Paramiko (exposición)

Paramiko es una biblioteca de Python creada en 2003 por unos tres sujetos de entre los cuales el más destacable es uno llamado Robey Pointer y es actualizado por múltiples desarrolladores hasta la fecha. Puede que el nombre suene extraño, pero proviene de un idioma artificial conocido como esperanto donde Paramiko significa «llave pequeña».

A grandes rasgos, Paramiko vendría a ser otra biblioteca de Python de entre los miles que hay, la cual tiene el enfoque de ser una implementación del protocolo SSH2 para comunicaciones cifradas en red (la diferencia entre SSH2 con el original es que, además de tener un 2 ahí pegado, incrementa cuestiones de seguridad a través de mejores cifrados y un mejor manejo de claves para la autenticación).

La verdad es que también íbamos a añadir una diapositiva para decirles cómo se instala, pero sólo tienen que abrir su entorno virtual de Python y escribir las palabras mágicas [*pip install paramiko*], esto instalará la biblioteca junto con todas sus dependencias; ahora pueden añadir a su CV que ya saben instalar paquetes de Python.

Paramiko ofrece funcionalidad de cliente y servidor para comunicarse entre sí remotamente e incluso para compartir archivos con literalmente cualquier cosa que tenga el protocolo SSH2 activado. La comunicación se da a través de la consola de comandos del dispositivo en cuestión, lo que da pie a la automatización de tareas gracias a la ejecución de múltiples comandos.

El usuario que use Paramiko tiene un control absoluto sobre la conexión SSH, donde podemos englobar cosas como:

- Conexión como tal (dirección y puerto SSH).
- Autenticación.
- Claves.
- Sesiones.
- Canales.
- Tiempos de espera.
- Algoritmos de cifrado.

Como mencioné anteriormente, un punto fuerte de Paramiko y SSH2 como tal es la seguridad, por lo que puede manejar accesos ya sea con contraseñas o con llaves (archivos de formato .pem). Finalmente, hay que recordar que Paramiko permite una conexión con todo dispositivo que tenga activo el protocolo SSH2, el cual puede partir desde ser un enrutador hasta un Sistema Operativo basado en Unix.

RUP (exposición)

RUP propone un proceso iterativo e incrementa de trabajo donde el proyecto es dividido en múltiples proyectos más pequeños para equilibrar el desarrollo de la arquitectura y los casos de uso. Cada iteración puede verse como una cascada del modelo secuencial lineal que todos conocen junto a varias actividades adicionales.

Cada iteración comprende una parte de la funcionalidad total y todas son evaluadas al final de tal modo que la retroalimentación delimite el rumbo de las siguientes.

RUP divide el proceso en unas cuatro etapas:

La de Inicio y elaboración se enfoca en la comprensión del problema, la delimitación del ámbito y la identificación y eliminación de riesgos. Se ven aspectos del negocio durante las iteraciones finales.

La de elaboración comprende el desarrollo de la arquitectura, flujos de trabajo de requerimientos y —en general— análisis y diseño de la implementación.

Durante la fase de construcción se refina el análisis de los casos de uso para su implementación (o sea, desarrollo) y pruebas.

Finalmente, en la fase de transición se busca garantizar una entrega nueva para la comunidad de usuarios.

En todas las fases participan todos los involucrados del proyecto.

La calidad de todos los artefactos se evalúa constantemente a lo largo de todo el desarrollo del proyecto, principalmente al final de cada una de las iteraciones.

Se realizan evaluaciones objetivas donde se busca detectar inconsistencias en el análisis, el diseño y la implementación, poniendo énfasis en aquellos que ya se detectaron como de mayor riesgo.

Se prueba y verifica todo desde el inicio, todos los casos de uso son evaluados desde los requerimientos hasta la codificación y se busca que todo este proceso de verificación se realice de forma automática.

Se sigue una filosofía donde si hay algo que no se encuentre verificado al 100%, entonces no está completo.

RUP tiene también especial cuidado en la gestión de los cambios a ser factores de riesgo muy altos. Los cambios se presentan a lo largo de todo el proceso de desarrollo por motivos variados, principalmente por cambios de requisitos y las dificultades del trabajo colaborativo.

Los cambios se ministran monitoreando modificaciones en las iteraciones, controlando los artefactos de software, gestionando todas las versiones, con espacios de trabajo seguros y aislados e implementando métricas de estado y avance.