ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Contenido

Introducción:

En el presente documento trabajaremos en conocer el patrón de desarrollo conocido por su sigla MVC, que es utilizado en innumerables organizaciones y empresas dedicadas al software. Comenzaremos definiendo dicha sigla, y aprenderemos como el framework Django lo aplica con una pequeña variante, dando lugar a MTV.

MVC:

MVC es un patrón de diseño de software, una forma en la que vamos a estructurar/organizar nuestro código. Lo que este patrón o tipo de arquitectura nos indica, son un conjunto de técnicas y procesos para resolver problemas o situaciones de manera simple y ordenada.
En ese sentido el patrón MVC, nos sugiere que se debe “agrupar” u organizar tres aspectos clave de todo desarrollo para cumplir con esta arquitectura de trabajo.
Tendremos tres capas fundamentales:

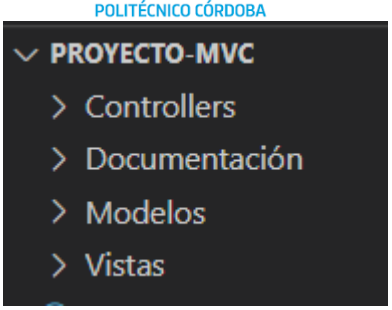
1. La lógica de acceso a la base de datos.
2. La lógica de presentación.
3. La lógica de negocio.

El punto número uno, queda comprendido en la M que significa Model (o modelo en español), y es la capa que nos ofrece la conexión e interacción con la base de datos.

El punto número dos, queda comprendido en la V que significa View (o vista en español), y es la capa que maneja la interacción con el usuario. Que se muestra, como se muestra, con qué estilos, todo se trabaja en esta capa.

El punto tres, queda comprendido en la C que significa Controller (o controlador en español), y es la capa que maneja la lógica de negocio del producto que se está desarrollando. Comprende las interacciones que provienen de la capa de las Views e interpreta según lo ingresado lo que tiene que pedir a la capa de Model para ser procesado y retornado a la capa View.

Entonces, al pensar nuestra organización del repositorio deberíamos tener esta estructura:



En esta organización de repositorio, vemos que está representada la arquitectura MVC o patrón de diseño MVC con sus carpetas para contener los archivos correspondientes a cada agrupación.
Además, con un agregado de una carpeta para tener organizada la documentación del proyecto.

MTV:

Django fue diseñado para desarrollar sitios web promoviendo el acople débil y la separación estricta entre las piezas de una aplicación/sitio. Siguiendo esta forma de trabajo, es fácil hacer cambios en un lugar específico de la aplicación/sitio sin afectar al resto.

Siguiendo con la lógica de MCV, en la sección de vistas, vemos la importancia de separar la lógica de negocios de la lógica con la que presentamos al usuario, utilizando un sistema de plantillas. En la capa de base de datos, aplicamos el mismo pensamiento para el acceso lógico de datos. En estas piezas juntas, (lógica de acceso a base de datos, de negocios y presentación) vemos claramente representado el modelo MVC.

En Django estas agrupaciones o definiciones cambian y se agrupan de la siguiente manera:

- M, significa Model (Modelo) que contiene la capa de acceso a la base de datos. Contendrá toda la información sobre los datos: cómo vamos a acceder, cuál es el comportamiento que tendrán, sus validaciones, relaciones, entre otros.

- T, significa Template (Plantilla) y contendrá la capa de presentación que reúne las decisiones que se tomaron en relación a la presentación (referido a una página web u otro tipo de documento interviniente).

- V, significa View (Vista) que en este modelo representa a la lógica de negocios y contendrá la lógica de acceso a los modelos y administración del template apropiado. Es como un puente entre los modelos y las plantillas.

MVC vs MTV:

En la interpretación de Django de MVC, la "vista" describe los datos que son presentados al usuario; no necesariamente el cómo se mostrarán, pero sí cuáles datos son presentados. En contraste, Ruby on Rails y frameworks similares sugieren que el trabajo del controlador incluya la decisión de cuales datos son presentados al usuario, mientras que la vista sea estrictamente el cómo serán presentados y no cuáles.
Dado lo expresado anteriormente, no hablaremos de un versus o una comparación entre el modelo MVC y el MTV, sino se deberá entender como la adaptación del framework Django hace del MVC para aprovechar sus beneficios al máximo en base a la estructura del mismo.

Referencias:

https://docs.djangoproject.com/es/3.0/faq/general/ (28/04/2023).

https://www.desarrollolibre.net/blog/python/el-patron-de-diseno-mtv-en-django-y-su-relacion-con-el-mvc (28/04/2023).

https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv (28/04/2023).