UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de manejo, gestión y control en actividades de actualización para la formación y desarrollo profesional de los docentes

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A N

Carolina Kennedy Villa

César Mauricio Ramos Villaseñor

DIRECTORA DE TESIS

M.ED. María del Rosario Barragán Paz

Ciudad Universitaria, Cd. Mx., 2024

Agradecimientos

A nuestra tutora la Mta. María del Rosario Barragán Paz, de cariño Chary, por ser nuestra guía, asesora y mentora durante este proyecto, así como durante gran parte de nuestra estancia en la Facultad de Ingeniería. Gracias por la dedicación, por la compañía y por la experiencia otorgada durante estos años trabajando juntos.

A nuestras familias por brindarnos la paciencia y el apoyo necesario desde nuestra educación básica hasta llegar a nuestra meta de ser profesionistas. Gracias por los abrazos, los cafés calientes en los desvelos, los almuerzos y todo el esfuerzo que realizaron y sacrificaron para que nosotros pudiéramos llegar hasta aquí. No hay palabras para describir la gratitud que siempre les tendremos.

A nuestros amigos que nos acompañaron a lo largo de todos los semestres. Aquellos que siempre estuvieron para levantarnos el ánimo, contar algunos chistes y hacer amenos nuestros días. En este camino para nuestra fortuna no se viaja sólo, de lo contrario sería el doble de difícil.

A UNICA por brindarnos un segundo hogar en donde estudiar, en donde experimentar sin temor y crear nuestras primeras experiencias como profesionistas. Sin UNICA no podríamos ser los ingenieros que hoy deseamos ser.

Y a nosotros, por no abandonar el camino a la mitad ni al final, por resistir y darnos apoyo y compañerismo mutuo. Gracias por siempre confiar el uno al otro.

Índice de contenido

Introducción

La capacitación docente es un proceso fundamental en el ámbito educativo que busca mejorar las habilidades, conocimientos y competencias de los profesionales de la enseñanza, con el objetivo de elevar la calidad de la educación que brindan a los estudiantes.

Las organizaciones cuya misión u objetivo principal es la impartición de actividades como cursos, diplomados, seminarios y conferencias orientadas a la capacitación docente e incluso orientadas a la divulgación del conocimiento, llegan a presentar áreas de oportunidad en cuanto al manejo de su información y las herramientas que implementan para el desarrollo y preparación de sus labores.

Existen organizaciones cuyo manejo de la información consiste en cantidades considerables de archiveros que ocupan salas de trabajo, o sistemas cuyo tiempo de respuesta suele alentar las capturas de información y generación de reportes estadísticos.

El cometido de este trabajo es ofrecerles a aquellas instituciones, organizaciones y dependencias sin fines de lucro o meramente pertenecientes al sector educativo, una herramienta con la cual estas áreas de oportunidad se vean cubiertas, mejorando así el desempeño de sus labores que traerá como consecuencia un impacto directo en la enseñanza.

Bajo esta visión nace el sistema MAGENTA: Manejo, gestión y control de la información para la capacitación docente.

Objetivos

Objetivo General

Realizar un software que se encargue de almacenar la información acerca de actividades para la capacitación docente, tales como cursos, diplomados, ciclos de conferencias, seminarios, eventos, foros y talleres.

El software tendrá como funciones la gestión y almacenamiento de dichas actividades y sus usuarios, así como la generación de evaluaciones, reportes, reconocimientos y constancias, automatizando diversos procesos dentro de la rutina diaria de los institutos u organizaciones encargadas del desarrollo docente, para conseguir un aprovechamiento de recursos que mejore su calidad.

Objetivos Particulares

- Investigar los antecedentes del proyecto para ampliar la visión de éste.
- Definir las reglas de negocio del caso de estudio del proyecto.
- Definir las herramientas y tecnologías con las que se desarrollará el sistema.
- Desarrollar el sistema de acuerdo con lo establecido en los requerimientos y el maquetado, utilizando las herramientas establecidas por el equipo de trabajo.
- Realizar pruebas con el sistema desarrollado.

Capítulo 1. Antecedentes

1.1 Aplicaciones web actuales

Las aplicaciones web son un software (conjunto de instrucciones e información que opera una computadora para ejecutar una tarea en específico), almacenado dentro de servidores de internet y, a diferencia de cualquier otra aplicación, no requieren de un proceso de instalación en alguna memoria secundaria en los dispositivos. Para su uso, únicamente requieren una conexión a internet además de algunos datos básicos de acceso como usuario y contraseña.

Estos softwares pertenecen a un subconjunto de sitios web donde despliegan información de todo tipo; sin embargo, las aplicaciones web no se destinan únicamente a esparcir información estática. En la actualidad pueden ofrecer un servicio (por ejemplo, para ordenar comida o comprar artículos); o bien pueden funcionar como herramienta para una tarea en específico (como tomar cursos en línea o mensajería instantánea).

Su funcionamiento consiste en una interfaz de usuario con la que se puede interactuar a través de botones y menús, los cuales reciben las solicitudes de éste para así ejecutar la tarea solicitada y generar la información requerida o almacenarla en la base de datos.

Como se mencionó previamente, las únicas condiciones para acceder a las aplicaciones web son tener una conexión a internet y un navegador (software instalado). Contar con este último requerimiento evitará que existan problemas de compatibilidad, pues en todo dispositivo existe al menos un navegador incorporado.

Un detalle importante por mencionar es el hecho de que todas las aplicaciones a las cuales ingresamos no ocupan espacio en el disco duro y esto se debe a que se encuentran almacenadas en la red. De esta forma, obtenemos otras ventajas como recibir actualizaciones automáticas del software y de seguridad informática, así como ser escalables. (Londoño, 2023), (Yeeply, s.f.), (GCF Global, s.f.)

Existe una amplia gama de aplicaciones web donde podemos encontrar prácticamente todo lo que nos haga la vida más sencilla. Como ejemplos podemos mencionar el correo electrónico donde se ubican los sitios de Gmail y Outlook; las de trabajo colaborativo que permiten edición y creación de archivos además de ofrecer otras herramientas de trabajo remoto como Google Docs y Microsoft Office; las de comercio electrónico que permiten comprar productos en línea tales como Amazon y Mercado Libre; o bien, las aplicaciones web usadas por las instituciones bancarias que permiten a los usuarios ingresar a sus cuentas para realizar pagos, transferencias de dinero, consulta de saldos, entre otras actividades.

1.2 Beneficios de las nuevas tecnologías

Sin duda, la tecnología se volvió parte de nuestra vida cotidiana a raíz de la revolución que vino a generar en el mundo y han sido tan radicales los cambios, que ahora somos prácticamente dependientes de ella, no sólo por la comodidad de realizar las actividades en forma más sencilla, sino por la rapidez con que podemos lograrlas.

Cada día se descubren cosas nuevas como lo fue en su momento la electricidad o las ondas de radio (Ismail, 2017), entre otras; estos descubrimientos han ido mejorando la calidad de vida y el trabajo de las personas, además de lograr grandes avances en cuestión de ciencia y medicina.

Estos avances se han aplicado también al internet, debido a esto es que hoy en día, existen un sin fin de aplicaciones que resuelven diferentes problemáticas, todas ellas al alcance de nuestros dedos.

Actualmente, el 60% de la población tiene acceso a internet, del cual 3,000 millones cuentan con un teléfono inteligente (Santander, 2021). De acuerdo con la Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) desarrollada por el INEGI, se estimó que "en 2021 había 88.6 millones de personas usuarias de internet, lo que representó 75.6% de la población de seis años o más en México" (Instituto Federal de Telecomunicaciones, 2021, párrafo primero). Gracias a esto, la innovación tecnológica nos ha dado la capacidad de vivir cómodamente otorgándoles el acceso a la información. De acuerdo con el estudio Data Never Sleeps 7.0 efectuado por la empresa de Domo Business Cloud, se realizan 4.5 millones de búsquedas por minuto desde cualquier lugar en el motor de búsqueda más popular de internet que es Google.

Pruebas fehacientes de los beneficios tecnológicos pudieron vivirse durante la pandemia por Covid19. En primera instancia, se revolucionó la manera de comunicarnos, pues gracias a las videollamadas o videoconferencias pudimos romper cualquier distancia y -aunque antes ya se hacía con una simple llamada telefónica- la gran diferencia fue que ya conseguimos ver a nuestros interlocutores.

Durante este período también se dio el auge en los contenidos de entretenimiento, pues todos los sitios web de audio o videos en directo dominaron el internet. Lo anterior de acuerdo con el estudio realizado por Domo donde menciona que en 2019 se vieron por minuto 4.5 millones de videos en YouTube y un millón de transmisiones en Twitch.

Hablar de tecnología no es enfocarse únicamente en el entretenimiento. Existen infinidad de áreas de aplicación; por ejemplo, en la medicina, donde gracias al desarrollo de equipos y maquinaria que ayudan a detectar todo tipo de enfermedades o padecimientos en los seres humanos, se ha logrado incrementar la esperanza de vida.

La educación es otra de las áreas de aplicación en las que se puede sacar más provecho de la tecnología, ya que Internet tiene disponible la información necesaria en todos los niveles educativos para así complementar y fortalecer los conocimientos transmitidos en las distintas instituciones educativas. Además, las herramientas para transmitir estos conocimientos, como cámaras web, micrófonos o programas de chat en línea, se mejoraron en forma sustancial logrando que la calidad de comunicación mediante una computadora se hiciera óptima. Por ello, las clases en línea se han convertido en una buena opción para muchos profesores y estudiantes en distintos niveles.

1.3 Problemática que resolver

Sin duda, el mundo ha cambiado su perspectiva después de la pandemia que afectó a millones de personas dejando a su paso crisis en todos los niveles: económico, político, social y educativo. Sin embargo, también ha dejado una estela de nuevos conocimientos, un auge tecnológico histórico, así como el aprendizaje autodidacta, el cual, hasta antes de ese periodo, no había sido del todo necesario.

En México se ha venido observando que en el campo de la educación existen muchas áreas de oportunidad. Uno de los principales motivos es la forma en la que los docentes a cualquier nivel son escasamente capacitados, ya sea en nuevas materias o en las técnicas de enseñanza.

Hacer de la capacitación algo frecuente lograría un cambio muy beneficioso en cualquier persona interesada en la enseñanza, mejorando su labor profesional y, en consecuencia, elevando su calidad educativa que impactaría en forma directa la educación a nivel nacional.

Enseñar a un docente, no es tarea fácil. Se requiere del apoyo de distintas instituciones u organizaciones que estén al nivel para lograrlo y que ofrezcan programas educativos de excelencia, de fácil acceso además de ser planeados específicamente para las áreas, materias y niveles en los que se desarrollen.

Al involucrarse distintos factores para llevar a cabo una tarea de capacitación (en este caso nos referimos a instituciones que impartirán los cursos, locación, material didáctico, instructores, reconocimientos, etc.) se deriva uno de los problemas principales de la actualización. En cada una de estas etapas, se involucra a varias personas para que manejen toda la logística como el registro de los participantes, los cursos a impartirse, el número de aulas, evaluaciones y expedición de constancias. Si el proceso se realiza en forma ineficiente, la probabilidad de tener éxito será muy baja.

Algunas instituciones utilizan, como almacenes de datos, estantes de carpetas con formatos impresos donde se anexan los registros y fotocopias de documentos oficiales de los participantes. Algunas otras se ayudan con paquetería de software que cuenta con datos genéricos y no específicos ni del uso ni de la organización (como la paquetería Office o LibreOffice). También existen otras organizaciones que utilizan sistemas diseñados e implementados con más de 20 años de antigüedad, esto ocasiona que sea cada vez más difícil darles mantenimiento, además de que no poseen planes de respaldo o restauración de su información.

De acuerdo con el Instituto Nacional de Estadística y Geografía INEGI, durante el año 2020-2021 se contabilizaron 2 millones 10 mil 989 maestros en la encuesta (El Financiero, 2022) "Maestros y escuelas por entidad federativa según nivel educativo", los cuales desempeñan su labor profesional a lo largo de 250 mil 698 instituciones educativas.

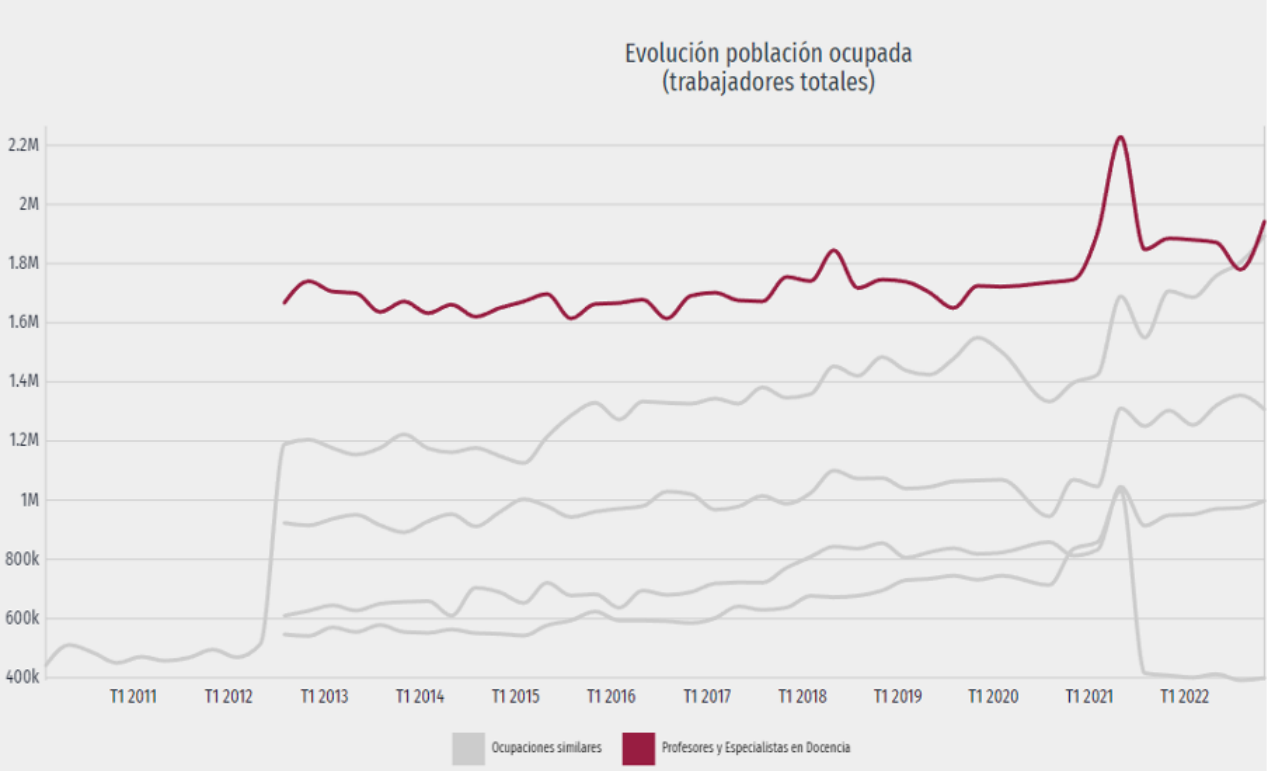


Ilustración 1. Gráfica de Profesores y Especialistas en Docencia (Data México, s.f.)

Bajo esta premisa, consideramos que una aplicación web actual y adaptable a las necesidades de cualquier organización, que destine parte de sus recursos a la capacitación docente, podría aumentar la eficiencia y producción de su labor, impactando fehacientemente no sólo en la calidad sino en las técnicas de enseñanza de su plantilla docente, traduciendo este esfuerzo en mejoras sustanciales, primero a nivel local y, posteriormente ya con el éxito probado, a nivel nacional.

La aplicación web sugerida debe desarrollarse con herramientas actuales robustas que tengan un futuro asegurado en el ramo tecnológico, garantizando su uso por varios años y su sistema únicamente sufra breves y continuas actualizaciones, sin dejar de estar a la vanguardia, tanto en su diseño como en la experiencia del usuario.

Propiamente, la aplicación web debe realizar tres actividades en concreto para aumentar la productividad y alcanzar su objetivo: generación de documentación automatizada, almacenamiento y gestión de datos eficientes y evaluaciones de calidad automatizadas.

1.4 Ejemplo de caso de estudio: Unidad de Servicios de Cómputo Académico en la Facultad de Ingeniería de la UNAM

Para poder desarrollar nuestra propuesta, fue necesario encontrar una entidad u organización que necesitara de ella. Al considerar los parámetros que nuestra aplicación debía cumplir, concluimos que podía ser aprovechada en su totalidad dentro de nuestra Facultad.

En la Facultad de Ingeniería dentro de la Universidad Nacional Autónoma de México (UNAM) existe la Unidad de Servicios de Cómputo Académico (UNICA), una organización que tiene como objetivo **proporcionar eficaz y eficientemente en el ámbito institucional, los servicios de Tecnologías de la Información y la Comunicación TIC y de las Tecnologías de Aprendizaje y el Conocimiento TAC, que coadyuven al proceso integral de formación del capital humano en la Facultad de Ingeniería.** (Unidad de Servicios de Cómputo Académico, 2023) Como puede verse, esta institución se encuentra comprometida con la capacitación docente y el desarrollo de la educación superior en México.

A partir de esto fue que, trabajando y recopilando información con dicha institución, desarrollamos un caso de estudio con el propósito de ejemplificar y realizar la aplicación web deseada que ayudaría a cualquier institución ajustándose a sus necesidades. A continuación, se detalla específicamente el caso de estudio propuesto:

Existe una organización, en este caso UNICA, que tiene como misión y objetivos la difusión y apropiación del conocimiento de Tecnologías de la Información y las Comunicaciones. Una de las maneras de cumplir con ello es por medio de la realización de actividades de capacitación como cursos, seminarios, conferencias, diplomados y más; orientados a docentes que deseen expandir sus conocimientos.

El problema principal de la institución es que necesita una base de datos para almacenar el historial de actividades cursadas, así como la información personal de cada docente que se registre dentro de ella. También, para la correcta impartición de las actividades, el sistema deberá encargarse de las tareas de logística (generación de formatos, asignación de sedes de impartición, etc.). Finalmente, la institución necesita realizar una evaluación constante, para que sea capaz de tomar decisiones que aumenten la calidad de sus actividades con respecto a sus instructores, sus aulas y sus contenidos. El sistema debe ser manipulado por usuarios administradores pertenecientes a la institución que poseen diferentes jerarquías y, por lo tanto, diferentes privilegios.

En los capítulos siguientes se profundizará en este caso de estudio; se establecerán las reglas de negocio; posteriormente, se propondrá una arquitectura para la aplicación y se expondrá una versión preliminar de dicha aplicación con resultados, así como se ahondará en todo el proceso de su diseño.

Capítulo 2. Diseño del caso de estudio, toma de requerimientos.

2.1 Definición del sistema: ¿Qué será MAGENTA? La razón de su nombre y su significado

Como se comentó en el capítulo anterior, UNICA es una institución cuya misión tiene como objetivo la capacitación docente. Para lograrlo, le es necesaria una aplicación o un sistema que le permita fungir como una base de datos global donde se registren los profesores, las actividades y sus avances.

Este sistema debe cumplir con tres características primordiales: confidencialidad, integridad y disponibilidad.

De acuerdo con la Facultad de Estudios Superiores Acatlán perteneciente a la U.N.A.M podemos describir las características anteriores como la tríada CIA, por sus siglas en inglés: **confidentiality, integrity** and **availability**.

Se entiende por confidencialidad: "el evitar que la información clasificada pueda ser revelada intencionalmente o no a alguien que no tiene acceso a ello". (F.E.S. Acatlán, 2015) La integridad establece que "la información sólo puede ser borrada o modificada por usuarios autorizados" (F.E.S. Acatlán, 2015); finalmente, la disponibilidad garantiza "el acceso a la información en un tiempo razonable a usuarios autorizados". (F.E.S. Acatlán, 2015)

Para cumplir con la demanda de la triada, se debe pensar en una base de datos eficiente, administrable y adaptable a cualquier entorno, que permita un manejo pulcro de datos cifrados y restringidos a miembros autorizados de la organización, además de contar con las respectivas medidas de seguridad en

la arquitectura física donde la aplicación o sistema se encuentre almacenado y operando.

Una opción viable para desarrollar esta aplicación o sistema podría ser a través de un sistema privado accesible mediante el protocolo HTTP el cual se administraría y accedería por medio de una VPN (Virtual Private Network).

El protocolo HTTP es aquel que opera al nivel de aplicación del modelo OSI, y su objetivo es la transmisión de documentos hipermedia, como lo son los archivos HTML (Developer Mozilla, s.f.). Fue diseñado para la comunicación entre navegadores web y sus servidores web, por lo que opera en una arquitectura tipo cliente-servidor, utilizando certificados digitales por medio de cifrados SSL/TLS. Dichos certificados digitales suelen tener un precio ante las entidades regulatorias. (CloudFlare, 2023)

Por otro lado, una VPN otorga privacidad y anonimato creando una red privada del internet público; además, enmascara una dirección IP para que las acciones realizadas no puedan ser rastreadas. Así, los servicios de VPN establecen conexiones seguras y encriptadas cuya finalidad es proporcionar al usuario privacidad. (Norton, 2023)

A partir de las premisas descritas anteriormente, se pensó en una aplicación web que ayudará a UNICA a cubrir los parámetros mencionados, y surgió el nombre de MAGENTA, cuya misión no era otra que el Manejo, la Gestión y el Control de la capacitación docente.

Desde este momento y hasta el final de nuestro escrito, hablaremos de los resultados obtenidos en la aplicación MAGENTA.

2.2 Descripción y diseño de las reglas de negocio del caso de estudio propuesto

De acuerdo con García y André "las reglas de negocio permiten definir las condiciones para establecer un negocio o precisar de qué forma se controlará el comportamiento de los eventos dentro de éste." (García González & André Ampuero, 2015) Dichas reglas son utilizadas para delimitar los objetivos y funcionalidad de un sistema que se desea desarrollar para cumplir con la necesidad de una empresa, organización, etc. "Las reglas de negocio son requisitos que definen cómo el negocio debe operar." (García González & André Ampuero, 2015). Es decir, enunciados breves y concisos cuyo fin es detallar una normativa que se cumple en las operaciones de la organización o negocio, evitando la ambigüedad o redundancia a toda costa. A partir de las reglas de negocio, es posible construir los requerimientos de un sistema. Con ellas, seremos capaces de desarrollar diagramas de clases, encontrar un patrón de diseño óptimo para el sistema e idear una propuesta de la arquitectura lógica de éste.

Como fue mencionado en el apartado 1.4, orientamos nuestros resultados y nuestra propuesta hacia la organización UNICA, aunque es importante aclarar que nuestra propuesta de sistema es completamente adaptable a las necesidades de cualquier organización dedicada a la capacitación docente, puesto que nuestra propuesta puede actualizarse de acuerdo con las necesidades de cualquier institución, sin la necesidad de interactuar en forma significativa con el código fuente de la aplicación.

En dicho apartado se detallan las reglas de negocio originadas y diseñadas después de un proceso colaborativo entre la organización y los sustentantes, que en este caso consistió en la realización de varias entrevistas.

A continuación, detallamos el caso de estudio:

Existe una organización, "UNICA" que imparte de forma periódica actividades educativas para docentes que deseen ampliar sus conocimientos. Estas actividades se restringen a los siguientes tipos: seminarios, cursos, talleres, cursos-talleres, diplomados, foros, eventos y conferencias. Todas las actividades poseen los mismos atributos, a diferencia de los diplomados, que están conformados por diversos módulos y que se imparten una vez por año, debido a su rigor, duración y temario.

Cada actividad es impartida por uno o varios instructores, los cuales son registrados y asignados por la organización "UNICA"; estos instructores, a su vez, pueden cursar diferentes actividades sin limitaciones.

El contenido y manejo de cada actividad depende de un departamento asociado a ella. Los departamentos son grupos de administrativos dentro de la organización que se encargan de un área de conocimientos específica; por ejemplo, el Departamento de Desarrollo e Investigación o el Departamento de Redes.

Para este caso de estudio existen los siguientes departamentos:

- *Departamento de Investigación y Desarrollo:* Encargado de actividades relacionadas con el desarrollo de software, administración de bases de datos y de temáticas del ámbito científico.
- *Departamento de Servicios Académicos:* Encargado de actividades relacionadas con las materias administrativas, tales como la distribución de paquetería de software Office 365, gestión de cuentas de correo electrónico institucionales y los talleres de introducción a aplicaciones realizadas por la organización.

- *Departamento de Seguridad en Cómputo:* Encargado de funciones relacionadas con la seguridad informática. Realiza actividades técnicas orientadas a la aplicación de herramientas para la ciberseguridad y la divulgación.
- *Departamento de Redes y Operación de Servidores:* Encargado de actividades acerca del uso de servidores computacionales y administración de computadores con sistemas operativos UNIX.
- *Departamento de Pedagogía:* Encargado de actividades instructivas para mejorar la calidad de enseñanza de los instructores adscritos a la organización, así como de todo el profesorado. Realiza actividades con temáticas asociadas a las habilidades blandas, como son el liderazgo, la creatividad y la ponencia.

La organización posee varias sedes que utiliza conforme a la demanda de las actividades; éstas pueden ser virtuales (por medio de videoconferencias) o físicas (salones, auditorios, etc.)

La organización necesita conocer por cada profesor la división a la que pertenecen y sus puestos de trabajo. Las divisiones son secciones dentro del organigrama de la Facultad de Ingeniería de la U.N.A.M. que se encargan de una cantidad fija de materias pertenecientes a uno o varios planes de estudio para una o varias licenciaturas. Un profesor puede pertenecer a una o más divisiones, dependiendo de las materias que imparte o de su área de trabajo dentro de la Facultad de Ingeniería.

Por otro lado, dentro del organigrama de la Facultad de Ingeniería, a un académico se le asignan puestos de trabajo, tales como "Profesor de Asignatura A" o "Ayudante de Profesor B". La organización necesita estar enterada de la categoría y nivel, con el fin de designar a los instructores adscritos y saber si

los académicos están preparados para cursar ciertas actividades, tales como diplomados o seminarios específicos.

Un requerimiento importante para la organización en cuanto al sistema a desarrollar es la asignación de los perfiles para cada usuario, ya que puede haber administradores y usuarios temporales.

Los administradores deben ser personas adjuntas, asociados a cada departamento al que pertenecen para que registren las actividades, horarios y evaluaciones en el sistema. Existirán dos tipos de administradores en el sistema: aquellos con el rol de jefe del departamento y aquellos con el rol de ayudantes del departamento, la diferencia entre ambos será que los primeros podrán tener acceso a todas las secciones y funciones del sistema, mientras que los ayudantes únicamente tendrán acceso a secciones relacionadas con las actividades, los profesores y las sedes de impartición, excluyendo las funcionalidades relacionadas a los puestos de trabajo, departamentos y divisiones, así como no podrán administrar cuentas de acceso.

Los usuarios temporales tendrán como único objetivo contestar encuestas de evaluación de las actividades impartidas. Para este fin, deberán contar con una clave de grupo y su nombre de usuario; además de proporcionar un correo electrónico al momento de inscribirse a la organización; esta medida le permitirá al sistema detectar si dicho docente realmente cursó la actividad en el grupo que está indicado por medio de su clave de adscripción (la cual funcionará como contraseña). El usuario solo podrá llenar dicha encuesta una vez.

Finalmente, la organización necesita que el sistema genere formatos estadísticos tomando en cuenta las evaluaciones registradas. Estos documentos varían dependiendo de si son generados por departamentos, por actividades,

por periodos, etc. También el sistema deberá encargarse de formatos logísticos, tales como publicidad de actividades, listas de asistencia, constancias, reconocimientos, diplomas, etc. Todos los formatos serán detallados en la sección 2.3 donde se desglosan los requerimientos funcionales y no funcionales del sistema.

2.3 Descripción de requerimientos funcionales y no funcionales

De acuerdo con el Departamento de Ciencias de la Computación e Inteligencia Artificial, definiremos requerimientos como "Propiedades o restricciones determinadas de forma precisa que deben satisfacerse", y se puede decir que para los proyectos de software son una descripción de lo que un programa debe hacer para satisfacer las necesidades de los usuarios y, sobre todo, servir como una guía de acciones específicas (Northware, 2022) que debe realizar el ingeniero de software durante el desarrollo.

Dentro de los requerimientos encontraremos dos tipos:

- *Requerimientos funcionales:* Describen el funcionamiento o servicios que prestará el sistema, generalmente las entradas, salidas y los datos que se almacenan en él. De igual forma, se puede indicar lo que no hará el sistema.
- *Requerimientos no funcionales:* Describen cualidades que debe tener el sistema relacionados con el rendimiento, tiempo de entrega, documentación, seguridad, entre otros. (Villa, 2017)

En otras palabras, los requerimientos funcionales definen qué debe hacer un sistema y los no funcionales indican cómo debe de ser. (Departamento de Ciencias de la Computación e I.A)

De acuerdo con la información descrita en el apartado 2.2, definimos los siguientes requerimientos:

2.3.1 Requerimientos funcionales

A continuación, se describen por secciones las acciones que el sistema debe ser capaz de realizar.

El sistema debe ser capaz de:

2.3.1.1 Actividades

- Almacenar un catálogo de actividades para, a posteriori, ser programadas en una fecha determinada con uno o varios instructores en específico. Las actividades registradas en el catálogo siempre podrán ser editadas y eliminadas en cualquier momento.
- Generar formatos específicos referentes a una actividad programada, como su lista de asistencia y su publicidad.
- Inscribir y enlistar participantes de una actividad programada.
- Capturar la información de cada participante en una actividad programada, como su calificación, montos de pago y cualquier dato referente a su inscripción.
- Asignar y mostrar los instructores de una actividad programada.
- Actualizar y eliminar la información referente a una actividad programada.
- Generar los reportes referentes a las actividades programadas, como el reporte de evaluación, el reporte de instructores y el reporte de sugerencias.
- Crear, editar y eliminar diplomados asignables a un catálogo de actividades que fungirá como un módulo de diplomado.

- Generar constancias y reconocimientos de una actividad programada, así como generar diplomas en el caso de los diplomados.

2.3.1.2 Profesores

- Almacenar, editar y eliminar la información personal y académica de los profesores o docentes inscritos en la organización.
- Consultar el registro de todos los profesores inscritos en la organización.
- Almacenar las divisiones y los puestos de trabajo a los que pertenece un profesor.
- Generar el reporte histórico de las actividades programadas que ha cursado un profesor.

2.3.1.3 Puestos de trabajo

- Almacenar y mostrar las categorías asociadas a los profesores.
- Crear, eliminar y editar una categoría dentro del catálogo.

2.3.1.4 Sedes

- Almacenar y mostrar un catálogo de sedes pertenecientes a la organización que serán asociadas a una actividad programada.
- Crear, eliminar y editar una sede dentro del catálogo.

2.3.1.5 Departamentos

- Almacenar y mostrar un catálogo de departamentos pertenecientes a la organización que serán asociados a actividades y administradores.
- Crear, eliminar y editar un departamento dentro del catálogo.

2.3.1.6 Divisiones

- Almacenar y mostrar un catálogo de divisiones pertenecientes a la organización que serán asociadas a los profesores.
- Crear, eliminar y editar una división dentro del catálogo.

2.3.1.7 Administradores

- Registrar administradores asociados a un departamento en específico, los cuales tendrán la posibilidad de crear cursos, generar formatos, inscribir participantes, registrar profesores, entre otras actividades. Poseerán acción únicamente sobre los cursos asociados a su departamento.
- Asignar el rol de jefe o ayudante a un administrador. Aquellos administradores con rol de jefe tendrán el privilegio de editar, crear y eliminar las demás cuentas de administrador de su departamento asociado, así como de modificar su rol.
- Siempre debe existir al menos una cuenta de administrador por cada departamento con rol de jefe, de tal forma que nunca deje de existir un administrador para cualquier departamento.

2.3.1.8 Misceláneos

- En las secciones deberá existir un buscador para identificar la información de manera rápida.
- Identificar a los diferentes tipos de usuario a través de un formulario para el inicio de sesión. De ser algún participante que sólo evaluará, el formulario será diferente para reconocer la actividad a evaluar.
- Tener una pantalla de inicio que contenga las acciones más utilizadas
- Realizar evaluaciones dentro del mismo sistema para las actividades cursadas, así como sus instructores relacionados con las mismas.

2.3.2 Requerimientos no funcionales

- El sistema deberá alojarse en un servidor web y su tiempo de respuesta no debe exceder de cinco minutos
- Se debe utilizar una base de datos relacional
- Los reportes y documentos generados no deberán ser almacenados en la base de datos, únicamente los datos de su contenido
- La sesión de usuario debe expirar a los cinco minutos de inactividad
- El sistema debe responder adecuadamente a múltiples peticiones de acceso a recursos en la base de datos.
- Información sensible como las contraseñas deben almacenarse cifradas en la base de datos.
- El sistema debe operar sin inconvenientes con al menos 10 sesiones de usuario abiertas. Un usuario puede tener muchas sesiones abiertas.
- El sistema deberá contar con un manual de usuario y tendrá la característica de ser entendible y legible para cualquiera que lo consulte.
- El sistema notificará con ventanas emergentes posibles errores o advertencias con mensajes claros y detallistas.
- El sistema estará disponible a través del protocolo HTTP y debe operar adecuadamente al menos para los siguientes exploradores de internet: Google Chrome, Microsoft Edge y Firefox. Será posible acceder a él por medio de cualquier dispositivo capaz de ejecutar dichos navegadores.
- No se necesita que el sistema sea responsivo a cualquier dimensión de pantalla.

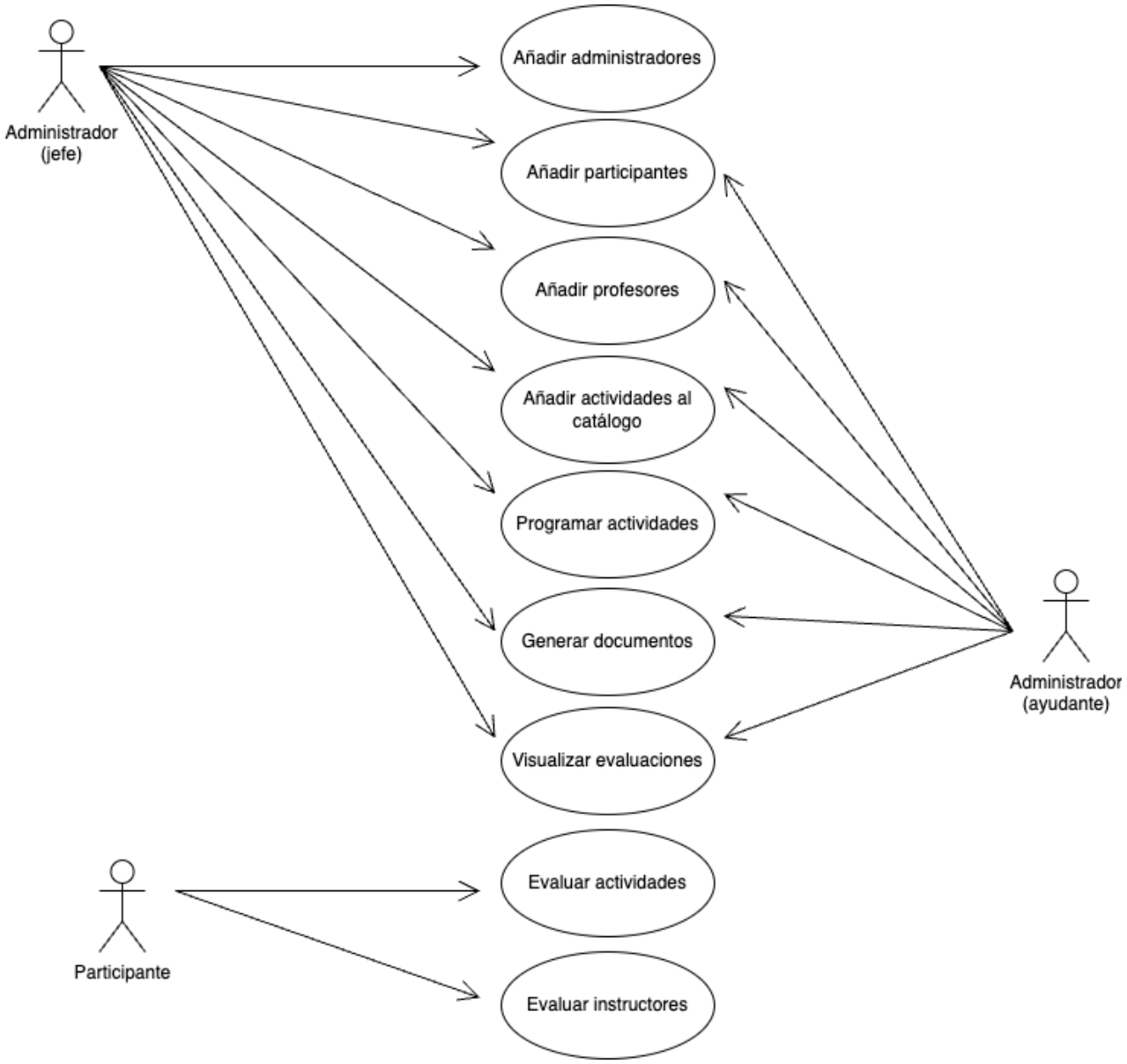


Ilustración 2. Diagrama UML

Tabla 1. Documentos requeridos

Documento	Descripción
Reporte de participantes	Documento que enlista todas las actividades del periodo dado, denotando su coordinación y el número de participantes inscritos en cada una.
Criterio de aceptación	Documento que enlista por departamento seleccionado el valor del criterio de aceptación por los cuatro periodos del año más un promedio general. Este se calcula a partir de la pregunta cuatro de la encuesta de evaluación de una actividad. Si un periodo no tiene evaluaciones, no se cuenta en el promedio general. Únicamente son tomadas en cuenta para los promedios por periodo las actividades que cuenten con al menos una encuesta contestada. Los totales de los promedios se toman a partir del total de encuestas registradas, no de participantes.
Reporte de evaluación	Documento que genera un reporte estadístico tomando en cuenta todas las evaluaciones realizadas en el departamento dentro de un periodo dado. Contendrá la información de cada actividad, así como los instructores, evaluaciones y comentarios de los participantes correspondientes a sus actividades.

Historial de profesor	Documento que enlista todas las actividades que un profesor ha acreditado, con el número total de actividades y las horas totales que ha tomado. Por cada actividad se enlista su periodo y su duración.
Libro de folios	Documento que enlista en una tabla los folios de constancias y reconocimientos existentes en la BD asociados a su actividad y participantes o instructores.
Reporte general	Documento que enlista todas las actividades en un periodo, anotando sus fechas, sede, etc.
Reporte de sugerencias	Documento que enlista todas las sugerencias de evaluaciones en un periodo dado.
Publicidad	Documento que genera una hoja para la publicidad de la actividad, contiene los objetivos, antecedentes, instructores, horarios, costos, etc. de una actividad.
Constancias	Documento que genera la constancia de un participante por haber acreditado el curso.
Reconocimientos	Documento que genera un reconocimiento a los instructores.
Reporte de evaluación de actividad	Documento que genera un reporte estadístico tomando en cuenta todas las evaluaciones realizadas de la actividad.

Verificación de datos	Documento que enlista a todos los participantes con su información de contacto y un recuadro blanco para marcar confirmación de datos.
Hoja de asistencia	Documento que genera una lista de asistencia para los instructores de la actividad, denotando el nombre de cada participante y datos de la actividad como la fecha, la sede y el horario.
Reporte de evaluación de instructores	Documento que genera un reporte estadístico tomando en cuenta todas las evaluaciones realizadas en la coordinación en un periodo dado.
Diplomas	Documentos que se generan por participantes que hayan acreditado un diplomado.
Exportación de Excel	Documento que enlista en una tabla la mayor parte de información de las actividades en la BD.

2.4 Delimitación, definición y descripción de las entidades y sus atributos

2.4.1 ADMINISTRADOR (ADMINISTRATOR*)*

Persona que posee un cargo en concreto dentro del organigrama de la organización de nuestro caso de estudio (diferente de un profesor y sus puestos de trabajo). Son los usuarios del sistema. Para nuestro caso de estudio únicamente existen jefes de departamento, cuentas de soporte o mantenimiento que no se asocian a un departamento y cuentas de trabajadores asociados a un departamento.

Tabla 2. Entidad Administrador y sus atributos

Atributo	Descripción	Restricción
NOMBRE *(NAME)*	El nombre del administrador, por ejemplo, Jaime.	OBLIGATORIO
APELLIDO PATERNO *(LAST NAME)*	El apellido paterno del administrador, por ejemplo, Ruiz.	OBLIGATORIO
APELLIDO MATERNO *(MOTHER'S LAST NAME)*	El apellido materno del administrador, por ejemplo, Mateo.	NO OBLIGATORIO

ABREVIATURA DE GRADO *(DEGREE)*	La abreviación del grado de estudios del profesor, por ejemplo, Lic., M.I., etc.	NO OBLIGATORIO
GÉNERO *(GENDER)*	La identidad de género del administrador. Sus valores son únicamente M (masculino) o F (femenino).	NO OBLIGATORIO
USUARIO *(USERNAME)*	Nombre de usuario con el cual el administrador entrará al sistema.	OBLIGATORIO
CONTRASEÑA *(PASSWORD)*	Contraseña de acceso al sistema.	OBLIGATORIO
ADMINISTRADOR *(ADMIN)*	Valor para indicar si el administrador posee privilegios en el sistema.	OBLIGATORIO

2.4.2 DEPARTAMENTO (DEPARTMENT)

Es una parte de nuestra organización del caso de estudio encargada de un área en concreto, por ejemplo: "El departamento de recursos humanos". A ella se vinculan administradores ya sea jefes de departamento o trabajadores, así como catálogos de actividades. De esta forma se generan reportes para analizar sus estadísticas, y también se facilita la filtración de contenido en el sistema.

Tabla 3. Entidad Departamento y sus atributos

Atributo	Descripción	Restricción
NOMBRE *(NAME)*	El nombre de la coordinación, por ejemplo, Coordinación Didáctico Pedagógica.	OBLIGATORIO
ABREVIACIÓN (*ABBREVIATION)*	La abreviación del nombre de la coordinación, por ejemplo C.O.	OBLIGATORIO

2.4.3 DIVISIÓN (DIVISION)

Este órgano es fundamental de la organización de nuestro caso de estudio que tiene como funciones sustantivas la docencia, la generación del conocimiento y su difusión, acerca de ciertas ramas de una profesión, por ejemplo, en caso de la Facultad de Ingeniería de la UNAM la DIMEI o la DIE. Se almacenan en el sistema para vincularse a un profesor, indicando que éste labora en dicha(s) división(es), éstas podrían incluir otras funciones administrativas como Becarios y Secretarías.

Tabla 4. Entidad División y sus atributos

Atributo	**Descripción**	**Restricción**
NOMBRE *(NAME)*	Nombre de la división, por ejemplo, División de Ingeniería Eléctrica.	OBLIGATORIO
ABREVIACIÓN (*ABBREVIATION)*	La abreviación del nombre de la división, como por ejemplo D.I.E.	OBLIGATORIO

2.4.4 PUESTO DE TRABAJO (WORK POSITION)

Refiere al trabajo que realiza un profesor dentro de la organización de nuestro caso de estudio. Por ejemplo, la categoría y nivel de un académico dentro de la Facultad de Ingeniería de la U.N.A.M., como Ayudante de Profesor A. Estas posiciones de trabajo se vinculan a un profesor, el cual puede tener más de una, no es necesario que un profesor se vincule forzosamente con un puesto de trabajo.

Tabla 5. Entidad Puesto de Trabajo y sus atributos

Atributo	**Descripción**	**Restricción**
NOMBRE *(NAME)*	Nombre de la categoría.	OBLIGATORIO
ABREVIACIÓN (*ABBREVIATION)*	Abreviación del nombre de la categoría para la simplificación del despliegue de la información; por ejemplo, ADJ PROF A.	OBLIGATORIO

2.4.5 PROFESOR (PROFESSOR) MIEMBRO

Académico o profesionista externo o interno a la organización de nuestro caso de estudio que desea tomar o posee la característica de impartir (si se aprueba bajo las decisiones de la organización) las actividades que se desarrollan en la organización. Se indica su proveniencia, puede pertenecer a una o varias divisiones de la organización, así como poseer diversos puestos de trabajo dentro de la misma.

Tabla 6. Entidad Profesor y sus atributos

Atributo	**Descripción**	**Restricción**
NOMBRE *(NAME)*	El o los nombres del profesor.	OBLIGATORIO
APELLIDO PATERNO *(LAST NAME)*	Apellido paterno del profesor.	OBLIGATORIO
APELLIDO MATERNO *(MOTHER'S LAST NAME)*	Apellido materno del profesor.	NO OBLIGATORIO

RFC *(RFC)*	El RFC del profesor, puede ser con homoclave o sin ésta. Su longitud puede ser de 13 o 10 caracteres únicamente y es único.	NO OBLIGATORIO
NÚMERO DE TRABAJADOR *(WORKER NUMBER)*	El número de trabajador del profesor.	NO OBLIGATORIO ES ÚNICO
FECHA DE NACIMIENTO *(BIRTHDATE)*	Fecha de nacimiento del profesor.	NO OBLIGATORIO
NÚMERO DE TELÉFONO *(PHONE NUMBER)*	Número de teléfono o celular del profesor.	NO OBLIGATORIO
GRADO DE ESTUDIOS *(DEGREE)*	Abreviatura del grado de estudios del profesor, por ejemplo: M.I.	NO OBLIGATORIO

CORREO ELECTRÓNICO *(EMAIL)*	Correo electrónico del profesor.	NO OBLIGATORIO
GÉNERO *(GENDER)*	La identidad de género del profesor. Sus valores son únicamente M (masculino) o F (femenino).	NO OBLIGATORIO
SEMBLANZA *(SEMBLANCE)*	Descripción de la semblanza del profesor para formatos informativos.	NO OBLIGATORIO
ES INSTRUCTOR *(IS INSTRUCTOR)*	No todo profesor puede ser instructor. Rubro falso o verdadero de aprobación para poder ser instructor de actividades en nuestra organización.	OBLIGATORIO
PROVENIENCIA *(PROVENANCE)*	Proveniencia del profesor.	NO OBLIGATORIO

2.4.6 CATÁLOGO DE ACTIVIDAD (ACTIVITY CATALOGUE)

Refiere a la actividad que la organización de nuestro caso de estudio realizará para cumplir su objetivo; es decir, un curso, un taller, un seminario, un módulo de diplomado, etc. que tiene como objetivo fundamental capacitar a los profesores registrados en la organización.

Tabla 7. Entidad Catálogo de Actividad y sus atributos

Atributo	**Descripción**	**Restricción**
CLAVE *(KEY)*	Se genera tomando la coordinación a la que pertenece y el tipo de actividad que es, por ejemplo: DICU001. Si hay excepciones, se ingresa y altera de forma manual. La utilidad de este dato radica en los reportes de la organización. NO CONTIENE GUIONES.	OBLIGATORIO
NOMBRE *(NAME)*	El nombre de la actividad.	OBLIGATORIO
CANTIDAD DE HORAS *(HOURS)*	La cantidad numérica de horas que tomará la actividad.	OBLIGATORIO

TIPO *(TYPE)*	El tipo de actividad. Su valor se especifica de acuerdo con las reglas de la organización. Su valor -únicamente para nuestro caso de estudio- puede ser CU (Curso), CT (Curso-Taller), TA (Taller), SE (Seminario), FO (Foro), EV (Evento), DI (Módulo de Diplomado), CO (Conferencia).	OBLIGATORIO
DIRIGIDO A *(AIMED AT)*	Descripción del público al que va dirigida la actividad.	NO OBLIGATORIO
OBJETIVO *(OBJECTIVE)*	La descripción del objetivo de la actividad.	NO OBLIGATORIO
CONTENIDO *(CONTENT)*	La descripción del temario o contenido de la actividad.	NO OBLIGATORIO
ANTECEDENTES *(BACKGROUND)*	La descripción de los antecedentes necesarios para entender la actividad.	NO OBLIGATORIO

FECHA DE CREACIÓN *(CREATION DATE)*	Fecha en la que se registró la actividad al sistema. En formato dd-mm-yyyy.	OBLIGATORIO
MÓDULO *(MODULE)*	Dado que la organización de nuestro caso de estudio realiza diplomados, este atributo indica el número de módulo asignado en caso de que la actividad sea de tipo Módulo de Diplomado. Únicamente es obligatorio en ese caso.	NO OBLIGATORIO

2.4.7 ACTIVIDAD (ACTIVITY)

Una actividad instanciada a partir del catálogo de actividades, que posee una fecha específica en concreto y se asocia a instructores y participantes. Representa el momento en el tiempo de la realización de una actividad que surgió a partir de su catálogo ya definido. Para el caso de estudio tomado, las actividades se realizan durante periodos de seis meses. El periodo de cada actividad consta de un año, su número (1 o 2) y su tipo (semestral o intersemestral). Esto facilita la visualización estadística.

Tabla 8. Entidad Actividad y sus atributos

Atributo	Descripción	Restricción
AÑO *(YEAR)*	Año en el cual se llevará a cabo la actividad.	OBLIGATORIO
NÚMERO DE SEMESTRE *(NUM)*	Número del semestre en el que se impartirá la actividad. Su valor puede ser únicamente 1 o 2.	OBLIGATORIO
TIPO DE SEMESTRE *(TYPE)*	El tipo del semestre en el que se impartirá la actividad. Su valor únicamente puede ser i (intersemestral) o s (semestral).	OBLIGATORIO

FECHA DE INICIO *START (DATE)*	La fecha específica en la que la actividad comienza, en formato dd-mm-yyyy.	OBLIGATORIO
FECHA DE FIN *(END DATE)*	La fecha específica en la que la actividad finaliza, en formato dd-mm-yyyy.	OBLIGATORIO
FECHA MANUAL *(MANUAL DATE)*	La fecha que se usará para distintos formatos generados automáticamente; ésta especifica en forma de cadena la fecha de inicio, la de fin y de forma posible los días y sesiones. Por ejemplo: El 4, 5 y 6 de mayo del 2022, Del 5 de junio al 6 de agosto del 2024, Los lunes, miércoles, y viernes del 7 al 8 de agosto del 2020.	OBLIGATORIO
DÍAS DE LA SEMANA *(DAYS WEEK)*	Los días de la semana en los que se impartirá la actividad. Su valor únicamente puede ser Lunes (L), Martes (M), Miércoles (I), Jueves (J), Viernes (V), Sábado (S) o Domingo (D), y pueden seleccionarse varios, ej: [L,M,I,J,V].	OBLIGATORIO

ACREDITACIÓN *(CREDITS TO CREDIT*, CTC)	La calificación necesaria (si es requerida) para aprobar una actividad. Si hablamos de un Foro o Evento, la calificación no es especificada y los participantes no dependen de ella para acreditar.	NO OBLIGATORIO
COSTO *(COST)*	El costo de inscripción de la actividad. Puede ser cero (gratis) o más.	OBLIGATORIO
CUPO MÁXIMO *(MAX QUOTA)*	El número de participantes máximo que pueden inscribirse a la actividad. Debe ser arriba de cero.	OBLIGATORIO
CUPO MÍNIMO *(MIN QUOTA)*	El número de participantes mínimo que es necesario para impartir la actividad. Debe ser arriba de cero.	OBLIGATORIO
CLAVE DE GRUPO *(KEY)*	Atributo derivado del identificador lógico de la actividad programada separado por un guion de la clave de catálogo. Por ejemplo, tomando la clave de catálogo DICU001 la clave de	

	grupo sería DICU001-12, el doce corresponde al identificador lógico de la actividad programada, siendo de forma práctica el número de actividad programada creada en el sistema.	

2.4.8 SEDE (VENUE)

Indica el lugar físico o virtual en el cual se llevará a cabo una actividad (Evento, Curso, Seminario, etc.)

Tabla 9. Entidad Sede y sus atributos

Atributo	Descripción	Restricción
NOMBRE *(NAME)*	El nombre del lugar. Por ejemplo: Salón 2013.	OBLIGATORIO
CAPACIDAD *(CAPACITY)*	La capacidad máxima que soporta el lugar. Debe ser arriba de cero.	OBLIGATORIO
LOCACIÓN *(LOCATION)*	La ubicación física (edificio, plantel) o virtual (enlace de conferencia virtual). Ejemplo: Edificio A F.I. U.N.A.M.	OBLIGATORIO

2.4.9 DIPLOMADO (DIPLOMA)

Una colección de actividades de tipo módulo de diplomado del Centro de Docencia que otorga un reconocimiento profesional a quien lo acredita. Gracias a él es posible generar diplomas una vez que se acreditaron todos sus módulos para un participante. Un diplomado dura varios períodos, pero abarca el mismo año; por ejemplo, puede poseer 6 módulos programados en 2022-1s 2022-2s 2022-1i y 2022-2i, pero ningún módulo puede programarse con diferente año. A diferencia de una actividad común, los módulos sólo pueden programarse una vez por año; es decir, no pueden ser programados dos módulos del mismo catálogo ni del mismo diplomado con el propósito de impartirlos en el mismo año, pero con diferentes instructores, periodo o sede.

Tabla 10. Entidad Diplomado y sus atributos

Atributo	Descripción	Restricción
NOMBRE *(NAME)*	El nombre del diplomado.	OBLIGATORIO

2.4.10 PARTICIPANTE (PARTICIPANT)

Profesor inscrito a una actividad, siendo partícipe de ésta como alumno.

Tabla 11. Entidad Participante y sus atributos

Atributo	Descripción	Restricción
ASISTIÓ *(ATTENDANCE)*	Valor falso o verdadero que indica si el participante cumplió el mínimo de asistencias de la actividad.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
ACREDITÓ *(ACCREDITED)*	Valor falso o verdadero que indica si el participante cumplió el mínimo de requisitos para acreditar.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
CONFIRMÓ *(CONFIRMATION)*	Valor falso o verdadero que indica si el participante confirmó o no su registro.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.

CANCELÓ *(CANCELED)*	Valor falso o verdadero que indica si el participante canceló su registro.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
DESCUENTO *(DISCOUNT)*	Valor numérico en decimal que indica el porcentaje de descuento para este participante; su valor lo decidirá el usuario final, de acuerdo con la información registrada del profesor que se inscribirá a la actividad como participante. Puede ser del cero al cien por ciento.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
PAGO *(DEPOSIT)*	Monto numérico que indica la cantidad que un participante ha pagado sobre el costo total de la actividad.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
A DESTIEMPO *(MISTIMED)*	Valor booleano que indica si el participante se inscribió una vez que el curso ya dio inicio.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el

		momento de su creación.
ADICIONAL *(ADDITIONAL)*	Valor booleano que indica si el participante se inscribió una vez que el cupo máximo de la actividad ya se había cubierto.	OBLIGATORIO, sin embargo, a nivel de base de datos puede ser nulo en el momento de su creación.
CAUSA DE NO ACREDITACIÓN *(NAC)*	Descripción del por qué un participante no acreditó la actividad.	NO OBLIGATORIO
CALIFICACIÓN *(GRADE)*	Es nula para actividades que no sean Curso, Curso-Taller, Taller o Módulo de Diplomado. Funciona para saber si un participante acreditó o no.	NO OBLIGATORIO
COMENTARIO (*COMMENT)*	Descripción o notas acerca del participante.	NO OBLIGATORIO

FOLIO *(KEY)*	Clave o Folio asociado a una constancia por haber acreditado una actividad. Es el identificador único y principal de la constancia. La organización define su estructura, pero se concatena a una secuencia de tres dígitos.	NO OBLIGATORIO ES ÚNICO

2.4.11 INSTRUCTOR (INSTRUCTOR)

Profesor que impartió una actividad, siendo partícipe de ésta como el ponente de la actividad.

Tabla 12. Entidad Instructor y sus atributos

Atributo	Descripción	Restricción
FOLIO *(KEY)*	Clave o Folio asociado a un reconocimiento por haber impartido una actividad. Es el identificador único y principal del reconocimiento. La organización define su estructura; sin embargo, se concatena a una secuencia de tres dígitos.	NO OBLIGATORIO ES ÚNICO

2.4.12 TEMA DE SEMINARIO (SEMINAR TOPIC)

Tópico asociado a una actividad de tipo seminario que imparte específicamente un instructor.

Tabla 13. Entidad Tema de Seminario y sus atributos

Atributo	Descripción	Restricción
NOMBRE *(NAME)*	Nombre del tema que se impartirá dentro de la actividad tipo seminario.	OBLIGATORIO
HORAS *(HOURS)*	Cantidad de horas que tomará la impartición de dicho tema.	OBLIGATORIO
FECHA DE EXHIBICIÓN *(EX DATE)*	Fecha en la que un instructor impartirá este tema. En formato dd-mm-yyyy.	OBLIGATORIO

2.4.13 EVALUACIÓN INSTRUCTOR (INSTRUCTOR EVALUATION)

Serie de preguntas referentes a un instructor en concreto, que evalúan el desempeño de éste durante una actividad de acuerdo con las respuestas de un participante. Estas preguntas se responden con un valor numérico correspondiente a NULO o 0 (SIN RESPUESTA), 50 (Mala), 60 (Regular), 80 (Buena), 95 (Muy buena), 100 (Excelente).

Tabla 14. Entidad Evaluación Instructor y sus atributos

Atributo	Pregunta	Restricción
PREGUNTA 1 *(QUESTION 1)*	Considero la experiencia del instructor como…	OBLIGATORIO
PREGUNTA 2 *(QUESTION 2)*	La planeación y organización de las sesiones y lecturas de acuerdo con los temas fue…	OBLIGATORIO
PREGUNTA 3 *(QUESTION 3)*	La puntualidad del instructor fue…	OBLIGATORIO

PREGUNTA 4 *(QUESTION 4)*	La forma de utilizar el equipo y materiales de apoyo al curso fue…	OBLIGATORIO
PREGUNTA 5 *(QUESTION 5)*	La manera de aclarar las dudas planteadas por los participantes fue…	OBLIGATORIO
PREGUNTA 6 *(QUESTION 6)*	Las técnicas grupales utilizadas por el (la) instructor(a) fueron…	OBLIGATORIO
PREGUNTA 7 *(QUESTION 7)*	La forma de interesar a los participantes durante el curso fue…	OBLIGATORIO
PREGUNTA 8 *(QUESTION 8)*	La actitud y el trato del (de la) instructor(a) respecto a la comunicación con los participantes fue…	OBLIGATORIO

2.4.14 EVALUACIÓN DE ACTIVIDAD (ACTIVITY EVALUATION)

Serie de preguntas referentes a una actividad en concreto, que evalúan el desempeño de ésta de acuerdo con las respuestas de un participante. Estas preguntas se responden con un valor numérico correspondiente a NULO o 0 (SIN RESPUESTA), 50 (Mala), 60 (Regular), 80 (Buena), 95 (Muy buena), 100 (Excelente), exceptuando las últimas, que en su descripción se menciona cómo se registran.

Tabla 15. Entidad Evaluación de Actividad y sus atributos

Atributo	Pregunta	Restricción
PREGUNTA 1 1 *(QUESTION 1 1)*	Las actividades de aprendizaje estuvieron vinculadas a los objetivos y contenidos de manera…	OBLIGATORIO
PREGUNTA 1 2 *(QUESTION 1 2)*	La suficiencia de los contenidos para el logro de los objetivos propuestos fue…	OBLIGATORIO
PREGUNTA 1 3 *(QUESTION 1 3)*	La utilidad del material proporcionado durante el curso fue…	OBLIGATORIO

PREGUNTA 1 4 *(QUESTION 1 4)*	La motivación para el estudio independiente de las sesiones fue…	OBLIGATORIO
PREGUNTA 1 5 *(QUESTION 1 5)*	La aplicación de los temas tratados en mi desarrollo académico es…	OBLIGATORIO
PREGUNTA 2 1 *(QUESTION 2 1)*	Mi puntualidad fue…	OBLIGATORIO
PREGUNTA 2 2 *(QUESTION 2 2)*	Mi participación fue…	OBLIGATORIO
PREGUNTA 2 3 *(QUESTION 2 3)*	Mi actitud durante el curso fue…	OBLIGATORIO

PREGUNTA 2 4 *(QUESTION 2 4)*	La forma en la que aprovecharé este curso será…	OBLIGATORIO
PREGUNTA 3 1 *(QUESTION 3 1)*	La coordinación del curso desde su difusión, inscripción, hasta el cierre fue…	OBLIGATORIO
PREGUNTA 3 2 *(QUESTION 3 2)*	La calidad del servicio en cuanto a trato personal fue…	OBLIGATORIO
PREGUNTA 3 3 *(QUESTION 3 3)*	La calidad del servicio en cuanto a instalaciones, ventilación, iluminación, mobiliario y equipo fue…	OBLIGATORIO
PREGUNTA 3 4 *(QUESTION 3 4)*	La limpieza, el orden y acústica de las instalaciones fue…	OBLIGATORIO

PREGUNTA 4 *(QUESTION 4)*	¿RECOMENDARÍA EL CURSO A OTROS PROFESORES? Opción múltiple: falso o verdadero	OBLIGATORIO
PREGUNTA 5 *(QUESTION 5)*	¿CÓMO SE ENTERÓ DEL CURSO? Selección múltiple: I (Internet), P (Publicidad), J (Jefes de la división) u O (Otro). Además pueden ser varias opciones. Ejemplo: [I,J,O].	OBLIGATORIO
PREGUNTA 6 1 *(QUESTION 6 1)*	Lo mejor del curso fue… Pregunta abierta.	NO OBLIGATORIO
PREGUNTA 6 2 *(QUESTION 6 2)*	Sugerencias y recomendaciones. Pregunta abierta.	NO OBLIGATORIO
PREGUNTA 6 3 *(QUESTION 6 3)*	¿Qué otros cursos, talleres, seminarios o temáticas le gustaría que se consideraran para próximas actividades? Pregunta abierta.	NO OBLIGATORIO

PREGUNTA 7 1 *(QUESTION 7 1)*	Área de conocimiento. Selección múltiple: P (Pedagogía), H (Desarrollo humano), C (Cómputo), O (Otro). Ejemplo: [H, O].	OBLIGATORIO
PREGUNTA 7 2 *(QUESTION 7 2)*	Temáticas. Pregunta abierta.	NO OBLIGATORIO
PREGUNTA 8 1 *(QUESTION 8 1)*	¿En qué horarios semestrales le gustaría que se impartiesen los cursos, talleres, seminarios o diplomados? Pregunta abierta.	NO OBLIGATORIO
PREGUNTA 8 2 *(QUESTION 8 2)*	¿En qué horarios intersemestrales le gustaría que se impartiesen los cursos, talleres, seminarios o diplomados? Pregunta abierta.	NO OBLIGATORIO

2.5 Desarrollo del primer maquetado de la interfaz de usuario

El maquetado de este sistema se realizó en *AdobeXD* el cual se encuentra en el siguiente link para visualizarlo: https://xd.adobe.com/view/c6d232ee-65f4-4e10-a6b7-cc52f72a7011-9381/?fullscreen

Ilustración 3. Propuesta inicio del sistema

Al inicio del sistema, se presentarán dos opciones:

- *Contestar encuesta*: A los participantes inscritos a las actividades de la organización se les pedirá el correo electrónico con el cual se registraron, así como la clave de grupo que les proporcionará el instructor de la actividad.

Ilustración 4. Propuesta Inicio de Sesión

El sistema lo redirigirá a la encuesta correspondiente a su inscripción y evaluará la actividad y al/los instructores relacionados con ella.

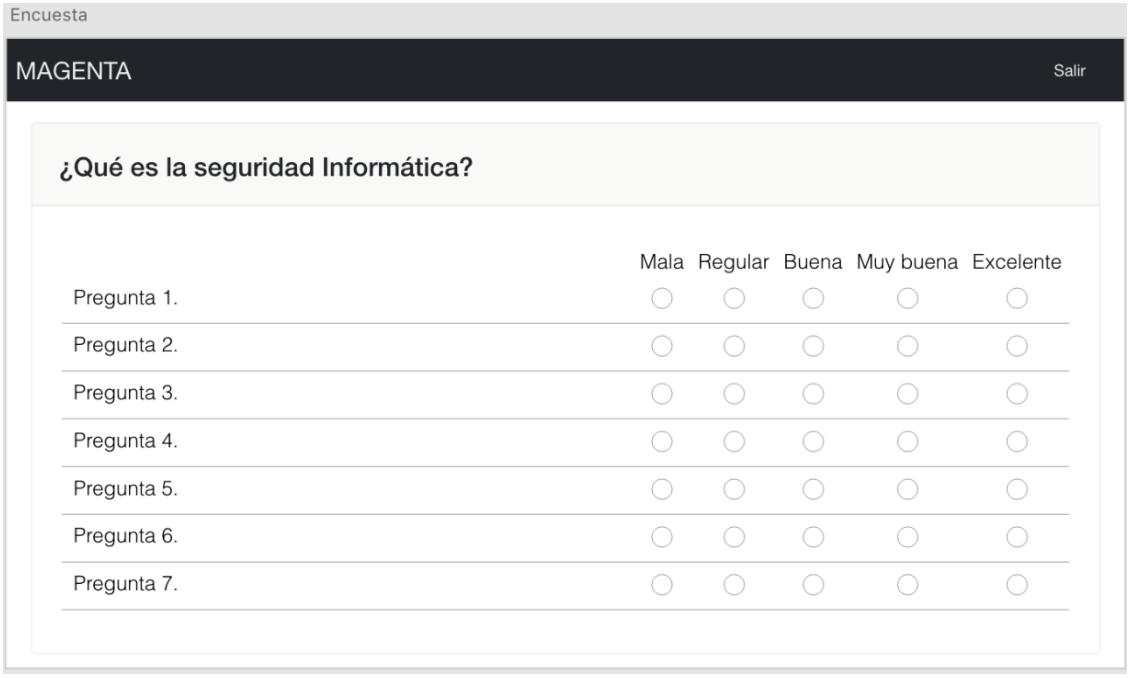


Ilustración 5. Propuesta Formulario de Evaluación

- *Ingresar*: A las personas de la organización autorizadas en el sistema se pedirá usuario y contraseña proporcionados por los administradores.

El sistema lo redirigirá a la siguiente vista donde se muestra un menú de todas las opciones disponibles y al centro se encuentran los accesos rápidos que más se utilizarán en MAGENTA.

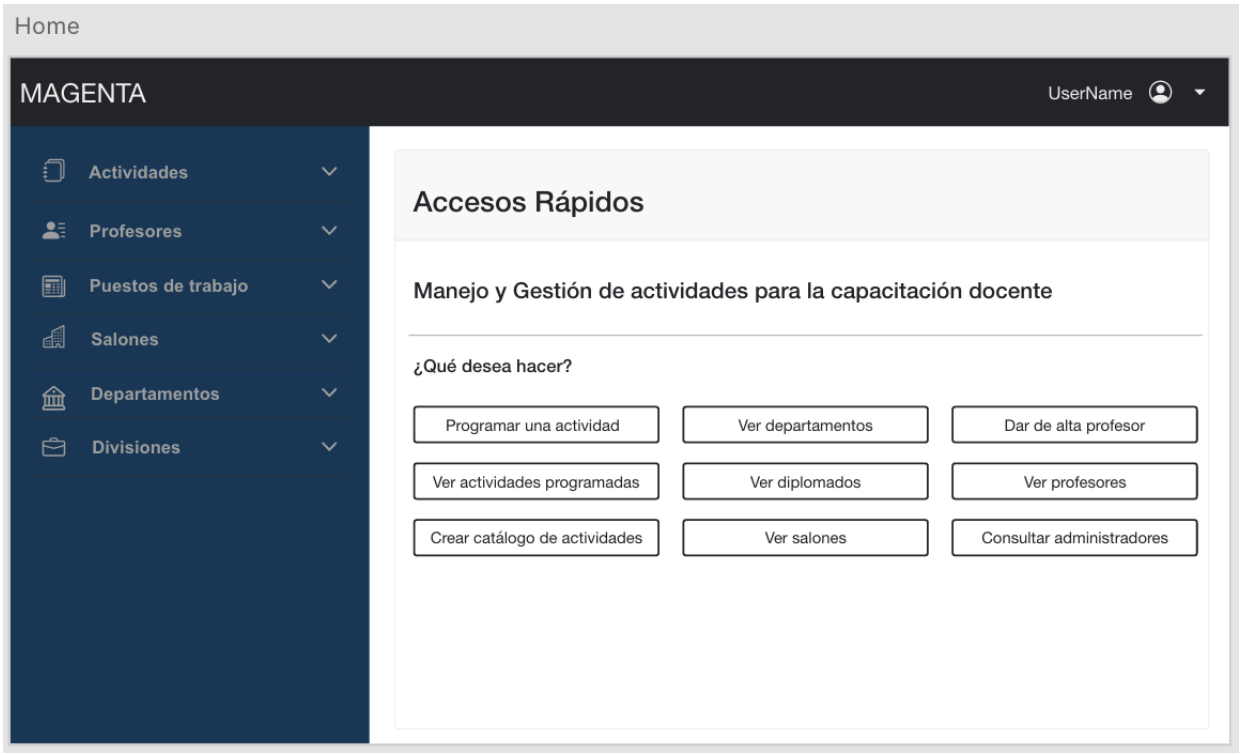


Ilustración 6. Propuesta de vista principal

En la mayoría de las secciones habrá una vista para dar de alta datos relacionados con la opción seleccionada.

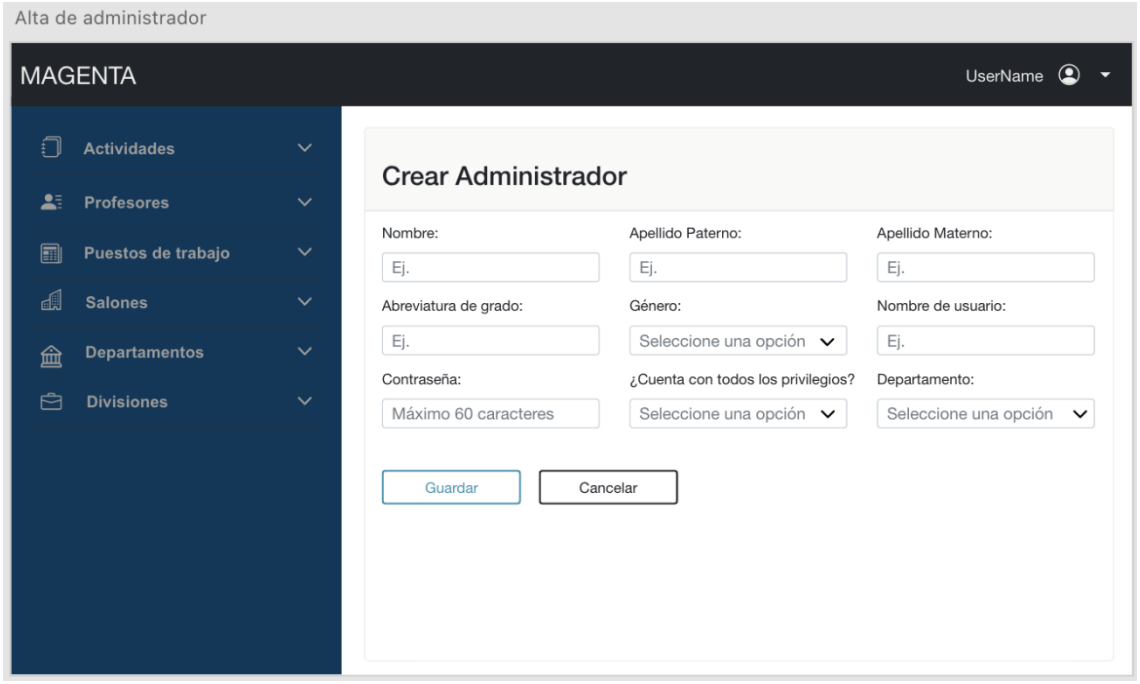


Ilustración 7. Propuesta vista Crear Administrador

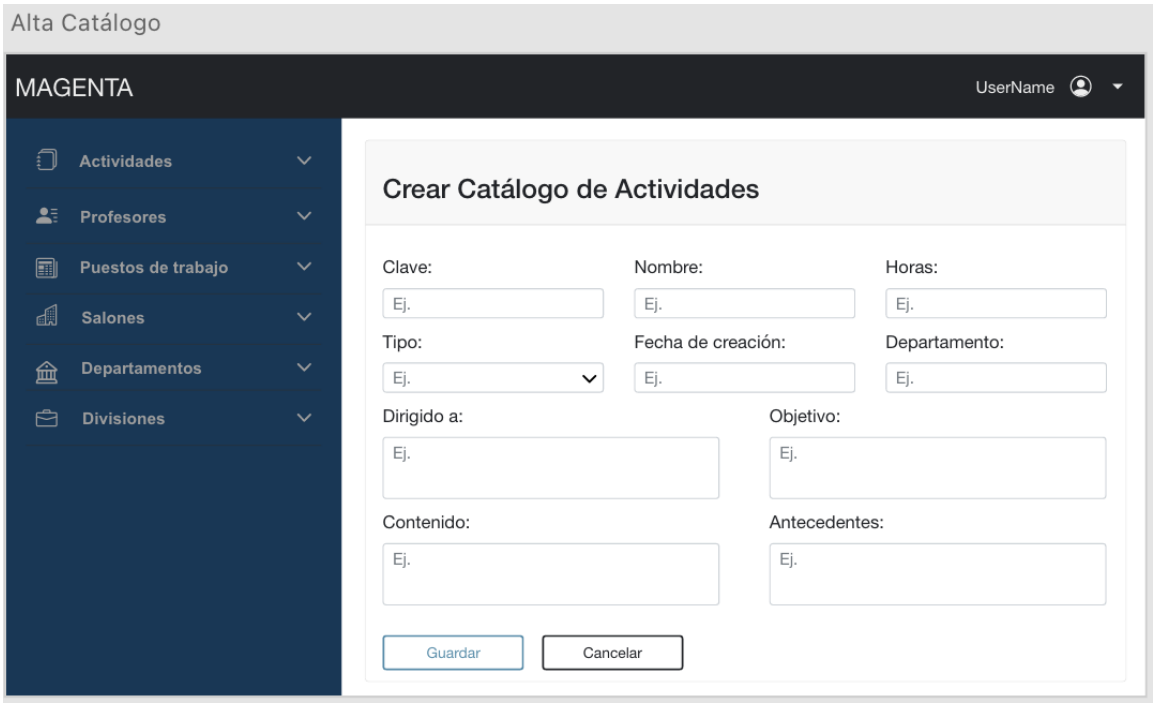


Ilustración 8. Propuesta vista Crear Catálogo de Actividades

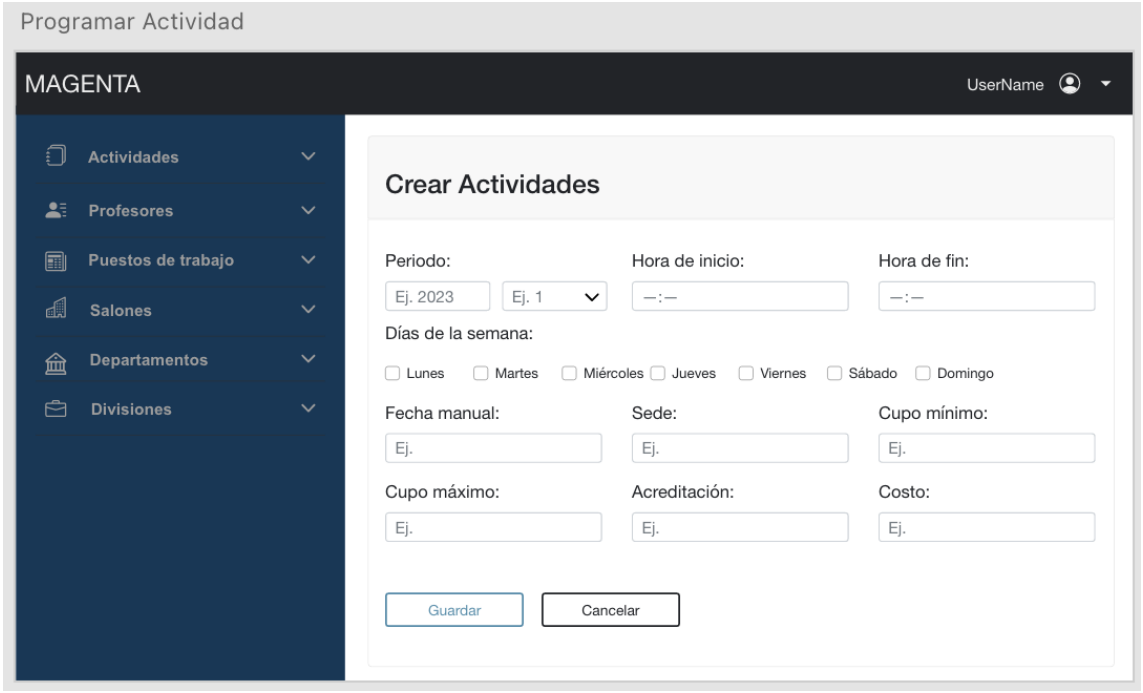


Ilustración 9. Propuesta vista Crear Actividades

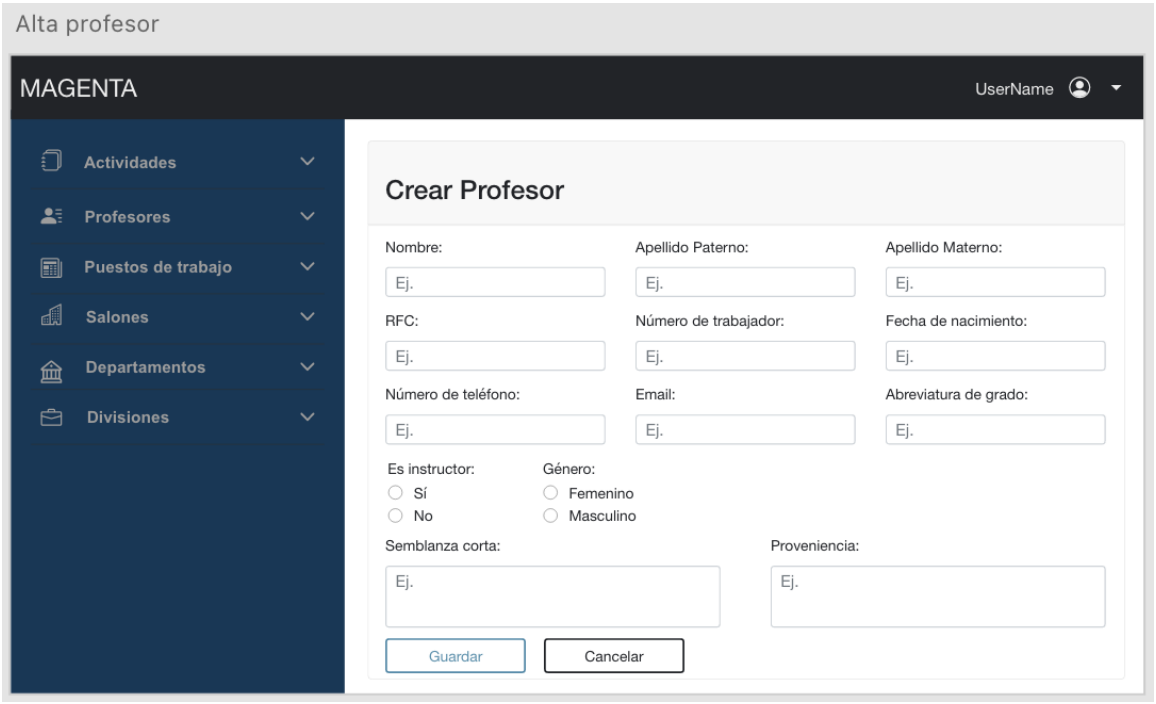


Ilustración 10. Propuesta vista Crear Profesor

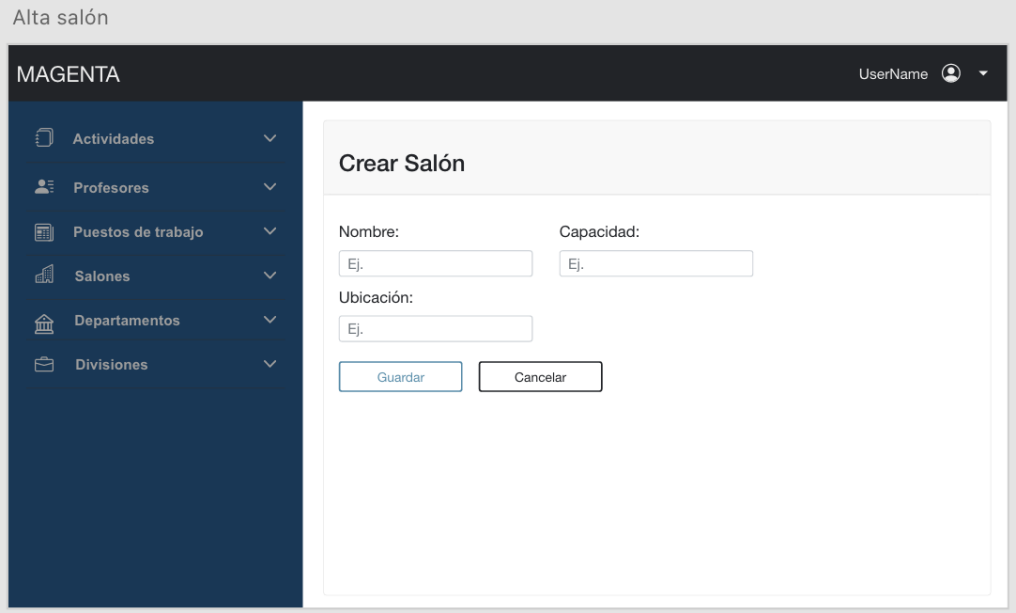


Ilustración 11. Propuesta vista Crear Salón

De igual manera, existirá otra vista para que los participantes e instructores que deseen inscribirse a una actividad, lo hagan sin necesidad de repetir su información, en virtud de haber sido proporcionada previamente al darse de alta.

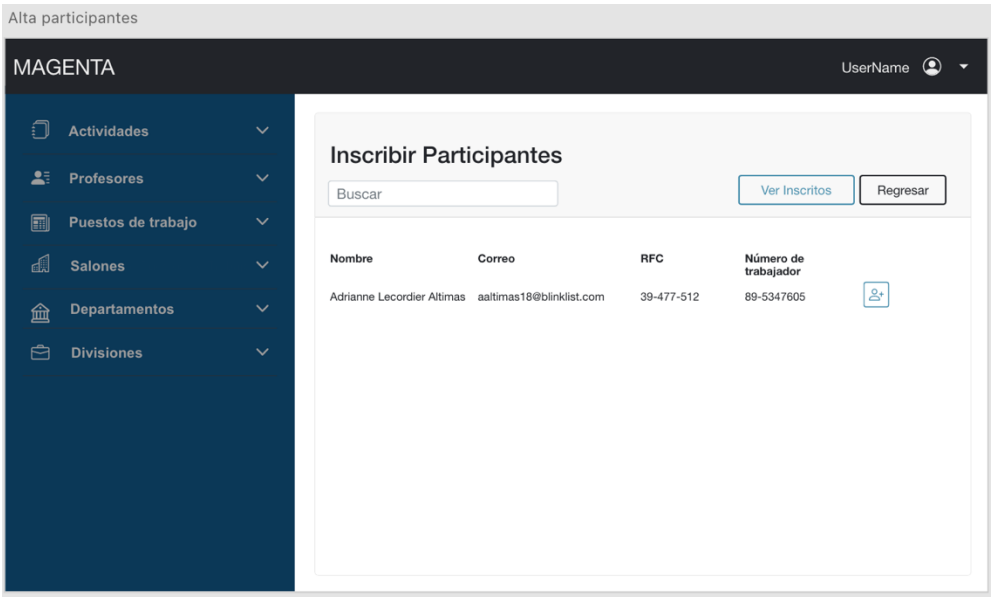


Ilustración 12. Propuesta vista Inscribir Participantes

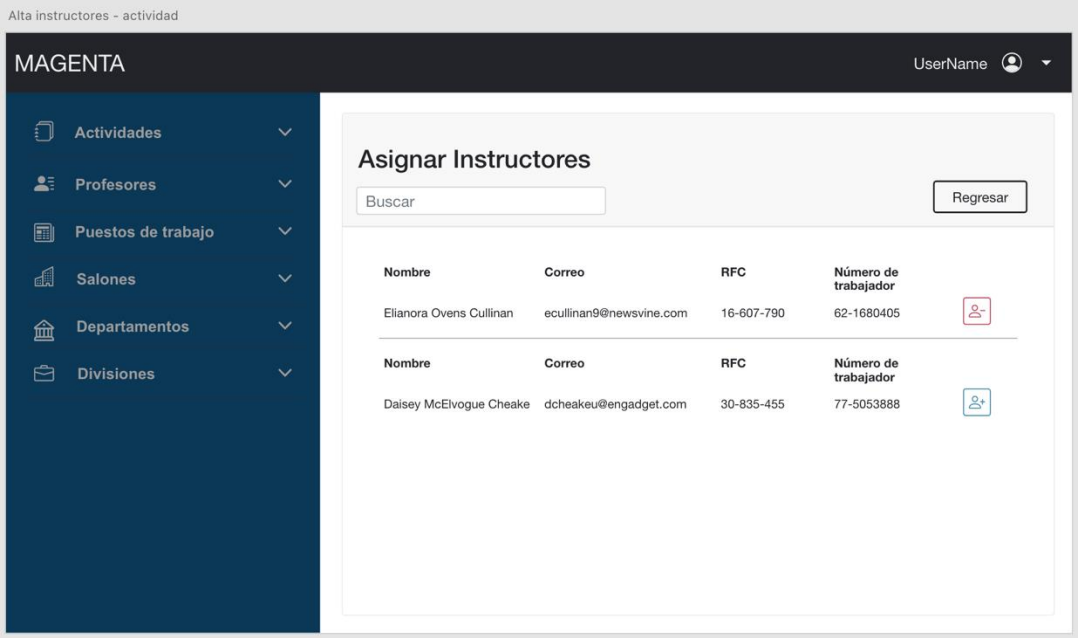


Ilustración 13. Propuesta vista Asignar Instructores

Habrá algunas secciones donde se pidan pocos datos, por ello aparecerán en la misma vista donde se muestran los demás registros.

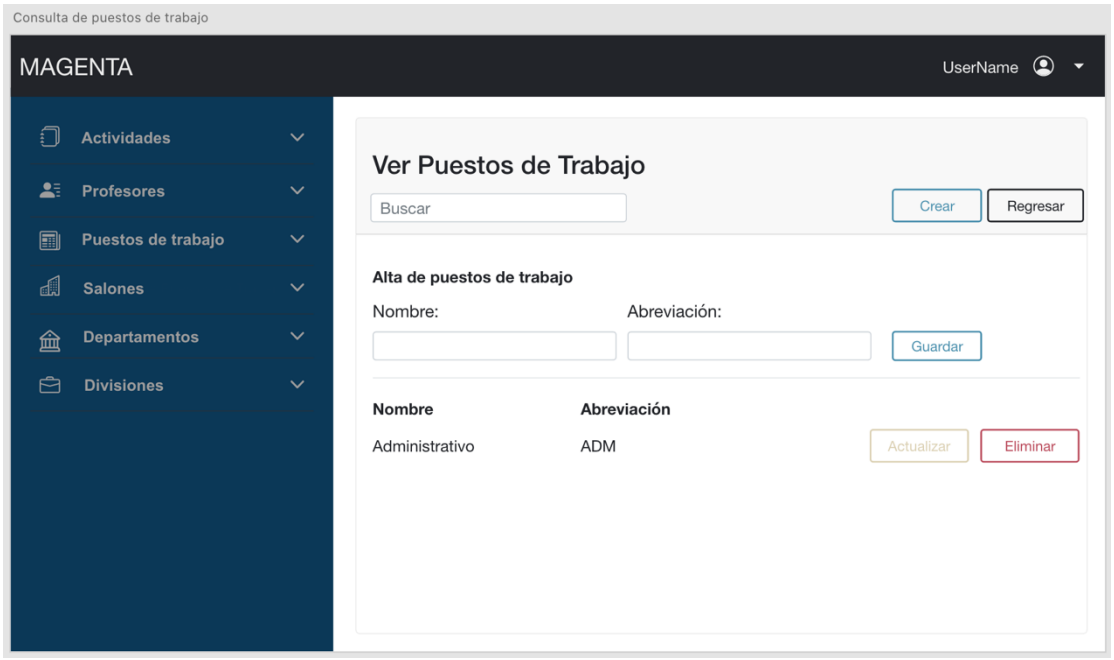


Ilustración 14. Propuesta vista Ver Puestos de Trabajo

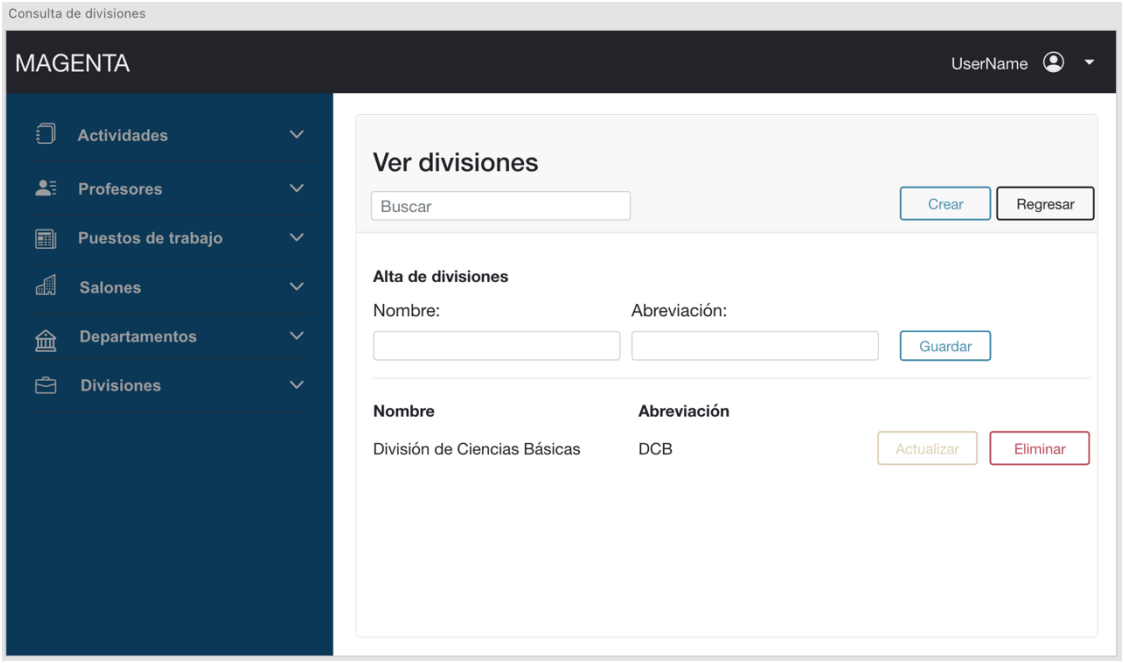


Ilustración 15. Propuesta vista Ver Divisiones

De igual forma se contará con las vistas para actualizar la información, donde retornarán los datos que sea necesario modificar.

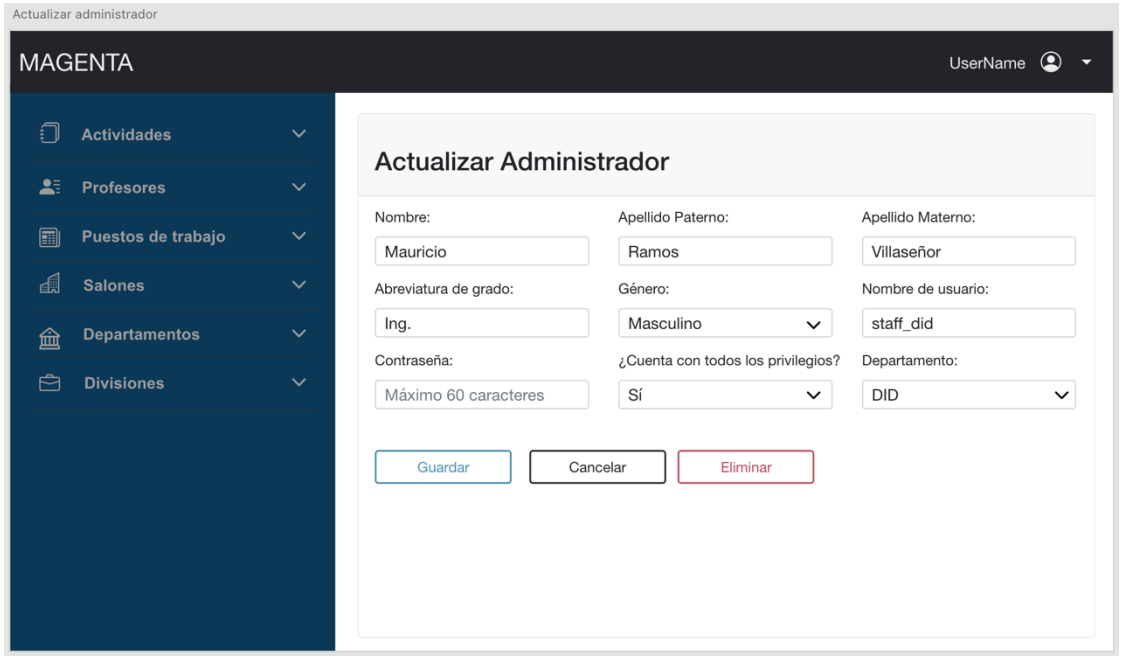


Ilustración 16. Propuesta vista Actualizar Administrador

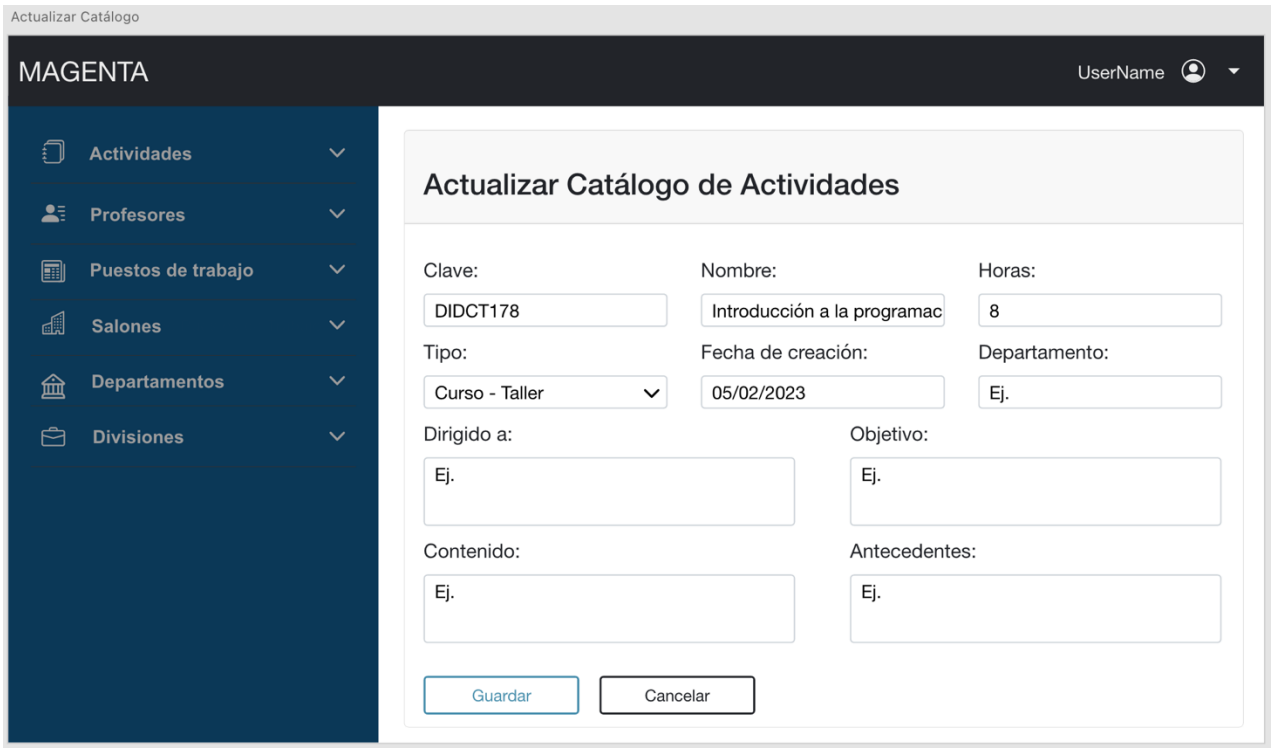


Ilustración 17. Propuesta vista Actualizar Catálogo de Actividades

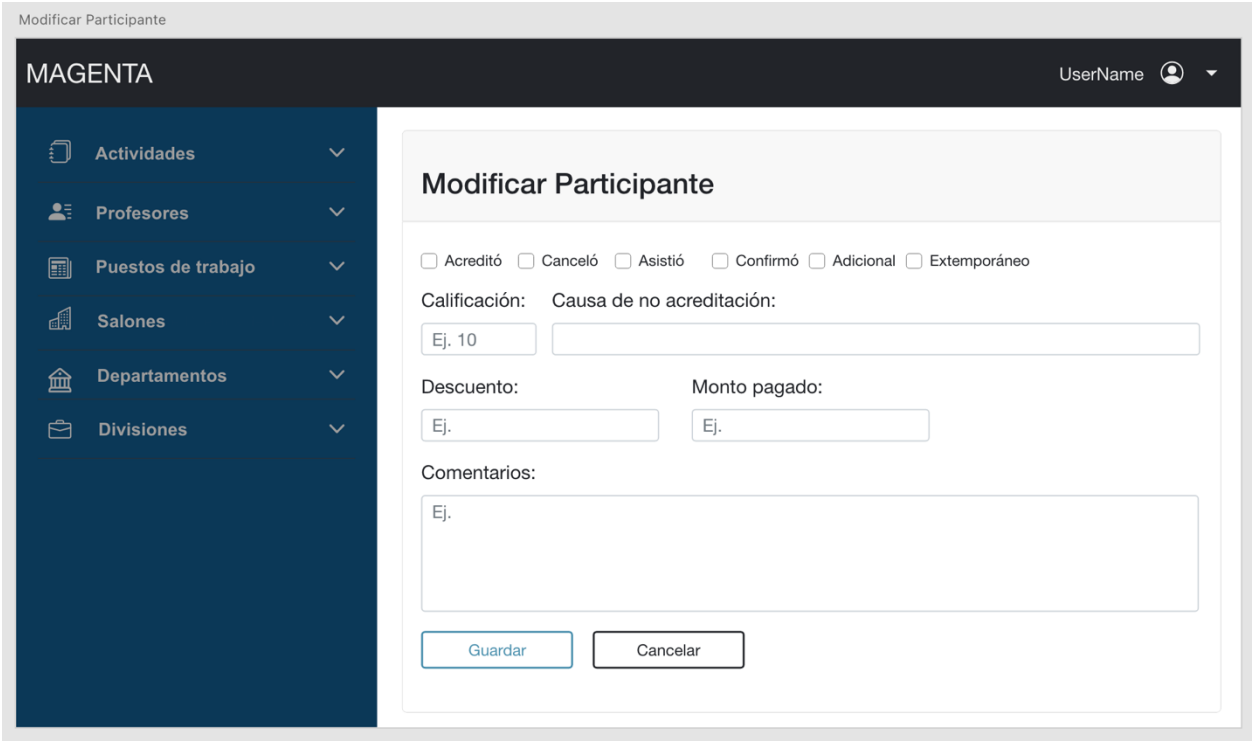


Ilustración 18. Propuesta vista Modificar Participante

De igual forma, en caso de necesitar actualizar secciones donde haya poca información, se abrirá un cuadro en medio de la ventana llamado *Modal.*

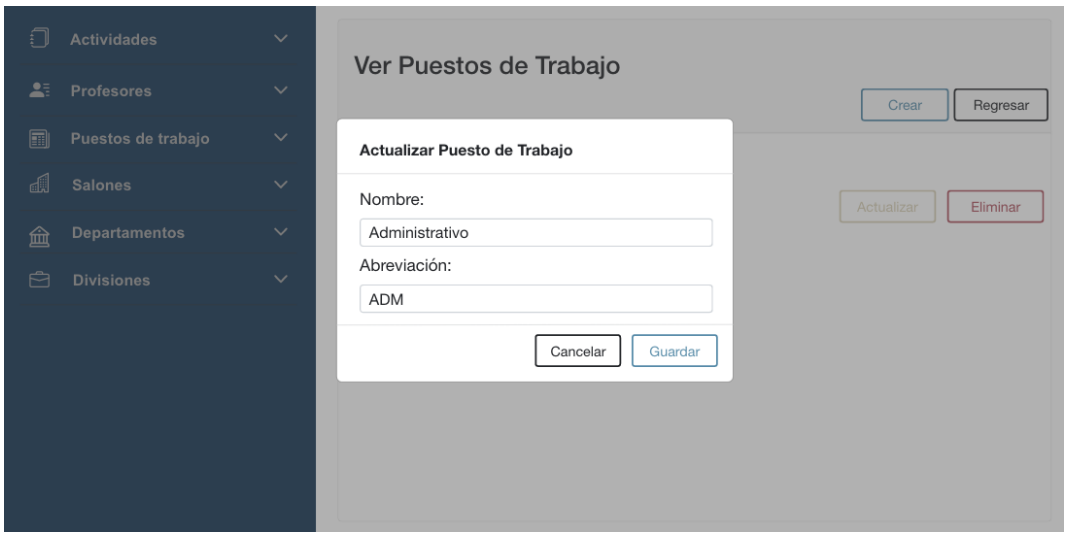


Ilustración 19. Propuesta vista Actualizar Puesto de Trabajo

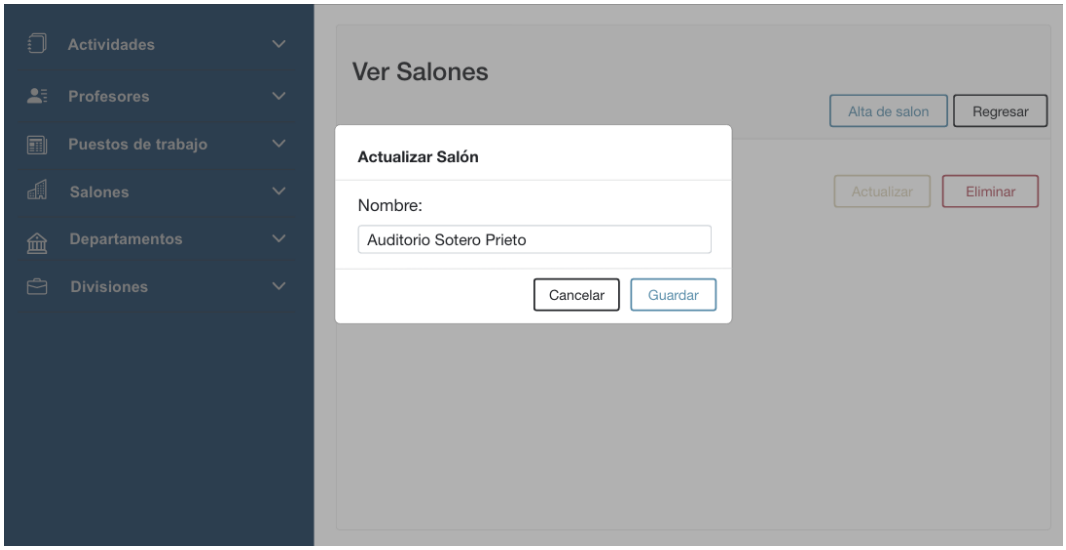


Ilustración 20. Propuesta vista Actualizar Salón

Asimismo, en caso de intentar eliminar algún otro dato de cualquiera de las secciones, aparecerá un *Modal* solicitando la confirmación.

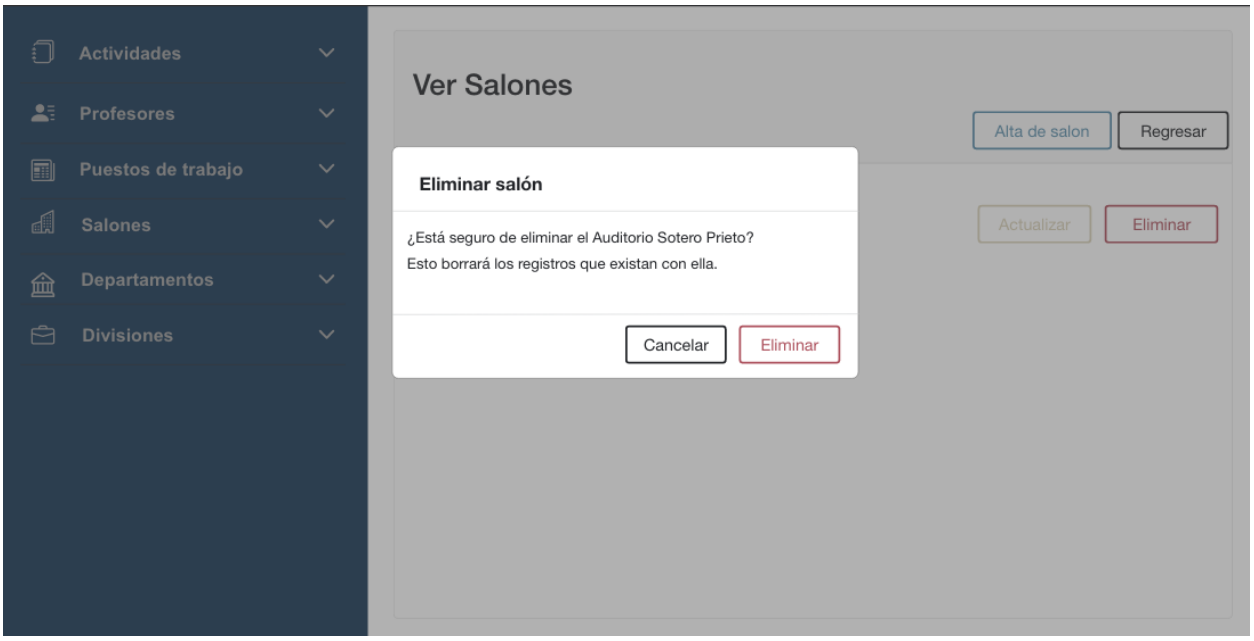


Ilustración 21. Propuesta vista Eliminar Salón

Como podemos observar, para nuestro caso de estudio se realizaron varias reuniones con el cliente, a fin de obtener con precisión cada uno de los requerimientos para hacer del proyecto un sistema eficiente y funcional, que permita gestionar fácilmente los registros de los participantes e instructores, así como de las actividades a realizar.

Capítulo 3. Análisis de las herramientas y metodologías necesarias para el desarrollo.

3.1 ¿Qué es un Framework? ¿Por qué es necesario? ¿Cómo se implementa?

De acuerdo con el diccionario de Cambridge, la palabra *framework* puede definirse como "a supporting structure around which something can be built" (Cambridge Dictionary, s.f.), esto es "una estructura de soporte alrededor de la cual se puede construir algo".

Si nos basamos en la definición anterior, podemos decir entonces que el término técnico hace referencia a una herramienta de software que permite organizar una plantilla y un esquema de trabajo para una aplicación con un propósito específico.

Existen *frameworks* cuyos propósitos pueden orientarse al *marketing* o bien a la edición de video (Open Bootcamp, s.f.) entre muchos otros. En el caso de programación, además, podemos encontrar diversos subtipos, como los que se enlistan a continuación:

- *Para software development:* Proporcionan una base para que los desarrolladores puedan mejorar la estructura de su aplicación.
- *Para desarrollo web:* Existen diferentes bases dependiendo del desarrollador, ya sea frontend, backend o fullstack.
    - Framework frontend: Está compuesto por plantillas y herramientas de HTML, CSS y JavaScript. Ejemplo: Angular, AngularJS, Vue.
    - Framework backend: Proporciona herramientas para trabajar con una base de datos o con el servidor donde se encuentran

almacenadas las páginas web. Ejemplo: Django, Rails, Laravel, CakePHP

- *Para gestión de contenido:* Proporciona herramientas para crear páginas web con plantillas prediseñadas. Ejemplo: WordPress, Joomla, Kentico.

En la actualidad, se pueden utilizar una gran cantidad de frameworks para el desarrollo de aplicaciones web, cuyo uso puede diferir dependiendo del lenguaje. Para un programador el uso de estos esqueletos se vuelve necesario, pues simplifica mucho el trabajo al permitirles tener un proyecto limpio, consistente y en menor tiempo, brindándoles adicionalmente ciertas ventajas, tales como (Open Bootcamp, s.f.), (La Universidad en Internet, 2022):

1. Gracias a la estructura básica que proporcionan, hay tareas que un programador no necesita realizar por sí mismo.
2. Son útiles en el desarrollo de aplicaciones de complejidad alta o media.
3. Mejoran las prácticas de programación, ya que la mayoría de los *frameworks* están basados en patrones de diseño. De igual forma, mejora la organización del código al proporcionar una estructura de dónde situar los diferentes archivos del proyecto. (Technology Consulting, 2018)
4. Su código se puede utilizar en varios proyectos sin tener que programarlo de nuevo.
5. Al saber la estructura básica con la que se está trabajando, facilita mucho el trabajo colaborativo con otros desarrolladores; un *framework* permite organizar y estructurar los archivos y directorios que comprende un proyecto, provocando también una mayor legibilidad y limpieza en el código del proyecto.
6. Tomando en cuenta el punto anterior, si todos los miembros de un equipo trabajan de la misma forma, el mantenimiento es más fácil de realizar.

Algunas de las desventajas que pueden presentarse al trabajar con *frameworks* son las siguientes:

1. El tiempo de aprendizaje es tardado, ya que se debe familiarizar con la estructura de sus archivos y la forma en la que se comunican entre ellos.
2. Como están en constante actualización con el fin de cumplir con las nuevas tecnologías y políticas de seguridad, los *frameworks* pueden sufrir versiones inestables por las incompatibilidades con otras bibliotecas o errores de seguridad.
3. En aplicaciones muy exigentes los *frameworks* pueden resultar poco apropiados, porque consumen más recursos que una aplicación creada desde cero.
4. Si la aplicación es pequeña, estaría ocupando más espacio del que necesita.

En nuestro caso de estudio, era necesario un esquema completo con muchas características y herramientas que nos ayudara a desarrollar el proyecto. Por este motivo, se decidió utilizar el *framework* Laravel y para implementarlo necesitamos lo siguiente:

Instalación de herramientas

a. **PHP**: Es el lenguaje base de Laravel y nos permite crear aplicaciones web dinámicas.
b. **COMPOSER**: Administra las dependencias, es decir, gestiona las bibliotecas y paquetes externos que necesitan instalarse en el proyecto.
c. **PostgreSQL**: Administra y almacena los datos del proyecto.

Configuraciones generales

- Al crear el proyecto, se generará un archivo con extensión **.env** donde se podrán establecer los detalles de la base de datos como nombre, usuario, contraseña; o, si se utiliza algún servicio de correo, también se puede configurar en este archivo.

Finalmente, antes de realizar la implementación de cualquier diseño, se deben tomar en cuenta las características de los equipos donde correrán dichos proyectos para no tener inconvenientes al utilizarlos.

3.2 Análisis y elección del patrón de diseño a implementar

3.2.1 Patrones de diseño

Los patrones de diseño surgieron de la necesidad de resolver problemas habituales en la programación orientada a objetos. Fue en el año 1995 que Erich Gamma, John Vlissides, Ralph Johnson y Richard Helm (Shvets, Historia de los patrones, 2023) retomaron la idea de Christopher Alexander para aplicar y describir los patrones de diseño en la programación. Al inicio se presentaban 23 modelos, pero con el tiempo se descubrieron decenas de ellos.

Se les define entonces como una especie de planos prefabricados que se pueden adecuar con el fin de resolver un problema de diseño en el código. Éstos no funcionan como algoritmos o librerías, sino como una descripción general que solventa problemáticas en particular, además de permitir observar los resultados a los que se debe llegar.

Existen patrones de diseño que sólo se aplican a un único lenguaje de programación conocidos como *idioms*; otros que son más universales aplicados en cualquier lenguaje y, finalmente, los clasificados por su propósito que se dividen en 3 grupos:

1. **Patrones creacionales**: Definen cómo crear un objeto. (Junta de Andalucía, s.f.) Ejemplos:
    a. *Factory Method:* Crea una base del objeto en la superclase y permite que las demás subclases lo modifiquen de acuerdo con las necesidades.
    b. *Builder Pattern:* Permite la construcción gradual de objetos complejos para generar diversas representaciones de éste utilizando el mismo código de construcción.
    c. *Singleton:* Garantiza la existencia de una única instancia y no permite la creación de otras, asegurando que todos los componentes tengan acceso a ella.
2. **Patrones estructurales**: Explican cómo ensamblar objetos y clases en estructuras más grandes. (Shvets, El catálogo de patrones de diseño, 2023) Ejemplos:
    a. *Facade:* Oculta la complejidad interna del sistema, mientras que el cliente ve la versión simplificada.
    b. *Flyweight:* Optimiza la memoria RAM reutilizando y manteniendo un número mayor de objetos compartidos dentro de los límites disponibles.
    c. *Proxy:* Procesa las solicitudes para realizar algo antes de acceder al objeto original y proseguir con las tareas posteriores.
3. Patrones de comportamiento: Indican las responsabilidades entre objetos. Ejemplos:
    a. *Iterator:* Permite recorrer los elementos de una colección, como una lista, una pila o un árbol, y colocarlos en un objeto separado llamado iterador.
    b. *Mediator:* Restringe las comunicaciones entre objetos forzándolos a pasar mediante un objeto mediador.

c. *Template Method:* Define el esqueleto de una clase, pero el algoritmo de las subclases pueden sobreescribirse sin cambiar la estructura.

Para nuestro diseño, se implementó el patrón *Facade* con el fin de que el cliente accediera a los módulos en forma sencilla.

3.2.2 Patrones de arquitectura

Los patrones de arquitectura son los encargados de definir la estructura que debe de tener el software, así como las partes que se construirán y la forma de juntarlas para trabajar con ellas. En otras palabras, se puede decir que es el responsable del esqueleto y la infraestructura del software (Fullstack Tutorials, s.f.) a desarrollar.

En cierta forma son similares a los patrones de diseño, pero la diferencia radica en que los de arquitectura tienen un alcance más amplio y se centran más en la visión abstracta de la idea, mientras que los de diseño se enfocan en su implementación.

Algunos ejemplos son:

1. *Patrón de capas:* Estructura los programas en diferentes grupos de tareas. Las capas más comunes son: presentación (interfaz de usuario), aplicación (capa de servicio), lógica de negocios (capa de dominio) y capa de acceso de datos (capa de persistencia). (Ccori Huaman, 2018)

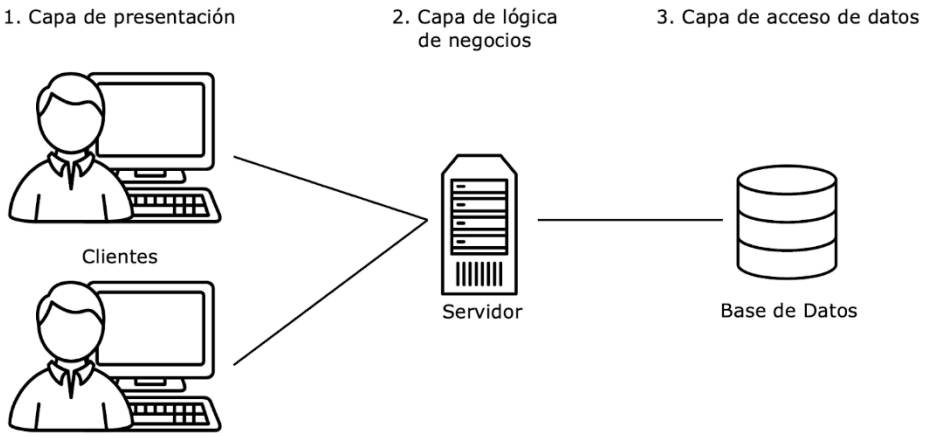


Ilustración 22. Patrón de capas

2. *Patrón de bus de eventos:* Permite detectar eventos y actuar sobre ellos en tiempo real, sin tener que esperar a una respuesta para poder pasar a la siguiente tarea.

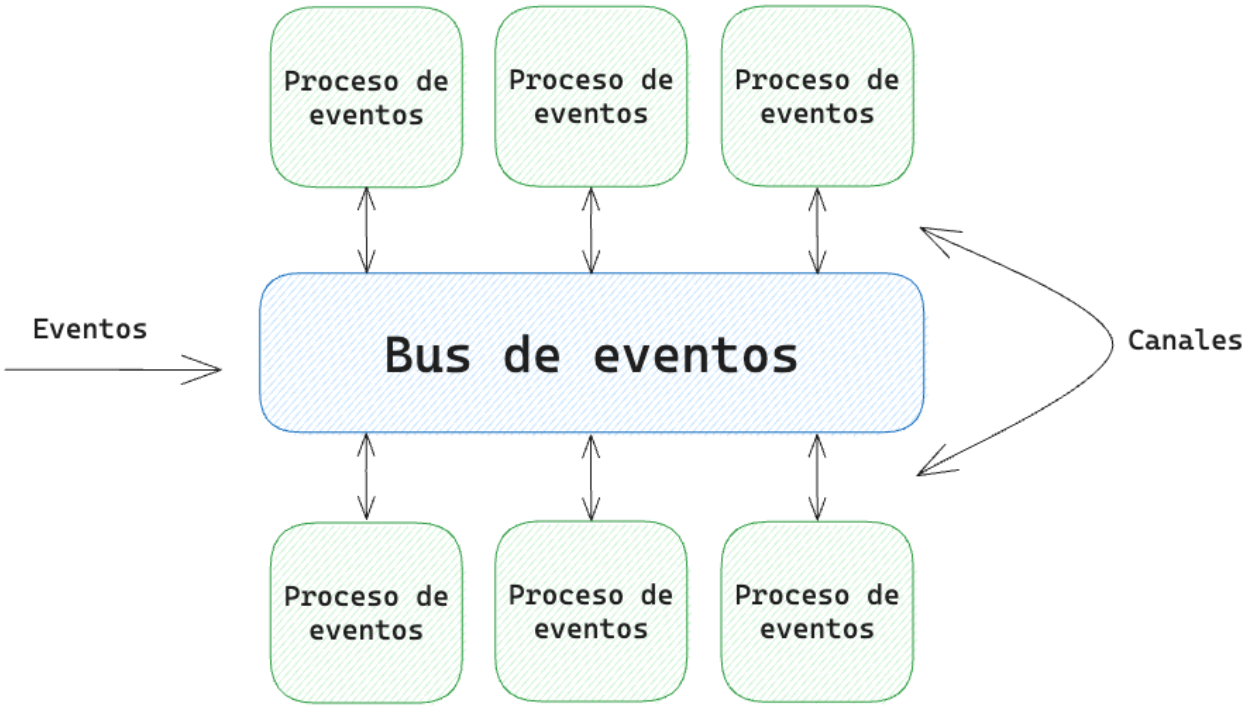


Ilustración 23. Patrón de bus de eventos

3. *Modelo-Vista-Controlador*:

a. Modelo: Representa la lógica de negocio (funcionalidad) y los datos de la aplicación.
b. Vista: Presenta los datos al usuario, es decir, representa la interfaz de usuario.
c. Controlador: Es el intermediario del modelo y la vista. Recibe la entrada del usuario a través de la vista, actualiza el modelo y su interfaz si es que lo requiere.

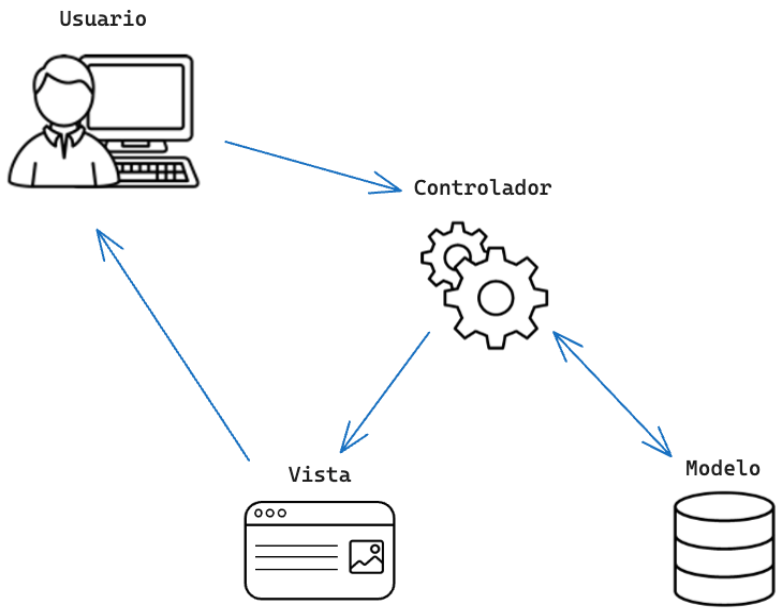


Ilustración 24. Modelo - Vista - Controlador

Se decidió utilizar el patrón Modelo Vista Controlador con el objetivo de crear varias interfaces más allá de un servicio de una sola interfaz.

3.3 Herramientas para el desarrollo de la base de datos

3.3.1 Breve descripción de una base de datos y una base de datos relacional

Podemos denominar el concepto *base de datos* como una colección de archivos almacenada en un espacio físico o lógico. En palabras del Ing. Jorge Alberto Rodríguez Campos, una BD puede ser una "Colección de datos interrelacionados que representan información de interés para un sistema de información y/o usuario final."

Algunos ejemplos de bases de datos pueden ser archivadores llenos de documentos, librerías, libros de Excel o directorios de archivos en un sistema operativo. Por lo tanto, las bases de datos suelen clasificarse por diferentes atributos: de red, por su modelado, lógicas o físicas, relacionales, entre otras.

Una base de datos relacional es aquella que se basa en el modelado de datos relacional creado por E.F. Codd en 1970, éste propone vincular entidades similares a las clases en la programación orientada a objetos, de acuerdo con sus características y la forma en que interactúan entre ellas. Parte de conceptos lógicos y matemáticos, principalmente sobre la teoría de conjuntos. "En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos". (ORACLE, s.f., párrafo primero).

Es importante mencionar esta definición, ya que -para este ejercicio- ocuparemos una base de datos lógica que será almacenada en sistemas computacionales, así como una BD relacional.

3.3.2 Manejadores de bases de datos

Cada vez que se hace referencia a un *Database Management System (DBMS)* hablamos de un software que se encarga de la administración y almacenamiento de los datos dentro del hardware de sistemas computacionales o en unidades de almacenamiento secundario, tales como discos duros y unidades de estado sólido también conocidos como *clusters*.

El software utiliza una serie de herramientas que se encargan de trabajar los datos a un lenguaje de bajo nivel, capaz de volver su acceso y almacenamiento lo más eficiente y rápido posible por medio de data files, archivos dentro de un sistema operativo. Generalmente, los DBMS poseen un software que actúa como motor de almacenamiento y una serie de programas que permiten interactuar con él por medio de CLI (interfaz de línea de comandos). A su vez, algunos DBMS son capaces de utilizar protocolos de comunicación por medio de *Internet Protocol* (IP) para comunicarse a través de una dirección y un puerto. De tal forma que las bases de datos puedan ser accesibles a través de la red. Algunos motores de almacenamiento utilizados por DBMS son *InnoDB* y *Aria*.

3.3.3 Lenguaje SQL

Sus siglas significan *Structured Query Language*. Es un lenguaje diseñado para interactuar con bases de datos relacionales con el propósito de diseñar una estructura de almacenamiento y ejecutar consultas de información.

SQL es un estándar mundial para el cual existen una infinita cantidad de bibliotecas de software capaces de implementarlo para consultar una base de datos a través de diferentes lenguajes de programación. Cada manejador de BD suele amplificar el lenguaje estándar SQL para transformarlo en un lenguaje de procedimientos *"Procedural language"* que permite escribir componentes

tales como disparadores *"triggers"*, procedimientos almacenados o funciones. Algunos ejemplos de estos lenguajes *son Transact-SQL* desarrollado por Microsoft y *PL/SQL* desarrollado por Oracle.

A su vez, el lenguaje SQL, para muchos profesionales, se puede factorizar en cinco sublenguajes que se encargan de clasificar sus principales operaciones:

- **Lenguaje de Definición de Datos:** Crea y maneja o altera los componentes dentro de una base de datos relacional. Las operaciones que utiliza son *CREATE*, *ALTER*, *DROP*, *TRUNCATE*, *RENAME* y *COMMENT*.
- **Lenguaje de Modificación de Datos:** Son las operaciones que involucran propiamente el contenido de los datos almacenados, tales como *INSERT*, *UPDATE*, *DELETE* y *MERGE*.
- **Lenguaje de Consulta de Datos:** Para muchos profesionistas, la instrucción *SELECT* debe considerarse dentro del lenguaje de modificación de datos, mientras que para algunos merece su propia división dada la extensa sintaxis y contextos en los cuales se puede aplicar. Esta instrucción se encarga de mostrar u obtener los datos almacenados.
- **Lenguaje de Control de Datos:** Aquí se encuentran las operaciones *GRANT* y *REVOKE*. Son operaciones que involucran los privilegios de los objetos dentro de una base de datos relacional. Un ejemplo de ello puede ser indicar qué cuentas de usuario tienen acceso de lectura a objetos como una tabla dentro de la base de datos.
- **Lenguaje de Control de Transacciones:** Para que una base de datos respete sus propios principios, disponibilidad e integridad, es necesaria la implementación de un flujo que permita llevar un orden dentro de las diversas conexiones y operaciones en ejecución que tiene una base de

datos relacional. Las instrucciones como *COMMIT*, *ROLLBACK*, *SET TRANSACTION* y *SAVEPOINT* se encargan de ello.

3.3.4 PostgreSQL

Dentro de los manejadores de bases de datos relacionales más comunes se encuentran *MySQL*, *MariaDB*, *Oracle* y *PostgreSQL*. En este trabajo dedicaremos total atención a *PostgreSQL*. Su diseño orientado a objetos amplía el lenguaje *SQL,* combinando diversas características que de forma segura almacenan y escalan las más complicadas cargas de trabajo de datos. Una de las particularidades de *PostgreSQL* es ser de los pocos manejadores de bases de datos de código abierto; es decir, que cualquier persona pudiese modificar, distribuir y utilizar el software a conveniencia.

El proyecto de desarrollo de *PostgreSQL* surgió en 1986, en la Universidad de California en Berkeley y continúa en constante actualización. Se encuentra disponible desde cualquier distribuidor de paquetes en los sistemas operativos basados en Linux más comunes; desde Windows, es posible descargar un instalador certificado desarrollado por EnterpriseDB a través de una página web.

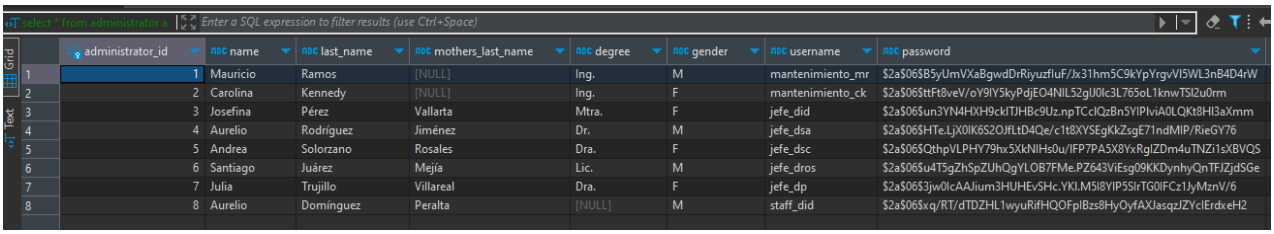
PostgreSQL ofrece, a diferencia de otros manejadores, tipos de datos específicos como *JSON*, *XML*, *Geométricos*, *IPv4* e *IPv6*, entre otros. También dispone de un control de concurrencia a través de puntos de guardado e indexado. Sin duda la mejor de sus características es ofrecer extensiones creadas por usuarios capaces de otorgar funcionalidades para casos de estudio específicos.

En este proyecto, un ejemplo utilizado fue la extensión *unaccent* capaz de funcionar como un diccionario de caracteres que puede transformar caracteres acentuados y especiales por caracteres del alfabeto inglés estándar.

3.3.5 Interfaces gráficas de administración de bases de datos relacionales

Además de utilizar el manejador de bases de datos relacional existen clientes gráficos los cuales son programas conectados al manejador que, por medio de SQL, permiten realizar consultas y visualizarlas de una forma más amigable para el usuario. También permiten la exportación de las consultas a archivos con diversos formatos: *xlsx*, *csv*, *pdf*, *txt*, entre otros.

El cliente más utilizado para trabajar con PostgreSQL es *PGAdmin*; sin embargo, en nuestro proyecto decidimos utilizar *DBeaver Community*, una herramienta multi plataforma para desarrolladores, administradores de bases de datos y analistas de datos. Soporta conexiones a la mayoría de manejadores de bases de datos e incluso es capaz de desarrollar diagramas relacionales para una mejor lectura y análisis de la base de datos.



	administrator_id	name	last_name	mothers_last_name	degree	gender	username	password
1	1	Mauricio	Ramos	[NULL]	Ing.	M	mantenimiento_mr	$2a$06$B5yUmVXaBgwdDrRiyuzfluF/Jx31hm5C9kYpYrgvVI5WL3nB4D4rW
2	2	Carolina	Kennedy	[NULL]	Ing.	F	mantenimiento_ck	$2a$06$ttFt8veV/oY9lY5kyPdjEO4NIL52gIJ0lc3L765oL1knwTSl2u0rm
3	3	Josefina	Pérez	Vallarta	Mtra.	F	jefe_did	$2a$06$un3YN4HXH9ckITJHBc9Uz.npTCclQzBn5YlPlviA0LQKt8HI3aXmm
4	4	Aurelio	Rodríguez	Jiménez	Dr.	M	jefe_dsa	$2a$06$HTe.LjX0IK6S2OJfLtD4Qe/c1t8XYSEgKkZsgE71ndMIP/RieGY76
5	5	Andrea	Solorzano	Rosales	Dra.	F	jefe_dsc	$2a$06$QthpVLPHY79hx5XkNIHs0u/IFP7PA5X8YxRgIZDm4uTNZi1sXBVQS
6	6	Santiago	Juárez	Mejía	Lic.	M	jefe_dros	$2a$06$u4T5gZhSpZUhQgYLOB7FMe.PZ643ViEsg09KKDynhyQnTFJZjdSGe
7	7	Julia	Trujillo	Villareal	Dra.	F	jefe_dp	$2a$06$3jw0lcAAJium3HUHEvSHc.YKIJM5I8YIP5SIrTG0IFCz1JyMznV/6
8	8	Aurelio	Domínguez	Peralta	[NULL]	M	staff_did	$2a$06$xq/RT/dTDZHL1wyuRifHQOFplBzs8HyOyfAXJasqzJZYclErdxeH2

Ilustración 25. Ejemplo de consulta SQL en DBEaver

3.4 Herramientas para el desarrollo del sistema

3.4.1 HTTP

Hypertext Transfer Protocol, mejor conocido por sus siglas HTTP, es un protocolo basado en el modelo cliente-servidor que nos permite intercambiar información en la web; es decir, el cliente envía una solicitud al servidor esperando la respuesta del recurso solicitado, como una página web o un archivo. Generalmente, el cliente puede ser un navegador web (Developer

Mozilla, s.f.), aunque también puede ser un programa-robot que se encargue de explorar y adquirir los datos.

Los mensajes HTTP están estructurados dependiendo de su tipo:

- Peticiones
    - Método: Indica la acción que el cliente quiere realizar. Los más comunes son:
        - *GET:* Solicita a un servidor que envíe un recurso (Lima, 2022)
        - *POST:* Envía datos a un recurso en específico para crear nuevos.
        - *PUT:* Edita y actualiza los datos solicitados (Developer Mozilla, s.f.)
        - *DELETE:* Borra datos en específico
    - URL: Indica la dirección del recurso solicitado.
    - Versión HTTP que se esté usando.
    - Cabeceras opcionales: Pueden aportar información adicional al servidor. (Developer Mozilla, s.f.)
- Respuestas
    - Versión HTTP que se esté usando
    - Código de estado: Indican si la petición fue exitosa o no. Los códigos se agrupan en cinco clases:
        - Informativos (100-199)
        - Satisfactorios (200-299)
        - Redirecciones (300-399)

        - Errores de los clientes (400-499)
        - Errores de los servidores (500-599)
    - Mensaje de estado: Descripción del código de estado. Algunos de los más usados son:
        - Código 200: (OK)
        - Código 201: (CREADO)
        - Código 404: (NO ENCONTRADO)
        - Código 500: (ERROR INTERNO DEL SERVIDOR)
    - Cabeceras HTTP: Pueden aportar información adicional al servidor
    - Recurso solicitado, puede desplegarse en algún formato de texto o un lenguaje de marcado de texto. Ejemplos pueden ser el formato *JSON, XML o HTML*.

Entre cada petición y respuesta encontramos los intermediarios *proxies* que se encargan de gestionar las solicitudes y enviarlas a los servidores correspondientes, así como reenviar la respuesta al cliente. Proxy HTTP tiene varias funciones como:

1. *Caching:* Almacena la caché para solicitudes futuras con el fin de proporcionar la misma información en poco tiempo.

2. *Filtrado*: Puede aplicar restricciones basadas en las direcciones IP, contenido o palabras clave.

3. *Balanceo de carga de peticiones:* Las solicitudes se distribuyen en varios servidores para evitar la sobrecarga, con ayuda de ciertos algoritmos, y no a uno en específico.

4. *Anonimato:* Se aplica cuando se desea conectar a internet anónimamente, por lo tanto, la dirección IP estará bien protegida.

5. *Registro de eventos:* Mantiene el histórico de los eventos que se producen.

Gracias a estas funciones, HTTP ha logrado proporcionar más control y funcionalidad en la web, lo que lo ha convertido en el protocolo estándar de comunicación mundial para desarrollar aplicaciones web, páginas de internet y servicios web.

3.4.2 Lenguajes de marcado HTML y CSS

El lenguaje de etiquetas HTML *"HyperText Markup Language"* es la base fundamental de la mayoría de las páginas web, que generalmente se complementa con las hojas de estilo cascada mejor conocidas por sus siglas en inglés como CSS *"Cascading Style Sheets"*. Ambas se utilizan para el desarrollo de la tecnología *front-end*, pues se emplea HTML para estructurar el contenido de las páginas, mientras que CSS se encarga de darle formato a los elementos. (Escuela de Diseño Madrid, s.f.)

En cuanto a su funcionamiento, HTML utiliza *etiquetas* o *tags* que se colocan entre corchetes (<>), antes y después del elemento a definir para así poder estructurar el texto. Sin embargo, existen etiquetas que no tienen un cierre llamadas *etiquetas vacías.* Por ejemplo *
,<hr>, <img>, <input>, <link>*, entre otras. Se les puede colocar opcionalmente una barra antes del signo > para simular el cierre *(<img/>)*.

Una de las ventajas de la versión HTML5 (ENIUN - Diseño Web y Marketing Digital, 2023) es la incorporación de los elementos semánticos, ya que contribuyen a definirlos mejor y evitan el uso excesivo de * *.
En la siguiente imagen, se muestra la distinción entre la sintaxis de la versión anterior y la actual de HTML. Es evidente que, aunque es posible escribir todo el contenido utilizando solo la etiqueta * * y asignar nombres a través de

clases o identificadores, resulta más legible cuando empleamos las etiquetas apropiadas para cada elemento.

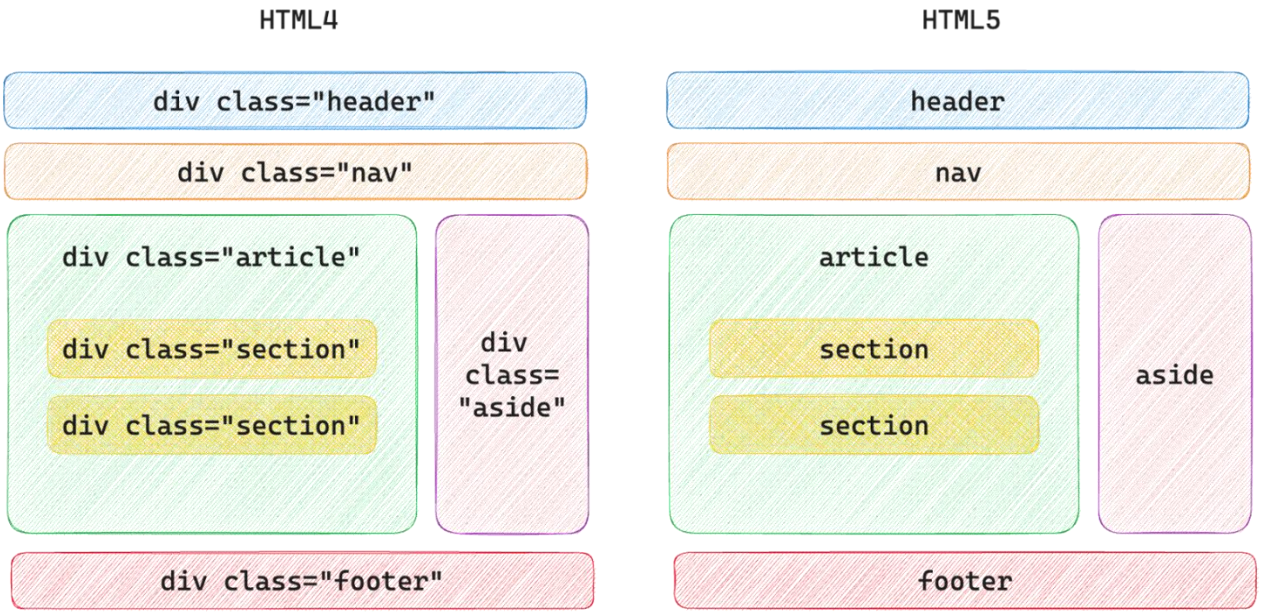


Ilustración 26. Ejemplificación de HTML4 y HTML5

En el anexo A y B están ejemplificadas algunas etiquetas de HTML que sirven para completar el esquema anterior.

Existen alrededor de 112 etiquetas que no sólo se catalogan como estructurales o semánticas. Podemos agruparlas dependiendo de las funciones de cada una, ya sean de texto, de tablas, de listado, de secciones, multimedia, entre otros.

En cuanto a CSS, éste se basa en *reglas* que se aplican sobre los elementos HTML con el propósito de alterar su estilo y se compone de:

- **Selector:** Es el nombre de la clase, identificador o elemento donde se aplicarán los cambios.
- **Propiedad:** Es el atributo que se modificará. Por ejemplo, en un texto puede ser el color o el tipo de fuente.

- **Valor:** Le indica a la propiedad la característica a aplicar. Ejemplo, el color del texto.

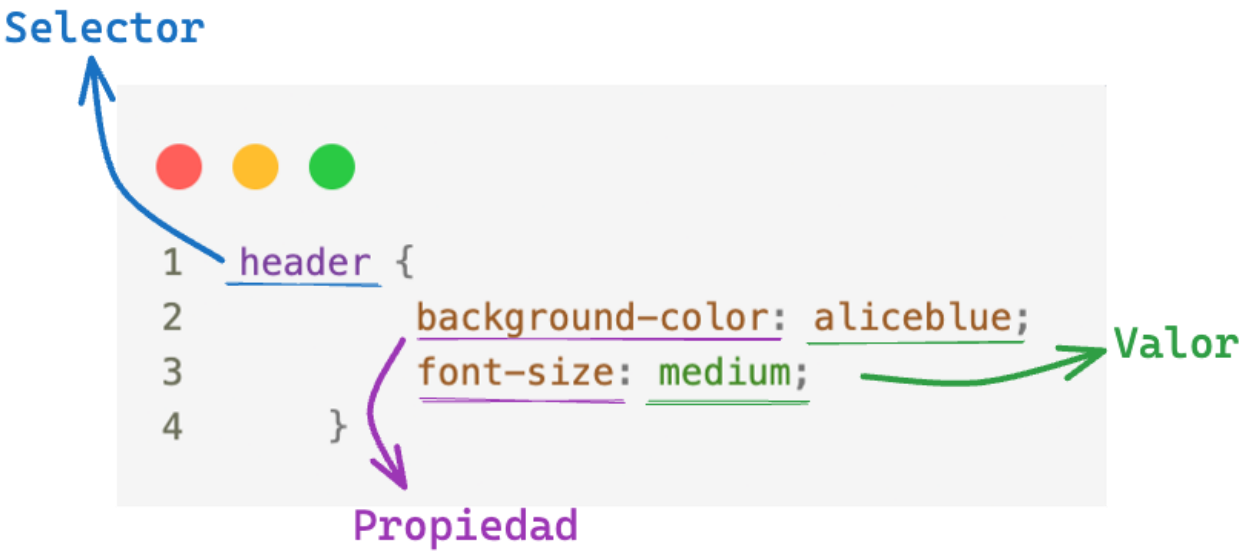


Ilustración 27. Componentes del CSS

En CSS, las propiedades y valores se separan utilizando dos puntos (:); al juntarlas se les llama declaración o regla, con la cual podemos tener varias de ellas en un selector separadas entre sí con punto y coma (;).

Es conveniente mencionar que, cuando se habla de 'Cascada', se refiere al algoritmo que sigue el navegador al aplicar los estilos CSS. Este algoritmo sirve para resolver conflictos donde existan los mismos selectores, pero con diferentes valores. Por esta razón, la cascada revisa los siguientes puntos:

- **Importancia:** Se añade el texto *important* al final de una declaración para que el navegador le dé prioridad a la hora de evaluar los estilos sobre otros.
- **Origen:** Existen 3 orígenes y cada uno con una prioridad específica:
    - Agente de usuario: Estilos que aplica el navegador por defecto.

    - CSS de usuario: Estilos que añade el usuario para personalizar.
    - CSS de autor: Estilos que crea el desarrollador en la página web.
- **Orden de aparición:** Es el orden en el que se procesan los estilos CSS y pueden ser por:
    - Etiqueta HTML <style>
    - Bloques de estilo <style> en el documento HTML
    - Archivo .css externo referenciado con <link rel="stylesheet"> (Manz, La cascada de CSS: Importancia, Especificidad y Orden|, s.f.)
- **Especificidad:** Cuando aún no se elimina la ambigüedad, se calcula el selector más específico:
    - *Selectores de tipo:* Elementos por nombre de nodo (Developer Mozilla, s.f.) (por ejemplo <a>, <h1>) y pseudo-elementos (*::before, ::after*).
    - *Selectores de clase:* Elementos con el atributo *class*, selectores de atributos (type="radio") y pseudo-clases (:hover).
    - *Selectores de ID:* Elementos con el atributo id.
- **Capas:** Los estilos se pueden aislar en capas fusionables.

Con HTML y CSS se pueden crear páginas más presentables, pero si se requiere obtener la atención del usuario y que no se vean estáticas, podemos agregar más dinamismo con JavaScript.

3.4.3 Lenguaje JavaScript

Para realizar páginas web interactivas es necesaria la implementación de un lenguaje de programación, en virtud de que ni HTML ni CSS lo son.

En la actualidad, JavaScript es la opción más utilizada pues mejora la experiencia del usuario, hace más atractivos los sitios web, muestra animaciones, así como cualquier efecto simple o complejo de los componentes.

Este lenguaje de programación es "interpretado"; es decir, que existe un motor de JavaScript, el cual está integrado en todos los navegadores web, que se encarga de ejecutar el código línea por línea. La forma en que el cliente verá su funcionamiento se resume a continuación:

1. Carga la página web solicitada y convierte todos sus elementos en una estructura de datos denominada *Modelo de Objetos de Documento* (Amazon, s.f.) (o por sus siglas en inglés DOM) la cual es una estructura árbol de los elementos que se cargan. Con ella se pueden acceder mediante el objeto *document* para posteriormente modificar sus atributos HTML, cambiando contenido, entre otros. (Manz, ¿Qué es el DOM?, s.f.)

2. El navegador mostrará un nuevo *DOM* cuando diferentes eventos, como hacer un clic a algún elemento, manden a llamar un código de JavaScript.

Una cualidad de este lenguaje es su dinamismo y puede generar contenido al instante tanto del lado del cliente como del servidor, con la diferencia de que el primero sólo puede acceder a los recursos del navegador y el último puede acceder a todos los recursos de la máquina cuando es necesario.

De igual forma, existen bibliotecas que nos pueden facilitar la programación de nuestros proyectos, pues están escritas por otros desarrolladores que nos permiten reutilizar funciones para mejorar el código. Las más utilizadas son

aquellas que sirven para la visualización de datos en forma de gráficos o mapas, para la manipulación de DOM, en su mayoría son animaciones para menús o, bien, galerías de imágenes, entre otros.

Podemos decir que al juntar JavaScript con HTML y CSS se desarrolla una experiencia completa para el usuario, ya que no sólo es funcional, sino que le permite tener más herramientas para actualizar el contenido de forma ágil y rápida.

3.4.4 Lenguaje PHP

PHP son las siglas recursivas en inglés de *Hypertext Preprocessor.* Es un lenguaje de programación que se usa para *incrustar* código en HTML. Se utiliza para desarrollo web, especialmente para la tecnología backend, pues se ejecuta del lado del servidor para después ser enviado al cliente, el cual únicamente recibirá el resultado del *script*.

Con este lenguaje se le pueden añadir nuevas funcionalidades al código HTML para que una página no sea estática. Al igual que JavaScript, le añade dinamismo a la página con la diferencia de que PHP lo hace del lado del servidor, mientras que JS lo hace del cliente.

Este lenguaje tiene tanto ventajas como desventajas; entre las ventajas encontramos la fácil integración con las bases de datos pues es multiplataforma, esto significa que es aceptado por todos los navegadores populares; adicionalmente, brinda seguridad, lo que lo convierte en una opción confiable para evitar ataques informáticos. Aunque es relativamente sencillo de aprender y puede usarse en proyectos simples, también es adecuado para proyectos complejos y permite trabajar con grandes volúmenes de datos, lo que lo hace ideal para sitios web populares.

La desventaja de este lenguaje es que PHP no puede ocultar el código fuente de las páginas desarrolladas, además de tener un rendimiento lento que puede afectar la experiencia de usuario. (Assemble Institute of Technology, s.f.)

3.4.5 BashScripting y PowerShell

Utilizaremos el término *scripting* para definir la acción de realizar un programa que tiene como propósito automatizar las tareas de mantenimiento o administrativas en un sistema operativo que regularmente está operando como un servidor.

Dichos programas están construidos con lenguajes diseñados para ejecutarse a través de una *CLI* o interfaz de línea de comandos. En nuestro proyecto implementamos el *scripting* para automatizar la configuración de nuestro servidor y su base de datos, así como la carga de información masiva. Las *CLIs* que utilizamos son *Zsh* (para sistemas basados en Linux y MacOS) y *PowerShell* (para sistemas Windows).

Zsh es una terminal interactiva y un poderoso lenguaje de *scripting.* Su desarrollo está basado en *Bash* (otra terminal para sistemas operativos UNIX/Linux). Sus principales diferencias con otras terminales son la característica de realizar cambios de directorio automáticos solamente ingresando el nombre del directorio, la abreviación de rutas de directorios, la corrección gramatical o de sintaxis y una comunidad de desarrolladores que construyen extensiones y temas para la terminal.

PowerShell es una terminal diseñada para reemplazar la famosa *CMD* de Windows. Al contrario de *Windows PowerShell*, *PowerShell* está construida con la tecnología *.NET Core 2.0.* Es una herramienta multiplataforma, esto es que puede ejecutarse en sistemas operativos diferentes a Windows, funcionando como una *CLI* y un lenguaje de *scripting.* Se encarga de ejecutar *cmdlets*,

comandos que permiten realizar una acción en específico, así como los archivos binarios de Zsh. La principal diferencia con otras *CLIs* es que se encarga de retornar objetos de .NET, lo que amplía los límites de programación de comandos.

3.4.6 Laravel Framework

Un desarrollador se enfrenta con varias tareas monótonas al realizar sus aplicaciones web en ciertos lenguajes de programación. Algunas de ellas pueden ser la administración de dependencias, las conexiones a bases de datos, el diseño de la autenticación, la administración de archivos y recursos o, bien, la generación de código compilado.

Es por esta razón que, como lo mencionamos en el capítulo anterior, contar con *frameworks* que faciliten la automatización y brinden soluciones sobre el manejo y mantenimiento de dichas tareas, es de suma importancia para un desarrollador.

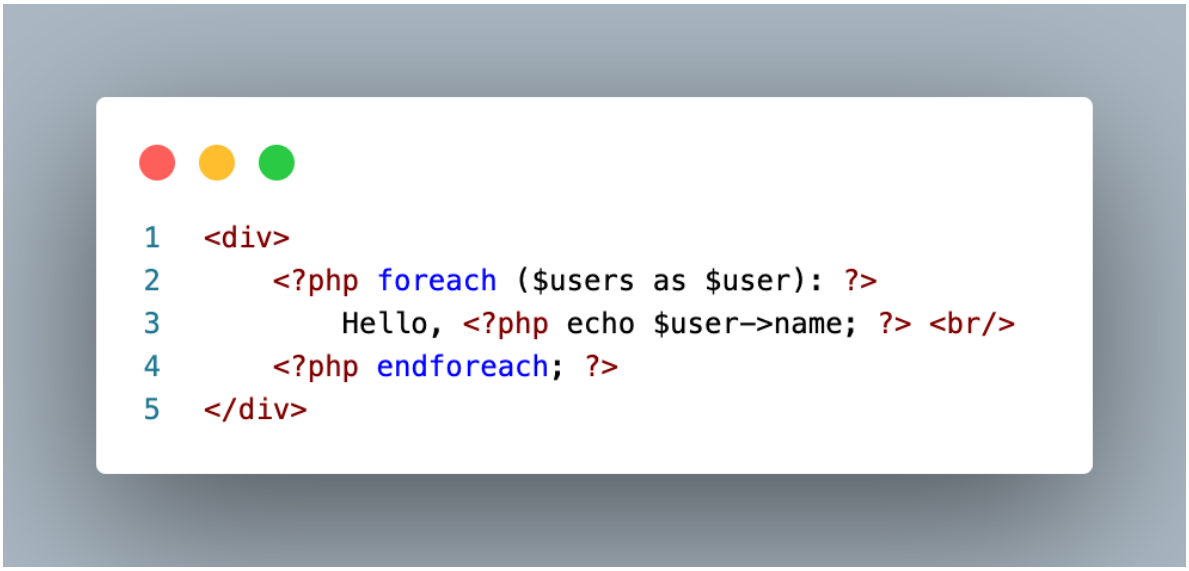
Algunos ejemplos de *frameworks* son: *Spring* en el lenguaje de programación Java; *Angular* para TypeScript; *VueJS* en JavaScript y *Laravel* para PHP.

Laravel provee una estructura de archivos y clases en PHP que permiten crear en forma rápida una aplicación web. Ofrece a los desarrolladores una sintaxis cómoda y sencilla que les permite enfocarse más en la lógica y la funcionalidad de su aplicación en vez de los detalles de su configuración. Es amigable con el desarrollador novato y le ofrece al experimentado una amplia variedad de herramientas para atender la inyección de dependencias, las pruebas unitarias y los eventos de tiempo real. Otra de sus principales características es que es escalable y con una comunidad de desarrolladores bastante grande.

Laravel nos permite manejar diferentes tipos de arquitectura. Es posible diseñar una aplicación monolítica en la cual las interfaces están directamente vinculadas con la lógica y el acceso a los datos. Las peticiones realizadas directamente por el cliente obtienen la información dentro de la misma infraestructura. Por otro lado, también permite el diseño de *API's* a las cuales una aplicación hará solicitudes, teniendo así las interfaces separadas del funcionamiento lógico y la conexión a la información, permitiendo que a nuestra API se conecten diversas aplicaciones, como móviles, web o de escritorio.

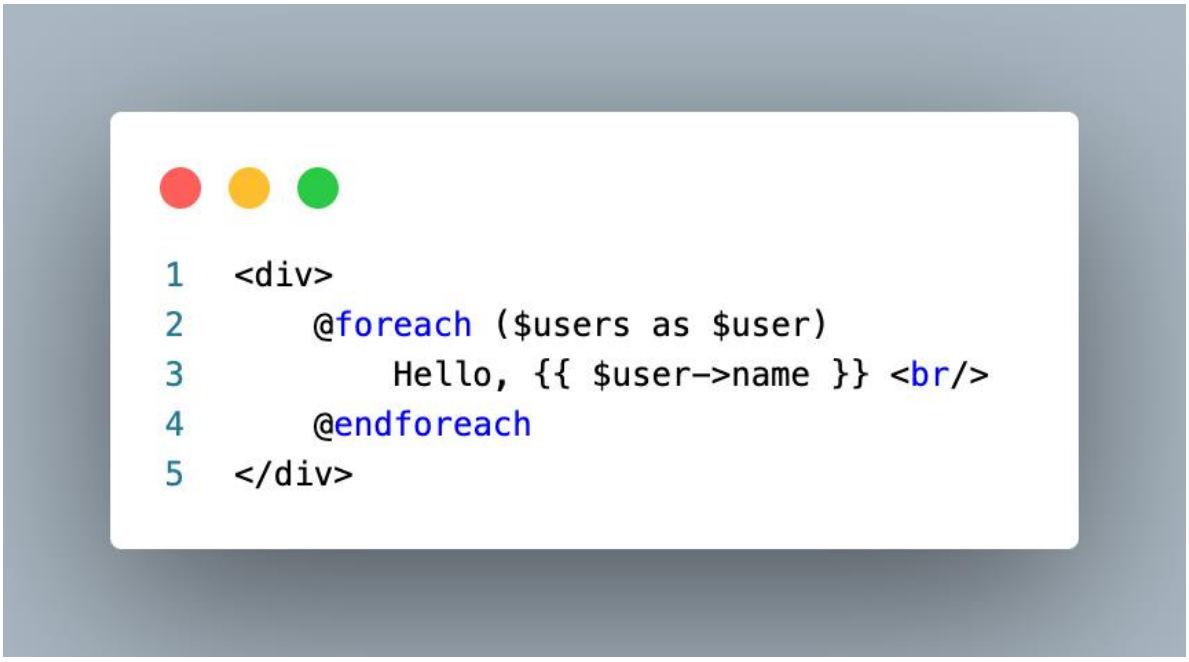
3.4.6.1 Motor de plantillas Blade

Para el diseño de las interfaces o vistas de una aplicación web, Laravel implementa sintaxis de PHP en forma reducida a través de *Blade*. Antes de que el archivo PHP sea procesado para ser renderizado a lenguaje HTML, Laravel lo hace pasar por una etapa de pre renderizado que se encarga de transcribir la sintaxis de Blade a sintaxis de PHP. Esto permite reducir muchas líneas de código, controlar nuestros patrones de diseño y fomentar el mantenimiento fácil y una legibilidad sencilla.



```
1  <div>
2      <?php foreach ($users as $user): ?>
3          Hello, <?php echo $user->name; ?> <br/>
4      <?php endforeach; ?>
5  </div>
```

Ilustración 28. Sintaxis PHP



```
1  <div>
2      @foreach ($users as $user)
3          Hello, {{ $user->name }} <br/>
4      @endforeach
5  </div>
```

Ilustración 29. Sintaxis Blade

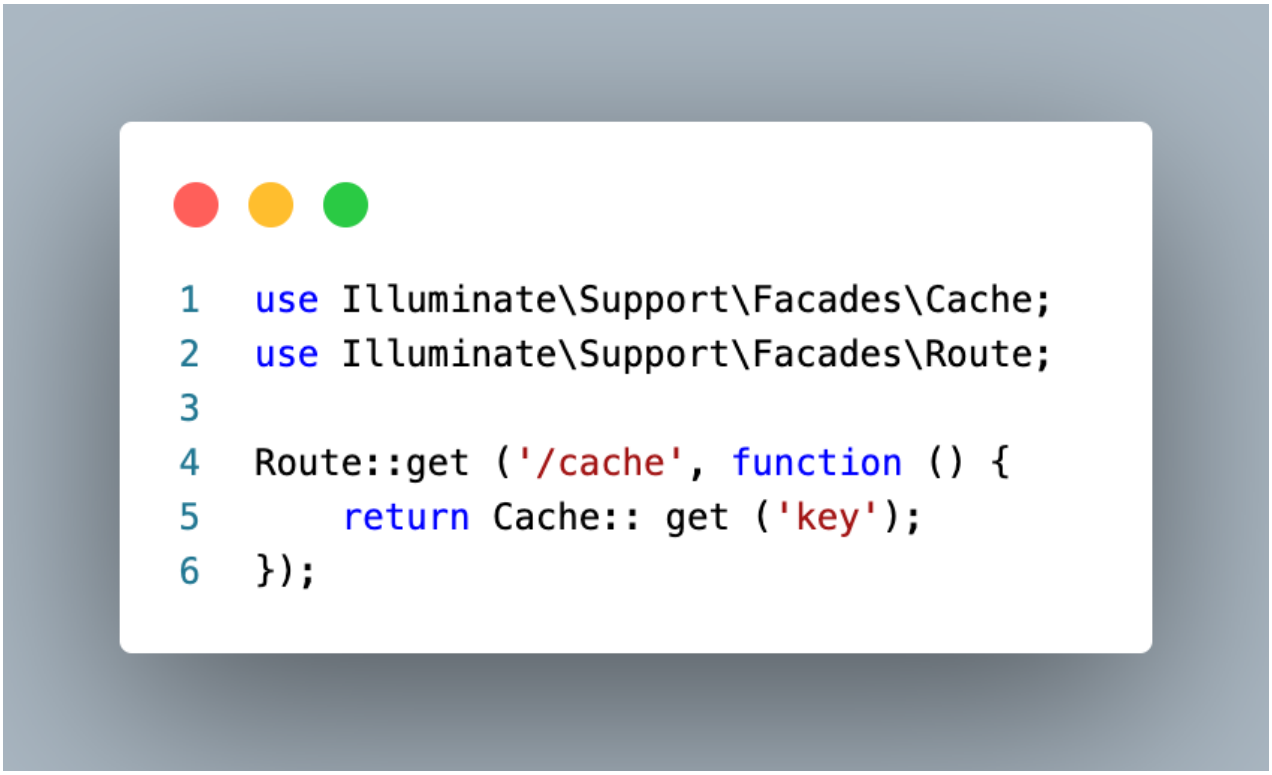
3.4.5.2 Facades

Dentro de la teoría del paradigma de programación orientada a objetos se encuentra el concepto de clases estáticas, interfaz y métodos estáticos. En Laravel, un *Facade* provee una interfaz estática a las clases que se definen dentro de la aplicación (recordando que PHP es un lenguaje que implementa el paradigma orientado a objetos).

Estas *Facades* o fachadas funcionan como una especie de proxies estáticos. Permiten generar una sintaxis memorable y breve dentro de la definición de las clases de nuestra aplicación, sin tener que recordar nombres de clases largos. Su implementación habla de un patrón de diseño estructural.

Podemos entonces decir que un *Facade* es una clase que representa un sistema entero que permite un acceso más cómodo a sus funcionalidades de manera simple y expresiva. Podemos pensar en los *Facades* como una forma de "envoltorio" alrededor de las clases subyacentes que nos permite acceder a ellas de manera más conveniente. Para utilizar un *Facade*, simplemente se realiza una llamada estática al método correspondiente en la fachada.

Por ejemplo, si necesitamos realizar un envío de correo electrónico, en lugar de importar e implementar la clase diseñada para ello, accedemos a la clase por medio de un *Facade* con métodos más reducidos y breves.



```
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\Route;

Route::get ('/cache', function () {
    return Cache:: get ('key');
});
```

Ilustración 30. Utilización del Facade Route y Cache para acceder a un recurso

3.4.5.3 Eloquent Object-Relational Mapper

Dentro de las funcionalidades de Laravel se encuentra *Eloquent,* una herramienta encargada de transformar lo más simple posible, consultas a una base de datos relacional en instancias de objetos. El objetivo de *Eloquent* es definir una clase modelo que corresponda a una tabla en la base de datos.

A dicha clase se le puede definir la llave primaria de la tabla, así como cada columna puede formar parte de los atributos de la clase. Sus métodos pueden ser desde relaciones con otras tablas, hasta formas de manipular los datos obtenidos de los registros en las tablas. Esto facilita la sintaxis y permanencia del código gracias al modelo vista controlador.

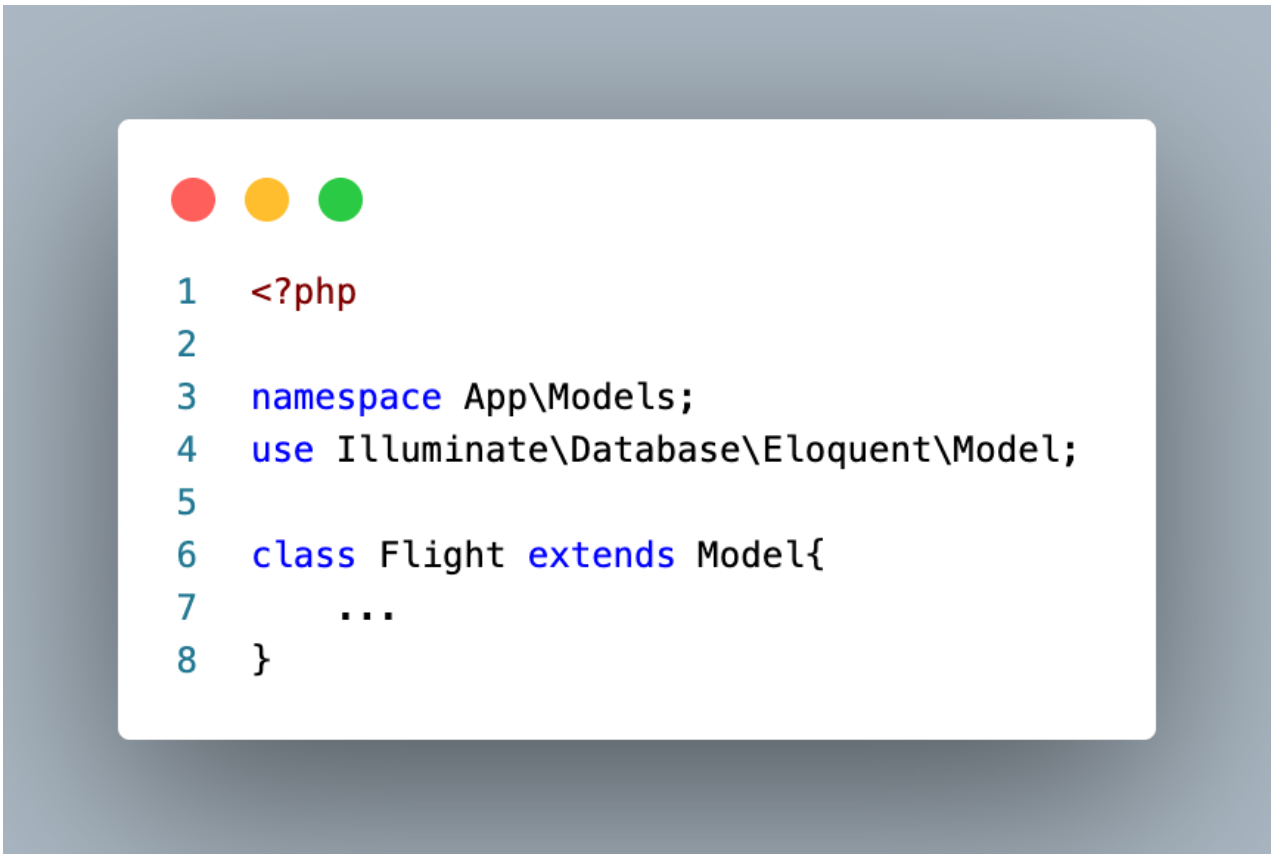


Ilustración 31. Ejemplo de Modelo en Laravel

3.5 Herramientas para la conexión entre módulos del sistema.

3.5.1 Docker

Docker Inc. es la empresa encargada del desarrollo de diversas tecnologías que tienen como objetivo la virtualización sencilla y eficiente. Docker Engine es una tecnología basada en el concepto de "contenedores" la cual empaqueta software en unidades de dicho nombre incluyendo bibliotecas, núcleos de sistemas operativos Linux, y todo lo necesario para virtualizar, de una forma liviana, un ambiente capaz de poder crear, probar e implementar aplicaciones rápidamente. (Amazon Web Services, n.d.)

Gracias a Docker Engine es posible crear varios contenedores operando de manera independiente que pueden comunicarse entre sí, hospedados en un sistema operativo anfitrión. De esta forma se optimizan los costos de infraestructura y facilita el desarrollo de sistemas. Su disponibilidad en Windows es reciente y es gracias a la tecnología Windows Subsystem For Linux (WSL).

Por medio de Docker Hub, es posible crear instancias de contenedores a través de "imágenes" sin proyectos creados por los usuarios. De esta forma es posible crear contenedores precargados con diversos sistemas operativos que tengan instaladas herramientas para su desarrollo, tales como: NodeJS, PHP, frameworks, servidores web, manejadores de bases de datos, entre otros.

A través de Docker Desktop obtenemos una interfaz gráfica para Windows y macOS capaz de gestionar las instancias de contenedores creadas por nuestro anfitrión. Evita la comunicación por parte del usuario con Docker Engine a través de una CLI; facilita el aprendizaje y reduce los tiempos de configuración; permite la creación de contenedores, su inicio, su detención, su eliminación e incluso su publicación o descarga a Docker Hub.

Si se omite el uso de Docker Desktop, será necesario aprender los comandos para la administración de los contenedores y ejecutarlos por medio de una terminal.

Hemos decidido la utilización de Docker Engine junto con Docker Desktop para reducir nuestros tiempos de desarrollo y facilitar el despliegue del sistema. En nuestra arquitectura definimos dos contenedores: uno dedicado a fungir como servidor web que alojará el código fuente del sistema y otro dedicado a almacenar la base de datos relacional, esto es, el lugar donde se guardará la información generada por el sistema.

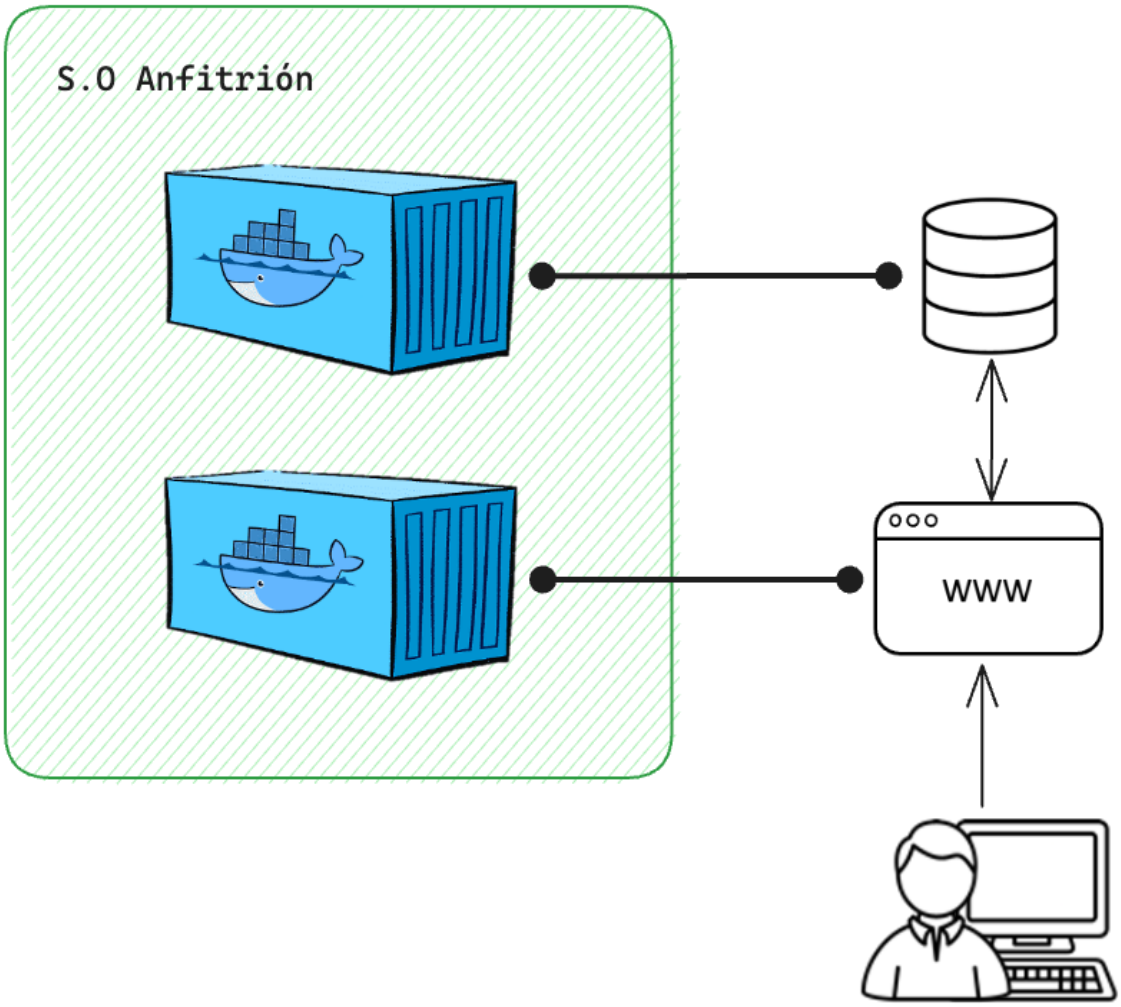


Ilustración 32. Arquitectura seleccionada para el desarrollo del sistema

3.6 Metodología ágil

Existen ocasiones en donde, al inicio del proyecto, el cliente no tiene bien definidos sus requisitos; esta indecisión genera cambios imprevistos que impactan de forma negativa al desarrollo. Por tal razón, surgen las metodologías ágiles, las cuales definiremos como aquellas que se adaptan a los nuevos requisitos con rapidez y flexibilidad, dando beneficios a cualquier proyecto que implique tener cambios de forma frecuente. (Think, Technology Consulting, 2018), (Garrido Sotomayor, 2023)

Existe un manifiesto ágil el cual cuenta con 12 principios:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9.La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia. (PRAGMA, n.d.)

En síntesis, deben cumplir lo siguiente:

- Individuos e interacciones sobre procesos y herramientas
    - Las personas que participan e implementan un proyecto, determinan los procesos y las herramientas con las cuales ayudarán a terminarlo con éxito.
- Software funcionando sobre documentación extensiva
    - Aunque la documentación es útil y necesaria, es mejor entregar un software funcional a lo largo del ciclo de vida del producto.
- Colaboración con el cliente sobre negociación contractual
    - Los clientes se consideran como colaboradores durante el desarrollo del proyecto y no sólo al inicio y final.
- Responder ante el cambio sobre seguir un plan
    - El desarrollo del proyecto debe permitir la incorporación de cambios en lugar de seguir un plan. (Think, Technology Consulting, 2018)

3.6.1 Metodología Kanban

El método Kanban fue desarrollado por primera vez por Toyota en la década de los 50’s, el cual consiste en un sistema para gestionar el flujo de trabajo y

mejorar la productividad. Kanban no sólo se utilizó en la industria del automóvil, sino que se aplicó en otros sectores como la informática y el desarrollo de software (TIBA México, 2016), surgiendo oficialmente a principios de 2007.

Kanban se centra en "hacer las cosas", y sus fundamentos se desglosan en dos principios y seis prácticas (Businessmap, n.d.):

Principio de la gestión del cambio

1. Comienza por lo que haces ahora
    - Al utilizar el método sobre los procesos existentes, éste reconoce que tienen un valor importante por lo que vale la pena conservarlos.
2. Aceptar el cambio incremental y evolutivo
    - Fomenta la administración continua de pequeños cambios incrementales mediante la aplicación de formas de colaboración y retroalimentación.
3. Fomentar los actos de liderazgo a todos los niveles
    a. Cada observación compartida fomenta una mejora continua (Businessmap, n.d.).

Principios de prestación de servicios

1. Centrarse en las necesidades y expectativas del cliente
    - La entrega de valor al cliente debe ser el centro de toda organización (Businessmap, n.d.).
2. Gestionar el trabajo
    - La capacidad de autoorganizarse permite centrarse en los resultados deseados.
3. Revisar periódicamente la red de servicios
    - La evaluación continua fomenta la mejora de los resultados entregados.

La palabra japonesa *kanban* significa *tarjeta visual* o *señal* y consiste en la elaboración de un tablero, donde se visualizan por lo menos tres columnas: pendientes, en proceso y terminadas, se pueden agregar más dependiendo de la complejidad del proyecto. De esta forma, todos los miembros del equipo pueden visualizar lo que hay que hacer y cuándo, además de evitar que las tareas se repitan o que se olviden de alguna de ellas.

En resumen, aplicar las herramientas y tecnologías manejadas actualmente en nuestro proyecto, permitió mejorar y proporcionar al cliente un sistema dinámico y visualmente atractivo. Para ello, tuvimos que analizar sus requerimientos verificando cuál de estas herramientas era más compatible para su desarrollo además del correcto funcionamiento. En forma adicional, basándonos en la metodología Kanban, logramos que el proyecto se gestionara en forma eficiente.

Capítulo 4. Desarrollo del sistema.

4.1 Forma de trabajo y control de versiones.

Como se mencionó en el capítulo anterior, utilizamos Kanban al desarrollar nuestro proyecto con la finalidad de atender cada uno de los requerimientos solicitados por el cliente.

Para lograr nuestro objetivo, creamos una lista de tareas y un tablero de control de los procesos donde podíamos verificar nuestro avance. El esquema era el siguiente: Una lista con tareas *por comenzar*, *en desarrollo*, *en pruebas* y *finalizadas*. De esta forma las funcionalidades se atendían por una persona estableciendo un periodo de tiempo para su desarrollo. Asimismo, con ayuda de Git, se iba actualizando cada uno de los requerimientos establecidos en el tablero y semanalmente, se realizaban minutas o reuniones para presentar los avances del proyecto y recibir retroalimentación de los usuarios del sistema.

Durante este proceso, pudimos comprobar que la principal problemática de esta metodología es que, si no se mantiene un estricto orden en la administración de las listas, la información puede perderse y los requerimientos definidos en un inicio pueden alterarse continuamente sin llegar a una definición final, provocando que el desarrollo del sistema se prolongue mucho más de lo esperado; de ahí la importancia de definir clara y estrictamente los requerimientos y alcance del sistema.

En cuanto al manejo y distribución del código fuente del proyecto, como ya se ha comentado en capítulos anteriores, se utilizaron dos herramientas: Git y GitHub. La primera nos proporcionó un robusto sistema o programa que se encarga de la administración y control de las versiones de nuestro código fuente; mientras que la segunda nos permitió distribuirlo entre los

desarrolladores y los usuarios; además, en caso de ser necesario, podría ser compartido con cualquier usuario de internet registrado en GitHub.

Git suele trabajar a través de un grafo dirigido. Cada nodo del grafo representa un punto del estado en el proyecto. Al realizar un cambio o finalizar una funcionalidad, nos encargamos de hacer un *commit*, el cual representa meramente un nuevo nodo que Git se encarga de registrar e identificar. Esta estructura de datos que Git implementa nos permite hacer saltos en el tiempo y manejar ramas o caminos distintos surgidos a partir de un nodo, creando la posibilidad de que dos personas editen un mismo archivo en distintos momentos y, después, puedan unificar sus cambios en forma sencilla, o bien, puedan trabajar al mismo tiempo en distintas partes de él.

Existen diferentes formas de manejar el flujo de trabajo en Git. A continuación, describimos dos, de las cuales la primera fue la que se decidió para el desarrollo de nuestro proyecto.

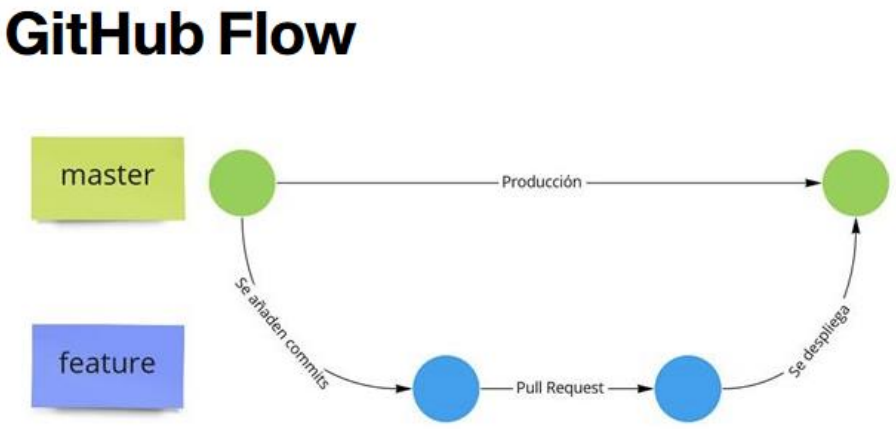


Ilustración 33. Flujo tipo GitHub

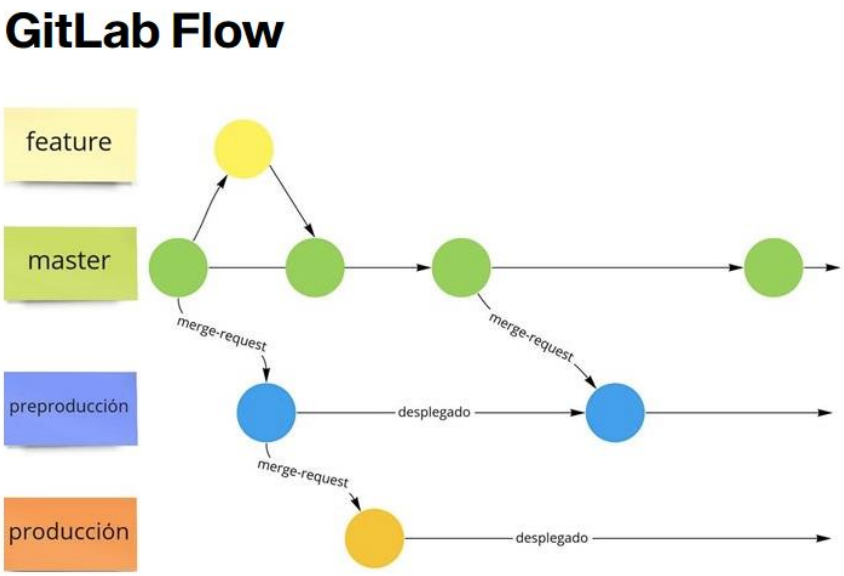


Ilustración 34. Flujo tipo GitLab

```
*   825b368 - (8 months ago) Merge pull request #48 from MauRamos334455/docs - Mauricio
|\
| * aa01c80 - (8 months ago) Diploma Ready - UstehKenny
| *   cddf72f - (8 months ago) Merge pull request #47 from MauRamos334455/master - Carolina Kennedy
| |\
| * \   7fce145 - (9 months ago) Merge pull request #46 from MauRamos334455/master - Carolina Kennedy
| |\ \
* | | | 9339768 - (8 months ago) Add counter to the report - Mauricio
| |_|/
|/| |
* | | fadce8b - (9 months ago) bug with diplomas - Mauricio
| |/
|/|
* | 8c8c65c - (9 months ago) Merge pull request #44 from MauRamos334455/docs - Mauricio
|\|
| * 9f9dcf2 - (9 months ago) Update activity-evaluation-report.blade.php - UstehKenny
* |   e9a256f - (9 months ago) Merge pull request #45 from MauRamos334455/diplomas - Carolina Kennedy
```

Ilustración 35. Ejemplo de commits realizados en la realización del proyecto

Al definir la arquitectura de nuestro proyecto se decidió utilizar dos componentes independientes, por lo cual se crearon y manejaron dos repositorios. En uno está el código asociado a la creación y administración de la base de datos; el segundo posee todo el contenido de la aplicación. Ambos

componentes son capaces de ser instalados en un servidor, ya sea en forma independiente, o bien, pueden estar juntos dentro del mismo servidor.

Para su correcto funcionamiento basta con indicar en la configuración de la aplicación la IP del servidor de la base de datos adecuada, ya sea que pertenezca a un ***localhost*** o a un servidor en la red. La razón por la cual se decidió no separar la aplicación en más componentes (el respectivo ***frontend*** y ***backend***) fue simplemente por fines prácticos, ya que no se requería esta forma de operar.

La aplicación fue pensada para escalar de una forma vertical con poco crecimiento y una extensión de almacenamiento en la base de datos mínima, ya que no opera por microservicios interconectados y tampoco fue necesario crear más componentes asociados a ella; además, no recibirá actualizaciones constantes pues será utilizada únicamente por medio de una red interna, aunque en determinado caso, sí podría ser usada a través de internet.

La arquitectura implementada es modular y en capas. El utilizar Laravel provocó que encapsuláramos la interfaz de usuario, la lógica de negocio y el acceso a los datos. No podemos argumentar que el diseño de la aplicación es monolítico debido a que la base de datos opera de forma totalmente independiente, pues pensamos en la facilidad de alterar y administrarla por largo del tiempo. Más allá de su funcionalidad, se planea el uso de datos que irán creciendo con el transcurso de los años y probablemente necesiten ser migrados continuamente.

4.2 Modelado de la base de datos.

Para nuestro sistema se consideró únicamente una fuente de información que es una base de datos relacional gestionada por PostgreSQL en un contenedor de Docker que quedará expuesta a través de un puerto lógico de conexión con la cual, la aplicación web podrá establecer comunicación. Para el diseño de la

base de datos se tomó como premisa el análisis realizado en el capítulo dos. A partir de él se desarrolló un modelo relacional.

Dentro de nuestro proyecto se creó un archivo de secuencia de comandos "*script*" que se encarga de inicializar y crear un contenedor de *Docker* en el cual posteriormente se cargará la base de datos. Existen dos versiones en el repositorio: Una para terminal *bash* (Linux) y otra para *PowerShell* (*Windows*). En la imagen siguiente se puede apreciar el código.

```
$container = "magenta-db"
$pg_password = "postgres"
$host_port = 5432
$container_port = 5432

Write-Output "Stopping the container..."
docker stop ${container}

Write-Output "Deleting previous container..."
docker rm ${container}

Write-Output "Creating our container..."
# Execute script inside scripts folder or change $PWD
docker run -e POSTGRES_PASSWORD=${pg_password} -d --name ${container} `
  -p ${host_port}:${container_port} `
  -v ${PWD}/source:/tmp/source postgres:latest

Write-Output "Starting container..."
docker start ${container}

# Copy this directory everytime that exists changes in the files
# and for the first time.
Start-Sleep -Seconds 3

Write-Output "Executing container..."
docker exec -it -u postgres -w /tmp/source ${container} psql
```

Ilustración 36. Código para inicializar y crear un contenedor de Docker

Para la ejecución del *script* es necesario inicializar *Docker*, esto se puede hacer desde la aplicación de escritorio de *Docker* o desde la terminal.

El *script* se ejecuta de la siguiente forma:

```
mauri@DESKTOP-5K9LJ5J A:\..\..\magenta-db main .\scripts\docker-init.ps1
Stopping the container...
magenta-db
Deleting previous container...
magenta-db
Creating our container...
e9292afe59bc31ac5f8dc5f2027de6ce98c172c0edef0646cac0befb11bb1e97
Starting container...
magenta-db
Executing container...
psql (15.1 (Debian 15.1-1.pgdg110+1))
Type "help" for help.

postgres=#
```

Ilustración 37. Ejecución del código

Posteriormente, se ejecuta el script de inicialización de la base de datos:

```
postgres=# \i s-00-main.sql
CREATE ROLE
CREATE DATABASE
GRANT
You are now connected to database "magenta" as user "magenta".
CREATE EXTENSION
CREATE EXTENSION
```

Ilustración 38. Código de inicialización de la base de datos

Este script se encarga de ejecutar todo el código relacionado a la base de datos: crea las tablas, secuencias, usuarios, otorga permisos y realiza una carga inicial de datos de prueba para interactuar con la aplicación.

4.2.1 Modelo Relacional

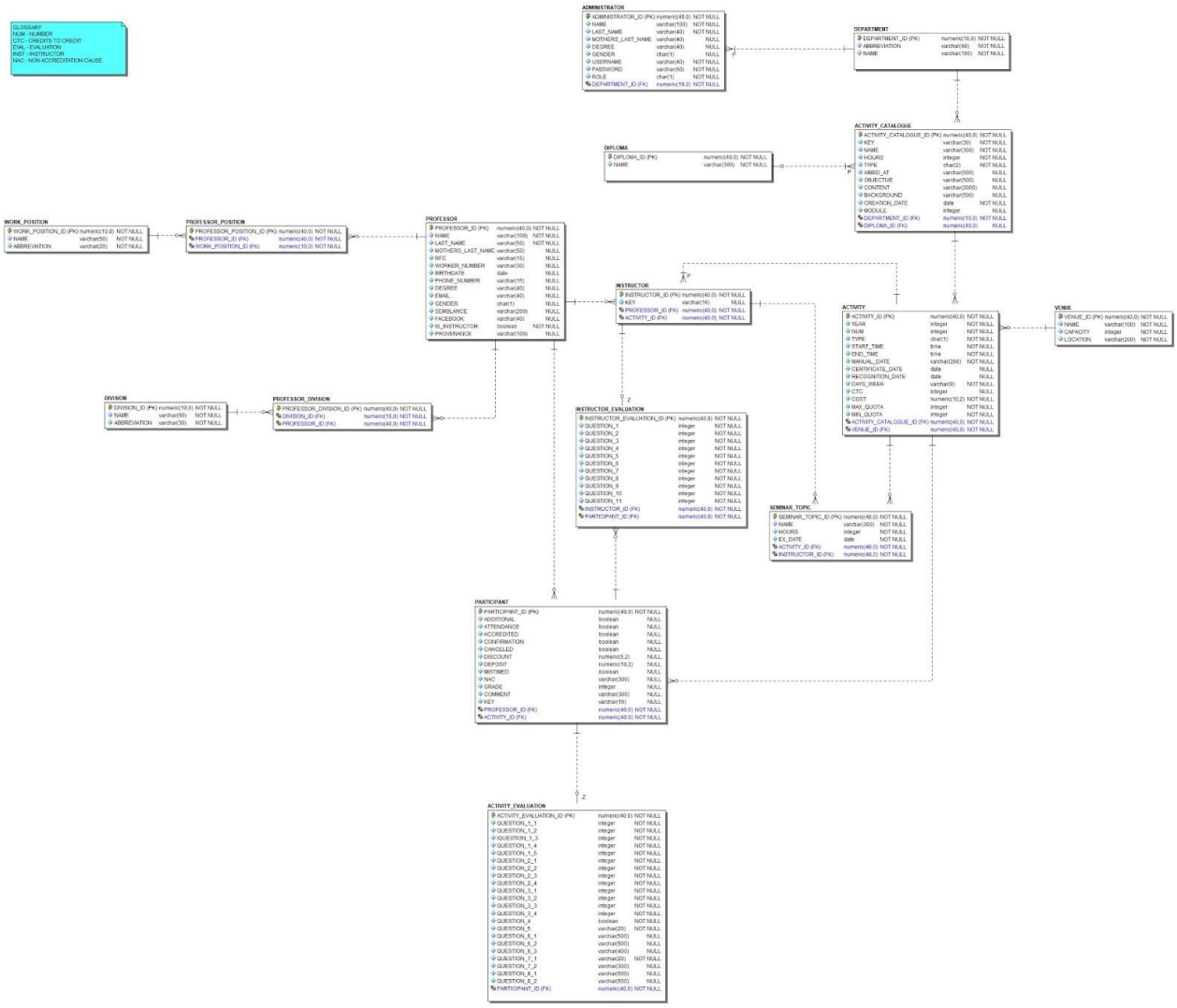


Ilustración 39. Modelo Relacional de la base de datos

A continuación, se describen las relaciones de las entidades:

1. Un administrador pertenece forzosamente a un departamento.

2. Un departamento posee de uno a muchos administradores.

3. Un departamento puede tener de ninguno a muchos catálogos de actividades.

4. Un catálogo de actividades está asociado a un departamento únicamente.

5. Un catálogo de actividades puede ser parte de uno o ningún diplomado.

6. Un diplomado posee de uno a muchos catálogos de actividades.

7. Una sede está asociada a ninguna o muchas actividades.

8. Una actividad sólo tiene una sede.

9. Una actividad pertenece forzosamente a un catálogo de actividades.

10. Un catálogo de actividades se asocia a al menos una actividad.

11. Una actividad posee de uno a varios instructores.

12. Un instructor lo es de una sola actividad.

13. Una actividad puede asociarse a ninguno o varios temas de un seminario.

14. Un instructor imparte de uno a varios temas de un seminario.

15. Un profesor posee de una a muchas posiciones de trabajo y de una a muchas divisiones.

16. Una posición de trabajo puede serlo de uno a muchos profesores, al igual que una división puede serlo.

17. Un profesor puede ser participante de ninguna o muchas actividades.

18. Un profesor puede ser instructor de ninguna o muchas actividades.

19. Un participante contesta únicamente una encuesta sobre una actividad.

20. Un participante puede contestar una encuesta de instructor por cada instructor de la actividad asociada.

21. Una encuesta de instructor se asocia únicamente a un instructor.

4.2.2 Modelo Lógico

El diseño de la base de datos contempla el manejador seleccionado; en este caso, PostgreSQL requiere únicamente de un esquema en el cual será creada la base de datos y dos usuarios que podrán poseer un acceso al servidor y, por tanto, a dicha base.

Estos usuarios tendrán las siguientes facultades:

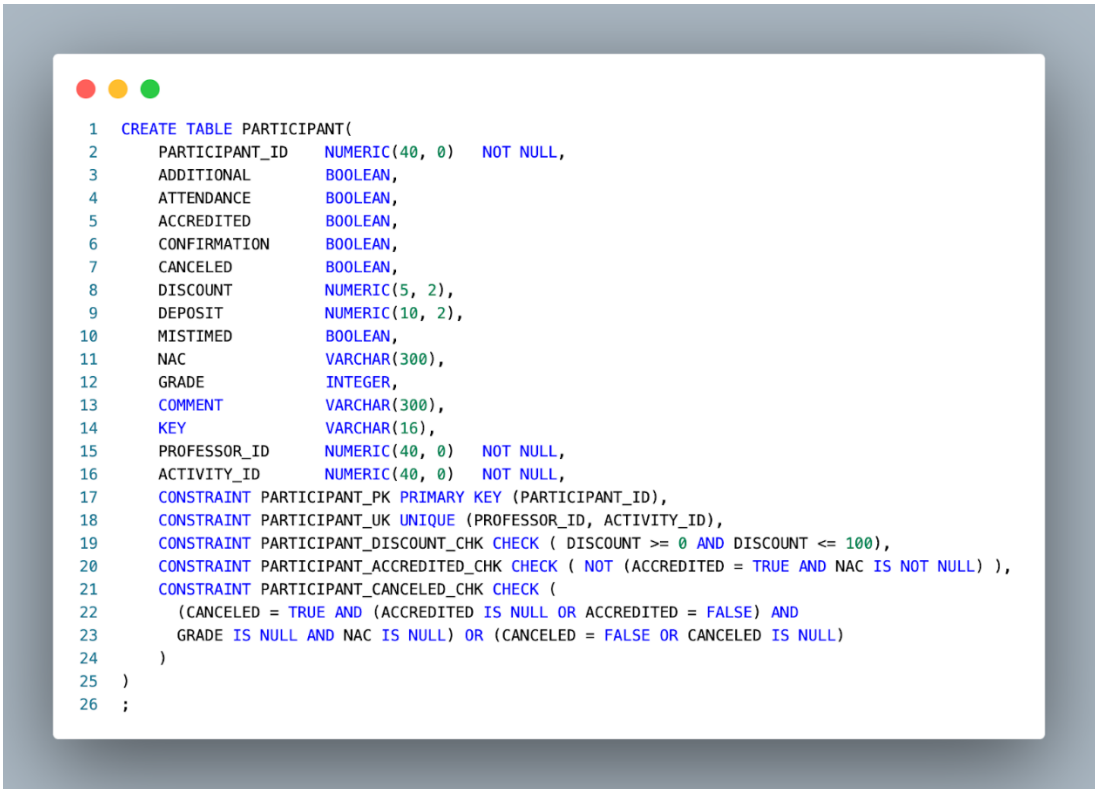
- dbo: Privilegio de lectura, escritura de datos y creación/modificación de objetos.
- usr: Privilegio de lectura y escritura de datos.

4.3 Explicación del desarrollo y evidencias de la base de datos.

El desarrollo de la base de datos se separó principalmente en dos tipos de *scripts.* Uno cuyo propósito era contener DDL y otros que contenían DML. Todos los scripts son invocados por medio del archivo *s-00-main.sql*, el cual invoca el siguiente orden de instrucciones para crear la base de datos:

1. Creación de tablas definidas en el diseño.
2. Creación de secuencias para llaves primarias.
3. Inserción de datos.

A continuación, se explica el diseño de una de las tablas para argumentar las decisiones tomadas a lo largo del diseño.



```sql
CREATE TABLE PARTICIPANT(
    PARTICIPANT_ID    NUMERIC(40, 0)   NOT NULL,
    ADDITIONAL        BOOLEAN,
    ATTENDANCE        BOOLEAN,
    ACCREDITED        BOOLEAN,
    CONFIRMATION      BOOLEAN,
    CANCELED          BOOLEAN,
    DISCOUNT          NUMERIC(5, 2),
    DEPOSIT           NUMERIC(10, 2),
    MISTIMED          BOOLEAN,
    NAC               VARCHAR(300),
    GRADE             INTEGER,
    COMMENT           VARCHAR(300),
    KEY               VARCHAR(16),
    PROFESSOR_ID      NUMERIC(40, 0)   NOT NULL,
    ACTIVITY_ID       NUMERIC(40, 0)   NOT NULL,
    CONSTRAINT PARTICIPANT_PK PRIMARY KEY (PARTICIPANT_ID),
    CONSTRAINT PARTICIPANT_UK UNIQUE (PROFESSOR_ID, ACTIVITY_ID),
    CONSTRAINT PARTICIPANT_DISCOUNT_CHK CHECK ( DISCOUNT >= 0 AND DISCOUNT <= 100),
    CONSTRAINT PARTICIPANT_ACCREDITED_CHK CHECK ( NOT (ACCREDITED = TRUE AND NAC IS NOT NULL) ),
    CONSTRAINT PARTICIPANT_CANCELED_CHK CHECK (
      (CANCELED = TRUE AND (ACCREDITED IS NULL OR ACCREDITED = FALSE) AND
      GRADE IS NULL AND NAC IS NULL) OR (CANCELED = FALSE OR CANCELED IS NULL)
    )
)
;
```

Ilustración 40. Ejemplo de creación de tabla

Para todas las llaves primarias, el tipo de dato escogido es *NUMERIC* de hasta 40 dígitos enteros, a excepción de una tercia de tablas que por su naturaleza requieren únicamente hasta 10 dígitos. Esto debido a que creímos conveniente el uso de llaves primarias artificiales e incrementales.

Desde nuestra visión las llaves naturales nunca bastan para identificar un registro y además se propagan en todas las demás tablas, impactando en el desempeño por su tamaño. Un ejemplo de lo anterior, lo observamos en una columna de tipo VARCHAR cuya longitud es máxima.

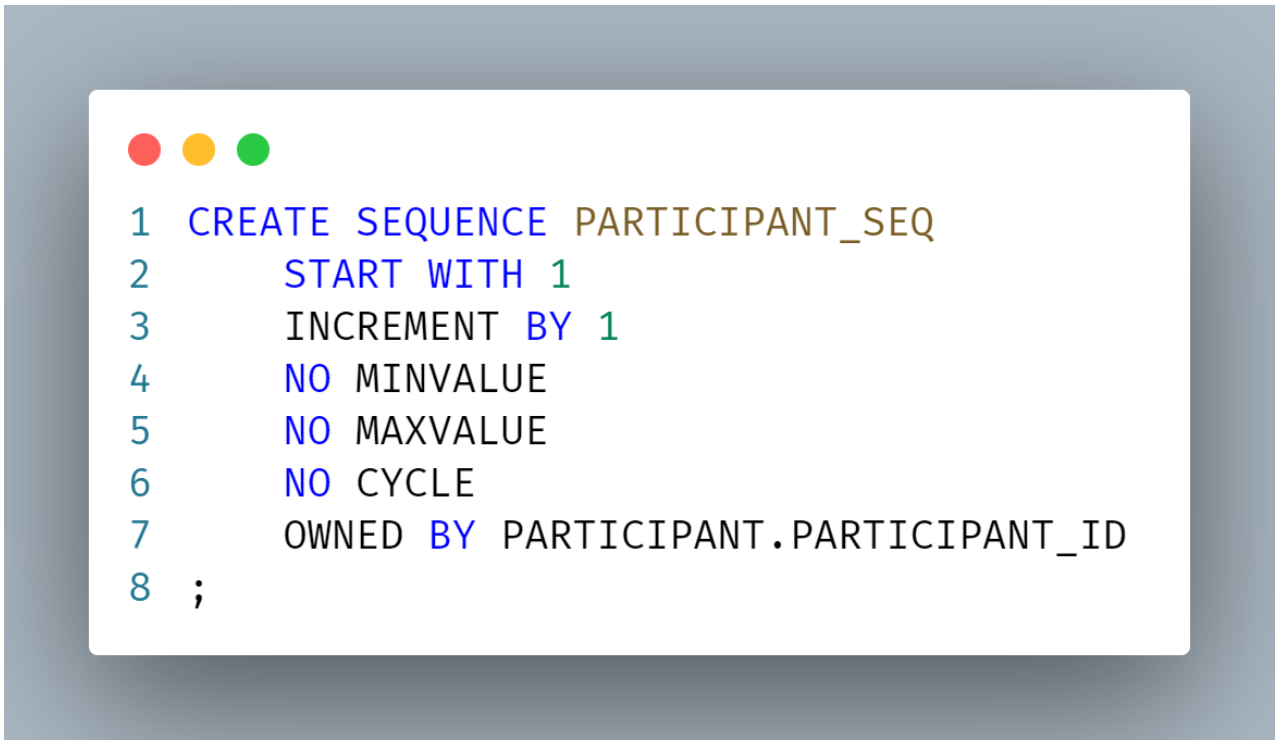
Por esta razón, a todas las llaves primarias se les asoció una secuencia, con el fin de que en la inserción de datos no se involucre a la llave primaria, es decir, que siempre en una inserción el valor por defecto para el nuevo registro es el siguiente valor en la secuencia asociada. Esto también permite evitar colisiones

en todas las tablas e identificar cada registro de una forma única sin que dependa de las reglas de negocio o el caso de estudio.

Los *constraints* repartidos en todas las tablas siguen la sintaxis descrita en el ejemplo. Su razón de existir fue detallada en capítulos anteriores. En la imagen podemos observar cinco ejemplos: un *constraint* que asocia la llave primaria a una columna; uno que se encarga de evaluar la unicidad entre la relación con la tabla profesor y la tabla actividad, además de constraints tipo *check*, que evalúan los valores de las columnas involucradas de tal forma que cumplan con cierta normatividad. En el primer *constraint* de tipo *check* de esta tabla se evalúa que el descuento ingresado sea siempre mayor o igual a cero y siempre igual o menor a 100, debido a que hablamos de un porcentaje.

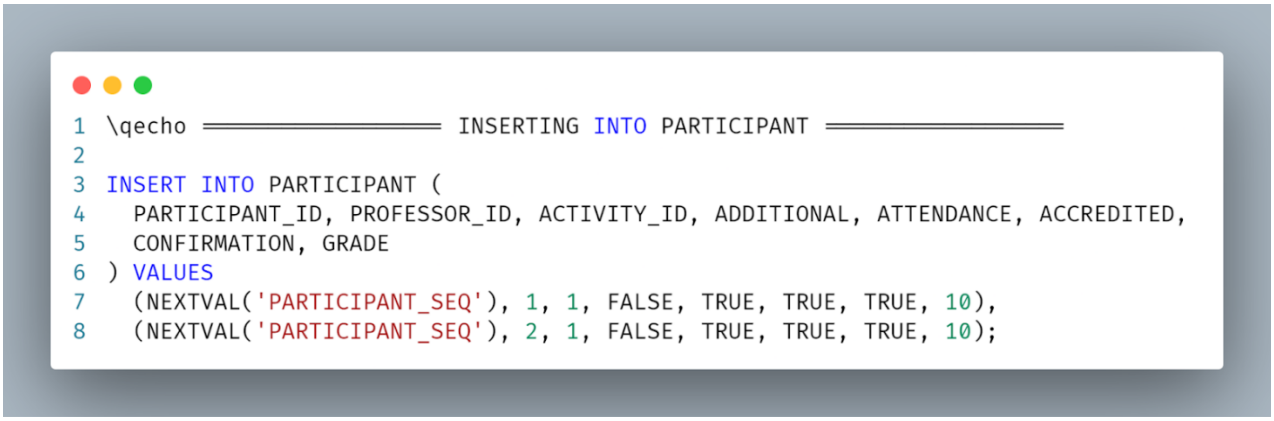
En la definición de las columnas dicotómicas utilizamos el tipo de dato *BOOLEAN*, el cual asigna únicamente un bit para saber si el valor de la columna representa verdadero o falso.

Todas las fechas son almacenadas con el tipo de dato *DATE* con el formato aaaa-mm-dd. Cantidades numéricas como pagos y calificaciones se manejan con el tipo de dato *NUMERIC* a una escala de dos decimales y hasta ocho cifras enteras. En ninguna parte del diseño se implementaron columnas de tipo *BLOB* o *TIMESTAMP,* pero consideramos que son un área de oportunidad si el sistema se enfrenta a constantes auditorías que exijan una bitácora que almacene cómo ha cambiado la información a través del tiempo en las tablas.



```sql
CREATE SEQUENCE PARTICIPANT_SEQ
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    NO CYCLE
    OWNED BY PARTICIPANT.PARTICIPANT_ID
;
```

Ilustración 41. Ejemplo de creación de secuencia



```sql
\qecho ═══════════════ INSERTING INTO PARTICIPANT ═══════════════

INSERT INTO PARTICIPANT (
  PARTICIPANT_ID, PROFESSOR_ID, ACTIVITY_ID, ADDITIONAL, ATTENDANCE, ACCREDITED,
  CONFIRMATION, GRADE
) VALUES
  (NEXTVAL('PARTICIPANT_SEQ'), 1, 1, FALSE, TRUE, TRUE, TRUE, 10),
  (NEXTVAL('PARTICIPANT_SEQ'), 2, 1, FALSE, TRUE, TRUE, TRUE, 10);
```

Ilustración 42. Ejemplo de inserción de datos

4.4 Explicación del desarrollo y evidencias del sistema

Como lo mencionamos en el capítulo 3, la herramienta con la que trabajamos fue Laravel donde implementamos el patrón Modelo-Vista-Controlador.

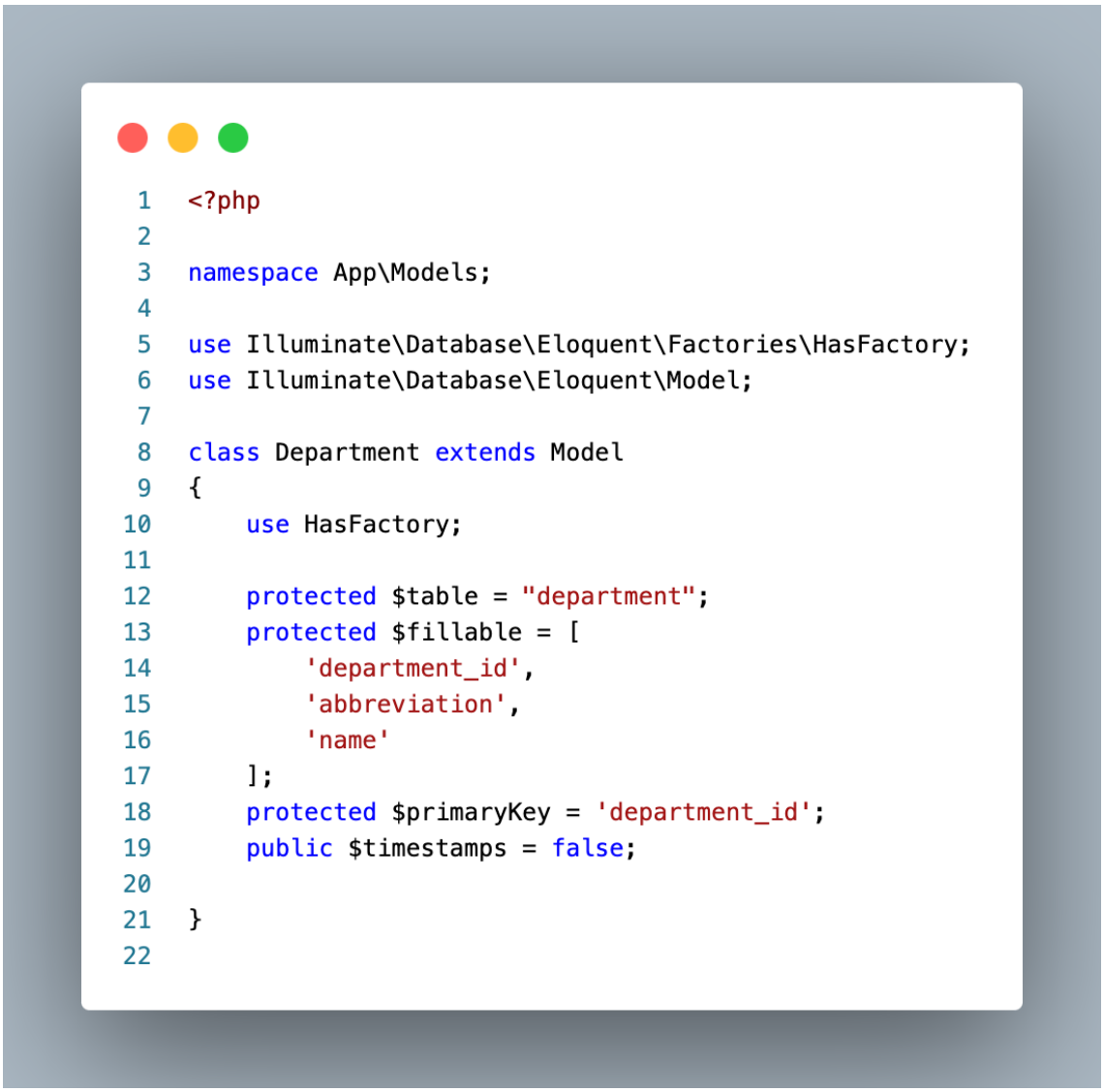
La estructura de este *framework* consiste en los siguientes directorios y archivos:

- **app**: Contiene la lógica principal de la aplicación como los Controladores y los Modelos.
- **bootstrap**: Contiene el archivo de arranque del *framework,* además de un directorio que almacena archivos generados por éste para su optimización como son archivos asociados al enrutamiento y servicios. Este directorio no es manipulado por el usuario.
- **config**: Se encuentran todos los archivos de configuración de la aplicación. Desde este directorio se configura por ejemplo la conexión o definición del controlador de la base de datos, el servidor de correo electrónico o el servicio de autenticación.
- **database**: Contiene los archivos relacionados con la base de datos como las migraciones y las semillas de ésta.
- **public**: Contiene los archivos accesibles públicamente, como los estilos CSS, código JavaScript o fuentes. Dichos archivos han pasado por un proceso de compilación o simplificación para ser optimizados dentro de las peticiones web.
- **resources**: Se almacenan las vistas desarrolladas además de archivos CSS o JavaScript con la característica de ser legibles o explícitos, a

diferencia de aquellos que se almacenan en el directorio *public*. Estos archivos también pueden ser de tipo Sass o Less.

- **routes**: Contiene todas las rutas HTTP de la aplicación. En nuestro diseño constan principalmente del archivo web.php, el cual posee rutas que invocan métodos contenidos en los controladores.
- **storage**: Almacena archivos generados por la aplicación como el caché, los logs y archivos de sesión.
- **tests**: Contiene archivos con clases creados para realizar pruebas automatizadas.
- **vendor**: Contiene las dependencias de Composer de la aplicación o proyecto.
- **.env**: Archivo que se utiliza para configurar variables de entorno específicas de la aplicación.
- **.gitignore**: Especifica los archivos y directorios que se deben de ignorar en Git.
- **composer.json**: Archivo de configuración de Composer que define las dependencias del proyecto a instalar.

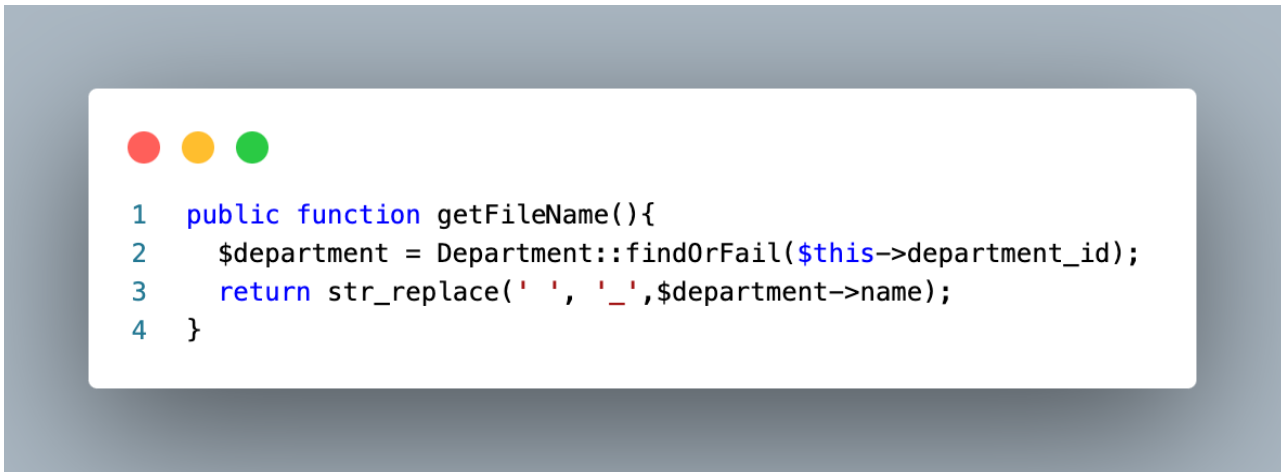
Conociendo la estructura de Laravel, el patrón con el que trabajamos y los datos que requerimos de la base de datos, procedimos con la creación de los modelos en la carpeta *app* > *Models.* En esta carpeta asociamos los atributos de nuestro modelo o clase a la tabla correspondiente de la base de datos, para así poder acceder a ellos en la aplicación.



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Department extends Model
{
    use HasFactory;

    protected $table = "department";
    protected $fillable = [
        'department_id',
        'abbreviation',
        'name'
    ];
    protected $primaryKey = 'department_id';
    public $timestamps = false;

}

```

Ilustración 43. Ejemplo del modelo 'Departamento'

De igual forma, se pueden agregar funciones e ir creando las vistas que se habían establecido en el maquetado del apartado 2.5.



```
1  public function getFileName(){
2    $department = Department::findOrFail($this->department_id);
3    return str_replace(' ', '_',$department->name);
4  }
```

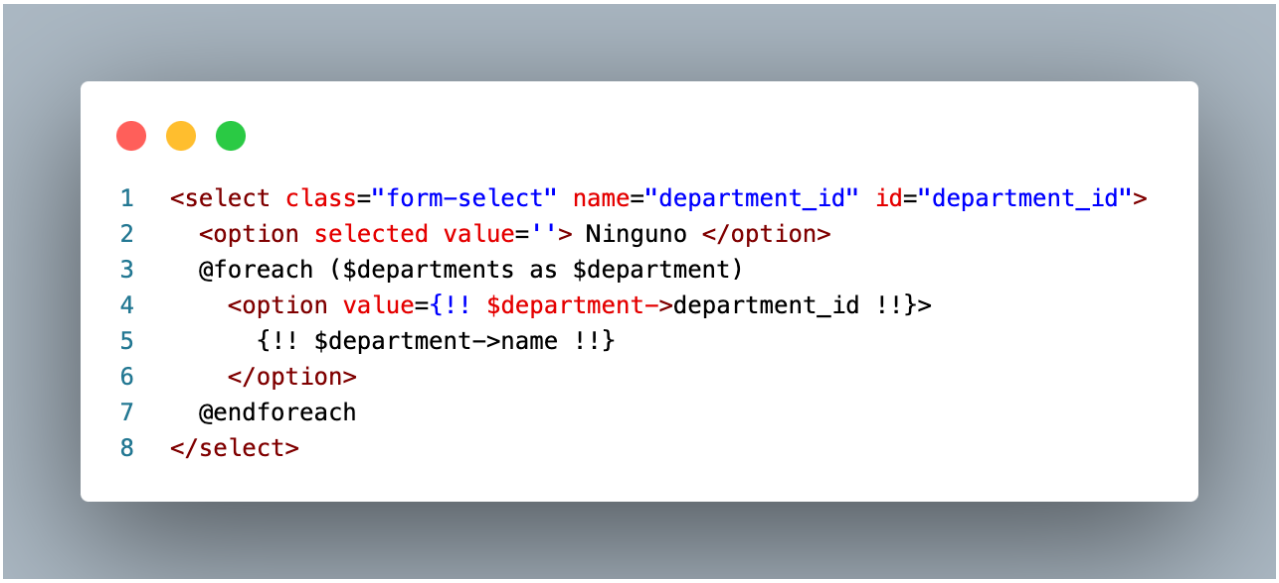
Ilustración 44. Ejemplo de función dentro de un modelo

Con respecto a las vistas, éstas podremos encontrarlas en la carpeta *resources* > *views* y, adicionalmente, se añadirán las carpetas:

- **docs**: Se encuentran los formatos que se generarán en PDF.
- **layouts**: Contiene el encabezado y la barra lateral que se requieren en todas las vistas.
- **pages**: Se encuentran todas las vistas correspondientes a las secciones.
- **partials**: Contiene los diferentes mensajes que se desplegarán para indicar si algo se realizó con éxito o con error.

Todos los archivos de las vistas son escritos en HTML y además utilizarán el motor Blade, cuyo funcionamiento fue explicado en el apartado 3.4.6.1.

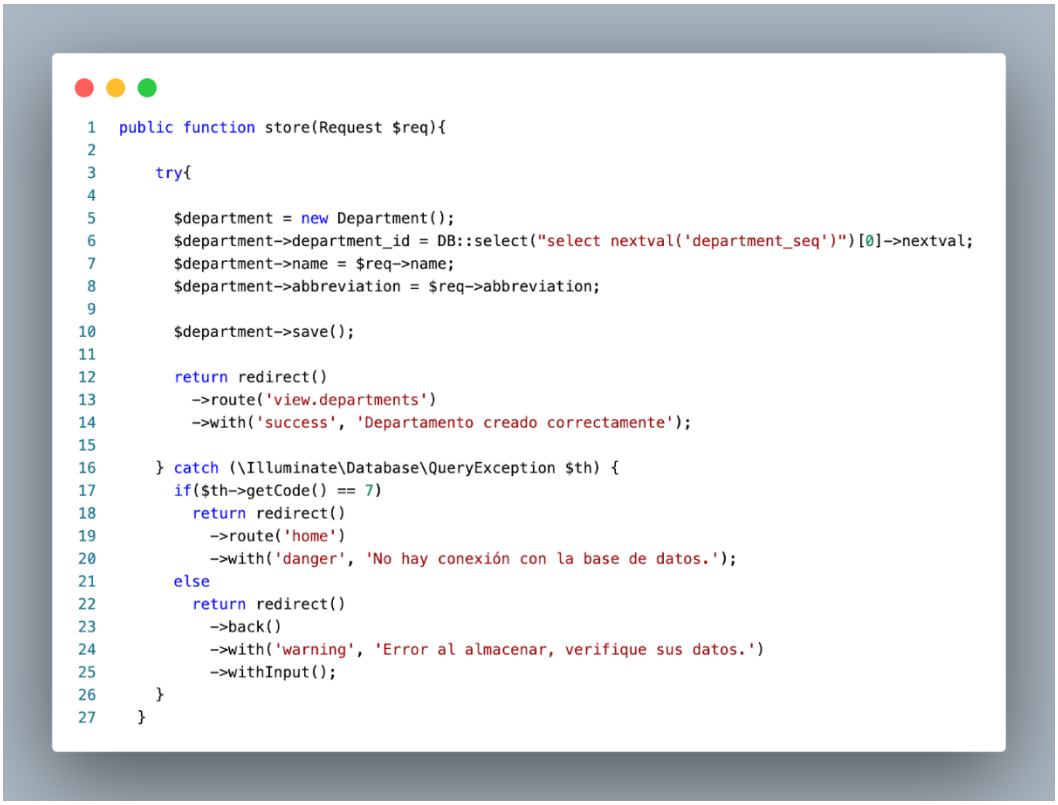
Blade nos ayuda a reducir el código y hacerlo más legible.



```
<select class="form-select" name="department_id" id="department_id">
  <option selected value=''> Ninguno </option>
  @foreach ($departments as $department)
    <option value={!! $department->department_id !!}>
      {!! $department->name !!}
    </option>
  @endforeach
</select>
```

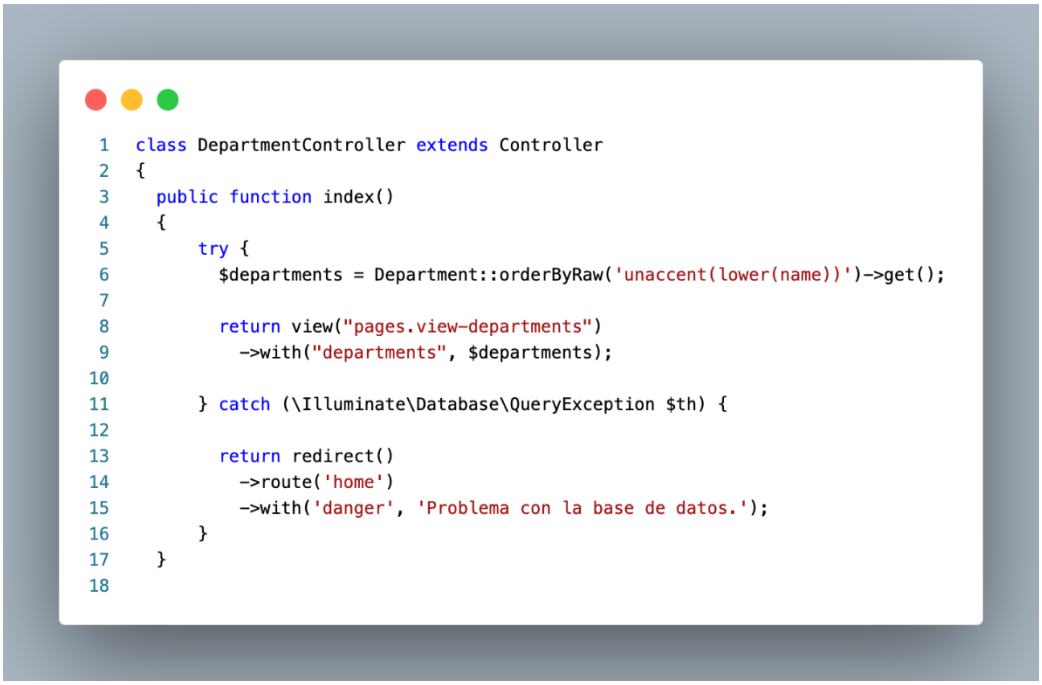
Ilustración 45. Ejemplo de fragmento de vista con Blade

Para obtener el valor de las variables utilizamos un controlador que recupera la información de la base de datos. Inicialmente, escribimos las operaciones básicas para gestión de información, también conocidas por el acrónimo *CRUD*, cuyas siglas significan; *Create* (Crear), *Read* (Leer), *Update* (Actualizar) y *Delete* (Eliminar).



```php
public function store(Request $req){

    try{

      $department = new Department();
      $department->department_id = DB::select("select nextval('department_seq')")[0]->nextval;
      $department->name = $req->name;
      $department->abbreviation = $req->abbreviation;

      $department->save();

      return redirect()
        ->route('view.departments')
        ->with('success', 'Departamento creado correctamente');

    } catch (\Illuminate\Database\QueryException $th) {
      if($th->getCode() == 7)
        return redirect()
          ->route('home')
          ->with('danger', 'No hay conexión con la base de datos.');
      else
        return redirect()
          ->back()
          ->with('warning', 'Error al almacenar, verifique sus datos.')
          ->withInput();
    }
  }
```

Ilustración 46. Ejemplo del método Create

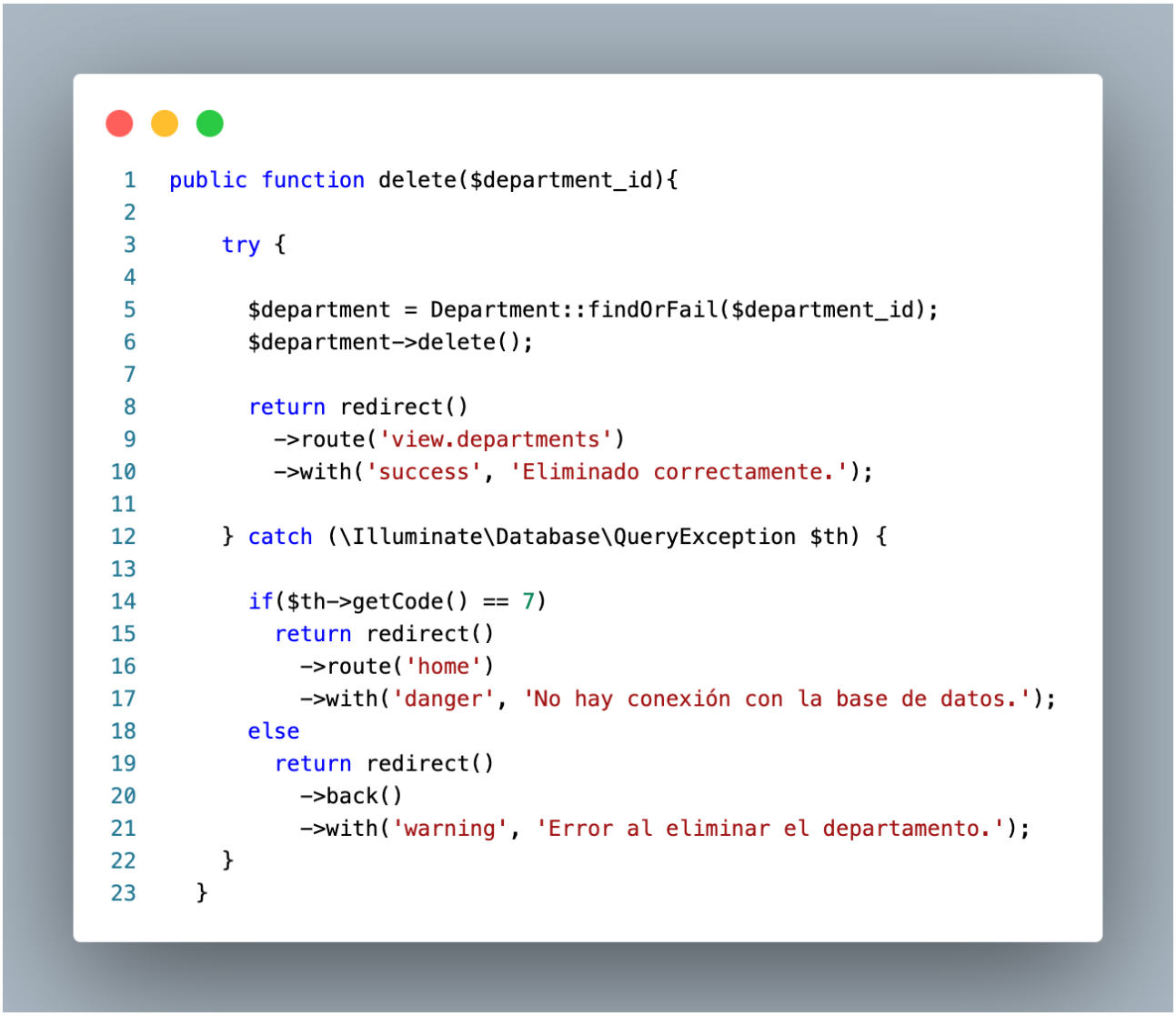


```php
class DepartmentController extends Controller
{
  public function index()
  {
      try {
        $departments = Department::orderByRaw('unaccent(lower(name))')->get();

        return view("pages.view-departments")
          ->with("departments", $departments);

      } catch (\Illuminate\Database\QueryException $th) {

        return redirect()
          ->route('home')
          ->with('danger', 'Problema con la base de datos.');
      }
  }

```

Ilustración 47. Ejemplo del método Read

```php
public function update(Request $req, $department_id){

    try {

      $department = Department::findOrFail($department_id);
      $department->name = $req->name;
      $department->abbreviation = $req->abbreviation;

      $department->save();

      return redirect()
        ->route('edit.department', $department->department_id)
        ->with('success', 'Cambios realizados.');

    } catch (\Illuminate\Database\QueryException $th) {

      if($th->getCode() == 7)
        return redirect()
          ->route('home')
          ->with('danger', 'No hay conexión con la base de datos.');
      else
        return redirect()
          ->back()
          ->with('department', $department)
          ->with('warning', 'Error al almacenar, verifique sus datos.');
    }
  }
```

Ilustración 48. Ejemplo del método Update



```php
public function delete($department_id){

    try {

      $department = Department::findOrFail($department_id);
      $department->delete();

      return redirect()
        ->route('view.departments')
        ->with('success', 'Eliminado correctamente.');

    } catch (\Illuminate\Database\QueryException $th) {

      if($th->getCode() == 7)
        return redirect()
          ->route('home')
          ->with('danger', 'No hay conexión con la base de datos.');
      else
        return redirect()
          ->back()
          ->with('warning', 'Error al eliminar el departamento.');
    }
  }
```

Ilustración 49. Ejemplo del método Delete

El controlador es invocado por medio de una petición a la ruta definida en nuestro archivo web.php. Posteriormente, nuestro controlador, sea cual sea el método invocado, devolverá siempre una vista o una redirección a ésta.

```php
//Route Department
Route::get('departamentos', "DepartmentController@index")
  ->name("view.departments");
Route::get('departamentos/crear', "DepartmentController@create")
  ->name("create.department");
Route::post('departamentos/almacenar', "DepartmentController@store")
  ->name("store.department");
Route::get('departamentos/actualizar/{department_id}', "DepartmentController@edit")
  ->name("edit.department");
Route::put('departamentos/guardar/{department_id}', "DepartmentController@update")
  ->name('update.department');
Route::delete('departamentos/eliminar/{department_id}', "DepartmentController@delete")
  ->name("delete.department");
```

Ilustración 50. Ejemplo de rutas en el archivo web.php

En resumen, los componentes en el diseño que implementamos del Modelo vista controlador interactúan de la siguiente manera en el código escrito: A cada vista se le asocia un controlador y un controlador puede o no ocupar varios modelos. Los métodos de un modelo no deben contener o utilizar otros modelos y a cada modelo se le asocia una tabla de la base de datos.

4.5 Cargas extensas de datos de prueba

Para comprobar la correcta ejecución del sistema se crearon diversos scripts de cargas de datos aleatorias generados con una herramienta online llamada *Mockaroo*. Dicha herramienta es capaz de crear datos aleatorios especificando ciertas características que deben cumplirse, como el tipo de dato asociado, patrones de texto, rangos de números, longitudes escalares, etc.

Esta herramienta nos permite descargar la información en varios formatos como CSV, SQL, JSON, XML, entre otros. Para nuestro proyecto, utilizamos el formato SQL editando después el código para darle cierto formato y modificamos ciertos valores que no podían automatizarse.

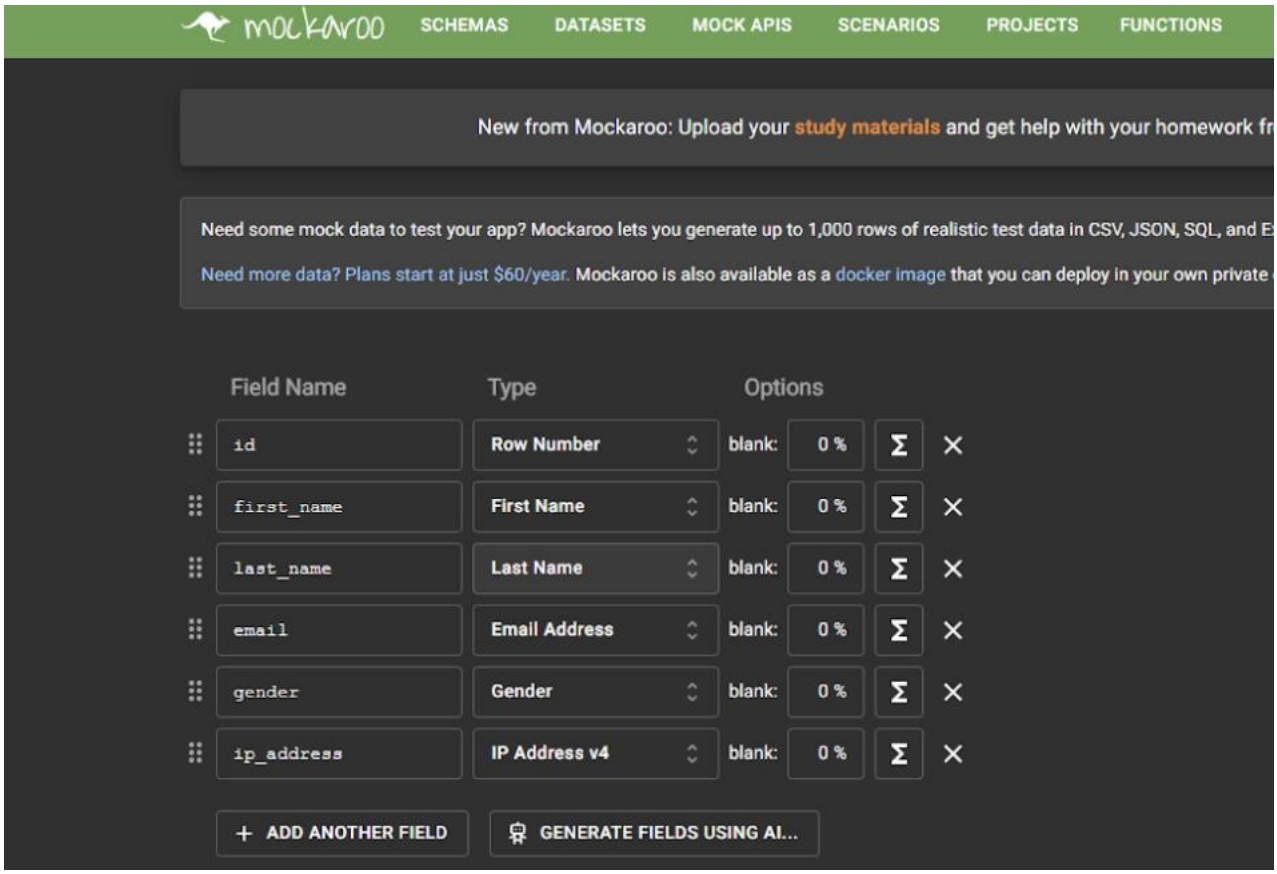
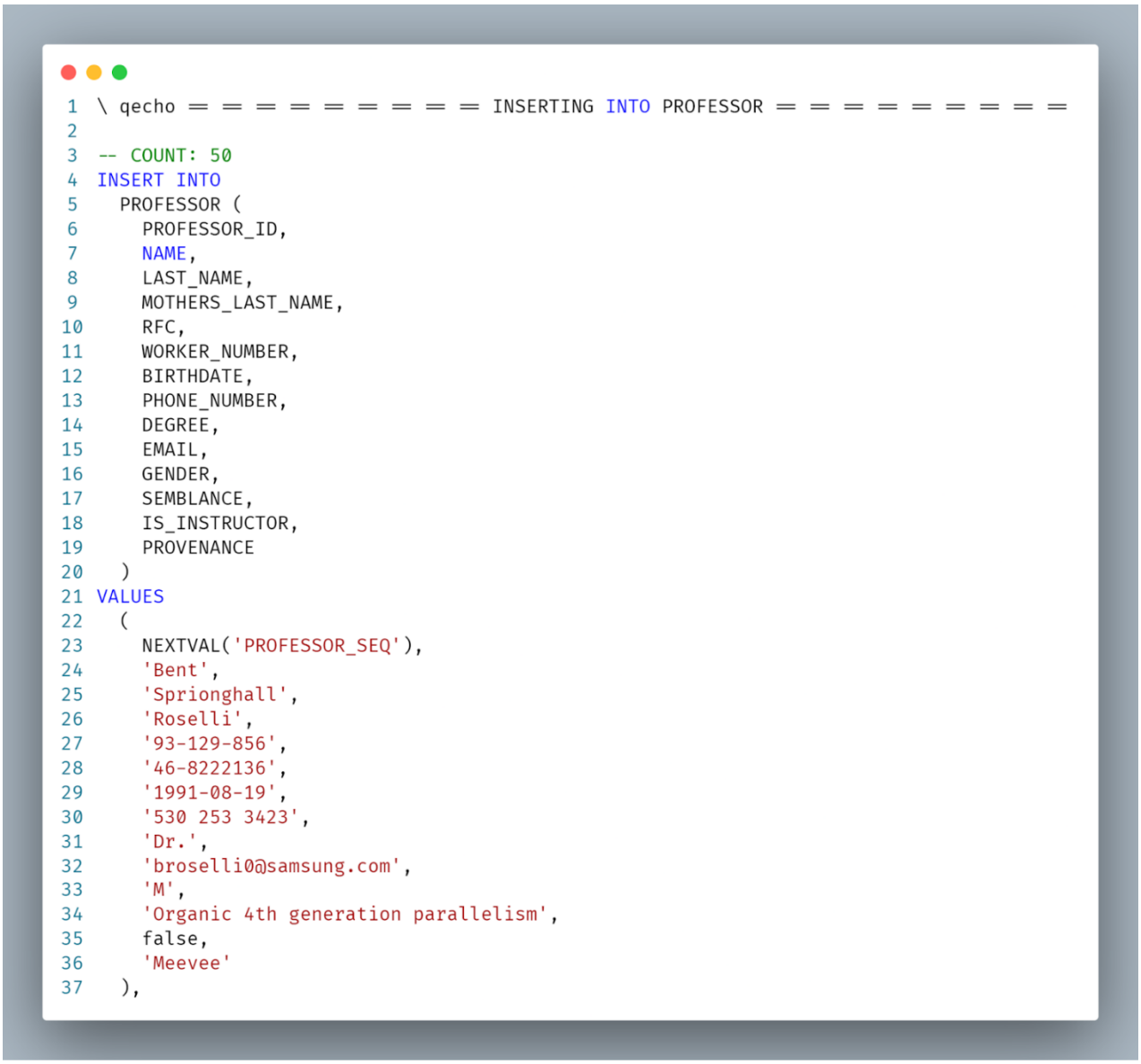


Ilustración 51. Ejemplo de creación de datos con Mockaroo

Se creó una configuración específica para cada tabla de la base de datos de nuestra aplicación; luego fueron creados algunos registros para poblar nuestro sistema, excepto por aquellas tablas que debían contener pocos registros a consecuencia de las reglas de negocio dadas; un ejemplo de esto es la tabla de Administradores y Departamentos. Las tablas con más datos fueron las de Actividad y Profesores.



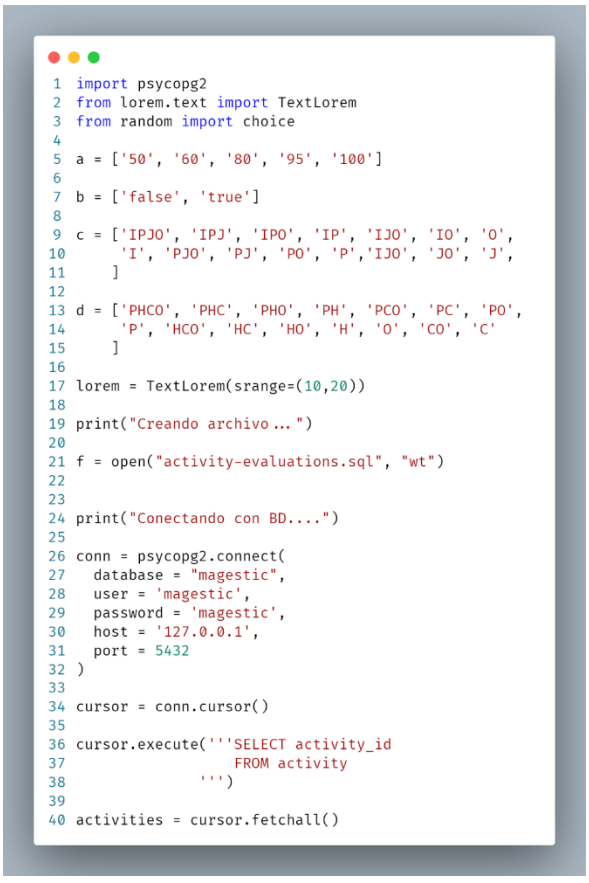
```
\ qecho = = = = = = = = = = INSERTING INTO PROFESSOR = = = = = = = = = =

-- COUNT: 50
INSERT INTO
  PROFESSOR (
    PROFESSOR_ID,
    NAME,
    LAST_NAME,
    MOTHERS_LAST_NAME,
    RFC,
    WORKER_NUMBER,
    BIRTHDATE,
    PHONE_NUMBER,
    DEGREE,
    EMAIL,
    GENDER,
    SEMBLANCE,
    IS_INSTRUCTOR,
    PROVENANCE
  )
VALUES
  (
    NEXTVAL('PROFESSOR_SEQ'),
    'Bent',
    'Sprionghall',
    'Roselli',
    '93-129-856',
    '46-8222136',
    '1991-08-19',
    '530 253 3423',
    'Dr.',
    'broselli0@samsung.com',
    'M',
    'Organic 4th generation parallelism',
    false,
    'Meevee'
  ),
```

Ilustración 52. Ejemplo de datos generado con Mockaroo

Una particularidad de automatizar nuestra carga de datos fueron las tablas resultantes de las evaluaciones de los participantes sobre la actividad y sus instructores. Dichas tablas necesitan las llaves primarias de las tablas con las que respectivamente se relacionan, así como textos largos y valores específicos construidos en la definición de las tablas.

A pesar de que *Mockaroo* es una excelente herramienta tiene ciertas limitantes, debido a lo anterior, desarrollamos un par de *scripts* o programas en lenguaje *Python* que generan los registros de manera específica para la sección de evaluaciones.

El código es bastante sencillo, se inicializa una conexión a la base de datos por medio del controlador o *driver* y funciones dentro de la biblioteca *psycopg2* para PostgreSQL. Con ayuda del paquete *TextLorem*, pudimos crear grandes párrafos de texto aleatorio que llenarían nuestras encuestas y, en consecuencia, darían mejores resultados en la generación de los PDF's de demostración. Con ayuda del paquete *choice* pudimos generar objetos (números) aleatorios de un conjunto específico definido.



```python
import psycopg2
from lorem.text import TextLorem
from random import choice

a = ['50', '60', '80', '95', '100']

b = ['false', 'true']

c = ['IPJO', 'IPJ', 'IPO', 'IP', 'IJO', 'IO', 'O',
     'I', 'PJO', 'PJ', 'PO', 'P','IJO', 'JO', 'J',
    ]

d = ['PHCO', 'PHC', 'PHO', 'PH', 'PCO', 'PC', 'PO',
     'P', 'HCO', 'HC', 'HO', 'H', 'O', 'CO', 'C'
    ]

lorem = TextLorem(srange=(10,20))

print("Creando archivo ... ")

f = open("activity-evaluations.sql", "wt")


print("Conectando con BD....")

conn = psycopg2.connect(
  database = "magestic",
  user = 'magestic',
  password = 'magestic',
  host = '127.0.0.1',
  port = 5432
)

cursor = conn.cursor()

cursor.execute('''SELECT activity_id
                  FROM activity
              ''')

activities = cursor.fetchall()
```

Ilustración 53. Código en Python para generar datos

```
print("Realizando archivo ... ")
for activity in activities :

  cursor.execute('''SELECT participant_id
                    FROM participant
                    WHERE activity_id = '''
                    +
                    str(activity[0])
                )
  participants = cursor.fetchall()



  for participant in participants:

    sql_sentence = ('''INSERT INTO ACTIVITY_EVALUATION '''
    '''(ACTIVITY_EVALUATION_ID, QUESTION_1_1, '''
    '''QUESTION_1_2, QUESTION_1_3, QUESTION_1_4, '''
    '''QUESTION_1_5, QUESTION_2_1, QUESTION_2_2, '''
    '''QUESTION_2_3, QUESTION_2_4, QUESTION_3_1, '''
    '''QUESTION_3_2, QUESTION_3_3, QUESTION_3_4, QUESTION_4, '''
    '''QUESTION_5, QUESTION_6_1, QUESTION_6_2, QUESTION_6_3, '''
    '''QUESTION_7_1, QUESTION_7_2, QUESTION_8_1, '''
    '''QUESTION_8_2, PARTICIPANT_ID) VALUES ('''
    '''NEXTVAL('ACTIVITY_EVALUATION_SEQ'), '''    +
    choice(a)  + ''', ''' + choice(a)  + ''', ''' + choice(a)  + ''', ''' +
    choice(a)  + ''', ''' + choice(a)  + ''', ''' + choice(a)  + ''', ''' +
    choice(a)  + ''', ''' + choice(a)  + ''', ''' + choice(a)  + ''', ''' +
    choice(a)  + ''', ''' + choice(a)  + ''', ''' + choice(a)  + ''', ''' +
    choice(a)  + ''', ''' + choice(b)  +
    ''', \''''   + choice(c)        + '''\', \'''' + lorem.sentence() +
    '''\', \'''' + lorem.sentence() + '''\', \'''' + lorem.sentence() +
    '''\', \'''' + choice(d)        + '''\', \'''' + lorem.sentence() +
    '''\', \'''' + lorem.sentence() + '''\', \'''' + lorem.sentence() +
    '''\', ''' +
    str(participant[0]) + ''');\n''')

    f.write(sql_sentence)


print("Cerrando conexión y archivo....")
conn.close
f.close()
```

Ilustración 54. Código en Python para insertar los datos

Capítulo 5. Realización de pruebas, muestra de resultados.

5.1 Muestra del funcionamiento general del sistema

5.1.1 Ingresando como administrador

En este capítulo realizaremos una guía sobre cómo manejar nuestro proyecto paso a paso, además de mostrar su funcionamiento y versatilidad para ser adaptado a otras instituciones.

Iniciaremos con abrir el programa, el cual fue previamente instalado en un servidor local donde también se han cargado ya algunos usuarios y contraseñas en modo *Prueba*. Una vez que aparezca la pantalla principal de MAGENTA se puede iniciar la sesión; si se ingresa como Administrador, seleccionaremos la opción "Ingresar".

Ilustración 55. Vista de Inicio del sistema

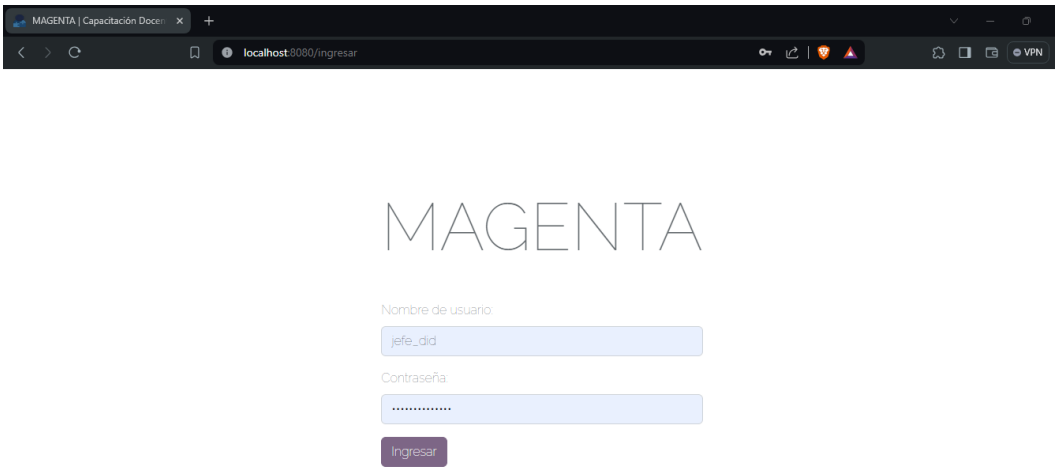


Ilustración 56. Vista de Inicio de Sesión

La pantalla mostrará accesos rápidos con las opciones más utilizadas del sistema. Del lado izquierdo, están todas las secciones restantes a las que tiene acceso el administrador.

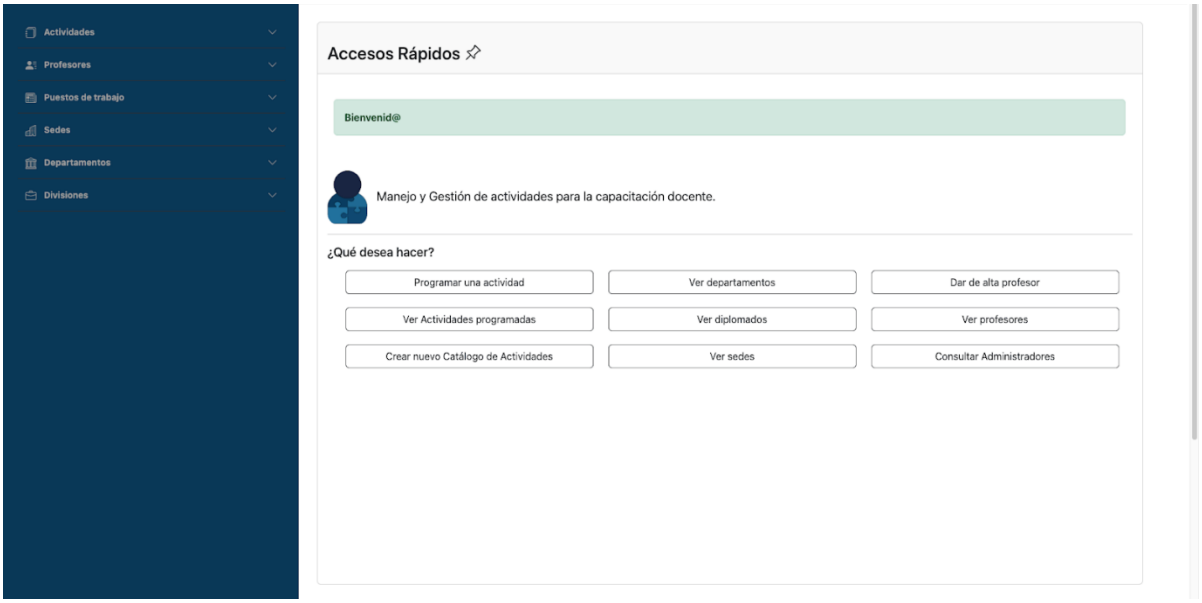


Ilustración 57. Vista Principal como usuario administrador

Si nos vamos a la sección de *"Ver departamentos"*, se podrá visualizar la información cargada previamente con los departamentos del caso de estudio de nuestra organización.

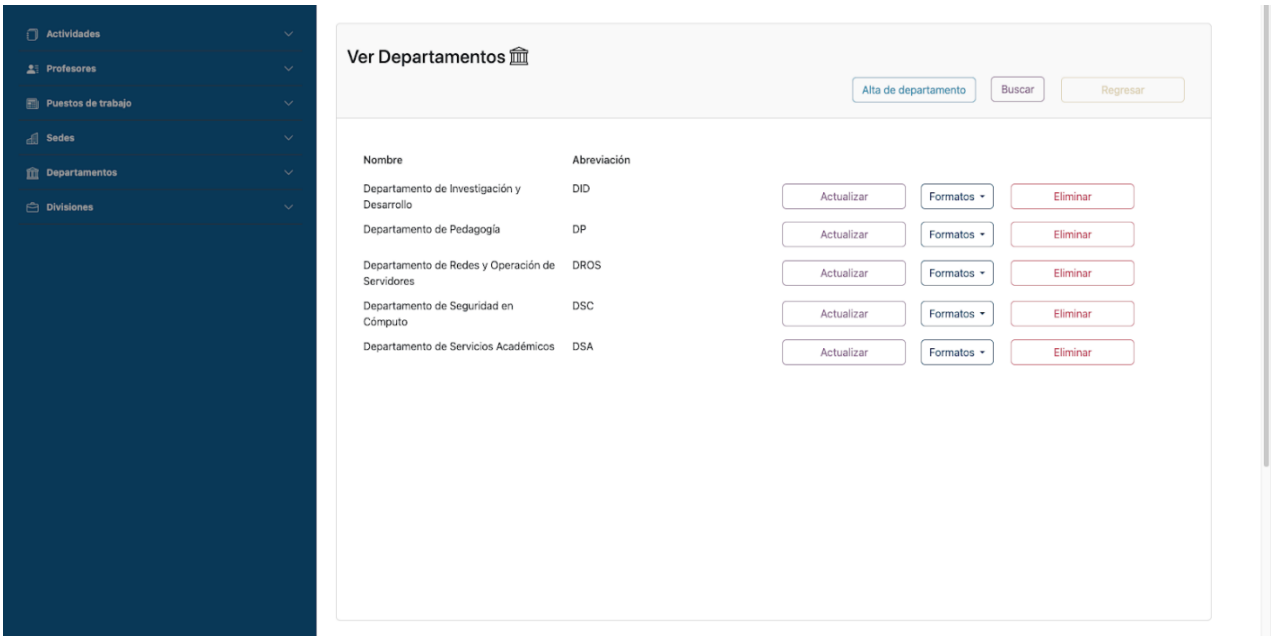


Ilustración 58. Vista Ver Departamentos como usuario administrador.

Si se tienen que agregar más departamentos, podemos dirigirnos a la opción de *"Alta de departamento"*, que se encuentra en el menú izquierdo o en la misma vista de *Ver departamentos.*

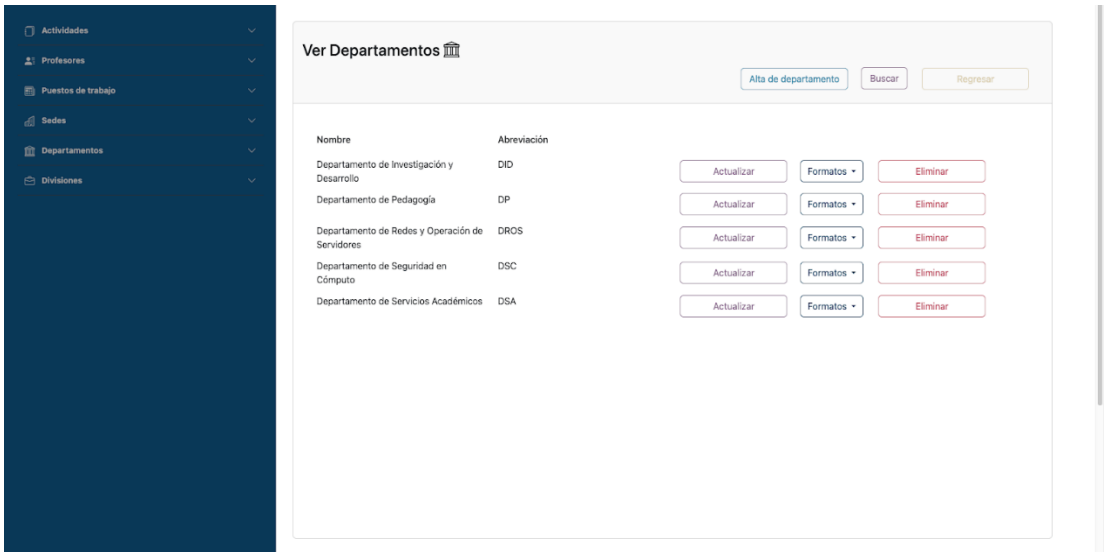


Ilustración 59. Opción Alta de departamento como usuario administrador.

Lo dirigirá a una nueva vista que pedirá el nombre y la abreviatura del nuevo departamento. Una vez establecidos, le daremos clic en *Guardar.*

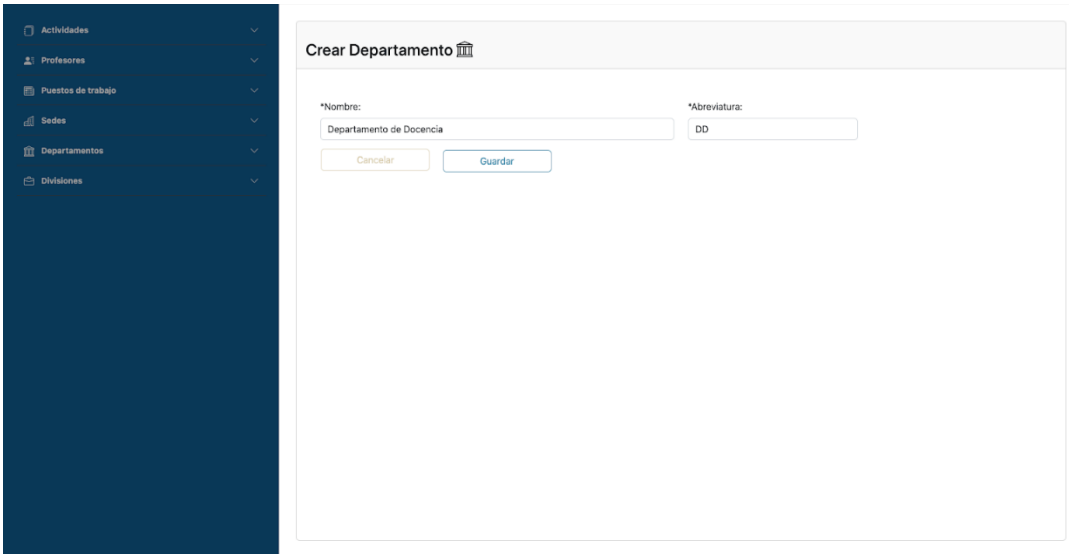


Ilustración 60. Vista Crear Departamento como usuario administrador.

Nos redirigirá a *Ver departamentos*, donde se visualizará el dato recién agregado.

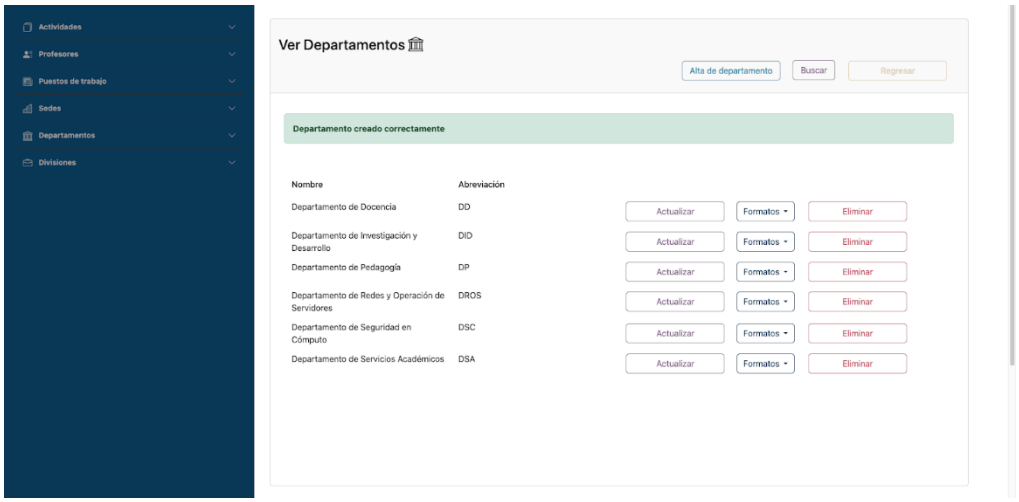


Ilustración 61. Mensaje de Departamento creado

También podemos modificar los datos con el botón *Actualizar*, éste lo dirigirá a la vista correspondiente para modificar el nombre y la abreviatura.

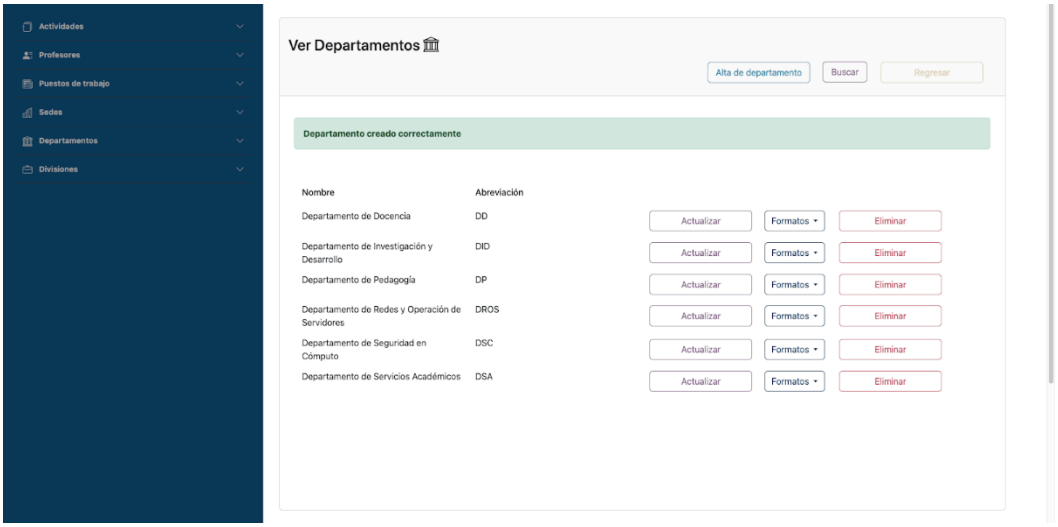


Ilustración 62. Opción de Actualizar como usuario administrador.

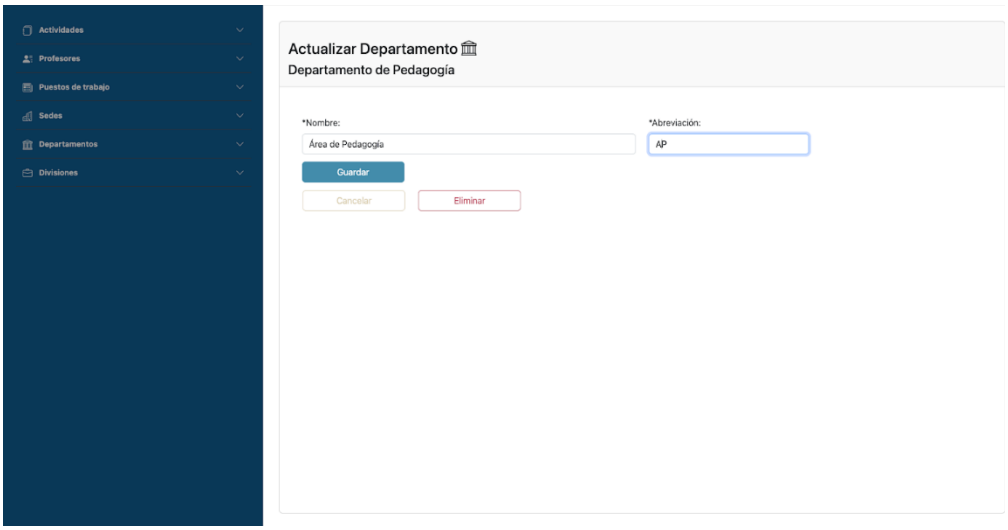


Ilustración 63. Vista Actualizar Departamento como usuario administrador.

Si los cambios se realizaron satisfactoriamente, mandará el mensaje de *Cambios realizados* y permanecerá en la misma página por si se desea realizar otro; de lo contrario, puede hacer clic en *Cancelar* para regresar a *Ver departamentos* o continuar su navegación con el menú lateral.

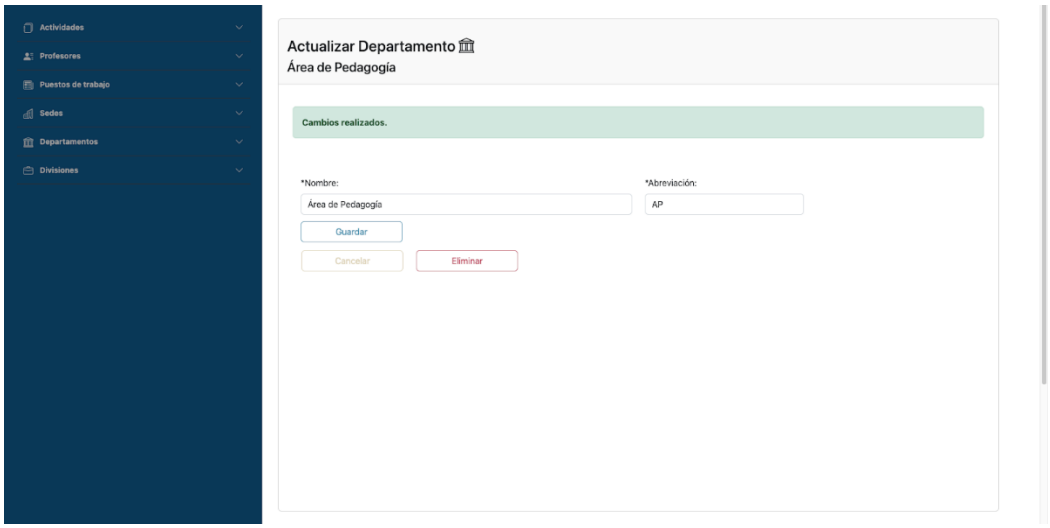


Ilustración 64. Mensaje de Cambios realizados

Para eliminar algún dato, podemos hacerlo desde la vista de *Actualizar* o en *Ver departamentos*, con el botón *Eliminar.*

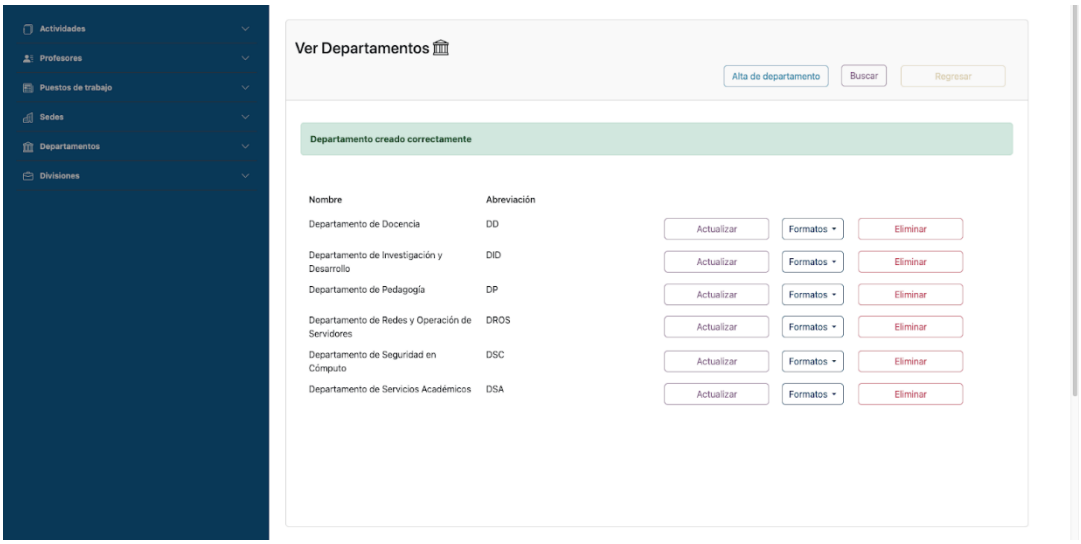


Ilustración 65. Opción de Eliminar como usuario administrador.

Dicho botón desplegará una alerta o *modal* para confirmar el dato a eliminar.

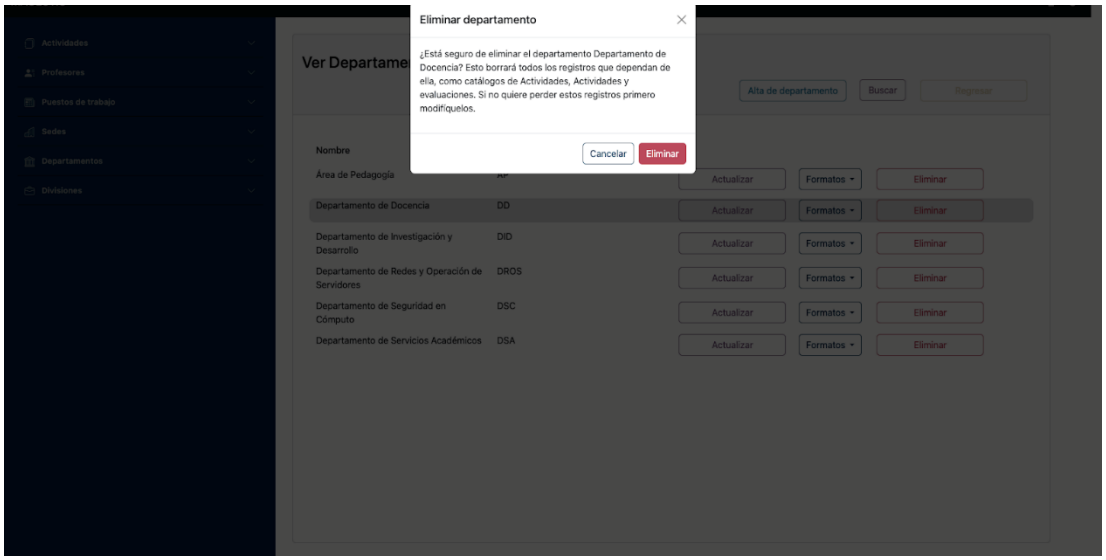


Ilustración 66. Alerta al eliminar un Departamento como usuario administrador.

Si no hubo ningún error al eliminarlo, mandará el mensaje de *Eliminado correctamente* en la vista de *Ver departamentos.*

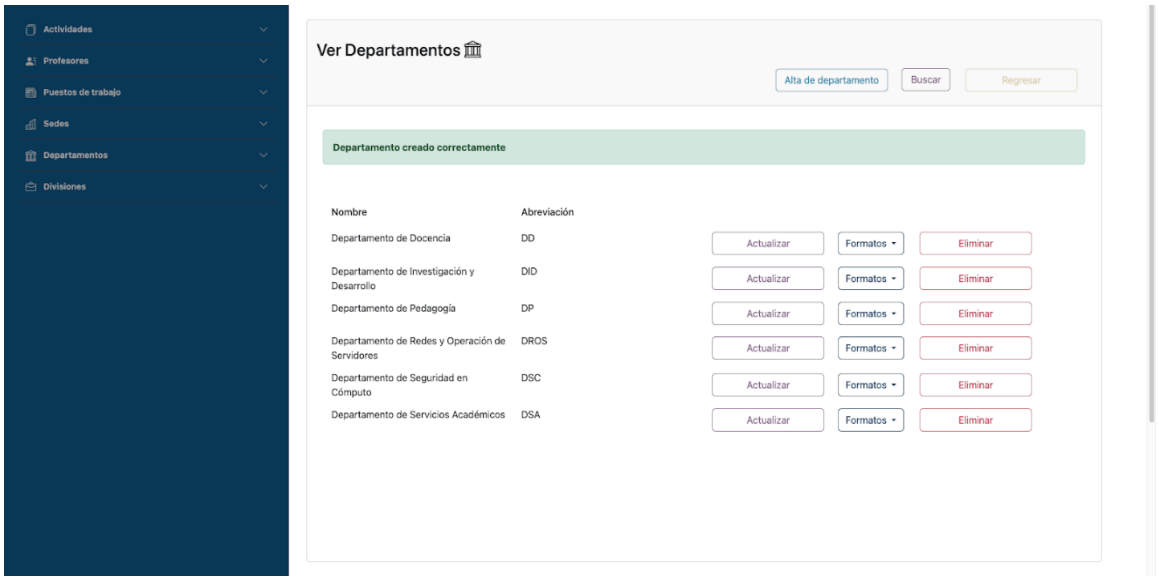


Ilustración 67. Mensaje de Eliminado correctamente

Dentro de la sección de *Departamentos*, también encontraremos la sección de *Administrador*, debido a que se requiere asignar uno o más administradores al dar de alta un departamento nuevo.

Ilustración 68. Opción Alta de Administrador.

Desde esta vista, se pueden agregar los usuarios que ingresen como administradores o ayudantes, estableciendo a qué departamento pertenecen.

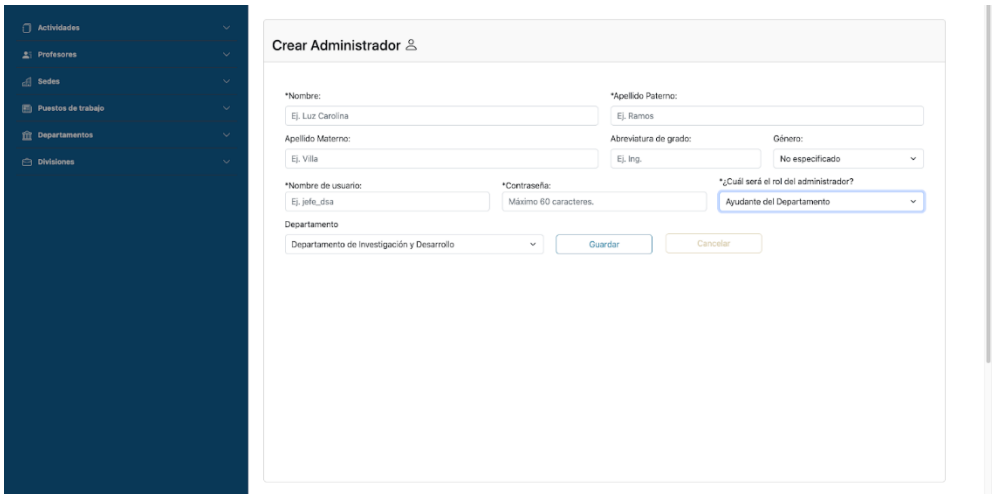


Ilustración 69. Vista Crear Administrador como usuario administrador.

Otra opción que tenemos para visualizar la sección de los administradores es accediendo por el menú izquierdo en *Consulta de administradores,* o bien, en los accesos rápidos dando clic en *Consultar Administradores*.

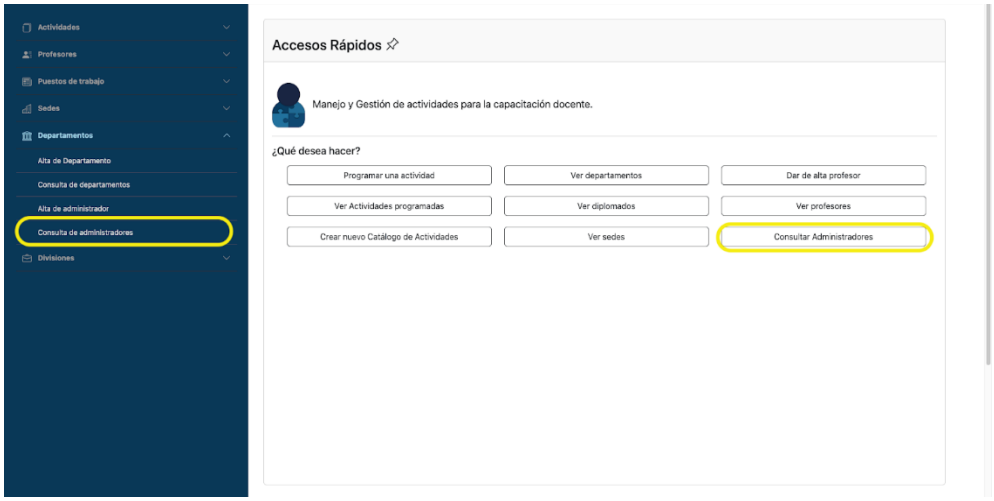


Ilustración 70. Opción Consulta de administradores como usuario administrador.

Además, dentro de esta vista se encuentran las opciones para actualizar y eliminar los datos.

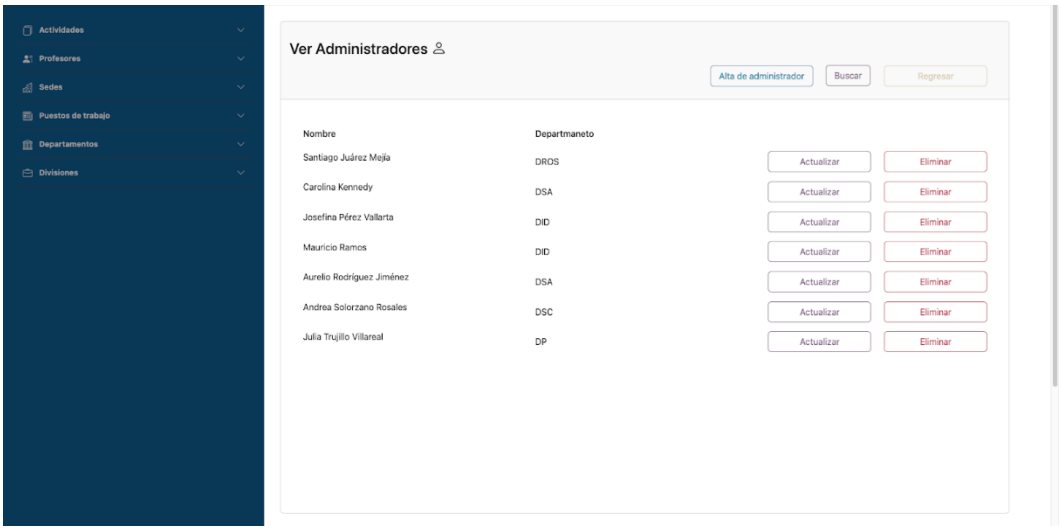


Ilustración 71. Vista Ver Administradores como usuario administrador.

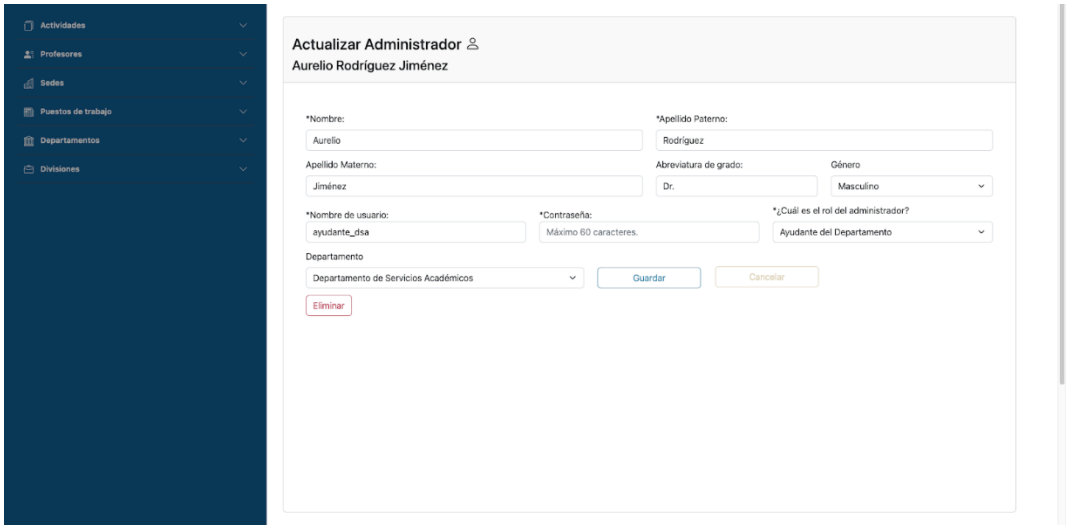


Ilustración 72. Vista Actualizar Administrador como usuario administrador.

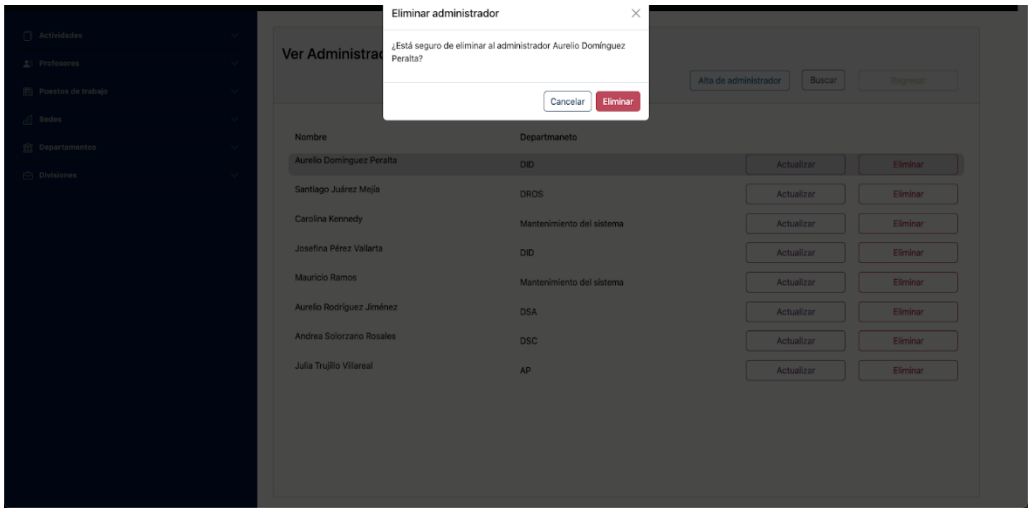


Ilustración 73. Alerta al eliminar un Administrador como usuario administrador.

Una vez teniendo nuestros administradores y departamentos asociados, podemos agregar actividades a nuestro catálogo; para ello, nos dirigimos al apartado de *Actividades* > *Alta de Actividad.* Otra forma de acceder es a través de los accesos rápidos donde se encontrará como *Crear nuevo Catálogo de Actividades*.

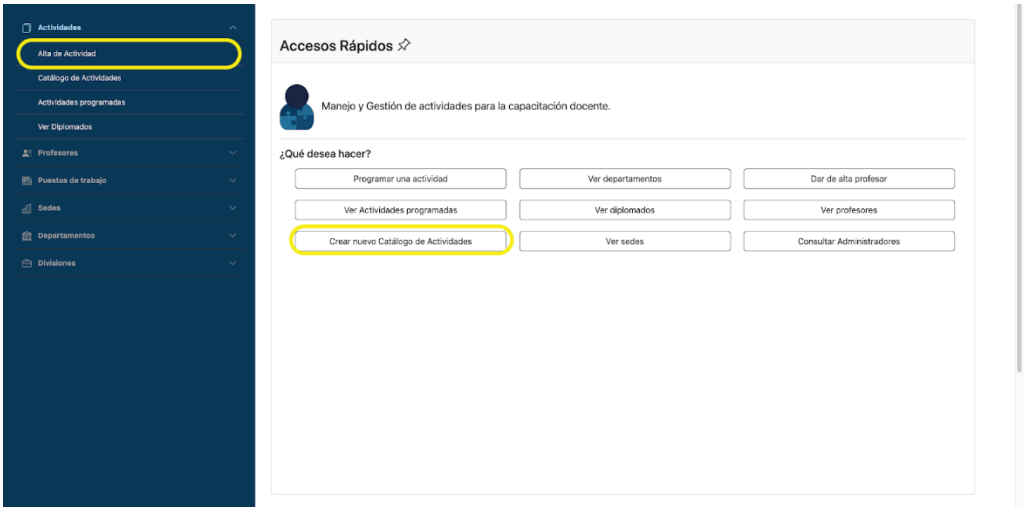


Ilustración 74. Opción Alta de Actividad como usuario administrador.

Dentro de esta vista, además de pedir los datos generales de la actividad, se podrá definir su tipo. Puede ser: Curso, Curso-Taller, Taller, Seminario, Foro, Evento, Conferencia y Módulo de Diplomado.

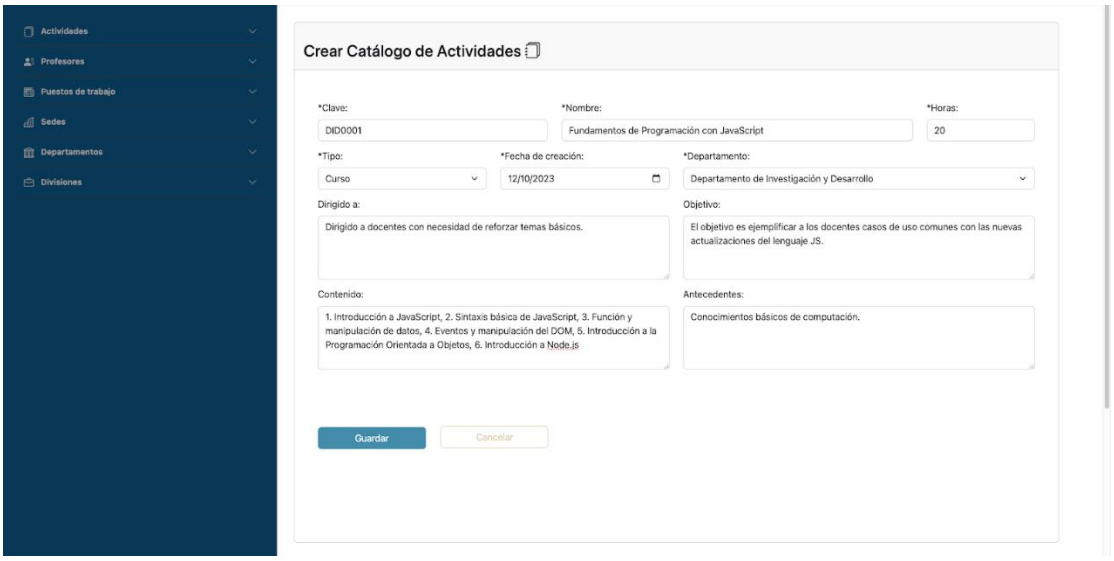


Ilustración 75. Vista Crear Catálogo de Actividades como usuario administrador.

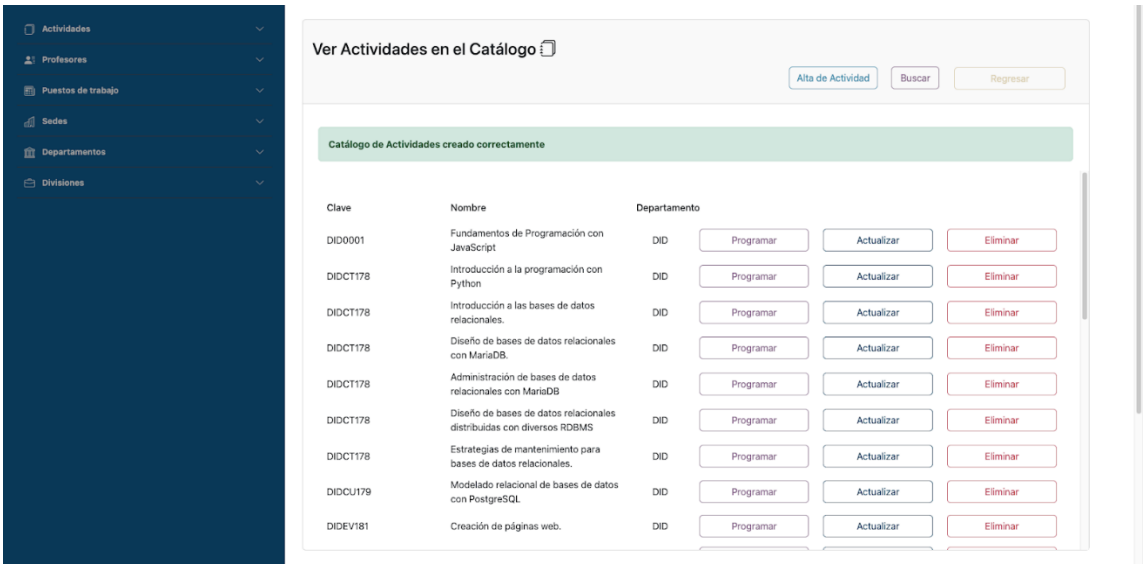


Ilustración 76. Mensaje de Catálogo creado como usuario administrador.

Los *Diplomados* son una sección adicional dentro de las *Actividades*, cuyos módulos están relacionados en el catálogo de actividades.

Por ejemplo, creamos un Diplomado en la sección de *Actividades* > *Ver Diplomados* > *Crear*.

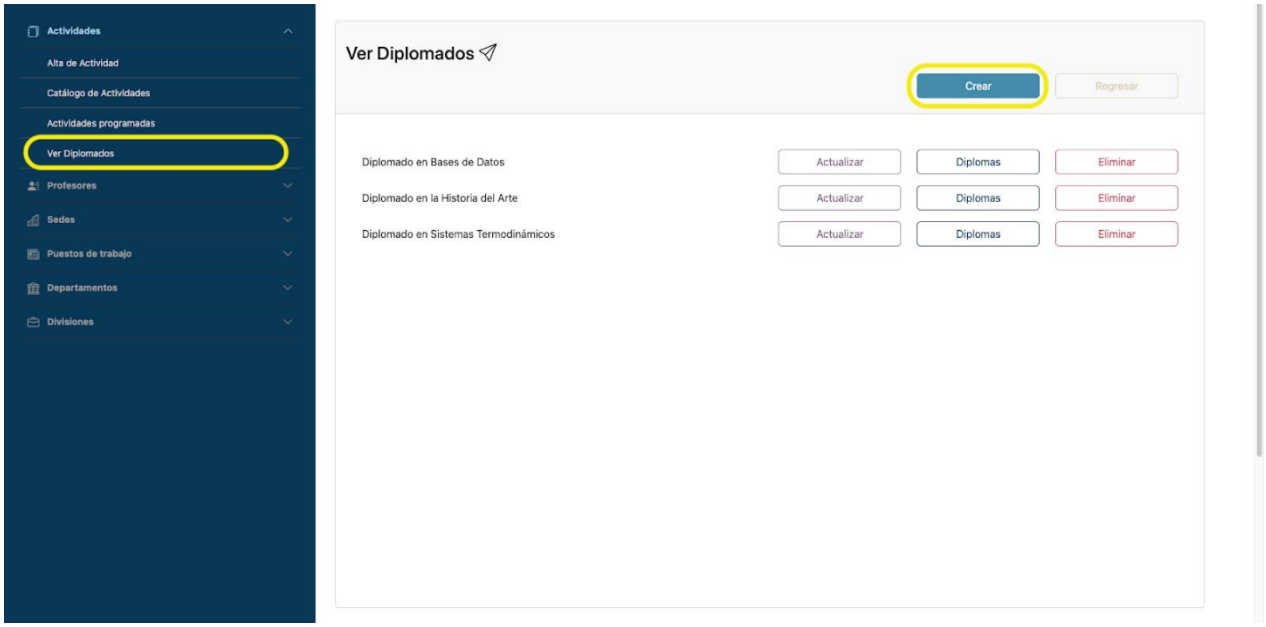


Ilustración 77. Opción Crear Diplomado como usuario administrador.

Se abrirá una nueva sección pidiendo el nombre del diplomado.

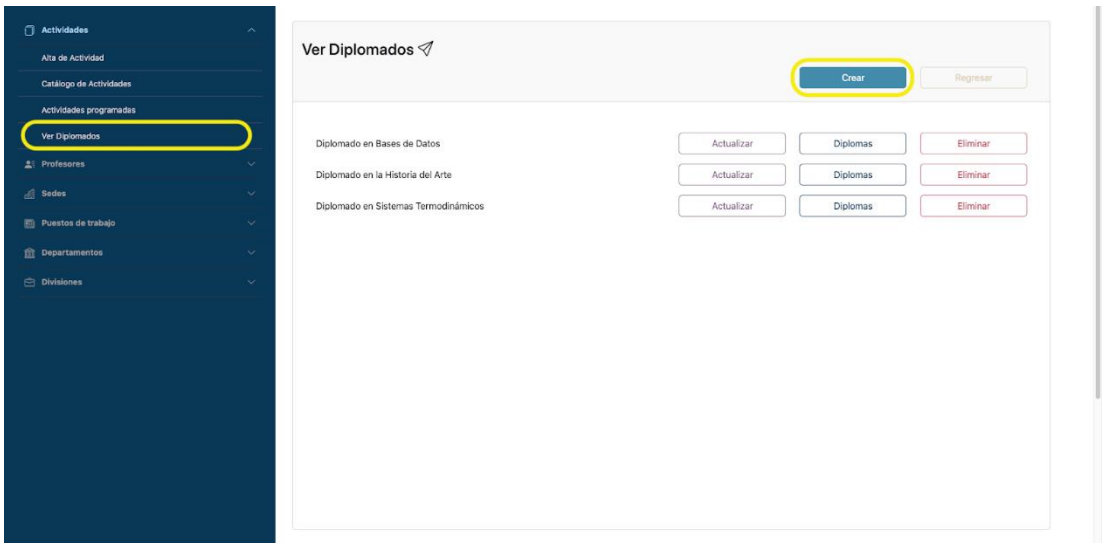


Ilustración 78. Sección Alta Diplomado como usuario administrador.

Una vez creado, procedemos a agregar los módulos desde el catálogo de actividades.

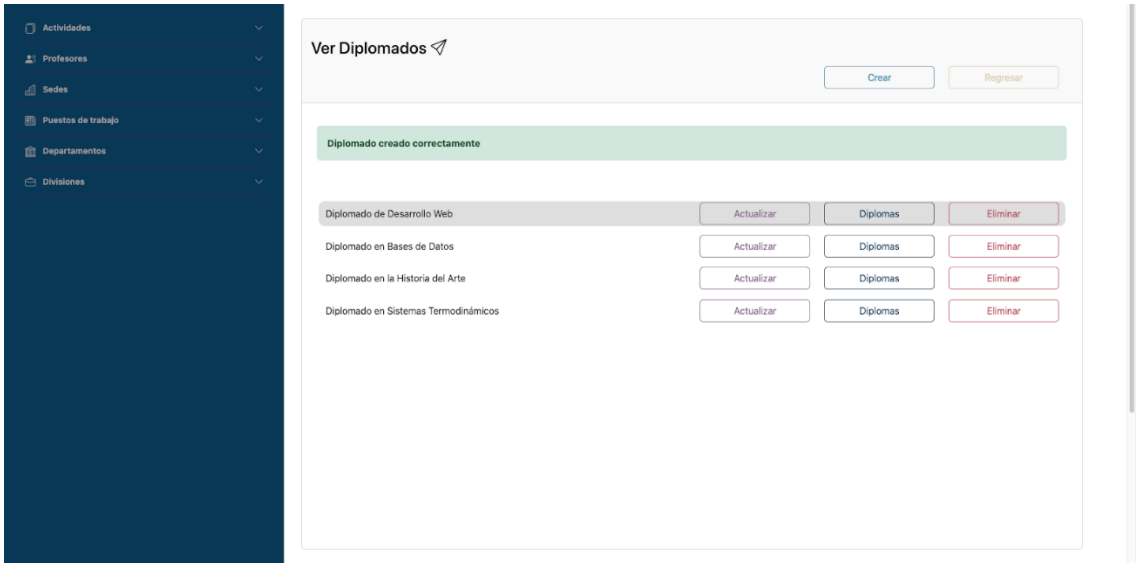


Ilustración 79. Mensaje de Diplomado creado

Para crear o actualizar alguna actividad dentro de nuestro catálogo, sólo se necesita que el tipo sea *Módulo de Diplomado*, y aparecerá una nueva sección pidiendo el *Número de módulo* y a qué diplomado pertenecerá.

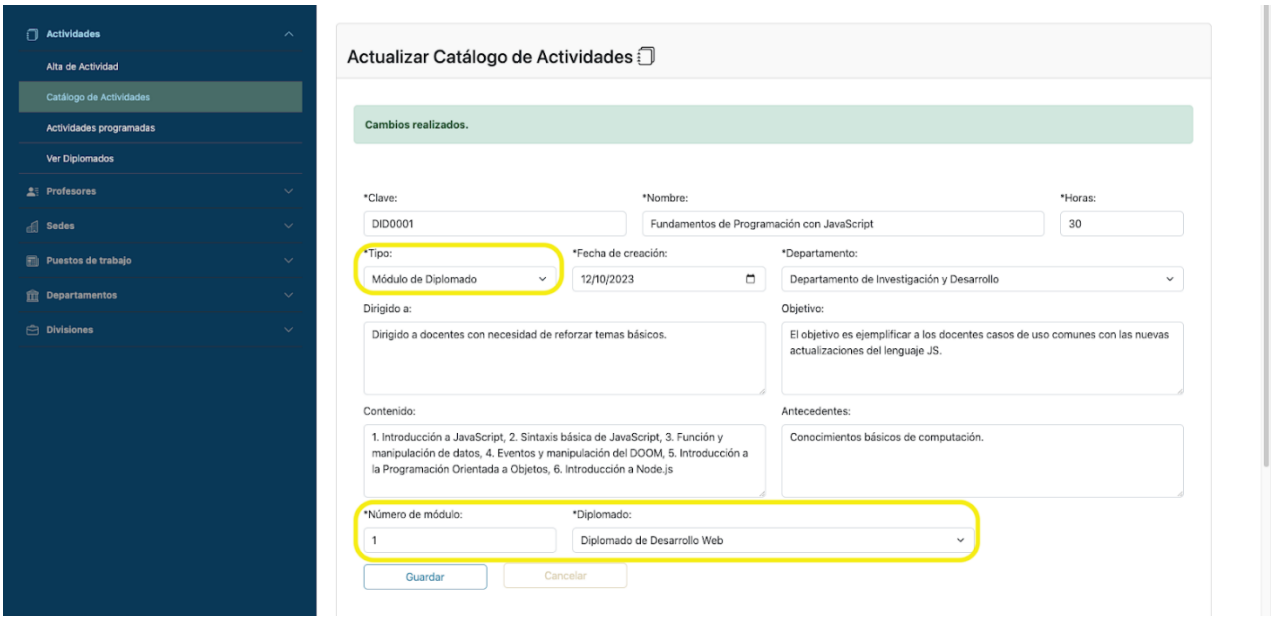


Ilustración 80. Vista Actualizar Catálogo de Actividades como usuario administrador.

Una vez terminado el diplomado y después de haber evaluado a sus participantes, se pueden generar los diplomas correspondientes. Tomaremos como ejemplo el *Diplomado de Bases de Datos*, el cual ya cuenta con evaluaciones.

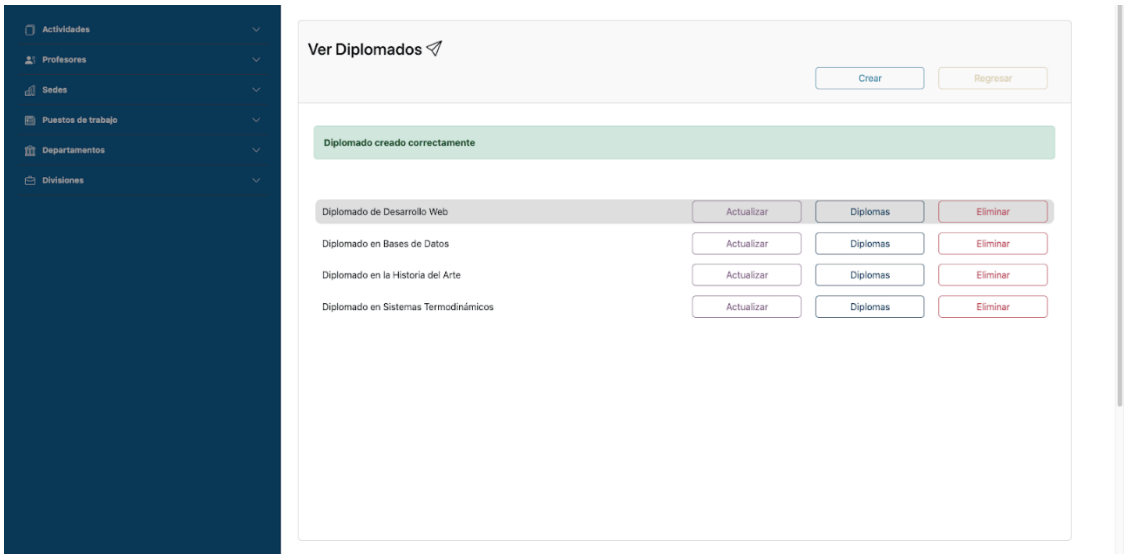


Ilustración 81. Opción para generar diplomas

Se pueden personalizar los diplomas, para ello, se pedirá toda la información requerida antes de crearlos, así como el número de firmantes que llevarán cada uno de ellos.

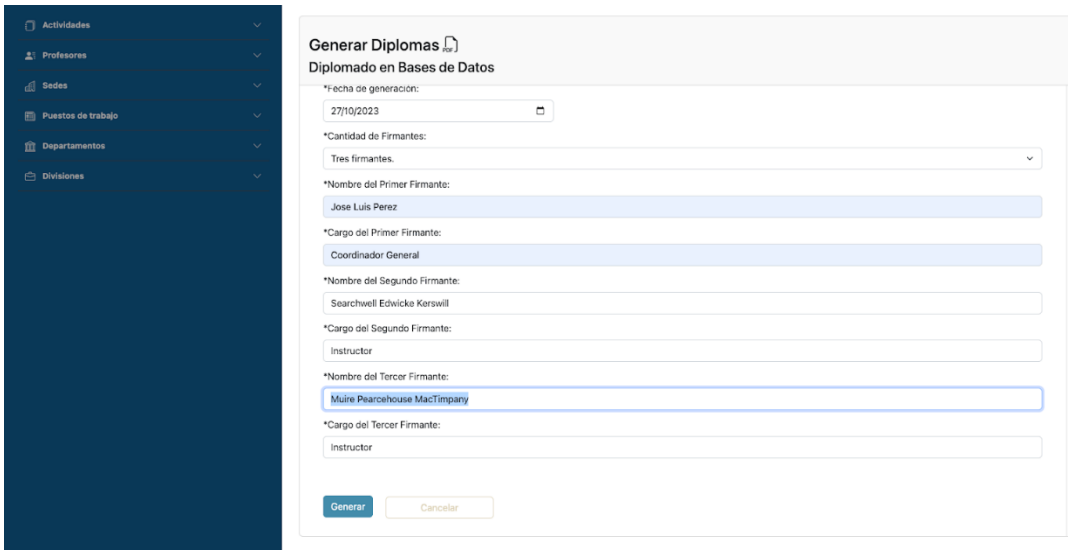
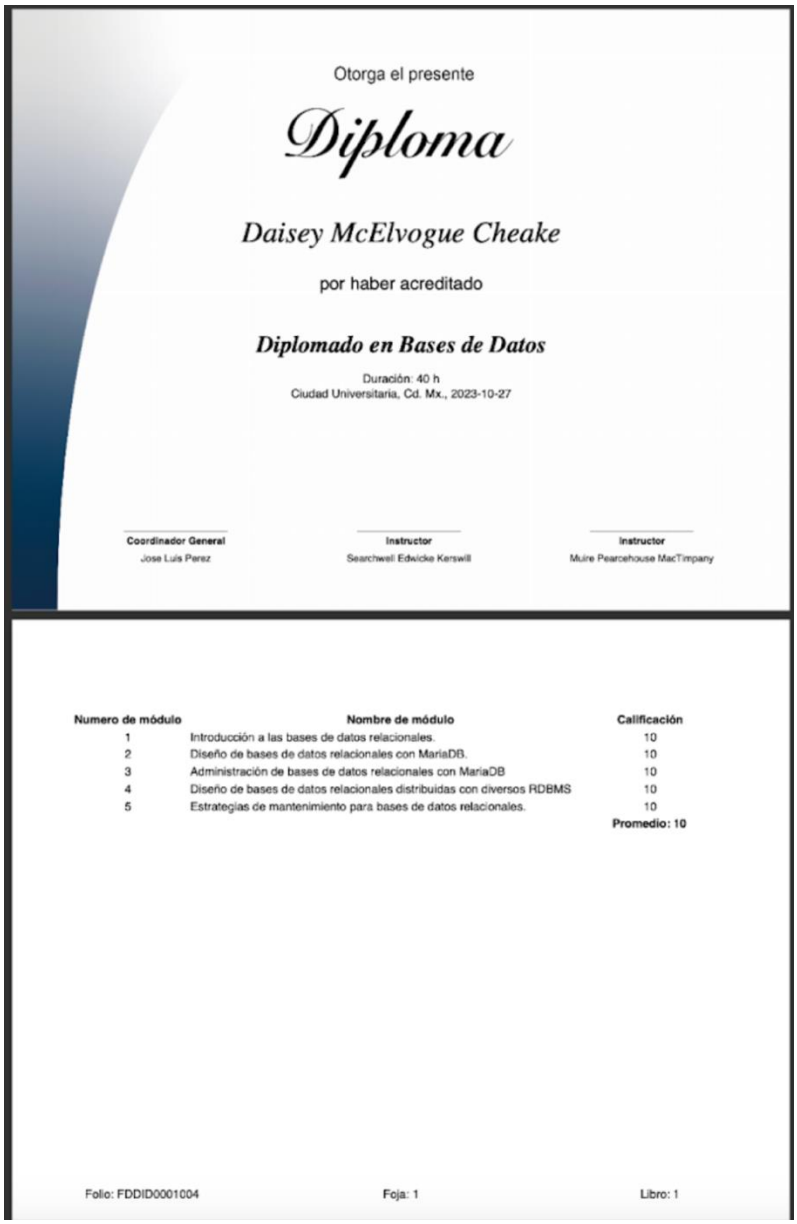


Ilustración 82. Vista Generar Diplomas

Un archivo zip será descargado conteniendo todos los diplomas de los participantes capturados. Además de la información general del diplomado, se incluirá la calificación de cada módulo y el promedio final.



Otorga el presente

Diploma

Daisey McElvogue Cheake

por haber acreditado

Diplomado en Bases de Datos

Duración: 40 h
Ciudad Universitaria, Cd. Mx., 2023-10-27

Coordinador General
Jose Luis Perez

Instructor
Searchwell Edwicke Kerswill

Instructor
Muire Pearcehouse MacTimpany

Numero de módulo	Nombre de módulo	Calificación
1	Introducción a las bases de datos relacionales.	10
2	Diseño de bases de datos relacionales con MariaDB.	10
3	Administración de bases de datos relacionales con MariaDB	10
4	Diseño de bases de datos relacionales distribuidas con diversos RDBMS	10
5	Estrategias de mantenimiento para bases de datos relacionales.	10
		Promedio: 10

Folio: FDDID0001004 Foja: 1 Libro: 1

Ilustración 83. Ejemplo de Diploma

Regresemos al ejemplo de un curso y actualicemos su catálogo.

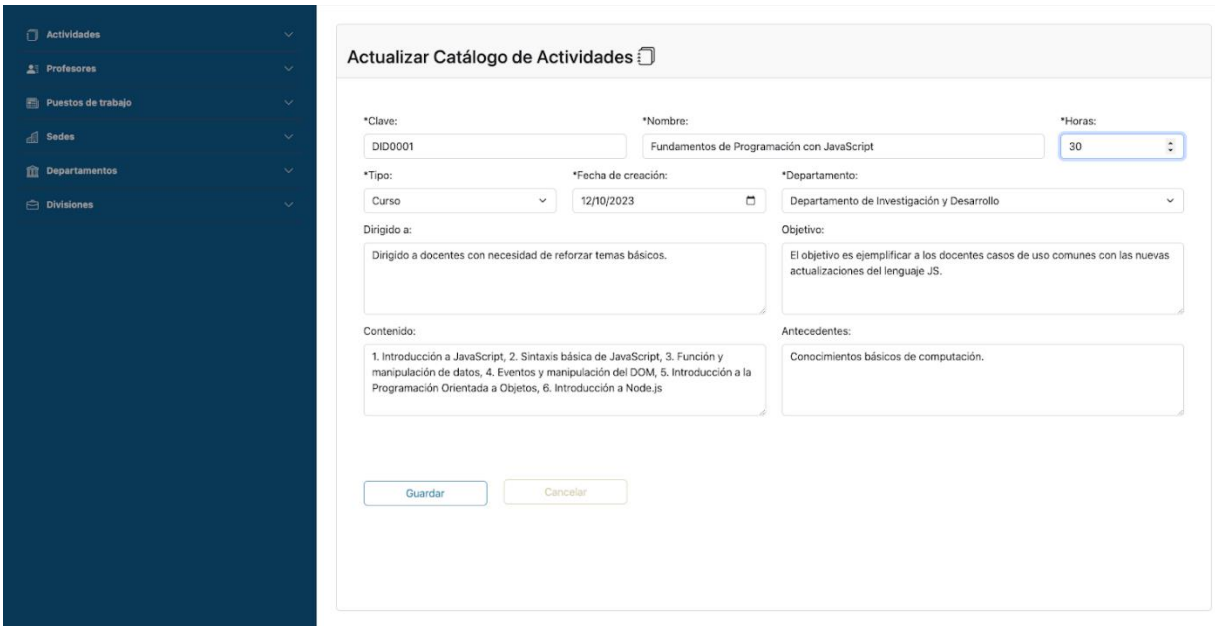


Ilustración 84. Modificando Catálogo de Actividades como usuario administrador.

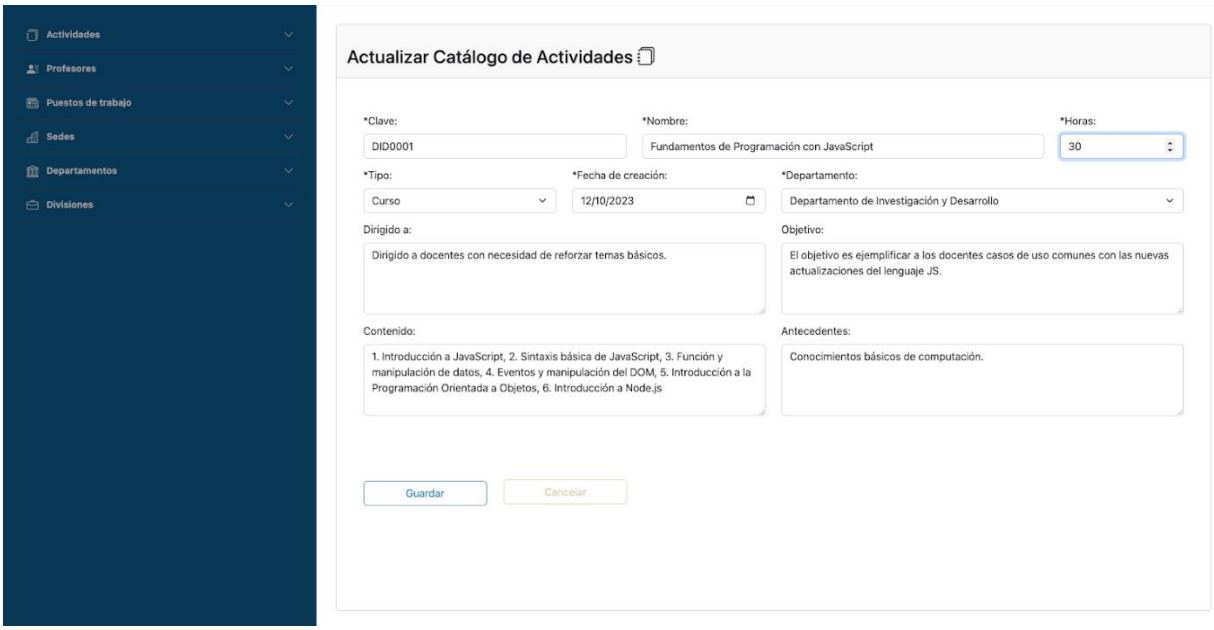


Ilustración 85. Mensaje de Cambios realizados

Para definir las fechas y el periodo de las actividades, necesitamos programarlo desde el catálogo.

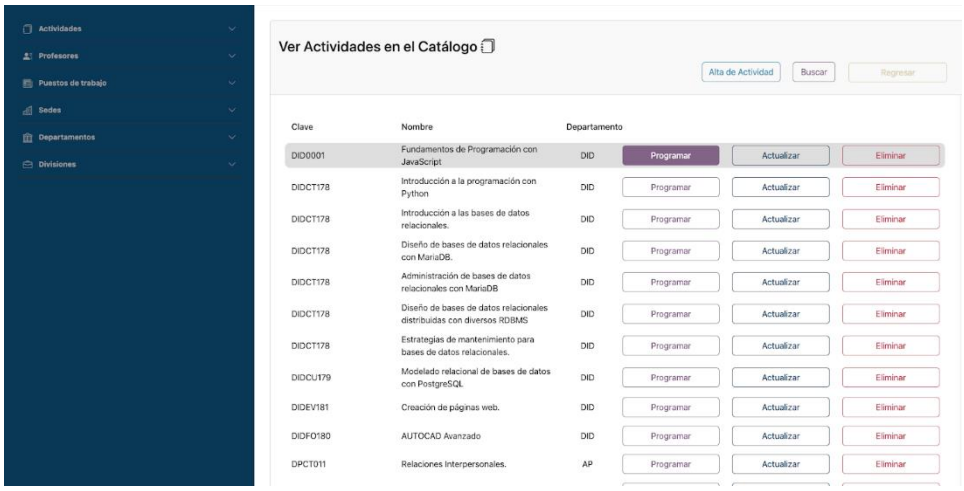


Ilustración 86. Opción de Programar Catálogo como usuario administrador.

Aquí especificaremos las fechas, el cupo, la sede donde se impartirá, los criterios de evaluación y el costo de la actividad.

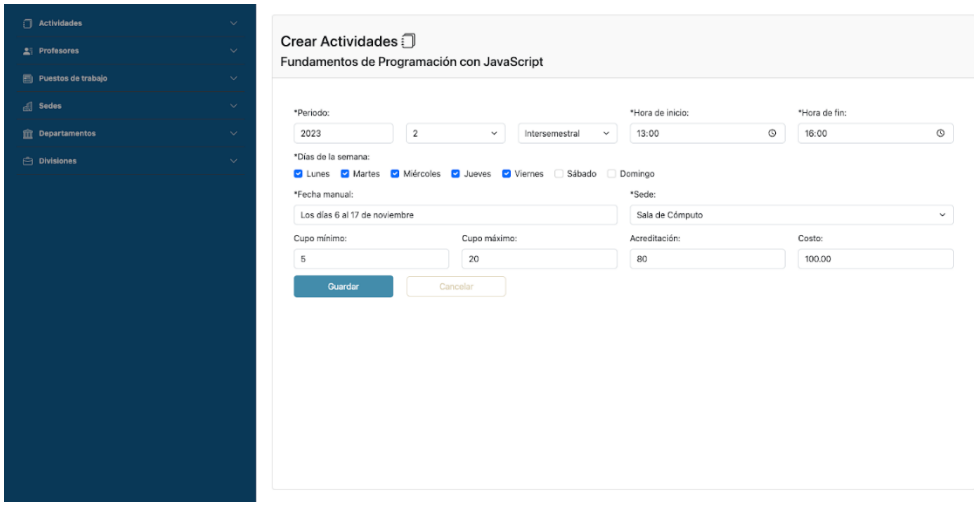


Ilustración 87. Vista Crear Actividad como usuario administrador.

Cuando guardemos los cambios, nos redirigirá a la vista de *Actividades Programadas*.

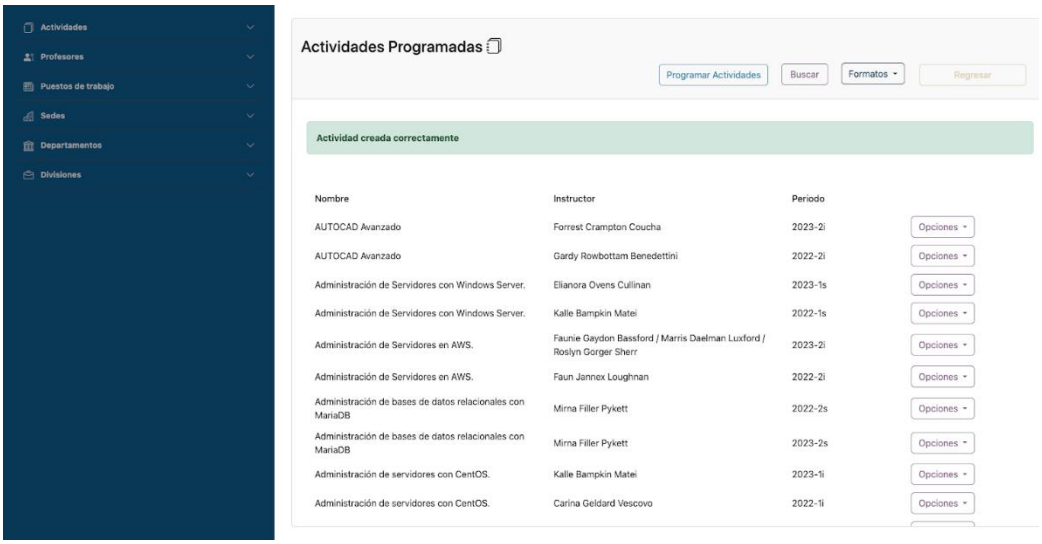


Ilustración 88. Mensaje de Actividad creada

En cada sección existirá un buscador para facilitar la obtención de los datos; en este caso se filtra por *nombre, instructor o periodo.*

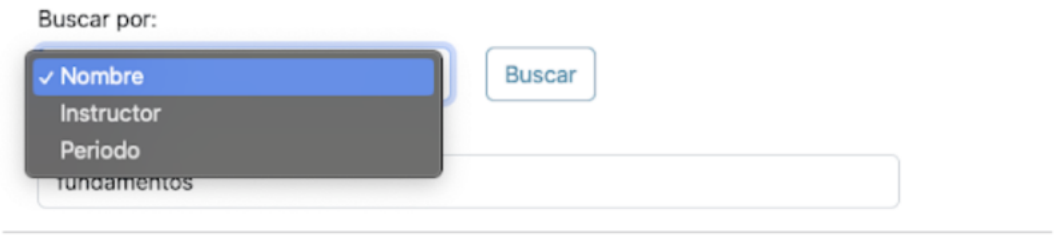


Ilustración 89. Opciones de búsqueda

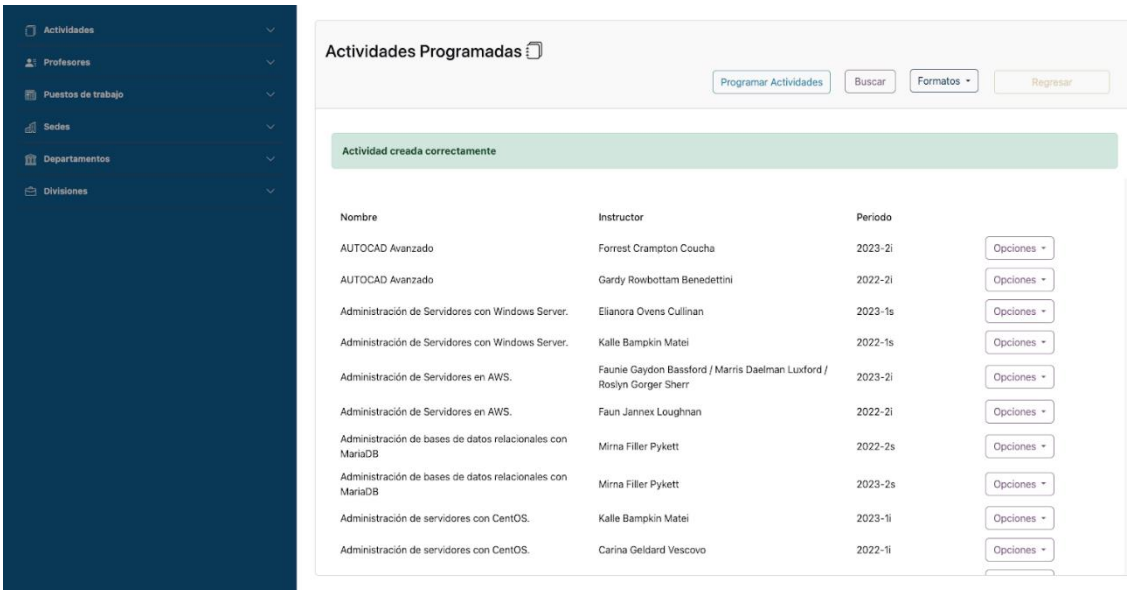


Ilustración 90. Ingresando curso a buscar

El buscador nos regresará todos los resultados con nombres similares o los que estén relacionados al periodo seleccionado.

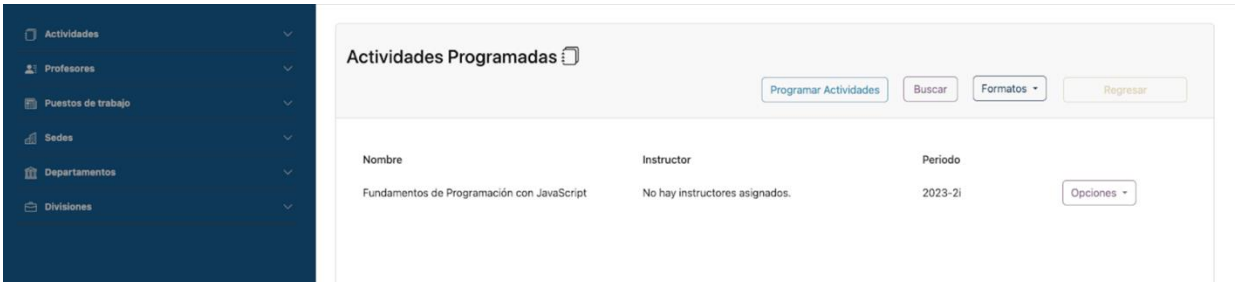


Ilustración 91. Resultados de búsqueda

También en esta sección de *Actividades Programadas* podemos generar diferentes formatos.

Ilustración 92. Formatos de Actividades Programadas como usuario administrador.

El formato de *Exportación* es un documento en Excel que contiene la mayor parte de información de las actividades en la BD.

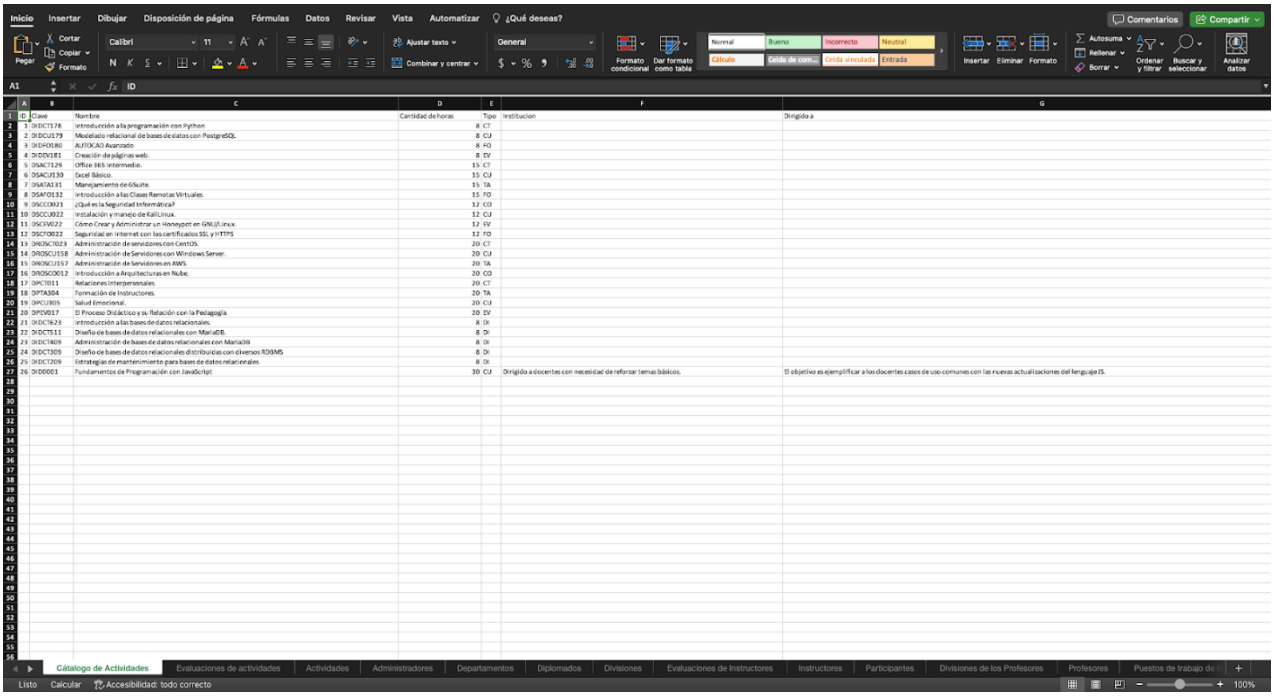


Ilustración 93. Formato de Exportación

El *"libro de folios"* enlista en una tabla los folios de constancias y reconocimientos existentes en la BD asociados a su actividad y participantes o instructores.

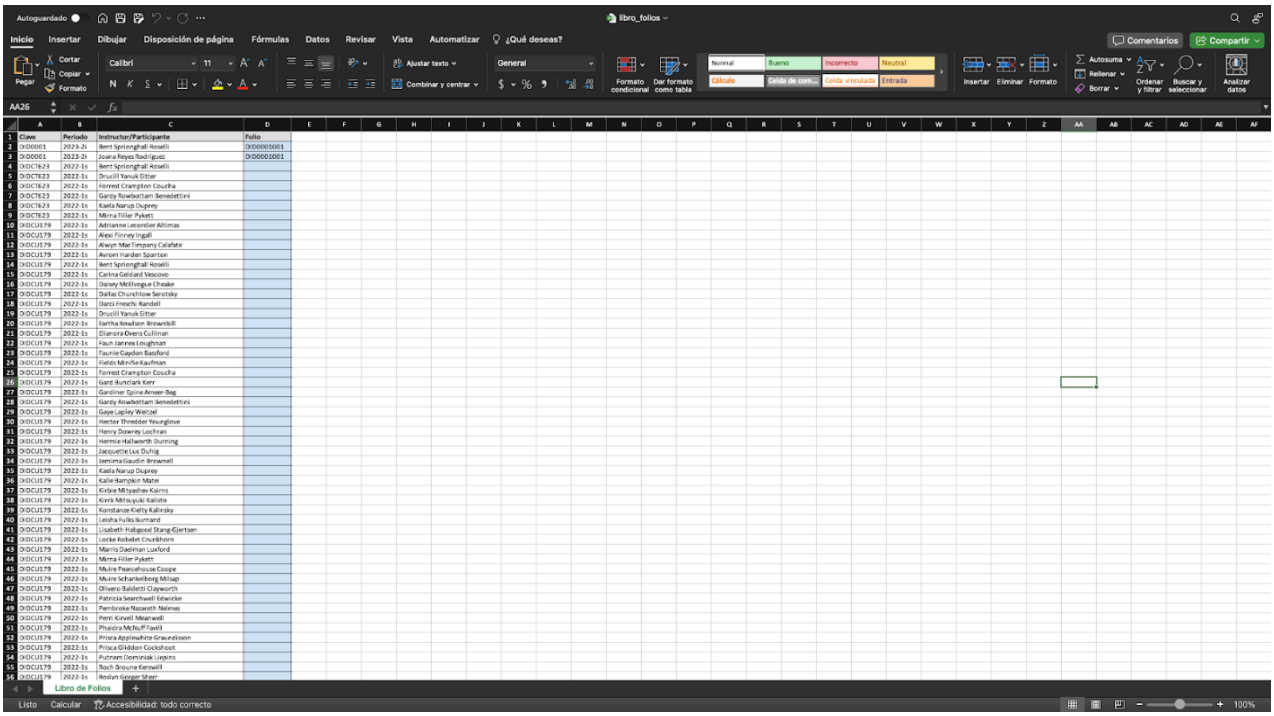


Ilustración 94. Formato Libro de Folios

El *"reporte general de actividades"* enlista todas las actividades en un periodo dado con la información de las fechas, horarios y sedes.

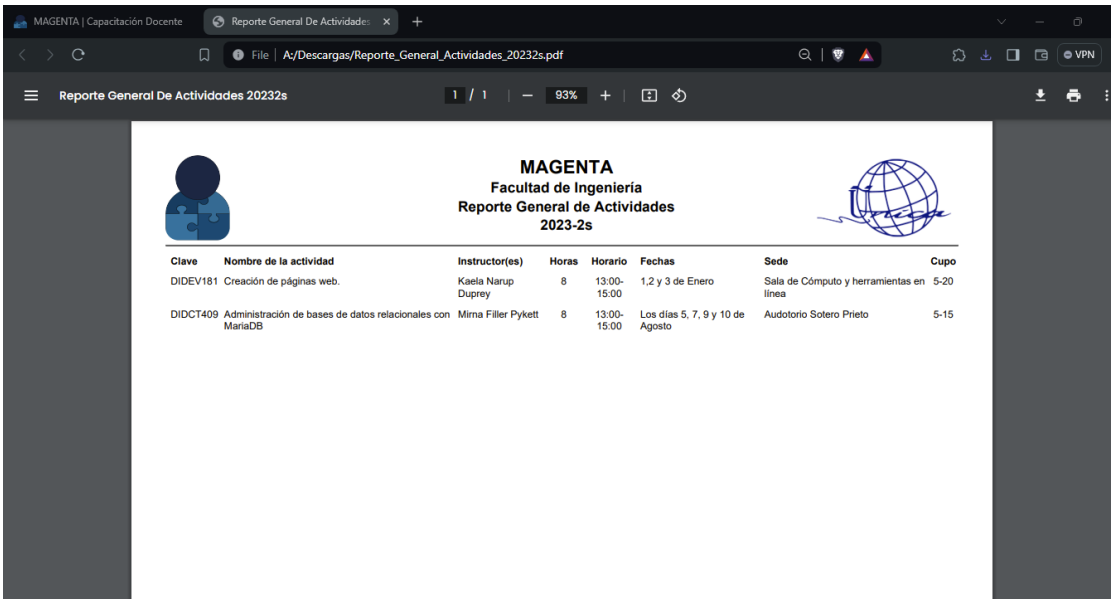


Ilustración 95. Reporte General de Actividades

El *"reporte de sugerencias"* enlista por actividad todos los comentarios de los participantes en un periodo específico.

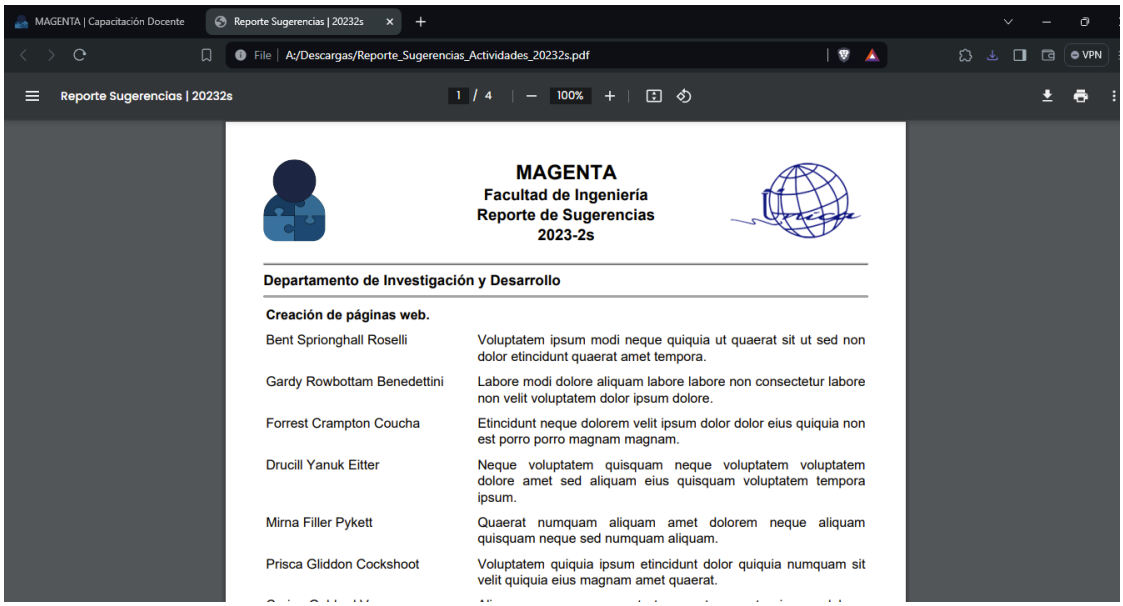


Ilustración 96. Reporte de Sugerencias

Como nuestro curso es nuevo, falta asignarles instructores. Podemos dar de alta nuevos instructores desde *Alta de Profesor*, ya sea en el menú izquierdo o en los accesos rápidos.

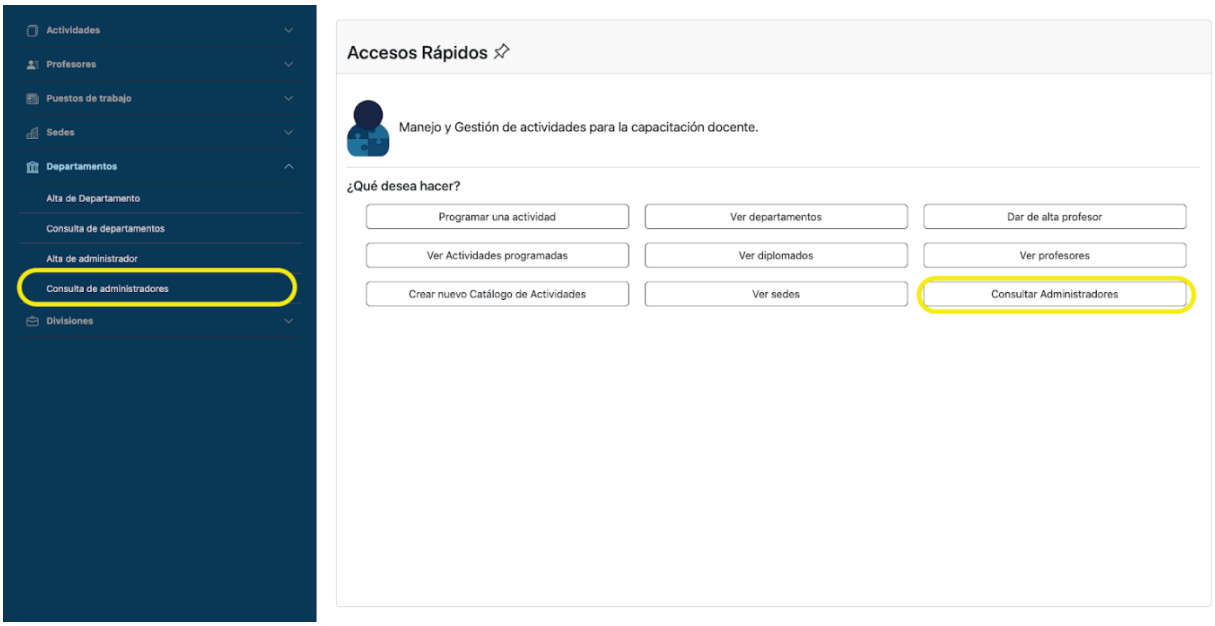


Ilustración 97. Opción Alta de Profesor como usuario administrador.

Dentro de esta vista, además de llenar todos sus datos, se debe especificar que *sí es instructor*, para que el sistema lo detecte al momento de asignarlo a la/las actividades.

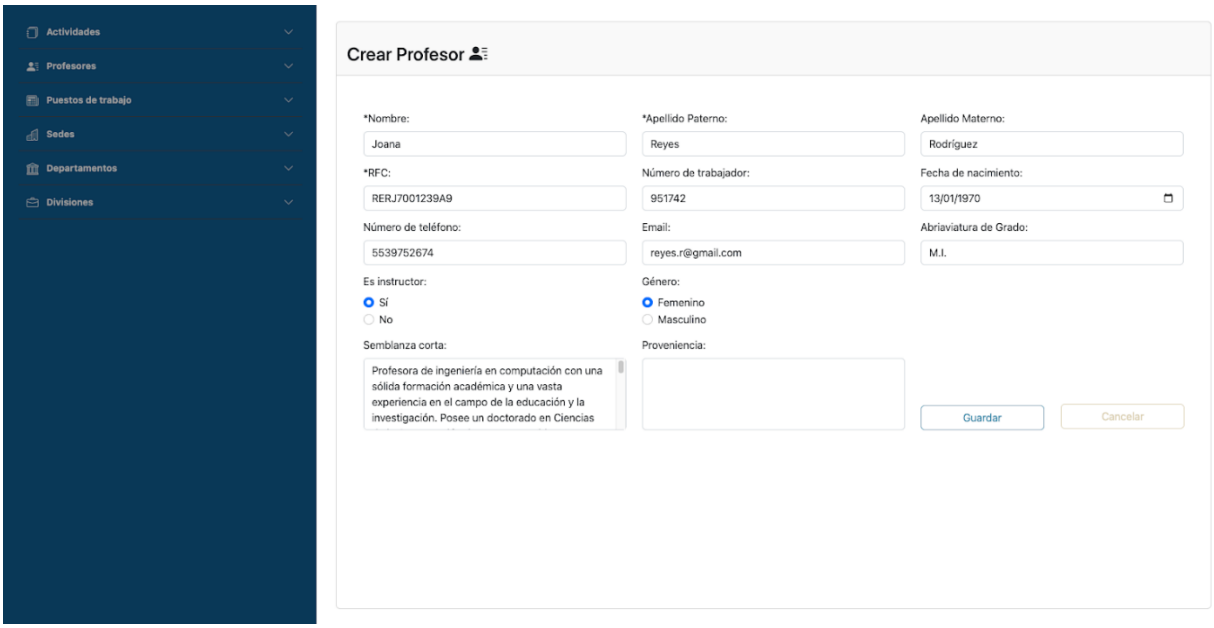


Ilustración 98. Creación de Profesor como usuario administrador.

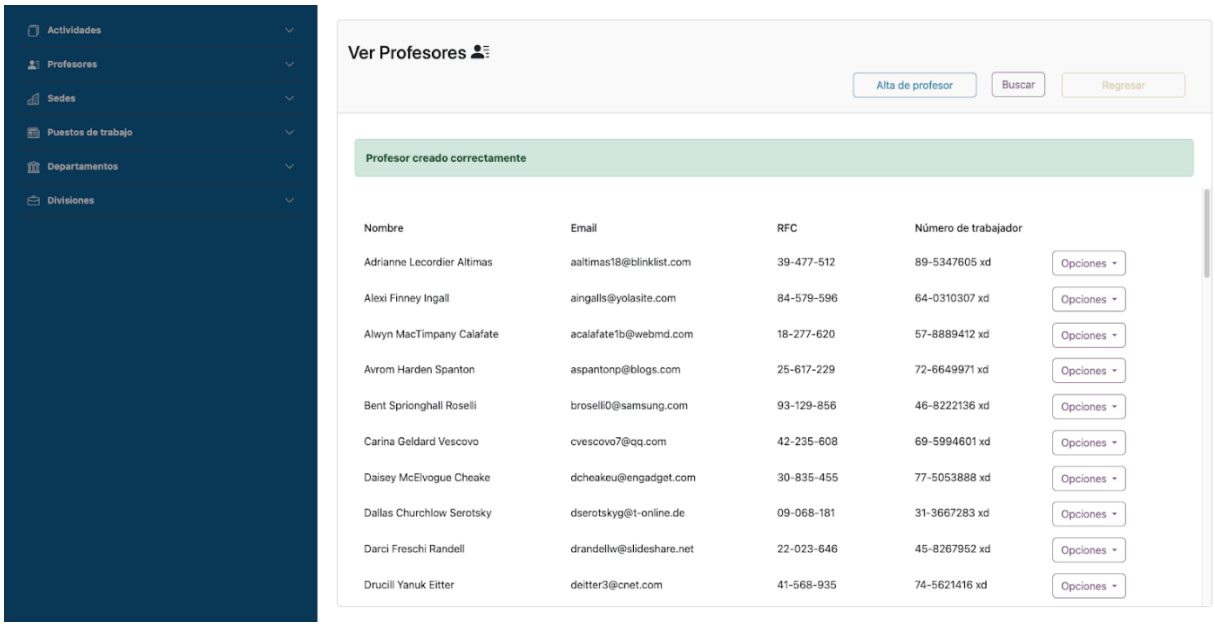


Ilustración 99. Mensaje de Profesor creado

Al profesor creado podemos asignarle divisiones que estén registradas en el sistema.

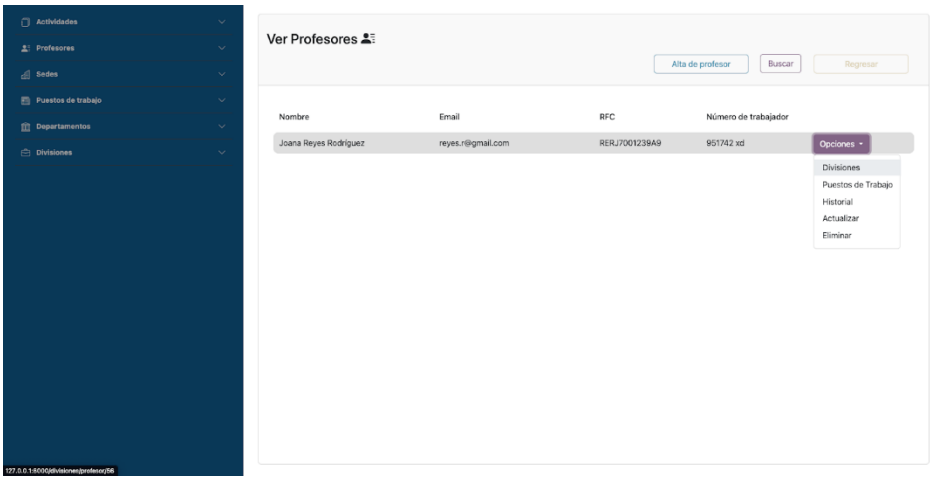


Ilustración 100. Opción de asignar Divisiones a Profesor como usuario administrador.

Si es la primera vez, aparecerá el siguiente mensaje: *No hay divisiones asignadas al profesor en la base de datos,* entonces, daremos clic en el botón de *Asignar*.

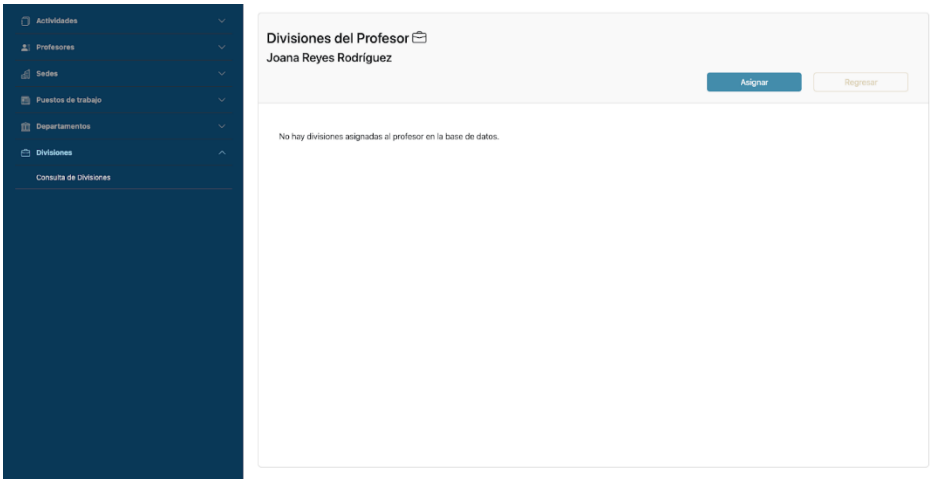


Ilustración 101. Vista Divisiones del Profesor y botón de Asignar como usuario administrador.

Nos aparecerá un selector con las divisiones disponibles en la base de datos y daremos clic en *Guardar.*

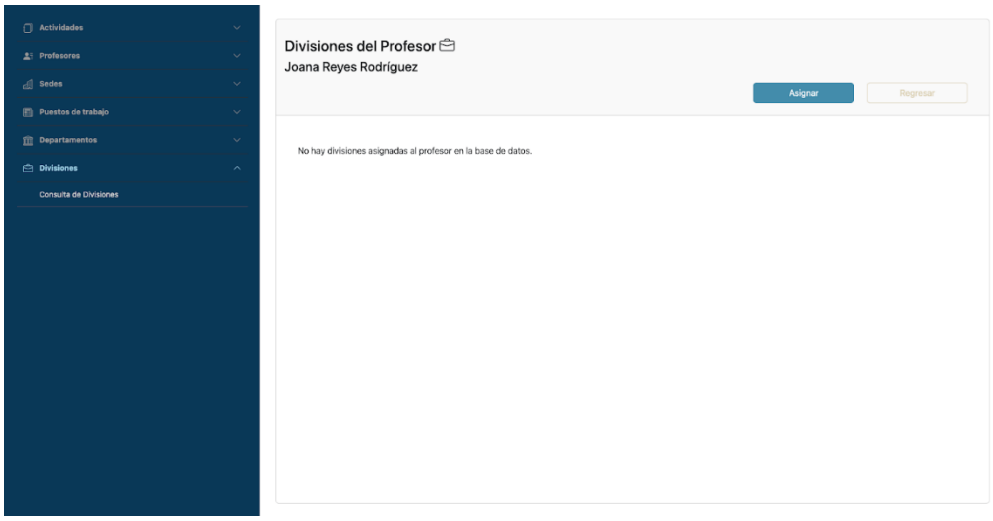


Ilustración 102. Divisiones disponibles

Se mandará un mensaje de que se asignó correctamente. Será en esta vista donde se puedan agregar todas las divisiones relacionadas con el profesor.

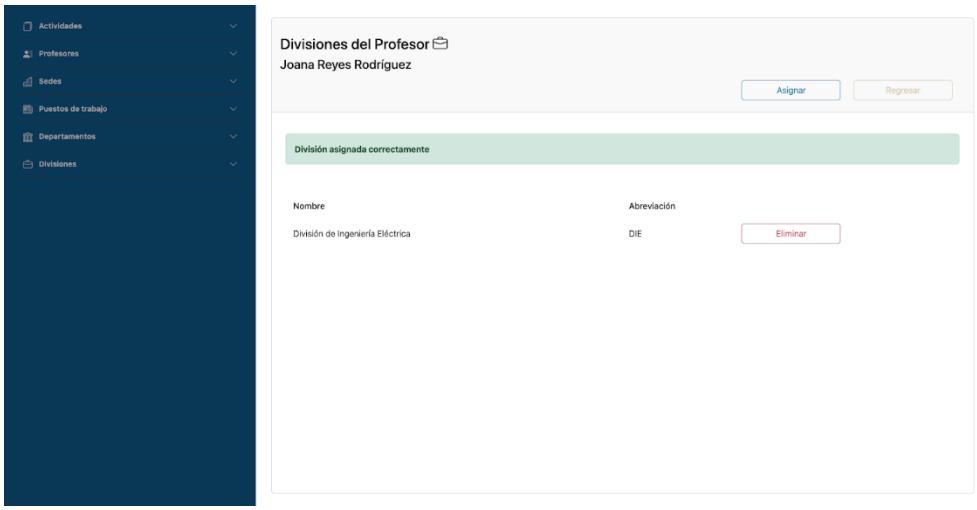


Ilustración 103. Mensaje de División de Profesor asignada

Las divisiones se pueden gestionar desde la sección de *Divisiones* > *Consulta de Divisiones*.

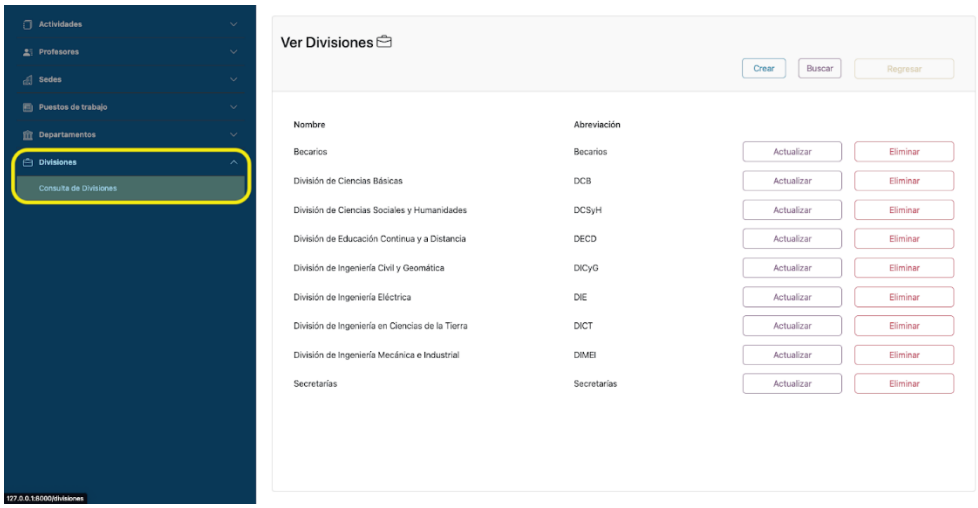


Ilustración 104. Opción Consulta de Divisiones como usuario administrador.

De igual manera, se pueden asignar los puestos de trabajo del profesor que tiene dentro de la institución.

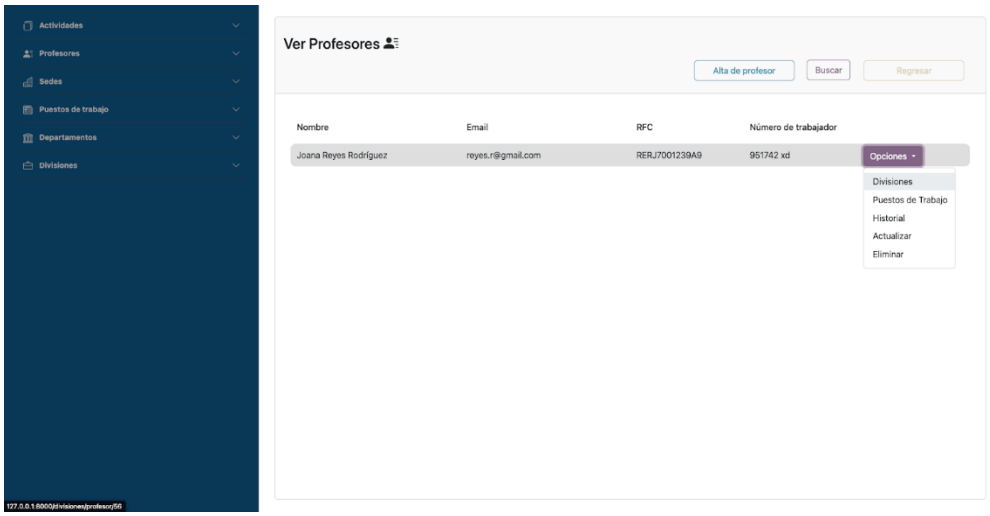


Ilustración 105. Opción de asignar Puesto de Trabajo a Profesor como usuario administrador.

La vista es similar a las de *divisiones* y el proceso para asignarlos es el mismo.

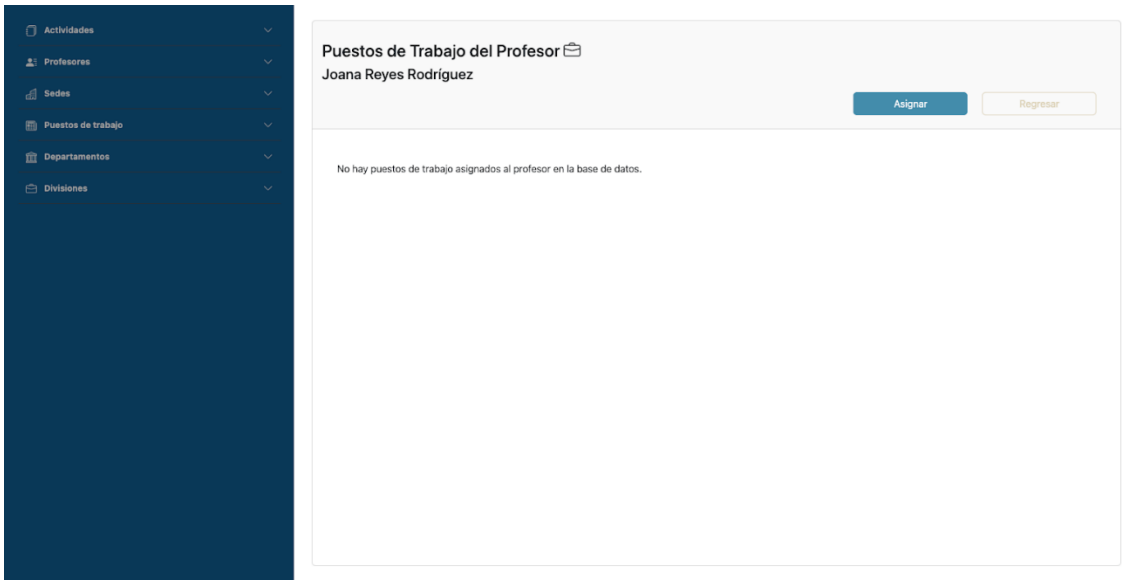


Ilustración 106. Vista de Puestos de Trabajo a Profesor y botón de Asignar como usuario administrador.

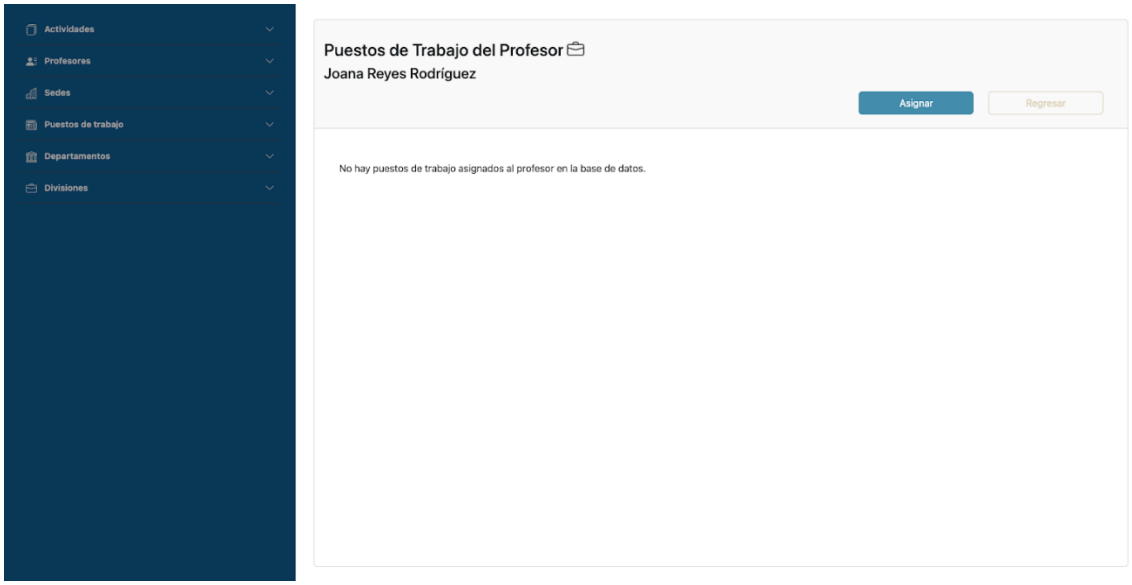


Ilustración 107. Puestos de Trabajo disponibles

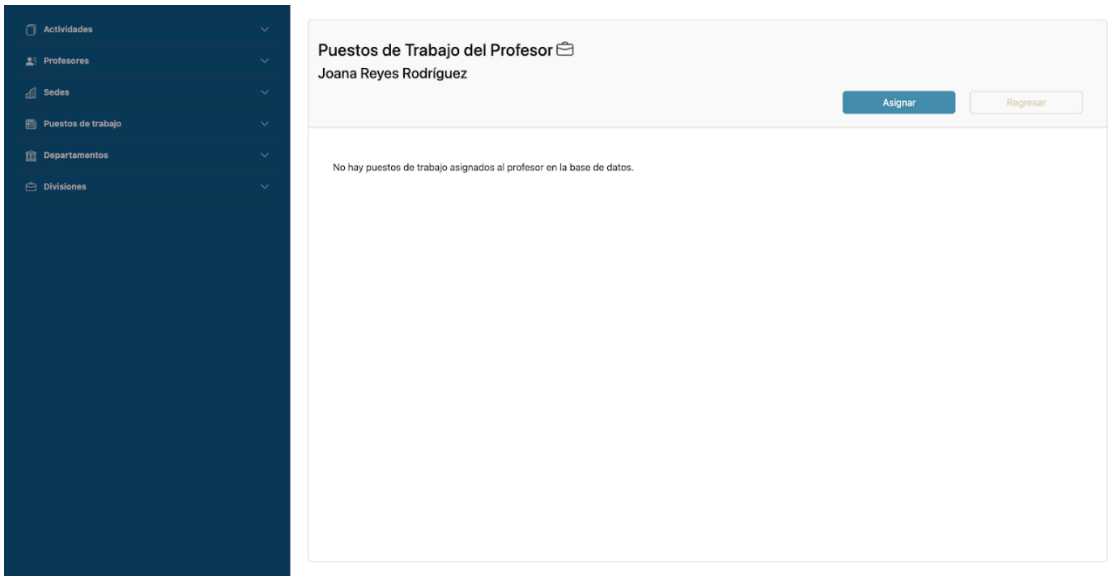


Ilustración 108. Mensaje de Puesto de Trabajo asignado

Asimismo, podemos gestionar estos datos desde la sección de *Puestos de trabajo > Consulta de Puestos de Trabajo*.

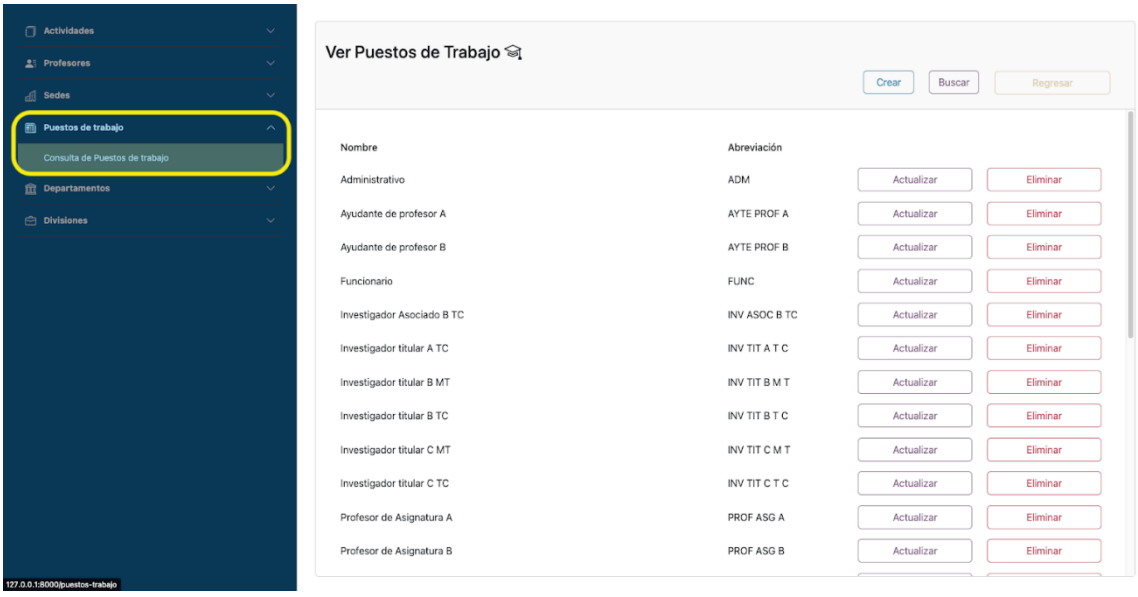


Ilustración 109. Opción Consulta de Puestos de Trabajo como usuario administrador.

En la sección de *Profesores*, también se puede generar un formato que muestre su historial con todas las actividades que ha aprobado dentro de la organización.

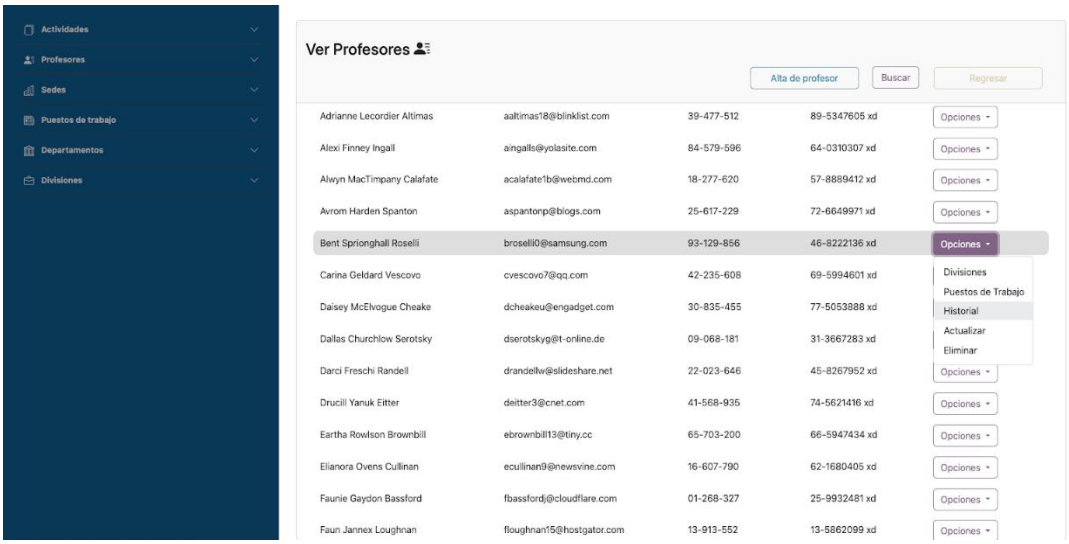


Ilustración 110. Opción Historial del Profesor como usuario administrador.

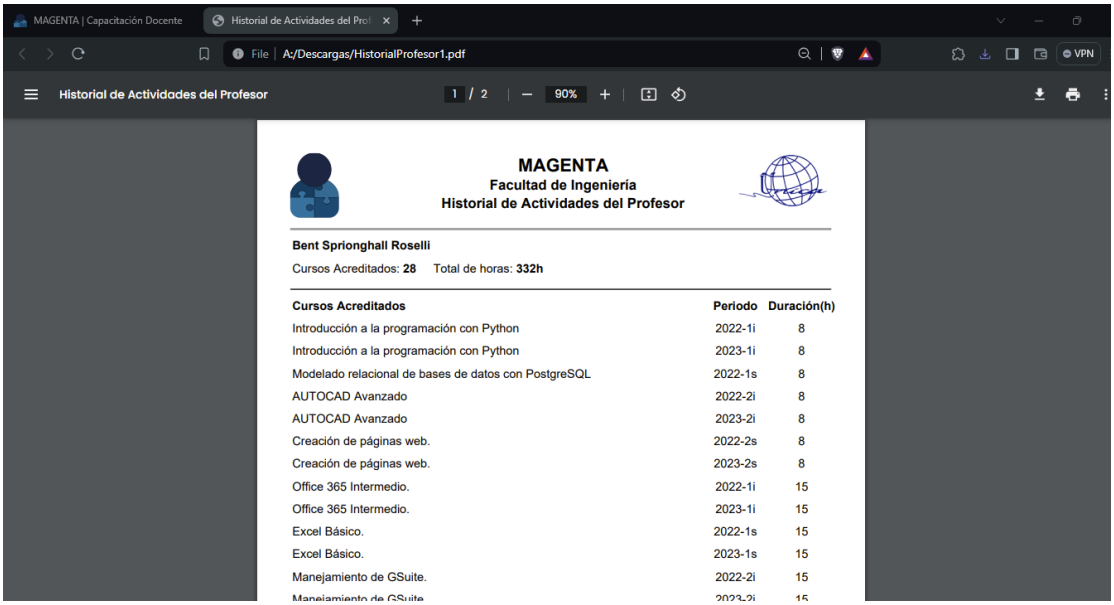


Ilustración 111. Historial del Profesor

Ahora que ya creamos a nuestro profesor, podemos asignarlo como instructor en el curso que creamos. Nos dirigimos a *Actividades* > *Actividades*

Programadas y buscamos nuestro curso para después seleccionar en las opciones *Instructores*.

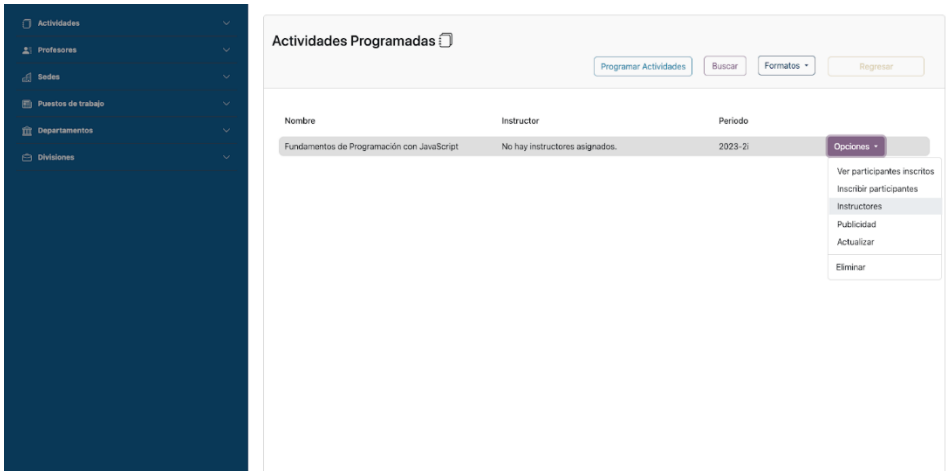


Ilustración 112. Opción de asignar Instructores a Actividades como usuario administrador.

Buscaremos al instructor creado.

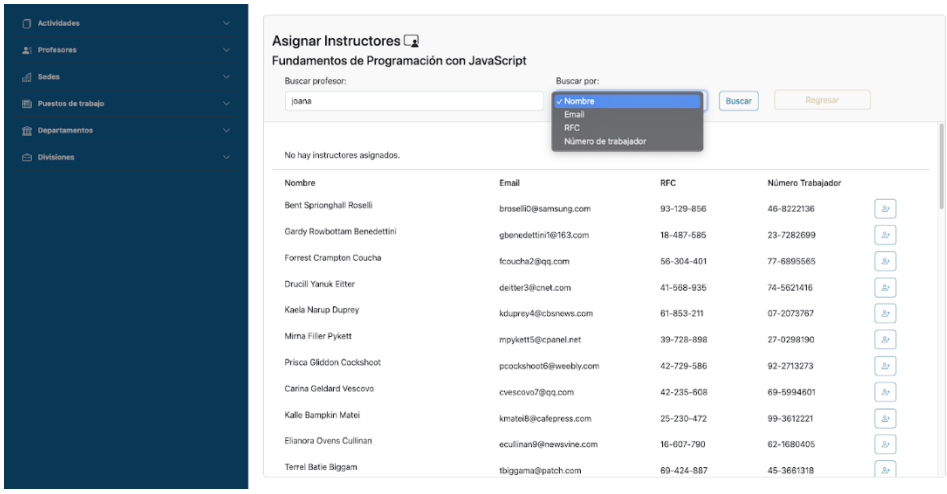


Ilustración 113. Búsqueda de Instructor

Y daremos clic en el ícono de *agregar*.

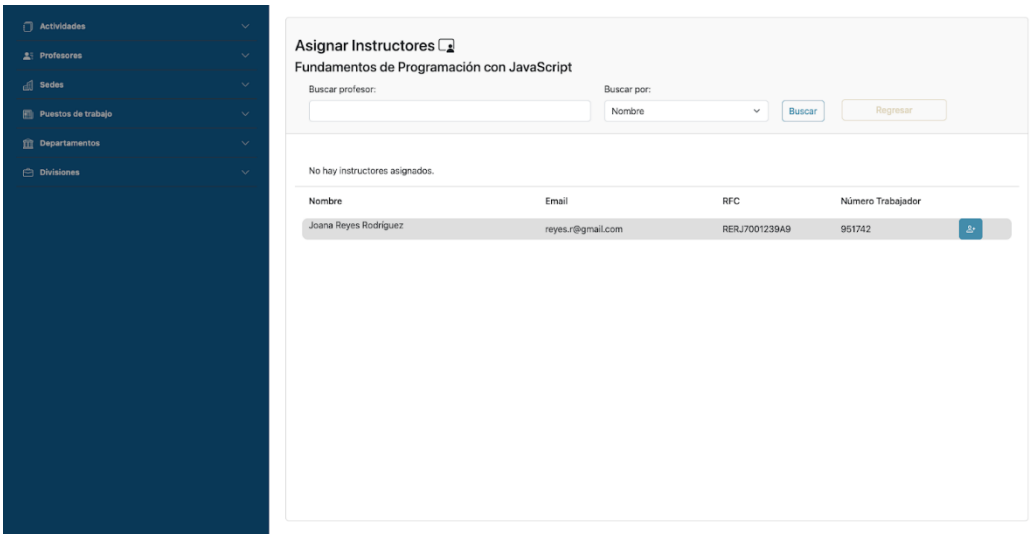


Ilustración 114. Ícono de agregar Instructor

Cuando agregamos instructores, éstos aparecerán al inicio de la lista y tendrá la opción de eliminarlos de esa actividad en concreto.

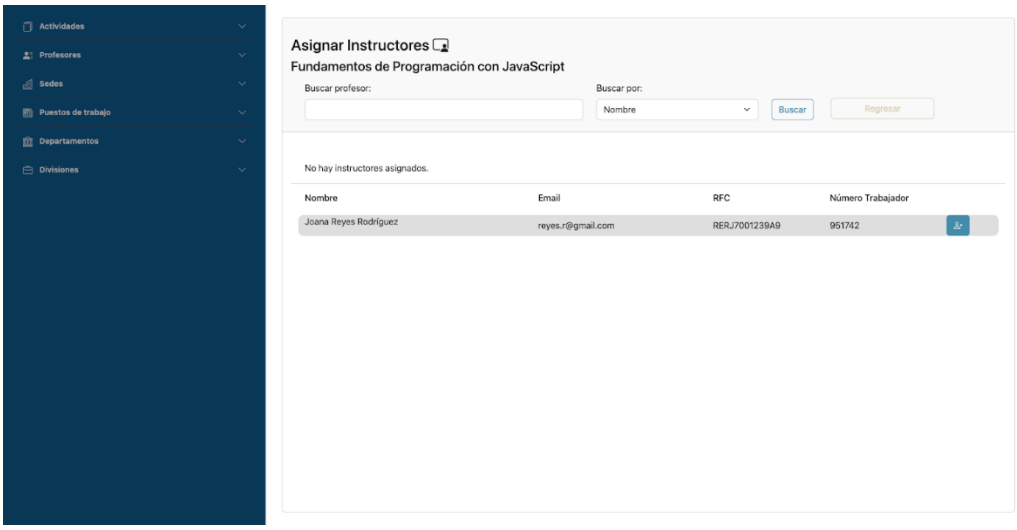


Ilustración 115. Ícono de Eliminar Instructor

Otro rubro dentro de la actividad programada es el documento de *Publicidad*.

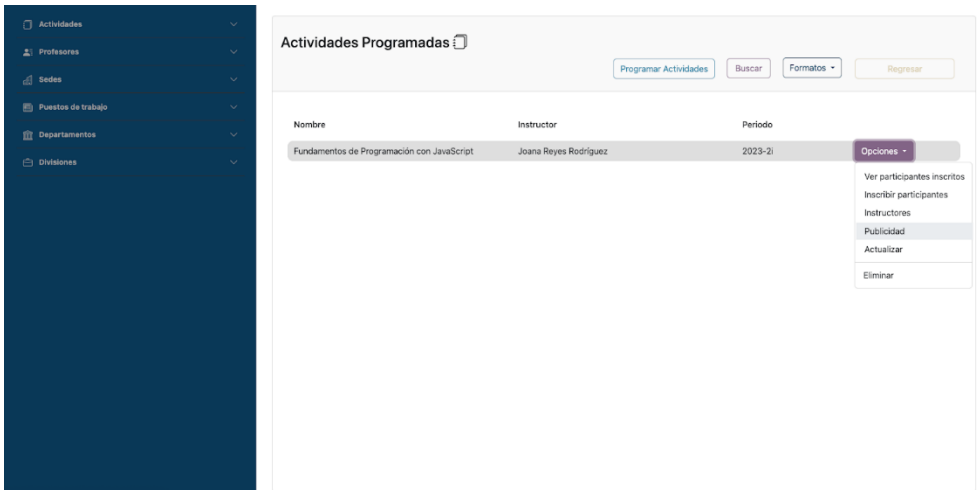


Ilustración 116. Opción para generar documento de Publicidad

Dicho documento contiene la información general como el contenido, las fechas, el costo y los instructores de la actividad.

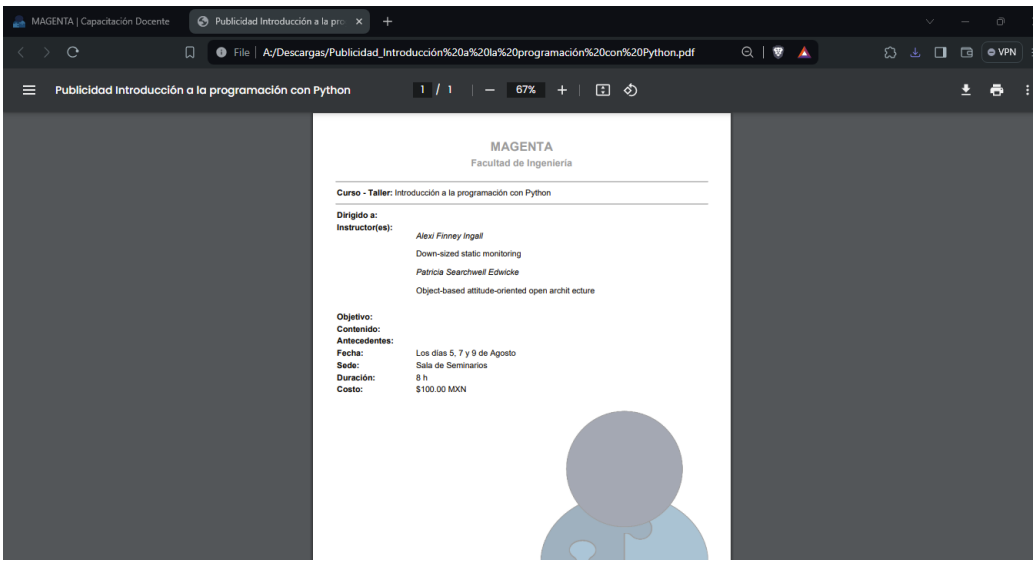


Ilustración 117. Documento de Publicidad

La opción restante es para *Inscribir participantes*.

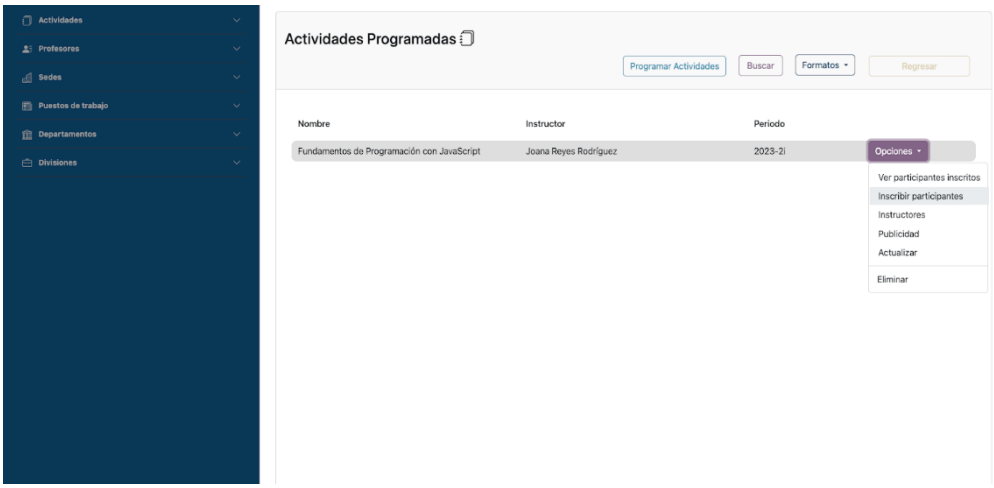


Ilustración 118. Opción Inscribir Participantes como usuario administrador

Tiene una vista similar a la de *asignar instructores*, la diferencia estriba en que no aparecerá la lista de los inscritos en esta vista, sino en la opción *Ver Inscritos,* o bien en *Ver participantes inscritos* de la página anterior.

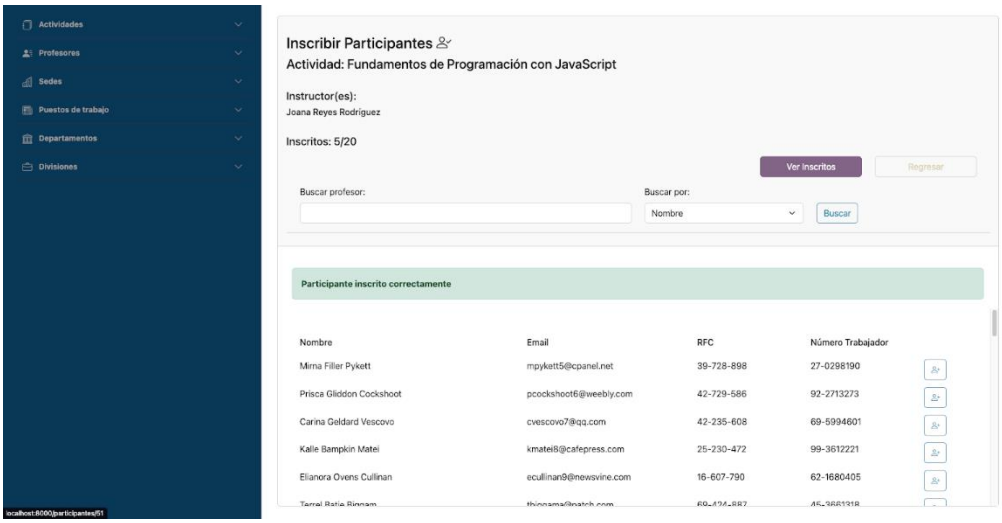


Ilustración 119. Opción Ver Inscritos como usuario administrador.

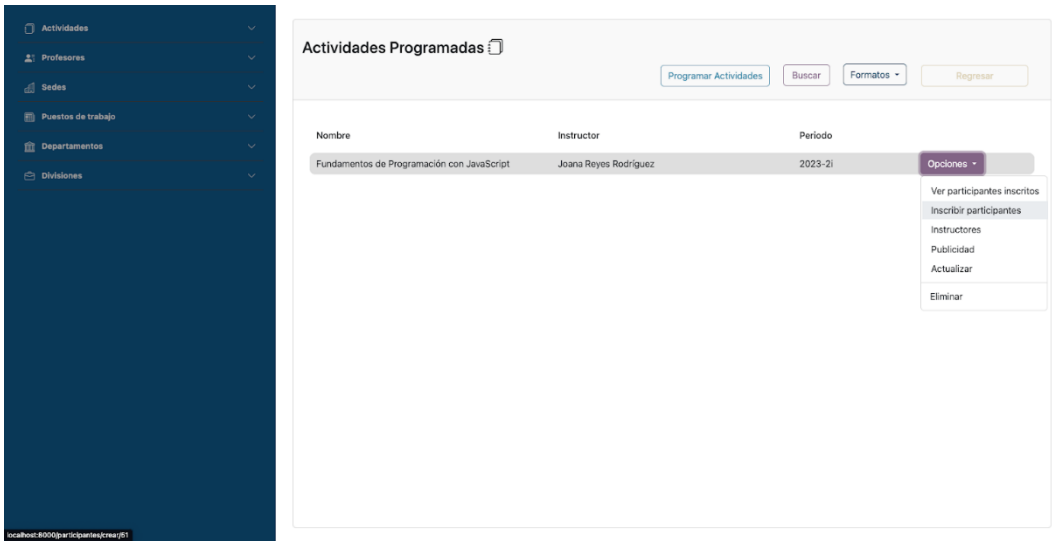


Ilustración 120. Opción Ver Participantes Inscritos como usuario administrador

En esta vista, se podrán visualizar todos los participantes, evaluarlos, ver las encuestas que contestaron y generar formatos relacionados con esa actividad.

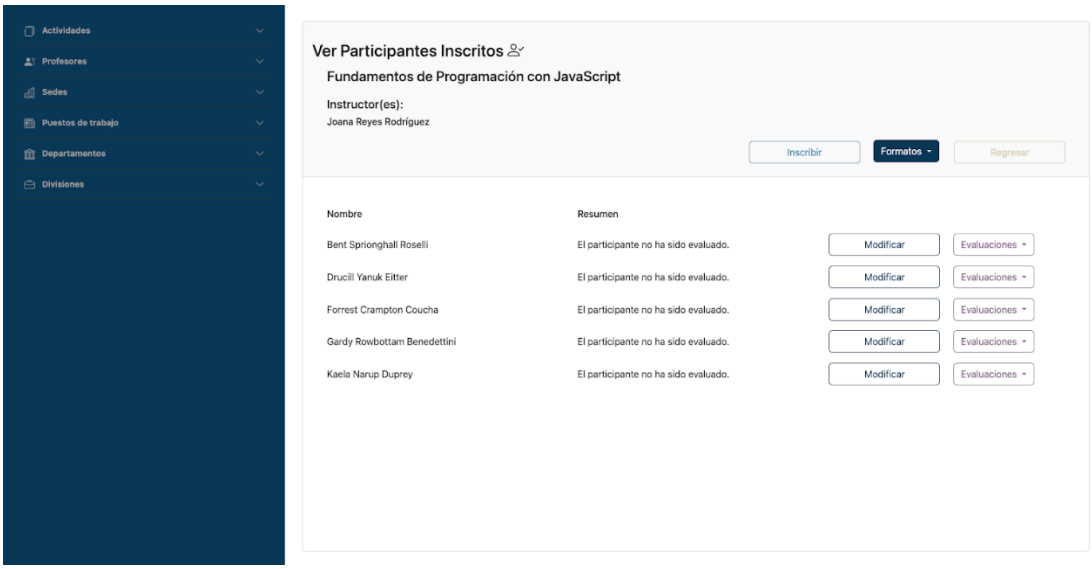


Ilustración 121. Opción de Formatos para Participantes Inscritos

Dentro de la opción *Formatos*, el primer documento "Verificación de datos", cada uno de los participantes debe revisar y marcar su información como correcta o incorrecta.

Ilustración 122. Opción Verificación de Datos

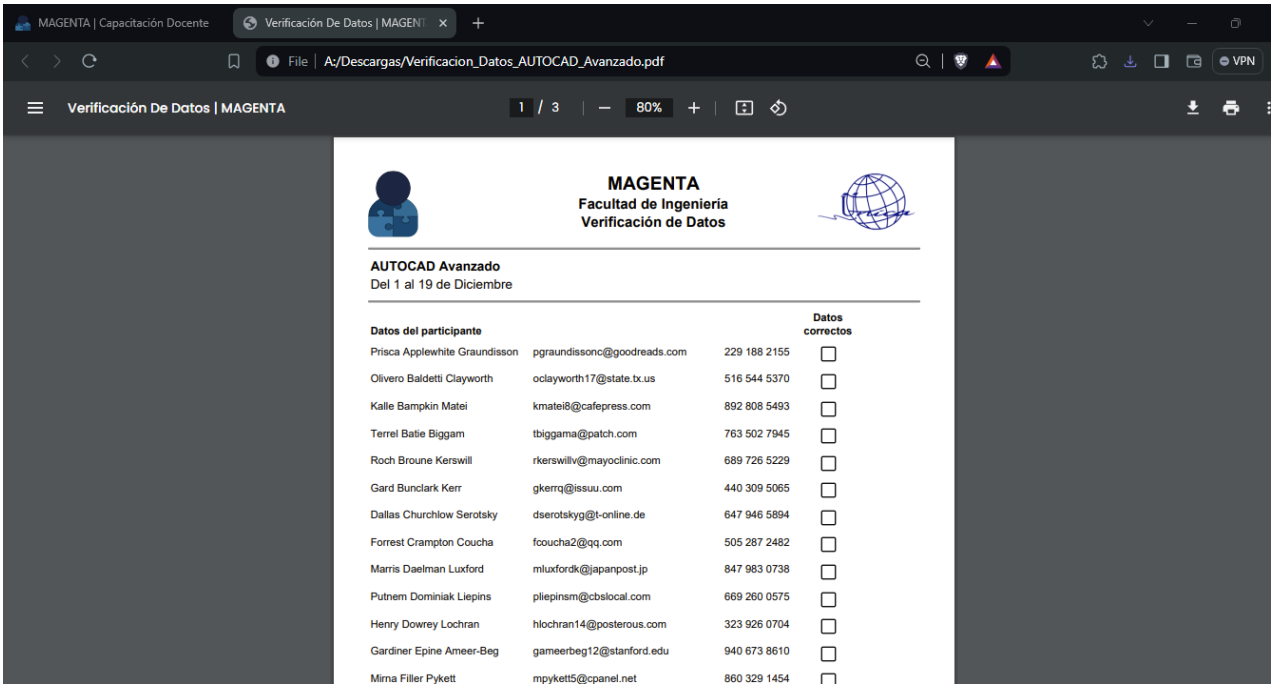


Ilustración 123. Documento de Verificación de Datos

El formato “Identificadores” creará, en caso de requerirlos, su nombre impreso a los participantes durante el curso.

Ilustración 124. Documento con Identificadores de los Participantes

La "hoja de asistencia" es para el instructor, con ella podrá llevar el control de los participantes.

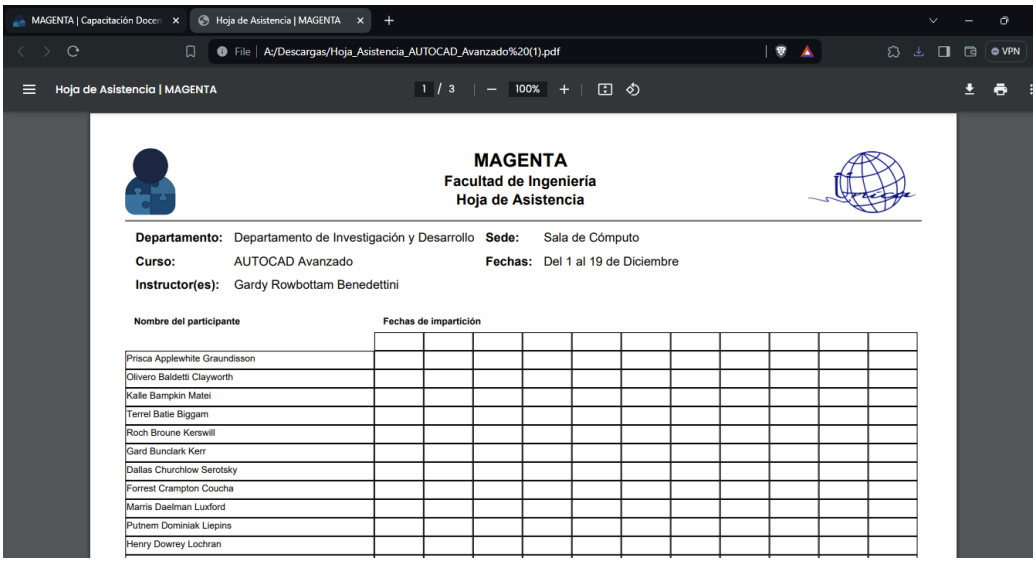


Ilustración 125. Formato de Hoja de Asistencia

Los reportes de "Evaluación" y "de instructores" son para calificar tanto la actividad como a los instructores mediante encuestas. Si el participante no hiciera las evaluaciones, el sistema mandará una advertencia para que se

conteste al menos una; de lo contrario, al generar el formato, sólo marcará *Sin evaluaciones.*

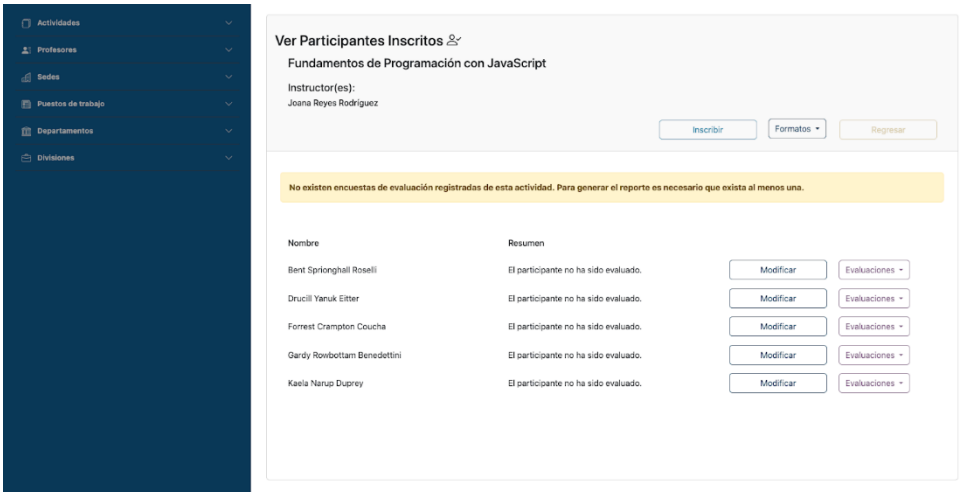


Ilustración 126. Mensaje de advertencia al generar el Reporte de Evaluación

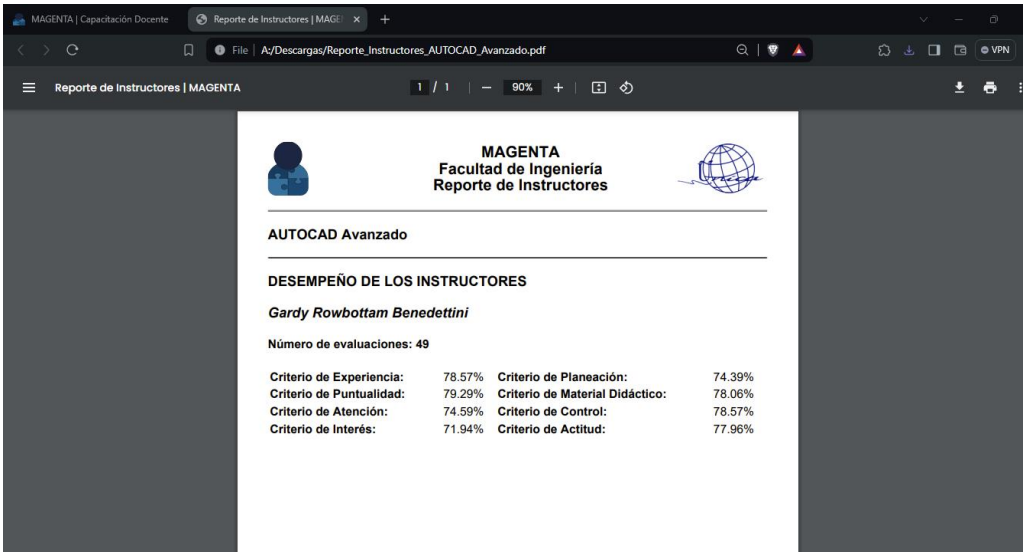


Ilustración 127. Ejemplo de Reporte de Instructores sin evaluaciones

Dentro de esta vista, podremos subir las evaluaciones de los participantes durante la actividad utilizando el botón *Modificar*.

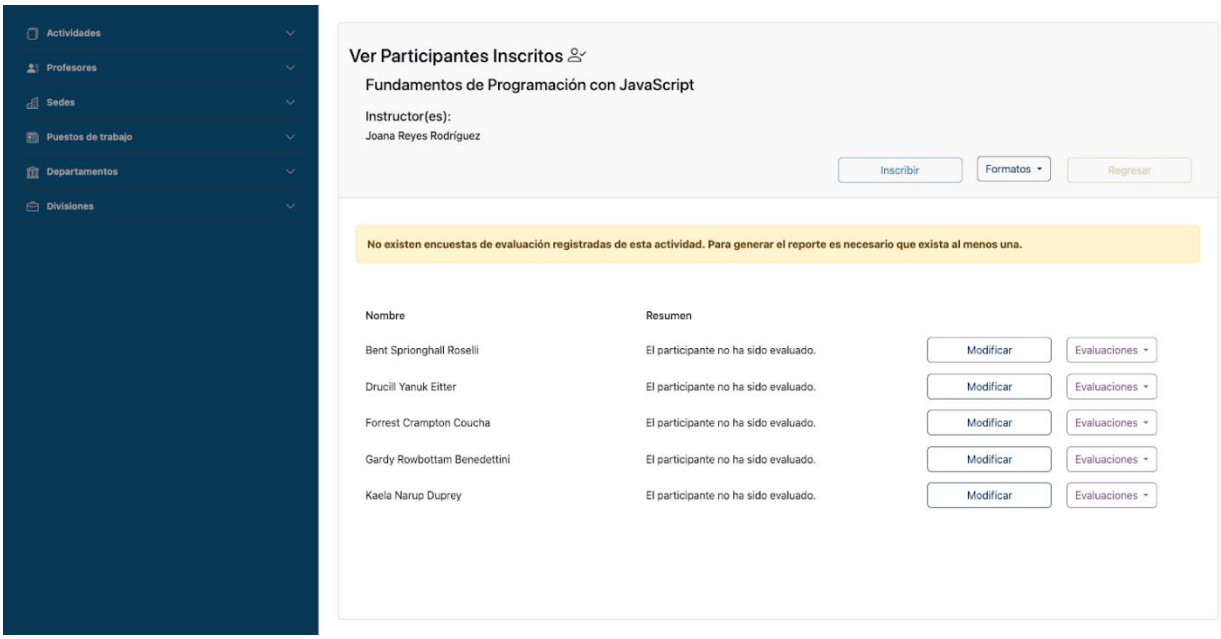


Ilustración 128. Opción Modificar Participante como usuario administrador.

Aquí se indicará si el participante acredita o no la actividad y, en caso necesario, aparecerá la retroalimentación correspondiente. Asimismo, se pedirán otros datos que son relevantes para la organización, como su asistencia durante la actividad, el pago realizado o cualquier comentario adicional por parte del instructor.

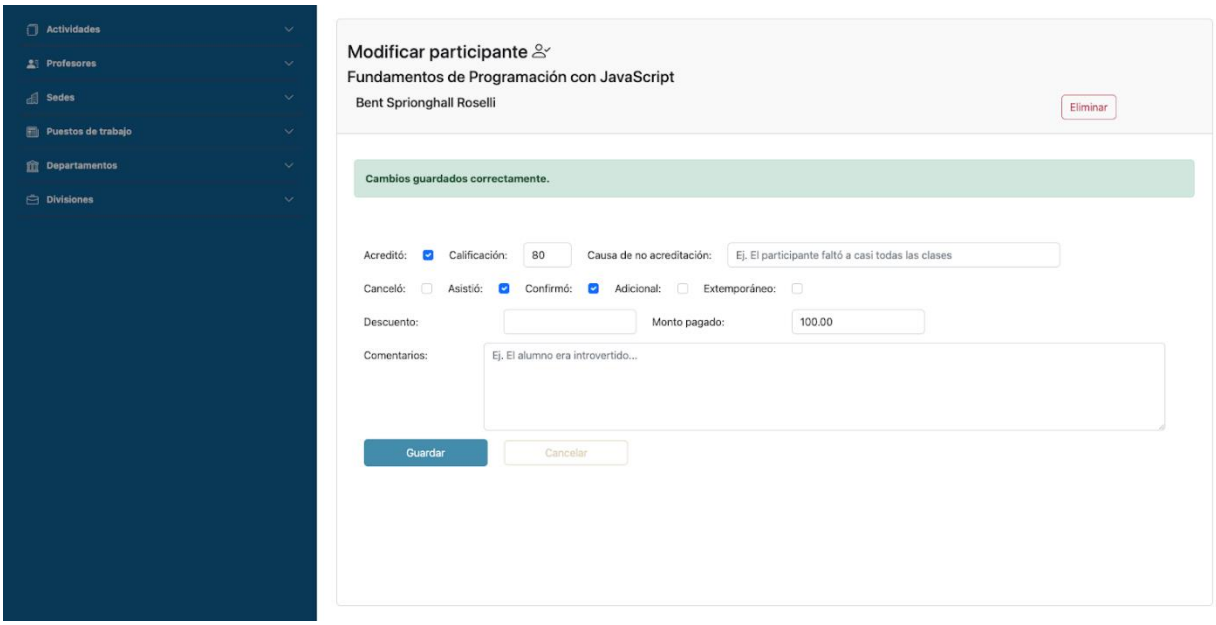


Ilustración 129. Ejemplo de participante que acreditó la actividad

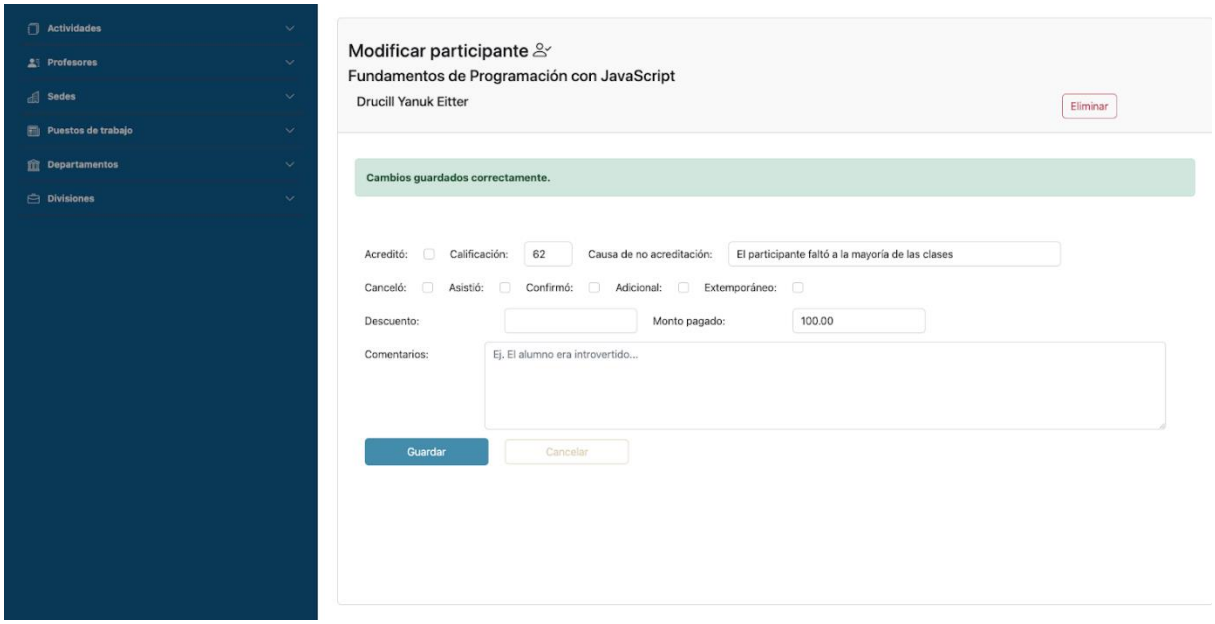


Ilustración 130. Ejemplo de participante que no acreditó la actividad

Al guardar los cambios, el sistema realizará un resumen de cada participante e incluso indicará si aún no ha sido evaluado.

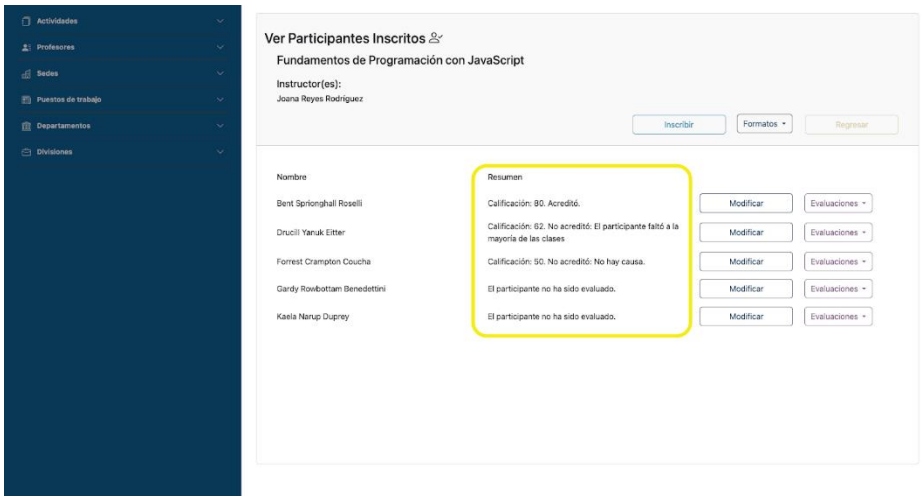


Ilustración 131. Resumen de Participantes

Una vez que se haya evaluado a los participantes, se podrán generar sus constancias. Al igual que los diplomas, éstos son personalizables y se puede elegir el texto que llevará, así como el número de firmantes.

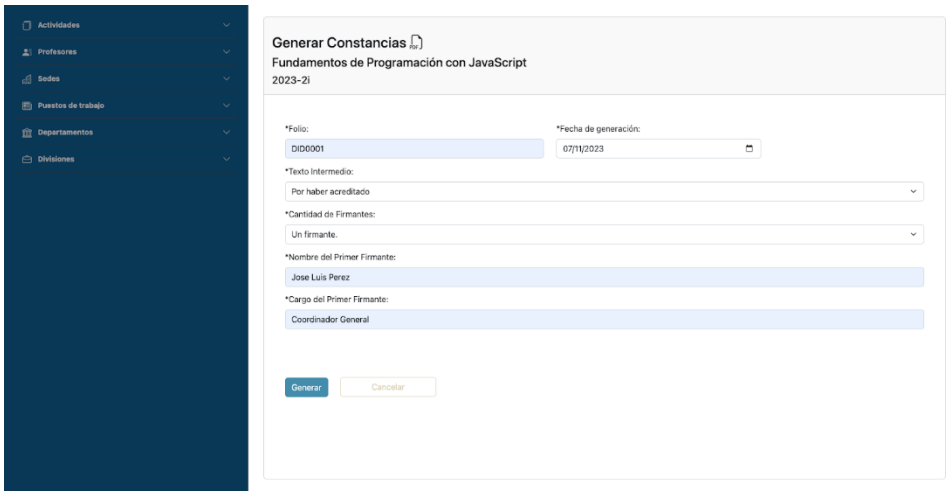


Ilustración 132. Vista Generar Constancias

Se generará un archivo zip con las constancias de todos los participantes que se consideren aprobados.

Ilustración 133. Primera constancia generada con terminación 001

Ilustración 134. Última constancia generada con terminación 003

El último formato que se puede generar es el de los *Reconocimientos*, los cuales son para los instructores; al igual que los diplomas y constancias, es completamente personalizable.

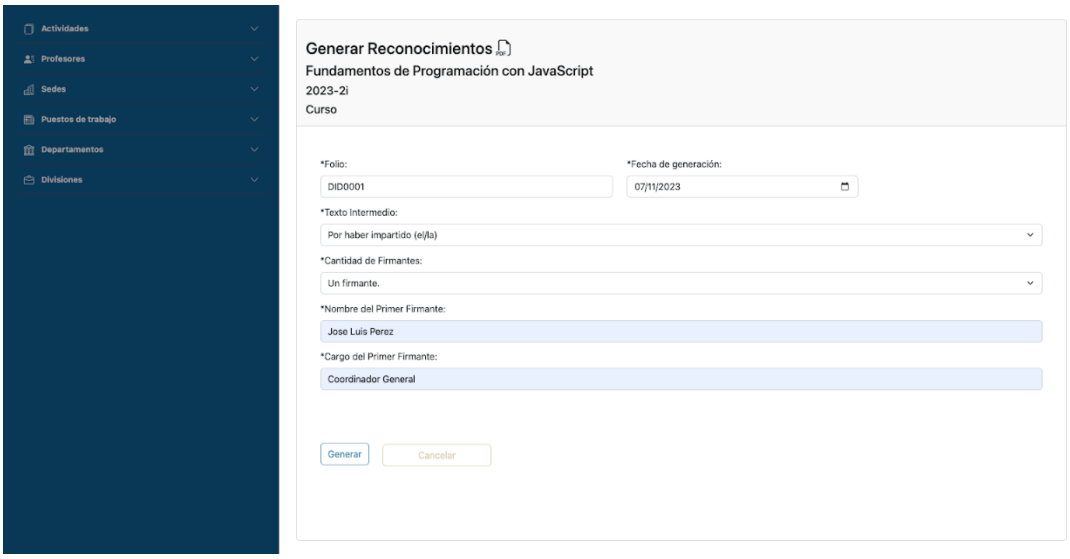


Ilustración 135. Vista Generar Reconocimientos

Ilustración 136. Ejemplo de Reconocimiento

Por último, en este apartado de *Participantes Inscritos*, se pueden ver las encuestas contestadas por los participantes.

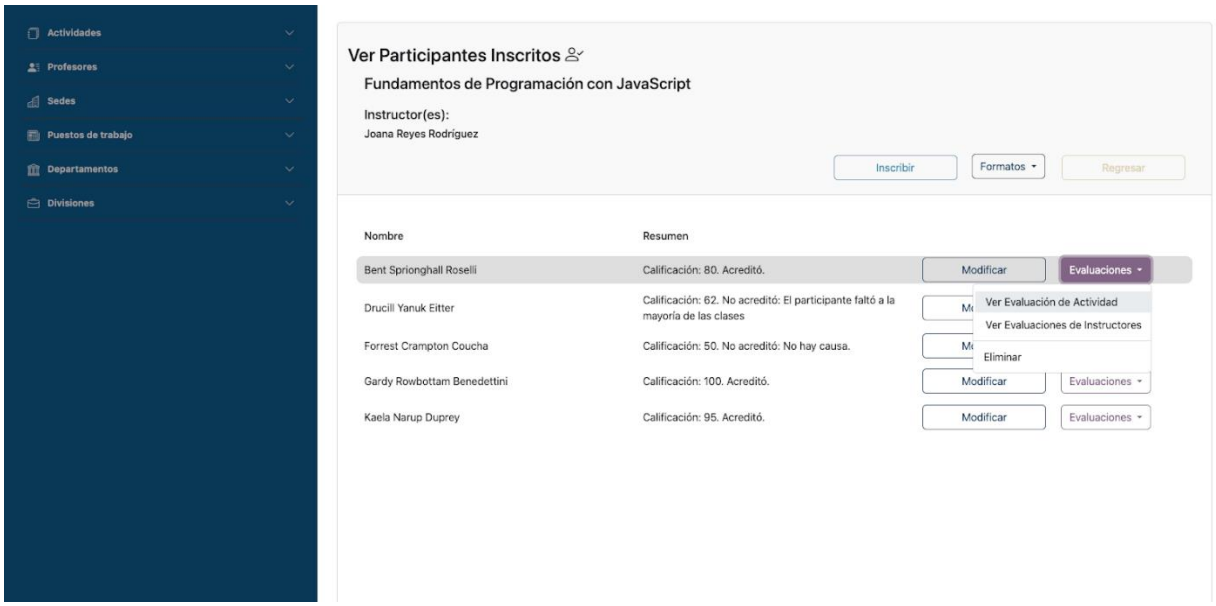


Ilustración 137. Opción Ver Evaluación de Actividad

En este ejemplo, como aún no se ha realizado ninguna evaluación, el sistema mandará mensaje diciendo que el participante no ha evaluado la actividad o al/los instructores.

Como se explica en el apartado 5.1.2, se debe ingresar como usuario temporal/participante para contestar las evaluaciones.

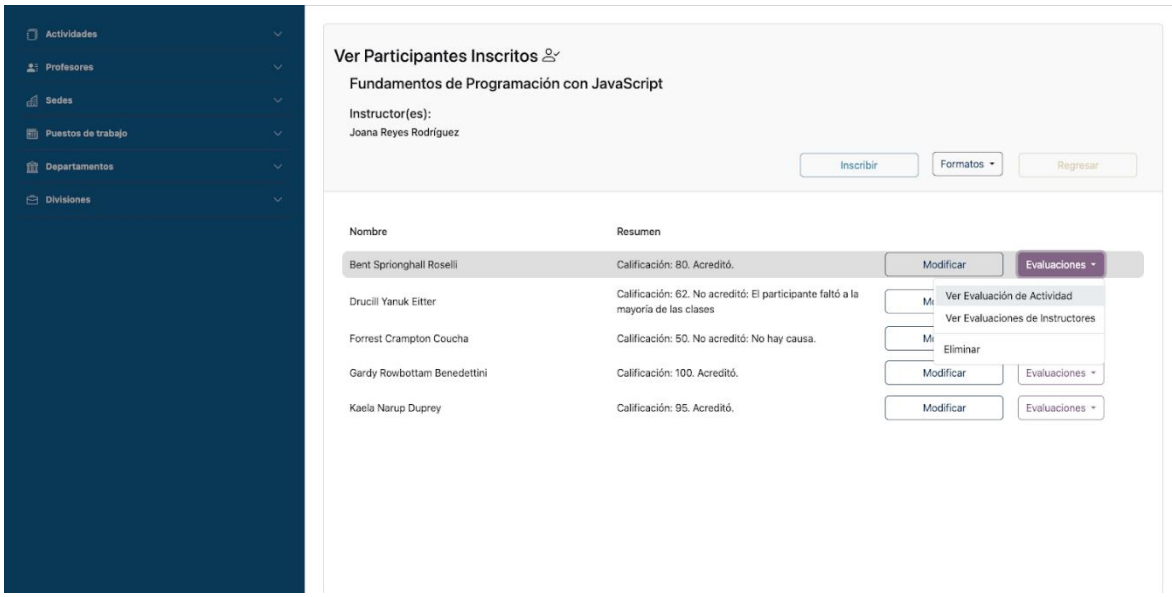


Ilustración 138. Mensaje de Evaluación de Actividad

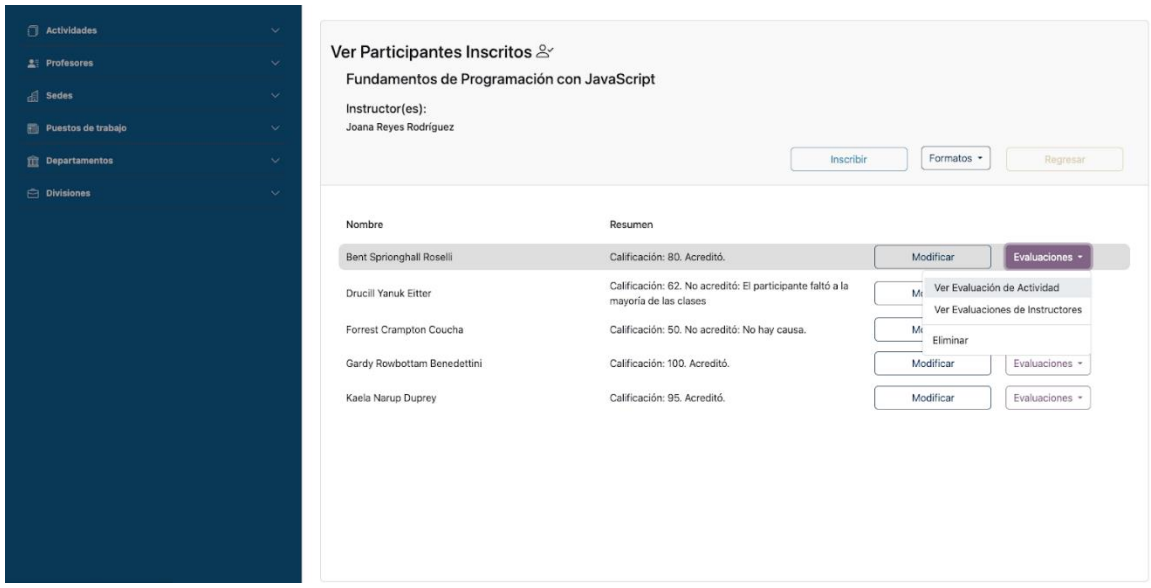


Ilustración 139. Mensaje de Evaluación de Instructor

Ahora bien, en el caso de los *administradores*, también pueden generarse los reportes de evaluación de acuerdo con su departamento.

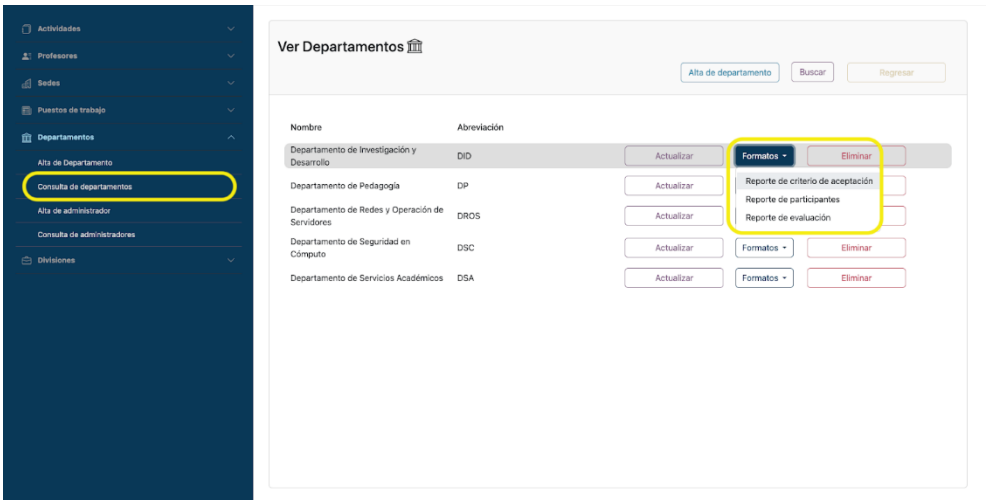


Ilustración 140. Reportes que pueden generar los distintos departamentos

El reporte de *criterio de aceptación* pedirá un año para poder generar el formato.

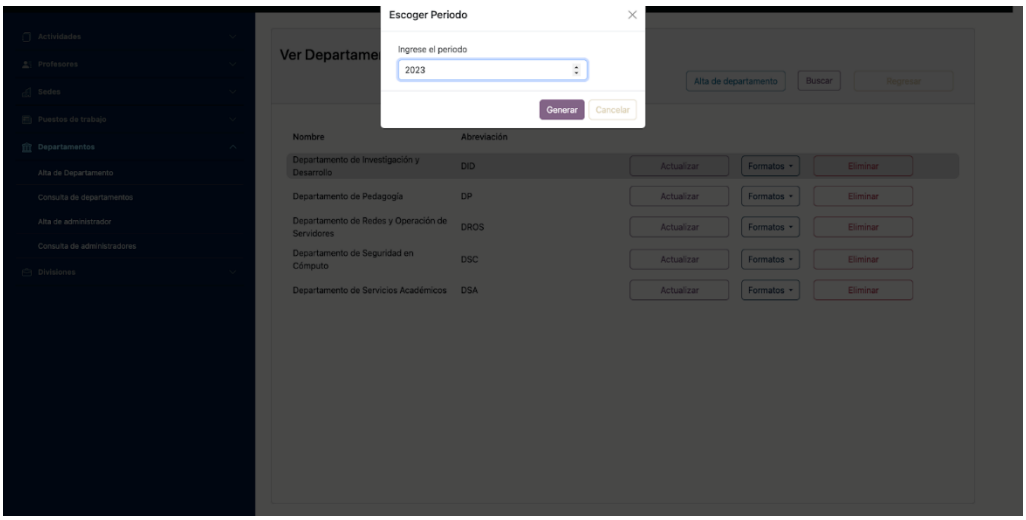


Ilustración 141. Modal para elegir periodo

Con eso, regresará todos los resultados obtenidos de sus actividades correspondientes hasta ese año.

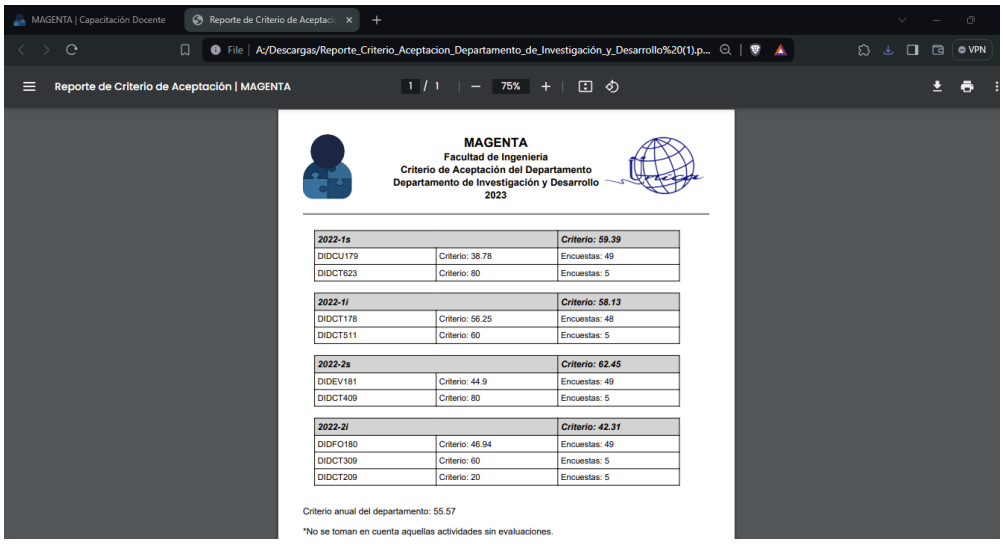


Ilustración 142. Ejemplo de reporte de Criterio de Aceptación del Departamentos

El *Reporte de Participantes* pedirá un periodo ya sea semestral o intersemestral para generar los resultados de esas actividades registradas durante ese tiempo.

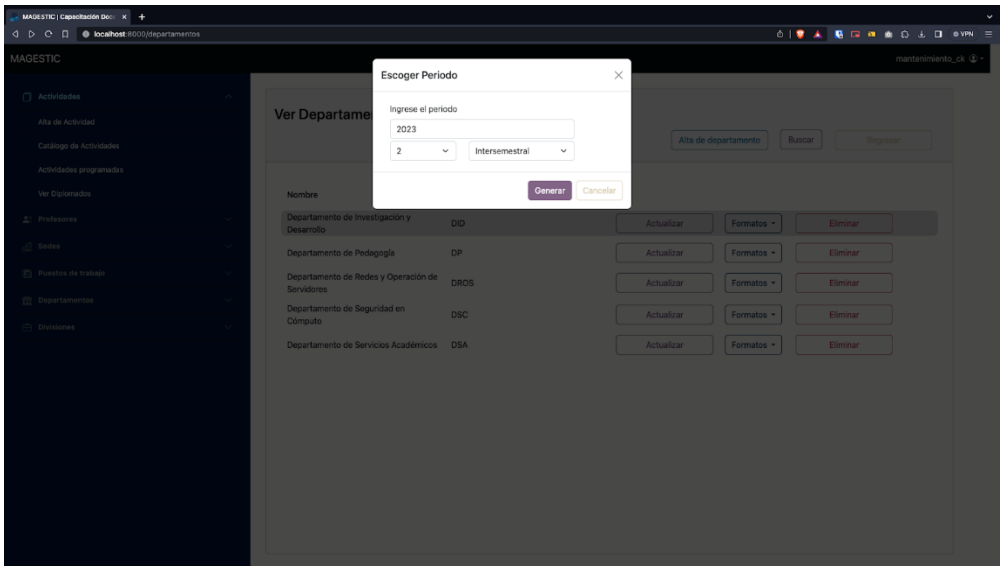


Ilustración 143. Modal para elegir periodo intersemestral o semestral

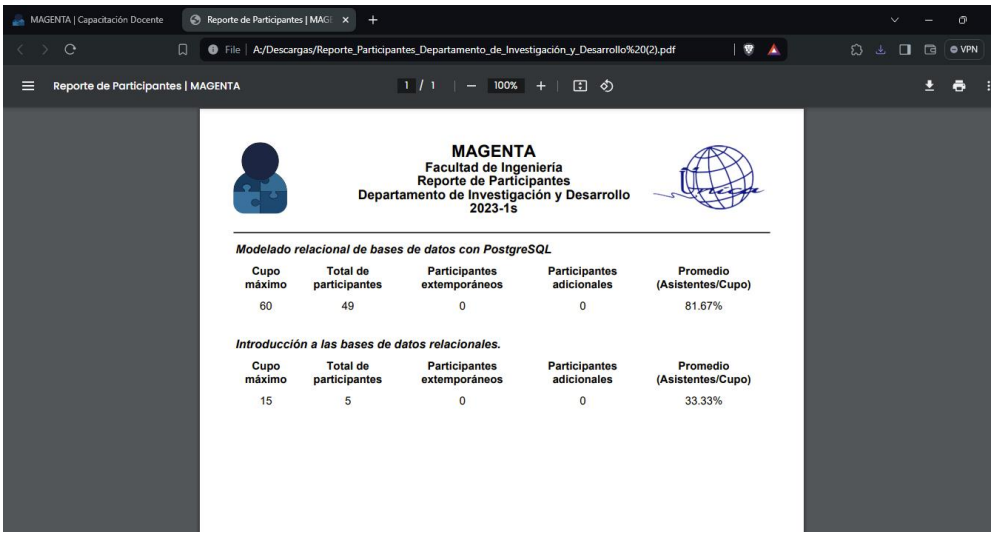


Ilustración 144. Ejemplo de Reporte de Participantes

El *Reporte de Evaluación* pedirá, al igual que el anterior, un periodo en específico para generar el reporte de las actividades impartidas en ese tiempo.

Éste es un reporte general que toma en cuenta todas las respuestas de los participantes, sus comentarios y las evaluaciones de los instructores.

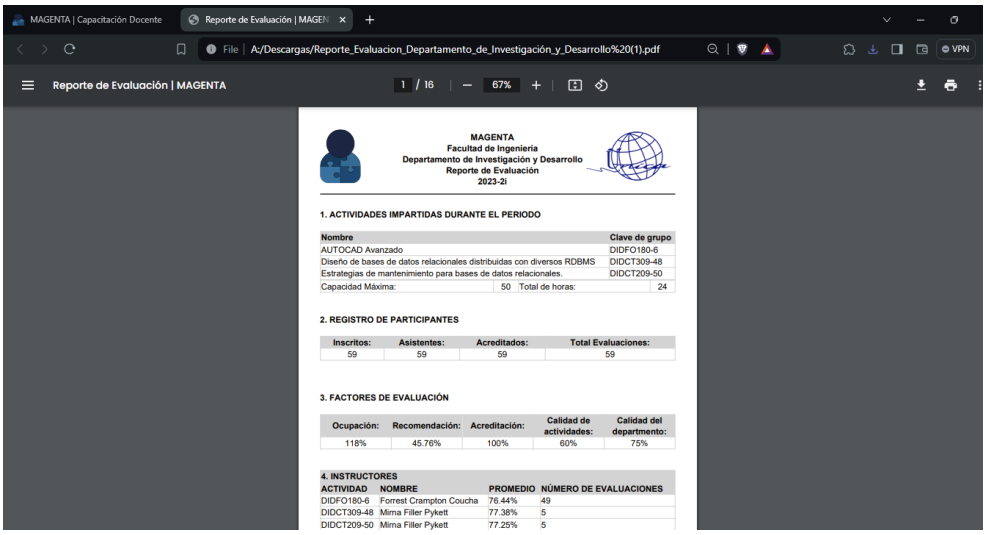


Ilustración 145. Ejemplo de Reporte de Evaluación

5.1.2 Ingresando como usuario temporal/participante

Al iniciar el sistema, daremos clic en *Contestar Encuesta*.

MAGENTA

Ilustración 146. Opción Contestar Encuesta

El formulario pedirá el correo electrónico del participante que evaluará y la clave de grupo que les proporcionará el instructor.

Ilustración 147. Inicio de Sesión del Participante

Si los datos son correctos, la primera evaluación que se realizará será la de la actividad.

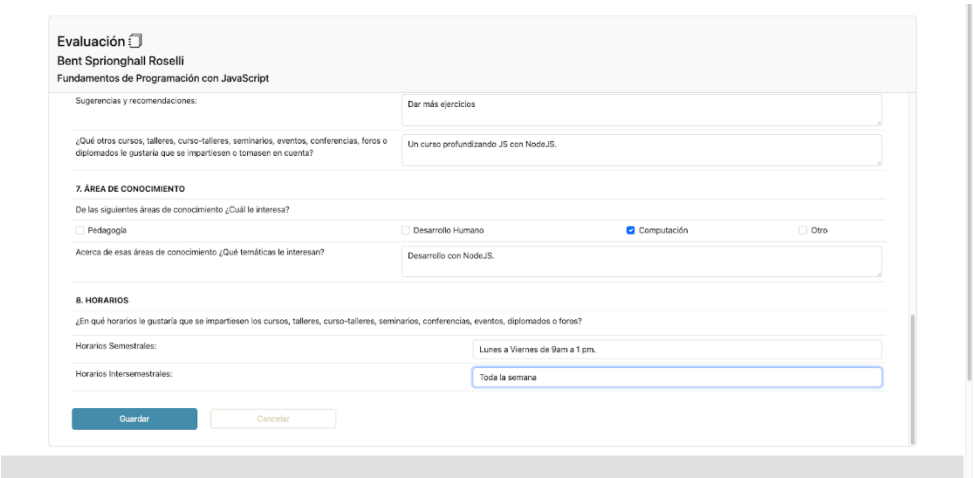


Ilustración 148. Formulario de Evaluación de Actividad

Al guardar los cambios, cambiará de página para realizar la evaluación(es) del/los instructores de la actividad.

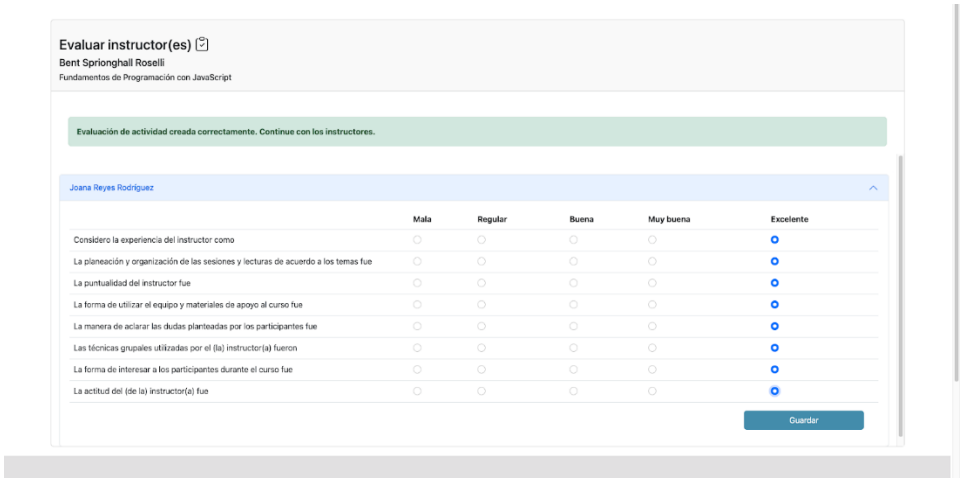


Ilustración 149. Formulario de Evaluación de Instructores

Una vez finalizada la encuesta, regresará al formulario con un mensaje de *Encuesta contestada totalmente*. Si intenta ingresar con los mismos datos, regresará la misma alerta.

Ilustración 150. Mensaje de Encuesta contestada

5.1.3 Ingresando como ayudante

Al iniciar el sistema, damos clic en *Ingresar*.

MAGENTA

Ilustración 151. Opción Ingresar

Ingresamos el usuario y contraseña proporcionados.

Ilustración 152. Vista Inicio de Sesión

A diferencia del administrador, el ayudante tiene funciones limitadas, ya que sólo requiere trabajar con lo más básico del sistema como la gestión de las actividades, profesores y sedes.

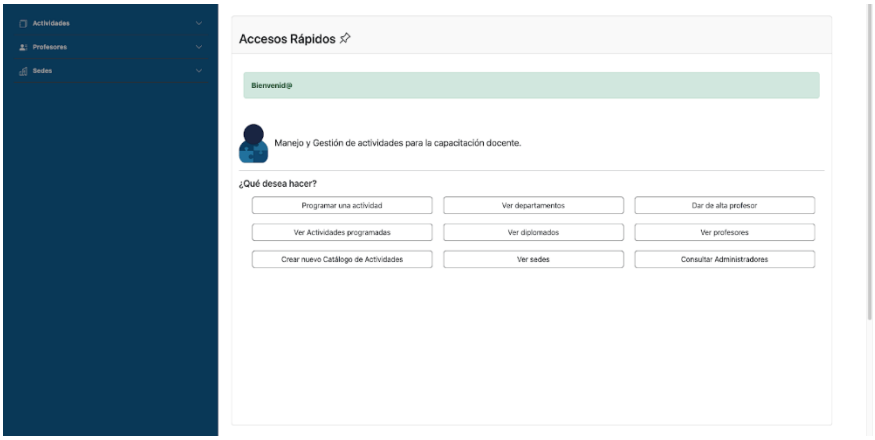


Ilustración 153. Vista principal como usuario ayudante.

Si el usuario intenta ingresar a otra sección, como la de *Administradores*, el sistema regresará un mensaje diciendo que no es posible acceder a esa parte.

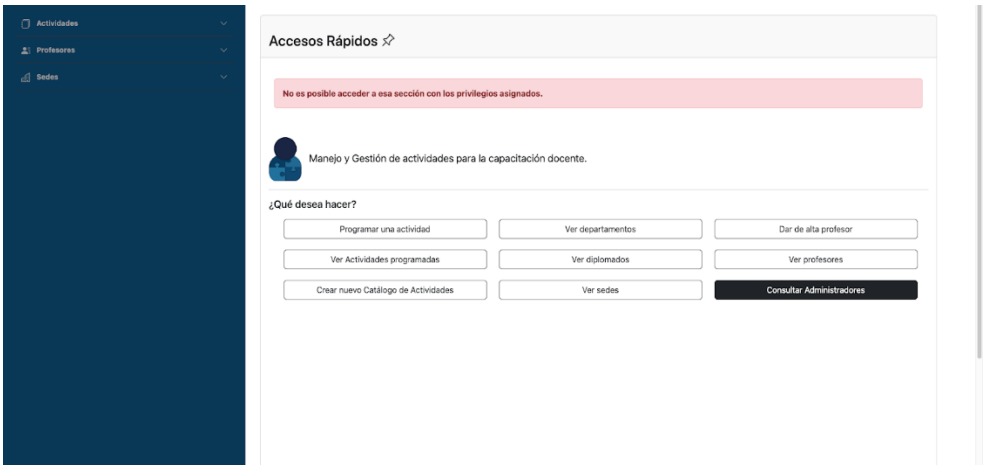


Ilustración 154. Mensaje de error por falta privilegios en el sistema

Dentro de las funciones del ayudante en el sistema, se podrá gestionar todo lo relativo a Actividades, Profesores y Sedes, es decir puede añadir, eliminar y actualizar la información de estas secciones. Asimismo, puede ver las evaluaciones y generar los reportes disponibles correspondientes al departamento al que pertenece.

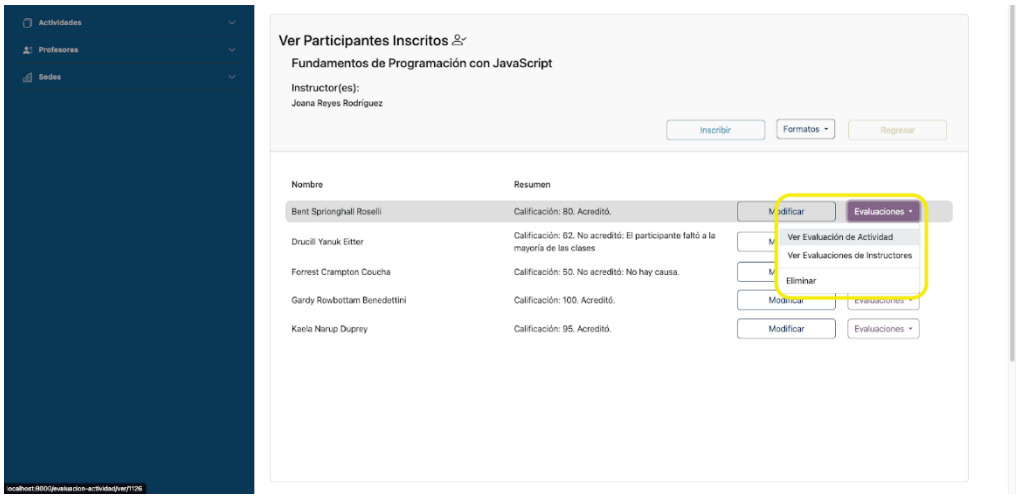


Ilustración 155. Opciones para visualizar evaluaciones como usuario ayudante.

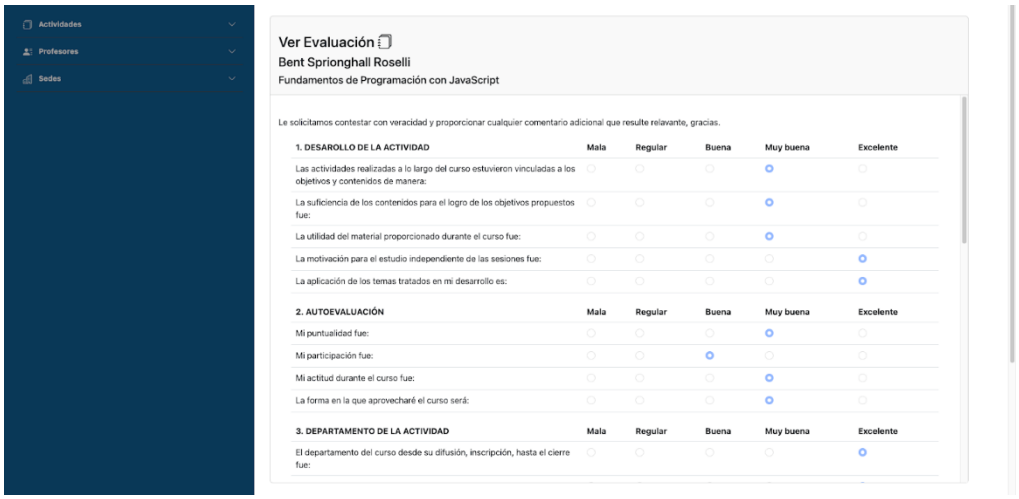


Ilustración 156. Visualización de Evaluación de Actividad

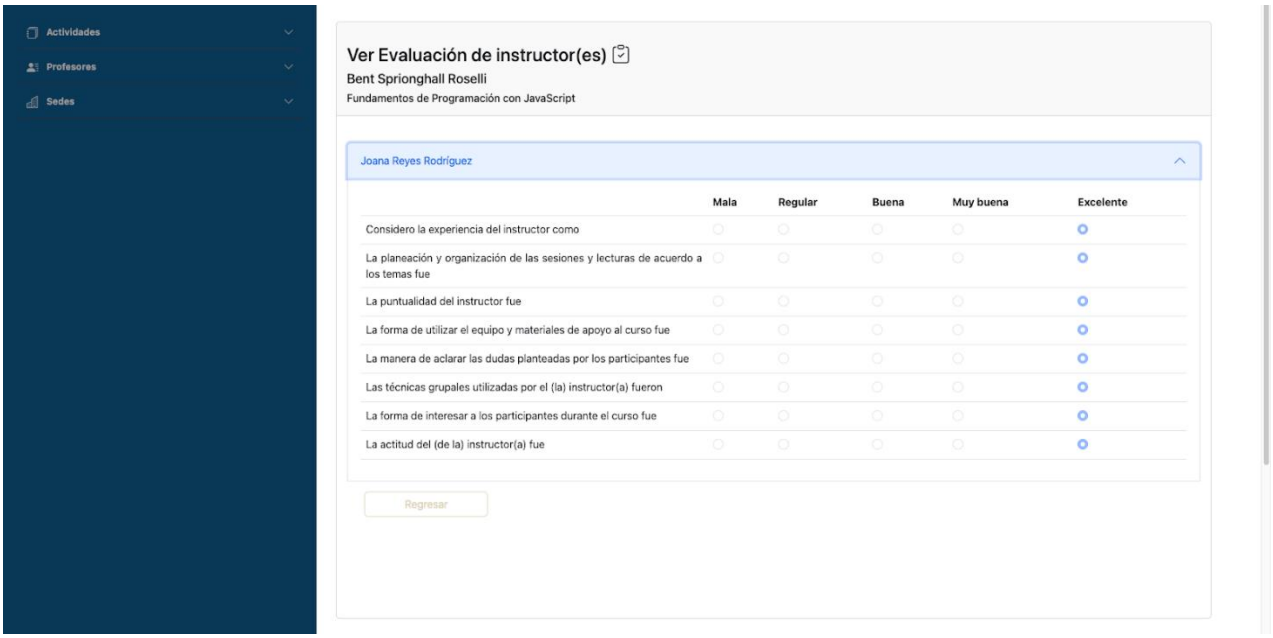


Ilustración 157. Visualización de Evaluación de Instructor

Para este ejemplo ya realizamos las evaluaciones como participantes, por lo que ya podremos generar los reportes de evaluación y de instructores.

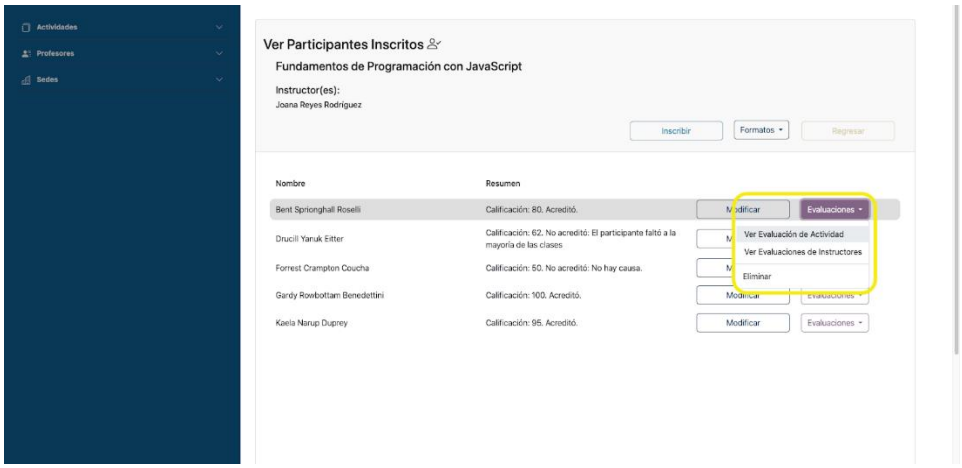


Ilustración 158. Opciones para generar Reportes de Evaluación de Actividades e Instructores como usuario ayudante.

El *"reporte de evaluación"* es un documento que contiene de manera resumida el resultado de las encuestas de una actividad en específico.

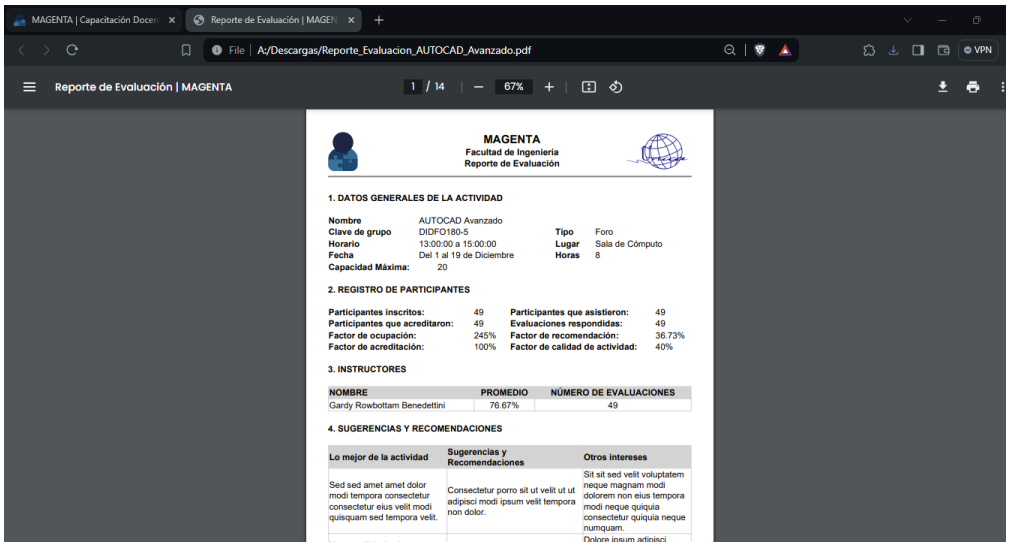


Ilustración 159. Ejemplo de Reporte de Evaluación

El siguiente formato sólo incluye en porcentajes los resultados de la evaluación de instructores que impartieron la actividad.

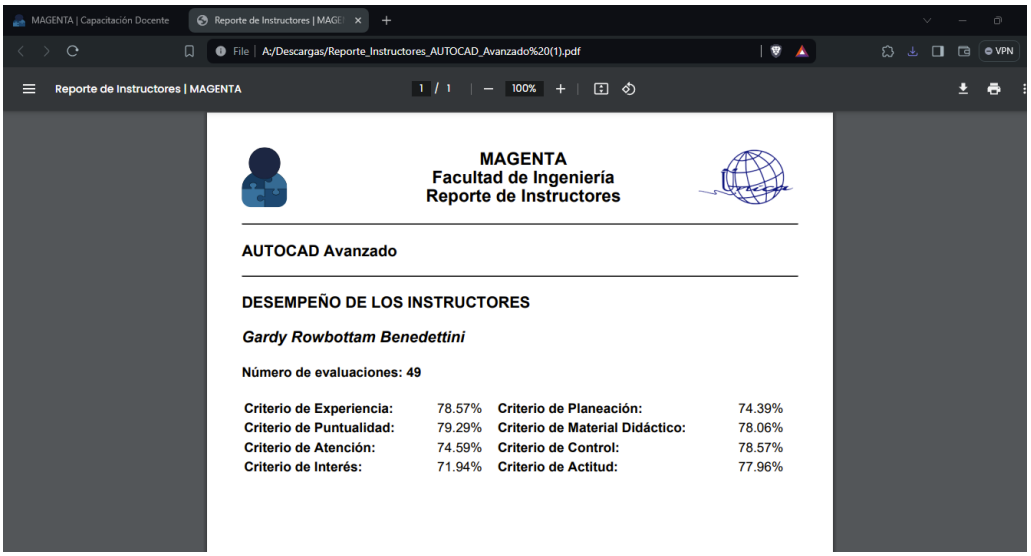


Ilustración 160. Ejemplo de Reporte de Instructores

5.2 Descripción del entregable final

El sistema MAGENTA facilita la gestión de actividades, profesores, evaluaciones y genera los formatos relacionados con cada uno de ellos.

Este sistema se ha desarrollado con el objetivo de mejorar la eficiencia y efectividad de los programas de formación o capacitación en entornos académicos.

MAGENTA tiene como funciones principales lo siguiente:

- Seguridad y Acceso:
    - Gestión de usuarios y roles.
    - Protección de datos sensibles.
- Gestión de Actividades
    - Registro y administración de actividades.

    - Programación de fechas, horarios y ubicaciones.
    - Gestión de instructores y participantes.
- Registro de Profesores:
    - Perfiles detallados de docentes.
    - Información académica y experiencia.
    - Asignación a cursos y seguimiento de desempeño.
- Evaluaciones:
    - Creación y programación de evaluaciones.
    - Seguimiento de resultados y análisis de datos.
    - Retroalimentación a docentes y participantes.
- Generación de Formatos:
    - Creación de certificados y reportes.
    - Personalización de formatos de evaluación.
    - Exportación en diversos formatos (PDF, Excel).

El sistema ha sido diseñado con un enfoque en la adaptabilidad, esto significa que puede ser implementado con éxito en cualquier organización en busca de mejorar sus programas de capacitación docente.

Hemos trabajado para generalizar sus características y funcionalidades, asegurando ajustarlas a una amplia gama de necesidades; sin embargo, entendemos que cada institución es única y, por lo tanto, estamos dispuestos

a realizar modificaciones personalizadas para satisfacer los requisitos específicos de cada cliente.

Además, para facilitar la transición, podemos exportar datos preexistentes de la organización al sistema, tal como se hizo en nuestro caso de estudio donde fue incluida la información predeterminada sobre divisiones, puestos de trabajo y cuentas de administradores.

Todo el código de programación desarrollado se encuentra disponible en los siguientes dos repositorios de GitHub para su consulta:

- Aplicación WEB: https://github.com/MauRamos334455/magenta
- Base de datos: https://github.com/MauRamos334455/magenta-db

Conclusiones

En esta tesis se desarrolló una solución capaz de atender las necesidades descritas y observadas al comienzo de nuestro trabajo escrito, en la cual se ofrece un software simple y potente que cumple con el objetivo principal de almacenar la información relacionada con las actividades para capacitación docente.

Todo sistema debe ser pensado para facilitar y agilizar la carga administrativa de los usuarios; por lo tanto, consideramos, al analizar el desempeño de MAGENTA, que es capaz de brindar una plataforma dinámica donde se concentra toda la información que una organización necesita para llevar el control adecuado de sus actividades de capacitación.

Más allá de eso, el núcleo de MAGENTA (tanto en cuestión técnica como su alcance) es adaptable no solamente a una organización sin fines de lucro y orientada a la capacitación docente, sino a cualquier institución, organización y/o empresa que se dedique a la divulgación de conocimiento por medio de actividades de capacitación; es decir, aquellas pequeñas y medianas empresas que se dedican al desarrollo de cursos, talleres, seminarios de cualquier índole o materia.

Futuros trabajos

A pesar de que el sistema presenta un muy buen desempeño en las pruebas realizadas, carecemos de un análisis completo elaborado a partir de herramientas automáticas que existen hoy en día para la elaboración de pruebas unitarias.

La función de dicho análisis es que nos provea de resultados certeros acerca de posibles fallos de seguridad o áreas de oportunidad en el código desarrollado. Por esta razón, uno de nuestros siguientes trabajos será programar e implementar pruebas unitarias automatizadas que se encarguen de evaluar los requerimientos y funcionalidades principales del sistema hasta cubrir la mayor cantidad posible de fallos. El objetivo principal de las pruebas unitarias es asegurarse de que cada unidad de código funcione correctamente de manera independiente antes de integrar en un sistema más amplio.

Como se sabe, la ingeniería de software establece que los sistemas necesitan mantenimiento constante, así como nuevas funcionalidades. Uno de los pasos a seguir es diseñar e implementar nuevas características al sistema, así como mejorar las ya existentes con base en experiencias de los usuarios y sus nuevos requerimientos.

Nuestro alcance abarca únicamente el sector de aplicaciones web; sin embargo, pensamos que puede ampliarse a una aplicación de escritorio, ejecutable nativamente en el sistema operativo sin necesidad de un navegador web y una aplicación móvil. Por esta razón, nuestra meta a largo plazo es llevar la opción a más dispositivos y, por ende, a más usuarios. El reto será adaptar nuestro diseño al de un servicio web.

Finalmente, parte de nuestra filosofía es diseñar soluciones accesibles para todo el mundo; por lo tanto, nos parece adecuado implementar protocolos y estándares de accesibilidad en el código desarrollado para que nuestra aplicación web y el software a largo plazo que desarrollemos sean accesibles a la mayor cantidad de personas posibles, incluyendo a aquellas con incapacidad visual o auditiva, con ayuda de herramientas existentes como NVDA.

Anexos

Anexo A

Tabla 16. Elementos Estructurales

Etiquetas más utilizadas en Elementos *Estructurales*	
<!DOCTYPE html>	Indica que se utilizará HTML5
<html>	Contenedor principal que indica el inicio y el fin del código HTML.
<head>	Incluye la información requerida para el documento como lo pueden ser los estilos o scripts a ejecutar.
<title>	Es el nombre de la página la cual aparece en la pestaña del navegador.
<link>	Enlaza documentos externos, generalmente para los estilos,
<meta>	Proporciona información adicional (metadatos) por ejemplo, los tipos de caracteres utilizados en el documento, palabras clave o descripción del contenido de la página, entre otros.
<style>	Escribe los estilos CSS internos.
<script>	Escribe un script interno o se enlaza a uno externo de JavaScript.

Anexo B

Tabla 17. Elementos Semánticos

Etiquetas más utilizadas en Elementos *Semánticos*	
<body>	Es todo el contenido del documento HTML. Sólo puede existir este elemento una vez.
<main>	Contenido principal del documento. Sólo puede existir este elemento una vez.
<section>	Define el contenido en secciones por algún tema en particular.
<nav>	Especifica un menú de navegación.
<article>	Contenido de la página que se puede utilizar de forma aislada.
<aside>	Define una barra lateral de la página.
<header>	Encabezado de la página.
<footer>	Pie de página del documento o cualquier otro elemento que lo requiera.

Índice de Ilustraciones

Índice de Tablas

Bibliografía

Amazon. (s.f.). *¿Qué es JavaScript (JS)?* Recuperado el Julio de 2023, de AWS: https://aws.amazon.com/es/what-is/javascript/

Amazon Web Services. (s.f.). *¿Qué es Docker?* Recuperado el Noviembre de 2023, de https://aws.amazon.com/es/docker/

Assemble Institute of Technology. (s.f.). *¿Qué es PHP y para qué sirve?* Recuperado el Julio de 2023, de Assemble Institute of Technology: https://assemblerinstitute.com/blog/que-es-php/#:~:text=PHP%20es%20un%20lenguaje%20de,tiene%20numerosas%20utilidades%20en%20frontend

Businessmap. (s.f.). *¿Qué es Kanban? Explicación para principiantes.* Recuperado el Noviembre de 2023, de https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

Cambridge Dictionary. (s.f.). *Meaning of framework in English*. Recuperado el Junio de 2023, de Cambridge Dictionary: https://dictionary.cambridge.org/us/dictionary/english/framework

Ccori Huaman, W. (07 de Septiembre de 2018). *Los 10 patrones comunes de arquitectura de software*. Recuperado el Junio de 2023, de Medium: https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b

CloudFlare. (2023). *What is HTTPS?* Recuperado el Marzo de 2023, de CloudFlare: https://www.cloudflare.com/learning/ssl/what-is-https/

Data México. (s.f.). *Profesores y Especialistas en Docencia*. Recuperado el Febrero de 2023, de Data México: https://datamexico.org/es/profile/occupation/profesores-y-especialistas-en-docencia

Departamento de Ciencias de la Computación e I.A. (s.f.). *Especificación de requerimientos - Diseño de bases de datos.* Recuperado el Marzo de 2023, de Departamento de Ciencias de la Computación e I.A: https://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf

Developer Mozilla. (s.f.). *Generalidades del protocolo HTTP*. Recuperado el Marzo de 2023, de mdn web docs: https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Developer Mozilla. (s.f.). *Métodos de petición HTTP*. Recuperado el Julio de 2023, de mdn web docs: https://developer.mozilla.org/es/docs/Web/HTTP/Methods

Developer Mozilla. (s.f.). *Selectores de tipo*. Recuperado el Julio de 2023, de mdn web docs: https://developer.mozilla.org/es/docs/Web/CSS/Type_selectors

El Financiero. (15 de Mayo de 2022). *Día del Maestro 2022: ¿Cuántos docentes hay en todo México?* Recuperado el Febrero de 2023, de El Financiero: https://www.elfinanciero.com.mx/nacional/2022/05/15/dia-del-maestro-2022-cuantos-docentes-hay-en-todo-mexico/

ENIUN - Diseño Web y Marketing Digital. (2023). *HTML5, estructura básica y elementos semánticos*. Recuperado el Julio de 2023, de ENIUN: https://www.eniun.com/html5-estructura-basica-elementos-semanticos/

Escuela de Diseño Madrid. (s.f.). *Qué es HTML y CSS: los básicos del desarrollo web*. Recuperado el Julio de 2023, de Escuela de Diseño Madrid: https://esdima.com/que-es-html-y-css/

F.E.S. Acatlán. (19 de Noviembre de 2015). *El triángulo de la seguridad*. Recuperado el Febrero de 2023, de F.E.S. Acatlán - UNAM: http://blogs.acatlan.unam.mx/lasc/2015/11/19/el-triangulo-de-la-seguridad/

Fullstack Tutorials. (s.f.). *Architectural Patterns vs Design Patterns*. Recuperado el Junio de 2023, de Fullstack Tutorials: https://www.fullstacktutorials.com/architectural-patterns-vs-design-patterns-57.html

García González, J. C., & André Ampuero, M. (Abril de 2015). *Implementación del enfoque de reglas de negocio utilizando motores de reglas en el desarrollo de aplicaciones Java.* Recuperado el Marzo de 2023, de Ciencias de la Información Vol. 46, No. 1: https://biblat.unam.mx/hevila/Cienciasdelainformacion/2015/vol46/no1/6.pdf

Garrido Sotomayor, S. (14 de Noviembre de 2023). *IEBS*. Recuperado el Noviembre de 2023, de https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

GCF Global. (s.f.). *Informática Básica: ¿Qué son las aplicaciones web?* Recuperado el Febrero de 2023, de GCF Global: https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/

Instituto Federal de Telecomunicaciones. (04 de Julio de 2021). *Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en*

los Hogares (ENDUTIH) 2021. (Comunicado de prensa) 4 de julio. Recuperado el Febrero de 2023, de Instituto Federal de Telecomunicaciones - Comunicación y Medios: https://www.ift.org.mx/comunicacion-y-medios/comunicados-ift/es/encuesta-nacional-sobre-disponibilidad-y-uso-de-tecnologias-de-la-informacion-en-los-hogares-endutih

Ismail, N. (10 de Abril de 2017). *Modern technology: advantages and disadvantages*. Recuperado el Febrero de 2023, de Information age: https://www.information-age.com/modern-technology-advantages-disadvantages-5283/

Junta de Andalucía. (s.f.). *Patrones de diseño*. Recuperado el Junio de 2023, de Marco de Desarrollo de la Junta de Andalucía: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/13

La Universidad en Internet. (22 de Septiembre de 2022). *Framework: qué es, para qué sirve y algunos ejemplos*. Recuperado el Junio de 2023, de UNIR - Ingeniería y Tecnología: https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/

Lima, G. (18 de Noviembre de 2022). *REST: Concepto y Fundamentos*. Recuperado el Julio de 2023, de ALURA LATAM: https://www.aluracursos.com/blog/rest-concepto-y-fundamentos

Londoño, P. (04 de Enero de 2023). *Qué son las aplicaciones web y 8 ejemplos*. Recuperado el Febrero de 2023, de HubSpot: https://blog.hubspot.es/website/que-es-aplicacion-web

Manz. (s.f.). *¿Qué es el DOM?* Recuperado el Julio de 2023, de MANZ.DEV: https://lenguajejs.com/javascript/dom/que-es/

Manz. (s.f.). *La cascada de CSS: Importancia, Especificidad y Orden*|. Recuperado el Julio de 2023, de MANZ.DEV: https://lenguajecss.com/css/cascada-css/que-es-cascada/

Northware. (26 de Mayo de 2022). *Requerimientos en el desarrollo de software y aplicaciones.* Recuperado el Marzo de 2023, de Northware: https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/

Norton. (23 de Febrero de 2023). *What is a VPN?* Recuperado el Marzo de 2023, de Norton: https://us.norton.com/blog/privacy/what-is-a-vpn

Open Bootcamp. (s.f.). *Qué es un framework*. Recuperado el Junio de 2023, de Open Bootcamp: https://open-bootcamp.com/aprender-programar/que-es-un-framework

ORACLE. (s.f.). *¿Qué es una base de datos relacional (sistema de gestión de bases de datos relacionales)?* Recuperado el Junio de 2023, de ORACLE: https://www.oracle.com/mx/database/what-is-a-relational-database/

PRAGMA. (s.f.). *PRAGMA*. Recuperado el Noviembre de 2023, de https://www.pragma.com.co/blog/la-introduccion-al-manifiesto-agil

Santander. (02 de Diciembre de 2021). *¿Cuáles son las ventajas y desventajas de la tecnología actual?* Recuperado el Febrero de 2023, de Santander Universidades: https://www.santanderopenacademy.com/es/blog/ventajas-y-desventajas-de-la-tecnologia.html

Shvets, A. (2023). *El catálogo de patrones de diseño*. Recuperado el Junio de 2023, de Refactoring.Guru: https://refactoring.guru/es/design-patterns/catalog

Shvets, A. (2023). *Historia de los patrones*. Recuperado el Junio de 2023, de Refactoring.Guru: https://refactoring.guru/es/design-patterns/history

Technology Consulting. (29 de Agosto de 2018). *Framework o librerías: ventajas y desventajas*. Recuperado el Junio de 2023, de ti Think: https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/

Think, Technology Consulting. (16 de Octubre de 2018). *Metodologías ágiles ¿qué son y para qué sirven?* Recuperado el Julio de 2023, de https://www.tithink.com/es/2018/10/16/metodologias-agiles-que-son-y-para-que-sirven/

TIBA México. (6 de Diciembre de 2016). *El sistema Kanban en la industria automotriz*. Recuperado el Noviembre de 2023, de https://www.tibagroup.com/mx/sistema-kanban

Unidad de Servicios de Cómputo Académico. (Enero de 2023). *Misión*. Recuperado el Febrero de 2023, de UNICA - Facultad de Ingeniería: https://www.ingenieria.unam.mx/unica/mision.php

Villa, R. (20 de Enero de 2017). *Requerimientos Funcionales y No Funcionales*. Recuperado el Marzo de 2023, de Universidad Técnica de Machala: https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/20/requerimientos-funcionales-y-no-funcionales/

Yeeply. (s.f.). *Los 5 mejores ejemplos de desarrollo de aplicaciones web que debes conocer*. Recuperado el Febrero de 2023, de Yeeply: https://www.yeeply.com/blog/ejemplos-desarrollo-de-aplicaciones-web/