FORMACIÓN PRÁCTICA REMOTA S4

202120-PIAD-415-NRC_23944

PROYECTO FINAL DEL CURSO

PEA TAREA 4

AVANCE 01

Nro. Grupo : 04

Integrantes : Ferrel Julca, Rufo Piero

Mallqui Torres, Miguel Angel

Rojas Barrios, Alexander Paolo

Vigilio Lavado, Elmer

¿QUÉ ES PYTHON?

Python es un lenguaje de programación de alto nivel, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas.
Es muy atractivo en el campo del Desarrollo Rápido de Aplicaciones (RAD).

Los desarrolladores pueden leer y traducir el código Python mucho más fácilmente que otros lenguajes.

Comprobar la versión de Python:

```
C:\>python --version
Python 3.10.0
```

Ejecutar Python desde el símbolo del sistema.

```
C:\>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Usar Python:

```
>>> a = int(input("Ingresar primer número: "))
Ingresar primer número: 5
>>> b = int(input("Ingresar segundo número: "))
Ingresar segundo número: 9
```

```
>>> rpta = a + b
>>> print(rpta)
14
>>>
```

Cerrar Python:

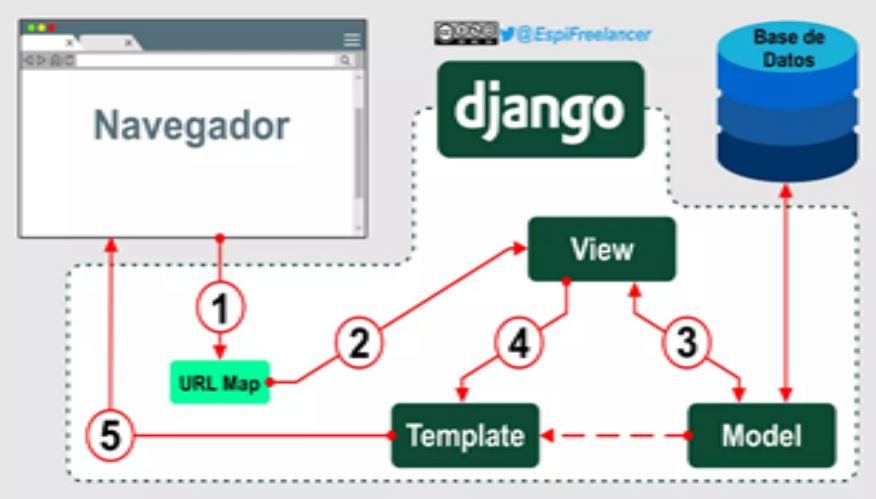
```
>>> exit()

C:\>
```

¿QUÉ ES PATRON DE DISEÑO MTV?

Django es conocido como un Framework MTV

- *M* significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- *T* significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web o otro tipo de documento.
- *V* significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.



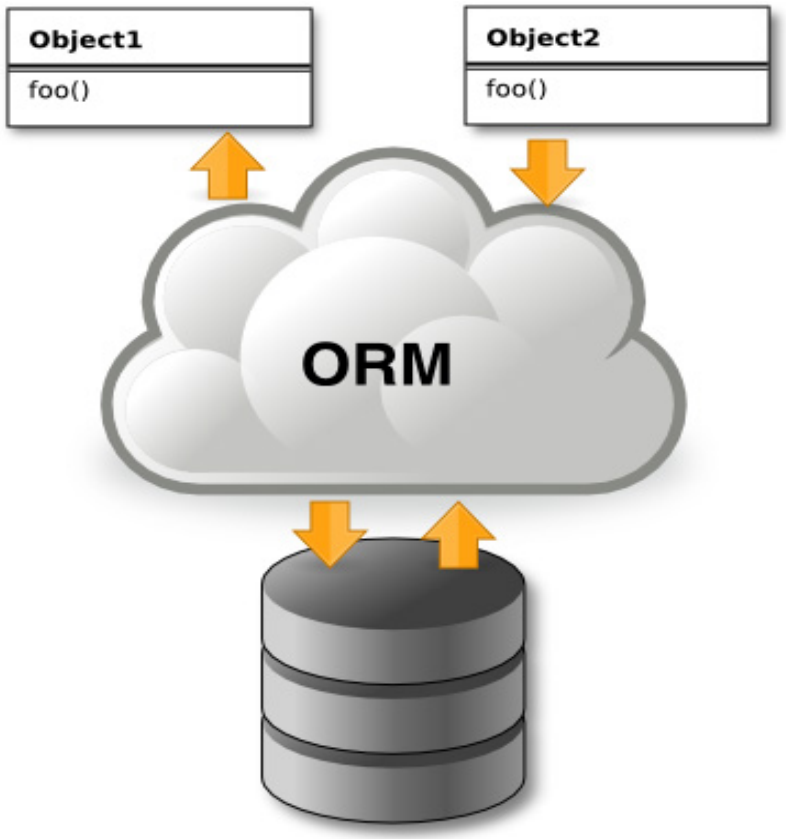
¿QUÉ ES ORM?

Es una técnica de programación que permite convertir datos entre el sistema de tipos empleado en el lenguaje de programación orientado a objetos y el utilizado en la base de datos relacional, empleando lo que se conoce como un motor o framework de persistencia.

- **Agilidad en el Desarrollo**. La curva de aprendizaje no es demasiado alta y cuando se domina el ORM es mucho más rápido y fácil de implementar el modelo de datos.
- **Abstracción de la base de datos.** Al emplear este sistema conseguimos un acceso transparente al sistema de base de datos, pudiendo cambiar en un futuro el motor sin notar efecto sobre nuestro sistema.

Seguridad. Los ORM implementan sistemas para evitar tipos de ataques maliciosos

- **Mantenimiento y reutilización del código.** Nos proporciona un mejor mantenimiento del código ya que nos permite separar los modelos en un paquete además de desacoplarlos de otras partes de la aplicación.
- **Encapsulación**: La capa ORM encapsula la lógica de los datos desacoplándolo del resto de la aplicación.



¿QUÉ SON LAS MIGRACIONES?

Las migraciones son la forma en que Django propaga los cambios que realiza en sus modelos (agregando un campo, eliminando un modelo, etc.) en el esquema de su base de datos. Están diseñados para ser en su mayoría automáticos, pero necesitará saber cuándo realizar migraciones, cuándo ejecutarlas y los problemas comunes con los que podría encontrarse.

migrate , que es responsable de aplicar y no aplicar las migraciones.

makemigrations , que es responsable de crear nuevas migraciones basadas en los cambios que ha realizado en sus modelos.

sqlmigrate , que muestra las instrucciones SQL para una migración.

showmigrations , que enumera las migraciones de un proyecto y su estado.

```
(py37_dj21) λ manage.py migrate --fake catalogos zero
Operations to perform:
  Unapply all migrations: catalogos
Running migrations:
  Rendering model states... DONE
  Unapplying catalogos.0004_producto... FAKED
  Unapplying catalogos.0003_auto_20181116_1140... FAKED
  Unapplying catalogos.0002_auto_20181115_1808... FAKED
  Unapplying catalogos.0001_initial... FAKED
```

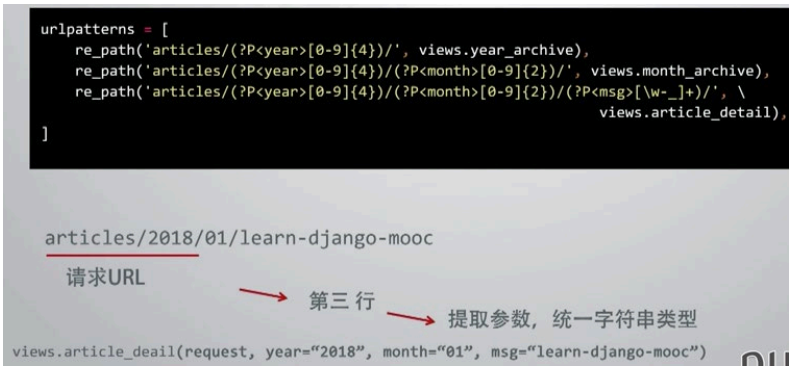
¿CÓMO SE GESTIONAN LAS RUTAS EN DJANGO?

En Django las rutas se gestionan con **path** y antiguamente con **re_path**.

La función **path** ya no acepta URL de expresiones regulares, debe usar la nueva sintaxis de URL <slug:title> en lugar de pasar una expresión regular para que coincida con los parámetros. El re_path solo funciona con URL formateadas con expresiones regulares (la forma antigua en la que creamos URLS).

Re_path, es una implementación de la forma 'antigua' de manejar las URL, que anteriormente (Antes de la versión 2.0 de Django) se hacía URL desde **django.conf.urls.** re_path podría ser una mejor opción cuando se requiere un convertidor muy personalizado y se alcanza el límite de lo que es factible con convertidores personalizados para 'path'. Fuera de eso, es recomendable usar path siempre que se pueda.

Path, se introdujo con el objetivo de simplificar las cosas, que es claramente la dirección en la que los desarrolladores de Django quieren ir. Por lo tanto, al usar path, está siguiendo esta dirección, minimizando así el riesgo de tener que adaptar su base de código a nuevos cambios.



¿QUÉ ES UN SISTEMA DE CONTROL DE VERSIONES?

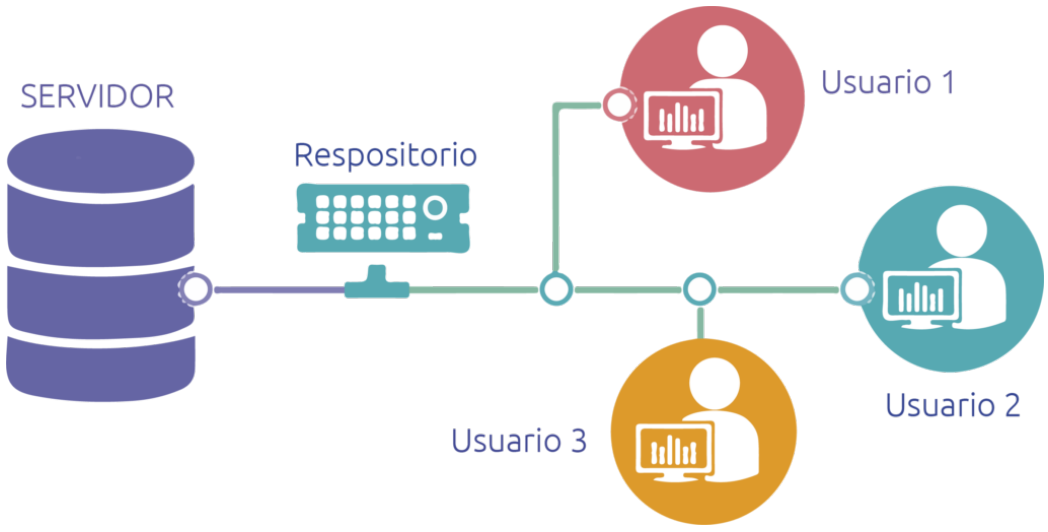
Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo.

Ayudan a los equipos de software a trabajar de forma más rápida e inteligente. Son especialmente útiles para los equipos de **DevOps**, ya que les ayudan a reducir el tiempo de desarrollo y a aumentar las implementaciones exitosas.

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Un buen software de control de versiones soporta el flujo de trabajo preferido de un desarrollador sin imponer una forma determinada de trabajar. Idealmente, también funciona en cualquier plataforma, en vez de ordenar qué sistema operativo o cadena de herramientas deben utilizar los desarrolladores.

DEOPS: es un conjunto de prácticas que trabaja para automatizar e integrar los procesos entre el desarrollo de software y los equipos de TI, para que puedan construir, probar y lanzar software de manera más rápida y confiable.



¿CUÁL ES LA DIFERENCIA ENTRE GIT Y GITHUB?

Git es un software de control de versiones para desarrolladores.

El control de versiones se refiere al proceso de guardar diferentes archivos o «versiones» a lo largo de las diferentes etapas de un proyecto. Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden que quieren revertir algunos de los cambios que han hecho.

Esto es útil por varias razones. Por ejemplo, facilita la resolución de errores y la corrección de otros errores que puedan ocurrir durante el desarrollo. También puede anotar los cambios en cada versión para ayudar a cualquier miembro del equipo a mantenerse al día sobre lo que se ha completado y lo que aún queda por hacer.

Sin embargo, hay algunos inconvenientes para manejar el desarrollo de esta manera. Como el software local está instalado en su máquina individual, Git no puede leer las ediciones que otros desarrolladores puedan estar haciendo en tiempo real. Esto significa que, si usted y un compañero de equipo están trabajando en un proyecto simultáneamente, no podrán ver el trabajo del otro.

GitHub facilita la colaboración con Git. Es una plataforma que puede mantener repositorios de código en almacenamiento basado en la nube para que varios desarrolladores puedan trabajar en un solo proyecto y ver las ediciones de cada uno en tiempo real.

LA DIFERENCIA ENTRE GIT Y GITHUB

Git es un software que permite a los desarrolladores guardar instantáneas de sus proyectos a lo largo del tiempo. Generalmente es mejor para uso individual.

GitHub es una plataforma basada en la web que incorpora las características de control de versiones de Git para que puedan ser utilizadas de forma colaborativa. También incluye características de gestión de proyectos y equipos, así como oportunidades para la creación de redes y la codificación social.

¿QUÉ UTILIDAD TENDRÍA UTILIZAR GITHUB PARA EL DESARROLLO DE SU PROYECTO?

GitHub es una buena herramienta para trabajar en equipo entre desarrolladores de software o sitio web utilizando el sistema de control de versiones Git, así facilitando la gestión de los proyectos.

Con esta plataforma muchas personas pueden trabajar de forma simultánea desde cualquier parte del planeta.

Esta nos permite desarrollar proyectos que están guardados en repositorios de forma gratuita y se puede entrar en los repositorios de otras personas pueden comentar sobre cómo mejorar contribuir al código, al igual que reportar errores a los desarrolladores.

También tiene herramientas útiles para el trabajo en equipo y las que destacan son:

Wiki: sirve para el mantenimiento de las distintas versiones de las páginas.

Visor de ramas: con la que se puede comparar los distintos progresos realizados por las ramas en un repositorio.

Sistema de seguimiento de problemas: permite a los miembros del equipo detallar algún problema con el software o dar sugerencias.

CUENTAS EN GITHUB

Rojas Barrios Alexander: https://github.com/alexander300903

Mallqui Torres Miguel: https://github.com/Miguelito-Angel-Torres

Ferrel Julca Rufo: https://github.com/Fer-Rel

Vigilio Lavado Elmer: https://github.com/elmervigiliolavado