

## 1. Instalación PIP, Python & Django

- Descargar **Python 3.6**  
Se recomienda configurar en carpeta d:\ y path, durante la instalación
- Descargar **PIP** y ejecutar en la carpeta donde bajo

```
python get-pip.py
```

O

```
python -m pip install -U pi
```

- Configurar ambiente MSDOS, y rutas a Python

C:\Program Files\Python36

C:\Program Files\Python36\Scripts

C:\Program Files\Python36\Lib\site-packages

C:\Program Files\Python36\Lib\site-packages\django\bin

```
C:\Windows\System32;C:\Python27;C:\Python27\python.exe;C:\Python\27\Lib\site-packages;C:\Python27\Lib\site-packages\django\bin;c:\Python27\Scripts;
```

- Instalar **virtualenv**.

```
pip install virtualenv
```

- Descargar **Django**, la versión mas reciente

```
pip install django==2.0.5
```

## 2. Empezar Proyecto Nuevo

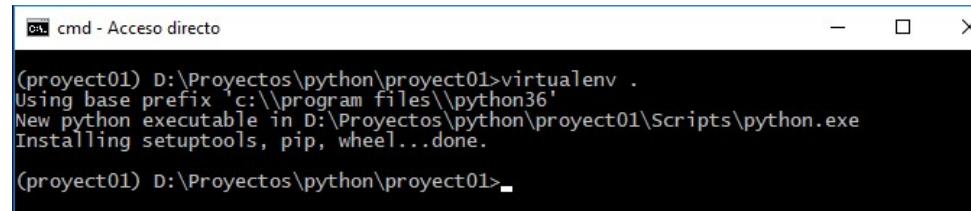
- Se debe cambiar la ruta al escritorio del usuario o en su defecto, al configurar MS-DOS, asegurar iniciar en D:\usuarios\" el usuario" \escritorio o en todo caso utilice la ruta mas confiable para guardar sus proyectos.

D:\Proyectos\python>**mkdir proyecto01**

D:\Proyectos\python>**cd proyecto01**

- Creando entorno virtual

D:\Proyectos\python\proyecto01>**virtualenv .**



```
(proyecto01) D:\Proyectos\python\proyecto01>virtualenv .
Using base prefix 'c:\\program files\\python36'
New python executable in D:\\Proyectos\\python\\proyecto01\\Scripts\\python.exe
Installing setuptools, pip, wheel...done.
(proyecto01) D:\\Proyectos\\python\\proyecto01>
```

- Ejecutando scripts

D:\\Proyectos\\python\\proyecto01>.\Scripts\\activate

D:\\Proyectos\\python\\proyecto01>**pip freeze**

```
(proyecto01) D:\Proyectos\python>mkdir proyecto01
(proyecto01) D:\Proyectos\python>cd proyecto01
(proyecto01) D:\Proyectos\python\proyecto01>virtualenv .
Using base prefix 'c:\\program files\\python36'
New python executable in D:\Proyectos\python\proyecto01\Scripts\python.exe
Installing setuptools, pip, wheel...done.
(proyecto01) D:\Proyectos\python\proyecto01>.\Scripts\activate
(proyecto01) D:\Proyectos\python\proyecto01>pip freeze
(proyecto01) D:\Proyectos\python\proyecto01>
```

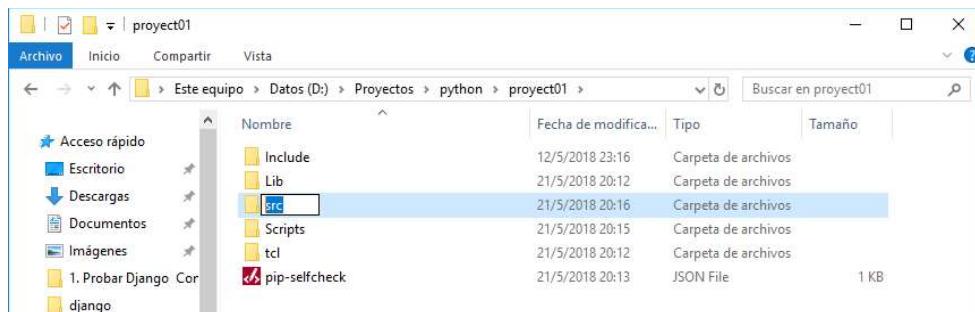
- Instalar Django en el ambiente virtual, en este caso la versión 2.0.5 mas reciente  
D:\Proyectos\python\proyecto01>**pip install django==2.0.5**

```
(proyecto01) D:\Proyectos\python\proyecto01>pip install django==2.0.5
Collecting django==2.0.5
  Using cached https://files.pythonhosted.org/packages/23/91/2245462e57798e9251de87c88b2b8f996d10ddcb68206a8a020561ef7bd3/Django-2.0.5-py3-none-any.whl
Collecting pytz (from django==2.0.5)
  Using cached https://files.pythonhosted.org/packages/dc/83/15f7833b70d3e067ca91467ca245bae0f6fe56ddc7451aa0dc5606b120f2/pytz-2018.4-py3-none-any.whl
Installing collected packages: pytz, django
Successfully installed django-2.0.5 pytz-2018.4
(proyecto01) D:\Proyectos\python\proyecto01>
```

- Iniciamos el proyecto recientemente creado  
D:\Proyectos\python\proyecto01>**python .\Scripts\django-admin.py startproject proyecto01**

```
(proyecto01) D:\Proyectos\python\proyecto01>python .\Scripts\django-admin.py startproject proyecto01
(proyecto01) D:\Proyectos\python\proyecto01>
```

- Una vez instalado debemos verificar en el escritorio y cambiar la ruta repetitiva 'proyecto01' por 'src' que se encuentran la carpeta de mismo nombre, para evitar confusiones.



```
Selezionar cmd - Acceso directo

(proyecto01) D:\Proyectos\python\proyecto01>python .\Scripts\django-admin.py startproject proyecto01
(proyecto01) D:\Proyectos\python\proyecto01>dir
El volumen de la unidad D es Datos
El n mero de serie del volumen es: 922C-5B01

Directorio de D:\Proyectos\python\proyecto01

21/05/2018 20:18 <DIR> .
21/05/2018 20:18 <DIR> ..
21/05/2018 23:16 <DIR> Include
21/05/2018 20:12 <DIR> Lib
21/05/2018 20:13 61 pip-selfcheck.json
21/05/2018 20:15 <DIR> Scripts
21/05/2018 20:16 <DIR> SRC
21/05/2018 20:12 <DIR> tcl
               1 archivos          61 bytes
               7 dirs  589.728.485.376 bytes libres

(proyecto01) D:\Proyectos\python\proyecto01>
```

- Ejecutando servidor de desarrollo  
**D:\Proyectos\python\proyecto01\src>python manage.py runserver**

```
Selezionar cmd - Acceso directo - python manage.py runserver

(proyecto01) D:\Proyectos\python\proyecto01>cd src
(proyecto01) D:\Proyectos\python\proyecto01\src>dir
El volumen de la unidad D es Datos
El n mero de serie del volumen es: 922C-5B01

Directorio de D:\Proyectos\python\proyecto01\src

21/05/2018 20:16 <DIR> .
21/05/2018 20:16 <DIR> ..
21/05/2018 20:16      556 manage.py
21/05/2018 20:16 <DIR> proyecto01
               1 archivos          556 bytes
               3 dirs  589.728.911.360 bytes libres

(proyecto01) D:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

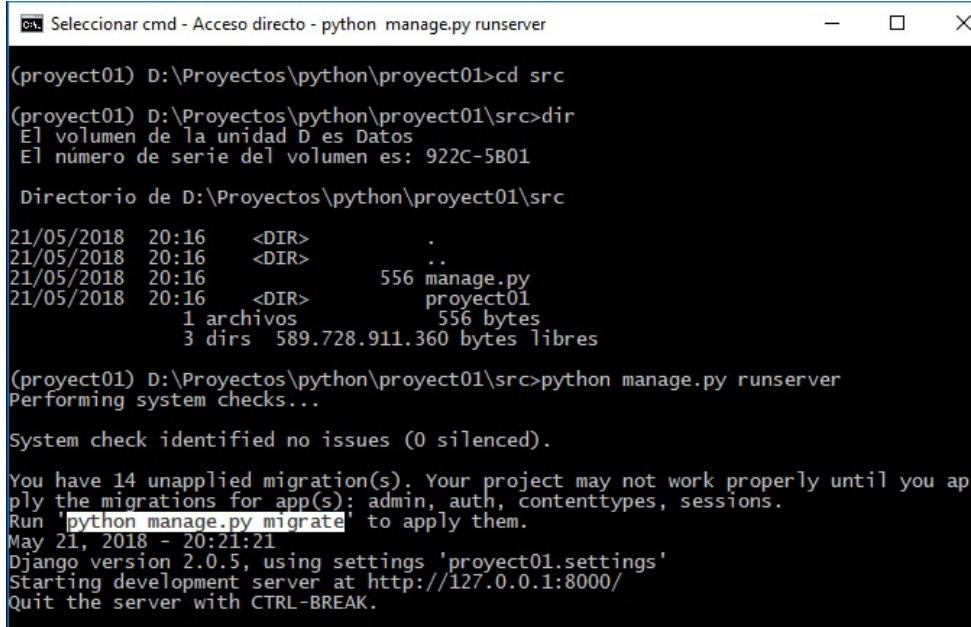
You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 21, 2018 - 20:21:21
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Copiamos ruta de desplegué y pegamos en browser:  
<http://127.0.0.1:8000> o <http://localhost:8000>



### 3. Primera Migración

- Se debe ejecutar cuando nos muestra el mensaje al cargar el servidor web. De esta forma sincronizamos el proyecto con la base de datos.



```
(proyecto01) D:\Proyectos\python\proyecto01>cd src
(proyecto01) D:\Proyectos\python\proyecto01\src>dir
El volumen de la unidad D es Datos
El n mero de serie del volumen es: 922C-5B01

Directorio de D:\Proyectos\python\proyecto01\src

21/05/2018 20:16 <DIR> .
21/05/2018 20:16 <DIR> ..
21/05/2018 20:16 556 manage.py
21/05/2018 20:16 <DIR> proyecto01
    1 archivos 556 bytes
    3 dirs 589.728.911.360 bytes libres

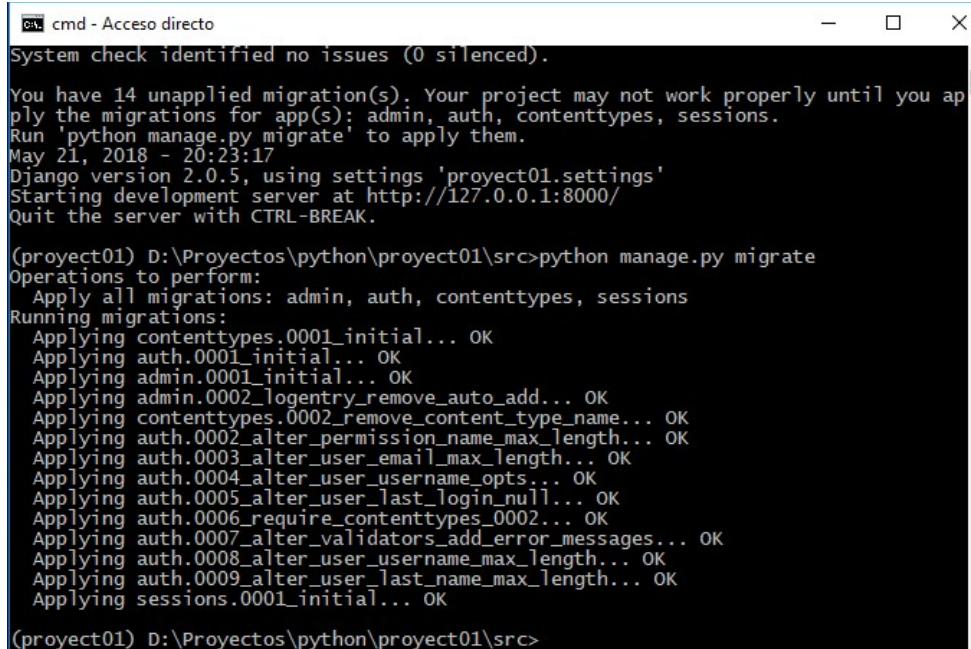
(proyecto01) D:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 21, 2018 - 20:21:21
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Por lo tanto, cerramos el servidor y ejecutamos.

D:\Proyectos\python\proyecto01\src>**python manage.py migrate**

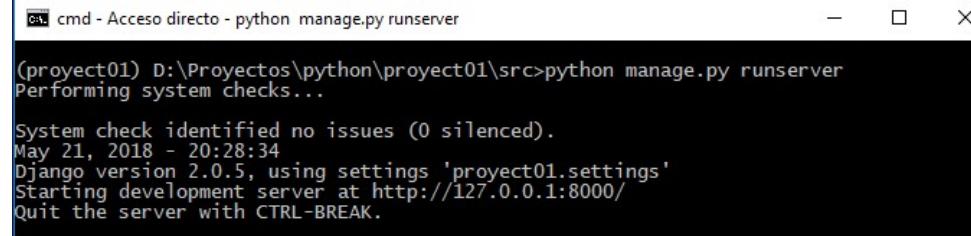


```
System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 21, 2018 - 20:23:17
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

(proyecto01) D:\Proyectos\python\proyecto01\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK

(proyecto01) D:\Proyectos\python\proyecto01\src>
```



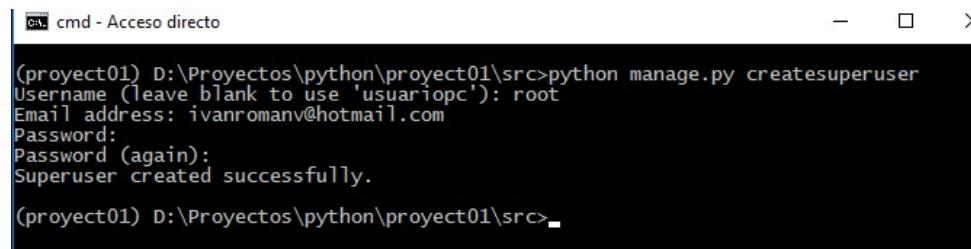
```
(proyecto01) D:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
May 21, 2018 - 20:28:34
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Usando el editor de textos se procede a modificar/crear el archivo, de preferencia aquí se está trabajando con db.sqlite3, que es la base que viene por omisión en la implementación.  
Habrá que guardar el proyecto en la carpeta proyecto01 en, pero dependerá del editor que se esté utilizando a nivel de proyecto para acceder a él más rápido y directamente.

## 4. Crear superusuario + Administración de Django

- Creamos el superusuario con el siguiente comando:  
D:\Proyectos\python\proyecto01\src>**python manage.py createsuperuser**



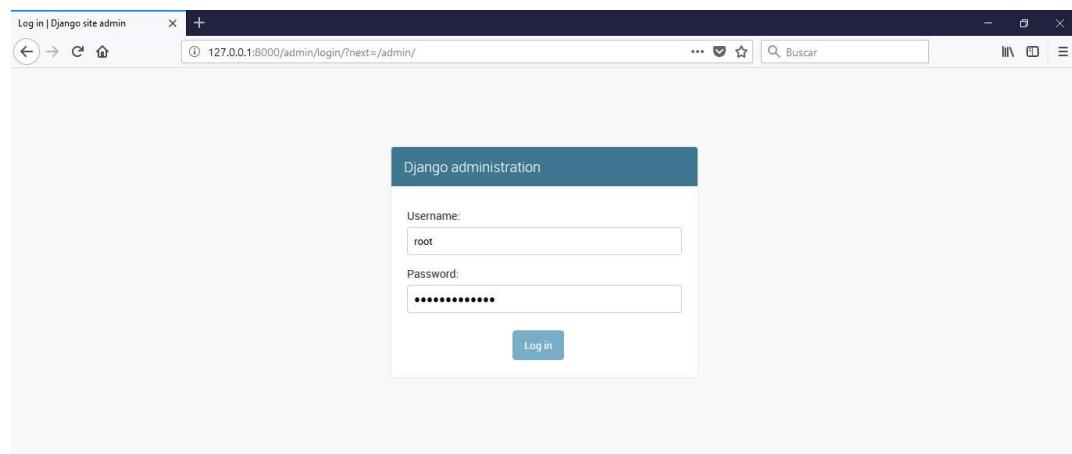
```
(proyecto01) D:\Proyectos\python\proyecto01\src>python manage.py createsuperuser
Username (leave blank to use 'usuariopc'): root
Email address: ivanroman@hotmail.com
Password:
Password (again):
Superuser created successfully.

(proyecto01) D:\Proyectos\python\proyecto01\src>
```

**Username :** admin  
**Email address :** [elusuario@hotmail.com](mailto:elusuario@hotmail.com)  
**Password :** sucontraseña  
**Password again:** sucontraseña

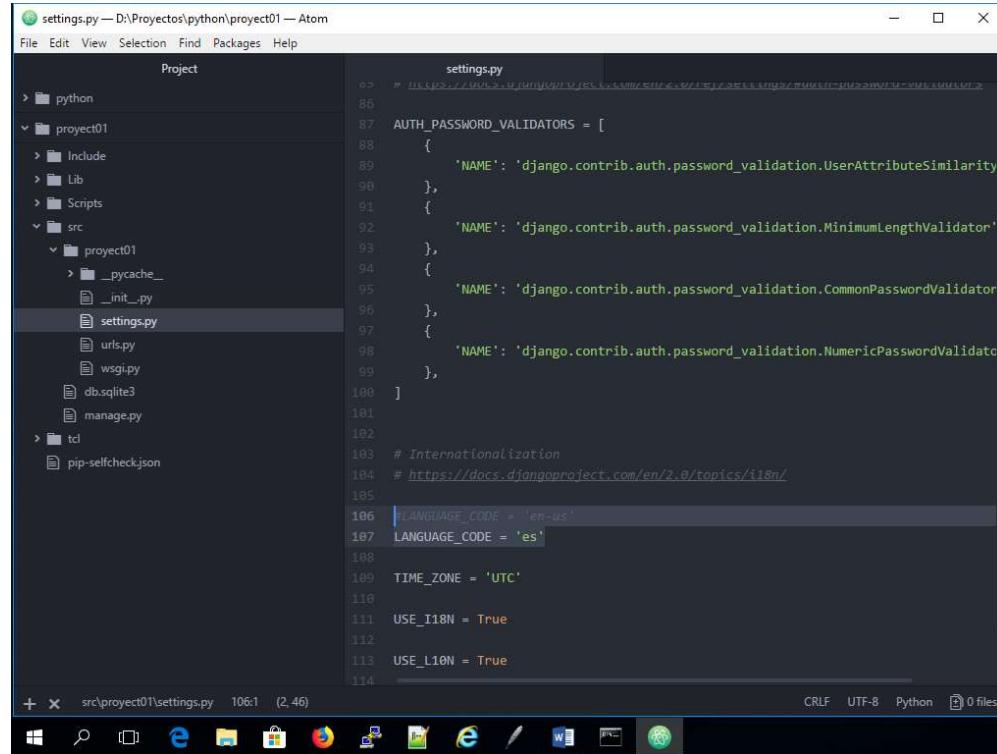
- Recargamos el servidor de Django a través del browser  
D:\Proyectos\python\proyecto01\src>**python manage.py runserver**

<http://127.0.0.1:8000/admin>



Colocamos el username, password y accedemos a la página principal

- Para cambiar el idioma vamos al archivo **settings.py**, lo abrimos y en donde dice **LANGUAGE\_CODE = 'us-en'** lo cambiamos por 'es' para español.

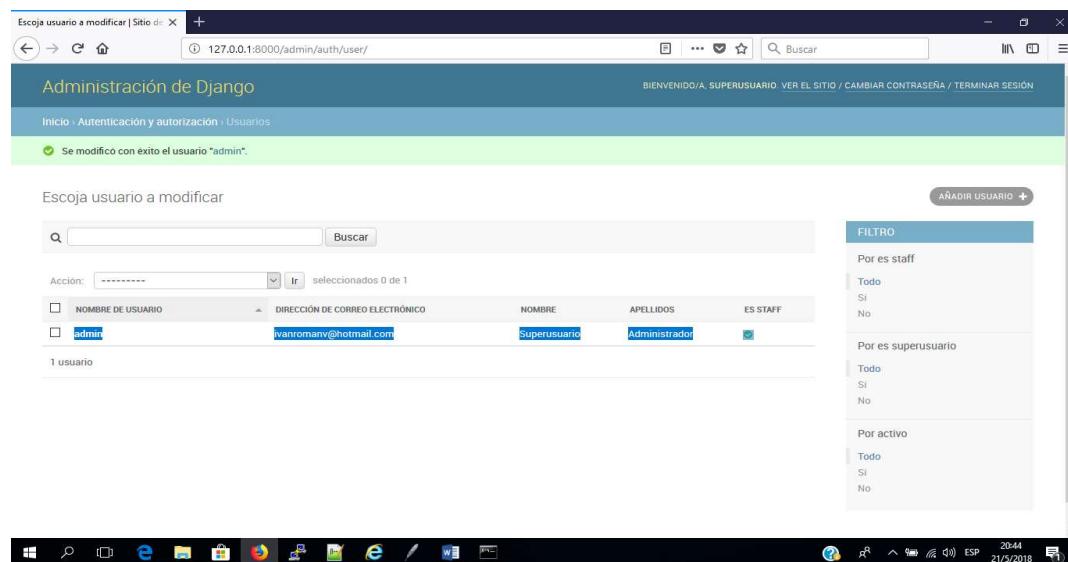


```

settings.py — D:\Proyectos\python\project01 — Atom
File Edit View Selection Find Packages Help
Project
src\project01\settings.py
87 AUTH_PASSWORD_VALIDATORS = [
88     {
89         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
90     },
91     {
92         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
93     },
94     {
95         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
96     },
97     {
98         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'
99     },
100 ]
101
102 # Internationalization
103 # https://docs.djangoproject.com/en/2.0/topics/i18n/
104
105 #LANGUAGE_CODE = 'en-us'
106 LANGUAGE_CODE = 'es'
107
108 TIME_ZONE = 'UTC'
109
110 USE_I18N = True
111
112 USE_L10N = True
113
114
src\project01\settings.py 106:1 (2,46) CRLF UTF-8 Python 0 files

```

- Recargamos el browser y veremos que se cambió el idioma a español  
Aquí podemos crear usuarios, grupos y perfiles, los mismos que nos servirán para acceder a la administración de Django.



Escoja usuario a modificar | Sitio de ... +

127.0.0.1:8000/admin/auth/user/

Administración de Django

Bienvenido/a, SUPERUSUARIO, VER EL SITIO / CAMBIAR CONTRASEÑA / TERMINAR SESIÓN

Inicio · Autenticación y autorización · Usuarios

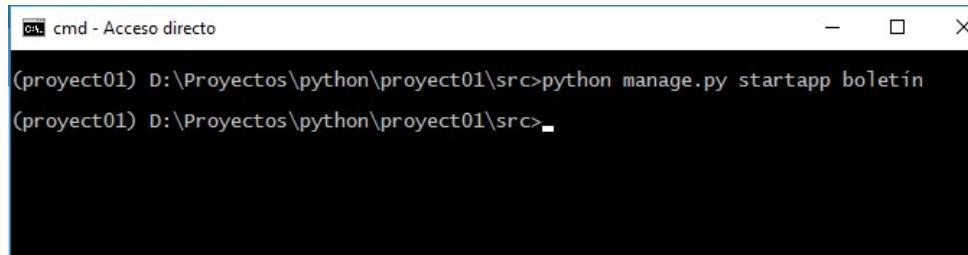
Se modificó con éxito el usuario "admin".

Escoja usuario a modificar					AÑADIR USUARIO +
<input type="text"/> Buscar					FILTRO
Acción: -----					Por es staff
<input type="checkbox"/> NOMBRE DE USUARIO <input type="checkbox"/> DIRECCIÓN DE CORREO ELECTRÓNICO <input type="checkbox"/> NOMBRE <input type="checkbox"/> APELLIDOS <input type="checkbox"/> ES STAFF					<input type="checkbox"/> Todo <input type="checkbox"/> Sí <input type="checkbox"/> No
<input type="checkbox"/> admin <input type="checkbox"/> evanromany@hotmail.com <input type="checkbox"/> Superusuario <input type="checkbox"/> Administrador					<input type="checkbox"/> Por es superusuario <input type="checkbox"/> Todo <input type="checkbox"/> Sí <input type="checkbox"/> No
1 usuario					<input type="checkbox"/> Por activo <input type="checkbox"/> Todo <input type="checkbox"/> Sí <input type="checkbox"/> No

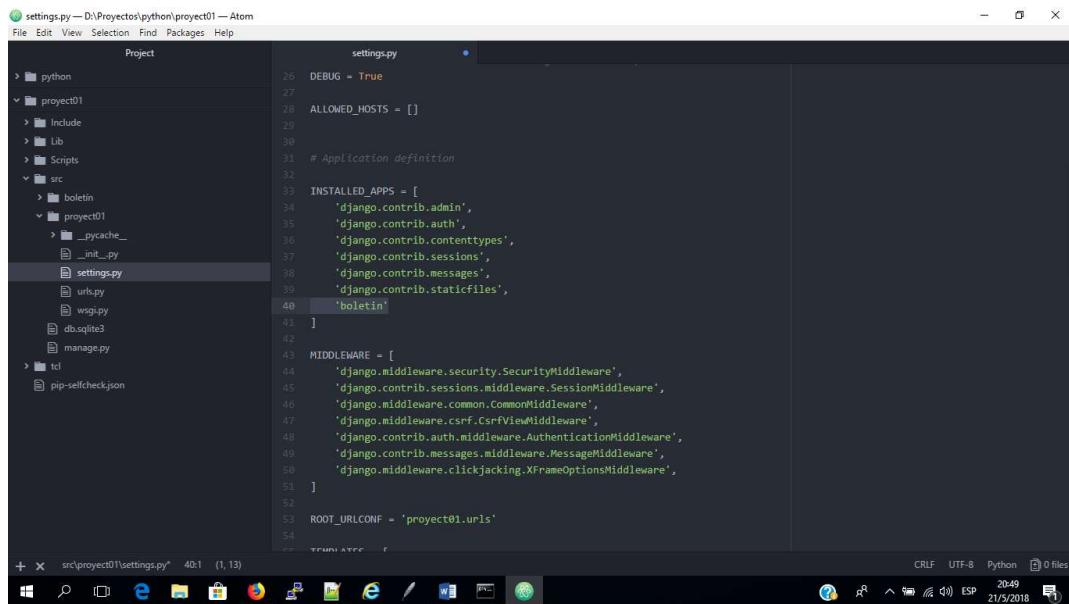
## 5. Primera Aplicación

- Debemos tener claro en que es una aplicación web y cual una web App
- Creamos la web app denominada boletín en la carpeta \proyecto01\src

```
D:\Proyectos\python\proyecto01>cd src
D:\Proyectos\python\proyecto01\src>python manage.py startapp boletin
```



- No vamos al editor en la carpeta proyecto01\src\project01\ archivo settings.py en donde agregamos la web app creada al final, es decir boletín debe ser colocado en INSTALLED\_APPS= [ .....  
'boletin',]



## 6. Primer Modelo

- Creando modelo para los campos en la web app creada denominada **boletin**, en su carpeta sobre el archivo **models.py**

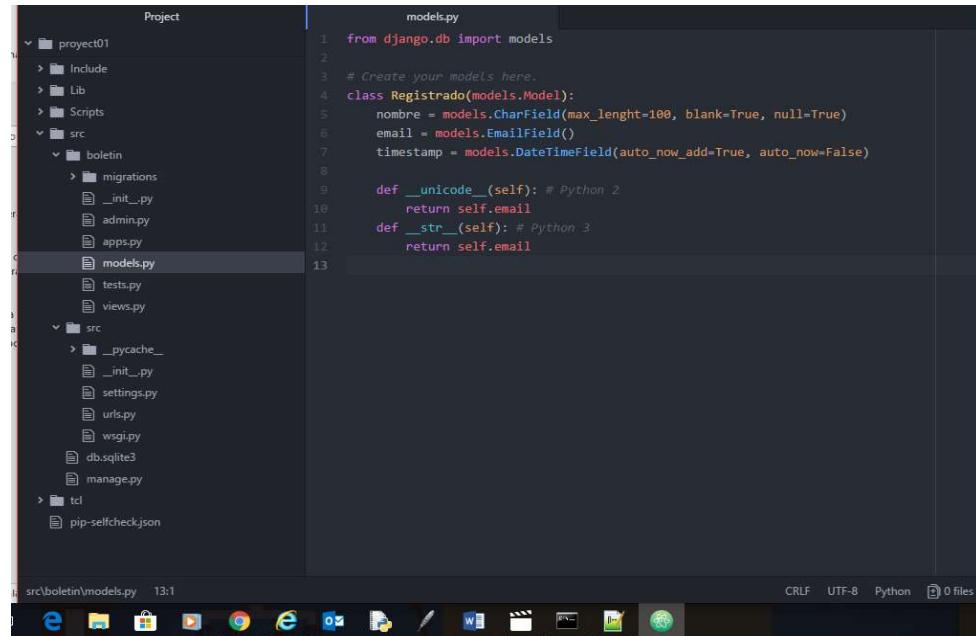
Archivo **models.py**:

```
#from __future__ import unicode_literals
from django.db import models
```

*# Create your models here.*

```
class Registrado(models.Model):
    nombre = models.CharField(max_length=100, blank=True, null=True)
    email = models.EmailField()
    timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)

    def __unicode__(self): # Python 2
        return self.email
    def __str__(self): # Python 3
        return self.email
```



Para obtener más información de los tipos de archivos y como usarlos consulta el link <https://docs.djangoproject.com/en/2.0/ref/models/fields/>

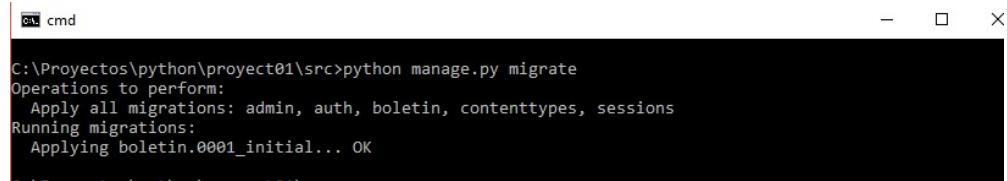
- Ahora se procede a empaquetarlas dentro del proyecto para el nuevo modelo  
D:\Proyectos\python\project01\src>**python manage.py makemigrations**

The terminal window shows the following output:

```
C:\Proyectos\python\project01\src>python manage.py makemigrations
Migrations for 'boletin':
  boletin\migrations\0001_initial.py
    - Create model Registrado

C:\Proyectos\python\project01\src>
```

- Ejecutamos los cambios para la creación de los campos en la base de datos db.sqlite3  
D:\Proyectos\python\project01\src>**python manage.py migrate**

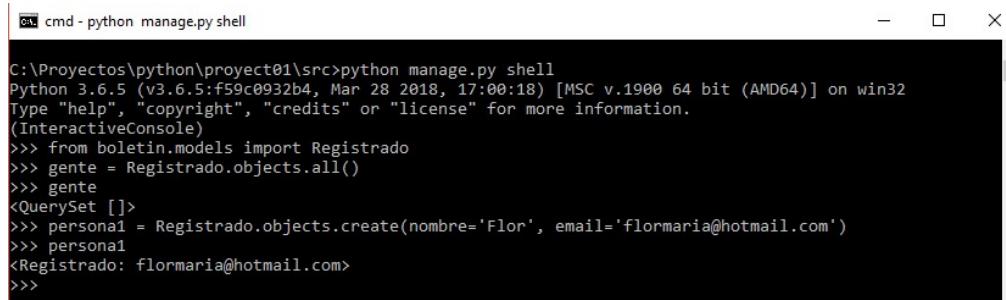


```
C:\Proyectos\python\proyect01\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, sessions
Running migrations:
  Applying boletin.0001_initial... OK
```

## 7. Crear Objetos en Python Shell + Registrar Model en Admin

- Empezamos a crear los objetos para guardarlos en la base de datos, lo cual podremos hacerlo mediante listas Python [] o accediendo directamente a la tabla boletin recientemente creada.
- Por medio de **listas []**, esto nos sirva para la importación de registros desde archivos planos.

```
C:\Proyectos\python\proyect01\src>python manage.py shell
>>> from boletin.models import Registrado
>>> gente = Registrado.objects.all()
>>> gente # Nos dice que es un variable tipo QuerySet []
>>> persona1 = Registrado.objects.create(nombre='Flor', email='flormaria@hotmail.com')
>>> persona1 # muestra el registro ingresado
<Registrado: flormaria@hotmail.com>
```



```
C:\Proyectos\python\proyect01\src>python manage.py shell
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from boletin.models import Registrado
>>> gente = Registrado.objects.all()
>>> gente
<QuerySet []>
>>> persona1 = Registrado.objects.create(nombre='Flor', email='flormaria@hotmail.com')
>>> persona1
<Registrado: flormaria@hotmail.com>
>>>
```

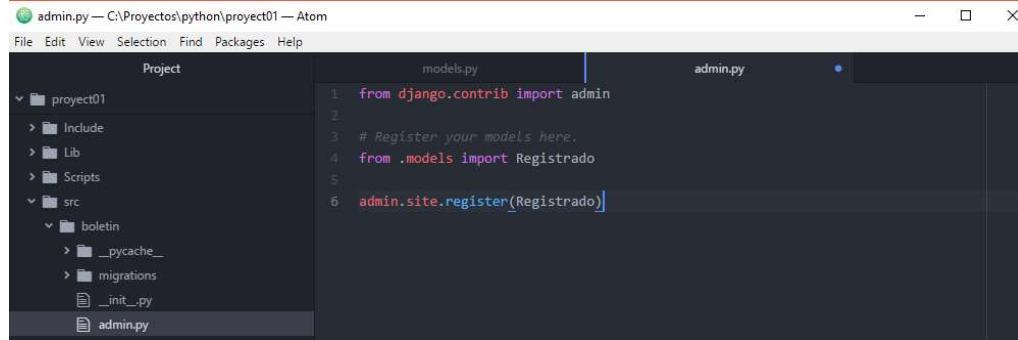
Notamos que nos mostrará el correo electrónico ya que se usa **unicode** el cual se refiere solo al campo email que deberá retornar. Si cambiamos por nombre, aparecerá ese campo a mostrar.

- Ahora se procede a registrar el modelo en la administración de Django escribiendo dentro del archivo **admin.py** de la carpeta boletin lo siguiente:

Archivo **admin.py**:

```
from django.contrib import admin
# Register your models here.
from .models import Registrado

admin.site.register(Registrado)
```



The screenshot shows the Atom code editor with two tabs open: 'models.py' and 'admin.py'. The 'models.py' file contains the following code:

```

1 from django.contrib import admin
2
3 # Register your models here.
4 from .models import Registrado
5
6 admin.site.register(Registrado)

```

The 'admin.py' file contains the following code:

```

from django.contrib import admin
# Register your models here.
from .models import Registrado
admin.site.register(Registrado)

```

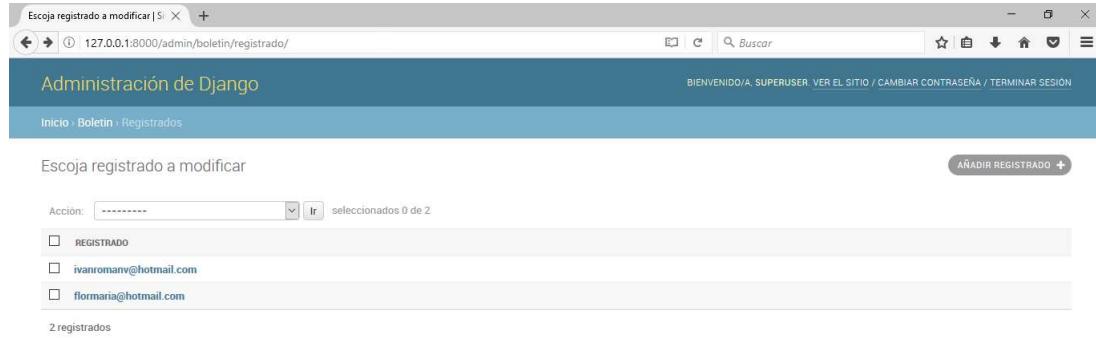
Grabamos y recargamos el servidor web <http://127.0.0.1:8000/admin>, en donde veremos la aplicación boletin, aparecen **Registrados**.

D:\Proyectos\python\proyecto01\src>python manage.py runserver



The screenshot shows the Django Admin interface at <http://127.0.0.1:8000/admin/>. The left sidebar shows the 'Boletin' application with 'Registrados' listed. The main area displays the 'AUTENTICACION Y AUTORIZACION' and 'BOLETIN' sections, each with 'Grupos', 'Usuarios', and 'Registrados' sub-sections. On the right, there are 'Acciones recientes' and 'Mis acciones' panels.

2. Por medio del sitio web admin de la aplicación boletin, en Registrados, en donde podremos agregar o eliminar registros.



The screenshot shows the 'Registrados' list view in the 'Boletin' application. It lists three entries: 'REGISTRADO', 'ivanromanv@hotmail.com', and 'flormaria@hotmail.com'. A search bar at the top is empty, and a button 'AÑADIR REGISTRADO' is visible on the right.

## 8. Personalizar Modelo en el Admin

- Si queremos personalizar los campos a mostrar dentro de la aplicación boletin, los mismos que se verán en el objeto Registrados, podemos hacer lo siguiente editando el archivo **admin.py** de la carpeta boletin.

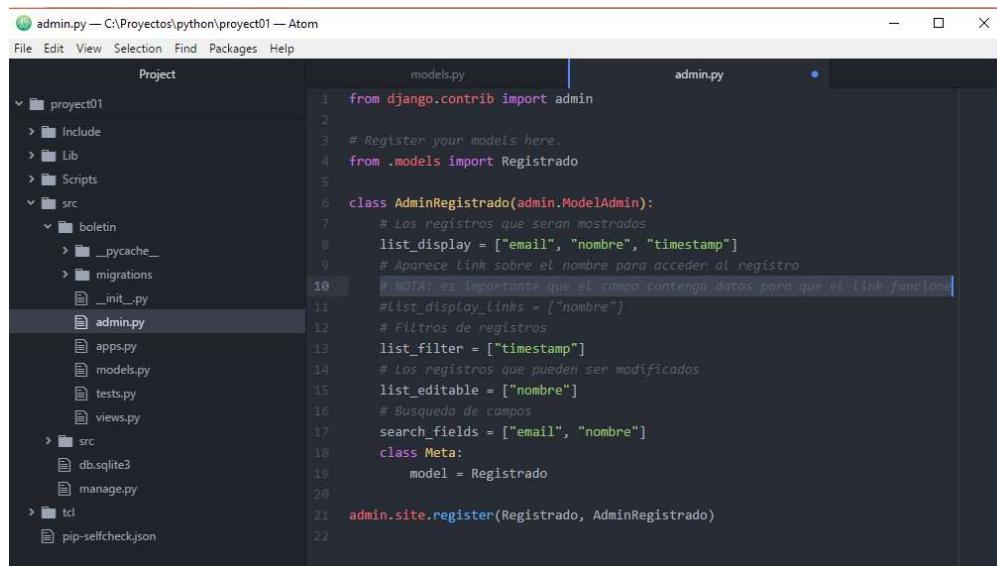
Archivo **admin.py**:

```
from django.contrib import admin
```

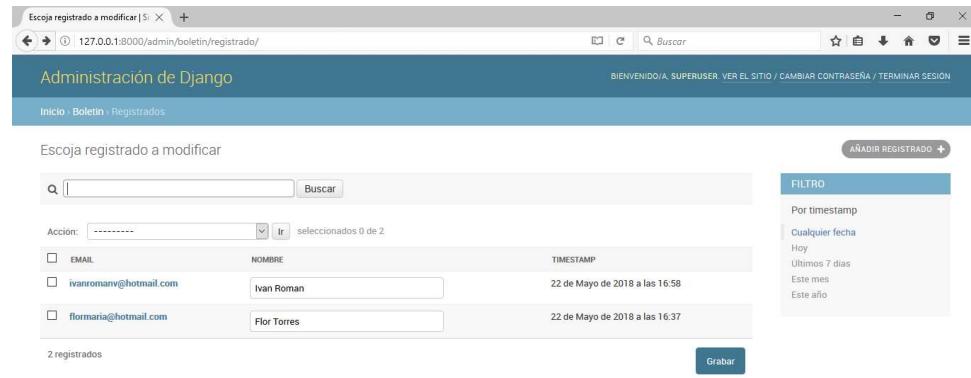
```
# Register your models here.
from .models import Registrado

class AdminRegistrado(admin.ModelAdmin):
    # Los registros que seran mostrados
    list_display = ["email", "nombre", "timestamp"]
    # Aparece link sobre el nombre para acceder al registro
    # NOTA: es importante que el campo contenga datos para que el link funcione
    #list_display_links = ["nombre"]
    # Filtros de registros
    list_filter = ["timestamp"]
    # Los registros que pueden ser modificados
    list_editable = ["nombre"]
    # Busqueda de campos
    search_fields = ["email", "nombre"]
    class Meta:
        model = Registrado

admin.site.register(Registrado, AdminRegistrado)
```



The screenshot shows the Atom code editor with two tabs open: 'models.py' and 'admin.py'. The 'admin.py' tab contains the Python code provided above. The 'models.py' tab is partially visible on the left.



The screenshot shows a browser window displaying the Django admin 'Boletín' section. The URL is 127.0.0.1:8000/admin/boletin/registrado/. The page title is 'Administración de Django'. It shows a list of registered users:

Acción	EMAIL	NOMBRE	TIMESTAMP
<input type="checkbox"/>	ivanromany@hotmail.com	Ivan Roman	22 de Mayo de 2018 a las 16:58
<input type="checkbox"/>	flormaria@hotmail.com	Flor Torres	22 de Mayo de 2018 a las 16:37

Below the table, it says '2 registrados' and has a 'Grabar' button. On the right, there are filter options for timestamp: 'Por timestamp' with choices 'Cualquier fecha', 'Hoy', 'Últimos 7 días', 'Este mes', and 'Este año'.

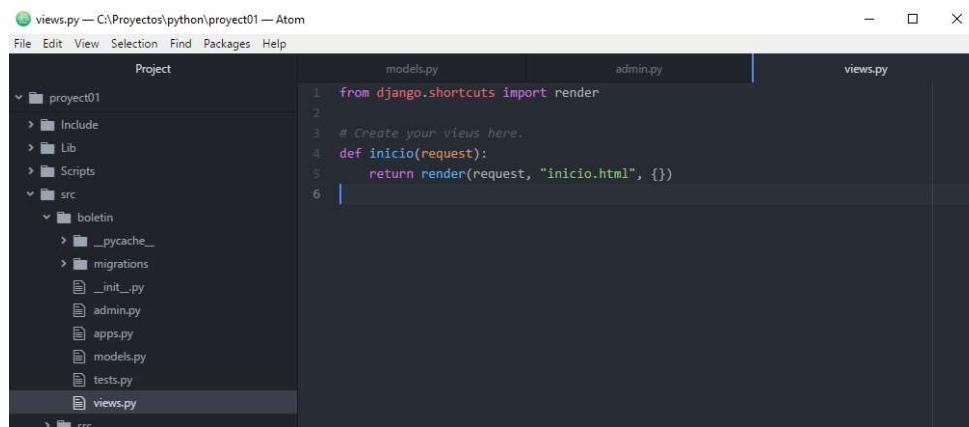
## 9. Primera Vista

- Tenemos que configurar la vista sobre la carpeta de la aplicación boletin en el archivo **views.py**, en donde añadiremos el código una función al cual pasaremos un request, le pasaremos un documento html om plantilla, para obtener un diccionario vacío.

Archivo **views.py**:

```
from django.shortcuts import render

# Create your views here.
def inicio(request):
    return render(request, "inicio.html", {})
```



The screenshot shows the Atom code editor interface. The title bar says "views.py — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The left sidebar shows a project structure under "Project": "proyecto01" (with "Include", "Lib", "Scripts", and "src" folders), and "src" (with "boletin" folder containing "\_\_pycache\_\_", "migrations", "\_\_init\_\_.py", "admin.py", "apps.py", "models.py", "tests.py", and "views.py"). The main editor area has tabs for "models.py", "admin.py", and "views.py". The "views.py" tab is active and displays the Python code for the "inicio" view function.

- Pasaremos una **url** para la vista del **proyecto01**, modificando el archivo **url.py** de la carpeta proyecto01. En donde **inicio** se refiere al template **inicio.html**, el cual al momento no está creado.

Archivo **url.py**:

```
from django.contrib import admin
from django.urls import path

from boletin import views
#from boletin.views import inicio

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio')
]
```

Por lo tanto, se mostrará el siguiente error al iniciar el servidor. <http://localhost:8000/>  
Ya que el template **inicio.html** no está creado en la url de lectura.

TemplateDoesNotExist at /

inicio.html

Request Method: GET  
Request URL: http://localhost:8000/  
Django Version: 2.0.5  
Exception Type: TemplateDoesNotExist  
Exception Value: 'inicio.html'  
Exception Location: C:\Proyectos\python\proyecto01\lib\site-packages\django\template\loader.py in get\_template, line 19  
Python Executable: C:\Proyectos\python\proyecto01\Scripts\python.exe  
Python Version: 3.6.5  
Python Path: ['C:\Proyectos\python\proyecto01\src', 'C:\Proyectos\python\proyecto01\Scripts\python36.zip', 'C:\Windows\system32\python36.dll', 'C:\Windows\python36.dll', 'C:\Proyectos\python\proyecto01\lib', 'C:\Proyectos\python\proyecto01\Scripts', 'C:\Program Files\python36\Lib', 'C:\Program Files\python36\DLLs', 'C:\Proyectos\python\proyecto01', 'C:\Proyectos\python\proyecto01\lib\site-packages']  
Server time: Mar, 22 May 2018 20:38:10 +0000

**Template-loader postmortem**

Django tried loading these templates, in this order:

Using engine django:

- django.template.loaders.app\_directories.Loader: C:\Proyectos\python\proyecto01\lib\site-packages\django\contrib\admin\templates\inicio.html (Source does not exist)
- django.template.loaders.app\_directories.Loader: C:\Proyectos\python\proyecto01\lib\site-packages\django\contrib\auth\templates\inicio.html (Source does not exist)

**Traceback** [Switch to copy-and-paste view](#)

C:\Proyectos\python\proyecto01\lib\site-packages\django\core\handlers\exception.py in inner

## 10. Configuración de Plantillas

- Tenemos que crear la carpeta denominada **templates** y dentro de ella crear el archivo **inicio.html** el mismo que estará dentro de la carpeta **proyecto01** que es la principal.

Abrimos el archivo **setting.py** y copiamos:

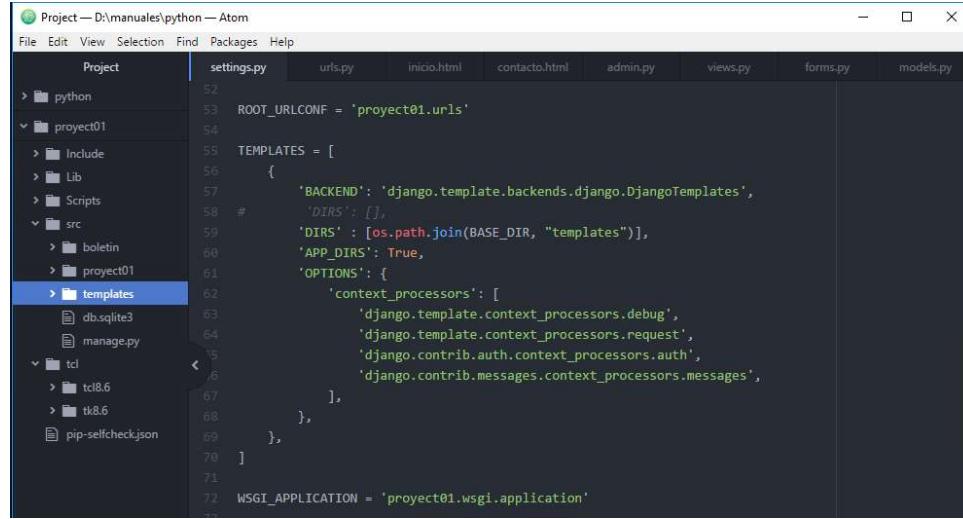
**os.path.join(BASE\_DIR, ...)**

Y lo pegamos en la ruta desde donde deberá leerse, Python utiliza este formato para encontrar las urls, osea:

```
#     'DIRS': [],
    'DIRS' : [os.path.join(BASE_DIR, "templates")],
    'APP_DIRS': True,
```

```
settings.py — D:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project settings.py urls.py inicio.html contacto.html admin.py views.py forms.py models.py
52
53 ROOT_URLCONF = 'proyecto01.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         #         'DIRS': [],
59         'DIRS' : [os.path.join(BASE_DIR, "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 = 'proyecto01.wsgi.application'
```

- Procedemos a crear la carpeta **templates**, a la altura de src.



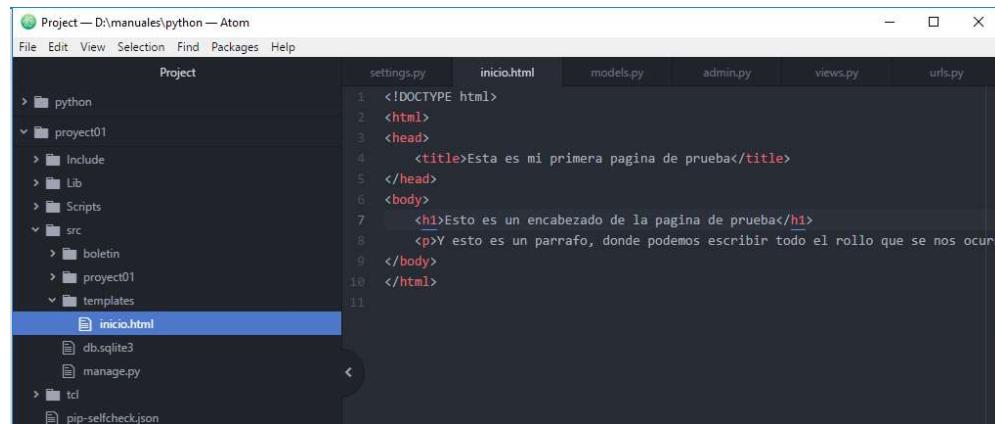
The screenshot shows the Atom IDE interface with the following details:

- Project:** D:\manuales\python
- File:** settings.py
- Content:** The code defines the project's configuration, including the root URLconf, template settings (BACKEND, DIRS, APP\_DIRS, OPTIONS), and the WSGI application.
- Structure:** The project structure is visible on the left, showing a nested folder 'python' containing 'proyect01' which has 'Include', 'Lib', 'Scripts', 'src', 'boletin', and 'proyect01'. 'proyect01' contains a 'templates' folder which in turn contains 'db.sqlite3', 'manage.py', and 'inicio.html'.

- Ahora bien, debemos crear dentro de esa carpeta el archivo **inicio.html**, con algún código representativo que indique que está funcionando, por ejemplo:

#### Archivo **inicio.html**:

```
<!DOCTYPE html>
<html>
<head>
    <title>Esta es mi primera página de prueba</title>
</head>
<body>
    <h1>Esto es un encabezado de la página de prueba</h1>
    <p>Y esto es un párrafo, donde podemos escribir todo el rollo que se nos ocurra.</p>
</body>
</html>
```



The screenshot shows the Atom IDE interface with the following details:

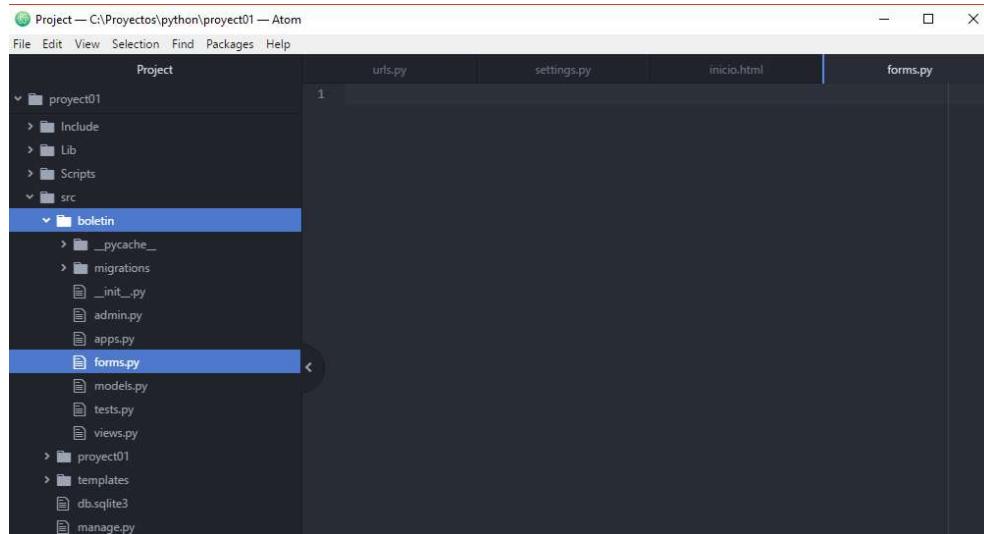
- Project:** D:\manuales\python
- File:** inicio.html
- Content:** The file contains the HTML code provided in the previous snippet.
- Structure:** The project structure is visible on the left, showing a nested folder 'python' containing 'proyect01' which has 'Include', 'Lib', 'Scripts', 'src', 'boletin', and 'proyect01'. 'proyect01' contains a 'templates' folder which in turn contains 'db.sqlite3', 'manage.py', and 'inicio.html'.

- Luego al activar el servidor con **python manage.py runserver**, e ir a la página inicial <http://localhost:8000/> podremos observar:



## 11. Escribir Formulario

- Escribimos el formulario dentro de la carpeta boletin con nombre **forms.py**, al cual le añadiremos la estructura necesaria para el ingreso de datos, dependiendo desde luego de los campos.

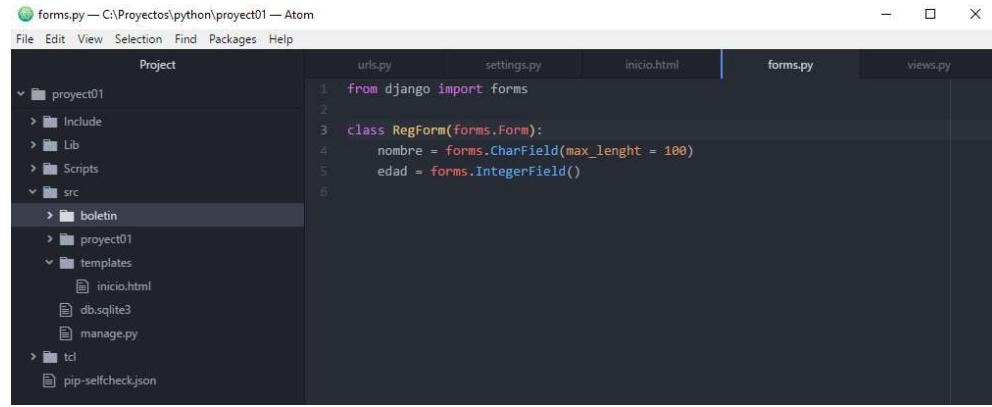


- Agregamos el código al archivo **forms.py**

Archivo **forms.py**:

```
from django import forms

class RegForm(forms.Form):
    nombre = forms.CharField(max_length = 100)
    edad = forms.IntegerField()
```



```

forms.py — C:\Proyectos\python\project01 — Atom
File Edit View Selection Find Packages Help
Project urls.py settings.py inicio.html forms.py views.py
  ✓ project01
    > Include
    > Lib
    > Scripts
    ✓ src
      > boletin
        > project01
        > templates
          inicio.html
          db.sqlite3
          manage.py
      > tcl
      pip-selfcheck.json

```

```

1 from django import forms
2
3 class RegForm(forms.Form):
4     nombre = forms.CharField(max_length = 100)
5     edad = forms.IntegerField()
6

```

## 12. Formulario en una Vista

- Procedemos a abrir la vista creada **views.py** en donde añadiremos código adicional para visualizar el formulario y renderizarlo.

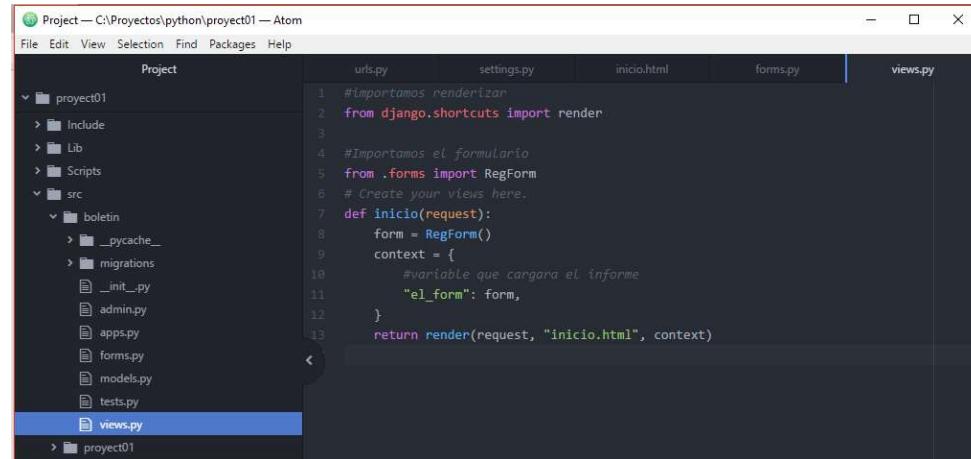
Archivo **views.py**:

```

#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
from .forms import RegForm
# Create your views here.
def inicio(request):
    form = RegForm()
    context = {
        #variable que cargara el informe
        "el_form": form,
    }
    return render(request, "inicio.html", context)

```



```

Project — C:\Proyectos\python\project01 — Atom
File Edit View Selection Find Packages Help
Project urls.py settings.py inicio.html forms.py views.py
  ✓ project01
    > Include
    > Lib
    > Scripts
    ✓ src
      > boletin
        > __pycache__
        > migrations
          __init__.py
          admin.py
          apps.py
          forms.py
          models.py
          tests.py
      > views.py
      project01

```

```

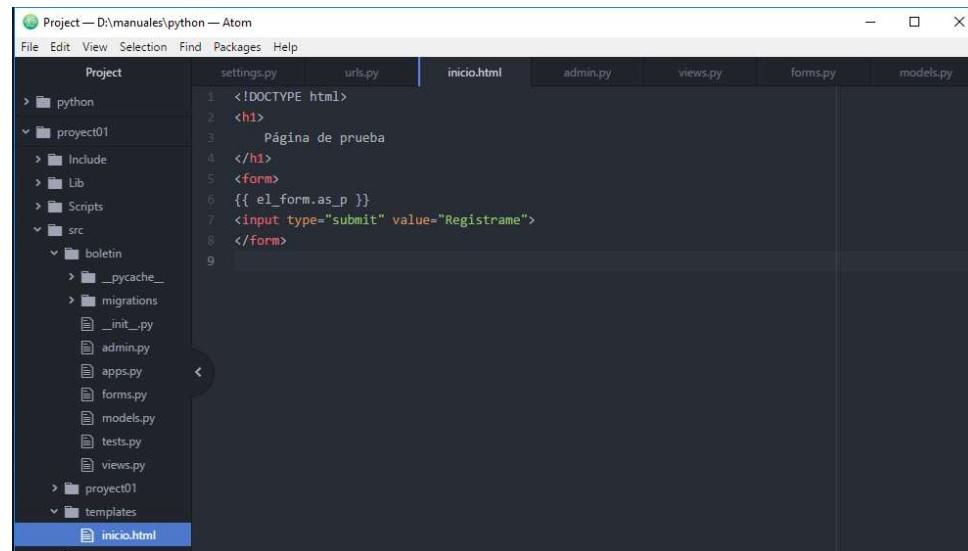
1 #importamos renderizar
2 from django.shortcuts import render
3
4 #Importamos el formulario
5 from .forms import RegForm
6 # Create your views here.
7 def inicio(request):
8     form = RegForm()
9     context = {
10         #variable que cargara el informe
11         "el_form": form,
12     }
13     return render(request, "inicio.html", context)

```

- Ahora vamos a la página de **inicio.html** en donde capturaremos agregando el formulario renderizado.

**Archivo inicio.html:**

```
<!DOCTYPE html>
<h1>
    Página de prueba
</h1>
<form>
{{ el_form.as_p }}
<input type="submit" value="Regístrate">
</form>
```



### 13. Método HTTP POST en Formulario

- Concepto de **CRUD**
  - C**      **Create (crear)** – añadir/guardar en la base de datos
  - R**      **Read or Retreive (recuperar)** – de la base de datos \*Quering (hacer consultas)
  - U**      **Update (actualizar)** – modificar un objeto guardado en la base de datos
  - D**      **Delete (borrar/eliminar)** – de una base de datos

**List (listar)** – listar los datos de la base de datos \*Queryset\*

**Search (buscar)** – hacer una búsqueda (parecido a retrieve)

**POST**, usado para cambiar el estado de la Base de Datos

- **C**      **Create (crear)** – añadir/guardar en la base de datos **POST**
- Por defecto siempre va el método **GET**, el cual permite que se visualicen los datos al registrarlos en la **url**, esto es incorrecto por motivos de seguridad.



## Página de prueba

Nombre:

Edad:

Comprobando método a **GET**:

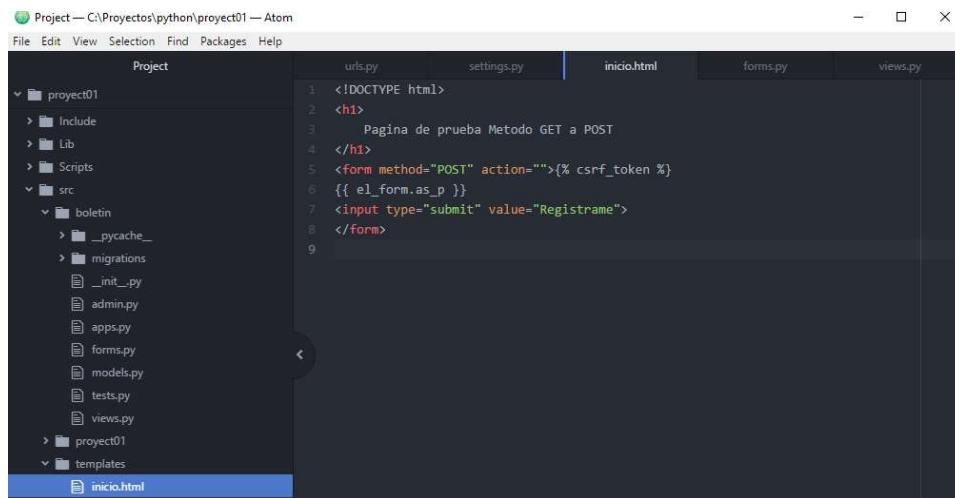
```
C:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
May 23, 2018 - 09:48:29
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[23/May/2018 09:48:30] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 97
[23/May/2018 09:48:30] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 97
[23/May/2018 09:48:31] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 97
[23/May/2018 09:48:32] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 97
[23/May/2018 09:48:33] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 97
[23/May/2018 09:48:44] "GET /?nombre=asdasd&edad=123 HTTP/1.1" 200 318
[23/May/2018 10:10:00] "GET /?nombre=Ivan&edad=51 HTTP/1.1" 200 318
```

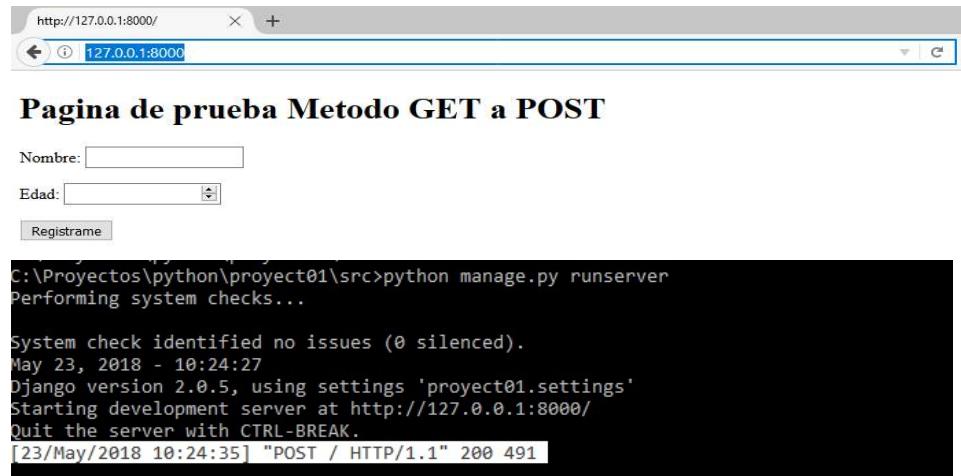
- Debemos por lo tanto cambiar al método POST en la página de **inicio.html** para agregar seguridad.

Archivo **inicio.html**:

```
<!DOCTYPE html>
<h1>
    Página de prueba Método GET a POST
</h1>
<form method="POST" action="">{% csrf_token %}
{{ el_form.as_p }}
<input type="submit" value="Registrate">
</form>
```



Comprobando cambio de método a **POST**:



#### 14. Validaciones Formulario pt.I

- Validación del formulario **request** para los campos obligatorios, datos limpios. Debemos agregar el código al archivo **views.py** de la carpeta boletin.

**TIP:** para ver los comandos o sentencias **form** que pueden ser usadas en un formulario

Archivo **views.py**:

```
#importamos renderizar
from django.shortcuts import render

#importamos el formulario
from .forms import RegForm
# Create your views here.

def inicio(request):
    form = RegForm(request.POST or None)
    print (dir(form))
    context = {
        #variable que cargara el informe
        "el_form":form,
    }
    return render(request, "inicio.html", context)
```

```
Seleccionar cmd - python manage.py runserver
Internal Server Error: /
Traceback (most recent call last):
  File "C:\Program Files\Python36\lib\site-packages\django\core\handlers\exception.py", line 35, in inner
    response = get_response(request)
  File "C:\Program Files\Python36\lib\site-packages\django\core\handlers\base.py", line 128, in _get_response
    response = self.process_exception_by_middleware(e, request)
  File "C:\Program Files\Python36\lib\site-packages\django\core\handlers\base.py", line 126, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
  File "C:\Proyectos\python\proyecto01\src\boletin\views.py", line 8, in inicio
    form = RegForm(request.POST or None)
NameError: name 'RegForm' is not defined
[23/May/2018 10:35:31] "POST / HTTP/1.1" 500 65464
Performing system checks...
System check identified no issues (0 silenced).
May 23, 2018 - 10:36:10
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__html__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_bound_fields_cache', '_clean_fields', '_clean_form', '_errors', '_html_output', '_post_clean', '_add_error', '_add_initial_prefix', '_add_prefix', '_as_p', '_as_table', '_auto_id', '_base_fields', '_changed_data', '_clean', '_data', '_declared_fields', '_default_renderer', '_empty_permitted', '_error_class', '_errors', '_field_order', '_fields', '_files', '_full_clean', '_get_initial_for_field', '_has_changed', '_has_error', '_hidden_fields', '_initial', '_is_bound', '_is_multipart', '_is_valid', '_label_suffix', '_media', '_non_field_errors', '_order_fields', '_prefix', '_renderer', '_use_required_attribute', '_visible_fields']
[23/May/2018 10:36:15] "POST / HTTP/1.1" 200 515
```

- Ahora bien, podemos también comprobar los datos limpios modificando el código con el comando **print**, pero desde luego solo es para comprobación, después habrá que comentarlo.

Archivo **views.py**:

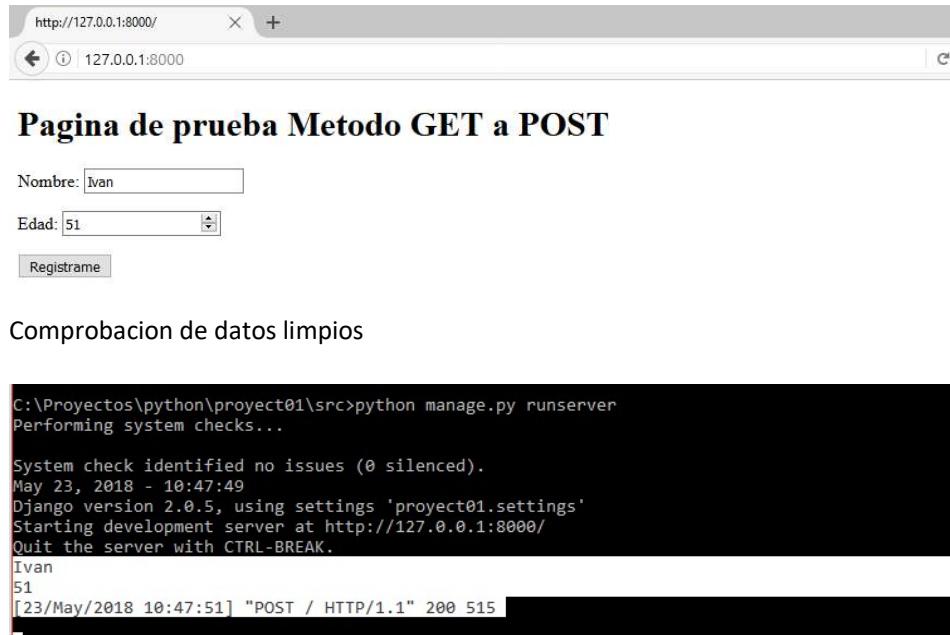
```
#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
from .forms import RegForm
# Create your views here.

def inicio(request):
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
        print (form_data.get("nombre"))
        print (form_data.get("edad"))

    context = {
        #variable que cargara el informe
        "el_form": form,
    }
    return render(request, "inicio.html", context)
```

Project	urls.py	settings.py	inicio.html	forms.py	views.py
proyecto01					
Include					
Lib					
Scripts					
src					
boletin					
__pycache__					
migrations					
__init__.py					
admin.py					
apps.py					
forms.py					
models.py					
tests.py					
views.py	1 #importamos renderizar 2 from django.shortcuts import render 3 4 #Importamos el formulario 5 from .forms import RegForm 6 # Create your views here. 7 def inicio(request): 8     form = RegForm(request.POST or None) 9     if form.is_valid(): 10         form_data = form.cleaned_data 11         print (form_data.get("nombre")) 12         print (form_data.get("edad")) 13     context = { 14         #variable que cargara el informe 15         "el_form": form, 16     } 17     return render(request, "inicio.html", context)				



Página de prueba Método GET a POST

Nombre:

Edad:

Comprobacion de datos limpios

```
C:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
May 23, 2018 - 10:47:49
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Ivan
51
[23/May/2018 10:47:51] "POST / HTTP/1.1" 200 515
```

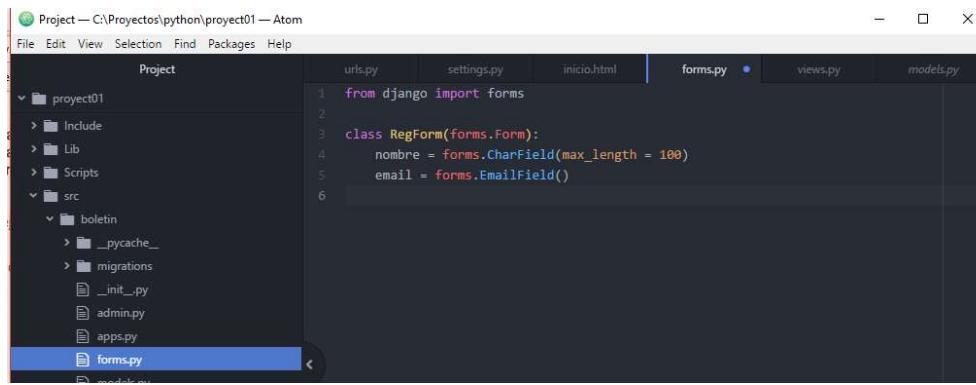
## 15. Guardar Datos del Formulario con el Modelo

- Como guardar objetos nuevos usando el modelo **Registrado**, importando el modelo y agregando variables a la vista **views.py**, de la carpeta boletín
- Primero cambiamos el **forms.py** para que coincida con nuestro modelo existente en la base de datos llamado Registrado.

Archivo **forms.py**:

```
from django import forms

class RegForm(forms.Form):
    nombre = forms.CharField(max_length = 100)
    email = forms.EmailField()
```



Project — C:\Proyectos\python\proyecto01 — Atom

File Edit View Selection Find Packages Help

Project

- project01
  - Include
  - Lib
  - Scripts
  - src
    - boletin
      - \_\_pycache\_\_
      - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - forms.py
      - models.py

urls.py    settings.py    inicio.html    forms.py    views.py    models.py

```
1  from django import forms
2
3  class RegForm(forms.Form):
4      nombre = forms.CharField(max_length = 100)
5      email = forms.EmailField()
```

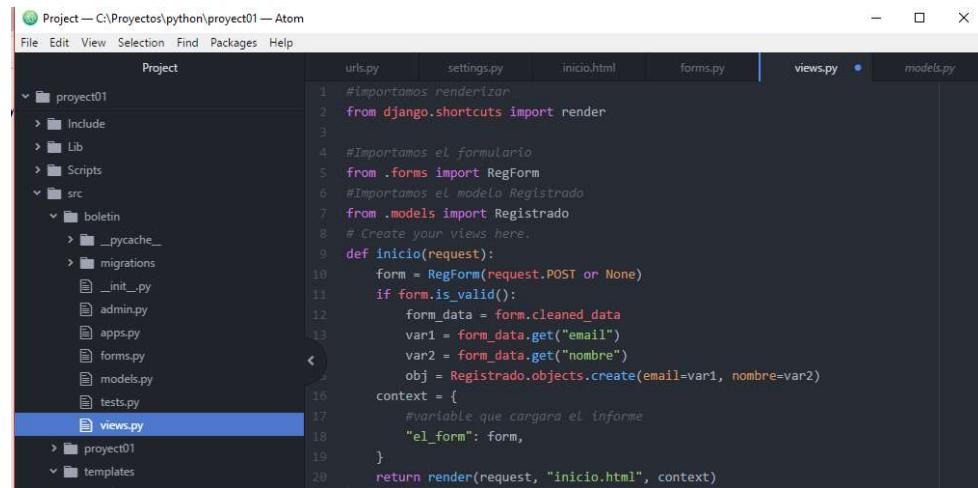
- Ahora modificamos **views.py** en donde agregamos:

**Archivo views.py:**

```
#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
from .forms import RegForm
#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.

def inicio(request):
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
        var1 = form_data.get("email")
        var2 = form_data.get("nombre")
        obj = Registrado.objects.create(email=var1, nombre=var2)
    context = {
        #variable que cargara el informe
        "el_form": form,
    }
    return render(request, "inicio.html", context)
```



The screenshot shows the Atom code editor interface. The title bar says "Project — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The top navigation bar has tabs for urls.py, settings.py, inicio.html, forms.py, views.py (which is selected), and models.py. The left sidebar shows the project structure: a root folder "proyecto01" containing "Include", "Lib", "Scripts", and a "src" folder. "src" contains "boletin" (with \_\_pycache\_\_ and migrations), "admin.py", "apps.py", "forms.py", "models.py", and "tests.py". "views.py" is highlighted in blue. The main editor area displays the Python code for "views.py" as shown in the previous code block.

- Ejecutando el formulario comprobaremos el ingreso en la base de datos con el admin de Django.

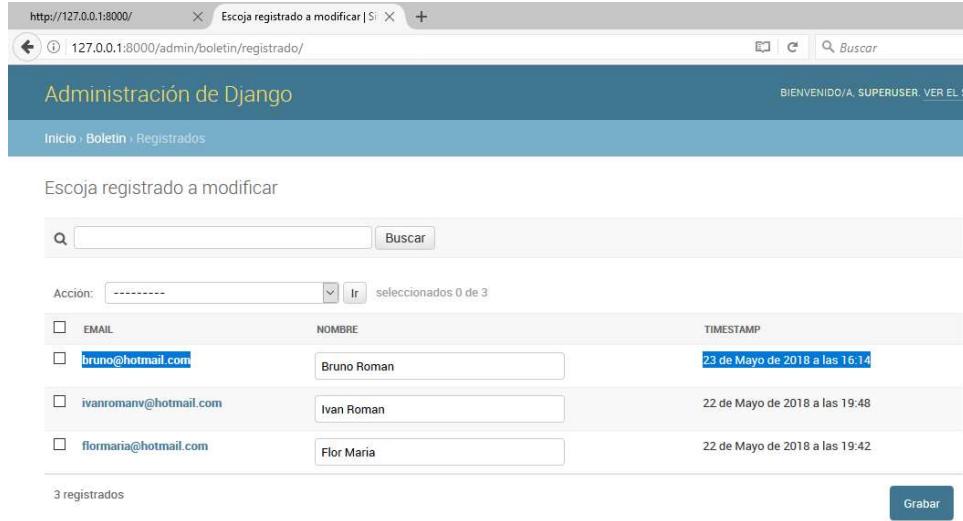


### Página de prueba Método GET a POST

Nombre:

• Este campo es obligatorio.

Email:



<input type="checkbox"/> EMAIL	NOMBRE	TIMESTAMP
<input type="checkbox"/> bruno@hotmail.com	Bruno Roman	23 de Mayo de 2018 a las 16:14
<input type="checkbox"/> ivanromanv@hotmail.com	Ivan Roman	22 de Mayo de 2018 a las 19:48
<input type="checkbox"/> flormaria@hotmail.com	Flor Maria	22 de Mayo de 2018 a las 19:42

3 registrados Grabar

## (PERSONALIZAR INTERFAZ ADMINISTRATIVA)

### 16. Model Form

- Para el aprovechamiento de los campos de nuestro modelo de forma más eficiente para personalizarlo, **dentro de la interfaz administrativa de Django**.
- Modificamos el **forms.py**, importando el modelo **Registrado** y creamos la clase **RegModelForm** para el modelo.

Archivo **forms.py**:

```
from django import forms
#Importamos el modelo Registrado
from .models import Registrado

#Agregamos clase RegModelForm
class RegModelForm(forms.ModelForm):
    class Meta:
        model = Registrado
        #Los campos que se deseen mostar
        fields = ["nombre", "email"]

    class RegForm(forms.Form):
        nombre = forms.CharField(max_length = 100)
        email = forms.EmailField()
```

The screenshot shows the Atom code editor interface. On the left, the project structure is displayed under 'Project' with the following files and folders:

- Project
  - proyecto01
    - Include
    - Lib
    - Scripts
    - src
      - boletin
        - \_\_pycache\_\_
        - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - forms.py**

The right pane contains the code for `forms.py`:

```

1 from django import forms
2 #Importamos el modelo Registrado
3 from .models import Registrado
4 #Agregamos clase RegistroForm
5 class RegModelForm(forms.ModelForm):
6     class Meta:
7         model = Registrado
8         #Los campos que se deseen mostrar
9         fields = ["nombre", "email"]
10
11     class RegForm(forms.Form):
12         nombre = forms.CharField(max_length = 100)
13         email = forms.EmailField()

```

- En el archivo **admin.py** hay que importar **RegModelForm** declarar el formulario con su mismo nombre.

#### Archivo **admin.py**:

```

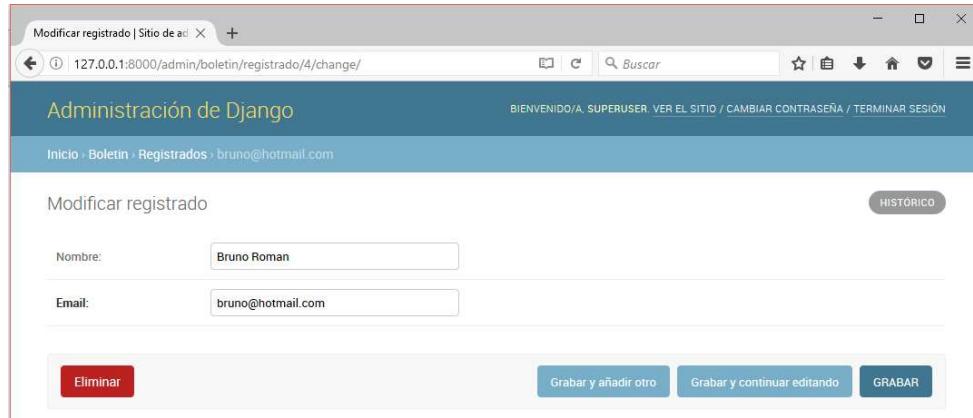
from django.contrib import admin

# Register your models here.
from .models import Registrado
# Importar modelo RegModelForm
from .forms import RegModelForm

class AdminRegistrado(admin.ModelAdmin):
    # Los registros que seran mostrados
    list_display = ["email", "nombre", "timestamp"]
    form = RegModelForm
    # Aparece link sobre el nombre para acceder al registro
    # NOTA: es importante que el campo contenga datos para que el link funcione
    #list_display_links = ["nombre"]
    # Filtros de registros
    list_filter = ["timestamp"]
    # Los registros que pueden ser modificados
    list_editable = ["nombre"]
    # Busqueda de campos
    search_fields = ["email", "nombre"]
    #class Meta:
    #    model = Registrado

admin.site.register(Registrado, AdminRegistrado)

```



## 17. Validaciones Model Form

- Por ejemplo, si requerimos validar nuestro correo electrónico que se ingrese al Model Form, para solo permitir direcciones “.edu” y no direcciones particular, se procedería a modificar el código en **forms.py**, se utiliza **self** para referirse a la función por si misma.

Archivo **forms.py**:

```
from django import forms
#Importamos el modelo Registrado
from .models import Registrado
#Agregamos clase RegModelForm
class RegModelForm(forms.ModelForm):
    class Meta:
        model = Registrado
        #Los campos que se deseen mostrar
        fields = ["nombre", "email"]

    def clean_email(self):
        email = self.cleaned_data.get("email")
        email_base, proveedor = email.split("@")
        dominio, extension = proveedor.split(".")
        if not extension == "edu":
            raise forms.ValidationError("Por favor ingrese un correo con extension .EDU")
        return email

class RegForm(forms.Form):
    nombre = forms.CharField(max_length = 100)
    email = forms.EmailField()
```

Project — C:\Proyectos\python\proyecto01 — Atom

File Edit View Selection Find Packages Help

Project	urls.py	settings.py	inicio.html	forms.py	admin.py	views.py
project01						
> Include						
> Lib						
> Scripts						
src						
> boletin						
> __pycache__						
> migrations						
__init__.py						
admin.py						
apps.py						
<b>forms.py</b>						
models.py						
tests.py						
views.py						
> proyecto01						
templates						
inicio.html						
db.sqlite3						

```

1  from django import forms
2  #Importamos el modelo Registrado
3  from .models import Registrado
4  #Agregamos clase RegModelForm
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          #Los campos que se deseen mostrar
9          fields = ["nombre", "email"]
10
11     def clean_email(self):
12         email = self.cleaned_data.get("email")
13         email_base, proveedor = email.split("@")
14         dominio, extension = proveedor.split(".")
15         if not extension == "edu":
16             raise forms.ValidationError("Por favor ingrese un correo con extension .EDU")
17         return email
18
19     class RegForm(forms.Form):
20         nombre = forms.CharField(max_length = 100)
21         email = forms.EmailField()

```

Modificar registrado | Sitio de ad +

127.0.0.1:8000/admin/boletin/Registrado/4/change/

Buscar

Administración de Django

BIENVENIDO/A. SUPERUSER. VER EL SITIO / CAMBIAR CONTRASEÑA / TERMINAR SESIÓN

Inicio • Boletín • Registrados • bruno@hotmail.com

Modificar registrado

Por favor, corrija los siguientes errores.

Nombre: Bruno Roman

Email: Por favor ingrese un correo con extension .EDU  
bruno@hotmail.com

Eliminar Grabar y añadir otro Grabar y continuar editando GRABAR

## 18. Contexto en la vista, plantilla

- Envía datos o variables hacia la pantalla de la vista, para ello se debe modificar el archivo **views.py** e **inicio.html**, entonces modificando **view.py**

Archivo **views.py**:

```

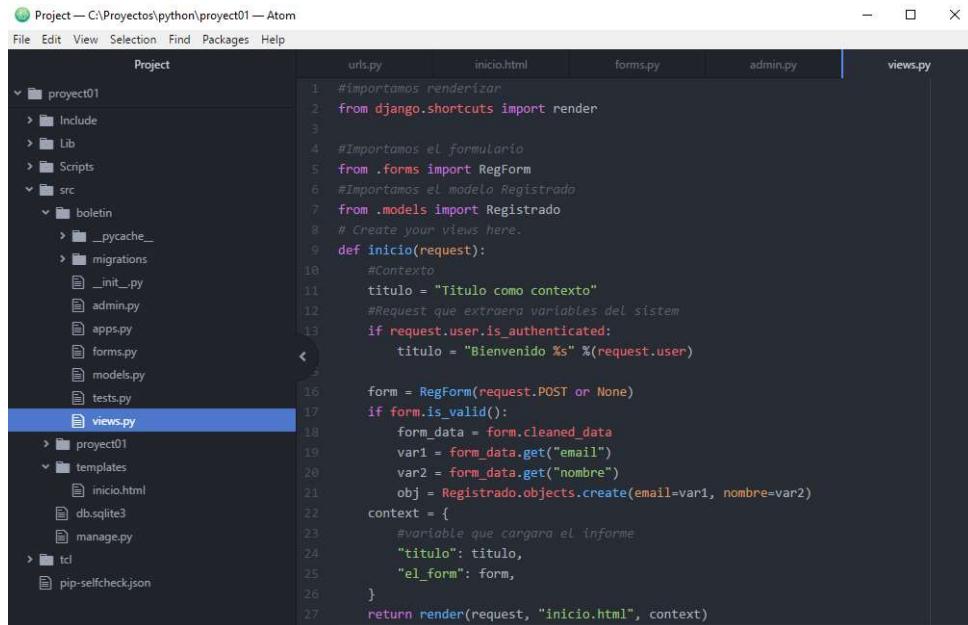
#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
from .forms import RegForm
#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Titulo como contexto"
    #Request que extraera variables del sistema
    if request.user.is_authenticated:

```

```
titulo = "Bienvenido %s" %(request.user)
```

```
form = RegForm(request.POST or None)
if form.is_valid():
    form_data = form.cleaned_data
    var1 = form_data.get("email")
    var2 = form_data.get("nombre")
    obj = Registrado.objects.create(email=var1, nombre=var2)
context = {
    #variable que cargara el informe
    "titulo": titulo,
    "el_form": form,
}
return render(request, "inicio.html", context)
```



The screenshot shows the Atom code editor interface. The title bar says "Project — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The left sidebar shows a project structure with a "views.py" file selected. The main editor area displays the following Python code:

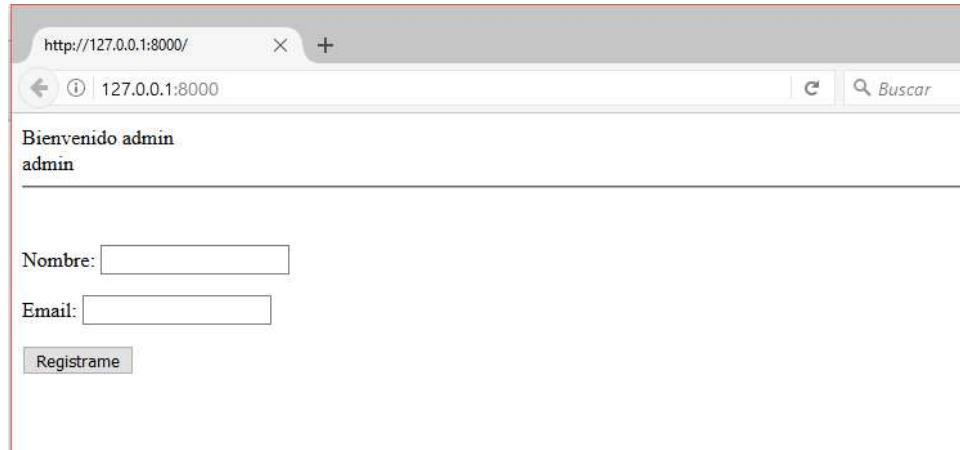
```
1 #importamos renderizar
2 from django.shortcuts import render
3
4 #Importamos el formulario
5 from .forms import RegForm
6 #Importamos el modelo Registrado
7 from .models import Registrado
8 # Create your views here.
9 def inicio(request):
10     #Contexto
11     titulo = "Titulo como contexto"
12     #Request que extraera variables del sistema
13     if request.user.is_authenticated:
14         titulo = "Bienvenido %s" %(request.user)
15
16     form = RegForm(request.POST or None)
17     if form.is_valid():
18         form_data = form.cleaned_data
19         var1 = form_data.get("email")
20         var2 = form_data.get("nombre")
21         obj = Registrado.objects.create(email=var1, nombre=var2)
22     context = {
23         #variable que cargara el informe
24         "titulo": titulo,
25         "el_form": form,
26     }
27
28 return render(request, "inicio.html", context)
```

#### Archivo inicio.html:

```
<!DOCTYPE html>
{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<form method="POST" action="">{% csrf_token %}
{{ el_form.as_p }}
<input type="submit" value="Registrate">
</form>
```

En donde vemos como se pasa a la vista un texto o una variable del sistema.



Bienvenido admin  
admin

---

Nombre:

Email:

## 19. ModelForm en la vista

- Sustituir el **RegForm** en la vista por **RegModelForm**, vamos a modificar **views.py**

Archivo **views.py**:

```
#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
from .forms import RegForm, RegModelForm
#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Titulo como contexto"
    #Request que extraera variables del sistem
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }

    if form.is_valid():
        instance = form.save(commit=False)
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        # En caso de la descripcion de persona no se llene por formulario
        if not instance.nombre:
            instance.nombre = "Persona nueva"
```

```

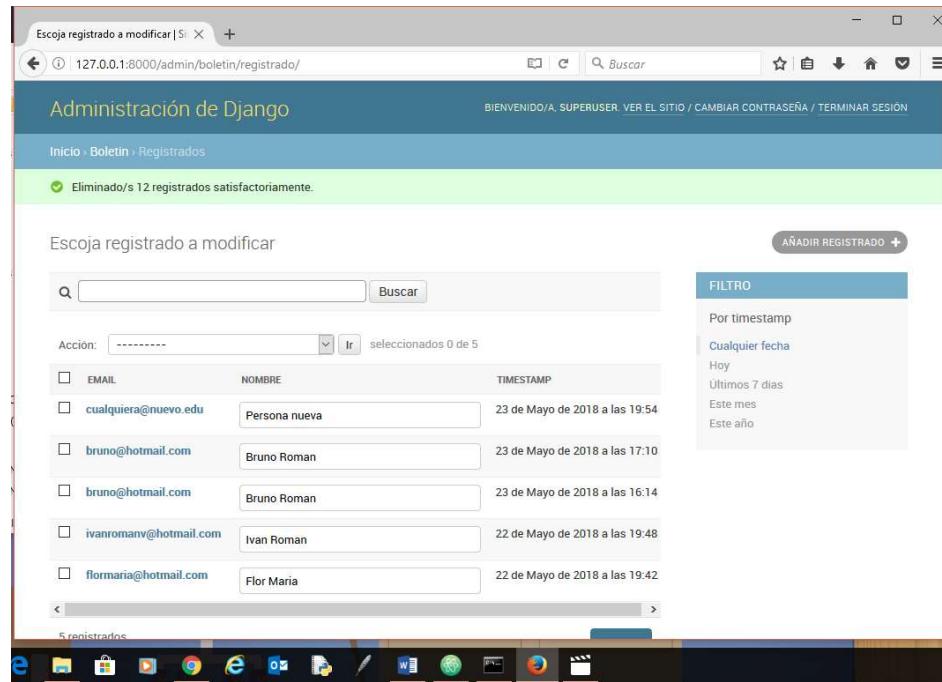
#Grabar instancia
instance.save()
#Envio de nuevo contexto con nombre
context = {
    "titulo": "Gracias %s!" %(nombre)
}
#Envio de nuevo contexto con email, el nombre de persona esta vacio
if not nombre:
    context = {
        "titulo": "Gracias %s" %(email)
    }

#Para mostrarse al cargar el browser
#print(instance)
#print(instance.timestamp)

# form_data = form.cleaned_data
# var1 = form_data.get("email")
# var2 = form_data.get("nombre")
# obj = Registrado.objects.create(email=var1, nombre=var2)

return render(request, "inicio.html", context)

```



Ahora el archive **inicio.html** si queremos que se apague el botón Registrarme

#### Archivo **inicio.html**:

```

<!DOCTYPE html>
{{ titulo }}<br/>

```

```

{{ request.user }}
<hr/>
<br/>
{% if form %}
<form method="POST" action="">{{ csrf_token }}
{{ el_form.as_p }}
<input type="submit" value="Registrate">
</form>
{% endif %}

```

## 20. Custom Form para Contacto

- Para conversión **RegForm** como formulario de **Contacto** para usarlo para el fin, usando la misma forma.

Modificamos el archivo **forms.py** y agregamos la forma de contacto **ContactForm**.

Archivo **forms.py**:

```

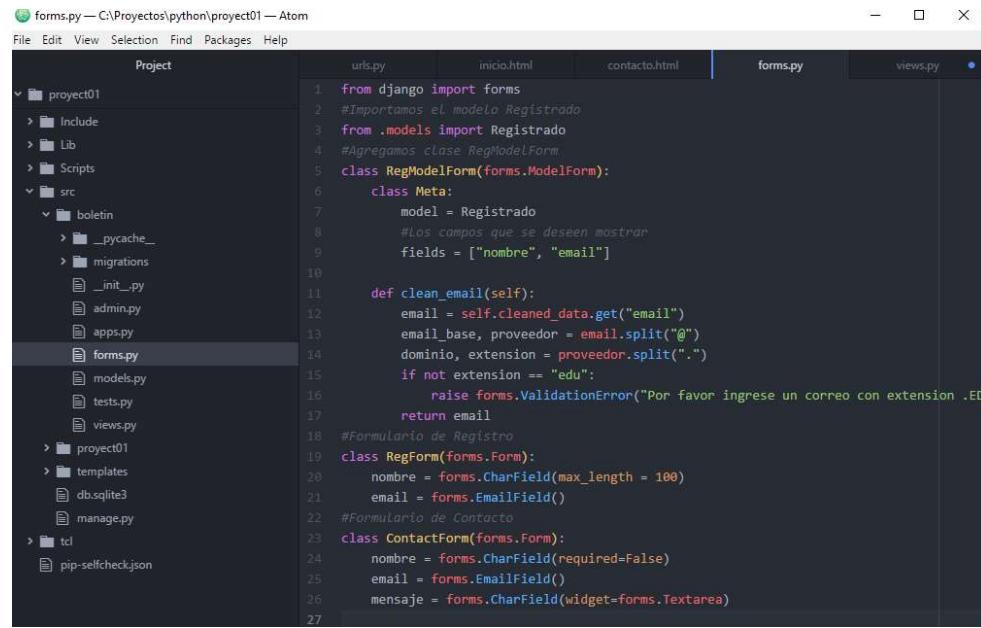
from django import forms
#Importamos el modelo Registrado
from .models import Registrado
#Agregamos clase RegModelForm
class RegModelForm(forms.ModelForm):
    class Meta:
        model = Registrado
        #Los campos que se deseen mostrar
        fields = ["nombre", "email"]

    def clean_email(self):
        email = self.cleaned_data.get("email")
        email_base, proveedor = email.split("@")
        dominio, extension = proveedor.split(".")
        if not extension == "edu":
            raise forms.ValidationError("Por favor ingrese un correo con extension .EDU")
        return email

    #Formulario de Registro
    class RegForm(forms.Form):
        nombre = forms.CharField(max_length = 100)
        email = forms.EmailField()

    #Formulario de Contacto
    class ContactForm(forms.Form):
        nombre = forms.CharField(required=False)
        email = forms.EmailField()
        mensaje = forms.CharField(widget=forms.Textarea)

```



The screenshot shows the Atom code editor with the following file structure:

```

Project
  - proyecto01
    - Include
    - Lib
    - Scripts
  - src
    - boletin
      - __init__.py
      - migrations
      - admin.py
      - apps.py
      - forms.py
      - models.py
      - tests.py
      - views.py
    - proyecto01
    - templates
    - db.sqlite3
    - manage.py
  - tcl
  - pip-selfcheck.json

```

The `forms.py` file contains the following code:

```

1  from django import forms
2  #Importamos el modelo Registrado
3  from .models import Registrado
4  #Agregamos clase RegModelForm
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          #los campos que se deseen mostrar
9          fields = ["nombre", "email"]
10
11     def clean_email(self):
12         email = self.cleaned_data.get("email")
13         email_base, proveedor = email.split("@")
14         dominio, extension = proveedor.split(".")
15         if not extension == "edu":
16             raise forms.ValidationError("Por favor ingrese un correo con extension .EDU")
17         return email
18
19     #Formulario de Registro
20     class RegForm(forms.Form):
21         nombre = forms.CharField(max_length = 100)
22         email = forms.EmailField()
23
24     #Formulario de Contacto
25     class ContactForm(forms.Form):
26         nombre = forms.CharField(required=False)
27         email = forms.EmailField()
28         mensaje = forms.CharField(widget=forms.Textarea)

```

En la vista **views.py** importamos el formulario **ContactForm** y agregamos código de la clase.

#### Archivo **views.py**:

```
#Importamos el formulario
from .forms import RegForm, RegModelForm, ContactForm
```

Agregamos al final la function **contacto** que carga la forma **ContactForm**:

```
def contacto(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        #Forma 3 de llenar
        for key, value in form.cleaned_data.items():
            print(key, value)
        #Forma 2 de llenar
        #for key in form.cleaned_data:
        #    print(key)
        #    print(form.cleaned_data.get(key))
        #Forma 1 de llenar
        #nombre = form.cleaned_data.get("nombre")
        #email = form.cleaned_data.get("email")
        #mensaje = form.cleaned_data.get("mensaje")
        #print(nombre, email, mensaje)

    context = {
        "contacto": form,
    }
    return render(request, "contacto.html", context)
```

```

views.py — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py inicio.html contacto.html forms.py views.py
src
  proyecto01
    > include
    > Lib
    > Scripts
    > src
      > boletin
        > __pycache__
        > migrations
        __init__.py
        admin.py
        apps.py
        forms.py
        models.py
        tests.py
        views.py
      > proyecto01
      > templates
      db.sqlite3
      manage.py
    > tcl
    pip-selfcheck.json

43     }
44     #     form_data = form.cleaned_data
45     #     var1 = form_data.get("email")
46     #     var2 = form_data.get("nombre")
47     #     obj = Registrado.objects.create(email=var1, nombre=var2)
48
49     return render(request, "inicio.html", context)
50
51 def contacto(request):
52     form = ContactForm(request.POST or None)
53     if form.is_valid():
54         #Forma 3 de Llenar
55         for key, value in form.cleaned_data.items():
56             print(key, value)
57         #Forma 2 de Llenar
58         #for key in form.cleaned_data:
59         #    print(key)
60         #    print(form.cleaned_data.get(key))
61         #Forma 1 de Llenar
62         #nombre = form.cleaned_data.get("nombre")
63         #email = form.cleaned_data.get("email")
64         #mensaje = form.cleaned_data.get("mensaje")
65         #print(nombre, email, mensaje)
66
67         context = {
68             "contacto": form,
69         }
70
71     return render(request, "contacto.html", context) Activar Windows
Vea la Configuración para activar Wind
CRLF UTF-8 Python 0 files
src\boletin\views.py 71:1

```

Ahora agregamos las rutas url de los archivos en urls.py

#### Archivo urls.py:

```

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
]

```

```

urls.py — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py inicio.html contacto.html forms.py views.py
src
  proyecto01
    > include
    > Lib
    > Scripts
    > src
      > boletin
        > __pycache__
        > migrations
        __init__.py
        admin.py
        apps.py
        forms.py
        models.py
        tests.py
        views.py
      > proyecto01
      > templates
      db.sqlite3
      manage.py
    > tcl
    pip-selfcheck.json

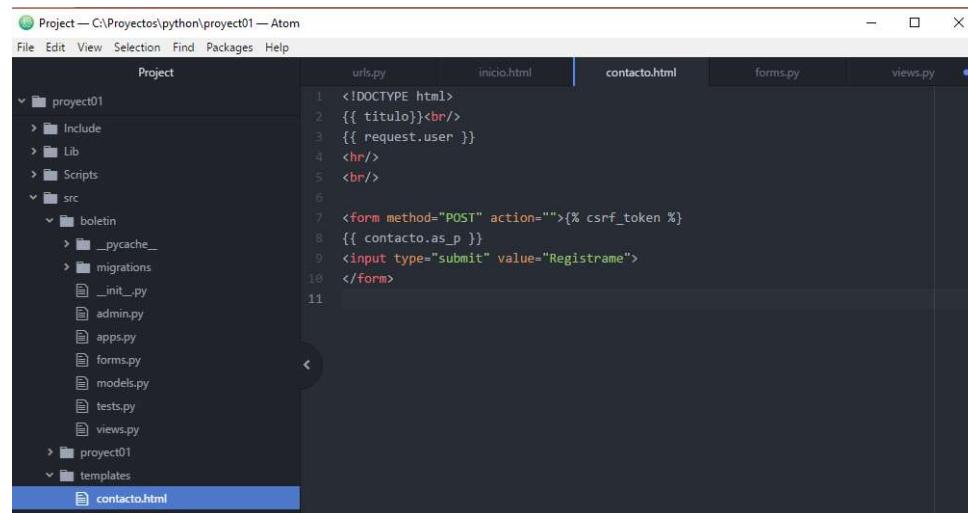
1 """proyecto01 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/2.0/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, 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 boletin import views
20 #from boletin.views inicio
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('inicio', views.inicio, name='inicio'),
25     path('contacto', views.contacto, name='contacto'),
26 ]

```

Creamos el formulario de Contacto en la carpeta **templates**, llamado **contacto.html**

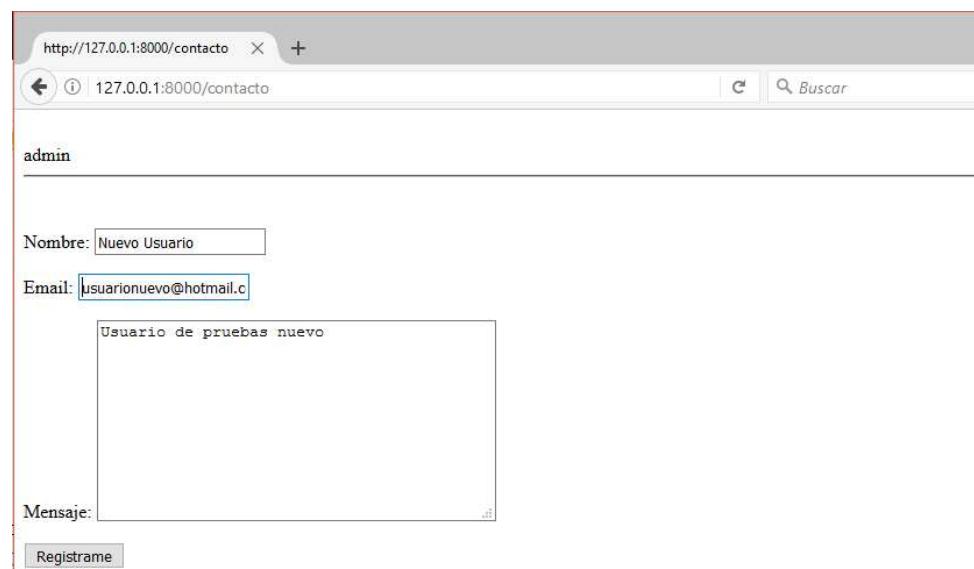
Archivo **contacto.html**:

```
<!DOCTYPE html>
{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>
<form method="POST" action="">{{ csrf_token %}}
{{ contacto.as_p }}
<input type="submit" value="Registrate">
</form>
```



- Accediendo a la url para probar formulario de contacto: <http://127.0.0.1:8000/contacto>.

**Nota:** Estos datos no se grabarán en ninguna base de datos, por ahora!



The screenshot shows a web browser window with the URL "http://127.0.0.1:8000/contacto". The page displays a form with the following fields:

- Nombre: Nuevo Usuario
- Email: usuariounuevo@hotmail.c
- Mensaje: Usuario de pruebas nuevo

At the bottom of the form is a "Registrate" button.

```
[23/May/2018 15:55:33] "POST /contacto HTTP/1.1" 200 613
[23/May/2018 15:56:03] "GET / HTTP/1.1" 200 492
[23/May/2018 15:56:11] "GET /contacto HTTP/1.1" 200 575
nombre Nuevo Usuario
email usuarionuevo@hotmail.com
mensaje Usuario de pruebas nuevo
[23/May/2018 15:58:04] "POST /contacto HTTP/1.1" 200 654
```

## 21. Configurar Email

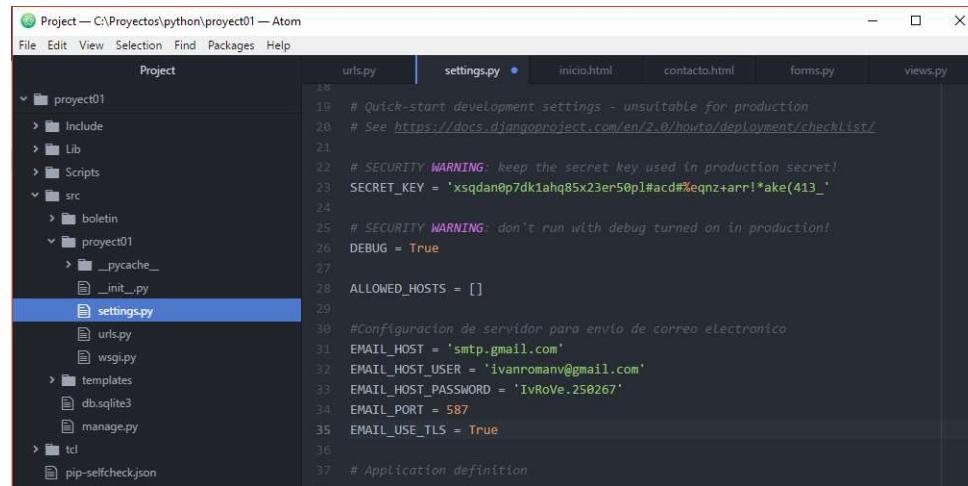
- Para esto se procede a la modificación del archivo **settings.py** de la carpeta de **proyect01**, agregando las siguientes líneas que corresponden al servidor smtp de gmail.

### Archivo **settings.py**:

```
ALLOWED_HOSTS = []
...
#Configuración de servidor para envío de correo electrónico
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'elusuario@gmail.com'
EMAIL_HOST_PASSWORD = 'sucontraseña'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

### IMPORTANTE:

```
#...
# Para usar gmail hay de desbloquear captcha
# https://accounts.google.com/displayunlockcaptcha
#...
```



The screenshot shows the Atom code editor interface. The top menu bar includes File, Edit, View, Selection, Find, Packages, and Help. Below the menu is a tab bar with urls.py, settings.py (which is currently selected), inicio.html, contacto.html, forms.py, and views.py. The main workspace displays the contents of the settings.py file. The file structure is as follows:

```

Project — C:\Proyectos\python\proyect01 — Atom
File Edit View Selection Find Packages Help
Project urls.py settings.py inicio.html contacto.html forms.py views.py
  ↓
  project01
    ↓ include
    ↓ lib
    ↓ scripts
      ↓ src
        ↓ boletin
        ↓ project01
          ↓ __pycache__
            ↓ __init__.py
        ↓ settings.py
          ↓ urls.py
          ↓ wsgi.py
        ↓ templates
        ↓ db.sqlite3
        ↓ manage.py
      ↓ tcl
      ↓ pip-selfcheck.json

```

The settings.py file content is:

```

18 # Quick-start development settings - unsuitable for production
19 # See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/
20
21 # SECURITY WARNING: keep the secret key used in production secret!
22 SECRET_KEY = 'xsqdan0p7dk1ahq85x23er50pl#acd#%eqnz+arr!*ake(413_'
23
24 # SECURITY WARNING: don't run with debug turned on in production!
25 DEBUG = True
26
27 ALLOWED_HOSTS = []
28
29 #Configuración de servidor para envío de correo electrónico
30 EMAIL_HOST = 'smtp.gmail.com'
31 EMAIL_HOST_USER = 'ivanromanv@gmail.com'
32 EMAIL_HOST_PASSWORD = 'IvRoVe.250267'
33 EMAIL_PORT = 587
34 EMAIL_USE_TLS = True
35
36 # Application definition
37
```

- Ahora en **views.py** procedemos a la importación de **send\_mail** y **settings** configurados, para luego configurar en la función **contacto** el mensaje a enviar.

### Archivo **views.py**:

```
#Importación de configuración para correo electrónico
from django.conf import settings
```

```
from django.core.mail import send_mail
#Importamos renderizar
from django.shortcuts import render

#Importamos el formulario
#from .forms import RegForm, RegModelForm
from .forms import RegForm, RegModelForm, ContactForm

#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Titulo como contexto"
    #Request que extraera variables del sistem
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }
    if form.is_valid():
        instance = form.save(commit=False)
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        # En caso de la descripcion de persona no se llene por formulario
        if not instance.nombre:
            instance.nombre = "Persona nueva"
        #Grabar instancia
        instance.save()
        #Envio de nuevo contexto con nombre
        context = {
            "titulo": "Gracias %s!" %(nombre)
        }
        #Envio de nuevo contexto con email, el nombre de persona esta vacio
        if not nombre:
            context = {
                "titulo": "Gracias %s" %(email)
            }
        # form_data = form.cleaned_data
        # var1 = form_data.get("email")
        # var2 = form_data.get("nombre")
        # obj = Registrado.objects.create(email=var1, nombre=var2)
```

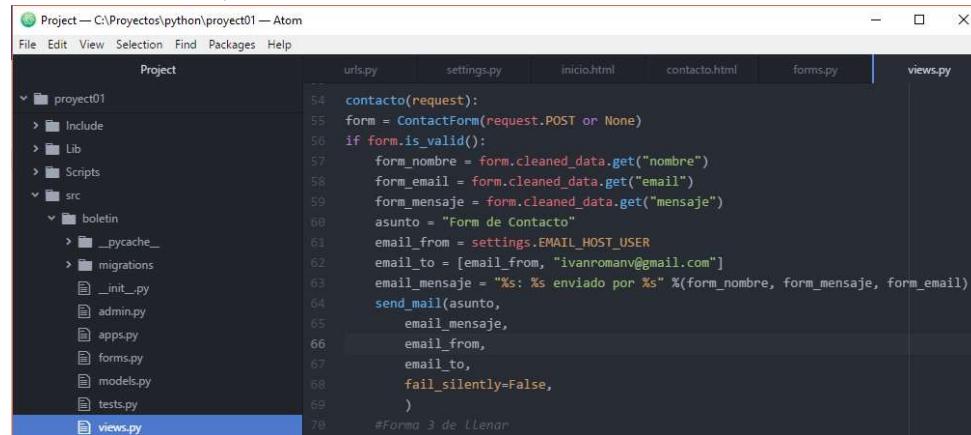
```

return render(request, "inicio.html", context)

def contacto(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        form_email = form.cleaned_data.get("email")
        form_nombre = form.cleaned_data.get("nombre")
        form_mensaje = form.cleaned_data.get("mensaje")
        asunto = "Form de Contaco"
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "elusuario@gmail.com"]
        email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_email,
form_mensaje)
        send_mail(asunto,
                  email_mensaje,
                  email_from,
                  email_to,
                  fail_silently=False,
                  )
    #Forma 3 de llenar
    #for key, value in form.cleaned_data.items():
    #    print(key, value)
    #Forma 2 de llenar
    #for key in form.cleaned_data:
    #    print(key)
    #    print(form.cleaned_data.get(key))
    #Forma 1 de llenar
    #nombre = form.cleaned_data.get("nombre")
    #email = form.cleaned_data.get("email")
    #mensaje = form.cleaned_data.get("mensaje")
    #print(nombre, email, mensaje)

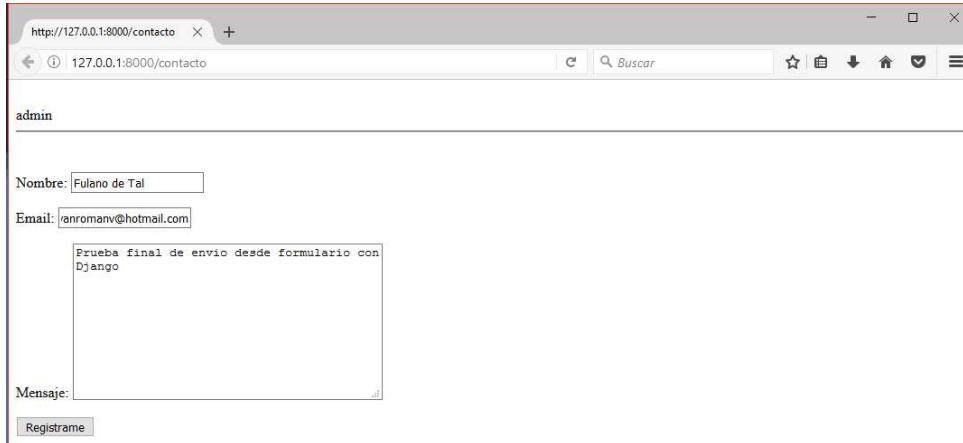
    context = {
        "contacto": form,
    }
    return render(request, "contacto.html", context)

```



The screenshot shows the Atom code editor interface. The title bar reads "Project — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. Below the menu is a toolbar with icons for Project, urls.py, settings.py, inicio.html, contacto.html, forms.py, and views.py. The main workspace shows a file tree on the left with a project structure: project01 (containing Include, Lib, Scripts, src, boletin, \_\_pycache\_\_, migrations, \_\_init\_\_.py, admin.py, apps.py, forms.py, models.py, tests.py, views.py). The right side of the screen displays the Python code for the 'views.py' file, which corresponds to the code block above.

- Ahora probaremos la funcionalidad implementada ejecutando el formulario de contacto.html en la ruta <http://127.0.0.1:8000/contacto>

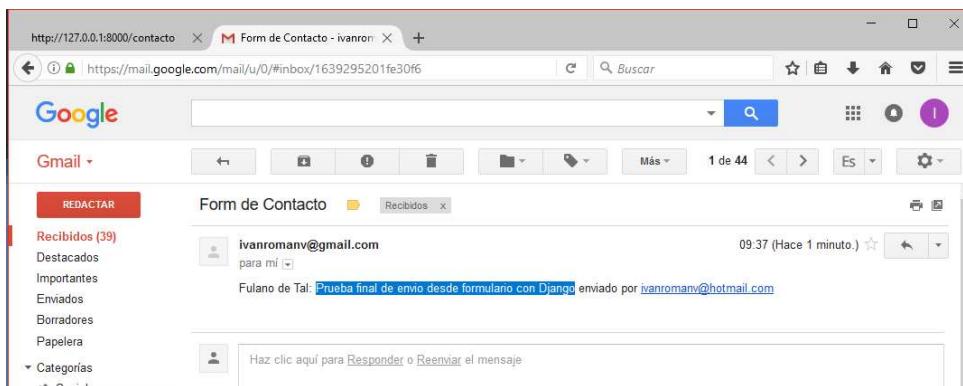


The screenshot shows a web browser window with the URL <http://127.0.0.1:8000/contacto>. The page contains a form with the following fields:

- Nombre: Fulan de Tal
- Email: anromany@hotmail.com
- Mensaje: Prueba final de envío desde formulario con Django

At the bottom of the form is a "Regístrate" button.

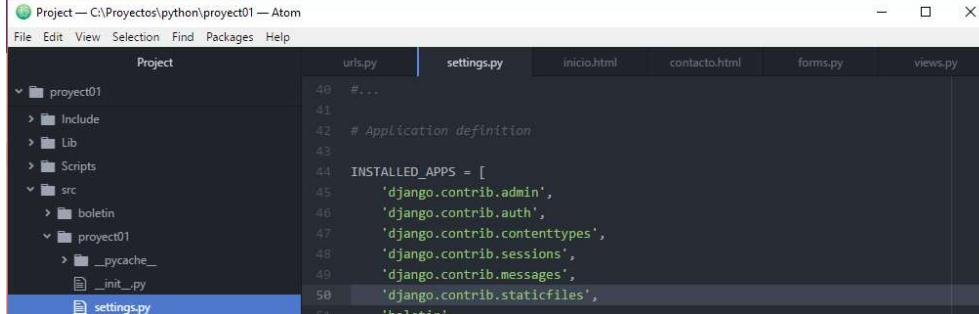
Se comprueba que el correo fue correctamente enviado desde servidor smtp.



## 22. Configuración de Archivos Estáticos

- El objetivo es la configuración de archivos estáticos (imágenes y javascripts) en un entorno de desarrollo y producción.  
Se encargará de la migración de los datos entre ambos ambientes, **así como la activación del mismo como desarrollo.**

1. Para este proceso modificamos el archivo **setting.py**. En donde **Verificamos** que este agregado en **INSTALLED\_APPS** el **staticfiles**.



```

Project— C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py settings.py inicio.html contacto.html forms.py views.py
  proyecto01
    > Include
    > Lib
    > Scripts
    > src
      > boletin
      > proyecto01
        > __pycache__
        __init__.py
settings.py
40 #...
41
42 # Application definition
43
44 INSTALLED_APPS = [
45     'django.contrib.admin',
46     'django.contrib.auth',
47     'django.contrib.contenttypes',
48     'django.contrib.sessions',
49     'django.contrib.messages',
50     'django.contrib.staticfiles',
51     'boletin'
52 ]

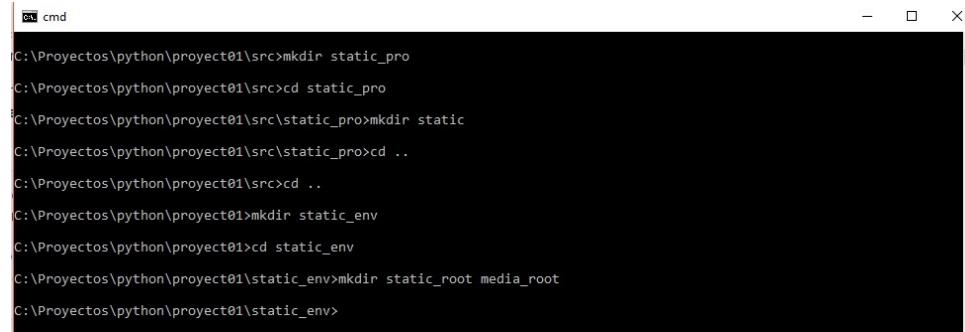
```

2. Creamos los directorios **a nivel de src** llamados:

**static\_pro** y debajo otro llamado **static**.

Creamos los directorios **antes** de **src** a nivel del proyecto llamados:

**static\_env** y debajo otro llamado **static\_root** y **media\_root**



```
cmd
C:\Proyectos\python\proyecto01>mkdir static_pro
C:\Proyectos\python\proyecto01>cd static_pro
C:\Proyectos\python\proyecto01\src>mkdir static
C:\Proyectos\python\proyecto01\src\static_pro>cd ..
C:\Proyectos\python\proyecto01>cd ..
C:\Proyectos\python\proyecto01>mkdir static_env
C:\Proyectos\python\proyecto01>cd static_env
C:\Proyectos\python\proyecto01\static_env>mkdir static_root media_root
C:\Proyectos\python\proyecto01\static_env>
```

3. Ahora agregando **STATIC\_URL** al archivo **settings.py**, agregando al final.

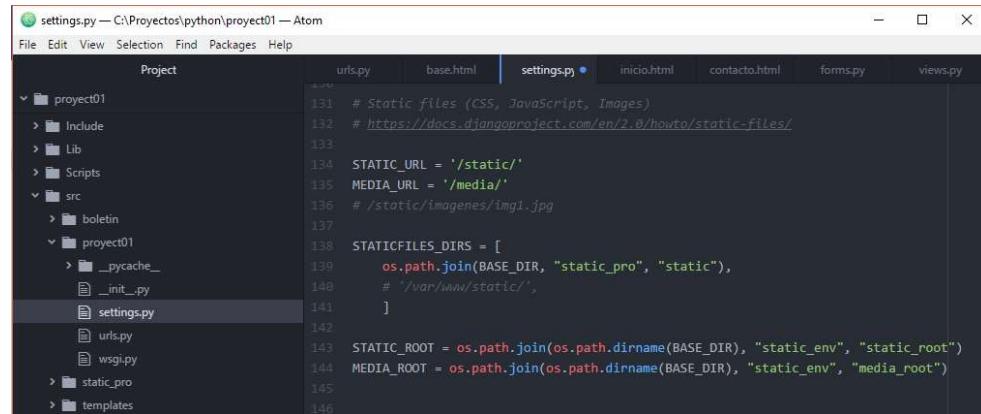
Archivo **settings.py**:

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/
```

```
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
```

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static_pro", "static"),
    # '/var/www/static/',
]
```

```
STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "static_root")
MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "media_root")
```



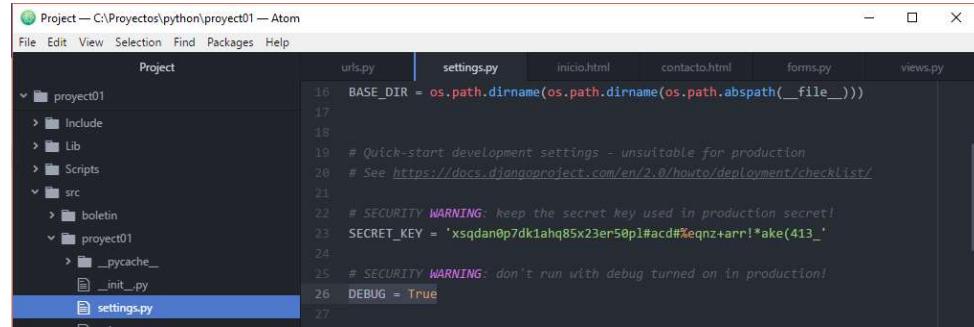
```
settings.py — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py base.html settings.py • inicio.html contacto.html forms.py views.py
  - □ ×
  project01
    > Include
    > Lib
    > Scripts
    > src
      > boletin
      > proyecto01
        > __pycache__
        > __init__.py
        > settings.py
        > urls.py
        > wsgi.py
      > static_pro
      > templates
131 # Static files (CSS, JavaScript, Images)
132 # https://docs.djangoproject.com/en/2.0/howto/static-files/
133
134 STATIC_URL = '/static/'
135 MEDIA_URL = '/media/'
136 # /static/imagenes/img1.jpg
137
138 STATICFILES_DIRS = [
139     os.path.join(BASE_DIR, "static_pro", "static"),
140     # '/var/www/static/',
141 ]
142
143 STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "static_root")
144 MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "media_root")
```

**IMPORTANTE:**

**Para desarrollo solamente,** Como medida de seguridad para evitar problemas la opción:

**NOTA:**

<b>DEBUG = TRUE</b>	<b>#Estamos en desarrollo</b>
<b>DEBUG = FALSE</b>	<b>#Estamos en producción</b>



The screenshot shows the Atom code editor with a project structure on the left. The current file is 'settings.py'. The line 'DEBUG = True' is highlighted with a blue selection bar. The code in 'settings.py' includes comments about security and deployment.

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py settings.py inicio.html contacto.html forms.py views.py
  proyecto01
    > Include
    > Lib
    > Scripts
    > src
      > boletin
      > proyecto01
        > __pycache__
          __init__.py
        settings.py
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'xsqdan0p7dk1ahq85x23er50p1#acd%eqnz+arr!*ake(413_'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27

```

4. Ahora configurando las **urls**, en el archivo **urls.py** importar **settings** y **static**, además agregar al final:

Archivo **urls.py**:

```

from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings
from django.conf.urls.static import static

from boletin import views
#from boletin.views inicio

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
]

if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

5. Ejecutamos a nivel de comandos en la ruta **src**, para actualizar los archivos en las réplicas.

D:\Proyectos\python\proyecto01\src>**python manage.py collectstatic**

The screenshot shows the Atom code editor with the file 'urls.py' open. The project structure on the left includes 'src' and 'static'. The 'src' folder contains 'boletin' and 'project01'. 'boletin' has files like \_\_init\_\_.py, admin.py, apps.py, forms.py, models.py, tests.py, and views.py. 'project01' contains db.sqlite3, manage.py, static\_env, and tcl. The 'static' folder contains templates with files base.html, contacto.html, and inicio.html. The 'urls.py' code is as follows:

```

urls.py — C:\Proyectos\python\project01 — Atom
File Edit View Selection Find Packages Help
Project urls.py base.html settings.py inicio.html contacto.html forms.py views.py
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 #Importando setting y static
19 from django.conf import settings
20 from django.conf.urls.static import static
21
22 from boletin import views
23 #from boletin.views.inicio
24
25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('inicio', views.inicio, name='inicio'),
28     path('contacto', views.contacto, name='contacto'),
29 ]
30
31 if settings.DEBUG:
32     urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
33     urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
34

```

At the bottom, it says 'CRLF' and 'Python'.

Finalmente, verificamos luego que los archivos que fueron migrados hacia las carpetas administrativas, pero aun no existen archivos estáticos, hasta cuando los carguemos.

The screenshot shows a Windows Command Prompt window titled 'Seleccionar cmd'. The command 'python manage.py collectstatic' was run, and the output shows the copying of static files from various Python library directories to the project's static folder. The output is as follows:

```

C:\Proyectos\python\project01\src>python manage.py collectstatic
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\autocomplete.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\base.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\changelists.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\dashboard.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\fons.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\forms.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\login.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\responsive.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\responsive_rtl.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\rtl.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\widgets.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\select2\LICENSE-SELECT2.md'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\select2\select2.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\css\select2\select2.min.css'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\fonts\LICENSE.txt'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\fonts\README.txt'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\fonts\Roboto-Bold-webfont.woff'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\fonts\Roboto-Light-webfont.woff'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\fonts\Roboto-Regular-webfont.woff'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\calendar-icons.svg'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\icon-addlink.svg'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\icon-alert.svg'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\icon-calendar.svg'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\icon-changelink.svg'
Copying 'C:\Program Files\Python36\lib\site-packages\django\contrib\admin\static\admin\img\icon-clock.svg'

```

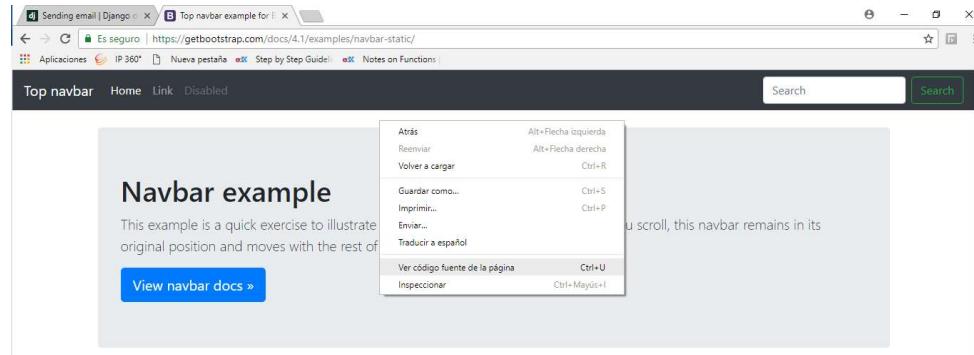
## 23. Configuración Bootstrap (<https://getbootstrap.com>)

- Con el objetivo de dar una apariencia mejor y diseño adaptable a nuestras aplicaciones web, vamos a ir a descargar **bootstrap** desde la página web <https://getbootstrap.com>. Con la cual descargaremos código css y javascript.

Para esto, para este ejemplo vamos a descargar una plantilla existente, la misma que se encuentra en el sitio web dentro de **examples**. Se escogió esta template, pero se puede escoger el que mejor convenga a su proyecto.

<https://getbootstrap.com/docs/4.1/examples/navbar-static/>

1. Vamos a la página, luego **botón derecho, ver código fuente de la página**. Sobre dicho código vamos a respaldar ciertos archivos en carpetas nuevas que vamos a crear sobre la estructura de nuestro proyecto.



#### Archivo navbar-static:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<link rel="icon" href="../../../../favicon.ico">
<title>Top navbar example for Bootstrap</title>
<!-- Bootstrap core CSS -->
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
<!-- Custom styles for this template -->
<link href="navbar-top.css" rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
<a class="navbar-brand" href="#">#Top navbar</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarCollapse">
<ul class="navbar-nav mr-auto">
<li class="nav-item active">
<a class="nav-link" href="#">#Home <span class="sr-only">(current)</span></a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">#Link</a>
</li>
<li class="nav-item">
<a class="nav-link disabled" href="#">#Disabled</a>
</li>

```

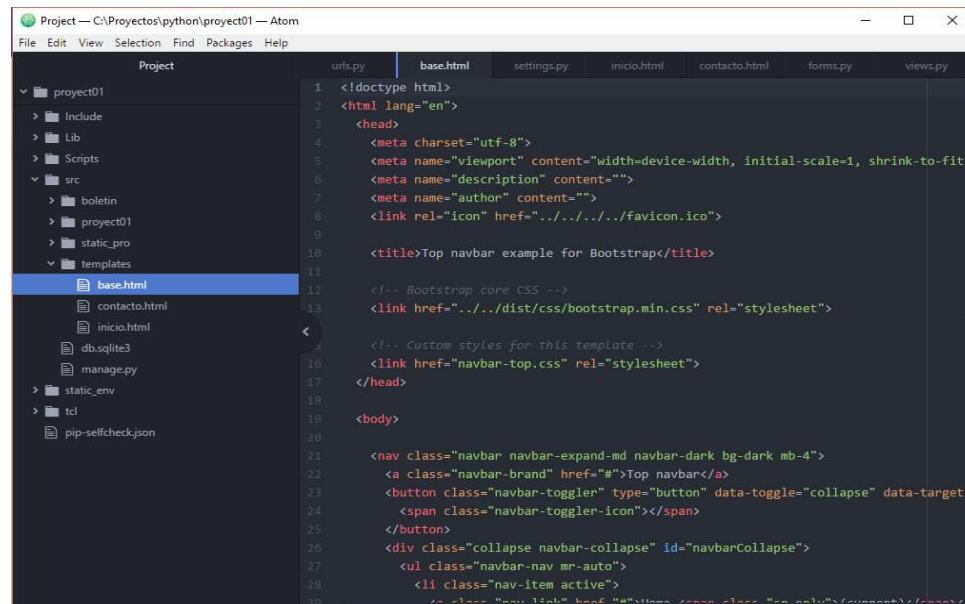
```

</ul>
<form class="form-inline mt-2 mt-md-0">
<input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
</form>
</div>
</nav>

<main role="main" class="container">
<div class="jumbotron">
<h1>Navbar example</h1>
<p class="lead">This example is a quick exercise to illustrate how the top-aligned navbar works. As you scroll, this navbar remains in its original position and moves with the rest of the page.</p>
<a class="btn btn-lg btn-primary" href="#">../components/navbar/https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo" crossorigin="anonymous"></script>
<script>window.jQuery || document.write('<script src="..../assets/js/vendor/jquery-slim.min.js"></script>')</script>
<script src="#">../assets/js/vendor/popper.min.js></script>
<script src="#">../dist/js/bootstrap.min.js></script>
</body>
</html>

```

- Copiamos todo el código y lo pegamos en la carpeta llamada **templates** con el nombre **base.html**



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with a 'templates' folder containing 'base.html'. The main editor area shows the content of 'base.html'.

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py base.html settings.py inicio.html contacto.html forms.py views.py
  |- proyecto01
    |- Include
    |- Lib
    |- Scripts
  |- src
    |- boletin
    |- proyecto01
    |- static_pro
  |- templates
    |- base.html
      |- contacto.html
      |- inicio.html
      |- db.sqlite3
      |- manage.py
    |- static_env
    |- tcl
    |- pip-selfcheck.json

```

```

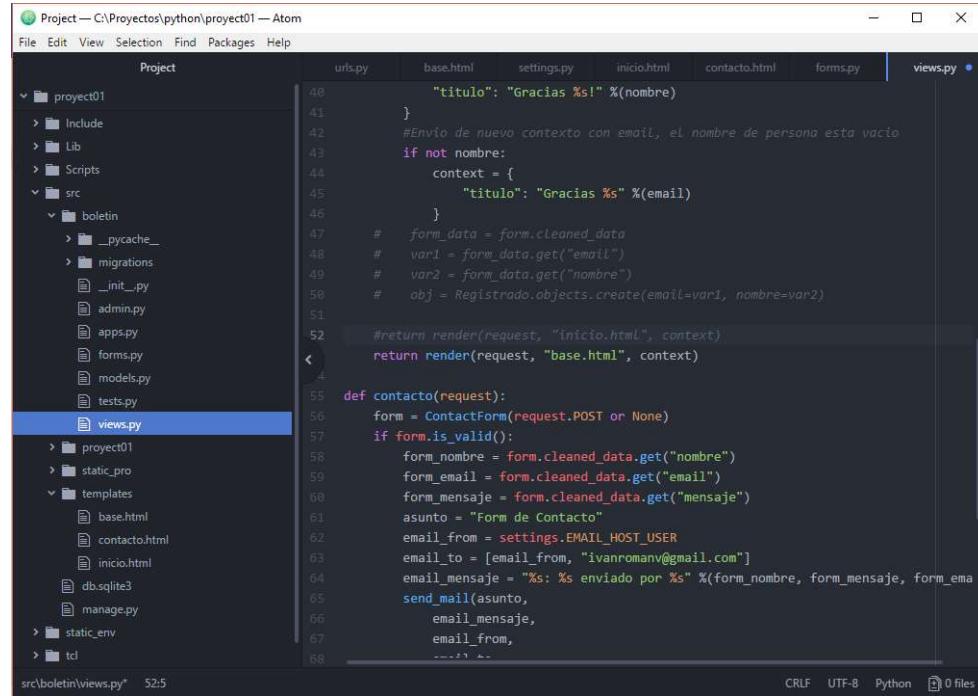
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit">
6      <meta name="description" content="">
7      <meta name="author" content="">
8      <link rel="icon" href="..../..../favicon.ico">
9
10     <title>Top navbar example for Bootstrap</title>
11
12     <!-- Bootstrap core CSS --&gt;
13     &lt;link href="..../dist/css/bootstrap.min.css" rel="stylesheet"&gt;
14
15     <!-- Custom styles for this template --&gt;
16     &lt;link href="navbar-top.css" rel="stylesheet"&gt;
17   &lt;/head&gt;
18
19   &lt;body&gt;
20
21     &lt;nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4"&gt;
22       &lt;a class="navbar-brand" href="#"&gt;Top navbar&lt;/a&gt;
23       &lt;button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse"&gt;
24         &lt;span class="navbar-toggler-icon"&gt;&lt;/span&gt;
25       &lt;/button&gt;
26       &lt;div class="collapse navbar-collapse" id="navbarCollapse"&gt;
27         &lt;ul class="navbar-nav mr-auto"&gt;
28           &lt;li class="nav-item active"&gt;
29             &lt;a class="nav-link" href="#"&gt;Home &lt;span class="sr-only">(current)</span></a>
30           </li>
31         </ul>
32       </div>
33     </nav>
34
35     <div class="container">
36       <h1>Top navbar example for Bootstrap</h1>
37       <p>This example is a quick exercise to illustrate how the top-aligned navbar works. As you scroll, this navbar remains in its original position and moves with the rest of the page.</p>
38       <a class="btn btn-lg btn-primary" href="#">../components/navbar/

```

- Vamos al archivo **views.py** y cambiamos la renderizacion de la página de **inicio.html** por **base.html**, luego observamos los cambios.

Archivo **views.py**:

```
#return render(request, "inicio.html", context)
return render(request, "base.html", context)
```

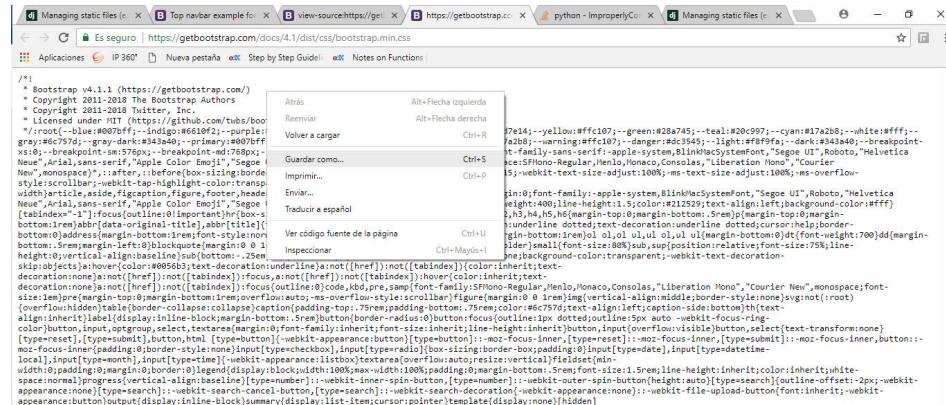


```
Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py base.html settings.py inicio.html contacto.html forms.py views.py
src
  proyecto01
    Include
    Lib
    Scripts
    src
      boletin
        __pycache__
        migrations
        __init__.py
        admin.py
        apps.py
        forms.py
        models.py
        tests.py
        views.py
      proyecto01
      static_pro
      templates
        base.html
        contacto.html
        inicio.html
      db.sqlite3
      manage.py
      static_env
      tcl
src\boletin\views.py 52:5
40           "titulo": "Gracias %s!" %(nombre)
41       }
42       #Envio de nuevo contexto con email, el nombre de persona esta vacio
43       if not nombre:
44           context = {
45               "titulo": "Gracias %s" %(email)
46           }
47       #   form_data = form.cleaned_data
48       #   var1 = form_data.get("email")
49       #   var2 = form_data.get("nombre")
50       #   obj = Registrado.objects.create(email=var1, nombre=var2)
51
52       #return render(request, "inicio.html", context)
53       return render(request, "base.html", context)
54
55 def contacto(request):
56     form = ContactForm(request.POST or None)
57     if form.is_valid():
58         form_nombre = form.cleaned_data.get("nombre")
59         form_email = form.cleaned_data.get("email")
60         form_mensaje = form.cleaned_data.get("mensaje")
61         asunto = "Form de Contacto"
62         email_from = settings.EMAIL_HOST_USER
63         email_to = [email_from, "ivanroman@gmail.com"]
64         email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje, form_email)
65         send_mail(asunto,
66                    email_mensaje,
67                    email_from,
68                    email_to)
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
```

- Deberemos crear las carpetas tanto para **css**, **js**, para las hojas de estilo por lo tanto debemos agregarlos al proyecto en `\src\static_pro\static\`

D:\Proyectos\python\proyecto01\src\static\_pro\static\mkdir css js img

- En estas carpetas grabaremos los archivos que se encuentran en el código `base.html` por el tipo de archivo, descargando cada uno de los links. **Botón derecho, guardar como:** (use el mismo nombre de archivo).

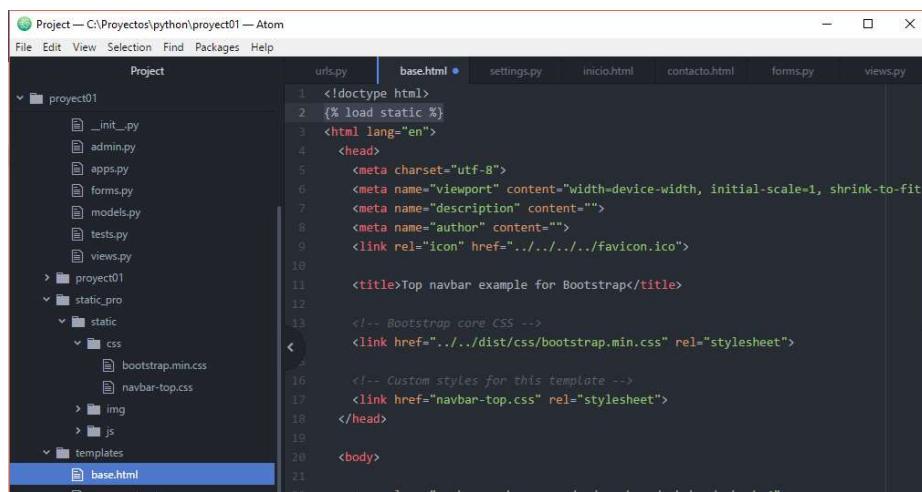


- Crear etiqueta en el `base.html` para llamar a la renderización `% load static %`

Archivo `base.html`:

```
<!doctype html>
```

```
{% load static %}
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
.....
```



The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

- Project
  - project01
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - forms.py
    - models.py
    - tests.py
    - views.py
  - project01
    - static\_pro
      - static
        - css
          - bootstrap.min.css
          - navbar-top.css
        - img
        - js
      - templates
        - base.html

7. Modificar el código **base.html** para cambiar las rutas de acceso a los archivos de estilo que fueron descargados en **css, js**, a través de la etiqueta.

Dependiendo del tipo de archivo clasificar las rutas donde se los tenga almacenados en la carpeta **\static\_pro\static\css, \static\_pro\static\js, \static\_pro\static\img**

Las rutas para **\static\_pro\static\img**:

/img/favicon.ico

Las rutas para **\static\_pro\static\css**:

/css/bootstrap.min.css

/css/navbar-top.css

Las rutas para **\static\_pro\static\js**:

/js/jquery-3.3.1.slim.min.js

/js/popper.min.js

/js/bootstrap.min.js

Archivo **base.html**:

```
{% load static %}
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<!--
<link rel="icon" href="../../../../favicon.ico">
.....
```

**Aquí agregamos el código necesario para la etiqueta -->**

```

<link rel="icon" href="{% static '/img/favicon.ico' %}">
<title>Top navbar example for Bootstrap</title>
<!-- Bootstrap core CSS -->
<!--
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
Aquí agregamos el código necesario para la etiqueta -->
<link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">

<!-- Custom styles for this template -->
<!--
<link href="navbar-top.css" rel="stylesheet">
Aquí agregamos el código necesario para la etiqueta -->
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
</head>

<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline mt-2 mt-md-0">
      <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
      <button type="submit" class="btn btn-outline-success my-2 my-sm-0">Search</button>
    </form>
  </div>
</nav>

<main role="main" class="container">
  <div class="jumbotron">
    <h1>Navbar example</h1>

```

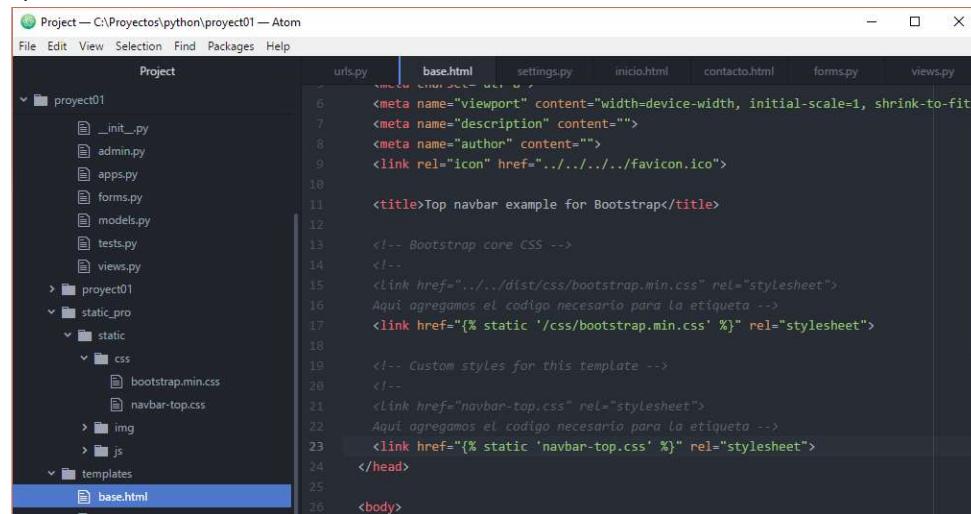
<p class="lead">This example is a quick exercise to illustrate how the top-aligned navbar works. As you scroll, this navbar remains in its original position and moves with the rest of the page.</p>

```

<a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">View navbar docs &gt;</a>
</div>
</main>

<!-- Bootstrap core JavaScript
=====
-->
<!-- Placed at the end of the document so the pages load faster
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK4JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
crossorigin="anonymous"></script>
Aquí agregamos el código necesario para la etiqueta-->
<script src="{% static '/js/jquery-3.3.1.slim.min.js' %}" integrity="sha384-q8i/X+965DzO0rT7abK4JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
crossorigin="anonymous"></script>
<script>window.jQuery || document.write('<script src="../../assets/js/vendor/jquery-slim.min.js"></script>')
<!--
<script src="../../assets/js/vendor/popper.min.js"></script>
Aquí agregamos el código necesario para la etiqueta-->
<script src="{% static '/js/popper.min.js' %}"></script>
<!--
<script src="../../dist/js/bootstrap.min.js"></script>
Aquí agregamos el código necesario para la etiqueta-->
<script src="{% static '/js/bootstrap.min.js' %}"></script>
</body>
</html>

```



The screenshot shows the Atom code editor with the following details:

- Project:** C:\Proyectos\python\proyect01
- File:** base.html
- Content:**

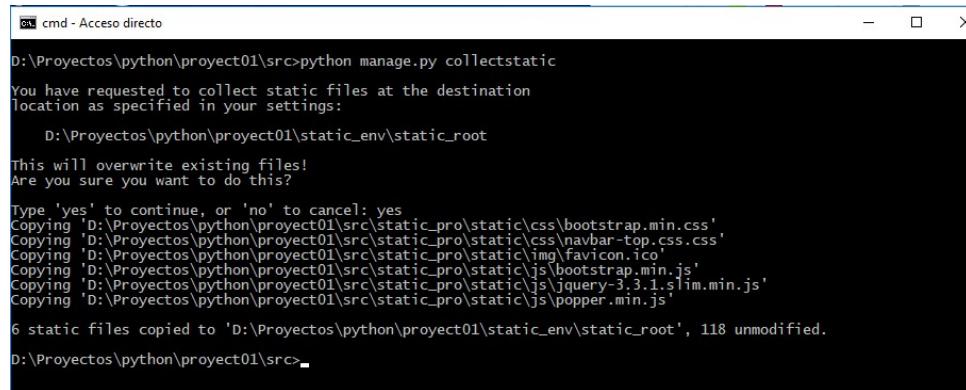
```

6   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit"
7   <meta name="description" content=""
8   <meta name="author" content=""
9   <link rel="icon" href="../../../../favicon.ico">
10
11  <title>Top navbar example for Bootstrap</title>
12
13  <!-- Bootstrap core CSS -->
14  <!--
15  <link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
16  Aquí agregamos el código necesario para la etiqueta -->
17  <link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">
18
19  <!-- Custom styles for this template -->
20  <!--
21  <link href="navbar-top.css" rel="stylesheet">
22  Aquí agregamos el código necesario para la etiqueta -->
23  <link href="{% static 'navbar-top.css' %}" rel="stylesheet">
24
25
26  </head>
<body>

```

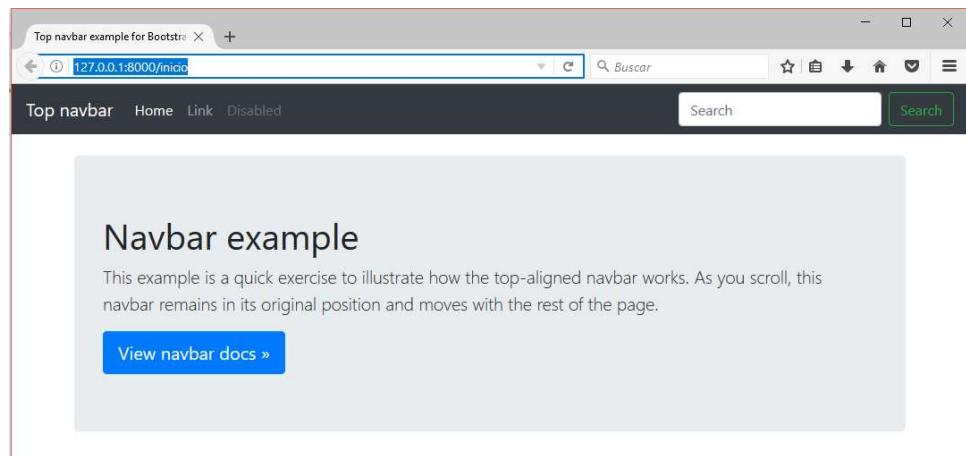
- Ejecutamos a nivel de comandos en la ruta **src**, para actualizar los archivos en las réplicas.

D:\Proyectos\python\proyect01\src>**python manage.py collectstatic**

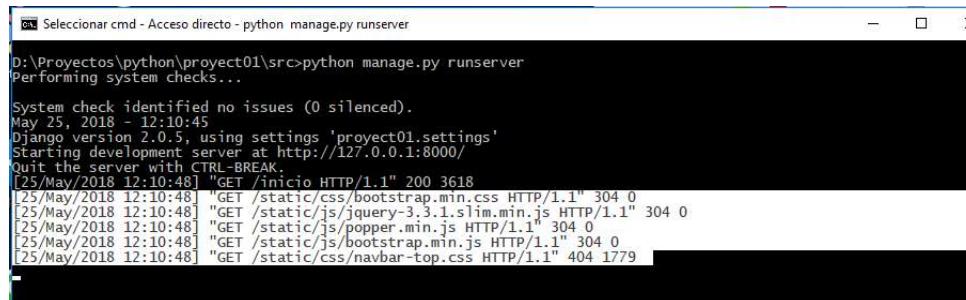


```
D:\Proyectos\python\proyecto01\src>python manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:
    D:\Proyectos\python\proyecto01\static_env\static_root
This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\css\bootstrap.min.css'
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\css\navbar-top.css.css'
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\img\favicon.ico'
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\js\bootstrap.min.js'
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\js\jquery-3.3.1.slim.min.js'
Copying 'D:\Proyectos\python\proyecto01\src\static_pro\static\js\popper.min.js'
6 static files copied to 'D:\Proyectos\python\proyecto01\static_env\static_root', 118 unmodified.
D:\Proyectos\python\proyecto01\src>
```

Probamos la interfaz web con <http://127.0.0.1:8000/inicio>



Aparecen los archivos en las rutas de instalación estáticas.



```
D:\Proyectos\python\proyecto01\src>python manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
May 25, 2018 - 12:10:45
Django version 2.0.5, using settings 'proyecto01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[25/May/2018 12:10:48] "GET /inicio HTTP/1.1" 200 3619
[25/May/2018 12:10:48] "GET /static/css/bootstrap.min.css HTTP/1.1" 304 0
[25/May/2018 12:10:48] "GET /static/js/jquery-3.3.1.slim.min.js HTTP/1.1" 304 0
[25/May/2018 12:10:48] "GET /static/js/popper.min.js HTTP/1.1" 304 0
[25/May/2018 12:10:48] "GET /static/js/bootstrap.min.js HTTP/1.1" 304 0
[25/May/2018 12:10:48] "GET /static/css/navbar-top.css HTTP/1.1" 404 1779
```

## 24. Plantillas Herencia, Include Tag, Blocks

- Para usar plantillas dentro de las plantillas evitar usar código repetitivo, mediante la plantilla **base.html** como fuente de todo el proyecto, la que será modificada y particionada en otros archivos para observar la funcionalidad de la **herencia, include tag y los blocks**.

### Include

Para el ejemplo crearemos en la carpeta **templates** una plantilla llamada **navbar.html** y **jumbotron**, en donde pegaremos el corte de la plantilla **base.html** que corresponde a la etiqueta **nav** y **jumbotron** para grabarlo dentro del archivo **.html** respectivamente. Luego podremos en el archivo **base.html** una etiqueta que llame al archivo **navbar.html**, asi: **{% include "navbar.html" %}** e **{% include "jumbotron.html" %}**

Archivo **navbar.html**:

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline mt-2 mt-md-0">
      <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
```

The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with files like urls.py, base.html, navbar.html, styles.less, inicio.html, contacto.html, admin.py, views.py, forms.py, and setting. The right pane shows the content of the navbar.html file:

```

1 <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
2   <a class="navbar-brand" href="#">Top navbar</a>
3   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse">
4     <span class="navbar-toggler-icon"></span>
5   </button>
6   <div class="collapse navbar-collapse" id="navbarCollapse">
7     <ul class="navbar-nav mr-auto">
8       <li class="nav-item active">
9         <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
10      </li>
11      <li class="nav-item">
12        <a class="nav-link" href="#">Link</a>
13      </li>
14      <li class="nav-item">
15        <a class="nav-link disabled" href="#">Disabled</a>
16      </li>
17    </ul>
18    <form class="form-inline mt-2 mt-md-0">
19      <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
20      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
21    </form>
22  </div>
23 </nav>
24

```

El archivo **jumbotron.html** contendrá:

#### Archivo jumbotron.html:

```

<main role="main" class="container">
<div class="jumbotron">
  <h1>Navbar example</h1>
  <p class="lead">This example is a quick exercise to illustrate how the top-aligned
  navbar works. As you scroll, this navbar remains in its original position and moves with
  the rest of the page.</p>
  <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">View navbar docs &gt;</a>
</div>
</main>

```

The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with files like urls.py, base.html, navbar.html, jumbotron.html, settings.py, inicio.html, contacto.html, forms.py, and view. The right pane shows the content of the jumbotron.html file:

```

1 <main role="main" class="container">
2   <div class="jumbotron">
3     <h1>Navbar example</h1>
4     <p class="lead">This example is a quick exercise to illustrate how the top-aligned
5     <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">View navbar docs &gt;</a>
6   </div>
7 </main>
8

```

Creamos el archivo **javascript.html**, que proviene del archivo **base.html**, no olvidar colocar **{% load static %}** en la cabecera del archivo.

#### Archivo javascript.html:

```
% load static%
<!-- Bootstrap core JavaScript
=====
<!-- Placed at the end of the document so the pages load faster
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK4JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
Aqui agregamos el codigo necesario para la etiqueta-->
<script src="{% static '/js/jquery-3.3.1.slim.min.js' %}" integrity="sha384-q8i/X+965DzO0rT7abK4JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<!-- <script>window.jQuery || document.write('<script'
src="../assets/js/vendor/jquery-slim.min.js"></script>')</script> -->
<script>window.jQuery || document.write('<script src="{% static '/js/jquery-
3.3.1.slim.min.js' %}"></script>')</script>
<!--
<script src="../assets/js/vendor/popper.min.js"></script>
Aqui agregamos el codigo necesario para la etiqueta-->
<script src="{% static '/js/popper.min.js' %}"></script>
<!--
<script src="../dist/js/bootstrap.min.js"></script>
Aqui agregamos el codigo necesario para la etiqueta-->
<script src="{% static '/js/bootstrap.min.js' %}"></script>
```

The screenshot shows the Atom code editor interface. The title bar reads "Project — D:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The left sidebar displays a project structure with a "Project" view and several sub-directories like "Include", "Lib", "Scripts", and "src". The "src" directory is expanded, showing sub-folders "boletin", "proyecto01", "static\_pro", and "templates". Inside "templates", files like "registration", "about.html", "base.html", "contacto.html", "head.css.html", and "inicio.html" are listed. The file "javascript.html" is currently selected and highlighted with a blue background. The main editor area contains the code provided in the previous block, which is a template for loading static assets like Bootstrap and jQuery.

Archivo **head\_css.html**, recientemente creado.

Archivo **head\_css.html**:

```
% load static%
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
<meta name="description" content="">
```

```

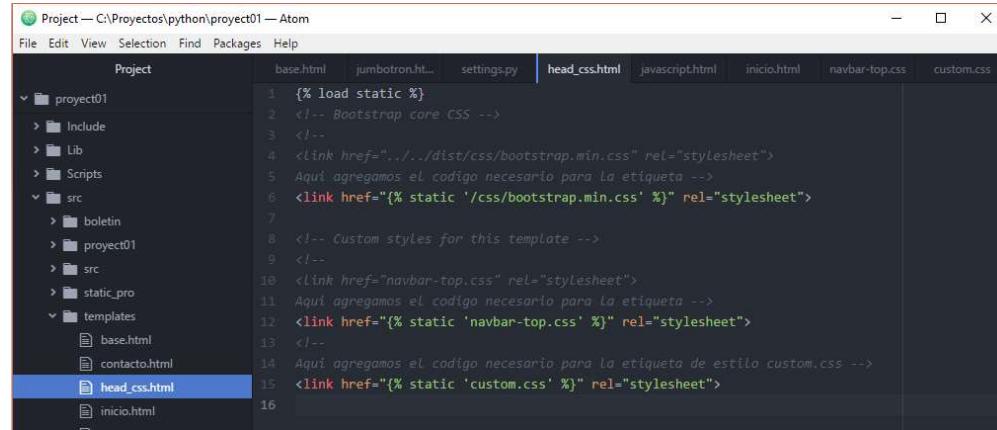
<meta name="author" content="">
<link rel="icon" href="{% static '/img/favicon.ico' %}">

<title>{% block titulo_sitio%}Proyecto01{% endblock %}</title>

<!-- Bootstrap core CSS -->
<!--
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">

<!-- Custom styles for this template -->
<!--
<link href="navbar-top.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">

```



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with a tree view of files and folders. The main editor area shows the content of the `head_css.html` file, which contains the code provided in the previous block. The code includes comments in Spanish explaining the purpose of each section.

El archivo **base.html** será el siguiente.

#### Archivo **base.html**:

```

<!doctype html>
{% load static %}
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <meta name="description" content="">
        <meta name="author" content="">
        <!--
        <link rel="icon" href="../../../../favicon.ico">
Aqui agregamos el codigo necesario para la etiqueta -->
<link rel="icon" href="{% static '/img/favicon.ico' %}">
<title>Top navbar example for Bootstrap</title>

```

```

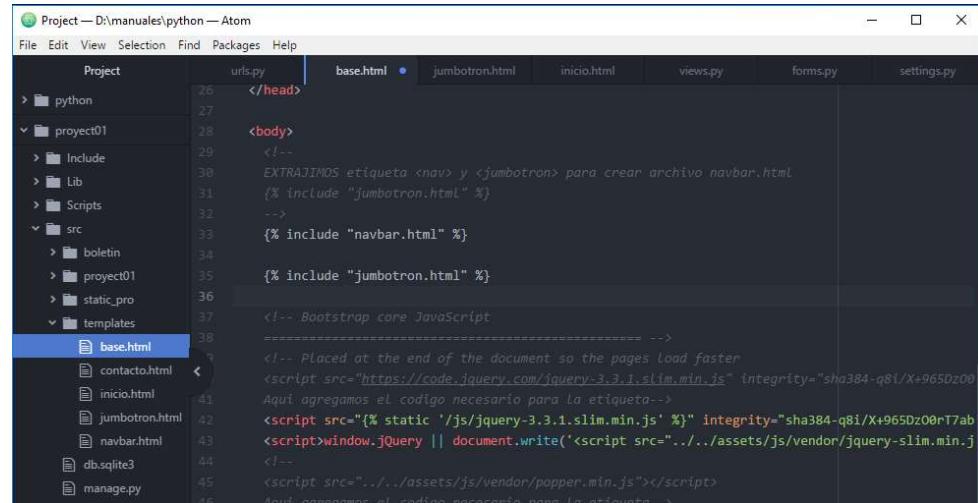
<!--
EXTRAJIMOS etiqueta <head> para crear archivo head_css.html
-->
{% include "head_css.html" %} </head>

<body>
<!--
EXTRAJIMOS etiqueta <nav>, <jumbotron> y <javascript>para crear archivo
navbar.html
-->
{% include "navbar.html" %}

{% include "jumbotron.html" %}

{% include "javascript.html" %}
</body>
</html>

```



The screenshot shows the Atom code editor interface. The project structure on the left includes files like urls.py, base.html (which is the active tab), jumbotron.html, inicio.html, views.py, forms.py, and settings.py. The base.html content is as follows:

```

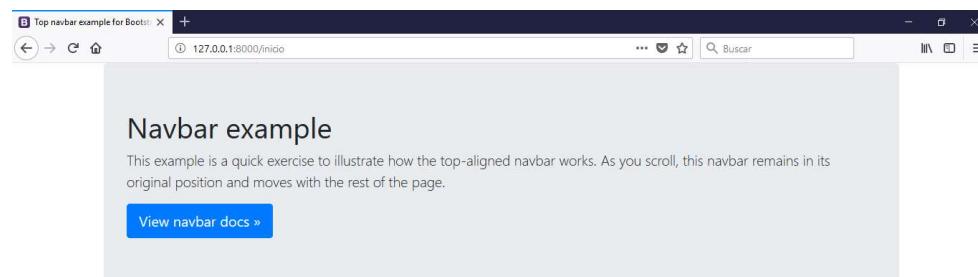
<!--
EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html
-->
{% include "jumbotron.html" %}

{% include "navbar.html" %}

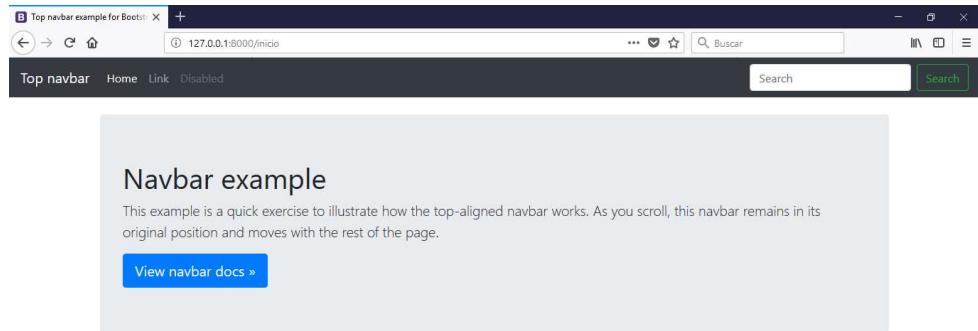
<!-- Bootstrap core JavaScript
=====
&lt;!-- Placed at the end of the document so the pages load faster
&lt;script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz0rT7ab
Aqui agregamos el código necesario para la etiqueta--&gt;
&lt;script src="{% static '/js/jquery-3.3.1.slim.min.js' %}" integrity="sha384-q8i/X+965Dz0rT7ab
&lt;script&gt;window.jQuery || document.write('&lt;script src="..../assets/js/vendor/jquery-slim.min.j
&lt;!--
&lt;script src="..../assets/js/vendor/popper.min.js"&gt;&lt;/script&gt;
Aqui agregamos el código necesario para la etiqueta--&gt;
</pre>

```

Si comentamos la etiqueta en **base.html**, no se mostrará la cabecera **nav**, como veremos a continuación.



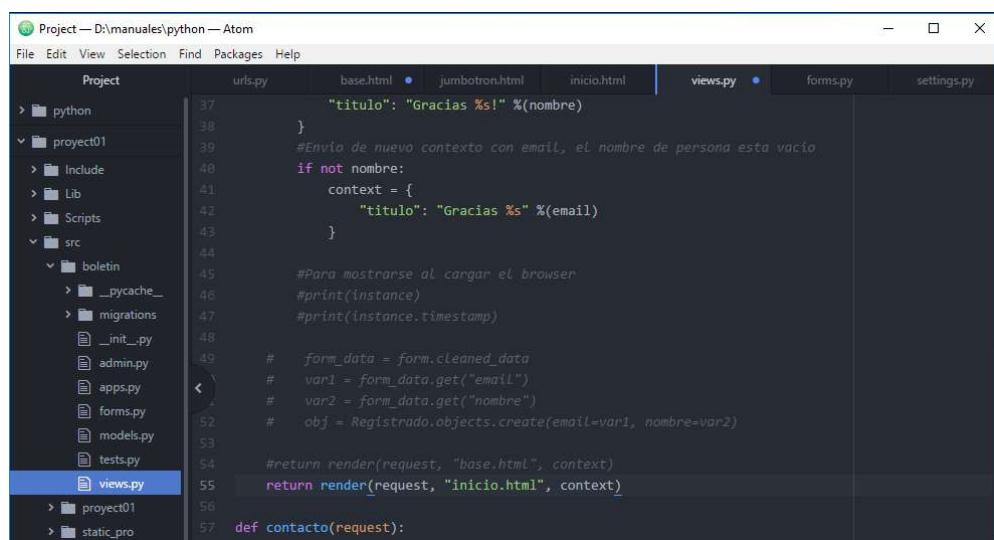
Al descomentar la línea veremos normalmente la página.



## Extends

Extiende código **html** o hereda proveniente de otra página. Por ejemplo, creamos un bloque dentro de **inicio.html** para decirle que vamos a heredar desde **base.html**

Para la página **inicio.html** tendremos que hacer cambios en el archivo **views.py** para decirle que cargue la página indicada como inicial.



```

Project -- D:\manuales\python -- Atom
File Edit View Selection Find Packages Help
Project urls.py base.html • jumbotron.html inicio.html views.py • forms.py settings.py
> python
> proyecto01
> Include
> Lib
> Scripts
> src
  > boletin
    > __pycache__
    > migrations
    > __init__.py
    > admin.py
    > apps.py
    > forms.py
    > models.py
    > tests.py
    > views.py
  > proyecto01
  > static_pro
37     "titulo": "Gracias %s!" %(nombre)
38   }
39   #Envío de nuevo contexto con email, el nombre de persona está vacío
40   if not nombre:
41       context = {
42           "titulo": "Gracias %s" %(email)
43       }
44       #Para mostrarse al cargar el browser
45       #print(instance)
46       #print(instance.timestamp)
47       #
48       # form_data = form.cleaned_data
49       # var1 = form_data.get("email")
50       # var2 = form_data.get("nombre")
51       # obj = Registrado.objects.create(email=var1, nombre=var2)
52       #
53       #return render(request, "base.html", context)
54       return render(request, "inicio.html", context)
55   def contacto(request):
56
57

```

Modificar el archivo **inicio.html** el cual contendrá las definiciones de la **extends** y los bloques de código definidos dentro de **base.html** que serán insertados dentro de la página de **inicio.html**

### Archivo **inicio.html**:

```

<!doctype html>
{% load static %}
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

```

```

<meta name="description" content="">
<meta name="author" content="">
<!--
<link rel="icon" href="../../../../favicon.ico">
Aqui agregamos el codigo necesario para la etiqueta -->
<link rel="icon" href="{% static '/img/favicon.ico' %}">
<title>Top navbar example for Bootstrap</title>

<!-- Bootstrap core CSS -->
<!--
EXTRAJIMOS etiqueta <head> para crear archivo head_css.html
-->
{% include "head_css.html" %} </head>
</head>

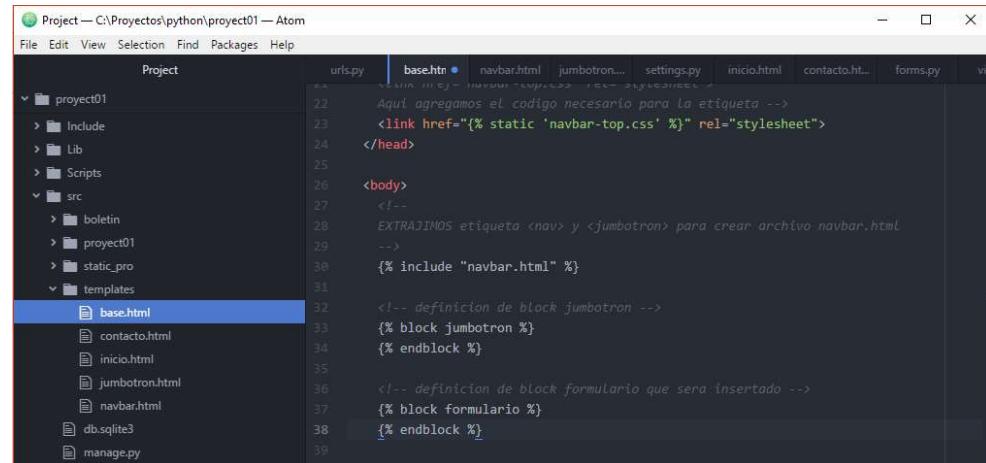
<body>
<!--
EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html
-->
{% include "navbar.html" %}

<!-- definicion de block jumbotron -->
{% block jumbotron %}
{% endblock %}

<!-- definicion de block formulario que sera insertado -->
{% block formulario %}
{% endblock %}

<!--
EXTRAJIMOS etiqueta <head> para crear archivo javascript.html
-->
{% include "javascript.html" %} </body>
</html>

```



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with folders like 'project01', 'Include', 'Lib', 'Scripts', 'src' (containing 'boletin' and 'project01'), 'static\_pro', and 'templates'. Inside 'templates', the 'base.html' file is selected and shown in the main editor area. The code in 'base.html' includes meta tags for description and author, a link to a favicon, a title, Bootstrap core CSS, and a head section containing a navigation bar and a jumbotron. It also includes definitions for 'jumbotron' and 'formulario' blocks, and a javascript section.

El archivo **inicio.html** quedara de la siguiente manera, en donde se definirán el block **jumbotron** y **formulario**, igualmente definidos dentro de **base.html**:

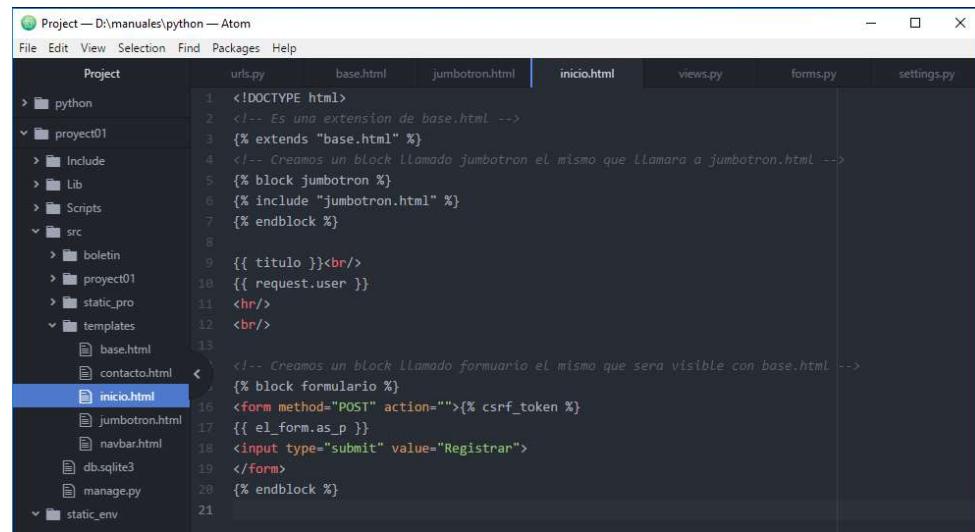
#### Archivo **inicio.html**:

```
<!DOCTYPE html>
<!-- Es una extension de base.html --&gt;
{% extends "base.html" %}
<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html --&gt;
{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

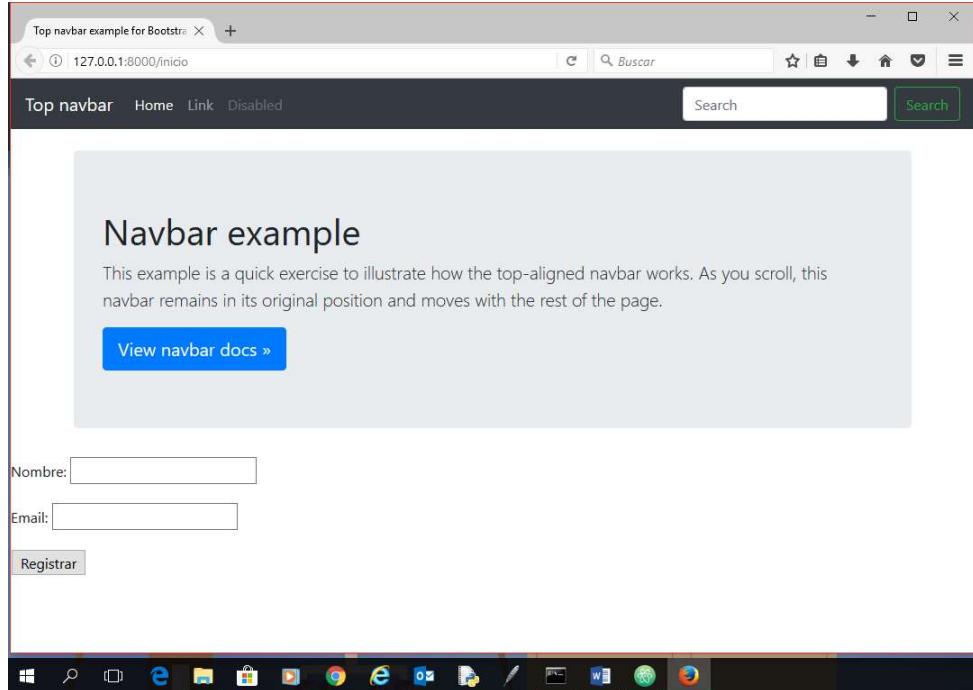
{{ titulo }}&lt;br/&gt;
{{ request.user }}
&lt;hr/&gt;
&lt;br/&gt;

<!-- Creamos un block llamado formulario el mismo que será visible con base.html --&gt;
{% block formulario %}
&lt;form method="POST" action=""&gt;{{ csrf_token %}}
{{ el_form.as_p }}
&lt;input type="submit" value="Registrar"&gt;
&lt;/form&gt;
{% endblock %}</pre>

```



The screenshot shows the Atom code editor interface. The top menu bar includes File, Edit, View, Selection, Find, Packages, Help, and a tab for the current file, 'inicio.html'. Below the menu is a toolbar with icons for Project, urls.py, base.html, jumbotron.html, inicio.html (which is highlighted), views.py, forms.py, and settings.py. The left sidebar displays the project structure under 'Project' with 'python' and 'proyecto01' expanded, showing 'Include', 'Lib', 'Scripts', 'src' (with 'boletin', 'proyecto01', 'static\_pro', and 'templates' subfolders), and 'static\_env'. The main editor area shows the code for 'inicio.html' with line numbers from 1 to 21. Lines 1-7 define the base structure and a 'jumbotron' block. Lines 8-12 show the 'request.user' variable and a horizontal separator. Lines 13-21 define the 'formulario' block, which includes a form with a csrf token, a form field, a submit button, and an endblock tag.



Podemos también colocar block donde queramos en este caso modificaremos el título del sitio, tanto en el archivo **base.html** e **inicio.html**, de la siguiente manera.

#### Archivo **base.html**:

```
<!doctype html>
{% load static %}
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <!--
        <link rel="icon" href="../../../../favicon.ico">
        Aqui agregamos el codigo necesario para la etiqueta -->
        <link rel="icon" href="{% static '/img/favicon.ico' %}">

    <title>{% block titulo_sitio%}Proyecto01{% endblock %}</title>

    <!-- Bootstrap core CSS -->
    <!--
        EXTRAJIMOS etiqueta <head> para crear archivo head_css.html
    -->
    {% include "head_css.html" %} </head>
</head>

<body>
```

```

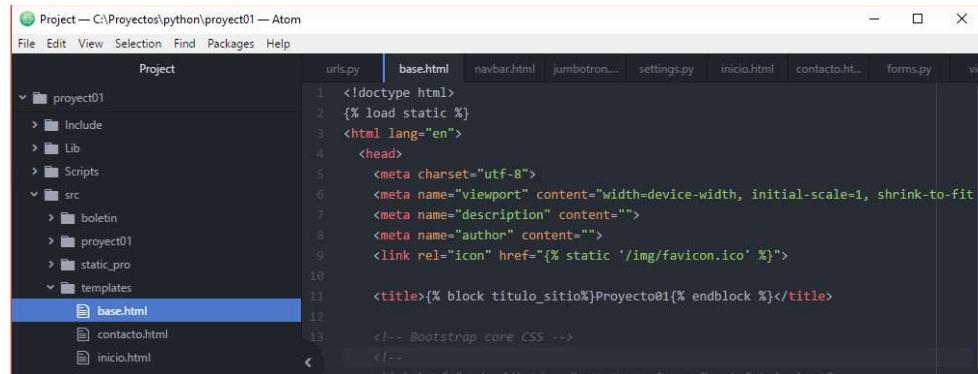
<!--
EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html
-->
{% include "navbar.html" %}

<!-- definicion de block jumbotron -->
{% block jumbotron %}
{% endblock %}

<!-- definicion de block formulario que sera insertado -->
{% block formulario %}
{% endblock %}

<!--
EXTRAJIMOS etiqueta <head> para crear archivo javascript.html
-->
{% include "javascript.html" %} </body>
</html>

```



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with folders like 'project01', 'Include', 'Lib', 'src', and 'templates'. Inside 'templates', the 'base.html' file is selected and shown in the main editor area. The code in 'base.html' includes standard HTML headers and meta tags, along with Django template syntax for blocks and includes.

#### Archivo inicio.html:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}

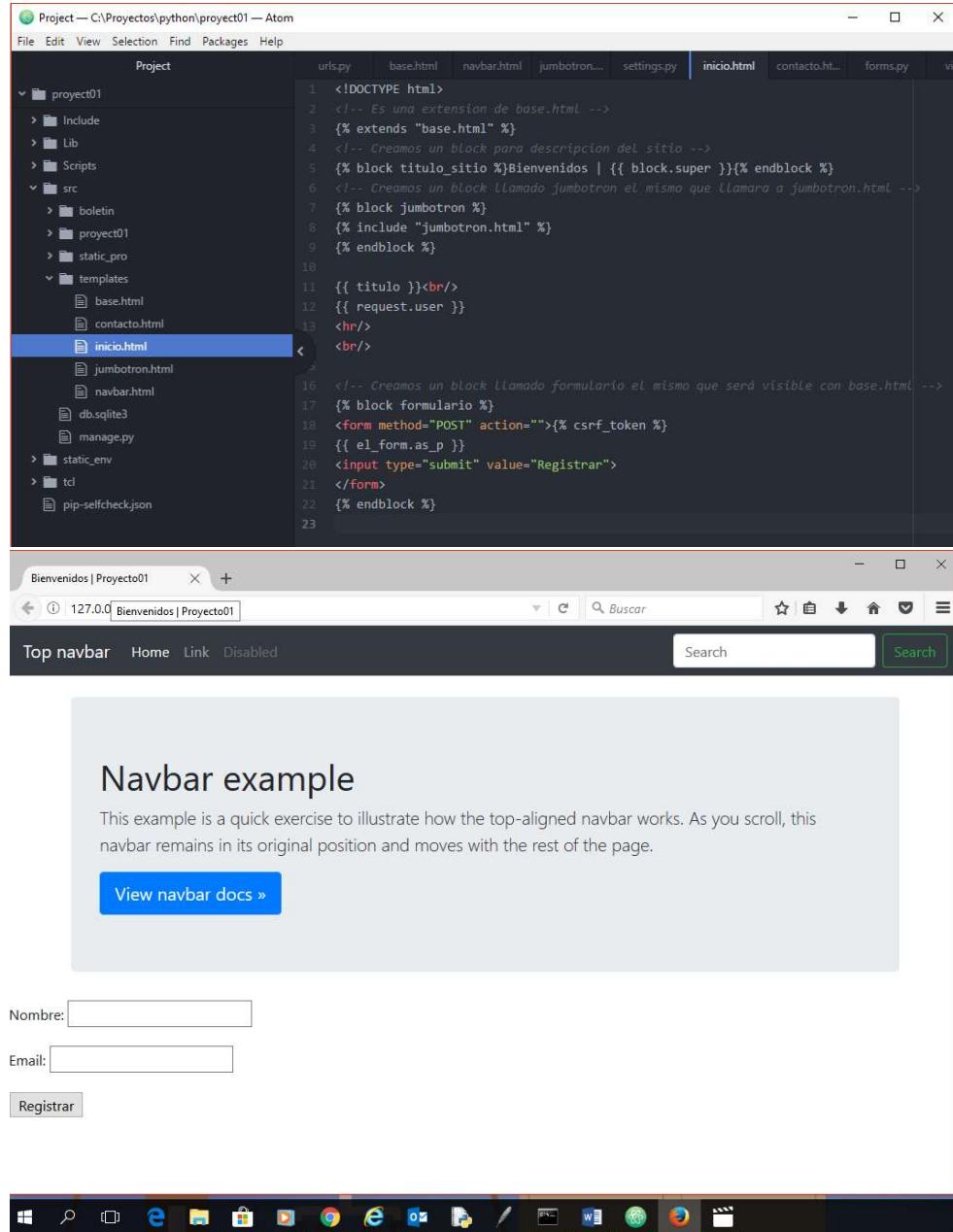
<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}

```

```
<form method="POST" action="">{% csrf_token %}
{{ el_form.as_p }}
<input type="submit" value="Registrar">
</form>
{% endblock %}
```



The screenshot shows a Django project structure in the Atom code editor:

- Project:** C:\Proyectos\python\project01 — Atom
- File:** Edit View Selection Find Packages Help
- Project Tree:**
  - project01
    - Include
    - Lib
    - Scripts
    - src
      - boletin
      - project01
      - static\_pro
      - templates
        - base.html
        - contacto.html
        - inicio.html**
        - jumbotron.html
        - navbar.html
      - db.sqlite3
      - manage.py
    - static\_env
    - tcl
    - pip-selfcheck.json
- Code Editor:** The `inicio.html` file contains the following code:

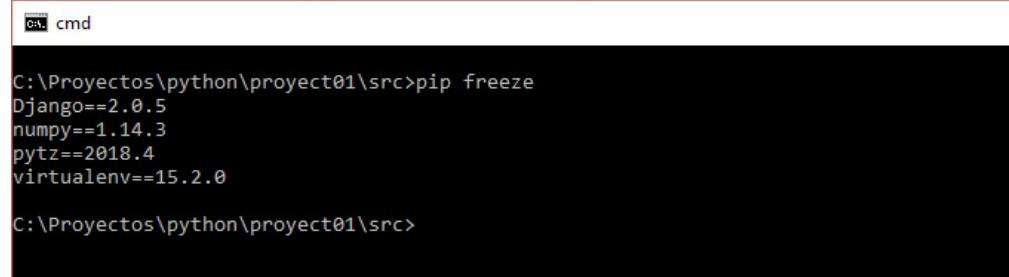
```
1  <!DOCTYPE html>
2  <!-- Es una extension de base.html -->
3  {% extends "base.html" %}
4  <!-- Creamos un block para descripcion del sitio -->
5  {% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
6  <!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
7  {% block jumbotron %}
8  {% include "jumbotron.html" %}
9  {% endblock %}
10 <hr>
11 {{ titulo }}<br/>
12 {{ request.user }}
13 <hr>
14 <br/>
15 <!-- Creamos un block llamado formulario el mismo que sera visible con base.html -->
16 {% block formulario %}
17 <form method="POST" action="">{% csrf_token %}
18 {{ el_form.as_p }}
19 <input type="submit" value="Registrar">
20 </form>
21
22 {% endblock %}
```
- Browser Preview:** A browser window titled "Bienvenidos | Proyecto01" shows the rendered HTML. The page has a top navigation bar with links: Top navbar, Home, Link, Disabled. Below the navbar, the main content area displays the "Navbar example" heading and a paragraph about how the top-aligned navbar works. A "View navbar docs »" button is present.
- Bottom Taskbar:** A Windows taskbar is visible at the bottom, showing various application icons.

## 25. Django Crispy Forms

- Paquete de Django permite añadir características visuales mejoradas los formularios

Comprobamos que el paquete este instalado mediante freeze.

D:\Proyectos\python\project01\src>**pip freeze**

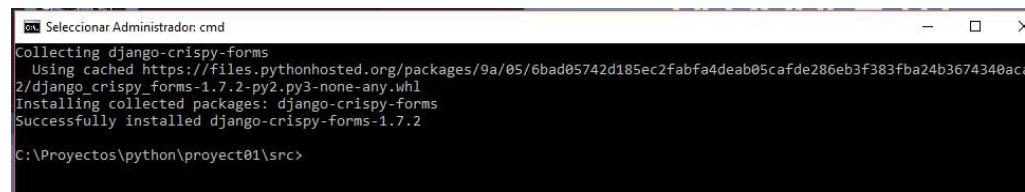


```
C:\Proyectos\python\proyecto01\src>pip freeze
Django==2.0.5
numpy==1.14.3
pytz==2018.4
virtualenv==15.2.0

C:\Proyectos\python\proyecto01\src>
```

Luego procedemos a la instalación manual.

D:\Proyectos\python\proyecto01\src>**pip install -U django-crispy-forms**



```
Se seleccionar Administrador: cmd
Collecting django-crispy-forms
  Using cached https://files.pythonhosted.org/packages/9a/05/6bad05742d185ec2fabfa4deab05cafde286eb3f383fba24b3674340ac2/django_crispy_forms-1.7.2-py2.py3-none-any.whl
Installing collected packages: django-crispy-forms
Successfully installed django-crispy-forms-1.7.2

C:\Proyectos\python\proyecto01\src>
```

Luego agregar la nueva app al archivo **settings.py**

Archivo **settings.html**:

*# Application definition*

```
INSTALLED_APPS = [
    # apps Django
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # apps de terceros
    'crispy_forms',
    # apps creadas
    'boletin',
]
```

Se procede a realizar las migraciones respectivas para que **Django** se actualice a nivel de apps.

D:\Proyectos\python\proyecto01\src>**python manage.py makemigrations**

D:\Proyectos\python\proyecto01\src>**python manage.py migrate**

```
C:\ Proyectos\python\proyecto01\src>python manage.py makemigrations
No changes detected

C:\ Proyectos\python\proyecto01\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, sessions
Running migrations:
  No migrations to apply.

C:\ Proyectos\python\proyecto01\src>
```

Vamos a template packs, ya que **crispy-forms** trabaja con **bootstrap**, entonces seleccionamos la versión correspondiente agregando en el archivo **settings.py** la ruta y versión, como sigue:

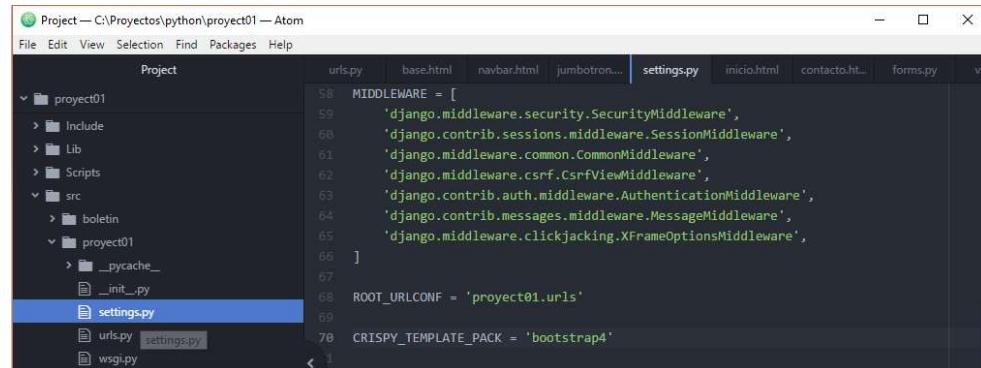
#### Archivo **settings.html**:

...

**ROOT\_URLCONF = 'proyecto01.urls'**

**CRISPY\_TEMPLATE\_PACK = 'bootstrap4'**

...



Finalmente, en el archivo **inicio.html** agregamos código para decirle que usaremos **crispy-form**, además cambiamos **form** por **crispy** con el símbolo “|”, para cargar formulario **el\_form**

#### Archivo **inicio.html**:

```
<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
```

```

{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

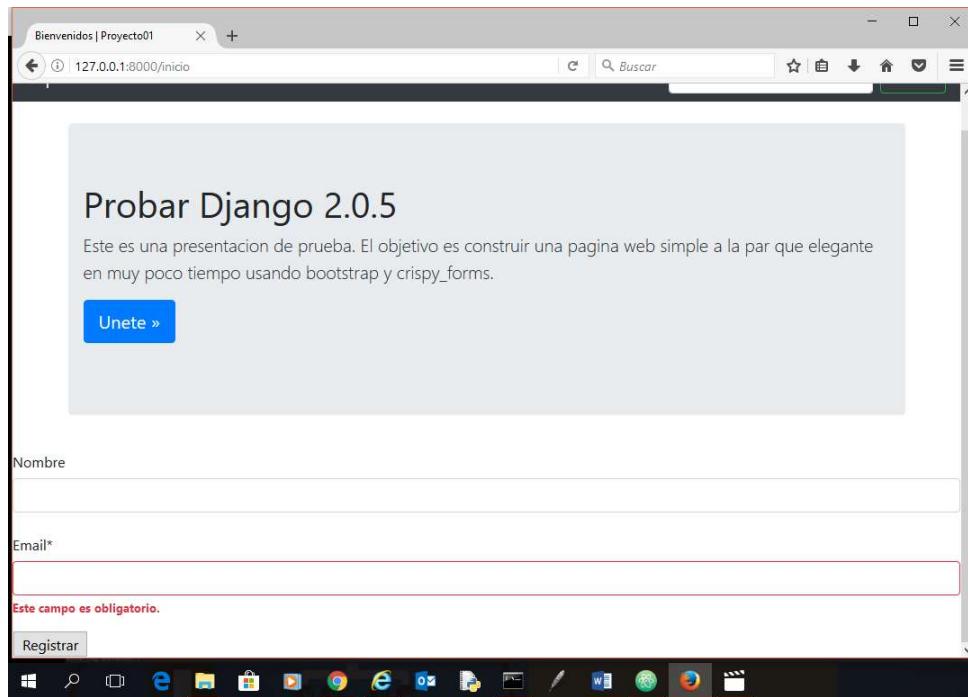
{{ titulo }}<br/>
{{ request.user }}

<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html --&gt;
{% block formulario %}
&lt;form method="POST" action=""&gt;{{ csrf_token }}
<!-- Cambiamos el form a crispy para mejor presentacion
{{ el_form.form }}
--&gt;
{{ el_form|crispy }}
&lt;input type="submit" value="Registrar"&gt;
&lt;/form&gt;
{% endblock %}
</pre>

```

La página web se mostrará de la siguiente manera, mejorando su presentación.



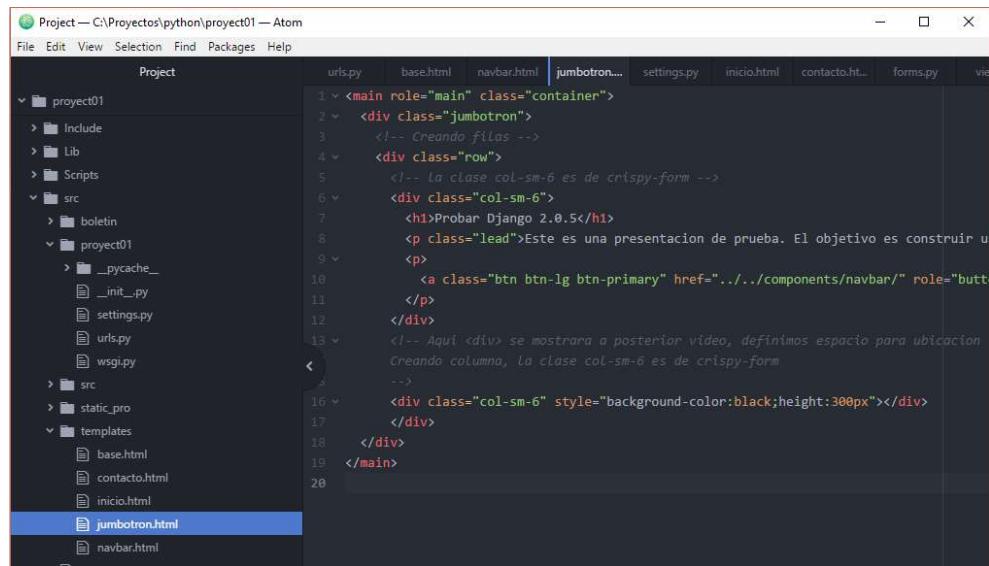
## 26. Estilo Bootstrap 1

- Dando estilo y formato a la página de **inicio.html**, previamente modificamos **jumbotron.html** agregando más **columnas** `<div>` y **párrafos** `<p>` con descripciones comprensibles.

Para el ejemplo crearemos una columna adicional en **jumbotron.html** donde colocaremos texto y a futuro un video, en **inicio.html**, en la parte de abajo el formulario más 3 columnas donde presentaremos texto.

#### Archivo jumbotron.html:

```
%load static%
<main role="main" class="container">
  <div class="jumbotron">
    <!-- Creando filas -->
    <div class="row">
      <!-- la clase col-sm-6 es de crispy-form -->
      <div class="col-sm-6">
        <h1>Probar Django 2.0.5</h1>
        <p class="lead">Este es una presentacion de prueba. El objetivo es construir una pagina web simple y elegante en muy poco tiempo usando bootstrap y crispy_forms.</p>
        <p>
          <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">Unete &raquo;</a>
        </p>
      </div>
      <!-- Aqui <div> se mostrara a posterior video, definimos espacio para ubicacion Creando columna, la clase col-sm-6 es de crispy-form -->
      <div class="col-sm-6" style="background-color:black;height:300px"></div>
    </div>
  </div>
</main>
```



#### Archivo inicio.html:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}
<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}
<div class="row">
<!--
    la clase col-sm-3 es de crispy-form numero de columnas 3
    la clase col-xs-12 es de crispy-form ajusta al comprimir la página
    la clase pull-right es de crispy-form alinea a la derecha
-->
<div class="col-sm-3 col-xs-12 pull-right">
    <p class="lead">{{ titulo }}</p>
    <form method="POST" action="">{{ csrf_token }}
    <!-- Cambiamos el form a crispy para mejor presentacion
    {{ el_form.form }}>
    <!--
        {{ el_form|crispy }}>
        <input class="btn btn-primary" type="submit" value="Inscribirse">
    </form>
</div>
<div class="col-sm-3">
    <p class="lead">Pagina creado con framework Django.</p>
</div>
<div class="col-sm-3">
    <p class="lead">Diseñado especialmente.</p>
</div>
<div class="col-sm-3">
    <p class="lead">Con codigo abierto.</p>
</div>
</div>
{% endblock %}

```

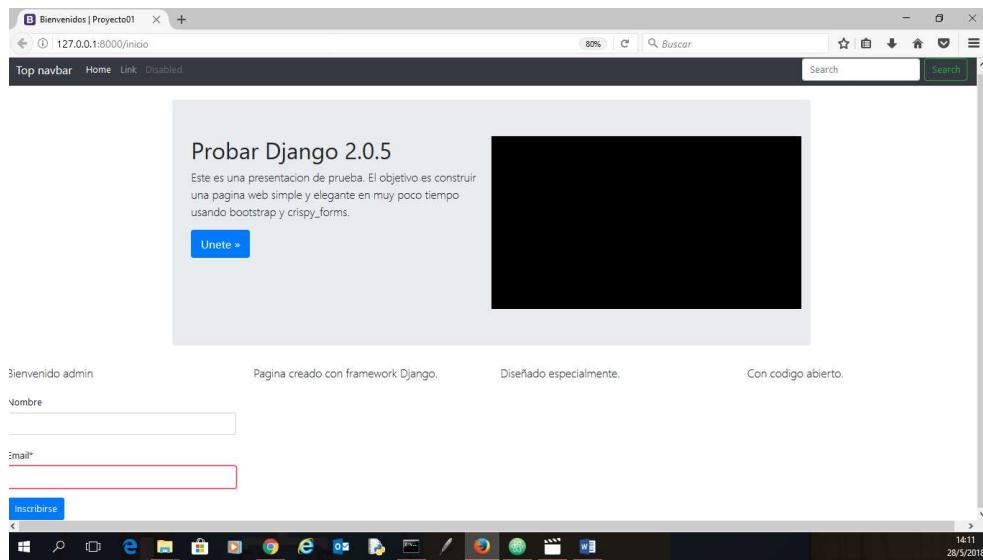
```

20  {% block formulario %}
21  <div class="row">
22  </!--
23  la clase col-sm-3 es de crispy-form numero de columnas 3
24  la clase col-xs-12 es de crispy-form ajusta al comprimir La pagina
25  la clase pull-right es de crispy-form alinea a la derecha
26  -->
27  <div class="col-sm-3 col-xs-12 pull-right">
28  <p class="lead">{{ titulo }}</p>
29  <form method="POST" action="">{{ csrf_token }}
30  <!-- Cambiamos el form a crispy para mejor presentacion
31  {{ el_form.form }}>
32  -->
33  {{ el_form|crispy }}
34  <input class="btn btn-primary" type="submit" value="Inscribirse">
35  </form>
36  </div>
37  <div class="col-sm-3">
38  <p class="lead">Pagina creado con framework Django.</p>
39  </div>
40  <div class="col-sm-3">
41  <p class="lead">Diseñado especialmente.</p>
42  </div>
43  <div class="col-sm-3">
44  <p class="lead">Con codigo abierto.</p>
45  </div>
46  </div>
47  {% endblock %}

```

src\templates\inicio.html 46:7

Obtendremos en el browser:



Si lo justificamos u observamos por el browser del celular, obtendríamos:



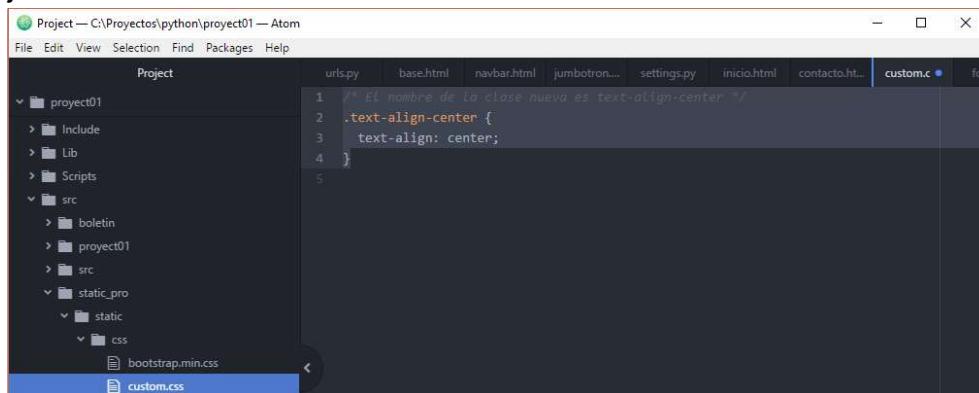
## 27. Estilo CSS Custom

- Crearemos un archivo denominado **custom.css** la misma que es un estilo nuevo en la carpeta **\proyecto01\static\css\**, en donde definiremos nuestras propias **clases personalizadas** para aplicar a nuestras etiquetas, de esta forma podremos llamarlas a lo largo del proyecto, así como también crearemos la ruta de su ubicación, para luego actualizarlas en nuestro servidor de producción.

Creamos el archivo **custom.css** el cuan contendrá.

Archivo **custom.html**:

```
/* El nombre de la clase nueva es text-align-center */
.text-align-center {
    text-align: center;
}
```



Procedemos a la creación del archivo **head\_css.html**, a partir del archivo **base.html** luego realizando el **%include%** respectivo en el archivo de **inicio.html**

Archivo **head\_css.html**, AGREGAREMOS la ruta del estilo **custom.css**, recientemente creado, no olvidar colocar **{% load static %}** en la cabecera del archivo.

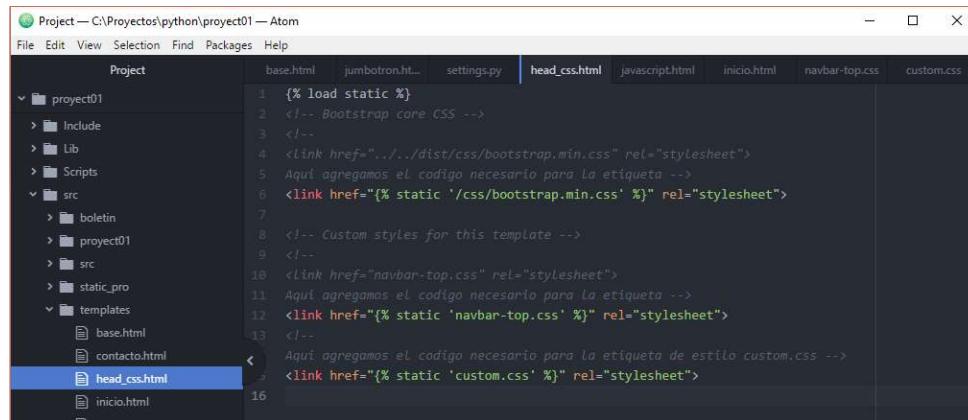
#### Archivo **head\_css.html**:

```

{% load static %}
<!-- Bootstrap core CSS -->
<!--
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">

<!-- Custom styles for this template -->
<!--
<link href="navbar-top.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
<!--
Aqui agregamos el codigo necesario para la etiqueta de estilo custom.css -->
<link href="{% static '/css/custom.css' %}" rel="stylesheet">

```



En el archivo **javascript\_css.html**, no olvidar colocar **{% load static %}** en la cabecera del archivo.

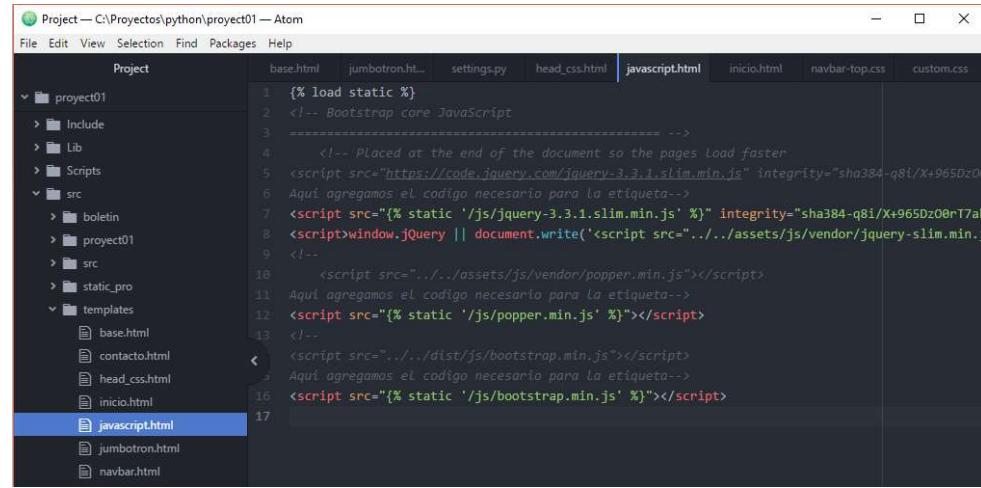
#### Archivo **javascript\_css.html**:

```

{% load static %}
<!-- Bootstrap core JavaScript
=====
<!-- Placed at the end of the document so the pages load faster
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzOOrT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
crossorigin="anonymous"></script>
Aqui agregamos el codigo necesario para la etiqueta-->

```

```
<script src="{% static '/js/jquery-3.3.1.slim.min.js' %}" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
crossorigin="anonymous"></script>
<script>window.jQuery || document.write('<script src="../../assets/js/vendor/jquery-slim.min.js"></script>')
<!--
<script src="../../assets/js/vendor/popper.min.js"></script>
Aqui agregamos el codigo necesario para la etiqueta-->
<script src="{% static '/js/popper.min.js' %}"></script>
<!--
<script src="../../dist/js/bootstrap.min.js"></script>
Aqui agregamos el codigo necesario para la etiqueta-->
<script src="{% static '/js/bootstrap.min.js' %}"></script>
```



Archivo **base.html**, es cada vez **MAS REDUCIDO**, optimizando el código.

#### Archivo **base.html**:

```
<!doctype html>
{% load static %}
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<!--
<link rel="icon" href="../../favicon.ico">
Aqui agregamos el codigo necesario para la etiqueta -->
<link rel="icon" href="{% static '/img/favicon.ico' %}">
<!--
<title>Top navbar example for Bootstrap</title> -->
<title>{{ block titulo_sitio }}Proyecto01{{ endblock }}</title>
<!--
```

```

EXTRAJIMOS etiqueta <head> para crear archivo head_css.html -->
{%
  include "head_css.html"
}

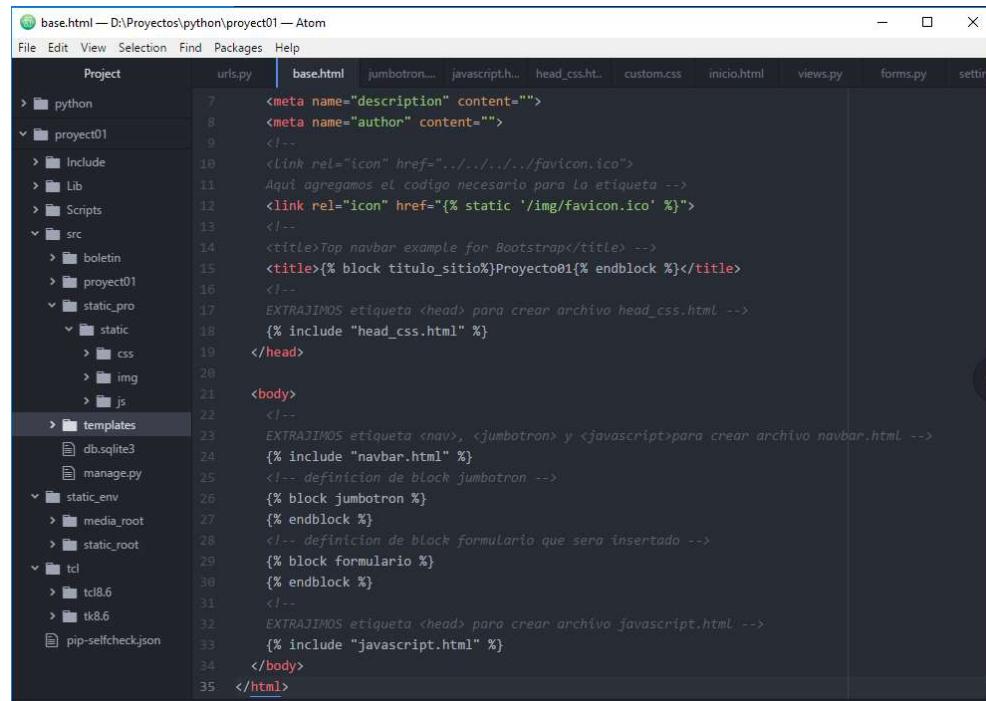
</head>

<body>
<!--

EXTRAJIMOS etiqueta <nav>, <jumbotron> y <javascript>para crear archivo
navbar.html -->
{%
  include "navbar.html"
}
<!-- definicion de block jumbotron -->
{%
  block jumbotron %
}
{%
  endblock %
}
<!-- definicion de block formulario que sera insertado -->
{%
  block content %
}
{%
  endblock %
}
<!-- definicion de block content para redux registration que sera insertado -->
{%
  block content %
}
{%
  endblock %
}
<!--

EXTRAJIMOS etiqueta <head> para crear archivo javascript.html -->
{%
  include "javascript.html"
}
</body>
</html>

```



```

base.html — D:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project urls.py base.html jumbotron... javascript.h... head_css.ht... custom.css inicio.html views.py forms.py setting
python
  proyecto01
    Include
    Lib
    Scripts
    src
      boletin
      proyecto01
      static_pro
        static
          css
          img
          js
      templates
        db.sqlite3
        manage.py
    static_env
    media_root
    static_root
    tcl
      tcl8.6
      tk8.6
    pip-selfcheck.json
7   <meta name="description" content="">
8   <meta name="author" content="">
9   <!--
10  <Link rel="icon" href="../../../../favicon.ico">
11  Aquí agregamos el código necesario para la etiqueta -->
12  <link rel="icon" href="{% static '/img/favicon.ico' %}">
13  <!--
14  <title>Top navbar example for Bootstrap</title> -->
15  <title>{{ block titulo_sitio }}Proyecto01{{ endblock }}</title>
16  <!--
17  EXTRAJIMOS etiqueta <head> para crear archivo head_css.html -->
18  {%
19    include "head_css.html"
20  %}
21  </head>
22  <body>
23  <!--
24  EXTRAJIMOS etiqueta <nav>, <jumbotron> y <javascript>para crear archivo navbar.html -->
25  {%
26    include "navbar.html"
27  }
28  <!-- definicion de block jumbotron -->
29  {%
30    block jumbotron %
31  }
32  <!-- definicion de block formulario que sera insertado -->
33  {%
34    block content %
35  }
36  <!--
37  EXTRAJIMOS etiqueta <head> para crear archivo javascript.html -->
38  {%
39    include "javascript.html"
40  %}
41  </body>
42  </html>

```

El archivo **inicio.html** contendrá.

#### Archivo **inicio.html**:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->

```

```

{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}
<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}<br/>
{% include "jumbotron.html" %}
{% endblock %}

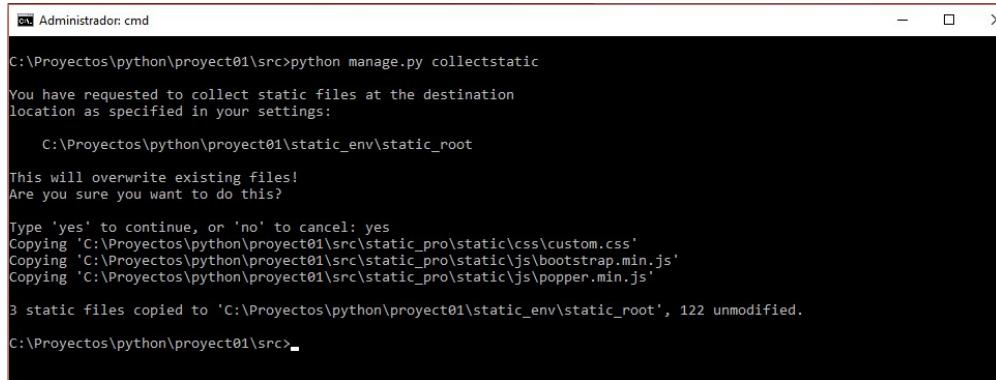
{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}
<div class="row">
<!--
  la clase col-sm-3 es de crispy-form numero de columnas 3
  la clase col-xs-12 es de crispy-form ajusta al comprimir la página
  la clase pull-right es de crispy-form alinea a la derecha
-->
<div class="col-sm-3 col-xs-12 text-left">
  <p class="lead text-center">{{ titulo }}</p>
  <form method="POST" action="">{{ csrf_token }}<br/>
  <!-- Cambiamos el form a crispy para mejor presentacion
  {{ el_form.form }}<br/>
  {{ el_form|crispy }}<br/>
  <input class="btn btn-primary" type="submit" value="Inscribirse">
</form>
</div>
<div class="col-sm-3">
  <p class="lead text-center">Pagina creado con framework Django.</p>
</div>
<div class="col-sm-3">
  <p class="lead text-center">Diseñado especialmente.</p>
</div>
<div class="col-sm-3">
  <p class="lead text-center">Con codigo abierto.</p>
</div>
</div>
{% endblock %}

```

Ejecutamos a nivel de comandos en la ruta **src**, para actualizar los archivos en las réplicas del servidor de producción.

D:\Proyectos\python\proyect01\src>**python manage.py collectstatic**



```
C:\Proyectos\python\proyecto01\src>python manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:
C:\Proyectos\python\proyecto01\static_env\static_root

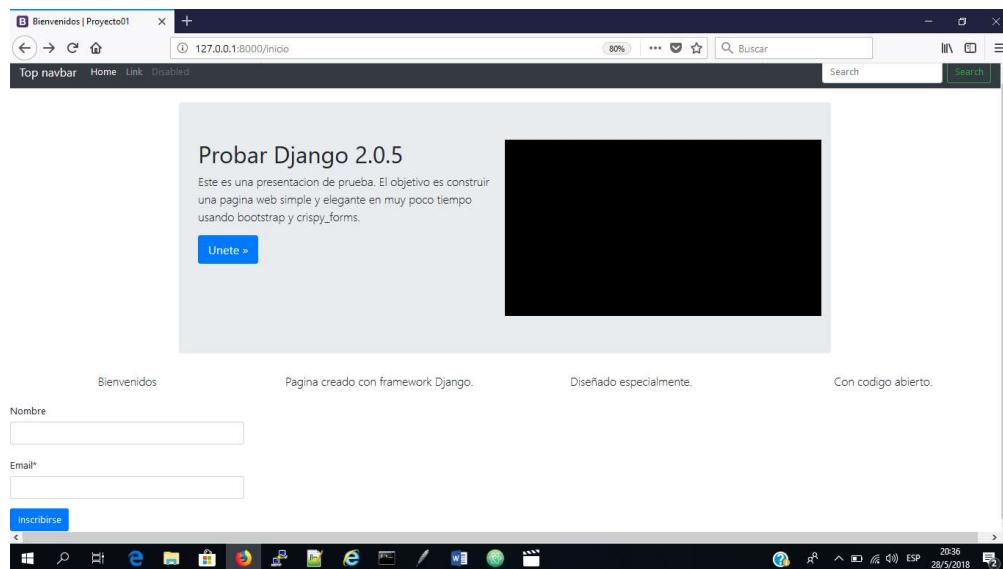
This will overwrite existing files!
Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\css\custom.css'
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\js\bootstrap.min.js'
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\js\popper.min.js'

3 static files copied to 'C:\Proyectos\python\proyecto01\static_env\static_root', 122 unmodified.

C:\Proyectos\python\proyecto01\src>
```

Se mostrará en el navegador como sigue:



## 28. Enlaces con Nombres URL

- Se refiere a los enlaces relativos y rutas hacia las páginas que se muestran en la barra de navegación desde donde haremos clic.
- Para esto en el template **navbar.html**, verificamos que contenga el código para el acceso hacia las páginas **/Contact** para **contacto.html** y **/About** para **about.html**. Deberá quedar más o menos como sigue ya que luego le haremos otras modificaciones.

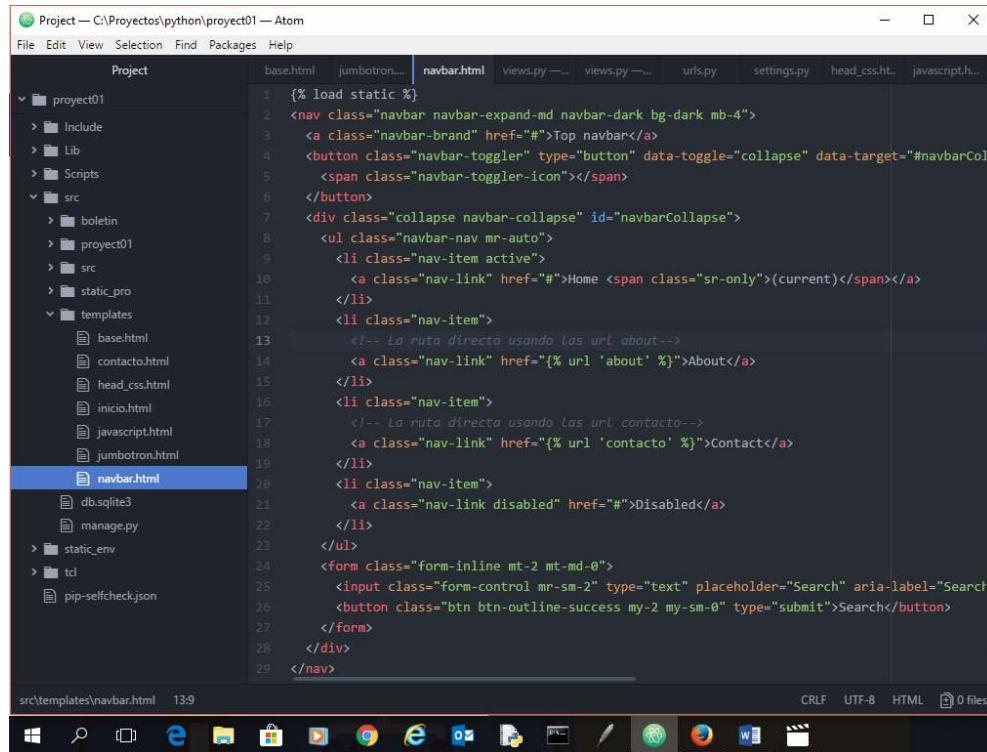
Archivo **navbar.html**:

```
{% load static %}
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
```

```

<div class="collapse navbar-collapse" id="navbarCollapse">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
      <a class="nav-link" href="#">#>Home <span class="sr-only">(current)</span></a>
    </li>
    <li class="nav-item">
      <!-- La ruta directa usando las url about-->
      <a class="nav-link" href="#">{ % url 'about' % } >About</a>
    </li>
    <li class="nav-item">
      <!-- La ruta directa usando las url contacto-->
      <a class="nav-link" href="#">{ % url 'contacto' % } >Contact</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" href="#">#>Disabled</a>
    </li>
  </ul>
  <form class="form-inline mt-2 mt-md-0">
    <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search" style="width: 150px; margin-right: 10px;" />
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</div>
</nav>

```



The screenshot shows the Atom code editor interface. The title bar reads "Project — C:\Proyectos\python\proyecto01 — Atom". The left sidebar displays a project structure with folders like "project01", "Include", "Lib", "Scripts", "src" (containing "boletin", "project01", "src", "static\_pro"), and "templates" (containing "base.html", "contacto.html", "head\_css.html", "inicio.html", "javascript.html", "jumbotron.html", "navbar.html"). The right pane shows the content of the "navbar.html" file, which contains the provided HTML code. The status bar at the bottom indicates "src\templates\navbar.html 13:9" and "CRLF UTF-8 HTML 0 files". The taskbar at the bottom includes icons for various applications like File Explorer, Task View, and Edge.

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project
  - project01
    - Include
    - Lib
    - Scripts
    - src
      - boletin
      - project01
      - src
      - static_pro
    - templates
      - base.html
      - contacto.html
      - head_css.html
      - inicio.html
      - javascript.html
      - jumbotron.html
      - navbar.html
    - db.sqlite3
    - manage.py
  - static_env
  - tcl
  - pip-selfcheck.json
src\templates\navbar.html 13:9
  1  {% load static %}
  2  <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  3    <a class="navbar-brand" href="#">#>Top navbar</a>
  4    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCol-
  5      <span class="navbar-toggler-icon"></span>
  6    </button>
  7    <div class="collapse navbar-collapse" id="navbarCollapse">
  8      <ul class="navbar-nav mr-auto">
  9        <li class="nav-item active">
 10          <a class="nav-link" href="#">{ % url 'about' % } >About</a>
 11        </li>
 12        <li class="nav-item">
 13          <!-- La ruta directa usando las url contacto-->
 14          <a class="nav-link" href="#">{ % url 'contacto' % } >Contact</a>
 15        </li>
 16        <li class="nav-item">
 17          <a class="nav-link disabled" href="#">#>Disabled</a>
 18        </li>
 19      </ul>
 20      <form class="form-inline mt-2 mt-md-0">
 21        <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search" style="width: 150px; margin-right: 10px;" />
 22        <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
 23      </form>
 24    </div>
 25  </nav>

```

### NOTA: FORMA 1 (Opcional):

Luego creamos las **views** para el archivo también usando las vistas fuera, es decir directamente sobre el proyecto de Django **\src\proyect01**, con el mismo nombre **views.py** que tenemos en la carpeta **\src\boletin\** aprovechando que se puede cargar desde ahí debido a que existe el archivo **\_\_init\_\_.py** de Python.

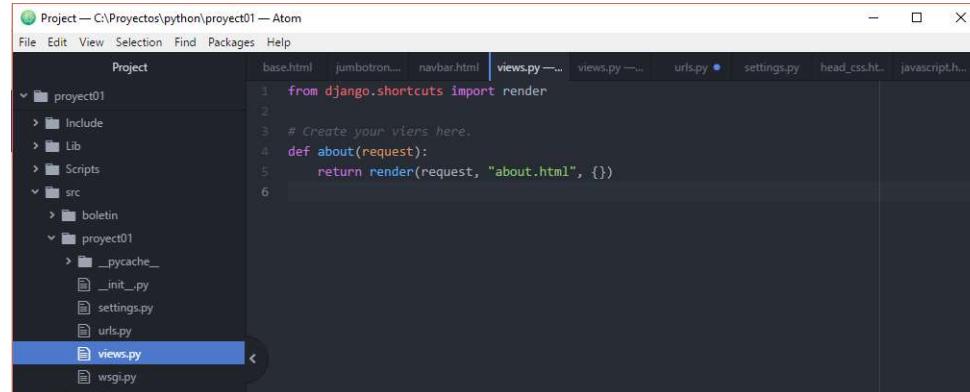
**NOTA:** luego probaremos si es posible definirlo dentro de la carpeta original del proyecto **\src\boletin**.

#### Archivo **\src\proyect01\views.py**

Archivo **views.py**:

```
from django.shortcuts import render
```

```
# Create your views here.
def about(request):
    return render(request, "about.html", {})
```



Abrimos **urls.py** y agregamos la **importación** relativa, más la ruta del archivo **about.html** que tendremos que crear posteriormente.

Archivo **urls.py**:

```
#from django.conf.urls import url
from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings
from django.conf.urls.static import static

from boletin import views
#from boletin.views inicio
# Creando la ruta relativa para archivo about.html
from .views import about

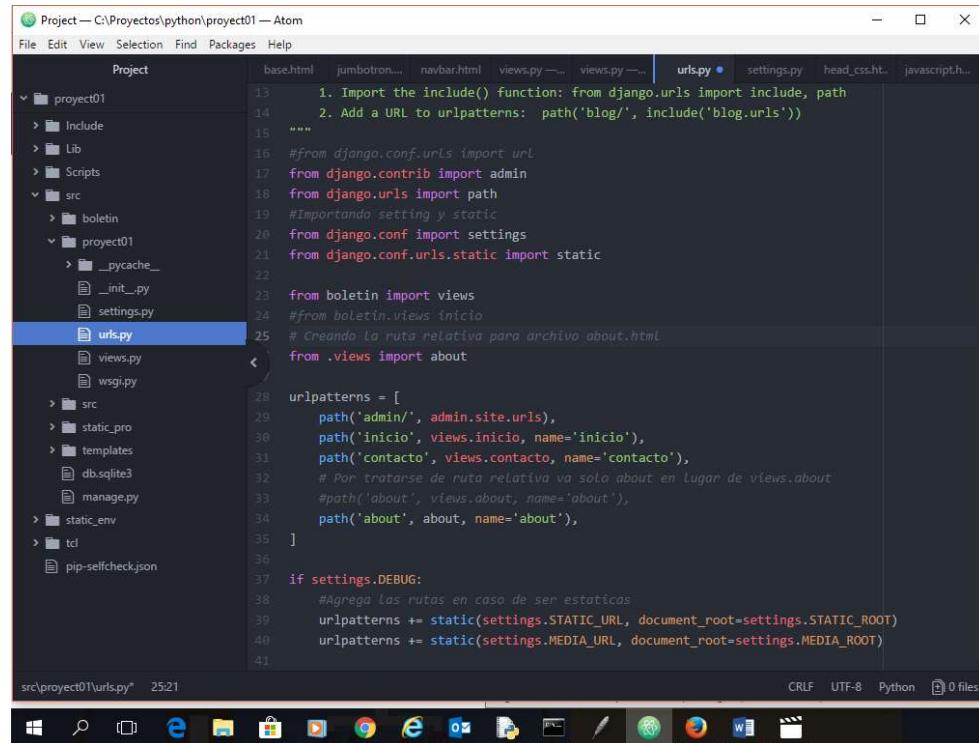
urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('about', views.about, name='about')
]
```

```

path('contacto', views.contacto, name='contacto'),
# Por tratarse de ruta relativa va solo about en lugar de views.about
#path('about', views.about, name='about'),
path('about', about, name='about'),
]

if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```



### NOTA: FORMA 2 (común):

La forma tradicional es no creando el archivo en `\src\proyecto01\views.py` y agregando el contenido del mismo al final del archivo inicial `\src\boletin\views.py`. Además, también crear en el archivo `urls.py` la ruta normal como se venía realizando, sin la importación de la vista personalizada, es decir.

#### Archivo `urls.py`:

```

#from django.conf.urls import url
from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings
from django.conf.urls.static import static

```

```

from boletin import views
#from boletin.views inicio
# Creando la ruta relativa para archivo about.html
#from .views import about

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
    # Por tratarse de ruta relativa va solo about en lugar de views.about
    #path('about', about, name='about'),
    path('about', views.about, name='about'),
]
if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

El archivo **\src\boletin\views.py**, quedaría normalmente así.

#### Archivo **views.py**:

```

#importacion de configuracion para correo electronico
from django.conf import settings
from django.core.mail import send_mail
#importamos renderizar
from django.shortcuts import render

#importamos el formulario
#from .forms import RegForm, RegModelForm
from .forms import RegForm, RegModelForm, ContactForm

#importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Bienvenidos"
    #Request que extraera variables del sistem
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }

```

```

        }

if form.is_valid():
    instance = form.save(commit=False)
    nombre = form.cleaned_data.get("nombre")
    email = form.cleaned_data.get("email")
    # En caso de la descripcion de persona no se llene por formulario
    if not instance.nombre:
        instance.nombre = "Persona nueva"
    #Grabar instancia
    instance.save()
    #Envio de nuevo contexto con nombre
    context = {
        "titulo": "Gracias %s!" %(nombre)
    }
    #Envio de nuevo contexto con email, el nombre de persona esta vacio
    if not nombre:
        context = {
            "titulo": "Gracias %s" %(email)
        }
    # form_data = form.cleaned_data
    # var1 = form_data.get("email")
    # var2 = form_data.get("nombre")
    # obj = Registrado.objects.create(email=var1, nombre=var2)

    #return render(request, "inicio.html", context)
    return render(request, "inicio.html", context)

def contacto(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        form_nombre = form.cleaned_data.get("nombre")
        form_email = form.cleaned_data.get("email")
        form_mensaje = form.cleaned_data.get("mensaje")
        asunto = "Form de Contacto"
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "elusuario@gmail.com"]
        email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje,
        form_email)
        send_mail(asunto,
                  email_mensaje,
                  email_from,
                  email_to,
                  fail_silently=False,
                  )
    #Forma 3 de llenar
    #for key, value in form.cleaned_data.items():
    #    print(key, value)

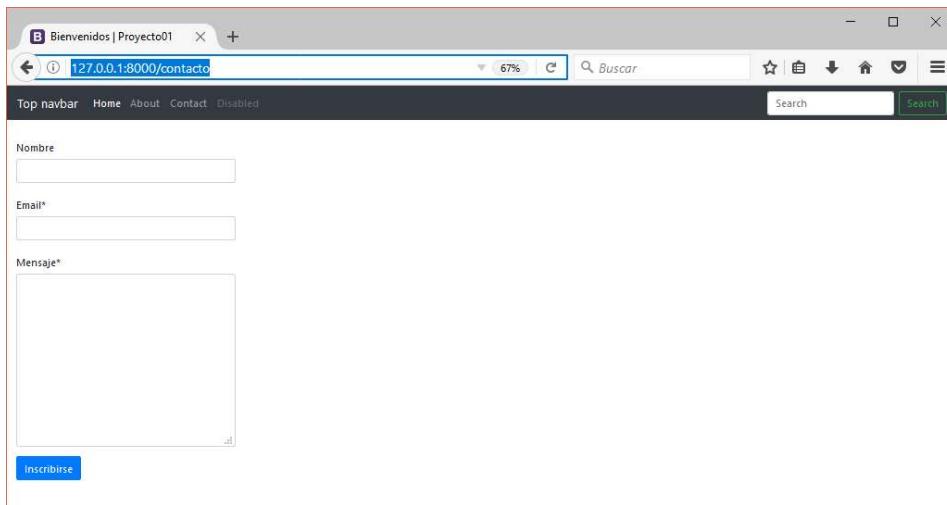
```

```
#Forma 2 de llenar
#for key in form.cleaned_data:
#    print(key)
#    print(form.cleaned_data.get(key))
#Forma 1 de llenar
#nombre = form.cleaned_data.get("nombre")
#email = form.cleaned_data.get("email")
#mensaje = form.cleaned_data.get("mensaje")
#print(nombre, email, mensaje)

context = {
    "contacto":form,
}
return render(request, "contacto.html", context)

def about(request):
    return render(request, "about.html", {})
```

Finalmente, cuando vamos a la barra de navegación podremos visualizar, por ejemplo **Contact**, el formulario de **contacto**.



## 29. Estilo Bootstrap 2

- Primero modificamos el archivo **views.py** en la función contacto y añadimos el contexto denominado “**Contacto**” más el parámetro a la forma contacto.

### Archivo **views.py**:

```
#Importacion de configuracion para correo electronico
from django.conf import settings
from django.core.mail import send_mail
#importamos renderizar
from django.shortcuts import render

#Importamos el formulario
```

```

#from .forms import RegForm, RegModelForm
from .forms import RegForm, RegModelForm, ContactForm

#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Bienvenidos"
    #Request que extraera variables del sistem
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }

    if form.is_valid():
        instance = form.save(commit=False)
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        # En caso de la descripcion de persona no se llene por formulario
        if not instance.nombre:
            instance.nombre = "Persona nueva"
        #Grabar instancia
        instance.save()
        #Envio de nuevo contexto con nombre
        context = {
            "titulo": "Gracias %s!" %(nombre)
        }
        #Envio de nuevo contexto con email, el nombre de persona esta vacio
        if not nombre:
            context = {
                "titulo": "Gracias %s" %(email)
            }
        # form_data = form.cleaned_data
        # var1 = form_data.get("email")
        # var2 = form_data.get("nombre")
        # obj = Registrado.objects.create(email=var1, nombre=var2)

        #return render(request, "inicio.html", context)
        return render(request, "inicio.html", context)

def contacto(request):

```

```

#Contexto
titulo = "Contacto"

form = ContactForm(request.POST or None)
if form.is_valid():
    form_nombre = form.cleaned_data.get("nombre")
    form_email = form.cleaned_data.get("email")
    form_mensaje = form.cleaned_data.get("mensaje")
    asunto = "Form de Contacto"
    email_from = settings.EMAIL_HOST_USER
    email_to = [email_from, "elusuario@gmail.com"]
    email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje,
form_email)
    send_mail(asunto,
              email_mensaje,
              email_from,
              email_to,
              fail_silently=False,
              )
#Forma 3 de llenar
#for key, value in form.cleaned_data.items():
#    print(key, value)
#Forma 2 de llenar
#for key in form.cleaned_data:
#    print(key)
#    print(form.cleaned_data.get(key))
#Forma 1 de llenar
#nombre = form.cleaned_data.get("nombre")
#email = form.cleaned_data.get("email")
#mensaje = form.cleaned_data.get("mensaje")
#print(nombre, email, mensaje)

context = {
    "titulo": titulo,
    "contacto": form,
}
return render(request, "contacto.html", context)

def about(request):
    return render(request, "about.html", {})

```

- Arreglando la presentación de la página de **contacto.html**, podemos hacer los siguientes cambios.

#### Archivo **contacto.html**:

```

<!DOCTYPE html>
<!-- Es una extension de base.html --&gt;
{% extends "base.html" %}
</pre>

```

```

<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}

{{ titulo }}

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block content %}


<!-- Pasamos titulo como contexto de view contacto -->
<h1 class="text-center">{{ titulo }}</h1>


<form method="POST" action="">{{ csrf_token }}
<!-- Cambiamos el form a crispy para mejor presentacion
{{ contacto.form }}-->
{{ contacto|crispy }}
<input class="btn btn-primary" type="submit" value="Inscribirse">
</form>


{% endblock %}

```

Se insertará un nuevo block de código para ser leído para Registro y Contact, About  
Archivo **base.html**:

```

<!doctype html>
{% load static %}
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<link rel="icon" href="{% static '/img/favicon.ico' %}">

<title>{{ block titulo_sitio }}Proyecto01{% endblock %}</title>
<!--
EXTRAJIMOS etiqueta <head> para crear archivo head_css.html
-->
{% include "head_css.html" %}
</head>

<body>

```

```

<!--
EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html
-->
{% include "navbar.html" %}

<!-- definicion de block jumbotron -->
{% block jumbotron %}
{% endblock %}

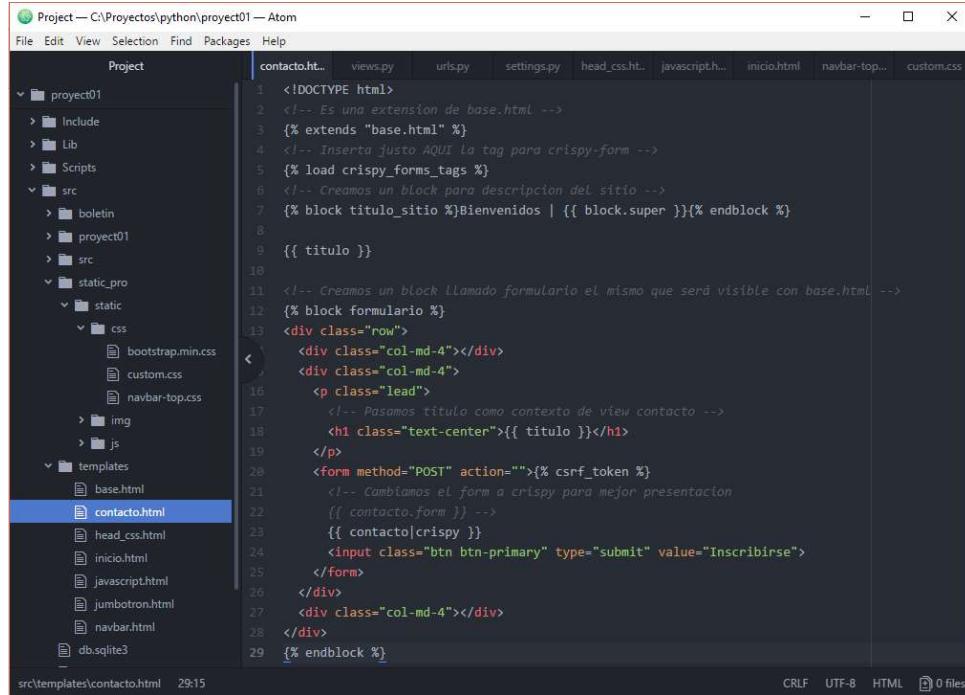
<!-- definicion de block formulario que sera insertado -->
{% block formulario %}
{% endblock %}

<!-- definicion de block content para que sera insertado -->
{% block content %}
{% endblock %}

<!--
EXTRAJIMOS etiqueta <head> para crear archivo javascript.html
-->
{% include "javascript.html" %}

</body>
</html>

```



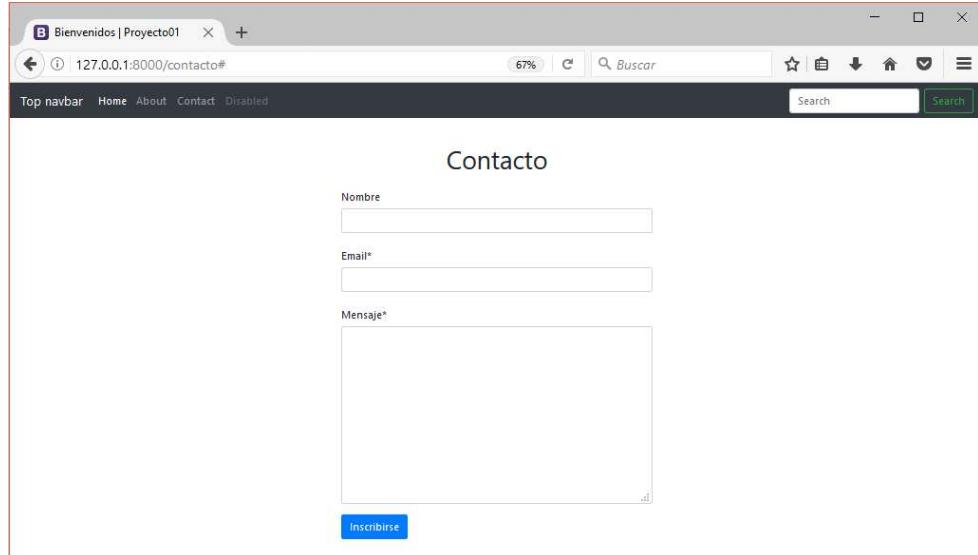
The screenshot shows the Atom code editor interface. The title bar reads "Project — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The left sidebar shows a project structure with a "project01" folder containing "Include", "Lib", "Scripts", "src" (which contains "boletin", "project01", "src", "static\_pro" (containing "static" with "css" subfolder), and "templates" (containing "base.html", "contacto.html", "head\_css.html", "inicio.html", "javascript.html", "jumbotron.html", "navbar.html", and "db.sqlite3"). The right pane displays the content of "contacto.html". The code is as follows:

```

1 <!DOCTYPE html>
2 <!-- Es una extension de base.html -->
3 {% extends "base.html" %}
4 <!-- Inserta justo AQUI la tag para crispy-form -->
5 {% load crispy_forms_tags %}
6 <!-- Creamos un block para descripcion del sitio -->
7 {% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}
8
9 {{ titulo }}
10
11 <!-- Creamos un block llamado formulario el mismo que sera visible con base.html -->
12 {% block formulario %}
13 <div class="row">
14   <div class="col-md-4"></div>
15   <div class="col-md-4">
16     <p class="lead">
17       <!-- Pasamos titulo como contexto de view contacto -->
18       <h1 class="text-center">{{ titulo }}</h1>
19     </p>
20     <form method="POST" action="">{{ csrf_token }}
21       <!-- Cambiamos el form a crispy para mejor presentacion
22       de contacto.form -->
23       {{ contacto|crispy }}
24       <input class="btn btn-primary" type="submit" value="Inscribirse">
25     </form>
26   </div>
27   <div class="col-md-4"></div>
28 </div>
29 {% endblock %}

```

The status bar at the bottom indicates "src\templates\contacto.html 29:15" and encoding "CRLF UTF-8 HTML". There are also icons for file count and file status.



Bienvenidos | Proyecto01

127.0.0.1:8000/contacto#

Top navbar Home About Contact: Disabled

Search

Contacto

Nombre

Email\*

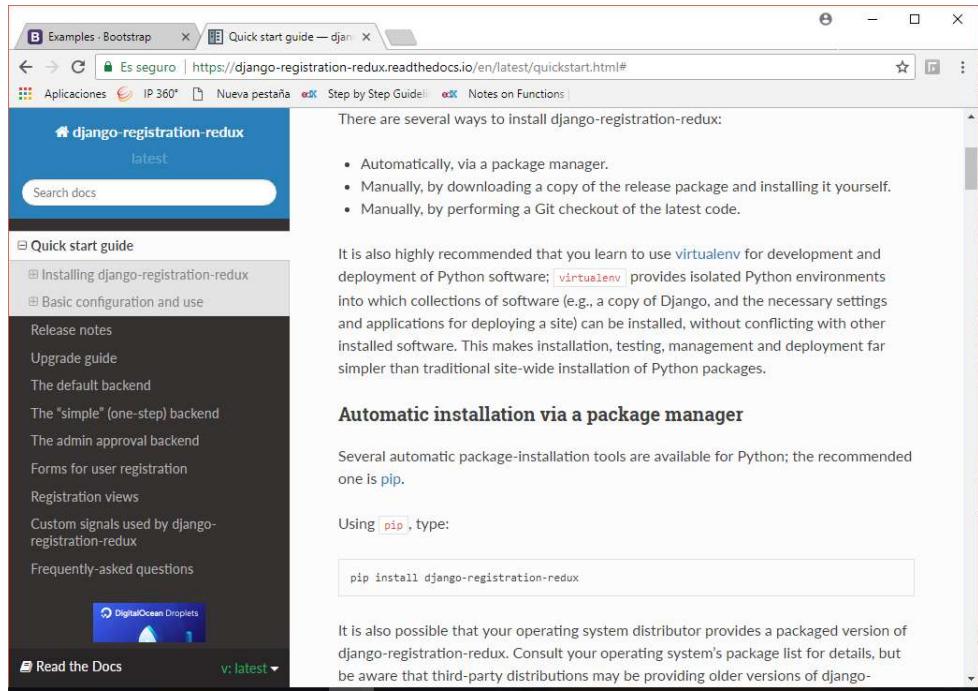
Mensaje\*

Inscribirse

## 30. Django Registration Redux 1

- Es un paquete de Django que permite añadir características para los formularios de acceso para registrarse o autenticarse dentro de la aplicación web.

<https://django-registration-redux.readthedocs.io/en/latest/quickstart.html>



There are several ways to install django-registration-redux:

- Automatically, via a package manager.
- Manually, by downloading a copy of the release package and installing it yourself.
- Manually, by performing a Git checkout of the latest code.

It is also highly recommended that you learn to use `virtualenv` for development and deployment of Python software; `virtualenv` provides isolated Python environments into which collections of software (e.g., a copy of Django, and the necessary settings and applications for deploying a site) can be installed, without conflicting with other installed software. This makes installation, testing, management and deployment far simpler than traditional site-wide installation of Python packages.

**Automatic installation via a package manager**

Several automatic package-installation tools are available for Python; the recommended one is `pip`.

Using `pip`, type:

```
pip install django-registration-redux
```

It is also possible that your operating system distributor provides a packaged version of django-registration-redux. Consult your operating system's package list for details, but be aware that third-party distributions may be providing older versions of django-

Comprobamos que el paquete esté instalado mediante freeze.

D:\Proyectos\python\proyecto01\src>**pip freeze**

```
C:\ Proyectos\python\proyecto01\src>pip freeze
Django==2.0.5
django-crispy-forms==1.7.2
numpy==1.14.3
pytz==2018.4
virtualenv==15.2.0

C:\ Proyectos\python\proyecto01\src>
```

Luego procedemos a la instalación manual (**OJO con RUTA**)

```
D:\Proyectos\python\proyecto01>pip install django-registration-redux
```

```
C:\ Proyectos\python\proyecto01\src>pip install django-registration-redux
Collecting django-registration-redux
  Downloading https://files.pythonhosted.org/packages/0a/64/2fd365f65c23e180306f3cc5b01f4a/django_registration_redux-2.4-py3-none-any.whl (202kB)
    100% |██████████| 204kB 486kB/s
Installing collected packages: django-registration-redux
Successfully installed django-registration-redux-2.4

C:\ Proyectos\python\proyecto01\src>_
```

Vamos al archivo **settings.py** para agregar las app '**registration**', el mismo que debe colocarse antes de '**django.contrib.admin**' y después de '**django.contrib.sites**'

Archivo **settings.py**:

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'xsqdan0p7dk1ahq85x23er50pl#acd#%eqnz+arr!*ake(413_'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

#Configuracion de servidor para envio de correo electronico
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'elusuario@gmail.com'
EMAIL_HOST_PASSWORD = 'Sucontraseña'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

```

#...
# Para usar gmail hay de desbloquear captcha
# https://accounts.google.com/displayunlockcaptcha
#...

# Application definition

INSTALLED_APPS = [
    # apps de terceros Redux registration
    'registration',
    # apps Django
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    # apps de terceros Redux registration
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # apps de terceros crispy
    'crispy_forms',
    # apps creadas
    'boletin',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'proyect01.urls'
# Crispy
CRISPY_TEMPLATE_PACK = 'bootstrap4'
# Redux registration
ACCOUNT_ACTIVATION_DAYS = 7
REGISTRATION_AUTO_LOGIN = True

TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    #     'DIRS': [],
    'DIRS': [os.path.join(BASE_DIR, "templates")],
}
]

```

```
'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
],
}

WSGI_APPLICATION = 'proyect01.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-passwordValidators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
            'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/

#LANGUAGE_CODE = 'en-us'
LANGUAGE_CODE = 'es'
```

```

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'
MEDIA_URL = '/media/'
# /static/imagenes/img1.jpg

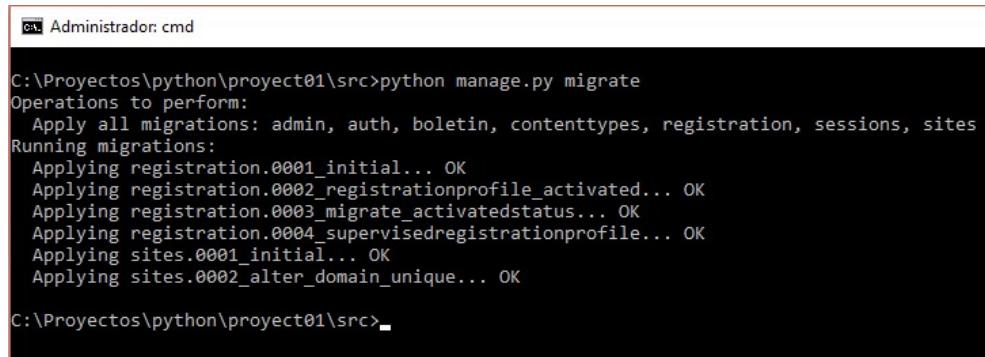
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static_pro", "static"),
    # '/var/www/static/,
]

STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "static_root")
MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "media_root")

```

Después vamos por comandos a ingresar para ejecutar lo siguiente:

D:\Proyectos\python\proyect01\src>**python manage.py migrate**



```

C:\Proyectos\python\proyect01\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, registration, sessions, sites
Running migrations:
  Applying registration.0001_initial... OK
  Applying registration.0002_registrationprofile_activated... OK
  Applying registration.0003_migrate_activatedstatus... OK
  Applying registration.0004_supervisedregistrationprofile... OK
  Applying sites.0001_initial... OK
  Applying sites.0002_alter_domain_unique... OK
C:\Proyectos\python\proyect01\src>

```

Ahora configuramos las **urls.py** agregando código para el **include** mas las rutas será entonces lo siguiente:

Archivo **urls.py**:

```

# Redux registration
from django.conf.urls import url, include
from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings

```

```

from django.conf.urls.static import static

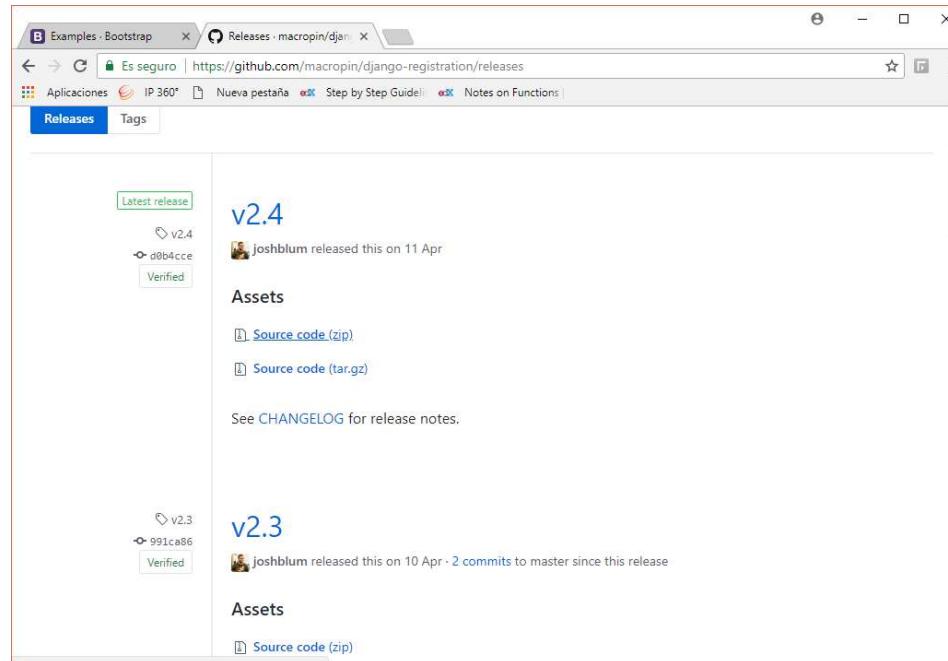
from boletin import views
#from boletin.views inicio
# Creando la ruta relativa para archivo about.html
#from .views import about

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
    # Por tratarse de ruta relativa va solo about en lugar de views.about
    #path('about', about, name='about'),
    path('about', views.about, name='about'),
    # Redux registration
    path('accounts', include('registration.backends.default.urls')),
]
if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

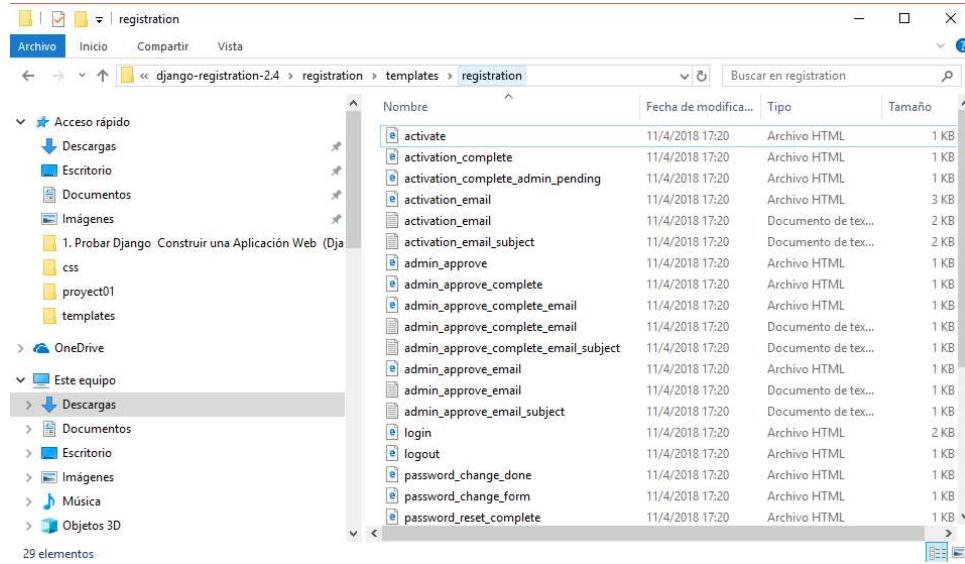
```

Procedemos a la descarga de las Templates dentro del repositorio github versión 2.4:

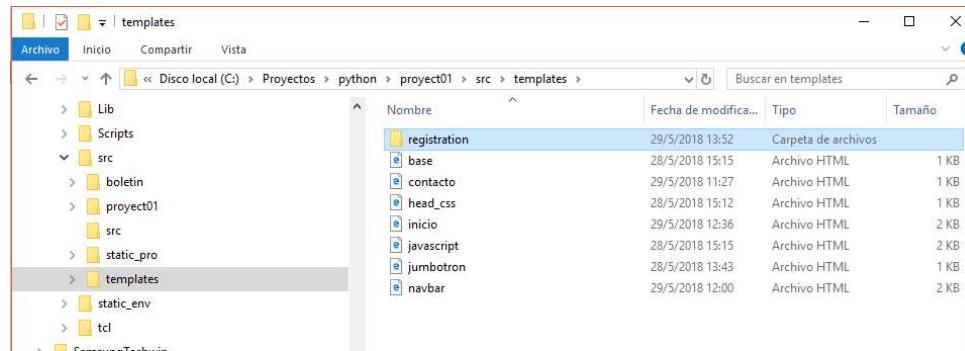
<https://github.com/macropin/django-registration/releases>



Una vez descargados, se descomprimen los archivos y se copia la carpeta **registration** debajo de **\src\templates\** con su mismo nombre.



Copiar los archivos en la ruta siguiente:



Habrá que modificar el archivo que viene incluido en la carpeta **\src\template\registration** llamado **registration\_form.html** al cual le agregaremos **crispy\_forms** para darle formato, de la siguiente manera:

Archivo **registration\_form.html**:

```

{% extends "registration/registration_base.html" %}
{% load i18n %}


{% load crispy_forms_tags %}

{% block title %}{% trans "Register for an account" %}{% endblock %}

{{ titulo }}

{% block content %}
<div class="row">
<div class="col-md-4"></div>
<div class="col-md-4">
```

```

<p class="lead">
    <h1 class="text-center">Registrese ahora </h1>
</p>
<form method="POST" action="">{% csrf_token %}
    <!-- Cambiamos el form a crispy para mejor presentacion
    {{ el_form.form }} -->
    {{ form|crispy }}
    <input class="btn btn-primary" type="submit" value="Inscribirse">
</form>
</div>
<div class="col-md-4"></div>
</div>
{% endblock %}

```

{% comment %}

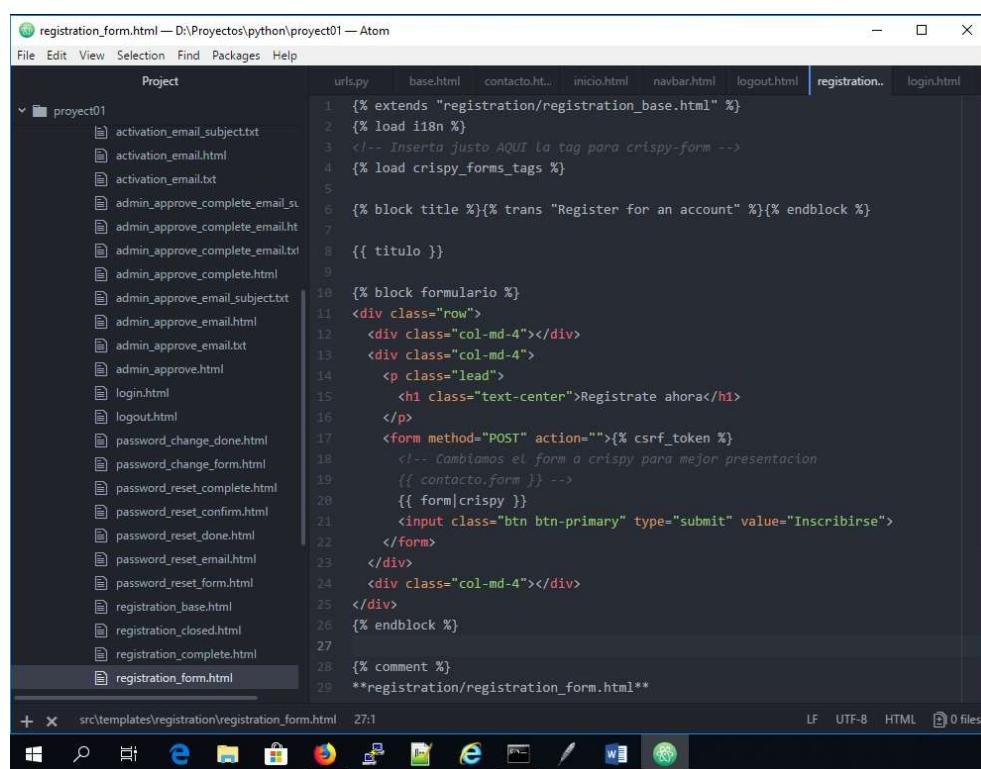
\*\*registration/registration\_form.html\*\*

Used to show the form users will fill out to register. By default, has the following context:

``form``

The registration form. This will be an instance of some subclass of ``django.forms.Form``; consult 'Django's forms documentation <<http://docs.djangoproject.com/en/dev/topics/forms/>>' for information on how to display this in a template.

{% endcomment %}



The screenshot shows the Atom code editor with the file 'registration\_form.html' open. The file contains Django template code for a registration form. The code includes HTML for a lead paragraph and a registration form, and a block comment section. The Atom interface shows a project structure on the left with files like activation\_email.\* and admin\_approve.\*. The status bar at the bottom indicates the file is 271 lines long and is saved in UTF-8 encoding.

```

registration_form.html — D:\Proyectos\python\project01 — Atom
File Edit View Selection Find Packages Help
Project
  project01
    activation_email_subject.txt
    activation_email.html
    activation_email.txt
    admin_approve_complete_email.srt
    admin_approve_complete_email.html
    admin_approve_complete_email.txt
    admin_approve_complete.html
    admin_approve_email_subject.txt
    admin_approve_email.html
    admin_approve_email.txt
    admin_approve.html
    login.html
    logout.html
    password_change_done.html
    password_change_form.html
    password_reset_complete.html
    password_reset_confirm.html
    password_reset_done.html
    password_reset_email.html
    password_reset_form.html
    registration_base.html
    registration_closed.html
    registration_complete.html
    registration_form.html
  urls.py base.html contacto.htm inicio.html navbar.html logout.html registration.. login.html
1  {% extends "registration/registration_base.html" %}
2  {% load i18n %}
3  <!-- Inserta justo AQUI la tag para crispy-form -->
4  {% load crispy_forms_tags %}
5
6  {% block title %}{% trans "Register for an account" %}{% endblock %}
7
8  {{ titulo }}
9
10  {% block formulario %}
11  <div class="row">
12  <div class="col-md-4"></div>
13  <div class="col-md-4">
14  <p class="lead">
15  <h1 class="text-center">Registrese ahora </h1>
16  </p>
17  <form method="POST" action="">{% csrf_token %}
18  <!-- Cambiamos el form a crispy para mejor presentacion
19  {{ contacto.form }} -->
20  {{ form|crispy }}
21  <input class="btn btn-primary" type="submit" value="Inscribirse">
22  </form>
23  </div>
24  <div class="col-md-4"></div>
25  </div>
26  {% endblock %}
27
28  {% comment %}
29  **registration/registration_form.html**

```

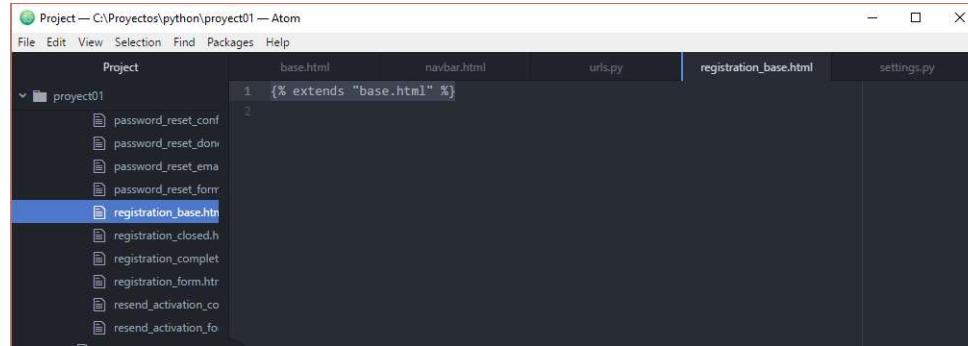
**IMPORTANTE:**

Es necesario tener la plantilla **base.html** antes de instalar **Registration Redux**, ya que serán llamadas como **extends** del mismo.

Confirmar que en archivo `\src\templates\registration\registration_base.html` de la carpeta **registration** contenga el nombre de la template “**base.html**” como sigue:

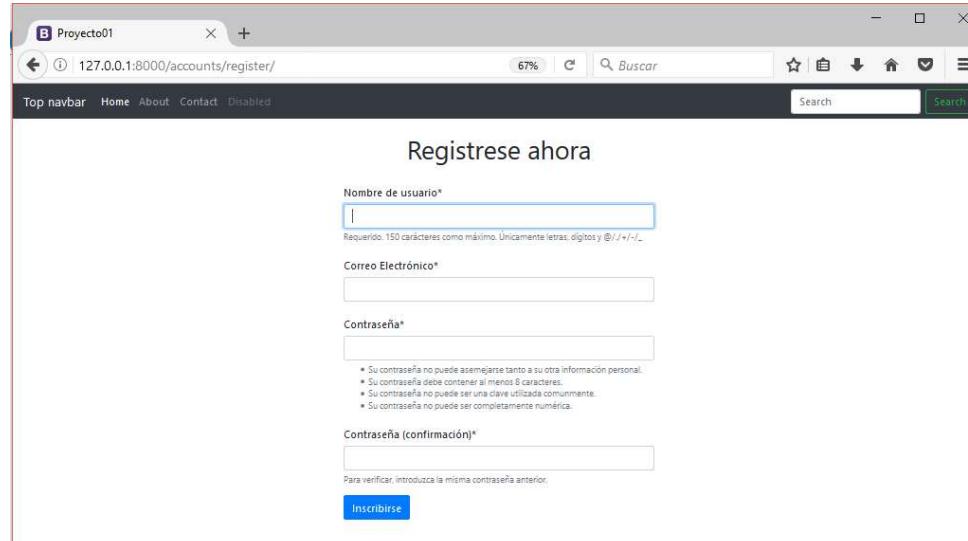
Archivo **registration\_base.html**:

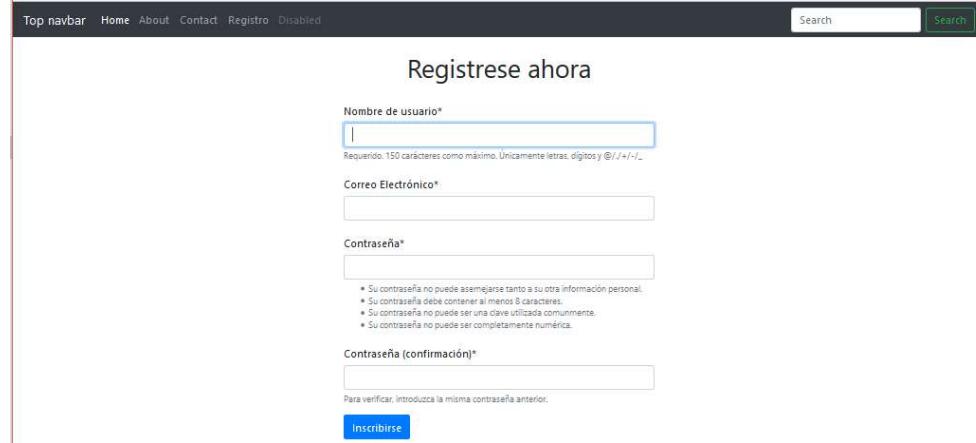
```
{% extends "base.html" %}
```



Cargamos el registro mediante el browser:

<http://127.0.0.1:8000/accounts/register>

A screenshot of a web browser window titled "Proyecto01". The address bar shows "127.0.0.1:8000/accounts/register/". The page header includes "Top navbar Home About Contact Disabled". The main content is a registration form with the title "Registrese ahora". It has four input fields: "Nombre de usuario\*" (with placeholder "Introduce tu nombre de usuario"), "Correo Electrónico\*" (with placeholder "Introduce tu correo electrónico"), "Contraseña\*" (with placeholder "Introduce una contraseña segura" and a note below it: "Su contraseña no puede asombrarse tanto a su otra información personal. Su contraseña debe contener al menos 6 caracteres. Su contraseña no puede ser una clave utilizada comúnmente. Su contraseña no puede ser completamente numérica."), and "Contraseña (confirmación)\*" (with placeholder "Introduce la misma contraseña anterior"). A "Inscribirse" button is at the bottom.



The screenshot shows a registration form titled "Registrese ahora". It includes fields for "Nombre de usuario\*", "Correo Electrónico\*", "Contraseña\*", and "Contraseña [confirmación]\*". Below the password fields are validation rules: "Su contraseña no puede asemejarse tanto a su otra información personal.", "Su contraseña debe contener al menos 8 caracteres.", "Su contraseña no puede ser una clave utilizada comúnmente.", and "Su contraseña no puede ser completamente numérica.". A note at the bottom says "Para verificar, introduzca la misma contraseña anterior." and a blue "Inscribirse" button.

## 31. Django Registration Redux 2

- Vamos a agregar a navbar links para login y logout para probar su funcionalidad, los mismos que también pueden ser ejecutados de la forma siguiente:

<http://127.0.0.1:8000/accounts/register/>  
<http://127.0.0.1:8000/accounts/login/>  
<http://127.0.0.1:8000/accounts/logout/>  
<http://127.0.0.1:8000/admin/>

**Registro** de usuarios nuevos  
**Login** de usuario  
**Logout** de usuario  
**Administracion** de web app Django

Archivo **navbar.html**:

```

{% load static %}

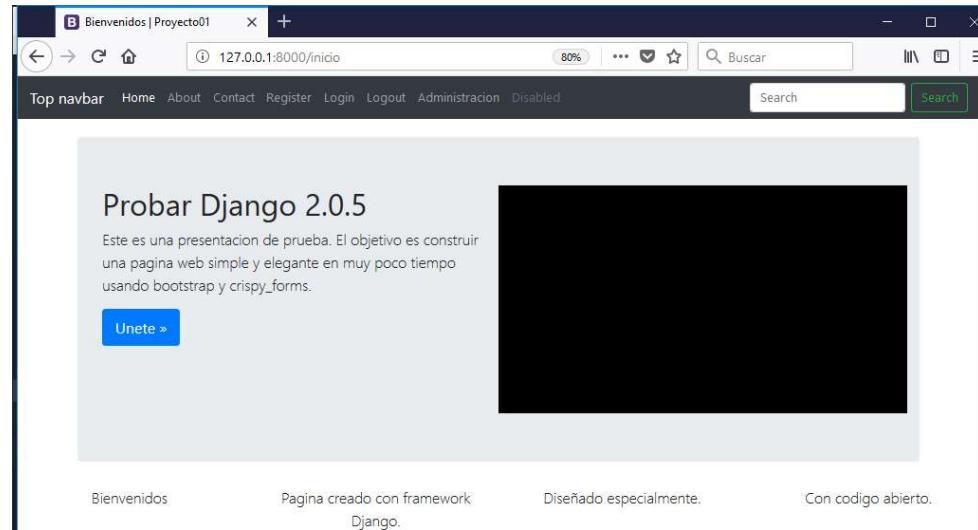
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="{% url 'inicio' %}">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <!-- La ruta directa usando las url about -->
        <a class="nav-link" href="{% url 'about' %}">About</a>
      </li>
      <li class="nav-item">
        <!-- La ruta directa usando las url contacto -->
        <a class="nav-link" href="{% url 'contacto' %}">Contact</a>
      </li>
      <li class="nav-item">
        <!-- La ruta directa usando las url register -->
      </li>
    </ul>
  </div>
</nav>

```

```

<a class="nav-link" href="/accounts/register">Register</a>
</li>
<li class="nav-item">
    <!-- La ruta directa usando las url Login -->
    <a class="nav-link" href="/accounts/login">Login</a>
</li>
<li class="nav-item">
    <!-- La ruta directa usando las url Logout -->
    <a class="nav-link" href="/accounts/logout">Logout</a>
</li>
<li class="nav-item">
    <!-- La ruta directa usando las url Admin -->
    <a class="nav-link" href="/admin">Administracion</a>
</li>
<li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
</li>
</ul>
<form class="form-inline mt-2 mt-md-0">
    <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-
label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
</form>
</div>
</nav>

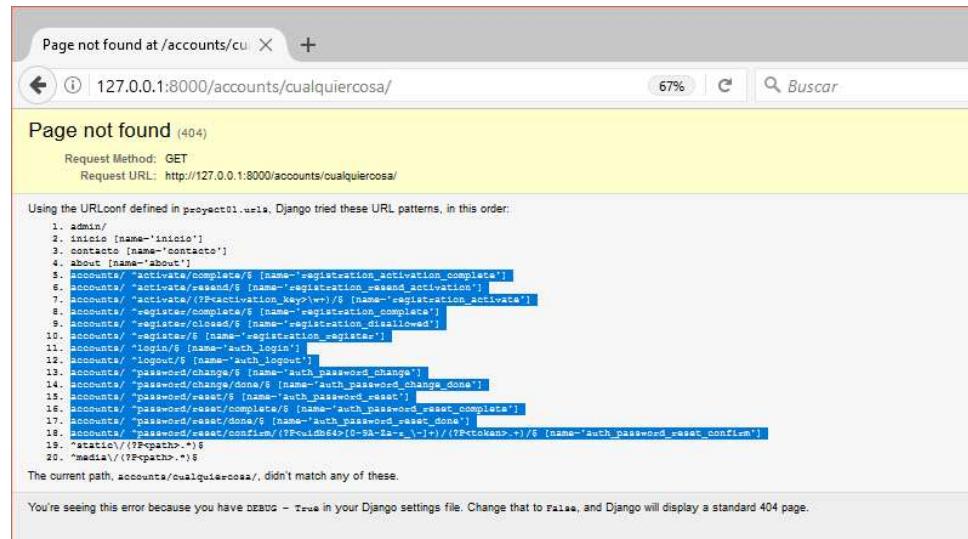
```



- Ahora bien, una vez montado **django-registration-redux**, notaremos que no existen link para el resto de apps instaladas.

Podremos comprobarlos simplemente escribiendo en el browser:

<http://127.0.0.1:8000/accounts/cualquiercosa/>



<http://127.0.0.1:8000/accounts/login/>

Para corregir este problema, editamos el archivo **settings.py**, añadiendo:

#### Archivo **settings.py**:

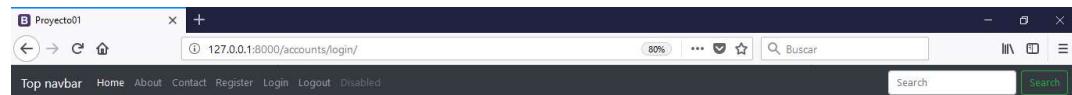
```

...
ROOT_URLCONF = 'proyecto01.urls'
# Crispy
CRISPY_TEMPLATE_PACK = 'bootstrap4'
# Redux registration
ACCOUNT_ACTIVATION_DAYS = 7
REGISTRATION_AUTO_LOGIN = True
# Activa el admin para la conexión
SITE_ID = 1
...

```

Notaremos que no aparece nada cuando intentamos ingresar al **login**, por lo tanto, tendremos que modificar dentro de **\templates\resgistration\login.html** y **logout.html** agregando código tal como se realizó con **registration\_form.html**.

<http://127.0.0.1:8000/accounts/login/>



#### Archivo **login.html**:

```
{% extends "registration/registration_base.html" %}  
{% load i18n %}  
<!-- Inserta justo AQUI la tag para crispy-form -->  
{% load crispy_forms_tags %}  
  
{% block title %}{% trans "Log in" %}{% endblock %}  
  
{% block content %}  
<div class="row">  
  <div class="col-md-4"></div>  
  <div class="col-md-4">  
    <p class="lead">  
      <h1 class="text-center">LOGIN</h1>  
    </p>  
    <form method="post" action="">  
      {% csrf_token %}  
      {{ form|crispy }}  
      <input type="submit" value="{% trans 'Log in' %}" />  
      <input type="hidden" name="next" value="{{ next }}" />  
    </form>  
    <p>{% trans "Forgot your password?" %} <a href="{% url 'auth_password_reset' %}">{% trans "Reset it" %}</a>.</p>  
    <p>{% trans "Not a member?" %} <a href="{% url 'registration_register' %}">{% trans "Register" %}</a>.</p>  
  </div>  
  <div class="col-md-4"></div>  
</div>  
{% endblock %}  
  
{% comment %}  
**registration/login.html**
```

*It's your responsibility to provide the login form in a template called registration/login.html by default. This template gets passed four template context variables:*

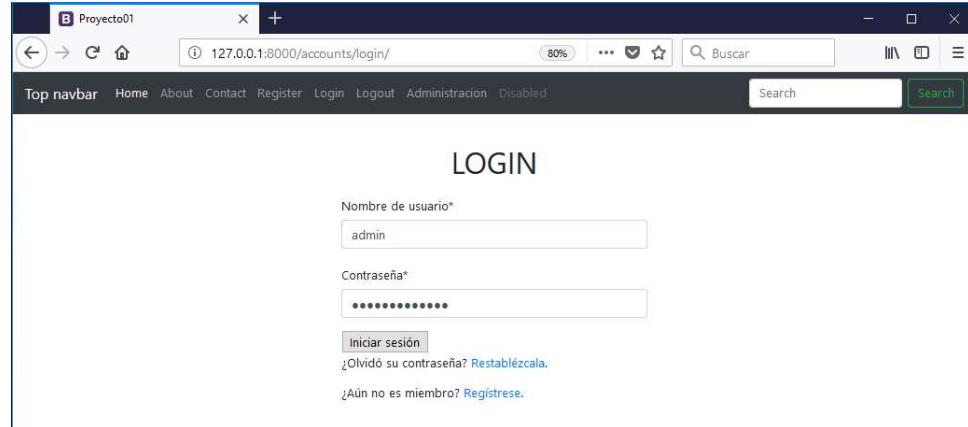
`form`  
A *Form* object representing the login form. See the forms documentation for more on Form objects.

**``next``**  
*The URL to redirect to after successful login. This may contain a query string, too.*

```site```  
*The current Site, according to the SITE\_ID setting. If you don't have the site framework installed, this will be set to an instance of RequestSite, which derives the site name and domain from the*

current *HttpRequest*.

```
``site_name``
An alias for site.name. If you don't have the site framework
installed, this will be set to the value of
request.META['SERVER_NAME']. For more on sites, see The
"sites" framework.
{%- endcomment %}
```



#### Archivo `logout.html`:

```
{% extends "registration/registration_base.html" %}
{% load i18n %}

<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

{% block title %}{% trans "Logged out" %}{% endblock %}

{% block content %}
<div class="row">
<div class="col-md-4"></div>
<div class="col-md-4">
<p class="lead">
<h1 class="text-center">LOGOUT</h1>
</p>
<p>{% trans "Successfully logged out" %}.</p>
</div>
<div class="col-md-4"></div>
</div>
{% endblock %}
```



### Archivo **password\_reset\_form.html**:

```

{% extends "registration/registration_base.html" %}
{% load i18n %}

<!-- Inserta justo AQUI la tag para crispy-form --&gt;
{% load crispy_forms_tags %}

{% block title %}{% trans "Reset password" %}{% endblock %}

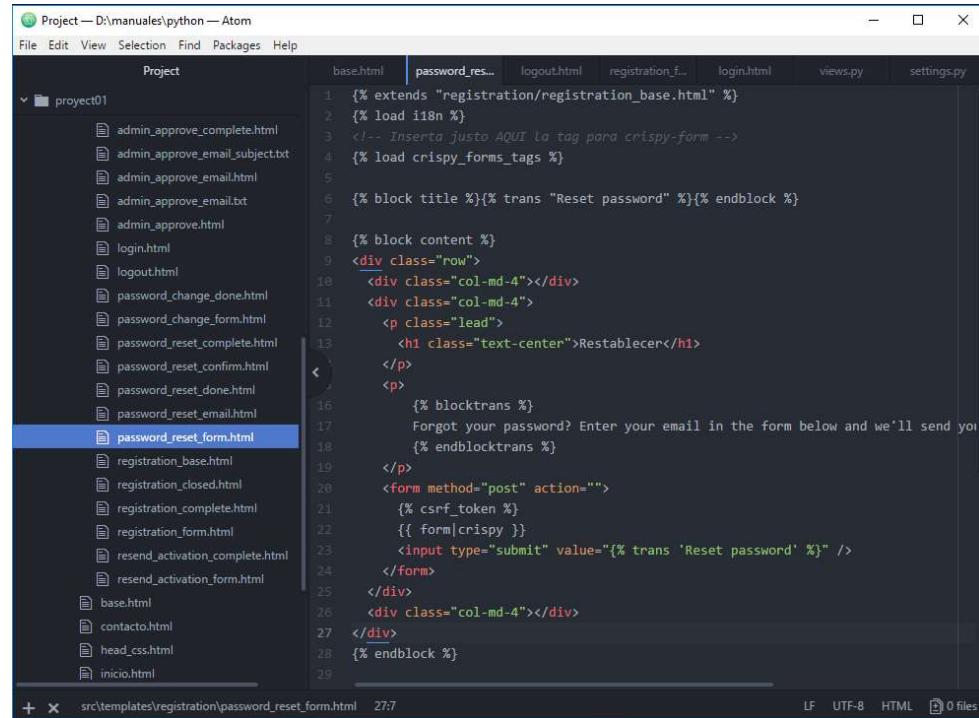
{% block content %}
&lt;div class="row"&gt;
  &lt;div class="col-md-4"&gt;&lt;/div&gt;
  &lt;div class="col-md-4"&gt;
    &lt;p class="lead"&gt;
      &lt;h1 class="text-center"&gt;Restablecer&lt;/h1&gt;
    &lt;/p&gt;
    &lt;p&gt;
      {% blocktrans %}
        <i>Forgot your password? Enter your email in the form below and we'll send you instructions for creating a new one.
      {% endblocktrans %}
    </p>
    <form method="post" action="">
      {% csrf_token %}
      {{ form|crispy }}
      <input type="submit" value="{{ trans 'Reset password' }}" />
    </form>
  </div>
  <div class="col-md-4"></div>
</div>
{% endblock %}

{# This is used by django.contrib.auth #}
<!--
ORIGINAL TEXT
<p>
  {% blocktrans %}
    Forgot your password? Enter your email in the form below and we'll send you instructions for creating a new one.
  {% endblocktrans %}
-->
```

```

</p>
<form method="post" action="">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="{% trans 'Reset password' %}" />
</form>
-->

```



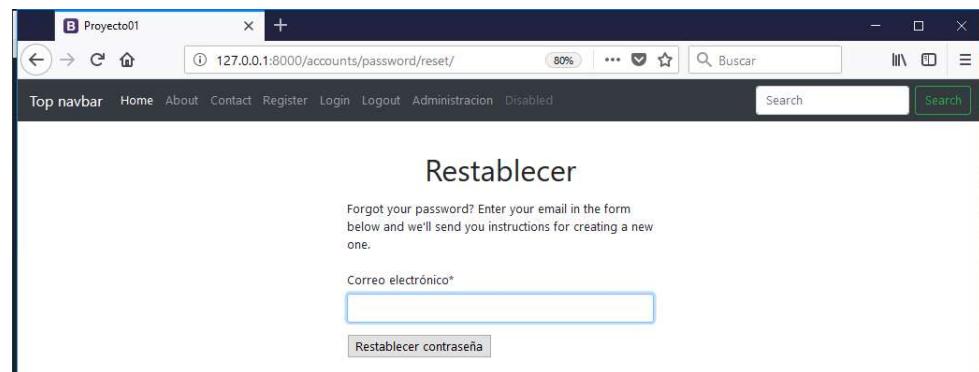
The screenshot shows the Atom code editor with the following details:

- Project:** D:\manuales\python — Atom
- File:** password\_res...
- Code View:** password\_reset\_form.html
- Content:**

```

1  {% extends "registration/registration_base.html" %}
2  {% load i18n %}
3  <!-- Inserta justo AQUÍ la tag para crispy-form -->
4  {% load crispy_forms_tags %}
5
6  {% block title %}{% trans "Reset password" %}{% endblock %}
7
8  {% block content %}
9  <div class="row">
10   <div class="col-md-4"></div>
11   <div class="col-md-4">
12     <p class="lead">
13       <h1 class="text-center">Restablecer</h1>
14     </p>
15     <p>
16       {% blocktrans %}
17         Forgot your password? Enter your email in the form below and we'll send you
18         an email with instructions.
19       {% endblocktrans %}
20     </p>
21     <form method="post" action="">
22       {% csrf_token %}
23       {{ form|crispy }}
24       <input type="submit" value="{% trans 'Reset password' %}" />
25     </form>
26   </div>
27 </div>
28 {% endblock %}

```
- Bottom Status Bar:** LF UTF-8 HTML 0 files



### 32. Cambiar URL Redirect Despu  s de Login

- Para que no vaya siempre a la pagina **127.0.1:8000/accounts/profile**, cada vez que cargamos **LOGIN**. Para corregir este problema, editamos el archivo **settings.py**, añadiendo:

Archivo **settings.py**:

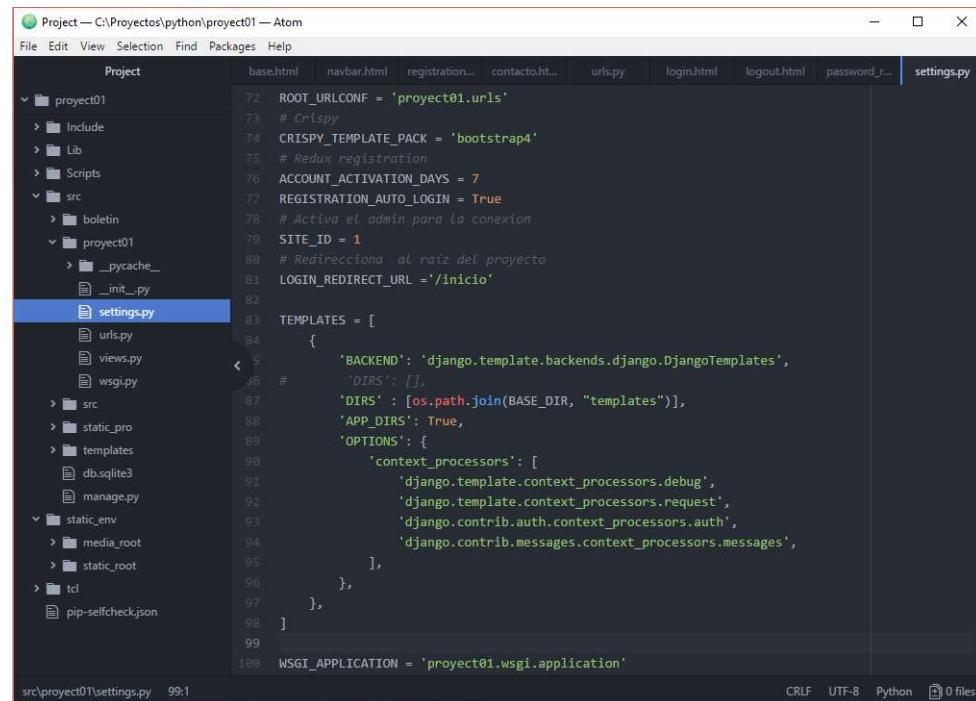
```

...
ROOT_URLCONF = 'proyect01.urls'

```

```
# Crispy
CRISPY_TEMPLATE_PACK = 'bootstrap4'
# Redux registration
ACCOUNT_ACTIVATION_DAYS = 7
REGISTRATION_AUTO_LOGIN = True
# Activa el admin para la conexión
SITE_ID = 1
# Redirecciona al raíz del proyecto
LOGIN_REDIRECT_URL = '/inicio'
...

```



The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project
  proyecto01
    Include
    Lib
    Scripts
    src
      boletin
      proyecto01
        __pycache__
        __init__.py
        settings.py
        urls.py
        views.py
        wsgi.py
      src
      static_pro
      templates
        db.sqlite3
        manage.py
    static_env
    media_root
    static_root
    tcl
    pip-selfcheck.json

```

The main editor window shows the `settings.py` file content:

```

72 ROOT_URLCONF = 'proyecto01.urls'
73 # Crispy
74 CRISPY_TEMPLATE_PACK = 'bootstrap4'
75 # Redux registration
76 ACCOUNT_ACTIVATION_DAYS = 7
77 REGISTRATION_AUTO_LOGIN = True
78 # Activa el admin para la conexión
79 SITE_ID = 1
80 # Redirecciona al raíz del proyecto
81 LOGIN_REDIRECT_URL = '/inicio'
82
83 TEMPLATES = [
84     {
85         'BACKEND': 'django.template.backends.django.DjangoTemplates',
86         'DIRS': [],
87         'DIRS' : [os.path.join(BASE_DIR, "templates")],
88         'APP_DIRS': True,
89         'OPTIONS': {
90             'context_processors': [
91                 'django.template.context_processors.debug',
92                 'django.template.context_processors.request',
93                 'django.contrib.auth.context_processors.auth',
94                 'django.contrib.messages.context_processors.messages',
95             ],
96         },
97     },
98 ]
99
100 WSGI_APPLICATION = 'proyecto01.wsgi.application'

```

At the bottom of the editor, it shows `src\proyecto01\settings.py 99:1`, `CRLF`, `UTF-8`, `Python`, and `0 files`.

### 33. Autenticación para Enlaces en la Navbar

- Con la finalidad de mejorar el acceso y presentación a nuestra página web, añadiremos código a `navbar.html` separando los links. Cuando no se esté conectado se mostrará en la parte derecha **Registrar - Login**, en cambio para cuando se lo esté mostrará **Administracion** (sitio django) y **Salir**.

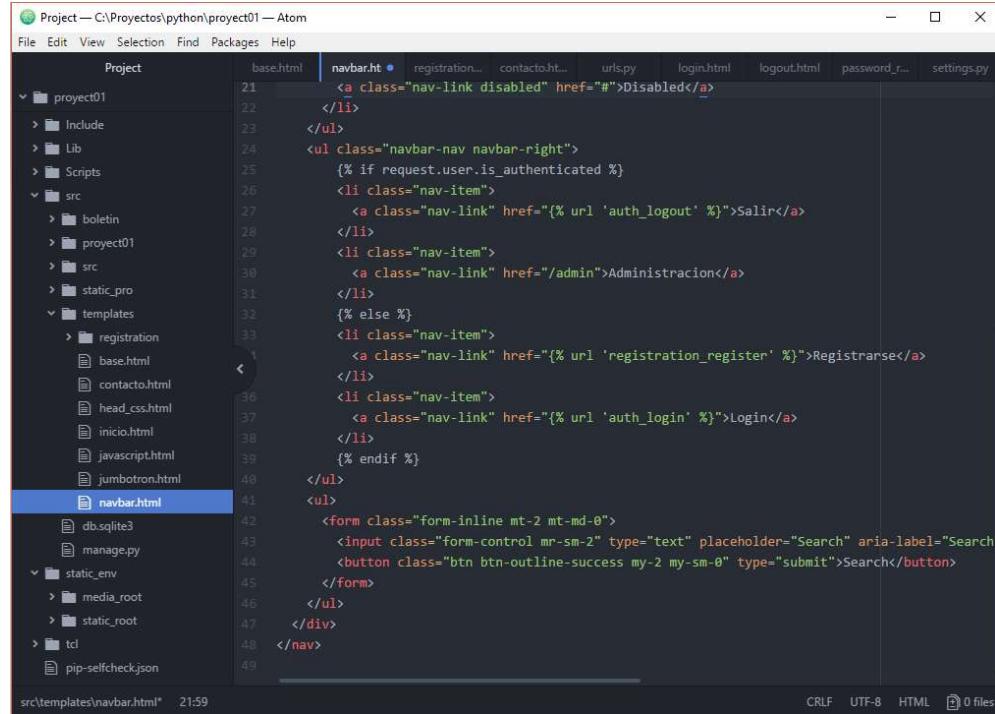
Archivo `navbar.html`:

```
{% load static %}
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
```

```

<ul class="navbar-nav mr-auto">
  <li class="nav-item active">
    <a class="nav-link" href="{% url 'inicio' %}">Home <span class="sr-only">(current)</span></a>
  </li>
  <li class="nav-item">
    <!-- La ruta directa usando las url about -->
    <a class="nav-link" href="{% url 'about' %}">About</a>
  </li>
  <li class="nav-item">
    <!-- La ruta directa usando las url contacto -->
    <a class="nav-link" href="{% url 'contacto' %}">Contact</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
<ul class="navbar-nav navbar-right">
  {% if request.user.is_authenticated %}
    <li class="nav-item">
      <a class="nav-link" href="{% url 'auth_logout' %}">Salir</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/admin">Administracion</a>
    </li>
  {% else %}
    <li class="nav-item">
      <a class="nav-link" href="{% url 'registration_register' %}">Registrarse</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'auth_login' %}">Login</a>
    </li>
  {% endif %}
</ul>
<ul>
  <form class="form-inline mt-2 mt-md-0">
    <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</ul>
</div>
</nav>

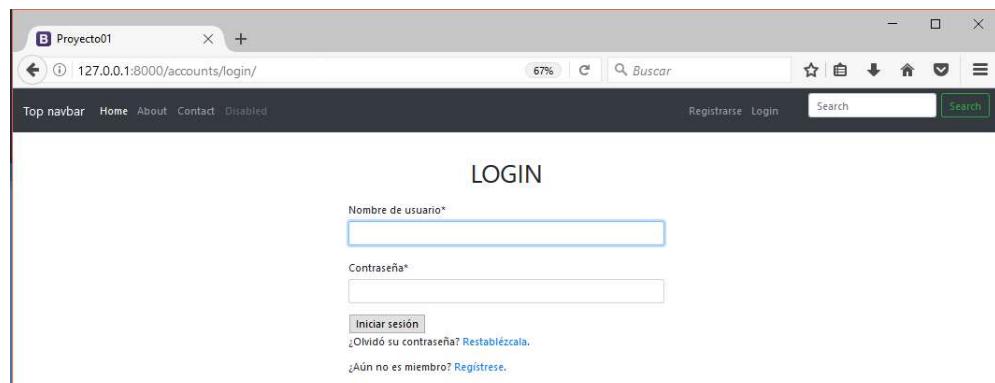
```



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for 'proyecto01' containing 'Include', 'Lib', 'Scripts', 'src' (with 'boletin' and 'proyecto01' subfolders), 'static\_pro', and 'templates' (with 'registration' and 'navbar' subfolders). The right pane shows the content of 'navbar.html'. The code includes logic for user authentication status, rendering different navigation items ('Salir', 'Administracion') based on whether the user is authenticated or not. It also includes links for registration and login. The bottom status bar shows 'src\templates\navbar.html' and line number '21:59'.

```

Project—C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project base.html navbar.htm registration... contacto.ht... urls.py login.html logout.html password_r... settings.py
  proyecto01
    > Include
    > Lib
    > Scripts
    > src
      > boletin
      > proyecto01
      > src
      > static_pro
    > templates
      > registration
        base.html
        contacto.html
        head_css.html
        inicio.html
        javascript.html
        jumbotron.html
      > navbar.html
        db.sqlite3
        manage.py
    > static_env
      > media_root
      > static_root
    > tcl
    pip-selfcheck.json
src\templates\navbar.html 21:59
  
```



### 34. Formulario de Login en la Navbar

- De esta forma agregamos código al archivo **navbar.html** con las validaciones respectivas que permitan una visualización adecuada cuando estemos dentro y no se repliquen innecesariamente las pantallas.

Archivo **navbar.html**:

```

{% load static %}

<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav ml-auto">
      {% if request.user.is_authenticated %}
        <li class="nav-item">
          <a class="nav-link" href="#">Disabled</a>
        </li>
        <ul class="navbar-nav navbar-right">
          {% if request.user.is_authenticated %}
            <li class="nav-item">
              <a class="nav-link" href="{% url 'auth_logout' %}">Salir</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/admin">Administracion</a>
            </li>
          {% else %}
            <li class="nav-item">
              <a class="nav-link" href="{% url 'registration_register' %}">Registrarse</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{% url 'auth_login' %}">Login</a>
            </li>
          {% endif %}
        </ul>
      </li>
    <ul class="form-inline mt-2 mt-md-0">
      <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search" name="q">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </ul>
  </div>
</div>
  
```

```

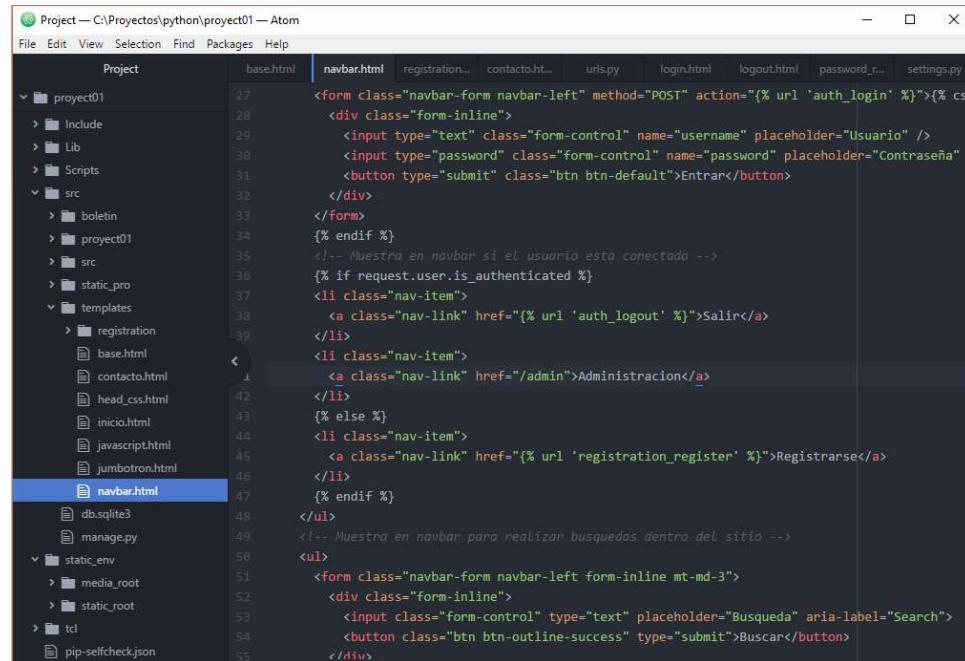
<ul class="navbar-nav mr-auto">
    <li class="nav-item active">
        <a class="nav-link" href="{% url 'inicio' %}">Home <span class="sr-only">(current)</span></a>
    </li>
    <li class="nav-item">
        <!-- La ruta directa usando las url about -->
        <a class="nav-link" href="{% url 'about' %}">About</a>
    </li>
    <li class="nav-item">
        <!-- La ruta directa usando las url contacto -->
        <a class="nav-link" href="{% url 'contacto' %}">Contact</a>
    </li>
    <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
    </li>
</ul>
<ul class="navbar-nav navbar-right">
    <!-- Muestra en navbar si el usuario no esta conectado -->
    {% if not request.user.is_authenticated and not "/accounts/login" in
request.get_full_path %}
        <form class="navbar-form navbar-left" method="POST" action="{% url
'auth_login' %}">{% csrf_token %}
            <div class="form-inline">
                <input type="text" class="form-control" name="username"
placeholder="Usuario" />
                <input type="password" class="form-control" name="password"
placeholder="Contraseña" />
                <button type="submit" class="btn btn-default">Entrar</button>
            </div>
        </form>
    {% endif %}
    <!-- Muestra en navbar si el usuario esta conectado -->
    {% if request.user.is_authenticated %}
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auth_logout' %}">Salir</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/admin">Administracion</a>
        </li>
    {% else %}
        <li class="nav-item">
            <a class="nav-link" href="{% url 'registration_register' %}">Registrarse</a>
        </li>
    {% endif %}
</ul>
<!-- Muestra en navbar para realizar busquedas dentro del sitio -->
<ul>

```

```

<form class="navbar-form navbar-left form-inline mt-md-3">
    <div class="form-inline">
        <input class="form-control" type="text" placeholder="Busqueda" aria-
label="Search">
        <button class="btn btn-outline-success" type="submit">Buscar</button>
    </div>
</form>
</ul>
</div>
</nav>

```

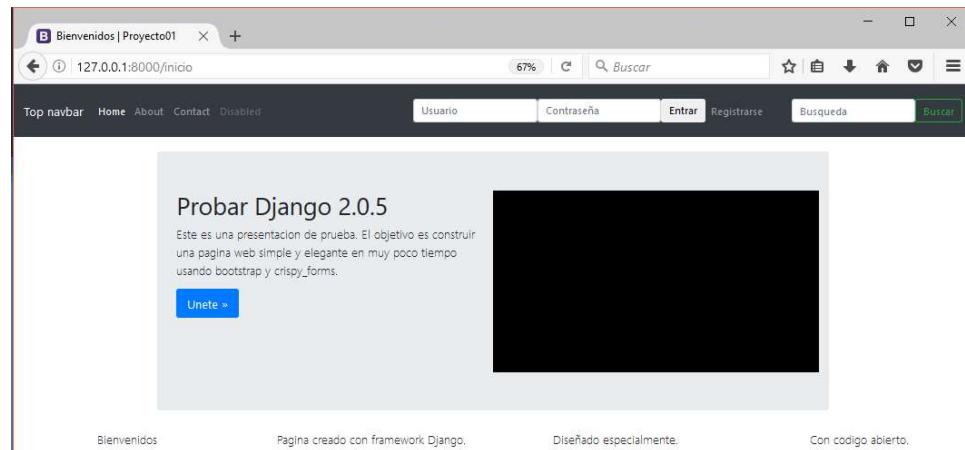


The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for 'proyecto01' containing 'Include', 'Lib', 'Scripts', 'src' (which includes 'boletin', 'proyecto01', 'src', 'static\_pro', and 'templates' which further includes 'registration', 'base.html', 'contacto.html', 'head\_css.html', 'inicio.html', 'javascript.html', and 'jumbotron.html'), and 'navbar.html'. The right pane shows the content of 'navbar.html':

```

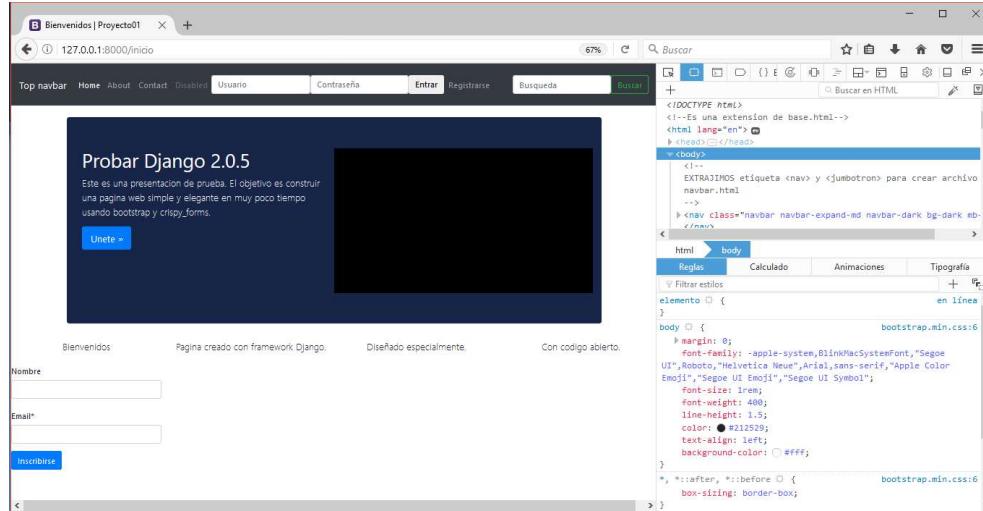
<form class="navbar-form navbar-left form-inline mt-md-3">
    <div class="form-inline">
        <input class="form-control" type="text" placeholder="Busqueda" aria-
label="Search">
        <button class="btn btn-outline-success" type="submit">Buscar</button>
    </div>
</form>
</ul>
</div>
</nav>

```



## 35. Estilo Personalizar CSS en Jumbotron & Navbar

- Recordemos y comprendamos que:
- Presionamos **F12**, para observar el entorno HTML, y seleccionamos la clase a la cual realizaremos los cambios.



- a.) El caso **JUMBOTRON**, lo tenemos dentro del archivo **jumbotron.html**, aquí realizaríamos cambios, no tenemos definidos los estilos, así que usariamos por medio de **block**.

Entonces para este caso, añadimos al archivo **base.html** un block llamado **style**, y en el archivo **jumbotron.html** agregamos el mismo código mas la definición del estilo de la clase.

#### Archivo **base.html**:

```
<!doctype html>
{% load static %}
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="{% static '/img/favicon.ico' %}">

    <title>{% block titulo_sitio%}Proyecto01{% endblock %}</title>
    <!--
        EXTRAJIMOS etiqueta <head> para crear archivo head_css.html -->
        {% include "head_css.html" %}
    </head>

    <body>
        <!-- EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html -->
        {% include "navbar.html" %}
        <!-- definicion de block jumbotron -->
        {% block jumbotron %}

        <!-- Creamos un block para styles personalizados .css para la clase jumbotron -->
        {% block style %}
```

```

    {% endblock %}
    {% endblock %}
    <!-- definicion de block formulario que sera insertado -->
    {% block formulario %}
    {% endblock %}
    <!-- definicion de block content para que sera insertado -->
    {% block content %}
    {% endblock %}
    <!-- EXTRAJIMOS etiqueta <head> para crear archivo javascript.html -->
    {% include "javascript.html" %}
</body>
</html>

```

The screenshot shows the Atom code editor interface. On the left is the project tree:

- Project — C:\Proyectos\python\proyecto01 — Atom
- File Edit View Selection Find Packages Help
- Project
  - └ proyecto01
    - > Include
    - > Lib
    - > Scripts
    - src
      - > boletin
      - > proyecto01
      - > src
      - > static\_pro
      - templates
        - > registration
        - base.html
        - contacto.html
        - head\_css.html
        - inicio.html
        - javascript.html
        - jumbotron.html
        - navar.html
        - db.sqlite3
        - manage.py
      - static\_env
        - > media\_root
        - > static\_root
      - tcl
      - pip-selfcheck.json

Two tabs are open in the main area:

  - jumbotron.html
  - base.html

The base.html file contains the following code:

```

8     <meta name="author" content="">
9     <link rel="icon" href="{% static '/img/favicon.ico' %}">
10
11     <title>{% block titulo_sitio%}Proyecto01{% endblock %}</title>
12     <!--
13     EXTRAJIMOS etiqueta <head> para crear archivo head_css.html -->
14     {% include "head_css.html" %}
15     </head>
16
17     <body>
18     <!-- EXTRAJIMOS etiqueta <nav> y <jumbotron> para crear archivo navbar.html -->
19     {% include "navbar.html" %}
20     <!-- definicion de block jumbotron -->
21     {% block jumbotron %}
22         <!-- Creamos un block para styles personalizados sobreescribiendo la .css-->
23         {% block style %}
24             {% endblock %}
25         {% endblock %}
26         <!-- definicion de block formulario que sera insertado -->
27         {% block formulario %}
28             {% endblock %}
29         <!-- definicion de block content para que sera insertado -->
30         {% block content %}
31             {% endblock %}
32         <!-- EXTRAJIMOS etiqueta <head> para crear archivo javascript.html -->
33         {% include "javascript.html" %}
34     </body>
35     </html>
36

```

#### Archivo jumbotron.html:

```

{% load static %}
<!-- Creamos un block para styles personalizados .css para la clase jumbotron -->
<style>
    {% block style %}
        .jumbotron {
            background-color: #172548;
            color: white;
        }
    {% endblock %}
</style>

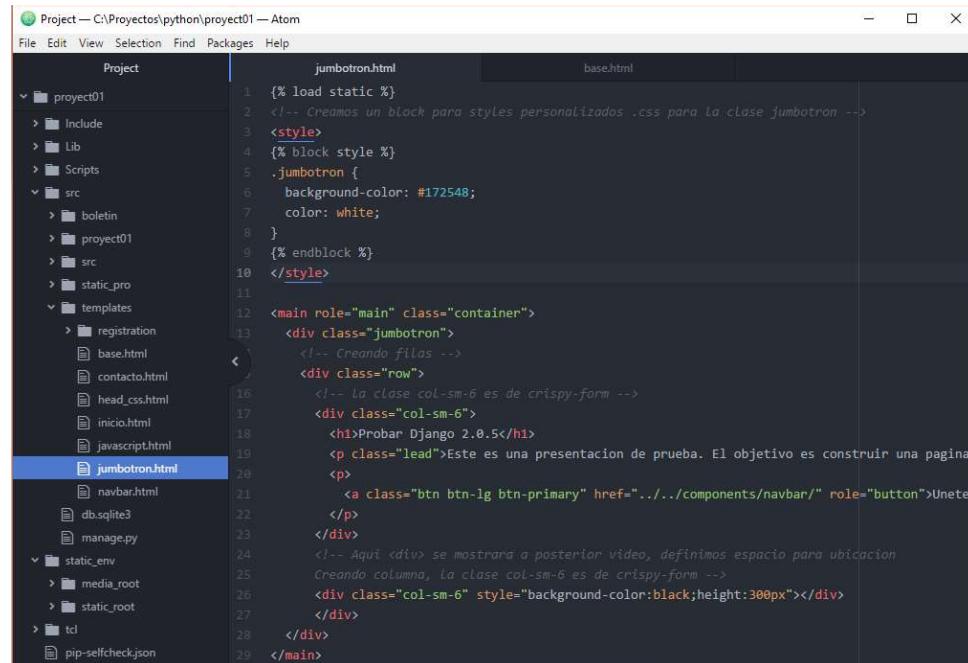
<main role="main" class="container">
    <div class="jumbotron">
        <!-- Creando filas -->

```

```

<div class="row">
    <!-- la clase col-sm-6 es de crispy-form -->
    <div class="col-sm-6">
        <h1>Probar Django 2.0.5</h1>
        <p class="lead">Este es una presentacion de prueba. El objetivo es construir una pagina web simple y elegante en muy poco tiempo usando bootstrap y crispy_forms.</p>
        <p>
            <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">Unete &raquo;</a>
        </p>
        </div>
        <!-- Aqui <div> se mostrara a posterior video, definimos espacio para ubicacion Creando columna, la clase col-sm-6 es de crispy-form -->
        <div class="col-sm-6" style="background-color:black;height:300px"></div>
    </div>
</div>
</main>

```



The screenshot shows the Atom code editor interface. The title bar says "Project — C:\Proyectos\python\proyecto01 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The left sidebar shows the project structure:

- project01
  - Include
  - Lib
  - Scripts
  - src
    - boletin
    - project01
    - src
    - static\_pro
    - templates
      - registration
      - base.html
      - contacto.html
      - head\_css.html
      - inicio.html
      - javascript.html
      - jumbotron.html**
      - navar.html
  - static\_env
    - media\_root
    - static\_root
  - tcl
  - pip-selfcheck.json

The main editor area displays the content of the **jumbotron.html** file:

```

1  {% load static %}
2  <!-- Creamos un block para styles personalizados .css para la clase jumbotron -->
3  <style>
4  {% block style %}
5      .jumbotron {
6          background-color: #172548;
7          color: white;
8      }
9  {% endblock %}
10 </style>
11
12 <main role="main" class="container">
13     <div class="jumbotron">
14         <!-- Creando filas -->
15         <div class="row">
16             <!-- La clase col-sm-6 es de crispy-form -->
17             <div class="col-sm-6">
18                 <h1>Probar Django 2.0.5</h1>
19                 <p class="lead">Este es una presentacion de prueba. El objetivo es construir una pagina
20                 <p>
21                     <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">Unete &raquo;</a>
22                 </p>
23             </div>
24             <!-- Aqui <div> se mostrara a posterior video, definimos espacio para ubicacion Creando columna, la clase col-sm-6 es de crispy-form -->
25             <div class="col-sm-6" style="background-color:black;height:300px"></div>
26         </div>
27     </div>
28 </main>

```

- b.) El caso **NAVBAR**, añadimos al archivo **navbar-top.css** el código mas la definición del estilo de la clase.

O también usando las clases nativas incluidas en la clase:

Archivo **navbar.html**:

```

<!--
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4"> -->
<nav class="navbar navbar-expand-md navbar-dark bg-primary mb-4">

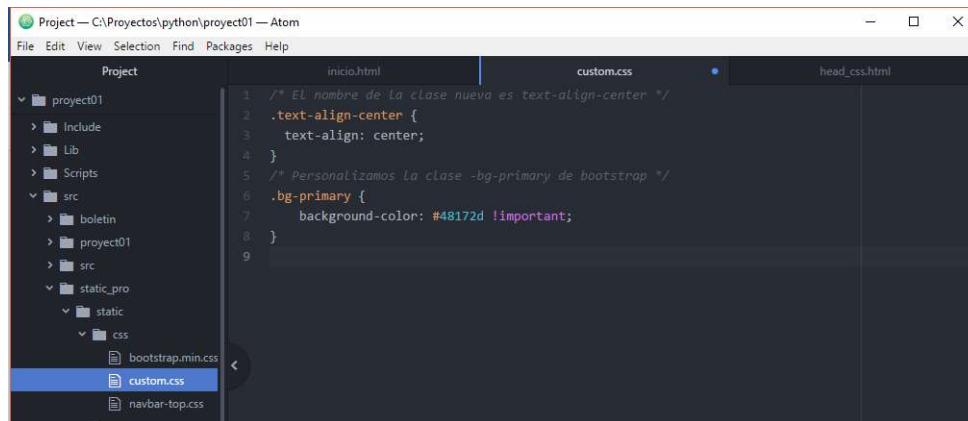
```

También podemos **custom.css** para darle personalidad a los estilos:

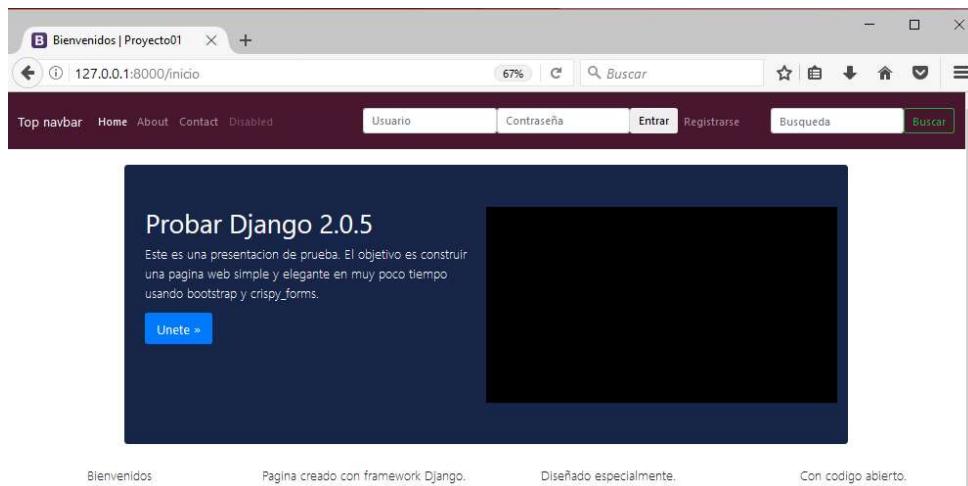
#### Archivo custom.css:

```
/* El nombre de la clase nueva es text-align-center */
.text-align-center {
    text-align: center;
}

/* Personalizamos la clase -bg-primary de bootstrap para el navbar */
.bg-primary {
    background-color: #48172d !important;
}
```

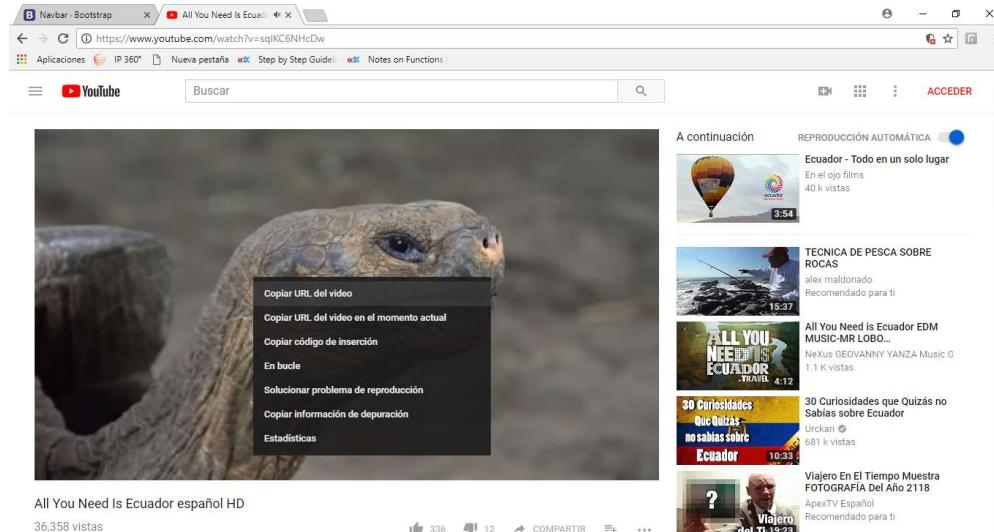


The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for 'Proyecto01' with folders like 'Include', 'Lib', 'Scripts', 'src', and 'static\_pro'. Inside 'static\_pro', there's a 'css' folder containing 'bootstrap.min.css', 'custom.css' (which is selected), and 'navbar-top.css'. The main editor area shows the content of 'custom.css'.



## 36. Añadir Video de YouTube al Jumbotron

- Incrustacion de video desde internet, bastara con copiar las url para cada uno y abregarlos dentro del jumbotron.html., (**ALLYOUNEEDCUADOR**)  
Justo donde antes se encontraba el espacio para la imagen



### Archivo jumbotron.html:

```
% load static %}

<!-- Creamos un block para styles personalizados .css para la clase jumbotron --&gt;
&lt;style&gt;
{%
block style %
.jumbotron {
background-color: #172548;
color: white;
}
{%
endblock %
&lt;/style&gt;

&lt;main role="main" class="container"&gt;
&lt;div class="jumbotron"&gt;
&lt;!-- Creando filas --&gt;
&lt;div class="row"&gt;
&lt;!-- la clase col-sm-6 es de crispy-form --&gt;
&lt;div class="col-sm-6"&gt;
&lt;h1&gt;Probar Django 2.0.5&lt;/h1&gt;
&lt;p class="lead"&gt;Este es una presentacion de prueba. El objetivo es construir una pagina web simple y elegante en muy poco tiempo usando bootstrap y crispy_forms.&lt;/p&gt;
&lt;p&gt;
&lt;a class="btn btn-lg btn-primary" href=".//components/navbar/" role="button"&gt;Unete &amp;raquo;&lt;/a&gt;
&lt;/p&gt;
&lt;/div&gt;
&lt;!-- Aqui &lt;div&gt; se mostrara a posterior video, definimos espacio para ubicacion Creando columna, la clase col-sm-6 es de crispy-form
&lt;div class="col-sm-6" style="background-color:black;height:300px"&gt;&lt;/div&gt;
&lt;/div&gt; --&gt;
&lt;!-- Añadir video de youtube --&gt;
&lt;div class="col-sm-6"&gt;</pre>

```

```

<iframe width="560" height="315"
src="https://www.youtube.com/embed/sqIKC6NHcDw?rel=0" frameborder="0"
allow="autoplay; encrypted-media" allowfullscreen></iframe>
</div>
</div>
</div>
</main>

```

The screenshot shows the Atom code editor with the file 'jumbotron.html' open. The code includes a CSS block for the '.jumbotron' class and an `<iframe>` tag with specific dimensions and attributes.

```

% load static %

<style>
{% block style %}
.jumbotron {
    background-color: #172548;
    color: white;
}
{% endblock %}
</style>

<main role="main" class="container">
    <div class="jumbotron">
        <!-- Creando filas -->
        <div class="row">
            <!-- la clase col-sm-6 es de crispy-form -->
            <div class="col-sm-6">
                <h1>Probar Django 2.0.5</h1>
                <p class="lead">Este es una presentación de prueba. El objetivo es construir una página</p>
                <a class="btn btn-lg btn-primary" href=".../components/navbar/" role="button">Unete</a>
            </div>
            <!-- Aquí <div> se mostrará a posterior video, definimos espacio para ubicación -->
            <div class="col-sm-6">
                <div with="560" height="315" src="https://youtu.be/sqIKC6NHcDw" frameborder="0" allowfullscreen></div>
            </div>
        </div>
    </div>
</main>

```

The screenshot shows a web browser displaying a Django application. The page has a dark header with a logo and navigation links. The main content area features a jumbotron with a video player. Below the jumbotron is a registration form with fields for 'Nombre' and 'Email\*', and a 'Inscribirse' button.

- Incrustacion de imágenes desde internet, podemos hacer dos cosas para renderizar desde el archivo **jumbotron.html**:
  - 1.) Usar los links para cada uno y abregarlos dentro del archivo

[https://www.eluniverso.com/sites/default/files/styles/powgallery\\_800/public/fotos/2016/12/data10146139.jpg](https://www.eluniverso.com/sites/default/files/styles/powgallery_800/public/fotos/2016/12/data10146139.jpg)

Archivo **jumbotron.html**:

```

<% load static %>
<!-- Creamos un block para styles personalizados .css para la clase jumbotron -->
<style>
{% block style %}
.jumbotron {
  background-color: #172548;
  color: white;
}
{% endblock %}
</style>

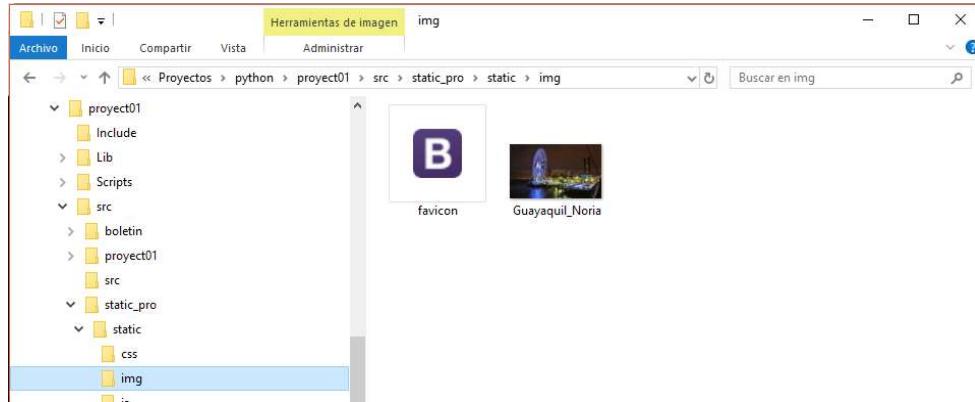
<main role="main" class="container">
<div class="jumbotron">
<!-- Creando filas -->
<div class="row">
<!-- la clase col-sm-6 es de crispy-form -->
<div class="col-sm-6">
  <h1>Probar Django 2.0.5</h1>
  <p class="lead">Este es una presentacion de prueba. El objetivo es construir una pagina web simple y elegante en muy poco tiempo usando bootstrap y crispy_forms.</p>
  <p>
    <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">Unete &raquo;</a>
  </p>
</div>
<!-- Aqui <div> se mostrara a posterior video, definimos espacio para ubicacion
Creando columna, la clase col-sm-6 es de crispy-form
<div class="col-sm-6" style="background-color:black;height:300px"></div>
</div> -->
<!-- Añadir video de youtube -->
<div class="col-sm-6">
  <!-- <iframe width="560" height=315 src="https://youtu.be/sqIKC6NHcDw"
frameborder="0" allowfullscreen></iframe> -->
  
</div>
</div>
</div>

```

</main>

- 2.) Siempre es preferible descargar las imágenes y copiarlas en las carpetas estáticas (**optimo**), ya que si falla algo podemos cargarla estáticamente en la carpeta `\src\static_pro\static\img`.

Imagen: Guayaquil\_Noria.jpg



## Archivo jumbotron.html:

```
{% load static %}  
<!-- Creamos un block para styles personalizados .css para la clase jumbotron -->  
<style>  
{% block style %}  
jumbotron {  
    background-color: #172548;  
    color: white;  
}  
{% endblock %}  
</style>  
  
<main role="main" class="container">  
    <div class="jumbotron">  
        <!-- Creando filas -->  
        <div class="row">  
            <!-- la clase col-sm-6 es de crispy-form -->  
            <div class="col-sm-6">  
                <h1>Probar Django 2.0.5</h1>  
                <p class="lead">Este es una presentacion de prueba. El objetivo es construir una  
pagina web simple y elegante en muy poco tiempo usando bootstrap y  
crispy_forms.</p>  
                <p>  
                    <a class="btn btn-lg btn-primary" href="..../components/navbar/"  
                    role="button">Unete &raquo;</a>  
                </p>  
            </div>  
        </div>  
        <!-- Aqui <div> se mostrara a posterior video, definimos espacio para ubicacion  
        Creando columna, la clase col-sm-6 es de crispy-form -->  
    </div>
```

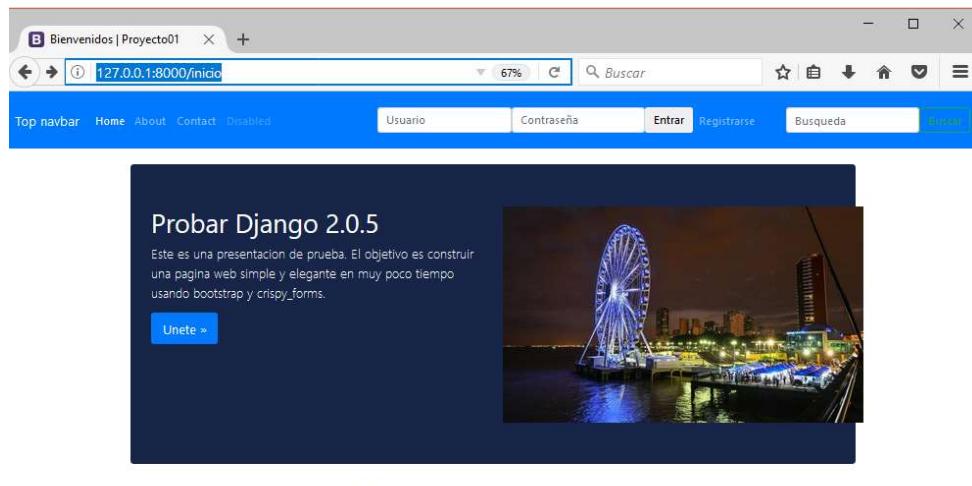
```
<div class="col-sm-6" style="background-color:black;height:300px"></div>
</div> -->
<!-- Añadir video de youtube -->
<div class="col-sm-6">
    <!-- <iframe width="560" height=315 src="https://youtu.be/sqIKC6NHcDw" frameborder="0" allowfullscreen></iframe> -->
    <!-- Añadir imagen estatica -->
    
</div>
</div>
</div>
</main>
```

The screenshot shows the Atom code editor interface. On the left is the project tree for 'proyecto01' containing 'Include', 'Lib', 'Scripts', 'src' (with 'boletin', 'proyecto01', 'src', 'static\_pro' subfolders), 'static' (with 'css', 'img' (containing 'favicon.ico' and 'Guayaquil\_Noria.jpg'), 'js', 'templates', 'db.sqlite3', 'manage.py'), 'static\_env', 'media\_root', 'static\_root', 'tcl', and 'pip-selfcheck.json'. The right pane shows the 'jumbotron.html' file with the provided code. The 'settings.py' file is also visible in the tab bar.

```
File Edit View Selection Find Packages Help
```

```
Project settings.py jumbotron.html
```

```
1 load static %}
2 -- Creamos un block para styles personalizados .css para la clase jumbotron -->
3 <style>
4     block style %
5     umbotron {
6         background-color: #172548;
7         color: white;
8     }
9     endblock %
10 <style>
11
12 <div role="main" class="container">
13     <div class="jumbotron">
14         <!-- Creando filas -->
15         <div class="row">
16             <!-- La clase col-sm-6 es de crispy-form -->
17             <div class="col-sm-6">
18                 <h1>Probar Django 2.0.5</h1>
19                 <p class="lead">Este es una presentacion de prueba. El objetivo es construir una pagina web simple y elegante en muy poco tiempo usando bootstrap y crispy_forms.</p>
20                 <a class="btn btn-lg btn-primary" href="../../components/navbar/" role="button">Unete &gt;</a>
21             </div>
22             </div>
23         </div>
24         <!-- Aquí <div> se mostrara a posterior video, definimos espacio para ubicacion= -->
25         <div class="col-sm-6">
26             <!-- <iframe width="560" height=315 src="https://youtu.be/sqIKC6NHcDw" frameborder="0" allowfullscreen-->
27             
28         </div>
29     </div>
30 
```



Ejecutamos a nivel de comandos en la ruta **src**, para actualizar los archivos en las réplicas del servidor de producción.

D:\Proyectos\python\proyecto01\src>**python manage.py collectstatic**

```
Administrator: cmd
C:\Proyectos\python\proyecto01\src>python manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:
    C:\Proyectos\python\proyecto01\static_env\static_root
This will overwrite existing files!
Are you sure you want to do this?

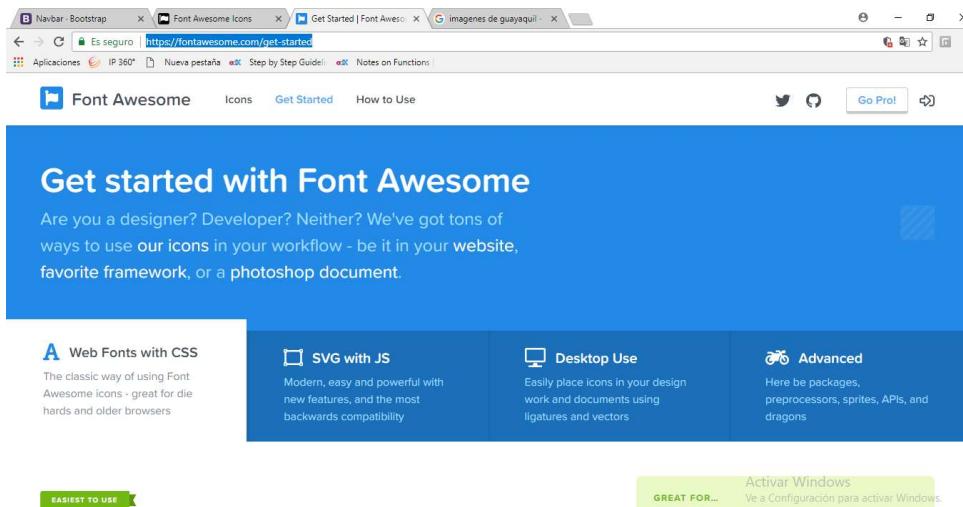
Type 'yes' to continue, or 'no' to cancel: yes
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\css\custom.css'
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\img\Guayaquil_Malecon.jpg'
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\img\Guayaquil_Noria.jpg'

3 static files copied to 'C:\Proyectos\python\proyecto01\static_env\static_root', 124 unmodified.
```

## 37. Fontawesome

- Iconos, fonts y animaciones.

<https://fontawesome.com/>



**FORMA 1:** Vamos a Get Started y copiamos la ruta completa free.

```
<link rel="stylesheet"
      href="https://use.fontawesome.com/releases/v5.0.13/css/all.css" integrity="sha384-DNOHZ68U8hZfKXOrtjWvJxusGo9WQnrNx2sqG0tfsghAvtVlRW3tvkXWZh58N9jp" crossorigin="anonymous">
```

Lo pegamos en el archivo head\_css.html en la cabecera.

**Nota:** si no estamos conectados a internet, se volverá lenta la carga de los iconos y animaciones o en todo caso descargar para poder usar directamente sobre producción.

Archivo **head\_css.html**:

```

{% load static %}
<!-- Bootstrap core CSS -->
<!--
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<!-- ASOWESOME iconos animados -->
<link   rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.13/css/all.css"
integrity="sha384-DNOHZ68U8hzfKXOrjWvixusGo9WQnrNx2sqG0tfsghAvtVIRW3tvkXWZh58N9jp"
crossorigin="anonymous">

<link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">
<!-- Custom styles for this template -->
<!--
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
<!--
Aqui agregamos el codigo necesario para la etiqueta de estilo custom.css -->
<link href="{% static '/css/custom.css' %}" rel="stylesheet">

```

#### Archivo **inicio.html**:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}

<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}
<div class="row">
<!--
la clase col-sm-3 es de crispy-form numero de columnas 3
la clase col-xs-12 es de crispy-form ajusta al comprimir la página

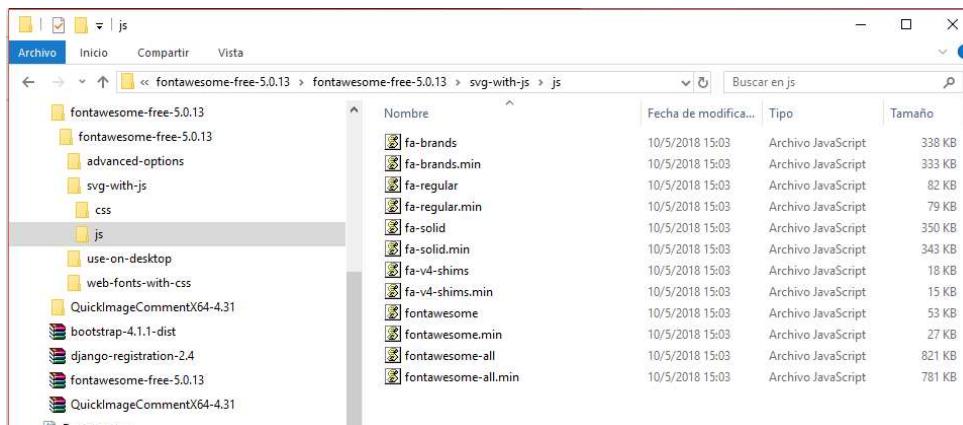
```

*la clase pull-right es de crispy-form alinea a la derecha*

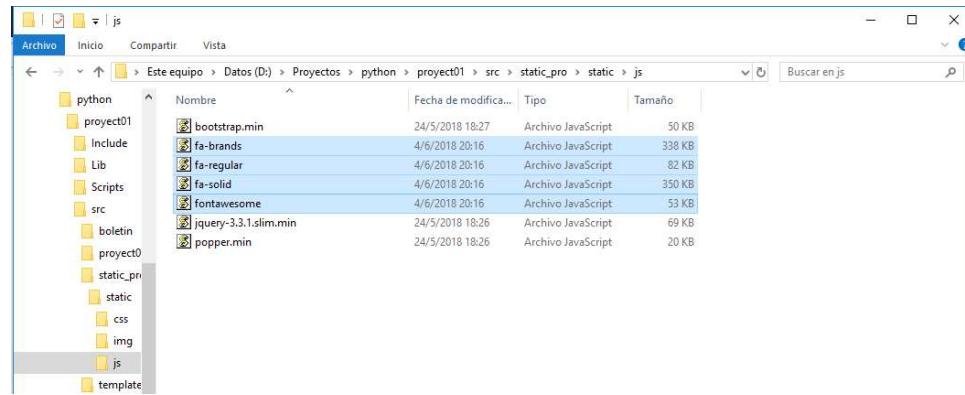
```
-->
<div class="col-sm-3 col-xs-12 text-left">
    <p class="lead text-center">{{ titulo }}</p>
    <form method="POST" action="">{{ csrf_token }}
    <!-- Cambiamos el form a crispy para mejor presentacion
    {{ el_form.form }}>
-->
    {{ el_form|crispy }}>
    <input class="btn btn-primary" type="submit" value="Inscribirse">
</form>
</div>
<div class="col-sm-3">
    <p class="lead text-center">Pagina creada con framework Django.
    <span class="fa-stack fa-5x">
        <i class="fa fa-circle-o-notch fa-stack-2x"></i>
        <i class="fa fa-bullhorn fa-stack-1x"></i>
    </span>
    </p>
</div>
<div class="col-sm-3">
    <p class="lead text-center">Diseñado especialmente.</p>
</div>
<div class="col-sm-3">
    <p class="lead text-center">Con codigo abierto.</p>
</div>
</div>
{% endblock %}
```

**FORMA 2:** (OPTIMA) Instalamos las .js dentro de nuestros archivos estaticos

Descargando **AWESOME** dentro de una carpeta y seleccionando los archivos .js, para luego enviarlos a **\src\static\_pro\static\js**:



Entonces nos quedaría la carpeta con los nuevos archivos de la siguiente manera:



D:\Proyectos\python\proyecto01\src\static\_pro\static\js\fa-brands.js  
D:\Proyectos\python\proyecto01\src\static\_pro\static\js\fa-regular.js  
D:\Proyectos\python\proyecto01\src\static\_pro\static\js\fa-solid.js  
D:\Proyectos\python\proyecto01\src\static\_pro\static\js\fontawesome.js

Modificamos los archivos **head\_css.html** e **inicio.html**, agregando código para cargar **ASOWESOME**, directamente en nuestro servidor, sin necesidad de conectarnos a su servicio.

#### Archivo **head\_css.html**:

```
% load static %}

<!-- ASOWESOME iconos animados -->
<script defer src="{% static '/js/fa-brands.js' %}"></script>
<script defer src="{% static '/js/fa-regular.js' %}"></script>
<script defer src="{% static '/js/fa-solid.js' %}"></script>
<script src="{% static '/js/fontawesome.js' %}"></script>
<!-- Bootstrap core CSS -->
<!--
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">
<!-- Custom styles for this template -->
<!--
<link href="navbar-top.css" rel="stylesheet">
Aqui agregamos el codigo necesario para la etiqueta -->
<link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
<!--
Aqui agregamos el codigo necesario para la etiqueta de estilo custom.css -->
<link href="{% static '/css/custom.css' %}" rel="stylesheet">
```

The screenshot shows the Atom code editor interface. On the left, the project structure is displayed under 'Project'. It includes a 'project01' folder containing '\_init\_.py', 'settings.py', 'urls.py', and 'wsgi.py'. Inside 'static\_pro' is a 'static' folder with 'css' (containing 'bootstrap.min.css', 'custom.css', and 'navbar-top.css'), 'img', and 'js' folders. The 'templates' folder contains 'registration', 'base.html', 'contacto.html', and 'head\_css.html'. The 'head\_css.html' file is currently selected and its content is visible in the main editor area.

```

Project — D:\manuales\python — Atom
File Edit View Selection Find Packages Help
Project
  project01
    __init__.py
    settings.py
    urls.py
    wsgi.py
  static_pro
    static
      css
        bootstrap.min.css
        custom.css
        navbar-top.css
      img
      js
    templates
      registration
      base.html
      contacto.html
      head_css.html
    head_css.html
    inicio.html
    javascript.html
    jumbotron.html
views.py          base.html          head_css.html          inicio.html
1  {% load static %}          1  <!--
2  <!--
3  <link rel="icon" href="../../../../favicon.ico">
4  Aquí agregamos el código necesario para la etiqueta -->
5  <link rel="icon" href="{% static '/img/favicon.ico' %}">
6  <title>Top navbar example for Bootstrap</title>
7  <!-- ASOMEASOME iconos animados -->
8  <script defer src="{% static '/js/fa-brands.js' %}"></script>
9  <script defer src="{% static '/js/static/fa-regular.js' %}"></script>
10 <script defer src="{% static '/js/fa-solid.js' %}"></script>
11 <script src="{% static '/js/fontawesome.js' %}"></script>
12 <!-- Bootstrap core CSS -->
13 <!--
14 <link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
15 Aquí agregamos el código necesario para la etiqueta -->
16 <link href="{% static '/css/bootstrap.min.css' %}" rel="stylesheet">
17 <!-- Custom styles for this template -->
18 <!--
19 <link href="navbar-top.css" rel="stylesheet">
20 Aquí agregamos el código necesario para la etiqueta -->
21 <link href="{% static '/css/navbar-top.css' %}" rel="stylesheet">
22 <!--
23 Aquí agregamos el código necesario para la etiqueta de estilo custom.css -->
24 <link href="{% static '/css/custom.css' %}" rel="stylesheet">

```

#### Archivo inicio.html:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}

<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}
  {% include "jumbotron.html" %}
  {% endblock %}

{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}
<div class="row">
  <!--
    la clase col-sm-3 es de crispy-form numero de columnas 3
    la clase col-xs-12 es de crispy-form ajusta al comprimir la página
    la clase pull-right es de crispy-form alinea a la derecha
  -->
  <div class="col-sm-3 col-xs-12 text-left">
    <p class="lead text-center">{{ titulo }}</p>
    <form method="POST" action="">{% csrf_token %}

```

```

<!-- Cambiamos el form a crispy para mejor presentacion
{{ el_form.form }}
-->
{{ el_form|crispy }}
<input class="btn btn-primary" type="submit" value="Inscribirse">
</form>
</div>
<div class="col-sm-3">
<p class="lead text-center">Pagina creado con framework Django.
<!-- ASOWESOME iconos animados -->
<span class="fa-stack fa-5x">
<i class="fas fa-circle-notch fa-spin fa-2x"></i> --
<i class="fa fa-bullhorn fa-stack-1x"></i>
</span>
</p>
</div>
<div class="col-sm-3">
<p class="lead text-center">Diseñado especialmente.
<!-- ASOWESOME iconos animados -->
<div><i class="fas fa-home fa-fw" style="background:MistyRose"></i>
Home</div>
<div><i class="fas fa-info fa-fw" style="background:MistyRose"></i> Info</div>
<div><i class="fas fa-book fa-fw" style="background:MistyRose"></i>
Library</div>
<div><i class="fas fa-pencil-alt fa-fw" style="background:MistyRose"></i>
Applications</div>
<div><i class="fas fa-cog fa-fw" style="background:MistyRose"></i>
Settings</div>
</p>
</div>
<div class="col-sm-3">
<p class="lead text-center">Con codigo abierto.
<!-- ASOWESOME iconos animados -->
<span class="fa-stack fa-5x">
<i class="fa fa-spinner fa-spin fa-stack-2x"></i>
<i class="fa fa-camera-retro fa-stack-1x"></i>
</span>
</p>
</div>
</div>
{% endblock %}

```

inicio.html — C:\Proyectos\python\proyect01 — Atom

File Edit View Selection Find Packages Help

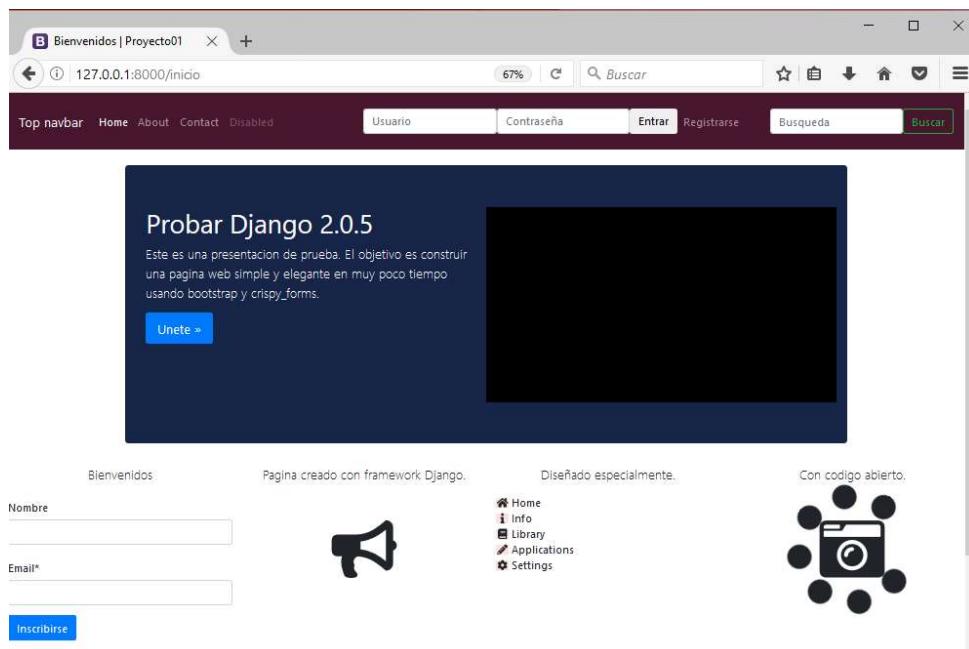
Project inicio.html head\_css.html

```

38 <div class="col-sm-3">
39   <p class="lead text-center">Pagina creado con framework Django.
40   <!-- ASOMEONESOME iconos animados -->
41   <span class="fa-stack fa-5x">
42     <i class="fas fa-circle-notch fa-spin fa-2x"></i> <br/>
43     <i class="fa fa-bullhorn fa-stack-1x"></i>
44   </span>
45 </p>
46 </div>
47 <div class="col-sm-3">
48   <p class="lead text-center">Diseñado especialmente.
49   <!-- ASOMEONESOME iconos animados -->
50   <div><i class="fas fa-home fa-fw" style="background:MistyRose"></i> Home</div>
51   <div><i class="fas fa-info fa-fw" style="background:MistyRose"></i> Info</div>
52   <div><i class="fas fa-book fa-fw" style="background:MistyRose"></i> Library</div>
53   <div><i class="fas fa-pencil-alt fa-fw" style="background:MistyRose"></i> Applications</div>
54   <div><i class="fas fa-cog fa-fw" style="background:MistyRose"></i> Settings</div>
55 </p>
56 <div class="col-sm-3">
57   <p class="lead text-center">Con codigo abierto.
58   <!-- ASOMEONESOME iconos animados -->
59   <span class="fa-stack fa-5x">
60     <i class="fa fa-spinner fa-spin fa-stack-2x"></i>
61     <i class="fa fa-camera-retro fa-stack-1x"></i>
62   </span>
63 </p>
64 </div>
65 </div>
66 </div>

```

src\templates\inicio.html\* 57:25 CRLF UTF-8 HTML 0 files



## ABOUT.HTML

Terminaremos la pagina que falta la cual es **about.html**, modificando los archivos **views.py**, **urls.py**.

Archivo **views.py**:

```

...
def about(request):
    return render(request, "about.html", {})
...

```

The screenshot shows the Atom IDE interface with the following details:

- Project:** C:\Proyectos\python\project01
- File:** views.py
- Code Content:**

```

67     email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje, form_email)
68     send_mail(asunto,
69         email_mensaje,
70         email_from,
71         email_to,
72         fail_silently=False,
73     )
74     #Forma 3 de llenar
75     #for key, value in form.cleaned_data.items():
76     #    print(key, value)
77     #Forma 2 de llenar
78     #for key in form.cleaned_data:
79     #    print(key)
80     #    print(form.cleaned_data.get(key))
81     #Forma 1 de llenar
82     #nombre = form.cleaned_data.get("nombre")
83     #email = form.cleaned_data.get("email")
84     #mensaje = form.cleaned_data.get("mensaje")
85     #print(nombre, email, mensaje)
86
87     context = {
88         "titulo": titulo,
89         "contacto": form,
90     }
91     return render(request, "contacto.html", context)
92
93 def about(request):
94     return render(request, "about.html", {})

```
- Project Structure:**
  - project01
    - Include
    - Lib
    - Scripts
    - src
      - boletin
        - \_pycache\_
        - migrations
        - \_\_init\_\_.py
        - admin.py
        - apps.py
        - forms.py
        - models.py
        - tests.py
        - views.py
      - static
      - static\_env
        - media\_root
      - static\_root
    - tcl
- Bottom Status Bar:** CRLF, UTF-8, Python, 0 files

#### Archivo urls.html:

```

# Redux registration
from django.conf.urls import url, include
from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings
from django.conf.urls.static import static

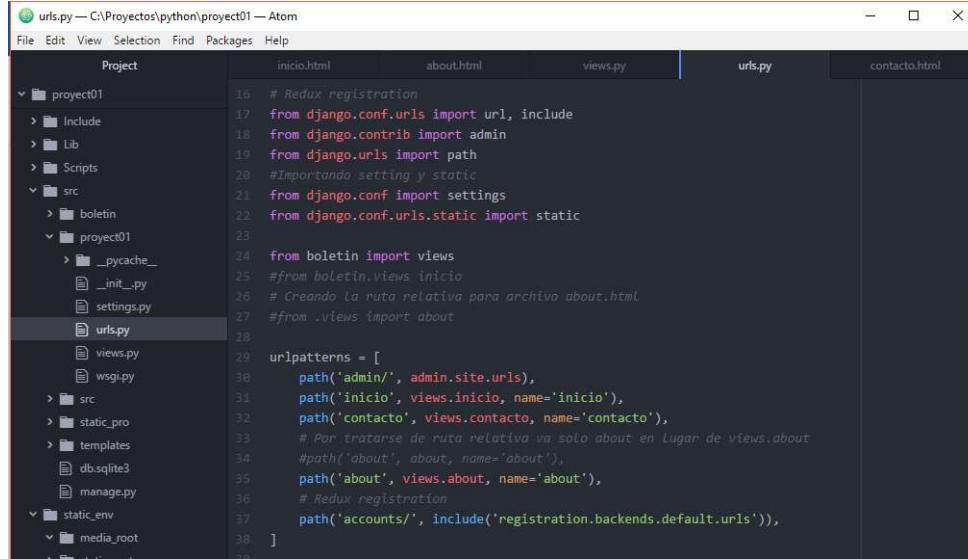
from boletin import views
#from boletin.views inicio
# Creando la ruta relativa para archivo about.html
#from .views import about

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
    # Por tratarse de ruta relativa va solo about en lugar de views.about
    #path('about', about, name='about'),
    path('about', views.about, name='about'),
    # Redux registration
    path('accounts/', include('registration.backends.default.urls')),
]

if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas

```

```
urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with a tree view of files and folders. The main editor area shows the content of the urls.py file. The code defines URL patterns for various views, including 'admin', 'inicio', 'contacto', and 'about'. It also includes imports for django.conf.urls, django.contrib.admin, and django.urls.static.

```
urls.py — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project inicio.html about.html views.py urls.py contacto.html
16 # Redux registration
17 from django.conf.urls import url, include
18 from django.contrib import admin
19 from django.urls import path
20 #Importando setting y static
21 from django.conf import settings
22 from django.conf.urls.static import static
23
24 from boletin import views
25 #from boletin.views inicio
26 # Creando la ruta relativa para archivo about.html
27 #from .views import about
28
29 urlpatterns = [
30     path('admin/', admin.site.urls),
31     path('inicio', views.inicio, name='inicio'),
32     path('contacto', views.contacto, name='contacto'),
33     # Por tratarse de ruta relativa va solo about en lugar de views.about
34     #path('about', about, name='about'),
35     path('about', views.about, name='about'),
36     # Redux registration
37     path('accounts/', include('registration.backends.default.urls')),
38 ]
```

#### Archivo about.html:

```
<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

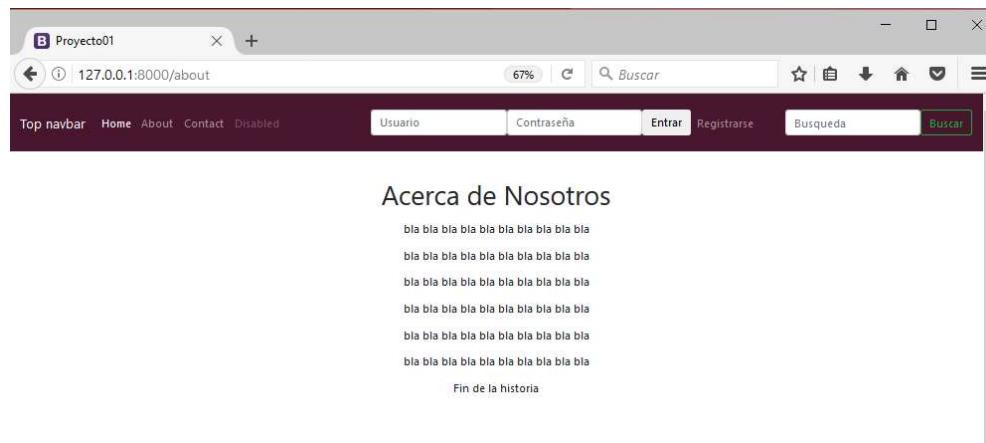
<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block content %}
<div class="row">
<div class="col-md-4"></div>
<div class="col-md-4">
<p class="lead">
<!-- Pasamos titulo como contexto de view contacto -->
<h1 class="text-center">Acerca de Nosotros</h1>
</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">bla bla bla bla bla bla bla bla</p>
<p class="text-center">Fin de la historia</p>
</div>
<div class="col-md-4"></div>
</div>
{% endblock %}
```

The screenshot shows the Atom code editor interface. On the left, the project tree displays a directory structure: 'proyecto01' containing 'Include', 'Lib', 'Scripts', 'src' (which contains 'boletin', 'project01', 'templates', and 'registration'), and 'static\_pro'. Inside 'src', there are files like '\_init\_.py', 'settings.py', 'urls.py', 'views.py', 'wsgi.py', and several HTML files including 'about.html', 'base.html', 'contacto.html', 'head\_css.html', 'inicio.html', 'javascript.html', 'jumbotron.html', and 'navbar.html'. The right pane shows the content of 'about.html'. The code is a template extending 'base.html' and defining a 'block content' block. It includes multiple paragraphs of placeholder text ('bla bla bla bla bla bla bla bla') and ends with a 'Fin de la historia' message.

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project inicio.html about.html views.py urls.py contacto.html
src
  proyecto01
    > Include
    > Lib
    > Scripts
    > src
      > boletin
      > proyecto01
        > __pycache__
        __init__.py
        settings.py
        urls.py
        views.py
        wsgi.py
      > src
      > static_pro
      > templates
      > registration
        about.html
        base.html
        contacto.html
        head_css.html
        inicio.html
        javascript.html
        jumbotron.html
        navbar.html
src\templates\about.html 26:15
CRLF UTF-8 HTML 0 files

```



Ejecutamos a nivel de comandos en la ruta **src**, para actualizar los archivos en las réplicas del servidor de producción.

D:\Proyectos\python\proyecto01\src>**python manage.py collectstatic**

The screenshot shows a Windows Command Prompt window titled 'Administrador: cmd'. The user is in the directory 'C:\Proyectos\python\proyecto01\src'. They run the command 'python manage.py collectstatic'. The output shows that the command is collecting static files to the destination location specified in the settings. It asks if the user wants to overwrite existing files, and the user types 'yes'. It then copies a file from 'C:\Proyectos\python\proyecto01\src\static\_pro\static\css\custom.css' to 'C:\Proyectos\python\proyecto01\static\_env\static\_root'. The process is completed successfully.

```

C:\ Proyectos\python\proyecto01\src>python manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:

C:\Proyectos\python\proyecto01\static_env\static_root

This will overwrite existing files!
Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes
Copying 'C:\Proyectos\python\proyecto01\src\static_pro\static\css\custom.css'

1 static file copied to 'C:\Proyectos\python\proyecto01\static_env\static_root', 130 unmodified.

C:\Proyectos\python\proyecto01\src>

```

## 38. Contenido para Usuarios Autenticados

- Permite que no se muestre el formulario de **Suscribirse** una vez estemos conectados dentro de la pagina, para esto agregamos el siguiente código al archivo inicio.html

Archivo **inicio.html**:

```

<!DOCTYPE html>
<!-- Es una extension de base.html -->
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form -->
{% load crispy_forms_tags %}

<!-- Creamos un block para descripcion del sitio -->
{% block titulo_sitio %}Bienvenidos | {{ block.super }}{% endblock %}

<!-- Creamos un block llamado jumbotron el mismo que llamara a jumbotron.html -->
{% block jumbotron %}
{% include "jumbotron.html" %}
{% endblock %}

{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

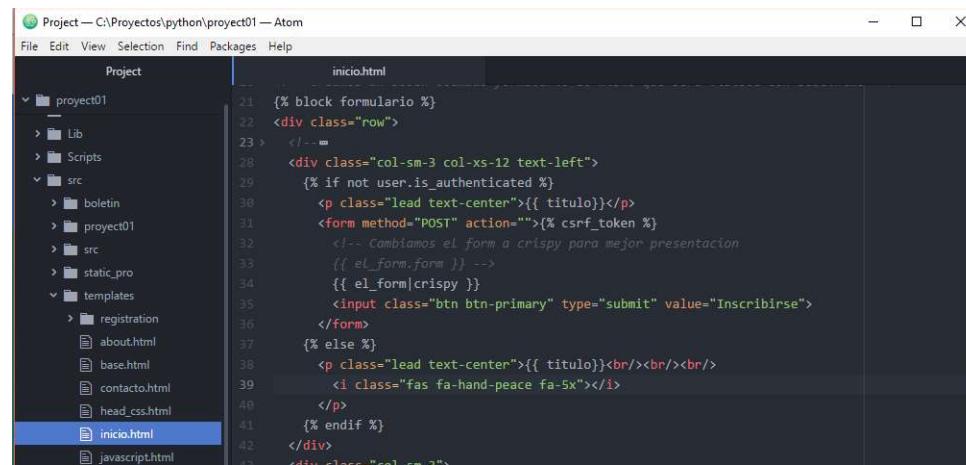
<!-- Creamos un block llamado formulario el mismo que será visible con base.html -->
{% block formulario %}
<div class="row">
<!--
  la clase col-sm-3 es de crispy-form numero de columnas 3
  la clase col-xs-12 es de crispy-form ajusta al comprimir la página
  la clase pull-right es de crispy-form alinea a la derecha
-->
<div class="col-sm-3 col-xs-12 text-left">
  {% if not user.is_authenticated %}
    <p class="lead text-center">{{ titulo }}</p>
    <form method="POST" action="">% csrf_token %
      <!-- Cambiamos el form a crispy para mejor presentacion
      {{ el_form.form }} -->
      {{ el_form|crispy }}
      <input class="btn btn-primary" type="submit" value="Inscribirse">
    </form>
  {% else %}
    <p class="lead text-center">{{ titulo }}<br/><br/><br/>
      <i class="fas fa-hand-peace fa-5x"></i>
    </p>
  {% endif %}
</div>
<div class="col-sm-3">

```

```

<p class="lead text-center">Pagina creado con framework Django.
<!-- ASOWESOME iconos animados -->
<span class="fa-stack fa-5x">
    <!-- <i class="fas fa-circle-notch fa-spin fa-2x"></i> -->
    <i class="fa fa-bullhorn fa-stack-1x"></i>
</span>
</p>
</div>
<div class="col-sm-3">
    <p class="lead text-center">Diseñado especialmente.
    <!-- ASOWESOME iconos animados -->
    <div><i class="fas fa-home fa-fw" style="background:MistyRose"></i> Home</div>
    <div><i class="fas fa-info fa-fw" style="background:MistyRose"></i> Info</div>
    <div><i class="fas fa-book fa-fw" style="background:MistyRose"></i> Library</div>
    <div><i class="fas fa-pencil-alt fa-fw" style="background:MistyRose"></i>
        Applications</div>
    <div><i class="fas fa-cog fa-fw" style="background:MistyRose"></i> Settings</div>
</p>
</div>
<div class="col-sm-3">
    <p class="lead text-center">Con codigo abierto.
    <!-- ASOWESOME iconos animados -->
    <span class="fa-stack fa-5x">
        <i class="fa fa-spinner fa-spin fa-stack-2x"></i>
        <i class="fa fa-camera-retro fa-stack-1x"></i>
    </span>
</p>
</div>
</div>
</div>
{% endblock %}

```



The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

- project01
  - Lib
  - Scripts
  - src
    - boletin
    - proyecto01
    - src
    - static\_pro
  - templates
    - registration
    - about.html
    - base.html
    - contacto.html
    - head\_css.html
    - inicio.html
    - javascript.html

The right pane shows the content of the `inicio.html` file:

```

21  {% block formulario %}
22  <div class="row">
23  <!-- =
24  <div class="col-sm-3 col-xs-12 text-left">
25  {% if not user.is_authenticated %}
26      <p class="lead text-center">{{ titulo }}</p>
27      <form method="POST" action="">{{ csrf_token }}
28          <!-- Cambiamos el form a crispy para mejor presentacion
29          {{ el_form.form }} -->
30          {{ el_form|crispy }}
31          <input class="btn btn-primary" type="submit" value="Inscribirse">
32      </form>
33  {% else %}
34      <p class="lead text-center">{{ titulo }}<br/><br/><br/>
35          <i class="fas fa-hand-peace fa-5x"></i>
36      </p>
37  {% endif %}
38  </div>
39  <div class="col-sm-2">

```



## 39. Introducción Básica a los Querysets

- Recupera datos desde la base de datos, leer a profundidad.

<https://docs.djangoproject.com/en/2.0/ref/models/querysets/>

Importamos el modelo del archivo `models.py` y agregamos el queryset al archivo `views.py`.

Podemos comprobarlo con la ejecución de estructura `for` del `queryset`, de la siguiente manera.

Archivo **views.py**:

```
#Importacion de configuracion para correo electronico
from django.conf import settings
from django.core.mail import send_mail
#importamos renderizar
from django.shortcuts import render
#Importamos el modelo de la base de datos llamado Registrado
from .models import Registrado
#Importamos el formulario
#from .forms import RegForm, RegModelForm
from .forms import RegForm, RegModelForm, ContactForm

#Importamos el modelo Registrado
from .models import Registrado
# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Bienvenidos"
    #Request que extraera variables del sistema
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }

    if form.is_valid():
        instance = form.save(commit=False)
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        # En caso de la descripcion de persona no se llene por formulario
        if not instance.nombre:
            instance.nombre = "Persona nueva"
        #Grabar instancia
        instance.save()
        #Envio de nuevo contexto con nombre
        context = {
            "titulo": "Gracias %s!" %(nombre)
        }
        #Envio de nuevo contexto con email, el nombre de persona esta vacio
        if not nombre:
            context = {
                "titulo": "Gracias %s" %(email)
```

```

        }
# form_data = form.cleaned_data
# var1 = form_data.get("email")
# var2 = form_data.get("nombre")
# obj = Registrado.objects.create(email=var1, nombre=var2)

# Si el usuario esta autenticado se podra mostrar el Queryset
if request.user.is_authenticated and request.user.is_staff:
    #Carga los datos hacia variable queryset
    queryset = Registrado.objects.all()
    #Comprobacion de los datos desde el queryset
    for instancia in queryset:
        print(instancia)
    context = {
        "queryset": queryset,
    }
    return render(request, "inicio.html", context)

def contacto(request):
    #Contexto
    titulo = "Contacto"

    form = ContactForm(request.POST or None)
    if form.is_valid():
        form_nombre = form.cleaned_data.get("nombre")
        form_email = form.cleaned_data.get("email")
        form_mensaje = form.cleaned_data.get("mensaje")
        asunto = "Form de Contacto"
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "elusuario@gmail.com"]
        email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje,
        form_email)
        send_mail(asunto,
                  email_mensaje,
                  email_from,
                  email_to,
                  fail_silently=False,
                  )
    #Forma 3 de llenar
    #for key, value in form.cleaned_data.items():
    #    print(key, value)
    #Forma 2 de llenar
    #for key in form.cleaned_data:
    #    print(key)
    #    print(form.cleaned_data.get(key))
    #Forma 1 de llenar
    #nombre = form.cleaned_data.get("nombre")
    #email = form.cleaned_data.get("email")

```

```

#mensaje = form.cleaned_data.get("mensaje")
#print(nombre, email, mensaje)

context = {
    "titulo": titulo,
    "contacto": form,
}
return render(request, "contacto.html", context)

def about(request):
    return render(request, "about.html", {})

```

```

[05/Jun/2018 08:50:18] "GET /static/js/bootstrap.min.js HTTP/1.1" 304 0
[05/Jun/2018 08:50:18] "GET /static/js/jquery-3.3.1.slim.min.js HTTP/1.1" 304 0
[05/Jun/2018 08:50:43] "GET /about HTTP/1.1" 200 5185
[05/Jun/2018 08:50:43] "GET /static/js/fa-regular.js HTTP/1.1" 404 1794
[05/Jun/2018 08:50:43] "GET /inicio HTTP/1.1" 200 7200
[05/Jun/2018 08:50:43] "GET /static/js/static/fa-regular.js HTTP/1.1" 404 1794
Performing system checks...

System check identified no issues (0 silenced).
June 05, 2018 - 10:55:38
Django version 2.0.5, using settings 'proyect01.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
flormaria@hotmail.com
ivanromanv@hotmail.com
bruno@hotmail.com
bruno@hotmail.com
cualquiera@nuevo.edu
[05/Jun/2018 10:55:38] "GET /inicio HTTP/1.1" 200 7204
[05/Jun/2018 10:55:38] "GET /static/css/navbar-top.css HTTP/1.1" 304 0
[05/Jun/2018 10:55:38] "GET /static/css/custom.css HTTP/1.1" 304 0
[05/Jun/2018 10:55:38] "GET /static/js/static/fa-regular.js HTTP/1.1" 404 1794
flormaria@hotmail.com
ivanromanv@hotmail.com
bruno@hotmail.com
bruno@hotmail.com
cualquiera@nuevo.edu
[05/Jun/2018 10:55:43] "GET /inicio HTTP/1.1" 200 7204
[05/Jun/2018 10:55:43] "GET /static/js/static/fa-regular.js HTTP/1.1" 404 1794

```

Hasta aqui las pruebas.....

Finalmente, procederemos a crear un nuevo archivo llamado **listado.html**, el mismo mostrara el “**listado de los registros**”, luego configuraremos dentro de **views.py** y **urls.py** agregando codigo, asi mismo le crearemos una ruta link en **navbar.html** que lo mostrara solo si estamos conectados.

#### Archivo **views.py**:

```

#Importacion de configuracion para correo electronico
from django.conf import settings
from django.core.mail import send_mail
#importamos renderizar
from django.shortcuts import render
#Importamos el modelo de la base de datos llamado Registrado
from .models import Registrado
#Importamos el formulario
#from forms import RegForm, RegModelForm

```

```

from .forms import RegForm, RegModelForm, ContactForm

# Create your views here.
def inicio(request):
    #Contexto
    titulo = "Bienvenidos"
    #Request que extraera variables del sistem
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)

    form = RegModelForm(request.POST or None)
    #Movemos el contexto para que se ejecute el de abajo
    context = {
        #variable que cargara el informe contexto
        "titulo": titulo,
        "el_form": form,
    }

    if form.is_valid():
        instance = form.save(commit=False)
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        # En caso de la descripcion de persona no se llene por formulario
        if not instance.nombre:
            instance.nombre = "Persona nueva"
        #Grabar instancia
        instance.save()
        #Envio de nuevo contexto con nombre
        context = {
            "titulo": "Gracias %s!" %(nombre)
        }
        #Envio de nuevo contexto con email, el nombre de persona esta vacio
        if not nombre:
            context = {
                "titulo": "Gracias %s" %(email)
            }
        # form_data = form.cleaned_data
        # var1 = form_data.get("email")
        # var2 = form_data.get("nombre")
        # obj = Registrado.objects.create(email=var1, nombre=var2)

        # Si el usuario esta autenticado se podra mostrar el Queryset
        if request.user.is_authenticated and request.user.is_staff:
            #Carga todos los datos hacia variable queryset
            queryset = Registrado.objects.all()
            #Queryset con filtro ordenado
            #queryset = Registrado.objects.all().order_by("-timestamp")#.filter(nombre__iexact="iv")

```

```

#Comprobacion de los datos desde el queryset
for instancia in queryset:
    print(instancia.nombre, instancia.email, instancia.timestamp)
context = {
    "queryset": queryset,
}
return render(request, "inicio.html", context)

def contacto(request):
    #Contexto
    titulo = "Contacto"

    form = ContactForm(request.POST or None)
    if form.is_valid():
        form_nombre = form.cleaned_data.get("nombre")
        form_email = form.cleaned_data.get("email")
        form_mensaje = form.cleaned_data.get("mensaje")
        asunto = "Form de Contacto"
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "elusuario@gmail.com"]
        email_mensaje = "%s: %s enviado por %s" %(form_nombre, form_mensaje,
form_email)
        send_mail(asunto,
                  email_mensaje,
                  email_from,
                  email_to,
                  fail_silently=False,
        )
    #Forma 3 de llenar
    #for key, value in form.cleaned_data.items():
    #    print(key, value)
    #Forma 2 de llenar
    #for key in form.cleaned_data:
    #    print(key)
    #    print(form.cleaned_data.get(key))
    #Forma 1 de llenar
    #nombre = form.cleaned_data.get("nombre")
    #email = form.cleaned_data.get("email")
    #mensaje = form.cleaned_data.get("mensaje")
    #print(nombre, email, mensaje)

    context = {
        "titulo": titulo,
        "contacto": form,
    }
    return render(request, "contacto.html", context)

def about(request):

```

```

return render(request, "about.html", {})

def listado(request):
    # Si el usuario esta autenticado se podra mostrar el Queryset
    if request.user.is_authenticated and request.user.is_staff:#
        #Carga todos los datos hacia variable queryset
        #queryset = Registrado.objects.all()
        #Queryset con filtro ordenado
        #queryset = Registrado.objects.all().order_by("-timestamp")#.filter(nombre__iexact="iv")
        queryset = Registrado.objects.all().order_by("-timestamp")
        #.filter(nombre__iexact="iv")
        for instance in queryset:
            #print(instance)
            context = {
                "queryset": queryset,
            }
    else:
        return render(request, "listado.html", {})
    return render(request, "listado.html", context)

```

The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

- Project
- src
  - boletin
    - \_\_pycache\_\_
    - migrations
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - forms.py
    - models.py
    - tests.py
  - views.py

The main editor area shows the content of `views.py`:

```

80     email_from,
81     email_to,
82     fail_silently=False,
83 )
84 >     #Forma 3 de llenar=
85
86     context = {
87         "titulo": titulo,
88         "contacto": form,
89     }
90     return render(request, "contacto.html", context)
91
92 def about(request):
93     return render(request, "about.html", {})
94
95 def listado(request):
96     # Si el usuario esta autenticado se podra mostrar el Queryset
97     if request.user.is_authenticated and request.user.is_staff:#
98         #Carga todos los datos hacia variable queryset#
99         queryset = Registrado.objects.all().order_by("-timestamp")#.filter(nombre__iexact="iv")
100        for instance in queryset:
101            #print(instance)
102            context = {
103                "queryset": queryset,
104            }
105
106    else:
107        return render(request, "listado.html", {})
108    return render(request, "listado.html", context)

```

The status bar at the bottom indicates the file is `src\boletin\views.py` and has 66:14 lines.

#### Archivo urls.py:

```

# Redux registration
from django.conf.urls import url, include
from django.contrib import admin
from django.urls import path
#Importando setting y static
from django.conf import settings

```

```

from django.conf.urls.static import static

from boletin import views
#from boletin.views inicio
# Creando la ruta relativa para archivo about.html
#from .views import about

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inicio', views.inicio, name='inicio'),
    path('contacto', views.contacto, name='contacto'),
    # Por tratarse de ruta relativa va solo about en lugar de views.about
    #path('about', about, name='about'),
    path('about', views.about, name='about'),
    path('listado', views.listado, name='listado'),
    # Redux registration
    path('accounts/', include('registration.backends.default.urls')),
]
]

if settings.DEBUG:
    #Agrega las rutas en caso de ser estaticas
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

The screenshot shows the Atom code editor interface with the following details:

- Project Path:** C:\Proyectos\python\proyecto01
- Open Files:**
  - views.py
  - urls.py (highlighted)
  - inicio.html
  - listado.html
  - navbar.html
- Code Editor Content (urls.py):**

```

16 # Redux registration
17 from django.conf.urls import url, include
18 from django.contrib import admin
19 from django.urls import path
20 #Importando setting y static
21 from django.conf import settings
22 from django.conf.urls.static import static
23
24 from boletin import views
25 #from boletin.views inicio
26 # Creando la ruta relativa para archivo about.html
27 #from .views import about
28
29 urlpatterns = [
30     path('admin/', admin.site.urls),
31     path('inicio', views.inicio, name='inicio'),
32     path('contacto', views.contacto, name='contacto'),
33     # Por tratarse de ruta relativa va solo about en lugar de views.about
34     #path('about', about, name='about'),
35     path('about', views.about, name='about'),
36     path('listado', views.listado, name='listado'),
37     # Redux registration
38     path('accounts/', include('registration.backends.default.urls')),
39 ]
40
41 if settings.DEBUG:
42     #Agrega las rutas en caso de ser estaticas
43     urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
44     urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```
- Status Bar:** CRLF, UTF-8, Python, 0 files

#### Archivo navbar.py:

```

{% load static %}
<nav class="navbar navbar-expand-md navbar-dark bg-primary mb-4">

```

```

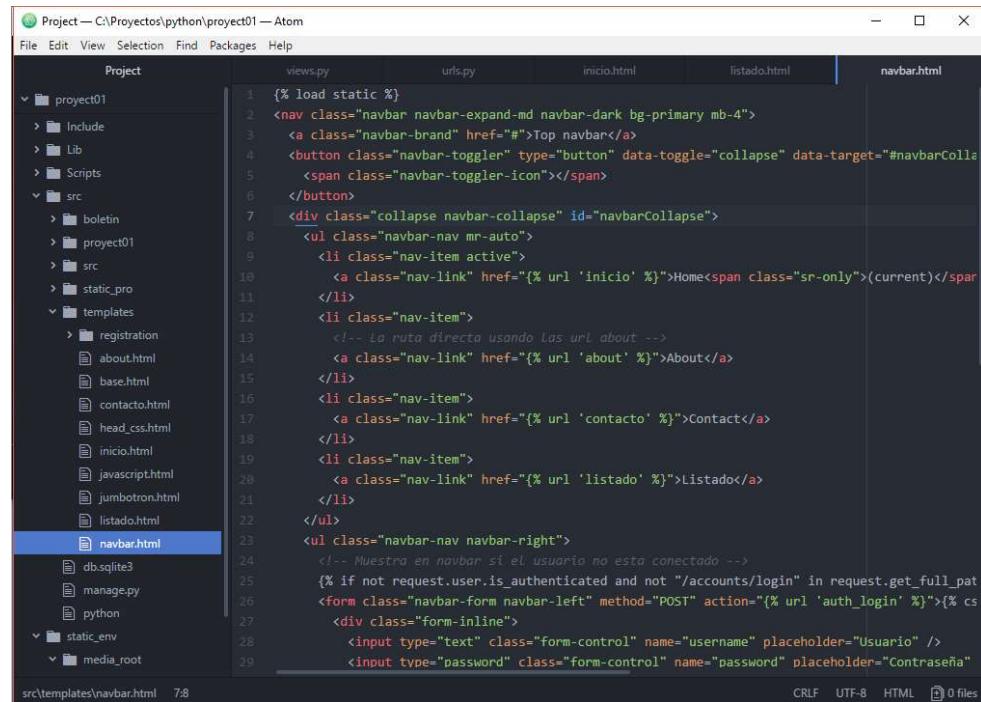
<a class="navbar-brand" href="#">Top navbar</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
  <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarCollapse">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
      <a class="nav-link" href="{% url 'inicio' %}">Home<span class="sr-
only">(current)</span></a>
    </li>
    <li class="nav-item">
      <!-- La ruta directa usando las url about -->
      <a class="nav-link" href="{% url 'about' %}">About</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'contacto' %}">Contact</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'listado' %}">Listado</a>
    </li>
  </ul>
  <ul class="navbar-nav navbar-right">
    <!-- Muestra en navbar si el usuario no esta conectado -->
    {% if not request.user.is_authenticated and not "/accounts/login" in
request.get_full_path %}
      <form class="navbar-form navbar-left" method="POST" action="{% url 'auth_login'
%}">{% csrf_token %}
        <div class="form-inline">
          <input type="text" class="form-control" name="username"
placeholder="Usuario" />
          <input type="password" class="form-control" name="password"
placeholder="Contraseña" />
          <button type="submit" class="btn btn-default">Entrar</button>
        </div>
      </form>
    {% endif %}
    <!-- Muestra en navbar si el usuario esta conectado -->
    {% if request.user.is_authenticated %}
      <li class="nav-item">
        <a class="nav-link" href="{% url 'auth_logout' %}">Salir</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/admin">Administracion</a>
      </li>
    {% else %}
      <li class="nav-item">

```

```

<a class="nav-link" href="{% url 'registration_register' %}">Registrarse</a>
</li>
{% endif %}
</ul>
<!-- Muestra en navbar para realizar busquedas dentro del sitio --&gt;
&lt;ul&gt;
&lt;form class="navbar-form navbar-left form-inline mt-md-3"&gt;
&lt;div class="form-inline"&gt;
&lt;input class="form-control" type="text" placeholder="Busqueda" aria-
label="Search"&gt;
&lt;button class="btn btn-outline-success" type="submit"&gt;Buscar&lt;/button&gt;
&lt;/div&gt;
&lt;/form&gt;
&lt;/ul&gt;
&lt;/div&gt;
&lt;/nav&gt;
</pre>

```



The screenshot shows the Atom code editor interface. On the left, the project tree displays a directory structure for 'proyecto01' containing 'Include', 'Lib', 'Scripts', 'src' (with subfolders 'boletin', 'proyecto01', 'src', 'static\_pro', and 'templates' which includes 'registration', 'about.html', 'base.html', 'contacto.html', 'head\_css.html', 'inicio.html', 'javascript.html', 'jumbotron.html', and 'listado.html'), 'static\_env', and 'media\_root'. The right pane shows the content of 'navbar.html'. The code is a template for a Bootstrap navbar, featuring a brand logo, a collapse button, and a navigation menu with links to 'Home', 'About', 'Contact', and 'Listado'. It also includes a search form and handles user authentication status.

```

Project — C:\Proyectos\python\proyecto01 — Atom
File Edit View Selection Find Packages Help
Project views.py urls.py inicio.html listado.html navbar.html
src
  templates
    registration
      about.html
      base.html
      contacto.html
      head_css.html
      inicio.html
      javascript.html
      jumbotron.html
      listado.html
      navbar.html
      db.sqlite3
      manage.py
      python
  static_env
  media_root
src\templates\navbar.html 7:8
1  {% load static %}
2  <nav class="navbar-expand-md navbar-dark bg-primary mb-4">
3    <a class="navbar-brand" href="#">Top navbar</a>
4    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse">
5      <span class="navbar-toggler-icon"></span>
6    </button>
7    <div class="collapse navbar-collapse" id="navbarCollapse">
8      <ul class="navbar-nav mr-auto">
9        <li class="nav-item active">
10          <a class="nav-link" href="{% url 'inicio' %}">Home<span class="sr-only">(current)</span>
11        </li>
12        <li class="nav-item">
13          <!-- La ruta directa usando las url about -->
14          <a class="nav-link" href="{% url 'about' %}">About</a>
15        </li>
16        <li class="nav-item">
17          <a class="nav-link" href="{% url 'contacto' %}">Contact</a>
18        </li>
19        <li class="nav-item">
20          <a class="nav-link" href="{% url 'listado' %}">Listado</a>
21        </li>
22      </ul>
23      <ul class="navbar-nav navbar-right">
24        <!-- Muestra en navbar si el usuario no esta conectado -->
25        {% if not request.user.is_authenticated and not "/accounts/login" in request.get_full_path %}
26        <form class="navbar-form navbar-left" method="POST" action="{% url 'auth_login' %}">{% csrf_token %}
27          <div class="form-inline">
28            <input type="text" class="form-control" name="username" placeholder="Usuario" />
29            <input type="password" class="form-control" name="password" placeholder="Contraseña" />

```

#### Archivo `listado.py`:

```

<!DOCTYPE html>
<!-- Es una extension de base.html --&gt;
{% extends "base.html" %}
<!-- Inserta justo AQUI la tag para crispy-form --&gt;
{% load crispy_forms_tags %}

<!-- Creamos un block llamado formulario el mismo que será visible con base.html --&gt;
{% block content %}
<!-- Los registros serán visibles si esta conectado al sistema --&gt;
</pre>

```

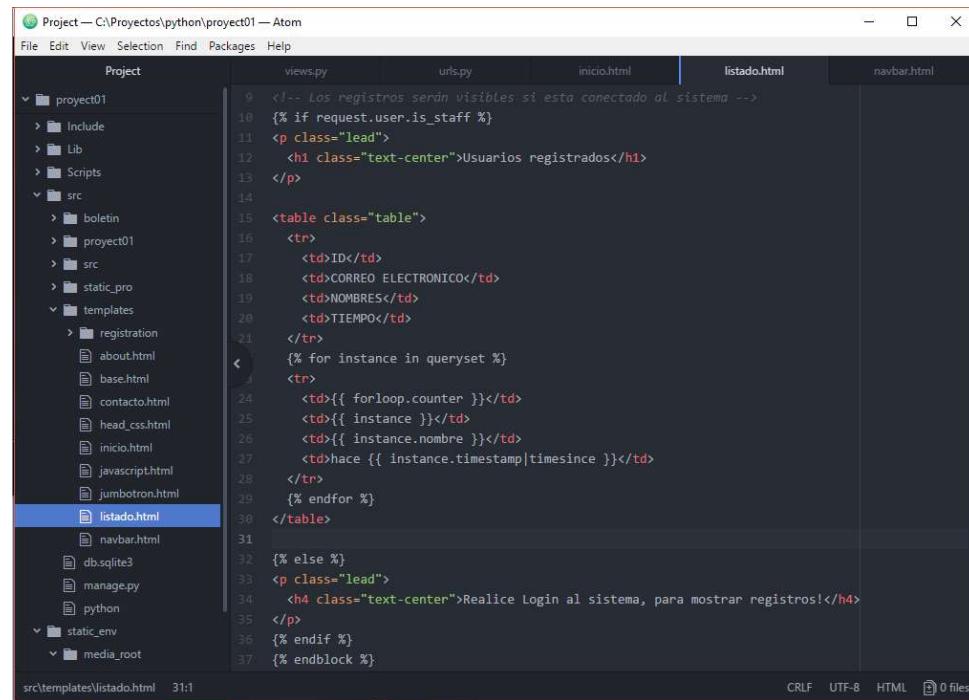
```

{% if request.user.is_staff %}
<p class="lead">
    <h1 class="text-center">Usuarios registrados</h1>
</p>

<table class="table">
    <tr>
        <td>ID</td>
        <td>CORREO ELECTRONICO</td>
        <td>NOMBRES</td>
        <td>TIEMPO</td>
    </tr>
    {% for instance in queryset %}
    <tr>
        <td>{{ forloop.counter }}</td>
        <td>{{ instance }}</td>
        <td>{{ instance.nombre }}</td>
        <td>hace {{ instance.timestamp|timesince }}</td>
    </tr>
    {% endfor %}
</table>

{% else %}
<p class="lead">
    <h4 class="text-center">Realice Login al sistema, para mostrar registros!</h4>
</p>
{% endif %}
{% endblock %}

```



The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

- Project — C:\Proyectos\python\project01 — Atom
- File Edit View Selection Find Packages Help
- Project
  - src
    - boletin
    - project01
    - src
    - static\_pro
    - templates
      - registration
        - about.html
        - base.html
        - contacto.html
        - head\_css.html
        - inicio.html
        - javascript.html
        - jumbotron.html
      - listado.html
      - navbar.html
    - views.py
    - urls.py
    - inicio.html
    - listado.html
    - navbar.html
  - static\_env
  - media\_root

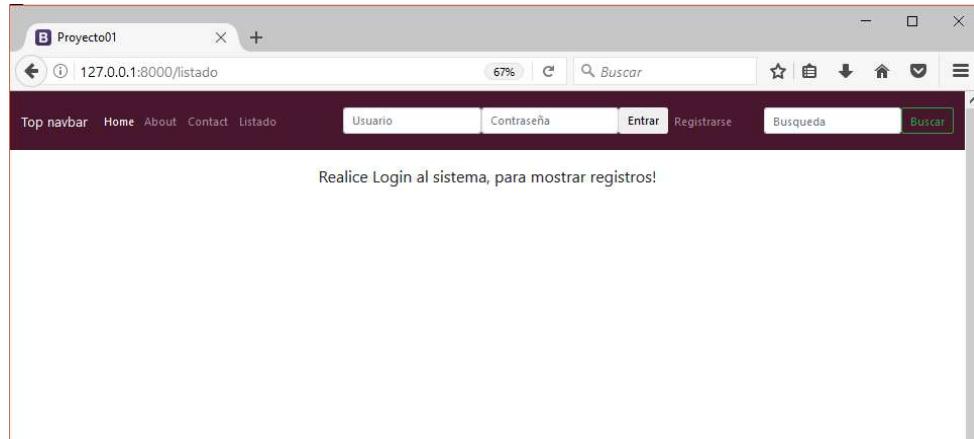
The right pane shows the content of the 'listado.html' file:

```

9     
10    {% if request.user.is_staff %}
11        <p class="lead">
12            <h1 class="text-center">Usuarios registrados</h1>
13        </p>
14
15        <table class="table">
16            <tr>
17                <td>ID</td>
18                <td>CORREO ELECTRONICO</td>
19                <td>NOMBRES</td>
20                <td>TIEMPO</td>
21            </tr>
22            {% for instance in queryset %}
23            <tr>
24                <td>{{ forloop.counter }}</td>
25                <td>{{ instance }}</td>
26                <td>{{ instance.nombre }}</td>
27                <td>hace {{ instance.timestamp|timesince }}</td>
28            </tr>
29            {% endfor %}
30        </table>
31
32    {% else %}
33        <p class="lead">
34            <h4 class="text-center">Realice Login al sistema, para mostrar registros!</h4>
35        </p>
36    {% endif %}
37    {% endblock %}

```

At the bottom, it shows the file path 'src\templates\listado.html' and line 31:1.



The screenshot shows a web browser window titled "Proyecto01" with the URL "127.0.0.1:8000/listado". The page has a dark purple header bar with the following navigation items: "Top navbar", "Home", "About", "Contact", "Listado", "Salir" (highlighted in red), "Administracion" (highlighted in green), and a search bar with "Busqueda" and "Buscar" buttons. Below the header, the title "Usuarios registrados" is displayed above a table. The table has columns: "ID", "CORREO ELECTRONICO", "NOMBRES", and "TIEMPO". The data is as follows:

ID	CORREO ELECTRONICO	NOMBRES	TIEMPO
1	cualquiera@nuevo.edu	Persona nueva	hace 1 semana, 5 días
2	bruno@hotmail.com	Bruno Roman	hace 1 semana, 6 días
3	bruno@hotmail.com	Bruno Roman	hace 1 semana, 6 días
4	ivanromanv@hotmail.com	Ivan Roman	hace 1 semana, 6 días
5	flormaria@hotmail.com	Flor Maria	hace 1 semana, 6 días