

video 1

en este video nos enseña a instalar python, crear un entorno virtual, descargar django en el entorno virtual e iniciar un nuevo proyecto desde django

para instalar python usamos "sudo apt install python3"

para hacer en entorno virtual he utilizado "python3 -m venv nombre_del_entorno_virtual"

para descargar el django he utilizado "pip3 install django"

para crear el proyecto tenemos que poner lo siguiente "python3
localizacion_De_django_admin.py startproject nombre_del_proyecto"

cabezazos: para encontrar "django-admin.py" tube que ebuscar carpeta por carpeta hasta encontrar porque en el video esta en otra ubicación posiblemente por que lo este haciendo en windows, el archivo está en
"ven/lib64/python3.6/site-packages/django/bin/django-admin.py"

video 2

en este video aprendemos a iniciar el servidor del entorno de desarrollo,
para ello tenemos que inciar el programa mange.py con la opcion de run server
"python3 mange.py runserver"



video 3

vamos a crear un vínculo entre la base de datos y el proyecto para crear el vínculo usamos “python3 manage.py migrate”

video 4

creamos un super usuario para el entorno de trabajo usando “python3 manage.py createsuperuser”

nos dirigimos a settings.py y cambiamos “LANGUAGE_CODE” a “es” para cambiar el idioma a español

```
LANGUAGE_CODE = 'es'
```

creamos un usuario desde el navegador

video 5

vamos a crear una aplicacion usando la terminal “python3 manage.py startapp boletin”

configuramos en settings.py para añadir la aplicacion al entorno de trabajo para ello vamos a modificar “INSTALLED_APPS” y añadimos “boletin”

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

video 6

vamos a crear un modelo para registrar a los usuarios para ello tenemos que modificar models.py

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

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

para guardar todas las modificaciones echas tenemos que utilizar “python3 manage.py makemigrations” y después “python3 manage.py migrate”

video 7

vamos a crear objetos y guardarlos en la base de datos desde la shell para acceder “python3 ”

tenemos que hacer una importación del modelo que hemos creado usando “from boletin.models import Registrado”

después creamos una nueva variable del modelo

```
>>> gente = Registrado.objects.all()
```

ahora vamos a guardar objetos en la lista

```
Registrado.objects.create(nombre='sergio',email='sergio@hostname')
```

para poder objetos en la lista desde al interfaz debemos importar el modelo en admin.py y registrarlo en administracion para acceder al el

```
from .models import Registrado  
  
admin.site.register(Registrado)
```

video 8

ahora vamos a personalizar el display del modelo modificando admin.py

```
class AdminRegistrado(admin.ModelAdmin):  
    list_display = ["email", "nombre", "timestamp"] # You, seconds ago • Uncommitted changes  
    #list_display_links = ["nombre"] #campo que tienen el link hacia la cuenta del usuario  
    list_filter = ["timestamp"]  
    list_editable = ["nombre"] #campo que se permite modificar  
    search_fiels = ["email", "nombre"] #campos que se pueden buscar  
    You, seconds ago | 1 author (You)  
    class Meta:  
        model = Registrado
```

The screenshot shows the Django Admin interface for the 'Registrados' model. The sidebar on the left contains navigation links for 'Inicio', 'Boletín', and 'Registrados'. The main content area displays a table with columns for 'EMAIL', 'NOMBRE', and 'TIMESTAMP'. A single record is visible with the email 'sergio@hostname' and name 'sergio'. The right-hand panel shows a 'FILTRO' (Filter) section with options for filtering by timestamp, including 'Cualquier fecha', 'Hoy', 'Últimos 7 días', 'Este mes', and 'Este año'. A 'Guardar' (Save) button is located at the bottom right of the table.

video 9

vamos a hacer una vista, para hacerla debemos dirigirlo a views.py para crear la vista y urls.py para asigne una url

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

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

video 10

configuramos la ubicación del directorio donde se tienen que buscar las plantillas, se configura en setting.py

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

y creamos en el directorio templates el archivo los archivos html que queramos

```
proyecto > templates > <> inicio.html
1 <h1>Hola mundo</h1>
```



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The main content area of the browser shows the text 'Hola mundo' in a large, bold, black serif font.

Video 11

vamos a crear formularios, para crear el formulario tenemos que crear un fichero forms.py en la aplicación y lo configuramos a nuestro gusto

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

video 12

vamos a añadir el formulario a la vista

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

después tenemos que añadir el formulario al html para que se puede rellenar

```
<form>  
{% el_form.as_p %}  
<input type="submit" value="Regístrame" />  
</form>
```



A screenshot of a web browser window. The address bar shows '127.0.0.1:8000'. The page content includes a heading 'Hola mundo', followed by two input fields labeled 'Nombre:' and 'Edad:'. Below these fields is a button labeled 'Regístrame'.

Video 13

vamos a configurar el formulario en el html para que guarde en la base de datos y hacer que no nos envíe a otra pagina



A screenshot of a web browser window, identical to the one in the previous block. It shows the same 'Hola mundo' heading, 'Nombre:' and 'Edad:' input fields, and the 'Regístrame' button.

video 14

vamos a poner "request.POST" para hacer referencia a lo que introducimos en el formulario y "or None" para que no nos salga mensaje de que es un campo obligatorio

```
def inicio(request):
    form = RegForm(request.POST or None)
    context = {
        "el_form" : form,
    }
    return render (request, "inicio.html", context)
```

Hola mundo

- Este campo es obligatorio.

Nombre:

- Este campo es obligatorio.

Edad:

después guardamos la información en el formulario en un formato limpio para poder usarla

```
def inicio(request):
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
    context = {
        "el_form" : form,
    }
    return render (request, "inicio.html", context)
```

video 15

vamos a hacer que el formulario almacene objetos nuevos usando nuestro modelo, primero tenemos que cambiar el formulario para que encaje con el modelo

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

después tenemos que almacenar esos datos en el modelo, para ello tenemos que importar el modelo y crear una variable donde se guarden los datos que queremos utilizar y otra para insertar los datos

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

# Create your views here.

def inicio(request):
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
        abc = form_data.get("email")
        abc2 = form_data.get("nombre")
        obj = Registrado.objects.create(email=abc, nombre=abc2)
    context = {
        "el_form" : form,
    }
    return render(request, "inicio.html", context)
```

<input type="checkbox"/>	EMAIL	NOMBRE
<input type="checkbox"/>	paco@localhost	<input type="text" value="paco"/>
<input type="checkbox"/>	sergio@hostname	<input type="text" value="sergio"/>

Hola mundo

Nombre:

Email:

video 16

vamos a crear un modelo de formulario par sustituir el que nos introduce django por defecto, tenemos que crear el modelo en forms.py y asigne los campos que queramos que muestre

```
You, 3 minutes ago | 1 author (You)
class RegModelForm(forms.ModelForm):
    You, 3 minutes ago | 1 author (You)
    class Meta:
        model = Registrado
        fields = ["nombre", "email"]
```

después nos vamos ha admin.py, importamos el modelo de formulario y lo sustituimos por el que tenemos por defecto

```
from django.contrib import admin

# Register your models here.
from .forms import RegModelForm
from .models import Registrado
You, seconds ago | 1 author (You)
class AdminRegistrado(admin.ModelAdmin):
    list_display = ["email", "nombre", "timestamp"]
    form = RegModelForm #modelo del formulario
    #list_display_links = ["nombre"] #campo que tienen el link haci
    list_filter = ["timestamp"]
    list_editable = ["nombre"] #campo que se permite modificar
    search_fiels = ["email", "nombre"] #campos que se pueden buscar
# class Meta:
#     model = Registrado

admin.site.register([Registrado, AdminRegistrado]) You, 2 days a
```

video 17

vamos a cambiar las restricciones del formulario, tenemos que editar el modelo del formulario y añadir las restricciones y el mensaje a cada

```
class RegModelForm(forms.ModelForm):
    You, 3 hours ago | 1 author (You)
    class Meta:
        model = Registrado
        fields = ["nombre", "email"]

    def clean_email(self):
        email = self.cleaned_data.get("email")
        email_base, proveedor = email.split("@")
        dominio, extension = proveedor.split(".")
        if not extension == "edu":
            raise forms.ValidationError("Por favor utiliza un email con la extension .EDU")
        return email
    You, seconds ago • Uncommitted changes
    def clean_nombre(self):
        nombre = self.cleaned_data.get("nombre")
        #validaciones
        return nombre
```

Añadir registrado

Por favor corrija el siguiente error.

Nombre:

Por favor utiliza un email con la extension .EDU

Email:

pepe@localhost.pe

video 18

primero vamos a poner el título como una variable que usaremos en html

```
You, 2 minutes ago | 1 author (You)
{{ titulo }}
<hr/>
<br/>
```

```
def inicio(request):
    titulo = "hola"
    form = RegForm(request.POST or None)
    if form.is_valid():
        form_data = form.cleaned_data
        abc = form_data.get("email")
        abc2 = form_data.get("nombre")
        obj = Registrado.objects.create(email=abc, nombre=abc2)
    context = {}
    "titulo" : titulo
    "el_form" : form,
```

HOLA

Nombre:

Email:

vamos ha hacer que si esta la cuenta iniciada nos saluden con el nombre de la cuenta

```
def inicio(request):
    titulo = "HOLA"
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)
```

Bienvenido patata

Nombre:

Email:

video 19

vamos a hacer que el formulario del html tenga el formato que hemos configurado para la página y vamos a hacer que si no inserta nombre se le asigne uno

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

# Create your views here.

def inicio(request):
    titulo = "HOLA"
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)
    form = RegModelForm(request.POST or None)

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

    if form.is_valid():
        instance = form.save(commic=False)
        if not instance.nombre:
            instance.nombre = "Persona"
        instance.save()

    context = {
        "titulo" : "Gracias %s" %(nombre)
    }

    if not nombre:
        context = {
            "titulo": "gracias %s" %(email)
        }
    # form_data = form.cleaned_data
    # abc = form_data.get("email")
    # abc2 = form_data.get("nombre")
    # obj = Registrado.objects.create(email=abc, nombre=abc2)

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

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

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

Bienvenido patata
patata

video 20

vamos a crear un formulario de contacto y como ya no utilizamos el Regform vamos a sustituirlo

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

tenemos que quitar la importación de Regform y poner el nuevo formulario

```
from .forms import RegModelForm, ContactForm
```

creamos la vista

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

creamos el fichero html y lo configuramos

```
o > templates > forms.html > form
{{ titulo }}<br/>
{{ request.user }}
<hr/>
<br/>

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

indicamos la url que le vamos a asignar

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

podemos usar una de las siguientes formas para obtener la información

```
if form.is_valid():
    for key, value in form.cleaned_data.iteritems():
        print key, value
    #for key in form.cleaned_data:
    #    print key
    #    print form.cleaned_data.get(key)
    #email = form.cleaned_data.get("email")
    #mensaje = form.cleaned_data.get("mensaje")
    #nombre = form.cleaned_data.get("nombre")
    #print email, mensaje nombre
```

Video 21

vamos a configurar el correo electrónico para enviar mensajes, vamos a utilizar la página de contacto para ello, primero tenemos que modificar settings para gmail

```
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'tu_email@gmail.com'
EMAIL_HOST_PASSWORD = 'tupassword'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

ahora tenemos que modificar views, le añadimos la importaciones para enviar email y nuestros setting

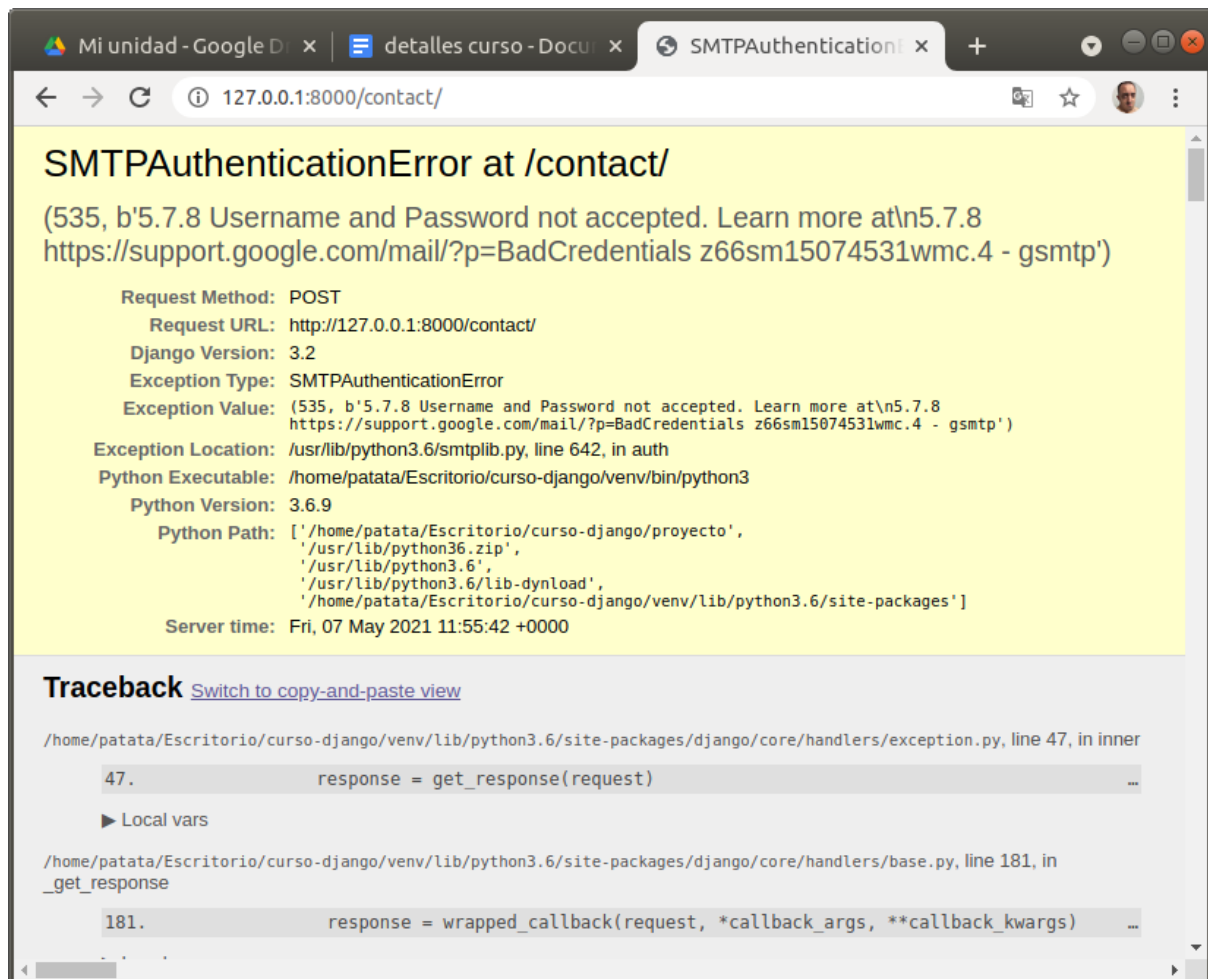
```
from django.conf import settings
from django.core.mail import send_mail
```

después configuramos la vista de contact

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

        #print email, mensaje nombre
    context = {
        "form": form,
    }
    return render(request, "forms.html", context)
```

comprobamos que funciona, es normal que nos salte el error ya que el correo que hemos como destinatario no existe, estamos biendo el error por que tenemos en la vista "fail_safety=True"



video 22

vamos a configurar nuestros archivos estáticos(css, images, javascript) para ello tenemos que comprobar que tenemos la opción 'django.contrib.staticfiles' y "STATIC_URL = '/static/'" en settings

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

```
STATIC_URL = '/static/'
```

indicamos dónde estarán los directorios estáticos y los que vamos a indicar como si fuera otro equipo

```
import os
```

```
STATICFILE_DIR = [
    BASE_DIR / "static_pro", "static"
    # "/var/www/static"
]

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

luego vamos a indicar donde se guardaran los fichero subido por los usuarios

```
STATIC_URL = '/media/'
```

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

primero vamos a confirmar que estamos en desarrollo, para ello tenemos que tener "DEBUG = True" en setting, des pues vamos configurar las url

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

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

por último vamos ha usar el siguiente comando para subir nuestro archivos estáticos al servidor y confirmamos

```
python3 manage.py collectstatic
```

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