Preguntas abp1 M7

repositorio del modulo 7 : https://github.com/el-javier/m7_individuales

¿Cuáles son las bases de datos soportadas por Django y en qué se diferencian?

SQLite: Es el motor de base de datos predeterminado de Django. Se almacena como un archivo en el sistema de archivos, lo que lo hace adecuado para aplicaciones más pequeñas o en desarrollo. Es ligero y no requiere una configuración de servidor de base de datos separado.

PostgreSQL: Es un potente sistema de gestión de bases de datos relacional y de código abierto. Ofrece una amplia gama de características avanzadas, como soporte para índices, transacciones ACID, replicación y consultas complejas.

MySQL / MariaDB: Estos son sistemas de gestión de bases de datos relacionales. Ambos son adecuados para aplicaciones de diversos tamaños.

Oracle: Es un sistema de gestión de bases de datos relacional de uso empresarial. Oracle es conocido por su escalabilidad, confiabilidad y capacidad para manejar grandes volúmenes de datos. Es una opción común para aplicaciones empresariales y de misión crítica.

Microsoft SQL Server: Es un sistema de gestión de bases de datos relacional desarrollado por Microsoft. Proporciona una amplia gama de características empresariales y se integra bien con el ecosistema de Microsoft. Es una opción popular para aplicaciones que se ejecutan en entornos de Windows.

- ¿Qué es una migración en Django y para qué se utiliza?

Una migración es una forma de gestionar cambios en la estructura de la base de datos de una aplicación, como cuando necesites realizar cambios en los modelos de datos, como agregar un campo, eliminar una tabla o modificar una columna. Las migraciones de Django te permiten realizar estos cambios de manera controlada y mantener la consistencia entre la definición de los modelos y la estructura de la base de datos.

Las migraciones de Django se definen en archivos Python que contienen instrucciones para modificar la estructura de la base de datos. Cada migración representa un conjunto de cambios atómicos y se registra en el sistema de control de versiones de tu proyecto. Cuando ejecutas las migraciones, Django se encarga de aplicar los cambios necesarios en la base de datos, creando o modificando tablas, índices y restricciones según las especificaciones definidas en las migraciones.

Las migraciones en Django se utilizan para controlar y gestionar cambios en la estructura de la base de datos de una aplicación web. Proporcionan portabilidad, control de versiones y seguridad en las actualizaciones, lo que facilita el desarrollo y mantenimiento de la aplicación a largo plazo.

- ¿Cuál es la diferencia entre usar consultas SQL y consultas ORM en Django?

La principal diferencia entre las consultas SQL y las consultas ORM de Django radica en el lenguaje utilizado, el nivel de abstracción, la portabilidad y la legibilidad del código. Las consultas SQL brindan un mayor control y están más orientadas a optimizar consultas específicas y utilizar características avanzadas del motor de base de datos. Por otro lado, las consultas ORM de Django ofrecen una capa de abstracción más alta, facilitando el desarrollo, la portabilidad y el mantenimiento del código. La elección entre ambos enfoques depende de los requisitos específicos de tu proyecto y de tus preferencias personales.

- ¿Cómo se instalan los paquetes de base de datos en Django y cuál es su importancia?

En Django, la instalación y configuración de los paquetes de base de datos depende del motor de base de datos que deseas utilizar. Los pasos generales a seguir para instalar y configurar los paquetes de base de datos en Django tomando de ejemplo Postgres
Por ejemplo, si deseas utilizar PostgreSQL, deberás instalar el paquete psycopg2 para interactuar con la base de datos PostgreSQL.

Puedes utilizar herramientas como pip (el gestor de paquetes de Python) para instalar los paquetes necesarios. Por ejemplo, puedes ejecutar el siguiente comando en la terminal para instalar psycopg2:

```
pip install psycopg2
```

Configuración en el archivo de configuración de Django: Después de instalar el paquete de base de datos, debes configurar Django para que utilice el motor de base de datos adecuado. Esto se realiza en el archivo **settings.py** de tu proyecto Django.

En la sección DATABASES del archivo settings.py, debes proporcionar los detalles de conexión específicos del motor de base de datos, como nombre de usuario, contraseña, host y nombre de la base de datos. También debes especificar el motor de base de datos que deseas utilizar.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

La importancia de los paquetes de base de datos en Django radica en que permiten la comunicación y la interacción con el motor de base de datos subyacente. Estos paquetes son esenciales para realizar consultas, migraciones y operaciones de CRUD (crear, leer, actualizar y eliminar) en la base de datos.

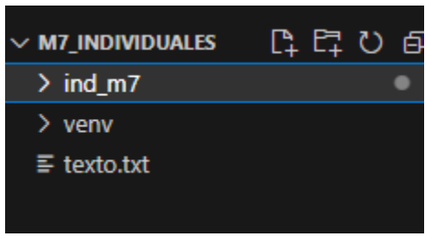
- ¿Qué ventajas ofrece Django como ORM para la integración con una base de datos?

Django como ORM ofrece ventajas significativas para la integración con una base de datos. Proporciona una capa de abstracción, portabilidad, mantenibilidad del código, funciones integradas y migraciones automáticas, lo que simplifica el desarrollo de aplicaciones web y facilita la gestión de la base de datos. Utilizando el ORM de Django, puedes interactuar con la base de datos de manera más eficiente y centrarte en la lógica de tu aplicación en lugar de preocuparte por los detalles específicos del motor de base de datos.

- Facilidad para escribir en el lenguaje que ya se está utilizando, ya que la mayoría de las personas no son expertas en SQL.
- Abstracción del sistema de base de datos, lo que facilita el cambio de un motor de base de datos a otro.
- Funciones avanzadas proporcionadas por el ORM, como transacciones, agrupación de conexiones, migraciones y población de tablas.
- Mejor rendimiento en las consultas, ya que el ORM puede optimizarlas automáticamente

Desarrollo del proyecto

Crear un proyecto en Django, con las aplicaciones que estimes necesarias para abordar el problema *del ejercicio*



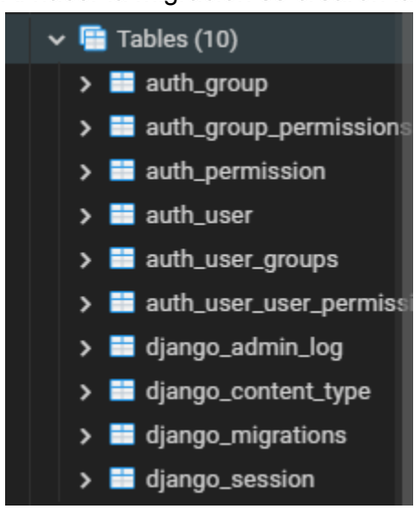
- Conectar el proyecto a una base de datos en PostgreSQL

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'js_m7',
        'USER': 'js_usuario',
        'PASSWORD': '1z2x3c4v.',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```


```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'js_m7',
        'USER': 'js_usuario',
        'PASSWORD': '1z2x3c4v.',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

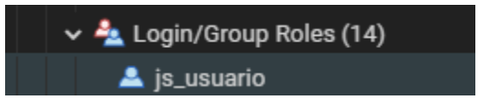

- Realizar la migración inicial, corroborando que ésta se ve reflejada en la base de datos PostgreSQL.

Al hacer la migración se crearon las tablas en postgres

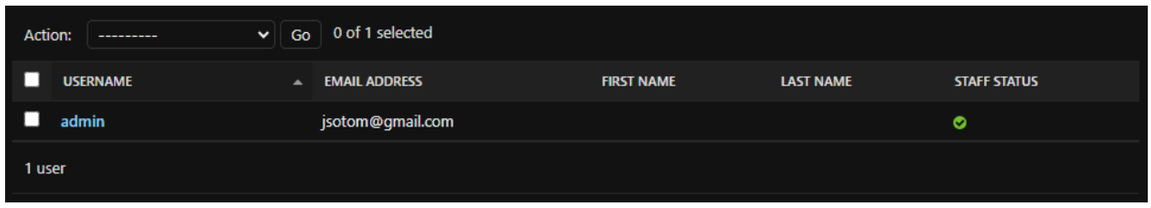


- Definir usuarios de tipo superuser y usuarios del sistema, considerando que todos podrán
interactuar con sus propias tareas.

usuario en postgres



super usuario en django “admin”



- La página principal será una página con el logo e imagen representativa de su aplicación y un link que lleve al Login de usuario.

- Implementar páginas de login y logout de usuarios. Login, deberá llevar a una página simple de Bienvenida, en donde, posteriormente, implementaremos la vista de Listado de Tareas

login

Al estar logueado además del saludo aparece la opción logout

ACCESO A DATOS CON DJANGO

Ejercicio 1 modulo m7

Bienvenido, admin

Cerrar sesión