**Introducción**

Somos un equipo de estudiantes del Colegio Secundario “Alte. Brown” y hemos decidido emprender un proyecto que tiene como objetivo la creación de una página web dedicada a la asistencia en reparaciones y reconstrucciones de computadoras. Este servicio está dirigido tanto al colegio como al alumnado y a los profesores.

Nuestra iniciativa surge del deseo de aplicar y compartir los conocimientos adquiridos durante nuestra formación, ofreciendo una contribución significativa a nuestra comunidad educativa. A través de esta plataforma, buscamos facilitar el mantenimiento y la mejora de los equipos informáticos, esenciales para el proceso de enseñanza y aprendizaje.

Nuestro compromiso es proporcionar estos servicios de manera completamente gratuita, como una forma de retribuir y agradecer la educación que hemos recibido. Creemos firmemente en el poder de la colaboración y el apoyo mutuo, y estamos entusiasmados por la oportunidad de ayudar a mejorar las condiciones tecnológicas del colegio.

Esperamos que este esfuerzo conjunto beneficie a todos los miembros de nuestra escuela y ayude con un entorno de aprendizaje más eficiente y accesible.

**Situación Problemática**

En el contexto actual, nos enfrentamos a un desafío significativo relacionado con la operatividad de las computadoras (PC) en nuestro colegio. Observamos que existe un número considerable de equipos que, a pesar de su importancia para las actividades educativas, presentan problemas técnicos y no están funcionando correctamente.

**Objetivo**

Nuestro principal objetivo es garantizar el funcionamiento correcto y sostenible de las computadoras, asegurando su pleno uso y mantenimiento a lo largo del tiempo en el colegio. Reconocemos la importancia de una infraestructura tecnológica confiable y eficiente para apoyar las actividades educativas.

Nos guiamos por la estrategia de establecer objetivos alcanzables, enfocándonos en metas a corto plazo que sean fácilmente alcanzables. Esta metodología no solo promueve un progreso continuo, sino que también motiva y alienta a todos los miembros del equipo involucrados en el proyecto.

Los objetivos específicos se detallarán de manera concreta en la documentación correspondiente.

**Documentación del código frontend:**

**Arreglo.html**

Hemos implementado una opción especializada en nuestro sistema de soporte técnico que permite a nuestros técnicos seleccionar y gestionar las computadoras disponibles en el taller para su reparación. Esta funcionalidad está diseñada para optimizar el proceso de diagnóstico y reparación, debido a que cada tecnico puede variar su eleccion segun la cantidad de conocimiento que tenga, asegurando que cada equipo reciba la atención necesaria de manera eficiente.

**Asesoramiento**

Se utilizaría para garantizar una atención óptima y resolver cualquier inquietud que pueda surgir, hemos implementado un sistema de comunicación directa respecto a dudas,sugerencias e inquietudes. Este canal está diseñado para que usted pueda expresar sus dudas de manera clara y efectiva, permitiéndonos ofrecerle un diagnóstico preciso y oportuno.

Nuestro equipo de servicio técnico está a su disposición para proporcionarle la información y asistencia necesarias, asegurando así que todas sus consultas sean atendidas de forma profesional y eficiente.

**Limpieza.html**

Esta sección está diseñada específicamente para facilitar la solicitud de servicios de limpieza. Aquí encontrará toda la información necesaria para coordinar y gestionar sus necesidades de limpieza de manera eficiente.

**Style.css**

Esta sección está dedicada a establecer las características superficiales de la página, abarcando cada vista de manera individual. Proporciona una guía detallada sobre la apariencia del sitio, asegurando una presentación coherente y atractiva. Aquí se encuentran los elementos clave que conforman la estética y la estructura de cada vista.

**Index.html**

La vista principal de la página actúa como el núcleo central de organización y distribución de todas las demás vistas. Es la principal fuente de datos y estructura de la página, proporcionando la base sobre la cual se construyen y desarrollan las fases posteriores del sitio. Esta vista integra y unifica todos los componentes, asegurando una coherencia y funcionalidad óptimas en la experiencia del usuario.

**Contactos.html**

Esta sección está destinada a facilitar la comunicación directa con el personal del proyecto. Aquí se encuentran datos específicos como dirección, correo electrónico, teléfono, entre otros. Este recurso es utilizado para establecer contacto fuera del ámbito de asesoramiento de la página, permitiendo una comunicación más personalizada y detallada con nuestro equipo.

**Justificación de la Integración de Bootstrap en el Proyecto FixApp**

Bootstrap ofrece diversas ventajas para el desarrollo rápido y prototipado de interfaces de usuario. Con su conjunto de componentes prediseñados y estilos CSS, es posible construir interfaces atractivas y funcionales de manera eficiente, ideal para proyectos con restricciones de tiempo.

Además, su enfoque "mobile-first" garantiza que las aplicaciones sean responsivas desde dispositivos móviles hasta escritorios, asegurando una experiencia consistente. Bootstrap también facilita mantener una apariencia visual uniforme mediante estilos predefinidos y componentes estructurados, mejorando la coherencia estética y la experiencia del usuario. Con un soporte activo y una comunidad robusta, ofrece recursos extensos de documentación y ejemplos para resolver problemas y personalizar proyectos según las necesidades específicas, combinando la facilidad de uso con la flexibilidad de la personalización.

**Importancia para la integridad y estilización de la página**

* **Integridad estructural:** Bootstrap proporciona una estructura coherente y bien definida para el desarrollo front-end, lo que facilita la organización y mantenimiento del código. Esto es esencial para proyectos grandes o equipos donde múltiples desarrolladores trabajan en el mismo código base.
* **Estilización coherente:** Al usar Bootstrap, te aseguras de que los elementos de tu página estén estilizados de manera uniforme, siguiendo las mejores prácticas de diseño web. Esto no solo mejora la apariencia visual, sino que también contribuye a una experiencia de usuario más agradable y profesional.

**Justificación de la Integración de Node.js en el Proyecto FixApp**

A diferencia de los entornos de ejecución tradicionales de JavaScript que funcionan en el navegador, Node.js permite ejecutar código JavaScript en el servidor. Esto posibilita el desarrollo de aplicaciones del lado del servidor con JavaScript, un lenguaje que tradicionalmente se ha utilizado principalmente en el desarrollo del lado del cliente.

**Node.js ofrece varias ventajas fundamentales que lo hacen ideal para el desarrollo de aplicaciones modernas:**

Node.js está diseñado con un enfoque en la asincronía y el no bloqueo, permitiendo manejar múltiples solicitudes simultáneamente sin bloquear el hilo principal. Esto optimiza el rendimiento y la escalabilidad, especialmente en aplicaciones web en tiempo real y de alta concurrencia.

El uso de JavaScript tanto en el frontend como en el backend simplifica el desarrollo al permitir a los desarrolladores trabajar con un único lenguaje en toda la pila de la aplicación. Esto facilita la colaboración entre equipos y elimina la necesidad de aprender un nuevo lenguaje para el desarrollo del lado del servidor.

Node.js cuenta con npm (Node Package Manager), uno de los ecosistemas de paquetes más extensos del mundo. Con acceso a una amplia variedad de librerías y módulos, facilita la implementación de funcionalidades complejas y acelera el desarrollo de aplicaciones.

Gracias al motor V8 de Google, Node.js ejecuta código JavaScript a alta velocidad, optimizando la eficiencia del servidor. Su naturaleza no bloqueante asegura una utilización eficiente de los recursos del sistema.

Node.js cuenta con una comunidad activa y en expansión, lo cual garantiza un amplio soporte a través de documentación, foros de discusión y contribuciones de código abierto, asegurando que los desarrolladores siempre encuentren ayuda y recursos para mejorar sus proyectos.

**Explicación del código de conexión a la base de datos:**

**Instalar dependencias:**

* mysql2: Permite que Node.js hable con MySQL.
* dotenv: Mantiene las contraseñas y configuraciones fuera del código en un archivo .env.
* Configurar variables de entorno:

**Crea un archivo .env en tu proyecto**.

Dentro del archivo .env, define:

* DB\_HOST: Dirección del servidor de la base de datos.
* DB\_USER: Nombre de usuario de la base de datos.
* DB\_PASSWORD: Contraseña de la base de datos.
* DB\_DATABASE: Nombre de la base de datos.

**Crear el archivo de conexión a la base de datos:**

**Crea un archivo db.js.**

Importa mysql2 y dotenv.

* Usa dotenv para cargar las variables de entorno.
* Configura la conexión a la base de datos con las variables del archivo .env.
* Conéctate a la base de datos y muestra un mensaje si la conexión es exitosa o hay un error.
* Exporta la conexión para usarla en otros archivos. Usar la conexión en tu aplicación

Se realizo la conexión con el archivo server.js

Este mismo implementa la conexión a la base de datos con las siguientes herramientas y librerías:

const express = require('express'):

* Express permite crear servidores web, manejar rutas, y gestionar solicitudes y respuestas HTTP de manera eficiente.

const mysql = require('mysql2'):

* La biblioteca mysql2 es una librería de Node.js que permite interactuar con bases de datos MySQL y MariaDB desde una aplicación Node.js.

const path = require('path'):

* Facilita la manipulación y resolución de rutas en sistemas de archivos, permitiendo trabajar de manera consistente en diferentes sistemas operativos (Windows, macOS, Linux).

const bodyParser = require('body-parser'):

* La biblioteca body-parser es un middleware para aplicaciones Express en Node.js que se utiliza para analizar y procesar los cuerpos de las solicitudes HTTP.

Documentacion del Codigo del archivo server.js:

### Descripción general:

1. **Dependencias Importadas**:
   * express: Framework para crear aplicaciones web en Node.js.
   * mysql2/promise: Biblioteca para interactuar con MySQL de manera asíncrona usando promesas.
   * path: Módulo para trabajar con rutas de archivos.
   * body-parser: Middleware para parsear los datos del cuerpo de las solicitudes HTTP.
   * express-session: Middleware para gestionar sesiones de usuario.
   * bcrypt: Biblioteca para encriptar contraseñas de manera segura.
2. **Configuración del Servidor**:
   * **Conexión a la Base de Datos**: Se crea un pool de conexiones a la base de datos MySQL appfix con las credenciales proporcionadas.
   * **Middlewares**:
     + express.json() y express.urlencoded(): Para procesar las solicitudes que contienen datos JSON o datos URL codificados.
     + express.static(): Para servir archivos estáticos como HTML, CSS y JavaScript.
     + bodyParser(): Permite analizar el cuerpo de las solicitudes para obtener los datos enviados desde formularios HTML.
     + **Session Middleware**: Utiliza express-session para manejar sesiones de usuario. La sesión almacena el estado de autenticación de los usuarios.
3. **Ruta Principal**:
   * La ruta '/' es la página de inicio que verifica si el usuario está logueado y si es un administrador o no. Si el correo del usuario es 'admin@gmail.com', lo redirige a una página de administración, de lo contrario, lo redirige a la página principal de la aplicación.
4. **Autenticación y Registro**:
   * **Registro de Usuario** (/register):
     + Permite a un nuevo usuario registrarse proporcionando un nombre, correo y contraseña.
     + Antes de registrar, verifica que el correo no esté registrado previamente.
     + La contraseña se encripta con bcrypt antes de ser almacenada en la base de datos.
   * **Login de Usuario** (/login):
     + Permite a los usuarios autenticarse con su correo y contraseña.
     + Si las credenciales son correctas, el usuario se guarda en la sesión y se redirige según si es administrador o no.
5. **Protección de Rutas**:
   * **Middleware ensureLoggedIn**: Asegura que solo los usuarios logueados puedan acceder a ciertas rutas. Si no están logueados, son redirigidos a la página de login.
6. **Laboratorios y Tickets**:
   * **Agregar un laboratorio** (/add-lab): Permite a un usuario agregar un laboratorio a su cuenta.
   * **Obtener laboratorios** (/labs): Recupera los laboratorios asociados al usuario logueado.
   * **Agregar un arreglo (ticket)** (/arreglo): Permite a un usuario solicitar un servicio de arreglo para un equipo. Valida la existencia del laboratorio y el tipo de servicio, luego inserta un ticket en la base de datos.
   * **Obtener Tickets** (/tickets): Recupera todos los tickets emitidos, incluyendo detalles sobre el laboratorio, equipo, y tipo de servicio.
   * **Eliminar Ticket** (/tickets/:ticketId): Permite eliminar un ticket y su equipo asociado.
7. **Función de Cierre de Sesión**:
   * **Cerrar sesión** (/logout): Elimina la sesión del usuario y redirige a la página de login.
8. **Formulario de Contacto**:
   * **Formulario de contacto** (/contacto): Permite que los usuarios envíen un mensaje de contacto, que luego se guarda en la base de datos.

### Explicación de Rutas Importantes:

1. **Ruta '/'**:
   * **Middleware ensureLoggedIn**: Verifica que el usuario esté logueado.
   * Si el usuario está logueado y es un administrador, se redirige a la página de administración. Si no es administrador, se le redirige a la página principal.
2. **Ruta POST /login**:
   * Recibe los datos de correo y contraseña.
   * Si el correo existe en la base de datos, se compara la contraseña con el valor encriptado usando bcrypt.
   * Si la contraseña es correcta, el usuario se guarda en la sesión y se redirige según el tipo de usuario (administrador o no).
3. **Ruta POST /register**:
   * Se reciben los datos de nombre, correo y contraseña.
   * Si el correo ya está registrado, devuelve un error. Si no, encripta la contraseña y guarda el nuevo usuario en la base de datos.
4. **Ruta POST /arreglo**:
   * Permite a los usuarios solicitar un servicio de arreglo para un equipo. Verifica que el laboratorio y el tipo de servicio existan antes de crear un ticket de arreglo.
5. **Ruta GET /labs**:
   * Obtiene todos los laboratorios asociados al usuario logueado.
6. **Ruta GET /tickets**:
   * Obtiene todos los tickets y los devuelve con detalles sobre el laboratorio y el tipo de servicio.
7. **Ruta DELETE /tickets/:ticketId**:
   * Elimina un ticket y su equipo asociado de la base de datos.
8. **Ruta POST /logout**:
   * Cierra la sesión del usuario y redirige al login.

### Seguridad:

* **Encriptación de contraseñas**: El código usa bcrypt para encriptar las contraseñas antes de almacenarlas en la base de datos, lo que es más seguro que almacenar contraseñas en texto plano.
* **Sesiones**: Utiliza express-session para mantener el estado de autenticación del usuario durante su navegación, evitando que un usuario tenga que autenticarse en cada solicitud.

### Consideraciones adicionales:

* **Rutas protegidas**: Algunas rutas, como la creación de tickets y la consulta de laboratorios, están protegidas por un middleware que asegura que el usuario esté logueado.
* **Redirección según el rol**: Dependiendo del correo del usuario, se lo redirige a la página adecuada (administrador o no).

FormSubmit

Proveedor de envió de formularios:

El formulario se conecta con el proveedor con el siguiente link, dentro de la sentencia del mismo formulario.

<form action="https://formsubmit.co/ tu@correoelectrónico.com " method="POST" />