

Aplicación Web DJANGO

1. Lo primero que debemos de tener es instalado python3 y python2.
2. Tener instalado ensurepip desde la página de pip.readthedocs.org.
3. Y también instalaremos pip mediante “sudo easy_install pip”
4. Instalamos virtualenv con “pip” y también “getpip”.
5. Crearemos nuestro directorio para realizar el proyecto, en mi caso lo he llamado pd110.
6. Creamos el virtualenv y lo activamos mediante source bin/activate

```
> virtualenv .
created virtual environment CPython3.8.10.final.0-64 in 218ms
creator CPython3Posix(dest=/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/josema/.local/share/virtualenv)
added seed packages: pip==22.0.4, setuptools==62.1.0, wheel==0.37.1
activators BashActivator,CShellActivator,FishActivator,MushellActivator,PowerShellActivator,PythonActivator
```

7. Instalamos django en el directorio

```
> pip install django
Collecting django
  Using cached Django-4.0.4-py3-none-any.whl (8.0 MB)
Collecting asgiref<4, >=3.4.1
  Using cached asgiref-3.5.0-py3-none-any.whl (22 kB)
Collecting backports.zoneinfo
  Using cached backports.zoneinfo-0.2.1-cp38-cp38-manylinux1_x86_64.whl (74 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)
Installing collected packages: sqlparse, backports.zoneinfo, asgiref, django
Successfully installed asgiref-3.5.0 backports.zoneinfo-0.2.1 django-4.0.4 sqlparse-0.4.2
```

8. Empezaremos el proyecto

```
> python ./bin/django-admin startproject pd110
```

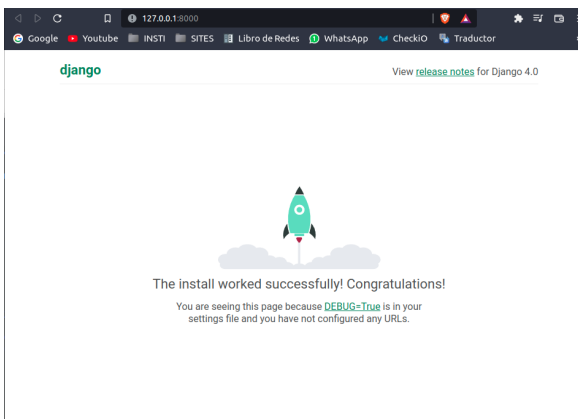
9. Podemos cambiar el nombre de dentro de nuestra raíz el cual tiene el mismo nombre, le he cambiado el nombre a src.

10. Iniciamos el servidor.

```
> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 17, 2022 - 20:43:25
Django version 4.0.4, using settings 'pd110.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[17/Apr/2022 20:43:29] "GET / HTTP/1.1" 200 10697
[17/Apr/2022 20:43:29] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[17/Apr/2022 20:43:29] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[17/Apr/2022 20:43:29] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[17/Apr/2022 20:43:29] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
Not Found: /favicon.ico
[17/Apr/2022 20:43:29] "GET /favicon.ico HTTP/1.1" 404 2109
```

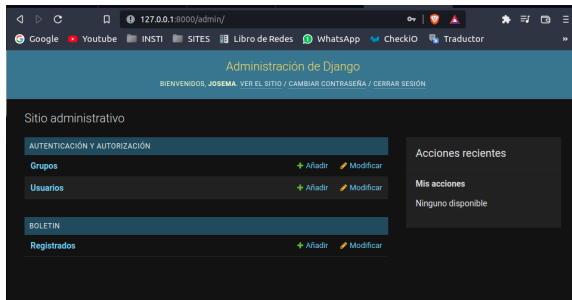
Veremos que si entramos al enlace de la terminal se nos abrirá el navegador con nuestra aplicación web en marcha.



11. Vamos a realizar migraciones en mi caso tuve que hacer python manage.py migrate

```
> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
```

12. Creamos un superusuario mediante python manage.py createsuperuser



13. Vamos a crear nuestra primera aplicación web.

```
> python manage.py startapp boletin
```

14. Añadimos en settings.py de nuestro primer proyecto la nueva app que acabamos de crear en el paso anterior en INSTALLED_APPS.

```
pd110 > src > pd110 > settings.py > ...
# SECURITY WARNING: keep the secret key used in production secret
SECRET_KEY = 'django-insecure-e#qxf2k8bmq(sm+7i89t@ya=24-j4_#3

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

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'boletin'
]
```

15. Escribimos nuestro primer modelo

```
pd110 > src > boletin > models.py > Registrados > __str__
1 from django.db import models
2
3 # Create your models here.
4
5
6 class Registrados(models.Model):
7     nombre = models.CharField(max_length=100, blank=True, null=True)
8     email = models.EmailField()
9     timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)
10
11     def __unicode__(self):
12         return self.email
13
14     def __str__(self):
15         return self.email
```

16. Hacemos las migraciones mediante python manage.py makemigrations

17. Mandamos las migraciones a nuestra base de datos mediante python manage.py migrate.

18. Creamos objetos en Python Shell.

```
>>> from boletin.models import Registrado
>>> gente = Registrado.objects.all()
>>> gente
<QuerySet []>
>>> personal1 = Registrado.objects.create(nombre="josema", email="josema@gmail.com")
>>> personal1
<Registrado: josema@gmail.com>
>>>
```

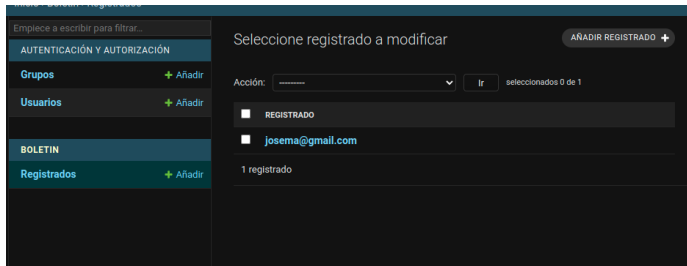
19. Registramos nuestro modelo en admin.py

```
src > boletin > admin.py
from django.contrib import admin

# Register your models here.
from .models import Registrado

admin.site.register(Registrado)
```

20. Ejecutamos nuevamente el servidor, nos metemos en registrados y veremos que está registrado el correo que pusimos anteriormente.



21. Vamos a proceder a personalizar el modelo en el admin.py

```
src > boletin > admin.py
from django.contrib import admin

# Register your models here.
from .models import Registrado

class AdminRegistrado(admin.ModelAdmin):
    list_display = ("email", "nombre", "timestamp")
    class Meta:
        model = Registrado

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

22. Añadimos filtros, enlaces y el poder editar el correo electrónico.

```
from django.contrib import admin

# Register your models here.
from .models import Registrado

class AdminRegistrado(admin.ModelAdmin):
    list_display = ("email", "nombre", "timestamp")
    #list_display_links = ["nombre"]
    list_filter = ("timestamp")
    list_editable = ["nombre"]
    search_fields = ("email", "nombre")
    class Meta:
        model = Registrado

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

23. Escribiremos nuestra primera vista

```
src > boletin > views.py
from django.shortcuts import render

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

24. Nos iremos ahora a pd100 en urls.py donde añadiremos, tendremos que copiar el ejemplo de arriba y poner el html que pusimos anteriormente. También importaremos el view.

```
'''
from django.contrib import admin
from django.urls import path
from boletin import views

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

25. Si intentamos acceder a nuestra página ahora nos saldrá obviamente que la página inicio no existe.
26. En el fichero settings en templates pondremos el directorio donde se va a encontrar nuestro fichero html

```
ROOT_URLCONF = 'pd110.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [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',
            ],
        },
    ],
]
```

27. Creamos una carpeta en src llamada templates donde guardaremos un fichero html llamado inicio el cual escribiremos un ejemplo y accederemos nuevamente a la página para ver el resultado.
28. Vamos a escribir nuestro primer formulario, por lo que en el directorio Boletín vamos a crear un fichero llamado forms.py donde crearemos una función de formulario donde importamos “forms”.

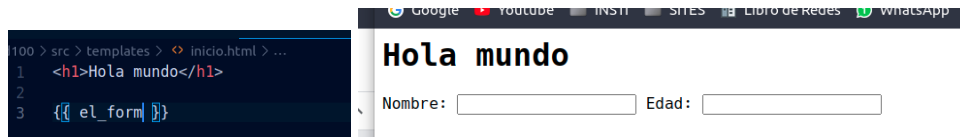
```
pd110 > src > boletin > forms.py > RegForm
1 from django import forms
2
3 class RegForm(forms.Form):
4     nombre = forms.CharField(max_length=100)
5     edad = forms.IntegerField()
```

29. Vamos a hacer un formulario en una vista, para ello editamos nuestra plantilla en views.py de boletin de esta forma añadiendo una nueva variable.

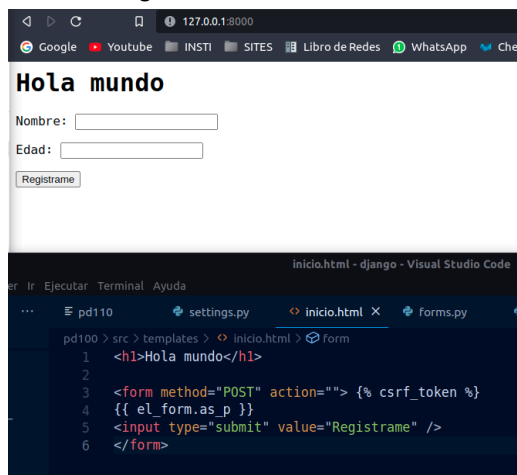
```
src > boletin > views.py > inicio
from django.shortcuts import render
from .forms import RegForm

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

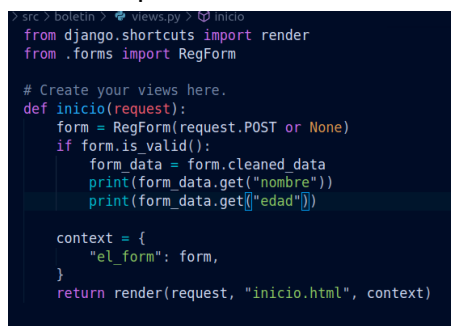
30. Nos vamos al fichero inicio.html y para importar nuestra función mediante {{ }} y dentro la variable que creamos volveremos a nuestra página y aparecerá el formulario.



31. Lo meteremos en la etiqueta form y añadiremos un botón que nos ponga regístrate.
32. Cambiamos a método post nuestra página para que cuando metamos datos no salgan en el enlace.



33. En views.py realizamos una sentencia if para que nos muestre el nombre y la edad limpios sin estar en un diccionario.



34. Vamos a guardar los datos del formulario, por lo que vamos a cambiar en forms la edad por el email.

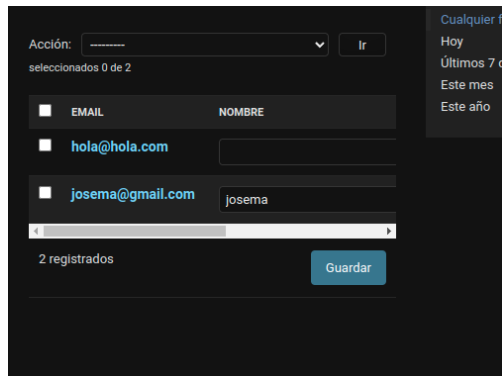


35. Editamos views.py de la siguiente manera.

```
> src > boletin > views.py > ...
from django.shortcuts import render
from .forms import RegForm
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
        abc = form_data.get("email")
        obj = Registrado.objects.create(email=abc)
    context = {
        "el_form": form,
    }
    return render(request, "inicio.html", context)
```

36. Si en el formulario ponemos el nombre y el correo, cuando vayamos a la página de admin en registrados, veremos que estará el correo pero el nombre no ya que el nombre es un campo obligatorio.



EMAIL	NOMBRE
<input type="checkbox"/> hola@hola.com	
<input type="checkbox"/> josema@gmail.com	josema

2 registrados

Guardar

37. Vamos a realizar el model form en el archivo forms.py importando la función model de Registrado y haremos la clase donde guardará los campos de email.

```
> src > boletin > forms.py > RegModelForm > Meta
from django import forms

from .models import Registrado

class RegModelForm(forms.ModelForm):
    class Meta:
        model = Registrado
        fields = ["email"]

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

38. Ya editamos el admin.py para importar la función forms de RegModel form y añadimos lo siguiente.

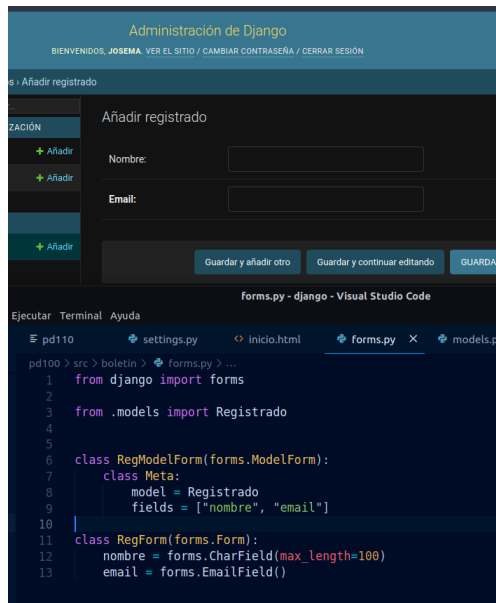
```
> src > boletin > admin.py > AdminRegistrado
from django.contrib import admin

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

class AdminRegistrado(admin.ModelAdmin):
    list_display = ["email", "nombre", "timestamp"]
    form = RegModelForm
    #list_display_links = ["nombre"]
    list_filter = ["timestamp"]
    list_editable = ["nombre"]
    search_fields = ["email", "nombre"]
    # class Meta:
    #     model = Registrado

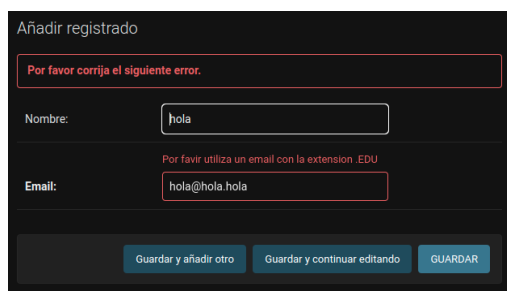
admin.site.register(Registrado, AdminRegistrado)
```

39. Editamos el forms.py para que nos salga el nombre y email en añadir registrado.



40. Vamos a realizar las validaciones del model Form, por lo que en forms.py vamos a definir una función que nos diga que si un email no termina en .edu no es válida.

```
def clean_email(self):
    email = self.cleaned_data.get("email")
    if not "edu" in email:
        raise forms.ValidationError("Por favor utiliza un email con la extension .EDU")
    return email
```



41. Y añadimos una función para limpiar el nombre.

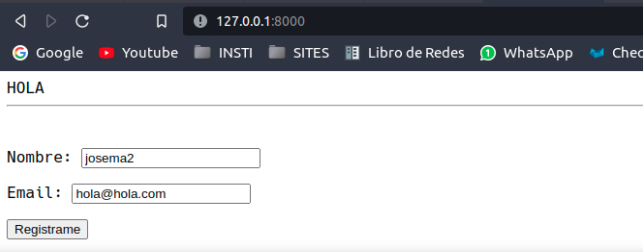
```
def clean_nombre(self):
    nombre = self.cleaned_data.get("nombre")
    #validaciones
    return nombre
```

42. Vamos a añadir un contexto en la vista de nuestra plantilla, en views.py vamos a añadir una nueva variable que nos sirva de título en la página del formulario.

```
src > Bolein > views.py > inicio
from django.shortcuts import render
from .forms import RegForm
from .models import Registrado

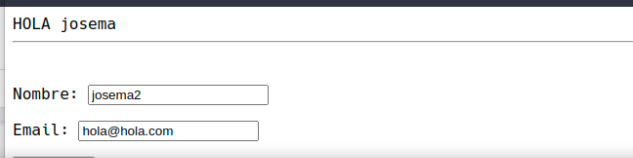
# Create your views here.
def inicio(request):
    titulo = "HOLA"
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
        abc = form_data.get("email")
        abc2 = form_data.get("nombre")
        obj = Registrado.objects.create(email=abc, nombre=abc2)

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



```
inicio.html - django - Visual Studio Code
pd100 > src > templates > inicio.html > br
1 <!-- <h1>Hola mundo</h1> -->
2 {{ titulo }}
3 <hr/>
4 <br/>
5
6 <form method="POST" action=""> {% csrf_token %}
7 {{ el_form.as_p }}
8 <input type="submit" value="Registrate" />
9 </form>
```

43. En el fichero html añadimos lo siguiente para que nos salga nuestro nombre en el título.



```
inicio.html - django - Visual Studio Code
Ver Ir Ejecutar Terminal Ayuda
pd100 > src > templates > inicio.html > form
1 <!-- <h1>Hola mundo</h1> -->
2 {{ titulo }}
3 {{ request.user }}
4
5 <hr/>
6 <br/>
7
8 <form method="POST" action=""> {% csrf_token %}
9 {{ el_form.as_p }}
10 <input type="submit" value="Registrate" />
11 </form>
```


44. Añadimos en views.py el import de RegModelForm donde comentaremos las líneas form_data, abc, abc2 y obj. Añadiremos una nueva línea de instancia que cuando pongamos nuestros datos en el formulario saldrá la instancia en la terminal.

```
from django.shortcuts import render
from .forms import RegForm, RegModelForm
from .models import Registrado

# Create your views here.
def inicio(request):
    titulo = "HOLA"
    form = RegModelForm(request.POST or None)
    if form.is_valid():
        instance = form.save(commit=False)
        print(instance)
        # form_data = form.cleaned_data
        # abc = form_data.get("email")
        # abc2 = form_data.get("nombre")
        # obj = Registrado.objects.create(email=abc, nombre=abc2)

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

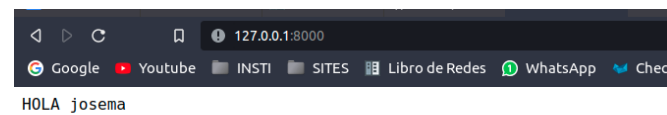
45. Añadimos instance.timestamp para que se guarde la hora que se creó el usuario.

```
# Create your views here.
def inicio(request):
    titulo = "HOLA"
    form = RegModelForm(request.POST or None)
    if form.is_valid():
        instance = form.save(commit=False)
        instance.save()
        print(instance)
        print(instance.timestamp)
        # form_data = form.cleaned_data
        # abc = form_data.get("email")
        # abc2 = form_data.get("nombre")
        # obj = Registrado.objects.create(email=abc, nombre=abc2)

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

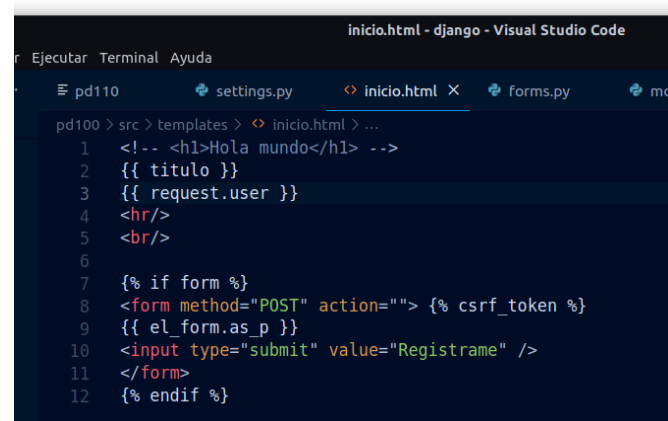
```
hola@hola.edu
2022-04-17 22:34:55.015386+00:00
[17/Apr/2022 22:34:55] "POST / HTTP/1.1" 200 583
```

46. Vamos a abrir inicio.html para añadir if form para quitar el botón.



127.0.0.1:8000

HOLA josema



```
inicio.html - django - Visual Studio Code

pd110 > src > templates > inicio.html > ...
1 <!-- <h1>Hola mundo</h1> -->
2 {{ titulo }}
3 {{ request.user }}
4 <hr/>
5 <br/>
6
7 {% if form %}
8 <form method="POST" action=""> {% csrf_token %}
9 {{ el_form.as_p }}
10 <input type="submit" value="Registrame" />
11 </form>
12 {% endif %}
```

47. Vamos a realizar un formulario de contacto. Para ello en forms.py donde teníamos RegForms lo cambiaremos por Contactform y en views donde importamos la función con el nombre anterior lo quitamos.

```
class ContactForm(forms.Form):
    nombre = forms.CharField()
    email = forms.EmailField()
    mensaje = forms.CharField(widget=forms.Textarea)
```

```
pd100 > src > boletin > views.py > contact
1 from django.shortcuts import render
2 from .forms import RegModelForm, ContactForm
3 from .models import Registrado
4
```

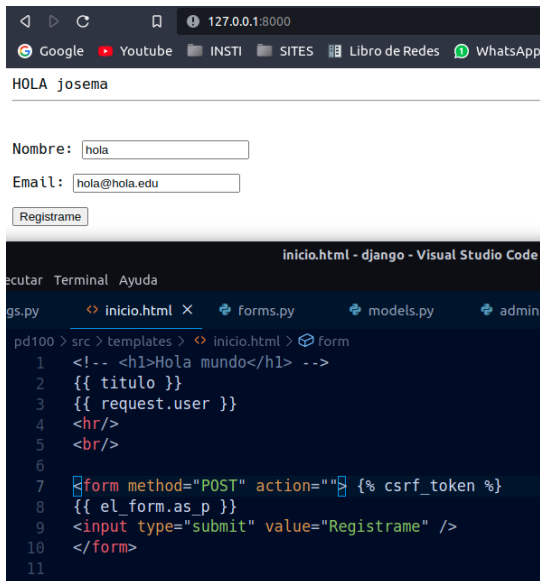
48. Crearemos una nueva función contact, donde crearemos un nuevo fichero html y lo crearemos igualmente en templates. Copiaremos el contenido de inicio.html en forms.html.

```
def contact(request):
    form = ContactForm(request.POST or None)
    context = {
        "form": form,
    }
    return render(request, "forms.html", context)
```

```
pd100 > src > templates > Forms.html > ...
1 {{ titulo }}
2 {{ request.user }}
3 <hr/>
4 <br/>
5
6 <form method="POST" action=""> {% csrf_token %}
7 {{ form.as_p }}
8 <input type="submit" value="Registrame" />
9 </form>
10
```

49. Añadiremos la url en urls.py del nuevo fichero html y quitamos el if de inicio para volver a tener el formulario.

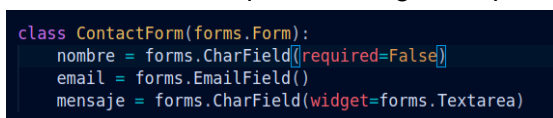
```
pd100 > src > pd110 > urls.py > ...
11 2. Add a URL to urlpatterns: path('...', name='inicio')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include
14 2. Add a URL to urlpatterns: path('blog/', include('django.contrib.sitemaps'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from boletin import views
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('contact/', views.contact, name='contact'),
23     path('', views.inicio, name='inicio')
24 ]
25
```



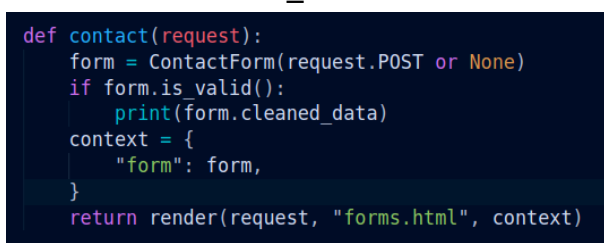
50. Nos metemos en la página de contact y veremos nuestro formulario.



51. Añadimos que sea obligatorio poner el email.

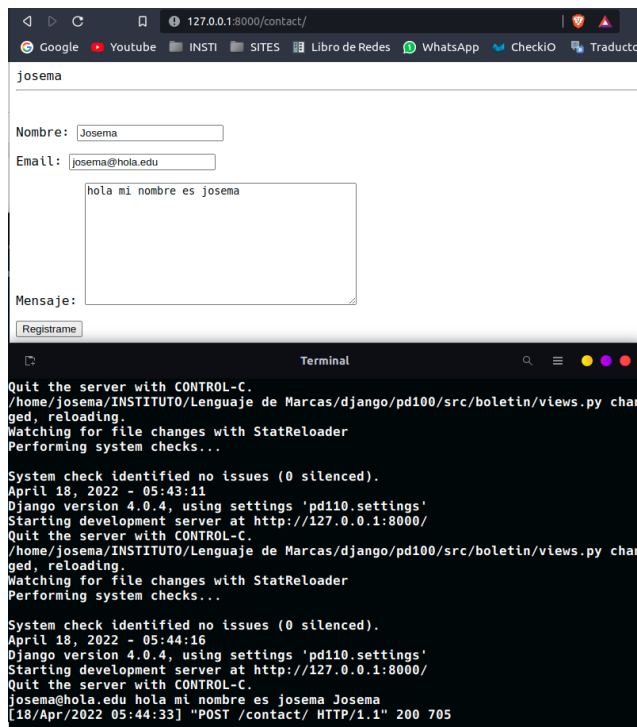


52. Añadimos en la función de contact en views.py que nos devuelva los datos limpios con el cleaned_data.



53. Añadiremos en la función contact las variables para que nos muestre en la terminal los datos que introducimos en el formulario.

```
def contact(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        email = form.cleaned_data.get("email")
        mensaje = form.cleaned_data.get("mensaje")
        nombre = form.cleaned_data.get("nombre")
        print(email, mensaje, nombre)
        context = {
            "form": form,
        }
    return render(request, "forms.html", context)
```



54. Vamos a configurar nuestro correo para poder enviar correos a nuestros usuarios. Para ello en settings.py añadimos los siguientes campos abajo de ALLOWED_HOSTS.

```
ALLOWED_HOSTS = []

EMAIL_HOST = "smtp.gmail.com"
EMAIL_HOST_USER = "tu_email@gmail.com"
EMAIL_HOST_PASSWORD = "tupassword"
EMAIL_PORT = 587
EMAIL_USER_TLS = True
```

55. En views.py importamos la función send_mail y los settings.

```
from django.conf import settings
from django.core.mail import send_mail
```

56. Ya rellenaremos en views la función de contact para que podamos enviar emails

```
def contact(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        form_email = form.cleaned_data.get("email")
        form_mensaje = form.cleaned_data.get("mensaje")
        form_nombre = form.cleaned_data.get("nombre")
        asunto = "Form de Contacto"
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "otroemail@gmail.com"]
        email_mensaje = "{}: {} enviado por {}".format(form_nombre, form_mensaje, form_email)
        send_mail(asunto,
                  email_mensaje,
                  email_from,
                  email_to,
                  fail_silently=False)
    )
```

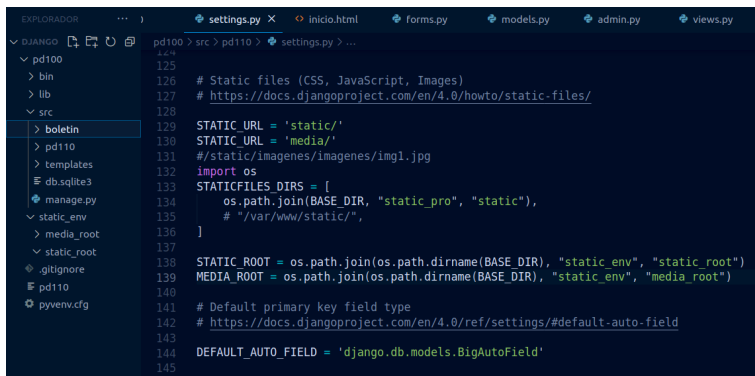
57. Configuraremos los archivos estáticos añadiendo lo siguiente en settings.py y añadiendo una nueva carpeta en la raíz llamada static_env

```
STATIC_URL = 'static/'
# /static/imagenes/imagenes/img1.jpg
import os
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static_pro", "static"),
    # "/var/www/static/",
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

58. Configuramos nuestro settings añadiendo archivos estáticos root y media y añadiendo en la carpeta static_env los directorios static_root y media_root.



```
129 STATIC_URL = 'static/'
130 STATIC_URL = 'media/'
131 # /static/imagenes/imagenes/img1.jpg
132 import os
133 STATICFILES_DIRS = [
134     os.path.join(BASE_DIR, "static_pro", "static"),
135     # "/var/www/static/",
136 ]
137
138 STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "static_root")
139 MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "media_root")
140
141 # Default primary key field type
142 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
143
144 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
145
```

59. Vamos a configurar nuestras url. Para ello en urls.py añadiremos el import que nos falta:

```
from django.conf.urls.static import static
```

60. Realizamos una sentencia if para el settings.DEBUG donde añadiremos el static_root y el media root. *TODO ESTO EN URL.PY*

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

61. Ahora ejecutaremos en la terminal el comando collectstatic

```
> python manage.py collectstatic
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory '/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/src/static_pro/static' in the STATICFILES_DIRS setting does not exist.

You have requested to collect static files at the destination
location as specified in your settings:

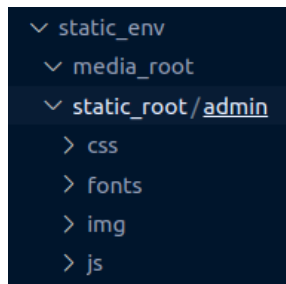
    /home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/static_env/static_root

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

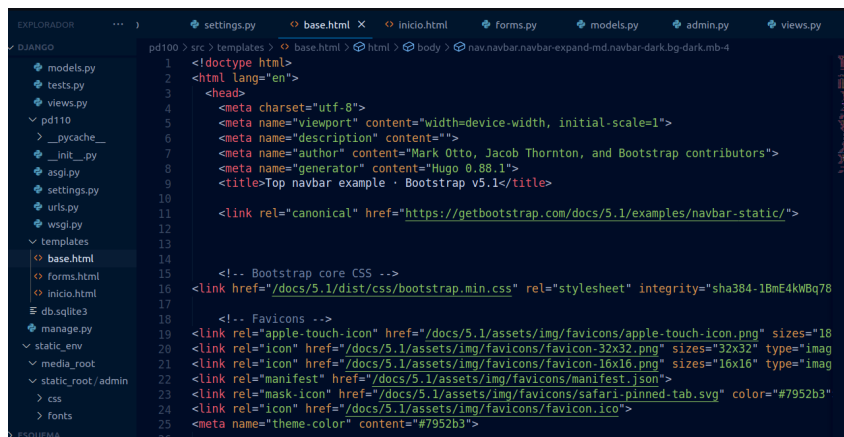
Type 'yes' to continue, or 'no' to cancel: yes

0 static files copied to '/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/static_env/static_root', 128 unmodified.
```

62. Se nos habrá creado los directorios dentro de static_root llamado admin con los siguientes directorios.



63. Vamos a realizar la configuración de bootstrap, por lo tanto en la página de bootstrap cogeremos una página de maqueta y crearemos en templates un fichero llamado base.html donde pegaremos el código fuente de dicha página.



64. Nos iremos a views.py y cambiaremos el render inicio.html por base.html e iremos a nuestra página donde veremos la maqueta.



65. En el código fuente descargamos el css de la página y lo pondremos en /src/static_pro/static y crearemos un directorio dentro llamado css. También crearemos los directorios en static de js e img.
66. En base.html pondremos el tag load static para renderizar nuestra imagen.

```
<!doctype html>
{% load static %}
<html lang="en">
```

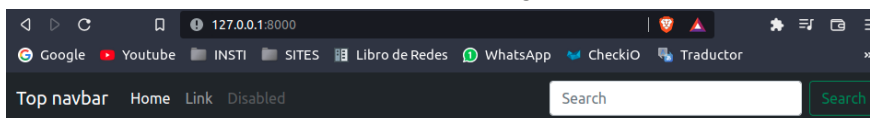
Redireccionaremos el enlace de base.html del css a nuestro fichero en la carpeta css poniéndole la etiqueta static

```
<!-- Bootstrap core CSS -->
<link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet" integrity="sha384-1BmE4kWBq78
<!-- Favicons -->
```

67. Realizamos lo mismo con Custom styles for this template descargando el fichero css y le ponemos la misma etiqueta que con el bootstrap.

```
<!-- Custom styles for this template -->
<link href="{% static 'navbar-top.css' %}" rel="stylesheet">
</head>
```

68. Cuando entremos en nuestra página nos saldrá tal que así.



Navbar example

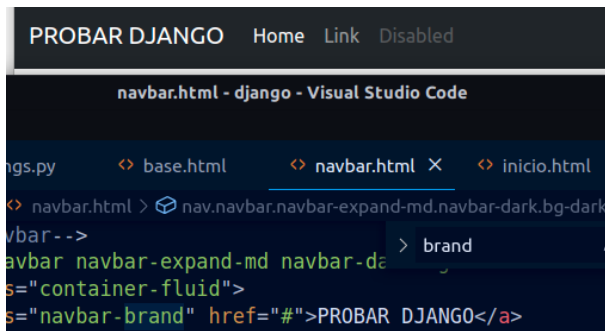
This example is a quick exercise to illustrate how the top-aligned navbar works. As you scroll, this navbar remains in its original position and moves with the rest of the page.

[View navbar docs »](#)

69. Realizamos un python manage.py collectstatic ya que hemos añadido ficheros css. Si nos vamos a static_env/static_root y en css veremos que están los ficheros css.
70. En el código fuente descargamos el fichero js y lo pegamos en static/js. Haremos nuevamente collectstatic para subir el fichero nuevo js y veremos que la página está como en el paso 73, habrá funcionado perfectamente.
71. Vamos a realizar ahora las plantillas con herencias, include tag y blocks. Crearemos en templates un fichero llamado navbar.html para la barra de navegación. Cogeremos del base.html toda la etiqueta nav y lo pegaremos en navbar.html y en base.html pondremos la etiqueta include. Veremos que la barra de navegación sale como si nada.

```
{% include "navbar.html" %}
```

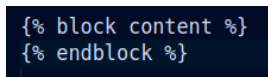
72. En navbar.html si tocamos navbar-brand y le ponemos un nombre se pondrá de título para la barra de navegación.



73. En el video nos muestra que copia la etiqueta jumbotron en inicio.html en mi código fuente no traía jumbotron. Lo añado tal y como tiene la que lo está mostrando.

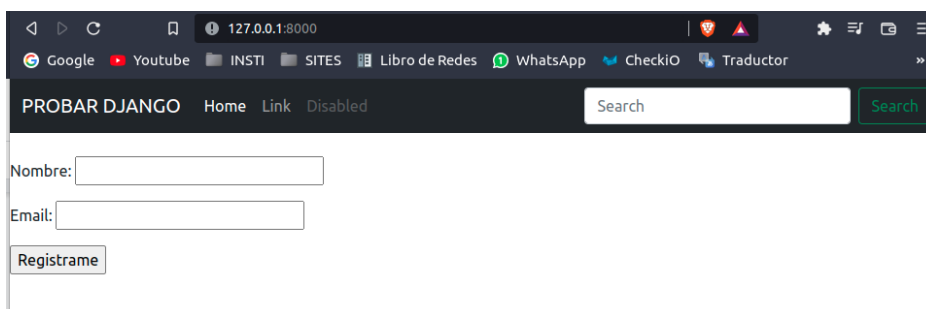
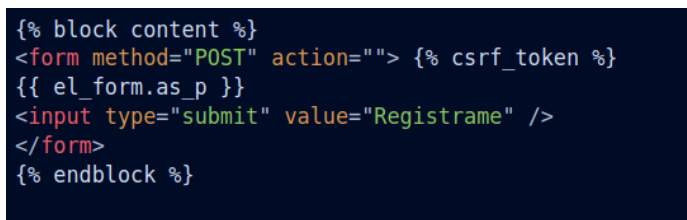


74. En base.html crearemos un bloque de contenido.



75. En views.py volvemos a cambiar el render a inicio nuevamente.

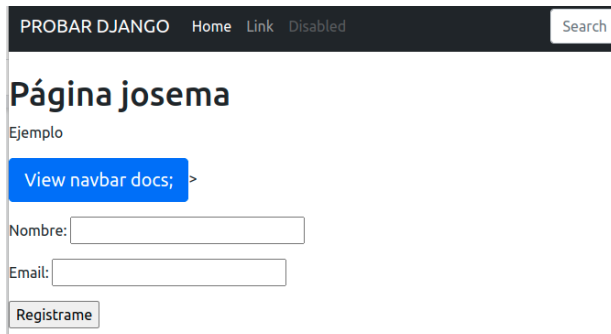
76. En inicio.html metemos en el bloque de contenido el método post y el botón de registrarse, una vez hecho esto y en base.html he tenido que quitar una clase para quitar el cuadro que salía de navbar example nos tendría que salir tal que así.



77. En base.html añadimos el bloque de jumbotron con la etiqueta de abrir y cierre. En inicio.html ponemos la misma etiqueta abriendo y cerrando el jumbotron quedando nuestra página así.

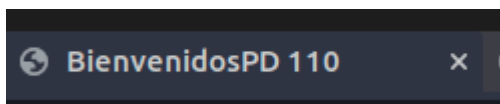
```
{% block jumbotron %}

<div class="jumbotron">
  <h1>Página josema</h1>
  <p>Ejemplo</p>
  <p>
    <a class="btn btn-lg btn-primary" href="../../component"
  </p>
</div>
{% endblock %}
```



78. Añadimos en base.html en el title el bloque head_title con su cierre y en inicio también lo añadimos cosa que en nuestra pestaña del navegador se verá lo que introducimos en ese bloque.

```
{% extends "base.html" %}
{% block head_title %}Bienvenidos{% endblock %}
{% block jumbotron %}
```



79. Seguimos limpiando el base.html copiando todo el css en un nuevo archivo en templates llamado head_css.html donde pondremos la etiqueta load_static y en base.html pondremos el include y el archivo css.

```
<!doctype html>
{% load static %}
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="Página de ejemplo de Django">
    <meta name="author" content="Mark O'Regan">
    <meta name="generator" content="Hugo">
    <title>{% block head_title %}{% endblock %}</title>

    {% include "head_css.html" %}
    <link rel="canonical" href="https://example.com/">

  </head>
  <body>

    {% include "navbar.html" %}
```

```
1 {% load static %}
2 <!-- Bootstrap core CSS -->
3 <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet" integrity="sha384-1BmE4kWBq78"
4
5
6 <!-- Favicon -->
7 <link rel="apple-touch-icon" href="/docs/5.1/assets/img/favicons/apple-touch-icon.png" sizes="180x180"
8 <link rel="icon" href="/docs/5.1/assets/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png"
9 <link rel="icon" href="/docs/5.1/assets/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png"
10 <link rel="manifest" href="/docs/5.1/assets/img/favicons/manifest.json">
11 <link rel="mask-icon" href="/docs/5.1/assets/img/favicons/safari-pinned-tab.svg" color="#7952b3">
12 <link rel="icon" href="/docs/5.1/assets/img/favicons/favicon.ico">
13 <meta name="theme-color" content="#7952b3">
14
15 <style>
16 .bd-placeholder-img {
17   font-size: 1.25rem;
18   text-align: center;
19   -webkit-user-select: none;
20   -moz-user-select: none;
21   user-select: none;
22 }
23
24 @media (min-width: 768px) {
25   .bd-placeholder-img-lg {
```

80. Cogemos el jumbotron de base.html donde crearemos un bloque nuevo dentro del div. Cortaremos el jumbotron en inicio dentro de la nueva etiqueta block jumbotron_content.

```
{% block head_title %}Bienvenidos | {{ block.super}}{% endblock %}
{% block jumbotron_content %}
    <h1>INICIO!!</h1>
    <p>Ejemplo</p>
    <p>
        <a class="btn btn-lg btn-primary" href="../../components/#navbar" role="button">View
    </p>
{% endblock %}
<!-- <h1>Hola mundo</h1> -->
```

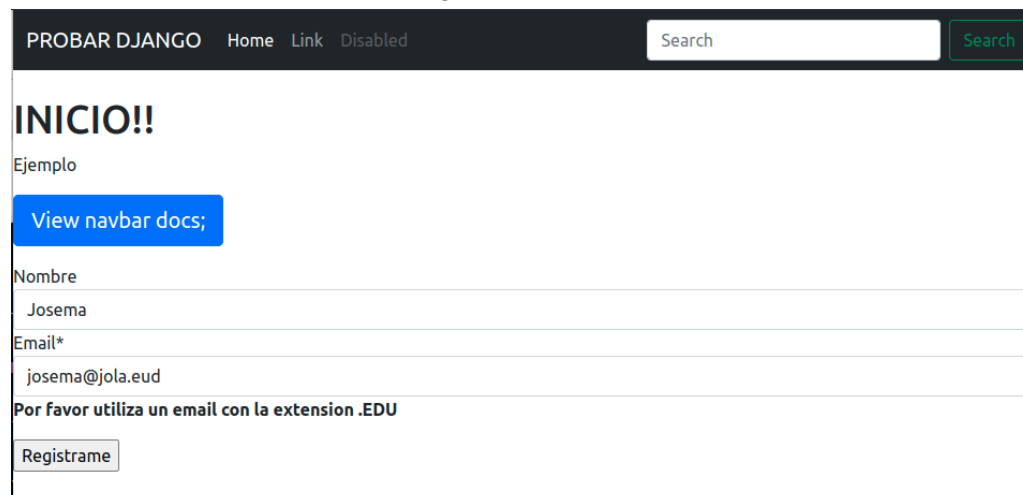
81. Ahora vamos a proceder a realizar django crispy forms. Actualizamos el django-crispy-forms mediante "pip install --upgrade django-crispy-forms"
82. Añadimos en settings en aplicaciones instaladas crispy_forms. Ahora migraremos mediante python manage.py migrate.
83. En settings.py añadimos lo siguiente:

```
CRISPY_TEMPLATE_PACK = 'bootstrap3'
```

84. Añadimos en inicio el tag crispy_forms_tags. Y en inicio el_form le agregamos crispy

```
{% block content %}
<form method="POST" action=""> {% csrf_token %}
{{ el_form|crispy }}
<input type="submit" value="Registrame" />
</form>
{% endblock %}
```

Que se verá el formulario de la siguiente manera.



PROBAR DJANGO Home Link Disabled Search Search

INICIO!!

Ejemplo

View navbar docs;

Nombre
Josema

Email*
josema@jola.eud

Por favor utiliza un email con la extension .EDU

Registrame

85. Mejoramos el estilo de bootstrap. Añadimos un container-fluid en el jumbotron de base.html y en navbar también añadimos container-fluid.

86. Editamos un poco nuestra página inicio para que quede de la siguiente forma.

PROBAR DJANGO

Home

Link

Disabled

Search

Search

Probar Django 1.10

Un proyecto para principiantes. El objetivo es contruir una página web simple a la par que elegante en muy poco tiempo

Únete »

Nombre

Josema

Email*

josema@jola.eud

Por favor utiliza un email con la extension .EDU

Regístrame

87. Editamos inicio.html añadiendo lo siguiente.

```
{% block content %}
<div class="row">
  <div class="col-sm-3 pull-right">
    <form method="POST" action=""> {% csrf_token %}
    {{ el_form|crispy }}
    <input class="btn btn-primary" type="submit" value="Regístrame" />
    </form>
  </div>
  <div class="col-sm-3">
    <p>Creado con Django & Bootstrap</p>
  </div>
  <div class="col-sm-3">
    <p>Y con mucho amor claro</p>
  </div>
  <div class="col-sm-3">
    <p>Código abierto, siempre.</p>
  </div>
</div>
<hr/>
{% endblock %}
```

PROBAR DJANGO

Home

Link

Disabled

Search

Search

Probar Django 1.10

Un proyecto para principiantes. El objetivo es contruir una página web simple a la par que elegante en muy poco tiempo

Únete »

Nombre

Josema

Created with Django & Bootstrap

Email*

josema@jola.eud

Y with much love clear

Código abierto, siempre.

Por favor utiliza un email con la extension .EDU

Regístrame

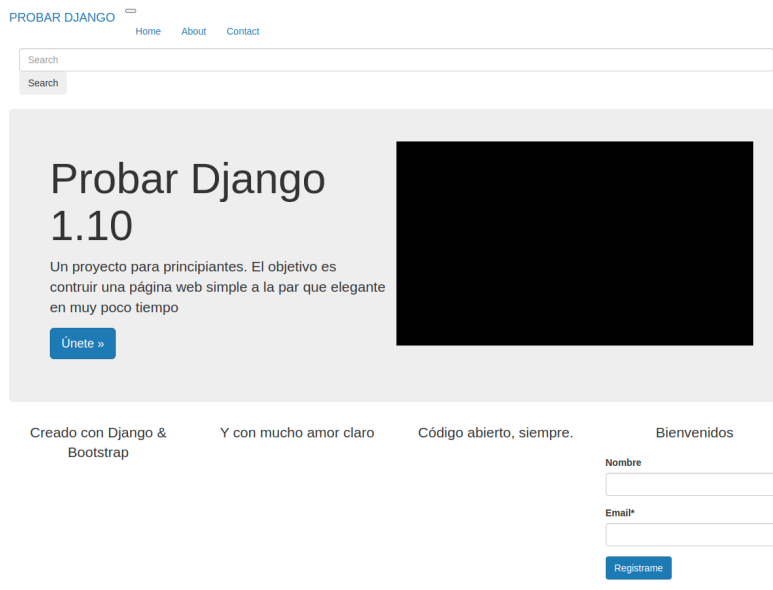
88. Editamos los parrafos para que cuando minimicemos la página salga correctamente el formulario uno encima de otro.

```
{% block content %}
<div class="row">
  <div class="col-sm-3 col-xs-12 pull-right">
    <p class="lead">{{ titulo }}</p>
    <form method="POST" action=""> {% csrf_token %}
      {{ el_form|crispy }}
    <input class="btn btn-primary" type="submit" value="Registrame" />
    </form>
  </div>
  <div class="col-sm-3">
    <p class="lead">Creado con Django & Bootstrap</p>
  </div>
  <div class="col-sm-3">
    <p class="lead">Y con mucho amor claro</p>
  </div>
  <div class="col-sm-3">
    <p class="lead">Código abierto, siempre.</p>
  </div>
</div>
<hr/>
{% endblock %}
```

Este es el resultado

89. Crearemos un fichero css customizado llamado custom.css donde crearemos nuevos estilo el cual añadiremos la ruta del fichero en head_css.html para que se apliquen los estilos quedando la página tal que así:

```
pd100 > src > static_pro > static > css > # custom.css > .text-align-center
1  .text-align-center {
2    text-align: center;
3  }
```



90. Al tocar los archivos estáticos haremos collectstatic para mandarlo a nuestro servidor.

91. Vamos a editar los enlaces con Nombres URL. Crearemos un nuevo fichero views.py en el directorio de /src/pd100 y pegaremos solo la funcion de inicio el cual lo llamaremos about y en urls.py añadiremos en urlpatterns la página about

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

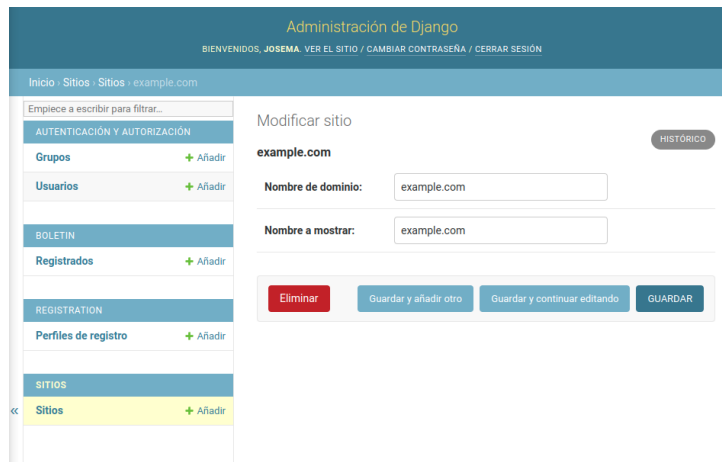
92. Añadimos las urls para que podamos acceder a about y contactos.

```
<!-- Static navbar -->  
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">  
  <div class="container">  
    <a class="navbar-brand" href="/">PROBAR DJANGO</a>  
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#na  
      <span class="navbar-toggler-icon"></span>  
    </button>  
    <div class="navbar-collapse collapse" id="navbar">  
      <ul class="nav navbar-nav">  
        <li><a href="/">Home</a></li>  
        <li><a href="{% url 'about' %}">About</a></li>  
        <li><a href="{% url 'contact' %}">Contact</a></li>  
      </ul>  
      <form class="d-flex">  
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search  
        <button class="btn btn-outline-success" type="submit">Search</button>  
      </form>  
    </div>  
  </div>  
</nav>
```


98. Añadiremos en settings.py la línea site_id=1 y en aplicaciones instaladas pondremos django.contrib.sites.

```
ACCOUNT_ACTIVATION_DAYS = 7
REGISTRATION_AUTO_LOGIN = True
SITE_ID = 1
```

99. Si entramos a la página de administrador podremos ver el dominio de nuestra web y podremos eliminar, guardar ,etc.



100. Al hacer migrate me salía un fallo de default_auto_field por lo que en settings.py he tenido que añadir una línea que ponga "DEFAULT_AUTO_FIELD = 'django.db.models.AutoField'"
101. En registration/activation_email veremos donde se podría poner el nombre de nuestro dominio con el activation key. Si nos registramos en nuestra página mediante /accounts/register.

Para tener todos los directorios de register he tenido que descargarlos de git hub

Registrarte Gratis!

Nombre de usuario*

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @/./+/_

Correo Electrónico*

Contraseña*

- 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.
- Su contraseña no puede ser completamente numérica.

Contraseña (confirmación)*

Para verificar, introduzca la misma contraseña anterior.

102. Nos saldrá error pero si nos vamos a perfiles de registro si nos saldrá como que está registrado.

```
SMTPAuthenticationError at /accounts/register/
(535, b'5.7.8 Username and Password not accepted. Learn more at\n5.7.8
https://support.google.com/mail/?p=BadCredentials f8-
20020a5d64c8000000b0020784359295sm13352708wri.54 - gsmtip')

Request Method: POST
Request URL: http://127.0.0.1:8000/accounts/register/
Django Version: 4.0.4
Exception Type: SMTPAuthenticationError
Exception Value: (535, b'5.7.8 Username and Password not accepted. Learn more at\n5.7.8
https://support.google.com/mail/?p=BadCredentials f8-20020a5d64c8000000b0020784359295sm13352708wri.54 - gsmtip')
Exception Location: /usr/lib/python3.8/smtp.py, line 655, in auth
Python Executable: /home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/bin/python
Python Version: 3.8.10
Python Path: ['/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/src',
'/usr/lib/python3.8.zip',
'/usr/lib/python3.8',
'/usr/lib/python3.8/lib-dynload',
'/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/lib/python3.8/site-packages']
Server time: Mon, 18 Apr 2022 11:32:54 +0000

Traceback: Switch to copy-and-paste view

/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/lib/python3.8/site-packages/django/core/handlers/exception.py, line 55, in inner
55. response = get_response(request)
└─ Local vars

/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/lib/python3.8/site-packages/django/core/handlers/base.py, line 197, in
_get_response
197. response = wrapped_callback(request, *callback_args, **callback_kwargs)
└─ Local vars

/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/lib/python3.8/site-packages/django/views/generic/base.py, line 84, in view
84. return self.dispatch(request, *args, **kwargs)
└─ Local vars

/home/josema/INSTITUTO/Lenguaje de Marcas/django/pd100/lib/python3.8/site-packages/django/utils/decorators.py, line 46, in _wrapper
46. return bound_method(*args, **kwargs)
```



103. No nos saldrá activado el usuario pero tendremos la clave de activación.

Registration information for Josema

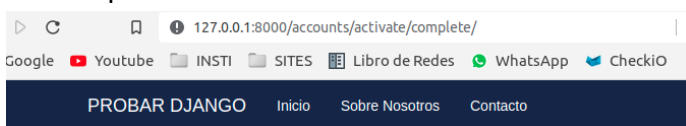
Usuario: [Josema](#)

Clave de activación:

☐ Activated

104. Para poder activar la cuenta tendremos que poner
`http://127.0.0.1:8000/accounts/activate/<clave de activación>`

105. Al activar su cuenta se activará la cuenta y podremos ver desde administrador que la cuenta estará activada.



Tu cuenta ha sido activada.

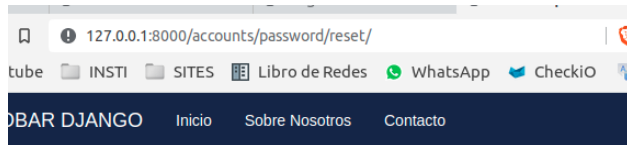
Veremos que estará activado

Usuario: [Josema](#)

Clave de activación:

☒ Activated

106. También si ponemos la url con /accounts/password/reset nos saldrá para poder restablecer la contraseña.



Restablecer Contraseña

Correo electrónico*

Enviar

107. Si intentamos iniciar sesión con el mismo nombre y correo nos saldrá que ya hay un usuario con el mismo nombre.

Registrarte Gratis!

Nombre de usuario*

Ya existe un usuario con ese nombre de usuario.

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @/./+/_

Correo Electrónico*

Contraseña*

- 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.
- Su contraseña no puede ser completamente numérica.

Contraseña (confirmación)*

Para verificar, introduzca la misma contraseña anterior.

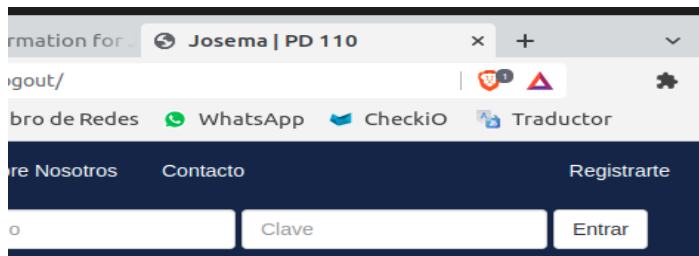
108. Vamos a configurar el Redirect para que al logear no nos lleve a otra página. Para ello añadiremos en settings.py la línea "LOGIN_REDIRECT_URL = '/'"
Con esto al loguearnos nos llevará a la página principal.

```
DEFAULT_AUTO_FIELD = 'django.db.models.AutoField'
```

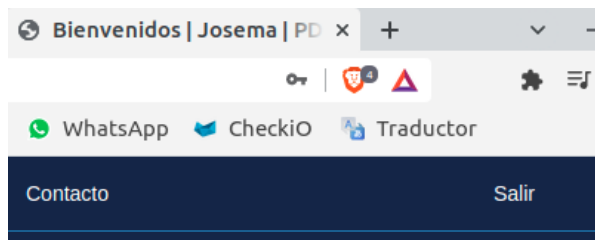
109. Vamos a realizar la autenticación de enlaces en navbar.html para que nos salga los botones de contacto como salir, entrar, registrarse... Cuando estemos logueados saldrá salir.

```
{% endif %}
</ul>
<div class='form-group'>
  <input type='password' class='form-control' name='password' placeholder='Clave' />
</div>
<button type='submit' class='btn btn-default'>Entrar</button>
</form>
{% endif %}
</div><!--/.nav-collapse -->
</div>
</nav>
```

Si no tenemos iniciada la sesión nos saldrá:



Una vez iniciado sesión nos saldrá:



110. Ahora haremos el formulario login para ello editaremos el fichero navbar.html para añadir nuestra clase para login:

```
<form class='navbar-form navbar-right' method='POST' action='{% url "auth_login" %}'>{% csrf_token %}
<div class='form-group'>
  <input type='text' class='form-control' name='username' placeholder='Usuario' />
</div>
```

111. Cuando nosotros nos logueamos se quitará el botón de entrar ya que estaremos dentro del usuario. Nos tendría que salir el botón de “salir”.

112. Con esta línea nos permite rellenar la contraseña sin que se vea:

```
<input type='password' class='form-control' name='password' placeholder='Clave' />
</div>
```



113. Añadiremos un parámetro para que cuando vayamos a /accounts/login no nos salga usuario y contraseña arriba a la derecha solo en la pantalla de inicio.

```
{% if not request.user.is_authenticated and not "/accounts/login" in request.get_full_path %}
```

114. En este paso aprenderemos a personalizar CSS en jumbotron y navbar, a ponerle un fondo del color que nosotros elijamos. En el inicio.html tendremos que añadir la etiqueta de style donde se aplican los estilos que nosotros pongamos.
115. Aprendemos a añadir un vídeo en la página de inicio sacado de youtube, para ello en inicio.html tendremos que crear una clase que tenga como atributo src con el enlace del video.

También si queremos añadir una imagen tendremos que crear un div con una etiqueta img con el atributo src con la dirección de la imagen que hayamos descargado, lo editaremos de tal forma que cuando minimicemos la página se adapte al contorno.

```
</div>
<div class='col-sm-6'><iframe width="560" height="315" src="https://www.youtube.com/embed/ChSvN5v_3aw" frameborder="0" allowfullscreen></iframe></div>
</div>
</div>
```

116. En este paso instalaremos fuentes para nuestra aplicación web desde la página fontawesome. En el fichero inicio.html añadiremos las etiquetas para aplicar las fuentes. También descargamos iconos para poder decorar un poco más nuestra aplicación web mediante la clase "i".

```
<div class='col-sm-3'>
  <p class='lead text-align-center'>Django &amp; Bootstrap.<br/><br/><span class="fa-stack fa-4x">
    <i class="fa fa-circle-o-notch fa-stack-2x"></i>
    <i class="fa fa-bullhorn fa-stack-1x" style="color: #47b78c;"></i></span>
  </div>
```

117. Vamos a aplicar contenido para los usuarios autenticados. Tendremos que insertar un bloque if para que cuando se autentique un usuario no salga para poner el usuario y la contraseña, en cambio nos salga Bienvenid@ y el usuario que se haya autenticado. Podemos añadir un icono para ocupar el espacio donde se encontraba el bloque de usuario y contraseña.

Creamos en templates el fichero about.html donde pondremos lo que queramos sobre la aplicación web.

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

{% block content %}
<div class="row">
  <div class="col-sm-6 col-sm-offset-3">
    <h2 class="text-align-center"><strong>Nuestra Historia</strong></h2><br/>
    <p class="text-align-center">Es una historia muy larga! Tienes tiempo? Te quedas un ratito para que te lo contemos?</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
    <p class="text-align-center">Blah blah blah blah blah blah</p>
  </div>
</div>
{% endblock %}
```

118. Ya en el último paso toqueteamos lo que viene siendo la introducción básica de los querysets. En views.py tendremos que importar la registrado de models e ir creando una función que nos muestre el email, nombre y la fecha de cuando inició sesión. La finalidad de los querysets es tener una página el cual se pueda ver un seguimiento de todos los usuarios que inician sesión, el nombre de usuario y correo que tiene cada uno.

```
if request.user.is_authenticated and request.user.is_staff:
    queryset = Registrado.objects.all().order_by("-timestamp")
    context = {
        "queryset": queryset,
    }
```