Hito 03

{desafío}
latam_

Recursos antes de iniciar

- A continuación, se comparten un conjunto de recursos audiovisuales que permitirán abordar los Hitos de nuestro proyecto.

- Antes de ejecutar la solución de los Hitos, es importante que cada estudiante vea estos recursos en video.

Contenido dinámico en Django

- En Django, la muestra de contenido en plantillas se realiza mediante el uso de condiciones, iteradores y filtros. Estos elementos permiten personalizar la presentación de información en las plantillas HTML de acuerdo con ciertas reglas y lógica.

Contenido dinámico en Django

Condiciones (If Statements)

- Las condiciones en las plantillas de Django se utilizan para mostrar o no mostrar ciertas partes de contenido basándose en evaluaciones lógicas.

- Sintaxis básica:

```
{% if condicion %}
    <!-- Contenido a mostrar si la condición es verdadera -->
{% else %}
    <!-- Contenido a mostrar si la condición es falsa -->
{% endif %}
```

```
{% if user.is_authenticated %}
    <p>Bienvenido, {{ user.username }}.</p>
{% else %}
    <p>Por favor, inicia sesión.</p>
{% endif %}
```

Ejemplo de usuario autenticado

Contenido dinámico en Django

Iteradores (For Loops)

- Los iteradores permiten recorrer elementos iterables, como listas, diccionarios, o consultas de bases de datos, para mostrar contenido repetitivo.
- Sintaxis básica:

```
{% for elemento in lista %}
    <!-- Contenido a mostrar por cada elemento en la lista -->
{% endfor %}
```

```
<ul>
    {% for nombre in nombres %}
        <li>{{ nombre }}</li>
    {% endfor %}
</ul>
```

Muestra una lista de nombres.

Contenido dinámico en Django

Filtros

- Los filtros se utilizan para modificar o formatear el contenido que se va a mostrar en la plantilla.
- Sintaxis básica:

```
{{ variable|filtro }}
```

```
<p>Fecha: {{ fecha|date:"F d, Y" }}</p>
```

Ejemplo de muestra de fecha en un formato específico.

Revisión de ejercicio

Revisa en plataforma el video **“Introducción a las condiciones, iteradores y filtros”**

Repliquemos juntos este ejercicio.

Recurso audiovisual

{desafío} latam_

Formularios en Django

- En Django, los formularios (forms) son una parte fundamental para manejar la entrada de datos del usuario en una aplicación web.

- Los formularios en Django proporcionan una manera sencilla y eficiente de recopilar, validar y procesar datos enviados por el usuario.

Veamos algunos conceptos claves asociados a los formularios

Formularios en Django

Conceptos clave

django.forms Module

- El módulo django.forms proporciona clases y funciones para definir y trabajar con formularios en Django.
- Puedes importar este módulo en tus archivos de Python para crear y manejar formularios.

Formularios en Django

Conceptos clave

forms.Form Class

- La clase base para definir formularios en Django es forms.Form.
- Puedes crear una clase que herede de forms.Form para definir los campos del formulario y las reglas de validación.

```
from django import forms

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

Campos del Formulario

- Los campos del formulario se definen como instancias de clases en el formulario. Cada campo representa un elemento del formulario, como un campo de texto, un campo de correo electrónico, etc.

Formularios en Django

Conceptos clave

Widgets

- Los widgets definen cómo se muestra un campo en el navegador. Django proporciona una variedad de widgets predefinidos (por ejemplo, TextInput, EmailInput) que puedes usar o puedes personalizarlos.

```
class MiFormulario(forms.Form):
    nombre = forms.CharField(widget=forms.TextInput(attrs={'placeholder': 'Tu nombre'}))
    email = forms.EmailField(widget=forms.EmailInput(attrs={'placeholder': 'Tu correo electrónico'}))
```

Validación de Datos

- Django realiza automáticamente la validación de los datos del formulario. Puedes especificar reglas de validación utilizando atributos en los campos o escribir métodos específicos de validación en la clase del formulario.

```
class MiFormulario(forms.Form):
    edad = forms.IntegerField()

    def clean_edad(self):
        edad = self.cleaned_data['edad']
        if edad < 18:
            raise forms.ValidationError("Debes ser mayor de edad.")
        return edad
```

Renderizado en Plantillas

- Puedes renderizar formularios en plantillas HTML utilizando etiquetas de Django. Esto incluye la representación de campos, la gestión de errores y la inclusión de tokens de seguridad CSRF.

```
<form method="post" action="{% url 'mi_vista' %}">
    {% csrf_token %}
    {{ formulario.as_p }}
    <button type="submit">Enviar</button>
</form>
```

Formularios en Django

Conceptos clave

Procesamiento de Datos del Formulario

- Puedes procesar los datos del formulario en tu vista usando el método HTTP adecuado (POST o GET).
- Después de la validación, los datos estarán disponibles en form.cleaned_data.

```
from django.shortcuts import render
from .forms import MiFormulario

def mi_vista(request):
    if request.method == 'POST':
        formulario = MiFormulario(request.POST)
        if formulario.is_valid():
            # Procesar los datos
    else:
        formulario = MiFormulario()

    return render(request, 'mi_template.html', {'formulario': formulario})
```

Revisión de ejercicio

Revisa en plataforma el video **“Introducción a los formularios”**

Repliquemos juntos este ejercicio.

Recurso audiovisual

Generadores en Django
HTML

Generadores de Contenido

- Puede que te refieras a la generación dinámica de contenido en las vistas de Django. En este contexto, las vistas pueden generar contenido dinámico que se renderiza en las plantillas HTML. Este contenido puede provenir de consultas a la base de datos, lógica del negocio, u otras fuentes.

```
from django.shortcuts import render
from .models import MiModelo

def mi_vista(request):
    datos = MiModelo.objects.all()
    return render(request, 'mi_template.html', {'datos': datos})
```

Generadores en Django
Formularios

Generadores de Formularios

- En el contexto de formularios en Django, podrías estar pensando en la generación de formularios dinámicamente. Django proporciona formularios que se pueden definir de manera declarativa y renderizar automáticamente en plantillas HTML.

```
from django import forms

class MiFormulario(forms.Form):
    nombre = forms.CharField()
    edad = forms.IntegerField()
```

Generadores en Django

Formularios

Generadores de URLs

- En el contexto de la creación de URLs en Django, puedes utilizar funciones como reverse para generar URLs de manera dinámica basándote en nombres de patrones de URL.

```
from django.urls import reverse

url = reverse('nombre_del_patron')
```

Revisión de ejercicio

Revisa en plataforma el video **“Generadores”**

Repliquemos juntos este ejercicio.

Recurso audiovisual

{desafío} latam_

Proyecto: Creación de sitio web para una PYME de venta de pasteles y postres

Para revisar en detalle, dirígete al documento “Proyecto” del Módulo 6.

{desafío} latam_

Aprendizajes Esperados

Forms en Django:

- Utiliza las clases provistas por el framework Django para la integración de un formulario básico.
- Procesa un formulario Django utilizando plantillas(templates) para dar solución a un requerimiento.
- Implementa plantillas de formulario reutilizables para dar solución a un requerimiento.
- Maneja mensajes de errores de formularios en plantillas(templates) para dar solución a un requerimiento.

HITO 3

{desafío} latam_

Requerimientos Hito 03

¡Revisemos los requerimientos del Hito 03 “Añadiendo interacción a nuestro sitio web (modelos y formularios)”!

Actividad 01

Creación del Modelo Flan, Migraciones y Administración de Django

Primero deberemos crear un modelo llamado ***Flan*** que contenga los siguientes atributos:

- ***flan_uuid*** del tipo UUIDField
- ***name*** del tipo CharField (largo máximo 64 caracteres)
- ***description*** del tipo TextField
- ***image_url*** del tipo URLField
- ***slug*** del tipo SlugField
- ***is_private*** del tipo BooleanField

Una vez definido el modelo ***Flan*** en el archivo ***models.py*** del app ***web***, generaremos las migraciones con el comando ***makemigrations*** y las aplicaremos con el comando ***migrate***. Luego registraremos el modelo ***Flan*** en el panel de administración de Django, más tarde crearemos las credenciales de super usuario con el comando **python manage.py createsuperuser**, para finalmente ejecutar el servidor y visitar http://127.0.0.1:8000/admin/.

Actividad 01

Creación del Modelo Flan, Migraciones y Administración de Django

Agregaremos al menos 8 nuevos elementos ***Flan*** con su descripción, imagen, etc.

Al finalizar, generaremos un “pantallazo” de la lista de estos 8 elementos dentro del panel de administración Django y lo guardaremos bajo el formato jpg o png.

Para guiarte en este proceso, te recomendamos leer en detalle el documento Hito 3, as{i como la Lectura Referida Hito 3.

{desafío} latam_

Actividad 02

Presentación de Flanes al Público

Es hora de mostrar tus Flanes recién creados al público. Agregaremos el resultado de todos los Flanes existentes en nuestro sitio web como contexto a la vista de la ruta **/** (Index o Índice) e "imprimiremos" los resultados en la plantilla ***index.html*** a través del componente card de bootstrap, utilizando el ciclo for de las plantillas de django para mostrar cada uno de los ***Flan*** anteriormente creados.

La lista de flanes puede ser obtenida de la siguiente manera:

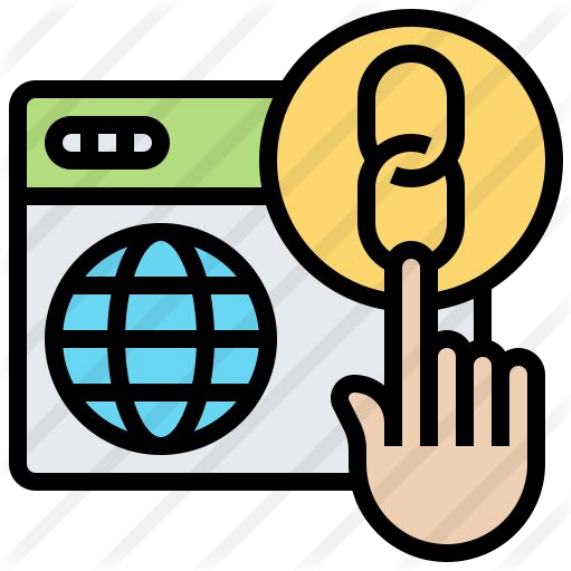
- `flanes = Flan.objects.all()`
- `flanes_privados = Flan.objects.filter(is_private=True)`
- `flanes_publicos = Flan.objects.filter(is_private=False)`

Para comenzar el trabajo de esta actividad, revisa en primer lugar el videotutorial del Hito 3, "Introducción a los condiciones, iteradores y filtros".

{desafío} latam_

Actividad 02

Presentación de Flanes al Público



Una vez hayamos logrado mostrar nuestros Flan recién creados en la página principal, deberemos agregar lo mismo en la página de bienvenida, esta vez diferenciando los contenidos de la siguiente manera:

- En la página principal(Index o Índice) se mostrarán solo los ***Flan*** cuyo atributo ***is_private*** es igual a ***False***
- En la página de bienvenida (Welcome o Bienvenida), se mostrarán solo los ***Flan*** cuyo atributo ***is_private*** es igual a ***True***

Finalizado lo anterior, visita las rutas **/** y ***/bienvenido*** de nuestro sitio web, realizaremos un "pantallazo" de cada una de las 2 rutas y lo guardaremos en formato jpg o png.

Recuerda revisar en detalle la Lectura Referida del Hito 3 para poder desarrollar de la mejor manera posible esta actividad.

{desafío} latam_

Actividad 03

Creación del Modelo ContactForm, Migraciones y Administración de Django

Primero deberemos crear un modelo llamado ***ContactForm*** que contenga los siguientes atributos:

- ***contact_form_uuid*** del tipo UUIDField, con valor por defecto uuid.uuid4, no editable
- ***customer_email*** del tipo EmailField
- ***customer_name*** del tipo CharField (largo máximo 64 caracteres)
- ***message*** del tipo TextField

Una vez aplicadas las migraciones, debemos registrar el modelo ***ContactForm*** en el archivo ***admin.py***
Crearemos una vista y una nueva url que permita mostrar el mensaje "Contacto" en la ruta ***/contacto***, esta ruta debe tener su propia plantilla html que extiende de la plantilla base. Agregaremos el enlace hacia esa ruta en el navbar de manera de permitir su fácil acceso.

{desafío} latam_

Actividad 03

Creación del Modelo ContactForm, Migraciones y Administración de Django

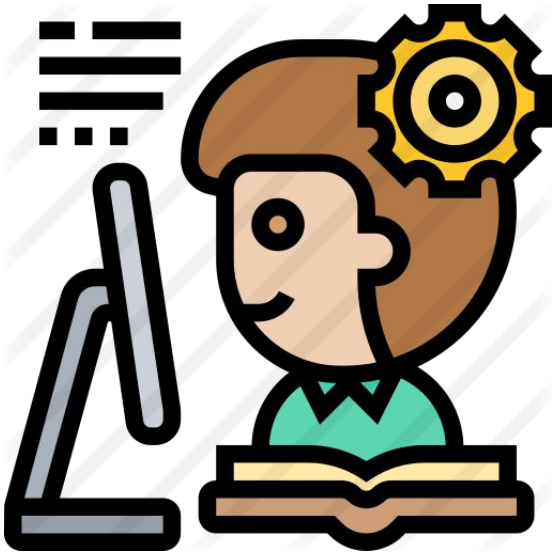
Para construir un formulario, debemos crear un archivo ***forms.py*** en el app ***web***, importar ***django.forms*** y crear un formulario llamado ***ContactFormForm***.
En la vista de contacto, debemos realizar las validaciones necesarias para validar el método de la solicitud(GET o POST), obtener su data, pasarle esa data a nuestro formulario, validarlo y luego redirigir al usuario a otra ruta.

Luego en nuestro formulario ***ContactFormForm*** debemos redirigir al usuario a la nueva url de éxito cuando el formulario sea válido, de manera de al llenar un formulario de contacto exitosamente, el usuario sea redirigido a la nueva vista de éxito y se le muestre un mensaje del estilo "Gracias por contactarte con OnlyFlans, te responderemos en breve", después debemos importar el modelo ***ContactForm*** en nuestro archivo ***web/views.py,*** luego procederemos a ejecutar el servidor de django y a navegar hasta el formulario de contacto, llenándolo con datos correctos que permitan la creación exitosa. Más tarde debemos modificar la plantilla de contacto de manera de mostrar los campos del formulario de forma individual, con el fin de lograr una página más estilizada. También quitaremos los colores de fondo de nuestros header, contenido y footer si es que aún los tenemos habilitados.

Actividad 03

Creación del Modelo ContactForm, Migraciones y Administración de Django

Una vez terminado todo lo anterior, modificaremos el contenido de la plantilla mostrada en la ruta ***/acerca*** para que contenga un texto sobre qué se trata la web, eliminaremos aquellos elementos sobrantes como la palabra “indice”, “bienvenido cliente”, “acerca” y “contacto”, finalmente generaremos los pantallazos que nos piden en formato jpg o png.

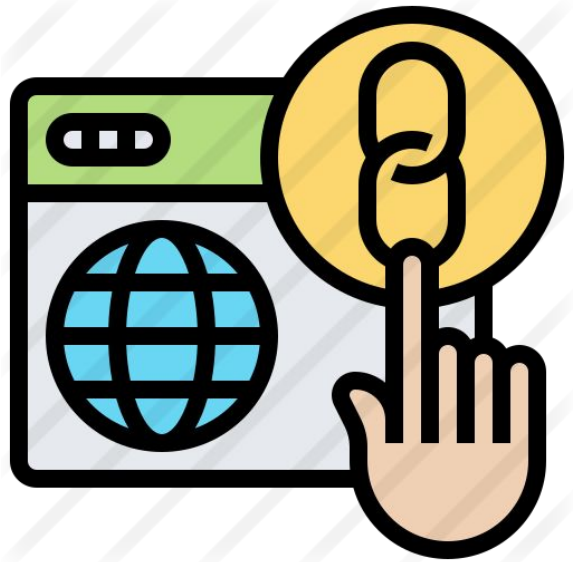


Para poder guiarte durante toda esta actividad, te sugerimos los siguientes materiales:

- **Lectura Referida Hito 3, especialmente el tutorial referido “Trabajando con formularios en Django (Español)”.**
- **Videotutorial “Introducción a los formularios”.**

Actividad 04

Migración del Formulario a ModelForm



Ya tenemos un sitio web que permite a sus usuarios enviar un formulario de contacto que queda registrado en nuestro panel de administración de Django. Lo que debemos hacer ahora es crear un modelo basado en ModelForm que permita reemplazar nuestro ***ContactFormForm***, lo podemos llamar ***ContactFormModelForm***.

Para poder comenzar con esto, revisaremos la Lectura Referida Hito 3, especialmente el apartado de “ModelForms (Inglés)”.

Actividad 04

Migración del Formulario a ModelForm

Luego reemplazaremos los llamados de ***ContactFormForm*** por ***ContactFormModelForm*** y probaremos el formulario de contacto para realizar los siguientes “pantallazos” que deberán ser guardados en formato jpg o png:

- Página Contacto(sin datos)
- Página Contacto(con dato de correo erroneo)
- Página Contacto(con dato de correo no erróneo)
- Página de éxito luego de enviar un contacto
- Detalle del contacto creado en el panel de administración de Django

Hint: El formulario debería lucir un poco diferente a como lo teníamos configurado en el punto anterior, no te preocupes por su estética, solo verifica su funcionamiento y genera los “pantallazos”, luego puedes volver a como tenias el formulario de contacto en el punto anterior o resolver los detalles estéticos de la forma que desees)

Actividades completadas

Ya llegado a este punto dentro del proyecto, deberías haber logrado lo siguiente:

- Crear el Modelo Flan, Migraciones y su administración en Django.
- Presentar los Flanes al Público.
- Crear el Modelo ContactForm, Migraciones y su administración en Django.
- Migrar el Formulario a ModelForm.

{desafío} latam_

{desafío} latam_

Academia de talentos digitales