Autenticación de APIs con Flask

1. Traballaremos sobre o código de tokens.

```
from flask import Flask, jsonify
from flask_jwt_extended import create_access_token, JWTManager
import os
from detenv import load_dotenv
import datetime
from flask_cors import CORS

load_dotenv()

app = Flask(__name__)

app.config['JSON_AS_ASCII'] = False
CORS(app)

app.config["JWT_SECRET_KEY"] = os.getenv('SECRET_KEY')
jwt = JWTManager(app)

def obter_payload():
    payload = {
        'user':'pepe',
        'rol':'ADMIN'
    }
    return payload

@app.post("/token")
def crear_token():
    token_config = {
        'payload': 'pepe',
        'exp':datetime.datetime.utcnow() + datetime.timedelta(hours=1)
    }
    token = create_access_token(token_config)
    return jsonify({"token":token})

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

2. Añadiremos unha importación máis ao código.

```
from flask import Flask, jsonify, request, make_response

from flask_jwt_extended import create_access_token, JWTManager, jwt_required
```

3. Añadimos unha nova petición POST

```
@app.post("/login")
def login():
    username = request.json['username']
    password = request.json['password']

    if username == "pepe" and password == "12345":
        return crear_token()
    else:
        respuesta = make_response({"error-401":"Error de autenticación"})
        respuesta.status_code = 401
        return respuesta
```

4. Añadimos unha nova petición GET

```
@app.get("/admin")
@jwt_required()
def dashboard():
    return "Este es el panel del administrador y para poder entrar en el voy a necesitar
```

5. Diriximonos a Postman e para probalo no Type de Authorization cambiamos JWT Bearer e enviamos a petición.