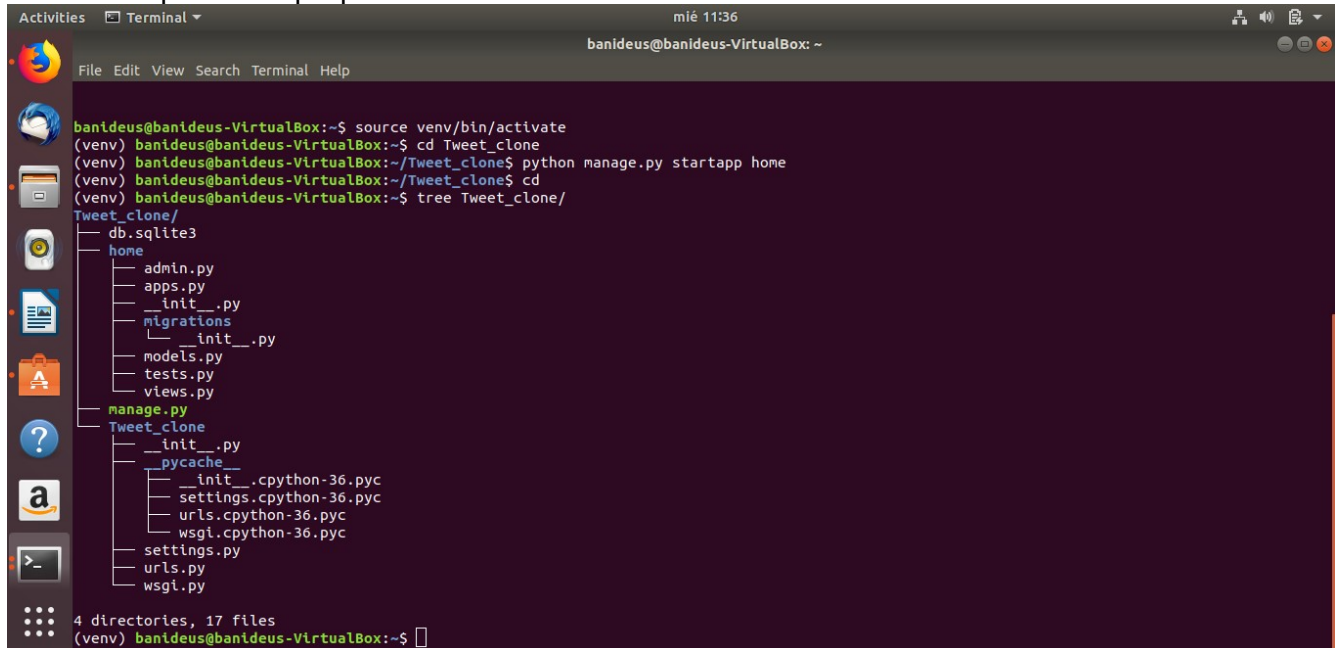


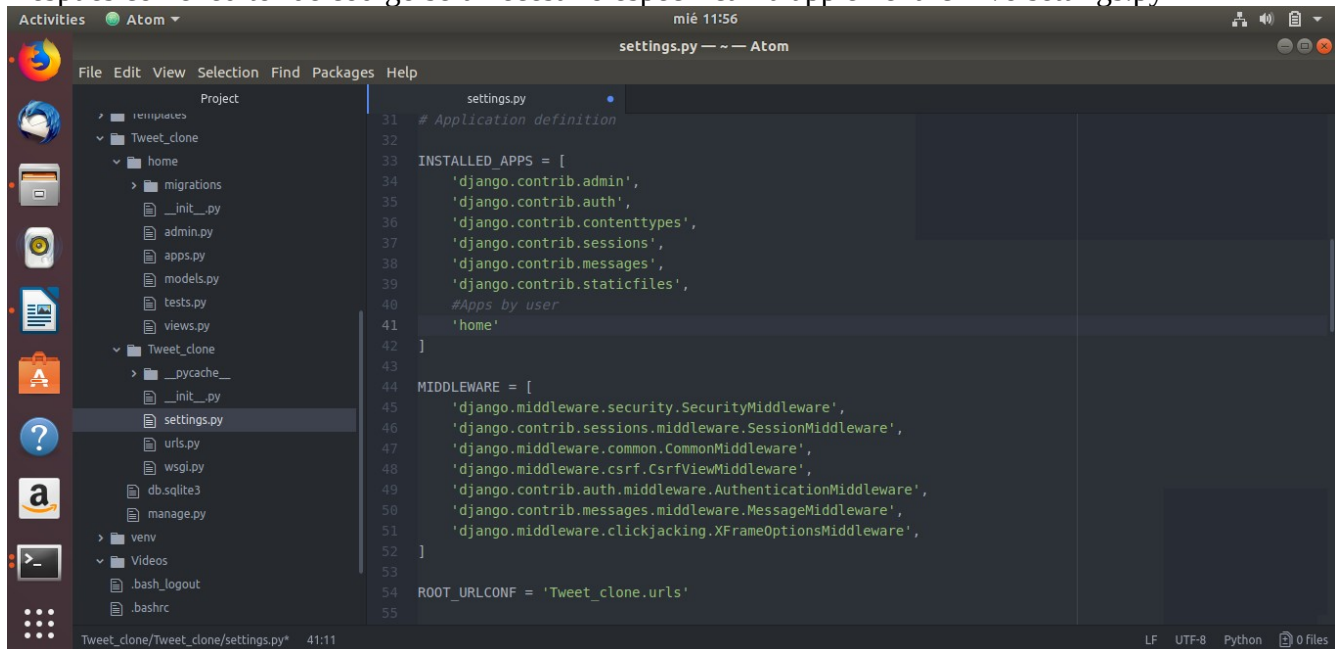
Práctica 2: Creación de aplicación

Para crear una nueva aplicación en django es necesario utilizar el comando startapp, esto nos creara una nueva aplicación que podremos visualizar el arbol de directorios con el comando tree



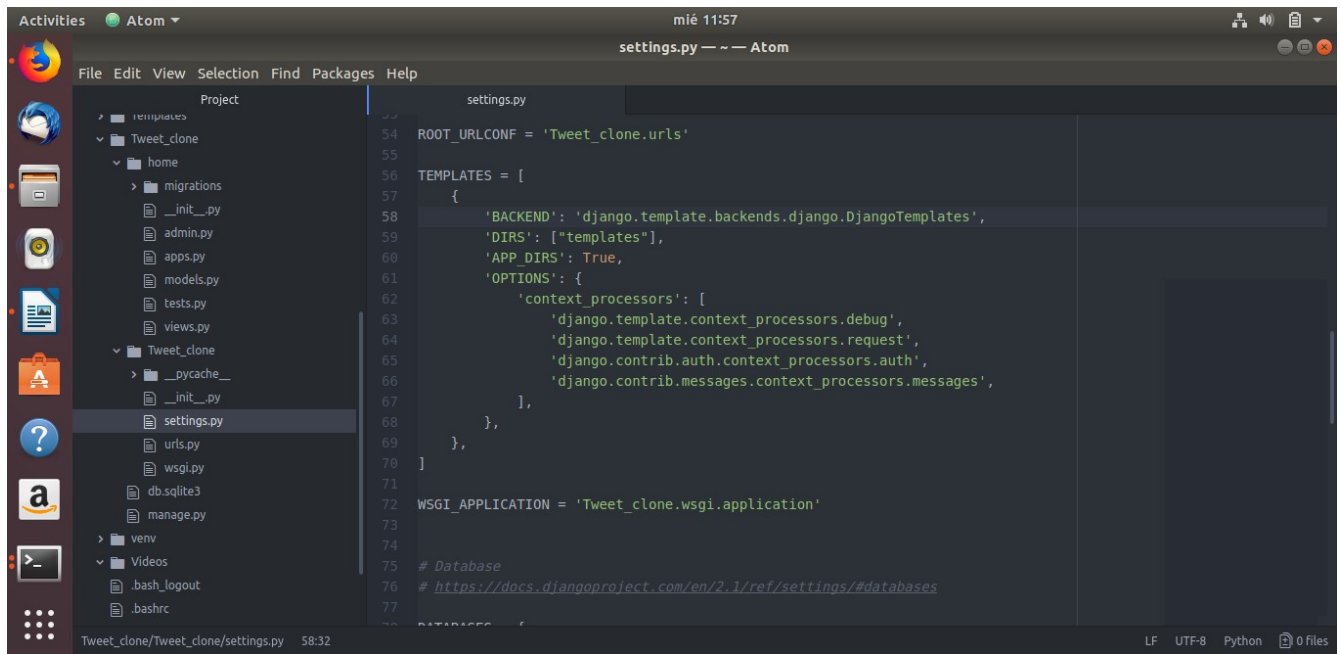
```
banideus@banideus-VirtualBox:~$ source venv/bin/activate
(venv) banideus@banideus-VirtualBox:~$ cd Tweet_clone
(venv) banideus@banideus-VirtualBox:~/Tweet_clone$ python manage.py startapp home
(venv) banideus@banideus-VirtualBox:~/Tweet_clone$ cd
(venv) banideus@banideus-VirtualBox:~$ tree Tweet_clone/
Tweet_clone/
├── db.sqlite3
├── home
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── __init__.py
│   │   └── 0001_initial.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
├── Tweet_clone
│   ├── __init__.py
│   ├── pycache
│   │   ├── __init__.cpython-36.pyc
│   │   ├── settings.cpython-36.pyc
│   │   ├── urls.cpython-36.pyc
│   │   └── wsgi.cpython-36.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
4 directories, 17 files
(venv) banideus@banideus-VirtualBox:~$
```

Despues con el editor de codigo será necesario especificar la app en el archiivo settings.py



```
settings.py
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     # Apps by user
41     'home'
42 ]
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 ROOT_URLCONF = 'Tweet_clone.urls'
55
```

Asi como también especificar de donde se tomarán los templates, que sera de una carpeta con el mismo nombre.

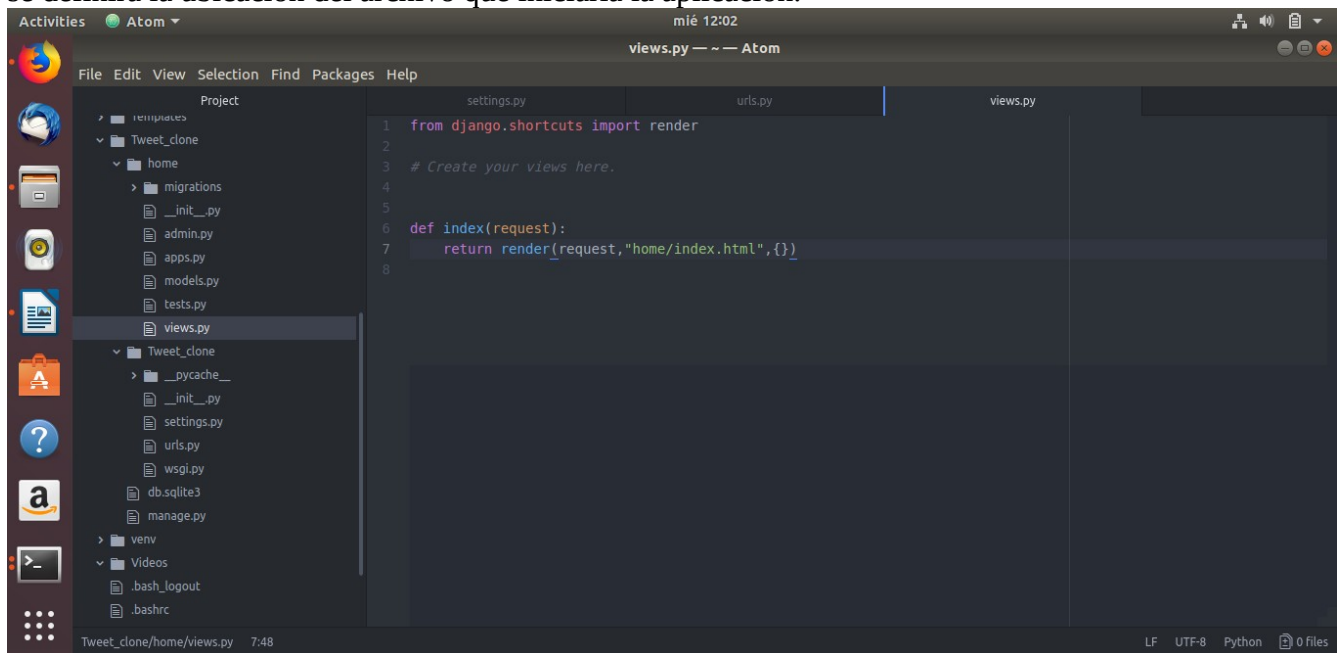


The screenshot shows the Atom text editor with the 'settings.py' file open. The left sidebar displays a project tree for 'Tweet_clone' with a file explorer. The main editor area shows the following Python code:

```
54 ROOT_URLCONF = 'Tweet_clone.urls'
55
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': ["templates"],
60         'APP_DIRS': True,
61         'OPTIONS': {
62             'context_processors': [
63                 'django.template.context_processors.debug',
64                 'django.template.context_processors.request',
65                 'django.contrib.auth.context_processors.auth',
66                 'django.contrib.messages.context_processors.messages',
67             ],
68         },
69     ],
70 ]
71
72 WSGI_APPLICATION = 'Tweet_clone.wsgi.application'
73
74
75 # Database
76 # https://docs.djangoproject.com/en/2.1/ref/settings/#databases
77
78 # Password validation
79 # https://docs.djangoproject.com/en/2.1/ref/settings/#password-validation
```

The status bar at the bottom indicates the file is 'Tweet_clone/Tweet_clone/settings.py' at line 58, column 32, using the LF line ending, UTF-8 encoding, Python syntax, and 0 files are open.

Despues habrá que editar el archivo views.py para crear una nueva funcion, la cual sera el index donde se definira la ubicación del archivo que iniciaria la aplicación.

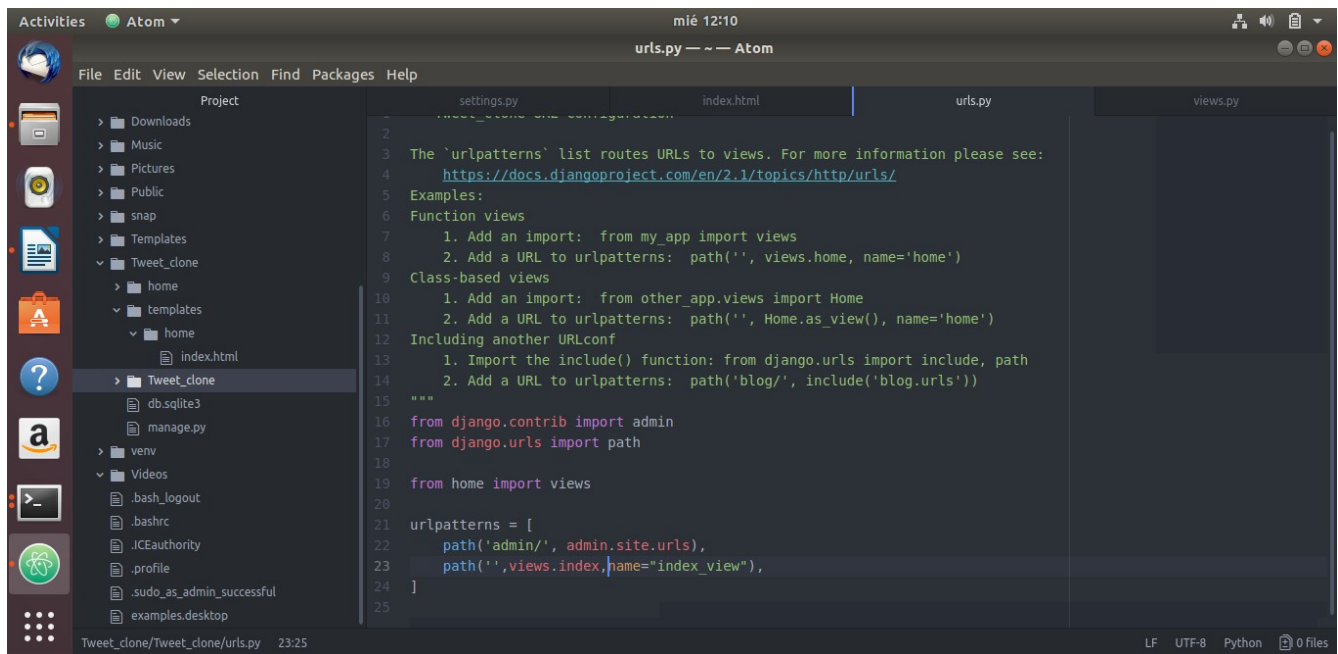


The screenshot shows the Atom text editor with the 'views.py' file open. The left sidebar displays the same project tree for 'Tweet_clone'. The main editor area shows the following Python code:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5
6 def index(request):
7     return render(request, "home/index.html", {})
8
```

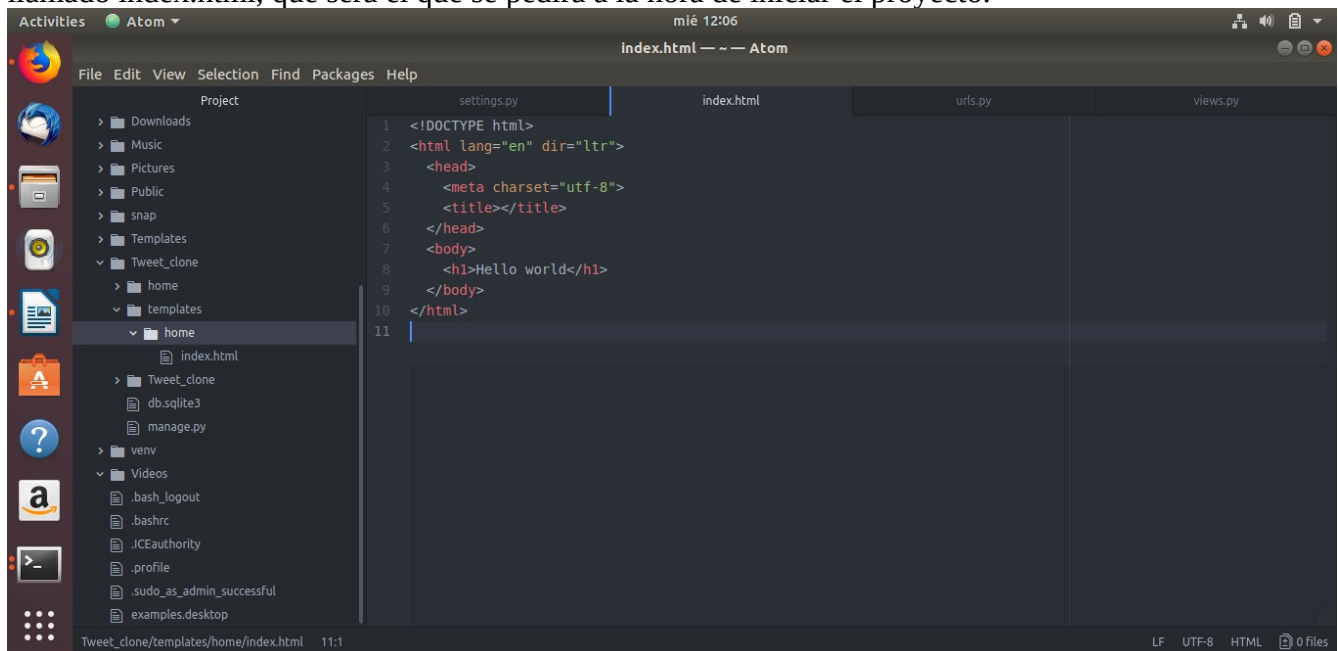
The status bar at the bottom indicates the file is 'Tweet_clone/home/views.py' at line 7, column 48, using the LF line ending, UTF-8 encoding, Python syntax, and 0 files are open.

También será necesario editar el archivo de `urls.py` para crear un nuevo path donde se enviará el parametro del archivo `index`, así como se importará las views de la carpeta `home`.



```
1 # -*- coding: utf-8 -*-
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4 https://docs.djangoproject.com/en/2.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 from home import views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', views.index, name='index_view'),
24 ]
25
```

Finalmente se crea la carpeta `templates` y dentro de ella la carpeta `home` y dentro de ella un archivo llamado `index.html`, que será el que se pida a la hora de iniciar el proyecto.



```
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   </head>
7   <body>
8     <h1>Hello world</h1>
9   </body>
10 </html>
11
```

Ejecutamos el servidor y accedemos al navegador con `localhost:8000` y veremos el resultado de nuestra aplicación.

