Informe

Marco Antonio Ortíz Ramírez

Servicio Nacional de Aprendizaje (SENA)

Centro de Electricidad, Electrónica y Telecomunicaciones

Análisis y Desarrollo de Software

Ficha N° 2558104

Instructora Isaura María Suarez Novoa

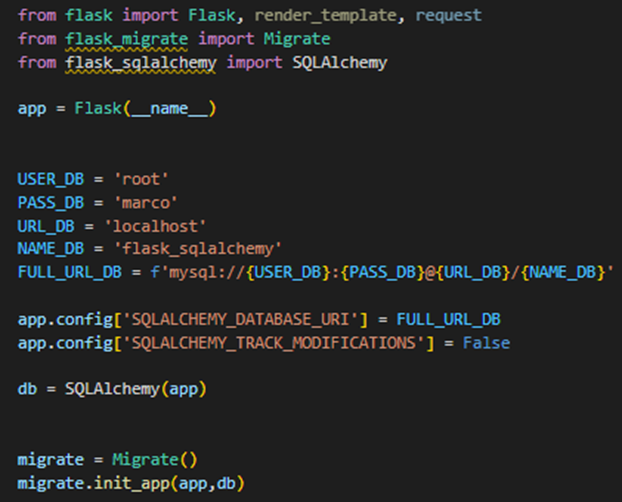
Bogotá D.C.

18 de agosto del 2023

Código



En el código anterior, lo configuré para postgresql. Tú deber es analizarlo y configurarlo para mysql. Pega aquí como quedo tu configuración.



Ahora colocamos las siguientes líneas de código



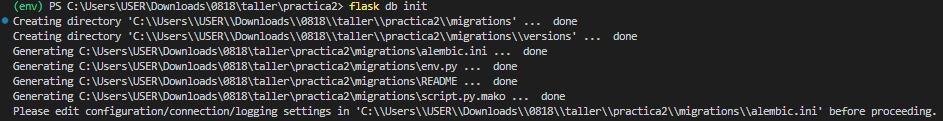
Ahora en la terminal, ejecuta el siguiente comando:



Revisa tu carpeta de proyecto y responde:

1. **¿Qué sucedió?**

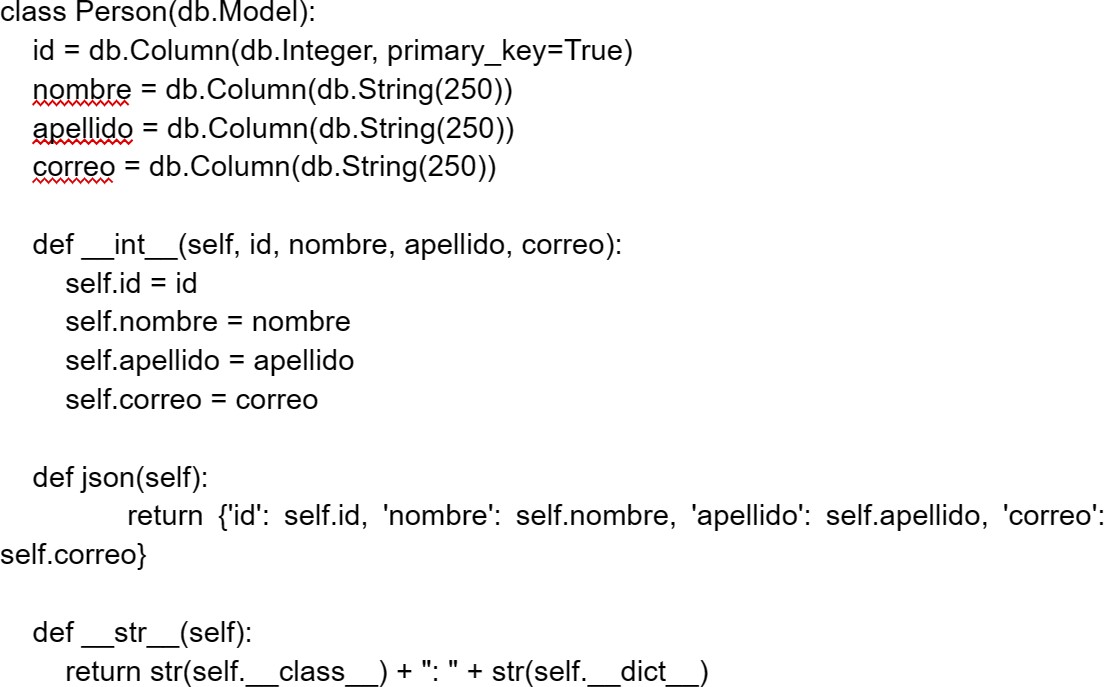
Salió lo siguiente en la consola:



Y se creó a su vez una carpeta en el proyecto, llamada “migrations”, con cosas dentro, como lo son archivos llamados: alembic.ini, env.py, un README, script.py.mako, y dos carpetas: Una llamada “\_\_pycache\_\_” y otra “versions”

Buen trabajo. ¡¡¡¡¡Continuemos!!!!!

Seguimos en el archivo app.py escribiendo líneas de código:



Responde:

**¿Qué crees tú que hace esa parte del código escrito anteriormente?**

Creo que hace algo que tiene que ver con, de alguna forma “crear” una tabla, con atributos; así como se crea un objeto con un primer “set” de los datos para luego pasarse a otro diccionario el cual sinceramente no entiendo del todo. Pero la idea general que tengo, es que aquel código puede llegar a ser algún tipo de archivo que se encargará de crear tablas en la base de datos sin necesidad de crearlas nosotros como programadores, pero manejando una sintaxis muy propia).

Bien, ahora en la terminal escribe el siguiente comando:



**Revisa la base de datos que creaste en worbench (no se te olvide actualizar) y comenta que pasó**

Lo que sucedió fue que en la base de datos se creó una tabla llamada alembic\_version

Bien, Ahora revisa la carpeta migrations de tu proyecto. Dentro de ella existe una carpeta llamada versions. Abre el archivo py.

**¿Explica que ves en ese archivo?. Sé muy explicito**

Lo primero que sale es como una descripción del archivo, hay un id de revisión (que es el mismo nombre del archivo), y está la fecha de creación del archivo.

Después se encuentran dos import, uno referenciando a “alembic” y otro que solamente importa un elemento que ya hemos usado (sqlalchemy), y le agrega un sobrenombre.

Luego sale más información describiendo algo, tanto del archivo como cosas que creería tienen que ver con el proyecto que tengamos y qué cosas lo compongan que sean importantes.

Luego se crea una función que complementa y lleva a mayor profundidad lo implementado dentro del archivo app.py, ya que parece que creara uno a uno los campos, haciéndolo con una sintaxis parecida a la usada al crear tablas con sentencias DDL (primero se pone el nombre de la columna, luego el tipo de dato y por último si podrá ser nulo o no).

Luego se crea una función que, al leer por encima el nombre que tiene y la función a la que se invoca dentro de ella, pareciera que es el fragmento de código encargado de crear la tabla en la base de datos.

Ahora, en la terminal escribe el siguiente comando:



1. **Revisa la base de datos en worbench (no se te olvide actualizar el esquema) y explica lo que sucedió.**

Lo que ocurrió fue que se creó una tabla en la base de datos, solo al haber corrido el comando flask db upgrade.

Según lo que entiendo, entonces, en app.py se hace como la primera instancia o llamada a la creación de la base de datos; al ejecutar el comando “flask db init” se crea la carpeta “migrations” que es como si fuera el espacio de conexión entre Python y la base de datos; luego, al ejecutar el comando “flask db migrate” se inicializa esa conexión y se enlaza con el archivo app.py para que, posteriormente, al llamar a la función upgrade, se cree correctamente la tabla procesada anteriormente dentro de la base de datos.

1. **Con base a la práctica, explica con tus palabras que es un ORM**

Un ORM es un sistema que le permite a Python (y por ende a formularios por parte de un frontend) generar una conexión con una base de datos, de tal forma que el mismo lenguaje (python) pueda interpretar los datos que le llegan del frontend, procesarlos, y posteriormente enviarlos a la base de datos sin necesidad de algún intermediario aparte del mismo lenguaje; aspecto que lo hace increíblemente útil ya que no se necesita hacer conversiones de datos en algún punto, ni usar la estructura que normalmente se usaría para crear una relación con una base de datos (sentencias DDL).

1. **Crea un repositorio y sube la practica1 (con su debida explicación. Fotos con el paso a paso) y la practica 2 (foto con el paso a paso, inluyendo foto de la terminal).**

**Practica 1:**

Primero creamos una carpeta en la que guardaremos nuestro proyecto, luego abrimos la terminal y escribimos “Python -m venv env”:



Así creamos nuestro proyecto.

Luego lo que tenemos que hacer es, escribir en la consola: “env\Scripts\activate”; así creamos nuestro entorno virtual:

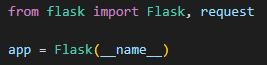


Luego, debemos ejecutar el siguiente comando, el cual instalará flask, que podríamos decir son un conjunto de módulos que le permiten a Python obtener mejores funcionalidades:



Posteriormente, creamos dentro de la carpeta del proyecto general, un archivo llamado app.py, en el que guardaremos nuestro programa.

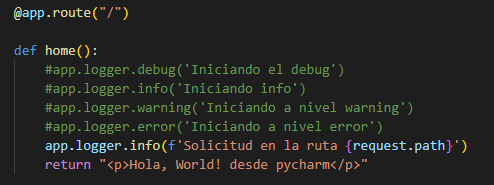
Allí escribiremos lo siguiente:

****

Este código permite importar de lo que podríamos llamar una dependencia o archivo del esqueleto del proyecto, una propiedad, llamada flask (que anteriormente instalamos dentro del proyecto).

Luego, creamos la ruta base o inicial del proyecto.

Hay que tener en cuenta crear el app.route que especifica que es en la ruta “/”. También es importante escribir la línea que dice app.logger.info, ya que con ella que “obtiene” la ruta a la que se referencia, y por último, la línea que será la forma de corroborar que todo funciona como debe:

****

Luego creamos otras dos rutas, las cuales recibirán, cada una, un dato en la URL y se usará luego para imprimir en pantalla una información (en la primera un saludo, en la segunda la edad).

**Texto

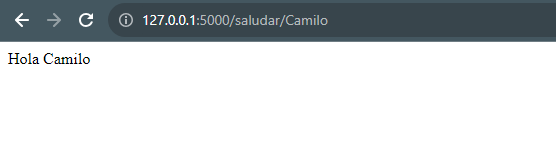
Descripción generada automáticamente**

Por último, tenemos la simulación de una api (comprendiendo el concepto de que recibe cierta información que es guardada en un objeto):

**Texto

Descripción generada automáticamente**

Por último, ya habiendo completado el código, escribimos en la terminal “flask --debug run”, para correr el código y visualizar lo que se supone que hace:

****

**Practica 2:**

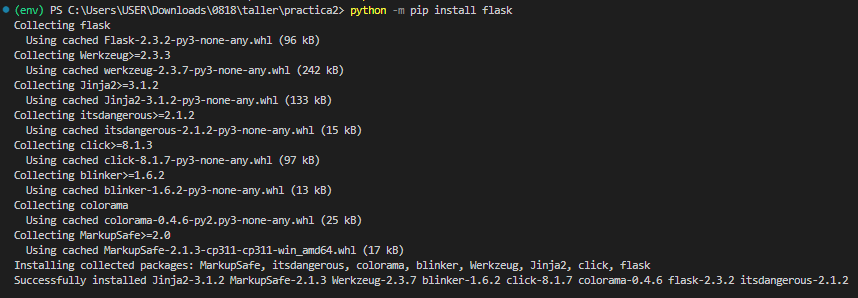
Primero, hacemos los mismos primeros pasos de la anterior practica (creamos la carpeta, ejecutamos el comando “Python -m venv env” y luego el comando “env\Scripts\activate”).



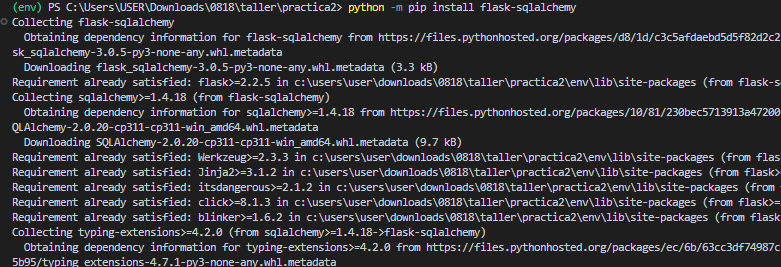
Luego de haber hecho lo anterior, pasamos ahora si a lo interesante:

En la terminal ejecutamos los comandos:

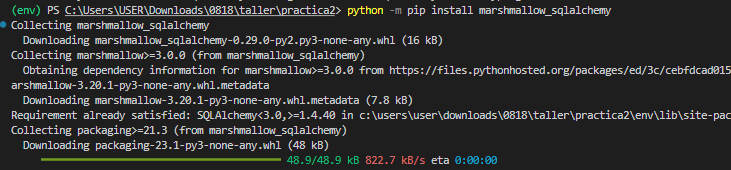
* Python -m pip install flask



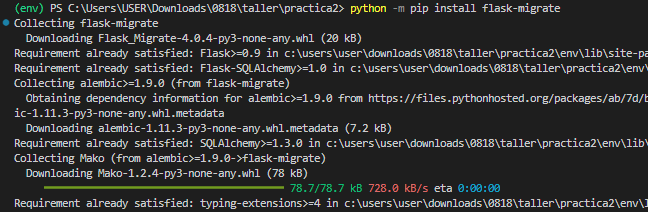
* Python -m pip install flask-sqlalchemy



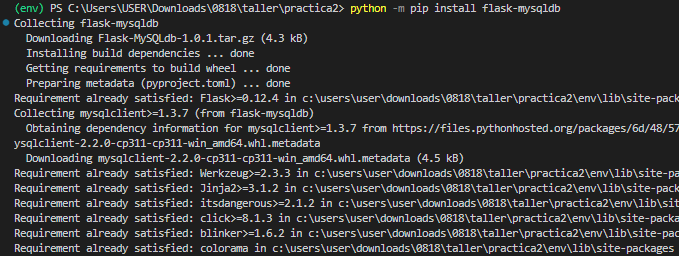
* Python -m pip install marshmallow\_sqlalchemy



* Python -m pip install flask-migrate



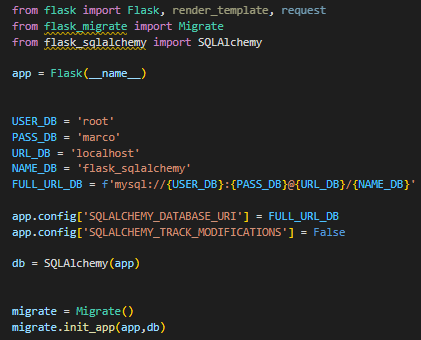
* Python -m pip install flask-mysqldb



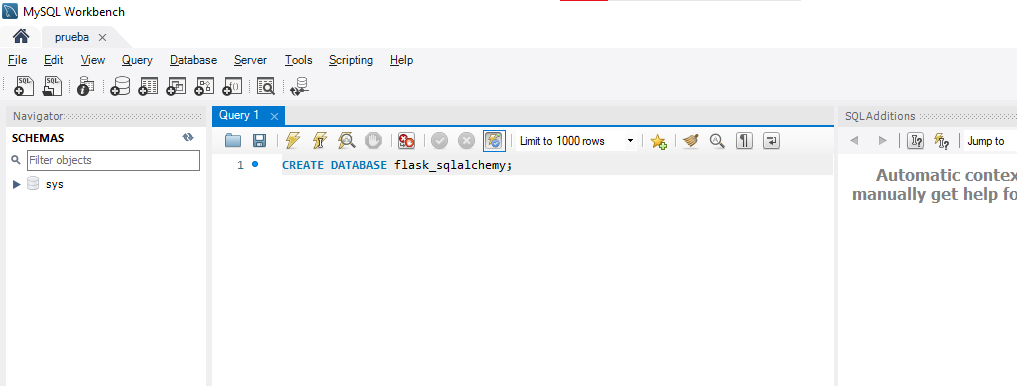
Con lo anterior ya tendremos todo lo necesario para crear nuestra aplicación. Empecemos:

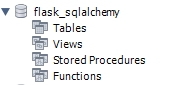
Creemos un archivo llamado app.py, dentro de nuestro proyecto, para alojar nuestro programa.

Luego de ello, creamos lo que al inicio de este archivo se decía y se sobrescribió para que funcionara con MySQL.

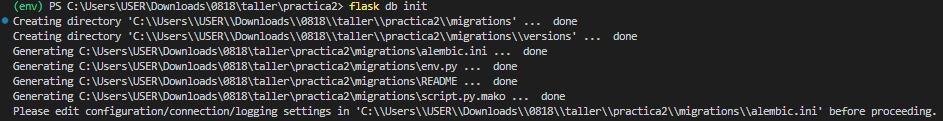


Ahora creamos una instancia en workbench y luego creamos una base de datos (en este caso se llama “flask\_sqlalchemy”):

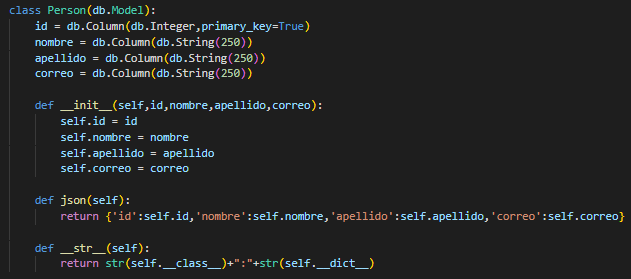




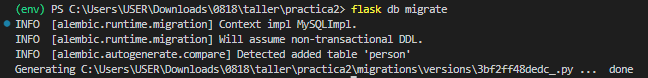
Ahora, ejecutamos en la consola: “flask db init”. Lo cual nos va a crear la carpeta “migrations” dentro de nuestro proyecto:

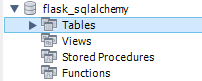


Ahora, dentro del archivo app.py, escribimos lo siguiente:

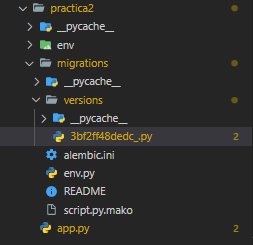


Ahora, volvemos a la terminal y escribimos “flask db migrate”, con lo que se agregará una tabla a la base de datos creada:

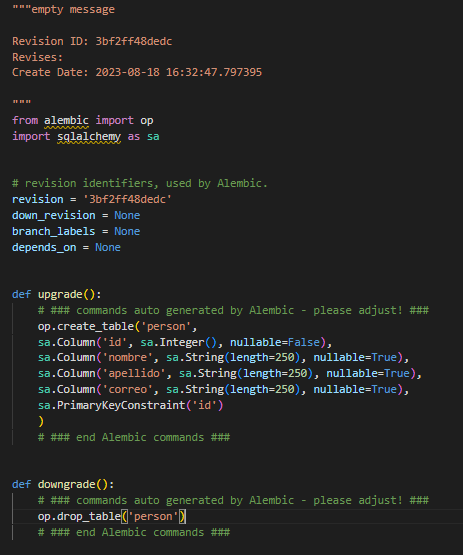




Así como se creará un nuevo archivo dentro de la carpeta versions dentro de la carpeta migrations de nuestro proyecto:

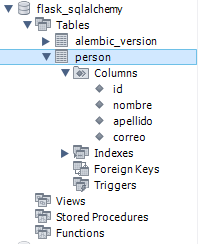


Y allí dentro habrá algo parecido a lo siguiente:



Ahora, en la terminal, ejecutamos el comando “flask db upgrade”. Con eso lo que haremos será crear una tabla llamada Person en la base de datos con los datos que se habían inicializado (recordemos que, precisamente la clase creada en app.py se llamaba Person, y los datos inicializados allí son los mismos que se existen ahora como campos en la base de datos. Recordando que al ejecutar el comando que termina en migrate, lo que se hizo fue algo parecido a llevar a un archivo de versión, lo que se tenía en ese archivo app.py para luego ser directamente añadido a la base de datos):





1. **Ingresa a territorium y adjunta este archivo con las preguntas resueltas, además del link del repositorio**

**🗸**