**Resumo sistemas operacionais**

Sistemas operacionais (SO) são programas de software fundamentais que gerenciam o hardware do computador e fornecem serviços para outros programas de software. Eles atuam como intermediários entre o usuário e o hardware, garantindo que diferentes aplicações possam funcionar simultaneamente e de forma eficiente.

**Tipos de sistemas operacionais**

**Windows**: Desenvolvido pela Microsoft, é o SO mais amplamente utilizado em desktops e laptops.

**macOS**: Desenvolvido pela Apple, é o SO para computadores Macintosh, conhecido por sua interface intuitiva e integração com outros produtos Apple.

**Linux**: Existem várias distribuições, como Ubuntu, Fedora e Debian, que oferecem diferentes interfaces e funcionalidades, e são conhecidos por sua flexibilidade e código aberto.

**Android**: Desenvolvido pelo Google, é o SO mais amplamente utilizado em smartphones e tablets.

**iOS**: Desenvolvido pela Apple, é o SO usado em iPhones e iPads, conhecido por sua integração com o ecossistema Apple e segurança.

**Sistemas operacionais – Histórico**

1ª geração - 1945/1955

° Computadores a válvula

° Linguagem de máquina

° Ausência de sistema operacional

2ª geração – 1955/1965

° Computadores com transistores

° Linguagem Assembly

° S.O do tipo lote

° Fila de execuções de programas

3ª geração - 1965/1980

° Circuitos integrados

° Sistemas OS/ 360 (IBM)

4ª Geração – 1980/1990

° circuito integrado de larga escala

° Surgimento dos DOS e WINDOWS

5ª Geração – 1990/ATUAL

° Difusão da internet

° suporte a TCP/IP

° Suporte a gráfico para jogos

° IOT

° Nuvem

**°** Computação ubígua

**Características dos Sistemas Operacionais Monotarefa**

**Execução Simples**:

**Processamento Único**: Apenas um único processo ou tarefa pode ser executado em qualquer momento. Não há suporte para multitarefa ou execução simultânea de múltiplos processos.

**Simplicidade**: Devido à sua simplicidade, esses sistemas são fáceis de projetar e implementar.

**Exemplo:** MS-DOS

**Características dos Sistemas Operacionais Multitarefa**

**Execução Simultânea**:

**Gerenciamento de Múltiplos Processos**: Permite a execução simultânea de vários processos ou aplicativos, proporcionando uma experiência de uso mais fluida e eficiente.

**Troca de Contexto**: Utiliza técnicas de troca de contexto para alternar entre processos, dando a impressão de que todos os processos estão sendo executados ao mesmo tempo.

**Exemplo:** Linux, OSX, Windows

**GIT**

**Para que serve o GIT ?**

O Git é um sistema que rastreia e gerencia mudanças em arquivos, principalmente código-fonte, ao longo do tempo. Ele permite que desenvolvedores acompanhem o histórico de alterações e revertam a versões anteriores se necessário.

**Principais Funcionalidades do Git**

**Rastreamento de Mudanças**:

**Commits**: Registra mudanças no repositório em pontos específicos no tempo. Cada commit inclui uma mensagem descritiva, o que facilita o acompanhamento das alterações e a colaboração em equipe.

**Branching e Merging**:

**Branches (Ramificações)**: Permite criar linhas de desenvolvimento paralelas, facilitando o trabalho em novas funcionalidades ou correções sem afetar a linha principal de desenvolvimento.

**Merging (Mesclagem)**: Integra alterações de diferentes branches de volta ao branch principal, permitindo a combinação de diferentes linhas de desenvolvimento.

**Trabalho Distribuído**:

**Repositórios Locais e Remotos**: Cada desenvolvedor tem uma cópia completa do repositório em sua máquina local, o que permite trabalhar offline e sincronizar as alterações com um repositório remoto quando estiver online.

**Resolução de Conflitos**:

**Merge Conflicts**: Quando duas mudanças conflitantes são feitas, o Git ajuda a identificar e resolver esses conflitos durante o processo de mesclagem.

**Histórico e Reversão**:

**Histórico Completo**: Mantém um histórico completo de todas as alterações feitas no repositório, o que facilita o rastreamento e a reversão de mudanças, se necessário.

**Colaboração**:

**Pull Requests**: Facilita a revisão e a integração de mudanças propostas por diferentes desenvolvedores, promovendo uma colaboração eficiente dentro de equipes.

**O que é o GitHub?**

**Plataforma de Hospedagem de Código**: GitHub permite armazenar e gerenciar repositórios Git na nuvem, facilitando o acesso remoto e a colaboração em projetos de software.

**Principais Funcionalidades do GitHub**

**Hospedagem de Repositórios**:

**Repositórios Git**: Armazena repositórios Git na nuvem, permitindo que desenvolvedores acessem e gerenciem seus projetos de qualquer lugar.

**Colaboração**:

**Pull Requests**: Permite que desenvolvedores proponham mudanças ao código, que podem ser revisadas e discutidas antes de serem integradas ao projeto principal.

**Issues**: Facilita o rastreamento de bugs, tarefas e discussões relacionadas ao projeto.

**Gerenciamento de Projetos**:

**Projetos e Painéis**: Ferramentas para organizar tarefas e acompanhar o progresso do projeto usando quadros Kanban e outras visualizações de gerenciamento de projetos.

**Comunidade e Redes Sociais**:

**GitHub Pages**: Permite criar e hospedar sites diretamente a partir de repositórios GitHub.

**Discussões e Gists**: Facilita a colaboração e a troca de ideias através de fóruns e anotações rápidas.

**Comandos básicos para o Prompt de comando**

**Criar diretório (pasta)**

**° Comando:** md

**Apagar diretório (pasta)**

° **Comando:** rd

Acessar diretório

° **comando:** cd

**Listar diretório (pasta)**

° **Comando:** dir

**Excluir arquivos**

° **Comando:** del

**Copiar arquivos**

° **Comando:** copy

**Mover arquivos**

° **Comando:** move

**Comandos GIT**

**Git – configurações**

**Nome:** git config --global user.name “ “

**Email:** git config –global user.email “ “

**Git – Iniciar um respositório**

**Comando:** git init

**Comandos depois de iniciar um repositório e adicionar arquivos**

git status

git add

git commit -m “comentário”

git remote add origin “url do repositório”

git remote -v

git push -u origin master