

**DOCUMENTACIÓN TÉCNICA PARA EL DESARROLLO DE
MICROSERVICIO RESTFUL PARA LA ADMINISTRACIÓN DE
TELÉFONOS.**

Examen práctico

20-03-2025

Luigi Hernández García

Contenido

1. Requerimiento Examen práctico.	3
2. Diagrama de Entendimiento para resolver el examen práctico.....	5
3. Diagrama de Arquitectura.	6
4. Obtener Código Fuente.	6
5. Entorno de desarrollo.	6
5.1 Instalar Java.	7
5.2 Instalar GIT.....	7
5.3 Instalar Visual studio code.	8
5.4 Instalar Docker.	8
5.5 Instalar Postman	9
5.6 Instalar Maven.....	9
6. Como Descargar, Ejecutar y Probar el proyecto.	10
6.1 Pasos para Iniciar Docker.....	10
6.2 Descargar el proyecto de Github.....	11
6.3 Ejecutar el proyecto.	13
6.4 Probamos el proyecto.	13
7. Como obtener el header obligatorio DATA	15
8. Servicios y documentación de la API tipos de respuesta.	17
9. Validar y Probar los servicios en postman.....	20

1. Requerimiento Examen práctico.

El siguiente ejercicio pretende medir tu habilidad para desarrollar código usando java y Spring y las buenas prácticas que aplicas a tu trabajo diario, por lo que te sugerimos realizarlo como si fuera un entregable de tu día a día. Se considerará la simplicidad que puedas implementar pero que logres cumplir con los objetivos solicitados.

Diseñar un microservicio RESTFUL para que permita la **administración de Teléfonos**, la información del teléfono se define más adelante.

El microservicio debe permitir:

1. **Consultar** los datos de un teléfono
2. **Crear** los datos de un teléfono
3. **Eliminar** los datos de un teléfono
4. **Actualizar** los datos de un teléfono
5. Crear un método de **consulta por IMEI**

La base de datos puede ser Relacional o no SQL(mayores puntos, **NoSQL**).

El microservicio debe recibir un header obligatorio de nombre "DATA". Como valor debe recibir una cadena hashada con sha256, la cadena a hashear es "MIEXAMENPRUEBA".

Dentro del servicio se tiene que validar que en todas las peticiones venga el header "DATA", si no contiene el valor "MIEXAMENPRUEBA" se debe rechazar la petición.

Agregar una Cache con redis en la consulta del cliente por IMEI. Solo se tiene que ir a BD la primera vez o cuando se cambie un valor del teléfono consultado.

Resetear el Cache al hacer un update del registro

Creación de **Test unitarios Junit** (utilizar mock y Mockito)

Consulta de todos los registros de los clientes con paginación.

Cada vez que se inserte o se actualice un registro en la base de datos de mongo se tiene que enviar un mensaje a una cola de mensajes, indicando que se creó o modifico un objeto con su respectivo modificador, de preferencia RabbitMQ - **Kafka**

Implementar el consumo de los mensajes del rabbitMq o **Kafka** e insertarlos en la base de datos

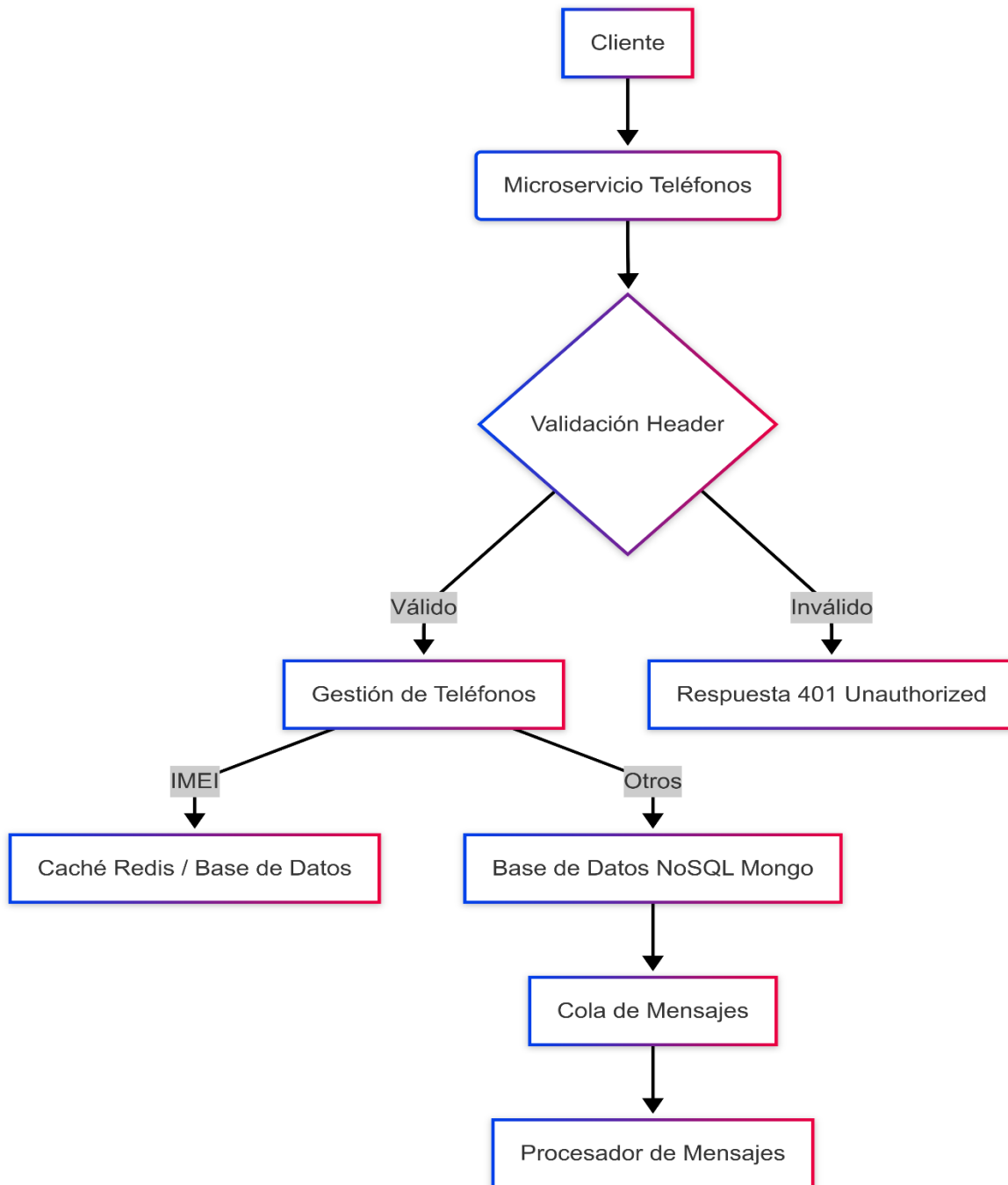
Los datos del teléfono son:

- Nombre (requerido, validar caracteres especiales)
- Marca (requerido, validar caracteres especiales)

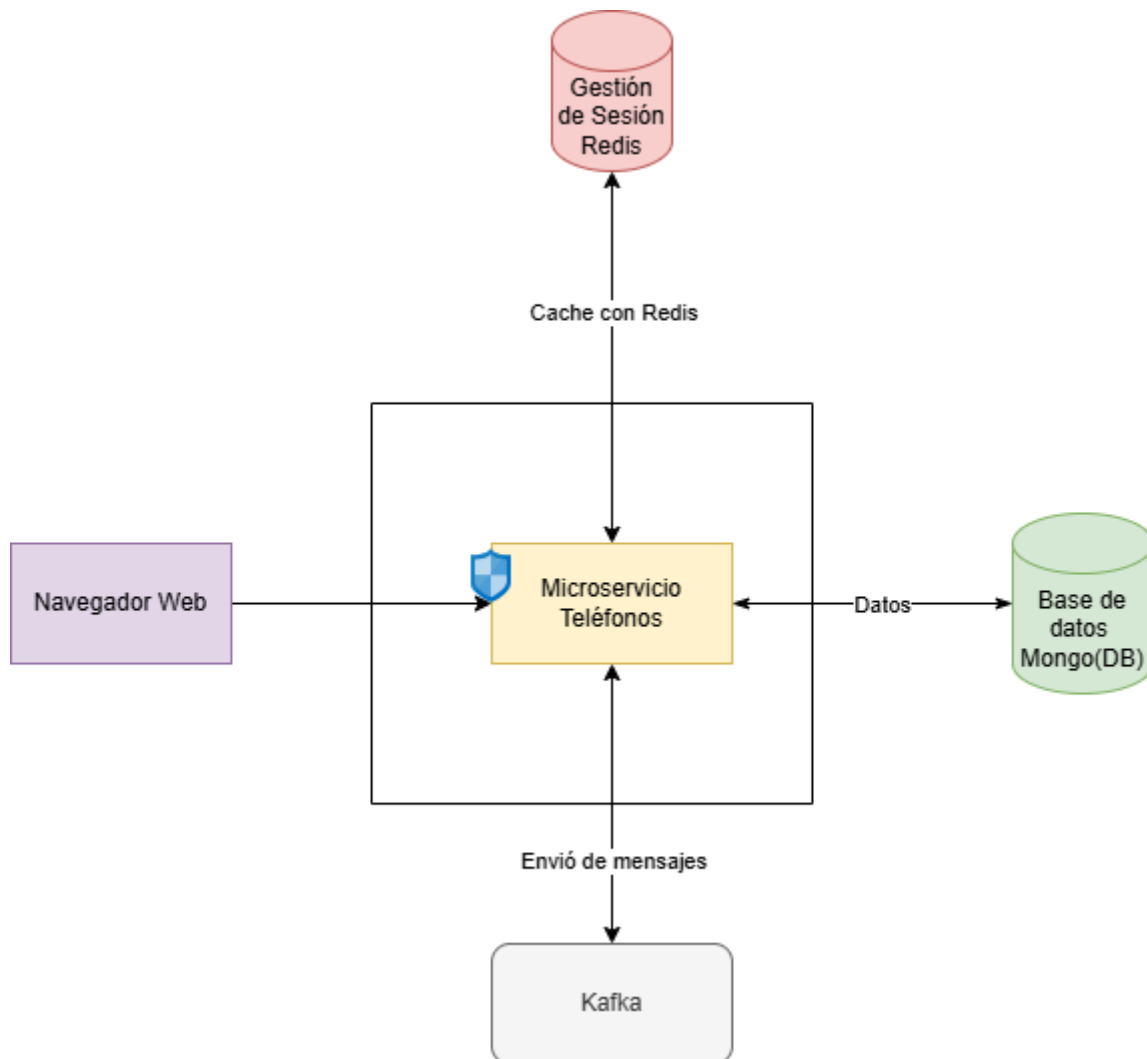


- Modelo (requerido, validar caracteres especiales)
- Nombre corto(requerido, validar caracteres especiales)
- Fecha de creación (requerido, validar caracteres especiales)
- IMEI (requerido y de valor único)
- Numero celular (opcional)
- Email Soporte(opcional, opcional si trae valor validar formato de Email)
- ¿Tiene sistema IOS? (Campo booleano)

2. Diagrama de Entendimiento para resolver el examen práctico.



3. Diagrama de Arquitectura.



4. Obtener Código Fuente.

El proyecto se encuentra en GITHUB

** Pre-condición tener instalado git ([ver sección 5.2](#))

git clone <https://github.com/luigihg/AdministrarTelefonos.git>

5. Entorno de desarrollo.

Instalar el siguiente software.

5.1 Instalar Java.

1. **Descargar el JDK (Version 17 o superior):**

Descarga un JDK (Java Development Kit) de Oracle o Adoptium (OpenJDK). Adoptium es una opción común y gratuita.

Aquí tienes el enlace directo a las descargas de Adoptium OpenJDK 17:

<https://adoptium.net/releases.html?variant=jdk&jvmVariant=hotspot&version=17>

2. **Instalar el JDK:**

Ejecuta el instalador y sigue las instrucciones.

3. **Configurar Variables de Entorno (Windows/Linux):**

Configura JAVA_HOME para que apunte a la instalación del JDK.

Agrega el directorio bin del JDK a la variable Path.

4. **Verificar la Instalación:**

Abre la línea de comandos y ejecuta `java -version` y `javac -version`

5.2 Instalar GIT

1. **Descargar Git:**

Descarga el instalador de la página oficial de Git.

La página principal para descargar Git es:

<https://git-scm.com/downloads>

2. **Instalar Git:**

Ejecuta el instalador y sigue las instrucciones.

3. **Verificar la Instalación:**

Abre la línea de comandos y escribe `git --version`.

4. **Configuración Inicial (Recomendado):**

Configura tu nombre de usuario y correo electrónico con los comandos `git config --global user.name "Tu Nombre"` y `git config --global user.email "tu_correo@ejemplo.com"`.

5.3 Instalar Visual studio code.

1. Descargar VS Code:

Descarga el instalador desde la página oficial de Visual Studio Code.

La página oficial para descargar Visual Studio Code es:

<https://code.visualstudio.com/>

2. Instalar VS Code:

Ejecuta el instalador y sigue las instrucciones.

3. Iniciar VS Code:

Abre la aplicación desde el menú de inicio, la carpeta de aplicaciones o la línea de comandos.

4. Instalar las siguientes extensiones: Extension Pack for Java, Maven for java, Spring Initializer.

5.4 Instalar Docker.

La página oficial para descargar Docker es:

- <https://www.docker.com/get-started>

En esta página, encontrarás enlaces para descargar Docker Desktop para diferentes sistemas operativos:

Docker Desktop for Windows

Pasos Generales para la Instalación

Los pasos exactos pueden variar ligeramente según tu sistema operativo, pero aquí tienes una descripción general:

1. Requisitos del Sistema

- * Asegúrate de que tu sistema operativo cumpla con los requisitos mínimos de Docker Desktop. Estos requisitos se detallan en la página de descarga y en la documentación de Docker.
- * Por lo general, necesitarás una versión reciente de Windows 10 u 11, macOS o una distribución de Linux compatible.
- * La virtualización de hardware debe estar habilitada en el BIOS/UEFI de tu computadora.

2. Descargar Docker Desktop

- * Ve a la página de descarga de Docker y descarga el instalador correspondiente a tu sistema operativo.

3. Instalar Docker Desktop

- * ****Windows/macOS:****

- * Ejecuta el instalador descargado (`.exe` en Windows, `.dmg` en macOS) y sigue las instrucciones en pantalla.

* Es posible que se te pida que reinicies tu computadora durante el proceso de instalación.

* ****Linux:****

* El proceso de instalación en Linux varía según la distribución.

* Docker proporciona instrucciones detalladas para diferentes distribuciones como Ubuntu, Debian, Fedora y CentOS.

* Generalmente, la instalación implica usar la línea de comandos para agregar el repositorio de Docker, instalar los paquetes necesarios y configurar el servicio de Docker.

4. Iniciar Docker Desktop

* Una vez instalado, inicia la aplicación Docker Desktop.

* Es posible que se te pida que aceptes los términos de servicio.

* Docker Desktop se ejecutará en segundo plano y mostrará un icono en la barra de tareas (Windows) o en la barra de menú (macOS).

5. Verificar la Instalación

* Abre una terminal o línea de comandos.

* Ejecuta el siguiente comando para verificar si Docker se ha instalado correctamente:

```
```bash docker --version ```
```

\* Deberías ver la versión de Docker instalada.

\* También puedes ejecutar:

```
```bash docker run hello-world ```
```

* Este comando descarga una imagen de prueba y ejecuta un contenedor "hello-world". Si todo está configurado correctamente, verás un mensaje de bienvenida.

5.5 Instalar Postman

1. Descargar Postman:

Descarga el instalador desde la página oficial de Postman.

La página oficial para descargar Postman es:

<https://www.postman.com/downloads/>

2. Instalar Postman:

Ejecuta el instalador y sigue las instrucciones.

3. Iniciar Postman:

Abre la aplicación Postman.

5.6 Instalar Maven

1. Requisitos Previos:

Instala el JDK (Java Development Kit) [Ver Sección 5.1](#).

2. Descargar Maven:

Descarga el archivo binario .zip o .tar.gz desde la página oficial de Maven.

La página oficial para descargar Apache Maven es:

<https://maven.apache.org/download.cgi>

3. Instalar Maven:

Extrae el archivo descargado en el directorio de instalación.

4. Configurar Variables de Entorno:

Configura JAVA_HOME (apuntando al JDK).

Configura M2_HOME o MAVEN_HOME (apuntando a la instalación de Maven).

Agrega el directorio bin de Maven a la variable Path.

5. Verificar la Instalación:

Abre la línea de comandos y ejecuta mvn -v.

6. Como Descargar, Ejecutar y Probar el proyecto.

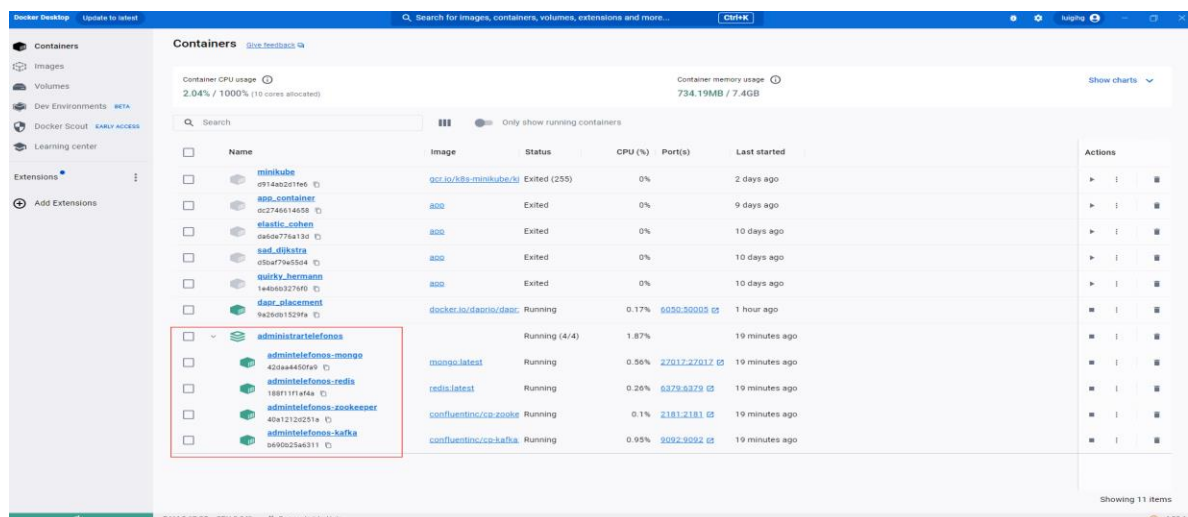
6.1 Pasos para Iniciar Docker

1. Buscar Docker Desktop:

- Puedes buscar "Docker Desktop" en el menú Inicio de Windows.

2. Iniciar Docker Desktop:

- Haz clic en la aplicación "Docker Desktop" para iniciar



3. Verificar el Estado de Docker:

- Una vez que Docker Desktop se esté ejecutando, verás el icono de Docker en la bandeja del sistema (la barra de tareas, generalmente en la esquina inferior derecha de la pantalla).
- Puedes hacer clic en el icono de Docker para ver el estado de Docker y acceder a las opciones de configuración.

4. Abrir una Terminal:

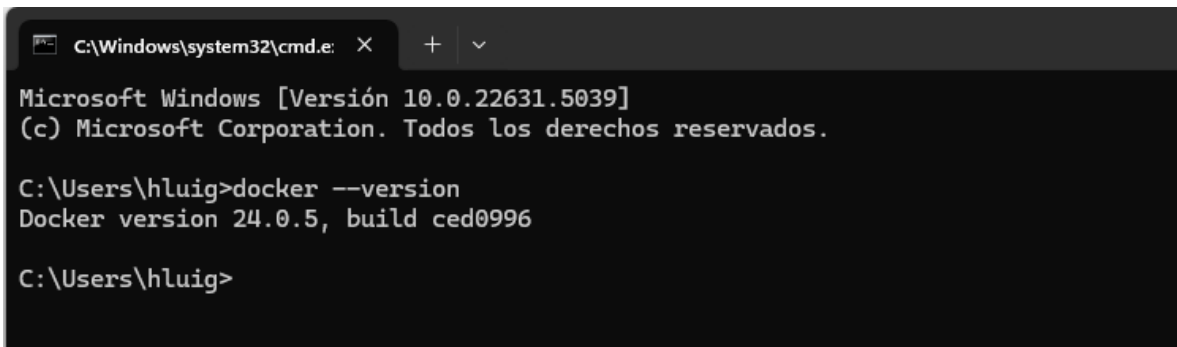
- Abre una terminal de Windows (PowerShell o Símbolo del sistema).

5. Verificar la Instalación de Docker (Opcional):

- Puedes verificar si Docker se ha iniciado correctamente ejecutando el siguiente comando en la terminal:

Bash

`docker --version`



```
C:\Windows\system32\cmd.e  X  +  v

Microsoft Windows [Versión 10.0.22631.5039]
(c) Microsoft Corporation. Todos los derechos reservados.

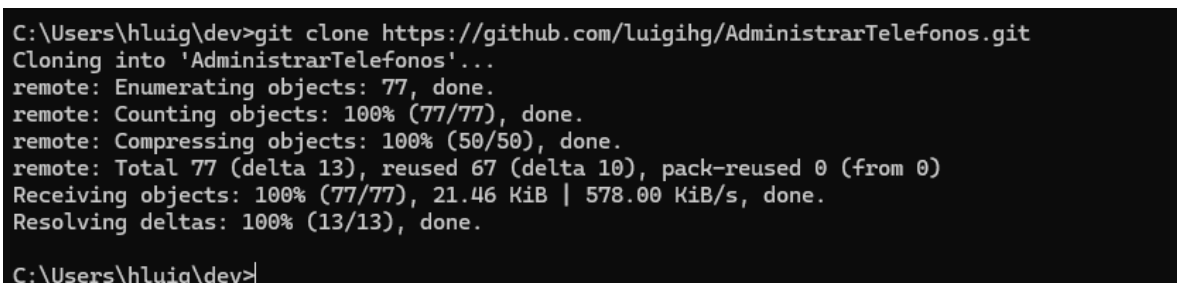
C:\Users\hluig>docker --version
Docker version 24.0.5, build ced0996

C:\Users\hluig>
```

6.2 Descargar el proyecto de Github.

1. En la consola de powershell o cmd ejecutamos el siguiente comando:

>git clone <https://github.com/luigihg/AdministrarTelefonos.git>



```
C:\Users\hluig\dev>git clone https://github.com/luigihg/AdministrarTelefonos.git
Cloning into 'AdministrarTelefonos'...
remote: Enumerating objects: 77, done.
remote: Counting objects: 100% (77/77), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 77 (delta 13), reused 67 (delta 10), pack-reused 0 (from 0)
Receiving objects: 100% (77/77), 21.46 KiB | 578.00 KiB/s, done.
Resolving deltas: 100% (13/13), done.

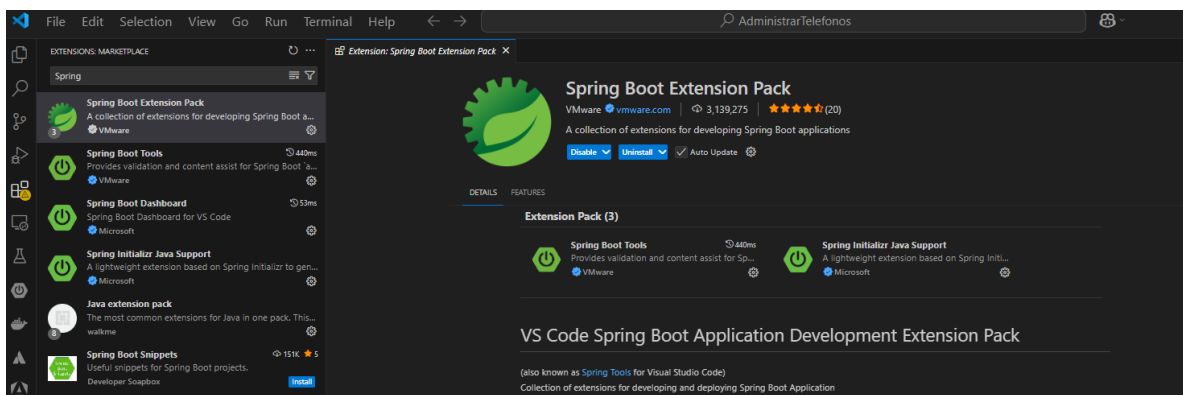
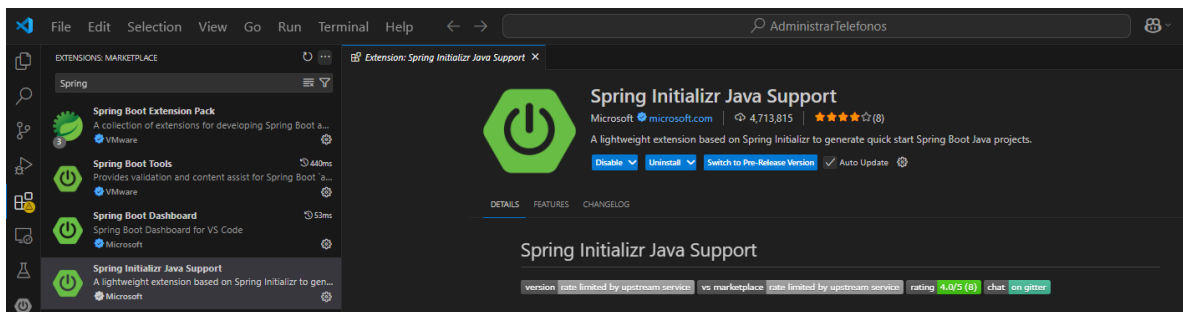
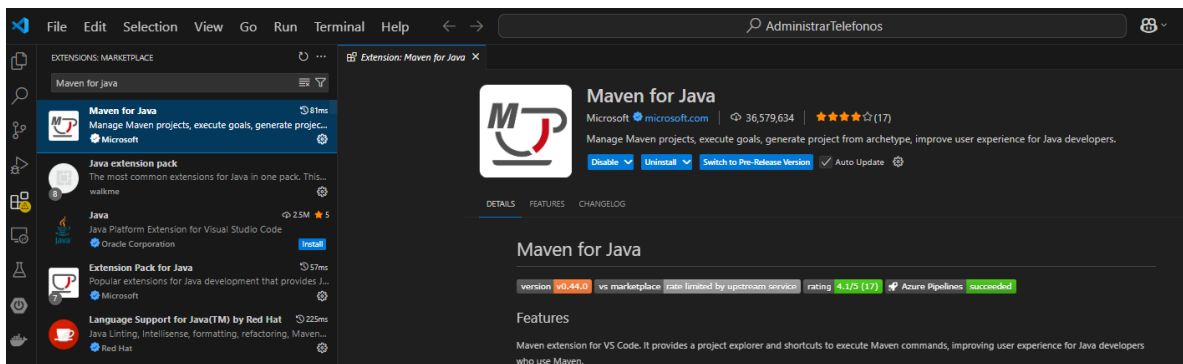
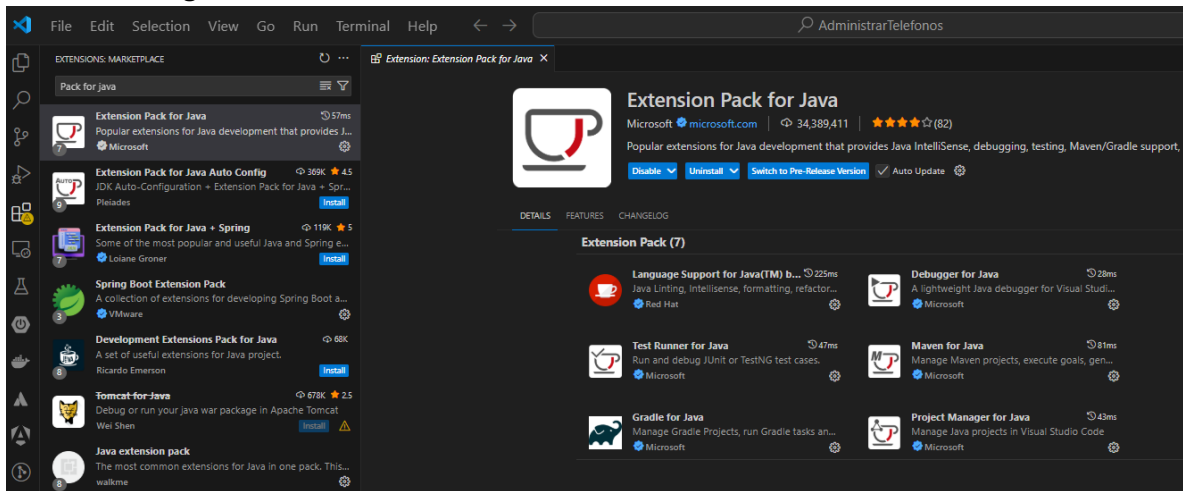
C:\Users\hluig\dev>
```

>cd AdministrarTelefonos

AdministrarTelefonos> code .



Verificar las siguientes extensiones estén instaladas:



Ir al menú Terminal> ejecutar el siguiente comando>

Elimina y Recrea la Red:

- A veces, la red Docker Compose puede entrar en un estado inconsistente. Eliminarla y recrearla puede solucionar el problema.

Bash

docker-compose down

`docker network prune -f` # Elimina todas las redes no utilizadas

docker-compose up --build

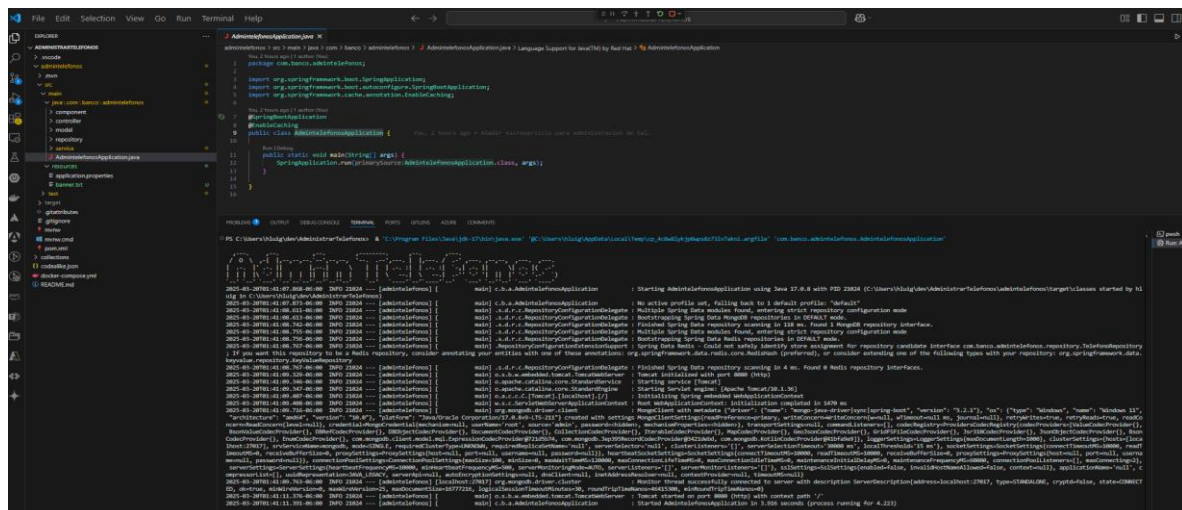
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE COMMENTS
● PS C:\Users\hluig\dev\AdministrarTelefonos> docker-compose up --build -d
[+] Running 9/9
  ✓ mongo 8 layers [██████████] 0B/0B Pulled
    ✓ 5a7813e071bf Already exists
    ✓ d67c4ebf9460 Pull complete
    ✓ 7afaf02f8c09e Pull complete
    ✓ 4e7ca17a42bd Pull complete
    ✓ 342a4f4728ff Pull complete
    ✓ d5baf1d14ffe8 Pull complete
    ✓ 0ec492c8e8cfd Pull complete
    ✓ 734719e891c0 Pull complete
[+] Running 3/3
  ✓ Network administrartelefonos_default Created
  ✓ Volume "administrartelefonos_mongo-data" Created
  ✓ Container administrartelefonos-mongo-1 Started
● PS C:\Users\hluig\dev\AdministrarTelefonos>

```

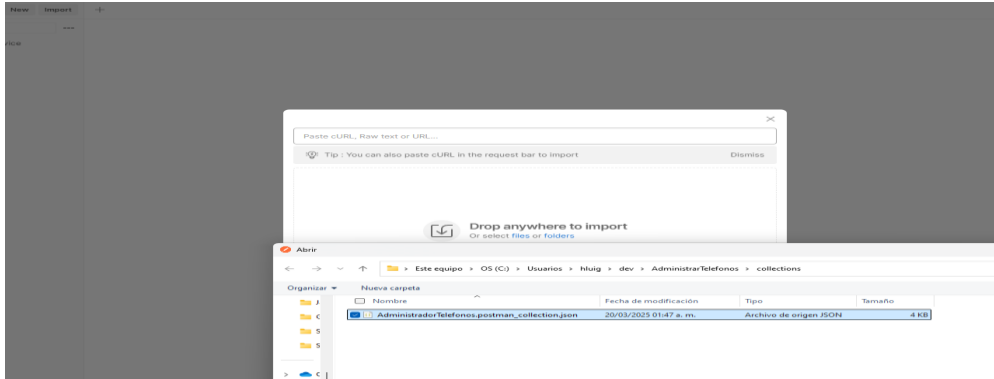
6.3 Ejecutar el proyecto.

Ejecutamos Run Java seleccionamos la clase AdmintelefonosApplication.java botón derecho Run Java.



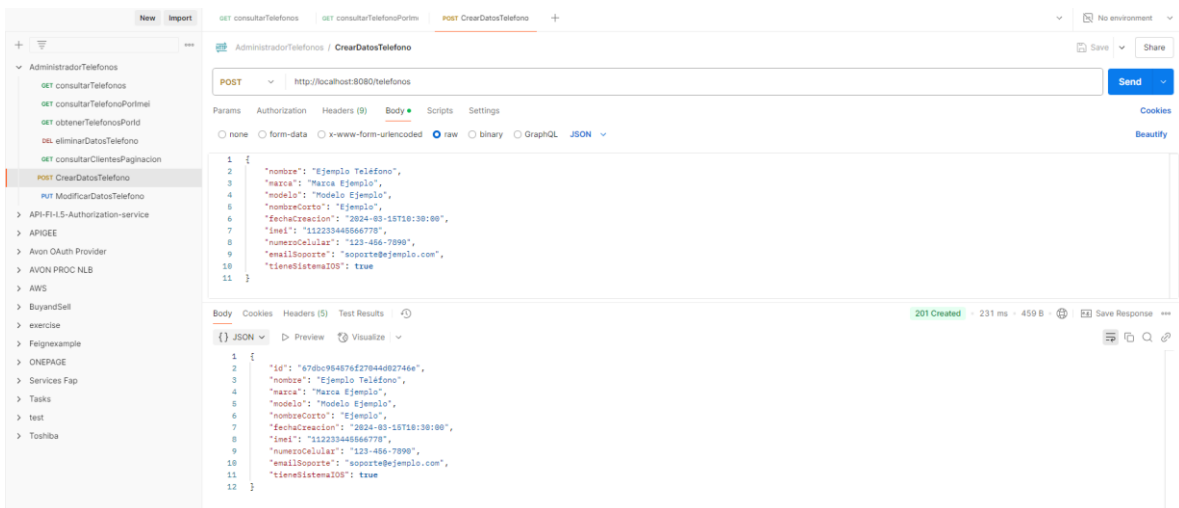
6.4 Probamos el proyecto.

1. Abrir postman
2. Importamos la collection que se encuentra en la raíz de la carpeta con el nombre `AdministrarTelefonos\collections\AdministradorTelefonos.postman_collection.json`

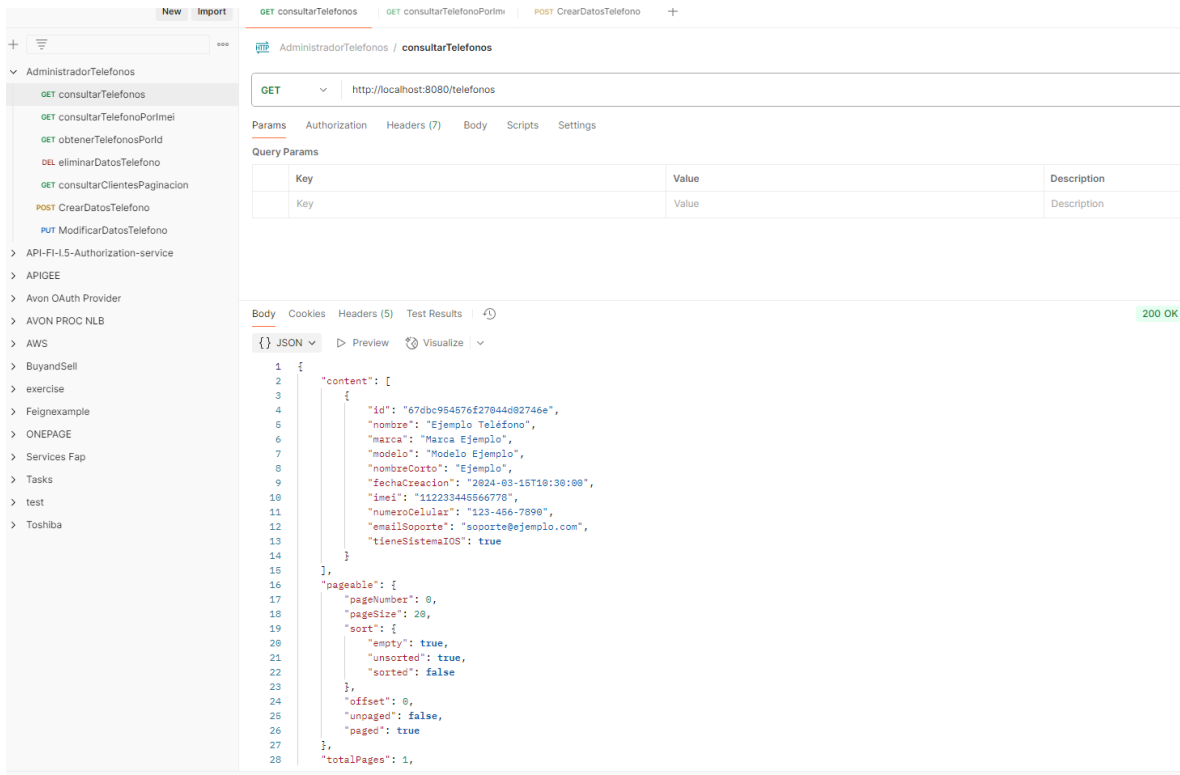


3. Abrimos la collection CrearDatosTelefono.

Ejecutamos para crear un nuevo teléfono.



Damos clic a ejecutar la collection consultarTelefonos



7. Como obtener el header obligatorio DATA

1. Obtener SHA-256 MIEXAMENPRUEBA

Ir a la página.

<https://emn178.github.io/online-tools/sha256.html>

Seleccionar OutputEncoding : Base 64

Input colocar: MIEXAMENPRUEBA

Verificar la salida: **bXIHGHDUTBaGk7VX3Uj+7dfHY5N6nwK684IKIYvucD8=**

2. Configurar Postman

- **Abre Postman.**
- **Crea una nueva solicitud.**
- **Ingresa la URL del endpoint que deseas probar.** Por ejemplo, `http://localhost:8080/telefonos`.
- **Selecciona el método HTTP apropiado.** Por ejemplo, GET, POST, PUT, DELETE.

3. Prueba con el Header "DATA" Correcto

- **Ve a la pestaña "Headers".**
- **Agrega un nuevo header con el nombre "DATA".**
- **Pega el hash SHA-256 que obtuviste en el paso 1 como el valor del header.**
- **Envía la solicitud.**

Resultado esperado:

- La solicitud debe ser procesada correctamente por tu controlador.
- Debes recibir la respuesta esperada del endpoint.
- El código de estado de la respuesta debe ser 200 (OK) o el código de estado apropiado para el endpoint.

4. Prueba con el Header "DATA" Incorrecto

- **Ve a la pestaña "Headers".**
- **Cambia el valor del header "DATA" a un hash incorrecto.** (Por ejemplo, un hash aleatorio o el hash de otra cadena).
- **Envía la solicitud.**

Resultado esperado:

- La solicitud debe ser rechazada por el filtro.
- Debes recibir una respuesta con código de estado 401 (Unauthorized).
- El cuerpo de la respuesta debe contener el mensaje "Unauthorized: Invalid DATA header".

5. Prueba sin el Header "DATA"

- **Ve a la pestaña "Headers".**
- **Elimina el header "DATA".**
- **Envía la solicitud.**

Resultado esperado:

- La solicitud debe ser rechazada por el filtro.
- Debes recibir una respuesta con código de estado 401 (Unauthorized).
- El cuerpo de la respuesta debe contener el mensaje "Unauthorized: Invalid DATA header".

8. Servicios y documentación de la API tipos de respuesta.

1. Descripción de los Servicios de la API de Teléfonos.

1. **Consultar** los datos de un teléfono
2. **Crear** los datos de un teléfono
3. **Eliminar** los datos de un teléfono
4. **Actualizar** los datos de un teléfono
5. Crear un método de **consulta por IMEI**
6. Consulta de todos los registros de los clientes con paginación

2. Servicios y Documentación de la API

Aquí se detalla cada endpoint, sus métodos, payloads, respuestas y posibles errores.

Descripción	Método	Endpoint
Lista teléfonos	GET	/telefonos
Obtiene un teléfono por su ID	GET	/telefonos/{id}
Obtiene un teléfono por su IMEI	GET	/telefonos/imei/{imei}
Crea un nuevo teléfono	POST	/telefonos
Actualiza un teléfono existente	PUT	/telefonos/{id}
Elimina un teléfono por su ID	DELETE	/telefonos/{id}

a) Obtener Teléfonos (Listar)

Payload de Entrada:

Ninguno (puede aceptar parámetros de paginación en la URL, como page y size).

Salida de Respuesta (Éxito - 200 OK):

Tipos de Errores:

Ninguno específico del endpoint (errores generales del servidor pueden ocurrir).

b) Obtener Teléfono por ID

Payload de Entrada:

Ninguno



Salida de Respuesta (Éxito - 200 OK):

Tipos de Errores:

404 Not Found: Si el teléfono con el ID especificado no existe.

c) Obtener Teléfono por IMEI

Payload de Entrada:

Ninguno

Salida de Respuesta (Éxito - 200 OK):

Tipos de Errores:

404 Not Found: Si el teléfono con el IMEI especificado no existe.

d) Crear Teléfono

Payload de Entrada:

```
{
  "nombre": "Ejemplo Telefono",
  "marca": "Marca Ejemplo",
  "modelo": "Modelo Ejemplo",
  "nombreCorto": "Ejemplo *",
  "fechaCreacion": "2024-03-15T10:30:00",
  "imei": "112233445566778",
  "numeroCelular": "5642877166",
  "emailSoporte": "soporte gmail.com",
  "tieneSistemaIOS": true
}
```

Salida de Respuesta (Éxito - 201 Created):

Tipos de Errores:

400 Bad Request: Si los datos de entrada no son válidos (por ejemplo, campos faltantes, formato incorrecto, IMEI ya existe).

○ **ErrorResponse:**

Códigos de error posibles:



1001: El nombre es requerido.

1002: El nombre no debe tener caracteres especiales.

1003: La Marca es requerida.

1004: La Marca no debe tener caracteres especiales.

1005: El Modelo es requerido.

1006: El Modelo no debe tener caracteres especiales.

1007: El Nombre corto es requerido.

1008: El Nombre corto no debe tener caracteres especiales.

1009: La Fecha de creación es requerida.

1011: La Fecha de creación tiene un formato invalido el formato valido es YYYY-MM-DDTHH:mm:ss

1012: El IMEI ingresado ya existe.

1013: El IMEI es requerido.

1014: El formato del correo electrónico es inválido.

e) Actualizar Teléfono

Payload de Entrada:

```
{  
  "nombre": "Ejemplo Telefono",  
  "marca": "Marca Ejemplo",  
  "modelo": "Modelo Ejemplo",  
  "nombreCorto": "Ejemplo",  
  "fechaCreacion": "2025-03-15T10:30:00",  
  "imei": "11223344556678",  
  "numeroCelular": "5642877166",  
  "emailSoporte": "soporte@gmail.com",  
  "tieneSistemaIOS": true  
}
```



Salida de Respuesta (Éxito - 200 OK):

Tipos de Errores:

- **400 Bad Request: Si los datos de entrada no son válidos.**

- **ErrorResponse:**

1001: El nombre es requerido.

1002: El nombre no debe tener caracteres especiales.

1003: La Marca es requerida.

1004: La Marca no debe tener caracteres especiales.

1005: El Modelo es requerido.

1006: El Modelo no debe tener caracteres especiales.

1007: El Nombre corto es requerido.

1008: El Nombre corto no debe tener caracteres especiales.

1009: La Fecha de creación es requerida.

1011: La Fecha de creación tiene un formato invalido el formato valido es YYYY-MM-DDTHH:mm:ss

1012: El IMEI ingresado ya existe.

1013: El IMEI es requerido.

1014: El formato del correo electrónico es inválido.

f) Eliminar Teléfono

Payload de Entrada:

Ninguno

Salida de Respuesta (Éxito - 204 No Content):

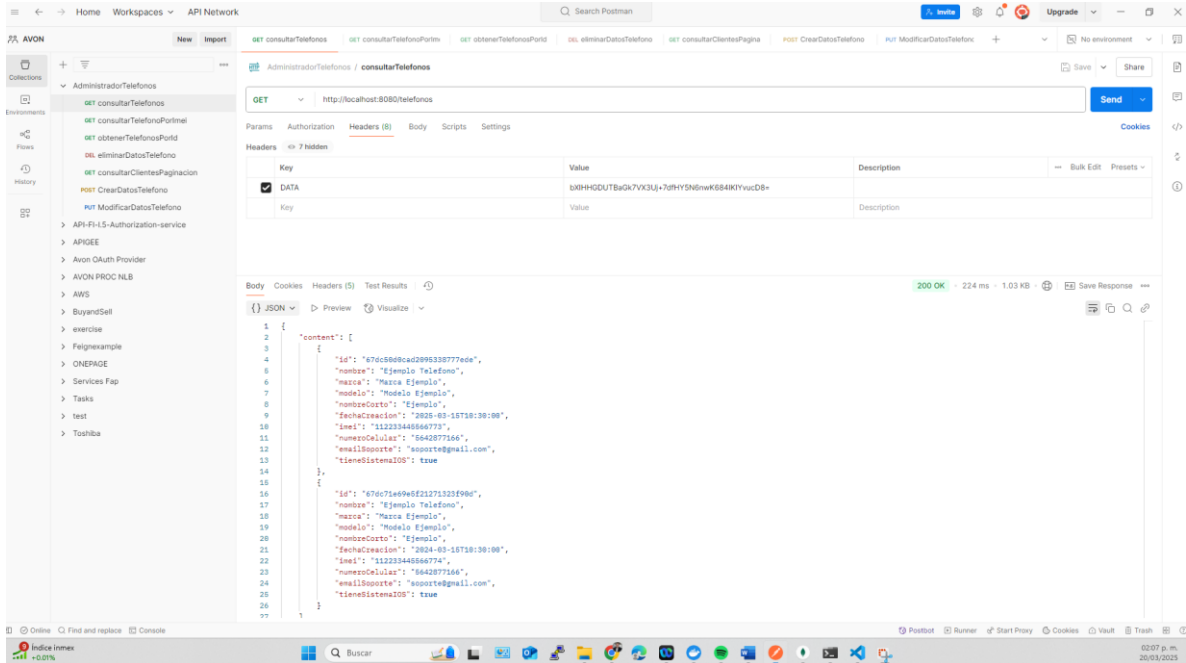
Sin cuerpo de respuesta

Tipos de Errores:

404 Not Found: Si el teléfono con el ID especificado no existe.

9. Validar y Probar los servicios en postman.

1. **Consultar** los datos de un teléfono



Administrador Telefonos / consultarTelefonos

GET http://localhost:8080/telefonos

Headers (7 hidden)

Key	Value	Description
DATA	bXHHGdUTBaG7VXUj+7dRySN6wK84K0YvucD6+	

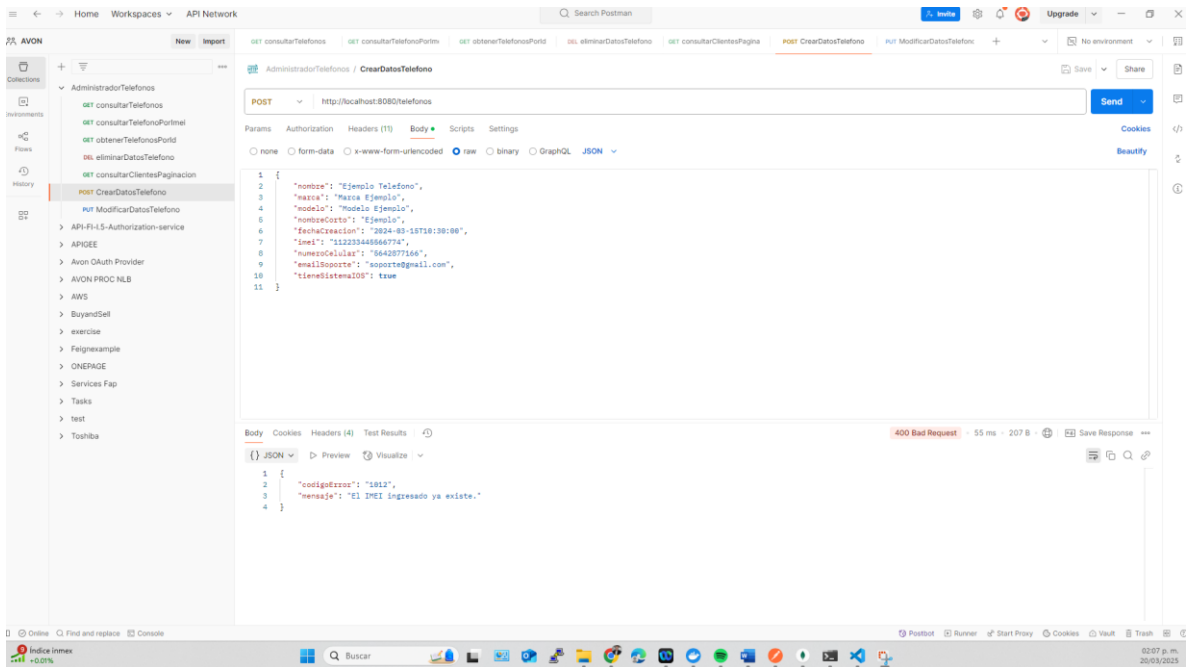
Body

```

1 {
2   "content": [
3     {
4       "id": "67c6860cad399532777ede",
5       "nombre": "Ejemplo Telefono",
6       "marca": "Marca Ejemplo",
7       "modelo": "Modelo Ejemplo",
8       "nombreCorto": "Ejemplo",
9       "fechaCreacion": "2024-03-18T18:38:08",
10      "imei": "112233445566775",
11      "numeroCelular": "5642877566",
12      "emailSoporte": "soporte@gmail.com",
13      "tieneSistemaIOS": true
14    },
15    {
16      "id": "67c675e9b6521271323e9b0",
17      "nombre": "Ejemplo Telefono",
18      "marca": "Marca Ejemplo",
19      "modelo": "Modelo Ejemplo",
20      "nombreCorto": "Ejemplo",
21      "fechaCreacion": "2024-03-18T18:38:08",
22      "imei": "112233445566774",
23      "numeroCelular": "5642877566",
24      "emailSoporte": "soporte@gmail.com",
25      "tieneSistemaIOS": true
26    }
27  ]
28 }

```

2. Crear los datos de un teléfono



Administrador Telefonos / CrearDatosTelefono

POST http://localhost:8080/telefonos

Body

```

1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-18T18:38:08",
7   "imei": "112233445566774",
8   "numeroCelular": "5642877566",
9   "emailSoporte": "soporte@gmail.com",
10  "tieneSistemaIOS": true
11 }

```

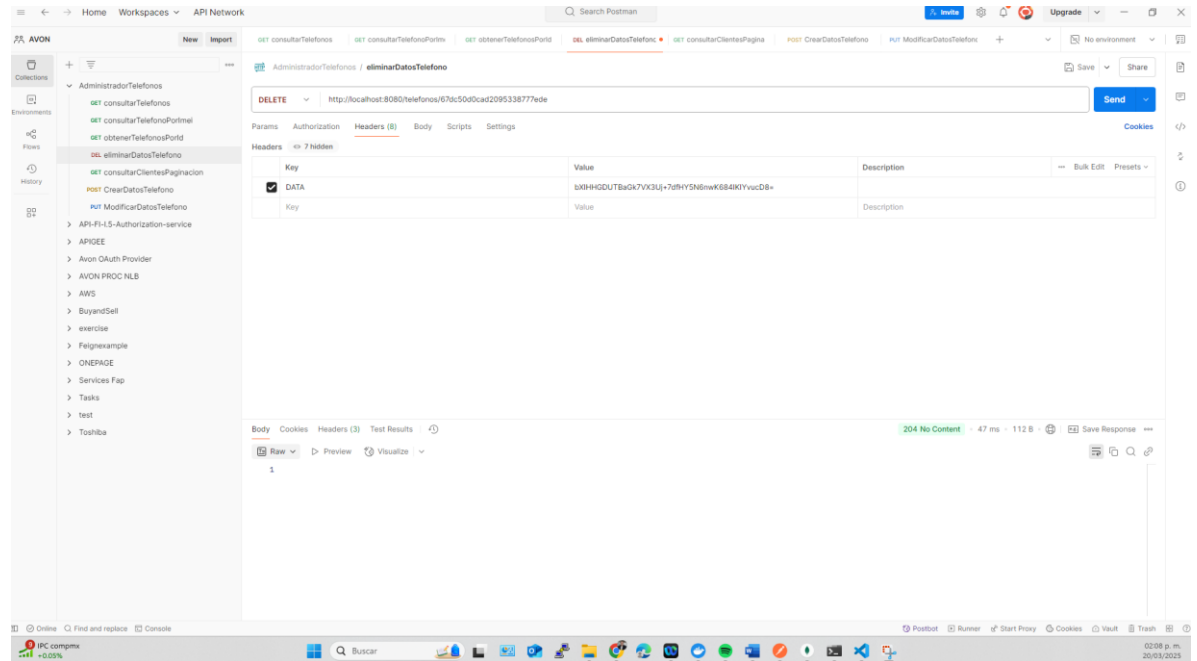
Body

```

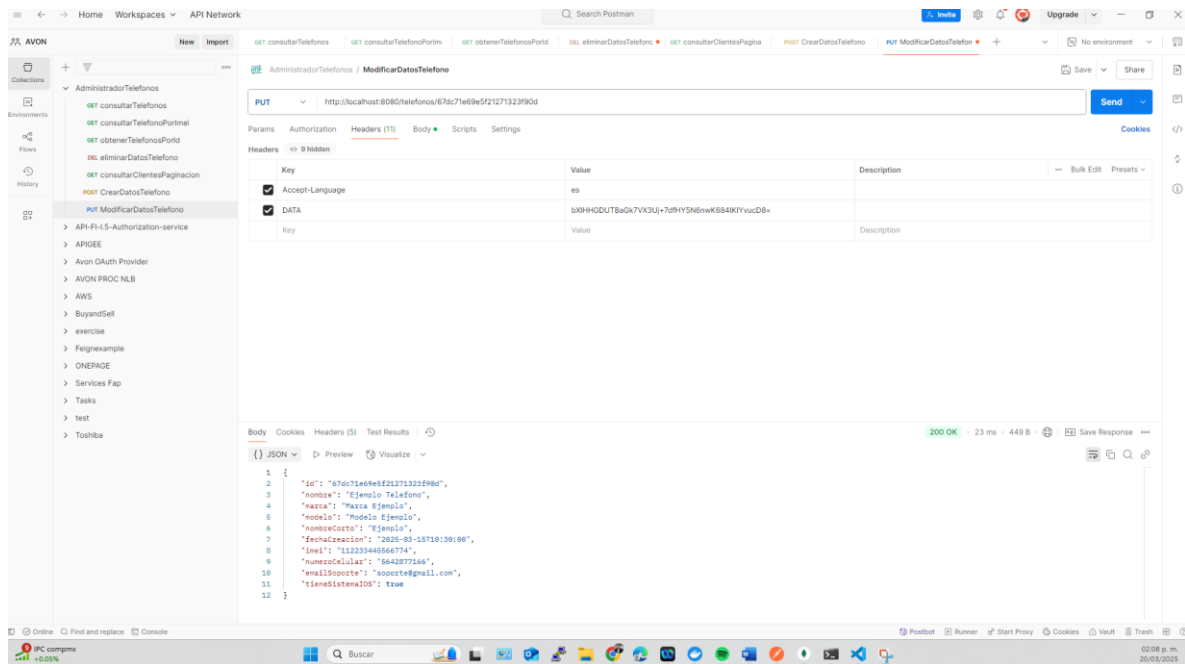
1 {
2   "codigoError": "1812",
3   "mensaje": "El IMEI ingresado ya existe."
4 }

```

3. Eliminar los datos de un teléfono



4. Actualizar los datos de un teléfono



5. Crear un método de consulta por IMEI



Postman interface showing a GET request to `http://localhost:8080/telefonos/imei112233445566774`. The response is a 200 OK with a JSON body:

```
{
  "id": "67dc71e69e521271323f96d",
  "nombre": "Ejemplo Telefono",
  "marca": "Marca Ejemplo",
  "modelo": "Modelo Ejemplo",
  "nombreCosto": "Ejemplo",
  "fechaCreacion": "2026-03-18T18:38:00",
  "imei": "112233445566774",
  "numeroCelular": "6642077166",
  "emailBoquete": "aboquete@gmail.com",
  "tieneSistemaIOS": true
}
```

6. Consulta de todos los registros de los clientes con **paginación**.

Postman interface showing a GET request to `http://localhost:8080/telefonos?page=0&size=10`. The response is a 200 OK with a JSON body:

```
{
  "content": [
    {
      "id": "67dc71e69e521271323f96d",
      "nombre": "Ejemplo Telefono",
      "marca": "Marca Ejemplo",
      "modelo": "Modelo Ejemplo",
      "nombreCosto": "Ejemplo",
      "fechaCreacion": "2026-03-18T18:38:00",
      "imei": "112233445566774",
      "numeroCelular": "6642077166",
      "emailBoquete": "aboquete@gmail.com",
      "tieneSistemaIOS": true
    }
  ],
  "pagination": {
    "pageNumber": 0,
    "pageSize": 10,
    "sort": {
      "empty": true,
      "unsorted": true,
      "sorted": false
    },
    "offset": 0,
    "paged": true,
    "unpaged": false
  },
  "last": true,
  "totalPages": 1
}
```

7. Validaciones campos.



GET consultarTelefonos

GET consultarTelefonoPorId

GET obtenerTelefonosPorId

DEL eliminarDatos

AdministradorTelefonos / CrearDatosTelefono

POST http://localhost:8080/telefonos

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "nombre": "Ejemplo Telefono *",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566774",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte@gmail.com",
10  "tieneSistemaIOS": true
11 }
```

Body Cookies Headers (11) Test Results

JSON Preview Visualize

```
1 {
2   "codigoError": "1002",
3   "mensaje": "El nombre no debe tener caracteres especiales."
4 }
```

AdministradorTelefonos / CrearDatosTelefono

POST http://localhost:8080/telefonos

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary

```
1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566774",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte@gmail.com",
10  "tieneSistemaIOS": true
11 }
```

Body Cookies Headers (4) Test Results

JSON Preview Visualize

```
1 {
2   "codigoError": "1012",
3   "mensaje": "El IMEI ingresado ya existe."
4 }
```


GET consultarTelefonosGET consultarTelefonoPorImGET obtenerTelefonosPorId

AdministradorTelefonos / CrearDatosTelefono

POSThttp://localhost:8080/telefonos

ParamsAuthorizationHeaders (11)BodyScriptsSettings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQL

```
1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566778",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte gmail.com",
10  "tieneSistemaIOS": true
11 }
```

BodyCookiesHeaders (4)Test ResultsTest Results

JSONPreviewVisualize

```
1 {
2   "codigoError": "1014",
3   "mensaje": "El formato del correo electrónico es inválido."
4 }
```

GET consultarTelefonosGET consultarTelefonoPorImGET obtenerTelefonosPorIdDEL eliminarDatosTelefonos

AdministradorTelefonos / CrearDatosTelefono

POSThttp://localhost:8080/telefonos

ParamsAuthorizationHeaders (11)BodyScriptsSettings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQLJSON

```
1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo *",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566778",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte gmail.com",
10  "tieneSistemaIOS": true
11 }
```

BodyCookiesHeaders (4)Test ResultsTest Results

JSONPreviewVisualize

```
1 {
2   "codigoError": "1004",
3   "mensaje": "La Marca no debe tener caracteres especiales."
4 }
```

GET consultarTelefonosGET consultarTelefonoPorImGET obtenerTelefonosPorIdDEL

AdministradorTelefonos / CrearDatosTelefono

POSThttp://localhost:8080/telefonos

ParamsAuthorizationHeaders (11)BodyScriptsSettings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQLJSON

```
1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo *",
5   "nombreCorto": "Ejemplo",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566778",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte gmail.com",
10  "tieneSistemaIOS": true
11 }
```

BodyCookiesHeaders (4)Test ResultsTest Results

JSONPreviewVisualize

```
1 {
2   "codigoError": "1006",
3   "mensaje": "El Modelo no debe tener caracteres especiales."
4 }
```

GET consultarTelefonosGET consultarTelefonoPorImGET obtenerTelefonosPorIdDEL elimin

AdministradorTelefonos / CrearDatosTelefono

POSThttp://localhost:8080/telefonos

ParamsAuthorizationHeaders (11)BodyScriptsSettings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQLJSON

```
1 {
2   "nombre": "Ejemplo Telefono",
3   "marca": "Marca Ejemplo",
4   "modelo": "Modelo Ejemplo",
5   "nombreCorto": "Ejemplo *",
6   "fechaCreacion": "2024-03-15T10:30:00",
7   "imei": "112233445566778",
8   "numeroCelular": "5642877166",
9   "emailSoporte": "soporte gmail.com",
10  "tieneSistemaIOS": true
11 }
```

BodyCookiesHeaders (4)Test ResultsTest Results

JSONPreviewVisualize

```
1 {
2   "codigoError": "1008",
3   "mensaje": "El Nombre corto no debe tener caracteres especiales."
4 }
```