Moises Serrano

Excellent

Contents

[NOMENCLATURA 3](#_Toc167208773)

[Project Python Django 4](#_Toc167208774)

[## 00 - Creation Project 4](#_Toc167208775)

[## 01 – Windows Prompt 4](#_Toc167208776)

[## 02 – Create virtual environment 4](#_Toc167208777)

[## 03 – python select interpreter 6](#_Toc167208778)

[## 04 - install Django 7](#_Toc167208779)

[## 05 – Creation Project 8](#_Toc167208780)

[## 06 – runserver 9](#_Toc167208781)

[## 07 – creation of application - tasks 10](#_Toc167208782)

[## 08 – Configuraciones Globales del projecto 11](#_Toc167208783)

[## 09 – Probar algo simple 11](#_Toc167208784)

[## 10 – Tambien puedo poner codigo html 13](#_Toc167208785)

[## 11 – Devolver un documento Entero 14](#_Toc167208786)

[## 12 – Formularios 15](#_Toc167208787)

[## 13 – Creation table Task 18](#_Toc167208788)

[## 14 – python manage.py makemigrations 19](#_Toc167208789)

[## 15 – python manage.py migrate 19](#_Toc167208790)

[## 16 – Como interactuar con la tabla 20](#_Toc167208791)

[## 17 – Panel administracion – admin 20](#_Toc167208792)

[#-- Creation Super Usuario 20](#_Toc167208793)

[--- Run server again: (venv) C:\Excellent>python manage.py runserver 20](#_Toc167208794)

[## 18 – connect como superuser to admin Panel 21](#_Toc167208795)

[## 19 – Refresh admin to reflect changes 21](#_Toc167208796)

[## 20 – agregar task desde admin 22](#_Toc167208797)

[## 21 – cambiar object information por columna 22](#_Toc167208798)

[## 22 – Quiero ver campos que son solo lectura en admin 24](#_Toc167208799)

[## 23 – Deseo crear tareas desde mi interfaz o aplicacion principal y no solo del administrador 25](#_Toc167208800)

[## 24 – Crear Formulario y anadir tareas 27](#_Toc167208801)

[## 26 – Campo oculto – protege Form: {% csrf\_token %} 29](#_Toc167208802)

[## 27 – Continuacion con create task – after save 30](#_Toc167208803)

[## 28 – save data a base de datos 32](#_Toc167208804)

[## 29 – Listar tareas – Todas (All) 36](#_Toc167208805)

[## 29 – Listar tareas con Filter 38](#_Toc167208806)

[## 30 – Listar tareas con mas Filter 39](#_Toc167208807)

[## 30 - Parentesis: DateCompleted optional 39](#_Toc167208808)

[## 31 – Podemos colocar condicionales dentro de ninja 40](#_Toc167208809)

[## 32 – Consultar detalles de una Tarea 40](#_Toc167208810)

[## 32 – Parentesis – problema con Task.object.get 42](#_Toc167208811)

[## 33 – Navegar rutas 43](#_Toc167208812)

[## 34 – Upadte base.html segun usuario authentificado 45](#_Toc167208813)

[## 35 – Editar/Actualizar una tarea 47](#_Toc167208814)

[## 36 – Crear su propio Formulario en task\_detail.html 48](#_Toc167208815)

[## 37 – Poner tarea Completada 49](#_Toc167208816)

[## 38 – Boton Delete 51](#_Toc167208817)

[## 39 – Listar Tareas Completadas 51](#_Toc167208818)

[## 40 – Ajustar la base 52](#_Toc167208819)

[## 41 – Proteger las rutas 53](#_Toc167208820)

[## 42 – Bootstrap 55](#_Toc167208821)

# NOMENCLATURA

## nn – Titulo de tema

#-- Nota especifica

Notas libres

# Project Python Django

## ## 00 - Creation Project

Drag folder Excellent to Visual Studio Code

A screenshot of a computer

Description automatically generated

## ## 01 – Windows Prompt

C:\>python --version

Python 3.12.3

## ## 02 – Create virtual environment

Me ubico dentro del folder del projecto

C:\Excellent>py -m venv venv

This create a structure from folder venv

A screenshot of a computer

Description automatically generated

## ## 03 – python select interpreter

FN + F1 or F1 depending of keyboard

Write down python select interpreter.

Select the version for venv that should be recommended.

El detecta que en venv hay un folder venv , un folder Script and python.exe

A screenshot of a computer

Description automatically generated

El terminal deberia reflejar : venv

Sino lo hace puedo jugar con comandos :

(venv) C:\Excellent>deactivate

C:\Excellent>.\venv\Scripts\activate

(venv) C:\Excellent>

## ## 04 - install Django

**Con esto django quedara instalado en el entorno**

(venv) C:\Excellent>pip install django

Collecting django

Using cached Django-5.0.6-py3-none-any.whl.metadata (4.1 kB)

Collecting asgiref<4,>=3.7.0 (from django)

Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)

Collecting sqlparse>=0.3.1 (from django)

Using cached sqlparse-0.5.0-py3-none-any.whl.metadata (3.9 kB)

Collecting tzdata (from django)

Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)

Using cached Django-5.0.6-py3-none-any.whl (8.2 MB)

Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)

Using cached sqlparse-0.5.0-py3-none-any.whl (43 kB)

Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)

Installing collected packages: tzdata, sqlparse, asgiref, django

Successfully installed asgiref-3.8.1 django-5.0.6 sqlparse-0.5.0 tzdata-2024.1

--- puedo comprobar que tengo commando nuevo: django-admin --version

C:\Excellent>django-admin --version

5.0.6

## ## 05 – Creation Project

django-admin startproject gleaming . (Reluciente)

El punto es para que cree elm projecto en la misma carpeta y no cree una subcarpeta

(venv) C:\Excellent>django-admin startproject gleaming .

(venv) C:\Excellent>

**Gleaming :**

**Es la carpeta principal, tiene la configuracion de todo el projecto**

**Contiene todas las aplicaciones**

A screenshot of a computer

Description automatically generated

## ## 06 – runserver

(venv) C:\Excellent>python manage.py runserver

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.

Run 'python manage.py migrate' to apply them.

May 20, 2024 - 13:46:49

Django version 5.0.6, using settings 'gleaming.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

Con esto ya tengo projecto djangoCreado

A screenshot of a computer

Description automatically generated

## ## 07 – creation of application - tasks

(venv) C:\Excellent>python manage.py startapp tasks

(venv) C:\Excellent>

A screenshot of a computer

Description automatically generated

## ## 08 – Configuraciones Globales del projecto

**Settings.py en gleaming**

**Agregamos la aplicacion tasks:**

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

**'tasks'**

]

## ## 09 – Probar algo simple

El archivo **tasks/views.py** nos permite probar algo cuando una URL sea visitada

**tasks/views.py**

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def helloworld(request):

    return HttpResponse('Hola Moises - Exito')

Para decirle cuando va a ser ejecutada : Debo agregar

En gleaming/urls.py :

from django.contrib import admin

from django.urls import path

from tasks import views # important views de tasks

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', views.helloworld), # ejecutar function helloworld

]

--- Ejecuto nuevamente runserver

(venv) C:\Excellent>python manage.py runserver

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.

Run 'python manage.py migrate' to apply them.

May 20, 2024 - 14:12:07

Django version 5.0.6, using settings 'gleaming.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

[20/May/2024 14:12:21] "GET / HTTP/1.1" 200 19

A computer screen with a number and a box

Description automatically generated

## ## 10 – Tambien puedo poner codigo html

**tasks/views.py**

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def helloworld(request):

    return HttpResponse('<h1>Hola Moises - Exito</h1>')

A screenshot of a computer

Description automatically generated

**Gleaming/urls**

from django.contrib import admin

from django.urls import path

from tasks import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',        views.helloworld),

    path('signup/', views.helloworld),

]

## ## 11 – Devolver un documento Entero

Como poner campos html es tedioso, es major crear paginas html

#-- Create folder templates dentro de aplicacion tasks

Ejemplo: signup.html

Aqui ya no necesito dar una respuesta html, por lo tanto :

No importo : from django.http import HttpResponse

Elimino linea en function helloword:

return HttpResponse('<h1>Hola Moises - Exito</h1>')

utilisare el metodo que nos provee django a traves de shortcuts : render

**tasks/views.py**

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def helloworld(request):

    return HttpResponse('<h1>Hola Moises - Exito</h1>')

**tasks/views.py**

from django.shortcuts import render

# Create your views here.

def helloworld(request):

#   necesito recibir el parametro request y enviar pagina html: signup.html

    return render(request, 'signup.html')

re-executo:

A screenshot of a computer

Description automatically generated

## ## 12 – Formularios

Como Django prove la funcoinalidad para signup a traves de formularios

Vamos a usar su libreria: UserCreationForm

from django.shortcuts import render

#from django.http import HttpResponse

from django.contrib.auth.forms import UserCreationForm

# Create your views here.

def helloworld(request):

#   return HttpResponse('<h1>Hola Moises - Exito</h1>')

#   necesito recibir el parametro request y enviar pagina html: signup.html

    return render(request, 'signup.html')

**#-- Otra alternativa – usando dicionario**

Signup.html

<h1>Signup</h1>

{{mytitle}}

Views/tasks.py

def helloworld(request):

    title = 'Hola soy titulo'

    return render(request, 'signup.html', {

        'mytitle': title

    })

**#-- Otra alternativa – usando Formulario**

Signup.html

<h1>Signup</h1>

{{form}}

Views/tasks.py

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

# Create your views here.

def helloworld(request):

    return render(request, 'signup.html', {

        'form': UserCreationForm

    })

**Me entrega un formulario completamente creado**

A screenshot of a computer screen

Description automatically generated

Signup.html

<h1>Signup</h1>

<form action="/signup/" method="POST">

    {% csrf\_token %} 🡺 create token de seguridad

    {{form.as\_p}} 🡺 as\_p lo ordena

    <button>

        Signup

    </button>

</form>

A screenshot of a computer screen

Description automatically generated

**#-- Usando home.html , signup.html**

**Gleaming/urls**

from django.contrib import admin

from django.urls import path

from tasks import views

urlpatterns = [

    path('admin/',  admin.site.urls),

    path('',        views.home),

    path('signup/', views.signup)

]

**Tasks/views:**

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

# Create your views here.

def home(request):

    return render(request, 'home.html')

def signup(request):

    return render(request, 'signup.html')

**home.html**

<h1>Home</h1>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# ## 13 – Creation table Task

**Ir a tasks/models.py**

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Task(models.Model):

    title = models.CharField(max\_length=100)

    description = models.TextField(blank=True)

    created = models.DateTimeField(auto\_now\_add=True)

    datecompleted = models.DateTimeField(null=True)

    important = models.BooleanField(default=False)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

# ## 14 – python manage.py makemigrations

Este comando permite generar la tabla

(venv) C:\Excellent>python manage.py makemigrations

Migrations for 'tasks':

tasks\migrations\0001\_initial.py

- Create model Task

A screenshot of a computer

Description automatically generated

# ## 15 – python manage.py migrate

Este comando permite ejecutar la tabla

A screen shot of a computer

Description automatically generated

# ## 16 – Como interactuar con la tabla

#-- python manage.py shell

Puedo importar los modelos, interactual con ellos

**Este metodo es tedioso, es mejor usar interfaz grafica**

# ## 17 – Panel administracion – admin

### #-- Creation Super Usuario

**(venv) C:\Excellent>python manage.py createsuperuser**

Username (leave blank to use 'moise'): moiadm

Email address: moisesstudytech@gmail.com

Password: moises01

Password (again):

Superuser created successfully.

### --- Run server again: (venv) C:\Excellent>python manage.py runserver

# ## 18 – connect como superuser to admin Panel

<http://127.0.0.1:8000/admin/> user :moiadm

Estoy en django administration

Aqui estan los usuarios y los grupos

Si deseo ver la tabla Tasks, debo registrat el modelo en la administracion.

Se debe editar el archive tasks/admin.py

**tasks/admin.py**

from django.contrib import admin

from .models import Task

# Register your models here.

admin.site.register(Task)

# ## 19 – Refresh admin to reflect changes

A screenshot of a computer

Description automatically generated

## 20 – agregar task desde adminA screenshot of a computer

Description automatically generated

# ## 21 – cambiar object information por columna

A close up of a text

Description automatically generated

**En tasks/models.py**

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Task(models.Model):

    title = models.CharField(max\_length=100)

    description = models.TextField(blank=True)

    created = models.DateTimeField(auto\_now\_add=True)

    datecompleted = models.DateTimeField(null=True)

    important = models.BooleanField(default=False)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    def \_\_str\_\_(self):

        return self.title

A screenshot of a computer

Description automatically generated

Y asi sucesivamente : …

    def \_\_str\_\_(self):

        return self.title + ' - by: ' + self.user.username

A screenshot of a computer

Description automatically generated

# ## 22 – Quiero ver campos que son solo lectura en admin

Ejemplo: created = models.DateTimeField(auto\_now\_add=True)

**En tasks/admin.py**

from django.contrib import admin

from .models import Task

**class TaskAdmin(admin.ModelAdmin):**

**readonly\_fields = ("created", ) # la coma porque es una tupla**

# Register your models here.

admin.site.register(Task, **TaskAdmin**)

A screenshot of a computer

Description automatically generated

# ## 23 – Deseo crear tareas desde mi interfaz o aplicacion principal y no solo del administrador

--- Crear create\_task.html

**tasks/templates/create\_task.html**

{% extends 'base.html' %}

{% block content %}

<h1>Create Task</h1>

{% endblock %}

--- Crear una function **create\_task** para llamar a este html:

**tasks/views.py**

def create\_task(request):

    return render(request, 'create\_task.html')

--- Luego ir a Gleaming (carpeta principal del projecto) , a urls.py para agregar la ruta:

**Gleaming/urls.py**

from django.contrib import admin

from django.urls import path

from tasks import views

urlpatterns = [

    path('admin/',          admin.site.urls),

    path('',                views.home,         name='home'),

    path('signup/',         views.signup,       name='signup'),

    path('tasks/',          views.tasks,        name='tasks'),

    path('tasks/create/',   views.create\_task,  name='create\_task'),

    path('logout/',         views.signout,      name='logout'),

    path('signin/',         views.signin,       name='signin')

]

# Example of explanation for create\_task.html

#   Path                : 'tasks/create/' para que luzca como una url

#   views.create\_task   : La function a ejecutar cuando visite esta url

#   name='create\_task'  : El nombre de la ruta

A screenshot of a computer

Description automatically generated

# ## 24 – Crear Formulario y anadir tareas

<!-- Create etiquete form.

debes enviarlo a /tasks/create a traves del metodo POST

para signin.html y signup.html recibimos el objeto form que es de django

pero tambien podemos crear nuestros propios objetos personalizados (Que estan basados en el modelo que ya creamos en tasks/models.py en la definicion class Task)

    class Task(models.Model):

    title = models.CharField(max\_length=100)

    description = models.TextField(blank=True)

    created = models.DateTimeField(auto\_now\_add=True)

    datecompleted = models.DateTimeField(null=True)

    important = models.BooleanField(default=False)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    def \_\_str\_\_(self):

        return self.title + ' - by: ' + self.user.username

Asi como hemos creado una tabla de tarea, podemos decir de crear un formulario a partir de los datos que habian en la tabla y eso se hace creando un archivo forms.py en tasks: tasks/forms.py

 -->

--- Creacion de formulario para Task que puede ser enviado al front-end

**Tasks/forms.py**

from django.forms import ModelForm  # esto es una clase

from .models import Task            # importamos el modelo de Task

class TaskForm(ModelForm):

    class Meta:

        model = Task # Este formulario es basado en Task (Esta importado arriba)

        fields = ['title', 'description', 'important']  # Campos que me interesan

**Modificamos: tasks/views**

# Al momento de enviar create\_tasks voy a importer mi formulario:

from .forms import TaskForm

def create\_task(request):

    return render(request, 'create\_task.html', {

            'form': TaskForm

        })

Y en create\_task.html, debemos modificar para recibor el formulario

Tambien agregamos button save

**tasks/templates/create\_task.html**

{% extends 'base.html' %}

{% block content %}

    <h1>Create Task</h1>

    <form action="/tasks/create/" method="POST">

        {% csrf\_token %} # se envia para proteccion,

# asi django se asegura que el lo genero

        {{form}} # le digo que me muestre este formulario

        <button>

        Save

        </button>

    </form>

{% endblock %}

Al refrescar, tenemos:

A screenshot of a computer

Description automatically generated

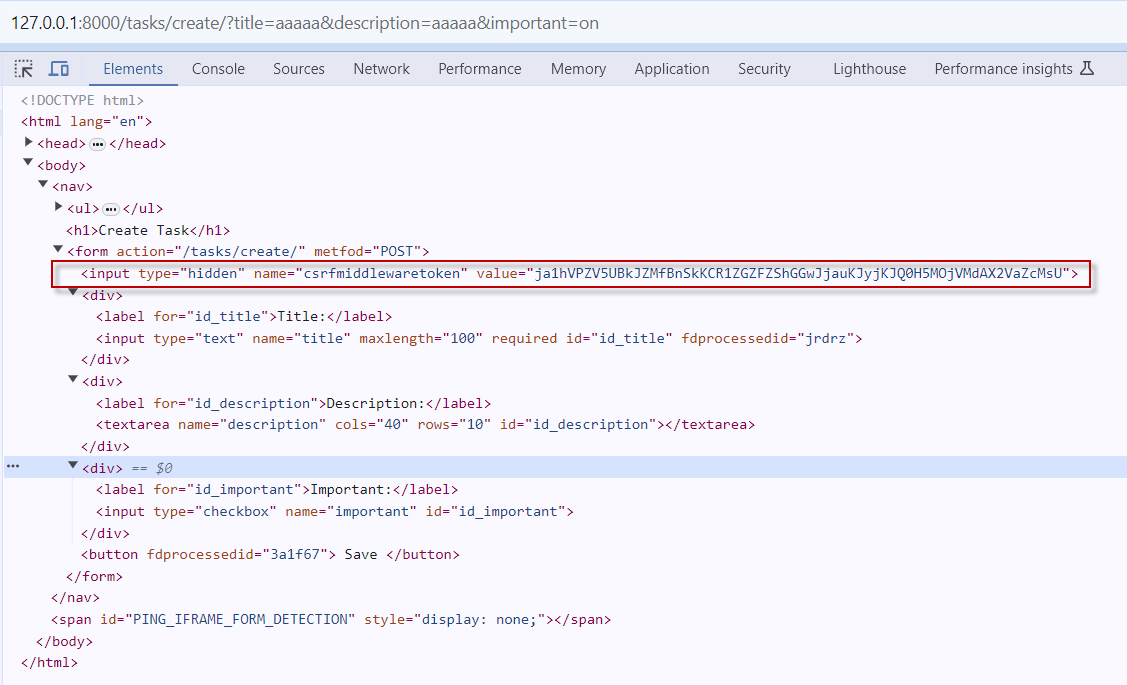
# ## 26 – Campo oculto – protege Form: {% csrf\_token %}

    <form action="/tasks/create/" method="POST">

        {% csrf\_token %}

  {{form}}

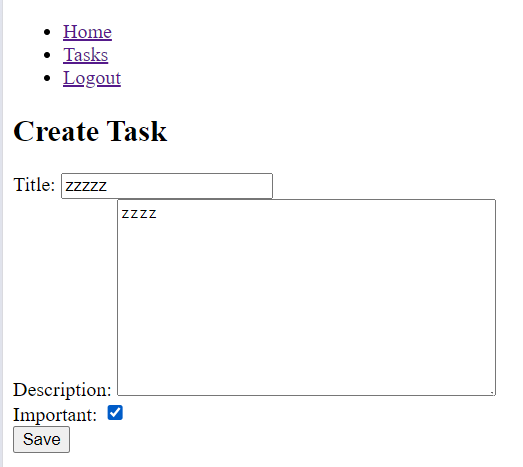
    </form>



# ## 27 – Continuacion con create task – after save

Estoy en <http://127.0.0.1:8000/tasks/create/>

Meto mis datos



Nos envia nuevamente a la lista : http://127.0.0.1:8000/tasks/create/

porque hace esto, en :

**Tasks/templates/create\_task.html**

{% extends 'base.html' %}

{% block content %}

    <h1>Create Task</h1>

**<form action="/tasks/create/" method="POST">**

        {% csrf\_token %}

        {{form}}

        <button>

           Save

        </button>

    </form>

{% endblock %}

Es la misma vista que renderiza la pagina :

**Tasks/views.py**

def create\_task(request):

    return render(request, 'create\_task.html', {

            'form': TaskForm

        })

Para solucionar esto, modifico :

**Tasks/views.py**

def create\_task(request):

    if request.method == 'GET': **#cuando recibo por primera vez el formulario**

        return render(request, 'create\_task.html', {

            'form': TaskForm

        })

    else: **# viene el formulario de nuevo pero con datos**

        # no me esta enviando por el metodo GET sino a travez de un button : Save

        print(request.POST) # pongo print para verificar que tengo los datos

        return render(request, 'create\_task.html', {

            'form': TaskForm

        })

**print(request.POST), contiene:**

<QueryDict: {'csrfmiddlewaretoken': ['9hiTQAN1UgcibmvirgT43VjeWQG3xOQKwD0V5fyPnFlHinq8CHkF5w2aZ5RnKI1Y'], 'title': ['hola'], 'description': ['homa desc'], 'important': ['on']}>

[21/May/2024 10:06:37] "POST /tasks/create/ HTTP/1.1" 200 1250

# ## 28 – save data a base de datos

Una forma es en models.py :

--- Importando y guardando datos yo mismo

**tasks/models.py**

# Create your models here.

class Task(models.Model):

    title = models.CharField(max\_length=100)

    description = models.TextField(blank=True)

    created = models.DateTimeField(auto\_now\_add=True)

    datecompleted = models.DateTimeField(null=True)

    important = models.BooleanField(default=False)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    def \_\_str\_\_(self):

        return self.title + ' - by: ' + self.user.username

Otra forma es en forms.py:

**tasks/forms.py**

from django.forms import ModelForm  # esto es una clase

from .models import Task            # importamos el modelo de Task

class TaskForm(ModelForm):

    class Meta:

        model = Task

        fields = ['title', 'description', 'important']

este forms se puede importar y utilizar para guardar datos ademas de generar formulario

**empezaremos con tasks/views.py :**

def create\_task(request):

    if request.method == 'GET':

        return render(request, 'create\_task.html', {

            'form': TaskForm

        })

    else:

        #print(request.POST)

# aqui recibo los datos y se los paso a la clase TaskForm y el va a

# generar por mi un formulario

TaskForm(request.POST)

# para verlo :

form = TaskForm(request.POST)

Print(form)

        return render(request, 'create\_task.html', {

            'form': TaskForm

        })

**Executing:**

<div>

<label for="id\_title">Title:</label>

<input type="text" name="title"

value="tarea 2"

maxlength="100" required id="id\_title">

</div>

<div>

<label for="id\_description">Description:</label>

<textarea name="description"

cols="40" rows="10" id="id\_description">

desc tarea 2

</textarea>

</div>

<div>

<label for="id\_important">Important:</label>

<input type="checkbox" name="important"

id="id\_important"

checked>

</div>

[21/May/2024 10:54:39] "POST /tasks/create/ HTTP/1.1" 200 1250

Luego de probar, tomaremos los datos del formulario y haremos save.

**tasks/views.py :**

from django.shortcuts import render, redirect

from django.contrib.auth.forms import UserCreationForm, AuthenticationForm

from django.contrib.auth.models import User

from django.contrib.auth import login, logout, authenticate

from django.db import IntegrityError

from .forms import TaskForm

# Create your views here.

def home(request):

    return render(request, 'home.html')

def signup(request):

    if request.method == 'GET':

        return render(request, 'signup.html', {

            'form': UserCreationForm

        })

    else:

        if request.POST['password1'] == request.POST['password2']:

            try:

                user = User.objects.create\_user(

                    username=request.POST['username'],

                    password=request.POST['password1']

                    )

                user.save()

                login(request, user)

                #return HttpResponse('User created sucessfully')

                return redirect('tasks')

            except IntegrityError:

                return render(request, 'signup.html', {

                    'form': UserCreationForm,

                    "error": 'Username already exists'

                })

        return render(request, 'signup.html', {

                    'form': UserCreationForm,

                    "error": 'Password do not match'

                })

def tasks(request):

    return render(request, 'tasks.html')

def create\_task(request):

    if request.method == 'GET':   #cuando recibo por primera vez el formulario

        return render(request, 'create\_task.html', {

            'form': TaskForm

        })

    else:                         # viene el formulario de nuevo pero con datos

        # no me esta enviando por el metodo GET sino a travez de un button : Save

        # print(request.POST)   # pongo print para verificar que tengo los datos

        # aqui recibo los datos y se los paso a la clase TaskForm que va a

# generar por mi un formulario, un test para verlo :

        # form = TaskForm(request.POST)

        # print(form)

        # utilisare el formulario para grabar los datos

        try:

            form = TaskForm(request.POST)

# solo quiero que devuelva los datos, por eso pongo commit False

new\_task = form.save(commit=False)

# Como no estoy entregando el usuario, genera un error.

# SE arregla al moverlo desde request

new\_task.user = request.user

            #print(new\_task)  # esto es una nueva tarea , la vere por consola

# una vez lo guardo, no quiero que me renderize,

# le dire que me redireccione a pagina tasks

            new\_task.save()

            # return render(request, 'create\_task.html', {

            #     'form': TaskForm

            # })

            return redirect('tasks')

        except ValueError:

            return render(request, 'create\_task.html', {

                        'form': TaskForm,

                        'error': 'Please provide valida data'

                    })

        except IntegrityError:

            return render(request, 'create\_task.html', {

                        'form': TaskForm,

                        'error': 'Hay un Error de integridad'

                    })

def signout(request):

    logout(request)

    return redirect('home')

def signin(request):

    if request.method == 'GET':

        return render(request, 'signin.html', {

            'form': AuthenticationForm

        })

    else:

        user = authenticate(request,

                            username=request.POST['username'],

                            password=request.POST['password'])

        if user is None:

            return render(request, 'signin.html', {

            'form': AuthenticationForm,

            "error": 'Username or Password Incorrect'

            })

        else:

            login(request, user)

            return redirect('tasks')

# ## 29 – Listar tareas – Todas (All)

Ya tenemos en **tasks/views.py** la renderizacion de tareas

def tasks(request):

    return render(request, 'tasks.html')

necesitamos importar el modelo de las tareas :

from .models import Task

**luego en : tasks/views:**

**def tasks(request):**

    tasks = Task.objects.all()  # devuelve todas las tareas de la base de datos

    return render(request, 'tasks.html', {'tasks':tasks})  # paso dato al Front

    #return render(request, 'tasks.html')

**Tambien en tasks/templates/task.html:**

{% extends 'base.html' %}

{% block content %}

<h1>Tasks</h1>

<ul>

    {% for task in tasks %}

    <li>

        <h1>{{task.title}}</h1>

        <p>{{task.description}}</p>

        <p>{{task.user.username}}</p>

    </li>

    {% endfor %}

</ul>

{% endblock %}

A screenshot of a computer

Description automatically generated

# ## 29 – Listar tareas con Filter

Pero aqui hay un problema, el usuario esta viendo la lista de tareas de todos los usuarios,

No solamente las de el.

Para esto en vez de usar un all, deberia ser un Filter

**Tasks/views.py**

def tasks(request):

    tasks = Task.objects.**all**()  # devuelve todas las tareas de la base de datos

    return render(request, 'tasks.html', {'tasks':tasks})  # paso dato al Front

def tasks(request):

# la propiedad user debe ser igual al user del request

    tasks = Task.objects.**filter**(user=request.user)

    return render(request, 'tasks.html', {'tasks':tasks})  # paso dato al Front

A screenshot of a computer

Description automatically generated

# ## 30 – Listar tareas con mas Filter

Se modifica esto para saber que tareas aun no han sido completadas

**Tasks/views.py**

def tasks(request):

    tasks = Task.objects.filter(user=request.user, **datecompleted\_\_isnull=True**)

    return render(request, 'tasks.html', {'tasks':tasks})

### ## 30 - Parentesis: DateCompleted optional

Como esta hinchando mucho el campo DateCompleted, lo dejaremos opcional:

**Para el administrador blank=True es opcional**

**Pero para la base de datos, si debemos indicarlo**

**En tasks/models.py:**

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Task(models.Model):

    title = models.CharField(max\_length=100)

    description = models.TextField(blank=True)

    created = models.DateTimeField(auto\_now\_add=True)

    datecompleted = models.DateTimeField(null=True, blank=True)

    important = models.BooleanField(default=False)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    def \_\_str\_\_(self):

        return self.title + ' - by: ' + self.user.username

# ## 31 – Podemos colocar condicionales dentro de ninja

**tasks/templates/task\_detail.html:**

{% extends 'base.html' %}

{% block content %}

<h1>Tasks</h1>

<ul>

    {% for task in tasks %}

    <li>

        {% if task.important %}

        <strong>{{task.title}}</strong>

        {% else %}

                {{task.title}}

        {% endif %}

        <p>{{task.description}}</p>

        <p>{{task.user.username}}</p>

    </li>

    {% endfor %}

</ul>

{% endblock %}

# ## 32 – Consultar detalles de una Tarea

**tasks/templates/task\_detail.html**

{% extends 'base.html' %}

{% block content %}

<h1>Task Details</h1>

{% endblock %}

**tasks/views.py :**

# Create your views here.

def task\_detail(request, task\_id):

    #print(task\_id)

# El task\_id me sirve para consultar

# Desde el modelo Task utilisare object y obtener un dato

# le digo que busque el dato donde el primary key = task\_id

    task = Task.object.get(pk=task\_id)

  #return render(request, 'task\_detail.html')

return render(request, 'task\_detail.html', {'task': task})

**gleaming/urls.py:**

from django.contrib import admin

from django.urls import path

from tasks import views

urlpatterns = [

    path('admin/',          admin.site.urls),

    path('',                views.home,         name='home'),

    path('signup/',         views.signup,       name='signup'),

    path('tasks/',          views.tasks,        name='tasks'),

    path('tasks/create/',   views.create\_task,  name='create\_task'),

    # un id sera guardado en esta variable: task\_id

    path('tasks/<int:task\_id>/',   views.task\_detail,  name='task\_detail'),

    path('logout/',         views.signout,      name='logout'),

    path('signin/',         views.signin,       name='signin')

]

**En tasks/templates/task\_detail.html**

{% extends 'base.html' %}

{% block content %}

<h1>{{task.title}}</h1>

{% endblock %}

# ## 32 – Parentesis – problema con Task.object.get

Esta sentencia no es muy buena cuando no existe el registro, hace caer al servidor

task = Task.object.get(pk=task\_id)

**cambiar en tasks/views.py:**

importar la libreria get\_object\_or\_404:

from django.shortcuts import render, redirect, get\_object\_or\_404

modificar sentencia:

task = get\_object\_or\_404(Task, pk=task\_id)

Este cambio ya no envia respuesta al servidor y no se cae

A screenshot of a computer

Description automatically generated

# ## 33 – Navegar rutas

**Modificar tasks/templates/tasks.html**

{% extends 'base.html' %}

{% block content %}

<h1>Tasks</h1>

<ul>

    {% for task in tasks %}

    <li>

        {% if task.important %}

        <strong>{{task.title}}</strong>

        {% else %}

                {{task.title}}

        {% endif %}

        <p>{{task.description}}</p>

        <p>{{task.user.username}}</p>

    </li>

    {% endfor %}

</ul>

{% endblock %}

A:

{% extends 'base.html' %}

{% block content %}

<h1>Tasks</h1>

<ul>

    {% for task in tasks %}

    <li>

        <a href="{% url 'task\_detail' task.id %}"> # url de ninja

            {% if task.important %}

            <strong>{{task.title}}</strong>

            {% else %}

                    {{task.title}}

            {% endif %}

            <p>{{task.description}}</p>

            <p>{{task.user.username}}</p>

        </a>

    </li>

    {% endfor %}

</ul>

{% endblock %}

Asi genera la salida:

A screenshot of a computer

Description automatically generated

# ## 34 – Upadte base.html segun usuario authentificado

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Excellent and Gleaming</title>

</head>

<body>

    <nav>

        <ul>

            <li>

                <a href="/">Home</a>

            </li>

            {% if user.is\_authenticated %}

**<li>**

**<a href="/tasks">Tasks</a>**

**</li>**

**<li>**

**<a href="/tasks/create/">Create task</a>**

**</li>**

            <li>

                <a href="/logout">Logout</a>

            </li>

            {% else %}

            <li>

                <a href="/signup">Signup</a>

            </li>

            <li>

                <a href="/signin">Signin</a>

            </li>

            {% endif %}

        </ul>

    {% block content %}

    {% endblock %}

</body>

</html>

A screenshot of a computer

Description automatically generated

# ## 35 – Editar/Actualizar una tarea

Ya temenos vista task\_detail.html nos permite mostrar titulo de tarea

Ahora crearemos formulario para mostrar la tarea

**En tasks/views.py**

def task\_detail(request, task\_id):

    task = get\_object\_or\_404(Task, pk=task\_id)

    form = TaskForm(instance=task) # Llama instance con valor de la tarea (task)

    # Le pasamos el valor al Front

    return render(request, 'task\_detail.html', {'task': task, 'form': form})

**En tasks/templates/task\_detail.html**

{% extends 'base.html' %}

{% block content %}

<h1>{{task.title}}</h1>

**{{form}}**

{% endblock %}

A screenshot of a computer

Description automatically generated

# ## 36 – Crear su propio Formulario en task\_detail.html

**Con esto tengo el formulario basico para editar.**

**En tasks/templates/task\_detail.html**

{% extends 'base.html' %}

{% block content %}

    <h1>{{task.title}}</h1>

    {{error}}

    <form method="POST">

        {%csrf\_token%}

        {{form.as\_p}}

        <button>

            Update

        </button>

    </form>

{% endblock %}

Falta la actualizacion

**En tasks/views.py: modificaremos esto**

    if request.method == 'GET':

        # task = get\_object\_or\_404(Task, pk=task\_id)

        # No debe modificar tarea que no le pertenece : user=request.user

        task = get\_object\_or\_404(Task, pk=task\_id, user=request.user)

        form = TaskForm(instance=task)

        return render(request, 'task\_detail.html', {'task': task, 'form': form})

    else:

        try:

            # No debe modificar tarea que no le pertenece : user=request.user

            task = get\_object\_or\_404(Task, pk=task\_id, user=request.user)

            form = TaskForm(request.POST, instance=task)

            form.save()

            return redirect('tasks')

        except ValueError:

            return render(request,

                          'task\_detail.html',

                          {'task': task,

                           'form': form,

                           'error': "Error Updating task"})

# ## 37 – Poner tarea Completada

En tasks/views.py

from django.utils import timezone

def complete\_task(request, task\_id):

    task = get\_object\_or\_404(Task, pk=task\_id, user=request.user)

    if request.method == 'POST':

        task.datecompleted = timezone.now()

        task.save()

        return redirect('tasks')

**en gleaming/urls.py:**

urlpatterns = [

   # para marcar tarea como completada

   path('tasks/<int:task\_id>/complete', views.complete\_task, name='complete\_task'

**en tasks/templates/task\_detail.html:**

{% extends 'base.html' %}

{% block content %}

    <h1>{{task.title}}</h1>

    {{error}}

    <form method="POST">

        {%csrf\_token%}

        {{form.as\_p}}

        <button>

            Update

        </button>

    </form>

**<form action="{% url 'complete\_task' task.id %}" method="POST">**

**{%csrf\_token%}**

**<button>**

**Complete**

**</button>**

**</form>**

{% endblock %}

**En tasks/templates/task.html**

Indicar que son las tareas no completadas

**<h1>Tasks Pending</h1>**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Desaparecio de la lista camisa correctisima**

A screenshot of a computer

Description automatically generated

# ## 38 – Boton Delete

**En gleaming/urls.py:**

urlpatterns = [

    # para deletear

    path('tasks/<int:task\_id>/delete',   views.delete\_task,  name='delete\_task'),

**En tasks/views.py**

def delete\_task(request, task\_id):

    task = get\_object\_or\_404(Task, pk=task\_id, user=request.user)

    if request.method == 'POST':

        task.delete()

        return redirect('tasks')

**en tasks/templates/task\_detail.html:**

    <form action="{% url 'delete\_task' task.id %}" method="POST">

        {%csrf\_token%}

        <button>

            Delete

        </button>

    </form>

# ## 39 – Listar Tareas Completadas

**En gleaming/urls.py:**

urlpatterns = [

path('tasks\_completed/<int:task\_id>/',

views. tasks\_completed,

name=' tasks\_completed'),

**En tasks/views.py**

def tasks\_completed(request):

    tasks = Task.objects.filter(

user=request.user,

datecompleted\_\_isnull=False).order\_by('-datecompleted')

    return render(request, 'tasks.html', {'tasks':tasks})

# ## 40 – Ajustar la base

        <ul>

            {% if user.is\_authenticated %}

            <li>

                <a href="/tasks">Tasks **Pending**</a>

            </li>

            <li>

**<!-- Ambas sentencias son equivalentes:**

**<a href="/tasks\_completed">Tasks Completed</a>**

**<a href="{% url 'tasks\_completed' %}">Tasks Completed</a>**

**-->**

**<a href="{% url 'tasks\_completed' %}">Tasks Completed</a>**

            </li>

            {% endif %}

        </ul>

# ## 41 – Proteger las rutas

Si un usuario pone task podria entrar directamente.

Si no se logea en ciertas opciones el sistema fallara.

Para eso jdango tiene cubierto este tema con el decorator login\_required

Este decorador se pone en cada funcion y de esta forma indico que la funcion esta protegida

**En tasks/views.py :**

from django.contrib.auth.decorators import login\_required

# Create your views here.

def home(request):

def signup(request):

**@login\_required**

def tasks(request):

**@login\_required**

def tasks\_completed(request):

**@login\_required**

def create\_task(request):

**@login\_required**

def task\_detail(request, task\_id):

**@login\_required**

def complete\_task(request, task\_id):

**@login\_required**

def delete\_task(request, task\_id):

**@login\_required**

def signout(request):

def signin(request):

**en Gleaming/settings.py**

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC\_URL = 'static/'

**LOGIN\_URL = '/signin'**

# ## 42 – Bootstrap

Es un framework de CSS

# ver sitio getbootstrap.com

Copier CDN links for CSS:

A computer screen shot of a computer

Description automatically generated

Link del sitio: https://getbootstrap.com/docs/5.3/getting-started/introduction/

Link a copiar: <https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css>

**En base.html:** # debajo del title

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Excellent and Gleaming</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/

bootstrap@5.3.3/dist/css/bootstrap.min.css"

</head>

Para trabajar con nav, puedo buscar en bootstrap.com Navbar

Copio el codigo de parte con multiples pestañas

A screenshot of a computer

Description automatically generated

Hago paste justo antes de mi nav:

<body>

    <nav class="navbar navbar-expand-lg bg-body-tertiary">

        <div class="container-fluid">

          <a class="navbar-brand" href="#">Navbar</a>

          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

            <span class="navbar-toggler-icon"></span>

          </button>

          <div class="collapse navbar-collapse" id="navbarSupportedContent">

            <ul class="navbar-nav me-auto mb-2 mb-lg-0">

              <li class="nav-item">

                <a class="nav-link active" aria-current="page" href="#">Home</a>

              </li>

              <li class="nav-item">

                <a class="nav-link" href="#">Link</a>

              </li>

              <li class="nav-item dropdown">

                <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">

                  Dropdown

                </a>

                <ul class="dropdown-menu">

                  <li><a class="dropdown-item" href="#">Action</a></li>

                  <li><a class="dropdown-item" href="#">Another action</a></li>

                  <li><hr class="dropdown-divider"></li>

                  <li><a class="dropdown-item" href="#">Something else here</a></li>

                </ul>

              </li>

              <li class="nav-item">

                <a class="nav-link disabled" aria-disabled="true">Disabled</a>

              </li>

            </ul>

            <form class="d-flex" role="search">

              <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>

    <nav>

        <ul>

            <li>

Tal cual sin editar nada se ve asi:

A screenshot of a computer

Description automatically generated

Cambie <nav class="navbar navbar-expand-lg bg-dark navbar-dark">

A screenshot of a computer

Description automatically generated

Modification: para que el container este centrado

        <div class="container">

A screenshot of a computer

Description automatically generated

Modificar:             <ul class="navbar-nav ms-auto">

A screenshot of a computer

Description automatically generated

# ## 43 – Modificando fachada de signup.html

{% extends 'base.html' %}

{% block content %}

<main class="container">

<div class="row">

    <div class="col-md-4 offset-md-4">

        <form method="POST" class="card card-body text-center">

            <h1>Signup</h1>

            {{error}}

            {% csrf\_token %}

            {{form}} # si elimino form y lo hago yo mismo (ver abajo)

            <button>

                Signup

            </button>

        </form>

    </div>

</div>

</main>

{% endblock %}

**Queda asi:**

{% extends 'base.html' %}

{% block content %}

<main class="container">

<div class="row">

    <div class="col-md-4 offset-md-4 mt-5">

        <form method="POST" class="card card-body">

            <h1 class="text-center">Signup</h1>

            {{error}}

            {% csrf\_token %}

            <div class="mb-3">

                <label for="username">Username:</label>

                <input type="text" name="username" id="username"

                class="form-control" placeholder="Write your username"

            </div>

            <div class="mb-3">

                <label for="password1">Password:</label>

                <input type="password" name="password1" id="password1"

                class="form-control" placeholder="Write your password"

            </div>

            <div class="mb-3">

                <label for="password2">Confirm your Password:</label>

                <input type="password" name="password2" id="password2"

                class="form-control" placeholder="Confirm your Password"

            </div>

            <button class="btn btn-primary">

                Signup

            </button>

        </form>

    </div>

</div>

</main>

{% endblock %}

A screen shot of a computer

Description automatically generated

# ## 44 – Modificando fachada de signin.html

Copiar mismo text de signup.html para signin.html

            <h1 class="text-center">Signup</h1>

            {{error}}

            {% csrf\_token %}

            <div class="mb-3">

                <label for="username">Username:</label>

                <input type="text" name="username" id="username"

                class="form-control" placeholder="Write your username"

            </div>

            <div class="mb-3">

                <label for="password1">Password:</label>

                <input type="password" name="password1" id="password1"

                class="form-control" placeholder="Write your password"

            </div>

            <div class="mb-3">

                <label for="password2">Confirm your Password:</label>

                <input type="password" name="password2" id="password2"

                class="form-control" placeholder="Confirm your Password"

            </div>

            <button class="btn btn-primary">

                Signup

            </button>

**En tasks/template/signin remove :**  {{form.as\_p}}

{% extends 'base.html' %}

{% block content %}

<h1>Signin</h1>

{{error}}

<form action="/signin/" method="POST">

    {% csrf\_token %}

    {{form.as\_p}} # to remove

    <button>

        Signin

    </button>

</form>

{% endblock %}

Queda asi:

{% extends 'base.html' %}

{% block content %}

<main class="container">

    <div class="row">

        <section class="col-md-4 offset-md-4 mt-5">

            <form action="/signin/" method="POST" class="card card-body">

                <h1 class="text-center">Signin</h1>

                {{error}}{% csrf\_token %}

                <div class="mb-3">

                    <label for="username">Username:</label>

                    <input type="text" name="username" id="username"

                    class="form-control" placeholder="Write your username"

                </div>

                <div class="mb-3">

                    <label for="password">Password:</label>

                    <input type="password" name="password" id="password"

                    class="form-control" placeholder="Write your password"

                </div>

                <button class="btn btn-primary">Signin</button>

            </form>

        </section>

    </div>

</main>

{% endblock %}

# ## 45 – Desplegar aplicacion – parte 1

Render.com

Signing with github

Render by render would like permission to:

Authorize Render

Panel control: Elegir New Web Service into Web Services

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Primero debemos configurar el projecto para que funcione en produccion

Primero Cancelar consola en Visual Studio Code

Vamos a reinicializar un repositorio de Git

Pero primero necesito colocar algunas configuraciones basicas

Desplegar aplicaciones Django no es tan simple como llegar y subir, sino que hay que anadir unas configuraciones a nuestro projecto

Asi que Render tiene contemplado esto, asi que debes buscar en browser algo como **Django Render deploy**, esto nos lleva a la documentacion de render.com/docs/deploy-django

La documentation dice de anadir Poetry (Para adminitracion de packages)

pero no usamos eso , asi que ignorarlo.

Luego la documentacion muestra todo lo que ya hemos hecho en nuestro projecto vis visual studio code.

Luego, hace incapie (hacer hincapié / emphasize / mettre l'accent) en un tema de seguridad.

En nuestro projecto, en settings, the variable: SECRET\_KEY

A screen shot of a computer

Description automatically generated

Render dice no utilizar esta informacion precedente, sino que mejor utilizar una variable de entorno :

**Adding basic security**

You should ensure that any production-level application is properly secured and configured. The Django documentation provides a useful [deployment checklist](https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/), which we’ll follow in this step.

1. Open mysite/settings.py and find the declaration of the SECRET\_KEY setting. We don’t want to store production secrets in source code, so we’ll fetch this value from an environment variable that we’ll create later:

La variable de entorno me la dara en produccion la nube en su momento

Entonces debo leerla en mi aplicacion en lugar de establecerlo aqui

Para leer esas variables python tiene un modulo llamado : **os**

**Y como sugiere Render**

A computer screen shot of a computer screen

Description automatically generated with medium confidence

**Entonces en Gleaming.settings.py :**

from pathlib import Path

import os

# SECURITY WARNING: keep the secret key used in production secret!

# Con esto leemos la variable de entorno que nos dara la nube

SECRET\_KEY = os.environ.get('SECRET\_KEY', default='your secret key')

Luego nos dice no activate debug en produccion :

A screenshot of a computer

Description automatically generated

Al mostrar Errores dejamos vulnerable la aplicacion a personas que encuentren ese error en la aplicacion

**En base a lo anterior y Continuando con Gleaming.settings.py :**

Podria set tan facil como cambiar este default a False, pero Render recomienda

Usar la variable RENDER, y si existe esta significara que DEBUG sera False en production, y sino existe sera verdaderoque es el caso de Development

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

DEBUG = True

Lo cambiamos a:

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

# DEBUG = True

DEBUG = 'RENDER' not in os.environ

Tambien Render dice:

A screenshot of a computer program

Description automatically generated

Render no permitira que cualquiera se conecte a tu aplicacion

Entonces tenemos que establecer el ALLOWED\_HOSTS

Y tenemos que darle por ejemplo las url permitidas, la direccion de tu IP

Pero como no sabemos que valor de IP nos va a dar el Servidor de la nube

Entonces podriamos darselo manualmente o usar la variable de entorno que no da RENDER a traves de la variable de entorno: RENDER\_EXTERNAL\_HOSTNAME

Copiamos esto debajo de ALLOWED\_HOSTS = []:

**RENDER\_EXTERNAL\_HOSTNAME = os.environ.get('RENDER\_EXTERNAL\_HOSTNAME')**

**Gleaming.settings.py :**

ALLOWED\_HOSTS = []

RENDER\_EXTERNAL\_HOSTNAME = os.environ.get('RENDER\_EXTERNAL\_HOSTNAME')

if RENDER\_EXTERNAL\_HOSTNAME:

    ALLOWED\_HOSTS.append(RENDER\_EXTERNAL\_HOSTNAME)

**La bases de datos:**

Muchos entorno de la nube no permiten que tengamos entornos con SQLite,

Sino que sea por otro servidor (Una base de datos de verdad por decirlo de alguna manera), que podrian ser postgreSQL, mysql, oracle, etc

Entonces para configurar Django para postgreSQL, Nos proveen de una variable de environment DATABASE\_URL.

Solamente debemos leer la variable DATABASE\_URL.

Para leer esa variable de entorno hay un paquete especialllamado dj-database-url

Para lo anterior debemos hacer:

pip install dj-database-url ( recordar de hacerlo en el entorno virtual )

(**venv) C:\Excellent>pip install dj-database-url**

Collecting dj-database-url

Downloading dj\_database\_url-2.1.0-py3-none-any.whl.metadata (11 kB)

Requirement already satisfied: Django>=3.2 in c:\excellent\venv\lib\site-packages (from dj-database-url) (5.0.6)

Collecting typing-extensions>=3.10.0.0 (from dj-database-url)

Using cached typing\_extensions-4.11.0-py3-none-any.whl.metadata (3.0 kB)

Requirement already satisfied: asgiref<4,>=3.7.0 in c:\excellent\venv\lib\site-packages (from Django>=3.2->dj-database-url) (3.8.1)

Requirement already satisfied: sqlparse>=0.3.1 in c:\excellent\venv\lib\site-packages (from Django>=3.2->dj-database-url) (0.5.0)

Requirement already satisfied: tzdata in c:\excellent\venv\lib\site-packages (from Django>=3.2->dj-database-url) (2024.1)

Downloading dj\_database\_url-2.1.0-py3-none-any.whl (7.7 kB)

Using cached typing\_extensions-4.11.0-py3-none-any.whl (34 kB)

Installing collected packages: typing-extensions, dj-database-url

Successfully installed dj-database-url-2.1.0 typing-extensions-4.11.0

(venv) C:\Excellent>

Luego lo importamos:

from pathlib import Path

import os

**import dj\_database\_url**

luego instalamos un modulo para PostgreSQL, para que postgreSQL se pueda conectar : **pip install psycopg2-binary**

(venv) C:\Excellent>pip install psycopg2-binary

Collecting psycopg2-binary

Downloading psycopg2\_binary-2.9.9-cp312-cp312-win\_amd64.whl.metadata (4.6 kB)

Downloading psycopg2\_binary-2.9.9-cp312-cp312-win\_amd64.whl (1.2 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 3.2 MB/s eta 0:00:00

Installing collected packages: psycopg2-binary

Successfully installed psycopg2-binary-2.9.9

(venv) C:\Excellent>

Luego RENDER nos dice que para utlizar seria de esta forma :

**Gleaming.settings.py :**

# DATABASES = {

#     'default': {

#         'ENGINE': 'django.db.backends.sqlite3',

#         'NAME': BASE\_DIR / 'db.sqlite3',

#     }

# }

DATABASES = {

    'default': dj\_database\_url.config(

        default='postgresql://postgres:postgres@localhost/postgres',

        conn\_max\_age=600

    )

}

Ahora Tambien Django que solo es un framework y no un servidor Web, necesita de un servidor Web.

Asi como C# y java tienen servidores como apache por ejemplo,

Djando dice cuando yo deba servir el contenido se lo voy a delegar a otra aplicacion

Para servir contendido estatico : html, css, javascripts, Render soluciona esto usando servicio llamado : WhiteNoise

Nosotros solo añadimos este servicio : que se llama whitenoise[brotli]

**Instalaremos : pip install whitenoise[brotli]**

**(venv) C:\Excellent>pip install whitenoise[brotli]**

Collecting whitenoise[brotli]

Downloading whitenoise-6.6.0-py3-none-any.whl.metadata (3.7 kB)

Collecting Brotli (from whitenoise[brotli])

Downloading Brotli-1.1.0-cp312-cp312-win\_amd64.whl.metadata (5.6 kB)

Downloading Brotli-1.1.0-cp312-cp312-win\_amd64.whl (357 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 357.3/357.3 kB 1.5 MB/s eta 0:00:00

Downloading whitenoise-6.6.0-py3-none-any.whl (19 kB)

Installing collected packages: Brotli, whitenoise

Successfully installed Brotli-1.1.0 whitenoise-6.6.0

(venv) C:\Excellent>

Dice la documentation de RENDER DE instalar en el MIDDLEWARE

**Gleaming.settings.py :**

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',

**'whitenoise.middleware.WhiteNoiseMiddleware',**

]

**Tambien, agregar esto despues de STATIC\_URL = 'static/':**

STATIC\_URL = 'static/'

if not DEBUG:   # Tell  Django to copy statics to the 'staticfiles' directory

    # in your application directory on Render.

    STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

    # Turn on WhiteNoise storage backend that takes care of compressing static files

    # and creating unique names for each version so they can safely be cached forever.

    STATICFILES\_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

Luego necesita hacer : 

(venv) C:\Excellent>python manage.py collectstatic

You have requested to collect static files at the destination

location as specified in your settings.

This will overwrite existing files!

Are you sure you want to do this?

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

Traceback (most recent call last):

File "C:\Excellent\manage.py", line 22, in <module>

main()

File "C:\Excellent\manage.py", line 18, in main

execute\_from\_command\_line(sys.argv)

File "C:\Excellent\venv\Lib\site-packages\django\core\management\\_\_init\_\_.py", line 442, in execute\_from\_command\_line

utility.execute()

File "C:\Excellent\venv\Lib\site-packages\django\core\management\\_\_init\_\_.py", line 436, in execute

self.fetch\_command(subcommand).run\_from\_argv(self.argv)

File "C:\Excellent\venv\Lib\site-packages\django\core\management\base.py", line 413, in run\_from\_argv

self.execute(\*args, \*\*cmd\_options)

File "C:\Excellent\venv\Lib\site-packages\django\core\management\base.py", line 459, in execute

output = self.handle(\*args, \*\*options)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Excellent\venv\Lib\site-packages\django\contrib\staticfiles\management\commands\collectstatic.py", line 209, in handle collected = self.collect()

^^^^^^^^^^^^^^

File "C:\Excellent\venv\Lib\site-packages\django\contrib\staticfiles\management\commands\collectstatic.py", line 135, in collect handler(path, prefixed\_path, storage)

File "C:\Excellent\venv\Lib\site-packages\django\contrib\staticfiles\management\commands\collectstatic.py", line 368, in copy\_file

if not self.delete\_file(path, prefixed\_path, source\_storage):

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Excellent\venv\Lib\site-packages\django\contrib\staticfiles\management\commands\collectstatic.py", line 278, in delete\_file if self.storage.exists(prefixed\_path):

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "C:\Excellent\venv\Lib\site-packages\django\core\files\storage\filesystem.py", line 165, in exists

return os.path.lexists(self.path(name))

^^^^^^^^^^^^^^^

File "C:\Excellent\venv\Lib\site-packages\django\contrib\staticfiles\storage.py", line 39, in path

raise ImproperlyConfigured(

django.core.exceptions.ImproperlyConfigured: You're using the staticfiles app without having set the STATIC\_ROOT setting to a filesystem path.

Esto no funciono ya que estamos con DEBUG yes segun el profe --- Estoy empezando a dudar. Jajaja

**Pero dice que en la nube probaremos de nuevo**

# ## 46 – Desplegar aplicacion – parte 2

Que lata: Se esta allargando mucho esta cosa

Bueno pero seguiremos como Arturo Prat

**Crear build.sh**

A screenshot of a computer program

Description automatically generated

**Executar commando: chmod a+x build.sh**

Pero debe ser en powershell – pero no funciona al profe

Entonces en bash:

Fue imposible hacerlo ya que visual studio code no tenia git bash

Entonces, installe Git Installation On Windows

Rebootee Visual studio code

Y aparecio git bash

**Execute en Git bash $ chmod a+x build.sh**

moise@Moises\_acer MINGW64 /c/Excellent

**instalar gunicorn**

(venv) C:\Excellent>pip install gunicorn

Collecting gunicorn

Using cached gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)

Collecting packaging (from gunicorn)

Using cached packaging-24.0-py3-none-any.whl.metadata (3.2 kB)

Using cached gunicorn-22.0.0-py3-none-any.whl (84 kB)

Using cached packaging-24.0-py3-none-any.whl (53 kB)

Installing collected packages: packaging, gunicorn

Successfully installed gunicorn-22.0.0 packaging-24.0

(venv) C:\Excellent>

# ## 47 – Desplegar aplicacion – parte 3

### ## 47 – paste de la guia de instalacion

Nota nunca encontre hasta ahora la misma guia de instalacion del profe.

Entonces, tome su url del video y llegue a la guia que buscaba

Este es el link : <https://docs.render.com/deploy-django>

Como no se si esta va a cambiar algun dia : hare un copy/paste de esta guia :

Deploy a Django App on Render

This guide walks through deploying a [Django](https://www.djangoproject.com/) Python app on Render. You can [use your existing Django project](https://docs.render.com/deploy-django#updating-an-existing-django-project) or [create one from scratch](https://docs.render.com/deploy-django#creating-a-new-django-project).

If you’re new to Django, we recommend first reading the official guide to [Writing your first Django project](https://docs.djangoproject.com/en/3.0/intro/tutorial01/).

Updating an existing Django project

To prepare an existing Django project for production on Render, we’ll make a couple adjustments to its configuration:

* We’ll update your project to use a [Render PostgreSQL database](https://docs.render.com/databases) instead of a SQLite database.
* We’ll configure the [WhiteNoise](https://whitenoise.evans.io/en/stable/django.html" \t "_blank) package to serve your project’s static files.
* We’ll define a build script to run with each deploy.

Use a Render PostgreSQL database

As part of deploying your project, we’ll also deploy a Render PostgreSQL database to serve as its backing datastore. To enable this, let’s add a couple of packages to your project:

| Package | Description |
| --- | --- |
| [psycopg2](https://www.psycopg.org/) | This is the most popular Python adapter for communicating with a PostgreSQL database. |
| [DJ-Database-URL](https://github.com/jacobian/dj-database-url) | This enables you to specify your database details via the DATABASE\_URL environment variable (you’ll obtain your database’s URL from the [Render Dashboard](https://dashboard.render.com/)). |

1. Run the following commands to install these packages:
2. $ pip install psycopg2-binary
3. $ pip install dj-database-url
4. # Add these dependencies to your requirements.txt file:

$ pip freeze > requirements.txt

1. Open settings.py in your project’s main directory (e.g., mysite/settings.py).

Make the following modifications:

# Import dj-database-url at the beginning of the file.

import dj\_database\_url

# Replace the SQLite DATABASES configuration with PostgreSQL:

DATABASES = {

'default': dj\_database\_url.config( # Replace this value with your local database's connection string. default='postgresql://postgres:postgres@localhost:5432/mysite', conn\_max\_age=600 )}

Set up static file serving

Django provides a [dedicated module](https://docs.djangoproject.com/en/5.0/howto/static-files/deployment/) for collecting your project’s static files (HTML, CSS, JavaScript, images, and so on) into a single place for serving in production. This module supports moving files from one place to another, relying on the end web server (such as Render’s default web server, or a tool like NGINX) to serve them to end users.

In this step, we’ll set up [WhiteNoise](https://whitenoise.evans.io/" \t "_blank) to serve these static assets from Render’s web server.

The following instructions summarize the setup described in the [WhiteNoise documentation](http://whitenoise.evans.io/en/stable/django.html" \t "_blank).

1. Add WhiteNoise as a dependency (adding [Brotli](https://en.wikipedia.org/wiki/Brotli" \t "_blank) support is optional, but recommended):
2. $ pip install 'whitenoise[brotli]'

$ pip freeze > requirements.txt

1. Open settings.py in your project’s main directory (e.g., mysite/settings.py).

Add the following to the MIDDLEWARE list, *immediately after* SecurityMiddleware:

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'whitenoise.middleware.WhiteNoiseMiddleware', ...

]

1. Still in settings.py, find the section where static files are configured.

Make the following modifications:

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.0/howto/static-files/

# This setting informs Django of the URI path from which your static files will be served to users

# Here, they well be accessible at your-domain.onrender.com/static/... or yourcustomdomain.com/static/...

STATIC\_URL = '/static/'

# This production code might break development mode, so we check whether we're in DEBUG mode

if not DEBUG: # Tell Django to copy static assets into a path called `staticfiles` (this is specific to Render)

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

# Enable the WhiteNoise storage backend, which compresses static files to reduce disk use

# and renames the files with unique names for each version to support long-term caching

STATICFILES\_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

All set! We’re ready to serve static content from our Django project on Render.

Create a build script

Whenever you deploy a new version of your project, Render runs a **build command** to prepare it for production. Let’s create a script for Render to run as this build command.

1. Create a new file called build.sh in your project’s root directory and paste in the following:
2. #!/usr/bin/env bash
3. # Exit on error
4. set -o errexit
5. # Modify this line as needed for your package manager (pip, poetry, etc.)
6. pip install -r requirements.txt
7. # Convert static asset files
8. python manage.py collectstatic --no-input
9. # Apply any outstanding database migrations

python manage.py migrate

Make sure the script is executable before adding it to version control:

chmod a+x build.sh

We’ll configure Render to run this build script whenever a new deploy is initiated.

1. We’ll run your project with [Uvicorn](https://www.uvicorn.org/" \t "_blank) and [Gunicorn](https://gunicorn.org/" \t "_blank). Add these dependencies to your project:
2. pip install gunicorn uvicorn

pip freeze > requirements.txt

1. Try running your project locally!

Replace mysite in the command below with your project’s name.

python -m gunicorn mysite.asgi:application -k uvicorn.workers.UvicornWorker

1. Visit [http://localhost:8000](http://localhost:8000/) in your browser to verify that your project is up and running.

Commit all changes and push them to your repository. Your project is ready to deploy to Render!

Deploying to Render

There are two ways to deploy your Django project on Render, either by [declaring your services within your repository](https://docs.render.com/infrastructure-as-code) using a render.yaml file or by manually setting up your services using the dashboard. In this tutorial, we will walk through both options.

Use render.yaml for deploys

1. Create a file named render.yaml in the root of your project. This file will define your Django **web service**, along with the [**database**](https://docs.render.com/databases) it connects to. Don’t forget to commit and push it to your repository.

The gunicorn command in the highlighted line below assumes your Django project is named mysite. Update it for your project as needed.

databases:

- name: mysitedb

plan: free

databaseName: mysite

user: mysite

services:

- type: web

plan: free

name: mysite

runtime: python

buildCommand: "./build.sh"

startCommand: "python -m gunicorn mysite.asgi:application -k uvicorn.workers.UvicornWorker" envVars:

- key: DATABASE\_URL

fromDatabase:

name: mysitedb

property: connectionString

- key: SECRET\_KEY

generateValue: true

- key: WEB\_CONCURRENCY

value: 4

1. In the Render Dashboard, go to the [Blueprints page](https://dashboard.render.com/blueprints) and click **New Blueprint Instance**.
2. Select the repository that contains your blueprint and click **Connect**.
3. Give your blueprint project a name and click **Apply**.

That’s it! Your project will be live at its .onrender.com URL as soon as the build finishes.

Manual deployment

1. Create a new [PostgreSQL database](https://docs.render.com/databases) on Render. Copy its **internal database URL** for now—you’ll need it later.
2. Create a new **web service** on Render, pointing it to your project’s GitHub/GitLab/Bitbucket repository (give Render permission to access it if you haven’t already).
3. Select Python for the runtime and set the following properties (replace mysite with your project’s name):

| Property | Value |
| --- | --- |
| **Build Command** | ./build.sh |
| **Start Command** | python -m gunicorn mysite.asgi:application -k uvicorn.workers.UvicornWorker |

1. Add the following environment variables under **Advanced**:

| Key | Value |
| --- | --- |
| DATABASE\_URL | The **internal database URL** for the database you created above |
| SECRET\_KEY | Click **Generate** to get a secure random value |
| WEB\_CONCURRENCY | 4 |

That’s it! Save your web service to deploy your Django application on Render. It will be live on your .onrender.com URL as soon as the build finishes.

Create a Django admin account

Once your application is live, create a new [Django admin account](https://docs.djangoproject.com/en/3.0/intro/tutorial02/#creating-an-admin-user) by running the following command in the Render Shell:

python manage.py createsuperuser

See [Setting your Python Version](https://docs.render.com/python-version) if you need to customize the version of Python used for your app.

Creating a new Django project

This section walks through setting up a Django project and adding an application with a simple view.

The finished code for this example is available on [GitHub](https://github.com/render-examples/django), and you can view the project running [here](https://django.onrender.com/).

This tutorial starts with a bare-bones installation and explains all required code modifications. Feel free to adapt it with custom configuration as needed.

Installation & setup

First, we’ll set up our local development environment and create a basic project structure.

We’ll call our project mysite. You can use a different name, but make sure to modify all commands that use mysite below to match the name you choose.

**1. Create your directory and virtual environment**

Run the following commands in your terminal (see comments for descriptions):

# Create a new project directory and cd into it

mkdir mysite

cd mysite

# Create a virtual environment using Python's venv package

python -m venv venv

# Activate the virtual environment to start installing other packages

source venv/bin/activate

**2. Install Django and create your project**

Run the following from your project’s root directory to install Django:

# This installs Django 5.0.1 (feel free to modify the version as needed)

pip install django==5.0.1

pip freeze > requirements.txt

Then run the following to initialize your mysite Django project:

django-admin startproject mysite .

You’ll end up with the following directory structure:

.

├── manage.py

├── mysite

│ ├── \_\_init\_\_.py

│ ├── asgi.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

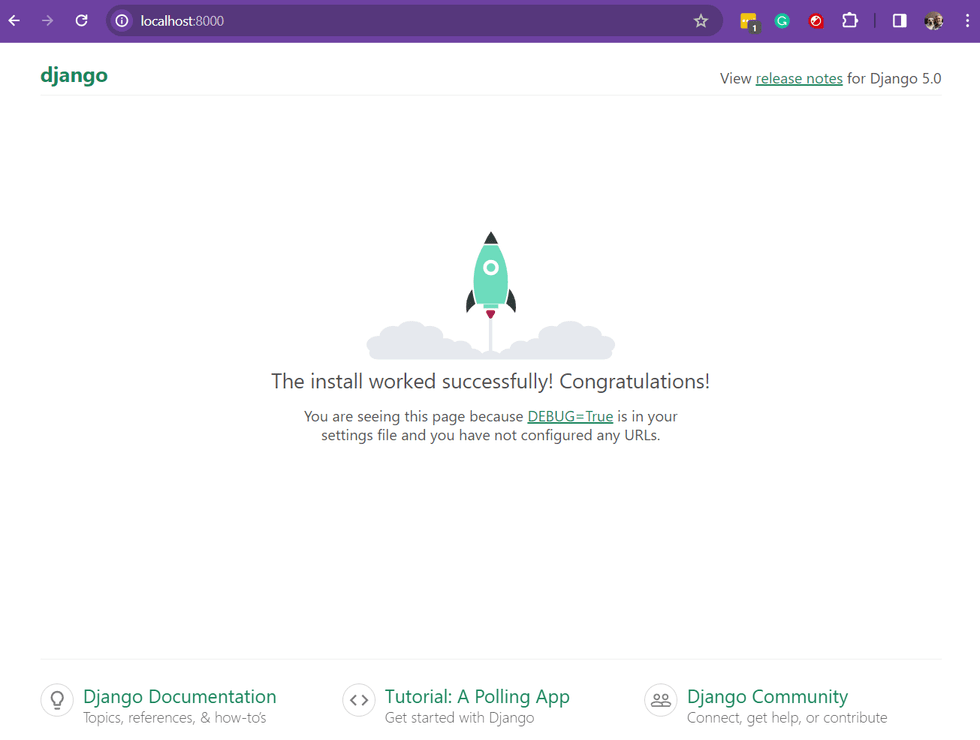
└── venv (you can ignore everything in here for now)

├──

You now have a fully functional scaffold for your new Django project! To verify, you can start the development server:

python manage.py runserver

Then visit [http://localhost:8000](http://localhost:8000/) in your browser:



Creating a landing page

Next, we’ll add an *application* to our Django project that serves a simple landing page.

Django *projects* contain one or more *applications*, which are wired together to form a complete service. Django provides several built-in applications, such as its [admin site](https://docs.djangoproject.com/en/3.0/ref/contrib/admin/).

**1. Create a new application**

1. Run the following command from your project’s root directory:

python manage.py startapp homepage

This creates a directory for a new application named homepage with following contents:

homepage

├── \_\_init\_\_.py

├── admin.py

├── apps.py

├── migrations

│ └── \_\_init\_\_.py

├── models.py

├── tests.py

└── views.py

1. We need to inform Django about this new homepage application. Open mysite/settings.py and find the the INSTALLED\_APPS setting. Add a reference to the HomepageConfig class to the *beginning* of the list:
2. INSTALLED\_APPS = [
3. 'homepage.apps.HomepageConfig', 'django.contrib.admin',
4. 'django.contrib.auth',
5. ...

]

For more information about the INSTALLED\_APPS setting, see the [Django settings information page](https://docs.djangoproject.com/en/5.0/ref/settings/#installed-apps)

**2. Write your first view**

Let’s create a simple view in our homepage application that we’ll serve from our service’s root path. First we’ll define the view, then we’ll configure its routing.

1. Create a new file at homepage/templates/homepage/index.html. Paste in the following HTML:
2. <!doctype html>
3. <html lang="en">
4. <head>
5. <meta charset="UTF-8">
6. <meta http-equiv="X-UA-Compatible" content="IE=edge" />
7. <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8. <title>Hello Django on Render!</title>
9. <link rel="stylesheet"
10. href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
11. integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
12. crossorigin="anonymous">
13. </head>
14. <body>
15. <main class="container">
16. <div class="row text-center justify-content-center">
17. <div class="col">
18. <h1 class="display-4">Hello World!</h1>
19. </div>
20. </div>
21. </main>
22. </body>

</html>

Nothing fancy

1. In the file homepage/views.py, add the following Python code for the index method:
2. from django.shortcuts import render
3. # Create your views here.

def index(request): return render(request, 'homepage/index.html', {})

This method renders the homepage/index.html template we just created.

1. Create a *new* file named homepage/urls.py and paste in the following:
2. from django.urls import path
3. from . import views
4. urlpatterns = [
5. path('', views.index, name='index'),

]

This file tells Django that you want to serve the index view from your service’s root URL.

1. Configure the Django project to point to the homepage application’s urls module. Open mysite/urls.py and update the from django.urls import path line to add the include module, and then include the homepage application’s URLs module to our urlpatterns, like so:
2. from django.contrib import admin
3. from django.urls import path, include
4. urlpatterns = [
5. path('admin/', admin.site.urls),

path('', include('homepage.urls')),]

Your folder structure should now look like this:

render

├── \_\_init\_\_.py

├── admin.py

├── apps.py

├── migrations

│ └── \_\_init\_\_.py

├── models.py

├── templates

│ └── render

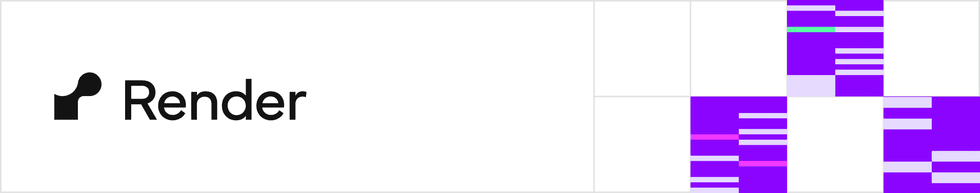
│ └── index.html

├── tests.py

├── urls.py

└── views.py

1. Add a static file to your application. Download this image and save it as homepage/static/homepage/render-banner.png:



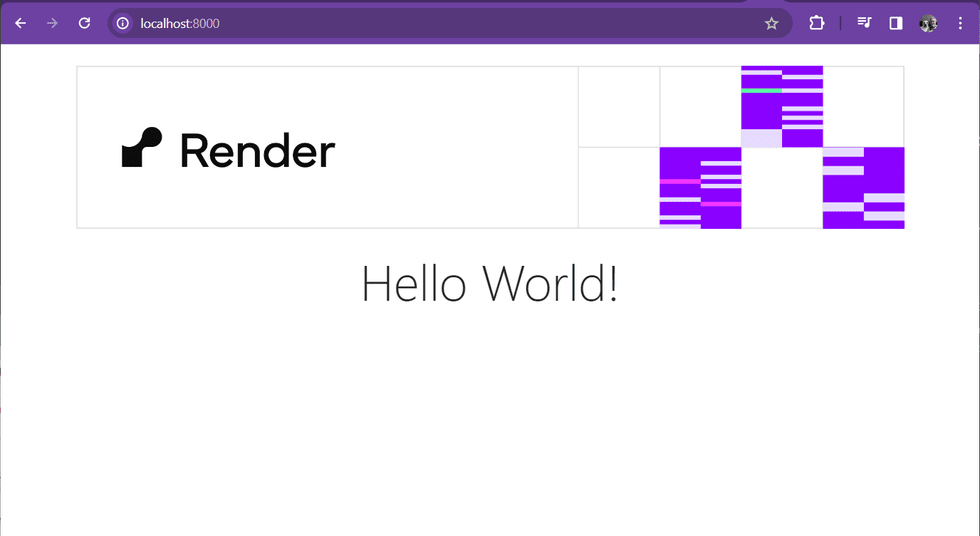
1. Return to your index.html template. Load the static module and reference the downloaded image in a new <header></header> block:
2. {% load static %}
3. <!doctype html>
4. <html lang="en">
5. ...
6. <body>
7. <header class="container mt-4 mb-4"> <a href="https://render.com"> <img src="{% static "homepage/render-banner.png" %}" alt="Homepage banner" class="mw-100"> </a></header>...
8. </body>

</html>

1. You can now verify your app is working with the following command:

python manage.py runserver

Reload your browser tab at localhost:8000 to make sure the banner and text are showing properly.



Adding basic security

You should ensure that any production-level application is properly secured and configured. The Django documentation provides a useful [deployment checklist](https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/), which we’ll follow in this step.

1. Open mysite/settings.py and find the declaration of the SECRET\_KEY setting. We don’t want to store production secrets in source code, so we’ll fetch this value from an environment variable that we’ll create later:
2. # Don't forget to import os at the beginning of the file

import os

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = os.environ.get('SECRET\_KEY', default='your secret key')

1. Find the declaration of the DEBUG setting. This setting should *never* be set to True in a production environment. You can detect that you’re running on Render by checking for the presence of the RENDER environment variable in the [application environment](https://docs.render.com/environment-variables):
2. # SECURITY WARNING: don't run with debug turned on in production!

DEBUG = 'RENDER' not in os.environ

1. When DEBUG is False, Django requires a suitable value for ALLOWED\_HOSTS. You can get the name of your web service host from the RENDER\_EXTERNAL\_HOSTNAME environment variable, which is automatically set by Render.
2. # https://docs.djangoproject.com/en/3.0/ref/settings/#allowed-hosts
3. ALLOWED\_HOSTS = []

RENDER\_EXTERNAL\_HOSTNAME = os.environ.get('RENDER\_EXTERNAL\_HOSTNAME')if RENDER\_EXTERNAL\_HOSTNAME: ALLOWED\_HOSTS.append(RENDER\_EXTERNAL\_HOSTNAME)

If you add a custom domain to your service, don’t forget to add it to this list.

Adding PostgreSQL support

We will add the [DJ-Database-URL](https://github.com/jacobian/dj-database-url) package, which allows us to specify databases in Django using connection strings. Render provides database connection strings in the [Render Dashboard](https://dashboard.render.com/), which you will provide to your web service via the DATABASE\_URL environment variable. We will also need to add [psycopg2](https://www.psycopg.org/) to the project.

1. Run the following command to add the necessary dependencies to your project:
2. $ pip install dj-database-url psycopg2-binary

$ pip freeze > requirements.txt

1. In mysite/settings.py, find the declaration of the DATABASES setting and modify it to look like the following:
2. # Import the dj-database-url package at the beginning of the file

import dj\_database\_url

# Database documentation https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {

'default': dj\_database\_url.config(

# Replace this value with your local database's connection string. default='postgresql://postgres:postgres@localhost:5432/mysite', conn\_max\_age=600 )}

This connection string assumes that you have PostgreSQL running on localhost, on port 5432, with a database named mysite and a user named postgres with the password postgres.

1. Apply our database migrations. A “migration” is a step-by-step process of creating database tables and fields for our Django project. We will use the migrate command to run the migrations:
2. python manage.py migrate
3. Operations to perform:
4. Apply all migrations: admin, auth, contenttypes, sessions
5. Running migrations:
6. Applying contenttypes.0001\_initial... OK
7. Applying auth.0001\_initial... OK
8. Applying admin.0001\_initial... OK
9. Applying admin.0002\_logentry\_remove\_auto\_add... OK
10. Applying admin.0003\_logentry\_add\_action\_flag\_choices... OK
11. Applying contenttypes.0002\_remove\_content\_type\_name... OK
12. Applying auth.0002\_alter\_permission\_name\_max\_length... OK
13. Applying auth.0003\_alter\_user\_email\_max\_length... OK
14. Applying auth.0004\_alter\_user\_username\_opts... OK
15. Applying auth.0005\_alter\_user\_last\_login\_null... OK
16. Applying auth.0006\_require\_contenttypes\_0002... OK
17. Applying auth.0007\_alter\_validators\_add\_error\_messages... OK
18. Applying auth.0008\_alter\_user\_username\_max\_length... OK
19. Applying auth.0009\_alter\_user\_last\_name\_max\_length... OK
20. Applying auth.0010\_alter\_group\_name\_max\_length... OK
21. Applying auth.0011\_update\_proxy\_permissions... OK
22. Applying auth.0012\_alter\_user\_first\_name\_max\_length... OK

Applying sessions.0001\_initial... OK

Finishing setup

As a final step, jump back to the top of the document and read the section about [Updating an existing Django project](https://docs.render.com/deploy-django#updating-an-existing-django-project) to make sure your project is production-ready. Then, follow the instructions in [Deploying to Render](https://docs.render.com/deploy-django#deploying-to-render) to deploy your project.

[Deploy a Django App on Render](https://docs.render.com/deploy-django)

* [**Updating an existing Django project**](https://docs.render.com/deploy-django#updating-an-existing-django-project)
  + [Use a Render PostgreSQL database](https://docs.render.com/deploy-django#use-a-render-postgresql-database)
  + [Set up static file serving](https://docs.render.com/deploy-django#set-up-static-file-serving)
  + [Create a build script](https://docs.render.com/deploy-django#create-a-build-script)
* [**Deploying to Render**](https://docs.render.com/deploy-django#deploying-to-render)
  + [Use render.yaml for deploys](https://docs.render.com/deploy-django#use-renderyaml-for-deploys)
  + [Manual deployment](https://docs.render.com/deploy-django#manual-deployment)
  + [Create a Django admin account](https://docs.render.com/deploy-django#create-a-django-admin-account)
* [**Creating a new Django project**](https://docs.render.com/deploy-django#creating-a-new-django-project)
  + [Installation & setup](https://docs.render.com/deploy-django#installation--setup)
  + [Creating a landing page](https://docs.render.com/deploy-django#creating-a-landing-page)
  + [Adding basic security](https://docs.render.com/deploy-django#adding-basic-security)
  + [**Adding PostgreSQL support**](https://docs.render.com/deploy-django#adding-postgresql-support)
  + [Finishing setup](https://docs.render.com/deploy-django#finishing-setup)

Did this page help?

* [Twitter](https://twitter.com/render)
* [LinkedIn](https://www.linkedin.com/company/renderco/)
* [GitHub](https://github.com/renderinc)

# ## 48 – Desplegar aplicacion – parte 4

El profe dice que debio instalar poetry porque hace muchas instalackiones con esto en la guia.

Pero ya no lo hizo asi que seguiremos a otromodo.

### #-- Generar archivo de requirements.txt

**(venv) C:\Excellent>pip freeze > requirements.txt**

(venv) C:\Excellent>

**El archivo requirements contiene todos los modulo que necesita mi projecto para funcionar :**

asgiref==3.8.1

Brotli==1.1.0

dj-database-url==2.1.0

Django==5.0.6

gunicorn==22.0.0

packaging==24.0

psycopg2-binary==2.9.9

sqlparse==0.5.0

typing\_extensions==4.11.0

tzdata==2024.1

whitenoise==6.6.0

**En el archivo build.sh temenos para reemplazar lo que seguramente poetry install hace : (Entonces con esta instruccion estan las dependencias)**

#poetry install

pip install -r requirements.txt

**puedo re-ejecutar de nuevo si deseo el comando : pip install -r requirements.txt**

**(venv) C:\Excellent>pip install -r requirements.txt**

Requirement already satisfied: asgiref==3.8.1 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 1)) (3.8.1)

Requirement already satisfied: Brotli==1.1.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 2)) (1.1.0)

Requirement already satisfied: dj-database-url==2.1.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 3)) (2.1.0)

Requirement already satisfied: Django==5.0.6 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 4)) (5.0.6)

Requirement already satisfied: gunicorn==22.0.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 5)) (22.0.0)

Requirement already satisfied: packaging==24.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 6)) (24.0)

Requirement already satisfied: psycopg2-binary==2.9.9 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 7)) (2.9.9)

Requirement already satisfied: sqlparse==0.5.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 8)) (0.5.0)

Requirement already satisfied: typing\_extensions==4.11.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 9)) (4.11.0)

Requirement already satisfied: tzdata==2024.1 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 10)) (2024.1)

Requirement already satisfied: whitenoise==6.6.0 in c:\excellent\venv\lib\site-packages (from -r requirements.txt (line 11)) (6.6.0)

# ## 49 – DESPLEGAR APLICACION – PARTE FINAL

**Init git**

(venv) C:\Excellent>git init

Initialized empty Git repository in C:/Excellent/.git/

**El profe quiere eliminar:**

(venv) C:\Excellent>del .git #en linux o mac puede hacer rm -rf .git

C:\Excellent\.git\\*, Are you sure (Y/N)? Y

**Poner archivos a ignorar:**

db.sqlite3

venv

\_\_pycache\_\_

A screenshot of a computer

Description automatically generated

Luego hacer :

(venv) C:\Excellent>git init

Initialized empty Git repository in C:/Excellent/.git/

Crear Repositorio :

(venv) C:\Excellent>git add .

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/metadata/importlib/\_\_init\_\_.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/metadata/importlib/\_compat.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/metadata/importlib/\_dists.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/metadata/importlib/\_envs.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/metadata/pkg\_resources.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/\_\_init\_\_.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/candidate.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/direct\_url.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/format\_control.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/index.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/installation\_report.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/link.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/scheme.py', LF will be replaced by CRLF the next time Git touches it

warning: in the working copy of 'venv/Lib/site-packages/pip/\_internal/models/search\_scope.py', LF will be replaced by CRLF the next time Git touches it

etc

etc

**Averiguando que es esto (ya que al profe no le paso) :**

segun link: https://stackoverflow.com/questions/5834014/lf-will-be-replaced-by-crlf-in-git-what-is-that-and-is-it-important

if you are a single developer working on a windows machine, and you don't care that git automatically replaces LFs to CRLFs, you can turn this warning off by typing the following in the git command line

**git config core.autocrlf true**

Git can handle this by auto-converting CRLF line endings into LF when you add a file to the index, and vice versa when it checks out code onto your filesystem. You can turn on this functionality with the core.autocrlf setting. If you’re on a Windows machine, set it to true – this converts LF endings into CRLF when you check out code:

**$ git config --global core.autocrlf true**

If you’re on a Linux or Mac system that uses LF line endings, then you don’t want Git to automatically convert them when you check out files; however, if a file with CRLF endings accidentally gets introduced, then you may want Git to fix it. You can tell Git to convert CRLF to LF on commit but not the other way around by setting core.autocrlf to input:

**$ git config --global core.autocrlf input**

This setup should leave you with CRLF endings in Windows checkouts, but LF endings on Mac and Linux systems and in the repository.

If you’re a Windows programmer doing a Windows-only project, then you can turn off this functionality, recording the carriage returns in the repository by setting the config value to false:

**$ git config --global core.autocrlf false**