Manual Técnico: Configuracion Frond-End

Data Filter

Versión: 1.0

**HISTORIAL DE REVISIÓN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **VERSIÓN** | **ELABORACIÓN** | | **REVISIÓN** | | **APROBACIÓN** | |
| **Fecha** | **Responsable** | **Fecha** | **Responsable** | **Fecha** | **Responsable** |
| 1.0 | 19/06/2021 | Luis Eduardo Rodriguez | 24/06/21 | Diana Maria Valencia | 24/06/21 | Diana Maria Valencia |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**CAMBIOS RESPECTO A LA VERSIÓN ANTERIOR**

|  |  |
| --- | --- |
| **VERSIÓN** | **MODIFICACIÓN RESPECTO VERSIÓN ANTERIOR** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Tabla de contenido**

[1. Introducción 4](#_heading=h.30j0zll)

[2. Alcance 4](#_heading=h.1fob9te)

[3. Definiciones, siglas y abreviaturas 4](#_heading=h.3znysh7)

[4. Responsables e involucrados 4](#_heading=h.2et92p0)

[5. Aspectos Técnicos 4](#_heading=h.tyjcwt)

[6. Requisitos de Configuración 4](#_heading=h.3dy6vkm)

[7. Proceso de Configuración o Despliegue 4](#_heading=h.1t3h5sf)

[8. Ingreso al Sistema 4](#_heading=h.4d34og8)

[9. Otras Consideraciones 4](#_heading=h.2s8eyo1)

# 1. Introducción

El presente documento explica detalladamente, cómo realizar el proceso de despliegue y configuración del front-end, que consiste en la conversión de datos de una interfaz gráfica para que el usuario pueda ver e interactuar con la información de forma digital. Para tener dicho resultado se empleó la tecnología de [Angular](https://angular.io/) que facilita la creación de componentes que pueden ser reutilizables.

El siguiente vínculo lo redirigirá en donde tenemos desplegado el aplicativo: [Data Filter](http://datafilter.pythonanywhere.com/)

# 2. Alcance

Para emplear una buena estructura en el desarrollo del front-end, fueron de vital importancia los servicios de terceros como: Git, Pythonanywhere para el alojamiento, incluyendo también el servidor de Node js y el framework de Angular con el gestor de paquetes npm, que permite la creación del proyecto.

# 3. Definiciones, siglas y abreviaturas

**Git :** Es un software de control de versiones, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente). Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

**Angula Js :** Es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

**Despliegue:** Se le conoce a la actividad de llevar tu código a producción. Si estás trabajando en un proyecto en tu computadora y quieres que más gente lo vea, con un dominio, en la nube. A ese proyecto le tendrás que hacer despliegue a un servidor en la nube.

**Npm:** Es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript por Isaac Schlueter, a través del cual podemos obtener cualquier librería con tan solo una sencilla línea de código, lo cual nos permitirá agregar dependencias de forma simple, distribuir paquetes y administrar eficazmente tanto los módulos como el proyecto a desarrollar en general.

**Código:** Es un conjunto de elementos que se combinan siguiendo unas ciertas reglas y que son semánticamente interpretables, lo cual permite intercambiar información.

**Node Js:** Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

**Servidor:** Es un conjunto de computadoras capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora, incluso en computadoras dedicadas a las cuales se les conoce individualmente como «el servidor».

**Visual Studio Code:** es un editor de código fuente desarrollado por [Microsoft](https://es.wikipedia.org/wiki/Microsoft) para [Wi](https://es.wikipedia.org/wiki/Microsoft_Windows)n[dows](https://es.wikipedia.org/wiki/Microsoft_Windows), Linux y macOS. Incluye soporte para la [depuración](https://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas), control integrado de [Git](https://es.wikipedia.org/wiki/Git), resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto),[1](https://es.wikipedia.org/wiki/Visual_Studio_Code#cite_note-ars-opensource-1)​[2](https://es.wikipedia.org/wiki/Visual_Studio_Code#cite_note-TechCrunch-2)​ aunque la descarga oficial está bajo [software privativo](https://es.wikipedia.org/wiki/Software_privativo) e incluye características personalizadas por Microsoft.

# 4. Responsables e involucrados

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo (Responsable/ Involucrado)** | **Rol** |
| Jorge Hernan Mapura | Responsable | Documentación, Control de Calidad, Programador |
| Christian Camilo Calle | Responsable | Analista, Tester, Programador. |
| Luis Eduardo Rodriguez Rendon | Responsable | Diseño, Control de Calidad, Programador. |

# 5. Aspectos Técnicos

La plataforma al estar alojada en un servidor en la nube, no requiere de una instalación de manera local, es decir en la computadora personal. Gracias a esto los usuarios, pueden interactuar en los siguientes navegadores Chrome, Microsoft edge, Opera, Mozilla firefox y Safari.

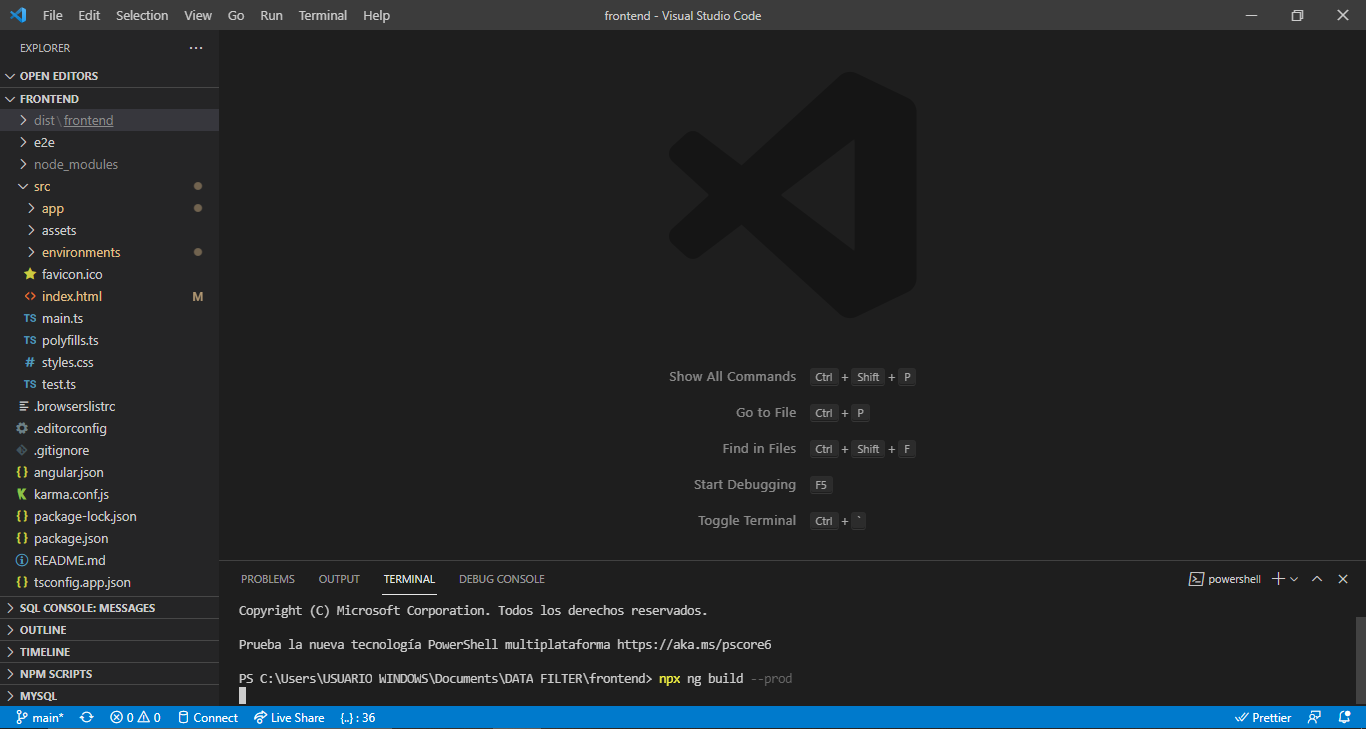
Como requisitos mínimos para un buen funcionamiento son necesarios los siguientes: 2GB de ram, disco duro de 250GB, un sistema operativo que sea de preferencia para el usuario como Windows, Linux o Mac OS. Cabe recalcar que al ser un aplicativo web es necesario contar una conexión estable a internet de al menos 2 megas.

# 6. Requisitos de Configuración

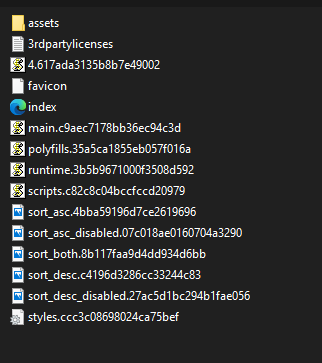
Para la construcción y configuración del entorno de desarrollo es necesario contar con un editor de código, en nuestro caso se utilizó Visual Studio Code, pero usted puede hacer uso del editor con el que se sienta más cómodo, siguiendo con los requisitos es necesario contar con el servidor de [Node Js](https://nodejs.org/es/) e incluir el gestor de paquetes npm para la instalación de las dependencias.

# 7. Proceso de Configuración o Despliegue

**Nota:** Como servidor de alojamiento para desplegar nuestro aplicativo web utilizaremos [pythonanywhere](https://www.pythonanywhere.com/).

* Para pasar la aplicacion al servidor de alojamiento primeramente tenemos que compilar el proyecto, para eso utilizaremos el siguiente código ‘ng build --prod”

* Una vez termine de realizar este proceso queda una carpeta con el nombre ‘dist’, esta contiene todo el proyecto compilado y con la siguiente estructura.



* Para el siguiente paso tenemos que tener instalado [python](https://www.python.org/downloads/), este proceso es sencillo y solo tiene que dirigirse a la página oficial y descargarlo, cuando termine de realizar la instalación, también tiene que instalar el framework de [flask](https://pypi.org/project/Flask/), para su instalación tiene que abrir con permisos de administrador la consola cmd y en ella ingresa el siguiente comando ‘pip install Flask’.
* Una vez realizado el paso anterior, vamos a dividir el proyecto en diferentes carpetas siguiendo esta estructura.

1. static

- ccs

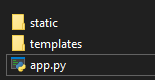
- js

1. templates

tomaremos todos los archivo con la extensión .js y los copiaremos dentro de la carpeta static/js y realizaremos el mismo proceso con el archivo css, pero esta vez lo copiaremos en static/css, el archivo .ico solo lo agregaremos a static ,y por último dentro de la carpeta templates estará el index.



El proyecto deberá quedar ordenado de esta manera, luego de esto agregaremos un archivo python para esto, en mi caso usare sublime text, lo nombraremos app.py.



El archivo app.py debe contener lo siguiente:

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/register')

def register():

return render\_template('index.html')

@app.route('/login')

def login():

return render\_template('index.html')

@app.route('/activated-account/<int:id>')

def activated\_account(id):

return render\_template('index.html')

@app.route('/recovery/<int:id>')

def recoveryid(id):

return render\_template('index.html')

@app.route('/recovery')

def recovery():

return render\_template('index.html')

@app.route('/consult')

def consult():

return render\_template('index.html')

@app.route('/environments')

def environments():

return render\_template('index.html')

@app.route('/profile')

def profile():

return render\_template('index.html')

@app.route('/environments/users/<int:id>')

def environments\_users(id):

return render\_template('index.html')

@app.route('/about-us')

def about\_us():

return render\_template('index.html')

@app.route('/environment/<int:id>')

def environment(id):

return render\_template('index.html')

@app.route('/environment/join/<code>')

def environment\_join(code):

return render\_template('index.html')

@app.route('/environments/report/<int:id>')

def environment\_report(id):

return render\_template('index.html')

@app.route('/contact\_us')

def contact\_us():

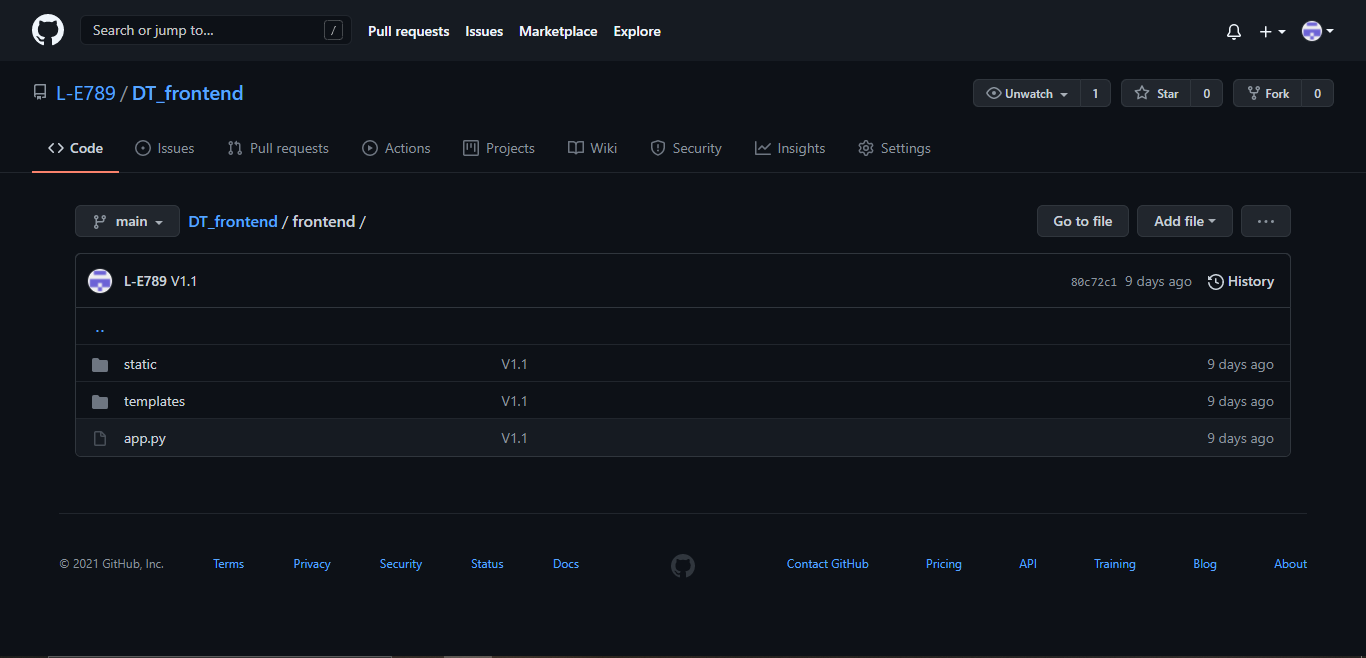
return render\_template('index.html')

@app.route('/3RjZgfU&rZVRLC7fzTNf1IRgxRFPvQ5G1ekFXJZ9/Sd89AsYwD912')

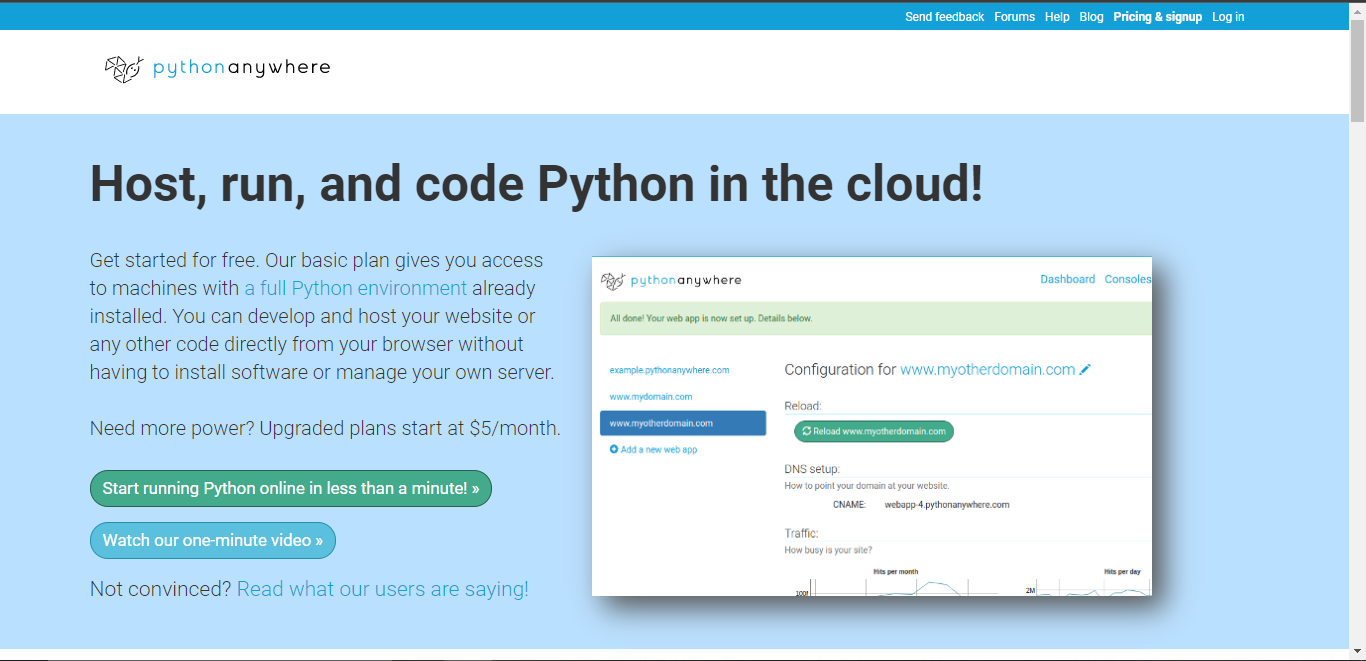
def login\_admin():

return render\_template('index.html')

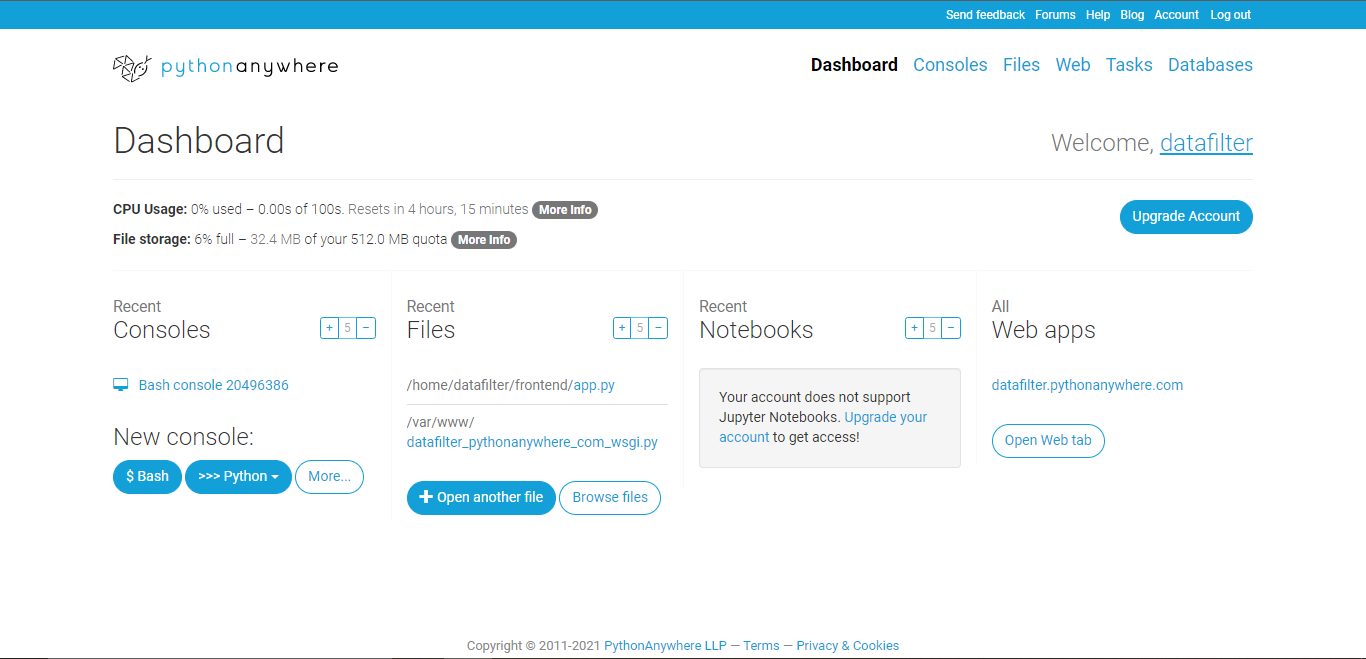
* Ahora pasaremos a subir nuestro proyecto a un repositorio, en mi caso utilizare github, pero usted pueden utilizar el que mas les guste:



* Una vez hechos los pasos anteriores ya podemos ingresar a [pythonanywhere](https://www.pythonanywhere.com/)



deberá diligenciar los datos que se les piden para registrarse en el aplicación, una vez ya registrados y logueados, les mostrara una ventana como esta:



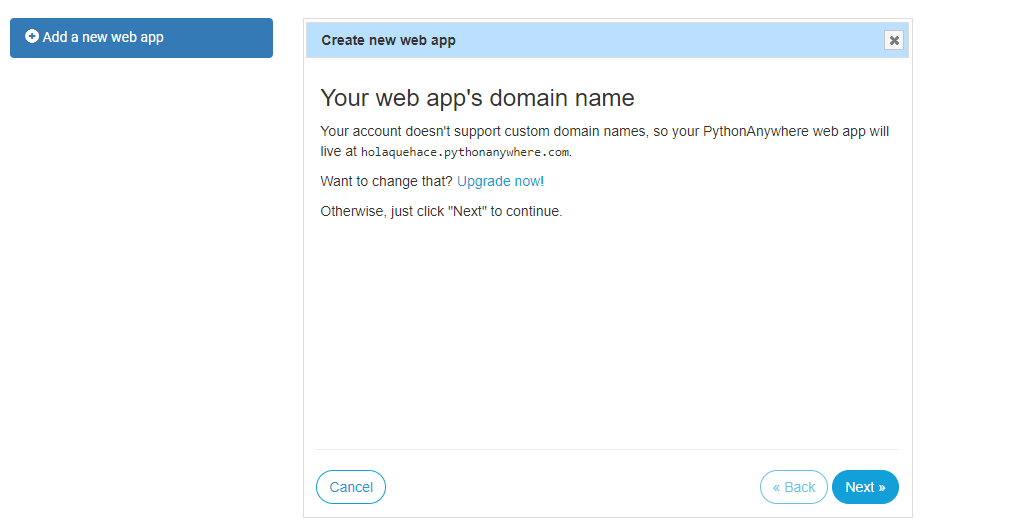
Ahora seleccionan en la barra derecha web.



después daremos clic en:

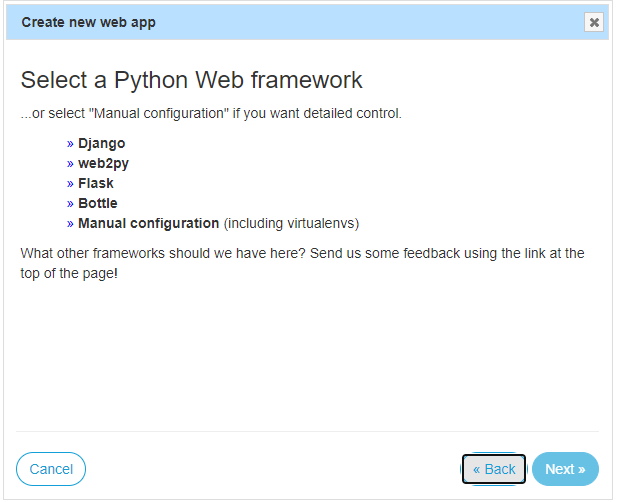


les aparecerá una ventana como la siguiente.

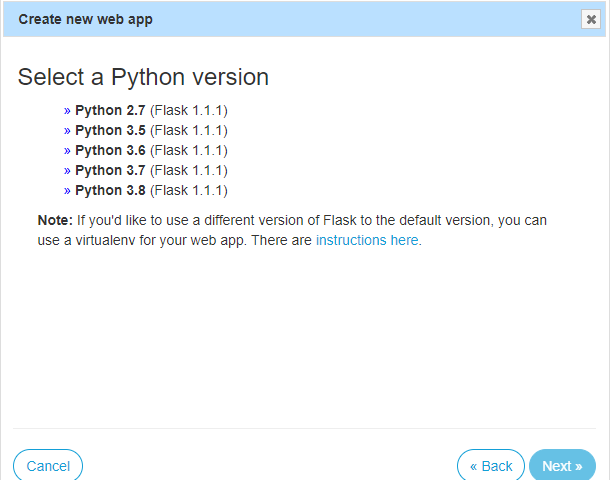


en esta solo daremos clic en next.

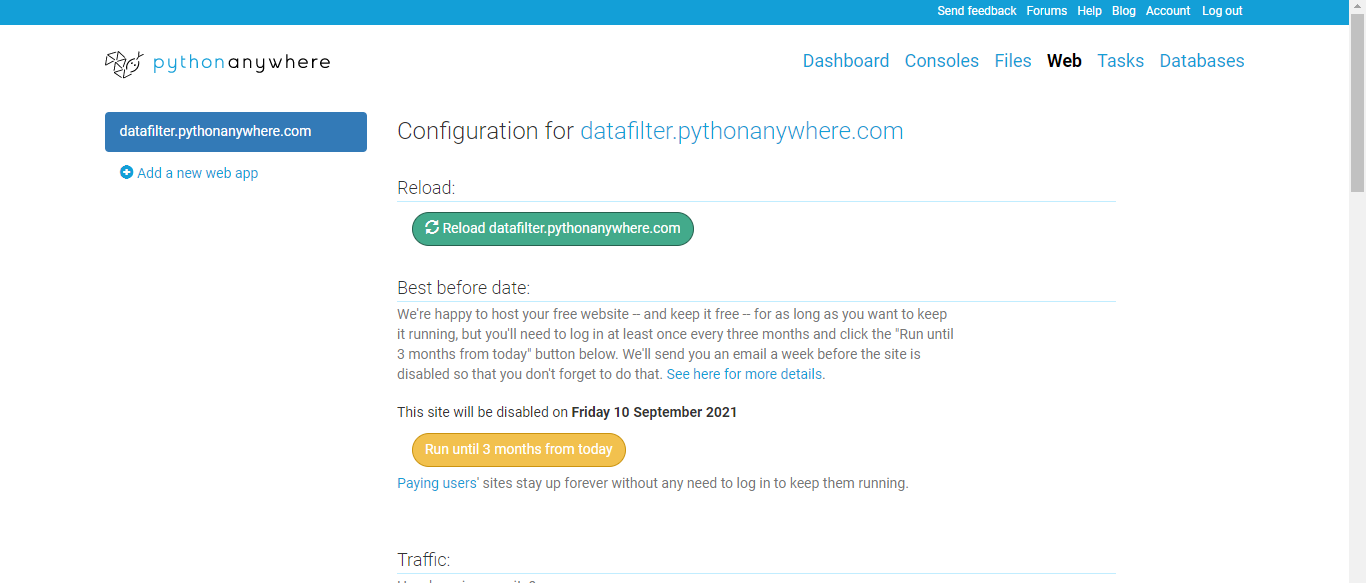
Ahora nos piden que seleccionemos el framework, en nuestro caso utilizaremos flask.



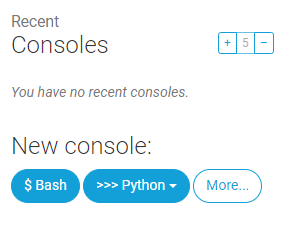
y por último nos piden que elijamos la versión de python que utilizaremos en este caso será la 3.8.



Si todo el proceso fue correcto les debe quedar una ventana con la siguiente información de su servidor.



* Ya hecho todo lo anterior, ahora tendremos que configurar el servidor, para esto nos regresamos a ‘Dashboard’, aquí iniciaremos una consola bash



luego de esto se nos mostrará una consola como la siguiente.

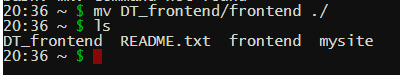


ahora vamos a irnos a github o donde crearon su repositorio y copiaremos la ruta para clonar el repositorio, en mi caso mi ruta es la siguiente <https://github.com/L-E789/DT_frontend.git>, una vez tengan la ruta, colocaremos el siguiente código en la consola bash.

ejemplo:

git clone <https://github.com/L-E789/DT_frontend.git>.

una vez finalizado este proceso, para darse cuenta que de los archivos se clonaron correctamente , pueden utilizar el siguiente comando ‘ls‘



Ahora utilizaremos mv nombre de su repositorio y nombre de la carpeta, tal y como se ve en la imagen anterior, de esta manera sacaremos todos los archivos de la carpeta DT\_frontend el cual es el nombre que le he puedo a mi repositorio, en su caso este nombre puede variar.

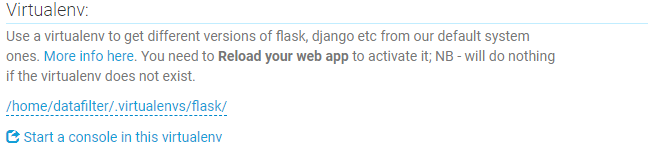
El siguiente paso será crear un entorno virtual para el manejo de las librerías que necesitaremos instalar, para esto utilizaremos el siguiente código ‘mkvirtualenv nombre que deseen -p python3’.

Ejemplo:

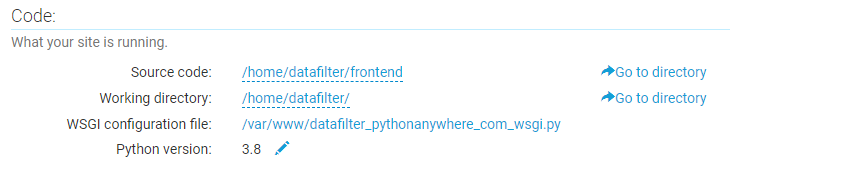
mkvirtualenv datafilter -p python3

Hecho lo anterior la consola nos deja posicionado en la ruta de nuestro entorno virtual, ahora procederemos a instalar Flask, y para eso utilizaremos el siguiente código ‘pip install Flask’.

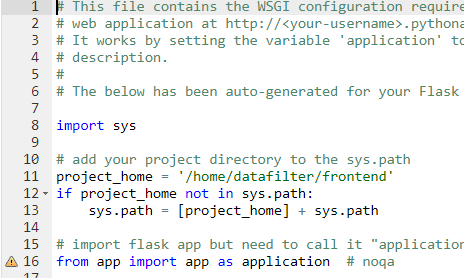
Regresamos a Dashboard y en la barra de navegación damos clic en web, nos desplazamos hasta encontrar virtualenv y colocamos la ruta que nos arroja la consola bash .



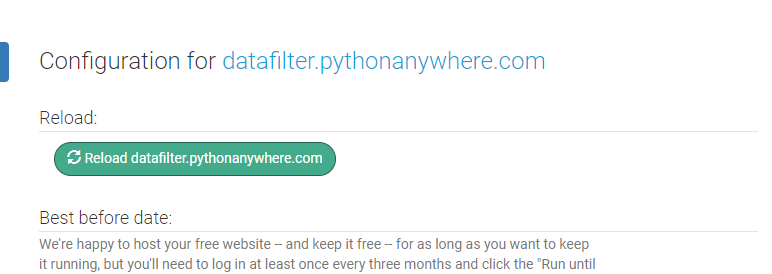
Una vez hecho el proceso anterior, nos dirigimos donde dice code, aquí cambiaremos la ruta /home/datafilter/misite a /home/datafilter/frontend.



Luego de esto seleccionaremos donde dice WSGI configuration file y deberán configurarlo tal y como lo verán en la siguiente imagen



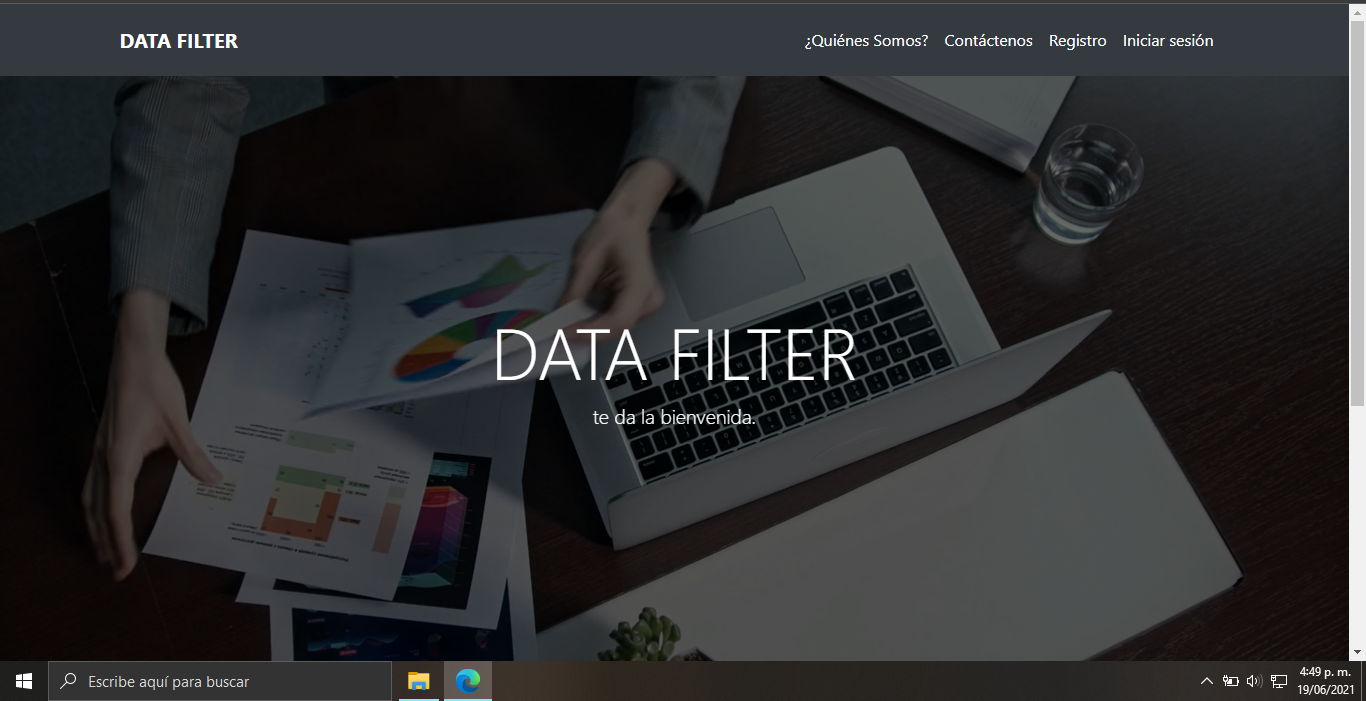
Con esto ya finalizamos el proceso de configuración, ahora solo sería recargar el servidor, para eso damos clic en reload. Y para acceder al aplicativo solo consta de dar clic en configuration for, como lo muestra la siguiente imagen.



# 

# 8. Ingreso al Sistema

Si el proceso de configuración anterior fue satisfactorio, al dirigirnos al aplicativo mediante la ruta [Data Filter](http://datafilter.pythonanywhere.com/) nos mostrará la siguiente ventana:



# 9. Otras Consideraciones

N/A