Laboratorio #3

Django & ASP.NET

Parte II - Teoría

1. Objetivos

- Definir los conceptos asociados a entornos virtuales en Python.
- Conocer el funcionamiento del framework Django.
- Definir las bases correspondientes a ASP.NET.

2. Estructura

Este laboratorio está dividido en las siguientes secciones:

Parte II - Teoría

- **Definiciones**
  - √ Entorno Virtual Python.
  - √ Django.
  - √ ASP.NET.

3. Definiciones

Entorno Virtuales en Python

Para comprender el concepto de un entorno virtual analizaremos la situación tomando en cuenta el siguiente caso práctico:

Imagine que usted trabaja como desarrollador freelance, y actualmente se encuentra realizando 2 proyectos al mismo tiempo.

Ahora bien, si asumimos que cada proyecto es de una tecnología similar, y por motivos de lógica de negocio, uno de los proyectos es realizado en python3, y el otro en python, podrían surgir algunas problemáticas en torno a las tecnologías a utilizar; por ejemplo, se desea usar una versión más antigua de un módulo en python para uno de los proyectos, mientras que para el segundo proyecto se necesita de otra más actualizada.

De la misma manera, puede suceder que se necesiten servidores diferentes, y sí, hay soluciones para esto; sin embargo, qué pasaría si te dijera que existen entornos virtuales que tu puedes configurar a tu manera para instalar las tecnologías que necesites para un determinado proyecto, y que dicho proyecto ***"resida"*** en ese entorno o ambiente virtual, sin verse afectado por las versiones.

Prácticamente estamos hablando de un contenedor, en simples y vagas palabras un entorno virtual en python puede ser visto como una caja virtual, similar a un docker, que va a contener a tu proyecto, y todas las dependencias y tecnologías que involucren al mismo sin afectar al resto de los componentes del sistema.

Una de las herramientas más utilizadas al momento de crear entornos virtuales es ***Virtualenv***, y puede obtener más información desde *su página oficial*.

En la práctica veremos cómo instalar, configurar, y utilizar los entornos virtuales en Python.

Django

En Aplicaciones con la Tecnología Internet utilizamos ***Python/Flask***, básicamente es un microframework que permite realizar las operaciones básicas que provee un framework web. A pesar de su extensibilidad, se pudo notar que este framework por sí solo no provee una gran robustez y alcance, ya que es necesario incorporar una infinidad de módulos si queremos realizar un gran desarrollo.

La solución ante los problemas mencionados anteriormente es utilizar un framework que presente una gran robustez y un gran alcance, este framework es Django.

Existe una diversidad de documentación asociada a Django, de hecho, en *su página oficial* podemos encontrar guías y recursos que nos permitan dar nuestros primeros pasos.

De acuerdo a la fuente *Developer Mozilla*, Django ayuda a escribir software que es:

- **Completo**

Django sigue la filosofía "Baterías incluidas" y provee casi todo lo que los desarrolladores quisieran hacer "de fábrica". Porque todo lo que necesitas es parte de un único "producto", todo funciona a la perfeccion, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación.

- **Versátil**

Django puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web – desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios nuevos. Puede funcionar con cualquier framework cliente-servidor, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc).

- **Seguro**

Django ayuda a los desarrolladores a evitar varios errores comunes de seguridad. Por ejemplo, Django, proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando asi errores comunes como colocar la informacion de la sesión en cookies donde es vulnerable (en lugar de eso, las cookies solo contienen una clave y los

datos se almacenan en la base de datos) o se almacenan directamente las contraseñas en un hash de contraseñas.

Un ***hash de contraseña*** es un valor de longitud fija creado al enviar la contraseña a una cryptographic hash function. Django puede validar si la contraseña ingresada es correcta enviandola a traves de una funcion hash y comparando la salida con el valor hash almacenado. Sin embargo debido a la naturaleza "unidireccional" de la función, incluso si un valor hash almacenado se ve comprometido es dificil para un atacante resolver la contraseña original.

Django permite protección contra algunas vulnerabilidades de forma predeterminada, incluida la inyección SQL, scripts entre sitios, falsificación de solicitudes entre sitios y clickjacking (consulte Seguridad de sitios web para obtener más detalles sobre dichos ataques).

- **Escalable**

Django usa un componente basado en la arquitectura “shared-nothing” (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario). 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 cache, servidores de bases de datos o servidores de aplicación. Algunos de los sitios mas concurridos han escalado a Django para satisfacer sus demandas (por ejemplo, Instagram y Disqus, por nombrar solo dos).

- **Mantenible**

El código de Django está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable. En particular, utiliza el principio No te repitas "Don't Repeat Yourself" (DRY) para que no exista una duplicación innecesaria, reduciendo la cantidad de código. Django también promueve la agrupación de la funcionalidad relacionada en "aplicaciones" reutilizables y en un nivel más bajo, agrupa código relacionado en módulos (siguiendo el patrón Model View Controller (MVC)).

- **Portable**

Django está escrito en Python, el cual se ejecuta en muchas plataformas. Lo que significa que no está sujeto a ninguna plataforma en particular, y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X.

Hasta ahora hemos hablado acerca del alcance del framework, y las utilidades que ofrece; sin embargo, ¿qué podemos decir con respecto a su estructura?

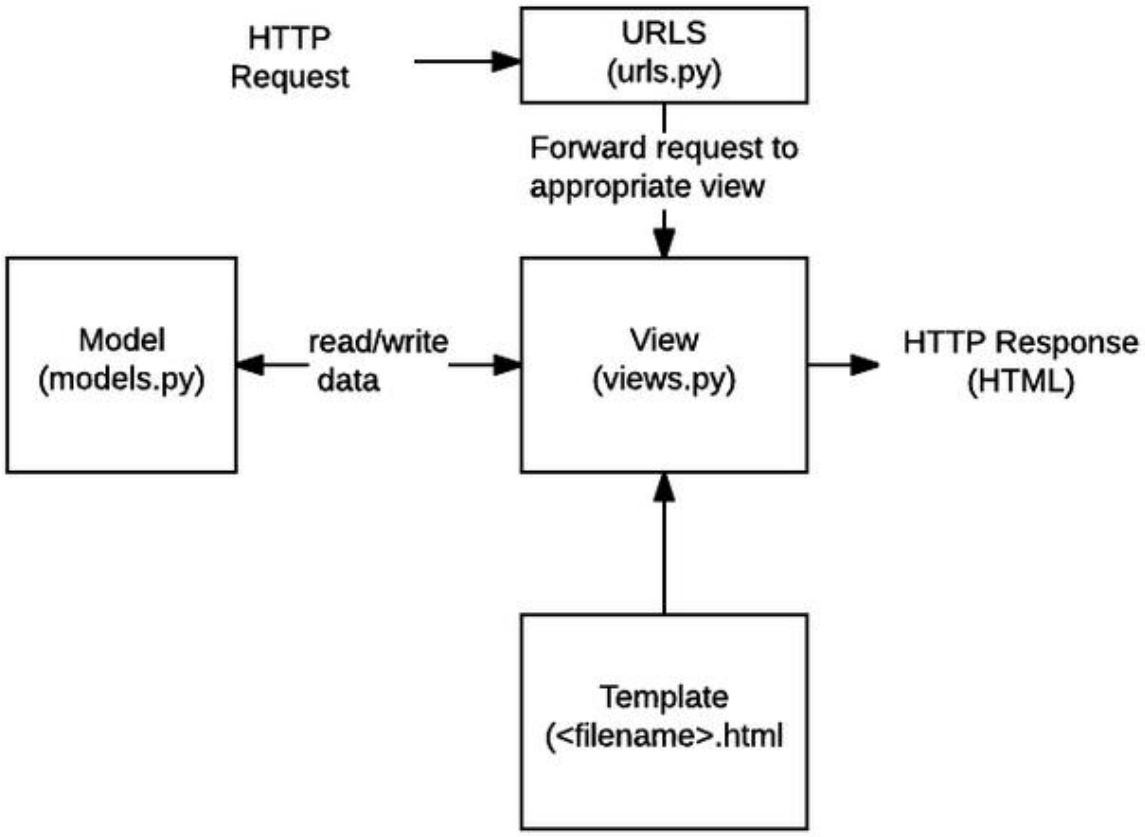

Figura 1. Componentes de un proyecto construido con el framework Django

En el gráfico anterior podemos observar los siguientes componentes:

- **URLS:** este componente es el encargado de realizar el map de las URL's asociadas a cada una de las peticiones HTTP para hacer su correspondiente direccionamiento a la vista respectiva.
- **Model:** son objetos que definen la estructura de la aplicación, y están directamente relacionados con la base de datos; es decir, a través de ellos se realizan las operaciones CRUD sobre la base de datos.
- **Template:** corresponde a los documentos (generalmente HTML) que son renderizados por la vista de la aplicación; ¿quiere decir que son las vistas?. No, las vistas representan prácticamente al controlador, la vista se encarga de gestionar los datos que provee la capa de modelo y enviarlas a la capa de template (o viceversa), para que la misma sea percibida visualmente por el usuario a través de una plantilla.
- **View:** es la capa que actúa como controlador, es por ello que el modelo de referencia utilizado por Django se conoce como **MVT (Model-View-Template)**, en lugar de lo que comúnmente se conoce como **MVC (Model-View-Controller)**.

Es importante resaltar que si deseas conocer realmente acerca de Django, la documentación es muy amplia, en especial por todas las funcionalidades que nos ofrece. Entre otro de los conceptos más importantes se encuentran las migraciones, mejor conocidas como migrations, que se describe como la forma de propagar los cambios a los modelos (por ejemplo, al momento de agregar un campo, borrar un modelo, actualizar algún campo, e.t.c) dentro del esquema de tu base de datos.

Lo mencionado anteriormente señala que Django tiene una capa de abstracción para la implementación de ORM, por lo tanto, también es relevante señalar el soporte que tiene este

framework con respecto a los SMBD que se utilizan durante el procesamiento del backend, el cual se lista a continuación:

- **PostgreSQL:** Django provee soporte para este SMBD; sin embargo, se debe tener en cuenta que al momento de crear una nueva columna se inicialice la misma con null=True si y solo sí dicha columna se encuentra vacía, para así no congestionar el espacio utilizado por la aplicación.
- **MySQL:** uno de los problemas que se presentan al usar MySQL es el fallo de migraciones; es decir, si falla una migración, los cambios se deben deshacer manualmente, ya que no es posible realizar un roll back.
- **SQLite:** no se recomienda correr una migración SQLite en un entorno de producción, se debe tener conciencia de los riesgos y limitaciones que podrían surgir si la migración realiza un cambio erróneo en el sistema.
- **Otros:** si como desarrollador tiendes a utilizar otros SMBD, puedes instalar módulos que te permiten una gestión más sencilla de los mismos.

En la sección práctica repasaremos todos los conceptos que fueron abarcados, y realizaremos una aplicación mediante el uso de Django.

ASP.NET

De acuerdo a la *página oficial de ASP.NET*, se define como un framework de código abierto cuyo propósito es construir aplicaciones web modernas y servicios con .NET. Adicionalmente, ASP.NET crea websites basados en HTML5, CSS, y JavaScript, que son simples, rápidos, y pueden ser escalables para millones de usuarios.

Incluso, con .NET es posible usar múltiples lenguajes, editores, y bibliotecas para construir aplicaciones no sólo en el ámbito web, también desktop, mobile, gaming, e IoT.

Pero, ¿entonces .NET es un lenguaje? No, ese es el punto, .NET es un framework, una plataforma de Microsoft de desarrollo y ejecución de aplicaciones. Realmente si quieres desarrollar aplicaciones en .NET, necesitas escribirlas en lenguajes compatibles con el Common Language Runtime (CLR), entre los lenguajes más utilizados se encuentran C#, F#, JScript.NET, y Visual Basic.

Por otra parte, ASP.NET es un modelo de desarrollo web unificado que incluye los servicios para crear aplicaciones web empresariales con el código mínimo. De manera general es posible decir que ASP.NET incluye:

- Marco de trabajo de páginas y controles
- Compilador de ASP.NET
- Infraestructura de seguridad
- Funciones de administración de estado
- Configuración de la aplicación
- Supervisión de estado y características de rendimiento
- Capacidad de depuración
- Marco de trabajo de servicios Web XML

- Entorno de host extensible y administración del ciclo de vida de las aplicaciones
- Entorno de diseñador extensible

Ahora, si hablamos con respecto al modelo arquitectónico de ASP.NET, se trata de un Modelo Vista Controlador (MVC), donde se tienen los siguientes componentes:

- **Modelo:** que son los objetos de la aplicación los cuales implementan la lógica de la capa de datos, muy similar a lo definido en el componente model de Django.
- **Vista:** que al contrario de Django, las vistas representan los componentes que muestran la interfaz de usuario de la aplicación. Sin embargo, la lógica es la misma, la vista vendría representando a cada uno de los templates en Django.
- **Controlador:** encargado de la interacción entre la capa de vista y la capa de datos, es aquella entidad que consulta datos del modelo, los procesa, y luego los escribe en la interfaz (la vista de la aplicación).

Además, el marco de ASP.NET MVC nos ofrece las siguientes ventajas:

- Facilita la administración de la complejidad, al dividir una aplicación en el modelo, la vista y el controlador
- No usa el estado de vista ni formularios basados en servidor. Esto hace que el marco de MVC sea ideal para los desarrolladores que deseen un control completo sobre el comportamiento de una aplicación
- Usa un modelo de controlador frontal que procesa las solicitudes de la aplicación web a través de un controlador único. Esto permite diseñar una aplicación que admite una infraestructura de enrutamiento avanzada
- Proporciona una mayor compatibilidad con el desarrollo basado en pruebas (TDD)

- Funciona bien para las aplicaciones web en las que trabajan equipos grandes de desarrolladores y para los diseñadores web que necesitan un alto grado de control sobre el comportamiento de la aplicación

Entre las características que presenta el marco de ASP.NET MVC de acuerdo a la página oficial de Microsoft se tienen:

- Separación de tareas de aplicación (lógica de entrada, lógica de negocios y lógica de la interfaz de usuario), facilidad para pruebas y desarrollo basado en pruebas (TDD). Todos los contratos principales del marco de MVC se basan en interfaz y se pueden probar mediante objetos ficticios, que son objetos simulados que imitan el comportamiento de objetos reales en la aplicación. Puede hacer una prueba unitaria de la aplicación sin tener que ejecutar los controladores en un proceso de ASP.NET, lo cual hace que las pruebas unitarias sean rápidas y flexibles. Puede usar cualquier marco de pruebas unitarias que sea compatible con .NET Framework
- Un marco extensible y conectable. Los componentes del marco de ASP.NET MVC están diseñados para que se puedan reemplazar o personalizar con facilidad. Puede conectar su propio motor de vista, directiva de enrutamiento de URL, serialización de parámetros de método y acción, y otros componentes. El marco de ASP.NET MVC también admite el uso de los modelos de contenedor Inyección de dependencia (DI) e Inversión de control (IOC). DI permite insertar objetos en una clase, en lugar de depender de que la clase cree el propio objeto. IOC especifica que si un objeto requiere otro objeto, el primer objeto debe obtener el segundo objeto de un origen externo como un archivo de configuración.
- Amplia compatibilidad para el enrutamiento de ASP.NET, un eficaz componente de asignación de direcciones URL que le permite compilar aplicaciones que tienen direcciones URL comprensibles y que admiten búsquedas. Las direcciones URL no tienen que incluir las extensiones de los nombres de archivo y están diseñadas para

admitir patrones de nombres de direcciones URL que funcionan bien para la optimización del motor de búsqueda (SEO) y el direccionamiento de transferencia de estado representacional (REST, Representational State Transfer)

- Compatibilidad para usar el marcado en archivos de marcado de páginas de ASP.NET existentes (archivos .aspx), de controles de usuario (archivos .ascx) y de páginas maestras (archivos .master) como plantillas de vista. Puede usar las características de ASP.NET existentes con el marco de ASP.NET MVC, tales como páginas maestras anidadas, expresiones en línea (<%= %>), controles de servidor declarativos, plantillas, enlace de datos, localización, e.t.c

- Compatibilidad para usar el marcado en archivos de marcado de páginas de ASP.NET existentes (archivos .aspx), de controles de usuario (archivos .ascx) y de páginas maestras (archivos .master) como plantillas de vista. Puede usar las características de ASP.NET existentes con el marco de ASP.NET MVC, tales como páginas maestras anidadas, expresiones en línea (<%= %>), controles de servidor declarativos, plantillas, enlace de datos, localización, e.t.c

De la misma manera, para comprender mejor todos estos conceptos, realizaremos el desarrollo de un proyecto básico, pero completo mediante el uso de ASP.NET en la sección práctica.