

Manual de Instalación y Uso del Software

Integración de Flask con PostgreSQL

Introducción:

Este documento detalla el procedimiento para la instalación, configuración y uso de una aplicación web desarrollada con **Flask (Python)** y **PostgreSQL** como sistema de gestión de bases de datos. Está dirigido a profesionales del desarrollo de software, pero ha sido redactado con un enfoque didáctico que también permite su comprensión por desarrolladores con experiencia básica o intermedia.

⚠ Importante: siempre utilizar “Command Prompt” y no la “BASH” de la terminal. Además, Este proyecto incluye un entorno virtual preconfigurado llamado `.prueba`, que ya contiene todas las dependencias necesarias. Si el propósito es utilizar directamente el sistema, **no es necesario reinstalar dependencias ni crear un nuevo entorno virtual**.

1. Gestión del Entorno Virtual

El entorno virtual es una herramienta fundamental en Python para aislar las dependencias de un proyecto, evitando conflictos con otros entornos o aplicaciones instaladas globalmente.

◆ 1.1 Crear el entorno virtual (solo si estás creando el software desde cero)

```
py -3 -m venv .prueba
```

Este comando crea un entorno virtual llamado `.prueba` en el directorio actual.

◆ 1.2 Activar el entorno virtual (para trabajar en el proyecto)

```
.prueba\Scripts\activate
```

Activa el entorno virtual `.prueba`, haciendo que todos los paquetes se instalen y utilicen dentro de él.

◆ 1.3 Desactivar el entorno virtual

```
deactivate
```

Finaliza la sesión del entorno virtual y retorna al intérprete global de Python.

◆ 1.4 Reactivar el entorno virtual

```
.prueba\Scripts\activate
```

Este comando puede utilizarse cada vez que se desee volver a trabajar en el entorno.

✓ **Nota:** Para usuarios que desean utilizar el sistema sin modificarlo, **solo es necesario activar el entorno .prueba**. No deben crearse nuevos entornos virtuales ni reinstalar dependencias.

2. Instalación de Dependencias (solo en caso de crear el software desde cero)

Este paso **no es necesario** si se trabaja directamente en el entorno .prueba, ya que las dependencias requeridas ya están instaladas. En caso de crear un entorno virtual nuevo, se deben instalar los siguientes paquetes:

♦ 2.1 Flask (framework principal)

```
pip install flask
```

♦ 2.2 Conector para PostgreSQL

```
pip install psycopg2-binary
```

Este paquete permite conectar la aplicación Flask con PostgreSQL.

♦ 2.3 Flask-SQLAlchemy (ORM)

```
pip install flask-sqlalchemy
```

Facilita el trabajo con bases de datos mediante mapeo objeto-relacional (ORM).

♦ 2.4 Flask-Migrate (migraciones)


```
pip install flask-migrate
```

Permite manejar cambios estructurales de la base de datos.

♦ 2.5 Flask-WTF (gestión de formularios)

```
pip install flask-wtf
```

Facilita la implementación de formularios web seguros y eficientes.

 **Requisito obligatorio:** El archivo requirements.txt incluye todas las dependencias del proyecto y permite una instalación automatizada con:

```
pip install -r requirements.txt
```

3. Configuración y Migraciones de la Base de Datos

Antes de aplicar migraciones, es indispensable configurar la base de datos y los parámetros de conexión.

♦ 3.1 Creación de la base de datos en PostgreSQL

Desde PostgreSQL (por ejemplo, utilizando pgAdmin o la línea de comandos), crear una base de datos con el siguiente nombre:

```
supermarket
```

Guardar los datos de acceso, ya que deben configurarse en app.py:

```
# Configuración de conexión en app.py
```

```
USER_DB = 'postgres'
```

```
USER_PASSWORD = '2025'
```

```
SERVER_DB = 'localhost'
```

```
NAME_DB = 'supermarket'
```

◆ 3.2 Inicializar migraciones (solo la primera vez)

```
flask db init
```

Este comando crea la carpeta migrations/ donde se almacenarán los scripts de migración.

! Este paso **ya fue ejecutado** en el entorno .prueba. No es necesario repetirlo a menos que se cree un entorno nuevo desde cero.

◆ 3.3 Crear una nueva migración

```
flask db migrate
```

Este comando detecta cambios en los modelos y genera el archivo de migración correspondiente.

◆ 3.4 Aplicar las migraciones

```
flask db upgrade
```

Ejecuta las migraciones, actualizando la base de datos para reflejar los modelos definidos.

4. Desarrollo y Modo Debug

◆ 4.1 Habilitar modo depuración

```
set FLASK_DEBUG=True
```

Activa la depuración para facilitar el desarrollo, mostrando errores detallados en el navegador.

◆ 4.2 Definir el entorno como desarrollo

```
set FLASK_ENV=development
```

Permite que la aplicación se reinicie automáticamente ante cualquier cambio en el código fuente.

5. Verificación y Ejecución del Proyecto

◆ 5.1 Verificar sincronización con la base de datos

```
flask db stamp head
```

Marca el estado actual de la base de datos como sincronizado con la última migración.

◆ 5.2 Ejecutar el servidor

```
flask run
```

Inicia el servidor local en `http://localhost:5000` y pone la aplicación en funcionamiento.

Consideraciones Finales

- Si vas a utilizar el entorno virtual **ya configurado (.prueba)**, simplemente actívalo con:

```
.prueba\Scripts\activate
```

- Si decides crear un nuevo entorno virtual desde cero para experimentar, debes **eliminar o renombrar la carpeta .prueba** antes de iniciar el proceso. Mantener ambas versiones puede provocar conflictos en las migraciones de base de datos.
- Asegúrate de contar con PostgreSQL 17 instalado, y haber creado la base de datos supermarket antes de aplicar migraciones.
- Todas las dependencias necesarias están listadas en `requirements.txt`. Este archivo es obligatorio y forma parte de los requerimientos del sistema.