**Trabajo Práctico N.º 3: Django**

Facundo Manuel Benitez

Escuela técnica: Instituto Industrial Luis Augusto Huergo

Laboratorio de Algoritmos y Estructura de Datos

Profesor: Ignacio Miguel García

**Cuestionario:**

1. Django es un framework de alto nivel web y código abierto, escrito en Python, que facilita el rápido desarrollo de aplicaciones web robustas y seguras. Fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible. Prioriza la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes. Muchos utilizan Django para escalar de forma más rápida y flexible. Empezar a partir del marco de trabajo web Django es un modo más eficiente de crear una app web que comenzar desde cero, que requiere crear el backend, las API, javascript y los mapas del sitio. Con el marco de trabajo web Django, los desarrolladores web pueden centrarse en crear una aplicación exclusiva y obtienen una mayor flexibilidad que con las herramientas de desarrollo web.
2. El patrón MTV (Modelo-Vista-Pantalla) es la arquitectura en la que se basa Django. Este mismo se encarga de separar las diferentes capas de una aplicación web en componentes distintos. Esta separación mejora la mantenibilidad, la flexibilidad y la escalabilidad. La diferencia entre el patrón MTV y MVC (Modelo-Vista-Controlador) se encuentra en la forma sobre la cual se gestiona la lógica de la aplicación y la interacción del usuario. En MVC, el controlador se encarga de ser el intermediario entre el modelo y la vista , mientras que en el MTV, la vista se encarga de recuperar los datos del modelo y presentarlos a través de plantillas, siendo la vista quien maneja la interacción del usuario.
3. En Django, una app hace referencia a un componente modular y reutilizable que cumple una función específica dentro de un proyecto web más grande. Es decir, se encarga de realizar una tarea concreta, como la gestión de usuarios, la administración de un blog, o la creación de una API.

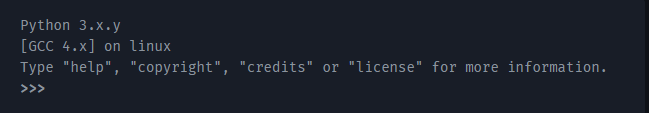
Cada aplicación en Django se caracteriza por lo siguiente:

* **Modularidad:** Las aplicaciones están diseñadas para ser independientes entre sí, por lo que pueden ser agregadas o eliminadas de un proyecto sin afectar a otras partes.
* **Reutilización:** Una app puede ser utilizada en distintos proyectos de Django, promoviendo la eficiencia y evitando la duplicación de código.
* **Especialización:** Cada app se enfoca en una función particular, como la autenticación, la gestión de contenido o el procesamiento de formularios.
* **Organización:** Ayuda a mantener el código bien estructurado, separando la lógica en diferentes partes del proyecto.

1. El flujo de request-response en Django hace referencia a cómo el framework se encarga de procesar una solicitud HTTP entrante y posteriormente, genera una respuesta HTTP para enviarla al cliente. Mediante este ciclo se maneja principalmente Django para gestionar las interacciones de los clientes y sirve aplicaciones web. El ciclo de solicitud se divide en varias partes:

* **Solicitud del cliente:** El ciclo comienza cuando un cliente envía una solicitud HTTP a un servidor web con Django, esta misma contiene información importante, como la URL solicitada, el método HTTP, las cabeceras y cualquier dato enviado en el cuerpo de la solicitud.
* **Enrutamiento de URL:** El despachador de URL de Django se encarga de examinar la URL de la solicitud entrante para determinar qué función de vista debe encargarse. El despachador de URL utiliza expresiones regulares o convertidores de rutas para asignar las URL a las funciones de vista.
* **Procesamiento de la función de vista:** Luego de que el despachador de URL identifique la función de vista adecuada, la invoca, pasando la solicitud HTTP como argumento. La función de vista contiene la lógica de la aplicación, procesa la solicitud, interactúa con modelos y base de datos, y prepara los datos para la respuesta.
* **Preparación de la respuesta:** Tras el procesamiento, la función de vista construye un objeto de respuesta HTTP. Este mismo contiene el contenido que se enviará al cliente, como contenido HTML, datos JSON o archivos binarios. También puede configurar encabezados de respuesta, cookies y códigos de estado dentro del objeto de respuesta.
* **Ejecución del middleware:** Previo al envió de respuesta al cliente, se ejecuta el middleware de Django. El middleware son funciones o clases que pueden realizar acciones como autenticación, modificación de solicitudes/respuestas, registro, etc. El middleware puede procesar la solicitud antes de que llegue a la función de vista o la respuesta antes de que salga del servidor.
* **Envío de la respuesta:** Finalmente, la respuesta HTTP se envía al cliente a través de la red. El navegador web del cliente procesa la respuesta, renderizando el contenido HTML, ejecutando JavaScript y mostrando la página web. En caso de que la respuesta incluya datos, el cliente también puede usarlos para interactuar o renderizar la página.
* **Repetir o finalizar:** El ciclo de solicitud-respuesta se repite cada vez que el cliente interactúe con la aplicación web mediante el envío de solicitudes adicionales. Cuando el cliente ya no necesita comunicarse con el servidor, el ciclo de solicitud-respuesta de esa interacción finaliza.

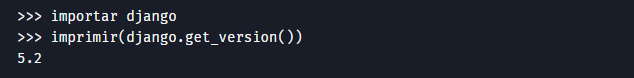
1. Un ORM, o Object-Relational Mapping, es un software diseñado para traducir entre las representaciones de datos utilizadas por bases de datos y las empleadas en la programación orientada a objetos. Estas dos formas de trabajar al no ser compatibles entre sí, un ORM permite salvar la brecha entre los diseños de datos de ambos sistemas. Un ORM permite trabajar con datos respaldados por bases de datos utilizando las mismas estructuras y mecanismos orientados a objetos que se usarían para cualquier tipo de datos internos. Los ORM funcionan como una capa de abstracción entre la aplicación y la base de datos, buscan aumentar la productividad del desarrollador eliminando la necesidad de código repetitivo y evitando el uso de técnicas complejas que podrían alterar los modismos y la ergonomía que se espera del lenguaje elegido.
2. Los templates de Django son los archivos HTML que definen la estructura y el contenido de tu aplicación web. Estos actúan como una capa visual de tu aplicación, permitiendo crear tus páginas web con elementos dinámicos. Estas plantillas generalmente, son utilizadas con vistas y modelos de Django para crear aplicaciones web completas. Las plantillas de Django permiten distinguir claramente entre el diseño visual y la lógica de la aplicación. Esto significa que es posible crear plantillas HTML que Django genera de manera dinámica utilizando los datos proporcionados. Para comprender este proceso, es necesario conocer algunos conceptos clave de Django, como: variables, etiquetas, filtros, herencia de plantillas y los resultados generados. Las etiquetas y los filtros en las plantillas de Django proporcionan herramientas poderosas para crear resultados HTML reutilizables. Las plantillas se estructuran en un lenguaje específico que separa la lógica del negocio de la lógica de presentac ión. Las etiquetas funcionan de manera similar a palabras clave o funciones, mientras que los filtros permiten modificar los datos antes de ser presentados en la interfaz.
3. Lectura
4. Para instalar Django primero debemos tener Python instalado. Para verificar que lo tengamos instalado podemos escribir python desde nuestra shell y deberíamos ver algo así:



Para instalar Django tenemos tres opciones:

* Instalar una versión oficial. Esta es la mejor opción para la mayoría de los usuarios.
* Instalar una versión de Django proporcionada por la distribución de su sistema operativo.
* Instalar la última versión de desarrollo. Esta opción es para entusiastas que buscan las últimas y mejores funciones y no temen ejecutar código completamente nuevo. Es posible que encuentres nuevos errores en la versión de desarrollo, pero reportarlos facilita el desarrollo de Django. Además, es menos probable que las versiones de paquete de terceros sean compatibles con la versión de desarrollo que con la última versión estable.

Para verificar que Python puede ver Django, escribimos python desde nuestra shell. Luego, en el prompt de Python, intente importar Django:



Y eso sería todo.

**Fuentes:**

**1)**<https://aws.amazon.com/es/what-is/django/> - <https://www.djangoproject.com>

**2)**<https://espifreelancer.com/mtv-django.html> <https://programacionfacil.org/blog/los-modelos-mvc-y-mvt-con-django/>

**3)**<https://access.workspace.google.com/remediate?urlparams=Aa6w02GPewRo_VBsMHjNRKkqWwG_tiNTTOneGbiu6Ov5mTtryF3M2rsVAK6L4Z92cWpUN2q3eOpJI_PfIdWLHqk&continue=https://translate.google.com/translate?u%3Dhttps://www.revsys.com/tidbits/what-is-a-django-app/%26hl%3Des%26sl%3Den%26tl%3Des%26client%3Drq>

**4)**[https://clouddevs.com/django/request-response-cycle/#:~:text=Django's%20request-response%20cycle%20is,and%20generates%20corresponding%20HTTP%20responses.](https://clouddevs.com/django/request-response-cycle/#:~:text=Django's%20request-response%20cycle%20is,and%20generates%20corresponding%20HTTP%20responses)

**5)**<https://clouddevs.com/django/request-response-cycle/#:~:text=Django's%20request%2Dresponse%20cycle%20is,and%20generates%20corresponding%20HTTP%20responses>.

**6)**<https://emmanueldav.medium.com/django-templates-mastering-the-basics-29a99813af9f>