



El futuro digital  
es de todos

MinTIC

```
te( "name" );  
"type" );
```

```
if ( type == "sprite" )  
  
std::string item_name = item->Attribute( "name" );  
std::string spritename = item->Attribute( "spritename" );  
float x = boost::lexical_cast<float>( item->Attribute( "x" ) );  
float y = boost::lexical_cast<float>( item->Attribute( "y" ) );  
float offset = boost::lexical_cast<float>( item->Attribute( "offset" ) );  
  
SpriteDescList::iterator sp = sprite_descs.begin();  
for( ; sp != sprite_descs.end(); ++sp )  
    if ( sp->name_ == spritename )  
        break;
```

## Ciclo 3:

Desarrollo de Software



**Misión  
TIC2022**

VERSIÓN 1.0

Unidad de educación  
continua y permanente  
Facultad de Ingeniería



Unidad Camilo Torres  
Calle 44 e 45-67  
Bloque 85 piso 1



(57) + 316 5000  
uec\_ibog@unaleduco

# Componente Lógico

## (Creación y Configuración Inicial del Proyecto)

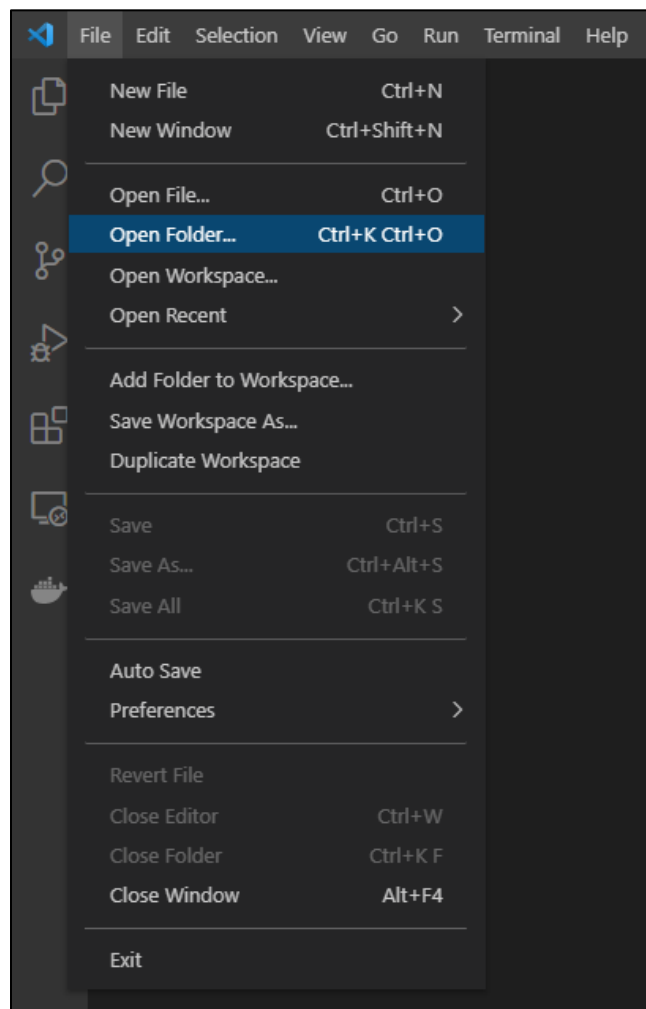
### Actividad Práctica

Una vez desarrollada la capa de datos, se desarrollará el componente asociado a la capa lógica: [bank\\_be](#). De esta forma, en esta actividad se realizará la creación y la configuración del proyecto respectivo, a partir del uso del framework *Django REST Framework*.

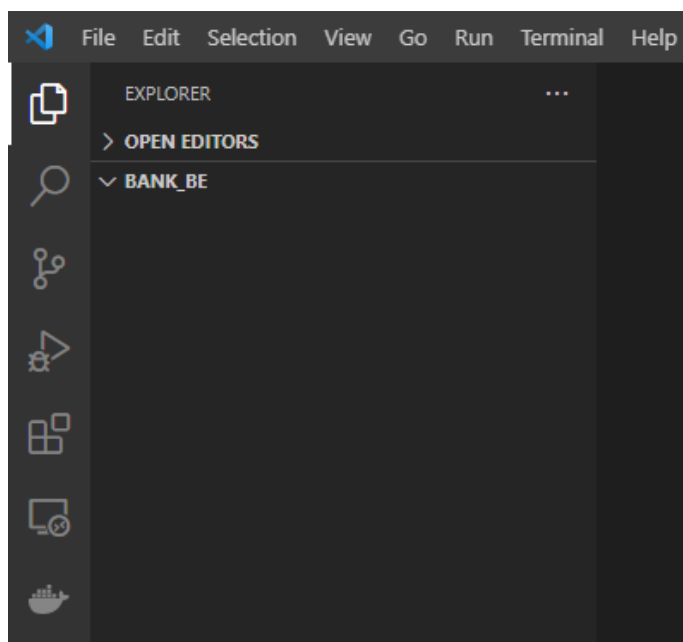
Para empezar, es necesario crear la carpeta donde se almacenarán todos los archivos del componente back-end; esta carpeta se llamará [bank\\_be](#). Para editar su contenido y ejecutar los comandos que se indicarán en esta guía se recomienda el uso de Visual Studio Code (VSC). De esta forma, el proceso que se debe seguir para la creación y configuración del componente, es el siguiente:

#### Abrir la carpeta del proyecto con Visual Studio Code

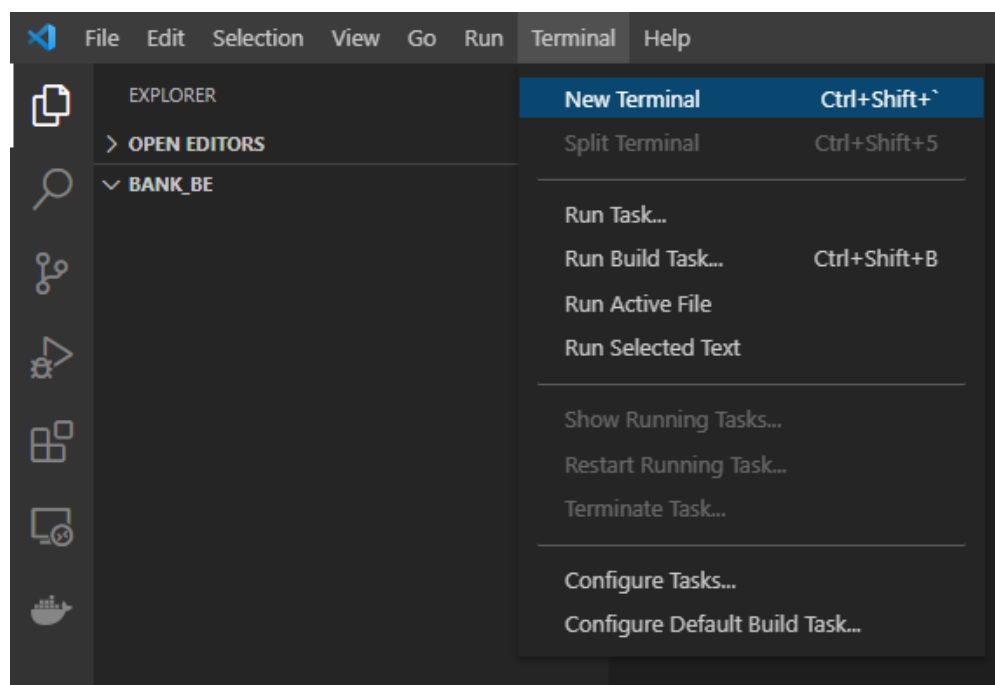
Lo primero que se debe hacer es abrir la carpeta [bank\\_be](#) en Visual Studio Code (VSC). Para ello, se expande la pestaña [File](#), y se elige la opción [Open Folder](#).



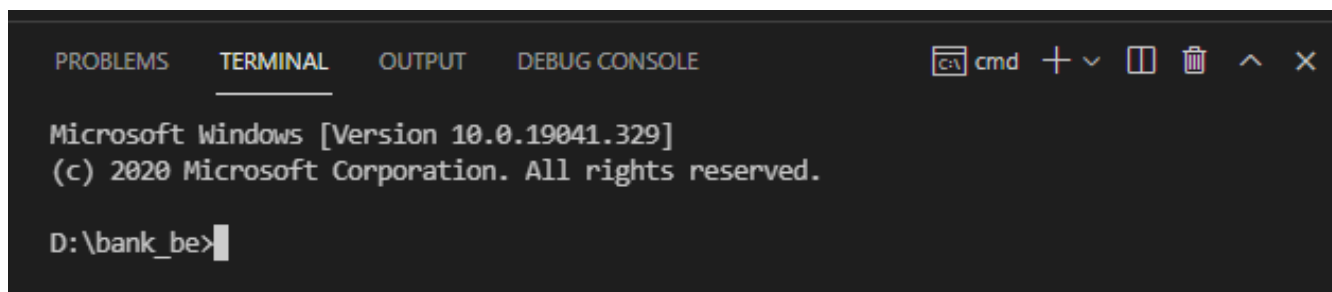
Se abrirá una ventana donde se debe buscar y seleccionar la carpeta creada anteriormente. Una vez se ha seleccionado, el panel izquierdo se verá similar a lo siguiente:



Una de las ventajas que ofrece el editor de texto VSC, es la posibilidad de abrir una terminal dentro de las carpetas en las que se trabaja. Para abrir una nueva terminal en la carpeta raíz del proyecto `users_ms`, se debe dirigir a la pestaña **Terminal**, y seleccionar la opción **New Terminal**:



Esto abre una terminal en la parte inferior de la aplicación, en la cual se ingresarán los comandos necesarios para el desarrollo del proyecto:



```

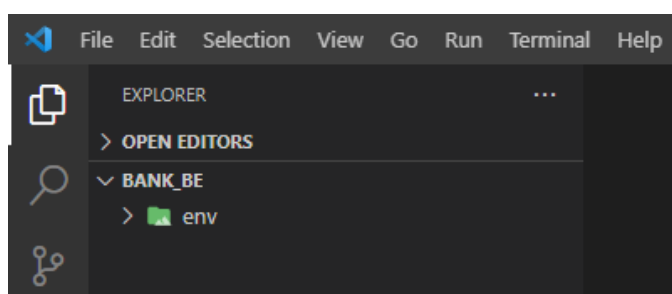
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE
Microsoft Windows [Version 10.0.19041.329]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\bank_be>

```

## Creación y ejecución del entorno de Python

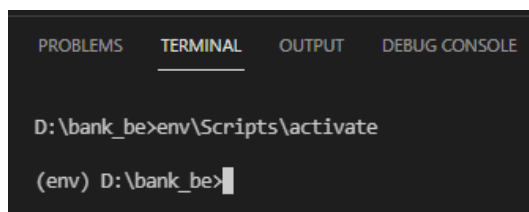
Para desarrollar el componente con Django REST Framework se debe crear un entorno virtual de Python, el cual permitirá la administración de los paquetes y las instalaciones necesarias para la ejecución del componente. Para ello, en la terminal de VSC, abierta anteriormente, se utiliza el comando `python -m venv env`. Una vez termine la ejecución del comando, se habrá creado una carpeta llamada `env` dentro de la carpeta del proyecto:



Ahora se debe iniciar el entorno virtual creado. De esta forma, de acuerdo al sistema operativo utilizado, se debe ejecutar un comando en la terminal:

1. Si se utiliza Windows: comando `env\Scripts\activate`
2. Si se utiliza una distribución de Linux o MacOS: el comando `source env/bin/activate`

Una vez se ejecute dicho comando, la terminal mostrará un indicador que permite verificar que el entorno virtual se encuentra activo:



```

PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

D:\bank_be>env\Scripts\activate

(env) D:\bank_be>

```

Es importante tener en cuenta que **siempre que se abra una nueva terminal para el desarrollo del proyecto, se debe ingresar el comando para iniciar el entorno virtual en la terminal**. Si no se hace, no se podrán ejecutar los comandos relacionados al proyecto y se mostrará un error.

### Creación del proyecto Django

Una vez se ha activado el entorno virtual en la terminal, se debe crear el proyecto Django. Para ello, se instala *Django* y *Django REST Framework*, utilizando los comandos *pip install django* y *pip install djangorestframework*:

```
(env) D:\bank_be>pip install django
Collecting django
  Using cached Django-3.2.7-py3-none-any.whl (7.9 MB)
Collecting asgiref<4,>=3.3.2
  Using cached asgiref-3.4.1-py3-none-any.whl (25 kB)
Collecting sqlparse>=0.2.2
```

```
(env) D:\bank_be>pip install djangorestframework
Collecting djangorestframework
  Using cached djangorestframework-3.12.4-py3-none-any.whl (957 kB)
Requirement already satisfied: django>=2.2 in d:\bank_be\env\lib\site-packages (
Requirement already satisfied: sqlparse>=0.2.2 in d:\bank_be\env\lib\site-packag
```

Posteriormente, se crea el proyecto con el comando *django-admin startproject projectName*, cambiando la palabra *projectName* por el nombre que se le desea dar al proyecto. En este caso se utilizará el nombre *authProject*.

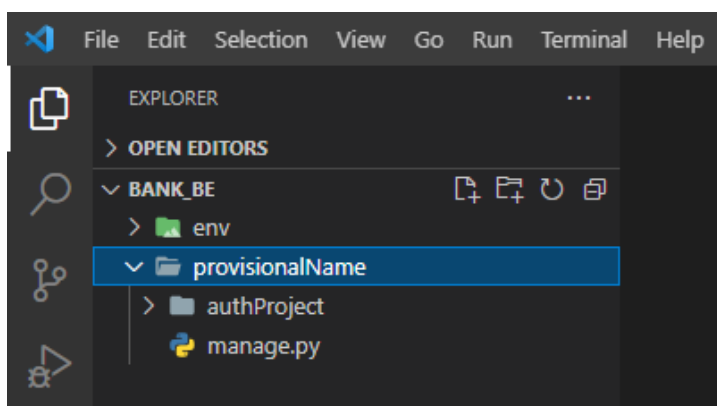
```
(env) D:\bank_be>django-admin startproject authProject

(env) D:\bank_be>
```

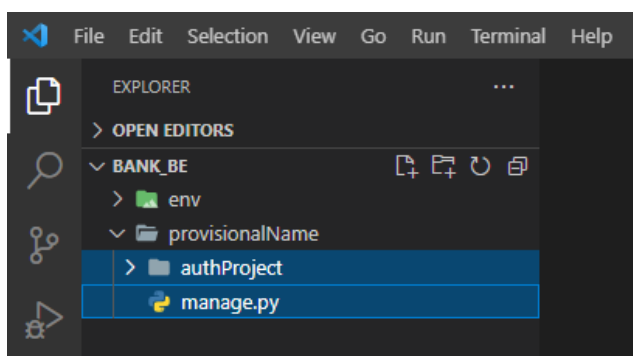
Una vez se ha ejecutado el comando, se habrá creado una carpeta con un archivo de Python (.py) y otra carpeta interna con el mismo nombre:



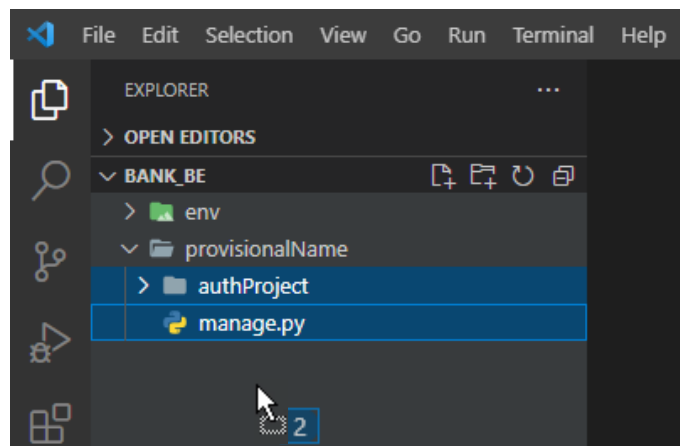
Para evitar futuras confusiones, es importante extraer la carpeta y el archivo `.py` fuera de la carpeta `authProject`. Para ello, primero se debe cambiar el nombre de la carpeta `authProject` externa por cualquier otro nombre:



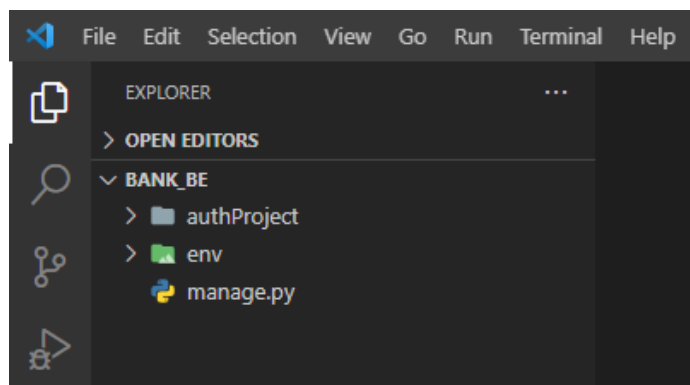
Luego, se selecciona la carpeta `authProject` y el archivo `manage.py`, manteniendo la tecla `CTRL`:



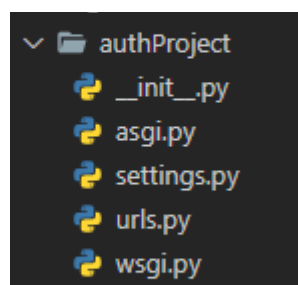
Ahora, manteniendo presionado el botón izquierdo del mouse, se arrastran ambos archivos fuera de la carpeta `provisionalName`:



Finalmente, se elimina manualmente la carpeta *provisionalName*. De esta forma, el proyecto debería verse de la siguiente manera:



Ahora, al abrir la carpeta *authProject* se pueden ver los siguientes archivos:



Todos estos archivos son de configuración. Sin embargo, los únicos que se van a utilizar son *settings.py* y *urls.py*. Estos sirven respectivamente para configurar los paquetes e instalaciones del proyecto, y para mapear las funcionalidades del componente a una URL del servidor. Por ahora, la única configuración del proyecto que se debe realizar es ingresar al archivo *settings.py* y buscar el arreglo con el nombre *INSTALLED\_APPS*:



```
settings.py X
authProject > settings.py
30
31 # Application definition
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 ]
41
```

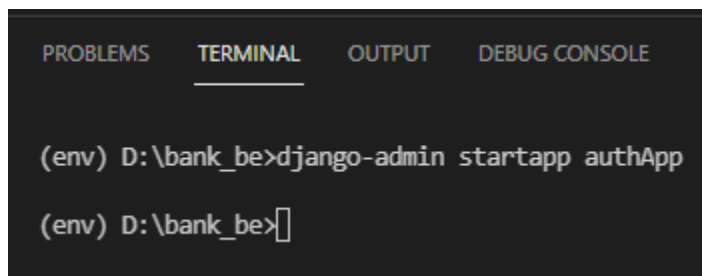
Allí, se debe insertar Django REST Framework como una aplicación instalada del proyecto, añadiendo `'rest_framework'` a la lista, y guardando los cambios del archivo:

```
settings.py X
authProject > settings.py > ...
30
31 # Application definition
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 ]
42
```

Notar que la coma luego de la cadena se utiliza para no olvidar separar los elementos que se añadan más adelante.

## Creación de la aplicación Django

Un proyecto puede tener múltiples aplicaciones; sin embargo, para el caso de estudio solo se utilizará una aplicación. Para crearla, se utiliza el comando `django-admin startapp appName`, cambiando la palabra *appName* por el nombre que se le desea dar a la aplicación. En este caso se utilizará con el nombre *authApp*:

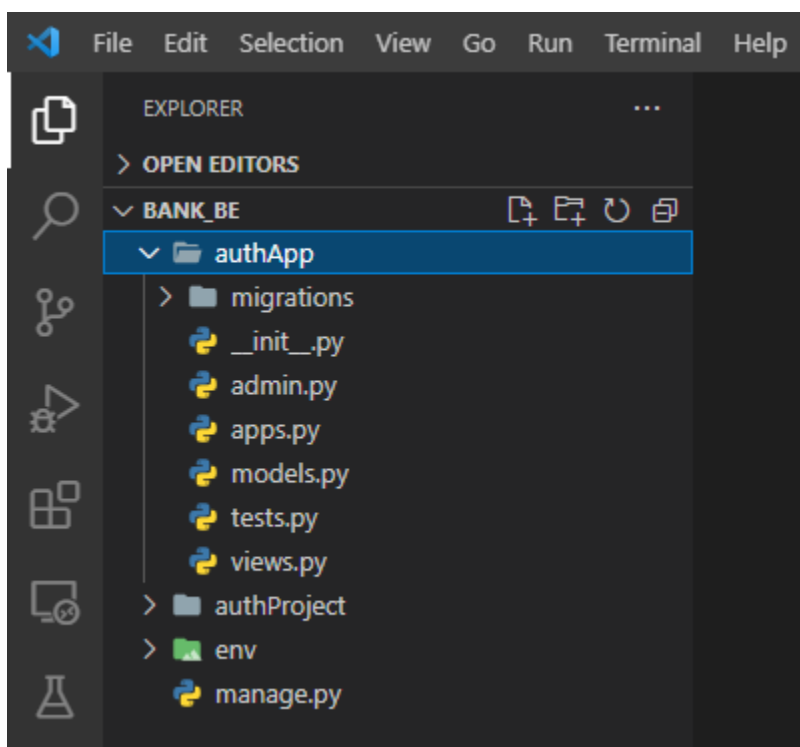


```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

(env) D:\bank_be>django-admin startapp authApp

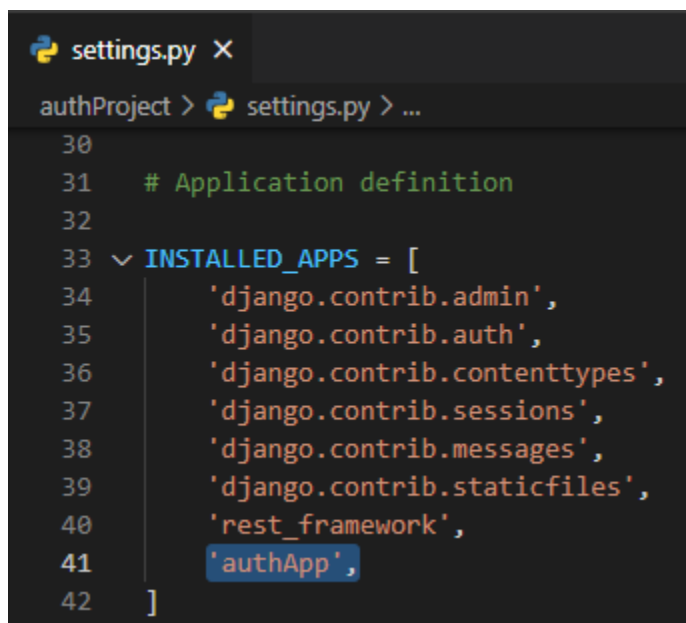
(env) D:\bank_be>
```

Una vez se ha ejecutado el comando, se habrá creado una carpeta con varios archivos de Python y la carpeta *migrations*:



La carpeta *migrations* es donde se almacenarán las migraciones de la base de datos, y los archivos allí presentes son aquellos archivos mínimos necesarios para que el componente funcione correctamente. De estos, los que se van a utilizar son los archivos *admin.py*, *models.py* y *views.py*. Cuyo nombre indica su función: El primero se utiliza para indicar a Django qué modelos utilizar en el proyecto, el segundo se utiliza para especificar dichos modelos, y el tercero se utiliza para especificar las vistas de la aplicación.

Cada vez que se crea una aplicación nueva, se debe configurar el proyecto para que este la reconozca y para que pueda mapear una URL a su funcionalidad en el archivo [urls.py](#). Para hacer dicha configuración, se debe dirigir al archivo [settings.py](#), dentro de la carpeta [authProject](#) y se debe editar la lista [INSTALLED\\_APPS](#) de la misma forma que se hizo anteriormente, añadiendo esta vez la cadena ['authApp'](#):



```
settings.py X
authProject > settings.py > ...
30
31 # Application definition
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 ]
```

### Ejecución del Componente

Ahora, para asegurarse que el proyecto se configuró correctamente, se ejecutará el servidor del asociado al componente. De esta forma, en la terminal se debe ejecutar el comando [python manage.py runserver](#), el cual se encarga de ejecutar el servidor localmente en la dirección [http://127.0.0.1:8000/](#):



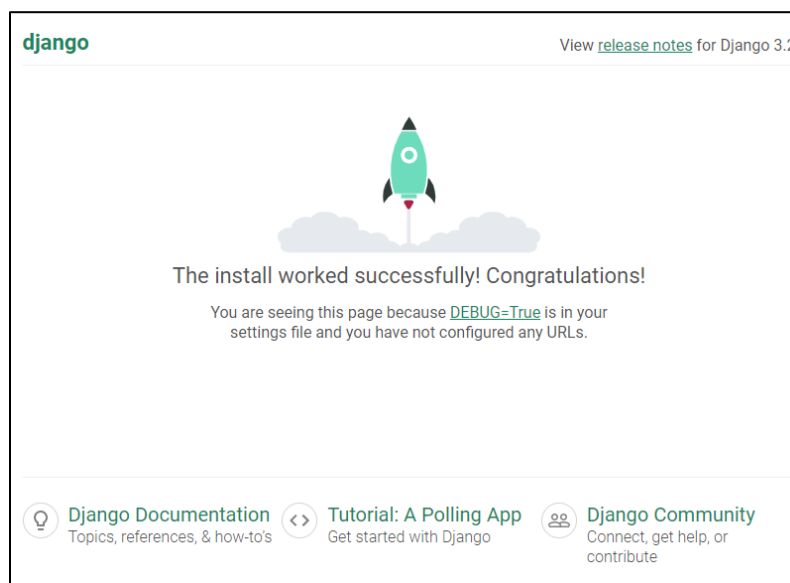
```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

(env) D:\bank_be>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

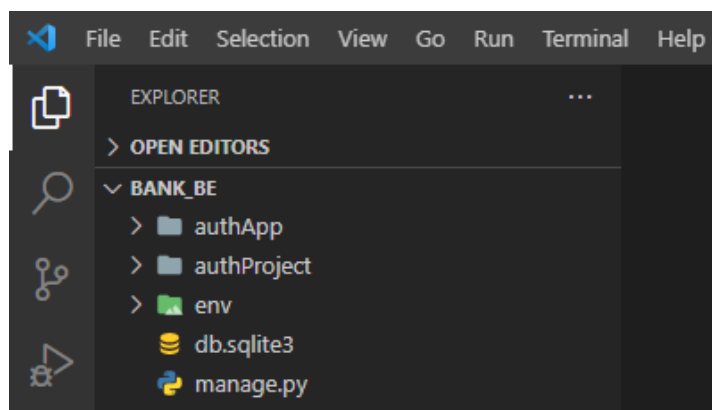
You have 18 unapplied migration(s). Your project may not work properly
Run 'python manage.py migrate' to apply them.
September 18, 2021 - 23:13:24
Django version 3.2.7, using settings 'authProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
█
```

Si se ingresa a la dirección en que se ejecuta el servidor, desde un navegador, se podrá ver la siguiente página:



Esto indica que las instalaciones y la creación del proyecto se completaron satisfactoriamente. En este momento, si se desea se puede detener la ejecución del servidor para no consumir recursos del computador innecesariamente. Para ello, en la terminal se deben pulsar las teclas **CTRL+C** al tiempo.

Notar que con la ejecución del componente se creó un archivo llamado **db.sqlite3**, esto se debe a que los proyectos creados por Django se conectan por defecto a una base de datos local SQLite.



Una vez se ha detenido la ejecución del servidor, se puede eliminar este archivo ya que más adelante, para el caso de estudio, se usará la base de datos remota PostgreSQL, desplegada en Heroku. Esta configuración se realizará en sesiones futuras.