Laboratorio de Algoritmos y Estructuras de Datos

# Trabajo Práctico Nº3


Estudiante: Franco Chichizola

Profesor: Ignacio Miguel Garcia

Curso: 4ºAO

Ins. Ind. LUIS A. Huergo

Fecha de entrega: 10/07/25

# ÍNDICE

# ¿Qué es Django?

Django es un *framework* web gratuito y de código abierto de alto nivel que permite el desarrollo rápido de aplicaciones web seguras y mantenibles que se crean en el lenguaje de programación Python.

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.

# ¿Por qué usar Django?

Versátil

Django puede ser usado para construir casi cualquier tipo de sitio web. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato.

Internamente, mientras ofrece opciones para casi cualquier funcionalidad que desees, también puede ser extendido para usar otros componentes si es necesario.

Seguro

Django ayuda a los desarrolladores a evitar varios errores comunes de seguridad al proveer un framework que ha sido diseñado para proteger el sitio web automáticamente.

Django puede validar si la contraseña ingresada es correcta enviándola a través de una función *hash* y comparando la salida con el valor hash almacenado.
Django permite protección contra algunas vulnerabilidades de forma predeterminada, incluida la *inyección SQL*, scripts entre sitios, falsificación de solicitudes entre sitios y *clickjacking*.

Escalable

Django usa un componente basado en la arquitectura *shared-nothing*. Teniendo en cuenta una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar hardware en cualquier nivel: servidores de caché, servidores de bases de datos o servidores de aplicación.

Velocidad

Django está diseñado para ayudar a los desarrolladores para hacer aplicaciones web lo más rápido posible.

# ¿Qué es el patrón MTV Django?

El patrón MTV es una forma de organizar el código en aplicaciones. Las siglas en inglés MTV corresponden a Model, Template y View. Model es la capa que se encarga de comunicar e interactuar con la base de datos, Template cumple la función de vista, y es la que se emplea para mostrar los datos al usuario, View es la capa que se encarga de manejar la lógica, a través de la que pasarán los datos del modelo a la plantilla.

# ¿Qué es el patrón MVC Django?

El patrón MVC es muy similar al patrón MTV aunque cambian las siglas y por ende cómo se compone, la idea general es parecida. Las siglas en inglés significan Model, View, Controller. Model representa los datos y la lógica de negocio, se encarga de acceder a la base de datos y manejar la información,  View muestra los datos al usuario, es decir, es la interfaz visual, y Controller es el intermediario que se encarga de recibir la solicitud del cliente y ejecutar lo que esté diseñado a hacer.

# Comparación MTV con MVC

| Aspecto a comparar | MTV | MVC |
|---|---|---|
| Siglas | Modelo - Vista - Controlador | Modelo - Plantilla - Vista |
| Función | Separar la lógica de negocio, la interfaz y el control de flujo | Separar la gestión de datos, la presentación y la lógica |
| Vista | Muestra los datos al usuario (interfaz de usuario) | Maneja la interacción con el usuario y procesa las solicitude |
| Plantilla | - | Se encarga de generar el código HTML/CSS que ve el usuario |
| Controlador | Recibe y procesa la entrada del | La Vista cumple tanto la función |

| | | |
|---|---|---|
| | usuario, actualiza el Modelo | de procesar la entrada como la de preparar la respuesta, rol que en MVC está dividido entre Controlador y Vista |
| Ejemplos de frameworks | Ruby on Rails, Angular. | Django |

# ¿Qué es una app en Django?

Para Django una app es un modulo o componente que funciona independientemente de la aplicación web que contiene una funcionalidad específica o un un conjunto de características. Las aplicaciones de Django están diseñadas para ser reutilizables, lo que facilita el desarrollo de aplicaciones web complejas con un código modular y fácil de mantener. Una app de django tiene esta estructura: Models que define la estructura de los datos y el esquema de la base de datos, Views se encargan de procesar las solicitudes y generar las respuestas, Template que contienen las plantillas HTML para mostrar las páginas, URLs que definen los patrones de URLs y el enrutamiento dentro de la aplicación, Static files que guarda los archivos CSS, JavaScript y otros recursos estáticos, Management Commands que proporciona comandos personalizados para administrar la app, y Tests que contiene pruebas unitarias para asegurar que la funcionalidad de la app funciona correctamente.

# ¿Qué es el flujo request-response en Django?

Django utiliza objetos de solicitud (request) y respuesta (response) para transmitir el estado a través del sistema. Cuando el cliente/usuario realiza una solicitud (request) empieza el ciclo, esta solicitud llega al servidor y Django la convierte en un *HttpRequest* que tiene la información que se necesita conteniendo los métodos, los datos enviados y la cabecera. Después en urls.py la ruta y ejecuta la vista que tiene asociada pasándole un objeto *HttpResponse* que tiene el contenido que ve el usuario, finalmente esto le llega al navegador que se lo muestra al usuario.

# ¿Qué es el concepto de ORM?

El mapeo objeto-relacional es una forma de conectar el código del programa con la base de datos. ORM utiliza descriptores de metadatos para crear una capa entre el lenguaje de programación y una base de datos relacional. Así, simplifica la interacción entre las bases de datos relacionales.

ORM se basa en la abstracción de los objetos. Esto permite acceder y manipular objetos sin preocuparse por cómo se relacionan con sus fuentes de datos. ORM permite a los programadores poder seguir trabajando con los objetos a lo largo del tiempo, incluso si cambian las fuentes que los generan, los destinos que los reciben o las aplicaciones que los usan.

Además, los desarrolladores pueden realizar operaciones de creación, lectura,

actualización y eliminación de datos (*CRUD*) en bases de datos relacionales sin usar SQL directamente.

# ¿Qué son los templates en Django?

Las plantillas de Django son los archivos HTML que definen la estructura y el contenido de su aplicación web. Actúan como la capa visual de la aplicación, permitiendo crear páginas web con elementos dinámicos. También, permite mantener el HTML separado del código Python, lo que mejora la organización, permite mostrar datos dinámicos y admite estructuras de control (if).

# ¿Cómo instalar Django?

Hay tres formas de instalar django  yo voy a explicar como instalar la versión oficial que usan la mayoría de usuarios. Primero se necesita instalar python, luego la página oficial recomienda instalarlo mediante pip utilizando un entorno virtual mediante el comando “$ python -m pip install Django” y listo ya tenés instalado django.

# Glosario

API: Es un conjunto de reglas y herramientas que permiten que dos programas se comuniquen entre sí.

Backend: se refiere a la parte del sistema que no es visible para el usuario, pero que es esencial para que la aplicación funcione correctamente.

Clickjacking: Es un ataque cibernético en el que un atacante engaña a un usuario para que haga clic en elementos de una página web que parecen ser diferentes a lo que realmente son.

CRUD (Create, Read, Update, Delete): Conjunto de operaciones básicas para gestionar datos en una aplicación.

Framework: conjunto de herramientas, bibliotecas y convenciones que proporcionan una estructura para construir aplicaciones de forma más eficiente y organizada.

Hash: Es el resultado de aplicar una función hash a unos datos. Esta función convierte los datos de entrada de cualquier longitud en una cadena de caracteres, generalmente de longitud fija, llamada valor hash o código hash. El hash es único para un conjunto de datos específico, lo que significa que incluso un pequeño cambio en los datos de entrada resultará en un valor hash completamente diferente.

HttpRequest:Objeto que representa una solicitud HTTP enviada por un cliente (generalmente un navegador) al servidor.

HttpResponse: Objeto que representa la respuesta HTTP que el servidor envía al cliente.

Inyección SQL: Es un tipo de ataque cibernético que explota vulnerabilidades en bases de datos, permitiendo a los atacantes ejecutar código malicioso en la base de datos de una aplicación web.

Javascript: Lenguaje de programación interpretado, orientado a objetos, que se ejecuta en el navegador y permite crear páginas web dinámicas e interactivas.

Mapa de sitio: Es un archivo o esquema que enumera todas las páginas importantes de un sitio web, con el objetivo de ayudar a los motores de búsqueda (como Google) a rastrear y entender la estructura del sitio. También puede ser útil para los usuarios como guía de navegación.

Shared-nothing: Es un modelo de computación distribuida donde cada nodo en un sistema es independiente y no comparte recursos con otros nodos.

# Referencias

Chirilov, A.. (2022). Apps in Django: Concept & samples. https://www.codementor.io/@chirilovadrian360/apps-in-django-concept-free-samples-294vudyim5

CloudDevs. (2023). Understanding Django's request/response cycle. https://clouddevs.com/django/request-response-cycle/

Desarrollo Libre. (2022). El patrón de diseño MTV en Django y su relación con el MVC. https://www.desarrollolibre.net/blog/python/el-patron-de-diseno-mtv-en-django-y-su-relacion-con-el-mvc

Developer Stacks. (2023). Django request-response cycle.
https://medium.com/@developerstacks/django-request-response-cycle-7165167f54c5

Django Girls. (2024). Django templates.
https://tutorial.djangogirls.org/es/django_templates/

Django Software Foundation. (2024). Installing Django.
https://docs.djangoproject.com/en/5.2/topics/install/#installing-official-release

Django Software Foundation. (2024). Request and response objects.
https://docs.djangoproject.com/en/5.2/ref/request-response/

Django Software Foundation. (2024). Start overview.
https://www.djangoproject.com/start/overview/

Emmanuel Dav. (2022). Django templates: Mastering the basics.
https://emmanueldav.medium.com/django-templates-mastering-the-basics-29a99813af9f

Espi Freelancer. (2022). MTV en Django.
https://espifreelancer.com/mtv-django.html

GeeksforGeeks. (2023). Difference between MVC and MVT design pattern.
https://www.geeksforgeeks.org/software-engineering/difference-between-mvc-and-mvt-design-pattern

IBM. (2023). ¿Qué es Django?.
https://www.ibm.com/es-es/topics/django

Mozilla. (2023). Introducción a Django.
https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django/Introduction#portable

Nilebits. (2023). Django request life cycle explained.
https://dev.to/nilebits/django-request-life-cycle-explained-ci6

TechTarget. (2023). Object-relational mapping (ORM).
https://www.theserverside.com/definition/object-relational-mapping-ORM