1. Introducción a la Informática
    1.1. Historia de la informática (Parte 1)

- Antecedentes Antiguos:

Los orígenes de la computación se remontan a la antigüedad, con el desarrollo de dispositivos mecánicos como el **ábaco** en Mesopotamia y China, y las primeras máquinas de calcular como el **astrolabio** y el **ábaco** chino.

- **Ábaco**:

Es uno de los dispositivos de cálculo más antiguos conocidos. Su origen se remonta a civilizaciones antiguas como la Mesopotámica, Egipcia, Griega, Romana, China e India. Consiste en un marco con cuentas o fichas que se mueven a lo largo de barras o alambres y se utiliza para realizar operaciones aritméticas básicas como la suma, la resta, la multiplicación y la división.

- **Astrolabio**:

Es un instrumento astronómico utilizado para medir la altura de los cuerpos celestes sobre el horizonte, aunque su principal función era la navegación y la astrología , también se usaba para realizar cálculos trigonométricos y resolver problemas matemáticos relacionados con la astronomía.

- **Ábaco Chino**:

También conocido como *suanpan*, es una versión más sofisticada del ábaco común la cual se utilizaba en China y otras partes de Asia, para realizar cálculos rápidos y precisos. Consiste en un marco rectangular con cuentas dispuestas en columnas y se utiliza el movimiento de las cuentas hacia arriba y hacia abajo para representar diferentes valores numéricos.

- La Era de la Mecánica:

En el siglo XVII, **Blaise Pascal** inventó la primera **calculadora mecánica**. Luego, en el siglo XIX, **Charles Babbage** diseñó la **"Máquina Diferencial"**, y la **"Máquina Analítica"**, considerada la *precursora* de las computadoras modernas.

- **Blaise Pascal y la Calculadora Mecánica**:

En 1642, el matemático y filósofo francés inventó la primera calculadora mecánica conocida como *Pascalina*. Este dispositivo mecánico fue diseñado para ayudar en la realización de cálculos aritméticos simples como las sumas y restas. La *Pascalina* estaba basada en un mecanismo de ruedas dentadas que permitía sumar cantidades predefinidas con facilidad. Aunque tenía limitaciones en términos de funcionalidad y capacidad, sentó las bases para el desarrollo de futuras máquinas de cálculo.

- **Máquinas Analíticas**:

A principios del siglo XIX, el matemático britanico Charles Babbage diseñó dos máquinas revolucionarias: La Máquina Diferencial y la Máquina Analítica.
En el caso de la Máquina Diferencial, concebida en 1822, estaba destinada a calcular tablas matemáticas mediante el uso de inferencias finitas. Su diseño influyó en el desarrollo posterior de las computadoras.
La Máquina Analítica, concebida por Babbage en la década de 1830, es considerada la precursora de las computadoras modernas. Esta máquina teórica fue diseñada para realizar

cualquier cálculo matemático siguiendo instrucciones almacenadas en tarjetas perforadas, lo que la convierte en la primera máquina programable. Incluye componentes clave como la Unidad de Control, la Unidad Aritmética y la Memoria, conceptos que son fundamentales en la arquitectura de las computadoras modernas.

A pesar de que las máquinas de Pascal y Babbage eran mecánicas y no eléctricas, sentaron las bases para el desarrollo de la computación moderna al introducir conceptos como la automatización de cálculos, el almacenamiento de datos y la programación. Sus innovaciones allanaron el camino para futuros avances en la tecnología y la computación que eventualmente llevarían al surgimiento de las computadoras electrónicas del siglo XX.

- La primera programadora:

Ada Lovelace, en la década de 1840, colaboró con **Charles Babbage** en el desarrollo de la **Máquina Analítica**. Sus notas incluyen el primer *algoritmo* diseñado para ser procesado por una *máquina*, lo que la convierte en la **primera programadora conocida**.

Nacida como Augusta Ada Byron en 1815, Ada Lovelace fue una matemática y escritora británica conocida principalmente por su trabajo pionero en el desarrollo de la computación temprana. Era hija del famoso poeta Lor Byron y Anna Isabella Milbanke.

Ada conoció a Charles Babbage en 1833 cuando solo tenía 17 años. Babbage, impresionado por su talento matemático, la reclutó para trabajar en su proyecto de la Máquina Analítica. Lovelace comprendió rápidamente los conceptos detrás de la máquina y comenzó a colaborar estrechamente con Babbage en su desarrollo.

En 1843 escribió una serie de notas sobre la Máquina Analítica que incluía un algoritmo detallado para calcular los números de Bernoulli. Este algoritmo se considera el primer algoritmo diseñado específicamente para ser procesado por una máquina, lo que convierte a Ada Lovelace en la primera programadora conocida en la historia.

Además de estas notas, Ada anticipó muchas de las ideas clave detrás de la informática moderna como los conceptos de bucles y subrutinas.

A pesar de que la Máquina Analítica de Babbage nunca se construyó completamente, el trabajo de Ada Lovelace sentó las bases para el desarrollo posterior de la informática, y su visión y comprensión de la máquina, así como su habilidad para anticipar su potencial, la han convertido en una figura icónica en el campo de la informática.

- Inteligencia Artificial:

En 1956, en la Conferencia de Dartmouth, se acuñó el término **“Inteligencia Artificial”** y se marcó el comienzo de la investigación formal en este campo. Pioneros como **John McCarthy**, **Marvin Minsky**, **Allen Newell** y **Herbert Simon** sentaron las bases de la IA.

    - John McCarthy:

Considerado el padre de la Inteligencia Artificial. Fue uno de los organizadores de la Conferencia de Dartmouth y es muy conocido por desarrollar el lenguaje de programación LISP en 1958, que se convirtió en uno de los principales lenguajes para la investigación de Inteligencia Artificial.

Además contribuyó significativamente al desarrollo de algoritmos y técnicas de Inteligencia Artificial.

- Marvin Misnky:

Fue otro de los líderes del campo de la Inteligencia Artificial. Junto con McCarthy fundó el Laboratorio de Inteligencia Artificial del MIT en 1959.
Minsky realizó importantes investigaciones en áreas de reconocimiento de patrones, percepción visual y sistemas expertos.

- Allen Newell y Herbert Simon:

Desarrollaron el Logic Theorist en 1956, un programa de computadoras capaz de demostrar teoremas en lógica proposicional. Este programa se considera uno de los primeros ejemplos de IA y sentó las bases para el desarrollo de sistemas expertos y resolución de problemas mediante el razonamiento simbólico.
Estos pioneros sentaron las bases teóricas y prácticas de la IA, explorando conceptos como el razonamiento lógico, el aprendizaje automático, la percepción y la toma de decisiones.

A partir de la Conferencia de Dartmouth la investigación en IA ha experimentado un crecimiento explosivo y ha dado lugar a numerosas aplicaciones prácticas en los campos de robótica, medicina, sistemas de recomendación, entre otros.

1.2. Historia de la informática (Parte 2)

- Informática Personal:

En la década de 1970, surgieron las primeras *computadoras personales*, como el **Altair 8800**, seguido por el **Apple I** y el **Apple II** de **Steve Wozniak** y **Steve Jobs**.

- Altair 8800:

En 1975 MITS lanza el Altair 8800, que fue una de las primeras computadoras personales disponibles comercialmente. Era un kit que requería montaje por parte del usuario y no tenía monitor, teclado, ni almacenamiento de datos integrados, sin embargo, su disponibilidad para entusiastas y aficionados marcó un hito importante en la historia de la computación personal.

- Apple I y Apple II:

Wozniak y Jobs fundaron Apple Computers en 1976 y lanzaron Apple I como su primer producto. Diseñado por Wozniak y comercializado por Jobs, el Apple I fue una computadora de placa única que marcó el comienzo de la empresa Apple. Pocos años después, en 1977, Apple lanzó Apple II, que fue un gran éxito comercial. El Apple II fue una de las primeras computadoras personales en tener un teclado integrado, capacidad para color y soporte para gráficos, lo que la convirtió en una opción muy popular para el uso doméstico y educativo.

- TRS 80 y Commodores Pet:

TRS 80 de RadioShack, ambas popularizaron el uso de las computadoras en el hogar y en la educación.

El surgimiento de las computadoras personales tuvo un impacto significativo en la sociedad y la cultura. Se volvieron más accesibles al público en general, lo que impulsó la alfabetización digital y la creación de una industria de software comercial.

También permite el desarrollo de nuevas formas de entretenimiento y comunicación como los videojuegos y redes sociales muchos años después.

- Internet:

En la década de 1960, **ARPANET**, una red de comunicaciones financiada por el gobierno de los EE.UU., sentó las bases de lo que más tarde se convertiría en Internet. **Tim Berners-Lee** creó la **World Wide Web** en 1989, revolucionando la forma en que compartimos y accedemos a la información.

- ARPANET:

Significa Advanced Research Projects Agency Network, es decir, la Red de la Agencia de Proyectos de Investigación Avanzada. Fue una red de computadoras desarrollada por esta Agencia del Departamento de Defensa de los EE. UU. en la década de 1960.

Fue diseñada para facilitar la comunicación en intercambio de información entre investigadores y científicos en instituciones académicas y militares.

ARPANET sentó las bases de los que más tarde se convertirían en Internet, estableciendo el protocolo de comunicación TCP/IP que sigue siendo la base de Internet hasta el día de hoy.

- World Wide Web:

Aunque ARPANET estableció la infraestructura básica de la Internet, la creación de la World Wide Web (WWW), revolucionó la forma en la que interactuamos con la información en línea.

En 1989 Tim Berners-Lee, científico de la computación del CERN, desarrolló un sistema de hipertexto que permitía a los usuarios navegar por documentos vinculados a través de internet. Este sistema, conocido como WWW, fue presentado por primera vez en 1991 y proporcionó una forma fácil y accesible de compartir y acceder a información de la red.

La web se expandió rápidamente y se convirtió en el principal motor de crecimiento de internet. Desde sus modestos comienzos internet ha experimentado un crecimiento exponencial y ha evolucionado para convertirse en la red global que conecta a miles de millones de personas alrededor del mundo. Se ha convertido en un recurso indispensable para la comunicación, la educación, el entretenimiento y mucho más.

El desarrollo de tecnologías, como los motores de búsqueda, los sitios webs interactivos, las redes sociales y el comercio electrónico han ampliado aún más las capacidades de internet y transformado la vida que vivimos y trabajamos.

- El Auge de las Empresas de Tecnología:

Empresas como **Microsoft**, fundada por **Bill Gates** y **Paul Allen** en 1975, y **Google**, fundada por **Larry Page** y **Sergey Brin** en 1998, se convirtieron en gigantes de la *Tecnología* que transformaron la informática y la sociedad.

Microsoft fue fundada en Alburquerque, Nuevo México, e inicialmente se centró en el desarrollo de software para una incipiente industria de la computación personal. Uno de los primeros productos exitosos de Microsoft fue el MS-DOS (MicroSoft-Disk Operating System) lanzado en 1981. DOS se convirtió en el Sistema Operativo estándar para las computadoras personales durante la década de los 80. En 1985 lanzó Windows, un sistema operativo con una interfaz gráfica de usuario que se convirtió en una plataforma dominante para las computadoras personales en el mundo. Luego continuó expandiendo su cartera de productos, incluyendo aplicaciones de productividad como Microsoft Office, herramientas de desarrollo como Visual Studio y servicios en la nube como Azure.

Google fue un proyecto de investigación mientras Larry Page y Sergey Brin estudian en la Universidad de Stanford, su objetivo era organizar la información en la web de manera más efectiva. Lanzaron su motor de búsqueda en línea en 1998 que rápidamente se convirtió en el motor de búsqueda más utilizado del mundo debido a su capacidad de ofrecer resultados relevantes y rápidos. A lo largo de los años Google ha diversificado sus servicios, incluyendo: Gmail, Google Maps, Google Drive, YouTube y Android, el sistema operativo móvil más utilizado en el mundo. Además de los servicios de consumo, también incursionaron en áreas de Inteligencia Artificial, la Computación de Nube (Google Cloud Platform) y la conducción autónoma a través de Waymo. Google también es reconocido por su impacto en la tecnología y la informática, y en la cultura empresarial innovadora y su compromiso con la investigación y el desarrollo.

- La Era de los dispositivos Móviles:

A partir de la década del 2000, el **iPhone** de **Apple** y los dispositivos **Android** cambiaron la forma en que interactuamos con la *tecnología*, llevando a la computación a nuestras manos en forma de **teléfonos inteligentes** y **tabletas**.

- Computación Cuántica:

En la actualidad, la **computación cuántica** promete revolucionar la informática al ofrecer una potencia de procesamiento sin precedentes. Empresas como **IBM**, **Google** y **Microsoft** están compitiendo para desarrollar esta tecnología.

- Machine Learning:

Es una rama de la Inteligencia Artificial, la cual se basa en desarrollar algoritmos o modelos en computadoras que les permitan aprender de forma automática sin la intervención humana directa en base a una experiencia previa que le ayuda a reconocer patrones y tomar decisiones en base a esa experiencia. **Open AI** y **Chat GPT**.

1.3. Hardware

- ¿Qué es el Hardware?

El hardware se refiere a los componentes físicos de un sistema informático que incluyen dispositivos como procesadores, memoria, unidades de almacenamiento, placas base, periféricos y más.

- **Componentes esenciales**: Procesador (CPU), Memoria RAM, Unidad de almacenamiento (Disco duro o SSD), Placa base, Tarjeta gráfica (en computadoras que lo requieran).
- **Dispositivos periféricos**: Teclado, Ratón, Monitor, Impresora, Altavoces, entre otros.
- **Variedad de hardware**: Puede variar desde computadoras de escritorio hasta dispositivos móviles, servidores, sistemas embebidos y más.
- **Importancia**: Es esencial para el funcionamiento de cualquier sistema informático y su rendimiento puede influir significativamente en la experiencia de usuario.

- Componentes Esenciales:

Son los elementos fundamentales que constituyen un sistema informático y son necesarios para su funcionamiento básico.

- **Procesador (CPU)**: Es el cerebro del ordenador, encargado de ejecutar las instrucciones y procesar los datos. Realiza operaciones aritméticas, lógicas y de control.
- **Memoria RAM (Memoria de Acceso Aleatorio)**: Es la memoria temporal utilizada por el sistema operativo y los programas en ejecución para almacenar datos y ejecutar procesos de manera rápida y eficiente.
- **Unidad de almacenamiento (Disco duro o SSD)**: Es el dispositivo donde se almacenan permanentemente los datos del sistema operativo, programas, archivos y documentos. Los discos duros tradicionales utilizan discos magnéticos para almacenar datos, mientras que los SSD (Solid State Drive) utilizan memoria flash, lo que los hace más rápidos y eficientes en términos de acceso de datos.
- **Placa base**: Es el componente principal de un ordenador, que conecta todos los demás componentes entre sí. Contiene el chipset, los puertos de conexión, las ranuras de expansión y otros circuitos necesarios para el funcionamiento del sistema.
- **Tarjeta gráfica (en computadoras que la requieran)**: Es responsable de procesar y generar las imágenes que se muestran en el monitor. En algunos casos, como en las computadoras de escritorio para juegos o diseño gráfico, se utiliza una tarjeta gráfica independiente para mejorar el rendimiento gráfico.

- Periféricos:

Los periféricos son dispositivos externos conectados a una computadora u otro dispositivo electrónico para ampliar sus capacidades de entrada, salida o interacción con el usuario. Estos dispositivos complementan las funciones principales del sistema y permiten realizar tareas específicas.

- **Teclado**: Permite ingresar datos al sistema mediante la pulsación de teclas. Esencial para la entrada de texto y comandos.
- **Ratón**: Proporciona una forma intuitiva de interactuar con la computadora mediante movimientos y clics. Se utiliza principalmente para seleccionar, arrastrar y hacer clic en elementos en la pantalla.
- **Monitor**: Muestra la salida visual del sistema, incluyendo texto, gráficos, imágenes y videos. Es el principal dispositivo de salida visual de la PC.

- **Impresora**: Permite imprimir documentos y gráficos en papel u otros medios físicos. Hay diferentes tipos de impresoras, como impresoras de inyección de tinta, láser y de matriz de puntos.
- **Altavoces**: Producen sonido para la reproducción de audio, incluyendo música, efectos de sonido y diálogos. Se utilizan para escuchar contenido multimedia y comunicarse en llamadas de voz y videoconferencias.
- **Micrófono**: Captura el sonido del ambiente o la voz del usuario para su procesamiento y grabación. Se utiliza en aplicaciones de grabación de audio, llamadas de voz y comandos de voz.

- Diferencias entre dispositivos:
    - **Computadoras de escritorio**:

Son sistemas informáticos completos, diseñados para utilizarse en un escritorio o lugar físico. Tienden a tener una mayor potencia de procesamiento, capacidad de almacenamiento y capacidad de expansión en comparación con dispositivos móviles. Son ideales para tareas que requieren un alto rendimiento, como edición de video, diseño gráfico, programación, etc. Suelen tener un factor de forma más grande y pueden incluir componentes adicionales como tarjetas gráficas dedicadas.

    - **Dispositivos móviles**:

Son dispositivos portátiles diseñados para ser utilizados sobre la marcha. Tienen limitaciones de hardware en comparación con las computadoras de escritorio, pero ofrecen una gran portabilidad y conectividad. Son ideales para tareas como navegación web, redes sociales, correo electrónico, reproducción de medios y aplicaciones móviles. Tienen pantallas táctiles y suelen depender de sistemas operativos móviles como iOS (iPhone, iPad) o Android.

    - **Servidores**:

Son sistemas informáticos diseñados para proporcionar servicios, recursos o funcionalidades a otros dispositivos o usuarios en una red. Están optimizados para tareas de almacenamiento, procesamiento y gestión de grandes cantidades de datos y tráfico de red. Pueden ser servidores web, servidores de bases de datos, servidores de correo electrónico, servidores de archivos, etc. Suelen operar de forma continua y están configurados para ser seguros y confiables.

    - **Sistemas embebidos**:

Son sistemas informáticos diseñados para realizar funciones específicas dentro de dispositivos más grandes o sistemas integrados. Tienen recursos limitados en comparación con las computadoras de escritorio o servidores, pero optimizados para tareas específicas. Se utilizan en una amplia gama de aplicaciones, como dispositivos médicos, electrodomésticos inteligentes, sistemas de control industrial, automóviles, etc. Suelen tener un diseño compacto, consumo de energía eficiente y pueden operar en condiciones ambientales adversas.

1.4. Software

- ¿Qué es el Software?

Es el conjunto de programas, instrucciones y datos necesarios para realizar diversas tareas en una computadora. Es una parte fundamental de cualquier sistema informático y desempeña un papel crucial en la realización de funciones específicas, desde procesar datos hasta ejecutar aplicaciones.

- Lenguaje de Máquina:

Es el conjunto de instrucciones directamente interpretable por el hardware de una computadora. Consiste en una secuencia de códigos binarios, representados por combinaciones de 0 y 1, que indican operaciones elementales como sumar, restar, mover datos, entre otros.

Cada procesador tiene su propio conjunto de instrucciones en el lenguaje de máquina, conocido como conjunto de instrucciones de arquitectura (ISA). Estas instrucciones se ejecutan en la unidad de procesamiento central (CPU) y controlan el flujo de datos y las operaciones realizadas por la computadora.

El lenguaje de máquina es fundamental para el funcionamiento de cualquier programa o aplicación, ya que proporciona la base para la ejecución de las operaciones a nivel de hardware. Aunque es el lenguaje más básico y difícil de entender para los humanos, todos los programas de software eventualmente se traducen a lenguaje máquina para ser ejecutados por el procesador.

- Lenguaje de Ensamblador:

También como lenguaje de ensamblador o Assembly, es un lenguaje de programación de bajo nivel que proporciona una representación simbólica de las instrucciones de máquina y otros recursos del sistema, lo que lo hace más legible para los humanos que el lenguaje máquina.

Cada instrucción en lenguaje ensamblador representa una instrucción del lenguaje de máquina correspondiente, facilitando la programación a un nivel más cercano al hardware.

Aunque requiere un mayor entendimiento de la arquitectura de la computadora, el lenguaje de ensamblador ofrece un mayor control sobre el hardware y la optimización de programas para tareas específicas.

- Lenguaje de Alto Nivel:

Los lenguajes de alto nivel son herramientas fundamentales en el desarrollo de software moderno.

Estos lenguajes están diseñados para ser más comprensibles para los programadores humanos, utilizando una sintaxis más cercana al lenguaje natural. En contraste con los lenguajes de bajo nivel, como el lenguaje de máquina y el lenguaje ensamblador, los lenguajes de alto nivel abstractizan la complejidad del hardware subyacente, permitiendo a los desarrolladores concentrarse en la lógica y la estructura de sus programas en lugar de preocuparse por los detalles específicos de la arquitectura del computador.

Ejemplo comunes de lenguajes de alto nivel incluyen Python, Java, C++ y Javascript. Estos lenguajes ofrecen una amplia gama de funcionalidades y paradigmas de programación, lo que los hace adecuados para una variedad de aplicaciones, desde desarrollo web hasta análisis de datos y aplicaciones móviles.

- Lenguaje de Programacion:

Es un conjunto de reglas y símbolos que permiten a los programadores escribir instrucciones para que la computadora las ejecute. Estos lenguajes actúan como intermediarios entre el ser humano y la máquina, permitiendo la creación de software para una amplia variedad de aplicaciones.

Desde aplicaciones móviles hasta sistemas de gestión empresarial, cada programa informático se construye utilizando uno o varios lenguajes de programación, cada uno con sus propias reglas y sintaxis.

- Sistemas Operativos:

Los sistemas operativos son el corazón de cualquier dispositivo informático. Actúan como intermediarios entre el hardware y el software, gestionando recursos como la memoria, el procesador, los dispositivos de entrada y salida, entre otros.

Además de proporcionar una interfaz para que los usuarios interactúen con la computadora, los sistemas operativos también ofrecen servicios como la gestión de archivos, la seguridad del sistema y la multitarea, permitiendo la ejecución simultánea de múltiples programas.

Su importancia radica en su capacidad para facilitar y optimizar el uso de la computadora, brindando una experiencia eficiente y segura al usuario.

- Software Libre vs. Propietario:

Software Libre	Software Propietario
El software libre se refiere a programas informáticos que otorgan a los usuarios la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Estos programas se basan en licencias de código abierto que permiten a la comunidad acceder al código fuente, modificando según sus necesidades y redistribuirlo libremente. Ejemplos populares de software libre incluyen el sistema operativo Linux, el navegador web Mozilla Firefox y la suite de oficina LibreOffice.	El software propietario es aquel cuyo uso, redistribución o modificación están restringidos por derechos de autor y licencias de software. Los usuarios deben adquirir una licencia o permiso para utilizar este tipo de software, y no tienen acceso al código fuente ni pueden modificarlo. Ejemplo comunes de software propietario incluyen el sistema operativo Windows, la suite de productividad Microsoft Office y Adobe Photoshop.

1.5. Servidores

- Introducción a los Servidores:

Los servidores son componentes fundamentales en el mundo de la tecnología de la información. Actúan como el corazón de las redes informáticas, proporcionando servicios, almacenamiento y recursos a otras computadoras en la red. Desde el alojamiento de sitios web hasta la gestión de bases de datos y el correo electrónico, los servidores desempeñan

un papel crucial en la entrega de aplicaciones y datos en entornos empresariales y de consumo.

- ¿Qué es un servidor?:

Es una computadora dedicada a proporcionar servicios, recursos y datos a otras computadoras en una red, conocidas como clientes. A diferencia de las computadoras de escritorio, que están diseñadas para ser utilizadas por un solo usuario, los servidores están optimizados para funcionar de manera eficiente en entornos de red y manejar múltiples solicitudes de manera simultánea.

Los servidores desempeñan un papel crucial en la infraestructura de tecnologías de la información (IT), sirviendo como centros de almacenamiento, procesamiento y distribución de datos. Pueden ejecutar una variedad de aplicaciones y servicios, como servidores web para hospedar sitios y aplicaciones en línea, servidores de correo electrónico para gestionar la comunicación electrónica, servidores de bases de datos para almacenar y administrar información estructurada, entre otros.

- Tipos de Servidores:
    - **Servidores Web**:

Los servidores web son los responsables de alojar y entregar sitios web y aplicaciones web a los usuarios a través de internet. Utilizan protocolos HTTP y HTTPS para comunicarse con los navegadores web.

- **Servidores de Correo Electrónico**:

Los servidores de correo electrónico gestionan la entrega y recepción de correos electrónicos. Utilizaron protocolos como SMTP, POP3 e IMAP para enviar, recibir y almacenar mensajes de correo electrónico.

- **Servidores de Bases de Datos**:

Los servidores de bases de datos almacenan y gestionan datos estructurados en una base de datos. Proporciona acceso concurrente a múltiples usuarios y aplicaciones garantizando la integridad, seguridad y disponibilidad de los datos.

- **Servidores de Archivos**:

Los servidores de archivos almacenan y comparten archivos y recursos en una red. Permiten a los usuarios acceder y compartir archivos y recursos en una red. Permiten a los usuarios acceder y compartir archivos de forma centralizada, facilitando la colaboración y la gestión de datos.

- **Servidores de Aplicaciones**:

Los servidores de aplicaciones ejecutan y gestionan aplicaciones empresariales servicios en una red. Proporcionan entornos de ejecución para aplicaciones web y empresariales, facilitando el desarrollo, implementación y administración de software.

- **Servidores de Impresión**:

Los servidores de impresión gestionan las impresoras y los trabajos de impresión en una red.

- Hardware de un Servidor:

El hardware de un servidor está diseñado para ofrecer un rendimiento confiable y escalable. Suele incluir componentes como procesadores potentes, memoria RAM amplia, unidades de almacenamiento de alta capacidad (discos duros o SSD).

- Seguridad de los Servidores:

La seguridad de los servidores es fundamental para proteger los datos y los recursos de la red contra amenazas como virus, malware, ataques cibernéticos y accesos no autorizados. Se utilizan medidas de seguridad como firewalls, cifrado de datos, autenticación de usuarios y actualizaciones de software para mitigar estos riesgos.

- Funciones de un Servidor:

Los servidores desempeñan diversas funciones en una red, como el almacenamiento y distribución de archivos, la gestión de bases de datos, la provisión de servicios de correo electrónico y el hospedaje de sitios web. También pueden proporcionar seguridad, control de acceso, monitoreo y administración de la red.

1.6. Interfaz de usuario

User Interface (UI): es el punto de encuentro entre el usuario y el sistema. Botones, entradas de texto, o una consola donde ingresar instrucciones. Lo importante es que este medio permita al usuario generar instrucciones al sistema de forma eficiente.

- Tipos de Interfaces de Usuario:

Las interfaces de usuario son puntos de interacción entre los usuarios y los sistemas informáticos. Hay varios tipos de interfaces, cada una con sus propias características y usos específicos.

- Interfaz de Línea de Comandos (CLI):

La interfaz de Línea de Comandos (CLI) es una interfaz de texto que permite a los usuarios interactuar con el sistema a través de comandos escritos. Los usuarios ingresan comandos específicos y reciben respuestas en forma de texto. Es especialmente eficiente para usuarios experimentados y tareas repetitivas, pero puede ser menos intuitiva para los principiantes.

- **Características:**
  - Utiliza comandos escritos.
  - No es visualmente atractiva.
  - Eficiente para tareas específicas.
  - **Ejemplo**: Terminal de Unix, Command Prompt en Windows.

- Interfaz Gráfica de Usuario (GUI):

La GUI utiliza elementos visuales como iconos, ventanas, y menús para facilitar la interacción del usuario. Es intuitiva y fácil de usar, adecuada para usuarios de todos los niveles de experiencia.

- **Características**:
  - Utiliza elementos visuales, fáciles de aprender y usar.
  - Atractiva visualmente.
  - **Ejemplos**: Windows, macOS, interfaces de aplicaciones móviles.

- Interfaz Nativa de Usuario (NUI):

La NUI permite la interacción del usuario a través de gestos, voz u otras formas naturales de comunicación. Elimina la necesidad de dispositivos de entrada tradicionales como teclados o ratones.

- **Características:**
    - Utiliza gestos, voz u otros medios naturales.
    - Enfoque en la experiencia del usuario.
    - Promueve la interacción intuitiva.
    - **Ejemplo**: Tecnología de reconocimiento de voz, interfaces táctiles en dispositivos móviles.

2. Interfaz de Usuario - Termina (CLI)
    2.1. Terminal

- Introducción a la Terminal:
    - **Definición y Fundamentos**:

La terminal es una **interfaz de línea de comandos (CLI)** que permite a los usuarios interactuar con un sistema operativo a través de **instrucciones de texto**. En lugar de utilizar *interfaces gráficas de usuario (GUI)*, como las que se encuentran en la mayoría de los sistemas operativos modernos, **la terminal** permite a los usuarios ejecutar comandos directamente al sistema.

Los fundamentos de **la terminal** incluyen la comprensión de conceptos como el **prompt** (el símbolo que indica que la terminal está lista para escribir un comando), los **directorios** (**carpetas**) y los **archivos** dentro del sistema de archivos del sistema operativo.

- **Consola, terminal y CLI**:

Estos términos se utilizan indistintamente para referirse al entorno en el que se ejecutan comandos de texto en un sistema informático. Aunque pueden tener matices ligeramente diferentes dependiendo del contexto, en general se refieren a la misma idea básica: una interfaz que permite a los usuarios interactuar con un sistema a través de comandos de texto en lugar de una interfaz gráfica de usuario.

Consola: Este término puede referirse tanto a la pantalla y el teclado físico conectado a una computadora como al software que emula dicha pantalla y teclado en un entorno virtual. En este último caso, la consola generalmente proporciona acceso a la terminal y a la CLI.

Terminal: Es el software o la aplicación que proporciona una interfaz de línea de comandos para interactuar con el sistema operativo. En sistemas Unix y Linux, la terminal es la principal forma de acceder a la CLI.

CLI (Interfaz de Línea de Comandos): Es un tipo de interfaz de usuario que permitió a los usuarios interactuar con un programa o sistema emitiendo comandos de texto simples. En el contexto de este tema, la CLI se utiliza a menudo a través de la terminal.

- **Ventajas de su uso**:

Es una herramienta versátil que ofrece a los usuarios la capacidad de interactuar directamente con el sistema operativo,

Eficiencia: La terminal permite a los usuarios realizar tareas de forma más rápida y eficiente, ya que pueden ejecutar comandos directamente sin necesidad de navegar a través de múltiples menús y opciones gráficas.

Automatización: Mediante el uso de scripts y la combinación de comandos, es posible automatizar tareas repetitivas en la terminal, lo que ahorra tiempo y reduce errores.

Acceso remoto: La terminal permite a los usuarios acceder y administrar sistemas de forma remota a través de conexiones de red, lo que resulta útil para administrar servidores y equipos sin necesidad de estar físicamente presentes en el lugar.

Potencia y flexibilidad: La terminal proporciona acceso directo a una amplia gama de herramientas y utilidades del sistema operativo, lo que brinda a los usuarios un mayor control y flexibilidad para realizar diversas tareas y personalizar su entorno de trabajo según sus necesidades específicas.

- **Estructura de la Terminal**:

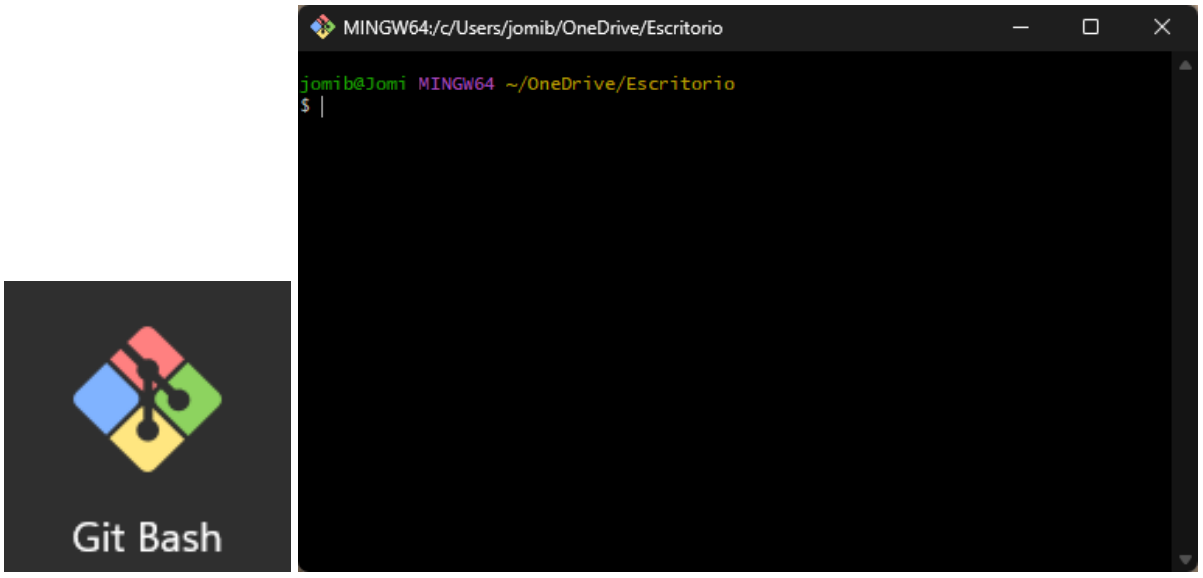
Prompt: Es el símbolo o texto que indica que la terminal está lista para recibir un comando. Por lo general, muestra información como el nombre de usuario, el nombre del equipo y la ubicación actual en el sistema de archivos.

Directorios Archivos: Los directorios son carpetas que contienen archivos y otros directorios. Los archivos son unidades de información que pueden contener datos. En la terminal, se utilizan comandos para navegar entre directorios (cd), listar archivos y directorios (ls) y crear (mkdir), copiar (cp) y eliminar (rm) archivos y directorios.

Permisos de Ejecución: En los sistemas Unix y Linux, cada archivo y directorio tiene permisos asociados que controlan quién puede leer, escribir o ejecutarlos. Los permisos de ejecución determinan si un archivo puede ser ejecutado como programa o script. Los comandos *chmod* y *chown* se utilizan para modificar los permisos de los archivos y directorios en la terminal.
Windows tiene un sistema de permisos, aunque la forma de gestionarlos es diferente a la de los sistemas Unix y Linux, y suele realizarse a través de la interfaz gráfica de usuario en lugar de la línea de comandos.

2.2. Instalar terminal Bash

https://git-scm.com/



2.3. Comandos básicos en la terminal

Comando	Descripción
“ls”	Lista los archivos y directorios en el directorio actual.
“ls -a”	Flag “-a” muestra archivos y directorios ocultos.
“cd [directorio]”	Cambia el directorio actual al especificado: “cd /home/usuario”.
“pwd”	Muestra la ruta del directorio actual.
“touch [nombre.extensión]”	Crea un archivo con el tipo de extensión especificado en el directorio actual. También se puede usar para actualizar la fecha de acceso de un archivo existente.
“rm [archivo/directorio]”	Elimina archivos o directorios. Los archivos eliminados no se pueden recuperar fácilmente.
“cp [origen] [destino]”	Copia archivos o directorios desde el origen especificado al destino especificado.
“mv [origen] [destino]”	Mueve archivos desde el origen al destino especificado.
“mv [nombre] [nombre nuevo]”	Cambia el nombre del archivo especificado al nombre especificado.
“cat [archivo]”	Muestra el contenido de un archivo en la terminal.
“grep [patrón] [archivo]”	Busca texto en archivos utilizando un patrón específico.
“find [directorio] [opciones]”	Busca archivos y directorios especificados según las opciones dadas.
“chmod [permisos] [archivo/directorio]”	Cambia los permisos de archivos y directorios.
“chown [usuario:grupo] [archivo/directorio]”	Cambia el propietario y el grupo de archivos y directorios.
“sudo [comando]“	Ejecuta un comando con privilegios de superusuario.

2.4. Diferencias con PowerShell de Windows

Comando	Descripción
“dir”	Lista los archivos y directorios en el directorio actual.
“cd [directorio]”	Cambia el directorio actual al especificado.
“cd ..”	Retrocede un nivel en la jerarquía de directorios.
“cd \”	Cambia al directorio raíz.
“cd $env:USERPROFILE”	Cambia al directorio de perfil de usuario.
“pwd”	Muestra la ruta del directorio actual.
“mkdir [directorio]”	Crea un directorio con el nombre especificado.
“rmdir [directorio]”	Elimina un directorio.
“del [archivo]”	Elimina el archivo especificado.
“copy [origen] [destino]”	Copia archivos o directorios desde el origen al destino especificado.
“move [origen] [destino]”	Mueve o renombra archivos y directorios.
“type [archivo]”	Muestra el contenido de un archivo en la terminal.
“gc [archivo]”	*Get-Content* muestra el contenido.
“Select-String [patrón] [archivo]”	Busca texto en archivos utilizando un patrón específico.
“Get-ChildItem [directorio]”	Obtiene los archivos y directorios en el directorio especificado.
“New-Item [directorio/archivo] -ItemType [tipo]”	Crea un nuevo directorio o archivo con el nombre especificado.
“Remove-Item [directorio/archivo]”	Elimina un directorio o archivo.

3. Git

3.1. Instalaciones Necesarias GIT

https://git-scm.com/
Distributed Version Control System
Agregar al PATH.
git --version

3.2. GIT (marco teórico)

- Introducción a Git:

Exploraremos los conceptos esenciales de Git, un VCS (Sistema de Control de Versiones) ampliamente utilizado en el desarrollo de software moderno. Desde sus fundamentos hasta su implementación práctica, descubriremos cómo Git facilita la gestión eficiente de versiones en proyectos de cualquier tamaño.

- Tipos de Versionados - Local:

En el contexto del control de versiones, existen varios enfoques para gestionar cambios en el código fuente y otros archivos. Uno de los enfoques más básicos es el versionado local. En este sistema, los cambios se realizan y registran sólo en el sistema de archivos local de un desarrollador, lo que significa que no hay colaboración directa ni historial de versiones compartido entre los miembros del equipo.

El versionado local presenta limitaciones significativas en entornos de desarrollo colaborativo, ya que dificulta la coordinación de cambios entre varios desarrolladores y no proporciona un historial centralizado de versiones para el proyecto.

Exploraremos cómo Git, un sistema de control de versiones distribuido, aborda estas limitaciones y proporciona una solución más robusta y flexible para la gestión de versiones en entornos colaborativos.

- Definición de VCS - Git y Alternativas:

Sistema de Control de Versiones (VCS)	Git	Alternativas a Git
Es una herramienta fundamental en el desarrollo de software que permite gestionar los cambios y otros archivos a lo largo del tiempo. Proporciona un historial detallado de las modificaciones, facilitando la colaboración entre equipos de desarrollo, el seguimiento de errores y la implementación de nuevas características de manera ordenada.	Es el VCS más utilizado en la actualidad, conocido por su velocidad, flexibilidad y potentes capacidades de ramificación y fusión. Desarrollado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux, Git se ha convertido en el estándar de facto en la industria del software.	Aunque Git es dominante, existen otras alternativas a VCS, como Subversion (SVN), Mercurial, CVS, entre otros. Cada uno tiene sus propias características y ventajas, pero Git destaca por su escalabilidad, rendimiento y robustez, lo que lo convierte en la elección preferida para la mayoría de los proyectos de desarrollo de software modernos.

- Instalación y Configuración:

1ro. Git puede ser instalado en una variedad de sistemas operativos, incluyendo Windows, macOS y Linux.
2do. En Windows, se puede descargar e instalar Git desde el sitio web oficial de Git (https://git-scm.com/).
3ro. En macOS, Git a menudo ya está disponible a través de la línea de comandos, pero también se puede instalar usando Homebrew u otras herramientas de gestión de paquetes.
4to. En Linux, Git puede ser instalado a través del gestor de paquetes de la distribución específica que estés utilizando (apt, yum, pacman, etc.).

- Configuración inicial:

Después de la instalación, es importante configurar Git con tu nombre de usuario y dirección de correo electrónico.

Esto se puede hacer usando el comando **git config**, por ejemplo:
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"

La opción **--global** indica que esta configuración se aplica globalmente a todos los repositorios en tu sistema.

- Estados de Archivos:

1ro. **Untracked (No rastreado)**: Archivos que existen en el directorio de trabajo pero aún no han sido añadidos al área de preparación (staging area). Git no rastrea cambios en estos archivos.
2do. **Unmodified (No modificado)**: Archivos que no han experimentado cambios desde la última confirmación (commit). Estos archivos están bajo seguimiento de Git y no se han modificado desde la última confirmación.
3ro. **Modified (Modificado)**: Archivos que han sido modificados desde la última confirmación, pero aún no se han añadido al área de preparación. Git rastrea estos cambios, pero aún no se han confirmado.
4to. **Staged (Preparado)**: Archivos que han sido añadidos al área de preparación (staging area) mediante el comando git add. Estos archivos están listos para ser confirmados en el próximo **commit**.

- Creación de Repositorio:

Utilizamos el comando **git init** en el directorio de nuestro proyecto para iniciar un repositorio Git localmente. Esto crea un nuevo subdirectorio llamado .**git** que contiene todos los archivos necesarios para el repositorio.

- Comandos de Git Básicos:

git init: Este comando se utiliza para inicializar un nuevo repositorio Git en un directorio. Crea un nuevo subdirectorio llamado .git, que contiene todos los archivos necesarios del repositorio.
git add: El comando git add se utiliza para agregar cambios al área de preparación (staging area) en Git. Esto prepara los archivos modificados para ser confirmados en el repositorio.

git commit: Después de agregar cambios con git add, se utiliza git commit para confirmar los cambios en el repositorio. Cada confirmación crea un nuevo punto en la historia del proyecto.
git config: Este comando se utiliza para configurar variables de entorno específicas de Git, como el nombre de usuario, la dirección de correo electrónico, etc. Esto es útil para personalizar la configuración de Git en un proyecto específico.

- Ignorar y Borrar Archivos:

En Git, es fundamental tener control sobre qué archivos y directorios se incluyen en el repositorio. Para ello, podemos utilizar el archivo .**gitignore** para especificar patrones de nombres de archivos o directorios que deben ser ignorados por Git.

Ejemplo de un archivo **.gitignore**:

```
# Ignorar archivos de respaldo de editor
*~
# Ignorar archivos de compilación
*.o
*.class
# Ignorar directorios de dependencias
/node_modules /vendor
```

Para borrar archivos del repositorio, utilizamos el comando **git rm**:
git rm [nombre de archivo]

3.3. ¿Para qué sirve GIT?

Git es una herramienta que se utiliza para gestionar el control de versiones en proyectos de software. Permite a los desarrolladores llevar un seguimiento detallado de los cambios realizados en los archivos, facilitando la colaboración en equipo y la gestión de múltiples versiones del mismo proyecto de manera simultánea. Con Git, es posible crear diferentes ramas del código para desarrollar nuevas funcionalidades o corregir errores, sin interferir con la versión principal del proyecto.

Entre las principales ventajas de Git destacan su capacidad para trabajar de forma distribuida, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto, lo que mejora la seguridad y flexibilidad. Además, ofrece una forma eficiente de colaborar, ya que permite a varios miembros del equipo trabajar en paralelo sin que se solapen sus cambios.

Una desventaja potencial de Git es que puede tener una curva de aprendizaje pronunciada para quienes no están familiarizados con conceptos de control de versiones, lo que puede llevar a errores si no se utiliza correctamente. Además, en proyectos muy grandes con muchos desarrolladores, la resolución de conflictos entre diferentes cambios puede llegar a ser compleja.

HEAD: es el punto de referencia de tu posición actual en el historial de commits de tu repositorio. Te indica dónde te encuentras y te permite moverte entre commits y ramas.

3.4. GIT Config

- **git config --global user.name “[nombre]”**:

Configuración global del nombre de usuario:

- **git config --global user.email “[correo]”**:

Configuración global del correo electrónico, el email debe coincidir con el mismo correo utilizado en cuentas de repositorios remotos.

- **git config --global core.editor “code --wait”**:

Configuración global del editor de código (IDE): **code** para VSC.

- **git config --global -e**:

Abre en el IDE el archivo de configuración: con **--wait** anterior, espera a que el IDE se cierre para volver a habilitar la consola.

- **git config --core.autocrlf false/input**:

Configura cuestiones técnicas de ingreso de texto, como saltos de línea y el formato de salto de línea de linux: **false** para Windows, **input** para macOS.

3.5. Git init

- **git init**:

Inicializa un repositorio local vacío: crea un archivo oculto (**ls -a**). Crea una carpeta con los archivos necesarios para crear el control de versiones. En VSC la solapa “Source Control”. En terminal, **Git Bash**.

3.6. GIT add, rm, status, restore

- **git add [archivo]**:

Cambia el estado de un archivo a Added o agregado, pasando a la etapa de Staged. Es un estado previo al commit para su posterior push.

- **git add .**:

Cambia el estado de todos los archivos a agregado, pasa todos los archivos a Staged.

- **git status**:

Muestra el estado de los archivos o del trabajo en general.

- **git rm --cached [archivo]**:

git rm por sí solo, elimina un archivo tanto de Stage como del directorio de trabajo, es decir, elimina el archivo del proyecto y lo marca para que se elimine del próximo commit.
Con la flag ***--cached***, elimina el archivo de Stage, pero lo deja en el directorio de trabajo, sigue existiendo el cambio, es decir, solo sale de Staged.

- **git restore [archivo]**:

Restaura el archivo a la última versión, ya sea en Stage o la última versión que esté registrada en el repositorio.

Un archivo puede estar Staged y Modified al mismo tiempo. Es decir, al agregar un archivo a Staged, puede ser Modificado y aparecerá como Modificado y Staged al mismo tiempo. Tendrá ambos estados hasta que se agregue a Staged, o se restaure.

3.7. GIT commit, log

- **git commit -m “[mensaje]”**:

El “commitean” los cambios, pasan a ser parte del repositorio actual. Todos los cambios van acompañados por un mensaje descriptivo de los cambios realizados. Está listo para ser incluido en el código final luego de hacer un push.

- **git log**:

Historial de cambios, aparecen absolutamente todos los cambios. **:q** para salir del log.

- **git log --online**:

Hace un listado del historial de cambios en una sola línea. Cada *log* tiene un ID que hay que considerar si se requieren hacer cambios.

3.8. Gitignore

Es un archivo **.gitignore**. Es un archivo de configuración propio de GIT que permite incluir dentro del mismo archivos y carpetas a los cuales no queremos y no necesitamos que se les haga un seguimiento (trackeo).
Por ejemplo:

- **.env**: archivo utilizado para variables de entorno.
- **node_modules/**: carpeta específica de entornos nodejs.
- ***.jpg**: excluye todos las imágenes con extensión jpg.
- ***.mp4**: excluye todos los videos con extensión mp4.

.gitignore es un archivo que debe ser trackeado y commiteado.

3.9. GIT diff

- **git status -s**:

Versión comprimida de git status, se ven los archivos modificados o no trackeados en una sola línea.

- **git diff**:

Luego de hacer un commit, hace un listado y detalle de los cambios en los archivos entre Modified o Untracked y el código original o en stage. Se debe salir con ***:q***.

3.10. Branches

- Ramas

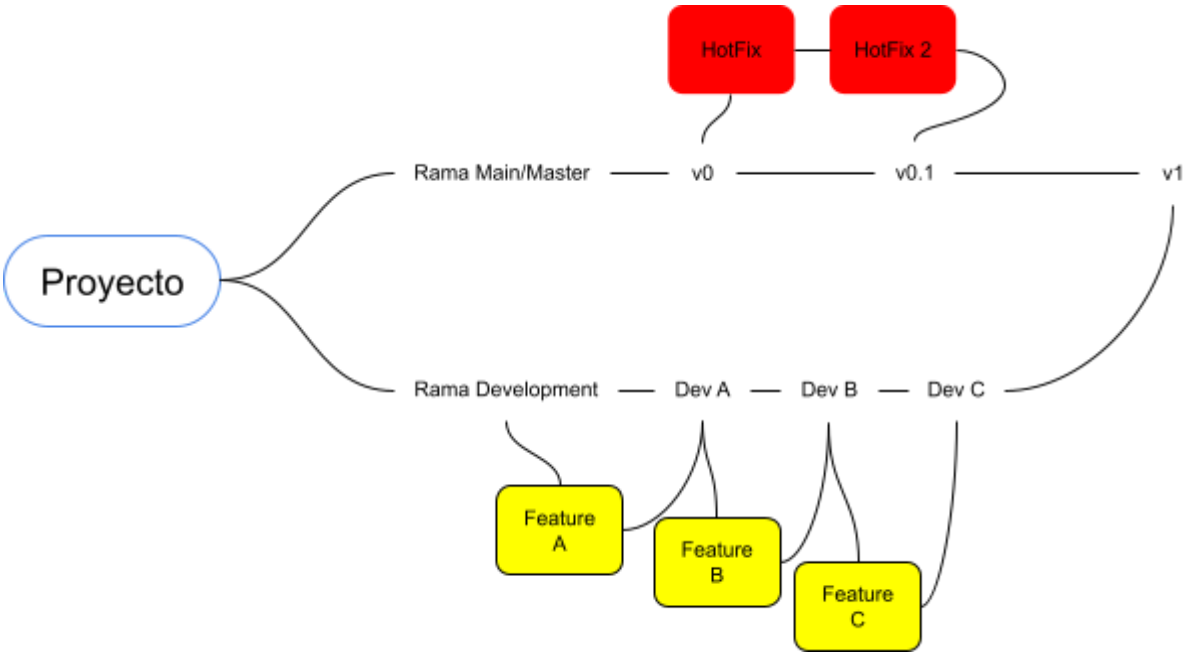
Las ramas en Git son versiones paralelas de un proyecto que permiten trabajar en nuevas características o correcciones sin afectar la rama principal (generalmente denominada “master” o “main”).

- ¿Por qué usar ramas?

Facilitan el desarrollo simultáneo de múltiples características o correcciones por parte de diferentes colaboradores.

Permiten experimentar con nuevas ideas sin afectar la estabilidad del proyecto principal.

Facilitan la organización y gestión de versiones en proyectos complejos.



- Principales Conceptos:

01. **Rama principal (master/main)**: Representa la versión estable y funcional del proyecto.
02. **Ramas de características (feature branches)**: Se utilizan para desarrollar nuevas funcionalidades de manera independiente.
03. **Ramas de corrección (bugfix branches)**: Se emplean para solucionar errores o problemas específicos.
04. **Ramas de versión (release branches**): Preparan el proyecto para un lanzamiento o versión específica.
05. **Ramas de desarrollo (development branches)**: Utilizadas para integrar y probar nuevas características antes de fusionarlas con la rama principal.

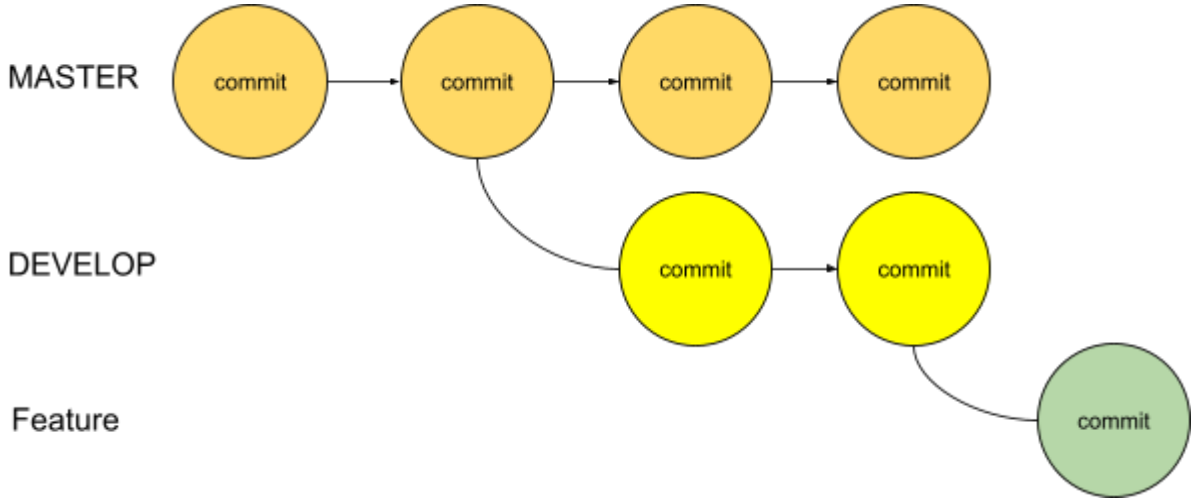
3.11. GIT branch, checkout

- **git branch**:

Provee un listado de las ramas disponibles en el repositorio, indicando la rama actual.

- **git checkout -b [nombre]**:

Crea una rama con el nombre asignado con los últimos cambios y se sitúa en la rama creada.



3.12. GIT Switch, merge

- **git switch [nombre]**:

Se cambia a la rama especificada en el nombre, al igual que ***checkout***. ***Checkout*** sirve para cambiar de rama como para cambiar a un commit específico o restaurar archivos, y ***switch*** se usa exclusivamente para cambiar de ramas.

- **git merge [nombre]**:

Sirve para fusionar ramas. Estando parado en una rama, se traen los cambios y commits de la rama especificada en el nombre a la rama donde se está parado.

3.13. GIT Graph

Extensión de VSC para ver gráficamente la estructura de ramas y commits de un repositorio GIT.

4. GitHub

4.1. ¿Qué es GitHub?

- Diferencias entre Git y Github:

En el contexto del control de versiones, es esencial comprender la distinción entre Git y GitHub. Mientras que Git es un sistema de control de versiones distribuido, GitHub es una plataforma de alojamiento de repositorios Git en la nube. Git, permite el seguimiento de cambios en archivos a lo largo del tiempo, facilitando la colaboración y el seguimiento de versiones en proyectos de software.

Por otro lado, GitHub actúa como un servicio de alojamiento para proyectos que utilizan Git como su sistema de control de versiones. Además de alojar repositorios, GitHub proporciona herramientas adicionales para la gestión colaborativa de proyectos, como seguimiento de problemas, solicitudes de extracción, integración continua y despliegue (CI/CD), entre otros.

En resumen, Git es el motor que impulsa el control de versiones local, mientras que GitHub agrega una capa social y colaborativa al permitir que los desarrolladores compartan, colaboren y contribuyan a proyectos de software de manera distribuida a través de la nube. Esta distinción es crucial para comprender cómo se interrelacionan Git y GitHub y cómo cada uno contribuye al proceso de desarrollo de software.

- Gestión de Ramas en GitHub:

Creación y Eliminación de Ramas:

- Permite crear nuevas ramas directamente desde la interfaz web de GitHub.
- La creación de ramas facilita el trabajo en paralelo en diferentes funcionalidades o correcciones.
- Las ramas pueden eliminarse una vez que ya no sean necesarias, manteniendo la limpieza del repositorio.

Fusionar Ramas mediante Solicitudes de Extracción (Pull Requests):

- Las solicitudes de extracción son mecanismo para fusionar cambios de una rama a otra.
- Facilitan la revisión del código por parte de otros colaboradores antes de fusionar los cambios.
- Permiten discutir y comentar los cambios propuestos antes de su integración.

Resolución de Conflictos durante la Fusión de Ramas:

- Los conflictos pueden surgir cuando los cambios en una rama entran en conflicto con los de otra.
- GitHub proporciona herramientas para resolver conflictos de una manera visual y colaborativa.
- Los colaboradores pueden trabajar juntos para resolver los conflictos antes de completar la fusión.

Visualización y Seguimiento del Historial de Cambios en cada Rama:

- GitHub ofrece una interfaz gráfica para visualizar el historial de cambios en cada rama.
- Permite revisar quién realizó cada cambio, cuando se realizó y los comentarios asociados.
- Facilita el seguimiento del progreso del desarrollo en diferentes ramas y versiones del proyecto.

4.2. Crear una cuenta en GitHub

https://github.com/

4.3. Crear un repositorio externo en GitHub

New en GitHub para crear un repositorio nuevo. Se le asigna un nombre, una descripción, público o privado y demás configuraciones.
Se crea un repositorio vacío con instrucciones para enlazar un repositorio local con el repositorio externo.

- **git remote add origin [url]**:

Enlaza el repositorio local con la url especificada que coincide con el repositorio remoto.

- **git push -u origin [rama]**:

“Empuja” el código local a una rama remota, que suele ser ***main*** o ***master***.

4.4. Git clone

- **git clone [url]**:

Copia un repositorio externo en local, lo “clona”, trayendo todo el código disponible y los datos del repositorio, ramas, commits, etc.

- **git push**:

Luego de agregar a Stage y Commitear cambios, se puede “empujar” el código local al repositorio remoto utilizando **git push**.

4.5. Git fetch, pull, push

- **git pull**:

Trae cambios remotos al entorno local. Descarga los cambios del repositorio remoto y los integra automáticamente.

- **git fetch**:

Trae la información remota al entorno local, como ramas y commits. Descarga los cambios del repositorio remoto sin integrarlos.

- **git push**:

Envía la información y cambios desde el entorno local al entorno remoto.

4.6. Borrado de ramas

- **git branch -d [nombre]**:

Borra la rama con el nombre especificado. Si no se pushea al entorno remoto, no hay problema, el problema está si la rama está en remoto y se borra localmente. Borrando una rama remota de forma local, se la puede recuperar.

- **git push origin [nombre]**:

Pushea las ramas especificadas en el nombre.

- **git push origin -d [nombre]**:

Borra de manera remota la rama especificada en el nombre.

Existe una inconsistencia: si se borra una rama remota y no local, localmente no se actualiza ni con **pull** ni con **fetch**. Para ello se usa:

- **git fetch --prune**:

Limpia todo lo que no está más, trayendo información más actualizada. Luego se puede borrar localmente para que localmente todo coincida con el repositorio remoto.

4.7. Comandos GIT

Comando	Descripción
git init	Inicializa un repositorio Git en un directorio local
git clone <URL>	Clona un repositorio Git desde una URL remota
git add <archivo>	Agrega cambios de un archivo al área de preparación
git add .	Agrega todos los cambios al área de preparación
git commit -m “Mensaje”	Guarda los cambios en el repositorio con un mensaje
git status	Muestra el estado actual del repositorio
git push	Envía los cambios locales al repositorio remoto
git pull	Obtiene cambios desde el repositorio remoto y los fusiona localmente
git branch	Lista todas las ramas del repositorio
git branch <nombre>	Crea una nueva rama con el nombre especificado
git checkout <rama>	Cambia a la rama especificada
git merge <rama>	Fusiona una rama específica con la rama actual
git log	Muestra el historial de confirmaciones
git reset <archivo>	Selecciona los cambios de un archivo en el area de preparacion
git reset --hard	Restablece el estado del repositorio a una confirmación específica

git stash	Guarda los cambios locales en una pila temprana
git stash pop	Recupera los cambios almacenados en la pila temporal
git remote -v	Muestra las URL de los repositorios remotos
git remote add <nombre> <URL>	Agrega un nuevo repositorio remoto con el nombre y la URL especificados
git rm <archivo>	Elimina un archivo del control de versiones
git mv <archivo1> <archivo2>	Cambia el nombre de un archivo en el control de versiones

5. Checkpoints de contenidos
   - ¿Qué innovación tecnológica se considera el primer programa de computadora, diseñado por Ada Lovelace para la máquina analítica?

Un algoritmo para calcular números de Bernoulli.

- ¿Cuál es la principal diferencia entre hardware y software?

El hardware se refiere a los componentes físicos de una computadora, mientras que el software son los programas que se ejecutan en ella.

- ¿Qué comando se usaría en una terminal para listar los archivos y directorios?

ls.

- ¿Qué comando de GIT se utiliza para crear un nuevo repositorio?

git init.

- ¿Cuál es la diferencia entre los comandos ***git fetch*** y ***git pull*** en GIT?

git fetch descarga los cambios del repositorio remoto sin integrarlos, mientras que git pull también los integra automáticamente.