**Cuestionario:**

1. ¿Qué es Django y por qué lo usaríamos?

Django es un software que podemos utilizar para desarrollar aplicaciones web de forma rápida y eficiente.Con Django, podes llevar tus aplicaciones web desde una idea hasta su lanzamiento en cuestión de horas. Django se encarga de gran parte de las complicaciones del desarrollo web, para poder concentrarte en escribir tu aplicación sin tener que invertir demasiado tiempo. Es gratuito y de código abierto.

Django incluye muchos extras que podes usar para gestionar tareas básicas de desarrollo web.Además, se encarga de la autenticación de usuarios, la administración de contenido, los mapas del sitio, las fuentes RSS.

Django prioriza seguridad y ayuda a los desarrolladores a evitar errores comunes, como la inyección SQL, los scripts entre sitios, la falsificación de solicitudes entre sitios y el clickjacking. Su sistema de autenticación de usuarios proporciona una forma segura de gestionar las cuentas de usuario y las contraseñas.

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

La arquitectura *Model-Template-View*, también conocida como MVT, es un patrón de diseño de software dentro de Django que separa las capas de una aplicación web en distintos componentes.

El primero es el *Model*,que maneja todo lo relacionado con la información, esto incluye cómo acceder a esta, la validación, relación entre los datos y su comportamiento.

Posteriormente, la *View*,es un enlace entre el *Model* y el *Template*. Decide qué información será mostrada y por cual *template*.

Por último, el Template decide cómo será mostrada la información.

El MTV es una versión específica de MVC. Si bien tienen muchas similitudes,

MVC divide la responsabilidad entre el *Model*, la *View* yel *Controller*, mientras que MVT asigna roles diferentes al *Model*, la *View* y el *Template*. La diferencia más notable es que en MVT, la *View* procesa las solicitudes y se coordina con el *Model*, mientras que en MVC, el Controller desempeña esa función.

Si vamos a trabajar con Django, el MTV es más recomendable ya que ayuda a un desarrollo más rápido.

1. ¿Qué entendemos por *app* en django?

Una "app" en Django es un módulo o componente autónomo de una aplicación web que encapsula una funcionalidad específica o un conjunto de características relacionadas. Estas están diseñadas para ser altamente modulares,autónomas y reutilizables, además de ser un componente conectable y adaptable a diferentes necesidades.

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

Un servidor web es un conjunto de software y hardware que utiliza HTTP (Protocolo de Transferencia de Hipertexto) y otras reglas para responder a las solicitudes de los clientes generadas con la ayuda de la World Wide Web. Su función principal es mostrar el contenido del sitio web a los usuarios mediante el almacenamiento, el procesamiento y la distribución de páginas web.

El flujo de *request-response* 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 flujo se conforma mediante una serie de pasos:

Comienza cuando un cliente envía una solicitud HTTP a un servidor web con Django. Luego, se despacha la URL examinando la solicitud entrante de esta para determinar qué función de vista debe gestionarla.

Una vez que el despachador de URL identifica la función de vista adecuada, la invoca, pasando la solicitud HTTP como argumento, es decir,se procesa la solicitud y se preparan datos para la respuesta. Más tarde, se prepara el objeto de respuesta HTTP, donde se encuentra el contenido que se enviará al cliente.

Antes de enviar la 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.Finalmente, la respuesta HTTP se envía al cliente a través de la red y el navegador web de este procesa la respuesta.

Para que este flujo funcione correctamente, se necesita la ayuda del WSGI. Esta es una herramienta que se desarrolló para vincular un servidor web a un marco web. WSGI se divide en dos partes: el servidor y la aplicación. El servidor ejecuta la aplicación y ofrece una función de devolución de llamada a la aplicación para gestionar la respuesta WSGI. Mediante la devolución de llamada, la aplicación ejecuta la solicitud y devuelve el resultado al servidor.

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

El Object-Relational Mapping (ORM) es un patrón de diseño que ayuda a agilizar la comunicación entre bases de datos relacionales y lenguajes de programación orientados a objetos como Java o Python .

ORM produce un mapa estructurado que revela las relaciones entre objetos y tablas o datos. Los objetos se convierten esencialmente en partes digeribles que una base de datos puede almacenar para su posterior recuperación. ORM conecta lenguajes o aplicaciones de programación orientada a objetos con una base de datos relacional, comunicando cualquier cambio realizado en un objeto a la base de datos, que a su vez modifica los datos en consecuencia

Dado que ORM supervisa y gestiona los cambios de interfaz, los desarrolladores no tienen que preocuparse por editar código constantemente. Esta técnica también permite a los equipos comprender la estructura de una base de datos sin necesidad de revisar el código, lo que facilita la comprensión de estados y códigos complejos generados por programas orientados a objetos.

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

Los templates en Django son los archivos HTML que definen la estructura y el contenido de tu aplicación web. Actúan como la capa visual de tu aplicación, permitiéndote crear páginas web con elementos dinámicos. Estas plantillas se suelen usar con vistas y modelos de Django para crear aplicaciones web completas.

Las plantillas de Django permiten separar el diseño de los datos de la lógica de la aplicación. Esto significa que se pueden crear plantillas HTML que Django genera dinámicamente a partir de los datos proporcionados. Para lograr esto, se necesita comprender ciertos aspectos de Django que incluyen: variables, etiquetas, filtros, herencia de plantillas y resultados.

Las etiquetas y filtros de plantilla de Django ofrecen una potente manera de generar resultados HTML reutilizables. Las plantillas se definen en su propio lenguaje para separar la lógica de negocio de la lógica de visualización. Las etiquetas son como las palabras clave y las funciones del lenguaje, mientras que los filtros permiten modificar los datos existentes antes de mostrarlos.

1. Leé con detenimiento este *django at a glance*: <https://docs.djangoproject.com/en/5.2/intro/overview/>
2. ¿Cómo se lo instala? <https://docs.djangoproject.com/en/5.2/intro/install/>

Ahora que saben un poco más sobre django, los invito a hacer el siguiente tutorial, hasta la parte 8 inclusive:

* <https://docs.djangoproject.com/en/5.2/intro/tutorial01/>

**Glosario:**

**Fuentes RSS:** Formato web que permite a los usuarios recibir actualizaciones de sus sitios web favoritos sin necesidad de visitar cada uno de ellos

**Clickjacking:** Ataque cibernético a un usuario donde se lo engaña para que haga clic en un elemento de una página web que parece inofensiva, pero en realidad está superpuesta a un elemento oculto que realiza una acción maliciosa.

**HTTP:** Protocolo de capa de aplicación para transferir virtualmente archivos y otros datos a través de Internet

**Framework:** estructura o conjunto de herramientas y componentes que nos proporciona una base para desarrollar aplicaciones o crear páginas web de una manera más organizada, robusta y escalable

**Fuentes:**

**Pregunta 1:**

1. [**https://www.djangoproject.com/start/overview/**](https://www.djangoproject.com/start/overview/)
2. [**https://aws.amazon.com/es/what-is/django/**](https://aws.amazon.com/es/what-is/django/)

**Pregunta 2:**

1. [**https://espifreelancer.com/mtv-django.html**](https://espifreelancer.com/mtv-django.html)
2. [**https://www.geeksforgeeks.org/software-engineering/difference-between-mvc-and-mvt-design-patterns/**](https://www.geeksforgeeks.org/software-engineering/difference-between-mvc-and-mvt-design-patterns/)
3. [**https://python.plainenglish.io/understanding-mtv-vs-mvc-design-patterns-a-guide-for-developers-28999bfd3109**](https://python.plainenglish.io/understanding-mtv-vs-mvc-design-patterns-a-guide-for-developers-28999bfd3109)

**Pregunta 3:**

1. [**https://www.codementor.io/@chirilovadrian360/apps-in-django-concept-free-samples-294vudyim5**](https://www.codementor.io/@chirilovadrian360/apps-in-django-concept-free-samples-294vudyim5)

**Pregunta 4:**

1. [**https://clouddevs.com/django/request-response-cycle/#:~:text=Django's%20request%2Dresponse%20cycle%20is,and%20generates%20corresponding%20HTTP%20responses**](https://clouddevs.com/django/request-response-cycle/#:~:text=Django's%20request%2Dresponse%20cycle%20is,and%20generates%20corresponding%20HTTP%20responses)**.**
2. [**https://medium.com/@ksarthak4ever/django-request-response-cycle-2626e9e8606e**](https://medium.com/@ksarthak4ever/django-request-response-cycle-2626e9e8606e)
3. [**https://medium.com/@developerstacks/django-request-response-cycle-7165167f54c5**](https://medium.com/@developerstacks/django-request-response-cycle-7165167f54c5)

**Pregunta 5:**

1. [**https://builtin.com/data-science/object-relational-mapping#:~:text=El%20mapeo%20objeto%2Drelacional%20es,actualiza%20los%20datos%20para%20reflejarlos**](https://builtin.com/data-science/object-relational-mapping#:~:text=El%20mapeo%20objeto%2Drelacional%20es,actualiza%20los%20datos%20para%20reflejarlos)**.**

**Pregunta 6:**

1. [**https://emmanueldav.medium.com/django-templates-mastering-the-basics-29a99813af9f**](https://emmanueldav.medium.com/django-templates-mastering-the-basics-29a99813af9f)