



El futuro digital  
es de todos

MinTIC

Misión  
TIC 2022

# CICLO 3

## Desarrollo de Software



Hechos  
QUE CONECTAN



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# Componente Lógico

## (Configuración Preliminar)

### Actividad Práctica

En la sesión anterior se creó el proyecto de Django que se va a utilizar para desarrollar el componente [bank\\_be](#). En esta guía se realizarán las configuraciones preliminares del proyecto que permitirán el debido desarrollo del componente en las próximas sesiones. Las configuraciones que se realizarán son, incluir Simple JWT, conectar el componente a la base de datos, y modificar la estructura del proyecto.

#### Instalación y configuración de Simple JWT

Para incluir Simple JWT en el proyecto, se debe abrir una terminal en la carpeta [bank\\_be](#) y activar el entorno virtual de Python en caso de no tenerlo activado (con el comando [env\Scripts\activate](#) o con el comando [source env/bin/activate](#), de acuerdo con el sistema operativo). Una vez está activo, se instala [Simple JWT](#) con el comando [pip install djangorestframework-simplejwt](#):

```
(env) D:\bank_be>pip install djangorestframework-simplejwt
Collecting djangorestframework-simplejwt
  Downloading djangorestframework_simplejwt-4.8.0-py3-none-any.whl (70 kB)
    | 70 kB 4.5 MB/s
Requirement already satisfied: django in d:\bank_be\env\lib\site-packages (
```

Cuando finaliza la instalación del paquete, este ya se puede utilizar en el componente [bank\\_be](#). Sin embargo, para asegurar su correcto funcionamiento, y para ajustar el comportamiento deseado, se deben realizar dos configuraciones en el archivo [settings.py](#), que se encuentra dentro de la carpeta [authProject](#).

La primera configuración que se debe realizar es indicarle al sistema de autenticación provisto por Django REST Framework que permita a Simple JWT el acceso a dicho sistema, pues es necesario para la creación y verificación de los tokens. Para esto, en [settings.py](#) se debe crear la variable [REST\\_FRAMEWORK](#), cuyo valor será un diccionario que indica las clases que tienen acceso al sistema de autenticación:

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.AllowAny',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

Esta variable se puede ubicar en cualquier parte del archivo, sin embargo, para mantener el orden de las configuraciones, se ubicará bajo la variable [MIDDLEWARE](#):



```
settings.py X
authProject > settings.py > ...
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
53
54 REST_FRAMEWORK = {
55     'DEFAULT_PERMISSION_CLASSES': (
56         'rest_framework.permissions.AllowAny',
57     ),
58     'DEFAULT_AUTHENTICATION_CLASSES': (
59         'rest_framework_simplejwt.authentication.JWTAuthentication',
60     )
61 }
62
```

La segunda configuración a realizar es indicarle a *Simple JWT* los atributos que se desea que tengan los tokens generados por la librería. Para ello, en *settings.py* se debe crear la variable *SIMPLE\_JWT*, cuyo valor será un diccionario que contiene los valores de cada atributo del token:

```
SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=5),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
    'ROTATE_REFRESH_TOKENS': False,
    'BLACKLIST_AFTER_ROTATION': True,
    'UPDATE_LAST_LOGIN': False,

    'ALGORITHM': 'HS256',
    'USER_ID_FIELD': 'id',
    'USER_ID_CLAIM': 'user_id',
}
```

Esta variable se puede ubicar en cualquier parte del archivo, sin embargo, para mantener el orden de las configuraciones, se ubicará bajo la variable *INSTALLED\_APPS*:

```
settings.py 2 X
authProject > settings.py > ...
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'rest_framework',
41     'authApp',
42 ]
43
44 SIMPLE_JWT = {
45     'ACCESS_TOKEN_LIFETIME': timedelta(minutes=5),
46     'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
47     'ROTATE_REFRESH_TOKENS': False,
48     'BLACKLIST_AFTER_ROTATION': True,
49     'UPDATE_LAST_LOGIN': False,
50
51     'ALGORITHM': 'HS256',
52     'USER_ID_FIELD': 'id',
53     'USER_ID_CLAIM': 'user_id',
54 }
55
```

Con este diccionario se está indicando que, entre otras cosas, los token access tendrán una duración de 5 minutos, los token refresh tendrán una duración de 1 día, y el algoritmo de encriptación utilizado para los tokens es HS256. Todas las llaves y sus posibles valores se pueden consultar en la documentación de Simple JWT: <https://django-rest-framework-simplejwt.readthedocs.io/en/latest/settings.html>.

Como se puede notar en la variable de configuración, dos valores están utilizando la función `timedelta`. Ya que esta función no se encuentra en el archivo `settings.py`, se debe importar desde la librería `datetime`. Para hacerlo se debe utilizar la línea de código:

```
from datetime import timedelta
```

Esta línea de código se puede ubicar en cualquier parte del archivo, sin embargo, para mantener el orden y las buenas prácticas, se ubicará al principio del archivo, bajo el `import` que ya existe:

```
settings.py X
authProject > settings.py > ...
12
13 from pathlib import Path
14 from datetime import timedelta
15
```

## Conexión con la Base de Datos

Para conectar el componente lógico con la base de datos, se debe instalar en el proyecto un paquete que se encargue de la administración de la conexión. Debido a que la base de datos desplegada en Heroku en sesiones anteriores es PostgreSQL, el paquete que se debe instalar es [psycopg2](#). Para ello, se debe utilizar el comando `pip install psycopg2` en la terminal:

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

(env) D:\bank_be>pip install psycopg2
Collecting psycopg2
  Downloading psycopg2-2.9.1-cp39-cp39-win_amd64.whl (1.2 MB)
    | 1.2 MB 3.3 MB/s
Installing collected packages: psycopg2
Successfully installed psycopg2-2.9.1

(env) D:\bank_be>
```

Una vez se ha instalado este paquete, se debe configurar la conexión en el archivo [settings.py](#), que se encuentra dentro de la carpeta [authProject](#). En este archivo se debe buscar la variable `DATABASES`:

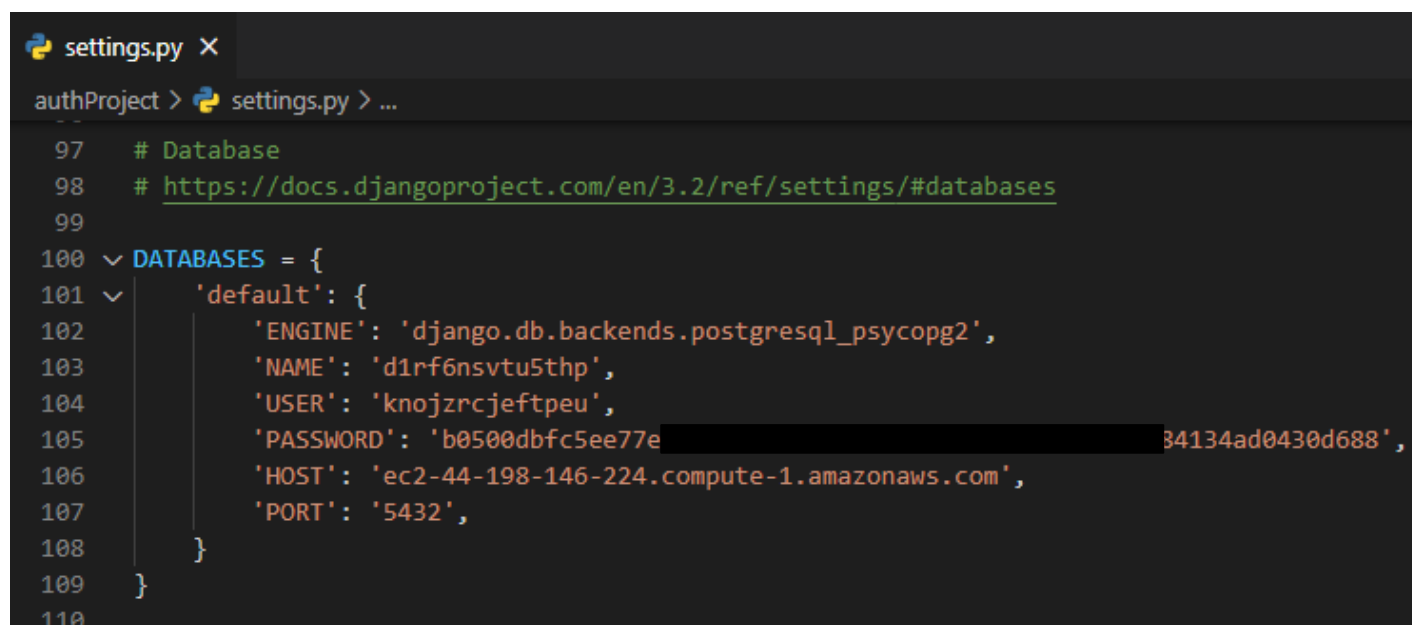
```
settings.py X
authProject > settings.py > ...

97  # Database
98  # https://docs.djangoproject.com/en/3.2/ref/settings/#databases
99
100 DATABASES = {
101     'default': {
102         'ENGINE': 'django.db.backends.sqlite3',
103         'NAME': BASE_DIR / 'db.sqlite3',
104     }
105 }
106
```

Una vez encontrada, se debe reemplazar con el formato que se debe seguir para ingresar los datos para la conexión del componente con la base de datos:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': '',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '5432',
    }
}
```

Una vez se ha reemplazado la variables, se deben llenar los campos vacíos con las credenciales obtenidas en la sesión de despliegue de la base de datos en Heroku, teniendo en cuenta que la llave '**NAME**' se refiere al valor *Database* de las credenciales. Al completar el formato, la variable debe verse similar a la siguiente imagen:



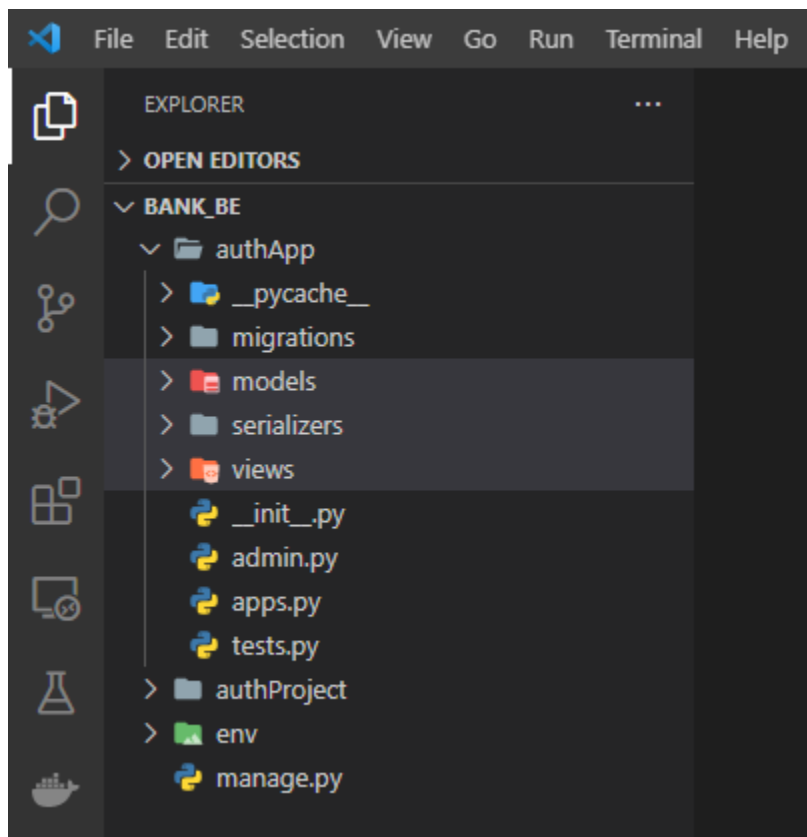
```
settings.py X
authProject > settings.py > ...
97 # Database
98 # https://docs.djangoproject.com/en/3.2/ref/settings/#databases
99
100 DATABASES = {
101     'default': {
102         'ENGINE': 'django.db.backends.postgresql_psycopg2',
103         'NAME': 'd1rf6nsvtu5thp',
104         'USER': 'knojzrcjeftepu',
105         'PASSWORD': 'b0500dbfc5ee77e34134ad0430d688',
106         'HOST': 'ec2-44-198-146-224.compute-1.amazonaws.com',
107         'PORT': '5432',
108     }
109 }
110
```

Con esto, ya se ha completado la conexión a la base de datos del proyecto, por lo cual, si se vuelve a ejecutar el servidor ya no se creará el archivo *db.sqlite3*. Sin embargo, para evitar inconvenientes de consistencia con la base de datos, **NO** ejecute el servidor hasta que se desarrolle la guía de la próxima sesión.

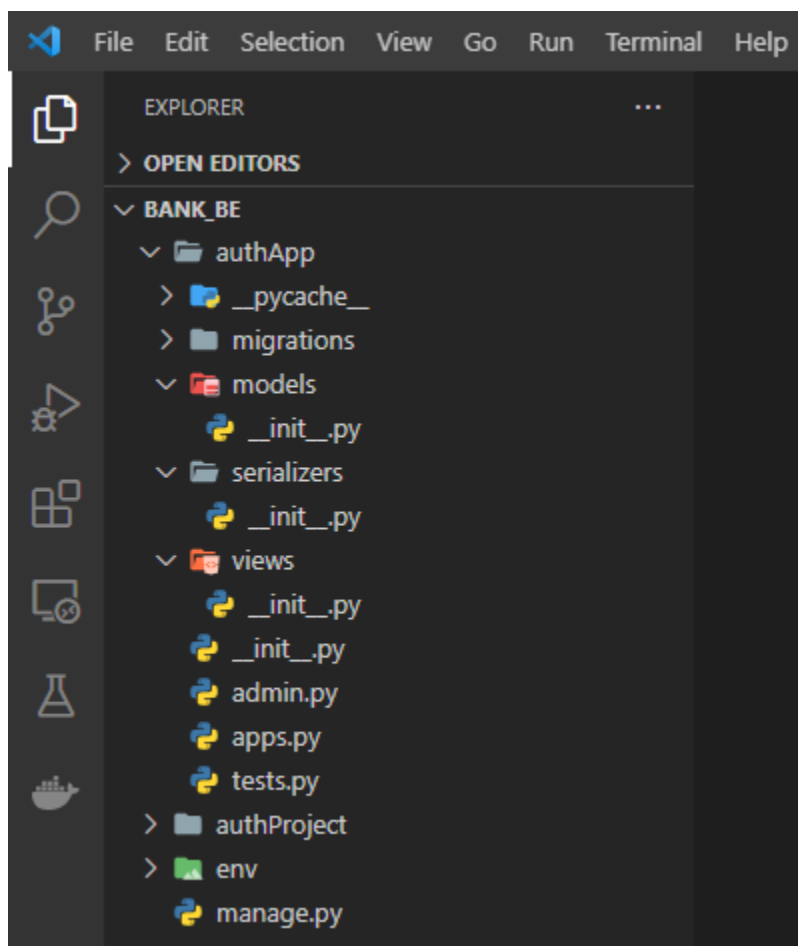


## Cambio en la estructura del proyecto

Django utiliza el patrón MVC en su implementación. Por esta razón, en la carpeta `authApp` se encuentran los archivos `models.py` y `views.py`, los cuales como sus nombres los indican, representan los modelos y las vistas respectivamente. Sin embargo, en futuras sesiones no se utilizarán archivos que contengan los modelos, las vistas y los controladores, en su lugar, se utilizarán carpetas que contengan el código de cada uno. Por esta razón, se deben borrar los archivos `models.py` y `views.py`, y se deben crear las carpetas `models`, `serializers` y `views` de tal forma que la carpeta `authApp` se vea así:



En cada una de estas carpetas se debe crear un archivo vacío con el nombre `__init__.py`, el cual convertirá la carpeta en un paquete, cuyo código podrá ser utilizado en cualquier parte del proyecto. Una vez se haga esto, la estructura del proyecto debe quedar así:



**Nota:** de ser necesario, en el material de la clase se encuentra un archivo *C3.AP.07. bank\_be.rar*, con todos los avances en el desarrollo del componente realizado en esta guía.