INDICE

1. Introducción

a. Presentación del proyecto

b. Objetivos del proyecto

c. Justificación del proyecto

2. Análisis de Requerimientos

a. Identificación de necesidades y requerimientos

b. Identificación de público objetivo

c. Estudio de mercado y competencia

3. Diseño y Planificación

a. Definición de la arquitectura del proyecto

b. Diseño de la interfaz de usuario

c. Planificación de las tareas y los recursos necesarios

4. Implementación y Pruebas

a. Desarrollo de las funcionalidades del proyecto

b. Pruebas unitarias y de integración

c. Corrección de errores y optimización del rendimiento

5. Documentación

a. Documentación técnica

b. Documentación de usuario

c. Manual de instalación y configuración

6. Mantenimiento y Evolución

a. Plan de mantenimiento y soporte

b. Identificación de posibles mejoras y evolución del proyecto

c. Actualizaciones y mejoras futuras

7. Conclusiones

a. Evaluación del proyecto

b. Cumplimiento de objetivos y requisitos

c. Lecciones aprendidas y recomendaciones para futuros proyectos

8. Bibliografía y Referencias

a. Fuentes utilizadas en el proyecto

b. Referencias y enlaces de interés

I. Introducción

**a. Presentación del proyecto**

El proyecto consiste en desarrollar una aplicación web para la gestión de averías de máquinas de clientes. Está orientada a la productividad de los equipos de trabajo y la trazabilidad de los servicios solicitados por los clientes. Los usuarios podrán acceder a la aplicación disponiendo de un navegador web y una conexión a internet sin necesidad de otro software adicional.

Se procura presentar una interfaz gráfica de usuario intuitiva y una vez que hemos trabajado un poco con ella que resulte amigable y con comandos que permitan que la experiencia de usuario sea positiva.

Está diseñada para visualizar en versión de escritorio para utilizar toda su funcionalidad, aunque también está disponible una versión para pantallas más reducidas con las funciones principales.

La aplicación incluye una base de datos con tablas para clientes, máquinas, avisos de averías, técnicos, usuarios y registros de acceso.

**b. Objetivos del proyecto**

* Facilitar y acelerar el proceso de reparación de averías reduciendo el tiempo de parada de las máquinas para que no interfiera en la operativa de los clientes.
* Dotar de una herramienta que permita gestionar de forma eficiente las tareas pendientes, así como proveer de información al respecto para una mejor toma de decisiones.
* Centralizar la información relacionada con clientes, máquinas e incidencias en los equipos para conseguir un acceso rápido y eficiente a la misma.
* Dejar la opción de hacer el proyecto escalable para futuras necesidades.

II. Análisis de Requerimientos

**a. Identificación de necesidades y requerimientos**

Para poder llevar a cabo el proyecto es necesario analizar los objetivos y tener en cuenta la posición de todas las partes implicadas, es decir las características de los clientes, el parque de máquinas, las incidencias que se producen y los técnicos que las asisten.

La aplicación tiene que poder dar de alta y gestionar clientes, máquinas, incidencias y usuarios, así como hacer un seguimiento de las averías y poder asignar las mismas a los técnicos ofreciendo la posibilidad de clasificar de incidencias por fecha y prioridad para mejorar la planificación de las mismas.

Como requisito mínimo de seguridad implementará un cifrado de contraseñas y un sistema de autentificación de usuario basado en roles para proteger la información.

Por último, ha de tenerse en cuenta la escalabilidad del proyecto para futuras necesidades.

Es por esto que debe constar de:

* Una base de datos para almacenar información de clientes, máquinas, avisos de averías, técnicos y usuarios.
* Una interfaz de usuario que de un acceso rápido a la información y permita su manejo.

**b. Identificación de público objetivo**

El público objetivo de la aplicación son los técnicos y administradores que gestionan la reparación de averías y planifican los trabajos para agilizar los tiempos de respuesta y analizar los datos obtenidos. Para ello consta de dos perfiles de acceso, uno de ellos es nivel usuario que le permite gestionar clientes, maquinas e incidencias; otro perfil administrador que le permite además gestionar usuarios. En un futuro podría existir un perfil cliente que pueda gestionar sus datos, sus máquinas y generar las incidencias propias.

**c. Estudio de mercado y competencia**

Para satisfacer la necesidad de información a tiempo real sobre la operatividad del parque de máquinas para planificar y garantizar la operativa interna de los clientes, surge la necesidad de dotar de una herramienta que ayude en la gestión de las incidencias de los clientes y sus máquinas, así como la organización del trabajo de los técnicos encargados de solucionar dichas incidencias.

Adaptando el modelo podría implementarse en otros casos de uso relacionados con el sector de los servicios de asistencia técnica como por ejemplo mantenimiento de instalaciones, equipos domésticos o mantenimiento de equipos informáticos.

III. Diseño y Planificación

**a. Definición de la arquitectura del proyecto**

La arquitectura propuesta sigue el patrón Modelo – Vista – Controlador (MVC) , por lo tanto se puede dividir en tres capas principales.

1. MODELO.

**Capa de datos**: Incluye una base de datos relacional responsable del almacenamiento y gestión de la información. El sistema gestor de BBDD que incluye las tablas para clientes, máquinas, avisos de averías, técnicos y usuarios en este caso será MariaDB.

**Patrón Singleton**: Se implementa en el Manager para asegurar que solo haya una instancia de esta clase en toda la aplicación. Esto se logra mediante una función autoejecutable que contiene una variable privada para almacenar la instancia única y un método estático para acceder a ella.

1. VISTA

**Capa de presentación (FrontEnd**): La capa de presentación es la interfaz de usuario de la aplicación web. Se encarga de la interacción con el usuario final y de mostrar los datos de manera amigable y atractiva. Está desarrollada en HTML que estructura la página web, CSS3 que nos permite dar estilo y diseño visual y JavaScript nativo para interactuar y dar funcionalidad a la página de manera dinámica.

1. CONTROLADOR.

**Capa de Lógica de Negocio (Backend)**: Los controladores que gestionan las peticiones del usuario, procesan la lógica de negocio, interactúan con el modelo y devuelven las respuestas adecuadas a la vista. Para gestionar el intercambio de información con la base de datos se utilizan DAO (Data Access Objects ) que abstraen y encapsulan las operaciones de acceso a los datos para facilitar el mantenimiento del código organizado en objetos permitiendo su escalabilidad. El lenguaje de programación es PHP nativo.

|  |
| --- |
| Cliente (Navegador)  ▼  Vista (HTML5, CSS3, JS)  ▼  Controlador (PHP)  Manejo de peticiones  Llamada a los métodos de servicio  ▼  Servicios (Lógica de Negocio)  Contiene la lógica de negocio  Llamadas a los métodos DAO  ▼  DAO (Data Access Object)  Gestiona las operaciones de acceso a los datos  ▼  Modelo (MariaDB)  Define la estructura de la base de datos  ▼  Base de Datos  tabla clientes  tabla máquinas  tabla avisos  tabla productores  tabla usuarios |

La aplicación consta de los archivos la2024App.js, la2024Model.js, la2024Controller.js y la2024View.js que es donde se estructura todo el patrón MVC.

* **la2024app.js**

Este archivo actúa como el punto de entrada de la aplicación. Aquí es donde se inicializan los componentes principales y se configura el controlador.

* Importaciones: Importa Manager (modelo), La2024Controller (controlador) y La2024View (vista).
* Instanciación: Crea una instancia del controlador La2024Controller pasando como parámetros una instancia única de Manager y una nueva instancia de La2024View.
* Exportación: Exporta la instancia del controlador como el módulo predeterminado.

|  |
| --- |
| import Manager from "./la2024Model.js";  import La2024Controller from "./la2024Controller.js";  import La2024View from "./la2024View.js";  const La2024App = new La2024Controller(Manager.getInstancia(), new La2024View());  export default La2024App; |

* **la2024model.js**

Es la parte del modelo, este archivo contiene la lógica del modelo. Gestiona los datos de la aplicación, define las propiedades privadas y utiliza el patrón Singleton para asegurar que solo haya una instancia de Manager.

|  |
| --- |
| const Manager = (function () {  let manager;  class Manager {  #clientes = new Map();  #maquinas = new Map();  #avisos = new Map();  #avisosPendientes = new Map();  #productores = new Map();  #usuarios = new Map();  #info = {};  constructor(name) {  if (!new.target) throw new LindeException();  Object.defineProperty(this, "name", {  enumerable: true,  get() {  return this.#name;  },  set(value) {  this.#name = value;  }  });  }  / aquí los métodos /  }//fin de clase  function init() {  return new Manager();  }  return {  getInstancia: function () {  if (!manager) {  manager = init();  }  return manager;  }  }  })();  export default Manager; |

* **la2024controller.js**

Este archivo contiene la lógica del controlador. El constructor recibe como parámetros la vista y el modelo gestionando las interacciones de ambos. Además, contiene los métodos para inicializar la aplicación y gestionar las peticiones del usuario. Se definen dos constantes MODEL y VIEW utilizando la función SYMBOL que se usan para encapsular.

|  |
| --- |
| 'use strict';  const MODEL = Symbol('La2024Model');  const VIEW = Symbol('La2024View');  class La2024Controller {  constructor(modelLa2024, viewLa2024) {  this[MODEL] = modelLa2024;  this[VIEW] = viewLa2024; |

* **la2024view.js**

Este archivo contiene la lógica de la vista. Se encarga de la presentación de la interfaz y la captura de interacciones del usuario. Contiene un constructor que inicializa sus componentes y los métodos para mostrar las vistas y las interacciones del usuario.

|  |
| --- |
| 'use strict';  class La2024View {  constructor() {  this.header = document.getElementsByTagName('header')[0];  this.main = document.getElementsByTagName('main')[0];  this.footer = document.getElementsByTagName('footer')[0];  this.pageSize = 50;  this.currentOffSet = 0;  }  / APARTADO BARRA NAVEGACION /  showNavBar(usuario, handler) { |

**b. Diseño de la interfaz de usuario.**

La intención es seguir los principios del diseño y crear una interfaz simple, usable, eficiente, ágil e intuitiva. Ha de ser clara, sencilla y fácil de navegar evitando la sobrecarga de información, minimizando los pasos necesarios para completar una tarea sin necesidad de instrucciones detalladas. Los elementos deben comportarse de manera predecible.

**Componentes de la Interfaz**

1. **Pantalla de Login**
   * **Formulario de Login:** Campos para usuario y contraseña.
   * **Validación de Contraseña:** Verificación de formato y alerta de errores.
   * **Registro de Nuevo Usuario:** Confirmación de contraseña con input de comprobación.
2. **Barra de Navegación Superior**
   * **Logotipo de la Empresa:** Situado en la esquina izquierda.
   * **Opciones del Menú:**
     + **Home:** Enlace a la página de inicio.
     + **Clientes:** Submenú con opciones de nuevo cliente y modificar cliente.
     + **Máquinas:** Submenú similar al de clientes.
     + **Avisos:** Submenú para crear nuevos avisos y gestionar avisos pendientes.
     + **Admin (sólo para administradores):** Gestión de usuarios.
   * **Información del Usuario Registrado:**
     + Nombre y perfil con icono interactivo.
     + Menú desplegable para cerrar sesión, cambiar contraseña o eliminar perfil.
3. **Vista Principal o Home**
   * **Estadísticas y Datos:**
     + Visualización de datos globales sobre clientes, máquinas e incidencias.
     + Utilización de gráficos y tablas dinámicas para presentar la información.
4. **Sección de Clientes**
   * **Formulario de Nuevo Cliente:**
     + Campos básicos validados antes del envío.
   * **Búsqueda de Clientes:**
     + Filtros en tiempo real (nombre, población, provincia, contacto).
     + Resultados en una tabla dinámica para edición o eliminación de datos del cliente.
5. **Sección de Máquinas**
   * Similar a la sección de clientes, con funcionalidad específica para máquinas.
6. **Sección de Avisos**
   * **Nuevo Aviso:**
     + Select de clientes y máquinas.
     + Rellenado y validación de campos del objeto aviso.
     + Muestra dinámica de tablas con la información en tiempo real de clientes y máquinas.
   * **Gestión de Avisos Pendientes:**
     + Modificación, asignación a técnicos, comentarios y gestión de propiedades.
     + Alertas visuales en función de la prioridad y la fecha.
7. **Sección de Admin (sólo con rol admin)**
   * **Nuevo Usuario**
     + Campos básicos validados antes del envío.

* **Administrar usuarios.**
  + Modificar nombre, rol y contraseña o eliminar registro.

La interfaz de usuario es básica y sencilla tratando ser eficiente, amigable e intuitiva. Al acceder encontramos con un menú de acceso en forma de modal que permite introducir un nombre de usuario y contraseña o crear un nuevo usuario con el rol de “user”. Verifica que la contraseña esté formada por letras y/o números y alerta de los posibles errores.

Una vez accedido a la aplicación, la parte superior está dividida en 3 partes, la primera muestra el logotipo de la empresa. La segunda parte está justo al lado derecho y consiste en una barra de navegación que da acceso a los apartados home, clientes, máquinas y avisos, indicando en que parte de la aplicación se encuentra el usuario en cada caso. Al pulsar en cada opción se desplegará un submenú con las distintas opciones disponibles en cada caso como por ejemplo en la parte de cliente las opciones de nuevo cliente y modificar cliente. Sucederá lo mismo en la parte de máquinas. En la parte de avisos se podrá crear un nuevo aviso y en otra opción llamada avisos pendientes se muestran los avisos en formato tabla permitiendo su manipulación. Si entras con un perfil de administrador encontrarás una quinta opción de “admin” que permite crear usuarios y gestionar los datos de los mismos. La opción “home” nos devuelve a la página principal.

La tercera parte de la barra de navegación nos muestra la información relativa al usuario que está registrado en ese momento, nombre y perfil con un icono que al pulsarlo nos ofrece un pequeño menú desplegable que nos permite abandonar la aplicación, cambiar la contraseña o eliminar el perfil.

La parte central de la vista principal o home muestra diversas estadísticas y datos que pretenden dar una visión global al usuario con datos de clientes, maquinas e incidencias pendientes de resolver.

La sección de cliente nos muestra un primer apartado con un formulario donde dar de alta un nuevo cliente con los datos básicos validando los mismos antes de enviarlos. En un segundo apartado tenemos un apartado de búsqueda donde podemos filtrar en tiempo real por nombre, población, provincia o contacto en un solo cuadro de entrada de datos para encontrar a un cliente específico y poder editar sus datos o eliminarlo de la aplicación. Los resultados los muestra en una especie de tabla dinámica (sin ser una tabla) donde pulsando en un botón de la fila despliega un cuadro con los datos del cliente permitiendo la edición de los datos o la eliminación en este caso del cliente.

La sección de máquinas funciona idénticamente a la de clientes.

El apartado de avisos tiene la opción de Nuevo aviso asignando a un campo SELECT los clientes y otro SELECT que dinámicamente carga las maquinas del cliente. Además, se pueden rellenar los demás campos con las propiedades del objeto aviso validando los mismos. En el apartado avisos pendientes puede hacer toda la gestión de avisos, es decir, modificarlos, asignarlos a los técnicos, darlos como finalizados, añadir comentarios y rellenar las distintas propiedades del objeto aviso. Dinámicamente mostrará alertas visuales cambiando el color de fondo dependiendo de prioridad y fecha de entrada.

La opción “admin” accesible solo para el perfil de administrador contiene la posibilidad de un formulario para crear un nuevo usuario y un listado con los usuarios registrados para consultar, modificar o eliminar los datos de los mismos.

**Guía de Estilos de Colores**

|  |  |  |
| --- | --- | --- |
| **Colores Primarios** | **Colores Secundarios** | **Colores de Resaltado y Advertencia** |
| |  |  | | --- | --- | | **Blanco** | #FFFFFF | | **Negro** | #000000 | | **Gris Oscuro**: | Darkgrey | | **Gris Claro:** Lightgrey | h3 cabecera etsadisticas | | |  |  | | --- | --- | | **Rojo** | : #b21122 | | **Gris** | : #e5e6e9 | | **Gris Medio** | #ccc | | **Gris Claro** | #f9f9f9 | | |  |  | | --- | --- | | **Rojo Claro** | rgb(239, 222, 222) | | **Amarillo Claro**: lightyellow | lightyellow | | **Coral Claro**: | lightcoral | | **Salmón Claro**: | Lightsalmon | |
| **Colores de Fondo** | **Colores de Texto** | **Colores de Enlaces y Botones** |
| |  |  | | --- | --- | | **Gris Claro:** #eefeff3 | fondo barra de navegación y menú login | | **Gris Claro:** Lightgrey | h3 cabecera etsadisticas | | **Gris Medio:** RGB(231, 231, 231) | fondo modal | | |  |  | | --- | --- | | **Negro**:  #000000 | Etiquetas, enlaces | | **Blanco**: #FFFFFF | Texto Botones | | **Rojo:**  Red | Mensajes de error | | **Gris Oscuro**: Darkgrey | Formularios | | |  |  | | --- | --- | | **Azul Claro**: Lightskyblue | Enlaces | | **zul**: Lightblue | Enlaces | | **Rojo Oscuro**: RGB(69, 2, 2) | Enlaces Activos | |

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Color** | **Código de Color** |
| Fondo General | Gris Claro | #f1f1f1 |
| Menú de Navegación | Gris Oscuro | #333 |
| Cabeceras | Gris Claro | #f1f1f1 |
| Enlaces y Textos del Menú | Negro | #000000 |
| Enlaces Activos | Negro | #B21122 |
| Botones | Maroon | #800000 |
| Botones Activos | Rojo Oscuro | #e5e6e9 |
| Fondo de Formularios | Gris Claro | #e5e6e9 |
| Fondo Cabeceras de Formularios | Rosa Claro | #efdede |
| Texto Cabeceras Formularios y login | Maroon | #800000 |
| Borde de Inputs | Gris Medio | #ccc |
| Fondo de Tabla (hover) | Gris Claro | #f1f1f1 |
| Input’s Editables | Amarillo Claro | #ffffe0 |

|  |  |
| --- | --- |
| MENU DE LOGIN | NUEVO REGISTRO |
|  |  |
| VISTA PRINCIPAL (HOME) | MENU DE USUARIO |
|  |  |
| FORMULARIO ALTA DE CLIENTE | VISTA DE MODIFICAR CLIENTES |
|  |  |
| FORMULARIO NUEVA MAQUINA | VISTA DE MODIFICAR MAQUINAS |
|  |  |
| FORMULARIO NUEVO AVISO | VISTA DE AVISOS PENDIENTES |
|  |  |
| NUEVO USUARIO | VISTA DE MODIFICAR USUARIOS |
|  |  |

**c. Planificación de las tareas y los recursos necesarios**

A continuación, se detalla de forma esquemática la planificación de las tareas llevadas a cabo y descritas en este documento.

**Análisis de Requerimientos (Descrito en el apartado 2 A)**

* Recopilación de información
* Documentación de requerimientos

**Diseño de la Arquitectura (Descrito en el apartado 3 A)**

* Definición del modelo de datos
* Diseño del sistema basado en MVC, Singleton y DAO
* Creación de diagramas UML

**Diseño de la Interfaz de Usuario**

* Creación de wireframes y prototipos
* Diseño de la interfaz (HTML, CSS)
* Pruebas de usabilidad

**Desarrollo (Descrito en el apartado 3 A)**

* Implementación del backend (Model, Controller, DAO)
* Implementación del frontend (View)
* Integración del sistema

**Pruebas**

* Pruebas unitarias
* Pruebas de integración
* Pruebas de usuario
* Corrección de errores y optimización

**Despliegue**

* Preparación del entorno de producción
* Despliegue de la aplicación
* Verificación y pruebas finales

**Mantenimiento y Evolución**

* Plan de mantenimiento
* Actualizaciones y mejoras

IV. Implementación y Pruebas

**a. Desarrollo de las funcionalidades del proyecto**

* **Registro de usuarios**: Permite a los nuevos usuarios crear una cuenta en el sistema.
* **Login seguro**: Autenticación segura mediante validación de credenciales.
* **Gestión de clientes y máquinas**: Permite agregar, modificar y eliminar información de clientes y máquinas.
* **Registro y seguimiento de avisos de averías**: Registro de incidencias y seguimiento del estado de cada avería.
* **Asignación de técnicos a las averías**: Asigna técnicos específicos para atender cada incidencia.

**b. Pruebas unitarias y de integración**

Pruebas unitarias:

* **Prueba de Registro de Usuarios**: Se comprobó la correcta creación de usuarios con datos válidos y la gestión de errores para datos inválidos.
* **Prueba de Login Seguro**: Validación de credenciales correctas e incorrectas, y manejo de intentos fallidos.

Pruebas de integración:

* **Prueba de Gestión de Clientes y Máquinas**: Verificación de la correcta interacción entre el módulo de gestión de clientes y el de máquinas.
* **Prueba de Asignación de Técnicos**: Confirmación de la correcta asignación de técnicos a las averías y la actualización de estados en el sistema.

**c. Corrección de errores y optimización del rendimiento**

Se llevaron a cabo varios procesos para corregir errores y optimizar el rendimiento del sistema:

* **Detección y Corrección de Errores**: Se identificaron y corrigieron errores en la funcionalidad de login, en la gestión de clientes y en la asignación de técnicos.
* **Optimización de Consultas a la Base de Datos**: Mejora de las consultas SQL para reducir tiempos de respuesta y aumentar la eficiencia.
* **Mejora del Rendimiento de la Interfaz de Usuario**: Optimización del código HTML, CSS y JavaScript para asegurar una experiencia de usuario fluida.

V. Documentación

**a. Documentación técnica**

**Descripción de la Arquitectura**

Como ya hemos hablado, la arquitectura del proyecto está basada en el patrón de diseño MVC (Model-View-Controller) con la implementación de los patrones Singleton y DAO (Data Access Object) para mejorar la modularidad y el acceso a datos.

* **Modelo (Model)**: Se encarga de la lógica de la aplicación y la gestión de datos. Implementa el patrón DAO para separar la lógica de acceso a datos del resto de la aplicación.
* **Vista (View)**: Es la interfaz de usuario, encargada de mostrar la información al usuario y recoger sus interacciones. Utiliza tecnologías como HTML, CSS y JavaScript.
* **Controlador (Controller)**: Gestiona la comunicación entre el modelo y la vista, procesando las acciones del usuario y actualizando la vista y el modelo según sea necesario.

**Estructura de la Base de Datos y Relaciones**

Se incluye un diagrama ER (Entidad-Relación) que muestra la estructura de las tablas de la base de datos y sus relaciones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TABLA USUARIOS | CLIENTES | MAQUINAS | AVISOS | PRODUCTORES |
| #id;  #username;  #pass;  #rol;  #baja; | #id;  #nombre; #direccion; #poblacion; #provincia; #contacto; #telefono; #email; #fechaCreacion; #bajaCliente; | #id; #modelo; #chasis;  #cliente; #fechaCreacion; #alquiler;  #contrato;  #reaco;  #baja;  #notas;  #numero; | #id;  #idCliente;  #chasis;  #averia;  #productor; #finalizado; #fechaInicio; #fechaFin; #resolucion; #presupuesto; #aceptado; #prioridad; | #id;  #nombre;  #baja; |

|  |  |
| --- | --- |
| DIAGRAMA ENTIDAD RELACION | |
|  |

**b. Documentación de usuario**

Incluye instrucciones para el usuario final.

**c. Manual de instalación y configuración**

Al trabajar en local, para el despliegue de esta aplicación he utilizado XAMP, que además de ser gratuito al ser de código abierto, es fácil de instalar en cualquier sistema operativo y compatible con varios de ellos como Windows, Linux o MacOS. Tiene todos los componentes necesarios para configurar y ejecutar un servidor web como Apache, MySQL y PHP. Tiene una interfaz amigable e intuitiva que facilita la configuración y administración del servicio web.

VI. Mantenimiento y Evolución

**a. Plan de mantenimiento y soporte**

El mantenimiento será de 3 tipos:

* Correctivo. Se solucionarán los problemas reportados por los usuarios y se mantendrán las dependencias actualizadas como las librerías, imágenes, etc.
* Preventivo. Evaluar periódicamente el rendimiento y realizar back-ups periódicos

**b. Identificación de posibles mejoras y evolución del proyecto**

Podría identificarse como otro tipo de mantenimiento, pero enfocado a la mejora continua del proyecto añadiendo nuevas funcionalidades, migrar a nuevas tecnologías y mejorar la interfaz de usuario.

**c. Actualizaciones y mejoras futuras**

En este caso como se podrían implementar selectores de filtros para generar las estadísticas personalizadas en tiempo real. Es decir, limitar la búsqueda a un número determinado de resultados según lo estipule el usuario. Por ejemplo, con un selector que limite la búsqueda a 5, 50 ó 500;

Otra posible mejora es implementar un selector de estadísticas configurable, pasando determinados parámetros para generar las consultas necesarias y obtener los resultados deseados.

VII. Conclusiones

**a. Evaluación del proyecto**

El proyecto ha cumplido con los objetivos iniciales. La aplicación desarrollada permite la gestión eficiente de averías de máquinas de clientes, mejorando la productividad de los equipos de trabajo y la trazabilidad de los servicios solicitados por los clientes. La arquitectura basada en los patrones de diseño MVC, Singleton y DAO ha demostrado ser efectiva para asegurar una mayor modularidad, reutilización de código y facilidad de mantenimiento.

**b. Cumplimiento de objetivos y requisitos**

Se ha logrado:

* **Registro de usuarios**: Implementado de manera exitosa, permitiendo la creación de nuevas cuentas de usuarios.
* **Login seguro**: Validación y autenticación de credenciales de manera segura.
* **Gestión de clientes y máquinas**: Sistema funcional para agregar, modificar y eliminar información de clientes y máquinas.
* **Registro y seguimiento de avisos de averías**: Implementado el registro y seguimiento de incidencias, permitiendo una gestión eficaz de las mismas.
* **Asignación de técnicos a las averías**: Funcionalidad de asignación de técnicos implementada correctamente.
* **Registro de accesos de usuarios**: Historial de accesos para monitorear la actividad del usuario.
* **Interfaz de usuario intuitiva**: Diseño de una interfaz amigable, eficiente y accesible tanto en versión de escritorio como en versión para pantallas más reducidas.
* **Optimización del rendimiento**: Mejoras en consultas a la base de datos y en el rendimiento de la interfaz de usuario.

Con estas conclusiones, se puede afirmar que el proyecto no solo ha cumplido con los objetivos y requisitos planteados inicialmente, sino que también ha demostrado ser una solución eficaz y robusta para la gestión de averías de máquinas de clientes.

**c. Lecciones aprendidas y recomendaciones para futuros proyectos**

La organización es fundamental y aunque cada desarrollador puede establecer su propio método es fundamental seguir unas pautas que ya están establecidas para el desarrollo de cualquier aplicación o proyecto.

Efectuar un análisis correcto de los requisitos y necesidades, elaborar una planificación y seguir una metodología es la clave de un resultado exitoso.

VIII. Bibliografía y Referencias

**a. Fuentes utilizadas en el proyecto**

* Imágenes de máquinas tomadas de la web de linde-mh.com
* **Chart.js**
* *Descripción*: Librería JavaScript para crear gráficos interactivos.
* *Versión utilizada*: 3.7.1
* *Enlace*: Chart.js - Documentation

**b. Referencias y enlaces de interés**

|  |  |
| --- | --- |
| <https://aulasfp2324.castillalamancha.es/auth/oidc/>    Como herramienta de aprendizaje | <https://www.w3schools.com/>    Como herramienta de consulta |
| <https://developer.mozilla.org/es/docs/Web/JavaScript>    Como herramienta de consulta | <https://code.visualstudio.com/>    Entorno de desarrollo.  Versión: 1.97.2 |
| HTML5 | CSS 3 |
| JAVASCRIPT | PHP 8.2.4 |
| Librería JS para gráficos dinámicos | Como paquete que incluye el servidor APACHE, MariaDB y PHP para el despliegue local. V3.3.0 |
| Base de datos relacional | Para la gestión de la BBDD |
| Para realizar pruebas de API permitiendo enviar solicitudes http y verificar las respuestas | Como alojamiento del código fuente y los archivos necesarios. |
| Para el control de versiones. |  |