



**El futuro digital
es de todos**

MinTIC



Ciclo 4a:

Desarrollo de aplicaciones web



**Misión
TIC 2022**

VERSIÓN 1.0

**Unidad de educación
continua y permanente
Facultad de Ingeniería**



Unidad Camilo Torres
Calle 44 # 45-67
Bloque B5 piso 1



(57) + 316 5000
uec.ftbog@unaleduco

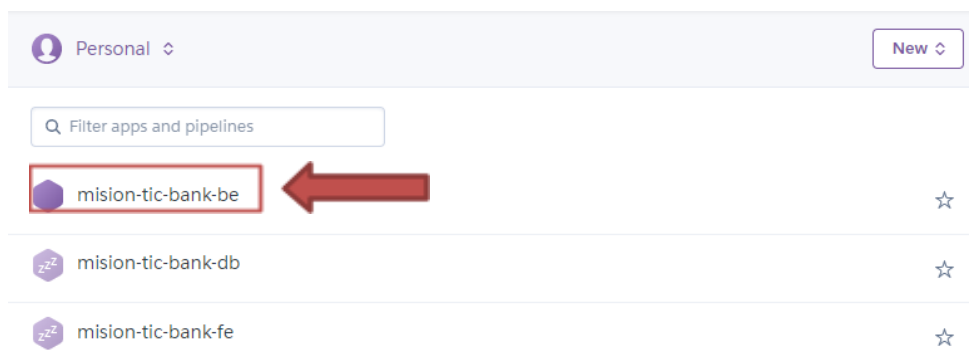
Actividad Práctica

(Despliegue de Microservicio AuthMS)

En sesiones anteriores se modificó el componente [bank_be](#) para crear el microservicio [auth_ms](#). Sin embargo, los cambios realizados no se subieron a la aplicación desplegada en Heroku, por ello, en esta guía se realizará el despliegue del microservicio, pero en lugar de utilizar un repositorio de GitHub para subir el proyecto del microservicio, se utilizará una imagen [Docker](#).

Cambiar el nombre del proyecto

Al igual como se cambió el nombre del proyecto [bank_be](#) en la guía anterior, también es necesario cambiar el nombre de la aplicación de Heroku en el cual se desplegará el microservicio, esto para mantener el orden de los componentes del sistema. Para ello, se debe dirigir a la consola de Heroku, y se debe abrir el proyecto donde se desplegó anteriormente el componente [bank_be](#) (ahora [auth_ms](#)):



Allí se debe dirigir a la pestaña [Settings](#):

mission-tic-bank-db

Overview Resources Deploy Metrics Activity Access **Settings**

Get a complete visualization of your app in a team-based continuous delivery environment with [Heroku Pipelines](#). [Hide](#) [Create a Heroku Pipeline](#)

Installed add-ons **\$0.00/month** [Configure Add-ons](#)

Latest activity [All Activity](#)

Heroku Postgres Hobby Dev postgresql-regular-05207

Dyno formation **\$0.00/month** [Configure Dynos](#)

This app is using free dynos

web gunicorn authProject.wsgi **ON**

missiontic2022@hotmail.com: Deployed 1d293d70 Sep 23 at 8:01 PM · v5

missiontic2022@hotmail.com: Build succeeded Sep 23 at 8:00 PM · [View build log](#)

missiontic2022@hotmail.com: @ref:postgresql-regular-05207 completed provisioning, setting DATABASE_URL Sep 15 at 9:37 PM · v4

En esta pestaña se encuentra el campo de texto *App Name*:

mission-tic-bank-be

Overview Resources Deploy Metrics Activity Access **Settings**

App Information

App Name

mission-tic-bank-be

Region United States

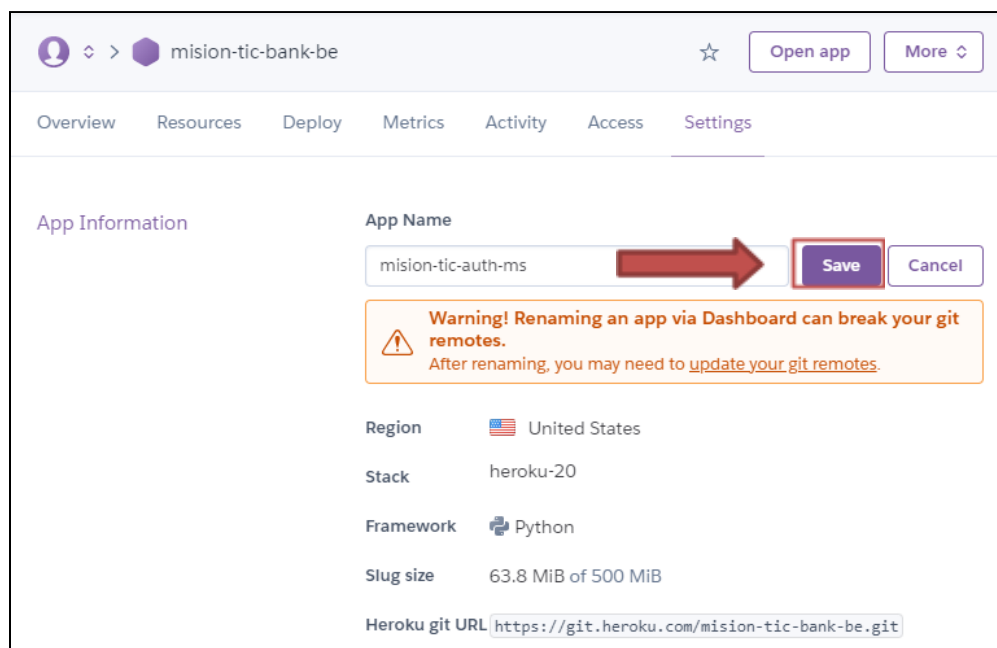
Stack heroku-20

Framework Python

Slug size 63.8 MiB of 500 MiB

Heroku git URL <https://git.heroku.com/mission-tic-bank-be.git>

En este se debe ingresar el nuevo nombre que se le desea dar a la aplicación, en este caso será *mission-tic-auth-ms*. Una vez ingresado, se debe pulsar el botón *Save*:



mision-tic-bank-be

Overview Resources Deploy Metrics Activity Access Settings

App Information

App Name

mision-tic-auth-ms Save Cancel

Warning! Renaming an app via Dashboard can break your git remotes.
After renaming, you may need to [update your git remotes](#).

Region United States

Stack heroku-20

Framework Python

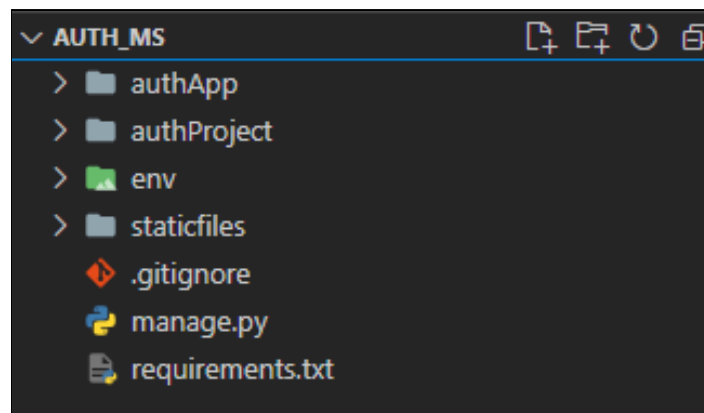
Slug size 63.8 MiB of 500 MiB

Heroku git URL <https://git.heroku.com/mision-tic-bank-be.git>

En este caso, ya que no se va a desplegar el microservicio utilizando un repositorio de GitHub, se hará caso omiso a la advertencia que indica Heroku.

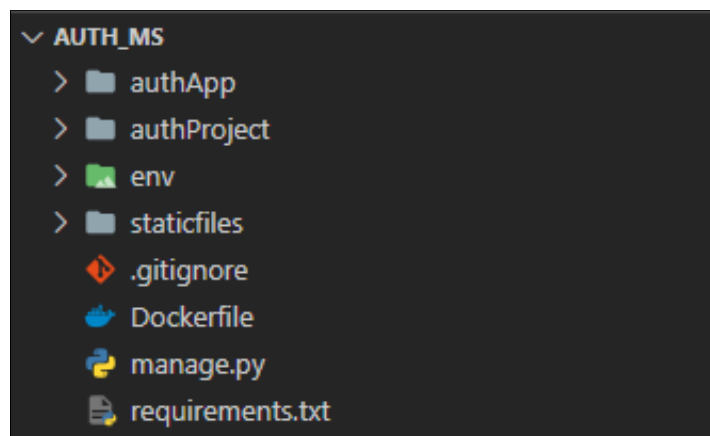
Modificaciones en la estructura del proyecto

Normalmente, cuando se hace el despliegue con Docker y Heroku, es necesario instalar los paquetes [django-heroku](#) y [gunicorn](#) en el proyecto, y realizar una pequeña configuración en el archivo [settings.py](#). Sin embargo, esto ya se realizó anteriormente dentro del proyecto, por lo cual, para el despliegue con Docker solo es necesario eliminar el archivo [Procfile](#), ya que este era utilizado para realizar el despliegue directo con Heroku, pero en este caso, al estar realizando el despliegue con Docker, resulta innecesario. Por otro lado, el archivo [.gitignore](#) y toda la carpeta de git, no serán utilizados en este despliegue con Docker; sin embargo, se pueden mantener con el único propósito de llevar el control de versiones del proyecto. Así, una vez eliminado el archivo, la estructura del proyecto debe ser la siguiente:



Creación del Dockerfile

Ahora, para desplegar el microservicio con Docker, se debe crear una imagen del proyecto, esto se realizará por medio de un [Dockerfile](#) que define los pasos para crear dicha imagen. Para ello, se debe crear un archivo llamado [Dockerfile](#) (sin extensión) en la raíz del proyecto, es decir en el mismo nivel del archivo [manage.py](#). Al hacerlo, la estructura del proyecto debe ser la siguiente:



Una vez se ha creado el [Dockerfile](#), dentro de este se deben indicar las siguientes instrucciones:

```
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /users
WORKDIR /users
ADD . /users/
RUN pip install -r requirements.txt
```

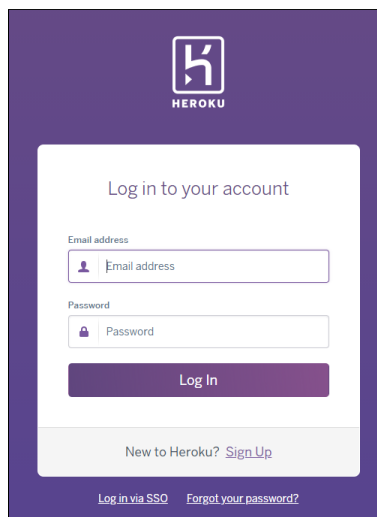
EXPOSE 8080

```
CMD python manage.py makemigrations && python manage.py migrate && python manage.py  
runserver 0.0.0.0:$PORT
```

Si bien el proceso que realiza Docker para crear la imagen puede parecer complejo, en general no lo es. Primero se está iniciando un proyecto basado en [python:3](#), luego se crea el respectivo entorno virtual en el cual se copian los archivos del proyecto y se instalan los paquetes necesarios con ayuda de [requirements.txt](#), y finalmente se realizan las migraciones correspondientes y se expone el microservicio en el puerto indicado.

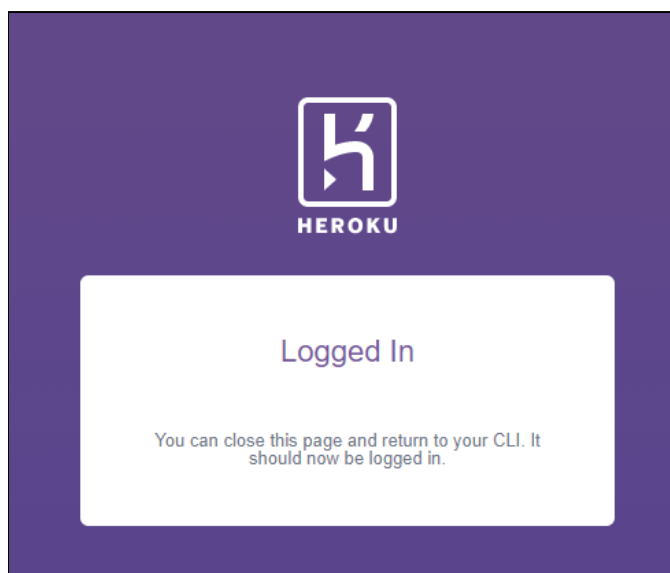
Despliegue del microservicio

Ahora que se ha hecho todo lo necesario, se puede realizar el despliegue. Para ello, primero se debe iniciar sesión en el CLI de Heroku, utilizando en una terminal el comando [heroku login](#). Al hacerlo, se abrirá una ventana en el navegador que permitirá iniciar sesión:



The image shows the Heroku login interface. At the top is the Heroku logo. Below it, the text "Log in to your account" is centered. There are two input fields: "Email address" with a user icon and "Password" with a lock icon. Below these fields is a purple "Log In" button. At the bottom, there is a link "New to Heroku? Sign Up" and two smaller links: "Log in via SSO" and "Forgot your password?".

Una vez se ingresan las credenciales y se pulsa el botón [Log In](#), se muestra el siguiente mensaje en el navegador, y se actualiza la terminal donde se ingresó el comando:



```
(env) D:\auth_ms>heroku login
» Warning: heroku update available from 7.56.1 to 7.59.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/9fd88
Logging in... done
Logged in as misiontic2022@hotmail.com
```

Ahora se debe crear la imagen del proyecto, para esto Docker se debe estar ejecutando en el sistema (simplemente se debe abrir la aplicación). Así con Docker corriendo, se debe conectar la consola con el servicio de Docker de Heroku, para esto se ejecuta comando `heroku container:login`:

```
(env) D:\auth_ms>heroku container:login
» Warning: heroku update available from 7.56.1 to 7.59.0.
Login Succeeded
```

Una vez se ha establecido la conexión, se puede crear la imagen con el comando, `heroku container:push web --app nombre-app`, donde `nombre-app` es el nombre de la aplicación de Heroku. Este comando puede tardarse varios minutos en ejecutarse, cuando finalice se tendrá construida la imagen del proyecto:



```
(env) D:\auth_ms>heroku container:push web --app mision-tic-auth-ms
» Warning: heroku update available from 7.56.1 to 7.59.0.
=== Building web (D:\auth_ms\Dockerfile)
[+] Building 86.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 286B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 48.92MB
=> [1/5] FROM docker.io/library/python:3@sha256:a487658b37559c499868dd4bdc6b18ed25cbfb5a02d054c9eafaf713d5aca
=> => resolve docker.io/library/python:3@sha256:a487658b37559c499868dd4bdc6b18ed25cbfb5a02d054c9eafaf713d5aca
=> => sha256:3d11a72c123ffc9326b86ad432480cef0ab73f02b7ed684873c2f8c35ce9a79f 2.22kB / 2.22kB
=> => sha256:4246fb19839fd033a0dd925c1f89cd1ad482c6b703d56f34bf0d2808b076e132 8.61kB / 8.61kB
=> => sha256:f02b617c6a8c415a175f44d7e2c5d3b521059f2a6112c5f022e005a44a759f2d 5.15MB / 5.15MB
=> => sha256:d32e17419b7ee61bbd89c2f0d2833a99cf45e594257d15cb567e4cf7771ce34a 10.87MB / 10.87MB
=> => sha256:a487658b37559c499868dd4bdc6b18ed25cbfb5a02d054c9eafaf713d5aca 2.60kB / 2.60kB
=> => sha256:bb7d5a84853b217ac05783963f12b034243070c1c9c8d2e60ada47444f3cce04 54.92MB / 54.92MB
=> => sha256:c9d2d81226a43a97871acd5afb7e8aafb4d6b62ae1709c870df3ee230bc3f5 54.57MB / 54.57MB
=> => sha256:3c24ae8b66041c09dabc913b6f16fb914d5640b53b10747a343ddc5bb5bd6769 196.50MB / 196.50MB
=> => sha256:8a4322d1621dec2def676639847e546c07a1da4ba0c035edb1036bf5fb45063b 6.29MB / 6.29MB
=> => extracting sha256:bb7d5a84853b217ac05783963f12b034243070c1c9c8d2e60ada47444f3cce04
=> => sha256:b777982287b6a9a7b6c2e8f090de34a1564abf3916be787cacd186d2b8605062 19.12MB / 19.12MB
=> => sha256:2c5fb32d4befd10c682ce3f658540f40db4ec2a0b03afa659bd64b98472b31c9 232B / 232B
```

Una vez creada la imagen del proyecto, solo falta realizar el despliegue, para ello se utiliza el comando `heroku container:release web --app nombre-app`, donde `nombre-app` es el nombre de la aplicación de Heroku:

```
(env) D:\auth_ms>heroku container:release web --app mision-tic-auth-ms
» Warning: heroku update available from 7.56.1 to 7.59.0.
Releasing images web to mision-tic-auth-ms... done
```

Una vez realizado el despliegue del microservicio `auth_ms`, se puede probar que las urls del servidor funcionan correctamente. Para esto se utilizará `Postman`, junto con la url de componente desplegado en heroku, cuyo formato, como ya se ha visto, es el siguiente:

<https://{nombre-app}.herokuapp.com/>

Toda funcionalidad se probará con la url de la aplicación de heroku seguida por el sufijo de la funcionalidad a probar. En este caso, solo se comprobará el funcionamiento de la vista que fue modificada (`UserDetailView`) y de la vista que fue creada (`VerifyTokenView`).

