# 📊 Diagramas UML Técnicos - PepsiCo Fleet Management

**Proyecto:** PepsiCo Fleet Management System

**Versión:** 1.0.0

**Fecha:** Octubre 15, 2024

**Equipo:** Joaquín Marín & Benjamin Vilches

---

## 📋 Índice

1. [Diagrama de Clases](#diagrama-de-clases)
2. [Diagrama de Base de Datos](#diagrama-de-base-de-datos)
3. [Diagrama de Flujo de Información](#diagrama-de-flujo-de-información)
4. [Diagrama de Secuencia](#diagrama-de-secuencia)
5. [Diagrama de Componentes](#diagrama-de-componentes)
6. [Diagrama de Despliegue](#diagrama-de-despliegue)

---

## 1. Diagrama de Clases

### 1.1 Clases del Backend

classDiagram  
 class User {  
 +String id  
 +String rut  
 +String firstName  
 +String lastName  
 +String email  
 +String password  
 +String phone  
 +String roleId  
 +String workshopId  
 +Boolean isActive  
 +DateTime lastLogin  
 +DateTime createdAt  
 +DateTime updatedAt  
 +login()  
 +logout()  
 +updateProfile()  
 }  
  
 class Role {  
 +String id  
 +String name  
 +String description  
 +DateTime createdAt  
 +DateTime updatedAt  
 +addPermission()  
 +removePermission()  
 }  
  
 class Permission {  
 +String id  
 +String resource  
 +String action  
 +String description  
 +DateTime createdAt  
 }  
  
 class Vehicle {  
 +String id  
 +String licensePlate  
 +String vehicleType  
 +String brand  
 +String model  
 +Int year  
 +String vin  
 +String fleetNumber  
 +String regionId  
 +String status  
 +Boolean isActive  
 +DateTime createdAt  
 +DateTime updatedAt  
 +create()  
 +update()  
 +delete()  
 }  
  
 class VehicleEntry {  
 +String id  
 +String entryCode  
 +String vehicleId  
 +String workshopId  
 +String driverRut  
 +String driverName  
 +String driverPhone  
 +DateTime entryDate  
 +DateTime exitDate  
 +Int entryKm  
 +Int exitKm  
 +String fuelLevel  
 +Boolean hasKeys  
 +String observations  
 +JSON photos  
 +String status  
 +String createdById  
 +DateTime createdAt  
 +DateTime updatedAt  
 +create()  
 +update()  
 +exit()  
 }  
  
 class WorkOrder {  
 +String id  
 +String orderNumber  
 +String vehicleId  
 +String entryId  
 +String workshopId  
 +String workType  
 +String priority  
 +String description  
 +Float estimatedHours  
 +String assignedToId  
 +String currentStatus  
 +DateTime startedAt  
 +DateTime completedAt  
 +Float totalHours  
 +String observations  
 +String createdById  
 +DateTime createdAt  
 +DateTime updatedAt  
 +create()  
 +update()  
 +assign()  
 +start()  
 +complete()  
 }  
  
 class SparePart {  
 +String id  
 +String code  
 +String name  
 +String description  
 +String category  
 +String unitOfMeasure  
 +Float unitPrice  
 +Int currentStock  
 +Int minStock  
 +Int maxStock  
 +String location  
 +Boolean isActive  
 +DateTime createdAt  
 +DateTime updatedAt  
 +adjustStock()  
 +request()  
 +deliver()  
 }  
  
 class Workshop {  
 +String id  
 +String code  
 +String name  
 +String regionId  
 +String address  
 +String city  
 +String phone  
 +Int capacity  
 +Boolean isActive  
 +DateTime createdAt  
 +DateTime updatedAt  
 +getStats()  
 +getSchedule()  
 }  
  
 class Region {  
 +String id  
 +String code  
 +String name  
 +Boolean isActive  
 +DateTime createdAt  
 }  
  
 class AuditLog {  
 +String id  
 +String userId  
 +String action  
 +String resource  
 +String resourceId  
 +JSON details  
 +String ipAddress  
 +String userAgent  
 +DateTime createdAt  
 +log()  
 }  
  
 class Notification {  
 +String id  
 +String userId  
 +String title  
 +String message  
 +String type  
 +String relatedTo  
 +String relatedId  
 +Boolean isRead  
 +DateTime readAt  
 +DateTime createdAt  
 +send()  
 +markAsRead()  
 }  
  
 %% Relaciones  
 User ||--o{ VehicleEntry : creates  
 User ||--o{ WorkOrder : creates  
 User ||--o{ WorkOrder : assigned  
 User ||--o{ AuditLog : generates  
 User ||--o{ Notification : receives  
 User }o--|| Role : has  
 User }o--o| Workshop : belongs  
  
 Role ||--o{ Permission : has  
  
 Vehicle ||--o{ VehicleEntry : has  
 Vehicle ||--o{ WorkOrder : has  
 Vehicle }o--|| Region : belongs  
  
 VehicleEntry ||--o{ WorkOrder : generates  
 VehicleEntry ||--|| KeyControl : has  
  
 WorkOrder ||--o{ WorkOrderStatus : has  
 WorkOrder ||--o{ WorkOrderPhoto : has  
 WorkOrder ||--o{ WorkPause : has  
 WorkOrder ||--o{ WorkOrderSparePart : has  
  
 SparePart ||--o{ WorkOrderSparePart : used  
 SparePart ||--o{ SparePartMovement : has  
  
 Workshop ||--o{ User : contains  
 Workshop ||--o{ VehicleEntry : receives  
 Workshop ||--o{ WorkOrder : processes  
 Workshop ||--o{ WorkshopSchedule : has  
 Workshop }o--|| Region : belongs  
  
 Region ||--o{ Workshop : contains  
 Region ||--o{ Vehicle : contains

### 1.2 Clases del Frontend

classDiagram  
 class AuthStore {  
 +User user  
 +Boolean isAuthenticated  
 +Boolean isLoading  
 +String error  
 +login(email, password)  
 +logout()  
 +getCurrentUser()  
 +refreshToken()  
 }  
  
 class ApiService {  
 +String baseURL  
 +Object headers  
 +get(url, params)  
 +post(url, data)  
 +put(url, data)  
 +delete(url)  
 +setAuthToken(token)  
 +handleResponse(response)  
 +handleError(error)  
 }  
  
 class AuthService {  
 +login(credentials)  
 +logout()  
 +refreshToken()  
 +getCurrentUser()  
 +changePassword(data)  
 }  
  
 class VehicleService {  
 +getAll(params)  
 +getById(id)  
 +create(data)  
 +update(id, data)  
 +delete(id)  
 +getStats()  
 }  
  
 class WorkOrderService {  
 +getAll(params)  
 +getById(id)  
 +create(data)  
 +update(id, data)  
 +assign(id, mechanicId)  
 +start(id)  
 +complete(id)  
 +pause(id, reason)  
 +resume(id)  
 }  
  
 class DashboardService {  
 +getStats(period)  
 +getActivity()  
 +getMechanicsPerformance()  
 }  
  
 class MainLayout {  
 +User user  
 +Array navigation  
 +Boolean sidebarOpen  
 +toggleSidebar()  
 +logout()  
 +render()  
 }  
  
 class AdminDashboard {  
 +Object stats  
 +Array recentActivity  
 +Array urgentOrders  
 +Array lowStock  
 +loadData()  
 +render()  
 }  
  
 class GuardiaDashboard {  
 +Array activeEntries  
 +Array recentEntries  
 +createEntry()  
 +registerExit()  
 +searchVehicle()  
 +render()  
 }  
  
 class RecepcionistaDashboard {  
 +Array pendingEntries  
 +Array availableMechanics  
 +createWorkOrder()  
 +assignMechanic()  
 +render()  
 }  
  
 class MecanicoDashboard {  
 +Array myOrders  
 +Array pendingOrders  
 +startWork()  
 +pauseWork()  
 +completeWork()  
 +requestSparePart()  
 +render()  
 }  
  
 class JefeTallerDashboard {  
 +Object workshopStats  
 +Array allOrders  
 +Array mechanics  
 +Array alerts  
 +assignOrder()  
 +superviseWork()  
 +render()  
 }  
  
 class InventarioDashboard {  
 +Array lowStock  
 +Array requests  
 +Array movements  
 +deliverSparePart()  
 +adjustStock()  
 +render()  
 }  
  
 %% Relaciones  
 AuthStore ||--|| ApiService : uses  
 ApiService ||--|| AuthService : provides  
 ApiService ||--|| VehicleService : provides  
 ApiService ||--|| WorkOrderService : provides  
 ApiService ||--|| DashboardService : provides  
  
 MainLayout ||--|| AuthStore : uses  
 MainLayout ||--o{ AdminDashboard : renders  
 MainLayout ||--o{ GuardiaDashboard : renders  
 MainLayout ||--o{ RecepcionistaDashboard : renders  
 MainLayout ||--o{ MecanicoDashboard : renders  
 MainLayout ||--o{ JefeTallerDashboard : renders  
 MainLayout ||--o{ InventarioDashboard : renders  
  
 AdminDashboard ||--|| DashboardService : uses  
 GuardiaDashboard ||--|| VehicleService : uses  
 RecepcionistaDashboard ||--|| WorkOrderService : uses  
 MecanicoDashboard ||--|| WorkOrderService : uses  
 JefeTallerDashboard ||--|| DashboardService : uses  
 InventarioDashboard ||--|| VehicleService : uses

---

## 2. Diagrama de Base de Datos

### 2.1 Modelo Entidad-Relación

erDiagram  
 USERS {  
 uuid id PK  
 string rut UK  
 string first\_name  
 string last\_name  
 string email UK  
 string password  
 string phone  
 uuid role\_id FK  
 uuid workshop\_id FK  
 boolean is\_active  
 timestamp last\_login  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 ROLES {  
 uuid id PK  
 string name UK  
 string description  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 PERMISSIONS {  
 uuid id PK  
 string resource  
 string action  
 string description  
 timestamp created\_at  
 }  
  
 ROLE\_PERMISSIONS {  
 uuid role\_id PK,FK  
 uuid permission\_id PK,FK  
 timestamp created\_at  
 }  
  
 VEHICLES {  
 uuid id PK  
 string license\_plate UK  
 string vehicle\_type  
 string brand  
 string model  
 integer year  
 string vin UK  
 string fleet\_number UK  
 uuid region\_id FK  
 string status  
 boolean is\_active  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 VEHICLE\_ENTRIES {  
 uuid id PK  
 string entry\_code UK  
 uuid vehicle\_id FK  
 uuid workshop\_id FK  
 string driver\_rut  
 string driver\_name  
 string driver\_phone  
 timestamp entry\_date  
 timestamp exit\_date  
 integer entry\_km  
 integer exit\_km  
 string fuel\_level  
 boolean has\_keys  
 text observations  
 json photos  
 string status  
 uuid created\_by\_id FK  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 KEY\_CONTROL {  
 uuid id PK  
 uuid entry\_id FK,UK  
 string key\_location  
 string delivered\_to  
 timestamp delivered\_at  
 string returned\_by  
 timestamp returned\_at  
 text observations  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 WORK\_ORDERS {  
 uuid id PK  
 string order\_number UK  
 uuid vehicle\_id FK  
 uuid entry\_id FK  
 uuid workshop\_id FK  
 string work\_type  
 string priority  
 text description  
 float estimated\_hours  
 uuid assigned\_to\_id FK  
 string current\_status  
 timestamp started\_at  
 timestamp completed\_at  
 float total\_hours  
 text observations  
 uuid created\_by\_id FK  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 WORK\_ORDER\_STATUSES {  
 uuid id PK  
 uuid work\_order\_id FK  
 string status  
 text observations  
 string changed\_by\_id  
 timestamp changed\_at  
 }  
  
 WORK\_ORDER\_PHOTOS {  
 uuid id PK  
 uuid work\_order\_id FK  
 string url  
 string description  
 string photo\_type  
 timestamp uploaded\_at  
 }  
  
 WORK\_PAUSES {  
 uuid id PK  
 uuid work\_order\_id FK  
 string reason  
 timestamp paused\_at  
 timestamp resumed\_at  
 integer duration  
 text observations  
 }  
  
 SPARE\_PARTS {  
 uuid id PK  
 string code UK  
 string name  
 text description  
 string category  
 string unit\_of\_measure  
 float unit\_price  
 integer current\_stock  
 integer min\_stock  
 integer max\_stock  
 string location  
 boolean is\_active  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 WORK\_ORDER\_SPARE\_PARTS {  
 uuid id PK  
 uuid work\_order\_id FK  
 uuid spare\_part\_id FK  
 integer quantity\_requested  
 integer quantity\_delivered  
 string status  
 timestamp requested\_at  
 timestamp delivered\_at  
 text observations  
 }  
  
 SPARE\_PART\_MOVEMENTS {  
 uuid id PK  
 uuid spare\_part\_id FK  
 string movement\_type  
 integer quantity  
 integer previous\_stock  
 integer new\_stock  
 string reason  
 string reference  
 timestamp created\_at  
 }  
  
 REGIONS {  
 uuid id PK  
 string code UK  
 string name  
 boolean is\_active  
 timestamp created\_at  
 }  
  
 WORKSHOPS {  
 uuid id PK  
 string code UK  
 string name  
 uuid region\_id FK  
 string address  
 string city  
 string phone  
 integer capacity  
 boolean is\_active  
 timestamp created\_at  
 timestamp updated\_at  
 }  
  
 WORKSHOP\_SCHEDULES {  
 uuid id PK  
 uuid workshop\_id FK  
 integer day\_of\_week  
 string open\_time  
 string close\_time  
 boolean is\_active  
 }  
  
 AUDIT\_LOGS {  
 uuid id PK  
 uuid user\_id FK  
 string action  
 string resource  
 string resource\_id  
 json details  
 string ip\_address  
 string user\_agent  
 timestamp created\_at  
 }  
  
 NOTIFICATIONS {  
 uuid id PK  
 uuid user\_id FK  
 string title  
 text message  
 string type  
 string related\_to  
 string related\_id  
 boolean is\_read  
 timestamp read\_at  
 timestamp created\_at  
 }  
  
 DOCUMENTS {  
 uuid id PK  
 string name  
 string type  
 string url  
 string related\_to  
 string related\_id  
 timestamp uploaded\_at  
 }  
  
 %% Relaciones  
 USERS ||--o{ ROLES : has  
 ROLES ||--o{ PERMISSIONS : has  
 USERS ||--o{ WORKSHOPS : belongs  
 USERS ||--o{ VEHICLE\_ENTRIES : creates  
 USERS ||--o{ WORK\_ORDERS : creates  
 USERS ||--o{ WORK\_ORDERS : assigned  
 USERS ||--o{ AUDIT\_LOGS : generates  
 USERS ||--o{ NOTIFICATIONS : receives  
  
 VEHICLES ||--o{ VEHICLE\_ENTRIES : has  
 VEHICLES ||--o{ WORK\_ORDERS : has  
 VEHICLES ||--o{ REGIONS : belongs  
  
 VEHICLE\_ENTRIES ||--o{ WORK\_ORDERS : generates  
 VEHICLE\_ENTRIES ||--|| KEY\_CONTROL : has  
 VEHICLE\_ENTRIES ||--o{ WORKSHOPS : received  
  
 WORK\_ORDERS ||--o{ WORK\_ORDER\_STATUSES : has  
 WORK\_ORDERS ||--o{ WORK\_ORDER\_PHOTOS : has  
 WORK\_ORDERS ||--o{ WORK\_PAUSES : has  
 WORK\_ORDERS ||--o{ WORK\_ORDER\_SPARE\_PARTS : has  
 WORK\_ORDERS ||--o{ WORKSHOPS : processed  
  
 SPARE\_PARTS ||--o{ WORK\_ORDER\_SPARE\_PARTS : used  
 SPARE\_PARTS ||--o{ SPARE\_PART\_MOVEMENTS : has  
  
 WORKSHOPS ||--o{ WORKSHOP\_SCHEDULES : has  
 WORKSHOPS ||--o{ REGIONS : belongs  
  
 REGIONS ||--o{ WORKSHOPS : contains  
 REGIONS ||--o{ VEHICLES : contains

### 2.2 Índices de Base de Datos

-- Índices para performance  
CREATE INDEX idx\_users\_email ON users(email);  
CREATE INDEX idx\_users\_rut ON users(rut);  
CREATE INDEX idx\_users\_role\_id ON users(role\_id);  
CREATE INDEX idx\_users\_workshop\_id ON users(workshop\_id);  
  
CREATE INDEX idx\_vehicles\_license\_plate ON vehicles(license\_plate);  
CREATE INDEX idx\_vehicles\_region\_id ON vehicles(region\_id);  
CREATE INDEX idx\_vehicles\_status ON vehicles(status);  
  
CREATE INDEX idx\_entries\_vehicle\_id ON vehicle\_entries(vehicle\_id);  
CREATE INDEX idx\_entries\_workshop\_id ON vehicle\_entries(workshop\_id);  
CREATE INDEX idx\_entries\_entry\_date ON vehicle\_entries(entry\_date);  
CREATE INDEX idx\_entries\_status ON vehicle\_entries(status);  
  
CREATE INDEX idx\_work\_orders\_vehicle\_id ON work\_orders(vehicle\_id);  
CREATE INDEX idx\_work\_orders\_entry\_id ON work\_orders(entry\_id);  
CREATE INDEX idx\_work\_orders\_workshop\_id ON work\_orders(workshop\_id);  
CREATE INDEX idx\_work\_orders\_assigned\_to\_id ON work\_orders(assigned\_to\_id);  
CREATE INDEX idx\_work\_orders\_current\_status ON work\_orders(current\_status);  
CREATE INDEX idx\_work\_orders\_created\_at ON work\_orders(created\_at);  
  
CREATE INDEX idx\_spare\_parts\_code ON spare\_parts(code);  
CREATE INDEX idx\_spare\_parts\_category ON spare\_parts(category);  
  
CREATE INDEX idx\_notifications\_user\_read ON notifications(user\_id, is\_read);  
CREATE INDEX idx\_notifications\_created\_at ON notifications(created\_at);  
  
CREATE INDEX idx\_audit\_user\_id ON audit\_logs(user\_id);  
CREATE INDEX idx\_audit\_resource ON audit\_logs(resource, resource\_id);  
CREATE INDEX idx\_audit\_created\_at ON audit\_logs(created\_at);

---

## 3. Diagrama de Flujo de Información

### 3.1 Flujo de Autenticación

flowchart TD  
 A[Usuario ingresa credenciales] --> B[Frontend valida formato]  
 B --> C[POST /api/auth/login]  
 C --> D[Backend valida credenciales]  
 D --> E{¿Credenciales válidas?}  
 E -->|No| F[Retorna error 401]  
 E -->|Sí| G[Genera JWT tokens]  
 G --> H[Retorna access + refresh token]  
 H --> I[Frontend guarda tokens]  
 I --> J[Redirige a dashboard]  
   
 F --> K[Usuario ve mensaje de error]  
 K --> A

### 3.2 Flujo de Creación de Ingreso

flowchart TD  
 A[Guardia inicia ingreso] --> B[Selecciona vehículo]  
 B --> C[Ingresa datos del conductor]  
 C --> D[Verifica estado del vehículo]  
 D --> E[Registra nivel de combustible]  
 E --> F[Control de llaves]  
 F --> G[Sube fotos opcionales]  
 G --> H[Envía POST /api/vehicle-entries]  
 H --> I[Backend valida datos]  
 I --> J{¿Datos válidos?}  
 J -->|No| K[Retorna errores de validación]  
 J -->|Sí| L[Inicia transacción]  
 L --> M[Crea registro de ingreso]  
 M --> N[Actualiza estado del vehículo]  
 N --> O[Registra control de llaves]  
 O --> P[Genera código de ingreso]  
 P --> Q[Envía notificación]  
 Q --> R[Retorna éxito]  
 R --> S[Frontend muestra confirmación]  
   
 K --> T[Usuario corrige errores]  
 T --> H

### 3.3 Flujo de Asignación de Orden

flowchart TD  
 A[Recepcionista crea orden] --> B[Selecciona vehículo]  
 B --> C[Define tipo de trabajo]  
 C --> D[Establece prioridad]  
 D --> E[Asigna mecánico]  
 E --> F[Envía POST /api/work-orders]  
 F --> G[Backend valida datos]  
 G --> H{¿Datos válidos?}  
 H -->|No| I[Retorna errores]  
 H -->|Sí| J[Inicia transacción]  
 J --> K[Crea orden de trabajo]  
 K --> L[Actualiza estado del vehículo]  
 L --> M[Envía notificación al mecánico]  
 M --> N[Registra en auditoría]  
 N --> O[Retorna éxito]  
 O --> P[Frontend actualiza lista]  
   
 I --> Q[Usuario corrige errores]  
 Q --> F

### 3.4 Flujo de Gestión de Inventario

flowchart TD  
 A[Mecánico solicita repuesto] --> B[Selecciona repuesto]  
 B --> C[Especifica cantidad]  
 C --> D[Envía POST /api/spare-parts/request]  
 D --> E[Backend valida stock]  
 E --> F{¿Stock disponible?}  
 F -->|No| G[Retorna error de stock]  
 F -->|Sí| H[Crea solicitud]  
 H --> I[Notifica a inventario]  
 I --> J[Inventario prepara repuesto]  
 J --> K[Entrega repuesto]  
 K --> L[Actualiza stock]  
 L --> M[Notifica al mecánico]  
 M --> N[Registra movimiento]  
 N --> O[Actualiza estado de orden]  
   
 G --> P[Usuario ve mensaje de stock]  
 P --> Q[Selecciona repuesto alternativo]  
 Q --> B

---

## 4. Diagrama de Secuencia

### 4.1 Secuencia de Login

sequenceDiagram  
 participant U as Usuario  
 participant F as Frontend  
 participant B as Backend  
 participant DB as Database  
   
 U->>F: Ingresa email y password  
 F->>F: Valida formato de datos  
 F->>B: POST /api/auth/login  
 B->>B: Valida credenciales  
 B->>DB: SELECT user WHERE email  
 DB-->>B: Datos del usuario  
 B->>B: Verifica password con bcrypt  
 B->>B: Genera JWT tokens  
 B->>DB: UPDATE last\_login  
 B-->>F: { accessToken, refreshToken, user }  
 F->>F: Guarda tokens en localStorage  
 F->>F: Redirige a dashboard  
 F-->>U: Muestra dashboard

### 4.2 Secuencia de Creación de Orden

sequenceDiagram  
 participant R as Recepcionista  
 participant F as Frontend  
 participant B as Backend  
 participant DB as Database  
 participant M as Mecánico  
   
 R->>F: Crea nueva orden  
 F->>F: Valida formulario  
 F->>B: POST /api/work-orders  
 B->>B: Valida permisos RBAC  
 B->>B: Valida datos de entrada  
 B->>DB: BEGIN TRANSACTION  
 B->>DB: INSERT work\_order  
 B->>DB: UPDATE vehicle status  
 B->>DB: INSERT work\_order\_status  
 B->>DB: COMMIT TRANSACTION  
 B->>B: Envía notificación  
 B->>M: Notificación push/email  
 B-->>F: { success: true, order }  
 F->>F: Actualiza lista de órdenes  
 F-->>R: Muestra confirmación

### 4.3 Secuencia de Asignación de Mecánico

sequenceDiagram  
 participant JT as Jefe Taller  
 participant F as Frontend  
 participant B as Backend  
 participant DB as Database  
 participant M as Mecánico  
   
 JT->>F: Asigna mecánico a orden  
 F->>B: PUT /api/work-orders/:id/assign  
 B->>B: Valida permisos  
 B->>DB: SELECT work\_order  
 B->>DB: SELECT mechanic availability  
 B->>DB: UPDATE work\_order assigned\_to  
 B->>DB: INSERT work\_order\_status  
 B->>B: Envía notificación  
 B->>M: Notificación de asignación  
 B-->>F: { success: true }  
 F->>F: Actualiza estado visual  
 F-->>JT: Confirma asignación

---

## 5. Diagrama de Componentes

### 5.1 Arquitectura de Componentes

graph TB  
 subgraph "Frontend Layer"  
 A[React App]  
 B[Auth Store]  
 C[API Service]  
 D[Dashboard Components]  
 E[Form Components]  
 F[Layout Components]  
 end  
   
 subgraph "API Layer"  
 G[Express Server]  
 H[Auth Middleware]  
 I[RBAC Middleware]  
 J[Validation Middleware]  
 K[Audit Middleware]  
 end  
   
 subgraph "Business Layer"  
 L[Auth Service]  
 M[User Service]  
 N[Vehicle Service]  
 O[WorkOrder Service]  
 P[Inventory Service]  
 Q[Dashboard Service]  
 end  
   
 subgraph "Data Layer"  
 R[Prisma ORM]  
 S[Database Client]  
 T[Query Builder]  
 end  
   
 subgraph "Database Layer"  
 U[PostgreSQL]  
 V[Tables]  
 W[Indexes]  
 X[Constraints]  
 end  
   
 A --> B  
 A --> C  
 A --> D  
 A --> E  
 A --> F  
   
 C --> G  
 G --> H  
 G --> I  
 G --> J  
 G --> K  
   
 H --> L  
 I --> M  
 J --> N  
 K --> O  
 L --> P  
 M --> Q  
   
 L --> R  
 M --> R  
 N --> R  
 O --> R  
 P --> R  
 Q --> R  
   
 R --> S  
 S --> T  
 T --> U  
   
 U --> V  
 U --> W  
 U --> X

### 5.2 Componentes del Frontend

graph TB  
 subgraph "App Component"  
 A[App.tsx]  
 B[Router]  
 C[AuthProvider]  
 end  
   
 subgraph "Layout Components"  
 D[MainLayout]  
 E[Sidebar]  
 F[Header]  
 G[Footer]  
 end  
   
 subgraph "Page Components"  
 H[LoginPage]  
 I[AdminDashboard]  
 J[GuardiaDashboard]  
 K[RecepcionistaDashboard]  
 L[MecanicoDashboard]  
 M[JefeTallerDashboard]  
 N[InventarioDashboard]  
 end  
   
 subgraph "Shared Components"  
 O[Card]  
 P[Button]  
 Q[Badge]  
 R[Modal]  
 S[Form]  
 T[Table]  
 end  
   
 subgraph "Service Components"  
 U[AuthService]  
 V[VehicleService]  
 W[WorkOrderService]  
 X[InventoryService]  
 Y[DashboardService]  
 end  
   
 A --> B  
 B --> C  
 C --> D  
   
 D --> E  
 D --> F  
 D --> G  
   
 B --> H  
 B --> I  
 B --> J  
 B --> K  
 B --> L  
 B --> M  
 B --> N  
   
 I --> O  
 J --> P  
 K --> Q  
 L --> R  
 M --> S  
 N --> T  
   
 H --> U  
 I --> V  
 J --> W  
 K --> X  
 L --> Y  
 M --> U  
 N --> V

---

## 6. Diagrama de Despliegue

### 6.1 Arquitectura de Despliegue

graph TB  
 subgraph "Internet"  
 A[Usuarios]  
 B[CDN]  
 end  
   
 subgraph "Frontend Hosting"  
 C[Vercel/Netlify]  
 D[Static Files]  
 E[Build Process]  
 end  
   
 subgraph "Backend Hosting"  
 F[Railway/Render]  
 G[Node.js App]  
 H[PM2 Process Manager]  
 end  
   
 subgraph "Database Hosting"  
 I[Neon/Supabase]  
 J[PostgreSQL]  
 K[Connection Pool]  
 end  
   
 subgraph "External Services"  
 L[Email Service]  
 M[File Storage]  
 N[Monitoring]  
 end  
   
 A --> B  
 B --> C  
 C --> D  
 C --> E  
   
 A --> F  
 F --> G  
 G --> H  
   
 G --> I  
 I --> J  
 J --> K  
   
 G --> L  
 G --> M  
 G --> N

### 6.2 Configuración de Ambientes

graph LR  
 subgraph "Development"  
 A[Local Frontend :5173]  
 B[Local Backend :3000]  
 C[Local PostgreSQL :5432]  
 end  
   
 subgraph "Staging"  
 D[Staging Frontend]  
 E[Staging Backend]  
 F[Staging Database]  
 end  
   
 subgraph "Production"  
 G[Production Frontend]  
 H[Production Backend]  
 I[Production Database]  
 end  
   
 A --> B  
 B --> C  
   
 D --> E  
 E --> F  
   
 G --> H  
 H --> I

---

## 7. Diagramas de Casos de Uso

### 7.1 Casos de Uso por Rol

graph TB  
 subgraph "Administrador"  
 A1[Gestionar Usuarios]  
 A2[Gestionar Roles]  
 A3[Ver Reportes]  
 A4[Configurar Sistema]  
 end  
   
 subgraph "Guardia"  
 B1[Registrar Ingreso]  
 B2[Registrar Salida]  
 B3[Buscar Vehículo]  
 B4[Control de Llaves]  
 end  
   
 subgraph "Recepcionista"  
 C1[Crear Orden]  
 C2[Asignar Mecánico]  
 C3[Gestionar Ingresos]  
 C4[Ver Estado Órdenes]  
 end  
   
 subgraph "Mecánico"  
 D1[Ver Mis Órdenes]  
 D2[Iniciar Trabajo]  
 D3[Pausar Trabajo]  
 D4[Solicitar Repuestos]  
 D5[Completar Trabajo]  
 end  
   
 subgraph "Jefe de Taller"  
 E1[Supervisar Taller]  
 E2[Asignar Órdenes]  
 E3[Ver Rendimiento]  
 E4[Gestionar Recursos]  
 end  
   
 subgraph "Encargado Inventario"  
 F1[Gestionar Stock]  
 F2[Entregar Repuestos]  
 F3[Ver Alertas]  
 F4[Ajustar Inventario]  
 end

### 7.2 Flujo de Casos de Uso

graph TD  
 A[Usuario accede al sistema] --> B{¿Autenticado?}  
 B -->|No| C[Mostrar login]  
 B -->|Sí| D[Verificar rol]  
   
 C --> E[Ingresar credenciales]  
 E --> F[Validar credenciales]  
 F --> G{¿Válidas?}  
 G -->|No| C  
 G -->|Sí| D  
   
 D --> H{¿Rol?}  
 H -->|Admin| I[Dashboard Admin]  
 H -->|Guardia| J[Dashboard Guardia]  
 H -->|Recepcionista| K[Dashboard Recepcionista]  
 H -->|Mecánico| L[Dashboard Mecánico]  
 H -->|Jefe Taller| M[Dashboard Jefe Taller]  
 H -->|Inventario| N[Dashboard Inventario]  
   
 I --> O[Gestionar sistema]  
 J --> P[Control vehicular]  
 K --> Q[Gestionar órdenes]  
 L --> R[Trabajar en órdenes]  
 M --> S[Supervisar taller]  
 N --> T[Gestionar inventario]

---

## 8. Diagramas de Estado

### 8.1 Estados de Órdenes de Trabajo

stateDiagram-v2  
 [\*] --> Pendiente : Crear orden  
   
 Pendiente --> Asignada : Asignar mecánico  
 Pendiente --> Cancelada : Cancelar orden  
   
 Asignada --> EnProgreso : Iniciar trabajo  
 Asignada --> Pendiente : Reasignar  
   
 EnProgreso --> Pausada : Pausar trabajo  
 EnProgreso --> Completada : Completar trabajo  
   
 Pausada --> EnProgreso : Reanudar trabajo  
 Pausada --> Completada : Completar trabajo  
   
 Completada --> [\*] : Finalizar  
   
 Cancelada --> [\*] : Finalizar

### 8.2 Estados de Vehículos

stateDiagram-v2  
 [\*] --> Activo : Registrar vehículo  
   
 Activo --> EnMantenimiento : Ingresar a taller  
 Activo --> Inactivo : Desactivar  
   
 EnMantenimiento --> Activo : Salir del taller  
 EnMantenimiento --> Inactivo : Desactivar  
   
 Inactivo --> Activo : Reactivar  
   
 Activo --> [\*] : Eliminar  
 Inactivo --> [\*] : Eliminar

---

## 9. Diagramas de Red

### 9.1 Topología de Red

graph TB  
 subgraph "Internet"  
 A[Usuarios]  
 B[CDN]  
 end  
   
 subgraph "Load Balancer"  
 C[Cloudflare]  
 end  
   
 subgraph "Application Layer"  
 D[Frontend Server]  
 E[Backend Server 1]  
 F[Backend Server 2]  
 end  
   
 subgraph "Database Layer"  
 G[Primary Database]  
 H[Replica Database]  
 end  
   
 subgraph "Storage Layer"  
 I[File Storage]  
 J[Backup Storage]  
 end  
   
 A --> B  
 B --> C  
 C --> D  
 C --> E  
 C --> F  
   
 E --> G  
 F --> G  
 G --> H  
   
 E --> I  
 F --> I  
 G --> J

### 9.2 Flujo de Datos en Red

sequenceDiagram  
 participant U as Usuario  
 participant CDN as CDN  
 participant LB as Load Balancer  
 participant F as Frontend  
 participant B as Backend  
 participant DB as Database  
   
 U->>CDN: Request  
 CDN->>LB: Forward request  
 LB->>F: Route to frontend  
 F->>B: API call  
 B->>DB: Query  
 DB-->>B: Response  
 B-->>F: API response  
 F-->>LB: Frontend response  
 LB-->>CDN: Forward response  
 CDN-->>U: Final response

---

## 10. Diagramas de Performance

### 10.1 Métricas de Performance

graph TB  
 subgraph "Frontend Metrics"  
 A[First Contentful Paint < 1s]  
 B[Largest Contentful Paint < 2s]  
 C[Cumulative Layout Shift < 0.1]  
 D[First Input Delay < 100ms]  
 end  
   
 subgraph "Backend Metrics"  
 E[Response Time < 200ms]  
 F[Throughput > 1000 req/s]  
 G[Error Rate < 1%]  
 H[Uptime > 99.5%]  
 end  
   
 subgraph "Database Metrics"  
 I[Query Time < 50ms]  
 J[Connection Pool > 20]  
 K[Index Usage > 95%]  
 L[Cache Hit Rate > 80%]  
 end

### 10.2 Optimizaciones Implementadas

graph TB  
 subgraph "Frontend Optimizations"  
 A[Code Splitting]  
 B[Lazy Loading]  
 C[Image Optimization]  
 D[Bundle Compression]  
 end  
   
 subgraph "Backend Optimizations"  
 E[Database Indexing]  
 F[Query Optimization]  
 G[Response Caching]  
 H[Connection Pooling]  
 end  
   
 subgraph "Database Optimizations"  
 I[Index Strategy]  
 J[Query Analysis]  
 K[Partitioning]  
 L[Replication]  
 end

---

**Última actualización:** Octubre 15, 2024

**Versión:** 1.0.0

**Mantenido por:** Joaquín Marín & Benjamin Vilches