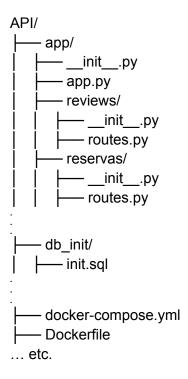
Blueprints

La estructura de la API quedaría gracias a los directorios.



Resumen

Los archivos __init__.py

Los archivos __init__.py son para que Python reconozca los directorios. (O sea las carpetas app, reviews, users, reservas, etc.):

- Con esto no tendremos problemas al meter los imports en el archivo app.py.
- Pueden estar comentados o vacíos, pero si no lo ponen las request pueden verse afectadas. (Ya probado con Postman, son necesarios).

Que necesito para armar mis endpoints.

A saber: cuando nuestra tarjeta requiera una tabla en base de datos, vamos a armar entonces un directorio con ese nombre.

Por ejemplo, tengo una tabla reviews, voy a necesitar un directorio reviews para mis endpoints.

Pasos:

- PRIMER PASO Creo un directorio nuevo (carpeta, folder) dentro de API/app

 SEGUNDO PASO: Creo en mi directorio un archivo __init__.py y un archivo routes.py.

 TERCER PASO: Ahora voy a trabajar sobre mi archivo routes.py para meter mis endpoints. Supongamos que quiero borrar una reviews de la base, entonces en mi archivo se verá un endpoint para borrar reviews por id, y el archivo se vera algo asi:

```
# ---Mis imports ---
from flask import Blueprint, jsonify, request, current_app
from sqlalchemy import text
from sqlalchemy.exc import SQLAlchemyError

# — Blueprint para mis reviews ---
reviews_bp = Blueprint('reviews', __name__)

# ---En las Rutas para mis reviews—
# ---Empiezo definiendo mi endpoint —
@reviews_bp.route('/<id>', methods=['DELETE'])
```

- Lo que introduce Blueprint aca:
- El import del objeto Blueprint y de current_app.
 Si se fijan en app.py, la app tiene guardada en la variable engine la cadena que nos permite conectarnos a la base de datos:

```
engine =
create_engine('mysql+mysqlconnector://app_user:****@**/')
app.config['engine'] = engine
```

Con **current_app** obtenemos de la configuración actual los datos para establecer la conexion asi:

```
engine = current_app.config('engine')
conexion = engine.connect()
```

reviews_bp = Blueprint('reviews', __name__)
 reviews_bp es nuestra instancia de Blueprint, esta nos permitirá
 definir nuestra ruta al estilo que veníamos haciendo con @app.route.

Solamente que esta vez debemos mandarle 2 valores por parámetros,

- el nombre del módulo __name___
- el nombre del Blueprint o como llamaremos a nuestra vista.

El nombre del Blueprint es importante y debe ser único, si se repite tira error. En este caso es 'reviews'.

Se utiliza principalmente para dos cosas:

1- Generación de URLs: Flask utiliza el nombre del Blueprint para generar URLs. Por ejemplo,

```
@reviews_bp.route('/list', etc.)
def mi_funcion():
    return "Lista de reviews."
```

para generar la URL para la función de vista **mi_funcion** en el Blueprint **'reviews'**:

url for('reviews.mi funcion')

- 2- Legibilidad: Esto es lo que más nos importa a nosotros.
- ÚLTIMO PASO. Una vez hecha nuestra hoja de rutas, queda importarlo y registrarlo en el archivo app.py

En los imports, para este ejemplo

from .reviews.routes import reviews_bp

Lo registro junto a los demas:

app.register_blueprint(reviews_bp, url_prefix='/reviews')

El prefijo del Blueprint es importante, la ruta para las request la define el prefijo:

Si registran el Blueprint con

entonces el prefijo para todas las rutas en ese Blueprint será '/reviews'.

Por lo tanto, la ruta @reviews_bp.route('/iist') sería accesible a través de la URL

'http://localhost:5000/reviews/list'.

Si registran el Blueprint con

seria

'http://localhost:5000/moncho/list'