CUNEF UNIVERSIDAD | 5 años

G-174. Programación de Servidores Web

Tema 2: Configuración del entorno de desarrollo

Por Andres Gorostidi - Andres.Gorostidi@cunef.edu

01

Instalación y configuración de Servidores Web

Insralación de Servidores Web

Servidor Web es un software que corre sobre el servidor, escucha las peticiones HTTP que le llegan y las satisface.

Dependiendo del tipo de la petición, el servidor Web buscara una página Web o bien ejecutara un programa en el servidor.

Siempre devolverá algún tipo de resultados HTML al cliente o navegador que realizo la petición.

Todas las aplicaciones Web se ejecutarán en él.

Tipos de Servidores Web

Los servidores más destacados son:

- Apache.
- Nginx.

Tipos de Servidores Web: Apache

Apache es uno de los servidores web más antiguos y ampliamente utilizados. Desarrollado y mantenido por la Apache Software Foundation, es conocido por su robustez y flexibilidad.

Tipos de Servidores Web: Apache

Sus características principales son:

- Módulos: Apache usa un sistema modular que permite añadir funcionalidades mediante módulos (e.g., mod_rewrite, mod_ssl).
- Configuración: Se basa en archivos de configuración (httpd.conf, .htaccess) que permiten una personalización detallada.
- Compatibilidad: Funciona en múltiples sistemas operativos (Windows, macOS, Linux) y soporta una gran cantidad de lenguajes de programación (PHP, Perl, etc.).
- Accesibilidad: Facilita la administración mediante interfaces gráficas y herramientas como cPanel.

Tipos de Servidores Web: Apache

Ventajas y desventajas de Apache:

- Ventajas:

Flexibilidad y Extensibilidad: Puede ser configurado para manejar una variedad de escenarios gracias a sus módulos.

Documentación y Comunidad: Amplia documentación y una gran comunidad de usuarios.

- Desventajas:

Rendimiento: Puede ser menos eficiente en términos de manejo de múltiples conexiones simultáneas en comparación con Ngnix.

Tipos de Servidores Web: Nginx

Nginx es un servidor web de alto rendimiento, conocido por su capacidad para manejar grandes volúmenes de tráfico con eficiencia. Desarrollado inicialmente por Igor Sysoev, se ha convertido en una opción popular para servicios de alto tráfico.

Tipos de Servidores Web: Nginx

Características principales:

- Asíncrono y Event-Driven: Utiliza un modelo de manejo de eventos asíncronos, lo que permite un manejo eficiente de conexiones concurrentes.
- Configuración: Se basa en un solo archivo de configuración (nginx.conf) que es sencillo y directo.
- Proxy Reverso y Balanceo de Carga: A menudo se utiliza como un proxy reverso o para balanceo de carga debido a su alta eficiencia.

Tipos de Servidores Web: Nginx

Ventajas y desventajas:

- Ventajas:

Rendimiento: Excelente rendimiento en escenarios con alta carga y tráfico, usando menos recursos de CPU y memoria.

Simplicidad en la Configuración: Archivo de configuración más simple en comparación con Apache.

- Desventajas:

Menos Módulos: Menos módulos disponibles en comparación con Apache, aunque muchos casos de uso están cubiertos por sus funcionalidades nativas.

02

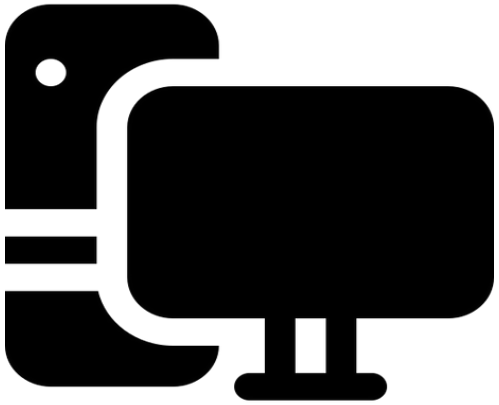
Herramientas de Diseño

Herramientas de diseño

En el desarrollo web, las herramientas de diseño son fundamentales para crear interfaces atractivas y funcionales.

Estas herramientas ayudan a diseñar y prototipar visualmente el contenido y la estructura de las páginas web antes de comenzar la codificación.

Unas de las herramientas de diseño más conocidas son Figma y Zeplin.



Herramientas de diseño: XAMPP

Existen opciones como XAMPP, un paquete de software que incluye Apache, MySQL, y PHP.

Esta herramienta facilita la creación de un entorno de servidor web local, necesario para desarrollar aplicaciones web.

Se puede descargar el instalador desde:
https://www.apachefriends.org/es/index.html

Herramientas de diseño: mongoDB

Es una base de datos NoSQL que almacena los datos en formato de documentos BSON (una extensión de JSON).

Herramientas de diseño: NODE JS

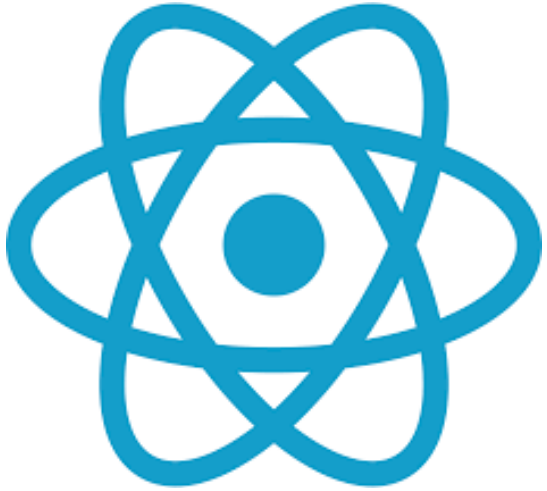
Es un entorno de ejecución de JavaScript basado en el motor V8 de Google Chrome que permite ejecutar Javascript en el servidor.

Herramientas de diseño: EXPRESS JS

Es un framework de Node.js que permite desarrollar el backend de una aplicación web.

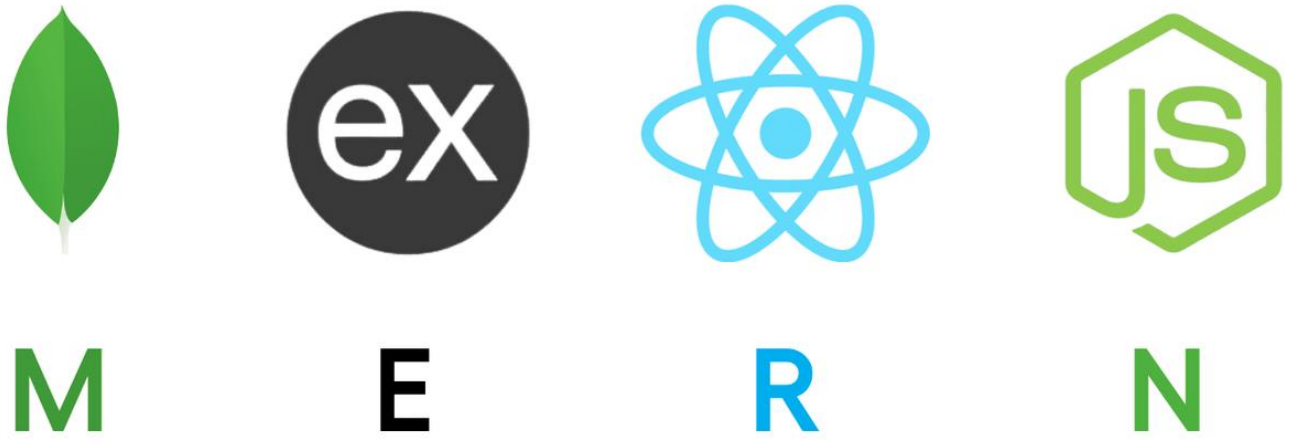
Herramientas de diseño: REACT

Es una biblioteca de JavaScript útil para construir interfaces de usuario dinámicas. Tiene una arquitectura basada en componentes y usa hooks para manejar el estado de la aplicación.



Herramientas de diseño: MERN

MERN (MongoDB, Express, React, Nodejs) es un stack de tecnologías web basado en JavaScript para crear aplicaciones web completas y escalables.



Herramientas de diseño: Figma

Figma es una herramienta de diseño, prototipado y colaboración en línea para equipos creativos. Está diseñada para que los diseñadores, desarrolladores, productores y equipos de marketing trabajen juntos para crear aplicaciones y productos de calidad.

Herramientas de diseño: Figma

Características principales:

- Diseño Colaborativo: Múltiples usuarios pueden trabajar en el mismo archivo simultáneamente.
- Prototipado Interactivo: Crear prototipos interactivos y animaciones.
- Componentes Reutilizables: Diseñar y reutilizar componentes en diferentes partes del proyecto.
- Accesibilidad: Funciona en cualquier navegador, lo que facilita el acceso desde diferentes dispositivos.

Herramientas de diseño: Zeplin

Zeplin es una herramienta colaborativa que facilita la comunicación y el "handoff" entre diseñadores y desarrolladores. Permite compartir diseños, realizar anotaciones y especificaciones de funciones, ayudando así a los equipos a conectar el diseño con el desarrollo y a cumplir con las expectativas del proceso creativo.

Herramientas de diseño: Zeplin

Características principales:

- Exportación de Activos: Generar automáticamente CSS, Android XML, y otros códigos a partir de los diseños.
- Colaboración y Comentarios: Permitir a los desarrolladores dejar comentarios directamente en los diseños.
- Integración: Compatible con herramientas de diseño como Figma, Sketch, y Adobe XD.
- Guías de Diseño: Crear y compartir guías de estilo y sistemas de diseño.

03

Herramientas de desarrollo

Herramientas de desarrollo

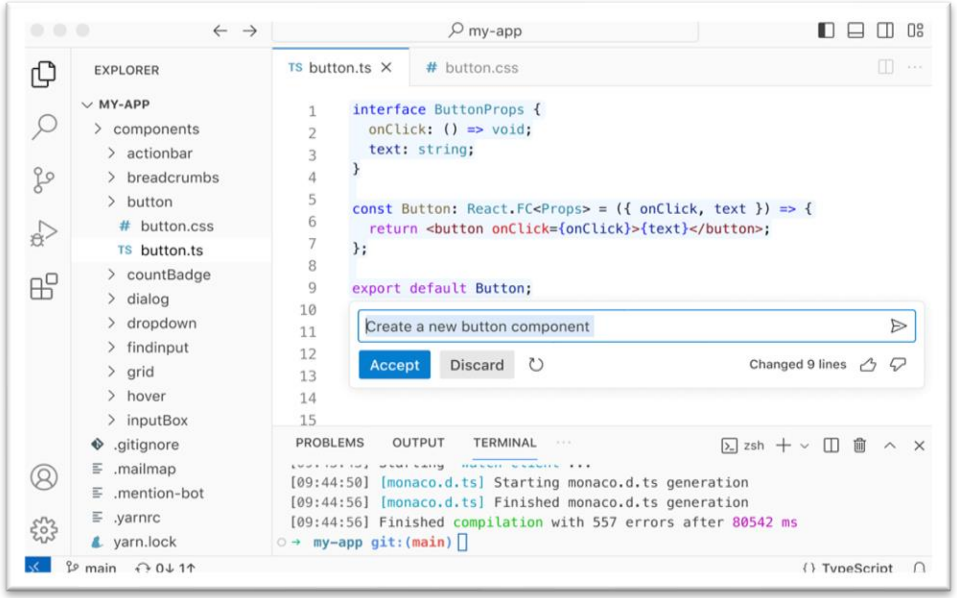
En el mundo del desarrollo web, contar con las herramientas adecuadas es fundamental para escribir código de manera eficiente, colaborar en equipo y asegurar la calidad del producto final. Las herramientas de desarrollo no solo facilitan la creación de código, sino que también permiten gestionar proyectos complejos, depurar errores de manera efectiva, y mantener un control estricto sobre las versiones del código.

Tres de las herramientas más esenciales que todo desarrollador web debería conocer y dominar:

- Visual Studio Code.
- IDE (Entorno de Desarrollo Integrado).
- Git.

Herramientas de Desarrollo: Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) de Microsoft utilizado para desarrollar aplicaciones en diversas plataformas y lenguajes de programación. Es conocido por sus potentes herramientas para la codificación, depuración y pruebas.



Herramientas de Desarrollo: Visual Studio

¿Para qué sirve?

- Edición de Código: Soporta una amplia variedad de lenguajes de programación como JavaScript, Python, Java, C++, entre otros.
- Depuración: Incluye un depurador integrado que permite a los desarrolladores ejecutar código paso a paso, identificar errores y probar aplicaciones.
- Extensiones: Permite la instalación de extensiones para añadir funcionalidades como la integración con Git, formateadores de código, temas de color, y más.

Herramientas de Desarrollo: Visual Studio

Características claves:

- Terminal integrado: Ejecuta comandos directamente en el editor sin necesidad de cambiar de ventana.
- Control de versiones: Integración nativa con Git para gestionar el control de versiones de los proyectos.
- Soporte para múltiples lenguajes: Detecta automáticamente el lenguaje de programación y proporciona autocompletado y resaltado de sintaxis.

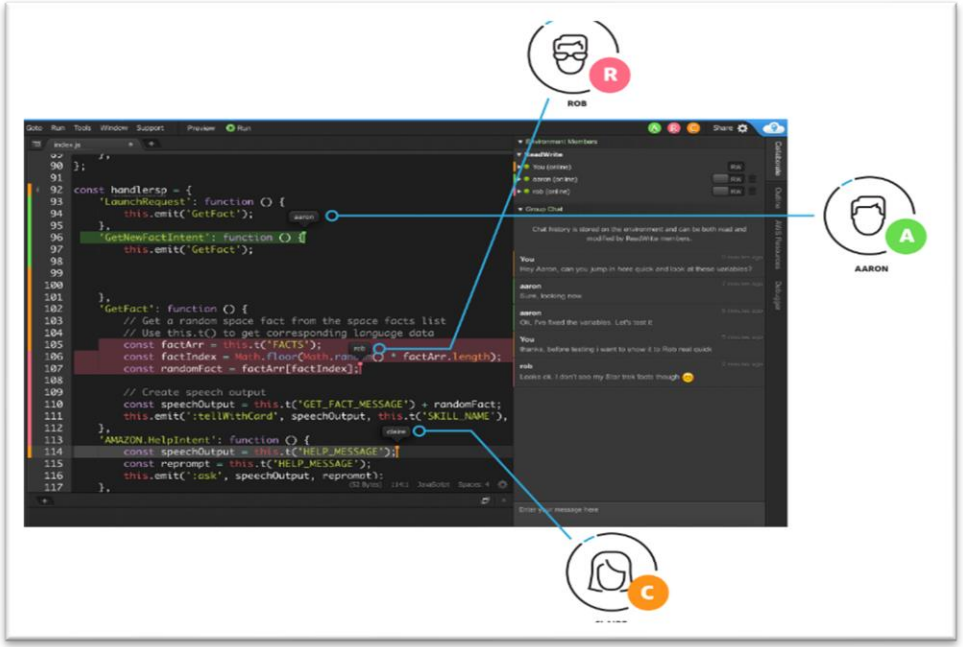
Herramientas de Desarrollo: Visual Studio

Extensiones:

- Prettier - Code Formatter: Una herramienta de formateo de código que asegura que todo tu código siga un estilo consistente. Soporta múltiples lenguajes y se integra directamente con VS Code.
- ESLint: Una extensión que se utiliza para identificar y corregir problemas de estilo y errores en JavaScript/TypeScript basándose en reglas predefinidas.
- GitLens: Expande las capacidades de Git en VS Code, proporcionando información detallada sobre las modificaciones en el código, como quién realizó cada cambio y cuándo.
- Live Server: Permite lanzar un servidor local con un solo clic y ver los cambios en tiempo real en el navegador a medida que editas tu código HTML, CSS y JavaScript.

Herramientas de Desarrollo: IDE

Un Entorno de Desarrollo Integrado (IDE) es una aplicación que proporciona herramientas integradas para desarrollar software. Combina un editor de código, un compilador o intérprete, herramientas de depuración y otros utilitarios en un solo paquete.



Herramientas de Desarrollo: IDE

¿Para qué sirve?

- Desarrollo Eficiente: Centraliza todas las herramientas necesarias para el desarrollo en un solo lugar, facilitando el flujo de trabajo y ahorrando tiempo.
- Depuración y Pruebas: Permite depurar y probar el código directamente desde el entorno sin tener que usar herramientas externas.
- Gestión de Proyectos: Los IDEs suelen incluir funcionalidades para gestionar proyectos grandes, facilitando la navegación y organización del código.

Herramientas de Desarrollo: IDE

Ejemplos de IDE populares

- IntelliJ IDEA: Popular entre los desarrolladores de Java por sus potentes herramientas de depuración y análisis de código.
- Eclipse: Un IDE extensible utilizado principalmente para desarrollar en Java, pero que también soporta otros lenguajes.
- PyCharm: Un IDE especializado en Python, que ofrece herramientas avanzadas de depuración y desarrollo web.

Herramientas de Desarrollo: Git

Git es un sistema de control de versiones distribuido creado por Linus Torvalds en 2005. Es una herramienta que permite a los desarrolladores gestionar y registrar los cambios realizados en su código fuente a lo largo del tiempo.

Herramientas de Desarrollo: Git

¿Para qué sirve?

- Control de Versiones: Git te permite llevar un historial detallado de todos los cambios en tu proyecto, facilitando la recuperación de versiones anteriores y la comparación entre ellas.
- Colaboración en Equipo: Con Git, varios desarrolladores pueden trabajar en el mismo proyecto simultáneamente, creando ramas independientes para cada función o corrección, y luego fusionando esos cambios en la rama principal.
- Seguridad y Copias de Respaldo: Al ser distribuido, cada desarrollador tiene una copia completa del historial del proyecto, lo que añade una capa extra de seguridad frente a la pérdida de datos.

Herramientas de Desarrollo: GitHub

GitHub es una plataforma basada en la web que utiliza Git como sistema de control de versiones. Ofrece un espacio centralizado donde los desarrolladores pueden almacenar sus repositorios de código, colaborar en proyectos, y compartir código con la comunidad global.

Herramientas de Desarrollo: GitHub

¿Para qué sirve?

- Almacenamiento de Repositorios: GitHub permite alojar repositorios de Git de manera pública o privada, facilitando la gestión de proyectos y la colaboración.
- Colaboración y Contribución: Proporciona herramientas para que los desarrolladores puedan contribuir a proyectos existentes a través de pull requests, revisiones de código, y gestión de issues.
- Documentación y Seguimiento: GitHub incluye funcionalidades para crear wikis, gestionar proyectos con tableros (project boards) y mantener un seguimiento de los problemas y solicitudes de características.

Herramientas de Desarrollo: GitHub

Uso correcto de GitHub

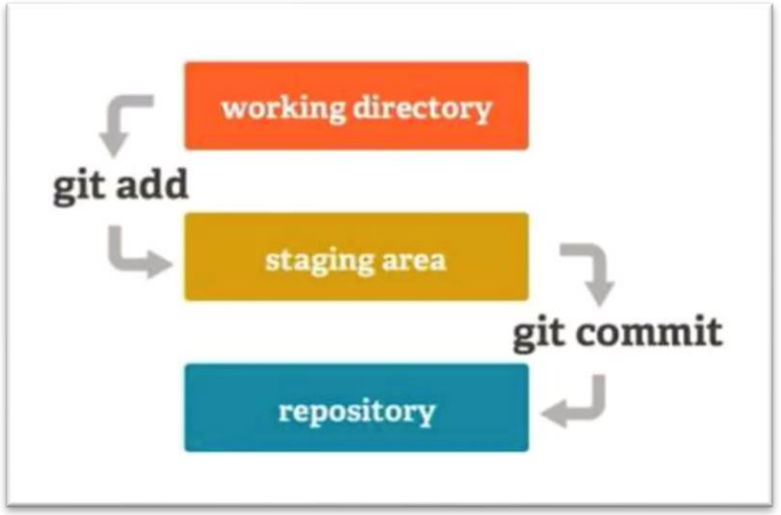
- Crear un Repositorio: Inicia un nuevo proyecto o clona uno existente en tu máquina local. Puedes crear un repositorio desde la interfaz de GitHub o mediante la línea de comandos con git init y git remote add origin <URL-del-repositorio>.
- Crear Ramas (Branches): Las ramas permiten trabajar en nuevas funcionalidades o correcciones sin afectar el código principal. Crea una nueva rama con git branch <nombre-de-la-rama> y cambia a ella con git checkout <nombre-de-la-rama>.
- Contribuir a un Proyecto: Al realizar cambios en tu rama, utiliza git add para agregar los cambios, git commit -m "Mensaje del commit" para confirmarlos, y git push para enviar esos cambios al repositorio remoto. Envía un pull request desde GitHub para que los mantenedores del proyecto revisen y, si es apropiado, integren tus cambios en la rama principal.

Herramientas de Desarrollo: GitHub

Comandos básicos

Los comandos de Git te permiten interactuar con tu repositorio local y remoto, gestionar tu código fuente, y colaborar con otros desarrolladores.

Aprender estos comandos básicos te dará el control necesario para manejar tus proyectos de manera efectiva en GitHub.



Herramientas de Desarrollo: GitHub

Comandos básicos

- git init: Inicializa un nuevo repositorio de Git en tu directorio actual.
- git clone: Clona un repositorio remoto en tu máquina local (git clone <URL-del-repositorio>).
- git status: Muestra el estado de los archivos en tu directorio de trabajo y en el área de preparación.
- git add: Añade cambios en los archivos al área de preparación (staging)- (git add <nombre-del-archivo> o git add).
- git commit: Guarda los cambios del área de preparación en el historial del proyecto (git commit -m "Mensaje descriptivo del commit").

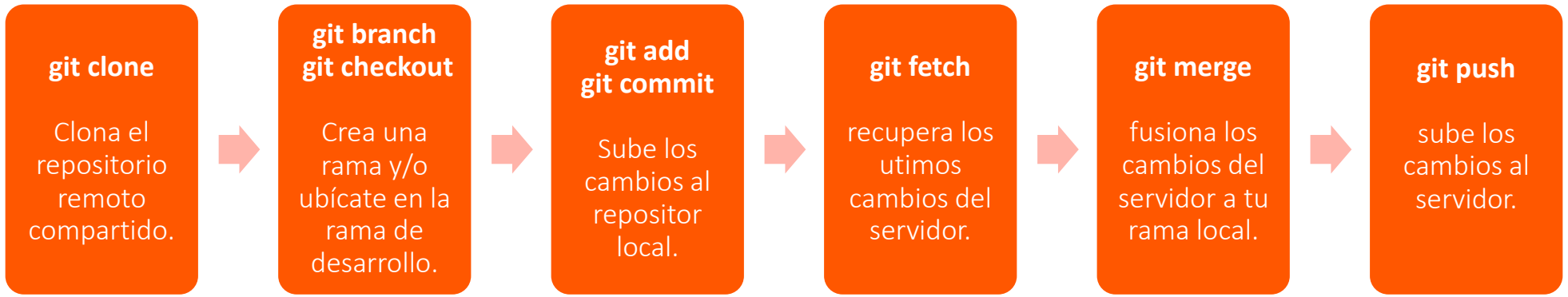
Herramientas de Desarrollo: GitHub

Comandos básicos

- git push: Sube los commits locales a un repositorio remoto (git push origin <nombre-de-la-rama>).
- git pull: Actualiza tu repositorio local con los cambios del repositorio remoto (git pull origin <nombre-de-la-rama>).
- git branch: Lista, crea o elimina ramas (git branch para listar, git branch <nombre-de-la-rama> para crear una nueva rama).
- git checkout: Cambia a una rama específica o revisa un archivo específico en el historial (git checkout <nombre-de-la-rama> o git checkout <commit-id> <archivo>).
- git merge: Fusiona los cambios de una rama en la rama actual (git merge <nombre-de-la-rama>).

Herramientas de Desarrollo: GitHub

¿Cómo colaborar en un proyecto Git?



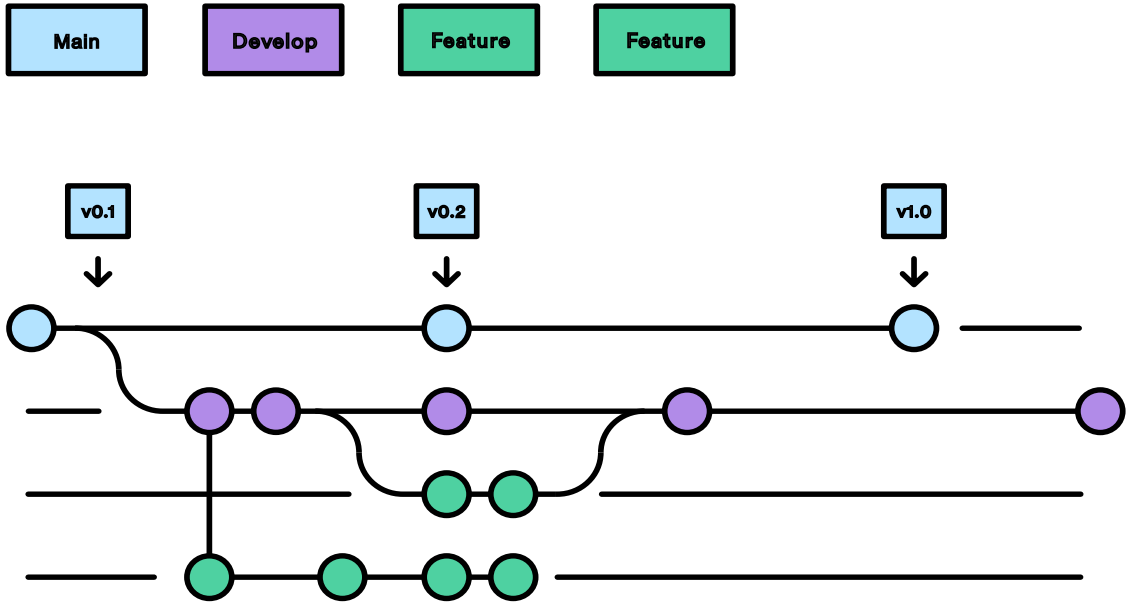
Herramientas de Desarrollo: GitHub

Flujo de trabajo

Herramientas de Desarrollo: GitHub

Flujo de trabajo (Git Flow)

El flujo de trabajo de Git Flow desarrollado por Vincent Driessen es un modelo estructurado de uso de Git que organiza el proceso de desarrollo en diferentes ramas para gestionar el desarrollo, las versiones de lanzamiento y las correcciones de errores de manera eficiente.



Herramientas de Desarrollo: GitHub

Estructura del flujo de trabajo (Git Flow)

Ramas Principales:

- master (o main): Esta es la rama principal que siempre contiene el código de producción estable. Cualquier versión final que se despliegue en producción debe estar en esta rama.
- develop: Es la rama de desarrollo principal donde se integran todas las nuevas características antes de un lanzamiento. Es la base para la creación de ramas de características (feature branches) y ramas de lanzamiento (release branches).

Herramientas de Desarrollo: GitHub

Estructura del flujo de trabajo (Git Flow)

Ramas de Soporte:

- feature/: Ramas para desarrollar nuevas funcionalidades. Se crean desde develop y se fusionan de nuevo en develop cuando la funcionalidad está completa.
- release/: Ramas creadas cuando el código en develop está listo para una nueva versión. Aquí se pulen detalles finales y se corrigen errores menores. Luego, se fusionan en master (para el lanzamiento) y en develop.
- hotfix/: Ramas para corregir errores críticos en producción. Se crean desde master, se soluciona el problema, y se fusionan tanto en master como en develop para que la corrección esté en futuras versiones.

Herramientas de Desarrollo: GitHub

Proceso de trabajo (Git Flow)

Ramas de Soporte:

- Desarrollo de Funcionalidades: Un desarrollador comienza creando una rama feature desde develop. Trabaja en la nueva funcionalidad en esta rama. Al finalizar, realiza un merge (fusión) de la rama feature en develop.
- Preparación de un Lanzamiento: Cuando se decide que el código en develop está listo para ser lanzado, se crea una release branch. Se realizan pruebas finales y correcciones menores en esta rama. Una vez listo, la release branch se fusiona en master y en develop. La rama master se etiqueta con la versión correspondiente.
- Corrección de Errores Críticos: Si surge un error crítico en producción, se crea una hotfix branch desde master. Se corrige el error y la hotfix branch se fusiona tanto en master como en develop.

CUNEF UNIVERSIDAD

5 años

GRACIAS POR VUESTRA ATENCIÓN