DSARQ – Descripción de la Arquitectura

# 1. Introducción

Este documento presenta la descripción de la arquitectura del Sistema de Estacionamiento Rotativo Tarifado (SERT) desarrollado para el cantón Puyo. Su propósito es establecer una visión técnica clara del sistema, definir los componentes principales, su interacción, y justificar las decisiones arquitectónicas tomadas durante el diseño del aplicativo móvil.

# 2. Visión general de la arquitectura

La arquitectura adoptada es del tipo Cliente-Servidor con un enfoque MVC (Modelo-Vista-Controlador) en el backend, y arquitectura modular para el frontend móvil. El sistema incluye una aplicación móvil desarrollada en Android, un servidor de aplicaciones basado en PHP, una base de datos MySQL y servicios integrados como la pasarela de pagos y sistema de notificaciones.

# 3. Componentes principales del sistema

- Aplicación móvil (cliente): desarrollada en Android Studio, utiliza geolocalización, notificaciones y conexión con APIs REST.  
- Servidor backend: implementado en PHP con patrón MVC, expone servicios mediante API RESTful.  
- Base de datos: MySQL, contiene información de usuarios, zonas, sesiones de parqueo, historial, pagos y sanciones.  
- Pasarela de pagos: servicio externo que permite realizar pagos electrónicos seguros (e.g., Datafast, PayPhone).  
- Panel web administrativo: interfaz web para configurar zonas tarifadas, gestionar usuarios y generar reportes.  
- Sistema de autenticación: manejo de sesiones mediante tokens JWT para seguridad en la app.

# 4. Tecnologías utilizadas

- Android Studio (Java/Kotlin) para el desarrollo móvil.  
- PHP 8 y Slim Framework para el backend.  
- MySQL como sistema gestor de bases de datos.  
- Firebase para notificaciones push.  
- Google Maps SDK para localización y visualización de zonas.  
- Git para control de versiones.  
- Postman para pruebas de APIs REST.

# 5. Interacción entre componentes

El flujo general de interacción es el siguiente:  
  
1. El usuario utiliza la aplicación móvil para iniciar sesión o registrarse.  
2. La app envía peticiones HTTP al backend mediante API REST (por ejemplo, crear sesión de parqueo).  
3. El backend valida la solicitud, consulta o actualiza datos en la base de datos.  
4. Si es necesario, se conecta con servicios externos como la pasarela de pagos o notificaciones.  
5. La respuesta es enviada nuevamente a la app móvil, que actualiza su interfaz en tiempo real.  
6. El panel administrativo accede a los mismos datos para monitoreo y generación de reportes.

# 6. Consideraciones de seguridad, rendimiento y escalabilidad

- Seguridad:  
 \* Autenticación basada en tokens JWT.  
 \* Cifrado de contraseñas usando bcrypt.  
 \* Validación de datos en cada punto de entrada.  
  
- Rendimiento:  
 \* Indexación de tablas críticas en la base de datos.  
 \* Uso de caché para zonas tarifadas.  
 \* Optimización de imágenes y datos en la app.  
  
- Escalabilidad:  
 \* Arquitectura desacoplada cliente-servidor.  
 \* Posibilidad de migrar a microservicios en futuras versiones.  
 \* Uso de contenedores Docker (en desarrollo).