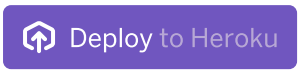
Application Server - Manual de Instalación



Aplicación en python (flask + gunicorn) con API RESTful para el app-server de Fiuber App.

Connect to… https://fiuber-app-server.herokuapp.com/

Para correr el servidor

Es necesario crear un entorno virtual sobre el que correr el servidor, para ello se utiliza el paquete *virtualenv* de python (`sudo pip install virtualenv`). Para crear un entorno virtual, estando en la carpeta del proyecto hacer `virtualenv venv` , se puede reemplazar *venv* por cualquier otro nombre, pero cuidado porque esto crea una carpeta y no se debería agregar a ningún commit (*venv* ya está en .gitignore).

Para entrar en el entorno virtual usar `source venv/bin/activate` , notar cómo el prompt indica entre paréntesis que se encuentra en el entorno venv. Aquí habría que instalar todos los paquetes que requiere la aplicación (si es que no se instalaron ya DENTRO del entorno virtual). Para facilitar las cosas están todos en el archivo *requirements.txt*, por lo que basta con hacer `pip install -r requirements.txt` . Se puede generar en cualquier momento un *requirements.txt* haciendo `pip freeze > requirements.txt` .

IMPORTANTE: para salir del entorno virtual, simplemente tipear `deactivate` .

Levantar el servidor localmente

Base de datos MongoDB

Es necesario tener corriendo una base de datos local si se quiere poder levantar el servido localmente. Para hacerlo hay que previamente tener instalado MongoDB Community Edition.

Una vez que se haya instalado, se deberá correr el servicio en segundo plano mediante `sudo service mongod start` .

Estando en el entorno virtual con las dependencias instaladas, se puede levantar el servidor con `make run` . Esto corre el comando `gunicorn --config config/gunicorn.conf --log-config config/logging.conf src.main.wsgi` seteando una variable de entorno `GUNICRON_BIND` que indica sobre qué puerto correr (ver makefile).

Nota: alternativamente a levantar una base de datos local, se puede setear una variable de entorno `MONGODB_URL` con la dirección de un base de datos remota (e.g. mlab), incluyendo en la misma la autenticación.

Conexión con el Shared Server

Por defecto la aplicación intenta conectarse a la API del Shared Server en la URL `http://localhost:5000/api` . Se puede cambiar este comportamiento mediante la variable de entorno `SS_URL` . Las conexiones con tal servidor utilizan un mecanismo de autenticación mediante un token. Antes de levantar una instancia del Application Server se tiene que estar autorizado previamente: solicitar un token al bussiness user pertienente (ver documentación del Shared server).

Una vez se tenga el token, se debe indicar el mismo a la aplicación mediante la variable de entonro `APP_TOKEN` .

Levantar el servidor en Heroku

Sencillamente se debe hacer `git push heroku +HEAD:master` para deployear en heroku la branch actual. Nótese que se utiliza *heroku* como remote, por lo tanto deberá estar configurado como tal. Usar `git remote -v` para ver la lista de remotes, y `git remote add` para agregar el remote con nombre heroku.

La application en heroku sobre la que deployar debe tener definidas todas las variables de entorno que se describieron en la sección de levantar el servidor localmente.

Para realizar requests por consolas

Para poder probar los distintos endpoints de la API se puede utilizar un navegador (que realiza request de tipo GET) o bien usar el comando `curl` y hacerlo por consola. En particular la estructura del comando sería:

```
curl -X _tipo_ -d _data_ http://someURL:somePort/someEndpoint
```

Donde *tipo* es POST, GET, etc; *data* son los datos a mandar en la request (un string representando un .json o '' para no enviar nada).

Para generar la documentación del código fuente

Ingresar el siguiente comando en la consola, desde el directorio raíz de la aplicación:

$ make

Los archivos de documentación se generarán automáticamente en la carpeta docs/documentation. Allí se podrá encontrar el subdirectorio correspondiente a cada formato disponible para visualizar la documentación (latex, html, etc.). Para visualizar la documentación en formato html, se debe ingresar a docs/documentation/html y seleccionar el archivo index.html. Para la documentación en latex, ingresar al directorio docs/documentation/latex y nuevamente ingresar el comando $make en la consola. Se generará un archivo .pdf con la documentación de la aplicación.

Uso de docker

Se debe tener instalado Docker Community Edition y configurado para poder correrlo sin sudo.

El archivo Dockerfile contiene toda la información sobre cómo generar el container. Usar los tasks `docker_build`, `docker_run` y `docker_stop` para crear una imágen, correr y detener un container.

Algunos comandos útiles para ver el estado de las imágenes/containers son:

Para crear una imagen a partir del Dockerfile en el directorio actual (notar el . al final del comando): `docker build -t image_name .`

Para crear un container a partir de una imagen ya creada usar: `docker container --name container_name -d -p 5000:8000 image_name`. El flag --name le da un nombre user friendly para poder referenciarlo luego; -d hace que se corra en segundo plano y -p mapea un puerto del host con algún puerto expuesto por el container.

Para listar todas las imágenes creadas: `docker images`. Para listar todos los containers corriendo `docker container ls`. Para detener un container `docker stop container_name` y para eliminar completamente un container `docker container rm container_name`.