**Proyecto Blog! [Python]**

Este repositorio contiene todos los materiales e instrucciones para poder realizar el proyecto de blog de programador python.

Este proyecto se acerca al tipo de trabajo y de desafios que tendrán en el curso de Python Django.

Para este proyecto ya cuenta con toda la parte de frontend resuelta, su deber será crear el backend de esta aplicación.

**Objetivo**

El objetivo es construir el backend de una aplicación de blog. El frontend ya se encarga del login del usuario y mostrar la información que provee el backend (enviar información y consultar al backend). El backend deberá:

* Proveer los endpoints que muestren las páginas HTML.
* Proveer los endpoints para la generación de posteos en el blog como también consultar los posteos realizados de un usuario.

Para lograr esto, además de crear todos los endpoints necesarios deberá crear la base de datos para almacenar toda la información (posteos) que la aplicación vaya generando.

En este repositorio cuenta con un video llamada "ejemplo\_funcionando.mp4" en el cual podrá ver como debería funcionar su aplicación una vez concluida.

**Recursos**

* Contará con todos los archivos necesarios de frontend en las carpeta templates y static.

**Como comenzar**

* Deberá crear un archivo "app.py" en el cual colocará todo el código necesario para realizar el backend del proyecto.
* Luego deberá crear el bloque principal if \_\_name\_\_ == "\_\_main\_\_":. Dentro del bloque principal deberá llamar al server:

app.run(host="127.0.0.1", port=5000)

* Deberá incluir la inicialización de su base de datos cuando se construye la aplicación:

# Este método se ejecutará la primera vez

# cuando se construye la app.

with app.app\_context():

# Crear aquí la base de datos

db.create\_all()

print("Base de datos generada")

**Base de datos**

Deberá crear la base de datos SQLite "blog.db". Utilizar SQLAlchemy para crear una clase que responda a la tabla "posteos". Dicha tabla "posteos" debe contener las siguientes columnas:

* id --> número (Integer) (autoincremental, primary\_key)
* usuario --> texto (String) (nombre del usuario que hizo el post)
* titulo --> texto (String) (título del post)
* texto --> texto (String) (texto/contenido del post)

**Endpoints del frontend (HTML)**

Dentro del archivo **app.py** deberá implementar los siguientes endpoints que responderan a las rutas del explorador del usuario:

**Endpoint login (/login)**

Cuando el usuario acceda a esta ruta desde el explorador, este endpoint deberá renderizar (render\_template) el archivo html "login.html"

**Endpoint de bienvenida o index (/)**

Cuando el usuario acceda a esta ruta desde el explorador, este endpoint deberá renderizar (render\_template) el archivo html "blog.html"

**Endpoints del backend (APIs)**

Dentro del archivo **app.py** deberá implementar los siguientes endpoints que responderán las peticiones GET / POST / etc:

**Endpoint post (/posteos/)**

Dentro de este endpoint deberá aceptar peticiones del tipo "GET" y del tipo "POST". Este endpoint recibe en la URL el nombre del usuario por parámetro. Deberá capturar el valor de "usuario" en la función del endpoint.

Para cada tipo de petición deberá realizar:

**Endpoint post (/posteos/) para peticiones GET**

Cuando este endpoint sea invocado por GET, el frontend le enviará en el endpoint el nombre del usuario logeado, luego:

* Deberá solicitar los posteos en la base de datos filtrados por ese usuario, y devolver los últimos (usar order\_by descendente) tres posteos realizados por ese usuario (limit = 3)
* De cada posteo obtenido extraer "titulo" y "texto" almacenando esos datos en un diccionario. Ej: {"titulo": ..... , "texto": .....}
* Cada diccionario creado de esos posteos se deberá almacenar en una lista llamada datos.
* Esa lista finalmente tendrá tres diccionarios, con los datos de esos tres posteos.
* Al finalizar deberá retornar la variable datos:

return jsonify(datos)

**Endpoint post (/posteos/) para peticiones POST**

Cuando este endpoint sea invocado por POST, el frontend le enviará los datos del posteo escrito (titulo y texto) en los parámetros de un formulario en "request.form".

* Deberá obtener el usuario enviado en el endpoint.
* Deberá obtener el titulo y texto de "request.form".
* Con esos datos deberá crear un nuevo posteo en la base de datos.
* Al finalizar deberá indicar que la petición se completó con éxito retornando código status 201:

return Response(status=201)

**Milla extra**

En caso que desee mejorar el sistema puede implementar para el endpoint "/posteos/" la petición DELETE de HTTP.

* Cuando este endpoint sea invocado por DELETE, deberá borrar todos los posteos de ese usuario.