Guia Tecnica de Aaron Vasquez

Nombre: **A**aron **J**affet **V**asquez **C**arrera

Asignatura: **P**rocess and **S**ervices **P**rogramming

Centro: **C.I. M**aria **A**na **S**anz

Ciclo: **2º**

Curso: **DAM**

Grupo: **B**

Fecha: 08/03/2025

Índice

1. Introducción

El presente documento detalla el diseño, desarrollo y configuración del proyecto Parking, un sistema distribuido compuesto por una API, un servidor y un cliente. Este proyecto tiene como finalidad gestionar y optimizar el control de acceso a los estacionamientos de manera eficiente y segura.

Se emplean técnicas de cifrado para proteger la comunicación entre los distintos componentes, garantizando la integridad y seguridad de los datos. A continuación, se detallarán los aspectos clave del sistema, su arquitectura y las decisiones técnicas que sustentan su desarrollo.

2. Arquitectura del Proyecto

El sistema está dividido en tres módulos principales:

- **ParkingAPI**: API REST encargada de gestionar la lógica de negocio y el almacenamiento de datos. Responde a solicitudes del servidor y proporciona la información necesaria para el correcto funcionamiento del sistema.
- **ParkingServer**: Servidor que actúa como intermediario entre el cliente y la API. Se encarga de gestionar las solicitudes, aplicar medidas de seguridad y garantizar una comunicación fluida y protegida.
- **ParkingClient**: Cliente que permite la interacción con el sistema, facilitando a los usuarios el acceso a la información del estacionamiento mediante una interfaz sencilla y eficiente.

3. Tecnologías Utilizadas

El desarrollo del sistema se ha basado en tecnologías modernas y ampliamente utilizadas en entornos empresariales:

- **Lenguaje de Programación:** C#
- **Framework:** .NET Core, utilizado para la construcción de la API y el servidor
- **Base de Datos:** Firebase, empleado para la persistencia de datos en tiempo real Y JSON empleado como base de datos de la API
- **Seguridad:** Implementación de cifrado RSA y generación de firmas digitales mediante SHA

4. Estructura del Código

El código del sistema está organizado en carpetas y archivos que facilitan su mantenimiento y escalabilidad. Cada módulo del sistema contiene los siguientes elementos:

- **ParkingAPI**
  - Controllers/ (Controladores que gestionan las solicitudes HTTP)
  - Models/ (Definición de las estructuras de datos utilizadas en el sistema)
  - Services/ (Implementación de la lógica de negocio y comunicación con la base de datos)
  - Program.cs (Archivo principal para la ejecución de la API)
  - Middleware/ (Implementación de filtros de seguridad y autenticación JWT)
- **ParkingServer**
  - Controllers/ (Gestión de la comunicación con el cliente y la API)
  - Security/ (RSAHelper y SHAHelper para la seguridad)
  - Program.cs (Punto de entrada del servidor)
- **ParkingClient**
  - Program.cs (Interfaz del cliente para interactuar con el sistema)
  - Security/ (Módulo encargado de cifrar las comunicaciones entre cliente y servidor)

5. Configuración y Despliegue

Configuración

Para la correcta ejecución del sistema, es necesario configurar los siguientes archivos:

- Archivos de claves RSA (privateKey.xml, publicKey.xml): Utilizados para el cifrado de las comunicaciones. Deben ser generados previamente y almacenados en ubicaciones seguras.

Despliegue

Para poner en marcha el sistema, se deben seguir los siguientes pasos:

1. Compilar y ejecutar ParkingAPI.
2. Iniciar ParkingServer.
3. Ejecutar ParkingClient para comenzar a interactuar con el sistema.

6. Seguridad y Cifrado

La seguridad es un aspecto fundamental del sistema. Para garantizar la protección de los datos, se han implementado los siguientes mecanismos:

- **RSAHelper.cs**: Utilizado para cifrar y descifrar mensajes mediante el algoritmo RSA.
- **SHAHelper.cs**: Implementa funciones hash para verificar la integridad de los datos intercambiados.
- **Autenticación y Claves:** Se utilizan claves públicas y privadas para el cifrado de la información, evitando accesos no autorizados.

7. Endpoints de la API

- **GET /api/usuarios**
  Devuelve una lista de todos los usuarios.

- **GET /api/usuarios/{id}**
  Devuelve un usuario específico según el ID proporcionado.

- **POST /api/usuarios**
  Crea un nuevo usuario enviando sus datos en el cuerpo de la solicitud.

- **POST /api/usuarios/bulk**
  Crea múltiples usuarios a partir de una lista de usuarios enviada en el cuerpo.

- **PUT /api/usuarios/{id}**
  Actualiza los datos de un usuario existente identificado por el ID en la URL.

- **PUT /api/usuarios/bulk**
  Actualiza múltiples usuarios enviando una lista con los datos de cada usuario.

- **DELETE /api/usuarios/{id}**
  Elimina un usuario específico identificado por el ID.

- **DELETE /api/usuarios/bulk**
  Elimina varios usuarios mediante una lista de IDs enviada en el cuerpo de la solicitud.

8. Flujo de Comunicación

1. El Usuario envía una solicitud cifrada al Servidor.
2. El Servidor descifra la solicitud y la cifra en SHA256.
3. El servidor consulta la API enviando la matrícula cifrada en SHA256 para verificar la información.
4. La API compara las matrículas guardadas en la base de datos en SHA256 con el SHA256 recibido del Servidor.
5. La API responde con los datos requeridos al Servidor (Acceso permitido o Acceso denegado).
6. El Servidor envía la respuesta al cliente.

9. Pruebas y Validaciones

Se han realizado pruebas unitarias para garantizar la calidad del sistema:

- **Pruebas de Integración:** Simulación de solicitudes a la API para validar el comportamiento del sistema, también simulación de solicitudes al Server para validar su comportamiento.
- **Pruebas de Seguridad:** Cifrado RSA y Cifrado AES.
- **Pruebas de Asincronía:** Puede usarlo más de un cliente a la vez

10. Conclusiones

El sistema Parking API & Sistema Cliente-Servidor ofrece una solución eficiente para el control de accesos, proporcionando un entorno seguro y optimizado.

11. Deudas Técnicas

- **Optimización del manejo de datos en Firebase**.
- **Automatización del despliegue** mediante contenedores Docker.
- **Mejoras en la autenticación** para incluir roles y permisos.
- **Interfaz Web** para hacerlo más atractivo e intuitivo.

Estas mejoras permitirán optimizar aún más el sistema y garantizar su escalabilidad en el tiempo.