En la presente, AdDynamics expone una propuesta de practica para el programa de pasantías llevadas a cabo por estudiantes de Alta Gracia organizado por el Ministerio de Educación de Alta Gracia.

En base a lo hablado con el grupo de estudiantes, sus experiencias previas e interés en ampliar conocimientos y prácticas en lo que respecta a la programación, se propone el desarrollo de una API en Python y con acceso a base de datos MySql, y opcionalmente una web para la visualización de los datos.

Se pretende que adquieran nociones básicas en el desarrollo de APIs, interfaces que son ampliamente utilizadas para conectar todo tipo de sistema.

El desarrollo de APIs no esta atado a un lenguaje en particular, sino que es posible en muchos de ellos. Se ha optado por Python ya que el grupo ha incursionado en este y facilitará su implementación.

Especificaciones del proyecto

API

Desarrollo de una API que permita registrar los datos reportados por un equipo de telemetría y dar acceso a la información almacenada a un sistema web para su visualización.

Los datos (BODY) deben enviarse y recibirse en formato JSON.

1 - Registro de telemetría
Se considera que el equipo de telemetría reporta los siguientes datos:

1. Serial: identificador único del equipo, es decir, no puede existir el mimo identificador para mas de un equipo. Valor alfanumérico de 12 caracteres. Campo obligatorio.
2. Fecha hora en que se obtuvieron los datos de los sensores. Campo obligatorio.
3. Nivel de batería. Valores posibles: 1 a 100%. Campo obligatorio.
4. Velocidad: numero entero positivo Campo obligatorio.
5. Temperatura de agua: valores numéricos negativos y positivos hasta con 2 decimales. Puede no enviarse.
6. Presión de aceite: valores numéricos negativos y positivos hasta con 2 decimales. Puede no enviarse.

Unicamente debe registrar los datos de los dispositivos que estén activos.

2 – Acceso a datos

La API, ademas de permitir el registro, debe proveer endpoints para el acceso a los datos que serán utilizados por la web de visualización, siendo estos:

1. La lista de los dispositivos activos: Identificador, Nombre
2. Un reporte del ultimo estado de todos los dispositivos activos: Identificador, fecha hora de ultima actualización, nivel de batería, velocidad, temperatura y presión
3. Métricas de un dispositivo en particular para un periodo de fecha específico:
    1. Parámetros: Identificador del dispositivo, Fecha Inicio, Fecha Fin
    2. Resultado: lista de métricas ordenas por fecha: Fecha, nivel de batería, velocidad, temperatura y presión

3 – Autenticación

Opcionalmente puede disponerse de una endpoint para la generación del Token.

En este caso, el Token debe utilizarse unicamente para la obtención de la información, no así para el registro.

Para la obtención del Token debe enviarse un nombre de usuario y contraseña definido.

4 - Modelo de datos

La base de datos debe estructurase con 3 entidades:

1. Dispositivo: Almacena información del dispositivo.
    1. Identificador: obligatorio
    2. Nombre: obligatorio
    3. Fecha Hora Alta: obligatorio
    4. Fecha Hora Baja: si es NULL, el dispositivo esta habilitado, caso contrario no deben registrarse las métricas.
2. Estado: Guarda el ultimo estado conocido. Solo debe haber un registro por dispositivo.
    1. Relación al dispositivo
    2. Fecha hora actualización
    3. Nivel de batería
    4. Velocidad
    5. Temperatura de agua
    6. Presión de aceite
3. Reportes: histórico de reportes de métricas. Almacena cada registro reportado por el dispositivo
    1. Relación al dispositivo
    2. Fecha hora
    3. Nivel de batería
    4. Velocidad
    5. Temperatura de agua
    6. Presión de aceite

4 - Resumen de rutas de acceso a API

1. Registro de telemetría mediante el dispositivo: POST con los datos en el BODY
2. Obtención de ultimo estado de los dispositivos: GET que devuelta todos los dispositivos activos y su estado
3. Obtención de lista de dispositivos: GET para obtener la lista de dispositivos activos
4. Obtención de histórico de métricas: GET para obtener las métricas de un dispositivo en un periodo de fecha.
5. Opcionalmente: Autenticación para obtener Token y utilizarlo en los GET como método de autenticación. POST que envía nombre de usuario y password para obtener Token.

Web de visualización

Como complemento puede desarrollarse una web en HTML / Javascript para mostrar los datos registrados.

1. Deberá incluirse un Login si se implementa la Tokenización
2. Una vista para listar todos el estado actual de todos los dispositivos activos
3. Algunas otras vistas
    1. Mostrar máximos, mínimos, promedio de valores para un dispositivos específico en un periodo determinado. Esto puede calcularse en la web o en la api (en este ultimo caso deberá agregarse un nuevo endpoint)
    2. Gráfico en tiempo de una variable en particular (ej, velocidad) para un dispositivo.

Ejemplo

Se muestra una parte de un código de API en Python con integración con Swagger, acceso la base de datos y uso de Token (JWT).

```python
# Endpoint para registrar una venta (requiere token)
@swagger.tags(['Ventas'])
class RegistrarVenta(Resource):
    @swagger.doc({
        'summary': 'Registrar una venta',
        'description': 'Permite registrar una nueva venta en la base de datos',
        'parameters': [{'name': 'body', 'in': 'body', 'required': True, 'schema': '
        'responses': {
            '201': {
                'description': 'Venta registrada exitosamente'
            }
        }
    })
    @jwt_required()
    def post(self):
        producto_id = request.json.get('producto_id')
        cantidad = request.json.get('cantidad')

        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO ventas(producto_id, cantidad) VALUES (%s, %s)", (
        mysql.connection.commit()
        return jsonify({"msg": "Venta registrada"}), 201

# Registrar los recursos en la API
api.add_resource(Login, '/login')
api.add_resource(Productos, '/productos')
api.add_resource(ActualizarPrecio, '/productos/<int:id>')
api.add_resource(RegistrarVenta, '/ventas')

if __name__ == '__main__':
    app.run(debug=True)
```

Parte de ejemplo de web en HTML con javascript

Archivo HTML

html Copiar código

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Consumo de API</title>
</head>
<body>
    <h1>Consumo de API</h1>

    <!-- Formulario de Login -->
    <h2>Login</h2>
    <form id="loginForm">
        <label for="username">Usuario:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>

    <!-- Lista de Productos -->
    <h2>Lista de Productos</h2>
    <button id="getProductosBtn">Obtener Productos</button>
    <ul id="productosList"></ul>

    <!-- Formulario de Registro de Venta -->
    <h2>Registrar Venta</h2>
    <form id="ventaForm">
        <label for="producto_id">ID del Producto:</label>
        <input type="text" id="producto_id" name="producto_id" required><br><br>
```

Archivo javascript

`app.js`

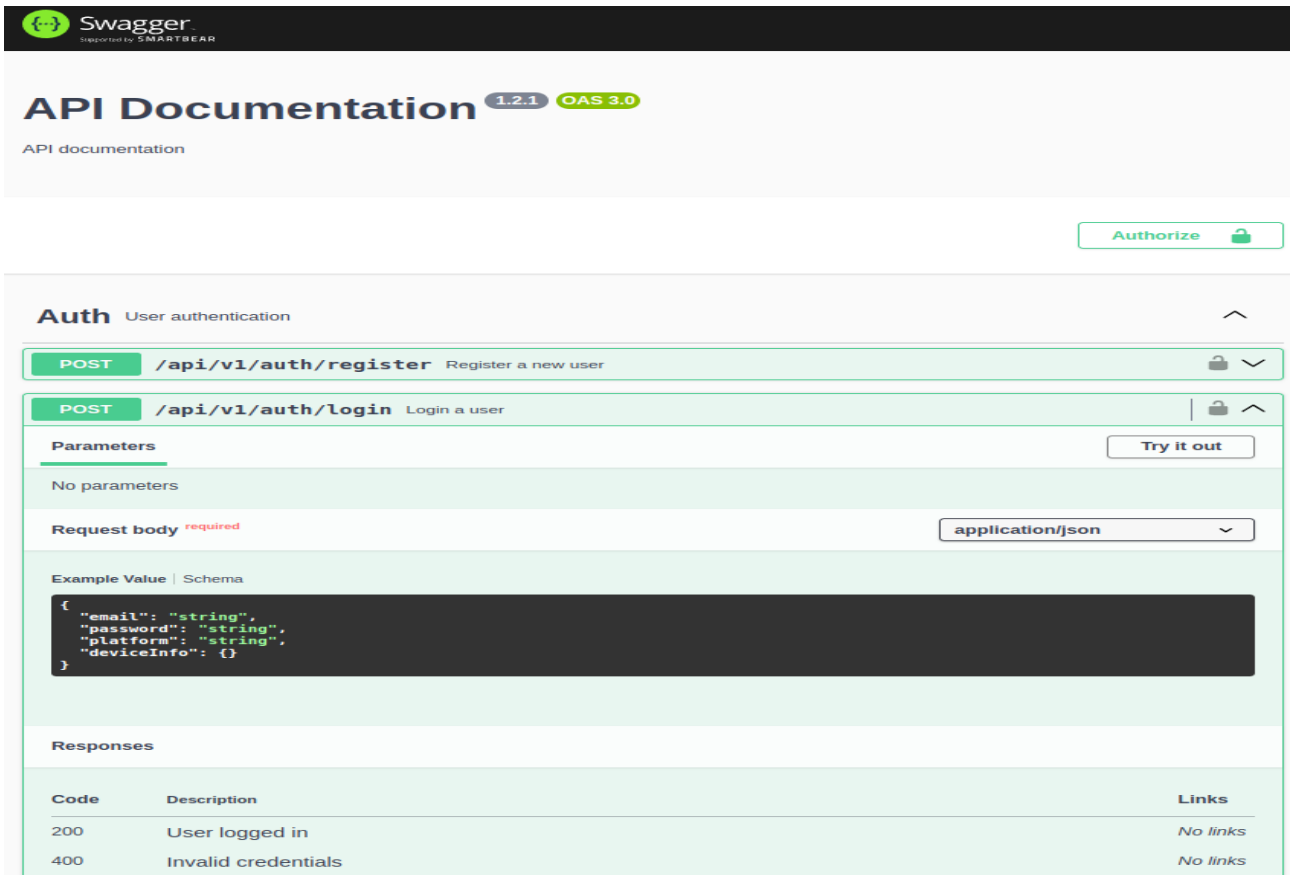
javascript Copiar código

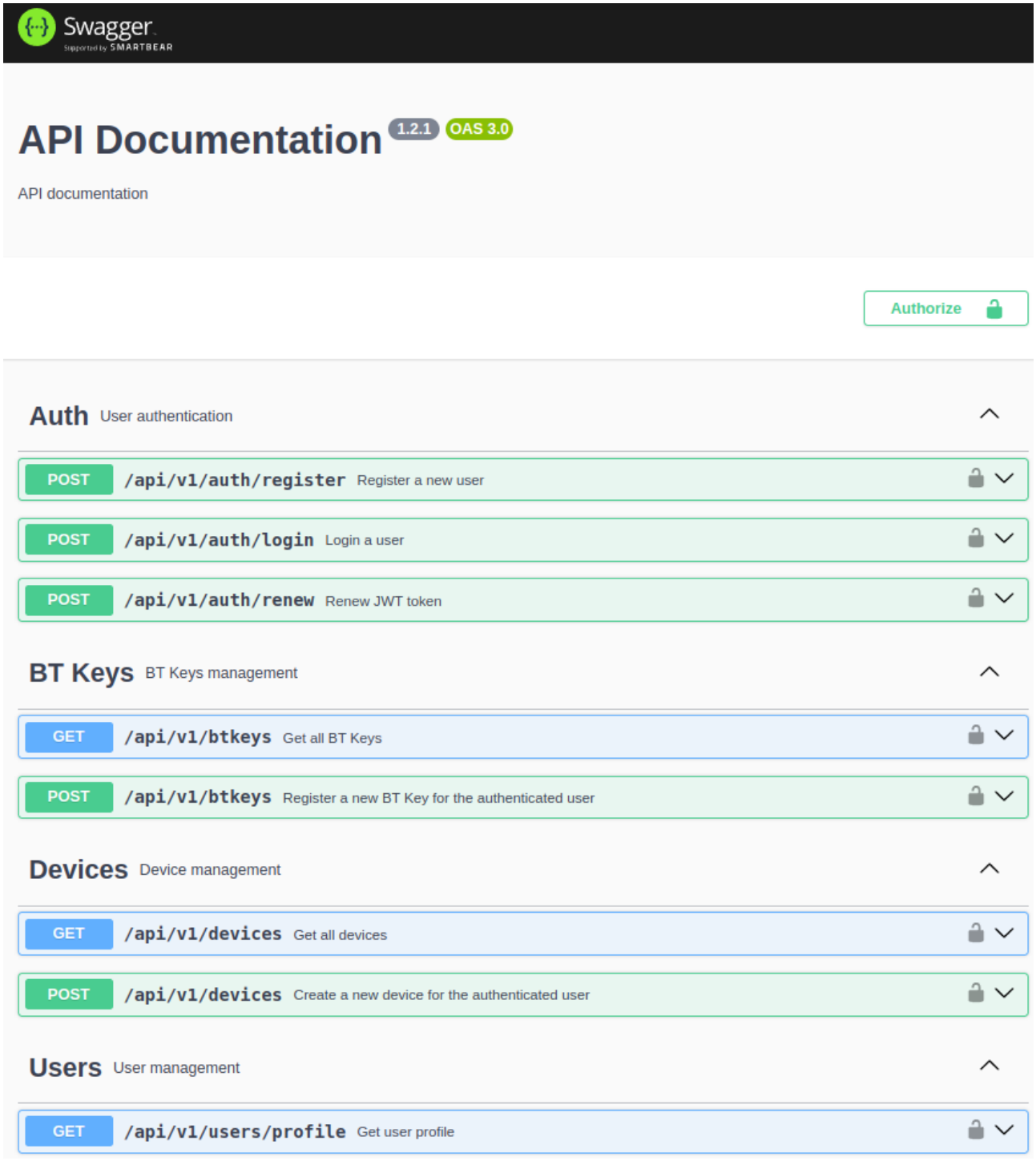
```javascript
let token = '';

// Manejar el login
document.getElementById('loginForm').addEventListener('submit', function(event) {
    event.preventDefault();
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    fetch('http://127.0.0.1:5000/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ username: username, password: password })
    })
    .then(response => response.json())
    .then(data => {
        if (data.access_token) {
            token = data.access_token;
            alert('Login exitoso');
        } else {
            alert('Login fallido');
        }
    })
    .catch(error => console.error('Error:', error));
});
```

Herramientas

SWAGGER

Es una herramienta que permite generar documentación y una interfaz para probar APIs. Se integra en el proyecto de la API.




Swagger
Supported by SMARTBEAR
API Documentation
1.2.1
OAS 3.0
API documentation
Authorize
Auth
User authentication
POST /api/v1/auth/register Register a new user
POST /api/v1/auth/login Login a user
POST /api/v1/auth/renew Renew JWT token
BT Keys
BT Keys management
GET /api/v1/btkeys Get all BT Keys
POST /api/v1/btkeys Register a new BT Key for the authenticated user
Devices
Device management
GET /api/v1/devices Get all devices
POST /api/v1/devices Create a new device for the authenticated user
Users
User management
GET /api/v1/users/profile Get user profile

POSTMAN

Es una herramienta que no es necesaria integrarla al proyecto. Permite consumir la API para realizar todo tipo de pruebas.