TP 3 - Tutorial y Cuestionario sobre Django

MATERIA: Laboratorio de algoritmos y base de datos

ALUMNO: Agustin Quintela

CURSO: 4°AO

PROFESOR: Ignacio Garcia

TEMA: Django

Índice

Introducción

En este documento voy a responder las preguntas solicitadas en el cuestionario hablando sobre que es, como funciona y las características y diferentes funciones que tiene django. Específico información precisamente sobre los patrones MTV y MVC, qué es y cómo funciona el ORM, información sobre la interacción request-response entre la página y los usuarios, que es una aplicación y cómo se utiliza, entre otras varias cosas. Además al final hay un breve tutorial de los pasos a seguir para iniciar a usar django y como instalarlo.

¿Qué es Django y por qué lo usaríamos?

Django es un framework web de alto nivel basado en Python el cual sirve para desarrollar webs de manera rápida, pragmática y con un diseño limpio. Se encarga de las complicaciones del desarrollo web para que puedas concentrarte en escribir tu aplicación sin tener que hacer todo desde 0 y perder tiempo. Además este ayuda a desarrollar páginas rápidamente, seguras y es fácilmente escalable.
Nosotros lo podríamos usar para agregarle una parte interactiva a nuestras páginas, con una estructura más profesional. Además Django permite agregar otras funcionalidades como inicio de sesión, un control de administrador, interacciones y otras cosas.

Glosario: Framework web: Es un conjunto de herramientas que proporcionan una estructura para el desarrollo eficiente de aplicaciones web.

Fuentes: https://www.djangoproject.com/start/overview/

¿Qué es el patrón MTV en Django? Comparación con MVC

El patrón MTV es el enfoque de Django para organizar el código base y el flujo de trabajo de una aplicación web. Además este patrón arquitectónico proporciona una forma estructurada de organizar el código, separando las tareas y promoviendo la reutilización. Está compuesta por tres componentes que realizan funciones específicas cada uno.

Modelo : La capa Modelo representa los datos y el esquema de la base de datos de su aplicación. Define la estructura de las tablas de su base de datos, incluyendo campos, relaciones y restricciones. Django utiliza el Mapeo Objeto-Relacional (ORM) para mapear las clases de Python a las tablas de la base de datos, lo que facilita el trabajo con bases de datos sin necesidad de escribir consultas SQL manualmente.

Plantilla :La capa de plantilla define cómo se presentarán los datos al usuario. Las plantillas son archivos HTML con marcadores de posición para contenido dinámico. El motor de plantillas de Django permite incrustar código Python en ellas, lo que permite generar HTML dinámico.
Vista : La capa Vista actúa como puente entre el Modelo y la Plantilla. Las vistas son funciones o clases de Python que gestionan las solicitudes HTTP, recuperan datos del Modelo y los pasan a la Plantilla para su renderizado. Las vistas definen la lógica de las páginas web y determinan qué datos se muestran. Veamos ahora cómo estos componentes funcionan juntos en Django.

Tanto MVC (Modelo-Vista-Controlador) como MVT (Modelo-Vista-Plantilla) son patrones de diseño arquitectónico que se utilizan para separar las tareas de una aplicación en diferentes componentes. Si bien comparten un propósito similar (dividir una aplicación en capas para facilitar su gestión y escalabilidad), su implementación y las funciones de los componentes difieren ligeramente.

El patrón MVC se utiliza ampliamente en el desarrollo web para separar la aplicación en tres componentes interconectados:

Modelo: Representa los datos o la lógica de negocio de la aplicación. Gestiona los datos y la lógica, y actualiza la vista cuando los datos cambian.
Vista: Representa los elementos de la interfaz de usuario (IU). Muestra los datos del modelo al usuario, generalmente en formato HTML, pero también puede representar cualquier elemento de la IU (botones, campos de texto, etc.).
Controlador: Actúa como intermediario entre el Modelo y la Vista. Gestiona la entrada del usuario (como clics en botones, enVíos de formularios, etc.), actualiza el Modelo y, posteriormente, la Vista.

Si bien tanto MVC como MVT buscan separar las tareas de una aplicación, MVC divide la responsabilidad entre el Modelo, la Vista y el Controlador, mientras que MVT asigna roles diferentes al Modelo, la Vista y la Plantilla. La diferencia más notable es que en MVT, la Vista procesa las solicitudes y se coordina con el Modelo, mientras que en MVC, el Controlador desempeña esa función. Pero ambos sirven para la separación de datos, lógica y la presentación de la página.

Este es un pequeño cuadro comparativo entre el modelo MVC y el MTV.

Aspecto	MVC (Modelo-Vista-Controlador)	MVT (Modelo-Vista-Plantilla)
Componentes	Modelo, Vista, Controlador	Modelo, Vista, Plantilla
El papel de la vista	Muestra los datos al usuario (UI)	Maneja la entrada y la interacción del usuario; procesa la solicitud
Función de la plantilla	N / A	La plantilla es responsable de renderizar el HTML/CSS (Vista en MVC)
Rol del controlador	Maneja la entrada del usuario, la procesa y actualiza el modelo.	La vista maneja tanto la entrada del usuario como la representación de la respuesta, que en MVC se divide entre el controlador y la vista.
División de responsabilidades	La vista solo se ocupa de representar y mostrar datos.	La vista es responsable de la lógica del manejo de solicitudes e interacciones del usuario.
Flujo de datos	El usuario interactúa con la Vista, que llama al Controlador	El usuario interactúa con la Vista, que maneja la lógica y devuelve una Plantilla para renderizar
Ejemplos de marcos	Ruby on Rails, Angular, ASP.NET MVC	Django

Fuentes:
https://www.freecodecamp.org/news/how-django-mvt-architecture-works/
https://python.plainenglish.io/creating-an-mtv-model-template-view-architecture-in-django-59c6502e15c8
https://www.geeksforgeeks.org/software-engineering/difference-between-mvc-and-mvt-design-patterns/

¿Qué entendemos por app en Django?

En Django, una app es un módulo o componente autónomo de una aplicación web que encapsula una funcionalidad específica o un conjunto de características relacionadas para realizar una función. Además las aplicaciones están diseñadas para ser reutilizables, lo que facilita el desarrollo de aplicaciones web complejas con código modular y mantenible.

Las aplicaciones en django son altamente modulables. Cada app debe tener un propósito o función específica. Como autenticación de usuario, gestión de blogs o catálogos. Además cada aplicación es autónoma no depende de otras aplicación del mismo proyecto para su funcionalidad y pueden ser reutilizadas en otros proyectos, por lo que también pueden reutilizar aplicaciones ya creadas por otros usuarios.

La estructura de una aplicación por lo general sigue estos componentes:

Modelos: Define la estructura de datos y el esquema de la base de datos.

Views: Maneja el procesamiento de solicitudes y genera respuestas.

Templates:Contiene plantillas HTML para renderizar páginas.

URLs: Define patrones de URL y enrutamiento para la aplicación.

Static files: Almacena CSS, JavaScript y otros activos estáticos.

Management Commands:Proporciona comandos de gestión personalizados para la aplicación.
Tests: Contiene pruebas unitarias para garantizar que la funcionalidad de la aplicación funcione como se espera.

Fuentes:
https://www.codementor.io/@chirilovadrian360/apps-in-django-concept-free-samples-294vudyim5

¿Qué es el flujo request-response en Django?

El ciclo de solicitud-respuesta de Django es un concepto fundamental que describe cómo el framework web Django procesa las solicitudes HTTP entrantes y genera las respuestas HTTP correspondientes. Este ciclo es el mecanismo principal mediante el cual Django gestiona las interacciones de los clientes y sirve aplicaciones web. Este proceso funciona así:

El ciclo comienza cuando un cliente envía una solicitud HTTP al servidor Django. Esta solicitud incluye información como la URL solicitada, el método HTTP, las cabeceras y posibles datos adicionales.
Luego, el despachador de URLs de Django, configurado en el archivo del proyecto, examina la URL para determinar qué función de vista debe manejar la solicitud. Si la URL contiene parámetros, estos se pasan como argumentos a la vista correspondiente.
La vista procesa la solicitud, interactúa con los modelos y la base de datos si es necesario, y construye un objeto de respuesta HTTP que contiene el contenido que será enviado al cliente.
Antes de que esta respuesta se envíe, interviene el middleware de Django. El middleware son clases que pueden realizar acciones adicionales, como autenticación, registro o modificaciones de la solicitud o la respuesta. Pueden ejecutarse tanto antes de llegar a la vista como después de su ejecución, antes de que la respuesta salga del servidor.

Finalmente, la respuesta es enviada al cliente, donde el navegador la procesa, renderiza el contenido (como HTML) y muestra la página. Si se trata de datos (por ejemplo, JSON), el cliente puede usarlos para actualizar o construir partes de la interfaz.

Fuentes: https://clouddevs.com/django/request-response-cycle/

¿Qué es el concepto de ORM?

ORM es una función integrada que ofrece django la cual es utilizada como una capa de abstracción entre la base de datos utilizada y la lógica de la aplicación. En lugar de escribir consultas SQL directamente, podemos trabajar con objetos y métodos proporcionados por Django para manipular los datos de la base de datos de una manera más simple y rápida.

En Django, la ORM, utiliza modelos, que son clases de python que representan tablas en la base de datos. Cada modelo define los campos y las relaciones entre ellos. Además El ORM en Django ofrece muchas más características y funcionalidades, como la definición de relaciones entre modelos, la realización de consultas anidadas y la optimización de consultas mediante la selección selectiva de campos. Esto nos permite trabajar de manera eficiente y estructurada con la base de datos sin tener que preocuparnos por los detalles de implementación subyacentes.

Fuentes: https://jorgecespedes.hashnode.dev/orm-en-django

¿Qué son los templates en Django?

Los templates o plantillas son una parte fundamental de la estructura MTV en Django. Una plantilla de Django es básicamente un archivo HTML que también puede incluir CSS y JavaScript . El framework de Django utiliza estas plantillas para generar dinámicamente páginas web con las que los usuarios interactúan. Dado que Django gestiona principalmente el backend, las plantillas se utilizan para renderizar contenido dinámico y definir la estructura visual de una página web.

Django ofrece dos formas de organizar las plantillas, dependiendo del tamaño y la estructura del proyecto:

Un directorio de plantillas a nivel de proyecto, compartido entre todas las aplicaciones.

Directorios de plantillas a nivel de aplicación, útiles en proyectos grandes o cuando se necesitan diferentes diseños para diferentes aplicaciones.

Fuentes: https://www.geeksforgeeks.org/python/django-templates/

¿Cómo se lo instala?

Principalmente para instalar Django debes tener instalado y actualizado python, ya que al ser un framework de python es necesario tenerlo. Para instalar Django hay diferentes formas, la más fácil y recomendada es a través del pip install.Una vez creado y activado el entorno virtual en el cual vas a trabajar debes insertar el comando “ python -m pip install .e django/ “ para instalarlo en ese entorno virtual. Es recomendable realizar un entorno virtual así no descargar todos los archivos y paquetes en todo el dispositivo y solo en ese entorno para no generar conflictos de compatibilidad.