Video 34

En el setting, el debug=True mientras trabajemos en local (modo desarrollo). Cuando querramos probar en el servidor, debug=False.

Cuando nuestra BBDD tiene una imagen, la misma queda referenciada en cualquier lado. Para dar un orden, se puede generar una carpeta nueva (por convención llamada media) y dentro del setting del proyecto:

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

Para ordenar las imágenes por apps, dentro del model:

imagen=models.ImageField(upload\_to='servicios')

Luego explica como agregar esas urls para poder ver las imágenes.

Video 35

Vinculamos el models de la app servicios a la vista que contiene la función con el renderizado de servicio que esta en la app ProyectoWebApp.

Explica como meter la querySet que devuelve un método llamado object.all() dentro de nuestra plantilla.

Video 36

En este video acomoda las vistas y templates de la app servicios dentro de la misma app (estaba todo en ProyectoWebApp)

Video 37

Da estilo al template de servicios.

Video 38

Crea la app con la BBDD y el registro de esa app para el admin para el contenido de blog.

Video 39

Crea el template de post, genera nuevos post para probar la pag.

Video 40

Sobre el template de post, agregamos el usuario que postea, mostramos las categorías con bucles anidados.

Video 41

Configuramos los paths de los blog para que los relacione

Video 42

Se linkea las categorías para que al hacer click dentro del blog, nos envie a la pagina que corresponde.

Video 43

Hacemos una nueva app para contacto, luego un formulario. Este lo hacemos usando una herramienta de Django.(básicamente una clase especial en un archivo llamado forms.py

Video 44

Damos formato al formulario. Breve comentario de csrf\_token (token de seguridad para formularios html para evitar que se envie info de otra web)

Video 45

Vemos que un método post guarda los valores en un diccionario. Eso lo podemos ver con un request.POST en el html

Video 46

Se arma la view para captar los datos enviado en el formulario.