**Sistema de Gestión de Flota**

PepsiCo Chile

STACK TECNOLÓGICO

*Tecnologías y Herramientas Utilizadas*

Fecha: 14 de octubre de 2025

Equipo: Joaquín Marín y Benjamin Vilches

Duoc UC - Proyecto de Título 2025

# DEFINICIÓN DEL STACK TECNOLÓGICO

Proyecto: Plataforma de Gestión de Ingreso de Vehículos - PepsiCo Chile

Fecha: 13/10/2025

Responsables: Joaquín Marín y Benjamin Vilches

Versión: 1.0

1. RESUMEN EJECUTIVO

Este documento define el stack tecnológico completo para el desarrollo de la

Plataforma de Gestión de Ingreso de Vehículos. Todas las tecnologías

seleccionadas son Open Source, cumpliendo con la restricción de presupuesto $0.

2. ARQUITECTURA GENERAL

Arquitectura: Cliente-Servidor (SPA - Single Page Application)

Patrón: API RESTful

Enfoque: Arquitectura de 3 capas

┌─────────────────────────────────────────────────────────────┐
│ CAPA DE PRESENTACIÓN │
│ React.js + TypeScript + Vite │
│ (Frontend - SPA) │
└─────────────────────────────────────────────────────────────┘
↕ HTTP/HTTPS
┌─────────────────────────────────────────────────────────────┐
│ CAPA DE APLICACIÓN │
│ Node.js + Express.js + TypeScript │
│ (Backend - API REST) │
└─────────────────────────────────────────────────────────────┘
↕ SQL
┌─────────────────────────────────────────────────────────────┐
│ CAPA DE DATOS │
│ PostgreSQL 15+ │
│ (Base de Datos) │
└─────────────────────────────────────────────────────────────┘

3. STACK TECNOLÓGICO DETALLADO

## 3.1. FRONTEND (Capa de Presentación)

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Tecnología │ Versión │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ React.js │ 18.3+ │ - Librería más popular y demandada │
│ │ │ - Componentización y reutilización │
│ │ │ - Gran comunidad y documentación │
│ │ │ - Curva de aprendizaje moderada │
│ │ │ - Virtual DOM para rendimiento │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ TypeScript │ 5.5+ │ - Tipado estático reduce errores │
│ │ │ - Mejor autocompletado en IDE │
│ │ │ - Refactorización más segura │
│ │ │ - Documentación de código implícita │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Vite │ 5.4+ │ - Build tool moderno y rápido │
│ │ │ - Hot Module Replacement (HMR) │
│ │ │ - Configuración mínima │
│ │ │ - Mejor experiencia de desarrollo │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ React Router │ 6.26+ │ - Navegación SPA estándar │
│ │ │ - Protección de rutas por rol │
│ │ │ - Carga diferida (lazy loading) │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ TanStack Query │ 5.56+ │ - Gestión de estado servidor │
│ (React Query) │ │ - Caché automático │
│ │ │ - Sincronización de datos │
│ │ │ - Manejo de estados de carga/error │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Zustand │ 4.5+ │ - Gestión de estado global simple │
│ │ │ - Alternativa ligera a Redux │
│ │ │ - Sin boilerplate │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Tailwind CSS │ 3.4+ │ - Desarrollo rápido de UI │
│ │ │ - Responsive design sencillo │
│ │ │ - Consistencia visual │
│ │ │ - Bundle optimizado │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ shadcn/ui │ Latest │ - Componentes accesibles (a11y) │
│ │ │ - Basados en Radix UI │
│ │ │ - Customizables │
│ │ │ - Copiar/pegar (no dependencia) │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Axios │ 1.7+ │ - Cliente HTTP robusto │
│ │ │ - Interceptores para autenticación │
│ │ │ - Manejo de errores centralizado │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ React Hook Form │ 7.53+ │ - Validación de formularios eficiente │
│ │ │ - Rendimiento optimizado │
│ │ │ - Integración con Zod │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Zod │ 3.23+ │ - Validación de esquemas │
│ │ │ - TypeScript-first │
│ │ │ - Reutilizable en front y back │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ date-fns │ 3.6+ │ - Manipulación de fechas ligera │
│ │ │ - Inmutable y modular │
│ │ │ - Soporte de i18n │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Recharts │ 2.12+ │ - Gráficos para reportes │
│ │ │ - Basado en React y D3 │
│ │ │ - Responsivo │
└──────────────────┴──────────────┴───────────────────────────────────────┘

## 3.2. BACKEND (Capa de Aplicación)

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Tecnología │ Versión │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Node.js │ 20 LTS │ - Runtime JavaScript más usado │
│ │ │ - Asíncrono y escalable │
│ │ │ - Mismo lenguaje que frontend │
│ │ │ - Amplio ecosistema (npm) │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ TypeScript │ 5.5+ │ - Consistencia con frontend │
│ │ │ - Prevención de errores │
│ │ │ - Mejor mantenibilidad │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Express.js │ 4.19+ │ - Framework minimalista y flexible │
│ │ │ - Gran comunidad │
│ │ │ - Middleware ecosystem │
│ │ │ - Documentación abundante │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Prisma ORM │ 5.20+ │ - ORM moderno type-safe │
│ │ │ - Migraciones automáticas │
│ │ │ - Excelente DX (Developer Experience) │
│ │ │ - Autocompletado de queries │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ JWT │ 9.0+ │ - Autenticación stateless │
│ (jsonwebtoken) │ │ - Estándar de la industria │
│ │ │ - Escalable sin sesiones en servidor │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ bcrypt │ 5.1+ │ - Hashing seguro de contraseñas │
│ │ │ - Protección contra rainbow tables │
│ │ │ - Configurable (salt rounds) │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Zod │ 3.23+ │ - Validación de datos de entrada │
│ │ │ - Compartido con frontend │
│ │ │ - Type safety │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ cors │ 2.8+ │ - Configuración de CORS │
│ │ │ - Seguridad en comunicación API │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ helmet │ 7.1+ │ - Headers de seguridad HTTP │
│ │ │ - Protección contra vulnerabilidades │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ express-rate- │ 7.4+ │ - Limitación de peticiones │
│ limit │ │ - Prevención de ataques DDoS │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ multer │ 1.4+ │ - Carga de archivos (imágenes) │
│ │ │ - Manejo de multipart/form-data │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ nodemailer │ 6.9+ │ - Envío de notificaciones por email │
│ │ │ - Soporte SMTP │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ pdfkit │ 0.15+ │ - Generación de reportes PDF │
│ │ │ - Personalización completa │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ exceljs │ 4.4+ │ - Generación de reportes Excel │
│ │ │ - Lectura/escritura .xlsx │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ dotenv │ 16.4+ │ - Gestión de variables de entorno │
│ │ │ - Configuración por ambiente │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ winston │ 3.14+ │ - Sistema de logging robusto │
│ │ │ - Múltiples transportes │
│ │ │ - Niveles de log configurables │
└──────────────────┴──────────────┴───────────────────────────────────────┘

# 3.3. BASE DE DATOS

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Tecnología │ Versión │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ PostgreSQL │ 15+ │ - Base de datos relacional robusta │
│ │ │ - ACID compliance │
│ │ │ - Soporte JSON (flexibilidad) │
│ │ │ - Excelente para relaciones complejas │
│ │ │ - Open Source completo │
│ │ │ - Mejor que MySQL en características │
│ │ │ - Triggers y stored procedures │
│ │ │ - Full-text search nativo │
└──────────────────┴──────────────┴───────────────────────────────────────┘

Alternativa evaluada: MySQL

**❌ Rechazada porque PostgreSQL ofrece:**

* Mejor manejo de concurrencia (MVCC)
* Tipos de datos más ricos
* Mejor cumplimiento de estándares SQL
* Extensibilidad superior

# 3.4. HERRAMIENTAS DE DESARROLLO

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Herramienta │ Propósito │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Git │ Control de │ - Estándar de la industria │
│ │ versiones │ - Trabajo colaborativo │
│ │ │ - Historial completo de cambios │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ GitHub │ Repositorio │ - Hosting gratuito │
│ │ remoto │ - GitHub Actions (CI/CD gratis) │
│ │ │ - Issues y project boards │
│ │ │ - Profesional y reconocido │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ VS Code │ IDE │ - Editor más popular │
│ │ │ - Extensiones para cada tecnología │
│ │ │ - IntelliSense potente │
│ │ │ - Gratuito y ligero │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Postman / │ Testing API │ - Pruebas de endpoints │
│ Thunder Client │ │ - Documentación de API │
│ │ │ - Colecciones compartidas │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Prisma Studio │ DB GUI │ - Interfaz visual para BD │
│ │ │ - Integrado con Prisma │
│ │ │ - Desarrollo y debugging │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ ESLint │ Linting JS/TS│ - Calidad de código │
│ │ │ - Detección temprana de errores │
│ │ │ - Estándares consistentes │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Prettier │ Formateo │ - Código consistente │
│ │ │ - Automático │
│ │ │ - Integración con ESLint │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Figma │ Diseño UI/UX │ - Colaborativo y gratuito │
│ │ │ - Mockups y prototipos │
│ │ │ - Handoff a desarrollo │
└──────────────────┴──────────────┴───────────────────────────────────────┘

# 3.5. TESTING

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Tecnología │ Tipo │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Vitest │ Unit Testing │ - Compatible con Vite │
│ │ (Frontend) │ - Sintaxis Jest-like │
│ │ │ - Rápido │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ React Testing │ Component │ - Testing de componentes React │
│ Library │ Testing │ - Buenas prácticas de testing │
│ │ │ - User-centric │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Jest │ Unit Testing │ - Estándar para Node.js │
│ │ (Backend) │ - Mocking potente │
│ │ │ - Cobertura de código │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Supertest │ API Testing │ - Testing de endpoints │
│ │ │ - Integración con Express │
└──────────────────┴──────────────┴───────────────────────────────────────┘

# 3.6. DESPLIEGUE (DEPLOYMENT)

────────────────────────────────────────────────────────────────

┌──────────────────┬──────────────┬───────────────────────────────────────┐
│ Servicio │ Uso │ Justificación │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Vercel │ Frontend │ - Deploy automático desde GitHub │
│ │ │ - CDN global │
│ │ │ - HTTPS gratuito │
│ │ │ - Plan gratuito generoso │
│ │ │ - Optimizado para React │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Railway / │ Backend │ - Deploy de Node.js simple │
│ Render │ │ - Plan gratuito disponible │
│ │ │ - Variables de entorno │
│ │ │ - Logs y monitoreo │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Neon / │ Base de │ - PostgreSQL serverless │
│ Supabase │ Datos │ - Plan gratuito (hasta 500MB) │
│ │ │ - Backups automáticos │
│ │ │ - Sin gestión de servidor │
├──────────────────┼──────────────┼───────────────────────────────────────┤
│ Cloudinary │ Almacenamiento│ - Hosting de imágenes │
│ │ de imágenes │ - Transformaciones automáticas │
│ │ │ - CDN incluido │
│ │ │ - Plan gratuito: 25 GB/mes │
└──────────────────┴──────────────┴───────────────────────────────────────┘

4. ESTRUCTURA DEL PROYECTO

## 4.1. MONOREPO (Recomendado)

────────────────────────────────────────────────────────────────

pepsico-fleet-management/

├── frontend/ # Aplicación React
│ ├── src/
│ │ ├── components/ # Componentes reutilizables
│ │ ├── pages/ # Páginas/vistas
│ │ ├── hooks/ # Custom hooks
│ │ ├── services/ # Llamadas a API
│ │ ├── store/ # Estado global (Zustand)
│ │ ├── utils/ # Utilidades
│ │ ├── types/ # Tipos TypeScript
│ │ ├── schemas/ # Esquemas Zod
│ │ └── App.tsx
│ ├── public/
│ ├── package.json
│ └── vite.config.ts
│
├── backend/ # API Node.js
│ ├── src/
│ │ ├── controllers/ # Controladores
│ │ ├── routes/ # Definición de rutas
│ │ ├── middlewares/ # Middlewares
│ │ ├── services/ # Lógica de negocio
│ │ ├── models/ # Tipos y interfaces
│ │ ├── utils/ # Utilidades
│ │ ├── config/ # Configuraciones
│ │ └── index.ts # Punto de entrada
│ ├── prisma/
│ │ ├── schema.prisma # Definición del modelo de datos
│ │ └── migrations/ # Migraciones
│ ├── tests/
│ ├── package.json
│ └── tsconfig.json
│
├── shared/ # Código compartido
│ ├── types/ # Tipos compartidos
│ └── schemas/ # Esquemas Zod compartidos
│
├── docs/ # Documentación
│ ├── api/ # Documentación API
│ ├── arquitectura/ # Diagramas
│ └── manuales/ # Manuales de usuario
│
├── .github/
│ └── workflows/ # CI/CD
│
├── README.md
└── package.json # Root package.json

5. COMPATIBILIDAD CON REQUERIMIENTOS

┌────────────────────────────────┬──────────────────────────────────────┐
│ Requerimiento │ Tecnología que lo soporta │
├────────────────────────────────┼──────────────────────────────────────┤
│ 50+ usuarios concurrentes │ Node.js (async), PostgreSQL (MVCC) │
│ Tiempo respuesta < 3s │ React Query (caché), índices en BD │
│ Interfaz responsiva │ Tailwind CSS, diseño mobile-first │
│ Control de roles (RBAC) │ JWT + middleware de autorización │
│ Carga de imágenes │ Multer + Cloudinary │
│ Reportes PDF/Excel │ pdfkit + exceljs │
│ Notificaciones │ nodemailer │
│ Seguridad │ bcrypt, JWT, helmet, CORS │
│ Disponibilidad 99.9% │ Vercel + Railway/Render (uptime) │
│ Backups automáticos │ Neon/Supabase (daily backups) │
└────────────────────────────────┴──────────────────────────────────────┘

# 6. COSTOS

TOTAL: $0 (USD)

Todos los servicios tienen planes gratuitos suficientes para el proyecto:

✓ Vercel Free Tier: 100 GB bandwidth/mes

✓ Railway Free Tier: 500 horas ejecución/mes

✓ Neon Free Tier: 500 MB storage, 0.5 vCPU

✓ Cloudinary Free: 25 GB/mes, 25,000 transformaciones

✓ GitHub: Repositorio ilimitado

7. RIESGOS TÉCNICOS Y MITIGACIÓN

┌────────────────────────────────┬──────────────────────────────────────┐
│ Riesgo │ Mitigación │
├────────────────────────────────┼──────────────────────────────────────┤
│ Curva de aprendizaje TypeScript│ Documentación oficial, tutoriales │
│ │ Puede usarse JS mientras se aprende │
├────────────────────────────────┼──────────────────────────────────────┤
│ Complejidad de Prisma │ Excelente documentación │
│ │ Prisma Studio para visualización │
├────────────────────────────────┼──────────────────────────────────────┤
│ Límites de planes gratuitos │ Optimización desde el inicio │
│ │ Monitoreo de uso │
├────────────────────────────────┼──────────────────────────────────────┤
│ Integración entre capas │ Tipos compartidos (TypeScript) │
│ │ Esquemas Zod compartidos │
└────────────────────────────────┴──────────────────────────────────────┘

8. VENTAJAS DE ESTE STACK

✓ Full TypeScript: Type safety en toda la aplicación

✓ Developer Experience: Herramientas modernas y eficientes

✓ Escalabilidad: Arquitectura preparada para crecer

✓ Mantenibilidad: Código limpio y bien estructurado

✓ Productividad: Menos boilerplate, más desarrollo

✓ Comunidad: Todas las tecnologías tienen gran soporte

✓ Empleabilidad: Stack muy demandado en la industria

✓ Costo: $0, cumple restricción presupuestaria

9. COMPARACIÓN CON ALTERNATIVAS

ALTERNATIVA 1: MERN Stack (MongoDB + Express + React + Node)

❌ MongoDB: Base de datos NoSQL no óptima para datos relacionales complejos

❌ Sin TypeScript nativo en muchos ejemplos

✓ Popular y con mucha documentación

Decisión: Rechazada por naturaleza relacional de los datos

ALTERNATIVA 2: Laravel + Vue

✓ Framework robusto y completo

✓ ORM Eloquent muy potente

❌ PHP menos demandado que JavaScript en el mercado

❌ Dos lenguajes diferentes (PHP + JavaScript)

❌ Curva de aprendizaje más alta

Decisión: Rechazada por stack tecnológico menos unificado

ALTERNATIVA 3: Django + React

✓ Python popular y fácil de aprender

✓ Django admin panel automático

❌ Menos demanda en frontend que React

❌ Dos lenguajes diferentes

❌ Configuración inicial más compleja

Decisión: Rechazada por preferir JavaScript full-stack

10. HOJA DE RUTA TECNOLÓGICA

FASE 1: Setup (Semana 5)

□ Configurar repositorio Git

□ Inicializar proyecto frontend (Vite + React + TS)

□ Inicializar proyecto backend (Express + TS)

□ Configurar Prisma y conectar PostgreSQL

□ Configurar ESLint y Prettier

□ Setup básico de CI/CD

FASE 2: Core Development (Semanas 5-8)

□ Implementar autenticación JWT

□ Desarrollar sistema de roles (RBAC)

□ Crear componentes base UI

□ Implementar CRUD de entidades principales

□ Integrar carga de imágenes

FASE 3: Features Avanzadas (Semanas 8-10)

□ Sistema de notificaciones

□ Generación de reportes

□ Dashboard con gráficos

□ Optimizaciones de rendimiento

FASE 4: Testing y Deploy (Semanas 10-12)

□ Pruebas unitarias

□ Pruebas de integración

□ Deploy a producción

□ Monitoreo y ajustes

11. CONCLUSIÓN

Este stack tecnológico representa una solución moderna, profesional y

completamente gratuita que cumple con todos los requerimientos del proyecto.

La elección de React + Node.js + PostgreSQL con TypeScript proporciona:

* Un ecosistema unificado (JavaScript/TypeScript)
* Excelente rendimiento y escalabilidad
* Gran demanda en el mercado laboral
* Herramientas de desarrollo de clase mundial
* Comunidad activa y soporte abundante

El stack está validado y listo para iniciar el desarrollo.

APROBACIÓN

**Elaborado por:**

Joaquín Marín - Gerente de Proyecto / Desarrollador Frontend

Benjamin Vilches - Gerente de Proyecto / Desarrollador Backend

Fecha: 13/10/2025

**Aprobado por:**

[ ] Alexis González - Patrocinador

[ ] Fabián Álvarez - Docente

Firma: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Fecha: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_