Datos del estudiante

Nombre y apellidos Mª Ángeles Domínguez Torvisco

Fecha de entrega 17/02/2026

Aplicación API REST con FastApi

Objetivos de la actividad

A través de esta prueba el alumnado deberá desarrollar una API RESTful utilizando FastAPI, aplicando los conceptos trabajados en clase. El objetivo es comprobar que el estudiante es capaz de arrancar una aplicación FastAPI correctamente, documentarla con Swagger, conectarla a una base de datos MySQL mediante SQLAlchemy e implementar un sistema básico de autenticación mediante JWT, protegiendo al menos un endpoint de la API.

La prueba está basada íntegramente en la metodología y ejemplos trabajados en clase durante el tema de FastAPI + MySQL + JWT

Base de datos proporcionada

Se trabajará con una base de datos MySQL **ya creada**, denominada fastapi_incidentes, que contiene una tabla llamada incidencias, destinada a gestionar incidencias de soporte técnico.

```
CREATE DATABASE IF NOT EXISTS fastapi_incidentes
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE fastapi_incidentes;

CREATE TABLE incidencias (
```

```
 id INT AUTO_INCREMENT PRIMARY KEY,
 titulo VARCHAR(150) NOT NULL,
 descripcion TEXT NOT NULL,
 prioridad VARCHAR(20) NOT NULL,
 estado VARCHAR(20) NOT NULL
);

INSERT INTO incidencias (titulo, descripcion, prioridad, estado) VALUES
('No arranca el equipo', 'El ordenador no pasa de la pantalla inicial', 'alta', 'abierta'),
('Error impresora', 'La impresora no responde al enviar trabajos', 'media', 'abierta'),
('Actualización software', 'Pendiente de actualizar el sistema', 'baja', 'cerrada');
```

Pautas de elaboración

1. Configuración del proyecto. Se deberá crear un proyecto FastAPI funcional, instalar las dependencias necesarias y arrancar el servidor correctamente mediante uvicorn. La documentación automática de la API deberá estar disponible en la ruta /docs.

2. Conexión con la base de datos. Se configurará la conexión a la base de datos MySQL utilizando SQLAlchemy. Para ello se crearán los archivos necesarios para la gestión de la base de datos y el modelo correspondiente a la tabla incidencias.

3. Endpoints de la API. Se implementará un endpoint GET /incidencias que devuelva el listado completo de incidencias almacenadas en la base de datos. Los datos deberán obtenerse de forma real desde MySQL.

4. Autenticación JWT. Se deberá implementar un sistema básico de autenticación mediante JWT, utilizando un usuario fijo (sin persistencia en base de datos). Se creará un endpoint POST /login que devuelva un token válido cuando las credenciales sean correctas.

5. Endpoints protegidos. Se creará al menos un endpoint protegido que obtenga información del token (por ejemplo, el nombre del usuario autenticado). Además, el endpoint POST /incidencias, encargado de insertar nuevas incidencias en la base de datos, deberá estar protegido mediante JWT y solo ser accesible con un token válido.

6. Pruebas de funcionamiento. Se comprobará el correcto funcionamiento de la API utilizando Swagger, verificando tanto los endpoints públicos como los protegidos.

Extensión y formato

La entrega de la actividad será obligatoria en dos partes:

1. **Documento PDF.** Se deberá entregar un documento en formato PDF que incluya:

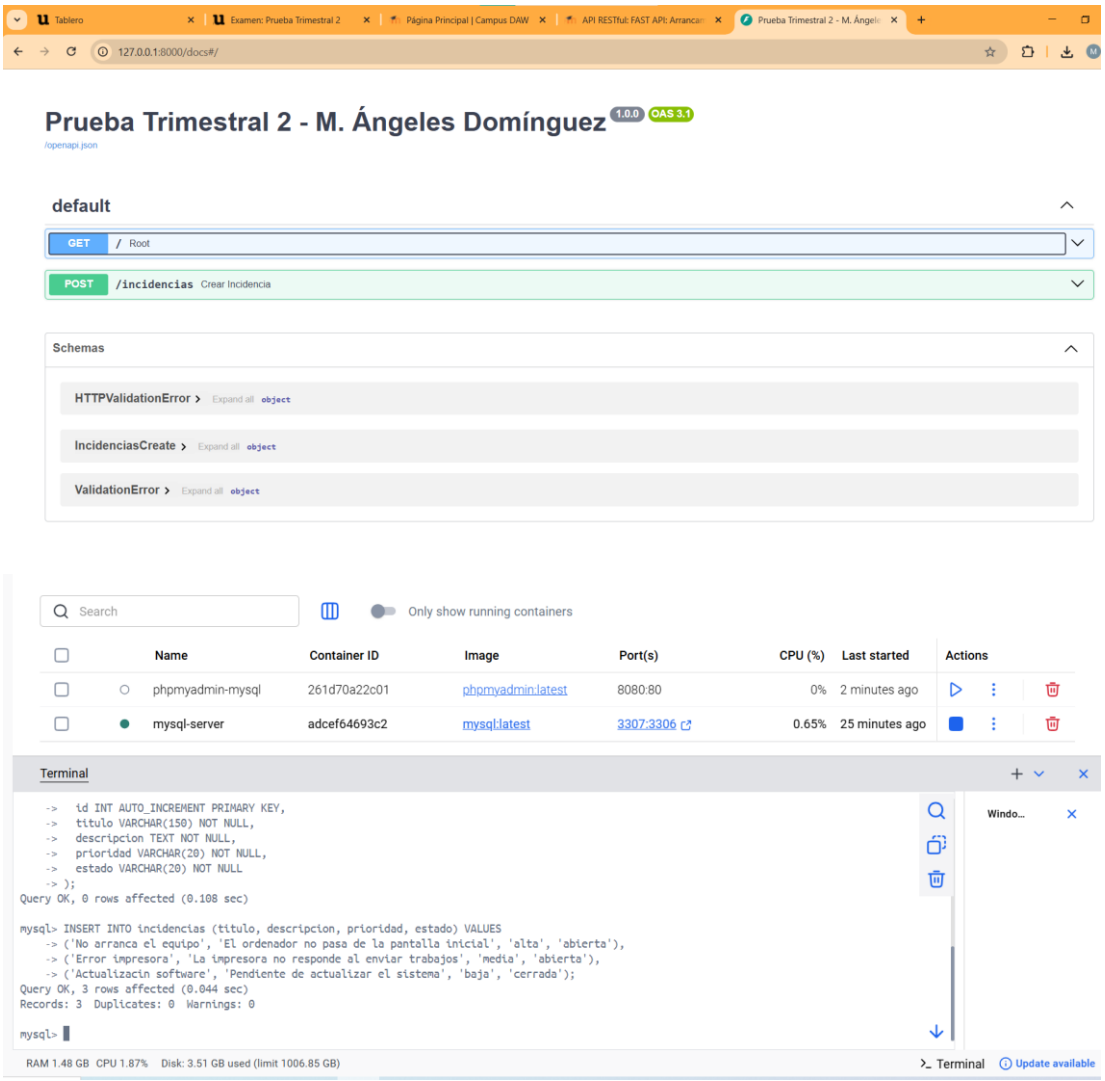
- Breve descripción del proyecto desarrollado.
- Capturas de pantalla de la documentación Swagger (/docs) mostrando los endpoints creados.
- Captura del endpoint de login y del uso del token JWT.
- Captura del acceso correcto a un endpoint protegido mediante JWT.

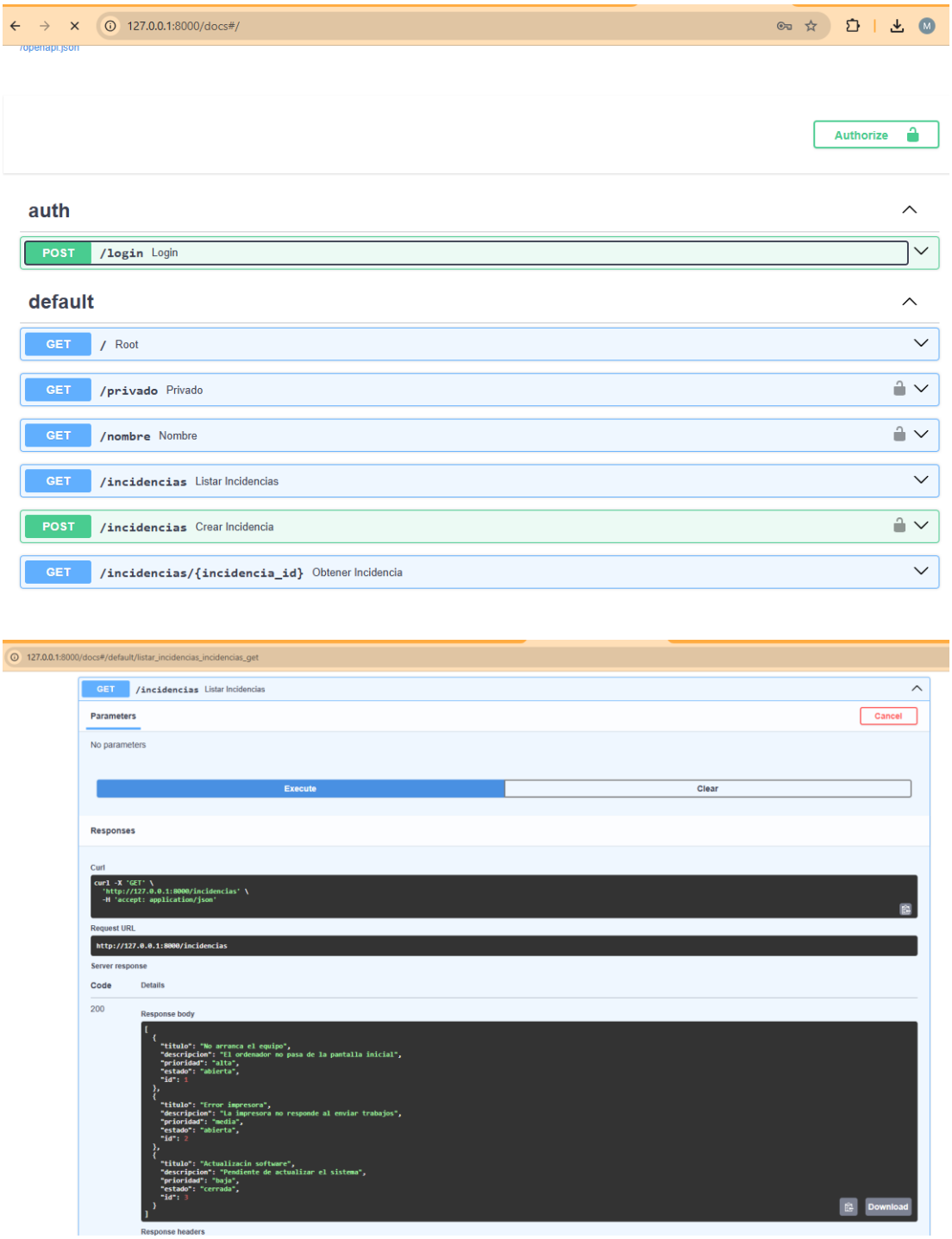
2. Proyecto en GitHub. Se deberá entregar el enlace a un repositorio público de GitHub que contenga:

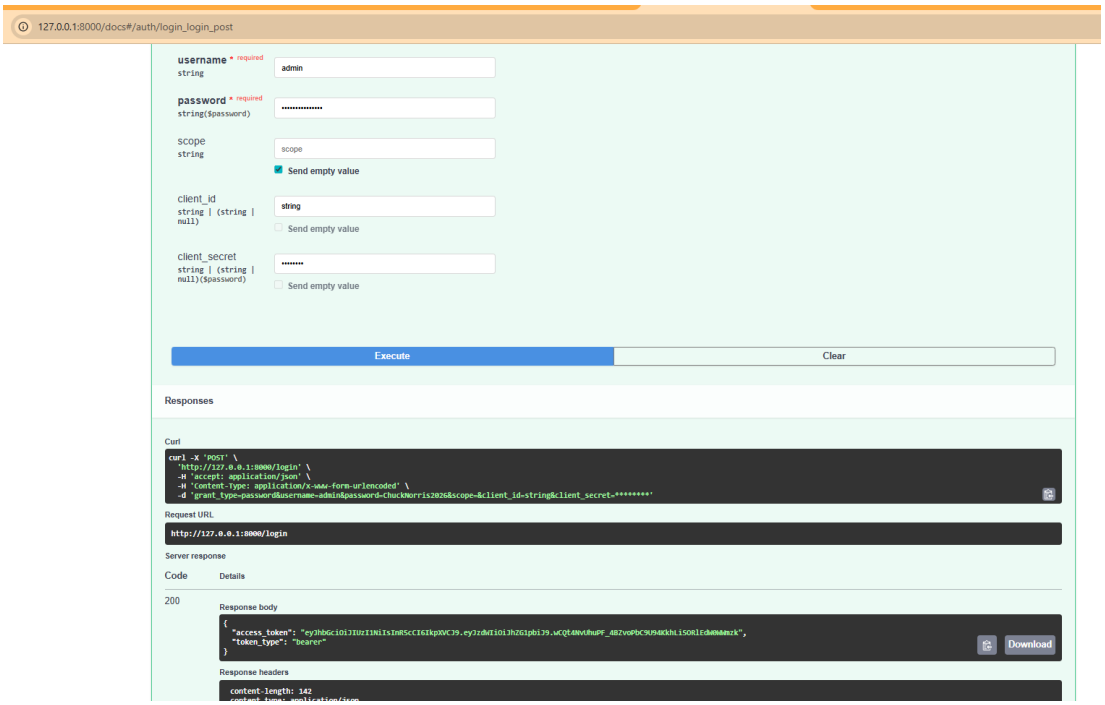
- El proyecto completo de FastAPI.
- Todo el código fuente necesario para ejecutar la aplicación.
- Estructura y archivos trabajados en clase.

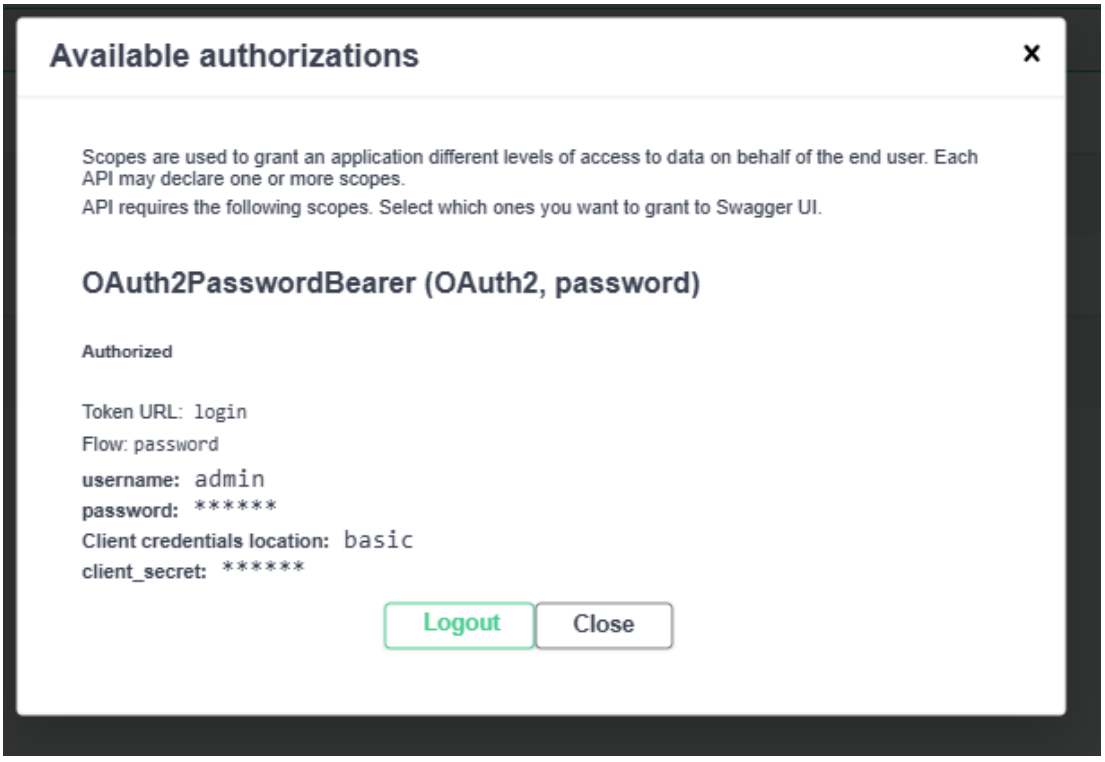
Criterios de evaluación

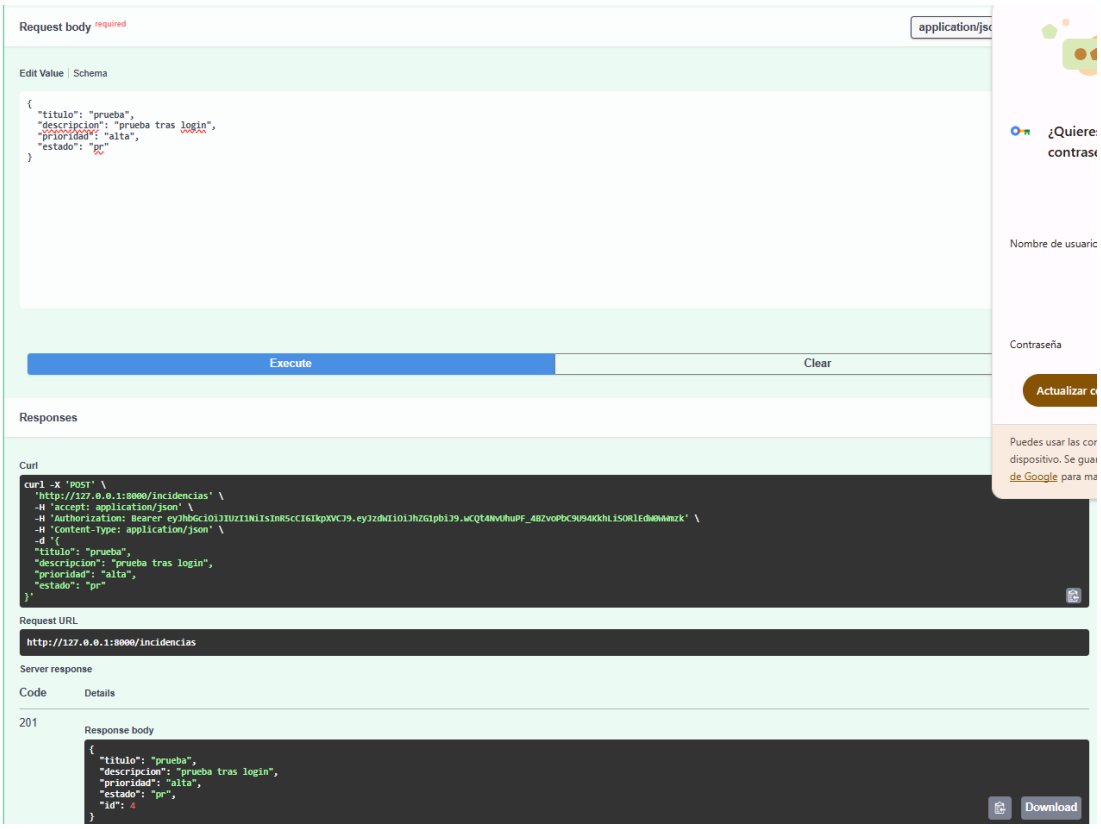
Título de la actividad	Descripción	Peso %
Conexión y uso de MySQL	La API arranca correctamente y muestra la documentación automática	30%
Configuración y uso de MySQL	Correcta conexión a la base de datos y uso del ORM	30%
Autenticación JWT	Implementación del login, generación del token y protección de endpoints	40%
		100 %




127.0.0.1:8000/docs#/
/openapi.json
Authorize
auth
POST /login Login
default
GET / Root
GET /privado Privado
GET /nombre Nombre
GET /incidencias Listar Incidencias
POST /incidencias Crear Incidencia
GET /incidencias/{incidencia_id} Obtener Incidencia
127.0.0.1:8000/docs#/default/listar_incidencias_incidencias_get
GET /incidencias Listar Incidencias
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/incidencias' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/incidencias
Server response
Code
Details
200
Response body
[
{
"titulo": "No arranca el equipo",
"descripcion": "El ordenador no pasa de la pantalla inicial",
"prioridad": "alta",
"estado": "abierta",
"id": 1
},
{
"titulo": "Error impresora",
"descripcion": "La impresora no responde al enviar trabajos",
"prioridad": "media",
"estado": "abierta",
"id": 2
},
{
"titulo": "Actualizacin software",
"descripcion": "Pendiente de actualizar el sistema",
"prioridad": "baja",
"estado": "cerrada",
"id": 3
}
]
Download
Response headers


127.0.0.1:8000/docs#/auth/login_login_post
username * required
string
admin
password * required
string($password)
scope
string
Send empty value
client_id
string | (string | null)
string
Send empty value
client_secret
string | (string | null)($password)
Send empty value
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/login' \
-H 'accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-d 'grant_type=password&username=admin&password=ChuckNorris2026&scope=&client_id=string&client_secret=********'
Request URL
http://127.0.0.1:8000/login
Server response
Code
Details
200
Response body
{
"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJhZG1pbiJ9.wCQt4NVtUhuPF_4BZvoPbC9U94KkhLiSORlEdW0Wmzk",
"token_type": "bearer"
}
Download
Response headers
content-length: 142


Available authorizations
Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.
OAuth2PasswordBearer (OAuth2, password)
Authorized
Token URL: login
Flow: password
username: admin
password: ******
Client credentials location: basic
client_secret: ******
Logout
Close



git@github.com:mangelesdt/Python.git