Aula Mentor
Aprendizaje a lo largo de la vida

Cisco Networking Academy

Cisco Devnet

Módulo #3: Desarrollo y Diseño de software

Autor:Paco Aldarias
francisco.aldarias@aulamentor.es

Fecha:
18-10-2021

¿Qué aprenderá en este módulo?

Título del módulo: Desarrollo y Diseño de software

Objetivos del Módulo: Utilizar las mejores prácticas de desarrollo y diseño de software.

Título del Tema	Objetivo del tema
Desarrollo de software	Compare las metodologías de desarrollo de software.
Patrones de diseño de software	Describa los beneficios de los diferentes patrones de diseño de software.
Sistema de Control de Versiones.	Implemente el control de versiones de software mediante Git.
Conceptos básicos de codificación	Explique los procedimientos recomendados de codificación.
Revisión de código y pruebas	Utilice la prueba unitaria de Python para evaluar el código.
Comprensión de los formatos de datos	Utilice Python para analizar los diferentes formatos de mensajería y datos.

3.1.- Desarrollo del software

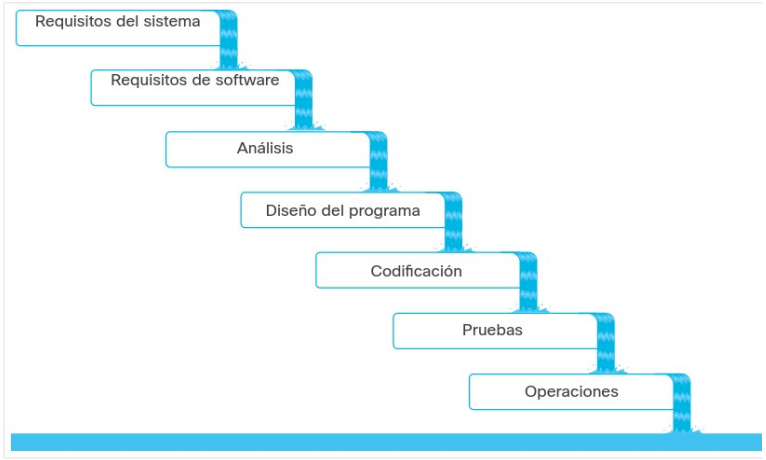
El proceso de desarrollo de software también se conoce como ciclo de vida de desarrollo de software (software development life cycle-SDLC).

Este proceso es más que simplemente codificar. También incluye recopilar requisitos, crear una prueba de concepto, probar y corregir errores.

3.1.- Desarrollo del software

Una metodología de desarrollo de software también se conoce como modelo de ciclo de vida de desarrollo de software.

- *Waterfall (Cascada)*
- *Agile*
- *Lean*

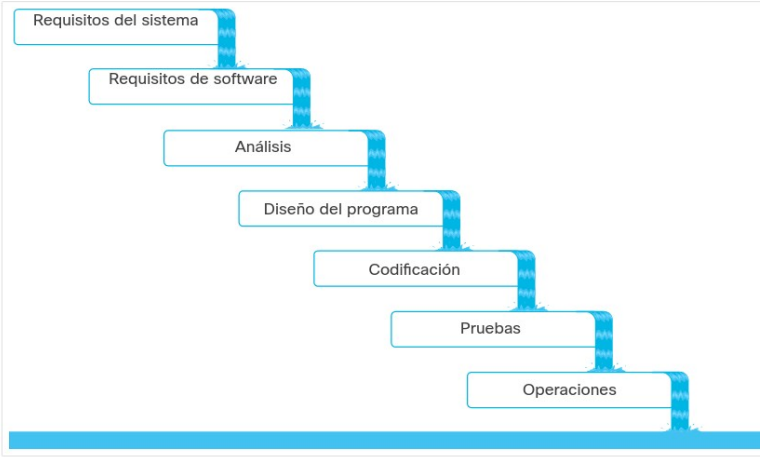


3.1.- Desarrollo del software

3.1.7. Desarrollo de software en Waterfall (Cascada)

El modelo de desarrollo en Cascada (Waterfall) es el modelo tradicional de desarrollo de software, y todavía se practica hasta el día de hoy. El modelo de desarrollo en Cascada (Waterfall) es casi idéntico al ciclo de vida de desarrollo del software ya que cada fase depende de los resultados de la fase anterior.

En este modelo en Cascada (Waterfall), el agua fluye en una sola dirección. Con el método en Cascada (Waterfall) el proceso va en una dirección, y nunca puede ir hacia atrás. Piense en ello como una carrera de relevos en la que un corredor tiene que terminar su distancia antes de entregar el testigo a la siguiente persona, que lo está esperando. La batuta siempre se mueve hacia adelante.



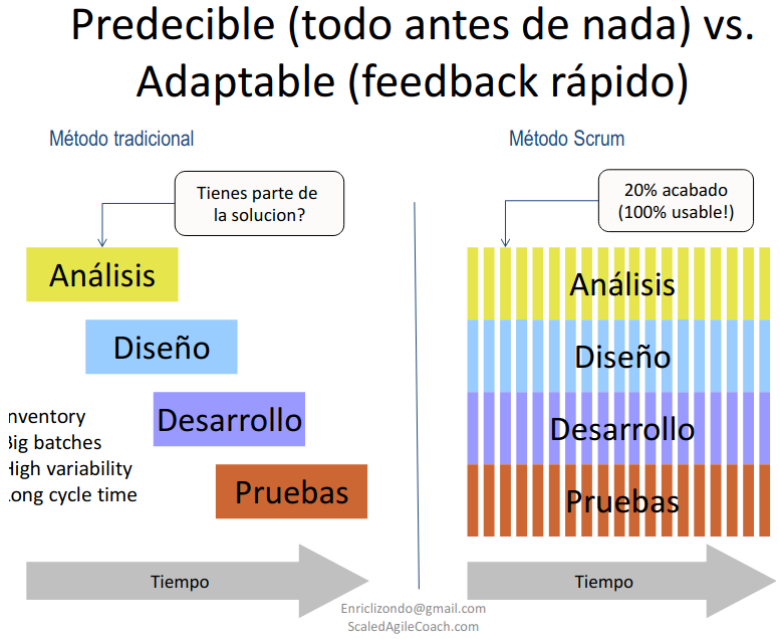
3.1.- Desarrollo del software

3.1.8 Desarrollo software Ágil (Agile)

Manifiesto Agile.

Según el Manifiesto Agile, sus valores de Agile son

- Individuos e interacciones sobre procesos y herramientas
- Software funcional sobre documentación completa
- Colaboración con clientes sobre negociación de contratos
- Respuesta al cambio sobre el seguimiento de un plan



DevNet Associate v1.0

Lab. 3.1.12 Explora herramientas desarrollo python

En este laboratorio, revisará la instalación de Python, PIP y entornos virtuales de Python. Usar Máquina Virtual

3.2.- Patrones de diseño

Modelo – Vista - Controlador

Los patrones de diseño de software son soluciones de mejores prácticas para resolver problemas comunes en el desarrollo de software. Los patrones de diseño son independientes del lenguaje.

El patrón de diseño Modelo-Vista-Controlador (MVC) a veces se considera un patrón de diseño arquitectónico. Su objetivo es simplificar el desarrollo de aplicaciones que dependen de interfaces gráficas de usuario (GUI). MVC abstrae el código y la responsabilidad en tres componentes diferentes: modelo, vista y controlador.

Componentes

Modelo - El modelo es la estructura de datos de la aplicación y es responsable de administrar los datos, la lógica y las reglas de la aplicación. Obtiene la entrada del controlador.

Vista - La vista es la representación visual de los datos. Puede haber varias representaciones de los mismos datos.

Controlador: El controlador es como el intermediario entre el modelo y la vista. Toma la entrada del usuario y la manipula para ajustarse al formato del modelo o la vista.

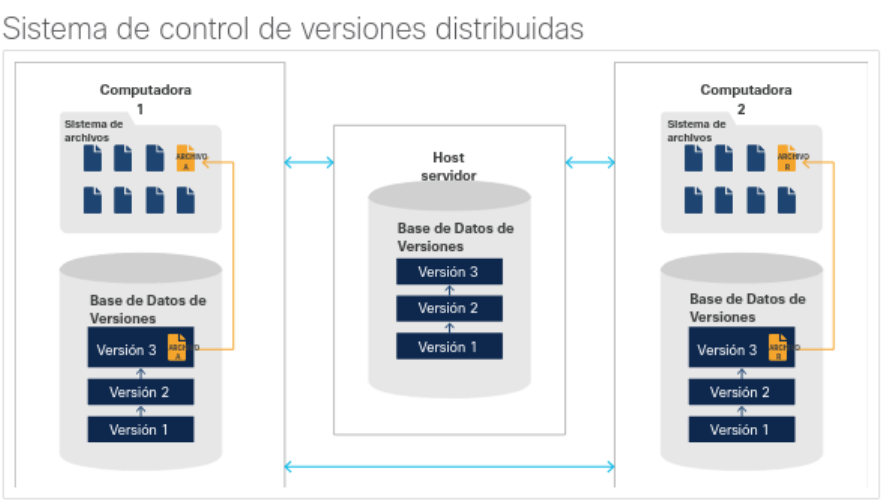
3.3.- Sistemas de control de versiones

3.3.1 Definición y tipos

El control de versiones, también llamado sistemas de control de versiones, control de revisiones o control de código, es una forma de administrar los cambios en un conjunto de archivos con el fin de mantener un historial de dichos cambios.

Existen tres tipos de sistemas de control de versiones:

- Local
- Centralizada
- Distribuida



3.3.- Sistemas de control de versiones

3.3.2 Git

Crear repositorio local:

git init

git add .

git commit -m “inicio”

Subir repositorio local al remoto:

git remote add origin “url del repositorio remoto”

git push origin master

Bajar repositorio remoto al local:

1ra vez) git clone “url”

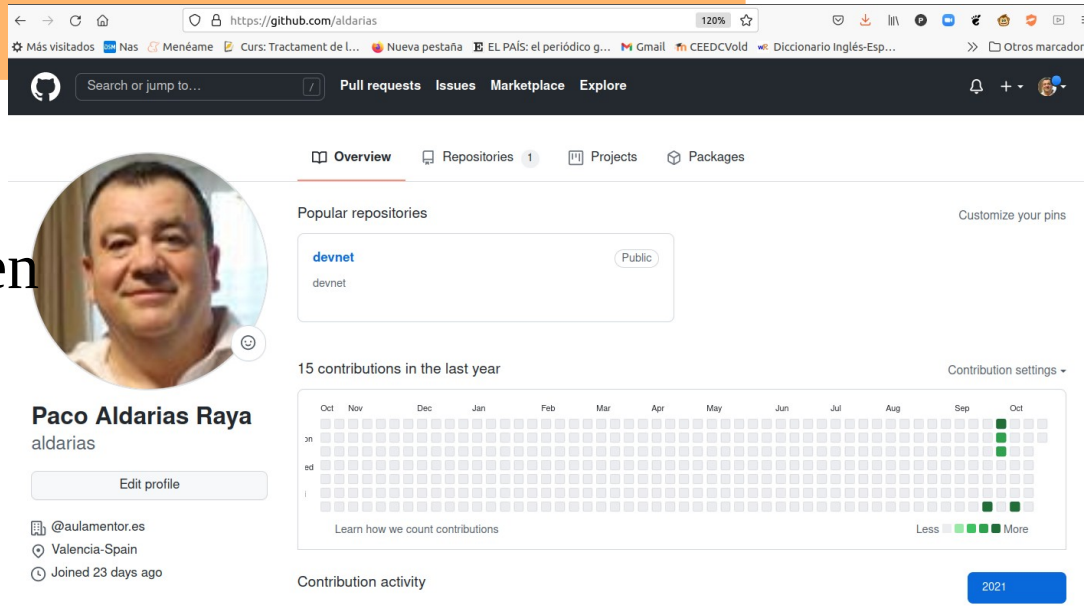
Resto) git pull origin master

3.3.- Sistemas de control de versiones

3.3.11 Laboratorio - Control de versiones de software con Git

https://github.com

GitHub Personal Access Token



DevNet Associate v1.0

3.4.- Codificación básica

Métodos, funciones, módulos y clases en python

Laboratorio 3.4.0. Explorar clases en Python

aula MENTOR

CISCO DevNet Associate v1.0

3.5.- Revisión de código y pruebas

Prueba unitaria
Prueba de la integración

Laboratorio 3.5.7. Crear prueba unitaria en python

3.6.- Formatos de datos

XML
JSON
YAML

Laboratorio 3.6.6. Analizar diferentes tipos de datos con python.