Proyecto Linea de Profundizacion III

Repositorio en github: <https://github.com/AlexNocua/Aplicacion-Web-con-python-Flask->

En este repositorio se va a encontrar el desarrollo de la aplicación web que va a ser desarrollada con la librería flask en Python, este se va a mantener el control de versiones con el fin de cada uno de los integrantes realicen sus respectivos commits, el trabajo espera ser realizado para línea de profundización III.

Integrantes y desarrolladores del proyecto:

Brayan Baquero.

Héctor Briceño

Brayan Cristancho

Alex Nocua

Inicio de desarrollo:

Para evitar los errores en cuanto a los controles de versiones en librerías de Python y la versión del mismo se va a ser la instalación y uso de entornos virtuales.

Comando para instalación de VirtualENV: pip install virtualenv

Creación de entorno virtual: instalado el entorno virtual se crea en la consola de visual studio, cabe resaltar que este entorno virtual tiene que ser instalado en la ubicación donde vayamos a trabajar el proyecto.

En este caso vamos a manejar el proyecto por control de versiones en github.

Para esto se necesitan algunas cosas, tener cuenta de github y descargar e instalar gitbash:



Aplicación para instalar, consola para comandos específicos y control de manejo de proyectos desde escritorio a nube.

Cuando ustedes tengan instalado el gitbash les va a salir lo siguiente:

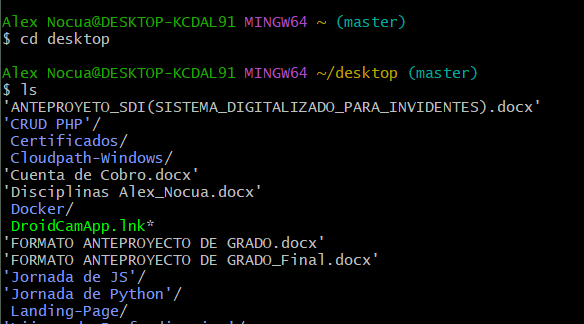
Texto

Descripción generada automáticamente

Consola de comandos, tener en cuenta que cada uno de los comandos para la navegación en la consola son “**Comandos de LINUX**”.

**Comandos básicos:**

* **cd “dirección o nombre de carpeta o archivo”:** El manejo de este es para ir a una ruta, ejemplo de ir a desktop: **cd desktop.**
* **cd .. :** El “**cd** ..” es para retroceder entre carpetas, si tengo una carpeta en el escritorio, ejemplo “**carpetaAlex**” la ruta de la misma se identificaría de la siguiente forma: C:users/desktop/carpetaAlex. Para eso yo quiero retrocedes a desktop, para ello solo ejecuto ese comando.
* **ls :** este comando sirve para ir a todos y cada uno de los elementos que se encuentren en la dirección donde este, ejemplo:



Estos serían los comandos básicos para moverse entre carpetas.

Ahora si seguimos con la clonación del repositorio de github.

**Clonación de repositorio de GitHub**

Para esto lo primero que necesitan hacer es crear una carpeta, donde ustedes quieran.

Cuando ya hayan creado la carpeta van a acceder a ella por medio de GitBash:



Estando ya en la carpeta pueden clonar el repositorio, pero antes de eso tienen que conectar su GitBash con sus cuentas de github, esto se hace con los siguientes comandos:

* git config --global user.name "Nombre de usuario de github sin las comillas"
* git config --global user.email "correo de cuenta de github sin comillas"

después de esto les va a salir de la siguiente manera:



Como el repositorio es publico para clonarlo no es necesario llaves de activación.

Ahora si clonamos el proyecto.

Para esto se utilizan los siguientes comandos:

**git clone** [**https://github.com/AlexNocua/Aplicacion-Web-con-python-Flask-**](https://github.com/AlexNocua/Aplicacion-Web-con-python-Flask-) **:** Este es para bajar y clonar el proyecto almacenado en un repositorio a su

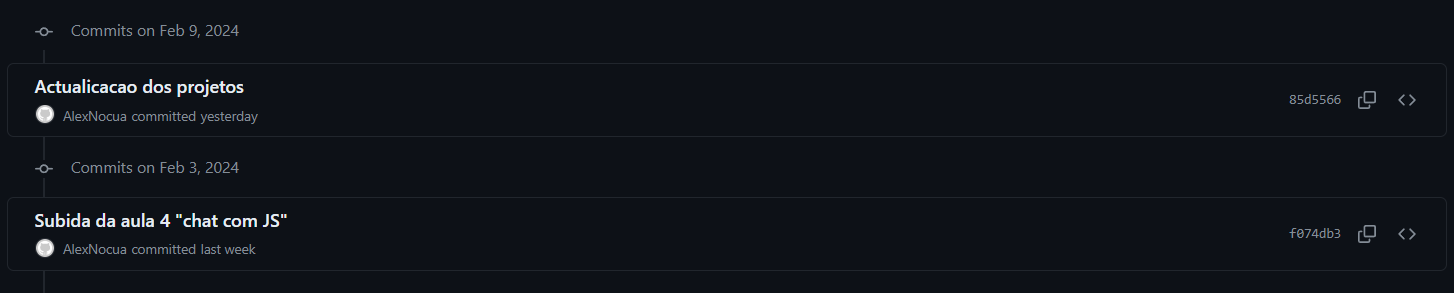
**git init :** Este inicia nuestro control de cambios de github y el mismo repositorio.

Cada cambio que vayan a realizar el mismo vs (Visual Studio) va a mostrar donde realizaron los cambios que hicieron de diferente y demás.

Para subir cambios se utilizan los siguientes comandos:

git add . : Este es para agregar los cambios a un guardado local que ofrece el mismo gitbahs.

git commit -m “” : este es el que va a guardar en un commit cada uno de los cambios realizados en el repositorio, entre las comillas por lo general se coloca una descriocion de los cambios que se hicieron para no perderse en el avance del proyecto, en github esto se va a mostrar de la siguiente manera:



En este se va a mostrar quien realizo los cambios y donde fueron.

Retomando lo anterior clonamos el proyecto dentro de la carpeta que crearon, esta clonación va a crear otra carpeta dentro de ella que es donde se van a almacenar las configuraciones de git para configurar el repositorio, quedando de la siguiente manera:

Texto

Descripción generada automáticamenteClonación

Texto

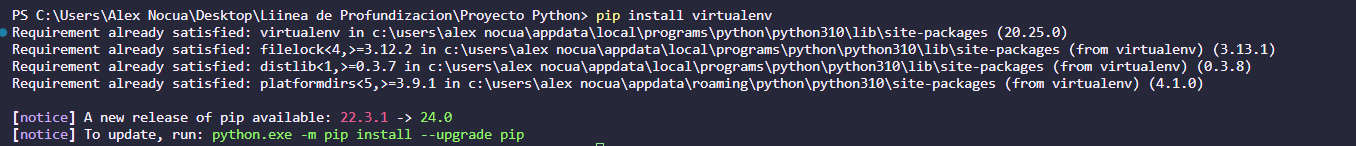
Descripción generada automáticamente

Inicialización del Proyecto, tener en cuneta que tiene que ingresar a la carpeta creada por la clonación.

Ya con esto esa misma carpeta se abre en el visual estudio para la creación del ambiente virtual y la instalación y manejo de cada una de las librerías a usar.

**Instalación de librerías y creación de entorno virtual**

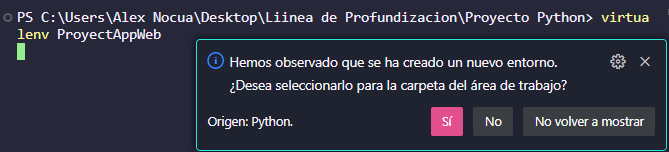
Retomamos al inicio de desarrollo donde se mencionó el comandopip install virtualenv, este va a ser ejecutado en la consola o terminal de visual Studio:



Muestra de instalación de virtual enviroment

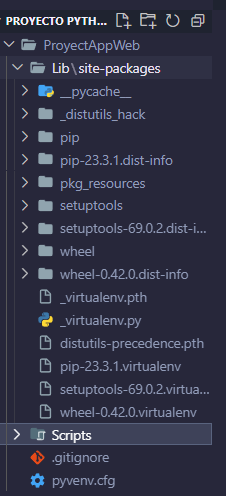
Ahora vamos con **creación del entorno virtual**.

Para esto se coloca el comando **virtualenv NombredelEntorno**, cuando lo creen el mismo visual estudio les va a decir si desea asignar las configuraciones de ese entorno virtual para la carpeta donde esta trabajando ese proyecto, se cliquea si, ¿pero porque?, Python tiene muchas versiones, así mismo tiene librerías disponibles con muchas versiones, esto por lo general se hace para que en ese entorno virtual se manejen versiones especificas de librerías en versiones específicas de Python que eviten la incompatibilidad de las mismas.



Muestra de VS.

Esto va a crear una carpeta dentro de nuestra carpeta de repositorio que va a tener las siguientes informaciones:



Allí es donde se va a manejar y almacenar cada una de las librerías a utilizar **no de forma global.**

Ya creado este entorno virtual ustedes tienen que activarlo dirigiéndose a “**nombredelentornocreado\Scripts\activate**”, digitan esto en la terminal de VS para activarlo.



Cuando ustedes crean el entorno virtual y lo activan si se fijan en la parte de la Dir sale entre paréntesis el entorno virtual, esto que quiere decir, cada librería o a instalar en este proyecto va a ser almacenado en el entorno virtual creado y **no de manera global**.

Comandos extras:

**deactivate**: Desactiva nuestro entorno virtual

**pip list:** Muestra las librerías con sus versiones instaladas.

**pip search nombredelalibreria**: Este es para buscar información de una librería especifica.

**pip uninstall nombredelalibreria**: Funciona para desinstalar una librería.

Hay muchos mas comandos, para instalar una versión especifica o eliminar la misma, configuraciones, etc.

**Instalación de flask**

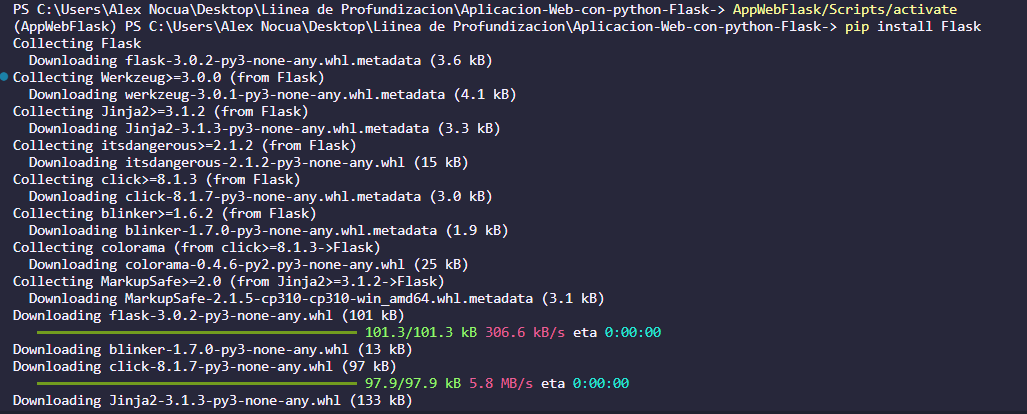
Para esto hay un sitio web donde puedo encontrar los comandos de instalación de cada librería, esta página es:[**https://pypi.org/**](https://pypi.org/)

O podemos buscar directamente Pypi flask em el navegador y nos va a redireccionar a la siguiente página:



Muestra del comando a ejecutar en la terminal de VS.

Ejecución del comando en terminal evidenciando su instalación:



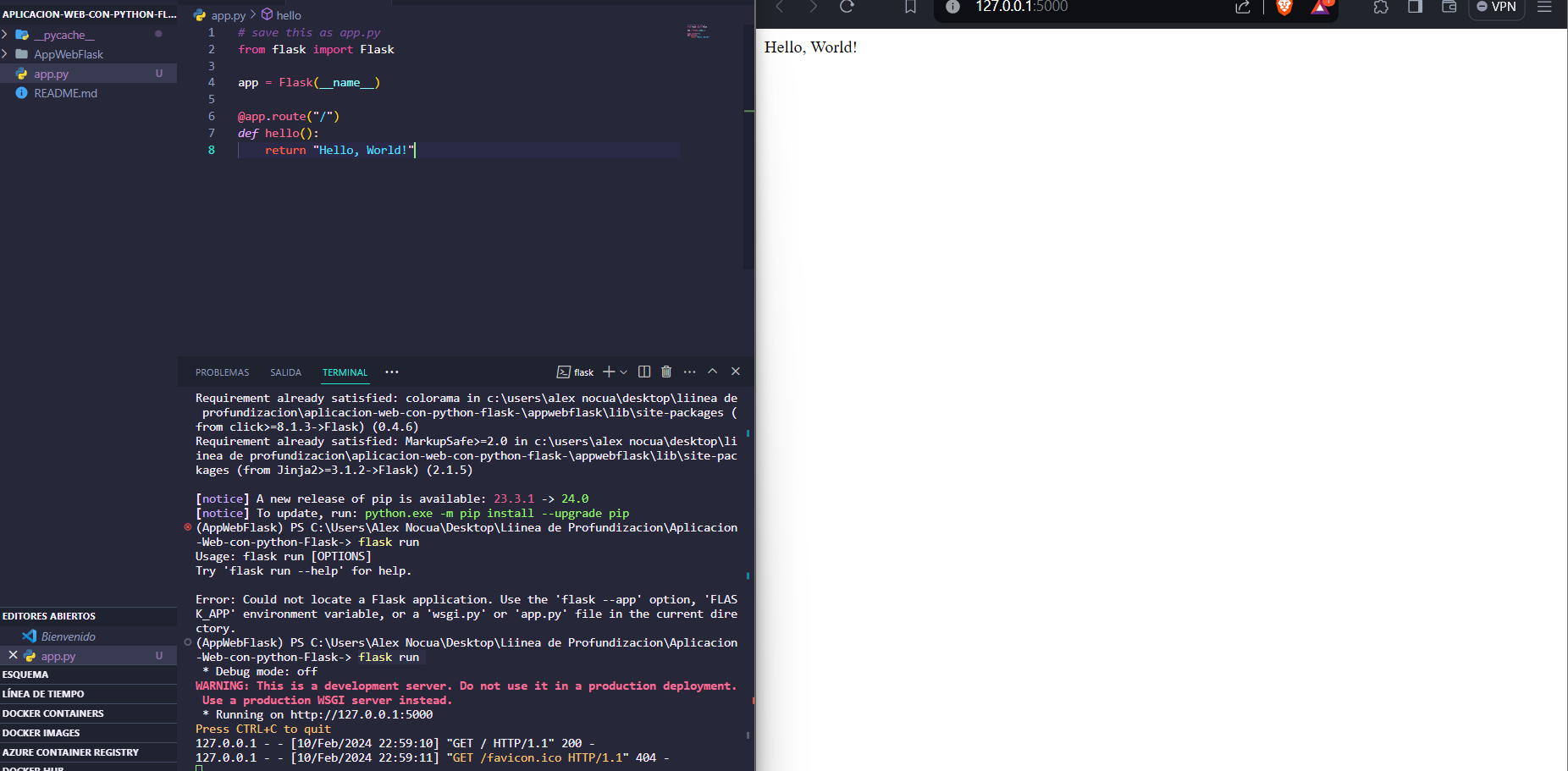
Muestra de instalación.

Ahora se crea un archivo y se siguen las normas según la documentación de flask para mostrar un hola mundo desde un puerto y dirección:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Creación de archivo básico y ejecución por terminal para habilitar puerto.

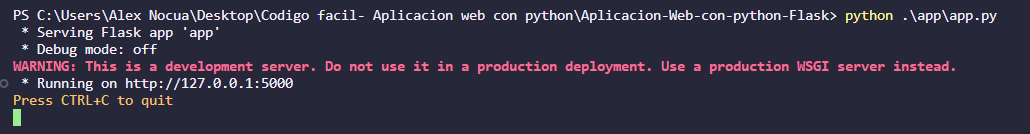


Muestra y código de la documentación.

Inicialización del proyecto:

Se utiliza el intérprete de Python y se ejecuta la siguiente instrucción.

python .\app\app.py



Muestra de creación del servidor.