**Trabajo práctico Laboratorio de Algoritmos y Estructuras de Datos**

Joaquin Hernan Rodriguez.

**Profesor**: Ignacio Miguel Garcia.

Instituto Superior Industrial Luis A. Huergo

**Materia:** Laboratorio de Algoritmos y Estructuras de Datos

**Fecha:** 7 de Julio del 2025

**Cuestionario:**

1. **¿Qué es Django y por qué lo usamos?**

Django es un software que se utiliza para el desarrollo de aplicaciones web de forma rápida y eficiente, dado que la gran mayoría de aplicaciones web poseen varias funciones comunes, coma la autenticación,administración de cookies, además de que muchos desarrolladores codifican funcionalidades similares en cada aplicación que desarrollan,Django facilita su trabajo agrupando funciones en módulos reutilizables, llamada marco de aplicaciones web, el cual es utilizado por los desarrolladores para organizar y escribir su código de manera más eficiente y más rápida.

Los desarrolladores utilizamos Django por:

* **Velocidad de desarrollo:** El marco Django está bien organizado y es fácil de instalar y aprender, por lo que puede comenzar en cuestión de horas. Los diseñadores de Django crearon el marco para implementar rápidamente cualquier arquitectura web en el código. Permite el desarrollo rápido y el diseño limpio. Puede escribir código en sólo unas pocas líneas porque Django proporciona una estructura lista para usar para varias tareas comunes de desarrollo web.
* **Rentabilidad:** Es un proyecto de Python gratuito y de código abierto con una comunidad activa que revisa y mantiene el software. Una organización llamada Django Software Foundation promueve y respalda el uso y el mantenimiento de Django completamente gratis, que también organizan encuentros regulares que animan a otros desarrolladores a revisar y contribuir con el proyecto, lo que genera un marco web de alta calidad y rico en funcionalidades gratuitas.
* **Popularidad:** Grandes proyectos y aplicaciones utilizan Django como son Instagram, FireFox, Pinterest, etc. Debido a su popularidad, el marco sigue evolucionando y cuenta con una sólida infraestructura de soporte.

1. **¿Qué es el patrón MTV (Model-Template-View) en Django? (simplificado en MVC). Compará MTV con MVC.**

Tanto el patrón MTV (Modelo-Plantilla-Vista) y el patrón MVC (Modelo-Vista-Controlador), son patrones de diseños que poseen un objetivo similar el cual es el facilitar el mantenimiento y las pruebas que se realizan gracias que los patrones de diseño ofrecen soluciones probadas a problemas recurrentes en el diseño de software. Ayudan a optimizar el proceso de desarrollo y a mejorar la legibilidad, la escalabilidad y el mantenimiento del código, pero se adaptan a diferentes necesidades y marcos de trabajo.

Por un lado el MTV se asocia ampliamente con Django, funcionando de la siguiente manera:

1. **Modelo**: Es la capa donde se define la estructura del código, además de que aquí es donde se conecta con la base de datos.
2. **Plantilla**: Es la capa donde se muestra la información en forma de un HTML o el lenguaje que se use.
3. **Vista**: Se encarga de analizar las solicitudes del usuario y de definir qué información mostrar en la pantalla.

Por otro lado el MVC es un patrón más tradicional y ampliamente utilizado, el cual a su veces se encuentra dividido en 3 partes las cuales funcionan muy similar al de el MTV, siendo que el Modelo del MTV similar al modelo del MVC, la Vista(MTV) a la Plantilla(MVC) y el Controlador(MVC) a la Vista(MTV).

Entonces si son tan similares que lo diferencia, bueno sus diferencias son:

| **Características** | **MTV** | **MVC** |
| --- | --- | --- |
| **Se usa en** | Django | Laravel,Ruby on Rails |
| **Rol de la vista** | Combina la lógica de vista y controlador del MVC | Vista y controlador son partes separadas e independientes |
| **Adaptabilidad** | No demasiada, mas que nada está orientada al desarrollo web | Es adaptable en distintos tipos de aplicaciones además de las páginas web |
| **Complejidad** | Es relativamente más sencillo | Es mas complejo de aprender |
| **Rendimiento** | Rápido(especialmente con Django) | Depende del framework |

1. **¿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. Django alienta a los desarrolladores a estructurar sus proyectos dividiéndolos en múltiples aplicaciones, cada una responsable de un aspecto particular de la aplicación.

Las aplicaciones Django están diseñadas para ser reutilizables, lo que facilita el desarrollo de aplicaciones web complejas con código modular y mantenible.

Algunas características principales de las aplicaciones son:

* **Modularidad:** Las aplicaciones de Django están diseñadas para ser modulares, es decir que tienen que tener un propósito o funcionalidad específica, tales como pueden ser la autenticación de usuarios o la gestión de blogs.
* **Independencia:** Las aplicaciones de Django son autónomas y no deberían tener fuertes dependencias de otras aplicaciones dentro del mismo proyecto. Esta separación permite conectar y usar aplicaciones en diferentes proyectos según sea necesario.
* **Estructura:** La estructura de la apps se basa en:
* Models: Define la estructura de datos y el esquema de la base de datos.
* Views: Gestiona 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 recursos estáticos.
* Magnet Commands: Proporciona comandos de administración personalizados para la aplicación.
* Test: Contiene pruebas unitarias para garantizar que la aplicación funcione correctamente.

Esas son algunas características de las aplicaciones.

1. **¿Qué es el flujo *request-response* en Django?**

Los flujos request-response en Django, se basa en que el usuario realiza una petición(el request) y la página le manda una respuesta(el response) siendo uno de los conceptos fundamentales de Django. Este proceso lo realiza de la siguiente manera:

1. **El usuario hace una petición HTTP:**

* Cuando alguien entra a un sitio web. el navegador (Chrome, Firefox, etc.) envía una solicitud HTTP al servidor.

1. **El servidor web (como Nginx o Apache) pasa la petición a Django:**

* Django no se comunica directamente con Internet. Necesita estar detrás de un servidor web, que hace de intermediario.

1. **La vista recibe el objeto HttpRequest:**

* Django crea una instancia del objeto HttpRequest, que encapsula toda la información de la petición del usuario. Este objeto contiene atributos como:
* request.method → "GET" o "POST"
* request.path → "/artículos/42"
* request.GET o request.POST → Parámetros enviados por el usuario
* request.user → Si el usuario está autenticado
* request.COOKIES → Cookies del navegador
* request.FILES → Archivos subidos

1. **Django busca una coincidencia en las URLs:**

* Django usa su sistema de enrutamiento (routing) definido en urls.py para determinar **qué vista debe ejecutar** según la URL solicitada.

1. **La vista procesa la lógica del negocio:**

* La función (o clase) vista se encarga de: Procesar parámetros, consultar la base de datos, validar formularios, ejecutar lógica de negocio, elegir qué template renderizar y preparar una respuesta personalizada

1. **La vista devuelve un objeto HttpResponse**

* Una vez que la vista procesa la información, **devuelve un objeto HttpResponse**, que puede incluir:
* **Contenido HTML, JSON, texto, etc.**
* **Código de estado HTTP** (200, 404, 500...)
* **Encabezados HTTP** (headers)
* **Cookies**.

1. **Django devuelve ese HttpResponse al servidor web:**

* Django toma esa respuesta final y la envía al servidor web que hizo la petición original (por ejemplo, Apache o Nginx).

1. El servidor web entrega la respuesta al navegador del usuario

* Finalmente, el servidor web entrega esa respuesta al navegador. El navegador interpreta el contenido (HTML, JSON, imagen, etc.) y lo muestra al usuario.

Este proceso dura milisegundos.

1. **¿Qué es el concepto de ORM (*Object-Relational Mapping*)?**

Un ORM (Mapeo Objeto-Relacional) es una herramienta que permite convertir los objetos de una aplicación a un formato que se puede almacenar en una base de datos, creando una base de datos virtual que vincula los datos de la aplicación con la base de datos final. Además de realizar esta conversión, los ORM eliminan la necesidad de usar el lenguaje complicado de SQL para las acciones CRUD (Crear, Leer, Actualizar, Eliminar), ya que el ORM se encarga de estas tareas.

Se menciona un ejemplo simple en C# para mostrar cómo el uso de un ORM puede hacer que el código sea más claro e integrado. Un aspecto clave del uso de un ORM es que no hay que preocuparse por actualizar una tabla o cambiar el motor de la base de datos, ya que el ORM se ocupa del mapeo necesario. Sin un ORM, habría que actualizar los métodos de creación y actualización al cambiar las tablas y revisar toda la sintaxis si hay un cambio de base de datos.

Se promete que en próximas entradas se presentarán los ORM que se utilizan en la empresa "dreams" para explicar por qué se elige uno u otro en diferentes situaciones.

1. **¿Qué son los *templates* en Django?**

En Django, los *templates* (o plantillas) son archivos que definen la estructura y presentación visual de las páginas web. Generalmente escritos en HTML, permiten insertar contenido dinámico mediante una sintaxis especial. A través del sistema de plantillas, Django separa la lógica del backend (escrita en Python) del diseño de la interfaz, facilitando así el desarrollo ordenado y mantenible.

Dentro de un template se pueden utilizar variables, que Django reemplaza con datos reales enviados desde las vistas, y también estructuras de control como bucles (for) o condiciones (if). Por ejemplo, se puede mostrar una lista de productos, el nombre de un usuario, o personalizar el contenido según la información proveniente de la base de datos.

Los templates suelen almacenarse en una carpeta llamada templates dentro de cada aplicación del proyecto, y se renderizan usando funciones como render() en las vistas. En resumen, los templates permiten generar páginas web dinámicas de forma flexible y eficiente.

1. **Lee con detenimiento este django at a glance:**

<https://docs.djangoproject.com/en/5.2/intro/overview/>

1. **¿Cómo se lo instala?**

<https://docs.djangoproject.com/en/5.2/intro/install/>

El proceso de instalación de Django:

1. Primero tenes que instalar python, que lo podes instalar desde la página oficial de python(<https://www.python.org/downloads/>).
2. Segundo abrir la consola.
3. Tercero verificar la versión de python que se instaló sea correcta o en caso de que ya lo tuvieras que este actualizado poniendo: python3 --version
4. Cuarto escribis en la consola lo siguiente: “pip install django”.
5. Quinto verificar la instalación de Django: django-admin --version

**Fuentes:**

Amazon Web Services. (s.f.). ¿Qué es Django? <https://aws.amazon.com/es/what-is/django/>

Chirilov, A. (2023). Apps in Django – Concept + Free Samples. Codementor.<https://python.plainenglish.io/understanding-mtv-vs-mvc-design-patterns-a-guide-for-developers-28999bfd3109>

Django Software Foundation. (2024). Installing Django.<https://www.codementor.io/@chirilovadrian360/apps-in-django-concept-free-samples-294vudyim5>

Dreams. (2023). ¿Qué es un ORM?<https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/que-es-un-orm>

Plain English. (2023). Understanding MTV vs MVC design patterns.<https://docs.djangoproject.com/en/5.2/intro/install/>