Arquitectura proyecto "Automatización y control de estacionamientos"

Target release	V 1.0.3 29-03-25
Preview release	V 1.0.2 28-03-25
Epic	Portfolio Analista
Document status	PUBLISH
Document owner	Instituto Duoc Uc
Designer	XForce Coders
Tech lead	David Nova, Juan Espinoza, Gonzalo Honorato
Technical writers	Gonzalo Honorato

Objetivo

Este documento describe la arquitectura del sistema de software "Automatización y Control de Estacionamientos". Tiene como objetivo alinear el diseño técnico con los objetivos funcionales y no funcionales del proyecto, facilitando su implementación, mantenimiento y escalabilidad.

Alcance del sistema

El sistema permite gestionar y automatizar el ingreso, salida y reserva de espacios de estacionamiento en instituciones educativas, mediante tecnologías como reconocimiento de patentes (OCR), códigos QR y una aplicación web progresiva (PWA).

Audiencia objetivo

Este documento está dirigido a desarrolladores, arquitectos de software, docentes evaluadores y colaboradores del equipo.

Definiciones y acrónimos

- **OCR**: Reconocimiento Óptico de Caracteres
- **PWA**: Progressive Web App
- **API**: Interfaz de Programación de Aplicaciones
- **Clean Architecture**: Patrón de arquitectura que separa responsabilidades

1. Visión general del sistema

El sistema está basado en una arquitectura cliente-servidor, donde:

- El **cliente** (PWA) permite a usuarios realizar reservas, acceder con QR, y a operadores monitorear el sistema.
- El **servidor** procesa autenticaciones, validación OCR, lógica de negocio y almacenamiento de datos.

El sistema incluye funcionalidades offline para mantener operatividad en caso de caída de red.

2. Estilo arquitectónico

Arquitectura Cliente-Servidor

Se adopta el estilo arquitectónico Cliente-Servidor debido a:

- Claridad en la separación de responsabilidades
- Escalabilidad horizontal del backend
- Posibilidad de uso en distintos dispositivos

3. Componentes del sistema

Cliente

- **Tecnología**: Vue 3, TypeScript, CSS
- **Funcionalidades**:
  - Registro y autenticación
  - Reserva de estacionamientos
  - Generación y escaneo de QR
  - Modo offline con SQLite local
  - Panel de control para administradores

Servidor

- **Tecnología**: Node.js o Go (a definir)
- **Funcionalidades**:
  - Autenticación con Google
  - Procesamiento de patentes (OCR)
  - APIs REST para funcionalidades del cliente
  - Almacenamiento en Google Cloud

Base de datos

- **Tecnología**: SQLite
- **Uso**: Almacenamiento de usuarios, reservas, registros de acceso, configuración

4. Diagramas arquitectónicos

Se incluirán:

- Diagrama de componentes (cliente, servidor, base de datos, OCR API, API institucional)
- Diagrama de despliegue (PWA, servidor local/VPS, base de datos)
- Diagrama de casos de uso

Diagrama de componentes

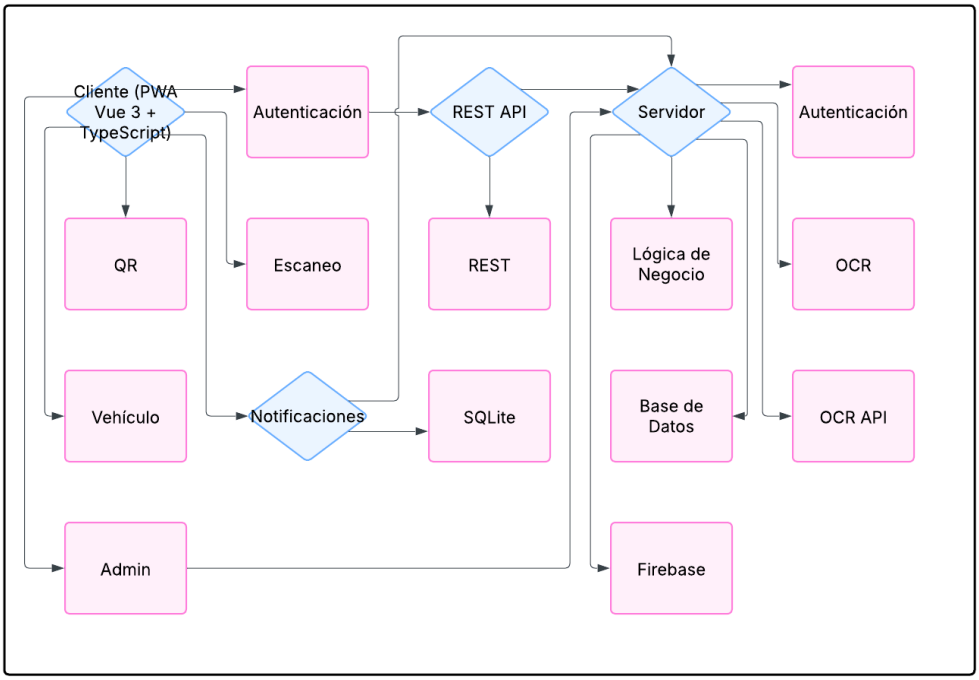
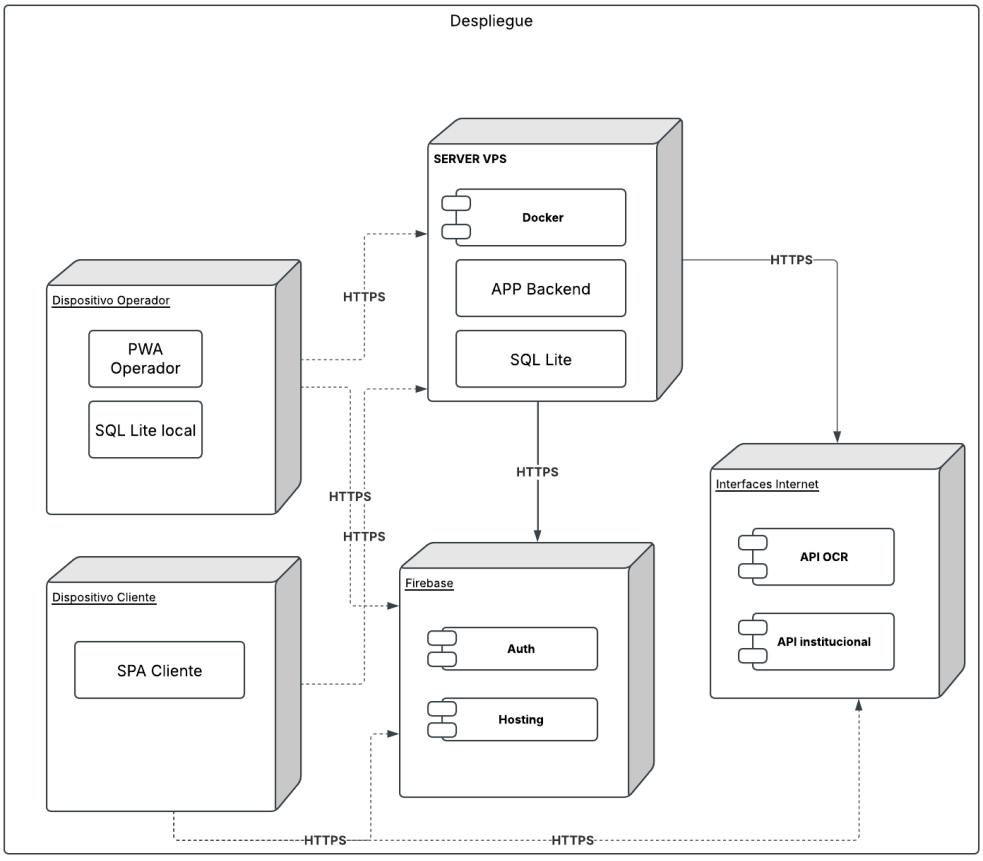
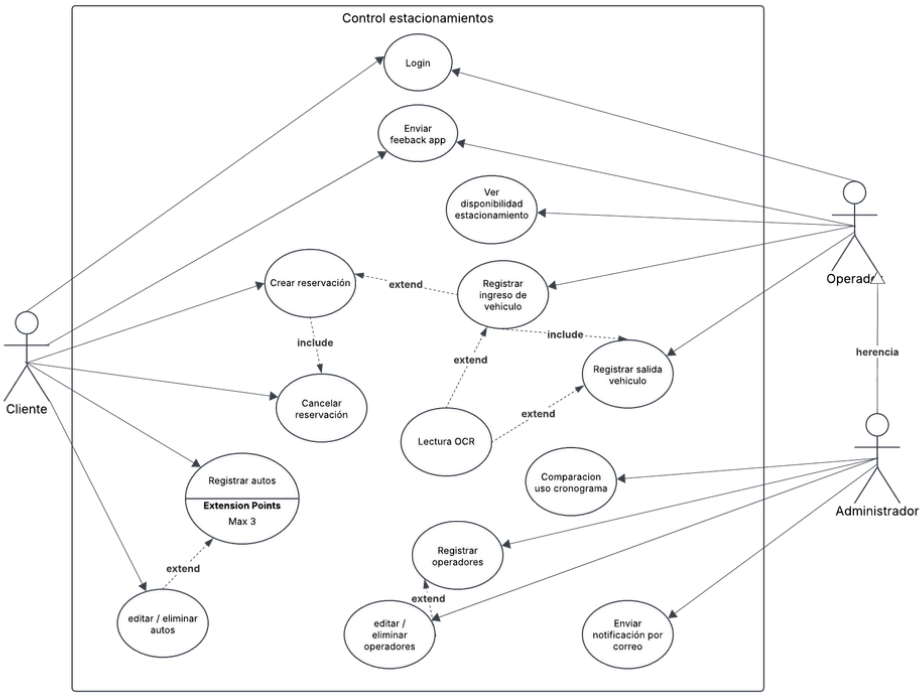


Diagrama de despliegue



Casos de uso



5. Interfaces

Interfaces externas

- **API institucional**: Consulta de cronograma académico (Simulada)
- **Google Authentication**
- **OCR API** (externa together.ai)

Interfaces internas

- **REST API** entre cliente y servidor (reservas, login, reportes)

6. Módulos principales

- **Módulo de reservas**: Gestión de espacios disponibles y QR
- **Módulo de autenticación**: Login Google, control de sesiones
- **Módulo OCR**: Validación de patentes
- **Módulo offline**: Operación local con SQLite en PWA
- **Módulo administrativo**: Dashboard, reportes, estado

7. Seguridad

- Autenticación firebase auth (Google)
- Validación de QR
- Acceso basado en roles (usuario, operador, administrador)
- Cifrado de tokens y almacenamiento seguro en cliente Json Web Tokens

8. Requisitos no funcionales

- **Disponibilidad**: Uso offline garantizado con PWA
- **Escalabilidad**: Posibilidad de migrar a base de datos centralizada
- **Rendimiento**: Procesamiento local para reservas, asincronía en OCR
- **Usabilidad**: Interfaz intuitiva, diseño responsivo
- **Mantenibilidad**: Código modular y basado en Clean Architecture

9. Patrones y principios

- Clean Architecture
- Programación orientada a objetos
- Programación orientada a eventos
- Integración continua con GitHub, mediante github actions para firebase hosting y functions.

10. Tecnologías y herramientas

- **Frontend**: Vue 3, TypeScript, CSS
- **Backend**: Node.js / Go
- **Base de datos**: SQLite
- **Servicios**: Google Auth, Cloud Storage
- **Herramientas**: GitHub, Jira, Figma, Docker

11. Despliegue

- Entorno local para pruebas
- VPS con Docker para servidor
- Simulación de entorno real en presentación final

Propiedad Intelectual y Responsabilidad