📌 Tecnologías Elegidas para el Sistema de Gestión de Taller Mecánico

Queremos desarrollar una aplicación versátil, accesible desde PC y móvil, con PostgreSQL como base de datos y posibilidad de integración con WhatsApp. Por ello, hemos seleccionado las siguientes tecnologías:

1.- Backend (Servidor y API REST)

📌 Tecnología: Node.js + Express.js

✅ ¿Por qué?

Rápido y escalable, ideal para aplicaciones web modernas.

Manejo eficiente de múltiples conexiones (útil si crece el número de usuarios).

Express.js simplifica la creación de API REST para conectar el frontend con PostgreSQL.

Se puede desplegar fácilmente en la nube (AWS, DigitalOcean, Railway).

📌 Base de datos: PostgreSQL

✅ ¿Por qué?

Más robusto que MySQL para manejar múltiples conexiones simultáneas.

Mejor manejo de datos estructurados y JSON (útil para reportes y facturación).

Escalable, ideal si en el futuro agregamos más funcionalidades.

📌 ORM: Sequelize

✅ ¿Por qué?

Facilita la conexión entre Node.js y PostgreSQL.

Evita escribir consultas SQL manualmente, reduciendo errores.

Mejor mantenimiento del código con modelos de datos claros.

📌 Autenticación: JWT + BCrypt

✅ ¿Por qué?

JWT (JSON Web Tokens) permite autenticación segura sin depender de cookies.

BCrypt cifra las contraseñas, protegiendo los datos de los usuarios.

2.- Frontend (Interfaz Gráfica Web y Móvil)

📌 Tecnología: React.js + Tailwind CSS

✅ ¿Por qué?

React.js permite construir una interfaz moderna y dinámica.

Tailwind CSS agiliza el diseño y lo hace 100% responsivo (adaptado a PC y móvil).

Se puede convertir en una PWA (Progressive Web App) para instalar en móviles.

📌 PWA (Progressive Web App)

✅ ¿Por qué?

Permite instalar la web como una "app nativa" en Android y iOS.

Funciona offline si configuramos un caché local.

Ahorra costos, ya que no necesitamos desarrollar una app móvil nativa separada.

3.- Integración con WhatsApp

📌 Tecnología: Twilio WhatsApp API

✅ ¿Por qué?

Automatiza mensajes (avisos de reparaciones listas, presupuestos, facturas).

Más estable y profesional que usar WhatsApp Web manualmente.

Permite escalar a más clientes sin bloquear la cuenta.

4.- Infraestructura y Despliegue

📌 Servidor: VPS con Docker (AWS, DigitalOcean, Railway)

✅ ¿Por qué?

Docker facilita la instalación y migración del sistema.

Railway o DigitalOcean ofrecen hosting barato y escalable.

📌 Base de datos en la nube: Supabase o PostgreSQL en un VPS

✅ ¿Por qué?

Supabase es PostgreSQL con hosting gratuito y API REST integrada.

PostgreSQL en VPS si necesitamos control total sobre los datos.

📌 Resumen Final

Tabla

El contenido generado por IA puede ser incorrecto.

Credenciales PostgresSQL:

Contraseña: PostgresSQL17021983BrfB

Puerto: 5432  
Configuración regional: Spanish

Sumario en Animus:

summary.installation.directory: C:\Program Files\PostgreSQL\17

summary.server.installation.directory: C:\Program Files\PostgreSQL\17

summary.data.directory: C:\Program Files\PostgreSQL\17\data

summary.database.port: 5432

summary.database.superuser: postgres

summary.serviceaccount: NT AUTHORITY\NetworkService

summary.databaseservice: postgresql-x64-17

summary.clt.installation.directory: C:\Program Files\PostgreSQL\17

summary.pgadmin.installation.directory: C:\Program Files\PostgreSQL\17\pgAdmin 4

summary.sbp.installation.directory: C:\Program Files\PostgreSQL\17

summary.installation.logfile: C:\Users\AlvaroGT\AppData\Local\Temp\install-postgresql.log

Sumario en Skainet:  
  
summary.installation.directory: C:\Program Files\PostgreSQL\17

summary.server.installation.directory: C:\Program Files\PostgreSQL\17

summary.data.directory: C:\Program Files\PostgreSQL\17\data

summary.database.port: 5432

summary.database.superuser: postgres

summary.serviceaccount: NT AUTHORITY\NetworkService

summary.databaseservice: postgresql-x64-17

summary.clt.installation.directory: C:\Program Files\PostgreSQL\17

summary.pgadmin.installation.directory: C:\Program Files\PostgreSQL\17\pgAdmin 4

summary.sbp.installation.directory: C:\Program Files\PostgreSQL\17

summary.installation.logfile: C:\Users\AlvaroGT\AppData\Local\Temp\install-postgresql.log

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Servicio de Supabase para albergar la base de datos PostgresSQL:

Url del proyecto: <https://gxcexgzaavrffqqmwvxu.supabase.co>

Clave de la API: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Imd4Y2V4Z3phYXZyZmZxcW13dnh1Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NDIwNDI0MDksImV4cCI6MjA1NzYxODQwOX0.AN7fiURec-0npZk0fdaA8SOB\_WleY1DSIYcSDIE9IB0

Para JavaScript:

import { createClient } from '@supabase/supabase-js'

const supabaseUrl = 'https://gxcexgzaavrffqqmwvxu.supabase.co'

const supabaseKey = process.env.SUPABASE\_KEY

const supabase = createClient(supabaseUrl, supabaseKey)

Para Dart:

const supabaseUrl = 'https://gxcexgzaavrffqqmwvxu.supabase.co';

const supabaseKey = String.fromEnvironment('SUPABASE\_KEY');

Future<void> main() async {

await Supabase.initialize(url: supabaseUrl, anonKey: supabaseKey);

runApp(MyApp());

}

Estructura de la que partimos al inicio del proyecto:

Tabla

El contenido generado por IA puede ser incorrecto.

A continuación se muestra un resumen descriptivo de cada carpeta y archivo principal de la estructura del proyecto, el cual puedes incluir en el dosier:

* **proyecto/**  
  Carpeta raíz del proyecto, que contiene todos los componentes (código fuente, configuraciones, documentación, etc.).
* **node\_modules/**  
  Directorio generado automáticamente por npm, que almacena todas las dependencias y paquetes instalados para el proyecto.
* **src/**  
  Carpeta que contiene el código fuente de la aplicación. Dentro se organizan los módulos del backend, frontend y, opcionalmente, la parte móvil.
  + **src/backend/**  
    Contiene todo el código del servidor (backend):
    - **configuracion/**  
      Archivos de configuración, como la conexión a la base de datos.
      * *base\_de\_datos.js*: Configuración y parámetros para conectar con la base de datos (por ejemplo, la base alojada en Supabase).
    - **controladores/**  
      Implementan la lógica de negocio para cada recurso o entidad.
      * *reparacion\_controlador.js*: Controlador que gestiona las operaciones relacionadas con reparaciones, facturas, etc.
    - **modelos/**  
      Define los modelos de datos (tablas) que se utilizarán en la aplicación, especialmente si se usa un ORM.
      * *Reparacion.js*: Modelo que representa una reparación en la base de datos.
    - **rutas/**  
      Contiene la definición de rutas y endpoints de la API del backend.
      * *reparacion\_rutas.js*: Define las rutas para gestionar reparaciones y otros procesos.
    - **intermedios/**  
      Incluye funciones middleware, por ejemplo, para autenticación o validación, que se ejecutan en el flujo de las peticiones.
      * *middleware\_autenticacion.js*: Middleware encargado de gestionar la autenticación de los usuarios.
    - **servicios/**  
      Encapsula la lógica de negocio compleja y los procesos relacionados con cada entidad.
      * *reparacion\_servicio.js*: Servicio que contiene funciones para gestionar las operaciones de reparaciones.
    - **utilidades/**  
      Proporciona helpers o funciones de utilidad que pueden ser usadas en distintas partes del backend.
      * *registro.js*: Función para llevar el registro o logs de las operaciones y errores.
    - *index.js*  
      Es el punto de entrada del backend; configura el servidor, carga middlewares globales y define la inicialización del servicio.
  + **src/frontend/**  
    Contiene el código del cliente (frontend) que se encarga de la interfaz de usuario:
    - **public/**  
      Almacena los archivos estáticos como hojas de estilo (CSS), scripts (JavaScript) e imágenes.
    - **vistas/**  
      Incluye las plantillas de las vistas (por ejemplo, con EJS, Pug o HTML estático).
      * *index.ejs*: La plantilla principal que renderiza la interfaz de la aplicación.
    - *index.js*  
      Código JavaScript del lado del cliente, que puede gestionar la interacción en la interfaz (opcional).
  + **src/mobile/**  
    (Opcional) Directorio destinado al código fuente o a las instrucciones para compilar la aplicación móvil, si se decide incluir una versión nativa o híbrida.
    - *README.md*: Documento con indicaciones para compilar o trabajar con la aplicación móvil.
* **.env**  
  Archivo que almacena las variables de entorno (como credenciales y parámetros de configuración). Debe estar incluido en el archivo .gitignore para evitar su publicación.
* **.gitignore**  
  Lista de archivos y carpetas que no se deben incluir en el control de versiones (por ejemplo, node\_modules y .env).
* **package.json**  
  Archivo central de npm que contiene la información del proyecto, scripts de ejecución y las dependencias instaladas.
* **README.md**  
  Documento con la documentación básica del proyecto, que explica su propósito, estructura y cómo configurarlo o ejecutarlo.