

**UNIVERSIDAD POLITÉCNICA DE ATLACOMULCO**

Ingeniería en Sistemas Computacionales

**PROYECTO DE ESTADIA**

**“DESARROLLO DEL SISTEMA**

**AUDITORIA POR CAPAS”**

Presentado por:

ORLANDO RAMOS PÉREZ

Profesor Asesor:

JUAN GARCIA VILLAFUERTE

|  |  |
| --- | --- |
| Calificación Asesor Empresarial: |  |
| Calificación y firma del Profesor Asesor: |  |
| Calificación Final: |  |

Atlacomulco, México. Diciembre 2024

Resumen

Resumen del proyecto. Una cara. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres. En un máximo de una cuartilla o 300 caracteres

**Palabras clave**: un par de líneas

**Usar salto de página y sección, NO usar enter.**

Índice general

1. [Introducción y objetivos 8](#_Toc184650169)

[1.1 Introducción 8](#_Toc184650170)

[1.2 Justificación 9](#_Toc184650171)

[1.3 Objetivo General 9](#_Toc184650172)

[1.3.1 Objetivos Específicos 9](#_Toc184650173)

[1.4 Problema por resolver 11](#_Toc184650174)

[1.5 Alcances 11](#_Toc184650175)

[1.6 Limitaciones 12](#_Toc184650176)

[2.1 Definición de Sistema web 13](#_Toc184650177)

[2.1.1 Elementos de un sistema web 13](#_Toc184650178)

[2.1.1 Funcionamiento de un Sistema web 14](#_Toc184650179)

[2.1.2 Tipos de sistema web 14](#_Toc184650180)

[2.2 Auditoria por capas 14](#_Toc184650181)

[2.2.1 Concepto de auditoria por capas. 15](#_Toc184650182)

[2.2.2 Ejemplo de auditoria por capas. 15](#_Toc184650183)

[2.2.3 Beneficios de las auditorias por capas. 15](#_Toc184650184)

[2.3 Mockups 16](#_Toc184650185)

[2.3.1 Herramientas utilizadas 16](#_Toc184650186)

[2.4 Base de datos 17](#_Toc184650187)

[2.4.1 Tipos de Bases de datos 17](#_Toc184650188)

[2.4.2 Tipos de Gestores de Base de Datos. 18](#_Toc184650189)

[2.5 LARAGON 18](#_Toc184650190)

[2.6 Que es un framework 19](#_Toc184650191)

[2.6.1 Ventajas de usar un framework. 19](#_Toc184650192)

[2.7 Codeigniter 4 20](#_Toc184650193)

[2.8 PHP 21](#_Toc184650194)

[2.8.1 ¿Por qué PHP? 21](#_Toc184650195)

[2.9 MVC 22](#_Toc184650196)

[2.9.1 ¿Qué significa MVC? 22](#_Toc184650197)

[2.9.2 Porque MVC 23](#_Toc184650198)

[2.10 Java Script 23](#_Toc184650199)

[2.10.1 Características principales de Java Script 23](#_Toc184650200)

[2.10.2 Java Script en el Frontend 24](#_Toc184650201)

[2.11 HTML 25](#_Toc184650202)

[2.12 CSS 25](#_Toc184650203)

[2.12.1 Que es CSS 25](#_Toc184650204)

[2.12.2 Conceptos claves sobre CSS 26](#_Toc184650205)

[2.13 Visual Studio Code 26](#_Toc184650206)

[2.13.1 ¿Por qué Visual Studio Code? 27](#_Toc184650207)

[2.14 Heidi SQL 27](#_Toc184650208)

[2.14.1 Características de HeidiSQL. 27](#_Toc184650209)

[2.15 NAVICAT 28](#_Toc184650210)

[2.16 GIT 28](#_Toc184650211)

[2.16.1 ¿Qué es Git y cómo funciona? 29](#_Toc184650212)

[2.16.2 Características principales de GIT 29](#_Toc184650213)

[2.16.3 Ventajas de GIT 30](#_Toc184650214)

[2.17 GITHUB 30](#_Toc184650215)

[2.17.1 GitHub VS GIT 30](#_Toc184650216)

[2.17.2 Características principales de GITHUB 31](#_Toc184650217)

[2.17.3 Ventajas de usar GITHUB 31](#_Toc184650218)

[2.17.4 GITLENS 32](#_Toc184650219)

[2.18 Firefox Developer 33](#_Toc184650220)

[2.18.1 Ventajas de usar el navegador Firefox Developer Edicion 33](#_Toc184650221)

[2.19 Bootstrap 34](#_Toc184650222)

[2.19.1 ¿Qué es y cómo se utiliza? 34](#_Toc184650223)

[2.19.2 Características de Bootstrap. 34](#_Toc184650224)

[2.20 W3-School 35](#_Toc184650225)

[2.21 Star UML 35](#_Toc184650226)

[2.22 DataTables 35](#_Toc184650227)

[2.23 SweetAlert2 35](#_Toc184650228)

[2.24 Select2 35](#_Toc184650229)

[2.25 Filtros en CodeIgniter 4 35](#_Toc184650230)

[2.26 Migraciones en CodeIgniter 4 35](#_Toc184650231)

[2.27 Rutas en CodeIgniter 4 35](#_Toc184650232)

[2.28 Librería Dompdf 35](#_Toc184650233)

[2.29 Logs in CodeIgniter 4 35](#_Toc184650234)

[3.1 Planificación del proyecto 36](#_Toc184650235)

[3.1.1 Exploración 36](#_Toc184650236)

[3.1.2 Historias de usuario 36](#_Toc184650237)

1. [Desarrollo 43](#_Toc184650238)

[4.1 Desarrollo de la base de datos. 43](#_Toc184650239)

[4.1.1 Entidades principales 45](#_Toc184650240)

[4.1.2 Tablas de soporte y gestión. 46](#_Toc184650241)

[4.1.3 Tablas de control del proceso 47](#_Toc184650242)

[4.2 Diagramas UML “inicio de sesión” 47](#_Toc184650243)

[4.2.1 Desarrollo de Mockups e inicio de sesión. 51](#_Toc184650244)

[4.3 Diagramas UML “olvide la contraseña” 52](#_Toc184650245)

[4.3.1 Desarrollo de Mockups y vista de “olvide la contraseña”. 55](#_Toc184650246)

[4.4 Diagramas UML “Crear Usuario” 57](#_Toc184650247)

[4.4.1 Desarrollo de Mockups y Crear Usuario. 59](#_Toc184650248)

[4.5 Diagramas UML “Editar Usuario” 61](#_Toc184650249)

[4.6 Diagramas UML “Crear Auditoria” 62](#_Toc184650250)

[4.6.1 Desarrollo de Mockups y Crear Auditoria. 62](#_Toc184650251)

[4.7 Desarrollo de Mockups y Ver Auditoria. 64](#_Toc184650252)

[4.8 Desarrollo de mockups y vista HOME del usuario (Auditor). 67](#_Toc184650253)

1. [Productos del proyecto 71](#_Toc184650254)

[9.1 Contenido 71](#_Toc184650255)

[9.2 Más información 71](#_Toc184650256)

[9.3 Más información 71](#_Toc184650257)

1. [Conclusiones y recomendaciones 73](#_Toc184650258)

[10.1 Contenido 73](#_Toc184650259)

[10.2 Más información 73](#_Toc184650260)

[10.3 Más información 73](#_Toc184650261)

1. [Referencias 74](#_Toc184650262)

Índice de figuras

[Figura 1 Modelo de proceso de java script 25](#_Toc184650263)

[Figura 2. Ejemplo de GitLens en visual studio code 34](#_Toc184650264)

[Figura 3. Diagrama de base de datos relacional 46](#_Toc184650265)

[Figura 4. Diagrama de flujo "inicio de sesión" 49](#_Toc184650266)

[Figura 5. Diagrama de Casos de uso de “inicio de sesión” 50](#_Toc184650267)

[Figura 6. Diagrama de actividades de "inicio de sesión" 50](#_Toc184650268)

[Figura 7. Diagrama de secuencia de "inicio de sesion" 51](#_Toc184650269)

[Figura 8Mockup de "inicio de sesión" 52](#_Toc184650270)

[Figura 9. Vista de "inicio de sesión" 52](#_Toc184650271)

[Figura 10. Diagrama de flujo de "olvide la contraseña" 53](#_Toc184650272)

[Figura 11. Diagrama de actividades de "olvide la contraseña" 54](#_Toc184650273)

[Figura 12. Diagrama de casos de uso de "olvide la contraseña" 54](#_Toc184650274)

[Figura 13.Diagrama de secuencia de "olvide la contraseña" 55](#_Toc184650275)

[Figura 14. Mockup de "olvide la contraseña" 56](#_Toc184650276)

[Figura 15. Vista de "olvide la contraseña" 56](#_Toc184650277)

[Figura 16. Correo recibido de "olvide la contraseña" 57](#_Toc184650278)

[Figura 17. Vista de "nueva contraseña" 57](#_Toc184650279)

[Figura 18. Diagrama de flujo de "crear usuario" 58](#_Toc184650280)

[Figura 19. Diagrama de casos de uso de "crear nuevo usuario" 59](#_Toc184650281)

[Figura 20. Diagrama de secuencia de "Crear nuevo usuario" 59](#_Toc184650282)

[Figura 21. Diagrama de actividades de "crear nuevo usuario" 60](#_Toc184650283)

[Figura 22. Mockup de crear nuevo usuario 60](#_Toc184650284)

[Figura 23. Vista del formulario de "crear nuevo usuario" 61](#_Toc184650285)

[Figura 24. Vista de los usuarios creados y enlistados. 61](#_Toc184650286)

[Figura 25. Vista de crear nuevo usuario. 61](#_Toc184650287)

[Figura 26. Mockup de “editar usuario”. 62](#_Toc184650288)

[Figura 27. Listado de usuarios para "editar" un usuario 62](#_Toc184650289)

[Figura 28. Modal de editar un usuario 63](#_Toc184650290)

[Figura 29. Mockup de "crear auditoria". 63](#_Toc184650291)

[Figura 30. Vista de "crear auditoria" 64](#_Toc184650292)

[Figura 31. Vista de una auditoria con integración de datos 65](#_Toc184650293)

[Figura 32. Mockup de "ver auditoria". 66](#_Toc184650294)

[Figura 33. Auditoria creada y activa. 66](#_Toc184650295)

[Figura 34. Mockup de "Detalles de la auditoria" 67](#_Toc184650296)

[Figura 35. Vista de "Detalles de la auditoria" 67](#_Toc184650297)

[Figura 36. Auditoria respondida con acciones tomadas y seguimiento. 68](#_Toc184650298)

[Figura 37 Mockup del auditor "Home" 69](#_Toc184650299)

[Figura 38. Vista del auditor "Home" 70](#_Toc184650300)

[Figura 39. Mockup donde se muestran las auditorias 71](#_Toc184650301)

[Figura 40. Vista de auditorías activas 71](#_Toc184650302)

Índice de tablas

[Tabla 1. Tipos de bases de datos 17](#_Toc184645640)

[Tabla 2. Gestores de bases de datos 18](#_Toc184645641)

[Tabla 3. Historia de usuario del administrador 37](#_Toc184645642)

[Tabla 4. Historias de usuario del usuario 39](#_Toc184645643)

[Tabla 5. Historias de usuario del supervisor 41](#_Toc184645644)

Capítulo 1

Introducción y objetivos

* 1. Introducción

**Dart de México, S. de R.L. de C.V.**, ubicada en el Parque Industrial Atlacomulco de Fabela, es una subsidiaria de Dart Container Corporation, con sede en Estados Unidos. Esta empresa se especializa en la fabricación de envases y utensilios desechables, como vasos de diferentes tamaños, contenedores y vajillas. Los productos fabricados en **DART** se elaboran principalmente con materiales como foam (poliestireno expandido) y papel, que dentro de la empresa se conocen comúnmente con ese nombre.

Desde la recepción de la materia prima, la empresa lleva a cabo rigurosos procesos de fabricación para garantizar la calidad de sus productos. Además, **DART** es una organización altamente automatizada, con sistemas operativos funcionando las 24 horas del día, lo que asegura una producción continua y eficiente.

Asimismo, la empresa busca constantemente mejorar la rapidez y eficiencia de sus procesos de trabajo. En este sentido, se considera fundamental el área de **Excelencia Operativa**, que tiene como objetivo garantizar que el trabajo se realice de manera eficiente, sin generar riesgos y siguiendo el proceso adecuado.

El **ingeniero en sistemas** desempeña un papel clave en el desarrollo de diversos sistemas, así como en el mantenimiento de la red, los equipos y los servidores dentro de la planta. En esta ocasión, se desarrollará el sistema de **"Auditoría por Capas"**, esta se desarrolla una vez por semana sobre hojas de papel en diferentes departamentos dentro de la empresa y es muy importante realizar estas auditorias ya que ayuda permite prevenir riesgos y solucionar problemas de manera efectiva dentro del área y así obtener los mejores resultados en sus productos y personal.

* 1. Justificación

La implementación de un sistema automatizado en base a las **auditorías por capas** es esencial para optimizar el control de procesos organizacionales en seguridad, calidad y productividad. Actualmente, la captura manual de datos consume tiempo, es propensa a errores y dificulta la toma de decisiones en tiempo real.

La automatización reducirá el tiempo de recolección de información, facilitará el seguimiento de acciones correctivas y preventivas, y acelerará la implementación de mejoras. Además, fortalecerá la retroalimentación entre niveles organizacionales, involucrando activamente a gerentes y supervisores, lo que fomentará una cultura de mejora continua y asegurará el cumplimiento de estándares. Este sistema es clave para optimizar la gestión de auditorías, mejorar el rendimiento organizacional y mantener la excelencia operativa.

* 1. Objetivo General

Desarrollar un sistema web que facilite la captura, control y monitoreo de información relacionada con el cumplimiento de procedimientos en áreas clave como seguridad, calidad y productividad, facilitando a gerentes y supervisores la detección y corrección de errores en los procesos organizacionales (Auditoria Por Capas).

### Objetivos Específicos

1. **El sistema será compatible con dispositivos móviles** asegurando que puedan realizar las auditorias en cualquier parte de la empresa y en ocasiones tomar evidencia.
2. **El sistema tendrá un sistema jerárquico** donde un usuario(auditor) podrá heredarle responsabilidades a su supervisor directo para solucionar los problemas.
3. **Implementar una base de datos centralizada** para almacenar los registros de auditorías realizadas, así como otros datos relevantes asegurando la eficiencia y funcionalidad además se su recuperación de datos para futuras requisiciones.
4. **Permitir el monitoreo en tiempo real** y facilitar la toma de decisiones en las áreas de seguridad, calidad y productividad.
5. **Desarrollar herramientas visuales como dashboard** que presenten métricas clave de desempeño, fortaleciendo el análisis organizacional.
6. **Facilitar el seguimiento de los hallazgos** detectados durante las auditorías y proporcione un mecanismo que permita a los líderes seguir el curso de acción necesario para corregir errores.
7. **Optimizar la retroalimentación** en todos los niveles de la organización para que los gerentes y supervisores puedan tomar decisiones rápidamente.
8. **Enviar notificación mediante correos** para que cada que inicie una auditoria o se suscite algún problema los usuarios estén alertados de los problemas.
   1. Problema por resolver

La ejecución manual de auditorías por capas dentro de **DART** consume un tiempo considerable en la captura de información, lo que dificulta la gestión eficiente de los hallazgos. Además, la falta de un mecanismo para registrar y almacenar evidencias de los problemas detectados impide realizar un seguimiento preciso de los resultados a lo largo de los diferentes procesos en las áreas de seguridad, calidad y producción, etc.

La ausencia de automatización también limita la capacidad de gerentes y supervisores para monitorear el cumplimiento de los procedimientos de manera eficiente, lo que provoca retrasos en la implementación de acciones correctivas y limita la mejora continua. Como consecuencia, esto afecta la agilidad y efectividad necesarias para optimizar los procesos organizacionales.

* 1. Alcances

1. El sistema web contará con una base de datos centralizada para almacenar toda la información de las auditorías, como evidencias, hallazgos, y acciones correctivas o preventivas, asegurando su fácil acceso y gestión.
2. Permitirá la captura y monitoreo de datos en tiempo real, mejorando la eficiencia del proceso de auditorías en diferentes áreas y departamentos de la empresa, y facilitando la detección oportuna de errores.
3. Incluirá un Dashboard que presentará métricas y estadísticas clave sobre las auditorías realizadas, proporcionando información para la toma de decisiones.
4. El sistema se desarrollará de manera local, garantizando un entorno controlado para pruebas y ajustes.
5. El código del sistema se implementará bajo la arquitectura Modelo-Vista-Controlador (MVC), asegurando una estructura ordenada, legible y fácil de mantener.
6. El desarrollo se realizará utilizando CodeIgniter 4 como framework base, con JavaScript para el frontend y PHP para el backend.
7. Los usuarios podrán acceder al sistema según los privilegios otorgados por el administrador, lo que les permitirá realizar tareas específicas dependiendo de su rol.
8. El sistema estará diseñado para ser accesible desde cualquier dispositivo con capacidad de navegar por la WEB.
9. Los usuarios en general no podrán acceder al sistema a través de la URL a menos que hayan iniciado sesión.
10. El sistema enviara correos a los diferentes usuarios cuando inicie una nueva auditoria y cuando se encuentre algún problema

## Limitaciones

1. Inicialmente, el sistema se integrará y probará localmente, siguiendo las especificaciones y condiciones definidas por el cliente.
2. El sistema estará enfocado únicamente en la gestión de auditorías por capas, sin incluir otros procesos organizacionales.
3. Los usuarios finales requerirán capacitación inicial para usar el sistema de manera eficiente.
4. El sistema no tendrá integradas reglas en base al servidor por lo que el sistema no nos solicitará datos precisos.

Capítulo 2: Marco teórico

* 1. Definición de Sistema web

Un sistema web se puede definir como una plataforma tecnológica que opera en un ambiente de internet o en algunos casos localmente y posee la habilidad de proporcionar servicios o características a los usuarios mediante navegadores web. Estos sistemas pueden incluir desde sencillos sitios remotos hasta aplicaciones sofisticadas que necesitan procesamiento tanto en el servidor como en el cliente. Los sistemas web desempeñan un rol vital en la vida cotidiana, simplificando desde el acceso a datos hasta la ejecución de actividades complicadas como las compras en línea, la administración de empresas o las plataformas de educación

### Elementos de un sistema web

Un sistema web común consta de tres capas fundamentales:

1. **Cliente (Frontend):** Esta capa es la plataforma con la que el usuario interactúa. Se crean empleando tecnologías como HTML, CSS y JavaScript, las cuales se utilizan para su desarrollo. La cual permite crear vistas atractivas e interactivas.
2. **Servidor (Backend):** La capa del servidor es el lugar donde se administran las operaciones lógicas y el manejo de información. Emplea tecnologías como PHP, JAVA, PYTHON, RUBY, etc. Las cuales funcionan para manejar las peticiones del cliente y proporcionar las respuestas adecuadas.
3. Base de datos: En este lugar se guarda toda la información del sistema, que incluye datos de usuarios, operaciones o contenido multimedia. Los sistemas web suelen utilizar bases de datos como MySQL, PostgreSQL, MongoDB, etc.

### Funcionamiento de un Sistema web

La operación de un sistema web se fundamenta en la interacción entre el usuario y el servidor. Cuando un usuario hace una petición mediante su navegador (por ejemplo, introduciendo una URL o enviando un formulario), esta petición es transmitida al servidor por medio del método GET, POST, PUT, etc. El cual la procesa y proporciona una respuesta. Generalmente, esta interacción se administra a través del protocolo HTTP o HTTPS.

### Tipos de sistema web

1. Sistemas de operaciones: Facilitan a los usuarios la realización de operaciones más complejas, tales como adquisiciones en línea por ejemplo mercado libre, Amazon, etc. operaciones de banca.
2. Sistemas de cooperación: Promueven la interacción entre diversos usuarios, tales como plataformas de redes sociales o de administración de proyecto.
3. Sistemas de Negocios o gestión empresarial: Son programas web destinados a administrar procesos internos de una compañía, tales como CRM (Administración de Relaciones con Clientes), ERP (Planificación de Recursos Empresariales), y otros sistemas de administración de recursos y relaciones.
4. Sistemas WEB: Son sistemas interactivos complejos que funcionan en el navegador. Se utilizan para funciones avanzadas como sistemas de gestión, comercio electrónico, redes sociales, etc. Se desarrollan con frameworks.

Existen más sistemas los cuales se ocupan dependiendo del área especifica, en este caso y para este proyecto nos enfocaremos en el último punto puesto que el sistema que se está desarrollando para que se maneje por medio de una URL y servirá para gestionar las peticiones que realicen por parte del frontend, a la lógica del backend y se consumirán datos o guardaran en la base de datos.

## Auditoria por capas

La auditoría por capas o LPA (Layered Process Audit) es un proceso que se utiliza mucho en los sectores industriales, especialmente en la manufactura o producción, para garantizar que los procesos se sigan correctamente en cada nivel de la organización. Su objetivo es detectar fallas de manera temprana, asegurando que se adhieran a los estándares de calidad, productividad, seguridad, etc. Llevan el nombre de auditorías por los distintos niveles jerárquicos o en este caso llamados “capas” dentro de la industria o empresa. Haga clic en Insertar y elija los elementos que desee de las distintas galerías. Los temas y estilos también ayudan a mantener su documento coordinado. Cuando haga clic en Diseño y seleccione un tema nuevo, cambiarán las imágenes, gráficos y gráficos SmartArt para que coincidan con el nuevo tema. Al aplicar los estilos, los títulos cambian para coincidir con el nuevo tema. Ahorre tiempo en Word con nuevos botones que se muestran donde se necesiten.

### Concepto de auditoria por capas.

La auditoría de niveles requiere que los empleados de diversos niveles, desde trabajadores hasta directivos o gerentes, auditen el mismo proceso o área en periodos de tiempo determinados. Esto posibilita que cada nivel posea una visión única, garantizando que las comprobaciones sean rigurosas y se identifiquen con rapidez las diferencias que puedan perjudicar la calidad o la seguridad del procedimiento y de esa manera poder tomar decisiones eficazmente.

### Ejemplo de auditoria por capas.

1. Primer nivel: Auditores de nivel de producción, como por ejemplo los encargados de línea, que llevan a cabo revisiones regulares diarias o semanales sobre los procesos bajo su supervisión y de esta manera encontrar posibles errores para su mejora.
2. Segundo panel: directores o encargados de área, capaces de llevar a cabo auditorías semanales o mensuales para garantizar que los supervisores cumplen con los estándares.
3. Tercer nivel: Supervisores o auditores externos, que examinan de manera regular el proceso global para identificar problemas sistémicos o fallos en el control.

### Beneficios de las auditorias por capas.

1. Identificación errores a temprana edad: Gracias a las auditorías realizadas en distintos niveles, es posible identificar y solucionar rápidamente los problemas.
2. Obligación en Todos los Escalones: Al involucrar a trabajadores de todas las áreas, se promueve la responsabilidad colectiva y el compromiso con la excelencia operativa de todas las áreas.
3. Implementación de Normas: Mejora la conformidad con los criterios de calidad y seguridad y otras áreas a través de una revisión continua.
4. Avance Constante: Las revisiones continuas generan una cultura de perfeccionamiento constante, en la que los trabajadores prestan mayor atención a los pormenores de su labor y así evitar fallas.

## Mockups

Un **mockup** es una herramienta de diseño que permite maquetar y plasmar visualmente un concepto, facilitando la comunicación de la idea del producto final. Su propósito principal es asegurar que el diseño esperado sea el entregado, evitando malentendidos en la fase de desarrollo. Los mockups son representaciones estáticas de las pantallas clave de un sistema y constituyen un paso fundamental para validar la experiencia de usuario (UX) antes de iniciar el desarrollo técnico.

### Herramientas utilizadas

En la actualidad, existen diversas herramientas, tanto gratuitas como de pago, que facilitan el desarrollo de interfaces gráficas, ya sea para sistemas web o áreas de marketing. Estas herramientas permiten optimizar el proceso de creación de vistas con el objetivo de hacer el trabajo más eficiente.

Inicialmente, los diseños de los mockups se realizaron en papel, lo que permitió a los interesados visualizar el flujo de las pantallas y aportar sus opiniones y recomendaciones para mejorar la usabilidad.

#### Excalidraw.

Es una herramienta que permite crear diagramas de manera sencilla, intuitiva y colaborativa. Se utiliza frecuentemente para crear wireframes, esquemas, y diagramas de flujo. Excalidraw se destaca por su capacidad para realizar bocetos visuales de forma rápida, facilitando la creación de prototipos preliminares. En este caso, se utilizó para digitalizar los diseños realizados en papel, permitiendo también la utilización de una paleta de colores para ofrecer una vista más realista de la interfaz final.

## Base de datos

Una base de datos es donde se organiza la información o datos a los cuales después se pueden ingresar para administrar o actualizar. Las bases de datos son una rama importante en las aplicaciones de informática como por ejemplo sistemas web o móviles puesto que gracias a que las bases de datos se pueden acumular y recuperar grandes cantidades de información de manera eficiente y así poder utilizarlos cuando el usuario lo necesite.

Existen administradores de bases de datos (DBMS) como MySQL, PostgreSQL y SQLite, las cuales son bases de datos estructuradas, por otro lado, existen las bases de datos NoSQL las cuales como por ejemplo MongoDB, Casandra, etc. Los cuales facilitan las operaciones de administrar, insertar, actualizar y eliminar información. Además, existen herramientas y entornos de desarrollo como XAMPP, WAMP, Laragon y MAMP, que facilitan a los programadores el uso de bases de datos en servidores locales.

### Tipos de Bases de datos

Existen diversos tipos de bases de datos lo cuales se utilizan ampliamente según el tipo de dato y las necesidades del usuario. A continuación, se mostrará una tabla sobre las bases de datos relacionales y no relacionales.

Tabla 1. Tipos de bases de datos

|  |  |  |
| --- | --- | --- |
| Característica | Bases de Datos Relacionales (RDBMS) | Bases de Datos No Relacionales (NoSQL) |
| Estructura de Datos | Tablas con filas y columnas | Diversos modelos (clave-valor, documentos, grafos, columnas) |
| Lenguaje de Consulta | SQL (Structured Query Language) | Varía según el modelo (por ejemplo, JSON para MongoDB, Gremlin para grafos) |
| Esquema | Esquema fijo y estructurado | Flexible y dinámico |
| Escalabilidad | Escalabilidad vertical (agregar más potencia al servidor) | Escalabilidad horizontal (agregar más nodos o servidores) |
| Transacciones | Soporta transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) | Soporta transacciones eventualmente consistentes (no todas soportan ACID) |
| Casos de Uso | Aplicaciones financieras, ERP, sistemas donde la integridad de datos es crítica | Big Data, aplicaciones en tiempo real, redes sociales, IoT |
| Ejemplos | MySQL, PostgreSQL, Oracle Database | MongoDB, Cassandra, Neo4j, Redis |
| Almacenamiento de Datos | Datos estructurados | Datos no estructurados y semi-estructurados |

### Tipos de Gestores de Base de Datos.

Como se mencionó anteriormente existen diversos tipos de bases de datos, pero también existen diversos tipos de herramientas para administrarlas. A continuación, se muestran algunos de ellos.

Tabla 2. Gestores de bases de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Características | XAMMP | LARAGON | WAMP | MAMP | EASYPHP |
| Compatibilidad de Sistemas | Windows, macOS, Linux | Windows | Windows | macOS, Windows | Windows |
| Componentes Incluidos | Apache, MySQL, PHP, Perl | Apache, MySQL/MariaDB, PHP, Node.js | Apache, MySQL, PHP | Apache, MySQL, PHP | Apache, MySQL, PHP |
| Instalación | Sencilla | Muy sencilla, portátil | Sencilla, pero está limitada a Windows | Muy sencilla | Sencilla |
| Facilidad de Uso | Interfaz simple | Intuitiva y amigable | Interfaz de usuario | Intuitiva | Interfaz sencilla |
| Portabilidad | No es portable | Portable | No es portable | No es portable | Portable |
| Gestión de Versiones | Limitada | Soporta múltiples versiones | Limitada | Limitada | Limitada |
| Virtual Hosts Automáticos | No | Si | Si | Limitado | No |
| Soporte SSL | Limitada | Si, muy sencillo | Alta | Popular en MacOS | Limitada |
| Popularidad | Alta | Mediana, en crecimiento | Sí | No | Media |

## LARAGON

Laragon es un recurso potente para equipos técnicos, diseñado para facilitar la creación de diversos entornos de desarrollo, lo que simplifica enormemente la labor con aplicaciones web y locales. Esta herramienta es ideal para desarrolladores que buscan configurar rápidamente entornos listos para trabajar, ya que incluye un servidor ligero y rápido, optimizado para el desarrollo además como se mencionara a continuación su importancia.

Con Laragon, tendremos la posibilidad de trabajar con lenguajes de programación de servidor como NodeJS, Ruby, Python y, naturalmente, PHP esto por mencionar alguno de los frameworks más utilizados e importantes. Además de estos lenguajes, Laragon soporta múltiples bases de datos como MySQL, MariaDB, PostgreSQL y SQLite, MongoDB, etc. Lo que brinda flexibilidad en el desarrollo de aplicaciones de diversa naturaleza y para diferentes ámbitos dependiendo del campo en el que se trabaje.

Una de las principales ventajas de Laragon es su facilidad de uso. Además de que es portable y de escritorio, como se mencionó en la tabla anterior de los diferentes gestores de bases de datos y para desarrollo web, pero de todos los antes mencionados Laragon cumplía con los estándares gracias a que es compatible con diferentes versiones de PHP, soporta diferentes tecnologías, genera los “virtual- host” automáticamente entre otras cosas que facilitan su uso.

## Que es un framework

Un **framework** es una estructura o conjunto de herramientas que proporciona una base estandarizada para el desarrollo de software. Los frameworks contienen código predefinido que los desarrolladores pueden usar y ampliar para crear aplicaciones, reduciendo el tiempo y el esfuerzo necesario para programar desde cero. Además, un framework ofrece convenios o estándares de cómo organizar el código creando buenas prácticas y así facilitar el mantenimiento y seguridad de aplicaciones a largo plazo.

Los frameworks se utilizan en diferentes áreas de programación incluyendo el desarrollo WEB, móvil o de escritorio estos varían su complejidad y intención.

### Ventajas de usar un framework.

1. **Eficiencia:** Al proporcionar componentes y archivos ya creados y configuradospermiten acelerar en grandes cantidades el tiempo de desarrollo. Por ejemplo, no es necesario desarrollar el código desde cero para gestionar las solicitudes HTTP o conexiones a bases de datos pues el framework se encargará de eso.
2. **Mantenimiento:** Un framework establecerá reglas sobre cómo desarrollar el código para que de esta manera se obtenga un código mas completo y la colaboración entre diferentes desarrolladores.
3. **Reutilización de Código:** Al utilizar componentes y módulos ya definidos, se evita la duplicación de código en algunos segmentos del código, las personas que desarrollen sobre el código pueden utilizar o personalizar algunos segmentos del código para soluciones más completas.
4. **Seguridad:** La mayoría de los frameworks contiene características de seguridad integradas, todo esto con el fin de brindar protección contra inyecciones SQL, o cualquier otro tipo de ataque y gestionar la autenticación.

## Codeigniter 4

**CodeIgniter 4** es un framework de PHP diseñado para ofrecer a los desarrolladores una base sólida, ágil y liviana sobre la cual edificar aplicaciones web. Es conocido por su simplicidad, bajo consumo de recursos y rendimiento rápido, lo que lo convierte en una opción popular entre los desarrolladores que buscan una solución eficaz para proyectos de cualquier tamaño.

Una de las razones por las que **CodeIgniter 4** es tan utilizado es porque sigue siendo fiel a su filosofía original de proporcionar un marco mínimo y modular, que permite a los desarrolladores añadir solo las herramientas y bibliotecas que necesiten. Esto ayuda a mantener el código limpio y fácil de mantener, lo que es crucial en proyectos a largo plazo .

Además, **CodeIgniter 4** se destaca por su curva de aprendizaje baja. Para aquellos que ya están familiarizados con PHP, la transición a CodeIgniter es sencilla, lo que hace que sea accesible tanto para desarrolladores novatos como para expertos. A diferencia de otros frameworks más complejos, CodeIgniter te permite empezar rápidamente, sin configuraciones complicadas ni la necesidad de aprender patrones de diseño avanzados antes de ser productivo.

Otra razón por la cual se utiliza es su excelente rendimiento. CodeIgniter está optimizado para ser rápido, incluso en servidores de bajo costo, lo que lo convierte en una opción ideal para aplicaciones web que requieren respuestas rápidas sin consumir demasiados recursos. Su núcleo liviano permite que las aplicaciones funcionen de manera eficiente sin sobrecargar los servidores.

**CodeIgniter 4** también es utilizado por su soporte para una arquitectura MVC (Modelo-Vista-Controlador), lo que fomenta la separación del código y hace que las aplicaciones sean más fáciles de escalar y mantener. Esta arquitectura permite organizar el código de una manera estructurada, lo que simplifica el trabajo en equipo y facilita el crecimiento del proyecto con el tiempo.

Además, **CodeIgniter 4** incorpora mejoras significativas en comparación con sus versiones anteriores, incluyendo soporte para PHP 7.3 o superior, una mejor gestión de rutas, y herramientas más avanzadas para la depuración y el manejo de errores, lo que lo convierte en una opción sólida para el desarrollo de aplicaciones modernas.

## PHP

**PHP (Hypertext Preprocessor)** es un lenguaje de programación de propósito general, ampliamente utilizado para el desarrollo web. Fue creado en 1994 por Rasmus Lerdorf y, a lo largo de los años, ha evolucionado hasta convertirse en uno de los lenguajes más populares en el ámbito del desarrollo web y para mi uno de los mejores lenguajes, especialmente para el desarrollo del lado del servidor. Su popularidad proviene en gran parte de su simplicidad, maleabilidad y capacidad para integrarse fácilmente con HTML, bases de datos y otros lenguajes.

### ¿Por qué PHP?

1. **Facilidad de uso**: PHP es relativamente sencillo de aprender y escribir. Su sintaxis es clara y accesible, lo que lo convierte en una opción popular tanto para desarrolladores novatos como experimentados.
2. **Desarrollo rápido**: PHP permite a los desarrolladores crear aplicaciones web dinámicas de manera rápida. Gracias a su capacidad de integrarse fácilmente con HTML y otras tecnologías web como CSS y JavaScript, los desarrolladores pueden crear aplicaciones completas en menos tiempo. Además de su adaptabilidad de hacia diferentes gestores de bases de datos.
3. **Código embebido en HTML**: Uno de los puntos fuertes de PHP es que puede insertarse directamente en el código HTML, lo que permite a los desarrolladores alternar entre HTML y PHP en una misma página sin complicaciones. Esta capacidad hace que PHP sea extremadamente flexible y fácil de usar para crear contenido web dinámico, puntos de venta, desarrollos hacia empresas, etc.
4. **Compatibilidad multiplataforma**: PHP es compatible con la mayoría de los sistemas operativos, incluidos Windows, Linux y macOS. Esto le permite ser ejecutado en una variedad de servidores y entornos, lo que le otorga flexibilidad para desarrolladores y empresas que trabajan con diferentes plataformas. Además, puede integrarse con varios servidores web como Apache y Nginx sin mayores inconvenientes.
5. **Rendimiento**: PHP ha mejorado significativamente su rendimiento a lo largo de los años, especialmente con las versiones más recientes como PHP 7.x y PHP 8.x, que ofrecen mejoras en la velocidad de ejecución y optimizaciones en el consumo de memoria. Esto lo hace ideal para aplicaciones que manejan grandes volúmenes de tráfico.
6. **Código abierto**: PHP es de código abierto, lo que significa que cualquier persona puede usarlo, modificarlo y distribuirlo de forma gratuita. Esto lo ha convertido en una opción preferida para startups, pequeñas empresas y desarrolladores independientes que buscan una solución económica para el desarrollo web.

## MVC

Modelo-Vista-Controlador es un patrón o arquitectura de diseño de software que se utiliza comúnmente en aplicaciones WEB. Este patrón organiza el código en tres grandes componentes principales: Modelo, Vista y Controlador, permitiendo una clara separación de responsabilidades dentro de la aplicación. Este enfoque mejora la organización, fascilita el manejo del código y permite la escalabilidad del proyecto con el paso del tiempo.

### ¿Qué significa MVC?

* **Modelo:** El modelo representa la parte de la aplicación donde maneja la lógica relacionada con los datos (consultas SQL). Es responsable de interactuar con la base de datos y como deben de ser manipulados. El modelo se define como se estructuran los datos y como deberían ser manipulados. Por ejemplo, si estamos construyendo un punto de venta, el modelo se encarga de acceder a la base de datos para gestionar, obtener, actualizar entre otras funciones para los usuarios dependiendo de lo que necesite.
* **Vista:** La vista es parte de la aplicación que gestiona la presentación de la información. En otras palabras, es la interfaz de usuario (UI) que los usuarios ven e interactúan. La vista toma datos proporcionados por el modelo y los muestra en un formato que los usuarios pueden entender, Siguiendo con el ejemplo del punto de control, la vista es donde se muestran precios e inventarios de los productos.
* **Controlador:** El controlador actúa como intermedio entre el modelo y la vista. Su tarea es recibir solicitudes del usuario a través de una URL o una acción en especifica, interactúan con el modelo para procesar los datos que se traen y luego pasarlos a la vista dinámicamente, nuevamente como ejemplo en el punto de venta el controlador se basa de gestionar las solicitudes de los usuarios, buscar, seleccionar algún producto en coordinación con la vista y el modelo en consecuencia se obtienen buenos resultados.

### Porque MVC

MVC es una arquitectura para desarrollar software más utilizado en el área de programación WEB puesto que los tres módulos (Modelo-Vista y controlador) son independientes y esto conlleva a grandes ventajas como.

* **Se separan las responsabilidades:** Al dividir el código en tres bloques es fácil deliberar responsabilidades a los desarrolladores para que cada uno pueda trabajar independientemente por ejemplo alguien podría trabajar en el backend mientras que otros en el frontend.
* **Mantenimiento:** Al tener separado cada módulo, y sobre cada módulo sus funciones cuando se le necesite dar alguna actualización esta se podrán desarrollar efectivamente puesto que el código estará organizado y separado por secciones y a largo plazo quien desee realizar cambios dentro del sistema lo realizara sin ningún problema. Por ejemplo, si se desea cambiar algo e la vista esto se podrá hacer sin modificar algo dentro del modelo o el controlador.
* **Escalabilidad:** La arquitectura MVC permite que las aplicaciones crezcan sin perder su claridad en la estructura del codigo, además si se utiliza un framework como lo que es CodeIgniter, Laragon, NodeJS, etc. Se podrá mejorar aún más sin perder claridad y si se busca que el sistema contenga nuevas funcionalidades o mejoras se podrá realizar sin mayor problema.
* **Desarrollo en equipo:** El patrón MVC se puede desarrollar además de facilitar el trabajo en equipo puesto que gracias a que se desarrolla en módulos se peuden centrar en diferentes áreas del proyecto, algunos en el backend otros en la lógica de las consultas, etc.

## Java Script

Java Script es un lenguaje de programación dinámico que se utiliza principalmente en el desarrollo para páginas que son dinámicas o interactivas y aplicaciones web modernas. Originalmente este lenguaje fue desarrollado para trabajar del lado del cliente (Servidor WEB), pero con el paso del tiempo también se ha pasado del lado del servidor. JavaScript es muy conocido por su flexibilidad, su capacidad para interactuar con el Document Obejct Modelo o por sus siglas (DOM) y su amplia adaptación a aplicaciones WEB o móviles.

### Características principales de Java Script

* **Lenguaje Interpretado:** Se ejecuta directamente en el navegador sin la necesidad de un proceso de compilación anterior. Esto lo convierte en un lenguaje rápido para desarrollar y probar código en tiempo real así facilitando las pruebas.
* **Interactividad WEB**: Es el lenguaje principal utilizado para hacer que las páginas web sean interactivas. Permite crear menús desplegables, formularios dinámicos, Tablas dinámicas, validación de entradas en el cliente, galerías de imágenes y mucho más, sin tener que recargar la página se podría decir que ejecuta código por secciones.
* **Compatibilidad con diversos navegadores**: JavaScript está soportado por todos los navegadores modernos, lo que significa que el código puede ser ejecutado en cualquier dispositivo sin importar el sistema operativo. Es compatible con Google Chrome, Firefox, Safari, Edge, entre otros.
* **Librerías:** Java Script tiene un ecosistema enorme de librerías y que constantemente se mejoran con el paso del tiempo. Existen librerías que van de la mano con frameworks para el desarrollo de interfaz de usuario las cuales han ampliado enormemente este lenguaje.
* **JSON (JavaScript Object Notation):** JavaScript también es conocido por ser la base del formato JSON, que es ampliamente utilizado para intercambiar datos entre el cliente y el servidor a través del controlador de manera eficiente. JSON es fácil de entender, ligero. Este a ganado mucha popularidad a lo largo del tiempo por su forma tan eficiente de utilizar.

### Java Script en el Frontend

JavaScript es la columna vertebral de la interactividad en las páginas web. Se utiliza para manejar eventos del usuario, manipular el DOM, y crear aplicaciones web dinámicas que pueden actualizarse sin recargar toda la página. Por ejemplo, formularios interactivos, menús dinámicos, y animaciones son comúnmente creados con JavaScript.



Figura 1 Modelo de proceso de java script

## HTML

HTML (Hypertext Markup Language) es fundamental cuando se habla de sistemas WEB pues este sirve como base sobre la que se construye las interfaces de usuario y se estructura el contenido visual en el navegador. Se trata de un lenguaje de marcado que permite organizar textos, imágenes, botones, enlaces, entre otros elementos multimedia que permiten al usuario interactuar con la información de la página de manera eficiente y clara.

HTML contiene diferentes tipos de etiquetas y elementos que permiten definir estructuras y formato a la página WEB. Por ejemplo, colocar textos en colores diferentes títulos de diferentes tamaños, enlaces que nos puedan dirigir a una página en concreto, colocar imágenes, insertar cuadros de dialogo, etc. HTML va de la mano con la vista lo cual pueden ser enlaces estáticos pero como se mencionó anteriormente con el modelo MVC podemos obtener un sistema dinámico el cual existe la oportunidad de que dependiendo de las características que el usuario necesite sobre su proyecto se pueden realizar cambios, retomando el ejemplo del punto de venta puede que tengamos una vista con la misma estructura HTML pero que presenta diferentes productos no es necesario tener ponerlos manualmente estos ya que se incrustan automáticamente sin la necesidad de modificar nada.

## CSS

**CSS (Cascading Style Sheets)** es un lenguaje utilizado para para describir la presentación, formato y diseño visual de las páginas WEB. La función principal de CSS es proporcionar como su nombre lo indica estilos y formato a los elementos de HTML por lo que permite que los desarrolladores y diseñadores creen paginas visualmente atractivas y bien organizadas. CSS muestra aspectos como colores, diferentes fuentes, tamaños de imágenes, vistas responsivas, márgenes y en generar diferentes diseños en la vista del usuario. Con esto se busca tener un código más llamativo, organizado y fácil de mantener con el paso del tiempo.

### Que es CSS

CSS es una hoja de estilo en forma de cascara que va de la mano con HTML para dar un formato más estructurado a la página WEB. Mientras que HTML define la estructura y el contenido, CSS se encarga de cómo se verá ese contenido en la pantalla de los dispositivos. Su importancia aplica en integrar múltiples elementos de una página esto con el fin de que sea llamativa, lo que facilita cambios rápidos en el diseño sin la necesidad de modificar la estructura HTML.

### Conceptos claves sobre CSS

A continuación, se muestran algunos conceptos clave sobre CSS y por qué.

* **Selectores de etiquetas:** Aplican estilos a todas las etiquetas especificas (por ejemplo, **p** para cada párrafo, **h1, h2, h3,** etc. Para los títulos).
* **Selectores de clases:** Aplica estilos a elementos con una clase en especifica, estos se utilizan utilizando el **“.”** antes de escribir el nombre de la clase, por ejemplo **.class** dentro del HTML y dentro de la hoja de estilos puedes agregarle el formato que desees.
* **Selector de ID:** Aplica estilos a un elemento con un ID único, usando el símbolo **“#”** antes de ID a utilizar por ejemplo **#Iduser.**
* **Selectores combinados:** Estos funcionan para definir estilos avanzados y dinámicos como: hover(cuando el ratón pasa sobre un elemento) entre otros elementos para darle más diseño al sistema.

En conclusión, las hojas de estilo son de suma importancia en el desarrollo WEB ya que gracias a esto podemos crear sistemas responsivos a lo que el usuario necesita además de que los estilos son adaptativos a cualquier dispositivo esto con el fin de tener una mejor eficiencia además de que es versátil y además de que se continua desarrollando.

## Visual Studio Code

Visual Studio Code es un software de editor de código gratuito, ligero y multiplataforma (funciona para Windows, MacOS y Linux) este software fue desarrollado por Microsoft y lanzado en 2015, se ha convertido rápidamente en una herramienta de suma importancia y populares ya que soporta diferentes tipos de lenguajes de códigos además de su flexibilidad, rendimiento, y su gran cantidad de extensiones disponibles que permiten que el desarrollo sea agradable y entendible. Este software no solo funciona para implementar la programación también funciona como depurador, control de versiones, terminal integrada, entre más características que hace que el trabajo sea fácil, ágil durante el proceso de desarrollo.

### ¿Por qué Visual Studio Code?

Visual Studio Code se ha convertido en una herramienta imprescindible para desarrolladores de cualquier nivel y en prácticamente cualquier lenguaje de programación. Una de sus mayores ventajas es que permite organizar y gestionar proyectos completos desde una única interfaz, ofreciendo la capacidad de explorar todas las carpetas y archivos del sistema con facilidad. Esto ayuda a mantener una visión clara del proyecto, independientemente de su tamaño.

Además, Visual Studio Code permite visualizar los cambios en tiempo real, lo que es especialmente útil al trabajar en proyectos web o aplicaciones donde es necesario ver cómo los cambios incorporados y su funcion al instante. Desde el mismo editor, también se pueden crear y organizar nuevos directorios y archivos, facilitando la estructura del proyecto sin necesidad de recurrir a herramientas externas.

Una de las características que hace destacar a VS Code es su amplio ecosistema de extensiones. Estas extensiones permiten añadir funcionalidades que ayudan a mejorar la calidad del código, como detectar errores de sintaxis, sugerencias para optimizar el rendimiento y recomendaciones de formato. Estas herramientas ayudan a los desarrolladores a corregir problemas rápidamente y a seguir buenas prácticas de desarrollo, lo que mejora la eficiencia y calidad del proyecto

## Heidi SQL

HeidiSQL es una herramienta gratuita y de código abierto para la gestión y administración de bases de datos. Es especialmente popular entre desarrolladores y administradores de bases de datos que necesitan una solución intuitiva y completa para trabajar con sistemas de gestión de bases de datos como **MySQL**, **MariaDB**, **Microsoft SQL Server**, **PostgreSQL**, y **SQLite**. Desarrollada inicialmente como un cliente para MySQL, HeidiSQL se ha expandido a lo largo de los años para incluir soporte a otros motores, lo que la convierte en una opción versátil y ligera para la administración de bases de datos.

### Características de HeidiSQL.

* **Plataforma sencilla e intuitiva:** HeidiSQL sobresale por su interfaz simple y de fácil manejo. Los programadores y gestores tienen la habilidad de desplazarse con rapidez entre tablas, bases de datos, vistas y otros elementos, sin requerir conocer lo suficiente del software. Su diseño gráfico facilita la comunicación con las bases de datos, incluso para quienes empiezan a utilizar este software y sus herramientas que incluye.
* **Soporte para múltiples motores de bases de datos**: Aunque fue diseñado originalmente para MySQL, HeidiSQL ahora acepta MariaDB, SQL Server, PostgreSQL y SQLite, lo que lo convierte en una herramienta de administración variable para aquellos que trabajan en diferentes ambientes.
* **Edición y visualización de datos en tablas**: HeidiSQL permite a los usuarios ver y editar datos de manera rápida y eficiente. Puedes editar el contenido de una tabla, añadir o eliminar registros y realizar consultas SQL directamente desde la interfaz, lo cual facilita las tareas de mantenimiento y ajuste sin la necesidad de ejecutar el navegador para visualizar la base de datos.
* **Emisión e importación de información:** La herramienta posibilita la exportación de bases de datos completas o tablas concretas a diversos formatos, tales como SQL, CSV, XML y HTML, lo que simplifica la transferencia de datos o la generación de copias de seguridad. Además, facilita la importación de datos en varios formatos, un recurso esencial para la incorporación de bases de datos en proyectos de gran envergadura.
* **Ejecución de múltiples consultas SQL**: HeidiSQL permite ejecutar múltiples consultas SQL en una misma sesión. Los usuarios pueden ver los resultados de cada consulta en pestañas separadas, lo que facilita el análisis de datos y la creación de reportes sin necesidad de abrir múltiples ventanas o cambiar de herramienta.

## NAVICAT

Navicat, junto con HeidiSQL, es un potente gestor de bases de datos fácil de utilizar, que permite desarrollar y administrar bases de datos de manera eficaz desde una única interfaz. Es compatible con una amplia variedad de sistemas de gestión de bases de datos, incluyendo MySQL, MariaDB, MongoDB, PostgreSQL, SQLite, Oracle y Microsoft SQL Server. Además, Navicat ofrece un gran soporte para la visualización de diagramas entidad-relación, lo que facilita la comprensión y diseño de la estructura de la base de datos. Durante el desarrollo del sistema, Navicat resultó ser una herramienta fundamental, ya que facilitó tanto el diseño como la administración de las bases de datos.

## GIT

GIT es un sistema de control de versiones distribuido, esta herramienta fue diseñada para rastrear y gestionar cambios en el código de proyectos de software. Git se ha convertido en una herramienta fundamental para desarrolladores y equipos de software de todo el mundo. Su fama se debe a su capacidad para gestionar proyectos de manera eficiente y colaborativa, permitiendo a varios desarrolladores trabajar en el mismo proyecto din interferir o chocar con el trabajo de los demás desarrolladores.

### ¿Qué es Git y cómo funciona?

Git permite a los desarrolladores realizar un seguimiento de los cambios en su código, volver a versiones anteriores, y trabajar en diferentes funcionalidades o "ramas" como se llama normalmente a los cambios del proyecto de forma independiente. Esto resulta especialmente útil en proyectos grandes donde múltiples colaboradores están modificando el código al mismo tiempo. Git no solo facilita la colaboración, sino que también permite mantener un historial completo de todos los cambios realizados, lo que es invaluable para depuración, auditoría y recuperación de código en caso de errores.

### Características principales de GIT

* **Distribuido**: Git es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio de código, incluido todo el historial de cambios, en su propia máquina. Esto permite trabajar sin conexión y asegura que los datos no dependan de un servidor central.
* **Control de versiones**: Git guarda un historial de todos los cambios realizados en el proyecto, lo que permite a los desarrolladores regresar a una versión anterior en cualquier momento. Esto es útil para depurar errores o deshacer modificaciones no deseadas.
* **Ramas y fusiones (branches y merges)**: Git permite crear ramas o "branches" para trabajar en funcionalidades, correcciones de errores o experimentos sin afectar el código principal. Las ramas pueden fusionarse con el proyecto principal una vez que están listas, lo que facilita el trabajo en equipo y el desarrollo de nuevas características sin interrupciones.
* **Colaboración y trabajo en equipo**: Git permite a múltiples desarrolladores trabajar en el mismo proyecto simultáneamente. Las funcionalidades de branching que permite que los colaboradores trabajen en sus propias ramas, lo que minimiza los conflictos y facilita la integración del código.
* **Comandos de Git**: Git ofrece una gran variedad de comandos para gestionar el código, como **commit** (guardar cambios), **push** (subir cambios a un repositorio remoto), **pull** (traer los cambios del repositorio remoto), **clone** (clonar un repositorio), entre otros. Estos comandos son esenciales para manipular y gestionar el código en Git.
* **Repositorios remotos**: Git permite trabajar con repositorios remotos, alojados en plataformas como **GitHub**, **GitLab** y **Bitbucket**, lo que facilita el acceso y la colaboración de equipos distribuidos. Estas plataformas ofrecen además herramientas adicionales para gestión de proyectos, como la creación de issues, wikis y revisiones de código.

### Ventajas de GIT

* **Historial detallado**: Git guarda un historial completo de todos los cambios en el código, lo que facilita la revisión, el deshacer errores y la restauración de versiones anteriores.
* **Eficiencia para trabajo en equipo**: Git facilita la colaboración en proyectos de gran escala, permitiendo a los desarrolladores trabajar en diferentes ramas y fusionar cambios sin conflictos importantes.
* **Desempeño rápido**: Al ser distribuido, Git permite trabajar en el proyecto y realizar commits sin conexión a internet, haciendo que el trabajo local sea rápido y eficiente.

## GITHUB

**GitHub** es una plataforma de desarrollo y colaboración basada en la web que utiliza **Git**, el sistema de control de versiones distribuido, para facilitar la gestión de proyectos y el trabajo en equipo. GitHub permite a los desarrolladores almacenar, compartir y colaborar en el código de manera agrupada, lo que es fundamental para proyectos de software, especialmente cuando involucran a múltiples colaboradores. Además de sus capacidades de alojamiento de repositorios Git, GitHub ofrece una amplia gama de herramientas de gestión de proyectos, automatización y colaboración, lo que la convierte en una de las plataformas preferidas para desarrolladores y empresas de software en todo el mundo.

### GitHub VS GIT

Es importante mencionar que estas tecnologías no son lo mismo a continuación se menciona por qué.

1. **GitHub** es una plataforma basada en la web que permite alojar repositorios Git y colaborar en línea. GitHub facilita compartir el código y colaborar con otros, además de ofrecer herramientas de gestión de proyectos como issues, pull requests, y revisiones de código. Otras plataformas similares a GitHub incluyen **GitLab** y **Bitbucket**.
2. **Git** es la tecnología de control de versiones que se ejecuta en tu máquina local, permitiendo gestionar el historial de cambios de tu proyecto.

### Características principales de GITHUB

* **Repositorios**: En GitHub, el código de un proyecto se almacena en un **repositorio** (o repo), que actúa como un contenedor para todos los archivos y el historial de cambios del proyecto. Los repositorios pueden ser **públicos** (accesibles para cualquier persona) o **privados** (accesibles solo para usuarios autorizados).
* **Branches (Ramas)**: Al igual que Git, GitHub permite trabajar en **ramas** o "branches" del proyecto. Las ramas son útiles para desarrollar nuevas características, corregir errores o realizar experimentos sin afectar el código principal. Cada rama se puede fusionar con el código principal mediante pull requests una vez que esté lista.
* **Pull Requests (PR)**: Los pull requests son una de las características más poderosas de GitHub. Permiten a las personas que desarrollan algún sistema proponer cambios en el código desde una rama, y que otros miembros del equipo pueden revisar, comentar y aprobar los cambios antes de que se fusionen con la rama principal del proyecto. Los PR son esenciales para mantener el código limpio y de alta calidad.

### Ventajas de usar GITHUB

* **Facilita la colaboración en equipo**: GitHub es ideal para trabajar en equipo gracias a herramientas como pull requests, issues y revisiones de código, que organizan y simplifican la colaboración.
* **Visibilidad y control**: Los desarrolladores pueden llevar un registro detallado de los cambios, ver el historial completo del proyecto y revisar el estado de las contribuciones, lo que mejora el control y la transparencia del proyecto.
* **Seguridad y control de acceso**: GitHub permite configurar repositorios públicos y privados y establecer permisos para proteger el código de accesos no autorizados.

### GITLENS

**GitLens** es una extensión para Visual Studio Code que mejora las capacidades de manejo de Git dentro del editor. Proporciona una serie de características avanzadas que facilitan la visualización y comprensión del historial de versiones de un proyecto.

#### ¿Qué hace GitLens?

**GitLens** extiende las funcionalidades de Git en el editor de código Visual Studio Code al ofrecer información detallada sobre el historial de cambios, autores, commits y ramas. A diferencia de las herramientas Git estándar, GitLens permite ver detalles de cada línea o bloque de código, así como que cambios se le realizan, lo cual resulta útil para comprender el contexto histórico y el flujo de trabajo del equipo. A continuación, se presentan algunas de las características principales de **GitLens.**

* **Anotaciones en línea (blame)**: GitLens permite ver quién fue el último en modificar cada línea de código directamente en el editor. Esta función, conocida como "blame", muestra el autor, la fecha y el mensaje del commit de cada línea, lo que facilita el rastreo de cambios y la identificación de posibles errores.
* **Explorador de historial y commits**: Con GitLens, puedes explorar el historial completo del repositorio, ver todos los commits y analizar los cambios realizados en cada uno. Esta vista de historial incluye detalles del commit, los archivos afectados, y las modificaciones específicas en cada archivo, lo que permite entender cómo ha evolucionado el código.
* **Comparación de ramas y commits**: GitLens permite comparar fácilmente entre diferentes ramas y commits, mostrando las diferencias entre versiones del código. Esto es útil para ver cómo una rama se diferencia de la principal (main/master) antes de realizar una fusión o para revisar los cambios entre versiones específicas.
* **Vista de cambios recientes**: La extensión muestra una lista de los archivos y cambios más recientes, permitiendo a los desarrolladores ver rápidamente las últimas modificaciones y revisar el progreso del proyecto.
* **Explorador de ramas y repositorios**: GitLens ofrece una vista completa de todas las ramas y etiquetas del repositorio, lo que facilita la navegación y permite a los usuarios cambiar de rama o realizar fusiones de forma rápida y sencilla.
* **Visualización de líneas de tiempo**: Con la línea de tiempo de GitLens, puedes ver una representación cronológica de los cambios en un archivo específico. Esto incluye una lista de commits y cambios en el archivo, mostrando la evolución del archivo desde su creación hasta el presente.
* **Anotaciones y comentarios en los commits**: GitLens permite visualizar los comentarios de cada commit y asociarlos con el código correspondiente, lo que facilita la revisión y comprensión de cada cambio. Además, puedes ver el impacto de un commit específico en diferentes archivos y líneas de código.

#### Ejemplo de GitLens en visual studio code.

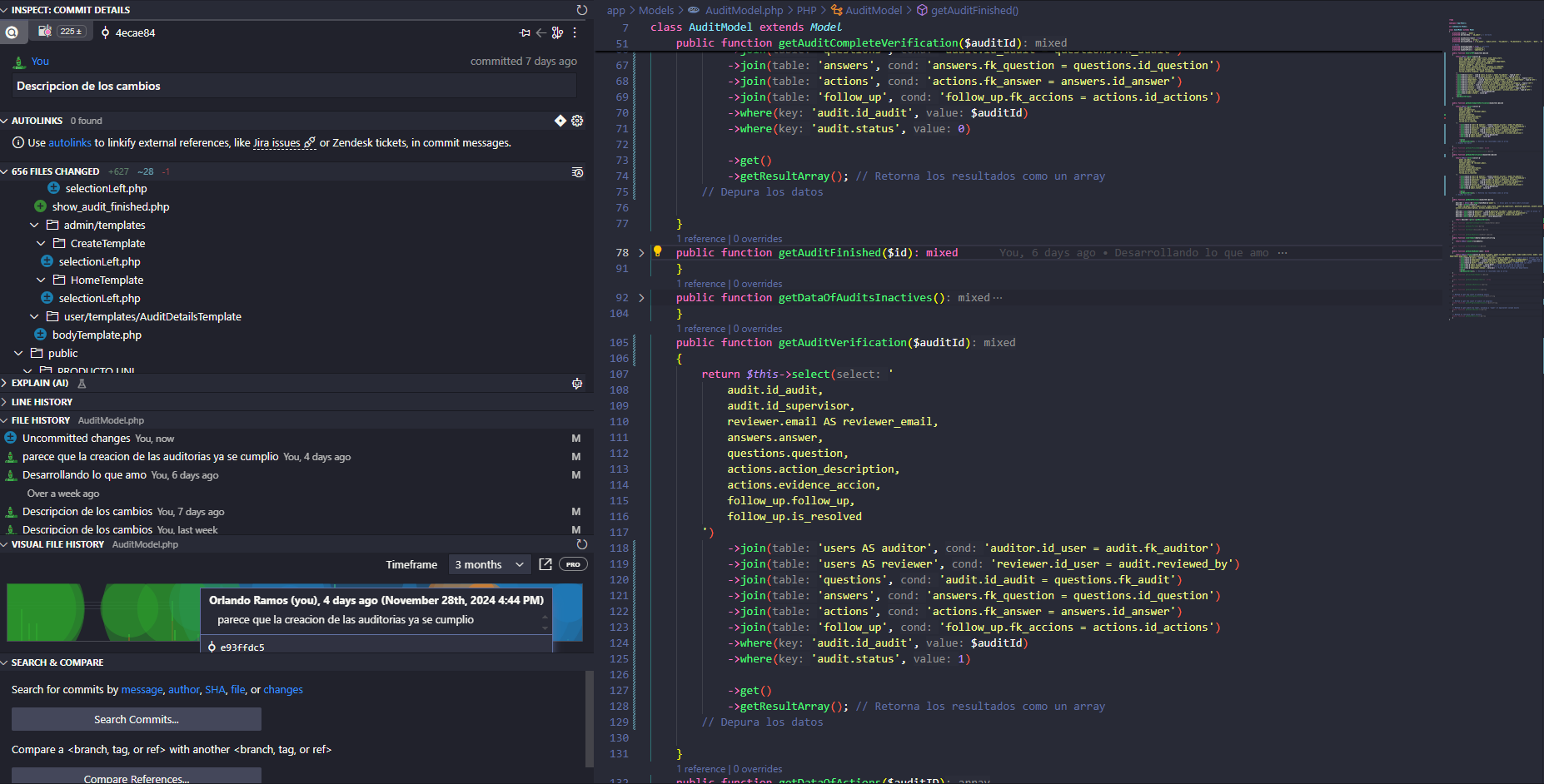


Figura 2. Ejemplo de GitLens en visual studio code

## Firefox Developer

**Firefox Developer Edicion** es una versión del navegador Firefox de Mozilla diseñada específicamente para desarrolladores web. Esta edición incluye herramientas avanzadas de desarrollo, características experimentales y soporte para estándares web emergentes, lo que permite a los desarrolladores crear, probar y depurar aplicaciones web de manera eficiente. Firefox Developer Edicion es gratuito y está utilizable en sistemas como Windows, macOS y Linux, ofreciendo un entorno optimizado para quienes trabajan en el desarrollo y depuración de aplicaciones web.

### Ventajas de usar el navegador Firefox Developer Edicion

* **Ambiente de pruebas y desarrollo especializado**: Firefox Developer Edition proporciona un entorno controlado donde los desarrolladores pueden probar sus aplicaciones web con características experimentales. Esto permite explorar nuevas APIs y tecnologías que aún no están disponibles en los navegadores convencionales.
* **Optimización para el flujo de trabajo de desarrollo**: La interfaz de Firefox Developer Edition está diseñada para facilitar el acceso a herramientas de desarrollo. La disposición de los menús y la estructura del navegador ayudan a que las herramientas sean fáciles de encontrar y utilizar, lo cual optimiza el flujo de trabajo diario de los desarrolladores.
* **Enfoque en el rendimiento**: La versión Developer Edition de Firefox está optimizada para pruebas de rendimiento. Herramientas como Firefox Profiler ayudan a analizar la eficiencia de la aplicación, facilitando la identificación y resolución de problemas de rendimiento.

## Bootstrap

**Bootstrap** es un framework de código abierto utilizado para el desarrollo de interfaces WEB responsivas y modernas. Creado originalmente en Twiter por Mark Otto y Jacob Thortnton, permite a los desarrolladores crear aplicaciones y sitios web adaptables a diferentes tamaños de pantalla, desde dispositivos móviles hasta pantallas de escritorios, con una mínima configuración y sin necesidad de diseñar desde cero. Esta herramienta se ha vuelto popular y de gran ayuda desde su lanzamiento, es ampliamente utilizada por desarrolladores, empresas y igniciones de todo el mundo.

### ¿Qué es y cómo se utiliza?

Bootstrap es una colección de herramientas y componentes CSS, JavaScript y HTML que facilitan el diseño de interfaces WEB de manera coherente y eficiente. Al ofrecer componentes predefinidos como botones, formularios, templetes, modales, etc. Bootstrap permite a los usuarios acelerar el proceso de diseño y asegurar que los elementos visuales se mantengan consistentes en diferentes secciones del código.

### Características de Bootstrap.

* **Componentes predefinidos:** Bootstrap incluye una amplia gama de componentes listos para usar, como botones, formularios, barras de navegación, tarjetas y alertas, entre otros. Estos componentes pueden personalizarse y combinan estilos y funcionalidades que facilitan la creación de interfaces coherentes y atractivas.
* **JavaScript integrado**: Además de sus componentes CSS, Bootstrap incorpora varios **plugins de JavaScript** que añaden funcionalidad interactiva a las interfaces. Estos plugins incluyen elementos como modales, carruseles de imágenes, desplegables, etc. que permiten a los desarrolladores añadir interactividad sin necesidad de codificar desde cero.
* **Compatibilidad multiplataforma y multicapa**: Bootstrap es compatible con todos los navegadores modernos y se basa en una filosofía de diseño multiplataforma. Esto significa que el diseño resultante es uniforme y consistente en diversos entornos de navegador y sistemas operativos, facilitando la accesibilidad y optimización del proyecto.

## W3-School

## Star UML

## DataTables

## SweetAlert2

## Select2

## Filtros en CodeIgniter 4

## Migraciones en CodeIgniter 4

## Rutas en CodeIgniter 4

## Librería Dompdf

## Logs in CodeIgniter 4

Capítulo 3: Metodología

Existen diversas metodologías agiles para desarrollar software, la metodología XP (Extreme Programming) es una metodología ágil de desarrollo de software desarrollada por Kent Beck autor del primer libro Extreme Programming Explained: Embrace Change (1999) [1]. Esta metodología se centrada en mejorar la calidad del software y la capacidad de respuesta ante los cambios. Se basa en valores clave como la comunicación, la simplicidad, el feedback, el coraje y el respeto. Las fases principales del ciclo de vida de un proyecto XP son las siguientes.

* 1. Planificación del proyecto

### Exploración

1. En esta fase, el equipo se familiariza con los requisitos del cliente. Se generan historias de usuario que refieren las funcionalidades del sistema desde el punto de vista del beneficiario.
2. El equipo explora diferentes opciones técnicas y prepara un plan preliminar.

### Historias de usuario

Las historias de usuario con el cliente tienen la finalidad de los casos de uso, pero con las diferencias de que solo constan de 3 o 4 líneas escritas sobre el objetivo que se desea obtener sin tener que incluir código ni dar mucho hincapié a lo que se va a realizar Estas son adecuadas para dar a conocer para tener un tiempo estimado de desarrollo y verificar que el programa se cumple. Cuando se desea desarrolla una historia de usuario los desarrolladores y el usuario se reúnen para concretar o detallar el tiempo y lo que tiene que hacer dicha historia.

#### Historias de usuario del administrador

Tabla 3. Historia de usuario del administrador

|  |  |  |  |
| --- | --- | --- | --- |
| ID | HISTORIA DE USUARIO | CRITERIOS DE ACEPTACION | PRIORIDAD |
| HU001 | Como **auditor**, deseo poder **crear auditorías** para registrar información. | * El auditor puede ingresar todos los datos necesarios. * La auditoría se guarda automáticamente en la base de datos. * El auditor recibe una notificación cuando se guarda correctamente. | Alta |
| HU002 | Como **administrador**, deseo poder **agregar usuarios** al sistema. | * El administrador puede ingresar todos los datos requeridos. * El sistema permite seleccionar información de la base de datos para facilitar el proceso. * Los datos se guardan automáticamente. | Alta |
| HU003 | Como **administrador**, deseo poder **activar y desactivar usuarios** para controlar el acceso al sistema. | * El administrador puede activar o desactivar usuarios. * Los cambios se reflejan en la base de datos y los usuarios no podrán ingresar si no están activos. | Alta |
| HU005 | Como **administrador**, deseo poder **ver las auditorías respondidas** por los usuarios. | * El administrador puede ver una lista de auditorías que ya han sido respondidas. | Alta |
| HU006 | Como **administrador**, me gustaría que el **usuario recibiera una notificación por correo** cuando tenga auditorías que no se han respondido. | * El usuario recibe una notificación automática por correo cuando tiene del total de auditorías pendientes. | Alta |
| HU007 | Como **administrador**, deseo poder **ver datos y gráficas** de las auditorías en general. | * El administrador puede visualizar gráficos con datos clave sobre las auditorías, preguntas, usuarios, etc. | Alta |
| HU008 | Como **administrador**, deseo que las **auditorías se generen automáticamente** cada semana. | * Las auditorías se crean automáticamente y se asignan a los auditores.  - El administrador no necesita crear manualmente nuevas auditorías cada semana. | Alta |
| HU009 | Como **administrador**, me gustaría poder **editar una auditoría** para agregar o quitar preguntas. | * El administrador puede editar una auditoría existente. * Los cambios se guardan automáticamente en la base de datos. | Alta |
| HU010 | Como **administrador**, deseo poder **actualizar datos de los usuarios**. | * El administrador puede seleccionar un usuario de la lista para editar sus datos. * Los datos actualizados se guardan automáticamente en la base de datos. | Media |
| HU011 | Como **administrador**, me gustaría poder **ver qué usuarios están activos e inactivos** en el sistema. | * El administrador puede visualizar los usuarios activos e inactivos en una tabla. | Media |
| HU012 | Como **administrador**, deseo **recibir una notificación por correo** cuando el usuario conteste la auditoría por capas de la semana. | * El administrador recibe una notificación cuando se completa una auditoría. * Los datos se guardan automáticamente en la base de datos. | Media |
| HU013 | Como **administrador**, me gustaría poder **ver todas las preguntas activas** en la base de datos. | * El administrador puede ver una lista de todas las preguntas activas en el sistema. | Baja |
| HU014 | Como **administrador**, me gustaría poder **activar o desactivar preguntas** en las auditorías. | * El administrador puede activar o desactivar preguntas, y los cambios se actualizan automáticamente en la base de datos. | Baja |

#### Historias de usuario del auditor

Tabla 4. Historias de usuario del usuario

|  |  |  |  |
| --- | --- | --- | --- |
| ID | HISTORIA DE USUARIO | CRITERIOS DE ACEPTACION | PRIORIDAD |
| HU001 | Como auditor deseo poder ingresar al sistema de auditorías. | * El usuario ingresara sus credenciales para poder ingresar al sistema. | Alta |
| HU002 | Como auditor deseo ver mis auditorias pendientes | * Cuando el usuario inicie sesión podrá ver un menú con las auditorias pendientes | Alta |
| HU003 | Como auditor deseo poder responder las preguntas correspondientes a la auditoria | * El usuario podrá obtener las preguntas correspondientes a su auditoria y responderlas | Alta |
| HU004 | Como auditor deseo que de preferencia se pueda tomar una foto cada que una pregunta no se cumpla. | * Cada que una pregunta no se cumpla podrá abrir su cámara y tomar una fotografía para evidenciar el problema | Alta |
| HU005 | Como auditor deseo que las preguntas que no se cumplan tengan una selección de retroalimentación o acciones tomadas | * Cuando una pregunta no se cumpla podrá verla en la sección de “Acciones tomadas” * Podrá responder con la acción que se implementará. | -Alta |
| HU006 | Como auditor puedo seleccionar un supervisor para dar a conocer los problemas que se encuentran durante la auditoria. | * El auditor que realice la auditoria podrá pasarle la auditoria a su supervisor para que verificar y seguir las acciones tomadas | Alta |
| HU007 | Como auditor deseo poder agregar un comentario cuando una pregunta no se cumpla | * El auditor agregara comentarios sobre las preguntas que no se cumplan | Media |
| HU008 | Como auditor deseo poder cerrar sesión cuando lo necesite. | * El supervisor tendrá un botón con el cual se podrá cerrar la sesión | Media |
| HU009 | Como auditor deseo que cada que me agreguen una auditoria me llegue una notificación por correo | * El auditor cada que le agreguen una auditoria podrá recibir una notificación para ver esa auditoria y responderla. | Media |
| HU010 | Como auditor me gustaría que cuando termine la auditoria la mande y no tenga nada pendiente | * Cuando el auditor termine su auditoria podrá enviarla y no tener nada pendiente o que se mantenga pendiente si aun no se responde. | Media |
| HU0011 | Como auditor deseo ver un dashboard. | * El auditor podrá visualizar un Dashboard relacionado a sus auditorias | Baja |
| HU0012 | Como auditor deseo poder agregar preguntas | * El usuario o auditor podrá agregar preguntas que le gustaría que se integraran en su auditoria * El administrador podrá ver su pregunta en su panel | Baja |

#### Historia de usuario del supervisor

Tabla 5. Historias de usuario del supervisor

|  |  |  |  |
| --- | --- | --- | --- |
| ID | HISTORIA DE USUARIO | CRITERIOS DE ACEPTACION | PRIORIDAD |
| HU0001 | Como supervisor deseo poder ver las auditorias que me han asignaron | * El supervisor podrá ver las auditorias que tiene asignadas | Alta |
| HU0003 | Como supervisor deseo poder ver las auditorias que me han mandado para darles inspección | * Como supervisor deseo poder ver los datos que me manden cuando tenga que revisar auditorias pendientes | Alta |
| HU0004 | Como supervisor deseo poder responder las preguntas de las auditorias | * Como supervisor podrá responder las preguntas de auditorías asignadas por parte del auditor | Alta |
| HU0005 | Como auditor deseo poder ver los datos que me manden para revisar | * El supervisor poder ver los datos de una auditoria pendiente o por revisar | Alta |
| HU0006 | Como supervisor deseo poder cerrar sesión | * El supervisor podrá cerrar sesión y no ingresar hasta que inicie nuevamente | Alta |
| HU0007 | Como auditor deseo poder ver mis auditorias concluida. | * El supervisor podrá ver las auditorias concluidas | Media |
| HU0008 | Como auditor me gustaría tener un Dashboard donde pueda ver algunas estadísticas sobre las auditorias | * El auditor tendrá un Dashboard donde podrá ver datos específicos de las auditorias | Media |

Capítulo 4

Desarrollo

Durante el desarrollo, se documentará el proceso de creación del Sistema de Auditoría por Capas, comenzando con el diseño del diagrama entidad-relación y avanzando hasta la implementación del sistema y su lógica. Esto permitirá comprender de manera más detallada el sistema, los inconvenientes encontrados durante el desarrollo y los resultados obtenidos.

## Desarrollo de la base de datos.

A continuación, se presenta un boceto de cómo se desarrolla las auditorias por capas dentro de **DART**, esto a fin de entender cómo se realizó el diagrama entidad relación y por su puesto la base de datos del sistema.



El primer enfoque se basa en la construcción de la base de datos. Se comenzó por definir el diagrama **entidad-relación** esto con el fin de poder organizar, gestionar y analizar cómo se desarrollaría el sistema de las auditorias, una herramienta fundamental para visualizar cómo se relacionarían las diferentes tablas y qué atributos serían necesarios con el fin de tener los necesarios para la correcta función. Esto permitió establecer una estructura sólida y coherente, siguiendo las tres formas normales para garantizar la integridad de los datos.

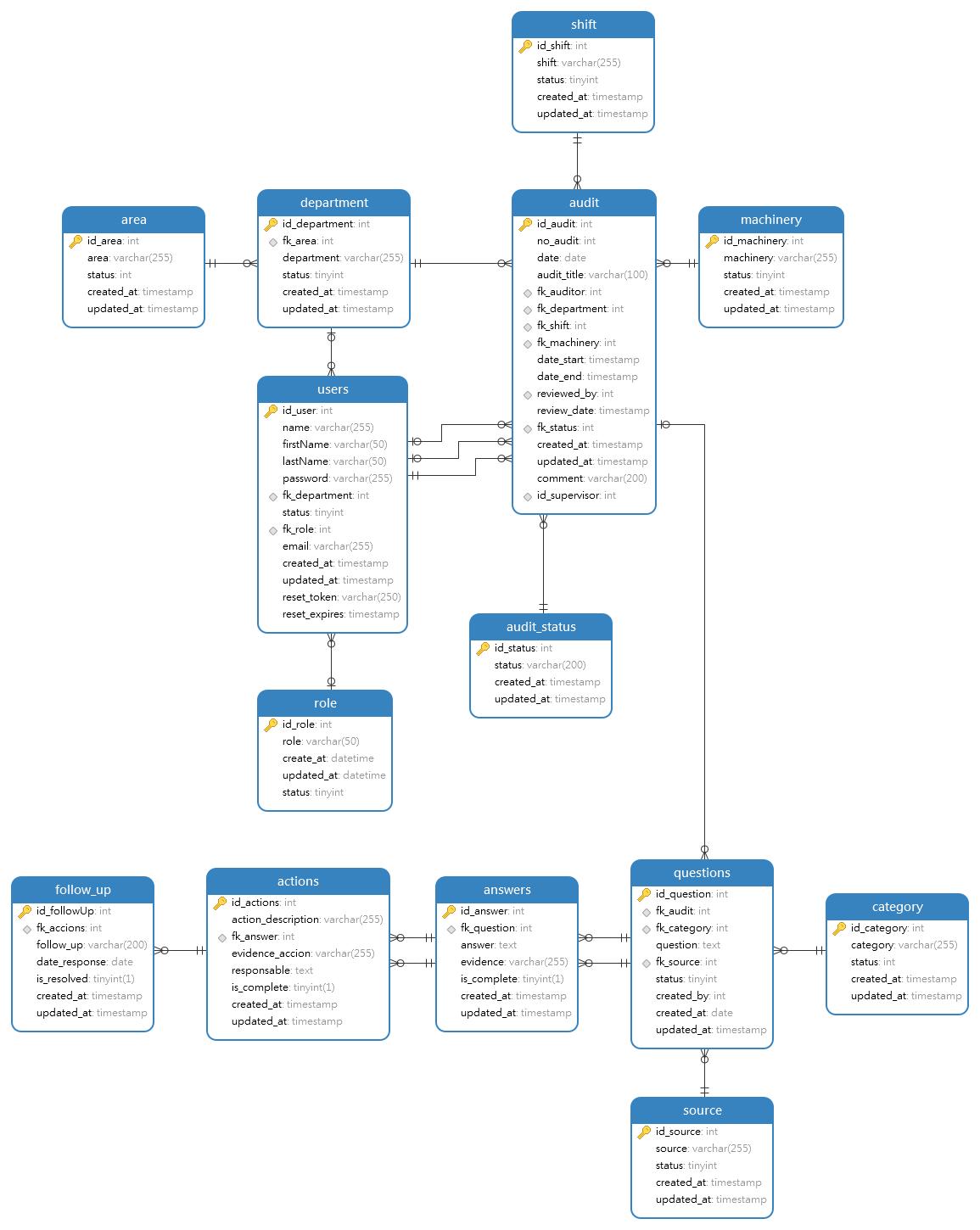


Figura 3. Diagrama de base de datos relacional

### Entidades principales

1. **audit**: Esta es la entidad central del sistema que representa cada auditoría realizada. Contiene atributos clave como `id\_audit`, `date` y `status`, entre otros. Estos atributos ayudan a identificar y gestionar cada auditoría individual y se relacionan con otras entidades que brindan información detallada sobre departament, machinery, categorías y preguntas evaluadas.
2. **question**: Define las preguntas que se aplican en la auditoría. Cada pregunta tiene su propia llave primaria y está asociada a una categoría a través del campo `id\_category`, lo que permite organizar las preguntas en grupos específicos según su categoría. También la constituye “fk\_source” la cual es de la fuente de donde proviene esa pregunta esto para tener un mejor control.
3. **category**: Representa las categorías de auditoría, como calidad, seguridad o productividad. Esta entidad facilita agrupar las preguntas relacionadas con cada tema, ayudando a estructurar el contenido de la auditoría en secciones lógicas.
4. **answer**: Almacena las respuestas proporcionadas a cada pregunta durante la auditoría. Esta entidad incluye el `fk\_question`, el `id\_question` correspondiente y otros atributos necesarios para registrar el resultado de cada evaluación.

### Tablas de soporte y gestión.

1. **área**: Define las áreas específicas de las que depende cada departamento que son auditadas. Esto permite un análisis más detallado y específico, ya que la auditoría puede realizarse en distintos sectores dentro de un área por ejemplo el área de papel tiene como departamento a calidad papel, producción papel, etc. Entonces cada departamento tendrá una auditoria.
2. **department**: Representa los distintos departamentos de la empresa en los cuales se pueden realizar auditorías. Esta entidad permite especificar el departamento auditado, facilitando la clasificación de auditorías por áreas de la organización.
3. **shift**: Representa los turnos en los cuales se realizan las auditorías. Esto permite una mejor gestión de los tiempos en los que se ejecuta cada evaluación, facilitando un seguimiento ordenado.
4. **user**: Almacena los datos de los usuarios que tienen acceso al sistema y ejecutan las auditorías. Esto incluye información como “email”, “contraseña”, “status”, que ayudan a controlar el acceso y las funciones disponibles para cada usuario para que de esta manera no cualquier usuario pueda acceder.
5. **machinery**: Representa la maquinaria sobre la cual se puede trabajar la auditoria. Esto permite que el usuario tenga un control exacto al momento de realizar su auditoria.

Todas estas son tablas o entidades que nos ayudan a tener un mejor control a la hora de realizar el sistema gracias a que no es necesario ingresar datos manualmente con el echo de seleccionar alguno de ellos se puede desarrollar una auditoria más completa además de que el proceso a la hora de realizar la misma es más sencilla para el administrador

### Tablas de control del proceso

1. **questions:** Esta tabla almacena las preguntas que deben ser respondidas por el usuario o auditor durante una auditoría. Está vinculada a las tablas **source** y **category**, las cuales son fundamentales para organizar y clasificar las preguntas. La tabla **source** es un identificador de la fuente a la que pertenece la pregunta, mientras que **category** agrupa las preguntas por temas específicos en base a la seguridad, calidad, etc. Facilitando su gestión y análisis durante el desarrollo de la auditoria.
2. **answer:** Esta es una tablaque va de la mano con la tabla pregunta esta es una tabla que nos ayuda a obtener respuestas concretas en base a las preguntas y conocer si se cumplen a fin de guardar las respuestas y saber que problemas se encuentran durante esa auditoria y poder actuar.
3. **accions:** Es una tabla que nos ayuda a dar soporte a los problemas encontrados de la auditoria, esta tabla es un punto referente para guardar las acciones que el auditor especula realizar para tener un control sobre los problemas.
4. **follow-up:** Esta tabla es la encargada de darle soporte a todo lo anterior puesto que en base a lo que los auditores comentaron que se realizaría para eliminar los problemas el supervisor del área tiene que darle un seguimiento y verificar que realmente se esté cumpliendo lo dicho.

## Diagramas UML “inicio de sesión”

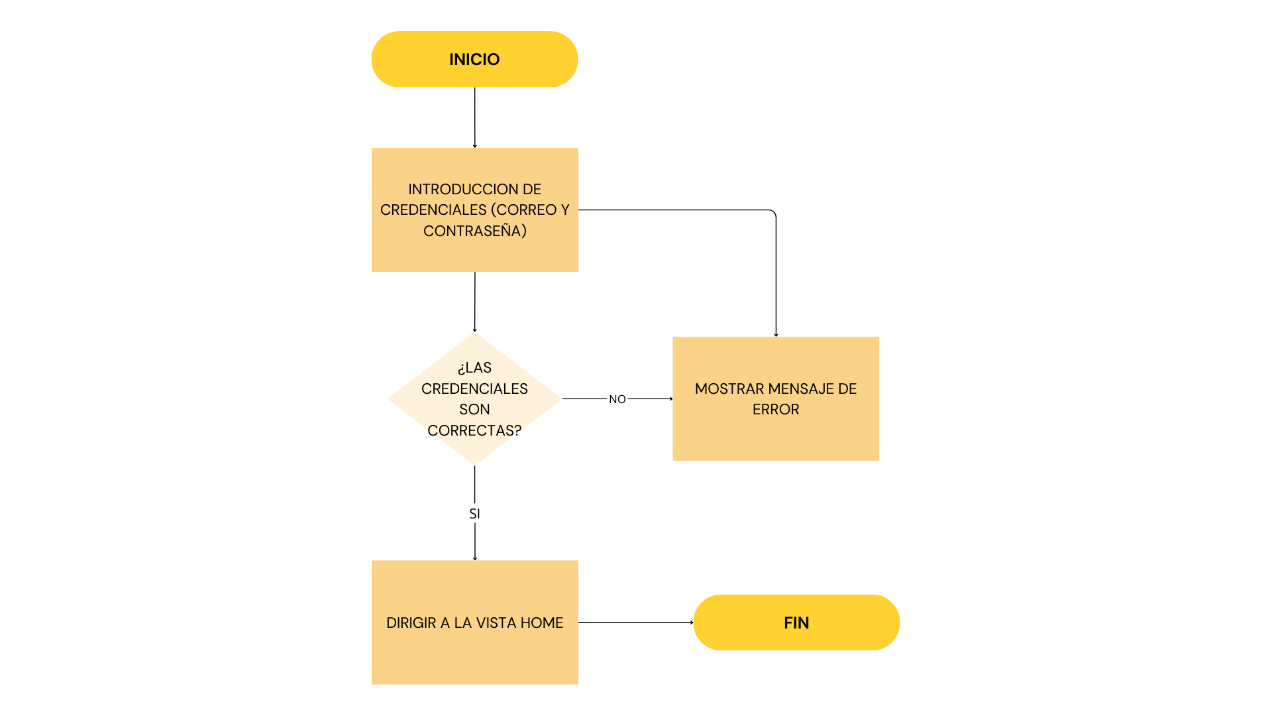
Los diagramas UML (Unified Modeling Language) son representaciones gráficas que permiten visualizar, especificar, construir y documentar los componentes de un sistema. Son una herramienta fundamental en el proceso de desarrollo de software, ya que facilitan la comunicación entre los diferentes miembros del equipo de desarrollo, permiten la comprensión de la estructura y el comportamiento del sistema, y ayudan a planificar la implementación. A continuación, se presentan los Diagramas UML ya que estos deben de considerarse para el buen desarrollo del sistema.

Figura 4. Diagrama de flujo "inicio de sesión"

1. Para el inicio se sesión como se muestra en la **figura**, el usuario que desea ingresar al sistema tiene que escribir sus credenciales (antes de esto el usuario tiene que estar registrado en el sistema). Ingresa su correo y su contraseña el sistema compara los datos introducidos con lo que esta dentro de la base de datos si el sistema encuentra los que son los correctos le permite ingresar a la vista **home** si encuentra algún error este manda una alerta y solicita nuevamente las credenciales.

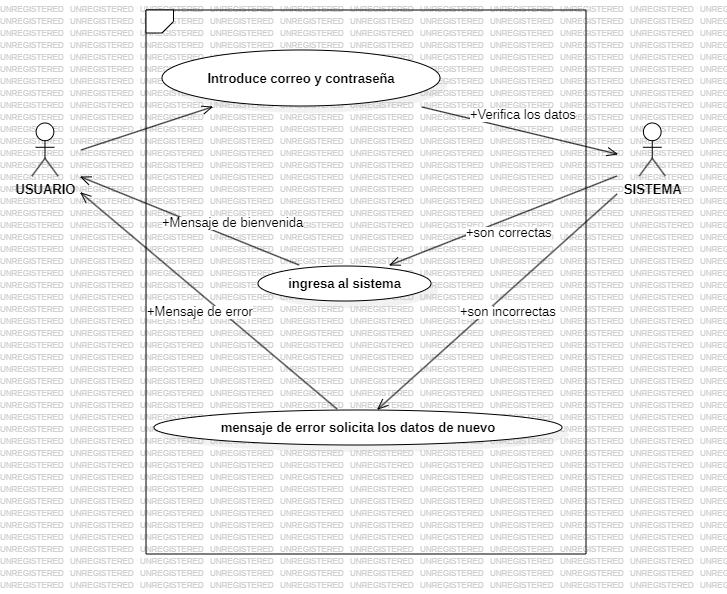


Figura 5. Diagrama de Casos de uso de “inicio de sesión”

1. El diagrama de casos como se muestra en la figura de uso es similar al de flujo lo que cambia es el usuario escribe sus credenciales el sistema los compara si son correctos se le permite el acceso al sistema si encuentra algún problema durante su verificación manda una alerta de error.

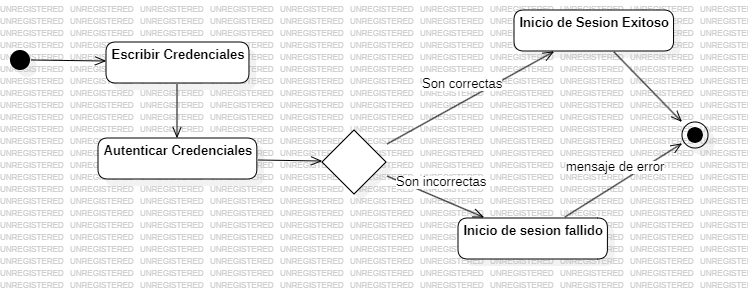


Figura 6. Diagrama de actividades de "inicio de sesión"

1. En el **diagrama de actividades** como se representa en la figura representa como se desarrolla el **inicio se sesión** de manera que da inicio con un “nodo inicial” para después mostrarnos la actividad de escribir contraseña consecuentemente realiza la actividad de autenticar las credenciales esto lo hace con una decisión o true o false donde si es verdadero nos permite ingresar en caso de se incorrecto nos manda un mensaje de error y concluir con un fin

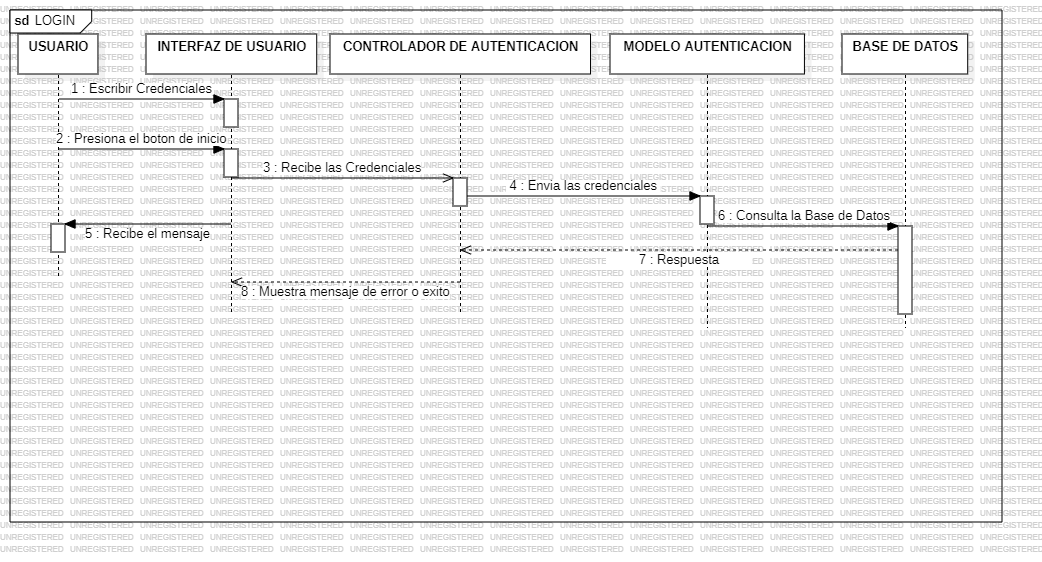


Figura 7. Diagrama de secuencia de "inicio de sesion"

En el **diagrama de secuencia** como se muestra en la figura viene un poco más desarrollado puesto que en este apartado se intenta dar a conocer cómo se comporta el sistema desde el backend hasta el frontend como actores tenemos al usuario, interfaz, controlador, modelo y la base de datos el cual la secuencia es la siguiente.

* + El usuario escribe sus credenciales las cuales pasan a la interfaz, el usuario presiona el botón de inicio
  + La interfaz recibe los datos y los manda al controlador (La parte que se encarga de mediar la vista y el modelo) este las recibe y las compara en el modelo
  + El modelo realiza una consulta a la base de datos la cual si son correctos manda un mensaje por decirlo así al controlador este la recibe la interpreta y manda un mensaje de error en caso de que durante el proceso algo este mal o lo mande a la siguiente vista en caso de estar en lo correcto.

### Desarrollo de Mockups e inicio de sesión.

A continuación, se hace representación gráfica de cómo se fue desarrollando el sistema esto a través de los mockups y la vista “final” de la vista



Figura 8Mockup de "inicio de sesión"

El primer mockup corresponde a el login o inicio de sesión donde un usuario podrá ingresar sus credenciales para entrar acceder al sistema como administrador o como otro tipo de usuario de un area especifica al ingresar los datos estos pasan a través de los pasos antes mencionados para autenticar al usuario. Véase en la **Figura .**



Figura 9. Vista de "inicio de sesión"

En la **Figura** se muestra la vista ya desarrollada de la vista de inicio se sesión. En la interfaz el sistema nos solicita el email y contraseña para poder iniciar sesión y realizar la acción de iniciar sesión.

## Diagramas UML “olvide la contraseña”

En el marco del sistema propuesto, garantizar la accesibilidad y la recuperación de credenciales por parte de los usuarios es fundamental para mantener una experiencia de usuario óptima y segura. Este proceso (Diagramas UML) se diseñó para permitir que los usuarios puedan restablecer sus contraseñas de forma autónoma, asegurando al mismo tiempo la integridad de sus datos personales.

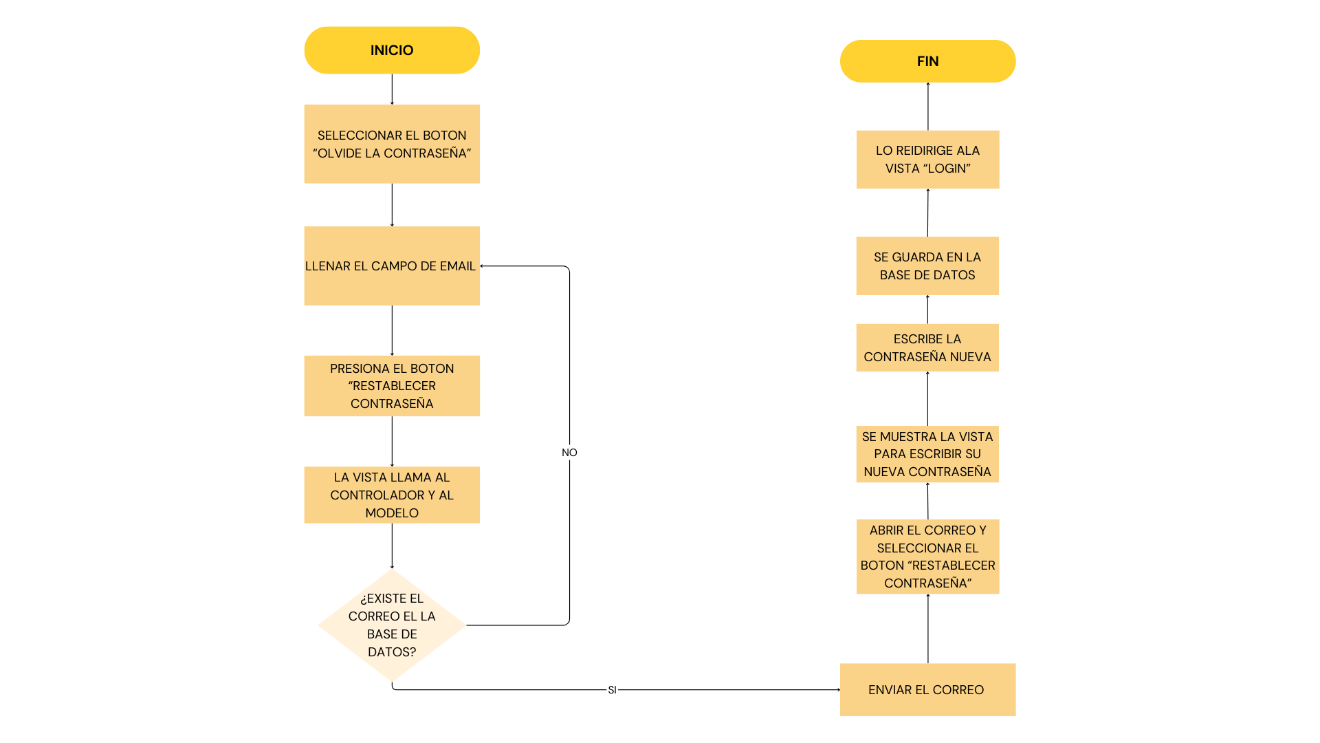


Figura 10. Diagrama de flujo de "olvide la contraseña"

El diagrama de flujo que se presenta en la imagen nos muestra el procedimiento de como se desenvuelve el sistema cuando a un usuario se le pierde la contraseña iniciando por seleccionar el botón de “olvide la contraseña”, dirige a una vista con un formulario para ingresar su correo, una vez que lo ingrese manda a llamar al controlador y al modelo, revisan si existe en la base de datos si existe le envía un correo de lo contrario le solicita un correo existente en el sistema, cuando el usuario obtiene el correo tiene que presionar el botón de recuperar contraseña, una vez hecho esto lo redirige a una vista donde le pedirá escribir su nueva contraseña al guardarla se actualizara en la base de datos y lo dirigirá al login.

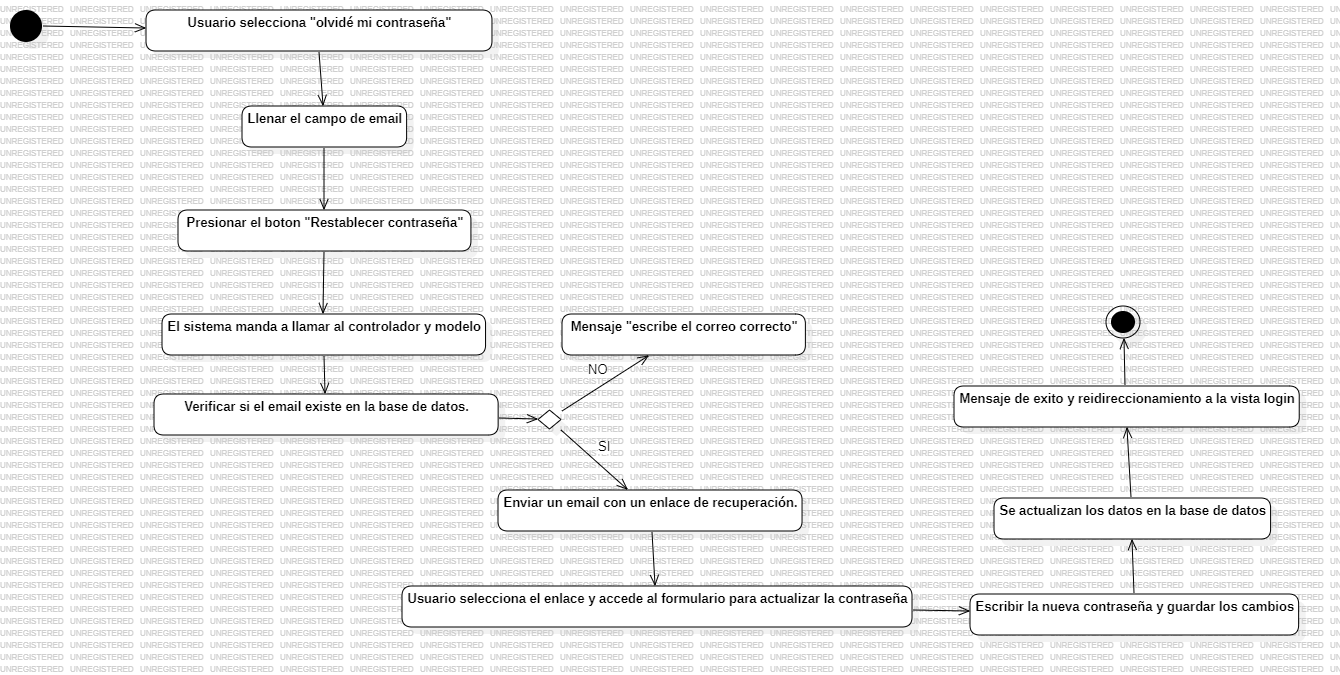


Figura 11. Diagrama de actividades de "olvide la contraseña"

El usuario selecciona la opción de olvide la contraseña, el sistema se encarga de dirigirlo a una vista donde este puede escribir el correo asociado a su cuenta despues mandarlo, el sistema se encarga de validar si es correcto, si lo es manda una liga con un token el cual le solicitara que escriba la nueva contraseña y le mostrara un mensaje de éxito y lo dirigirá a iniciar sesión, de lo contrario si no existe o es incorrecto le mostrara un mensaje donde le pedirá que revise si es correcto.

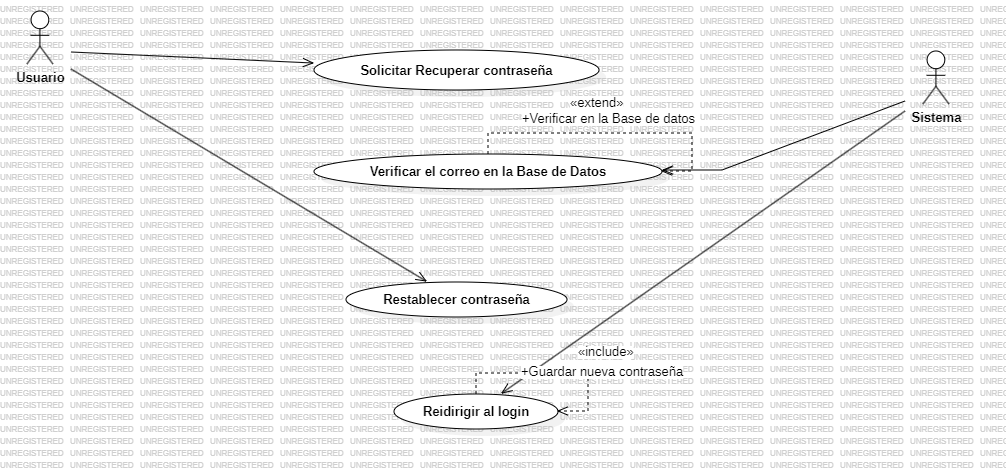


Figura 12. Diagrama de casos de uso de "olvide la contraseña"

El usuario necesita recuperar su contraseña; una vez estando en login da clic en el botón de **"Olvidé mi contraseña"**, el sistema le muestra un formulario donde escribe su correo electrónico (el que usa para iniciar sesión), el sistema revisa el correo el usuario da clic al botón de **"Restablecer contraseña"**, el sistema toma ese correo y se lo manda al controlador y despues al modelo para revisar en la base de datos si el correo está registrado, Si el correo no está registrado El sistema le dice al usuario: *"El correo no está asociado."* De lo contrario si el correo sí está registrado el sistema crea un enlace único para que el usuario pueda cambiar su contraseña se manda al correo del usuario, cuando el usuario abre el enlace el sistema lo lleva a una página donde puede escribir una nueva contraseña y actualización de la contraseña, el usuario escribe su nueva contraseña, le da clic a **"Guardar"**, y el sistema actualiza esa contraseña en la base de datos, por último, el sistema manda un mensaje de éxito y lo regresa al login.

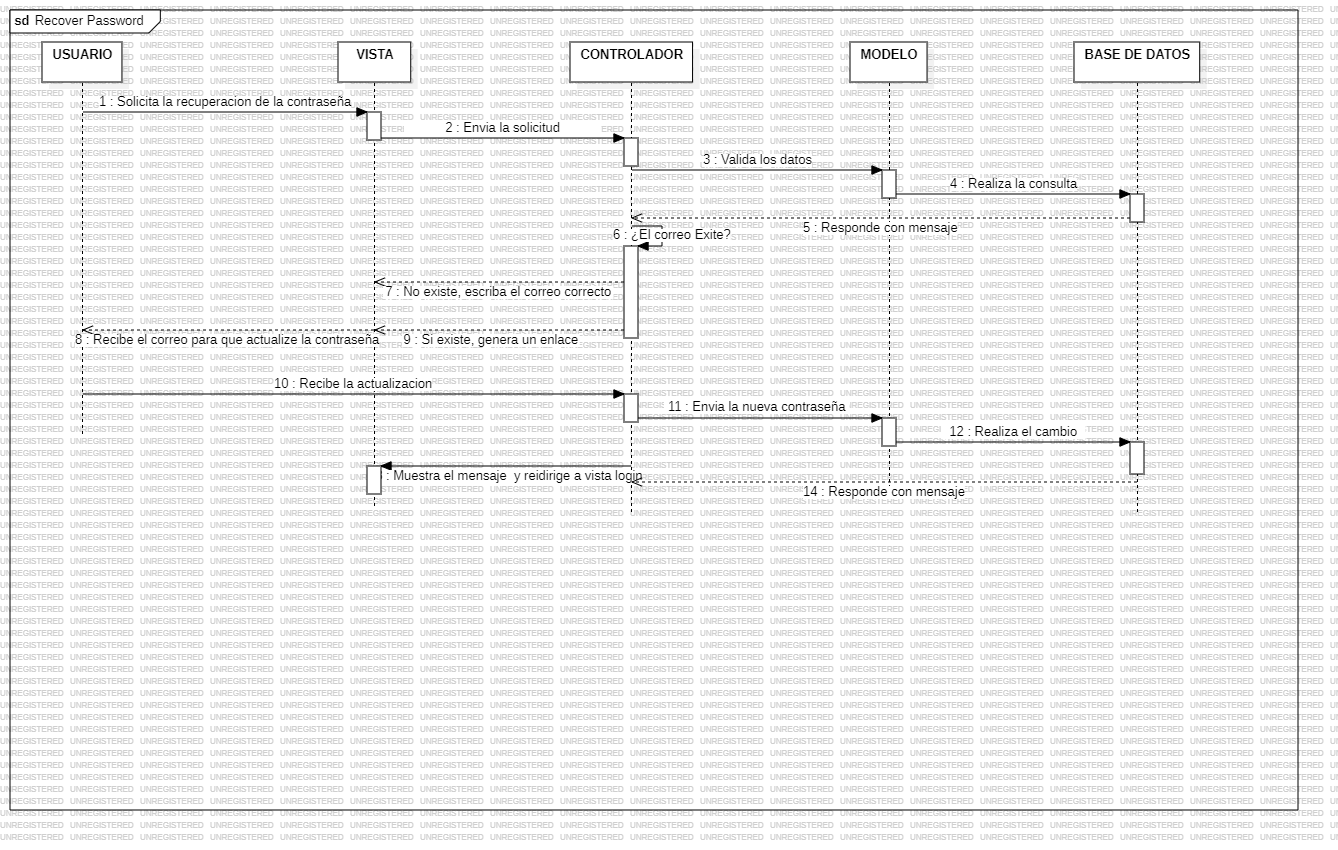


Figura 13.Diagrama de secuencia de "olvide la contraseña"

El diagrama de secuencia representa el flujo de labores que realiza el sistema para recuperar una contraseña. El proceso inicia con el usuario solicitando la recuperación desde la interfaz o vista, el sistema valida los datos en el controlador y el modelo y realiza la consulta la base de datos. Si los datos son válidos, se genera un enlace de recuperación, permitiendo al usuario restablecer su contraseña y finalizando con la redirección al inicio de sesión. Este proceso asegura la seguridad y correcta actualización de credenciales.

### Desarrollo de Mockups y vista de “olvide la contraseña”.



Figura 14. Mockup de "olvide la contraseña"

1. El mockup de la figura representa a **"Recuperar mi contraseña"**. En esta pantalla, el usuario podrá ingresar el correo electrónico asociado a su cuenta. Una vez ingresado, el sistema enviará un correo con un enlace único para que el usuario pueda restablecer su contraseña. Véase la **Figura .**



Figura 15. Vista de "olvide la contraseña"

En la **figura** se muestra la vista desarrollada de **“Recuperar contraseña”** donde el usuario debe ingresar su correo electrónico para que le mande un token y este pueda restablecer su contraseña mediante un enlace.

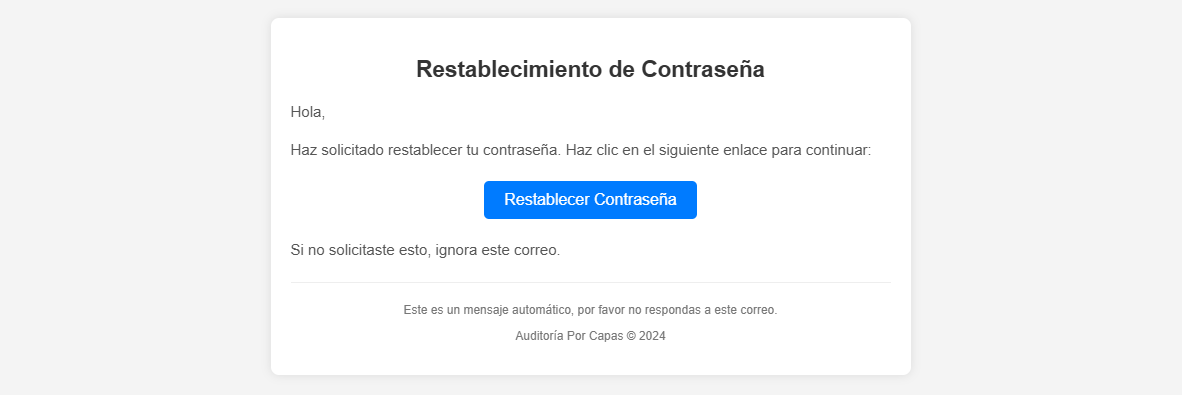


Figura 16. Correo recibido de "olvide la contraseña"

En la figura se muestra el correo que se manda cuando se solicita el restablecimiento de la contraseña, el cual va en función al correo que se integró anteriormente, por ejemplo, si el correo no esta asociado en la base de datos se mostrara una alerta con un error donde le solicitara revisar el correo enviado.

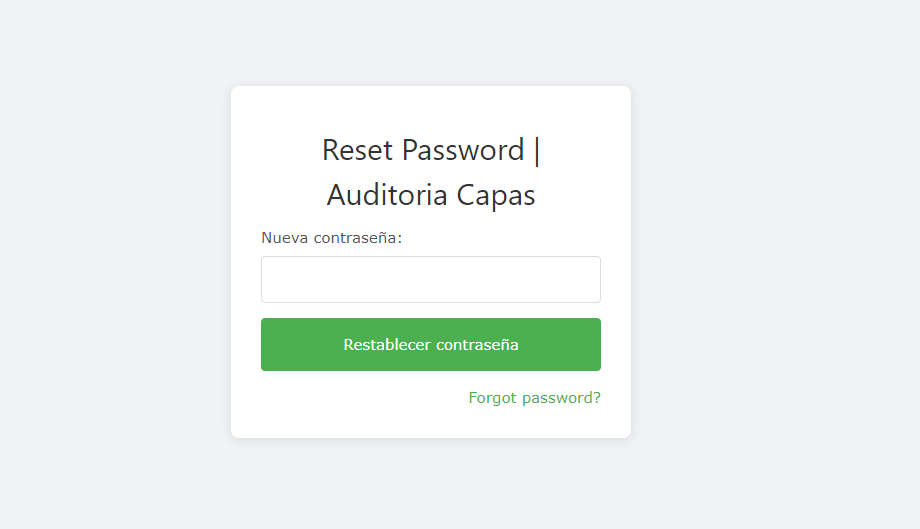


Figura 17. Vista de "nueva contraseña"

Al generarse el toquen de restablecer contraseña al usuario le abrirá una ventana como la que se muestra en la figura donde se introducirá la nueva contraseña, se enviara y se actualizara en la base de datos, para después poder ingresar con ella en el login.

## Diagramas UML “Crear Usuario”

Permite al administrador agregar un nuevo usuario al sistema para otorgarle acceso y poder realizar sus auditorias o verificar las auditorias por revisar.

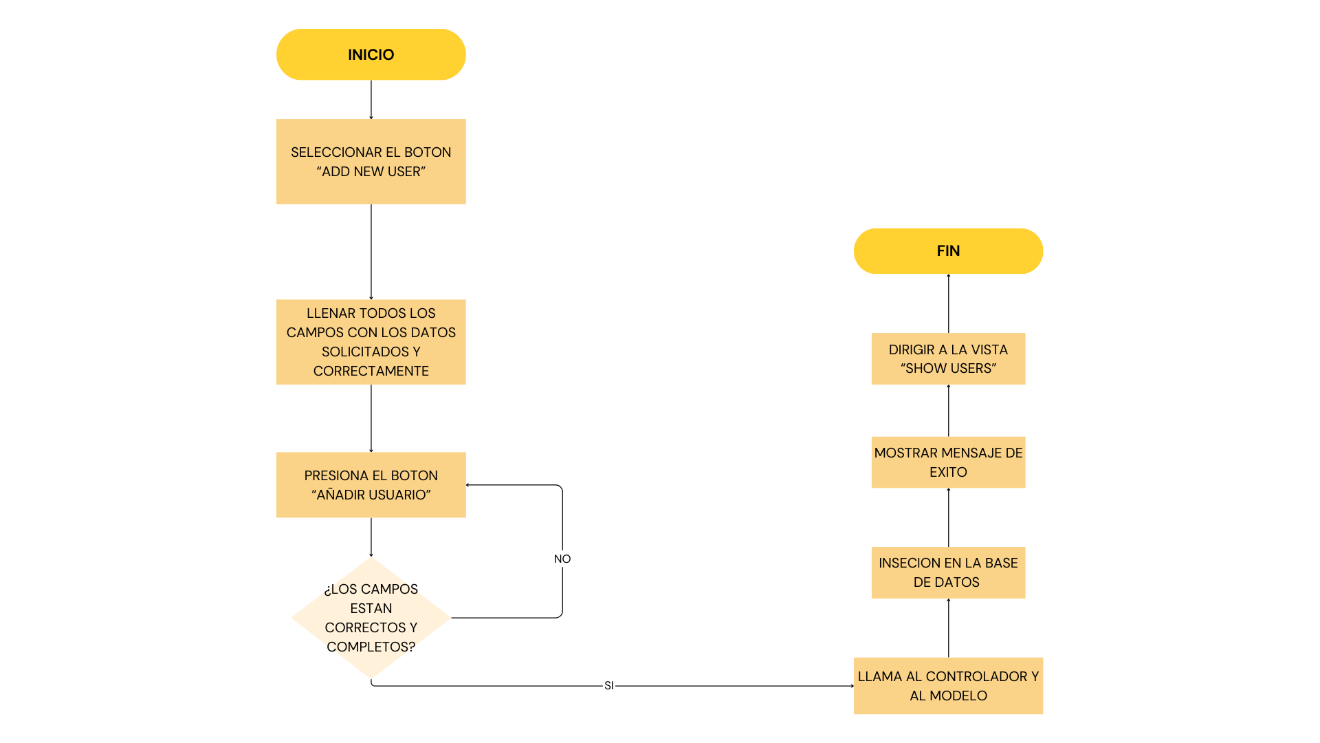


Figura 18. Diagrama de flujo de "crear usuario"

El diagrama de flujo que se muestra en la figura nos muestra cómo se crea un usuario el cual solo el administrador puede realizar esa acción. El administrador ingresa el al sistema, selecciona el botón crear usuario, llena todos los campos solicitados y selecciona los que se le presentan al presionar el botón “añadir usuario,” el sistema valida que los campos estén completos si falta alguno le solicita completarlo correctamente, y si están completos se llama la función del controlador y este al modelo la cual realiza la inserción a la base de datos retorna un mensaje de éxito.

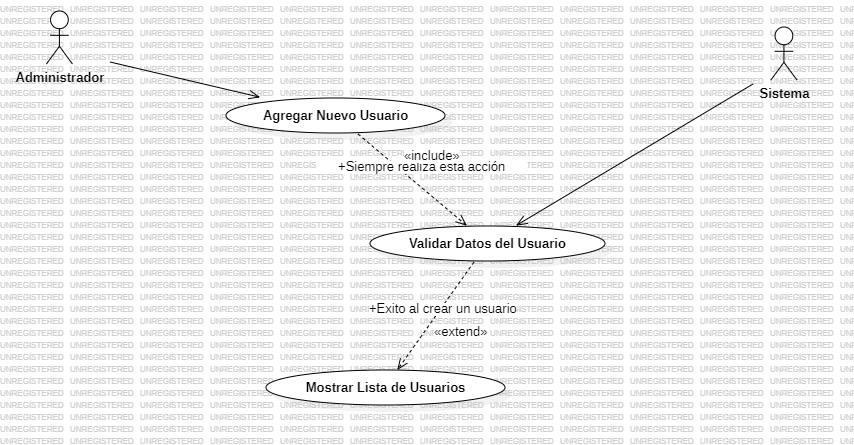


Figura 19. Diagrama de casos de uso de "crear nuevo usuario"

El administrador selecciona agregar nuevo usuario, el sistema le abre el formulario con los datos que necesita colocar, presiona el botón crear los manda al modelo este lo recibe y los mete en la base de datos los guarda y lo manda a la vista donde están todos los usuarios

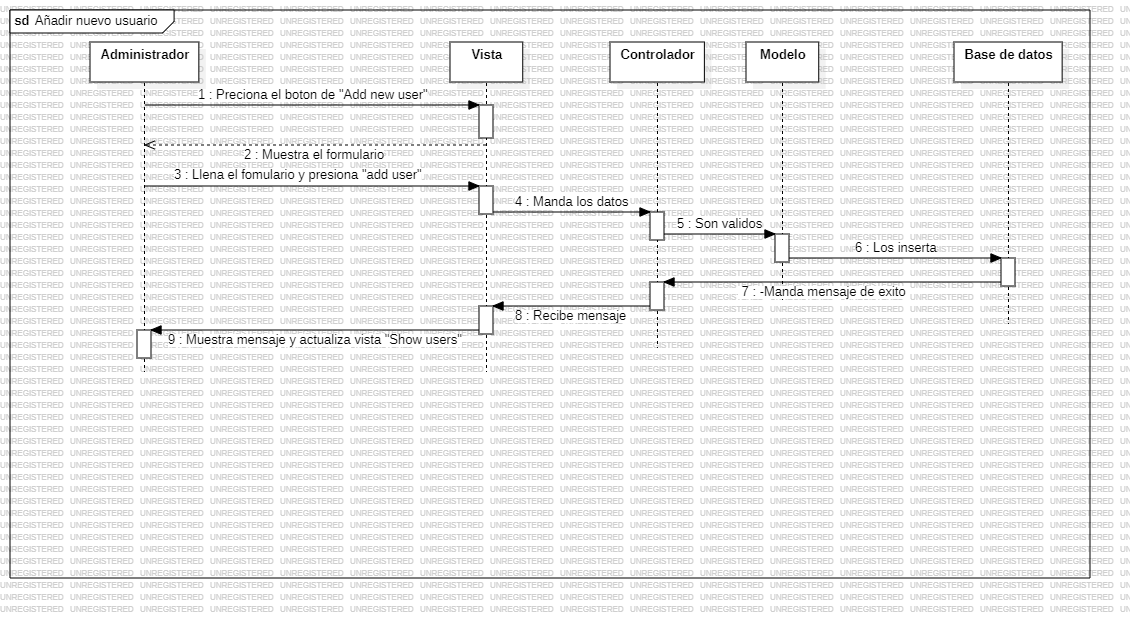


Figura 20. Diagrama de secuencia de "Crear nuevo usuario"

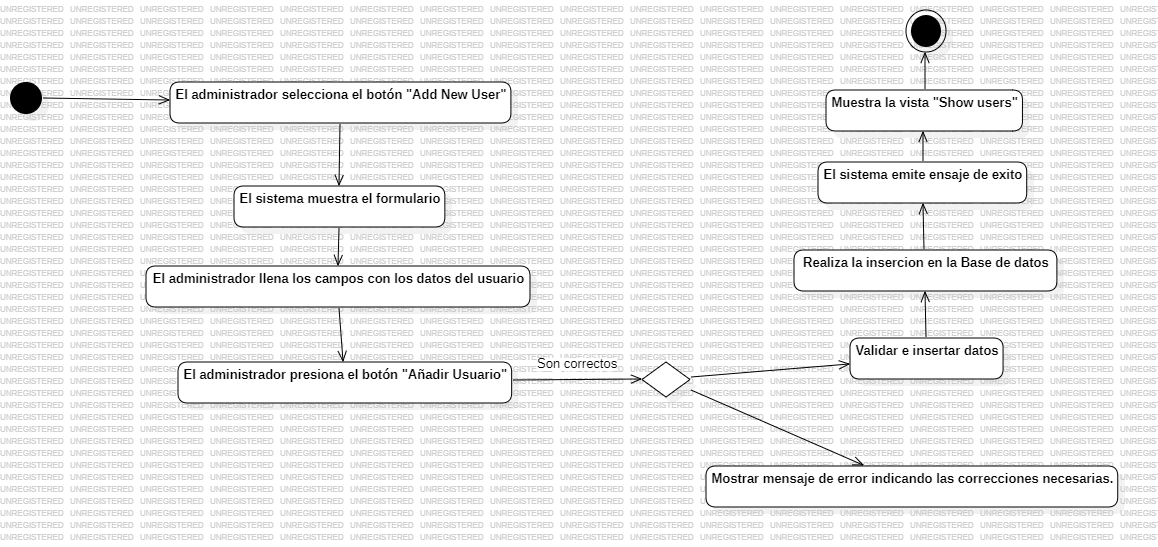


Figura 21. Diagrama de actividades de "crear nuevo usuario"

### Desarrollo de Mockups y Crear Usuario.



Figura 22. Mockup de crear nuevo usuario

Cuando un usuario ingresa sus credenciales como administrador, será redirigido a la página admin/home, donde se mostrará una tabla con los usuarios activos e inactivos. El administrador tendrá la capacidad de buscar, editar y agregar usuarios. Véase la **Figura**.



Figura 23. Vista del formulario de "crear nuevo usuario"

Al seleccionar el botón "Add User" (véase la Figura, sección marcada en rojo), se desplegará una nueva ventana en la página admin/create, donde el administrador deberá completar los campos correspondientes para agregar un nuevo usuario. Al hacer clic en el botón "Add", los datos ingresados serán guardados en la base de datos, y el nuevo usuario aparecerá en la tabla mostrada en la Figura 3.Desarrollo mockups del usuario.

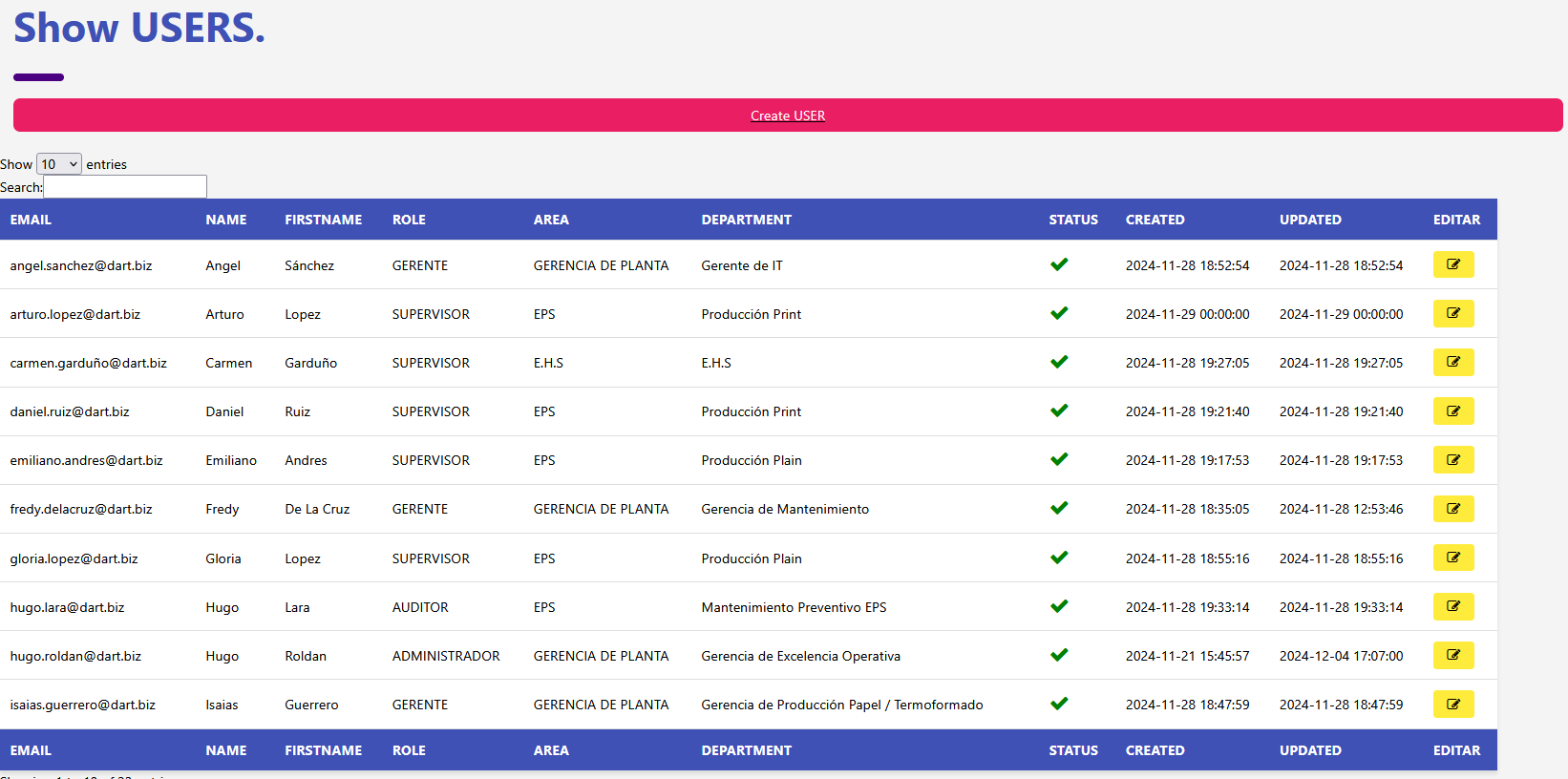


Figura 24. Vista de los usuarios creados y enlistados.

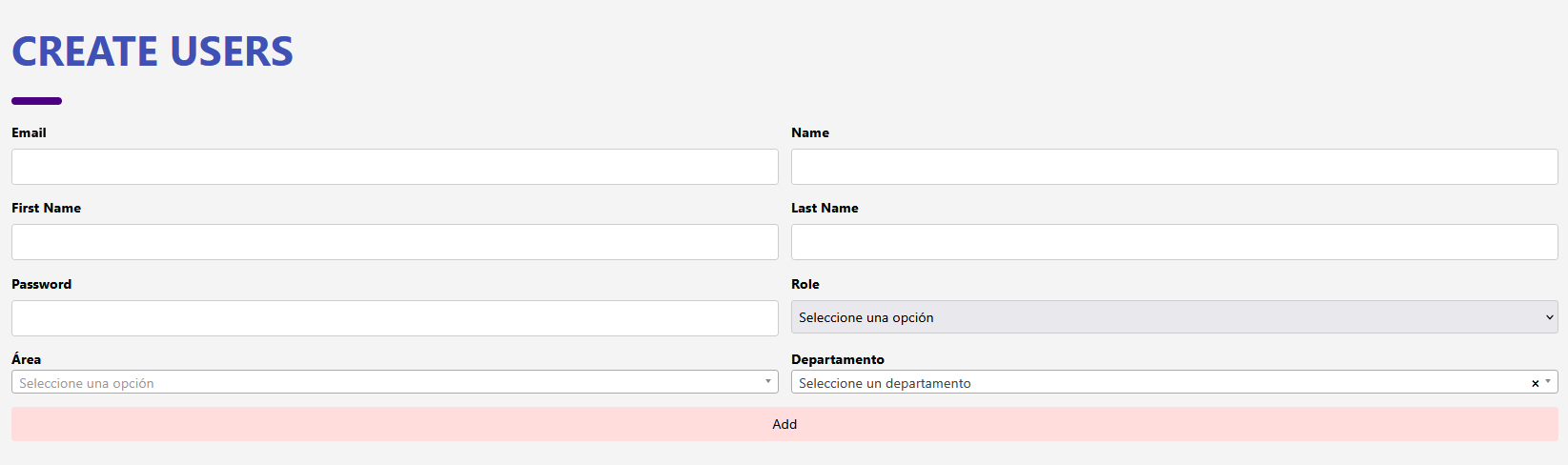


Figura 25. Vista de crear nuevo usuario.

## Diagramas UML “Editar Usuario”



Figura 26. Mockup de “editar usuario”.

Lo que se busca con este mockup es el lograr editar datos básicos de un usuario como seria el nombre, apellidos, el turno, etc. Solo datos básicos puesto que el administrador no podrá editar campos como la contraseña.

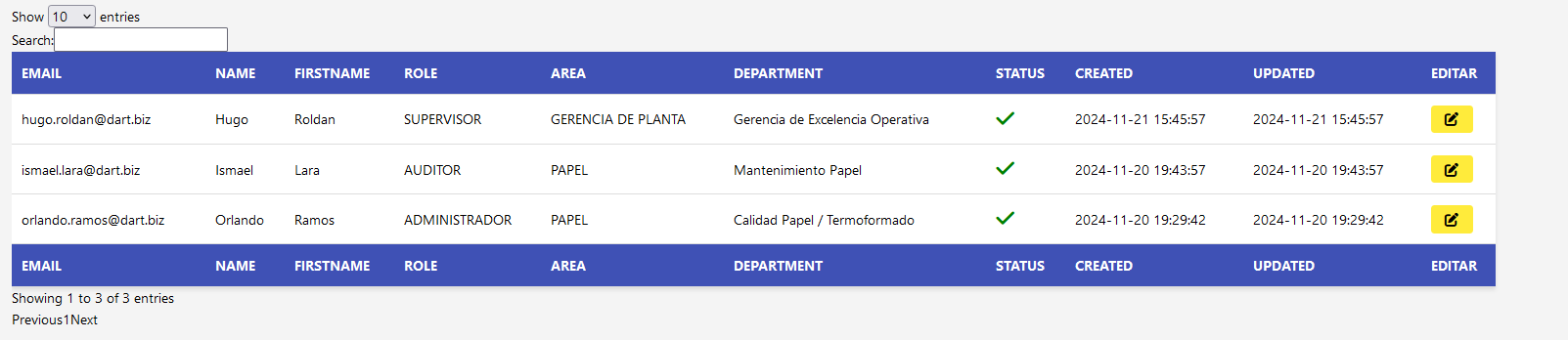


Figura 27. Listado de usuarios para "editar" un usuario

En la figura se muestran algunos datos de usuario como el nombre apellidos y a que área pertenecen puesto que gracias a esto se puede identificar a que departamento pertenecen además de que gracias a las “DataTables” podemos paginar los usuarios y buscar datos sobre ellos.

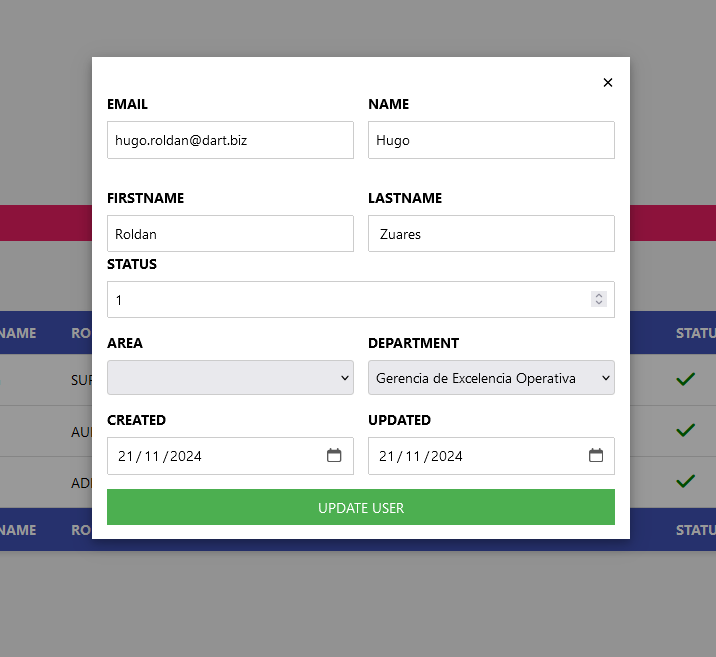


Figura 28. Modal de editar un usuario

En la figura se muestra un botón “editar” para que seleccionemos el usuario que deseamos editar esto con el fin de poder cambiar datos sobre él y guardarlo.

## Diagramas UML “Crear Auditoria”

### Desarrollo de Mockups y Crear Auditoria.



Figura 29. Mockup de "crear auditoria".

Al seleccionar la opción de “crear auditoria” del menú del sistema. Inicialmente se selecciona la maquinaria utilizada, el turno en el cual se realizará, el departamento involucrado y el nombre de auditor presente (todo eso traído desde la base de datos). Se necesitan plantear preguntas al departamento dirigido. Una vez completados todos los campos obligatoriamente. Se escribirán las preguntas las cuales están organizadas por categorías. Entonces cada que se escriba una pregunta el sistema nos mostrara la fuente y podremos seleccionar una y guardar en la base de datos, véase en la figura

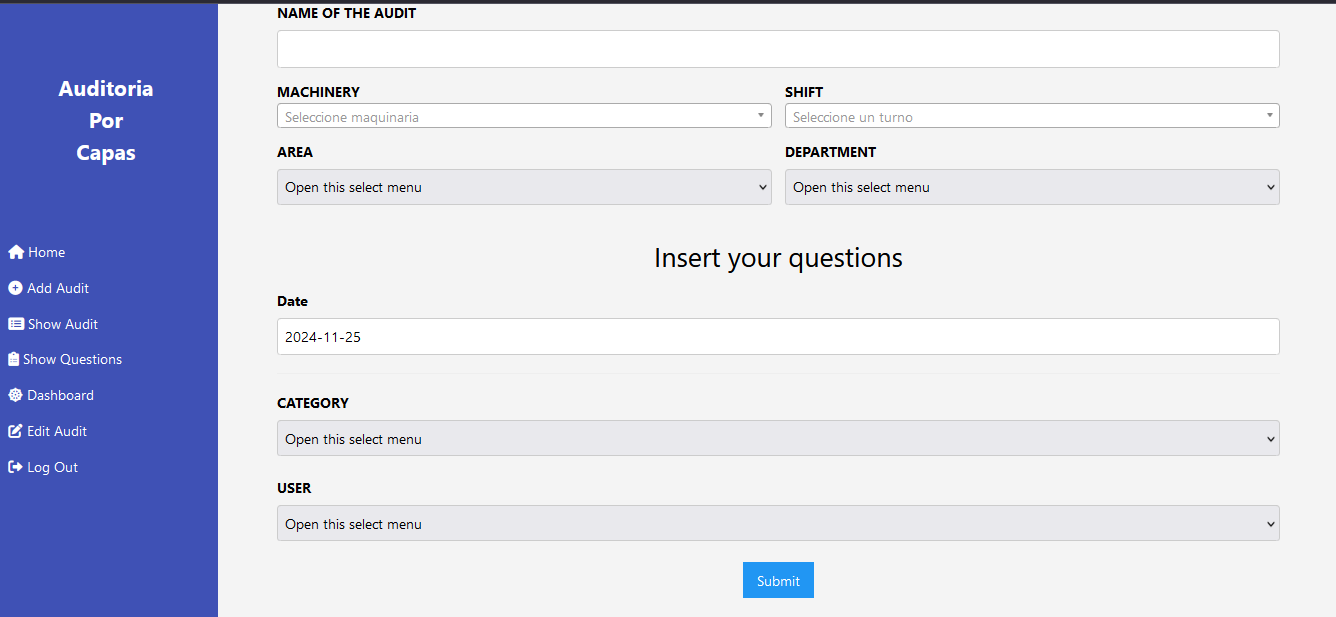


Figura 30. Vista de "crear auditoria"

Como se mencionó en el mockup de la figura anterior el desarrollo de crear una auditoria tenía como proceso escoger los datos principales como el nombre de la auditoria, la maquinaria con la que se trabajara, el turno, entre otros datos para consecuentemente escoger las preguntas que se utilizaran durante la auditoria en donde cada categoría tendrá una o más preguntas, y una pregunta tendrá una fuente.

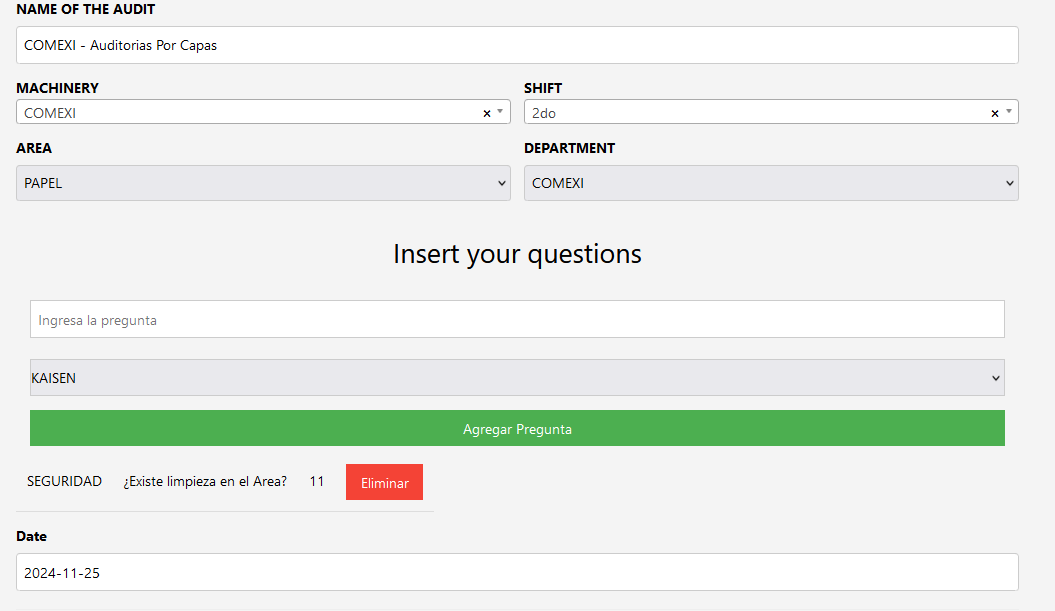


Figura 31. Vista de una auditoria con integración de datos

Como se muestra en la figura se colocan y/o seleccionan los datos solicitados, se escoge una categoría y en base a esa categoría poder ir escribiendo las preguntas y escoger la fuente a la que pertenecen una vez se hallan integrado todas las preguntas se presiona el botón de “agregar pregunta” de esta manera se garantiza que al escribir una pregunta se va almacenando y mostrando en la parte inferior, una vez integrando todas las preguntas se selecciona al usuario al que corresponde la auditoria creada y se le envía también se almacena la auditoria en la BD a fin de después mostrarla en una tabla para verificar lo que se realiza dentro de la auditoria y que el usuario a quien está destinada pueda responderla.

## Desarrollo de Mockups y Ver Auditoria.

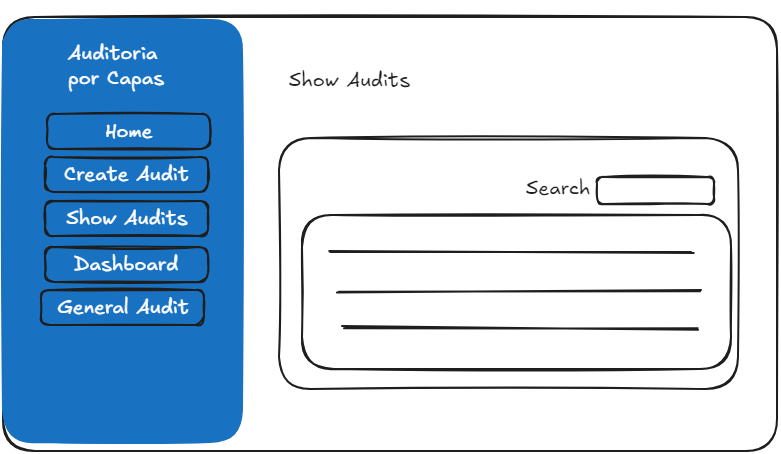


Figura 32. Mockup de "ver auditoria".

Cuando terminamos de crear una auditoria esta procede a mostrarse en una tabla donde se puede observar algunos atributos de la auditoria como pueden ser el turno, el departamento y el estado en el que se encuentra, teniendo esto en cuenta la auditoria se puede seleccionar con el fin de poder realizar cambios en caso de los que realizan la auditoria y revisar los cambios en caso del Administrador.

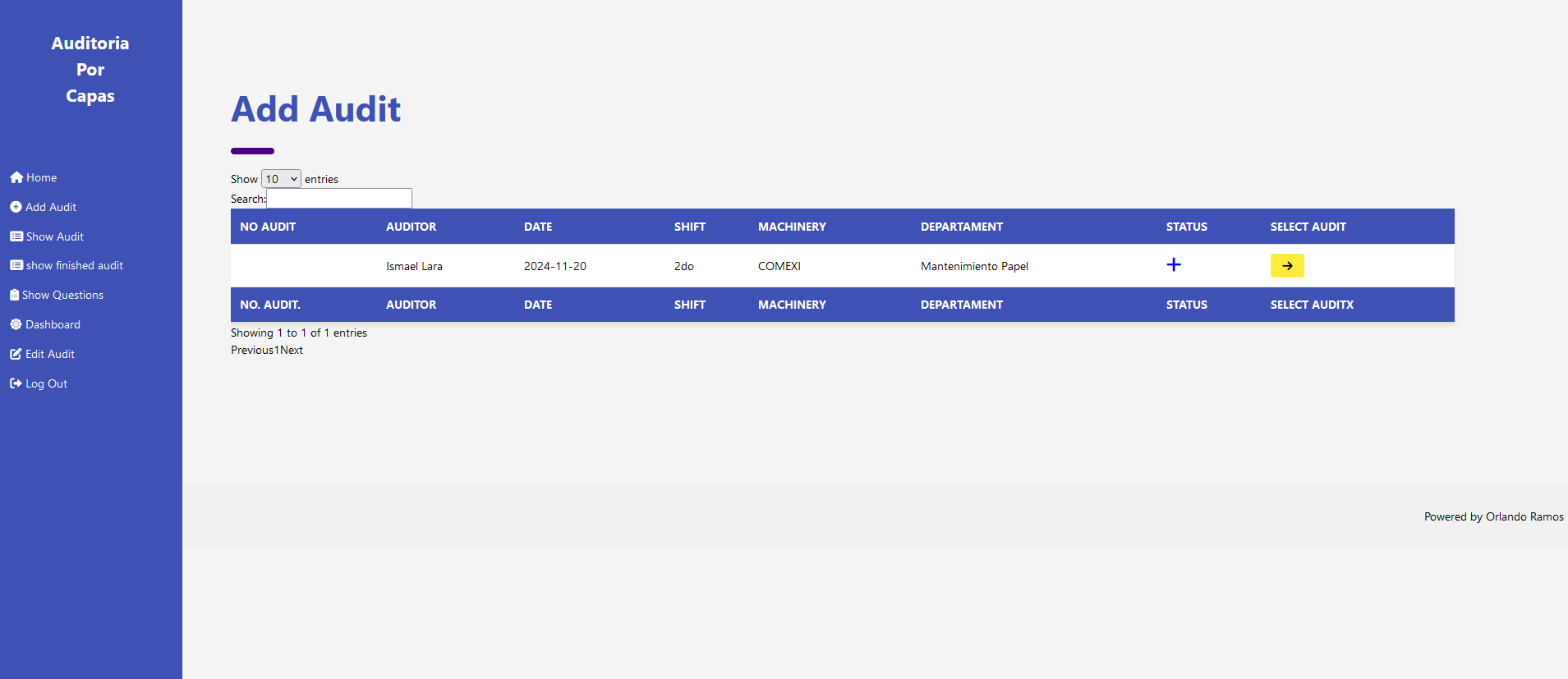


Figura 33. Auditoria creada y activa.

En la imagen y como se mencionó en los mockups se muestran algunos datos como el Auditor a quien va dirigido, el turno, etc. La auditoría se mantiene activa y se puede ver que cambios se están realizando dentro de esta a fin de lograr entender que respuestas se están dando.

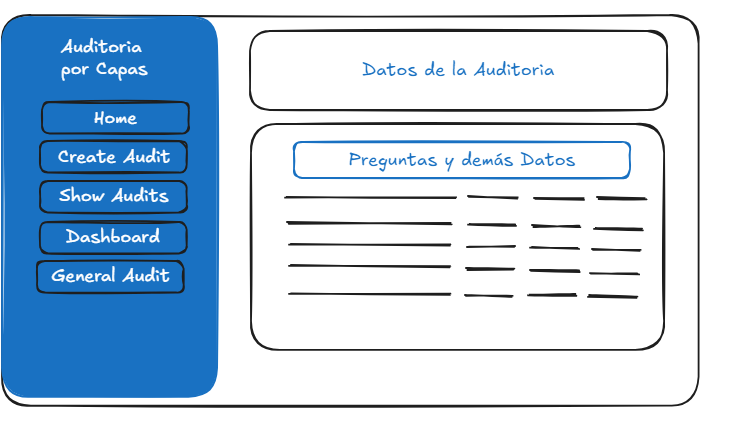


Figura 34. Mockup de "Detalles de la auditoria"

En el mockup de la figura se planea traer los datos del encabezado de la auditoria que se está realizando como podrían ser el título, el usuario que realizara la auditoria. Entre otros componentes y en la parte de abajo estarían las preguntas como se muestra en la figura

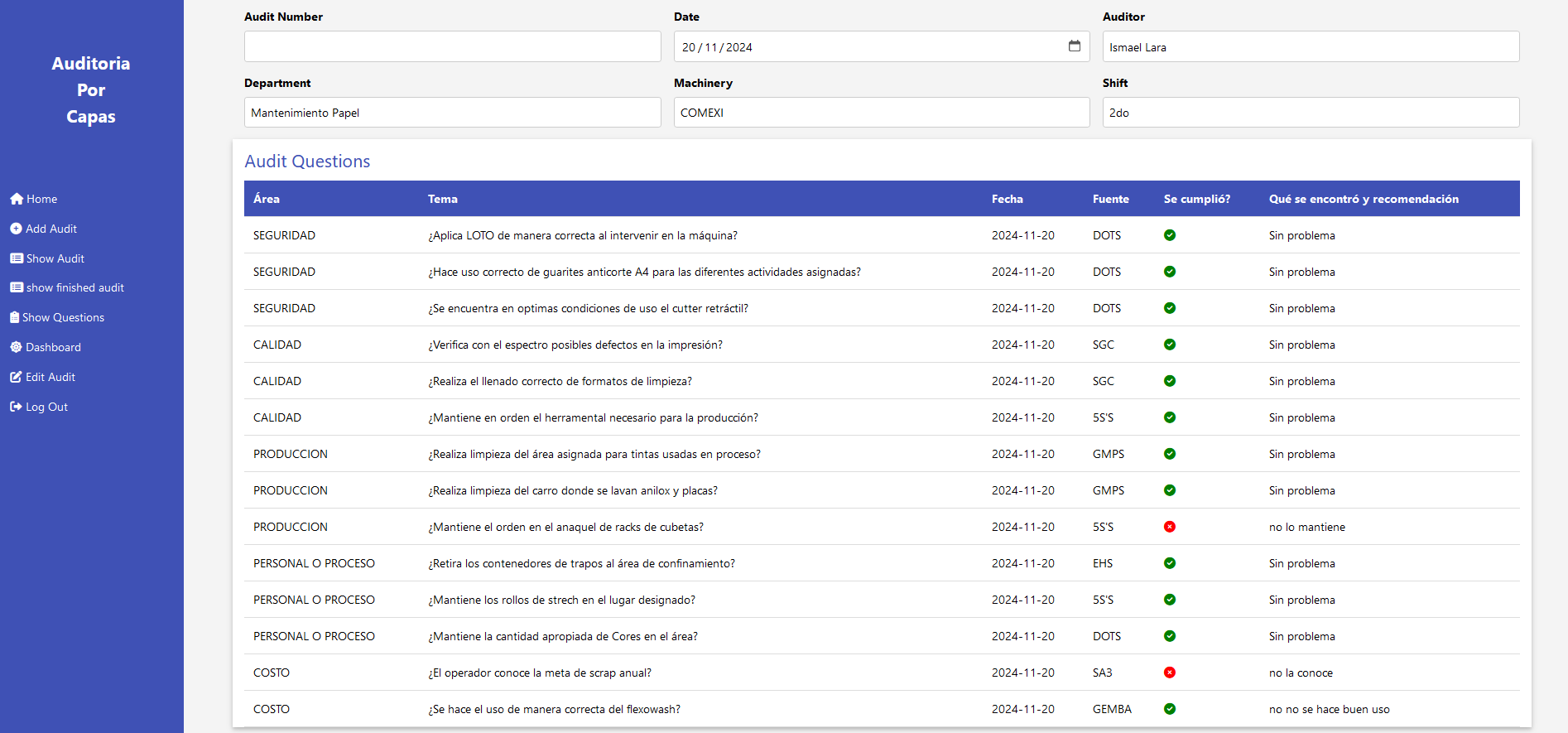


Figura 35. Vista de "Detalles de la auditoria"

Al seleccionar una o cualquier auditoria activa esta mostrara los datos que están presentes en el encabezado (Auditor, Departamento, Maquinaria y demás datos). Las preguntas que están activas y relacionada con esa auditoria, la fecha en la que se crearon las preguntas. Por parte del usuario que realice la auditoria y conforme se valla realizando la auditoria si se están cumpliendo y que problemas se encontraron.

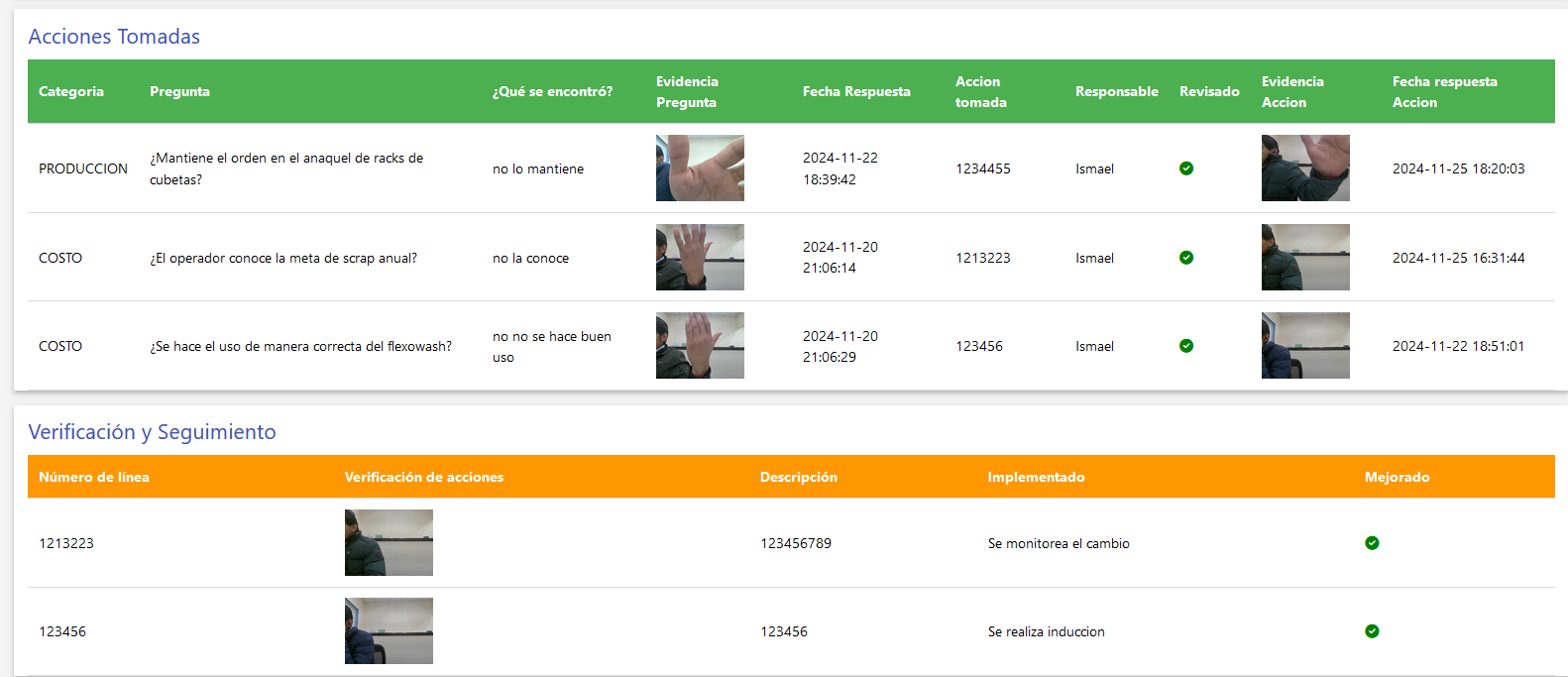


Figura 36. Auditoria respondida con acciones tomadas y seguimiento.

Durante el desarrollo de la auditoria y encontrar algún problema relacionado con la pregunta se encontrará algún problema durante el desarrollo de la auditoria se tomaría la evidencia de lo que se encontró, que acción se tomó conforme a el problema y que el supervisor o su jefe directo verifique el problema, esto para tener un control, recordando que todo esto lo puede ver solamente el administrador con el fin de poder revisar los cambios dentro de las auditorias.

## Desarrollo de mockups y vista HOME del usuario (Auditor).

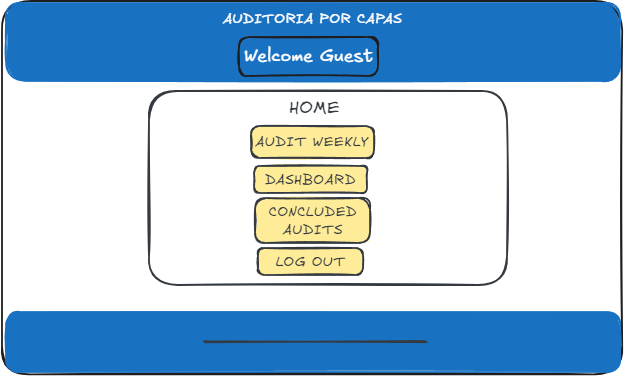


Figura 37 Mockup del auditor "Home"

Una vez que el usuario halla ingresado al sistema (dependiendo de su rol) podrá ver un menú como el que se muestra en la figura donde si es un usuario que desarrolla una auditoria tendrá la auditoria semanal, el Dashboard donde dé podrán ver algunas graficas con dato sobre lo que realiza, las auditorias que ha concluido y un logout donde podrá cerrar su sesión.

Al ingresar con sus datos en la parte de arriba se mostrar el nombre de quien ingresa y tendrá una sesión activa una vez que cierre su sesión no podrá ver más datos o sus pendientes.

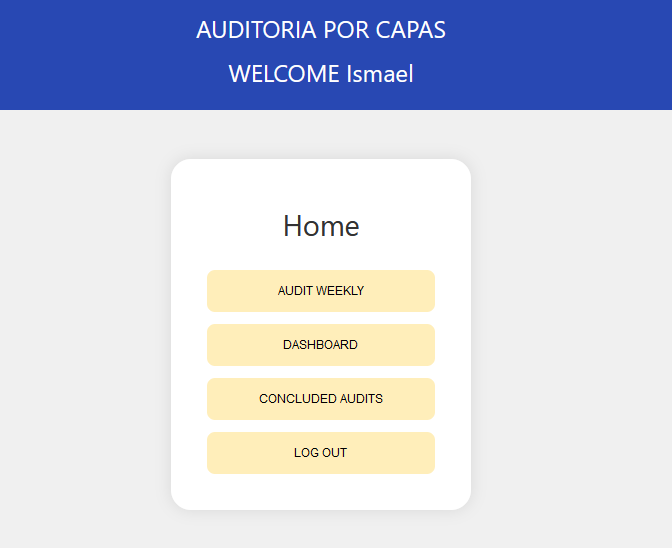


Figura 38. Vista del auditor "Home"

Como se mencionó en la imagen y mockup anterior se buscaba tener un menú en el centro donde se pudieran realizar las acciones que el usuario podrá realizar durante su estancia dentro del sistema el primer punto es revisar la auditoria pendiente el cual va a base de

revisar si tiene una auditoria pendiente.

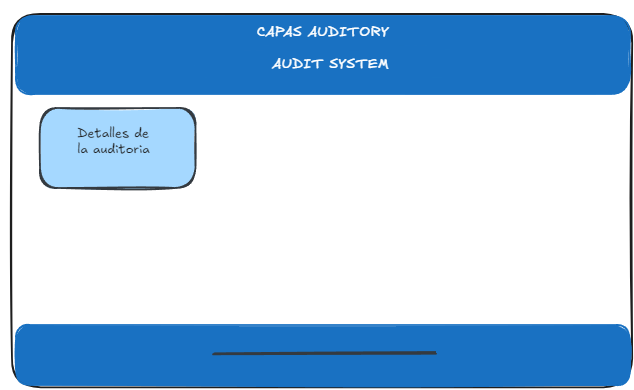


Figura 39. Mockup donde se muestran las auditorias

Una vez seleccionada la primera opción del menú mencionado en la figura nos lanzara a una vista donde se generarán tarjetas con algunos datos de la auditoria esto a fin de conocer que la auditoria va dirigida hacia nosotros para consecuentemente escogerla y nos de la opción de ir respondiendo la auditoria a como se presente.

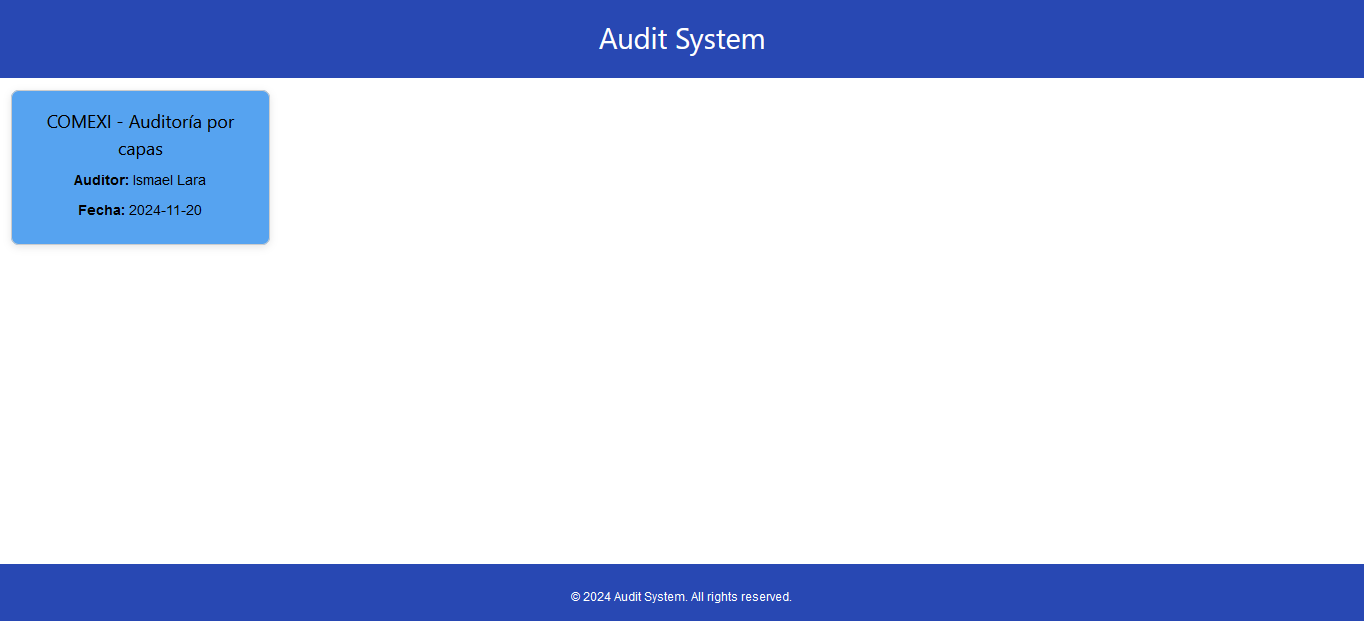


Figura 40. Vista de auditorías activas

En la vista tenemos las auditorias activas y generadas en las tarjetas las cuales se muestran en la vista del usuario auditor con el nombre del auditor que realizara la auditoria, el titulo de la auditoria y la fecha que se generó, la cual el usuario podrá seleccionar para obtener los datos de la auditoria y poder responderl

Capítulo 5

Productos del proyecto

* 1. Contenido

Manuales de usuario, códigos relevantes, infografías

* 1. Más información

El vídeo proporciona una manera eficaz para ayudarle a demostrar el punto. Cuando haga clic en Vídeo en línea, puede pegar el código para insertar del vídeo que desea agregar. También puede escribir una palabra clave para buscar en línea el vídeo que mejor se adapte a su documento. Para otorgar a su documento un aspecto profesional, Word proporciona encabezados, pies de página, páginas de portada y diseños de cuadro de texto que se complementan entre sí. Por ejemplo, puede agregar una portada coincidente, el encabezado y la barra lateral.

Haga clic en Insertar y elija los elementos que desee de las distintas galerías. Los temas y estilos también ayudan a mantener su documento coordinado. Cuando haga clic en Diseño y seleccione un tema nuevo, cambiarán las imágenes, gráficos y gráficos SmartArt para que coincidan con el nuevo tema. Al aplicar los estilos, los títulos cambian para coincidir con el nuevo tema. Ahorre tiempo en Word con nuevos botones que se muestran donde se necesiten.

* 1. Más información

El vídeo proporciona una manera eficaz para ayudarle a demostrar el punto. Cuando haga clic en Vídeo en línea, puede pegar el código para insertar del vídeo que desea agregar. También puede escribir una palabra clave para buscar en línea el vídeo que mejor se adapte a su documento. Para otorgar a su documento un aspecto profesional, Word proporciona encabezados, pies de página, páginas de portada y diseños de cuadro de texto que se complementan entre sí. Por ejemplo, puede agregar una portada coincidente, el encabezado y la barra lateral.

Haga clic en Insertar y elija los elementos que desee de las distintas galerías. Los temas y estilos también ayudan a mantener su documento coordinado. Cuando haga clic en Diseño y seleccione un tema nuevo, cambiarán las imágenes, gráficos y gráficos SmartArt para que coincidan con el nuevo tema. Al aplicar los estilos, los títulos cambian para coincidir con el nuevo tema. Ahorre tiempo en Word con nuevos botones que se muestran donde se necesiten.

Capítulo 6

Conclusiones y recomendaciones

* 1. Contenido

Resaltar lo esencial del proyecto realizado; comentar los logros, dimensionar los resultados en tiempo y espacio, mostrando medios para ir más lejos en desarrollos similares

* 1. Más información

El vídeo proporciona una manera eficaz para ayudarle a demostrar el punto. Cuando haga clic en Vídeo en línea, puede pegar el código para insertar del vídeo que desea agregar. También puede escribir una palabra clave para buscar en línea el vídeo que mejor se adapte a su documento. Para otorgar a su documento un aspecto profesional, Word proporciona encabezados, pies de página, páginas de portada y diseños de cuadro de texto que se complementan entre sí. Por ejemplo, puede agregar una portada coincidente, el encabezado y la barra lateral

* 1. Más información

El vídeo proporciona una manera eficaz para ayudarle a demostrar el punto. Cuando haga clic en Vídeo en línea, puede pegar el código para insertar del vídeo que desea agregar. También puede escribir una palabra clave para buscar en línea el vídeo que mejor se adapte a su documento. Para otorgar a su documento un aspecto profesional, Word proporciona encabezados, pies de página, páginas de portada y diseños de cuadro de texto que se complementan entre sí. Por ejemplo, puede agregar una portada coincidente, el encabezado y la barra lateral.

Referencias

|  |  |
| --- | --- |
| [1] | E. Place, PROGRAMACIÓN ORIENTADA A OBJETOS, www.surforce.com, 2009. |