

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 qu 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 manage.py con la opcion de run server
"python3 manage.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 email 'sergio@hostname' and name 'sergio'. The right sidebar contains a 'FILTRO' dropdown menu with options like 'Por timestamp', 'Cualquier fecha', 'Hoy', 'Últimos 7 días', 'Este mes', and 'Este año'.

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 import inicio # You, a minute ago  
  
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