

## JWT - JSON Web Token

Antes de empezar, debemos tener todas las vistas del caso de música, completas. Recuerda que debías completar las vistas y crear las rutas o endpoints. Si no lo realizaste, debes estar atento de realizar este paso a paso completo:

Debemos instalar el jwt a través del siguiente comando:  
`pip install Flask-JWT-Extended` . ¿Para realizarlo que pasos debemos tener presente? Escribe a continuación tu respuesta y toma captura de pantalla de la terminal:

Archivo vista.py

```
1  from flask import request
2  from ..modelos import db, Cancion, CancionSchema, Usuario, UsuarioSchema, Album, AlbumSchema
3  from flask_restful import Resource
4  from sqlalchemy.exc import IntegrityError
5  from flask_jwt_extended import jwt_required, create_access_token
6
```

```
7  cancion_schema = CancionSchema()
8  usuario_schema = UsuarioSchema()
9  album_schema = AlbumSchema()
10
11
12  class VistaCanciones(Resource):
13
14      def post(self):
15          nueva_cancion = Cancion(titulo=request.json["titulo"], minutos=request.json["minutos"], segundos=request.json["segundos"], interprete=request.json["interprete"])
16          db.session.add(nueva_cancion)
17          db.session.commit()
18          return cancion_schema.dump(nueva_cancion)
19
20      def get(self):
21          return [cancion_schema.dump(ca) for ca in Cancion.query.all()]
22
23  class VistaCancion(Resource):
24
25      def get(self, id_cancion):
26          return cancion_schema.dump(Cancion.query.get_or_404(id_cancion))
27
28      def put(self, id_cancion):
29          cancion = Cancion.query.get_or_404(id_cancion)
30          cancion.titulo = request.json.get("titulo", cancion.titulo)
31          cancion.minutos = request.json.get("minutos", cancion.minutos)
32          cancion.segundos = request.json.get("segundos", cancion.segundos)
33          cancion.interprete = request.json.get("interprete", cancion.interprete)
34          db.session.commit()
35          return cancion_schema.dump(cancion)
```

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

```
36
37     def delete(self, id_cancion):
38         cancion = Cancion.query.get_or_404(id_cancion)
39         db.session.delete(cancion)
40         db.session.commit()
41         return '', 204
42
43 class VistaLogin(Resource):
44     def post(self):
45         u_nombre = request.json["nombre"]
46         u_contrasena = request.json["contrasena"]
47         usuario = Usuario.query.filter_by(nombre=u_nombre, contrasena = u_contrasena).all()
48         if usuario:
49             return {'mensaje': 'Inicio de sesión exitoso'}, 200
50         else:
51             return {'mensaje': 'Nombre de usuario o contraseña incorrectos'}, 401
52
53
54 class VistaSignIn(Resource):
55
56     def post(self):
57         nuevo_usuario = Usuario(nombre=request.json["nombre"], contrasena=request.json["contrasena"])
58         token_de_acceso = create_access_token(identity=request.json['nombre'])
59         db.session.add(nuevo_usuario)
60         db.session.commit()
61         return {'mensaje': 'Usuario creado exitosamente', 'token_de_acceso': token_de_acceso}
62
63     def put(self, id_usuario):
64         usuario = Usuario.query.get_or_404(id_usuario)
65         usuario.contrasena = request.json.get("contrasena", usuario.contrasena)
66         db.session.commit()
67         return usuario_schema.dump(usuario)
68
```

```
68
69     def delete(self, id_usuario):
70         usuario = Usuario.query.get_or_404(id_usuario)
71         db.session.delete(usuario)
72         db.session.commit()
73         return '', 204
74
75 class VistaAlbumsUsuario(Resource):
76
77     @jwt_required()
78     def post(self, id_usuario):
79         nuevo_album = Album(titulo=request.json["titulo"], anio=request.json["anio"], descripcion=request.json["descripcion"], medio=request.json["medio"])
80         usuario = Usuario.query.get_or_404(id_usuario)
81         usuario.albumes.append(nuevo_album)
82
83         try:
84             db.session.commit()
85         except IntegrityError:
86             db.session.rollback()
87             return 'El usuario ya tiene un album con dicho nombre', 409
88
89         return album_schema.dump(nuevo_album)
90
91     @jwt_required()
92     def get(self, id_usuario):
93         usuario = Usuario.query.get_or_404(id_usuario)
94         return [album_schema.dump(al) for al in usuario.albumes]
95
```

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

```
95
96 class VistaCancionesAlbum(Resource):
97
98     def post(self, id_album):
99         album = Album.query.get_or_404(id_album)
100
101         if "id_cancion" in request.json.keys():
102
103             nueva_cancion = Cancion.query.get(request.json["id_cancion"])
104             if nueva_cancion is not None:
105                 album.canciones.append(nueva_cancion)
106                 db.session.commit()
107             else:
108                 return 'Canción errónea', 404
109         else:
110             nueva_cancion = Cancion(titulo=request.json["titulo"], minutos=request.json["minutos"], segundos=request.json["segundos"], interprete=request.json["interprete"])
111             album.canciones.append(nueva_cancion)
112             db.session.commit()
113             return cancion_schema.dump(nueva_cancion)
114
115     def get(self, id_album):
116         album = Album.query.get_or_404(id_album)
117         return [cancion_schema.dump(ca) for ca in album.canciones]
118
119 class VistaAlbum(Resource):
120
121     def get(self, id_album):
122         return album_schema.dump(Album.query.get_or_404(id_album))
123
```

```
123
124
125     def put(self, id_album):
126         album = Album.query.get_or_404(id_album)
127         album.titulo = request.json.get("titulo", album.titulo)
128         album.anio = request.json.get("anio", album.anio)
129         album.descripcion = request.json.get("descripcion", album.descripcion)
130         album.medio = request.json.get("medio", album.medio)
131         db.session.commit()
132         return album_schema.dump(album)
133
134     def delete(self, id_album):
135         album = Album.query.get_or_404(id_album)
136         db.session.delete(album)
137         db.session.commit()
138         return '', 204
139
```

Responde:

¿Qué observaste en el código? Explica con tus palabras

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

Archivo app.py

```
1 from flask import create_app
2 from flask_restful import Api
3 from .modelos import db
4 from .vistas import VistaCanciones, VistaCancion, VistaSignIn, VistaLogin, VistaAlbum, VistaAlbumsUsuario, VistaCancionesAlbum
5 from flask_jwt_extended import JWTManager
6
7 app = create_app('default')
8 app_context = app.app_context()
9 app_context.push()
10
11 db.init_app(app)
12 db.create_all()
13
14 api = Api(app)
15 api.add_resource(VistaCanciones, '/canciones')
16 api.add_resource(VistaCancion, '/cancion/<int:id_cancion>')
17 api.add_resource(VistaSignIn, '/signin')
18 api.add_resource(VistaLogin, '/login')
19 api.add_resource(VistaAlbumsUsuario, '/usuario/<int:id_usuario>/albumes')
20 api.add_resource(VistaAlbum, '/album/<int:id_album>')
21 api.add_resource(VistaCancionesAlbum, '/album/<int:id_album>/canciones')
22
23 jwt = JWTManager(app)
24
```

¿Qué observas en el código? Explica los cambios que observas y además explica que crees que realiza el código nuevo.

Probemos cómo funciona con postman:

Debes abrir una cuenta en el siguiente link: <https://www.postman.com/>

Y además deberás instalar el agente desktop

Cuando ya te abra la ventana, realiza lo siguiente:

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

The screenshot shows a REST client interface. At the top, a request bar is highlighted with a blue oval, containing the method **GET** and the URL `http://127.0.0.1:5000/canciones`. To the right of the URL bar is a blue **Send** button. Below the request bar are tabs for **Params**, **Authorization**, **Headers (6)**, **Body**, **Pre-request Script**, **Tests**, and **Settings**. The **Params** tab is active, showing a table for Query Params:

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Below the tabs are sections for **Body**, **Cookies**, **Headers (5)**, and **Test Results**. The **Body** section is active, showing a JSON response in the **Pretty** view:

```
1 {
2   "albumes": [],
3   "id": 1,
4   "titulo": "Prueba",
5   "minutos": 3,
6   "segundos": 35,
7   "interprete": "Maneski"
8 }
9
10
```

At the top right of the response area, the status is **200 OK** with a response time of **392 ms** and a size of **269 B**. There is also a **Save as example** button.

Observa que debes escoger el método https a probar y luego colocar el endpoint. Posteriormente dar clic en send o enviar. Y en la parte de abajo aparece el resultado de lo que está en la base de datos.

Ahora vamos a crear un usuario. Abrimos una nueva pestaña

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://127.0.0.1:5000/signin
- Body (JSON):**

```
{  "nombre": "Isaura",  "contrasena": "12345"}
```
- Response (JSON):**

```
{  "mensaje": "Usuario creado exitosamente",  "token_de_acceso": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsIm1hdCI6MTczMTM3ODIyMSwianRpIjoibNDMyODlmYjMtMzRmMi00YTJiLWZ0TctNWJlZDkzYj1jMTA3IiwidHlwZSI6ImFjY2VzcyIsInN1bGkiOiI6IklzYXVyYSIsIm5iZiI6MTczMTM3ODIyMSwiZXhwIjoxNzYxMzQ1MTI5fQ.wQb5gd-vlVNo9mdAiYDNR20B2yyXqbdMWRPDSSjVn4w"}
```

¿Qué observas? Explica con tus palabras.

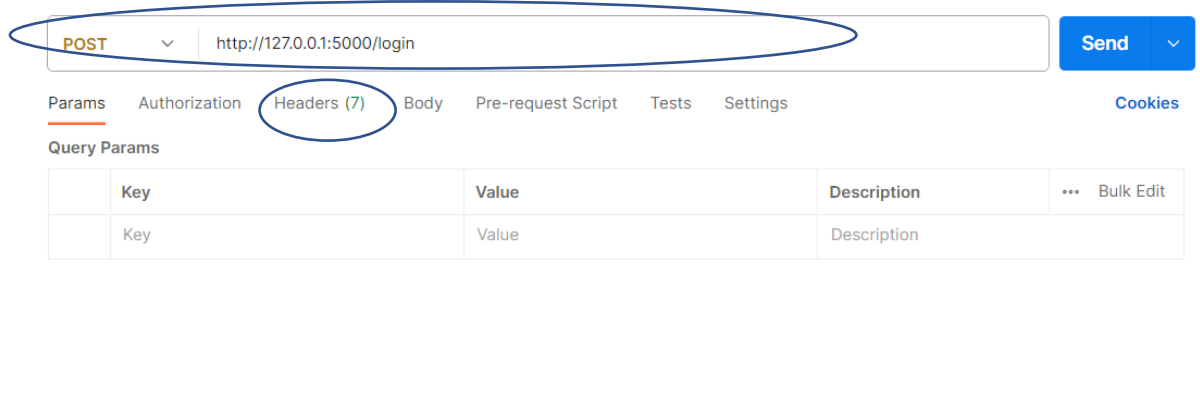
Ahora vamos a probar a que el usuario ingresa al sistema:

Abrimos una nueva pestaña:

Escogemos el método https según nuestro código: post

Colocamos la url: <http://127.0.0.1:5000/login>

*Servicio Nacional de Aprendizaje - SENA*  
*Practica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*



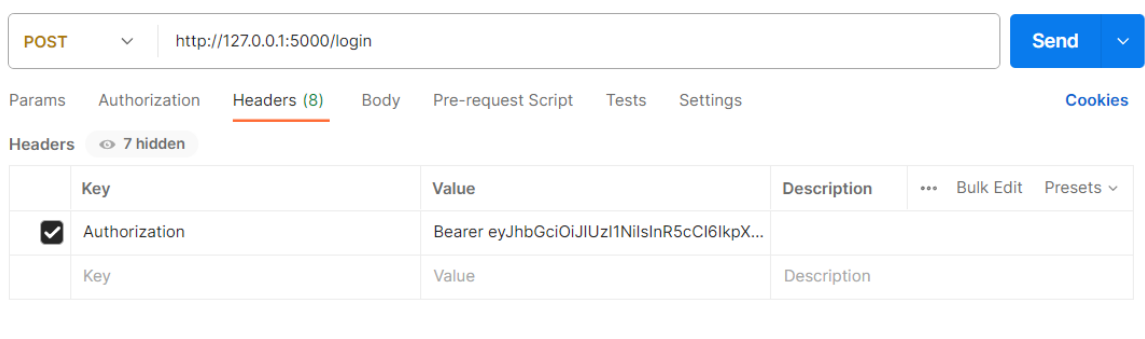
POST http://127.0.0.1:5000/login Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

En el header, configuramos lo siguiente:



POST http://127.0.0.1:5000/login Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX...				
	Key	Value	Description			

En el key colocamos: Authorization

Y en value, colocamos: Bearer <token que genero la creación del usuario>

Ahora en el body colocamos los datos de acceso del usuario que creamos y le damos enviar.

*Servicio Nacional de Aprendizaje - SENA*  
*Práctica Api*  
*Caso de Estudio*  
*Instructora Isaura Suarez*

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://127.0.0.1:5000/login
- Body:** A JSON object with the following structure:

```
1 {
2   "nombre": "Isaura",
3   "contrasena": "12345"
4 }
```
- Response:** A 200 OK status with a response body of:

```
1 {
2   "mensaje": "Inicio de sesión exitoso"
3 }
```
- Headers:** 5 headers are listed.
- Test Results:** 200 OK, 36 ms, 211 B.
- Buttons:** Send, Beautify, Save as example, and various view toggles (Pretty, Raw, Preview, Visualize, JSON).

Dee esta manera vamos probando cada uno de los endpoints de nuestra api.

Ahora, Explica con tus palabras y de acuerdo a la práctica que observaste, ¿qué es el JWT y cómo funciona?