**DOCUMENTO DE DISEÑO DETALLADO DE SOFTWARE**

1. Plantilla inspirada en el estándar IEEE 1016-2009 y adaptada a las necesidades del curso de Construcción de Software

(Plantilla compilada por Ph.D. Franklin Parrales B.)

***1. Selección de Tecnologías***

**Objetivo:**  
Definir y justificar el stack tecnológico para el sistema de gestión vehicular, garantizando escalabilidad, mantenibilidad y eficiencia en el desarrollo.

***2. Tecnologías Propuestas***

***2.1. Lenguajes de Programación***

* **Frontend:**
  + HTML5, CSS3 y JavaScript (ES6+)

**Fundamentación:** Tecnologías estándar para el desarrollo web, compatibles con todos los navegadores modernos y adecuadas para construir interfaces dinámicas y responsivas.

* **Backend (opcional para futuras etapas):**
* Node.js

**Fundamentación:** Permite utilizar JavaScript en el servidor, facilitando la integración con el frontend y reduciendo la curva de aprendizaje para equipos que ya dominan JavaScript.

***2.2. Frameworks y Bibliotecas***

* **Frontend:**
  + Bootstrap 5 o Tailwind CSS

**Fundamentación:** Frameworks CSS que agilizan el desarrollo de interfaces responsivas mediante componentes pre-diseñados y utilidades personalizables.

* + React.js (opcional para escalabilidad)

**Fundamentación:** Biblioteca que permite construir interfaces modulares y reutilizables, ideal para aplicaciones complejas que requieran alta interactividad.

* **Backend (opcional):**
  + Express.js

**Fundamentación:** Framework minimalista para Node.js, ideal para desarrollar APIs RESTful de manera eficiente.

***2.3. Herramientas de Desarrollo***

* **Entorno de Desarrollo Integrado (IDE):**
  + Visual Studio Code

**Fundamentación:** Editor ligero con soporte nativo para JavaScript, integración con herramientas de depuración y extensiones útiles (ESLint, Prettier, Live Server).

* **Control de Versiones:**
  + Git + GitHub/GitLab

**Fundamentación:** Sistemas estándar para gestión de código fuente, colaboración en equipo y despliegue continuo.

* **Gestión de Dependencias:**
  + npm o Yarn

**Fundamentación:** Gestores de paquetes ampliamente adoptados en el ecosistema JavaScript, necesarios para instalar bibliotecas externas.

***2.4. Almacenamiento de Datos***

* **Almacenamiento Local (implementación actual):**
  + LocalStorage

**Fundamentación:** Solución práctica para prototipos o aplicaciones pequeñas que no requieren persistencia de datos en servidor.

* **Alternativas para Escalabilidad:**
  + Firebase/Firestore

**Fundamentación:** Base de datos NoSQL en la nube con integración sencilla para aplicaciones web, ideal para proyectos que requieren autenticación y almacenamiento en tiempo real.

* + PostgreSQL

**Fundamentación:** Sistema de gestión de bases de datos relacionales robusto, adecuado para aplicaciones con modelos de datos complejos.

***2.5. Herramientas Complementarias***

* Testing:
  + Jest

**Fundamentación:** Framework de pruebas unitarias para JavaScript, esencial para garantizar la calidad del código en funciones críticas (ejemplo: cálculos financieros en órdenes de trabajo).

* Bundlers y Build Tools:
  + Webpack o Vite

**Fundamentación:** Herramientas para optimizar y empaquetar recursos estáticos (CSS, JS), mejorando el rendimiento en producción.

***3. Análisis y Justificación***

***3.1. Enfoque Actual (HTML/CSS/JS + LocalStorage)***

* **Ventajas:**
  + **Rápido desarrollo:** No requiere configuración de servidor.
  + **Bajo costo:** Despliegue sencillo en hosting estático (ejemplo: GitHub Pages, Netlify).
  + **Suficiente para MVP:** Adecuado para sistemas pequeños con uso local.
* **Limitaciones:**
  + **Escalabilidad:** LocalStorage no es óptimo para grandes volúmenes de datos o múltiples usuarios.
  + **Seguridad:** Los datos se almacenan en el cliente, sin cifrado avanzado.

***3.2. Recomendaciones para Escalabilidad***

1. **Migrar a una Arquitectura Full-Stack:**
   * **Frontend:** React.js para una interfaz modular.
   * **Backend:** Node.js + Express.js para construir una API REST.
   * **Base de Datos:** Firebase (para desarrollo ágil) o PostgreSQL (para modelos relacionales).
2. **Implementar Autenticación:**
   * Integrar Firebase Auth o JWT (JSON Web Tokens) para gestión segura de usuarios.
3. **Despliegue en la Nube:**
   * Usar servicios como Vercel (frontend) y Render/Heroku (backend) para mayor disponibilidad.