# Introduccion a Django

## python

Primero tenemos que descargar pyhton, en este caso uso pyhton 3.
Lo instalamos desde su repositorio.

```
apt install python3
```

```
franmola@ubuntu-cliente:~/Escritorio/karlita/src$ python3 --version
Python 3.8.10
```

Como vemos tenemos la version 3.8

## pip

Ahora instalamos tambien pip que es el administrador de paquetes, lo volvemos a hacer desde su repositorio.

## Virutalenv

Tambien tenemos que instalar el virutalenv donde instalaremos django.

Una vez descargado, creamos una carpeta llamada karlita y iniciamos el virtualenv dentro de la carpeta

Usamos los comandos correspondientes para activarlo.

```
franmola@ubuntu-cliente:~/Escritorio/karlita$ virtualenv karlita
created virtual environment CPython3.8.10.final.0-64 in 840ms
  creator CPython3Posix(dest=/home/franmola/Escritorio/karlita/karlita, clear=False, no_vcs_ignore=False, g
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/h
ualenv)
    added seed packages: pip==23.1.2, setuptools==67.7.2, wheel==0.40.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivat
franmola@ubuntu-cliente:~/Escritorio/karlita$ source bin/activate
(karlita) franmola@ubuntu-cliente:~/Escritorio/karlita$
```

## Django

Dentro del entorno virtual escribimos pip install Django, y se nos descargrá su ultima version.
Lo iniciamos con:

```
torio/karlita$ source bin/django-admin test_project
```

# Introduccion a Django

## Django

Probamos si funciona, hemos creado otra carpeta llamada src y es donde está.

```
python manage.py runserver
```

## superusuario

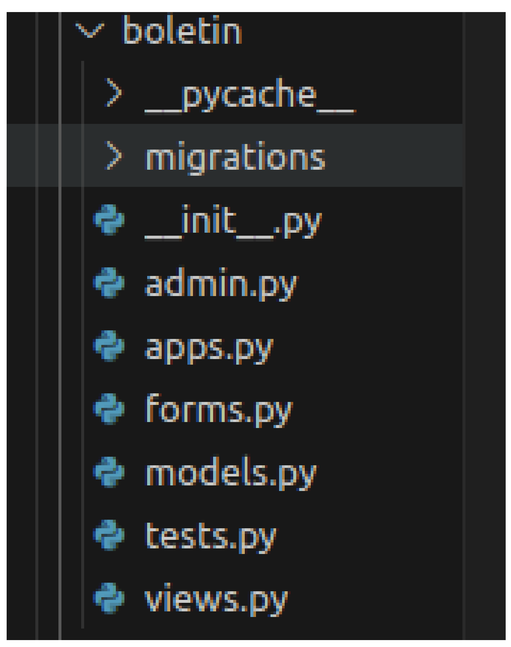
Ahora se entra al server http://127.0.0.1:8000/admin y nos indica un que añadamos un usuario, para ello nos volvemos a la terminal y creamos un superusuario para el servidor

```
C(karlita) franmola@ubuntu-cliente:~/Escritorio/karlita/test_project$ python manage.py createsuperu
sername (leave blank to use 'franmola'): franciscomola
mail address: franciscomanuel.monjo@gmail.com
assword:
assword (again):
his password is too short. It must contain at least 8 characters.
his password is entirely numeric.
ypass password validation and create user anyway? [y/N]: y
uperuser created successfully.
karlita) franmola@ubuntu-cliente:~/Escritorio/karlita/test_project$ python manage.py runserver
atching for file changes with StatReloader
erforming system checks...
```

Dentro podemos crear grupos, usuarios, controlar los permisos para usuarios etc.


## Aplicacion web

Crearemos la aplicación llamada boletin, nos vamos a la consola y lo creamos con pyhton manage.py startapp boletin, y se nos crea una carpeta con ese nombre con varios archivos dentros.




Nos dirigimos a settings.py y añadimos la aplicacion en su sitio correspondiente. Ademas aquí podemos cambiarle el idioma

```
# Application definition

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

# Introduccion a Django

## Primer modelo

Haciendo pyhton manage.py migrate/makemigration nos conectamos a la base de datos. cada vez que modificamos lo realizamos para mandar las modificaciones a la base de datos

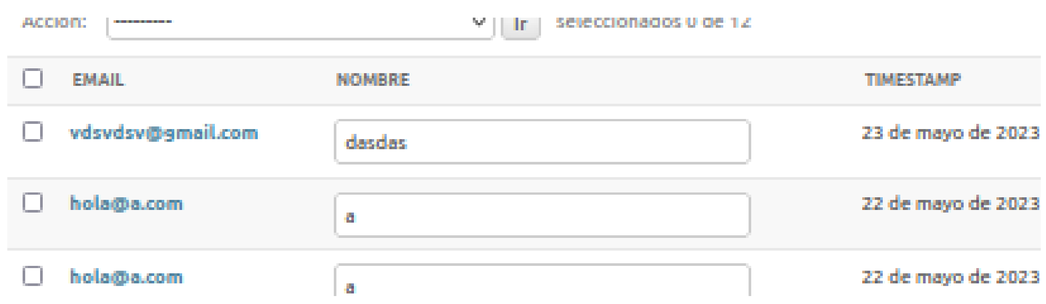
En el archivo models.py creamos nuestro modelo.

```python
from django.db import models

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

    def __str__(self):
        return self.email
```

Por defecto si no ponemos el auto_now, estará en True

## Admin

Personalizamos nuestro display del server en admin.py




```python
# src > boletin > admin.py
from django.contrib import admin
from .models import Registrados
from .forms import RegModelForm
class AdminRegistrados(admin.ModelAdmin):
    list_display = ["email", "nombre", "timestamp"]
    form = RegModelForm
    list_filter = ["timestamp"]
    list_editable = ["nombre"]
    search_fields = ["email", "nombre"]

admin.site.register(Registrados, AdminRegistrados)
```

## Primera vista

Creamos una funcion en views.py y añadimos la url en urls.py(importamos las views).

Creamos la plantilla inicio.html

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

## Templates

Creamos el directorio templates para poder tener ahí todas nuestras plantillas.

Añadimos la ubicación en settings.py

```python
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates'
        'DIRS': [
            BASE_DIR / 'templates',
        ],
        'APP_DIRS': True,
```

# Introduccion a Django

## forms.py

creamos un formulario que nos apetezca .

Nos dirigimos a incio.html y lo modificamos

```
<h1>Hola mundo</h1>

<form>
{{ el_form.as_p }}
<input type='submit' value='Registrame
</form>
```

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

## Metodo post

En inicio.html modificamos inicio.html y añadimos el metodo post y el token de django para la autentificacion.

```
<h1>Hola mundo</h1>

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

## Validacion formulario

en views modificamos para inlcuir los objetos nuevos

```
if form.is_valid():
    form_nombre = form.cleaned_data.get("nombre")
    form_email = form.cleaned_data.get("email")
```

```
def inicio(request):
    titulo = "Bienvenido"
    mensaje = None

    if request.user.is_authenticated:
        titulo = "Bienvenido %s" % request.user

    if request.method == "POST":
        form = RegModelForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            if not instance.nombre:
                instance.nombre = "persona"
            instance.save()
            mensaje = "Gracias por registrarte, %s!" % instance.nombre
    else:
        form = RegModelForm()

    context = {
        "title": titulo,
        "el_form": form,
        "mensaje": mensaje,
    }

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

Francisco Manuel Monjo Lancharro

# Introduccion a Django

## Model form

Usamos el formulario que trae django de la administracion.
Vamos al codigo forms.py e importamos el models de registrados, en admin añadimos form= RedModelForm, ademas de importar de nuevo aquí.
Con esto podemos añadir nuestras propias validaciones.

```
> src > boletin > admin.py
from django.contrib import admin
from .models import Registrados
from .forms import RegModelForm
class AdminRegistrados(admin.ModelAdmin):
    list_display = ["email", "nombre", "timestamp"]
    form = RegModelForm
    list_filter = ["timestamp"]
    list_editable = ["nombre"]
    search_fields = ["email", "nombre"]

admin.site.register(Registrados, AdminRegistrados)
```

```
class RegModelForm(forms.ModelForm):
    class Meta:
        model = Registrados
        fields = ["nombre", "email"]
```

## Validacion model form

Por lo tanto, nos dirigimos a forms.py y añadimos dos nuevas funciones, una para email y otra para nombre. y le añadimos nuestros requisitos que queremos que tengan.

```
def clean_email(self):
    email = self.cleaned_data.get("email")
    email_base, proveedor =email.split("@")
    dominio, extension= proveedor.split(".")
    if not extension == "com":
        raise forms.ValidationError("Use solo .com.")
    return email
def clean_nombre(self):
    nombre = self.cleaned_data.get("nombre")
    #para validar
    return nombre
```

## contexto plantilla

Añadimos titulo en el views.py y lo ponemos en incicio html como {{ titulo }}

```
def inicio(request):
    titulo = "Bienvenido"
    mensaje = None

    if request.user.is_authenticated:
        titulo = "Bienvenido %s" % request.user

    if request.method == "POST":
        form = RegModelForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            if not instance.nombre:
                instance.nombre = "persona"
            instance.save()
            mensaje = "Gracias por registrarte, %s!" % instance.nombre
    else:
        form = RegModelForm()

    context = {
        "title": titulo,
        "el_form": form,
        "mensaje": mensaje,
    }

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

# Introduccion a Django

## ModelForm en la vista

Con estas modificaciones estamos anadiendo el ModelForm a la vista, en lugar del formulario que teniamos antes, ademas de añadir un mensaje para que salga despues del registro

```
    form = RegModelForm(request.POST)
    if form.is_valid():
        instance = form.save(commit=False)
        if not instance.nombre:
            instance.nombre = "persona"
        instance.save()
        mensaje = "Gracias por registrarte, %s!" % instance.nombre
else:
    form = RegModelForm()
```

## custom form

```
def contact(request):
    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, 'franciscomanuel.monjo.lancharro.alu@iesferna
        email_mensaje = f"{form_nombre}: {form_mensaje} enviado por {form_ema

        send_mail(
            asunto,
            email_mensaje,
            email_from,
            email_to,
            fail_silently=True
        )

        print(form_email, form_mensaje, form_nombre)

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

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

El formulario que teniamos antes, lo reutilizamos para modificarlo y convertir lo en contacto.

En vista añadimosla funcion de contacto y tendremos que crear la url para contacto

```
path('contact/', views.contact, name='contact'),
```

## sendgrid

Con gmail ya no es posible si no se utiliza un tercero como sendgrid. Para la configuracion de sendgrid tenemos que crear una cuanta, verificarla y coger la api key. instalamos con pip install sendgrid.
Ahora en settings.py añadimos estos valores.

```
ALLOWED_HOSTS = []

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.sendgrid.net'
EMAIL_HOST_USER ='apikey'
EMAIL_HOST_PASSWORD ='SG.oNJoQ3_mQqeUiEpEqQdRfQ.3qBdfWPQcIfsHv9B9W55E5E
EMAIL_PORT= 587
EMAIL_USE_TLS = True
DEFAULT_FROM_EMAIL = 'franciscomanuel.monjo@gmail.com'
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
"""
Con gmail ya no es posible si no se utiliza un tercero como sendgrid
"""
```

# Introduccion a Django

## Archivos Estaticos

Tenemos que configurarlo para que funcione los archivos estáticos para desarrollo.
Todos estos se añade a settings.py

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

STATICFILES_DIRS = [
    BASE_DIR / "static_pro" / "static",
]

STATIC_ROOT = BASE_DIR.parent / 'static_env' / 'static_root'

MEDIA_ROOT = BASE_DIR.parent / 'static_env' / 'media_root'
```

Tenemos que crear todas estas carpetas en sus sitios correspondientes

static root es donde serán enviados nuestros archivos que estan es staticfiles_dirs

BASE.DIR.parent es la ruta padre del directorio

## Urls

Añadidos las irls y añadimos una condicion para que solo se use en desarrollo

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

```
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_RC
```

```
python manage.py collectstatic
```

Usamos el collectstatic en la terminal

Francisco Manuel Monjo Lancharro

# Introduccion a Django

## bootstrap

Vamos a la pagina de boostrap y buscamos una plantilla que nos guste por ejemplo esta: https://getbootstrap.com/docs/5.3/examples/navbar-static/ le damos click derecho ver codigo fuente y copiamos el codigo fuente en un archivo llamado base.html que pondremos en templates. Dentro de ese codigo tenemos varios codigos de css y javascripts que tambien tendremos que descargar y meter en unas carpetas que crearemos dentro de static_pro/statick llamadas css y js.

```
<link href="/docs/5.3/dist/css/bootstrap.min.css"

<script src="/docs/5.3/dist/js/bootstrap.bundle.min.js'

<head><script src="/docs/5.3/assets/js/color-modes.js"></script>

<link href="navbar-static.css" rel="stylesheet">
```

## configuracion

Para que las hojas de estilos carguen, tenemos que configurar y cargar los staticfiles.
Colocamos arriba del todo(base.html) esto:

```
<!doctype html>
{% load static %}
```

Ahora en cada link del codigo tendremos que cambiar la ruta y añadir la etiqueda static, por ejemplo así.

```
src="{% static 'js/color-modes.js' %}"></
```

```
<link href="{% static 'css/bootstrap.min.css' %}"
```

# Introduccion a Django

## Configuracion 2

Ahora modificaremos las plantillas bases y limpiaremos  base.html y lo pondremos en otras plantillas.

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

Pillamos cada trozo, por ejemplo navbar lo añadimos a navbar.html y en base le añadimos un include.

```
templates > navbar.html > nav.navbar.navbar-expand-md.navbar-dark.bg-dark.mb-4 > div.container-fluid > div#navbarCollapse.
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
    <div class="container-fluid">
      <a class="navbar-brand" href="/">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="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav me-auto mb-2 mb-md-0">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="/">Inicio</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="https://sites.google.com/iesfernandoaguilar.es/lmsgi">LMSG
```

## 2

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

{% load crispy_forms_tags %}

{% block head_title %}Bienvenidos | {% endblock %}
{% block jumbotron %}
<div class="bg-body-tertiary p-5 rounded">

<div class='row'>
    <div class="col-sm-6">
    <h1>Prueba con Django</h1>
    <p class="lead">Proyecto realizado por Francisco con el objetivo de probar y
    <a class="btn btn-lg btn-primary" href="/docs/5.3/components/navbar/" role="b
    </div>
    <div class="col-sm-6"><iframe src="MIPdf" width="650" height="400" style="bor
```

Tambien haremos que herede una parte del codigo una plantilla de otra.

Todo lo que queremos renderizar de inicio.html tenemos que meterlo en un bloque de contenido

```
<div class="container-fluid">
{% block jumbotron %}

{% endblock %}
```

## 3

Seguimos limpiando, esto lo metemos en head_css.html

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

Cuando se termine de limpiar, ya le cambiamos los titulos a cada cosa segun gusto de cada uno.

```
<link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet" integrity="sha384-KK94CHFLLe+


<style>
  .bd-placeholder-img {
    font-size: 1.125rem;
    text-anchor: middle;
    -webkit-user-select: none;
    -moz-user-select: none;
    user-select: none;
  }

  @media (min-width: 768px) {
    .bd-placeholder-img-lg {
      font-size: 3.5rem;
    }
  }
```

# Introduccion a Django

## Añadimos mejora de forms con crispy forms

En nuestra consola dentro del entorno virtual ponemos pip install --upgrade django-crispy-forms

Ahora añadimos en settings.py añadimos en apps las siguientes cosas.
(En este caso tuve que buscar que mas añadir porque en el video era antiguo y faltaban cosas)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    #apps de terceros
    'boletin',
    'crispy_forms',
    "crispy_bootstrap5",
```

Ahora por si acaso migramos con pyhton manage.py migrate.

Esto lo podemos añadir abajo del todo por ejemplo en settings.py

```
CRISPY_TEMPLATE_PACK = 'bootstrap5'
```

Añadimos y modificamos inicio.html para usar crispy y darnos un mejor aspecto.

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

{% load crispy_forms_tags %}
```

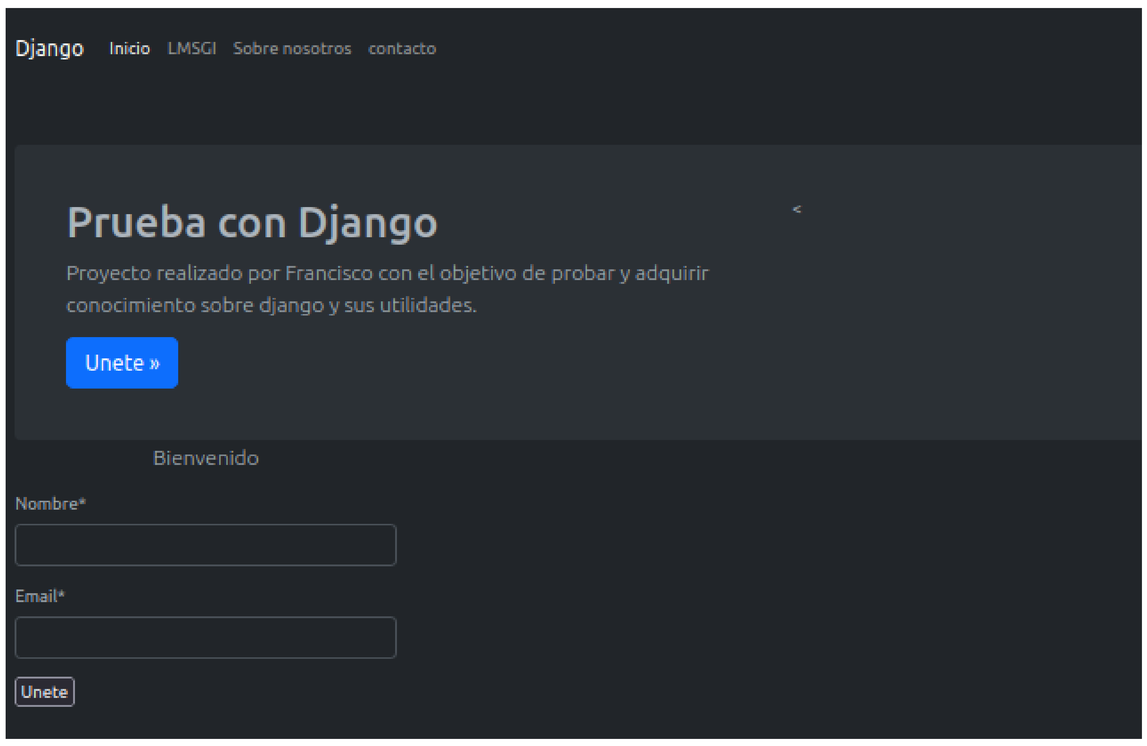
```
form method="POST" action
  {{ el_form|crispy }}
```

## Estilo con boostrap

Si añadimos -fluid en container, se estirará en toda la pantalla.

```
<main class="container-fluid">
```

Vamos añadiendo modificaciones en el inicio.html que nos parezca para dejarlo a nuestro gusto.



# Introduccion a Django

## Barra de navegacion

Vamos a agregar a la barra de navegacion de nuestra pagina una pagina de about, en el navbar la añadimos a igual de las que ya habia de ejemplos.

Gracias a que tenemos asignando nombres en las url en el navbar podemos añadirle el nombre y asi no pasa nada si cambiamos la url.

Esto lo añadimos a URLS.py

```
from .views import about
```

```
path('about/', about, name='about'),
```

Esto en views.py

```
from django.shortcuts import render


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

En el navbar.html

```
<li class="nav-item">
  <a class="nav-link" href="{% url 'about' %}">Sobre nosotros</a>
li>
```

## Contactos boostrap

Le añadimos los etilos que tenemos en base.html que metemos todos dentro del bloque de contenido. Como también vamos a usar crispy lo cargamos  también.

A destacar le hemos añadido un titulo, que estará situado en views.py

karlita > src > templates > form.html > ...

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

<h1>Contacto</h1>
<hr/>
<br/>
{% block content %}
<div class="row">
    <div class="col-sm-4 col-sm-offset-3">
{% if titulo %}
<h1>{{ titulo }}</h1>
{% endif %}
<form method="POST" action="">{% csrf_token %}
    {{ form|crispy }}
    <input class="btn-primary" type="submit" value="Enviar">
</form>
</div>
</div>
{% endblock %}
```

# Introduccion a Django

## Registration Redux 1

Lo que necesitamos es instalar la aplicacion con pip install django-registration-redux

Una vez instalada nos descargamos del github del tutorial los templates que necesitamos.

Ahora en settings tenemos que añadir varias configuraciones.

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

LOGIN_REDIRECT_URL = '/'
```

'django.contrib.sites,
'registration',

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    #apps de terceros
    'boletin',
    'crispy_forms',
    "crispy_bootstrap5",
    'registration',
```

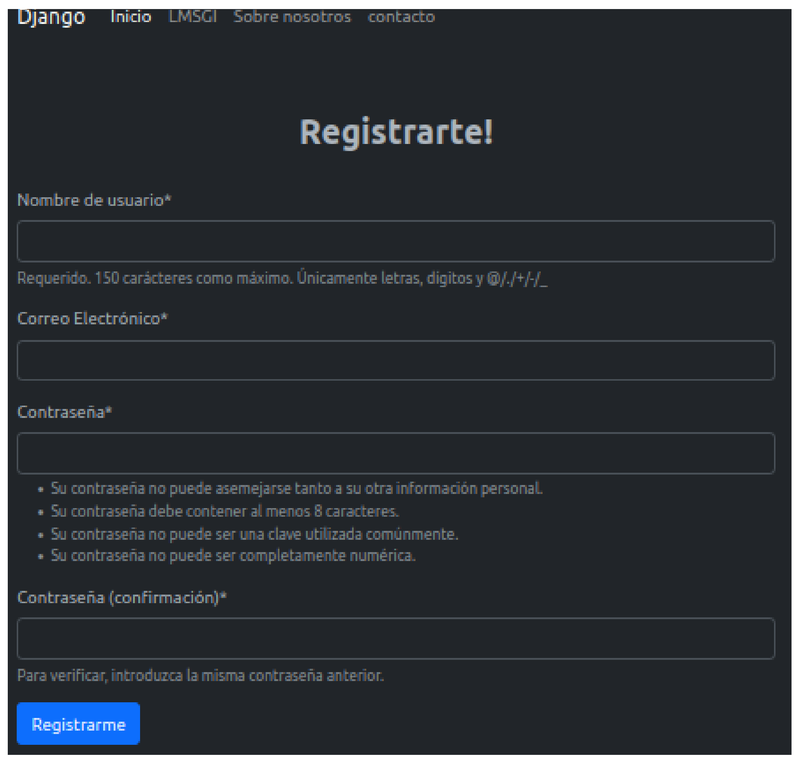
En urls.py

```
from django.urls import path, include
path('accounts/', include('registration.backends.default.urls'))
```

Con esto ya los usuarios podran entrar en registrar y login aunque aun queda configurarlo

## registration

Ahora modificamos las plantillas que descargamos del github para ponerle el mismo formato que anteriormente




Francisco Manuel Monjo Lancharro

# Introduccion a Django

## Registration Redux 2

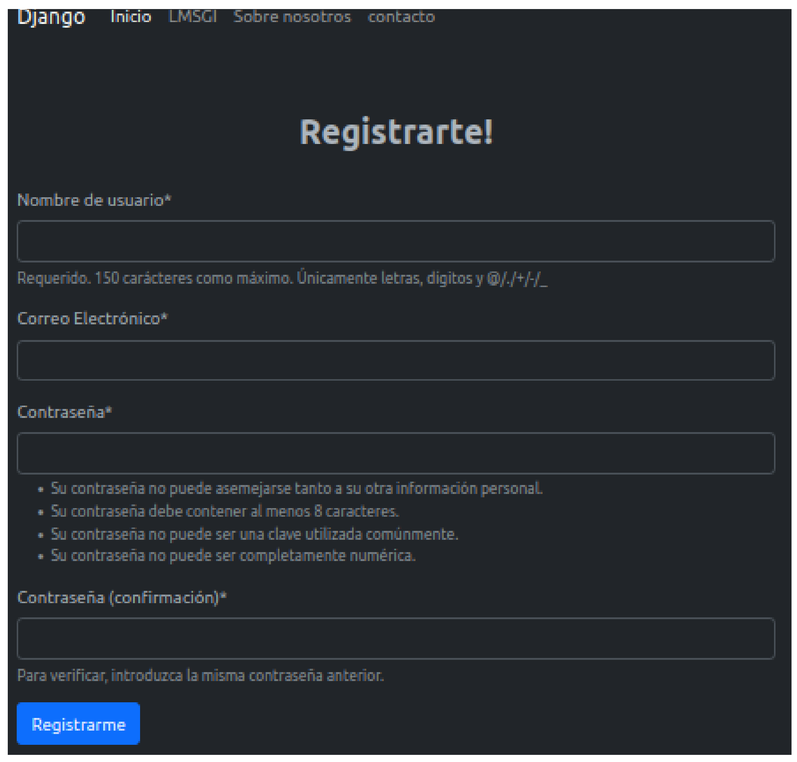
Ahora tenemos que configurar la activacion de cuentas.

```
{% load i18n %}

{% trans "Activa tu cuenta en" %} {{ site.name }}:

Hola,
Haz click en el enlace para activar tu cuena.
http://{{ site.domain }}{% url 'registration_activate' activation_key %}

{% blocktrans %}Enlace válido durante {{ expiration_days }} días.{% endblocktrans %}

-Team CFE
```

Con esto ya los usuarios podran entrar en registrar y login aunque aun queda configurarlo

## registration

Ahora modificamos las plantillas que descargamos del github para ponerle el mismo formato que anteriormente.
Hemos usado sendgrid para la validación del correo.
Cuando te registras te llega un correo electronico con lo necesario para hacerlo.



# Introduccion a Django

## Reedirigir

Hemos añadido una linea, para que cuando te loguees, te lleve de vuelta a la pagina de inicio

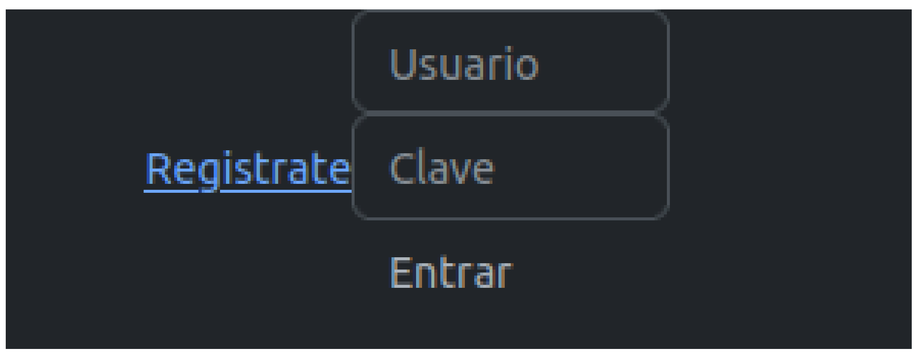
```
LOGIN_REDIRECT_URL = '/'
```

Ademas cuando estés registrado que no te salga de nuevo el menu de registro, si no, que detecte que ya estás autenticado.

```
{% if not request.user.is_authenticated and not "/accounts/login" in r
<form class='navbar-form navbar-right' method='POST' action='{% url "a
    <div class='form-group col-sm-5 pull-right'>
      <input type='text' class='form-control' name='username' placehol
    </div>
    <div class='form-group col-sm-5 pull-right'>
      <input type='password' class='form-control' name='password' plac
```

## Mini inicio

Hemos añadido un acceso rápido a loguearte sin tener que redirigirte a la pagina



# Introduccion a Django

## Añadir documento o video

Ahora en un hueco que hemos dejado a en la pagina, vamos a añadir este mismo documento, para que se pueda ver desde el inicio de la pagina web

```
<div class='row'>
    <div class="col-sm-6">
    <h1>Prueba con Django</h1>
    <p class="lead">Proyecto realizado por Francisco con el objetivo de probar
    <a class="btn btn-lg btn-primary" href="/docs/5.3/components/navbar/" role=
    </div>
    <div class="col-sm-6"><iframe src="MIPDF" width="650" height="400" style="b

    </div>
</div>
</div>
```

## Retoques esteticos

Si queremos hacer retoques esteticos a la pagina web podemos añadirles imagenes alguna pagina web, etc