Actividad Entorno de Desarrollo Virtual

Juliana Tique Ortiz

16/06/2025

Verificación de la versión Instalada

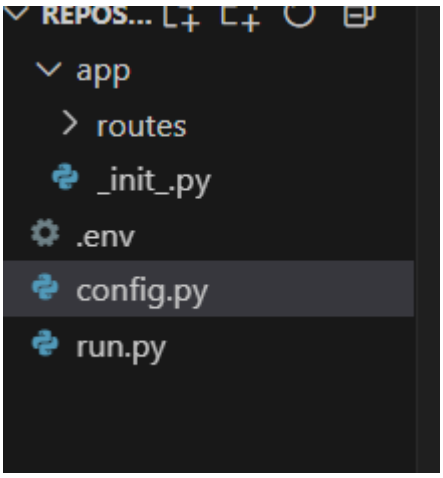
- Verificar la versión instalada de Python de la siguiente manera en el cmd ($ Python-versión)
- Creamos una carpeta y la abrimos en **Visual Studio Code**
- Dentro de la misma abrimos el terminal de **VS**, Seleccionamos la opción de la consola "Git Bash" en el icono+
- Primero que todo creamos el entorno Virtual llamado "Vemv", El cmd para créalo es el Siguiente (Python-M Venv "NombreDelEntorno")
- Para activar el entorno virtual debomos ejecutar el siguiente comando (**Source "**NombreDelEntorno**"/Scripts/Activate**)
- Al momento de activarse debería del aparecer el nombre de entorno entre () **EJ: (**vemv**)**

Actividad Repostería

Primero abrimos worbench y creamos una instancia y ingresamos el script

```sql
-- Crear la base de datos
CREATE DATABASE reposteria_db; -- Usar la base de datos
USE reposteria_db; -- Crear la tabla 'roles'
CREATE TABLE roles (
id INT AUTO_INCREMENT PRIMARY KEY,
role_name VARCHAR(50) NOT NULL UNIQUE
); -- Crear la tabla 'usuarios'
CREATE TABLE usuarios (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL UNIQUE,
password VARCHAR(255) NOT NULL,
role_id INT,
FOREIGN KEY (role_id) REFERENCES roles(id)
); -- Crear la tabla 'productos'
CREATE TABLE productos (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
price DECIMAL(10, 2) NOT NULL
); -- Insertar roles iniciales (opcional)
INSERT INTO roles (role_name) VALUES ('Administrador'), ('Vendedor'), ('Repostero');
```

Creamos una carpeta que va a almacenar diferentes archivos para el proyecto en Visual Studio Code



En el Archivo.env

Secret_key: es el nombre de un ajuste con un valor de **clave_muy_segura** este es una especie de contraseña o código secreto

Localhost: es un valor que significa que la base de datos(Donde el programa guarda su información)

Root: Es el nombre del usuario que el programa usa para entrar a la base de datos

Password(‘’): Significa que no hay contraseña para el usuario de la base de datos

Reposteria_db: El nombre de la base de datos en donde se conectará

Archivo config.py

Import os : Es un modulo de la biblioteca de Python, este proporciona una interacción con el sistema operativo. Su uso es para acceder a las variables de entorno (son leídas del archivo.env)

From dotenv import load:dotenv:

Dotenv: Es una biblioteca de pip

Load_dotenv: Es una función especifica de esa biblioteca. Esta se encarga de leer el archivo .env y carga las variables que contienen en el "Entorno" Python acede a las variables como si fueran del sistema

Load_dotenv(): esta es una lina clave que busca el archivo .env en la carpeta actual y carga todas las variables definidas allí como (**SECRET_KEY, DB_HOST)** ahora se pueden utilizar las variables

Class config= define una clase llamda config en programación orientada a objetos. Agrupa todas las cofiguraciones de la aplicación dentro de una clase config para mantenerlas organizadas y fáciles de acceder desde cualquier parte del programa

SECRET_KEY = os.getenv("SECRET_KEY") (y las líneas siguientes para DB_HOST, DB_USER, DB_PASSWORD, DB_NAME):

os.getenv("NOMBRE_DE_LA_VARIABLE"): Esta es la parte más importante

os es un modulo el que importamos al principio

GETenv()Es una función del modulo Os que se usa para obtener el valor de una variable de entorno

Archivo run.py

from app import create_app: importa una function llamada create_app desde otra parte del código app. Esta función sabe como crear y configurar el proyecto

app = create_app(): Ejecuta esa función y guarda la aplicación ya creada en una variable llamada app.

if __name__ == '__main__': Esta es una condición estándar en Python que significa “Si este Archivo (run.py) es el que está ejecutando directamente (y no es un módulo importado por otro archivo)”

app.run(debug=True): Entonces inicia el servidor web de desarrollo para app.run(). El debug=true muestra errores detallados y recargados automáticos cuando se hacen cambios

Archivo auth.py

from flask import Blueprint: Importa Blueprint, que es como un "mini-aplicación" o un "plano" para organizar rutas y funcionalidades de forma modular en Flask.

auth_bp = Blueprint('auth', __name__): Crea este “Plano” y le da el nombre ‘auth’ este agrupa todas las rutas que definas aquí bajo un prefijo común, haciéndolas más organizadas.

@auth_bp.route('/'): Esta es un ‘decorador’ que le dice a flask:” Cuando alguien visiste la url base de este plano (/ en este caso), ejecutar la función de abajo”

def home(): Define una función llamada “Home” Esta función es lo que es ejecuta cuando se accede a la ruta definida arriba.

return "Bienvenido al sistema de repostería": Esta función simplemente devuelve un mensaje de texto es lo que vera el usuario en su navegador.

Archivo_init_.py

Importaciones iniciales: Trae las herramientas principales:

Flask: El "cerebro" de tu aplicación web.

Config: Tus ajustes (como la base de datos) del archivo .env que vimos antes**.**

mysql.connector: La herramienta para conectar tu aplicación a una base de datos MySQL.

def create_app():: Define una función que se encarga de construir y configurar tu aplicación. Todo lo que sigue dentro de esta función se ejecuta cuando llamas a create_app().

app = Flask(__name__): Crea una nueva instancia para el proyecto Flask. app es ahora

app.config.from_object(Config): Le dice al proyecto que use los ajustes que se definieron en la clase Config (del archivo .env). Así, la aplicación sabe dónde está la base de datos, etc.

app.config['MYSQL_CONNECTION'] = mysql.connector.connect(...): Aquí el proyecto se conecta a la base de datos MySQL. Usa los datos (host, user, password, database) que leyó de tu Config (es decir, del .env). Una vez conectada, guarda esa conexión para usarla más tarde.

from app.routes.auth import auth_bp: Importa el "Blueprint" (el grupo de rutas) de autenticación que vimos en el archivo auth.py.

app.register_blueprint(auth_bp): Le dice al proyecto principal (app) que "registre" o incluya las rutas y funciones que están definidas en el Blueprint de autenticación (auth_bp). De esta forma, las rutas de auth.py forman parte de tu aplicación principal.

return app: La función create_app devuelve la aplicación Flask ya configurada y lista para funcionar.