



Universidad Nacional Autónoma de  
México

FACULTAD DE CIENCIAS

MANUAL PARA INGENIERÍA DE SOFTWARE

Autor:

Cristopher Alejandro Escamilla Soto

Curso: 2023-2

# 1. ¿Que es Git?

Git es un programa de control de versiones distribuido; Quiere decir que un conjunto de colaboradores utilizan recursos computacionales en varios equipos de cálculo distintos para lograr un objetivo compartido común por lo tanto es una herramienta útil para rastrear fácilmente los cambios en su código, colaborar y compartir.

En un sistema de control de versiones distribuido, cada desarrollador tiene una copia completa del historial del proyecto y del proyecto. A diferencia de los sistemas de control de versiones centralizados que alguna vez fueron populares, los DVCS no necesitan una conexión constante a un repositorio central. Git es el sistema de control de versiones distribuido más popular. Git se usa comúnmente para el desarrollo de software comercial y de código abierto, con beneficios significativos para individuos, equipos y empresas.

- Git permite a los desarrolladores ver la línea de tiempo completa de sus cambios, decisiones y progresión de cualquier proyecto en un solo lugar. Desde el momento en que acceden al historial de un proyecto, el desarrollador tiene todo el contexto que necesita para comprenderlo y comenzar a contribuir.
- Los desarrolladores trabajan en todas las zonas horarias. Con un DVCS como Git, la colaboración puede ocurrir en cualquier momento mientras se mantiene la integridad del código fuente. Usando sucursales, los desarrolladores pueden proponer de forma segura cambios en el código de producción.

Con Git puede rastrear los cambios que realiza en su proyecto para que siempre tenga un registro de lo que ha trabajado y pueda volver fácilmente a una versión anterior si es necesario.

Un control de versiones se refiere al proceso de guardar diferentes archivos o versiones a lo largo de las distintas etapas de un proyecto:

- Cuando se hizo un cambio.
- Quien hizo el cambio.
- Que cambios se hicieron.
- Identificador del cambio.

## 2. ¿Que es GitHub?

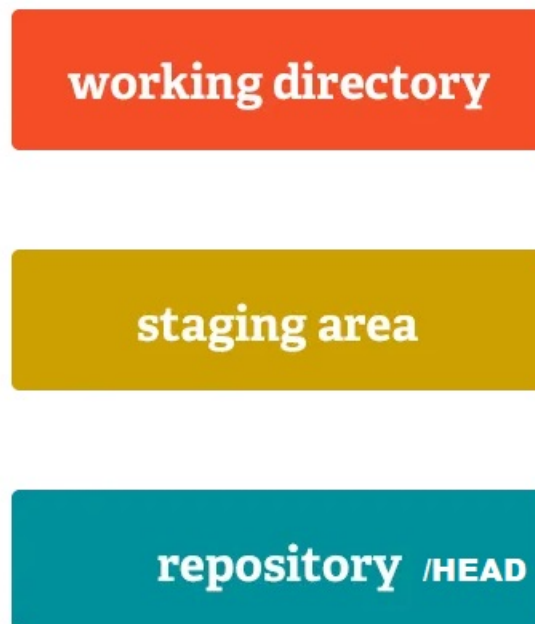
GitHub aloja repositorios Git y proporciona a los desarrolladores herramientas para enviar un mejor código a través de funciones de línea de comandos, problemas ( threaded discussions ), solicitudes de extracción (pull request), revisión de código. Con capas de colaboración como el flujo GitHub, una comunidad de 15 millones de desarrolladores y un ecosistema con cientos de integraciones, GitHub cambia la forma en que se construye el software.

GitHub construye la colaboración directamente en el proceso de desarrollo. El trabajo está organizado en repositorios donde los desarrolladores pueden describir los requisitos o la dirección y establecer expectativas para los miembros del equipo. Luego, utilizando el flujo de GitHub, los desarrolladores simplemente crean una rama para trabajar en las actualizaciones, comprometen cambios para guardarlos, abren una solicitud de extracción para proponer y discutir cambios, y fusionar solicitudes de extracción una vez que todos estén en la misma página

GitHub es una forma de usar el mismo poder de Git todo en línea con una interfaz fácil de usar. Se utiliza en todo el mundo del software y más allá para colaborar y mantener la historia de los proyectos.

### 3. Las 3 zonas de Git

- El directorio de trabajo: Será el directorio donde hayamos creado el repositorio, y en el tendremos los archivos que se quieran añadir al proyecto en algún momento determinado. En esta zona los ficheros están en el estado Untracked es decir, aun no son tomados en cuenta por git para controlar sus cambios
- Staging area: O área de preparación, es un área temporal de GIT donde se guardarán los archivos que están a punto de ser enviados al repositorio. En esta área git nos puede indicar en que estado se encuentra un archivo
- El repositorio: También llamado HEAD o zona de commits, será la zona donde los archivos del proyecto estarán almacenados, ordenados y controlados para ser recuperados en cualquier momento.



Un repositorio, o proyecto Git, abarca toda la colección de archivos y carpetas asociados con un proyecto, junto con el historial de revisiones de cada archivo. El historial del archivo aparece como instantáneas en el tiempo llamadas confirmaciones. Las confirmaciones se pueden organizar en múltiples líneas de desarrollo llamadas ramas. Debido a que Git es un DVCS, los repositorios son unidades autónomas y cualquiera que tenga una copia del repositorio puede acceder a toda la base de código y su historial. Usando la línea de comandos u otras interfaces de facilidad de uso, un repositorio Git también permite: interacción con el historial, clonación del repositorio, creación de ramas, confirmación, fusión, comparar cambios entre versiones de código y más.

## 4. Configuraciones iniciales.

Git utiliza un nombre de usuario para asociar las confirmaciones con una identidad. El nombre de usuario de Git no es tu mismo nombre de usuario de GitHub.

El cambio del nombre asociado a las confirmaciones de Git con `git config` solo afectará a las confirmaciones futuras y no cambiará el nombre que se usa en las anteriores.

El caso práctico más básico de `git config` es invocarlo con un nombre de configuración, que mostrará el valor definido con ese nombre. Los nombres de configuración son cadenas delimitadas por puntos que se componen de una "sección" una clave.<sup>en</sup> función de su jerarquía.

Establecer el nombre de usuario de Git para todos los repositorios del equipo:

```
\$ git config --global user.name "Mi nuevo nombre"
```

Establecer el nombre de usuario de Git para un repositorio en específico:

```
\$ git config user.name "Nuevo nombre"
```

Establecer para el correo del usuario de Git para todos los repositorios del equipo:

```
\$git config --global user.email "user.name@email.com"
```

Establecer para el correo del usuario de Git para un repositorio en específico:

```
\$git config user.email "user.name@email.com"
```

Establecer para que el nombre de la rama principal sea `main` en lugar de `master`:

```
\$git config --global init.defaultBranch main
```

Para ver mas opciones de configuración:

```
\$git config --help
```

Cuando se utilizan comandos como `git diff`, si el resultado es superior a una página se efectuará una paginación automática empleando `less` (comando UNIX que muestra el texto en una página). Si se busca navegar sin limitaciones en el documento. Para esto, se requiere definir una paginación de tipo `cat`:

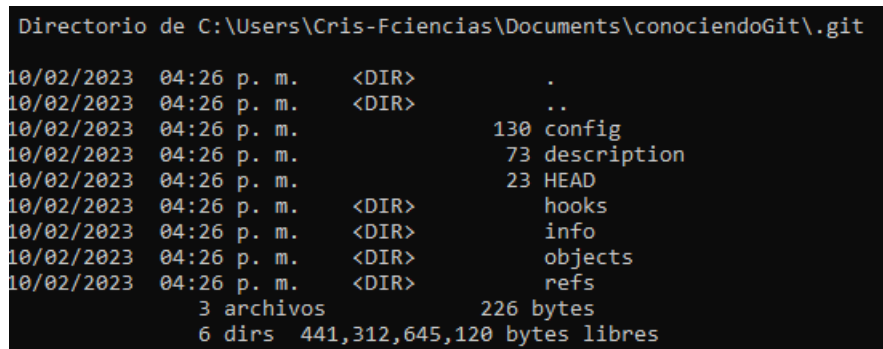
```
\$git config --global pager cat
```

## 5. Inicializar un proyecto nuevo.

Cuando deseamos versionar un nuevo proyecto con git, primero necesitamos crear un nuevo directorio o posicionarnos en el directorio que comenzaremos a versionar:

```
\$ git init
```

Este comando creará una carpeta llamada .git en la raíz del proyecto. dicha carpeta esta oculta para la mayoría de exploradores de archivos.



```
Directorio de C:\Users\Cris-Fciencias\Documents\conociendoGit\.git
10/02/2023  04:26 p. m.    <DIR>      .
10/02/2023  04:26 p. m.    <DIR>      ..
10/02/2023  04:26 p. m.           130 config
10/02/2023  04:26 p. m.           73 description
10/02/2023  04:26 p. m.           23 HEAD
10/02/2023  04:26 p. m.    <DIR>      hooks
10/02/2023  04:26 p. m.    <DIR>      info
10/02/2023  04:26 p. m.    <DIR>      objects
10/02/2023  04:26 p. m.    <DIR>      refs
                        3 archivos          226 bytes
                        6 dirs  441,312,645,120 bytes libres
```

El archivo .git alberga todo el contenido del repositorio utilizado por Git. Algunos de los mas importantes son:

HEAD: contiene la referencia del commit a partir del cual se trabaja. Es una referencia que depende de la rama sobre la cual nos encontramos. HEAD designará el commit más reciente de una rama y por defecto el commit más reciente de la rama actual.

branches: esta carpeta contiene las ramas del proyecto. Las ramas son versiones que se diferencian

Cuando se trabaja sobre un repositorio remoto y lo que necesitas es clonar el repositorio y ponerte manos a la obra, tendrás que posicionarte en el directorio donde se ubicara remotamente dicha copia y después:

```
\$ git clone
```

## 6. Markdown (Lenguaje de marcado)

Este lenguaje no es un lenguaje de presentación con una estructura XML como lo es HTML. Este lenguaje fue diseñado con el fin de ser legible directamente en modo texto, lo que lo ha hecho muy popular entre los desarrolladores. En efecto, un formato como el Markdown puede consultarse en modo texto y, por lo tanto, puede ser fácilmente versionado. Además, se puede exportar los archivos Markdown fácilmente a código HTML u otros formatos.

La sintaxis es muy sencilla y cuenta con varias opciones diferentes para algunos de sus elementos. Básicamente, se trata de añadir ciertos caracteres al inicio de la línea o antes y después de los elementos a los que vamos a aplicar el formato.

- Encabezados:  
# Esto es un H1 #  
## Esto es un H2 ##  
### Esto es un H3 ###
- Listas ordenadas:
  1. Primer objeto ordenado
  2. Segundo objeto ordenado
- Listas.
  - \* Listas no ordenadas
  - también listas no ordenadas
  - + lo mismo
- Links:  
[Soy el nombre del link (azulito) y voy a google](https://www.google.com)

## 7. Comandos básicos de GitHub

Para utilizar Git, los desarrolladores utilizan comandos específicos para copiar, crear, cambiar y combinar el código. Estos comandos pueden ejecutarse directamente desde la línea de comandos o utilizando una aplicación como GitHub Desktop:

```
\$ git init
```

Inicializa un repositorio nuevo de Git y comienza a supervisar el directorio existente. Este agrega una subcarpeta oculta dentro del directorio existente que hospeda la estructura de datos interna que se requiere para el control de versiones.

```
\$ git clone
```

Crea una copia local de un proyecto que ya existe remotamente. El clon incluye todos los archivos, historial y ramas del proyecto.

```
\$ git add
```

Almacena provisionalmente un cambio (stages a change). Este comando realiza pruebas, la primera parte de este proceso de dos pasos. Cualquier cambio que se pruebe, se convertirá en parte de la siguiente snapshot y también del historial del proyecto. Las pruebas y confirmaciones(staging and committing) por separado otorgan a los desarrolladores el control completo sobre el historial y sobre el proyecto sin cambiar la forma en la que codifican y trabajan.

```
\$ git add .
```

Añade todos los archivos (que no se enumeran en .gitignore) en todo el repositorio al "staging area"

```
\$ git commit
```

Guarda el snapshot del historial del proyecto y completa el proceso de seguimiento de los cambios. En resumen, una confirmación funciona tal como el tomar una fotografía. Todo lo que se haya almacenado provisionalmente con git add pasará a formar parte de la instantánea.

```
\$ git status
```

Muestra el estado de los cambios como untracked, modificados o staged provisionalmente.

```
\$ git branch
```

Muestra las ramas en las que se trabaja localmente.

```
\$ git branch -m nuevo-nombre
```



Renombra la rama en la que estás situado.

```
\$ git branch -d nombre-rama
```

Elimina la rama que especificas.

```
\$ git merge
```

Combina las líneas de desarrollo. Este comando habitualmente se utiliza para combinar los cambios que se realizan en dos ramas distintas. Por ejemplo, un desarrollador podría hacer una fusión cuando necesite combinar los cambios de una rama de característica en la rama de desarrollo principal.

```
\$ git pull
```

Actualiza la línea de desarrollo local con actualizaciones de sus contrapartes remotas. Los desarrolladores utilizan este comando si un compañero de equipo hizo confirmaciones en una rama en un repositorio remoto y quieren reflejarlos en su ambiente local.

```
\$ git push
```

Actualiza el repositorio remoto con las confirmaciones realizadas localmente en una rama.

```
\$ git reset fichero
```

Elimina todos los ficheros especificados que estén en "staging area".

```
\$ git log
```

Muestra el historial de commits que se han realizado en el branch.

```
\$ git log --oneline
```

Muestra el historial de commits que se han realizado en el branch de forma resumida.

```
\$ git log revert idcommit
```

Revierte los cambios del commit con id idcommit.

```
\$ git restore
```

Restaurar archivos del árbol de trabajo

## 7.1. Ejemplo: Contribuir con un repositorio existente

```
1  # descargar un repositorio de GitHub en tu PC
2  # Replace por la direccion correcta para clonar
3  git clone https://github.com/owner/repo.git
4
5  # Situarse en el directorio `repo`
6  cd repo
7
8  # Crear una nueva rama para guardar cualquier nuevo cambio
9  git branch my-branch
10
11 # Cambiar a la rama "my-branch"
12 git checkout my-branch
13
14 # Trabajar en tu parte del proyecto
15
16 # Almacenar provisionalmente un cambio en el "stage area"
17 git add file1.md file2.md
18
19 # Tomar un snapshot del staging area (cual quier cosa que este dentro)
20 git commit -m "my snapshot"
21
22 # subir cambio a github
23 git push --set-upstream origin my-branch
```

## 7.2. Ejemplo: Comienza un repositorio nuevo y publícalo en GitHub

```
1  # Crear un nuevo directori, y despues lo inicializamos en git
2  git init my-repo
3
4  # Situarnos en el directorio inicializado `my-repo`
5  cd my-repo
6
7  # Crear el primer fichero dentro del directorio
8  touch README.md
9
10 # git aun no esta enterado de la creacion del fichero, stage it!
11 git add README.md
12
13 # toma un snapshot del staging area
14 git commit -m "add README to initial commit"
15
16 # proporcione la ruta para el repositorio que creó en github
17 git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY-NAME.git
18
19 # push changes to github
20 git push --set-upstream origin main
```

### 7.3. Ejemplo: contribuye con una rama existente en GitHub

```
1  # Cambiarse al directorio `repo`
2  cd repo
3
4  # Actualizar todas las ramas de trackedas remoto y la rama actualmente registrada
5  git pull
6
7  # Cambiarnos a la rama `feature-a`
8  git checkout feature-a
9
10 # Trabajamos en nuestra parte del proyecto
11
12 # subimos al stage area los cambios o ficheros nuevos
13 git add file1.md
14
15 # toma una snapshot del staging area
16 git commit -m "edit file1"
17
18 # subir los cambio a github
19 git push
```

## 8. Backend

El backend se la parte de un sistema o aplicación web que se encarga de procesar la lógica de negocio, el almacenamiento, la gestión de datos y la comunicación con otros sistemas. En términos generales, el backend se enfoca en todo lo que sucede detrás de escena para hacer posible el funcionamiento de la parte visible del sistema o aplicación, que se conoce como frontend. El backend se desarrolla utilizando diferentes tecnologías y lenguajes de programación, y su objetivo es asegurar la estabilidad, la seguridad y el rendimiento del sistema o aplicación en general.

Existen varias opciones para programar un backend, aquí te menciono algunas de las herramientas y lenguajes más utilizados para desarrollar aplicaciones web en el lado del servidor:

- Lenguajes de programación: algunos de los lenguajes más utilizados para desarrollar backend son Python, Java, Ruby, PHP, y Node.js (basado en JavaScript).
- Frameworks: los frameworks te permiten acelerar el proceso de desarrollo del backend al proporcionar un conjunto de herramientas y funcionalidades listas para usar. Algunos de los frameworks más populares son Django y Flask para Python, Ruby on Rails para Ruby, Spring para Java, Laravel para PHP, y Express para Node.js.
- Servidores web: existen varios servidores web disponibles que te permiten alojar aplicaciones web en el lado del servidor, algunos de los más populares son Apache, Nginx y Microsoft IIS.
- Servidores web: existen varios servidores web disponibles que te permiten alojar aplicaciones web en el lado del servidor, algunos de los más populares son Apache, Nginx y Microsoft IIS.
- Servidores web: existen varios servidores web disponibles que te permiten alojar aplicaciones web en el lado del servidor, algunos de los más populares son Apache, Nginx y Microsoft IIS.
- Plataformas de nube: las plataformas de nube, como Amazon Web Services (AWS), Google Cloud Platform (GCP) y Microsoft Azure, ofrecen una amplia gama de herramientas y servicios para desarrollar, implementar y escalar aplicaciones web en el lado del servidor.

En resumen, hay muchas opciones para programar un backend, y la elección dependerá de los requisitos específicos de tu proyecto y de tus preferencias personales.

## 9. Frontend

El frontend, también conocido como "parte del cliente", se refiere a la parte de un sistema o aplicación web que interactúa directamente con el usuario final. Es la parte visible y accesible del sistema o aplicación, y está diseñada para permitir a los usuarios interactuar con la funcionalidad del sistema de una manera fácil e intuitiva.

El frontend generalmente se compone de tres tecnologías principales: HTML (HyperText Markup Language), que se utiliza para crear la estructura y el contenido de las páginas web; CSS (Cascading Style Sheets), que se utiliza para definir la presentación y el estilo visual de las páginas web; y JavaScript, que se utiliza para agregar interactividad y funcionalidad avanzada a las páginas web.

El frontend se puede desarrollar utilizando diferentes frameworks y librerías de JavaScript, como React, Angular, Vue.js, y otras tecnologías como TypeScript, Webpack, y Babel, entre otras. El objetivo principal del frontend es proporcionar una interfaz de usuario atractiva y fácil de usar, con el fin de mejorar la experiencia del usuario y aumentar la usabilidad del sistema o aplicación.

## 9.1. Vue.js

Vue.js es un framework progresivo de JavaScript utilizado para crear interfaces de usuario y aplicaciones web de una sola página (Single Page Applications - SPAs). Fue creado por Evan You en 2014 y desde entonces ha ganado popularidad gracias a su simplicidad y flexibilidad.

.js se basa en la arquitectura Modelo-Vista-Controlador (MVC) y utiliza una sintaxis declarativa basada en plantillas HTML y componentes reutilizables para crear interfaces de usuario. También utiliza la programación reactiva para actualizar la interfaz de usuario en función de los cambios en los datos subyacentes.

Entre las características principales de Vue.js se incluyen:

- Su simplicidad y facilidad de aprendizaje.
- Su flexibilidad y escalabilidad, lo que lo hace adecuado tanto para pequeñas aplicaciones como para aplicaciones empresariales complejas.
- Su capacidad de crear componentes reutilizables, lo que hace que el desarrollo sea más rápido y eficiente.
- Su enfoque modular, lo que permite a los desarrolladores trabajar en diferentes partes de la aplicación al mismo tiempo.
- Su capacidad de integrarse con otras bibliotecas y frameworks de JavaScript.
- Su enfoque en el rendimiento y la eficiencia, lo que lo hace adecuado para aplicaciones que necesitan una respuesta rápida y suave.

En resumen, Vue.js es un framework de JavaScript que se utiliza para crear interfaces de usuario y aplicaciones web de una sola página. Ofrece una sintaxis declarativa, es fácil de aprender y utilizar, y se enfoca en la modularidad, la reutilización de componentes y el rendimiento.

## 10. ¿Que es una api?

Una API (Application Programming Interface) es un conjunto de reglas, protocolos y herramientas que se utilizan para comunicar diferentes aplicaciones o sistemas entre sí. En resumen, una API permite que una aplicación o servicio pueda utilizar los datos o funcionalidades de otra aplicación o servicio de una manera estandarizada y segura. Las APIs son comúnmente utilizadas para crear integraciones entre diferentes sistemas y aplicaciones, permitiendo que éstos interactúen de manera eficiente y automatizada.

## 11. El MVC

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que se utiliza en el desarrollo de aplicaciones. El objetivo del patrón MVC es separar la lógica de la aplicación en tres componentes principales: el modelo, la vista y el controlador.

- El modelo representa los datos y la lógica del negocio de la aplicación.
- La vista se encarga de mostrar la información al usuario y de recoger su interacción.
- El controlador actúa como intermediario entre el modelo y la vista, procesando la entrada del usuario, actualizando el modelo y actualizando la vista en consecuencia.

El patrón MVC se utiliza para construir aplicaciones escalables y fáciles de mantener, ya que la separación de los componentes permite una mayor flexibilidad en el desarrollo y modificación de cada uno de ellos de forma independiente. Además, el uso de este patrón facilita la reutilización del código y la colaboración en equipo al tener una estructura clara y bien definida.

## 12. ¿Que es REST?

REST (Representational State Transfer) es un estilo arquitectónico utilizado en el diseño de aplicaciones web y servicios web. Se basa en la definición de un conjunto de restricciones y principios que permiten la creación de servicios web con una interfaz uniforme y de fácil uso.

En el modelo REST, los recursos de la aplicación son identificados por una URL única y son accedidos a través de los métodos HTTP estándar (GET, POST, PUT, DELETE, entre otros). Cada recurso es representado en un formato de datos estándar, como JSON o XML, y puede ser manipulado mediante las operaciones que ofrece el protocolo HTTP.

Algunas de las características principales de REST son la escalabilidad, la interoperabilidad, la simplicidad y la capacidad de cachear los datos. REST es ampliamente utilizado en la creación de servicios web para aplicaciones móviles, redes sociales, comercio electrónico, entre otros, y es considerado como una de las tecnologías más importantes en el desarrollo de aplicaciones web y servicios web.

## 13. JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero y fácil de leer para seres humanos y para las máquinas. Es utilizado principalmente para transmitir datos estructurados a través de una red, y se basa en la sintaxis de los objetos de JavaScript.

En JSON, los datos son representados como pares de clave-valor, separados por comas, y encerrados entre llaves . Las claves son cadenas de caracteres que identifican los datos, y los valores pueden ser cualquier tipo de dato soportado por el formato, como números, cadenas, arreglos o incluso objetos anidados.

JSON es ampliamente utilizado en aplicaciones web y servicios web para el intercambio de datos, ya que es fácil de leer y escribir, y es compatible con una amplia variedad de lenguajes de programación y plataformas. Además, es más ligero que otros formatos como XML, lo que lo hace más eficiente para el envío de grandes cantidades de datos a través de la red.

## 14. HTML

HTML (HyperText Markup Language) es un lenguaje de marcado utilizado para crear páginas web. Consiste en un conjunto de etiquetas y atributos que se utilizan para definir la estructura y el contenido de una página web.

En HTML, las etiquetas se utilizan para indicar al navegador web cómo debe mostrar el contenido de la página. Por ejemplo, la etiqueta `<h1>` se utiliza para indicar que el texto siguiente debe ser mostrado como un encabezado de primer nivel, mientras que la etiqueta `<p>` se utiliza para indicar un párrafo de texto.

Además de las etiquetas, HTML también utiliza atributos para proporcionar información adicional sobre las etiquetas y sus contenidos. Por ejemplo, el atributo `src` se utiliza para indicar la URL de una imagen que se va a mostrar en la página.

HTML es el lenguaje estándar para la creación de páginas web en la World Wide Web, y es compatible con la mayoría de los navegadores web modernos. Es comúnmente utilizado junto con otros lenguajes y tecnologías web, como CSS y JavaScript, para crear páginas web interactivas y atractivas.

## 15. HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de comunicación utilizado para transferir datos a través de la World Wide Web (WWW). Se utiliza para solicitar y enviar información entre clientes (como navegadores web) y servidores web.

HTTP funciona como un conjunto de reglas y convenciones para que los clientes y servidores se comuniquen entre sí. Cuando un cliente hace una solicitud HTTP, el servidor web responde con una respuesta HTTP que puede incluir datos en forma de texto, imágenes, audio, video, entre otros.

El protocolo HTTP se basa en una estructura cliente-servidor, donde el cliente envía una solicitud al servidor para obtener información y el servidor envía una respuesta con los datos solicitados. Las solicitudes y respuestas HTTP se dividen en mensajes, y cada mensaje se compone de una línea de solicitud o respuesta, seguida de una serie de cabeceras que proporcionan información adicional sobre el mensaje.

HTTP es el protocolo estándar utilizado en la mayoría de las comunicaciones entre navegadores web y servidores web en la World Wide Web. Además, existen varias versiones de HTTP, la más reciente de ellas es HTTP/3.



## 16. Java Script

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web interactivas y dinámicas. Es un lenguaje de programación de alto nivel, interpretado y de tipado dinámico que se ejecuta en el lado del cliente, es decir, en el navegador web.

JavaScript se utiliza para agregar interactividad a una página web, como animaciones, cambios de contenido en respuesta a eventos, validación de formularios y la creación de efectos visuales. También se utiliza en el desarrollo de aplicaciones web y en el desarrollo de aplicaciones del lado del servidor utilizando Node.js.

Es un lenguaje de programación basado en objetos, lo que significa que los datos y las funcionalidades se organizan en objetos que pueden interactuar entre sí. Además, cuenta con una gran cantidad de bibliotecas y frameworks que facilitan la creación de aplicaciones web.

Se ha convertido en uno de los lenguajes de programación más populares en la web y se encuentra en constante evolución, con nuevas funcionalidades y características que se agregan regularmente.

## 17. API REST

Una API REST (o API web RESTful) es una interfaz de programación de aplicaciones (API) basada en el protocolo HTTP y los principios del estilo de arquitectura REST (Representational State Transfer). Permite a los desarrolladores acceder a recursos y datos de una aplicación o sistema a través de solicitudes HTTP como GET, POST, PUT, DELETE, etc., y obtener respuestas en formato JSON, XML u otros formatos. Las APIs REST se utilizan comúnmente en aplicaciones web y móviles para permitir la integración y la comunicación entre diferentes sistemas y servicios.

## 18. ¿Que es un Framework?

Un framework (marco de trabajo) es un conjunto de herramientas, bibliotecas y convenciones que se utilizan para desarrollar software de una manera más estructurada, reutilizable y eficiente. En lugar de tener que crear todo el código desde cero, los desarrolladores pueden utilizar el framework para aprovechar funcionalidades predefinidas y concentrarse en la lógica específica de su aplicación. Los frameworks proporcionan una estructura básica y un conjunto de reglas y patrones para organizar y desarrollar el código, lo que puede acelerar el proceso de desarrollo y mejorar la calidad y la consistencia del software. Los frameworks se utilizan comúnmente en el desarrollo de aplicaciones web, móviles y empresariales, y pueden ser específicos del lenguaje de programación utilizado, como Java, Python, PHP, entre otros.

## 19. Spring

Spring es un framework de desarrollo de aplicaciones Java que se utiliza para crear aplicaciones empresariales escalables y de alta calidad. Proporciona una amplia gama de características y módulos, que incluyen gestión de dependencias, inyección de dependencias, control de transacciones, seguridad, servicios web, etc. Spring también facilita la integración con otras tecnologías y frameworks populares, como Hibernate, Struts, JavaServer Faces (JSF) y más recientemente, Spring Boot. Con Spring, los desarrolladores pueden crear aplicaciones Java eficientes y flexibles en términos de configuración, despliegue y mantenimiento.

## 20. Spring Boot Tools Suite 4

Spring Boot Tools Suite 4 (STS 4) es un entorno de desarrollo integrado (IDE) basado en Eclipse que se utiliza para desarrollar aplicaciones basadas en el framework Spring Boot. STS 4 proporciona una serie de características y herramientas específicas de Spring, como asistentes de configuración, análisis de código, navegación de código y soporte para la creación de aplicaciones Spring Boot desde cero. También incluye herramientas para el despliegue y la gestión de aplicaciones Spring Boot, como la capacidad de crear archivos JAR y WAR, así como la integración con herramientas de gestión de contenedores, como Kubernetes. STS 4 es una herramienta muy popular entre los desarrolladores de aplicaciones Java, especialmente aquellos que utilizan el framework Spring Boot.

## 21. Fuentes

<https://docs.github.com/en>

<https://docs.github.com/en/get-started/using-git/about-git>

[http : //agrega.juntadeandalucia.es/repositorio/20022017/c8/es-an20170220129123136/41\\_controladas\\_distribuidas](http://agrega.juntadeandalucia.es/repositorio/20022017/c8/es-an20170220129123136/41_controladas_distribuidas)

<https://www.atlassian.com/es/microservices/microservices-architecture/distributed-architecture> ChatGPT