramientas tecnológicas consolidadas: para la gestión de datos se emplea **MySQL**; para la interfaz de usuario se utilizan **HTML**, **CSS** y **JavaScript**; para conectar la interfaz con la base de datos se emplea **Python** con el *framework* **Django**; y para el alojamiento (*hosting*) del aplicativo se utiliza una Máquina Virtual dedicada en la red interna.

A continuación se presentan los objetivos del proyecto, los cuales sirvieron de directriz en cada etapa del desarrollo:

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar e implementar un sistema de inventario automatizado que permita llevar un registro detallado, confiable y actualizado de los activos tecnológicos de la empresa, gestionando con precisión el flujo de entradas, salidas, asignaciones, estados operativos y ubicaciones físicas dentro de la organización.

1.1.2. Objetivos Específicos

- Desarrollar una base de datos relacional con **MySQL** que permita almacenar, actualizar y consultar eficientemente la información correspondiente a los equipos tecnológicos.
- Crear una interfaz *web* funcional e intuitiva mediante **Django**, permitiendo una interacción fluida con la información del inventario desde los equipos autorizados.
- Construir un esquema robusto de autenticación y control de acceso basado en roles (**RBAC**), garantizando el acceso seguro a las funciones del sistema conforme a la jerarquía operativa de la empresa.
- Configurar el despliegue del sistema dentro de la infraestructura de la empresa mediante una Máquina Virtual dedicada con resolución por DNS local (`invconecta.local.net`), asegurando conectividad local y remota controlada.
- Automatizar la generación de reportes y resguardos en formato **PDF**, facilitando el seguimiento administrativo de los activos asignados tanto a proyectos como a empleados.

A lo largo de este informe se profundiza en los conceptos teóricos que sustentan el proyecto, la metodología empleada durante su desarrollo y las ventajas operativas que aporta frente a métodos manuales de control. Asimismo, se describen los retos técnicos solventados en las distintas capas del sistema y las decisiones de ingeniería adoptadas para lograr un resultado óptimo.

1.2. Índice Desglosado

El presente documento se estructura en cinco capítulos que exponen de forma progresiva los antecedentes, desarrollo, implementación, resultados y análisis del sistema:

Capítulo 1: Introducción. Expone el contexto general del proyecto, la problemática detectada, los objetivos planteados y el marco tecnológico utilizado.

Capítulo 2: Historia Laboral. Describe la estructura y actividad de la empresa, así como las responsabilidades y funciones desempeñadas durante la estancia profesional.

Capítulo 3: Desarrollo de un Sistema de Inventario. Presenta la justificación del proyecto y los fundamentos teóricos sobre gestión de inventarios, bases de datos relacionales, interfaces *web* y entornos de despliegue.

Capítulo 4: Implementación. Detalla el proceso técnico de construcción: reglas de negocio, modelado de datos, arquitectura de la aplicación en **Django**, topología de red, protocolo de pruebas, matriz de trazabilidad y evidencias visuales de operación.

Capítulo 5: Conclusiones. Analiza los logros de ingeniería alcanzados, las limitaciones operativas identificadas y las líneas de trabajo a futuro para la evolución de la plataforma.

El informe concluye con los apartados de **Glosario**, **Referencias** bibliográficas bajo la norma IEEE y **cuatro Anexos**, los cuales integran el Diagrama Entidad–Relación, el Diagrama Relacional y los fragmentos representativos de código fuente en SQL y Django.

Capítulo 2

Historia Laboral

El desarrollo del sistema de inventario que se expone en el presente informe fue elaborado durante el periodo que laboré en una empresa de licitaciones gubernamentales y privadas. Esta empresa se especializa en la ejecución de proyectos de carácter tecnológico, con gran énfasis en los contratos gubernamentales y, en menor medida, brinda atención al sector privado. Dicho esquema presenta el manejo de un gran volumen de equipos distribuidos en varias ubicaciones.

Mi trabajo en la empresa se centró en el diseño, desarrollo, implementación y despliegue de un sistema de inventario, con la capacidad de ofrecer acceso remoto mediante una interfaz web. Este proyecto representó el núcleo de mis responsabilidades durante mi relación laboral con la empresa.

Aunque el desarrollo del sistema de inventario representó mi actividad principal, también realicé tareas complementarias correspondientes al área de soporte web. Actividades como el mantenimiento, actualización, soporte y mejora del sitio web empresarial me fueron asignadas. Esta combinación de tareas me permitió poner en práctica habilidades adquiridas durante la carrera, todo ello en un entorno real. A la par, fortalecí mis habilidades técnicas, mejoré mi comprensión de temas que creía olvidados y corregí puntos débiles que ni sabía que tenía.

2.1. Empresa de Licitaciones Gubernamentales y Privadas

2.1.1. Historia

La empresa en la que me desempeñé fue fundada en el año 1999. Nació como respuesta a la necesidad de participar en grandes proyectos que tanto el sector gubernamental como el privado querían ejecutar. Aunque hoy por hoy su principal cliente es el Gobierno Federal, el sector privado aún se mantiene vigente como un actor relevante en los objetivos de la empresa, compitiendo principalmente por proyectos de diversa índole tecnológica.

Cuenta con una sólida trayectoria en la ejecución y gestión integral de proyectos tecnológicos, proveyendo soluciones que van desde el abasto de equipos especializados hasta la asignación de personal capacitado para el despliegue, instalación, mantenimiento y soporte técnico. Un factor diferenciador de la empresa se encuentra en su modelo de trabajo colaborativo, el cual promueve la integración de los distintos

actores del proyecto en un equipo de trabajo unificado que responde en conjunto a las necesidades del cliente y del equipo operativo, lo cual contribuye a la agilidad en la toma de decisiones, así como a la eficiencia en la ejecución de los proyectos.

A lo largo de su trayectoria, la empresa también ha consolidado relaciones con distintos proveedores, lo que permite responder con flexibilidad a los requerimientos logísticos de cada proyecto, otorgando tiempos de entrega y precios competitivos en el mercado. En el ámbito interno, la empresa se ha mantenido a la vanguardia mediante la mejora constante de sus procesos corporativos, logrando un seguimiento sólido, eficaz y dinámico de sus propuestas.

En lo referente al ámbito tecnológico, la empresa se mantiene actualizada mediante el monitoreo de las tendencias del mercado, eligiendo y proveyendo equipos que se adecuen de manera óptima a los requerimientos de cada proyecto, lo cual garantiza un alto nivel de eficacia, calidad y cumplimiento técnico, a un costo adecuado.

2.1.2. Misión y Visión

Misión

Mejorar la disponibilidad y el desempeño operativo de las diversas entidades en la totalidad del territorio nacional, mediante la ejecución de soluciones tecnológicas integrales. Estas soluciones comprenden el suministro de *hardware*, desarrollo y personalización de *software*, soporte técnico especializado y servicios administrados.

La filosofía empresarial se basa en la convicción de que su crecimiento y consolidación dependen del trabajo conjunto con sus principales grupos de interés: clientes, empleados, accionistas, proveedores y la sociedad en general. En concordancia con este enfoque, la empresa promueve los siguientes valores institucionales:

- Confianza y respeto por las personas.
- Atención a las necesidades de los involucrados.
- Consideración de expectativas comunes.
- Actitud de servicio proactiva.
- Trabajo colaborativo y en equipo.
- Responsabilidad en el cumplimiento de compromisos.

Visión

La empresa aspira a consolidarse como una organización líder en su sector, reconocida por sus clientes, colaboradores, accionistas, proveedores y la sociedad como una institución modelo. Su propósito es mantener un equilibrio entre el bienestar propio y el de sus aliados estratégicos, mediante la mejora continua de sus procesos y servicios, enfocados principalmente en soluciones informáticas orientadas a

corporativos y entidades gubernamentales. A largo plazo, se busca inspirar respeto y confianza con todos los actores con los que se relaciona.

2.1.3. Divisiones

La empresa está configurada en diversas secciones o departamentos, cada uno de ellos con labores y responsabilidades claramente definidas. Aunque cada área tiene su rol definido, la comunicación entre ellas resulta indispensable; esto permite una respuesta rápida a la publicación de licitaciones, además de otorgar una gran sinergia en la búsqueda de soluciones óptimas a los problemas que se presentan durante todo el proceso de ejecución de proyectos. A continuación se describen las principales secciones de la empresa:

Contabilidad: Es el área responsable de la salud financiera de la empresa. Entre sus funciones se encuentran: el control de gastos, el control de ingresos, la elaboración de nóminas, el análisis de costo-beneficio por proyecto y el cumplimiento de las obligaciones fiscales.

Licitaciones: Es el área encargada de llevar a cabo la correcta participación de la empresa en los concursos que se publican, tanto públicos como privados. Este departamento revisa los requisitos de cada convocatoria, elabora la documentación técnica y administrativa, da seguimiento al estatus de los concursos, y genera informes de retroalimentación en caso de no resultar ganadores.

Proyectos: Su función principal es la planificación, ejecución y supervisión de los proyectos adjudicados. Se encargan de la adquisición, configuración, distribución y mantenimiento de equipos tecnológicos, así como de la coordinación del personal técnico tanto en campo como en remoto.

Recursos Humanos: Este departamento gestiona el ciclo completo de vida del personal, desde el reclutamiento y la selección hasta su desarrollo profesional y bienestar organizacional [1]. Su objetivo es asegurar que la empresa cuente con personal competente, capacitado y alineado con los valores institucionales.

Soporte Técnico: Responsable del mantenimiento y correcto funcionamiento de la infraestructura tecnológica interna. Esto incluye computadoras, impresoras, servidores, *routers*, cableado estructurado y redes inalámbricas. También se encarga de la administración y actualización del sitio web institucional.

2.2. Historia laboral en la empresa

Durante el periodo laboral en el que se llevó a cabo el proyecto descrito en este informe, me desempeñé como **Desarrollador de Software** en el área de Sistemas y Tecnologías de la Información de la empresa. Mi responsabilidad primordial consistió en el análisis, diseño, desarrollo y puesta en marcha de soluciones tecnológicas a la medida, teniendo como eje central la creación de un sistema de inventario automatizado para centralizar y optimizar el control de los activos tecnológicos de la organización.

De manera complementaria a las actividades de desarrollo del sistema, asumí la administración técnica y el mantenimiento evolutivo del sitio web institucional desarrollado en *WordPress*. Estas labores combinaron la ingeniería de software con la gestión de infraestructura web, abarcando desde la programación y modelado de datos hasta el despliegue en servidores y la resolución de incidencias en producción.

2.2.1. Administración y Mantenimiento del Sitio Web

Como parte de las responsabilidades asignadas para la presencia digital de la empresa, asumí la administración técnica y operativa del portal web institucional. Las funciones desempeñadas en esta área incluyeron:

Actualización y mantenimiento: Ejecución periódica de actualizaciones de versiones del núcleo de *WordPress*, del intérprete *PHP* y de los *plugins* de seguridad, contacto y filtrado de correo no deseado (*antispam*).

Desarrollo de módulos y diseño web: Reestructuración y reprogramación de la sección principal (*home*) del sitio para optimizar la navegación del usuario y homologar la identidad corporativa.

Monitoreo y soporte de infraestructura: Supervisión de la disponibilidad del servidor y gestión de incidencias técnicas mediante la apertura y seguimiento de reportes con el proveedor de alojamiento (*hosting*).

2.2.2. Desarrollo y Modelado de la Base de Datos

En el marco del desarrollo del sistema de inventario automatizado de la empresa, asumí la responsabilidad técnica de diseñar, estructurar y administrar el motor de base de datos relacional que daría soporte a las operaciones corporativas. Mis funciones y responsabilidades en este rubro comprendieron:

- **Levantamiento de requerimientos con usuarios finales:** Coordinación de sesiones de trabajo y entrevistas con los responsables de almacén y proyectos para determinar los flujos de entrada, salida y resguardo de equipos.
- **Control de avances y documentación técnica:** Elaboración de bitácoras semanales y reportes de seguimiento para la dirección general sobre el estado del diseño y la mitigación de riesgos.
- **Definición de reglas y modelado conceptual:** Establecimiento de las reglas de negocio organizacionales y diseño de los esquemas conceptuales y lógicos de datos.
- **Implementación y optimización en MySQL:** Creación física del esquema, configuración de restricciones de integridad referencial, programación de disparadores automáticos (*triggers*) y optimización mediante índices.

2.2.3. Desarrollo de la Aplicación Web e Integración

Para la interacción de los colaboradores con la base de datos, se me encomendó la construcción integral de la plataforma web del sistema de inventario. Las responsabilidades desempeñadas abarcaron:

- **Construcción de la plataforma en Django:** Desarrollo de la lógica de negocio en el *backend* utilizando **Python** e integración de la interfaz de usuario mediante plantillas web dinámicas.
- **Implementación de seguridad y roles (RBAC):** Programación de los flujos de autenticación y asignación de permisos según los cuatro niveles jerárquicos definidos por la empresa.
- **Generación de reportes institucionales:** Automatización de la emisión de resguardos individuales y actas de proyecto en formato **PDF**.
- **Control de versiones y calidad:** Uso de **Git** y **GitHub** para la gestión del código fuente, asegurando la trazabilidad de los cambios y puntos de restauración durante el ciclo de vida del desarrollo.

2.2.4. Despliegue y Puesta en Operación en Infraestructura Corporativa

Una vez concluida la etapa de desarrollo y validación de la plataforma, se me asignó la responsabilidad de coordinar y ejecutar la puesta en marcha del sistema dentro de la infraestructura tecnológica de la empresa. Las actividades laborales desempeñadas en este rubro abarcaron:

- **Aprovisionamiento y configuración del entorno de ejecución:** Preparación de la Máquina Virtual (*VM*) dedicada provista por el área de TI, configurando el intérprete de **Python**, el entorno virtual aislado y la instalación de dependencias requeridas para producción.
- **Aseguramiento y variables de entorno:** Ajuste de los parámetros operativos en el archivo de configuración del aplicativo (`settings.py`), desactivando el modo de depuración (`DEBUG = False`) y definiendo las políticas de acceso mediante `ALLOWED_HOSTS`.
- **Integración con la red corporativa y DNS local:** Verificación del enlace con el motor de base de datos **MySQL** interno y comprobación de la resolución del nombre de dominio local (`invconecta.local.net`) para el acceso de los colaboradores desde la red de área local (*LAN*).
- **Capacitación y soporte inicial a usuarios:** Elaboración de guías operativas básicas y acompañamiento al personal administrativo y técnico durante la fase de transición y registro de los primeros activos en el sistema.

Desarrollo de un Sistema de Inventario

Para la implementación del sistema, se utiliza una base de datos relacional desarrollada en **MySQL**, con el propósito de aprovechar las ventajas que este motor ofrece en términos de integridad, consistencia y eficiencia en el manejo de datos estructurados.

El acceso al sistema se realiza a través de una interfaz web, diseñada para ser accesible desde cualquier equipo de cómputo con conexión a internet. Esta interfaz proporciona funcionalidades de consulta, registro, actualización y generación de reportes, adaptadas a los distintos perfiles de usuario definidos mediante un esquema jerárquico de permisos.

El sistema tiene como objetivo mejorar la trazabilidad de los activos de la empresa, reducir el error humano asociado a la gestión manual del inventario y mejorar la toma de decisiones, teniendo a disposición información clara y precisa en tiempo real.

3.1. Antecedentes

La puesta en marcha de un sistema de inventario requiere la integración de diversas áreas del campo de la informática. Entre las áreas que se incluyeron en el desarrollo se encuentran la gestión de datos, diseño *backend*, diseño *frontend*, automatización, virtualización y seguridad de la información. Para otorgar un sólido sustento a la metodología que se usó para el desarrollo del sistema de inventario, a continuación se presentan los fundamentos conceptuales que guiaron dicho desarrollo.

3.1.1. Gestión de Inventario

La gestión de inventario es un proceso crítico en el funcionamiento eficiente de las organizaciones, ya que permite mantener un control preciso sobre los bienes y recursos disponibles. Una administración adecuada del inventario contribuye a prevenir desabastos, evitar acumulaciones innecesarias de materiales, cumplir con los compromisos contractuales y optimizar los costos operativos.

Diversos autores han definido el inventario desde diferentes enfoques:

Meana Coalla (2017) establece que:

“El objetivo del inventario es confirmar o verificar el tipo de existencias de que disponemos en la empresa, mediante un recuento físico de los materiales existentes.” (p. 3) [2]

Fernández, A. C. (2018) lo describe como:

“Un inventario consiste en un listado ordenado, detallado y valorado de los bienes de una empresa.” (p. 2) [3]

Por su parte, Montes, J. L. (2014) ofrece una visión más amplia:

“Registro documental de los bienes y demás cosas pertenecientes a una persona o comunidad, hecho con orden y precisión, pero en el mundo empresarial es la relación ordenada de bienes y existencias, a una fecha determinada.” (p. 14) [4]

Aunque cada definición resalta distintos aspectos —ya sea el recuento físico, la valoración económica o la precisión documental— todas coinciden en que el inventario es una herramienta indispensable para lograr un control riguroso de los activos de una entidad. En el ámbito empresarial, esta gestión no se limita únicamente al registro de bienes, sino que se convierte en una herramienta estratégica para la planificación operativa, la optimización de recursos y la toma de decisiones fundamentadas.

3.1.2. Tipos de Bases de Datos

En el entorno tecnológico contemporáneo, las bases de datos representan un componente esencial para el almacenamiento, recuperación y gestión eficiente de información [5]. De manera general, estas se dividen en dos grandes categorías: bases de datos **SQL** (relacionales) y bases de datos **NoSQL** (no relacionales), las cuales ofrecen arquitecturas flexibles para datos semiestructurados [6]. Cada tipo responde a distintas necesidades de estructura, volumen y acceso a los datos, dependiendo del contexto de uso.

- **SQL**

Las bases de datos relacionales utilizan el lenguaje estructurado de consultas (**SQL**, por sus siglas en inglés) para la gestión de datos estructurados. La información se organiza en tablas con filas y columnas, lo que permite establecer relaciones lógicas claras entre diferentes entidades. Este tipo de base de datos es ideal para aplicaciones donde se requiere una estructura rígida, validaciones estrictas y consistencia transaccional, como en sistemas de contabilidad, gestión de clientes (*CRM*) o plataformas empresariales. Algunos ejemplos ampliamente utilizados incluyen **MySQL**, **PostgreSQL** y **Microsoft SQL Server**.

- **NoSQL**

Las bases de datos **NoSQL** están diseñadas para almacenar datos no estructurados o semiestructurados, ofreciendo una arquitectura más flexible en comparación con las bases de datos relacionales. Estas no requieren un esquema tabular fijo, lo que las hace ideales para manejar grandes volúmenes de datos distribuidos que cambian con frecuencia. Los modelos de almacenamiento en **NoSQL**

incluyen documentos (ej. *MongoDB*), pares clave-valor (ej. *Redis*), bases de datos en columnas (ej. *Cassandra*) y grafos (ej. *Neo4j*). Se utilizan comúnmente en aplicaciones que demandan alta escalabilidad, disponibilidad y rendimiento en tiempo real, como redes sociales, análisis de grandes volúmenes de datos y plataformas móviles.

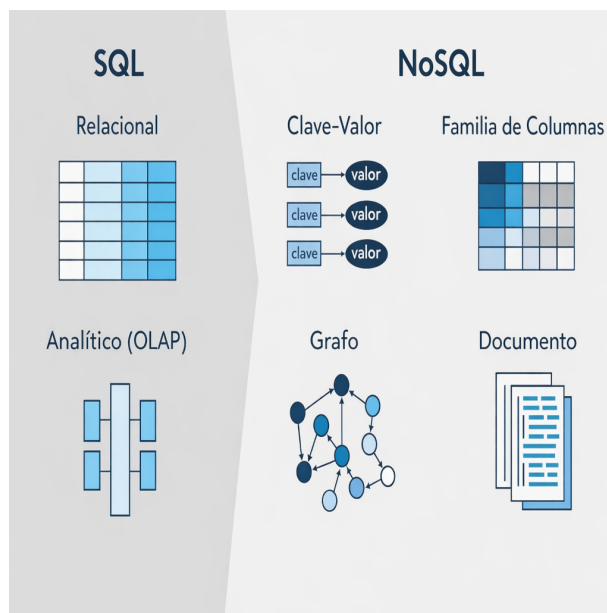


Figura 3.1: Tipos de Bases de Datos (Fuente: [7]).

Cada paradigma tiene ventajas específicas orientadas a distintos requerimientos de desarrollo. Las bases de datos **SQL** destacan por garantizar la **integridad de los datos** mediante restricciones lógicas (*constraints*, llaves foráneas y unicidad) y la **consistencia transaccional** bajo el cumplimiento de las propiedades ACID (*Atomicity, Consistency, Isolation, Durability*). Esto resulta indispensable en un sistema de inventario, donde operaciones críticas —como la transferencia de equipos entre almacenes, el registro simultáneo de salidas o la asignación de activos a proyectos— exigen que todas las modificaciones se ejecuten de manera atómica e indivisible para evitar estados inconsistentes o pérdidas de registro. Por su parte, las bases de datos **NoSQL** proporcionan escalabilidad horizontal y esquemas dinámicos, siendo preferidas en aplicaciones con volúmenes masivos de datos no estructurados o con alta tolerancia a la consistencia eventual.

En el caso del sistema de inventario desarrollado para la empresa, se optó formalmente por una base de datos relacional (**SQL**) implementada en **MySQL**. Dicha elección se fundamentó en la naturaleza estructurada de la información operativa y en la necesidad estricta de asegurar la consistencia transaccional y la integridad referencial en cada movimiento, resguardo y consulta de los activos tecnológicos.

3.1.3. Bases de Datos SQL

De acuerdo con Escofet (2002), **SQL** (*Structured Query Language*) es el lenguaje estándar ANSI/ISO para la definición, manipulación, consulta y control de bases de datos relacionales [8]. A diferencia de los lenguajes procedimentales, **SQL** es un lenguaje declarativo: permite a los usuarios especificar qué desean obtener de una base de datos, sin necesidad de indicar cómo debe lograrse ese resultado. Esta característica lo posiciona como una herramienta potente y de fácil uso, ya que su sintaxis se asemeja al lenguaje natural en inglés.

SQL cuenta con funciones básicas sumamente valiosas, como lo son la recuperación, inserción, actualización y eliminación de datos mediante sentencias simples y claras, pero sumamente expresivas. El funcionamiento general de SQL se puede conocer en cuatro pasos esenciales:

1. **Conexión a la base de datos:** El primer paso es establecer una conexión con un sistema de gestión de bases de datos (DBMS), el cual contiene la información persistente.
2. **Redacción de consultas:** Se requiere redactar una sentencia con instrucciones en SQL para solicitar a la base la consulta, inserción o modificación de datos.
3. **Ejecución de la consulta:** Una vez que se ha redactado la consulta, es necesario ejecutarla. El DBMS interpreta las instrucciones enviadas en la consulta y ejecuta el plan de acción más eficiente.
4. **Retorno de resultados:** Como paso final del proceso, el sistema devuelve los datos solicitados o confirma la operación realizada.

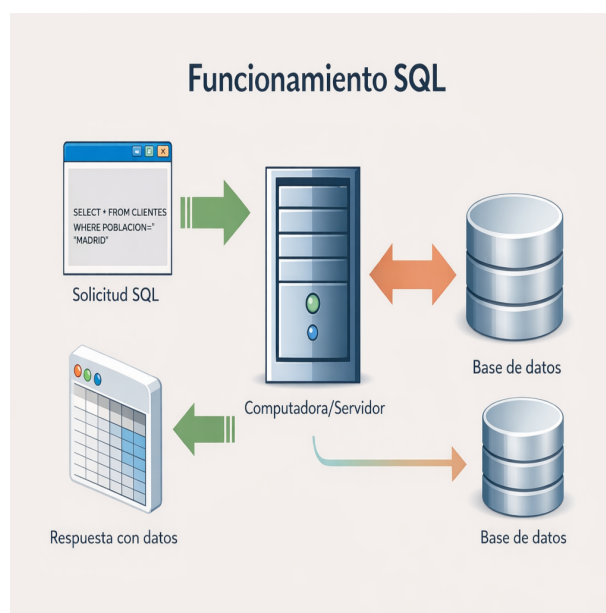


Figura 3.2: Funcionamiento de SQL (Fuente: [9]).

El motivo por el que SQL desempeña un papel fundamental en la gestión de datos responde a múltiples razones:

- **Estandarización:** SQL es un lenguaje normalizado que se rige bajo los estándares ANSI e ISO, lo cual asegura una correcta integración con una amplia variedad de sistemas como lo son PostgreSQL, MySQL, MariaDB, Microsoft SQL Server y Oracle, por mencionar algunos de los más relevantes.
- **Gestión eficiente de datos:** SQL permite un manejo ágil de grandes cantidades de información y el uso de operaciones de escritura y lectura sumamente optimizadas.
- **Consultas complejas:** Permite realizar operaciones avanzadas que involucran múltiples tablas, filtros condicionales y funciones de agregación para generar reportes analíticos.
- **Seguridad y control de acceso:** Proporciona mecanismos de autenticación y definición de permisos mediante roles para resguardar la integridad de la información.
- **Integración tecnológica:** Su naturaleza estándar le permite integrarse con sistemas *ERP*, plataformas de *Business Intelligence (BI)* y *frameworks* como **Django**.

En el caso del sistema de inventario desarrollado para la empresa, el uso de **SQL** mediante el motor **MySQL** resultó idóneo para estructurar y administrar la información de manera eficiente, segura y escalable, dada la necesidad de mantener relaciones claras entre entidades como equipos, almacenes, proveedores, empleados y proyectos.

3.1.4. Interfaces de Usuario - Web

La interfaz de usuario (*UI*, por sus siglas en inglés) constituye el punto de contacto entre una persona y un sistema digital. Es el medio por el cual los usuarios visualizan, interactúan y comunican sus necesidades a través del *software*.

Albornoz (2014) lo define de la siguiente manera:

“La Interfaz de Usuario es la parte del *software* que las personas pueden ver, oír, tocar, hablar; es decir, donde se pueden entender.” (p. 2) [10]

Aunado a la *UI*, converge el concepto de experiencia de usuario (*UX*), el cual hace referencia a la percepción general que tiene el usuario como resultado de su interacción con el sistema.

En palabras de Ramírez-Acosta (2017):

“Mientras que *UI (user interface)* se refiere a la interfaz visual de una herramienta de *software*, *UX (user experience)* se refiere específicamente a la experiencia del usuario mientras la utiliza.” (s/p) [11]

Desde una perspectiva funcional, la interfaz de usuario puede analizarse a partir de dos componentes esenciales:

Entrada: Es la forma mediante la cual una persona comunica una acción o intención al sistema. Entre los dispositivos de entrada más comunes se encuentran el teclado, el ratón, la pantalla táctil y la entrada por voz.

Salida: Es la respuesta del sistema hacia el usuario. Comúnmente se manifiesta a través de la pantalla, pero también puede incluir sonidos, notificaciones visuales o mensajes hablados.

Un buen diseño de una interfaz proporciona una interacción fluida entre el usuario y el sistema, lo cual permite que los procesos se lleven a cabo de manera precisa, rápida, intuitiva y sin errores, incrementando la eficiencia operativa del sistema y la satisfacción del usuario. Por otro lado, un diseño erróneamente ejecutado puede generar confusión, frustración e incremento en los errores, conllevando a una pérdida de productividad.

En el desarrollo de software actual, la UI se ha posicionado como un elemento decisivo. En la gran mayoría de casos, la UI es el primer contacto que tiene el usuario con el sistema, por lo que representa un factor crítico del éxito o fracaso de dicho sistema. No obstante, es conveniente aclarar que una buena interfaz por sí sola no es suficiente. Es necesaria una integración armónica con la calidad de los datos, la estructura lógica del sistema y la eficiencia del *backend*. Solo así será posible conseguir una solución útil, coherente y robusta [11].

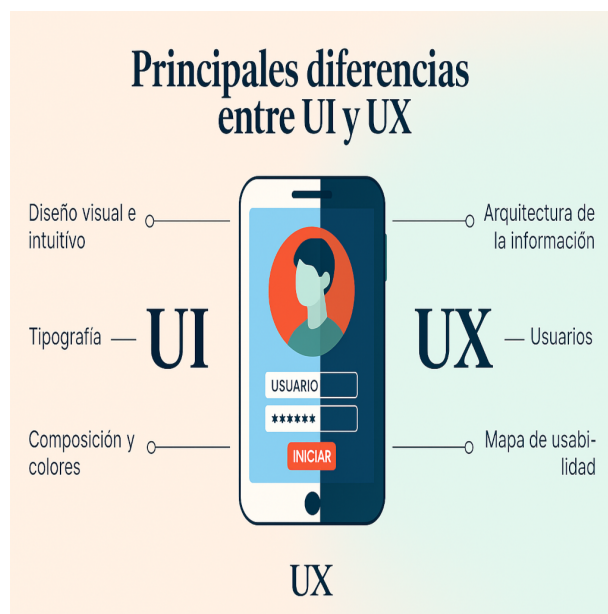


Figura 3.3: User Interface vs. User Experience (Fuente: [12]).

Durante el desarrollo de este proyecto, fue necesario crear una interfaz web que cumpla con los principios mencionados, buscando ofrecer siempre una experiencia accesible, clara y funcional, adaptada al perfil de los empleados de la empresa, los cuales serían los usuarios finales [10].

3.1.5. Conexión entre bases de datos y aplicaciones web

Un punto fundamental del desarrollo de software contemporáneo es la conexión entre la base de datos y la interfaz web. Esta integración permite un manejo eficiente de información dinámica en tiempo real de la base de datos a través del uso de interfaces accesibles que se despliegan desde cualquier navegador

web. Esto permite que los usuarios consulten, registren o modifiquen información almacenada en sistemas remotos, lo cual es especialmente útil en entornos distribuidos o multiusuario.

Según Coronel y Morris (2017), una base de datos es útil en tanto pueda comunicarse de forma eficiente con las aplicaciones que la utilizan, y esta comunicación debe garantizar integridad, velocidad y seguridad en el manejo de los datos [13]. En este contexto, las aplicaciones *web* operan como intermediarias entre el usuario y el sistema de almacenamiento, procesando las solicitudes mediante el *backend* y entregando respuestas visuales a través del *frontend*.

La conexión se realiza mediante tecnologías de integración como los mapeos objeto-relacionales (**ORM**, por sus siglas en inglés), los cuales permiten representar las estructuras de una base de datos como objetos dentro del código de la aplicación. Esto facilita la escritura, lectura y actualización de datos sin necesidad de recurrir directamente a sentencias **SQL**. Según Fowler (2003), la implementación de patrones como el *Data Mapper* facilita una arquitectura que mejora la mantenibilidad y escalabilidad del *software*, contribuyendo a un desarrollo más eficiente y seguro [14].

Para establecer esta comunicación, es necesario configurar parámetros como el nombre de la base de datos, el usuario y contraseña, el *host* y el puerto de acceso. Como se ilustra en la Figura 3.4, una vez activa, la aplicación puede ejecutar transacciones de forma automatizada y transparente para el usuario final.

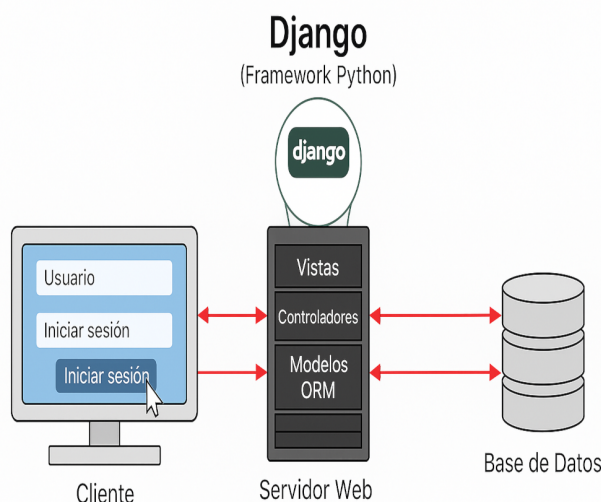


Figura 3.4: Conexión a la base de datos (Fuente: [15]).

La correcta implementación de esta conexión asegura la coherencia de la información, permite escalar el sistema según la demanda y mejora significativamente la experiencia del usuario, al permitirle interactuar con los datos de forma inmediata, desde cualquier ubicación con acceso a internet.

3.1.6. Servidor Web Empresarial (Hosting Propio)

En el desarrollo de aplicaciones *web*, un servidor *web* empresarial es un componente esencial para alojar y ejecutar sistemas que requieren acceso remoto. A diferencia de los entornos locales de desarrollo, un servidor empresarial está configurado para estar disponible de forma continua, permitiendo que usuarios autorizados puedan acceder a la aplicación desde diferentes estaciones de trabajo o ubicaciones remotas.

Un servidor *web* empresarial actúa como un *hosting* propio (*on-premise*), gestionado y controlado directamente por la organización, en lugar de depender de proveedores de alojamiento externos. Esto ofrece múltiples ventajas, como la posibilidad de personalizar configuraciones de seguridad, administrar directamente las actualizaciones y optimizar los recursos según las necesidades específicas del sistema, tal como se ilustra en la Figura 3.5.



Figura 3.5: ¿Cómo funciona el alojamiento web? (Fuente: [16]).

En términos técnicos, un servidor *web* empresarial incluye componentes como:

- Un servidor de aplicaciones (por ejemplo, **uWSGI**, **Gunicorn** o el entorno de ejecución del *framework*) que ejecuta la lógica del sistema.
- Un servidor **HTTP** (como **Nginx** o **IIS**) que gestiona las solicitudes de los usuarios y sirve archivos estáticos.
- Un sistema gestor de bases de datos (como **MySQL**, **PostgreSQL** o **SQL Server**) para almacenar y procesar la información.
- Configuraciones específicas de red y seguridad (puertos, cortafuegos, certificados **SSL/TLS** o aislamiento perimetral) para garantizar la confidencialidad e integridad de los datos [17], [18].

La utilización de un servidor empresarial o entorno virtualizado propio permite el despliegue de aplicaciones *web* las 24 horas del día. Este servicio es manejado directamente por la empresa y acoplado a las necesidades particulares del sistema, como lo son la escalabilidad, el acceso remoto controlado y el control de datos sensibles.

Este tipo de implementación es sumamente útil para empresas que requieren flexibilidad, seguridad y, sobre todo, control total sobre el despliegue de su infraestructura tecnológica, al posibilitarles operar sistemas críticos sin depender de servicios en la nube pública o alojamiento de terceros [19].

3.2. Definición del Problema

Dentro de la empresa, el manejo de los equipos asignados a diversos contratos y ubicaciones operativas se lleva a cabo de forma fragmentada, haciendo uso de métodos manuales o herramientas no especializadas como hojas de cálculo, archivos PDF, procesadores de texto y registros impresos. Dicho enfoque, aunque dio resultados en etapas iniciales, se mostró insuficiente ante el crecimiento de la empresa y la complejidad operativa asociada al manejo de diversos proyectos en simultáneo.

Este escenario presenta una serie de dificultades que erosionaba directamente la productividad y la eficiencia administrativa y operativa. Entre los principales inconvenientes encontrados se aprecian los siguientes:

- Dificultad para hallar equipos específicos de manera rápida y precisa.
- Falta de seguimiento en el registro de asignaciones y reubicaciones de los equipos.
- Persistencia de errores en los registros de las asignaciones, entradas y salidas de los equipos.
- Imposibilidad para acceder a la información de los equipos de forma remota, en situaciones donde era necesaria la toma de decisiones rápidas y precisas.

Aunado a esto, se encontró que el uso de métodos tradicionales conllevaba una carga de trabajo elevada para el personal responsable del inventario, con un alto riesgo de pérdida de información, inconsistencia en los datos y una baja capacidad de escalabilidad. Conforme la empresa incrementa el número de contratos y proyectos, aumenta la cantidad de equipos y ubicaciones donde almacenarlos, lo que evidenció aún más la necesidad de adoptar una solución más robusta y automatizada.

A partir del análisis realizado, se encontró como problema principal la falta de un sistema de inventario acoplado a las necesidades de la empresa, que a su vez esté digitalizado, centralizado y accesible remotamente, lo que permitiría una gestión eficiente de los movimientos de los equipos de la empresa, a la par que otorgaría un control sobre los registros históricos de cada activo y brindaría acceso seguro a la información desde cualquier ubicación autorizada y en cualquier momento.

3.3. Análisis de la Solución

Una vez que se definió el problema, se presentó como solución el desarrollo de un sistema de inventario digital basado en tecnologías informáticas y bases de datos relacionales. Esta metodología permite abordar los principales desafíos de disponibilidad, confiabilidad y trazabilidad en el manejo de los equipos de la empresa, todo ello mediante el uso de una infraestructura de fácil mantenimiento y alta capacidad de escalabilidad. Las principales ventajas de esta solución son:

Centralización de datos: La información del inventario se almacena en una base de datos **SQL** estructurada, permitiendo la consulta y modificación en tiempo real desde una única fuente de verdad.

Acceso remoto y seguro: La interfaz del sistema, desarrollada en **Django**, permite a los usuarios autorizados acceder al sistema desde cualquier ubicación mediante conexión a internet, implementando *login* y permisos según el rol del usuario.

Automatización de procesos: El sistema permite generar reportes automáticos en **PDF**, registrar entradas y salidas, y mantener un historial de movimientos, lo cual reduce la carga operativa del personal.

Escalabilidad y mantenimiento: Gracias al uso de tecnologías modernas y de código abierto como **Python** y **Django**, el sistema es fácil de mantener y escalar a futuro según las necesidades de la empresa.

La solución a la que se llegó integra distintas tecnologías para construir un sistema robusto, seguro y accesible, orientado a resolver los problemas de trazabilidad, disponibilidad y confiabilidad de la información del inventario de la empresa.

Implementación

La implementación del sistema de inventario realizado en la empresa se guió por un enfoque flexible, lo que permitió que las herramientas tecnológicas y metodológicas fueran seleccionadas libremente conforme al contexto operativo y los objetivos del proyecto.

Antes de elegir las herramientas, el proyecto comenzó con una fase de exploración e investigación comparativa, por lo cual analizamos diferentes tecnologías y marcos de desarrollo para las bases de datos y las aplicaciones web. Para ello, se revisó documentación técnica, se realizaron pruebas funcionales y se evaluaron las ventajas, desventajas y compatibilidades entre los distintos componentes.

Las herramientas y estrategias se escogieron para garantizar la robustez, escalabilidad, seguridad y facilidad de mantenimiento del sistema. La arquitectura tecnológica se compuso de **MySQL** como gestor de base de datos relacional y **Django** como *framework* de desarrollo web, además de metodologías de diseño iterativo para asegurar el mejoramiento constante del sistema durante su elaboración.

A continuación, se presenta la composición del sistema, la cual comprende el desarrollo de la base de datos, la construcción de la interfaz web, la configuración del acceso remoto y la validación del sistema mediante pruebas funcionales.

4.1. Metodología para el Desarrollo de la Base de Datos

En un sistema de inventario se requiere almacenar, organizar y permitir el acceso a toda la información relacionada con los equipos tecnológicos, sus movimientos, asignaciones y estados. Es por ello que una base de datos es fundamental; su diseño e implementación requirieron un enfoque sistemático, riguroso y alineado con las necesidades operativas de la empresa.

4.1.1. Reglas de negocio

Las reglas de negocio son las directrices formalmente establecidas que guían el comportamiento del sistema. Estas reglas deben establecer las restricciones, condiciones y relaciones del sistema, siguiendo la lógica operativa de la empresa. Estas directrices fueron definidas tras entrevistas con el personal involucra-

do, análisis de flujos operativos y documentación interna. A continuación, se presentan categorizadas por áreas funcionales:

1. Gestión de Almacenes

- Cada almacén debe tener un nombre único y una ubicación física específica.
- La cantidad de equipos en un almacén no puede ser negativa.
- La suma de equipos vendidos o desechados no puede superar la cantidad total existente.
- Cada modificación en el almacén debe registrar automáticamente la fecha de actualización.

2. Gestión de Equipos

- Todo equipo debe estar asociado a un registro válido en el catálogo de equipos.
- Cada equipo debe estar vinculado a un pedido y a un almacén al momento de su registro.
- Si se registra una salida, debe existir previamente una entrada y una relación válida en la tabla correspondiente.
- Un equipo solo puede estar vinculado a un proyecto existente con fechas definidas de inicio y conclusión.

3. Gestión de Proveedores

- Cada proveedor debe tener un **RFC** único; no se permiten duplicados.
- Se pueden registrar múltiples teléfonos por proveedor, siempre que sean distintos.
- Un pedido solo puede realizarse a proveedores registrados.
- Todo pedido debe incluir al comprador responsable y los equipos solicitados.

4. Gestión de Pedidos

- Los pedidos deben estar vinculados a un proveedor y a un empleado registrado como comprador.
- No se permite registrar pedidos con cantidades de equipos iguales o menores a cero.
- El precio unitario de cada equipo debe ser mayor que cero.
- El sistema debe calcular automáticamente el costo total del pedido, incluyendo el **IVA** (16%).

5. Gestión de Proyectos

- Cada proyecto debe registrar la fecha de inicio, la fecha de término y la duración calculada automáticamente en meses.
- Los proyectos deben asociarse a una dirección y a un contacto válidos.
- No se puede asociar un proyecto a contactos o ubicaciones no registradas previamente.

6. Gestión de Reparaciones

- Un equipo solo puede ser reparado si está previamente registrado en el sistema.
- La fecha de reparación debe ser posterior o igual a la fecha de ingreso del equipo.
- Cada reparación debe estar vinculada a un reparador con **RFC** único.
- No se permiten reparadores duplicados con el mismo **RFC**.

7. Gestión de Ingresos y Salidas de Equipos

- Todos los equipos deben tener una fecha de ingreso válida para ser operativos.
- Si el equipo tiene garantía, la fecha de fin debe calcularse automáticamente desde el periodo ingresado.
- No se puede registrar una fecha de salida anterior a la de ingreso.
- Toda venta debe incluir un precio mayor a cero.

8. Gestión de Empleados

- Cada empleado debe tener un **RFC** único.
- Es obligatorio que cada empleado tenga al menos un área de trabajo asignada y un correo electrónico válido.
- Un empleado debe estar previamente asociado como comprador para poder registrar pedidos.

9. Consultas y Reportes

- El sistema debe permitir consultar inventarios por almacén, proyecto o proveedor.
- Se deben generar reportes automáticos sobre equipos reparados, vendidos y desechados.
- Los reportes de pedidos deben incluir el desglose de cantidades, precios unitarios e **IVA**.

Consideraciones adicionales

Además de las reglas anteriores, se establecieron criterios específicos para garantizar la trazabilidad, el control de versiones y la consistencia en la información almacenada:

- Cada equipo cuenta con un identificador único (*Service TAG* o número de serie alfanumérico).
- Los equipos de cómputo incluyen la dirección **MAC** como identificador complementario.
- Se debe registrar el costo de compra y su estatus como activo de la empresa.
- El historial de reparaciones debe incluir la fecha de reparación, el nombre del reparador y el número de evento.
- Las características del catálogo, tales como marca, modelo y especificaciones, deben mantenerse actualizadas.

4.1.2. Diagrama Entidad-Relación (ER)

Cuando las reglas de negocio fueron definidas y validadas, el siguiente paso fue el Diagrama Entidad-Relación; este tuvo como objetivo representar las entidades fundamentales del sistema de inventario gráficamente, así como sus relaciones e interdependencias lógicas, lo que nos permitió visualizar la estructura global de la base de datos antes de su implementación física.

El diagrama **ER** constituyó la representación gráfica del proceso de modelado conceptual del sistema, permitiendo definir las entidades principales, sus atributos identificadores y las relaciones de cardinalidad (uno a uno, uno a muchos, muchos a muchos) entre ellas. Asimismo, facilitó la validación del modelo con base en los flujos operativos reales de la empresa.

Las entidades principales identificadas incluyen:

- Equipos.
- Empleados.
- Proyectos.
- Pedidos.
- Proveedores.
- Almacenes.
- Movimientos de entrada y salida.
- Reparaciones.
- Asignaciones.

Este modelo conceptual sirvió como base para construir el modelo lógico (Diagrama Relacional) y posteriormente el modelo físico en **SQL**. Su validación permitió garantizar que la estructura diseñada respondiera fielmente a los requerimientos operativos reales de la empresa.

*Nota: El diagrama completo puede consultarse en el **Anexo 1**, donde se incluye la representación detallada de las entidades, atributos y relaciones que conforman el modelo.*

4.1.3. Diagrama Relacional

A partir del Diagrama Entidad–Relación (**ER**), se procedió a construir el Diagrama Relacional, cuya finalidad es traducir el modelo conceptual a un esquema lógico implementable en un sistema de gestión de bases de datos, en este caso, **MySQL**. Este tipo de diagrama nos permite visualizar cada entidad como una tabla, mostrando de forma explícita sus claves primarias (PK), llaves foráneas (FK), sus atributos y sus tipos de relación entre ellas mediante un esquema visual fácil de entender.

Para la elaboración del diagrama se usó **ER/Studio**, un *software* especializado en el modelado de datos que proporciona una visualización integral de las estructuras relacionales, así como la capacidad de definir los tipos de datos, establecer restricciones de integridad y asegurar la coherencia entre el modelo lógico y las necesidades reales del sistema.

Durante esta etapa se prestó especial atención a establecer las reglas de integridad referencial, para lo cual se aplicaron restricciones como campo obligatorio (NOT NULL), claves únicas (UNIQUE) y cláusulas de borrado (ON DELETE) y edición (ON UPDATE) con el propósito de evitar posibles inconsistencias y mantener la coherencia de los datos a lo largo del tiempo.

Asimismo, se aplicó un proceso de normalización enfocado en reducir la redundancia en los datos. De forma complementaria, se aplicaron criterios de optimización orientados a:

- Facilitar la redacción de consultas frecuentes.
- Aumentar el rendimiento de lectura y escritura.
- Asegurar una estructura lógica flexible, clara, eficiente y escalable.

*Nota: El diagrama relacional completo, con todos los detalles asociados a él, incluyendo cada tabla y sus atributos, se encuentra disponible en el **Anexo 2** de este informe.*

4.1.4. Modelo Físico

El modelo físico es el resultado final del diseño de una base de datos. En esta etapa se conjuga la estructura lógica definida en las etapas previas dentro de un Sistema de Gestión de Bases de Datos (DBMS). Para el presente sistema se eligió utilizar **MySQL** como DBMS, decisión basada en su robustez, compatibilidad con Django, abundante información en línea en caso de fallas y amplio soporte en entornos empresariales.

En esta fase las entidades planteadas en los pasos previos fueron definidas en lenguaje SQL, se incorporaron índices en las columnas relevantes para acelerar las operaciones de búsqueda y filtrado y se implementaron restricciones de integridad que garantizan la coherencia de los datos. Dentro de cada entidad se establecieron claves primarias y foráneas, junto a reglas específicas para operaciones de borrado (ON DELETE) y edición (ON UPDATE) con el fin de mantener las relaciones entre las entidades del sistema.

Se diseñaron e integraron *triggers* (disparadores automáticos) con el objetivo de automatizar determinadas funciones críticas. Por ejemplo, se diseñó un *trigger* para actualizar el *stock* de equipos almacenados cada vez que se registra una entrada o salida de algún equipo, lo cual asegura una sincronización en tiempo real sin intervención humana.

El modelo físico fue evaluado mediante pruebas e iteraciones usando datos ficticios, con el fin de comprobar su rendimiento, posibles fallas de diseño, el correcto funcionamiento de *triggers* y restricciones (*constraints*), así como corroborar que los tipos de datos y la estructura de las entidades funcionasen adecuadamente. Este método de trabajo permitió afinar detalles técnicos antes de su uso en un entorno de producción.

A continuación, se muestra un fragmento de código SQL que describe una de las entidades principales del sistema:

```
CREATE TABLE almacen (  
  id_almacen          VARCHAR(15) PRIMARY KEY,  
  nombre              VARCHAR(50) UNIQUE NOT NULL,  
  ubicacion           VARCHAR(100) NOT NULL,  
  existentes           INT NOT NULL CHECK (existentes >= 0),  
  vendidos            INT DEFAULT 0 CHECK (vendidos >= 0),  
  desechados          INT DEFAULT 0 CHECK (desechados >= 0),  
  totales              INT AS (existentes + vendidos + desechados)  
  STORED,  
  fecha_actualizacion DATE  
);
```

Fragmento de código 4.1: Ejemplo de definición de tabla Almacen.

Fragmento de código 1. Ejemplo de definición de tabla Almacen.

*Nota: Una muestra con mayor detalle del código, la cual incluye la definición de otras tablas, claves, índices y triggers, se presenta en el **Anexo 3** de este informe.*

4.1.5. Pruebas

La primera versión del modelo físico fue implementada en **MySQL**. En cuanto finalizó, se procedió con el flujo de pruebas y validación con el fin de verificar el funcionamiento del modelo y corregir cualquier fallo o mal funcionamiento. Este proceso resultó esencial para asegurar que la estructura implementada cumpliera con los requisitos operativos de la empresa y que las reglas de negocio se acataran íntegramente.

Las pruebas se llevaron a cabo mediante un proceso iterativo de ajustes continuos, en el que cada etapa fue seguida de modificaciones, eliminación o adición de las estructuras necesarias para cumplir con los objetivos. Las actividades principales realizadas en estas pruebas incluyeron:

Inserción de datos de prueba: Para verificar el funcionamiento adecuado de las relaciones entre entidades, se registraron datos de prueba basados en escenarios reales.

Ejecución de consultas SQL: Se evaluó la capacidad de respuesta de SQL ejecutando consultas complejas, donde se demostró que estas devuelven resultados precisos y efectivos.

Validación de triggers y automatizaciones: Se verificó que las rutinas de los *triggers* definidos, como aquellos que actualizan el *stock* de equipos al registrar una salida, se ejecutaran de manera precisa.

Revisión de integridad referencial: El sistema se sometió a pruebas con la intención de provocar escenarios que pudiesen comprometer la integridad de los datos, con el fin de asegurar que las restricciones (*constraints*) definidas cumplieran su objetivo y mantuvieran la coherencia de la información. Entre las tareas más comunes estuvo la corroboración de que los campos obligatorios no aceptasen valores nulos y que se mantuvieran sincronizados los registros relacionales.

Este enfoque, orientado a la filosofía de "falla rápido, aprende rápido", permitió identificar los errores en etapas tempranas y realizar los ajustes necesarios para corregirlos. El resultado ha sido una base de datos sólida, lista para integrarse a la interfaz web y ser utilizada en un entorno operativo real.

4.2. Metodología para el Desarrollo de la Interfaz Web

La interfaz web es la conexión entre la experiencia de usuario y el sistema de inventario; es por ello que su diseño requirió una planificación estructurada, modular y que garantizara la seguridad. Se optó por utilizar el *framework* **Django**, una herramienta de desarrollo *web* de alto nivel basada en **Python**, que permite integrar de forma eficiente el *backend* (lógica del servidor y acceso a datos) con el *frontend* (interfaz visual).

4.2.1. Creación del Proyecto y las Aplicaciones en Django

El primer paso en el desarrollo del sistema fue la creación del proyecto principal en **Django**, al cual se le asignó el nombre `admin_inventory`. Esta instancia constituye el núcleo desde el cual se organizan todas las aplicaciones internas que componen el sistema.

Con el fin de garantizar una estructura ordenada, escalable y fácil de mantener, se definieron y desarrollaron seis aplicaciones (*apps*) independientes dentro del proyecto, cada una encargada de una funcionalidad específica:

- Empleados.
- Inventario.
- Pedidos.
- Proyectos.
- Reparaciones.
- Users (gestión de usuarios y control de permisos).

Esta arquitectura modular responde al principio de separación de responsabilidades, en el que cada aplicación gestiona su propio conjunto de modelos, vistas, formularios, plantillas y rutas. De esta manera, las tablas relacionadas con proyectos se encuentran encapsuladas en la aplicación Proyectos, mientras que las funcionalidades asociadas a movimientos de inventario están organizadas dentro de la *app* Inventario.

El uso de este enfoque ofrece múltiples ventajas:

Mantenimiento simplificado: Al estar cada funcionalidad contenida en su propia *app*, las actualizaciones o correcciones pueden realizarse sin afectar otras partes del sistema.

Escalabilidad: Nuevas funcionalidades pueden añadirse mediante nuevas aplicaciones, sin alterar la estructura base del proyecto.

Reutilización de código: Los componentes desarrollados en una *app* pueden integrarse o adaptarse fácilmente en otros contextos o proyectos futuros.

Esta organización del proyecto, basada en las capacidades nativas de **Django**, sentó las bases para el desarrollo coherente de toda la plataforma y permitió mantener un alto grado de control sobre la lógica y presentación de cada módulo funcional.

4.2.2. Conexión entre Django y MySQL

Una vez definida la estructura base del sistema mediante las aplicaciones internas de **Django**, el siguiente paso fue establecer la conexión con la base de datos **MySQL**, que constituye el motor de almacenamiento principal del sistema de inventario.

Esta conexión se configuró en el archivo `settings.py` del proyecto **Django**, donde se especificaron los parámetros requeridos para la autenticación y comunicación con el servidor de base de datos. Estos parámetros incluyeron el nombre de la base de datos, el usuario, la contraseña, el *host* y el puerto, de acuerdo con las credenciales establecidas durante la implementación del modelo físico.

Django cuenta con un sistema de Mapeo Objeto-Relacional (**ORM**, por sus siglas en inglés) que permite traducir las estructuras relacionales de la base de datos (tablas, campos y relaciones) en clases y atributos dentro del código **Python**. Este enfoque facilitó el acceso y manipulación de datos directamente desde los modelos definidos en cada aplicación (`models.py`), eliminando la necesidad de escribir sentencias **SQL** manuales para operaciones comunes como inserciones, consultas, actualizaciones o eliminaciones.

Entre las principales ventajas de utilizar el **ORM** de **Django** destacan:

- Abstracción de la lógica de base de datos, lo que simplifica el desarrollo y reduce la probabilidad de errores.
- Mayor seguridad, al prevenir vulnerabilidades como inyecciones **SQL**.
- Portabilidad, ya que el código es independiente del motor de base de datos subyacente.
- Integración nativa con herramientas de migración, lo que facilita la gestión de cambios en el esquema de datos.

Gracias a esta integración entre **Django** y **MySQL**, fue posible garantizar una sincronización eficiente entre el *backend* de la aplicación *web* y la estructura lógica previamente diseñada en el modelo de base de datos, logrando una interacción fluida, segura y modular entre ambos componentes.

4.2.3. Frontend y Backend

En la sección del *frontend*, **Django** otorga un paquete de plantillas (*templates*) el cual proporciona la capacidad de crear vistas HTML personalizadas, mediante la integración de variables, estructuras de automatización, segmentos condicionales y bucles de control directamente conectados al sistema de acceso. Estas funciones permiten construir una interfaz web adaptada a las necesidades del sistema, con contenido dinámico y elementos adaptados al perfil de cada usuario.

La sección del *backend* es gestionada mediante el **ORM**, lo cual permite una comunicación fluida con la base de datos mediante el uso de modelos basados en las entidades SQL definidos en **Python**, evitando redactar consultas SQL directamente. A la par, **Django** ofrece la capacidad de crear una estructura clara para implementar vistas (`views.py`), las cuales procesan las solicitudes de entrada y ejecutan la lógica del sistema para así devolver la respuesta correspondiente al usuario final.

Estas herramientas que ofrece **Django** fueron utilizadas para crear seis aplicaciones que definen completamente la lógica del sistema de inventario. Cada una de estas aplicaciones cuenta con sus propias vistas y métodos de control, encargadas de gestionar las solicitudes específicas de su área:

- Empleados.
- Inventario.
- Pedidos.
- Proyectos.
- Reparaciones.
- Users.

Este método de segmentación favorece la claridad del código, la reutilización de componentes y la posibilidad de escalar el sistema conforme a las necesidades de la empresa. Además, esta división del sistema en aplicaciones permite distribuir el desarrollo en fases, realizar pruebas conjuntas manteniendo la compartimentación de cada módulo y aislar los errores para corregirlos con mayor rapidez.

4.2.4. Autenticación

Una vez que el sistema y sus vistas principales se encontraban operativos, se procedió a implementar el mecanismo de autenticación de usuarios con el objetivo de controlar el acceso a las funcionalidades de la plataforma de manera estructurada y segura. Esta capa es fundamental en sistemas que manejan información crítica, como los inventarios corporativos, ya que garantiza que únicamente los usuarios autorizados interactúen con los datos.

Para este fin, se utilizaron las herramientas que **Django** proporciona de manera nativa, aprovechando su subsistema de autenticación integrado para gestionar credenciales, sesiones y permisos de forma robusta y centralizada.

Se implementó un flujo de inicio de sesión (*login*) basado en nombre de usuario y contraseña, lo cual permitió asociar a cada cuenta un perfil determinado con permisos específicos. **Django** incluye de forma predeterminada funciones esenciales para la gestión de acceso, entre las que destacan:

- **Almacenamiento seguro de contraseñas:** Generación de *hashes* criptográficos mediante funciones de derivación de claves (**PBKDF2** con algoritmo **SHA-256** e iteraciones reforzadas), evitando el almacenamiento de claves en texto plano.
- **Gestión de sesiones autenticadas:** Control de sesiones mediante *cookies* firmadas criptográficamente para prevenir la suplantación o manipulación de sesiones (*session hijacking*).
- **Validación y saneamiento de entradas:** Validación automática de formularios de acceso para mitigar el envío de datos anómalos o peticiones malformadas.
- **Vistas genéricas seguras:** Controladores estandarizados para los flujos de inicio de sesión (*login*) y cierre de sesión (*logout*).

La implementación de este esquema permitió establecer un acceso confiable y protegido frente a vulnerabilidades comunes a nivel de sesión y almacenamiento de credenciales. Cabe destacar que la mitigación de ataques por fuerza bruta se delega al perímetro de red corporativo (acceso restringido mediante la conexión inicial a la Máquina Virtual) y a las políticas internas de complejidad de contraseñas, complementando la seguridad nativa provista por el *framework*.

La autenticación constituye la primera barrera de seguridad de la plataforma, resultando determinante para salvaguardar la confidencialidad, integridad y disponibilidad de la información gestionada.

4.2.5. Jerarquías y Permisos

Uno de los requerimientos funcionales planteados por la empresa fue la implementación de un sistema de jerarquías de usuario, que permitiera restringir el acceso a determinadas funcionalidades del sistema según el rol asignado. Esta característica es esencial en aplicaciones que manejan información sensible, ya que promueve el principio de mínimo privilegio, garantizando que cada usuario interactúe únicamente con los componentes necesarios para el cumplimiento de sus funciones.

Para cumplir con este objetivo, se definieron cuatro niveles de usuario, cada uno con permisos específicos:

- **Administrador:** Tiene control total sobre el sistema. Puede gestionar usuarios (crear, editar, eliminar) y también realizar cualquier operación sobre los registros del sistema (consultar, registrar, editar y eliminar).
- **Gestor:** Tiene acceso completo a la gestión de registros, pero no puede acceder a la sección de administración de usuarios.
- **Editor:** Puede visualizar y modificar registros existentes, pero no tiene permisos para crear nuevos ni eliminar registros.
- **Lector:** Tiene un acceso exclusivamente consultivo. Puede visualizar los registros, pero no puede realizar modificaciones de ningún tipo.

La asignación de estos roles se llevó a cabo utilizando el sistema de permisos y grupos que **Django** proporciona por defecto, lo cual permitió una administración centralizada y eficiente del acceso a vistas, formularios y operaciones específicas.

La siguiente tabla resume los niveles de acceso establecidos:

Tabla 4.1: Tabla de Jerarquías.

Rol	Acciones Permitidas	Usuarios	Registros
Administrador	Crear, editar y eliminar usuarios y registros	✓	✓
Gestor	Consultar, crear, editar y eliminar registros	×	✓
Editor	Consultar y editar registros	×	✓
Lector	Solo consultar registros	×	✓

4.2.6. Pruebas

Al igual que en el desarrollo de la base de datos, la implementación de la interfaz *web* requirió una fase de pruebas exhaustiva, con el objetivo de garantizar el correcto funcionamiento de cada uno de sus componentes antes de su despliegue definitivo.

Las principales pruebas realizadas fueron:

Pruebas de autenticación: Se verificó que los usuarios pudieran iniciar sesión correctamente, que el sistema manejara las credenciales de forma segura y que las contraseñas fueran encriptadas conforme a los estándares de **Django**.

Pruebas de vistas: Se evaluó que cada vista (*view*) respondiera adecuadamente ante diferentes solicitudes. Esto incluyó la recuperación de datos, la visualización de formularios y la ausencia de errores en el *backend*.

Pruebas de permisos: Se comprobó que el sistema aplicara correctamente las restricciones por rol. Los usuarios cuyos permisos estuviesen asociados solo a Lectura no tendrían la capacidad de modificar ningún registro ni acceder a funciones administrativas, asegurando la consistencia del modelo.

Pruebas de rendimiento: Se llevaron a cabo pruebas para verificar el rendimiento del sistema; para ello se realizaron inserciones de registros con datos de prueba para posteriormente hacer consultas sencillas, moderadas y complejas con el fin de observar el comportamiento del sistema.

Todas las pruebas se realizaron de forma iterativa, aplicando ajustes progresivos a medida que surgían inconsistencias o se presentaban necesidades de optimización. Esta estrategia ayudó a prevenir e identificar errores, permitiendo ajustes estructurales, modificaciones en las restricciones y el refinamiento de procesos automáticos en etapas tempranas, dando como resultado una base de datos sólida [20].

4.3. Metodología para el Acceso Remoto

Para facilitar la experiencia del usuario y resolver la necesidad de movilidad, con el sistema desarrollado y validado en un entorno local, se procedió a implementar el acceso remoto para manejar y consultar la aplicación desde ubicaciones externas. Esto permitió que los usuarios autorizados pudieran interactuar con la plataforma desde fuera de las instalaciones de la empresa.

4.3.1. Cargar el Proyecto en los Servidores de la Empresa

El primer paso para habilitar el uso en modalidad remota fue cargar el proyecto en los servidores de la empresa, lo cual ayudó a gestionarlo de una forma centralizada, ágil y segura. Se utilizó un servidor dedicado configurado para replicar las condiciones de ejecución mediante dos fases principales:

Preparación del servidor: Instalación del intérprete de **Python**, el entorno virtual, **Django** y el motor de base de datos **MySQL**. Se instalaron las dependencias mediante `pip` y se definieron las variables de entorno para producción.

Configuración del entorno de producción: Ajustes en el archivo `settings.py`, incluyendo la conexión a la base de datos, la desactivación del modo de depuración (`DEBUG=False`), la definición de dominios permitidos (`ALLOWED_HOSTS`) y la configuración de rutas estáticas.

4.3.2. Pruebas de Conexión

Se llevaron a cabo pruebas técnicas para verificar la correcta integración de los componentes del sistema y asegurar que la comunicación entre el *backend*, la base de datos y los clientes *web* operara de forma fluida. Las validaciones se estructuraron en dos aspectos esenciales:

Conexión al servidor de base de datos: Se validó el enlace bidireccional entre **Django** y el motor **MySQL** ejecutado en la Máquina Virtual mediante herramientas administrativas de consola como `python manage.py migrate` (comprobación del estado del esquema) y `python manage.py`

dbshell (acceso interactivo), confirmando la accesibilidad de las tablas, relaciones e integridad de las restricciones.

Acceso a la aplicación web: Se verificó la disponibilidad de la interfaz desde distintas estaciones de trabajo conectadas a la red local (*LAN*) a través del nombre de dominio asignado (*invconecta.local.net*). Se comprobó el correcto enrutamiento de las solicitudes **HTTP**, la renderización de las plantillas y el suministro íntegro de los archivos estáticos (*CSS*, *JavaScript* e imágenes) sin errores de resolución o carga.

4.3.3. Pruebas de Funcionamiento

Tras validar la conexión, se realizaron pruebas de funcionamiento bajo condiciones reales de uso para detectar posibles fallos de rendimiento o brechas de seguridad. Las pruebas se agruparon en:

Validación de funcionalidades: Comprobación de operaciones críticas como el registro de datos, la edición de información y la generación de reportes en **PDF**, validando que el modelo de jerarquías restringiera el acceso según el rol.

Verificación de seguridad: Se llevó a cabo una evaluación práctica de los mecanismos de protección implementados en la plataforma:

- **Mitigación de inyecciones SQL:** Se verificó el comportamiento del ORM de Django frente a entradas maliciosas en parámetros de búsqueda y filtrado de equipos (e.g., cadenas con comillas simples y sentencias ' OR '1'='1). Las consultas parametrizadas del ORM neutralizaron las cargas útiles al tratarlas estrictamente como literales, garantizando la integridad de la base de datos MySQL.
- **Protección contra CSRF (*Cross-Site Request Forgery*):** Se validó que todos los formularios de captura (altas, modificaciones y bajas) exigieran el token criptográfico `{% csrf_token %}`. Las peticiones HTTP POST manipuladas manualmente o enviadas sin el token fueron rechazadas de forma inmediata por el servidor con una respuesta de error HTTP 403 Forbidden.
- **Control de acceso a rutas restringidas:** Se corroboró que el ingreso directo por URL a vistas administrativas o de edición estuviera bloqueado para usuarios no autenticados o con roles sin privilegios suficientes, redirigiendo automáticamente a la pantalla de inicio de sesión.

4.3.4. Despliegue e Integración en la Infraestructura Corporativa

Dadas las limitaciones de almacenamiento en el servidor físico principal de la empresa, el área de TI determinó aprovisionar una **Máquina Virtual (VM)** dedicada exclusivamente a la ejecución y administración del sistema de inventario. Esta estrategia de virtualización se implementó bajo criterios específicos para garantizar el rendimiento, la seguridad y la integridad de la plataforma.

Configuración del Entorno y Acceso Local El sistema de inventario fue estructurado desde su fase de diseño para operar dentro de la infraestructura interna de la empresa. Con el propósito de facilitar el acceso a los colaboradores conectados a la red de área local (*LAN*), se configuró un registro en el **DNS Local**, permitiendo el acceso mediante una dirección legible: `invconecta.local.net`. Bajo este esquema, los usuarios internos conectados a la *LAN* ingresan directamente a la interfaz web mediante sus credenciales de aplicativo (Administrador, Gestor, Editor o Lector), sin requerir autenticación previa a nivel de servidor.

Acceso Remoto y Seguridad Perimetral Con el objetivo de garantizar la disponibilidad del sistema desde puntos externos sin exponer la base de datos ni la aplicación directamente a la red pública de internet, se implementó un modelo de acceso seguro estructurado en dos niveles:

1. **Nivel de Infraestructura:** Cualquier usuario externo debe, en primera instancia, establecer una conexión remota segura hacia la **Máquina Virtual** utilizando las credenciales de red corporativas proporcionadas por el área de TI. Este primer paso actúa como una barrera perimetral que valida la identidad del usuario antes de otorgarle acceso a la red interna de la organización.
2. **Nivel de Aplicación:** Una vez dentro del entorno virtual, el usuario hace uso del navegador web para ingresar a la URL local `invconecta.local.net`. Tras desplegarse la interfaz del sistema, deberá autenticarse en el módulo de inicio de sesión (*login*) con su usuario y contraseña para acceder a las funciones permitidas por su rol asignado.

Esta arquitectura de despliegue asegura que el sistema opere en un entorno controlado, optimizando los recursos de virtualización disponibles y manteniendo una política de seguridad basada en el aislamiento de la infraestructura corporativa.

La Figura 4.1 ilustra el flujo de comunicación y los niveles de autenticación implementados para el acceso tanto local como remoto al sistema de inventario.

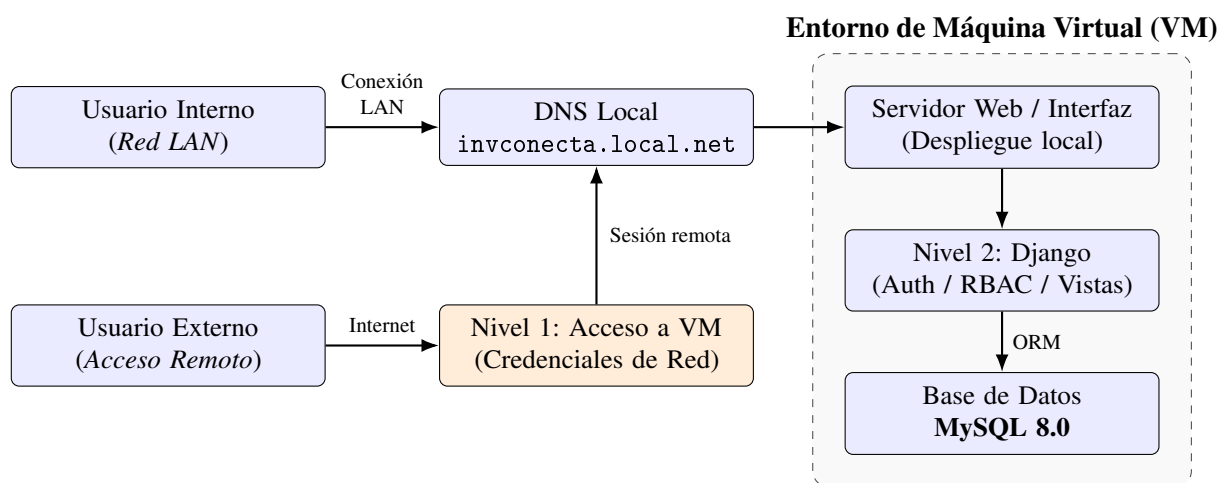


Figura 4.1: Diagrama de flujo de acceso y arquitectura de despliegue en dos niveles (Fuente: Elaboración propia).

4.4. Resultados

Con base en la metodología descrita en los apartados anteriores, se logró desarrollar e implementar un sistema de inventario funcional, seguro y accesible de manera remota para la empresa. El sistema integra una base de datos construida en **MySQL**, una interfaz *web* desarrollada con el *framework Django* y un entorno de producción gestionado a través de servidores locales configurados por la propia organización.

Las herramientas seleccionadas para el proyecto demostraron un alto grado de compatibilidad entre sí, lo que facilitó la integración entre el *backend*, la base de datos y el entorno de despliegue. Además, al tratarse de tecnologías ampliamente documentadas y con comunidades activas, fue posible acceder a numerosos recursos de soporte técnico y soluciones a problemas comunes en todas las etapas del desarrollo.

4.4.1. Validación y Cumplimiento de Objetivos

Para constatar de forma cuantitativa y cualitativa que el sistema de inventario satisface las necesidades operativas de la empresa, se diseñó una matriz de trazabilidad que vincula cada objetivo específico planteado al inicio del proyecto con su solución técnica y su evidencia documental de cumplimiento (Tabla 4.2).

Tabla 4.2: Matriz de trazabilidad y cumplimiento de objetivos.

Objetivo Específico	Solución Implementada	Evidencia / Métrica
Desarrollar una base de datos relacional con MySQL .	Modelo físico normalizado en 3FN con llaves primarias, foráneas, <i>triggers</i> y columnas calculadas.	Esquema DDL (<code>inventario_v7.sql</code>), diagrama relacional y 0% de inconsistencias en pruebas de integridad.
Crear una interfaz web funcional mediante Django .	Arquitectura modular MVT compuesta por seis aplicaciones independientes (<i>apps</i>) y vistas CBV.	Módulos operativos de altas, bajas, consultas y cambios accesibles vía navegador web.
Construir un esquema de autenticación y permisos por roles.	Implementación de control de acceso basado en roles (RBAC) con cuatro jerarquías mediante <i>mixins</i> de Django.	Tabla de jerarquías (4.1), validación de vistas protegidas y bloqueo de acciones no autorizadas.
Configurar el despliegue en la infraestructura corporativa.	Aprovisionamiento de una Máquina Virtual dedicada con DNS local (<code>invconecta.local.net</code>) y acceso seguro en dos niveles.	Diagrama de arquitectura (4.1) y conectividad remota exitosa validada por TI.
Automatizar la generación de reportes en PDF.	Módulo de exportación documental integrado en Django con formateo tabular de activos y resguardos.	Generación y descarga inmediata de reportes de pedidos, almacenes y asignaciones de equipo.

4.4.2. Pruebas Funcionales y de Rendimiento

Se sometió a la plataforma a un protocolo de pruebas funcionales y de carga controlada sobre la base de datos y la interfaz web. La Tabla 4.3 resume los casos de prueba ejecutados, el volumen de datos evaluado y los tiempos de respuesta obtenidos dentro de la red corporativa.

Tabla 4.3: Resultados del protocolo de pruebas funcionales y de rendimiento.

Caso de Prueba	Operación Evaluada	Volumen	Tiempo Prom.	Estado
Autenticación e inicio de sesión	Validación de credenciales con <i>hash</i> PBKDF2 y carga de sesión	1 usuario	< 0,35 s	Aprobado
Consulta de inventario general	Consulta de equipos con filtros por almacén y catálogo (JOIN múltiple)	1,500 registros	< 0,80 s	Aprobado
Registro masivo de entrada	Inserción en <i>ingreso_equipo</i> y actualización del stock en <i>almacen</i>	100 equipos	< 1,20 s	Aprobado
Ejecución de <i>triggers</i>	Verificación de actualización en <i>fecha_actualizacion</i> y bloqueo de duplicidad	50 eventos	< 0,05 s	Aprobado
Generación de reporte PDF	Extracción de registros, maquetación y renderizado en PDF	250 registros	< 1,50 s	Aprobado
Control de permisos RBAC	Intento de edición y borrado por usuario con rol Lector	10 intentos	Inmediato (403)	Aprobado

4.4.3. Beneficios Obtenidos del Sistema de Inventario

La implementación del sistema generó mejoras significativas en la operación diaria de la empresa, las cuales se detallan a continuación:

Centralización de la información: Todos los datos relacionados con equipos, proyectos, proveedores, almacenes y movimientos se almacenan ahora en una única base de datos centralizada. Esta estructura facilita el acceso compartido, reduce la redundancia de registros y mejora la eficiencia en la consulta por parte de los usuarios autorizados.

Consistencia en los datos: A través de la correcta implementación de las reglas de negocio que definieron las restricciones de integridad referencial (*constraints*), se logró garantizar que los datos ingresados en el sistema estén siempre completos y coherentes. Ejemplo de ello es la restricción que asegura que ningún equipo sea capaz de registrar una salida antes de su fecha de ingreso.

Homogeneidad en los registros: Al implementar un catálogo centralizado de equipos, es posible mantener una estructura uniforme en todos los datos registrados, asegurando que se cumplan las normativas internas de la empresa, que tiene entre sus objetivos contar con una clasificación clara de sus activos.

Robustez del sistema: La arquitectura modular da la capacidad de gestionar una gran cantidad de registros simultáneamente sin perjudicar el rendimiento global. La dupla Django-MySQL ofrece cimientos sólidos que otorgan la capacidad de soportar una futura expansión del sistema.

Escalabilidad: La estructura del sistema fue pensada y diseñada para ser capaz de crecer según las necesidades de la empresa, adaptándose al creciente número de usuarios, equipos y proyectos sin necesidad de realizar grandes modificaciones estructurales.

Consulta remota: La integración del acceso remoto ofrece a los usuarios la capacidad de consulta y gestión del sistema de inventario desde cualquier ubicación externa con conexión a internet. Mediante

credenciales de acceso se asegura mantener altos estándares de seguridad y control, lo cual es clave para las operaciones de la empresa.

Seguridad y control de accesos: El sistema de inventario integra un módulo de autenticación para otorgar el ingreso y un método de control de acceso basado en roles (RBAC) para la gestión de la plataforma, lo cual limita las funciones que puede realizar el usuario dependiendo de su rol asignado. A la par, se emplean protocolos de acceso seguro en las conexiones externas con el fin de aumentar la seguridad del sistema.

Los beneficios mencionados representan las mejoras más significativas en el manejo de las operaciones internas de la empresa. El sistema facilitó la gestión del inventario proporcionando información oportuna sobre los equipos disponibles y los proyectos en curso. La plataforma muestra una capacidad que va más allá de resolver las necesidades presentes, consolidándose como una herramienta adaptable y sólida capaz de evolucionar de acuerdo con las demandas futuras de la organización.

4.4.4. Evidencias de Operación del Sistema

Con el propósito de validar la integración visual, operativa y funcional descrita en los apartados anteriores, a continuación se presentan las interfaces clave del sistema de inventario desplegado en la infraestructura corporativa.

Acceso y Autenticación La Figura 4.2 muestra la pantalla de bienvenida y acceso a la plataforma a través del navegador web, validando la resolución mediante el dominio del DNS local (`invconecta.local.net`). Por su parte, la Figura 4.3 presenta el módulo de inicio de sesión protegido mediante los mecanismos de autenticación y validación de credenciales.

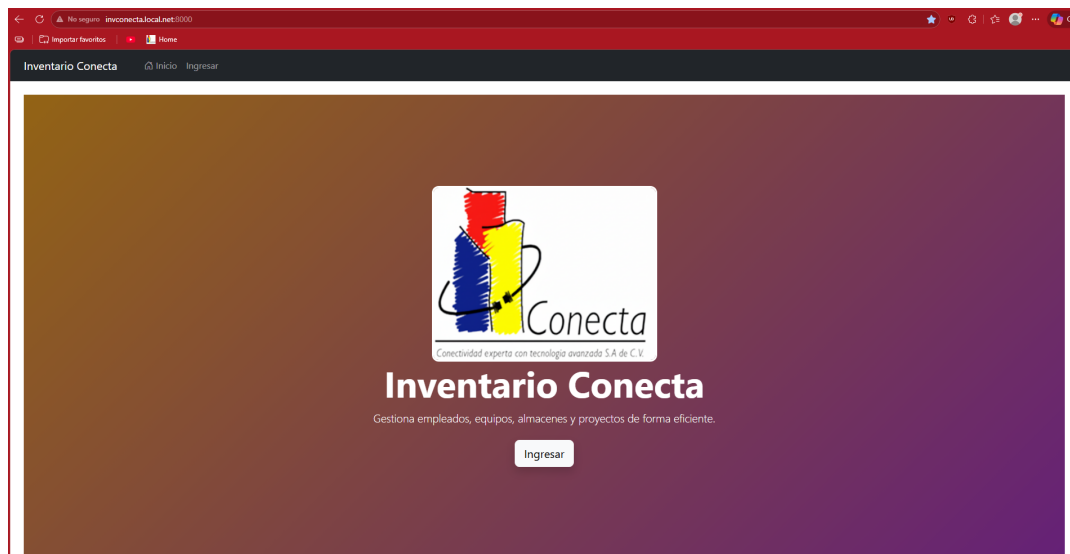


Figura 4.2: Pantalla de bienvenida y resolución mediante DNS local corporativo (Fuente: Elaboración propia).

Inventario Conecta

Iniciar Sesión

Usuario:

Contraseña:

Ingresar

Figura 4.3: Formulario de inicio de sesión del sistema de inventario (Fuente: Elaboración propia).

Consulta y Formularios de Captura La Figura 4.4 ilustra la vista de consulta general de inventario con filtros dinámicos por almacén, catálogo y tipo de activo. La Figura 4.5 muestra el formulario para el alta de colaboradores, estructurado para validar campos obligatorios como el RFC y correos de contacto.

Buscar

Catálogo

Todos

Activo Fijo

Todos

Aplicar Filtros

Limpiar

Lista de Equipos

+ Agregar Equipo

Volver al Dashboard

ID Equipo	Número Serie	Activo Fijo	MAC Address	Almacén	Acciones
eqpo-1	SN-LTP-001	Si	00:1A2B:3C4D:5E	Almacén Central	Ver Detalles
eqpo-2	SN-IMP-001	No	00:1A2B:3C4D:5F	Almacén Norte	Ver Detalles
eqpo-3	SN-PRJ-001	No	N/A	Almacén Sur	Ver Detalles
eqpo-4	SN004	No	N/A	Almacén Central	Ver Detalles

Figura 4.4: Módulo de consulta general y listado de equipos en inventario (Fuente: Elaboración propia).

Nuevo Empleado

RFC*

Nombre completo

Apellido

Nombre*

Apellido paterno*

Apellido materno*

Calle

No. ext.

No. int.

Colonia

Cp.

Municipio

Estado

Email

Teléfono personal

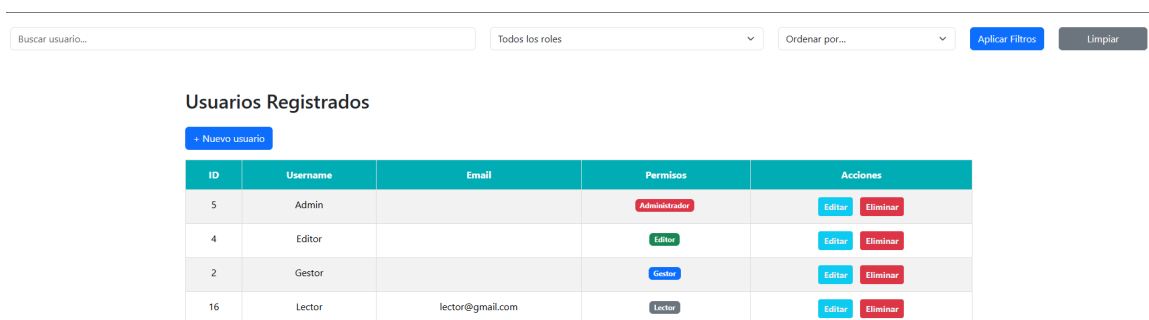
Teléfono ext.

Guardar

Cancelar

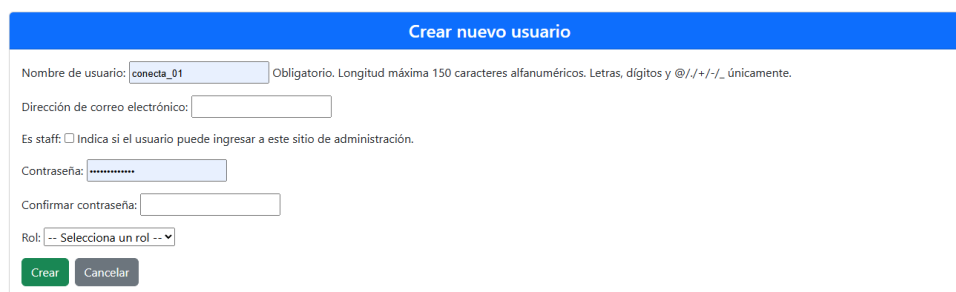
Figura 4.5: Formulario de captura y registro de empleado (Fuente: Elaboración propia).

Administración de Usuarios y Permisos (RBAC) La Figura 4.6 exhibe el panel de control de usuarios registrados junto con su jerarquía asignada (Administrador, Gestor, Editor y Lector). La Figura 4.7 presenta la interfaz para dar de alta nuevas cuentas asignándoles su respectivo rol de operación.



ID	Username	Email	Permisos	Acciones
5	Admin		Administrador	Editar Eliminar
4	Editor		Editor	Editar Eliminar
2	Gestor		Gestor	Editar Eliminar
16	Lector	lector@gmail.com	Lector	Editar Eliminar

Figura 4.6: Gestión de usuarios y asignación de jerarquías bajo el esquema RBAC (Fuente: Elaboración propia).



Crear nuevo usuario

Nombre de usuario: Obligatorio. Longitud máxima 150 caracteres alfanuméricos. Letras, dígitos y @/./+/_ únicamente.

Dirección de correo electrónico:

Es staff: ☐ Indica si el usuario puede ingresar a este sitio de administración.

Contraseña:

Confirmar contraseña:

Rol: -- Selecciona un rol --

Figura 4.7: Formulario de creación de usuario y asignación de rol (Fuente: Elaboración propia).

Vistas de Detalle y Generación de Reportes PDF El sistema implementa vistas de detalle (*Detail Views*) que centralizan la información consolidada de cada registro y disponen del accionador para generar el reporte documental correspondiente.

La Figura 4.8 muestra la vista detallada de un proyecto registrado; al accionar la opción de exportación, el *backend* procesa las partidas y renderiza dinámicamente el acta formal de entrega de insumos en PDF (Figura 4.9).

De igual manera, la Figura 4.10 exhibe la consulta detallada de una asignación de activo tecnológico, a partir de la cual se emite el formato oficial de resguardo y responsiva individual mostrado en la Figura 4.11.

Información Completa del Proyecto	
ID Proyecto:	proy-1
Nombre:	Proyecto Alfa
RFC:	PRJA900101AAA
Descripción:	Instalación de red
Dirección:	Av. Reforma, CDMX, CDMX
Contacto:	celular - 5588880001
Fecha de Inicio:	10 de Enero de 2025
Fecha de Conclusión:	10 de Junio de 2025
Duración (meses):	5

[Editar](#)
[Eliminar](#)
[Generar PDF](#)
[-- Regresar](#)

Figura 4.8: Vista de detalle y consulta consolidada de proyecto (Fuente: Elaboración propia).

Ciudad de México a 27 de February de 2026

Dirección de Administración y Finanzas
 Subdirección de Recursos Materiales y Servicios Generales
 Gerencia y Recursos Materiales
 Proyecto Alfa

Por medio de la presente y dando cumplimiento al proyecto "Instalación de red", realizado entre mi representada CONECTIVIDAD EXPERTA CON TECNOLOGÍA AVANZADA S.A. DE C.V. (CONECTA) y la contraparte en Av. Reforma, CDMX, CDMX, hago entrega de los siguientes insumos descritos:

PARTIDA	DESCRIPCIÓN	CANTIDAD	UNIDAD DE MEDIDA	NO. SERIE
1	Laptop Pro	2	Piezas	SN-LTP-001 SN004
2	Impresora Láser	1	Piezas	SN-IMP-001
2	Lapiz Tactil	1	Piezas	SN009
2	Proyector HD	1	Piezas	SN-PRJ-001
-	Laptop Pro	2	Piezas	SN005 SN006

Sin más por el momento, reciba un cordial saludo.

Atentamente,

 Firma del Responsable de Entrega

Figura 4.9: Reporte de entrega y desglose de equipos asociados a proyecto en formato PDF (Fuente: Elaboración propia).

Inventario Conecta
Bienvenido: conecta_01
Inicio
Empleados
Inventario
Pedidos
Proyectos
Reparaciones
III Panel de Control

Información Completa de Asignación

ID Asignación:	aseq-1
Empleado:	None
Equipos:	SN-LTP-001 (eqpo-1)
Fecha de Asignación:	15 de Enero de 2025
Fecha de Devolución:	No devuelto
Observaciones:	Laptop para Juan

Editar
Eliminar
Generar PDF
-- Regresar

Figura 4.10: Vista de detalle de asignación de equipo a colaborador (Fuente: Elaboración propia).

Resguardo de Equipo: Laptop Pro

Datos del Usuario

Nombre:

Correo:

Departamento:

Domicilio:
, No. , Col. , , , C.P.

Datos del Equipo

Etiqueta:
eqpo-1

No. Serie:
SN-LTP-001

Marca:
Dell

Modelo:
XPS13

Tipo de Movimiento:
ALTA

Fecha de Entrega:
15 de Enero de 2025

Observaciones:
Laptop para Juan

Condiciones de Resguardo

Con el presente resguardo, recibo el equipo que provee la empresa CONECTIVIDAD EXPERTA CON TECNOLOGÍA AVANZADA S.A DE C.V., aceptando la responsabilidad de su conservación y entrega en óptimas condiciones al concluir mi relación con la empresa. Asimismo, asumo la responsabilidad por cualquier daño, pérdida o irregularidad, comprometiéndome a notificar de inmediato a mi jefe directo y al departamento de soporte.

Nombre y Firma de quien recibe:

Figura 4.11: Documento de resguardo y responsiva de equipo de cómputo en formato PDF (Fuente: Elaboración propia).

Trazabilidad y Auditoría La Figura 4.12 exhibe el módulo de registro de eventos (*logs*), el cual almacena las acciones de actualización, modificación y eliminación ejecutadas por los usuarios, asegurando la auditoría de operaciones críticas.

Historial de Cambios

Fecha:

dd/mm/aaaa

Tipo de Acción:

-- Todos --

Filtrar

Limpiar Filtros

Eliminar Todos

Usuario Afectado	Acción	Realizador	Fecha	Detalles	Acciones
conecta_01	Actualizado	Sistema	27/02/2026 16:30:43	Usuario actualizado: conecta_01	Eliminar
conecta_01	Actualizado	Sistema	27/02/2026 15:57:54	Usuario actualizado: conecta_01	Eliminar
conecta_01	Actualizado	Sistema	27/02/2026 15:27:22	Usuario actualizado: conecta_01	Eliminar
conecta_01	Actualizado	Sistema	27/02/2026 14:56:52	Usuario actualizado: conecta_01	Eliminar
conecta_01	Actualizado	Sistema	27/02/2026 14:03:37	Usuario actualizado: conecta_01	Eliminar
conecta_01	Actualizado	Sistema	27/02/2026 14:01:35	Usuario actualizado: conecta_01	Eliminar

Figura 4.12: Módulo de auditoría e historial de modificaciones en el sistema (Fuente: Elaboración propia).

Conclusiones

El desarrollo e implementación del sistema de inventario documentado en el presente informe permitió alcanzar satisfactoriamente los objetivos planteados, entregando a la empresa una solución tecnológica integral, robusta y adaptada a su dinámica operativa. La plataforma centralizó el control de los activos tecnológicos, garantizó la integridad y trazabilidad de los movimientos de almacén y optimizó la emisión de responsivas y actas de entrega mediante la automatización documental.

A lo largo del proyecto se validó en un entorno de producción real la aplicación práctica de los fundamentos metodológicos y teóricos de la Ingeniería en Computación. El diseño de un esquema relacional normalizado en Tercera Forma Normal (3FN), la abstracción y desacoplamiento provistos por el patrón Arquitectónico Modelo–Vista–Template (MVT) de **Django**, y la configuración de una infraestructura de red segura y virtualizada, demostraron la viabilidad técnica de construir sistemas empresariales escalables mediante tecnologías de código abierto.

5.1. Logros Técnicos y Aportaciones de Ingeniería

La ejecución del proyecto consolidó aportaciones clave en las distintas capas del sistema:

- **Modelado y consistencia de datos:** Se diseñó una base de datos relacional en **MySQL 8.0** estructurada bajo reglas de negocio formales, restricciones de integridad referencial (*foreign keys*, **CHECK**) y automatizaciones a nivel de motor mediante *triggers* y columnas calculadas almacenadas (**STORED**), logrando un 0 % de inconsistencias en las pruebas de carga.
- **Arquitectura modular y mantenibilidad:** La lógica de la aplicación se desacopló en seis módulos funcionales independientes dentro de Django, facilitando el aislamiento de fallos, la reutilización de componentes y el mantenimiento evolutivo del código.
- **Seguridad y control de acceso granular:** Se implementó un esquema de autenticación robusto basado en funciones de derivación criptográfica (**PBKDF2** con **SHA-256**) y un modelo de control de acceso basado en roles (**RBAC**) que restringe las operaciones críticas conforme a cuatro perfiles jerárquicos (Administrador, Gestor, Editor y Lector).

- **Eficiencia y rendimiento operativo:** Las pruebas funcionales demostraron tiempos de respuesta inferiores a un segundo en consultas complejas sobre volúmenes de 1,500 registros, garantizando una interacción ágil tanto en la red local (*LAN*) mediante el DNS `invconecta.local.net` como a través de sesiones remotas seguras.
- **Automatización de reportes:** Se eliminó la dependencia de formatos impresos manuales mediante la generación automatizada de archivos **PDF** para actas de proyectos y resguardos individuales de equipo.

5.2. Limitaciones Identificadas

Durante las fases de desarrollo, pruebas y puesta en marcha se identificaron limitantes operativas y de arquitectura que acotan el alcance actual de la solución:

Infraestructura monolítica virtualizada: El despliegue en una única Máquina Virtual dedicada concentra los servicios web y el motor de base de datos en el mismo nodo. Aunque satisface la demanda actual, limita la escalabilidad horizontal en caso de un incremento exponencial de transacciones concurrentes.

Procesamiento síncrono de reportes: La renderización de documentos PDF se ejecuta de forma síncrona en el hilo principal de la petición HTTP. Para volúmenes moderados el tiempo es óptimo ($< 1,50$ s), pero ante solicitudes masivas simultáneas podría representar un cuello de botella en los recursos del servidor.

Dependencia del perímetro de red corporativo: El acceso remoto requiere la autenticación previa a nivel de sistema operativo en la Máquina Virtual antes de interactuar con el aplicativo web, lo cual añade un paso intermedio para los colaboradores externos al no disponer de una pasarela VPN dedicada o un esquema de autenticación federada (SSO).

5.3. Aplicaciones y Trabajo a Futuro

Por su naturaleza modular y su diseño basado en estándares abiertos, el sistema cuenta con un amplio potencial de transferencia y evolución tecnológica.

5.3.1. Ámbitos de Aplicación

Con ajustes mínimos en las entidades del catálogo y en los formularios de captura, la plataforma puede implementarse en diversos entornos:

- **Instituciones educativas y laboratorios:** Para la administración y préstamo controlado de instrumental y equipos de cómputo.

- **Consultorías y empresas de servicios:** En la asignación temporal de herramientas de trabajo asociadas a contratos y proyectos específicos.
- **Dependencias públicas y organismos descentralizados:** Para el control patrimonial de bienes y auditoría de resguardos institucionales.

5.3.2. Líneas de Trabajo Futuro

Para mitigar las limitaciones identificadas y potenciar las capacidades de la plataforma, se plantean las siguientes líneas de desarrollo:

- **Desacoplamiento mediante API RESTful:** Desarrollar una capa de servicios web (**Django REST Framework**) con documentación OpenAPI/Swagger para permitir la integración bidireccional con otros sistemas empresariales o plataformas de facturación y compras (**ERP**).
- **Procesamiento asíncrono de tareas:** Integrar colas de trabajo en segundo plano mediante **Celery** y **Redis** para procesar la exportación masiva de reportes y el envío automático de notificaciones por correo electrónico sin bloquear la interfaz de usuario.
- **Monitoreo e Inteligencia de Negocios (BI):** Implementar paneles gráficos interactivos para la visualización de indicadores clave de rendimiento (**KPIs**), permitiendo proyectar ciclos de vida de activos y planificar renovaciones de equipo.
- **Migración a contenedores y nube:** Empaquetar la solución utilizando contenedores (**Docker**) y orquestación para facilitar el despliegue continuo y la redundancia en infraestructuras de nube híbrida.

El desarrollo de este sistema representó una experiencia profesional y académica integral. Más allá de resolver la problemática concreta de gestión de inventarios en la organización, el proyecto consolidó el ejercicio del criterio ingenieril, la capacidad de análisis y la toma de decisiones fundamentadas como bases indispensables de la práctica profesional en la Ingeniería en Computación.

Glosario

- **Administrador:** Usuario con permisos completos sobre el sistema. Tiene acceso a todas las funcionalidades, incluida la gestión de usuarios, equipos, proyectos y configuraciones clave.
- **App (Aplicación):** En el contexto de **Django**, una *app* es un componente modular del proyecto que agrupa funcionalidades específicas, como la gestión de empleados, inventarios, pedidos, entre otros.
- **Base de Datos SQL:** Sistema de almacenamiento estructurado que utiliza el lenguaje **SQL** (*Structured Query Language*) para gestionar información. En este proyecto se empleó **MySQL** como gestor de base de datos.
- **Catálogo de Equipos:** Conjunto de datos estáticos que describen los tipos de equipos disponibles en la empresa (marca, modelo, especificaciones, tipo), utilizado como referencia en el registro de inventario.
- **Consulta SQL:** Instrucción escrita en lenguaje **SQL** que permite recuperar, insertar, modificar o eliminar datos en la base de datos. Es esencial para interactuar con la información almacenada.
- **Django:** *Framework* de desarrollo *web* en **Python** que permite construir aplicaciones robustas y seguras, integrando el *backend* y el *frontend* mediante herramientas preconfiguradas.
- **Entidad-Relación (ER):** Modelo visual que representa gráficamente las entidades (tablas) de una base de datos y las relaciones entre ellas. Se utiliza para el diseño conceptual del sistema.
- **Interfaz Web:** Conjunto de elementos visuales e interactivos que permiten a los usuarios comunicarse con el sistema a través de un navegador, facilitando la navegación y operación.
- **Modelo Django:** Clase definida en el código de **Django** que representa una tabla de la base de datos. Cada instancia del modelo corresponde a un registro (fila), y cada atributo a una columna.
- **MySQL Workbench:** Entorno gráfico de desarrollo utilizado para diseñar, administrar y consultar bases de datos **MySQL**. Incluye herramientas para modelado **ER**, ejecución de consultas y administración de usuarios.

- **Permisos de Usuario:** Reglas que definen las operaciones que un usuario puede ejecutar dentro del sistema (consultar, editar, eliminar, crear). Se asignan a través de roles.
- **Roles de Usuario:** Niveles jerárquicos de acceso definidos en el sistema, como Administrador, Gestor, Editor y Lector, cada uno con permisos específicos.
- **Servidor Local (localhost):** Entorno de pruebas configurado en una máquina local o red interna, que permite ejecutar el sistema antes de su despliegue en producción.
- **uWSGI:** Servidor de aplicaciones que actúa como intermediario entre el servidor *web* (por ejemplo, **Nginx**) y la aplicación *web* desarrollada en **Django**. Optimiza el rendimiento y la entrega de respuestas.
- **Visualización de Datos:** Representación gráfica de la información almacenada (reportes, estadísticas, gráficos), utilizada para facilitar el análisis y la toma de decisiones.
- **Vista (en Django):** Función o clase que responde a una solicitud *web*, procesando la lógica de negocio y devolviendo una respuesta (**HTML**, **JSON**, **PDF**, etc.).
- **XML:** Lenguaje de marcado utilizado para almacenar datos estructurados de forma jerárquica. Aunque no se emplea directamente en este proyecto, puede ser útil para exportación o interoperabilidad con otros sistemas.

Referencias

- [1] Oracle. *¿Qué es Recursos Humanos?* Accedido: 2026-02-23. URL: <https://www.oracle.com/mx/human-capital-management/hr/what-is-hr/#link3>
- [2] P. P. Meana Coalla, *Gestión de inventarios*. Ediciones Paraninfo, S.A., 2017.
- [3] A. C. Fernández, *Gestión de inventarios. COML0210*. IC Editorial, 2018.
- [4] J. L. Montes, *UF0476 - Gestión de inventarios*. Editorial Elearning, S.L., 2014.
- [5] C. J. Date, *Introducción a los sistemas de bases de datos*. Pearson Educación, 2001.
- [6] H. G. del Busto y O. Y. Enríquez, Bases de datos NoSQL, *Telemática*, vol. 11, n.º 3, págs. 21-33, 2012.
- [7] DbExperts. *Tipos de bases de datos*. Accedido: 2026-02-23. URL: <https://dbaexperts.tech/bases-de-datos/tipos-de-bases-de-datos-nosql/>
- [8] C. M. Escofet, *El lenguaje SQL*. UOC, La Universidad Virtual, 2002.
- [9] Nicoraudy's blog. *El lenguaje SQL*. Accedido: 2026-02-23. URL: <https://nicoraudy.wordpress.com/2016/03/29/el-lenguaje-sql/>
- [10] M. C. Albornoz, Diseño de interfaz gráfica de usuario, en *XVI Workshop de Investigadores en Ciencias de la Computación*, oct. de 2014.
- [11] K. Ramírez-Acosta, Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo, *Revista Tecnología en Marcha*, vol. 30, págs. 49-54, 2017.
- [12] Sortlist. *El diseño de la interfaz de usuario: qué es y guía para hacerlo*. Accedido: 2026-02-23. URL: <https://www.sortlist.es/blog/disenio-de-la-interfaz-de-usuario/>
- [13] C. Coronel y S. Morris, *Database Systems: Design, Implementation, & Management*, 12.^a ed. Cengage Learning, 2017.
- [14] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.
- [15] Sucerman. *Diferencias entre User Interface y User Experience*. Accedido: 2025-05-29. URL: <https://contenidos.sucerman.com/nivel2/web1/unidad4/leccion2.html>

- [16] Hostinger. *¿Cómo funciona el hosting o alojamiento web?* Accedido: 2025-05-29. URL: <https://www.hostinger.com/mx/tutoriales/que-es-un-hosting>
- [17] L. Welling y L. Thomson, *PHP and MySQL Web Development*, 5.^a ed. Addison-Wesley, 2018.
- [18] D. Parsons, *The Art of Software Testing*. John Wiley & Sons, 2020.
- [19] C. Morimoto, *Servidores Linux: Guía del administrador de sistemas*, 4.^a ed. Dattatec, 2008.
- [20] D. Spinellis, *Code Quality: The Open Source Perspective*. Addison-Wesley, 2006.

Anexo 1. Diagrama Entidad-Relación (ER) del sistema de inventario

El siguiente diagrama representa el modelo conceptual del sistema de inventario, diseñado a partir de las reglas de negocio y los requerimientos funcionales identificados durante el análisis de procesos de la empresa. Este esquema constituye la piedra angular del diseño de datos, permitiendo abstraer la complejidad operativa en una estructura lógica coherente.

Este modelo, elaborado mediante el *software* de código abierto **DIA**, permitió visualizar de forma estructurada las entidades principales del sistema, así como sus atributos, relaciones, cardinalidades y restricciones de integridad. La elaboración de este diagrama fue crucial para alcanzar los siguientes objetivos de ingeniería:

- **Identificación de Entidades Clave:** Definición precisa de los objetos del mundo real que el sistema debe rastrear, tales como *Equipos*, *Usuarios*, *Asignaciones* y *Mantenimientos*.
- **Definición de Relaciones y Cardinalidades:** Establecimiento de los vínculos lógicos entre entidades (por ejemplo, la relación uno-a-muchos entre un proyecto y los equipos asignados), garantizando que el flujo de información refleje fielmente la operación de la firma.
- **Normalización Conceptual:** Aplicación de reglas de diseño para evitar la redundancia de información y asegurar que cada atributo pertenezca a la entidad correcta, sentando las bases para una base de datos eficiente.
- **Validación de Requerimientos:** El diagrama sirvió como una herramienta de comunicación técnica para validar con los interesados que todos los procesos de gestión de contratos y activos fueron correctamente interpretados.

El Diagrama Entidad-Relación presentado a continuación sirvió como referencia fundamental para el desarrollo del modelo relacional (Anexo 2) y la posterior implementación física en el motor de base de datos **MySQL**.

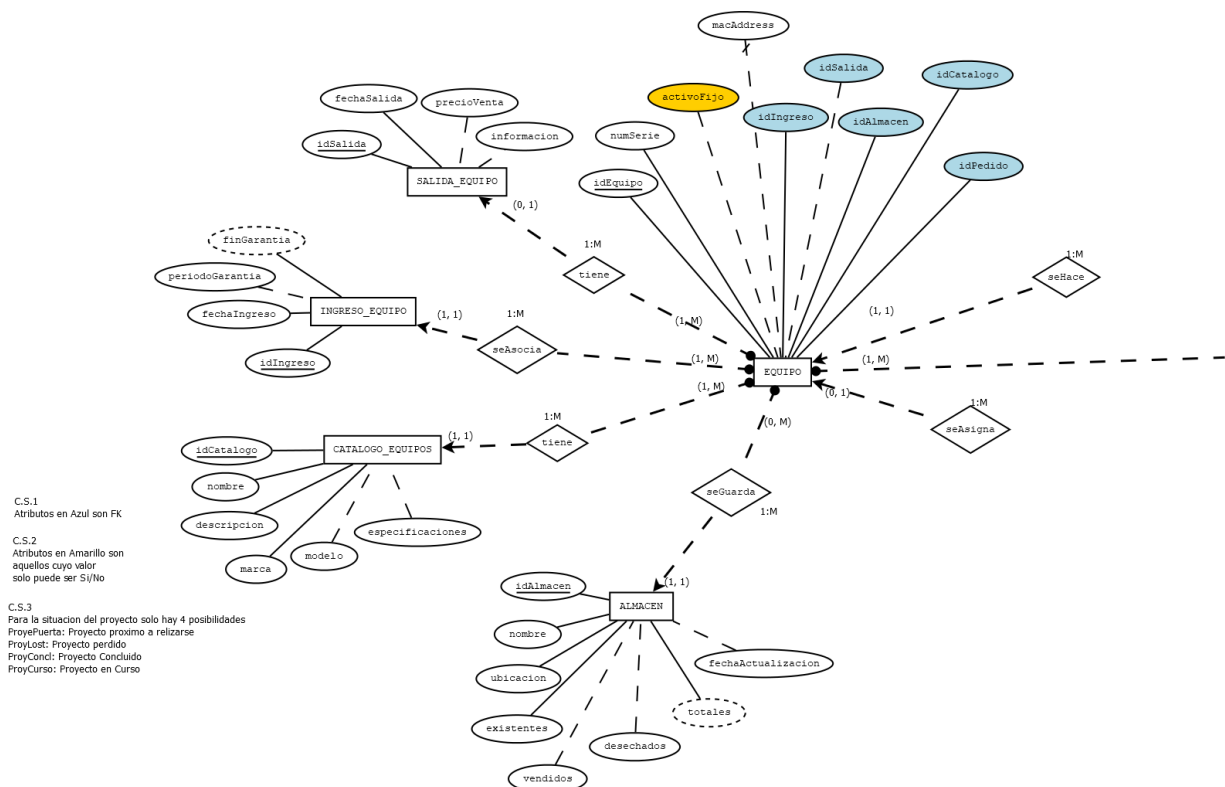


Figura 1: Diagrama Entidad-Relación (ER) del sistema de inventario, parte 1. Fuente: Elaboración propia con DIA.

Anexo 2. Diagrama Relacional del sistema de inventario

Este anexo presenta el Diagrama Relacional del sistema de inventario desarrollado para la empresa. El modelo relacional constituye una traducción formal del Diagrama Entidad–Relación (ER) al nivel lógico, adoptando una estructura tabular orientada a su implementación en el gestor de bases de datos **MySQL**.

A diferencia del modelo conceptual, esta representación lógica permite definir con precisión la arquitectura de almacenamiento, asegurando la integridad y consistencia de la información mediante los siguientes elementos técnicos:

- **Normalización:** El diseño ha sido sometido a un proceso de normalización (hasta la Tercera Forma Normal) para eliminar la redundancia de datos y prevenir anomalías en las operaciones de inserción, actualización y borrado.
- **Definición de Claves (Keys):** Se especifican de manera unívoca las Claves Primarias (*Primary Keys*) de cada tabla y las Claves Foráneas (*Foreign Keys*) que establecen los vínculos de integridad referencial entre las entidades.
- **Restricciones de Integridad:** Se incluyen reglas de negocio aplicadas a nivel de base de datos, tales como restricciones de obligatoriedad (**NOT NULL**), unicidad (**UNIQUE**) y acciones referenciales en cascada (**ON DELETE / ON UPDATE**).
- **Mapeo de Cardinalidades:** Traducción técnica de las relaciones 1:1, 1:N y N:N identificadas en el modelo conceptual, incluyendo la creación de tablas asociativas donde fue necesario para la resolución de relaciones muchos a muchos.

El diagrama fue elaborado utilizando la herramienta profesional **ER/Studio**, lo cual facilitó la generación de un esquema detallado que sirvió como base para la codificación del modelo físico y la posterior configuración del **ORM** de **Django**.

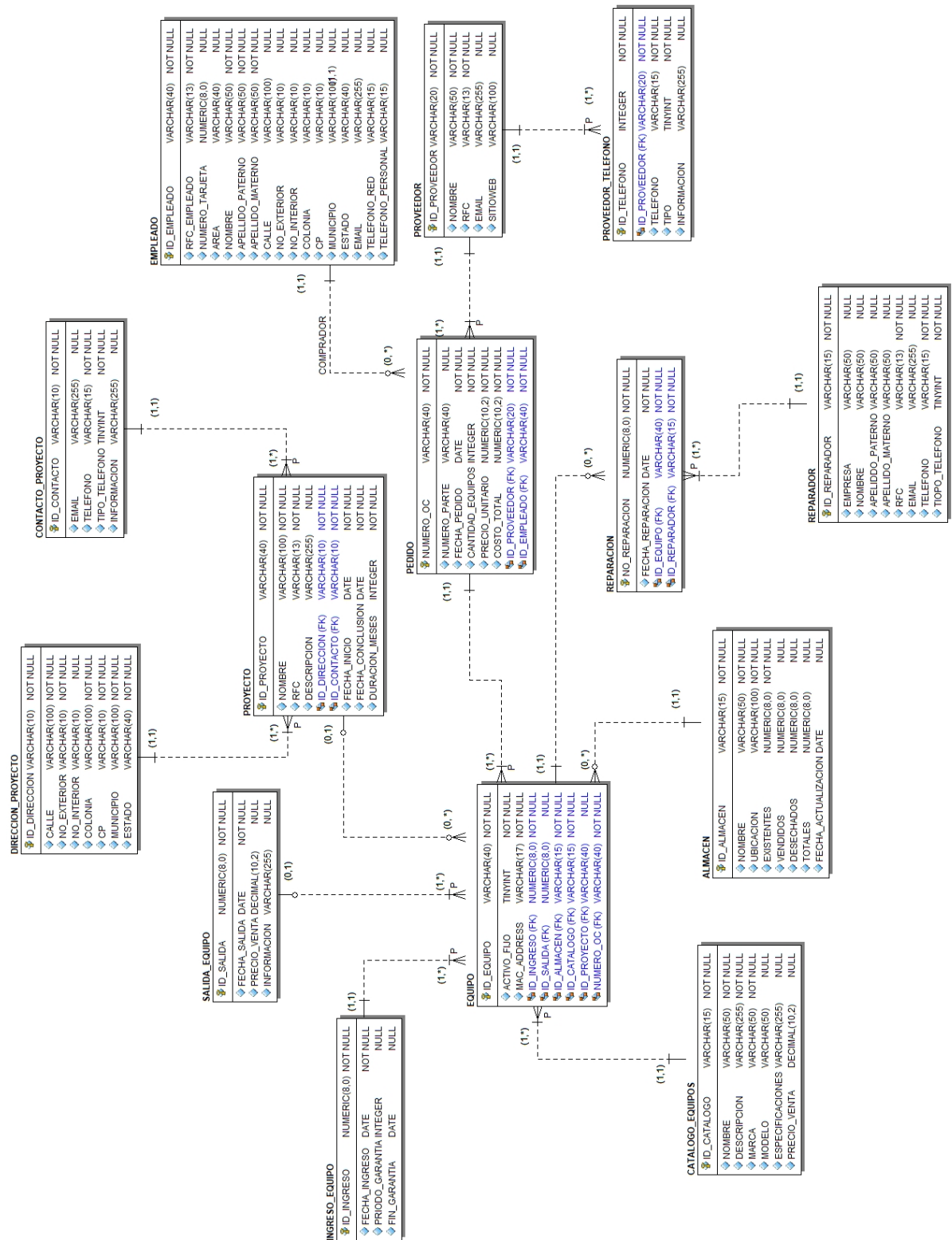


Figura 3: Diagrama relacional del sistema de inventario. Fuente: Elaboración propia con ER/Studio.

Anexo 3. Código SQL del modelo físico del sistema de inventario

Este anexo contiene una muestra representativa del código fuente en lenguaje **SQL** utilizado para definir el modelo físico de la base de datos correspondiente al sistema de inventario desarrollado para la empresa.

El código fue elaborado y probado en el entorno **MySQL Workbench 8.0 CE** y se diseñó conforme a los principios de modelado relacional, integridad referencial y optimización de consultas. El archivo completo incluye:

- Definición de tablas y restricciones de dominio.
- Asignación de claves primarias (*PK*) y foráneas (*FK*).
- Restricciones de integridad (*constraints*) como NOT NULL, UNIQUE, CHECK y expresiones regulares.
- Índices en los campos de consulta frecuente y relaciones foráneas.
- Columnas generadas (*STORED/VIRTUAL*) y *triggers* automatizados para mantener la integridad de los datos.

El *script* se ha estructurado en secciones comentadas para facilitar su lectura y seguimiento. A continuación, se presenta el fragmento de código correspondiente a la definición de tablas clave, disparadores e índices:

Tablas

```
-- Tabla de almacenes
CREATE TABLE almacen (
  id_almacen          VARCHAR(15) PRIMARY KEY,
  nombre              VARCHAR(50) UNIQUE NOT NULL,
  ubicacion           VARCHAR(100) NOT NULL,
  existentes           INT NOT NULL CHECK (existentes >= 0),
  vendidos            INT DEFAULT 0 CHECK (vendidos >= 0),
  desechados          INT DEFAULT 0 CHECK (desechados >= 0),
  totales              INT AS (existentes + vendidos + desechados)
    STORED,
  fecha_actualizacion DATE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
  utf8mb4_0900_ai_ci;

-- Tabla de equipos
CREATE TABLE equipo (
  id_equipo           VARCHAR(15) PRIMARY KEY,
  no_serie            VARCHAR(20) UNIQUE,
  activo_fijo         BOOLEAN,
  mac_address         VARCHAR(17)
  CHECK (mac_address IS NULL OR mac_address REGEXP '^[0-9A-Fa-f
    ]{2}[:-]{5}([0-9A-Fa-f]{2})$') UNIQUE,
  id_ingreso          INT NOT NULL,
  id_salida           INT,
  id_almacen          VARCHAR(15) NOT NULL,
  id_catalogo         VARCHAR(15) NOT NULL,
  id_pedido           VARCHAR(15) NOT NULL,

  CONSTRAINT fk_equipo_ingreso
  FOREIGN KEY (id_ingreso) REFERENCES ingreso_equipo(id_ingreso)
    ON DELETE RESTRICT,
  CONSTRAINT fk_equipo_salida
  FOREIGN KEY (id_salida) REFERENCES salida_equipo(id_salida)
    ON DELETE SET NULL,
  CONSTRAINT fk_equipo_almacen
  FOREIGN KEY (id_almacen) REFERENCES almacen(id_almacen)
    ON DELETE RESTRICT,
  CONSTRAINT fk_equipo_catalogo
  FOREIGN KEY (id_catalogo) REFERENCES catalogo_equipos(
    id_catalogo) ON DELETE RESTRICT,
  CONSTRAINT fk_equipo_pedido
  FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido)
    ON DELETE CASCADE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
utf8mb4_0900_ai_ci;
```

Fragmento de código 5.1: Ejemplo de definición de tablas en SQL. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.

Triggers

```
-- Trigger para actualizar el campo fecha_actualizacion en
almacen
DELIMITER //

CREATE TRIGGER before_update_almacen
BEFORE UPDATE ON almacen
FOR EACH ROW
BEGIN
-- Actualizar la fecha de modificacion si cambian las
existencias o conteos
IF NEW.existentes != OLD.existentes OR
NEW.vendidos != OLD.vendidos OR
NEW.desechados != OLD.desechados THEN
SET NEW.fecha_actualizacion = CURRENT_DATE;
END IF;
END //

DELIMITER ;

-- Trigger de validacion para asignacion de equipos
DELIMITER //

CREATE TRIGGER before_insert_asignacion
BEFORE INSERT ON asignacion_equipo
FOR EACH ROW
BEGIN
IF EXISTS (
SELECT 1 FROM asignacion_equipo
WHERE id_equipo = NEW.id_equipo AND fecha_devolucion IS NULL
) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'El equipo ya cuenta con una asignacion
activa.';
END IF;
END //

DELIMITER ;
```

Fragmento de código 5.2: Ejemplos de triggers utilizados en el sistema de inventario. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.

Índices

```
-- EQUIPO (Indices para llaves foraneas y busquedas)
CREATE INDEX idx_equipo_almacen ON equipo(id_almacen);
CREATE INDEX idx_equipo_catalogo ON equipo(id_catalogo);
CREATE INDEX idx_equipo_pedido ON equipo(id_pedido);
CREATE INDEX idx_equipo_ingreso ON equipo(id_ingreso);
CREATE INDEX idx_equipo_salida ON equipo(id_salida);

-- ASIGNACION EQUIPO (Indices para trazabilidad y consultas
    compuestas)
CREATE INDEX idx_asignacion_empleado ON asignacion_equipo(
    id_empleado);
CREATE INDEX idx_asignacion_proyecto ON asignacion_equipo(
    id_proyecto);
CREATE INDEX idx_asignacion_equipo ON asignacion_equipo(
    id_equipo);
CREATE INDEX idx_asig_equipo_fecha ON asignacion_equipo(
    id_equipo, fecha_asignacion);

-- PROVEEDOR
CREATE INDEX idx_proveedor_nombre ON proveedor(nombre);
CREATE INDEX idx_proveedor_rfc ON proveedor(rfc_proveedor);

-- PEDIDO
CREATE INDEX idx_pedido_proveedor ON pedido(id_proveedor);
CREATE INDEX idx_pedido_fecha ON pedido(fecha_pedido);
CREATE INDEX idx_pedido_comprador ON pedido(comprador);
CREATE INDEX idx_pedido_prov_fecha ON pedido(id_proveedor,
    fecha_pedido);
```

Fragmento de código 5.3: Índices definidos para optimizar consultas frecuentes. Fuente: Elaboración propia en MySQL Workbench 8.0 CE.

*Nota: El código **SQL** completo, que incluye la definición de todas las tablas, vistas, columnas generadas e índices, está disponible como archivo digital y puede consultarse públicamente en el siguiente repositorio:*

- **GitHub:** https://github.com/TastaturRitter/Base_Conecta

Este *script* fue validado con datos ficticios y comprobado para garantizar la integridad referencial, la coherencia estructural y la eficiencia en las consultas del sistema.

Anexo 4. Código representativo del sistema en Django

Este anexo presenta una selección estratégica de fragmentos de código fuente desarrollados bajo el *framework* **Django**, los cuales constituyen el núcleo de la lógica de negocio y la interfaz operativa del sistema de inventario. La estructura del sistema se sustenta en el patrón de desarrollo MVC (*Model-View-Template*), lo que asegura una separación bien definida entre el acceso a los datos, la lógica de control, la capa de presentación y los módulos de seguridad.

El sistema desarrollado y documentado en este informe muestra el uso de lineamientos modernos de desarrollo de *software*, entre los que se incluyen:

- **Modelado de datos mediante ORM:** Se usó la herramienta *Object-Relational Mapping* que incorpora Django para aislar la complejidad de las consultas SQL, con el fin de asegurar la integridad referencial y lograr una gestión eficiente de los registros del sistema.
- **Vistas Basadas en Clases (CBVs):** Se incorporó la lógica modular para las operaciones de tipo CRUD (*Create, Read, Update, Delete*), aspecto que facilitó el mantenimiento del código e incentivó la reutilización de componentes.
- **Seguridad y Control de Acceso (RBAC):** Se integraron *mixins* personalizados en el *backend* del sistema para implementar el Control de Acceso Basado en Roles que se solicitó desde el principio, esto con el fin de asegurar la protección de las funciones críticas del sistema frente a todos aquellos que no cuenten con los permisos necesarios.
- **Enrutamiento Modular:** Organización de rutas mediante el sistema de *URLs* de Django, permitiendo una navegación intuitiva y una estructura de proyecto escalable.

A continuación, se detallan los componentes más representativos del sistema:

inventario\models.py

El siguiente fragmento muestra la definición del modelo de datos en **Python**, utilizando el **ORM** para mapear la clase con la tabla física en **MySQL**.

```
class Almacen(models.Model):
    id_almacen = models.CharField(primary_key=True, max_length=15)
    nombre = models.CharField(max_length=50, unique=True)
    ubicacion = models.CharField(max_length=100)
    existentes = models.PositiveIntegerField()
    vendidos = models.PositiveIntegerField(default=0)
    desechados = models.PositiveIntegerField(default=0)
    fecha_actualizacion = models.DateField(null=True, blank=True)

    @property
    def totales(self):
        return self.existentes + self.vendidos + self.desechados

    def __str__(self):
        return f"{self.id_almacen} ({self.nombre}) - Ubicación: {self.ubicacion}"

    def get_absolute_url(self):
        return reverse('almacen_details', kwargs={'pk': self.pk})

    class Meta:
        db_table = 'almacen'
        managed = False
```

Fragmento de código 5.4: Definición del modelo Almacen en models.py. Fuente: Elaboración propia en Django.

inventario\views.py

Se presentan las vistas basadas en clases (*Class-Based Views*), integrando los *mixins* de seguridad para restringir el acceso según el nivel de jerarquía del usuario.

```
from django.views.generic import ListView, CreateView,
    UpdateView, DeleteView, DetailView
from django.urls import reverse_lazy

from .models import Almacen, CatalogoEquipos, Equipo,
    IngresoEquipo, SalidaEquipo
from .forms import AlmacenForm, CatEqoForm, EquipoForm,
    IngEqoForm, SalEqoForm
from utils.mixins import (
    RestrictedDeleteMixin, LectoresOSuperioresMixin,
    EditoresOSuperioresMixin, GestoresOSuperioresMixin)

class AlmacenListView(LectoresOSuperioresMixin, ListView):
    model = Almacen
    template_name = 'inventario/almacen_list.html'
    context_object_name = 'almacenes'
    paginate_by = 10

class AlmacenDetailView(LectoresOSuperioresMixin, DetailView):
    model = Almacen
    template_name = 'inventario/almacen_details.html'
    context_object_name = 'almacen'

class AlmacenCreateView(GestoresOSuperioresMixin, CreateView):
    model = Almacen
    form_class = AlmacenForm
    template_name = 'inventario/almacen_create.html'
    success_url = reverse_lazy('almacen_list')

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['cancel_url'] = reverse_lazy('almacen_list')
    return context
```

Fragmento de código 5.5: Ejemplo de vistas para listado y creación de almacenes en views.py. Fuente: Elaboración propia en Django.

admin_inventory\urls.py

Configuración de las rutas principales del proyecto, utilizando la función `include` para mantener la modularidad por aplicación.

```
from django.contrib import admin
from django.urls import path, include
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('supertabla/', include('super_tabla.urls')),
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
    path('empleados/', include('empleados.urls')),
    path('inventario/', include('inventario.urls')),
    path('pedidos/', include('pedidos.urls')),
    path('proyectos/', include('proyectos.urls')),
    path('users/', include('users.urls')),
    path('reparaciones/', include('reparaciones.urls')),
]
```

Fragmento de código 5.6: Configuración de rutas principales en `urls.py`. Fuente: Elaboración propia en Django.

Nota: El código completo que incluye la definición de todas las vistas, plantillas y funciones está disponible como archivo digital y puede consultarse públicamente en el siguiente repositorio:

- **GitHub:** https://github.com/TastaturRitter/Base_Conecta

Este proyecto fue validado para garantizar la integridad referencial y la eficiencia en las consultas del sistema en un entorno de producción real.