APRESENTAÇÃO
DE APOIO

PROGRAMAÇÃO PARA WEB

Andrea Konzen - Aula 01

PUCRS online uol edtech

Professores

ANDREA KONZEN
Professora Convidada

Formada em Ciência da Computação, com mestrado em Ciência da Computação na área de Inteligência Artificial pela (PUCRS) e doutorado em Informática na Educação na área de Inteligência Artificial com aplicação em Sistemas Educacionais pela (UFRGS). Além disso, possui pós-doutorado na área de Machine Learning em projetos voltados para exploração autônoma com múltiplos robôs. Atua como professora adjunta na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) e como coordenadora de projetos na área de Inteligência Artificial na Foreducation EdTech. É secretária regional da Sociedade Brasileira de Computação do Estado (SBC), pesquisadora de projetos na área de Inteligência Artificial com ênfase em Sistemas Inteligentes e Aprendizagem de Máquina aplicados em áreas como Saúde, Educação e Agricultura, com o propósito de reverter benefícios concretos e significativos para a sociedade.

LUIS FERNANDO PLANELLA GONZALEZ
Professor PUCRS

Doutor Ciências da Computação (PUCRS, 2018). Desenvolvedor e arquiteto Java com experiência profissional desde 1999, certificado pela Sun como programador e desenvolvedor de componentes web na plataforma Java. Entusiasta de software livre.

Ementa da disciplina

Estudo do desenvolvimento de aplicações com HTML, CSS e JavaScript. Estudo sobre Document Object Model (DOM). Utilização de forms em aplicações WEB. Desenvolvimento de aplicações responsivas e acessíveis.

Parte 1 da disciplina: Aula 1 e Aula 2

O que vamos ver?

1 - Internet e Protocolos de Transporte e Aplicação

2 - Conceitos de Front end e Back end

3 - Conceitos de SPA, MPA e PWA

4 - Projeto para Desenvolvimento Web

5 - Conceitos e aplicação de HTML

6 - Conceitos e aplicação de CSS

7 - Conceitos e aplicação de JavaScript

1 - Internet e Protocolos de Transporte e Aplicação

Breve Histórico

Breve histórico

década de 1960

Departamento de Defesa dos Estados Unidos criou a **ARPANET (Advanced Research Projects Agency Network)** para conectar computadores de diferentes universidades e centros de pesquisa em todo o país.

Breve histórico

1970, a ARPANET **se expandiu** para outros países, como Reino Unido e Noruega, e foi dividida em redes menores, conhecidas como "sub-redes".

1983, a ARPANET adotou o **protocolo TCP/IP**, que se tornou o padrão para a comunicação entre computadores na Internet.

Breve histórico

anos 1990, a Internet se tornou mais acessível ao público em geral, com o surgimento de **provedores de serviços de Internet** (ISPs) comerciais e a popularização dos navegadores da web.

1991 Tim Berners-Lee (conhecido como pai da web), criou a **linguagem HTML** - CERN (Organização Europeia para Pesquisa Nuclear) na Suíça. Sua equipe criou o primeiro cliente web – **World Wide Web, WWW**.

Breve histórico

anos 90 surgiram os primeiros **provedores Webmail, salas de bate-papo, fóruns** e, o primeiro **serviço de hospedagem de sites** (com ferramentas gratuitas para criação de páginas na internet)

anos 2000, a Internet continuou a crescer em popularidade e importância, **tornando-se um meio fundamental para comunicação, comércio, entretenimento e pesquisa em todo o mundo**

Breve histórico

A evolução da Internet é dividida em três fases:

Web 1.0 (estática)

• a produção de conteúdo ficava a cargo de provedores da internet
• a internet ainda copiava os modelos de distribuição de informações como o rádio e a televisão
• havia muito consumo de informação, mas **a criação de conteúdo figurava na mão de poucos, com pouco espaço para uma criação aberta**

Breve histórico

A evolução da Internet é dividida em três fases:

Web 2.0 (interativa)

- iniciou o processo de produção de conteúdo
- o público deixou de ser somente leitor e para ser o maior responsável pela criação de conteúdo online
- **surgiram os maiores nomes da internet, como o YouTube, a Wikipédia, o Google e as redes sociais**
- o volume de dados gerados diariamente deu um salto muito grande

Breve histórico

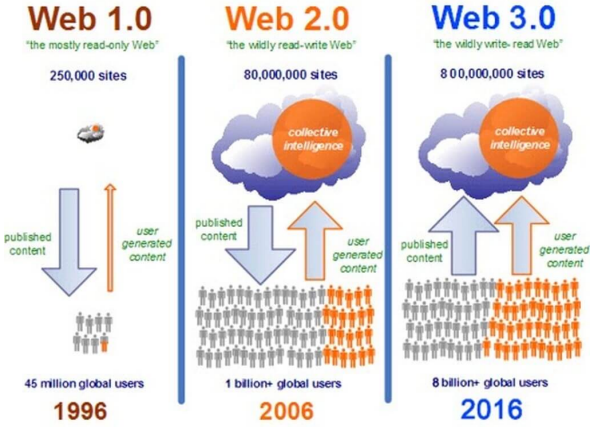
A evolução da Internet é dividida em três fases:

Web 3.0 (interativa inteligente)

- uso da **inteligência artificial**
- criação de **experiência personalizada e interativa**
- a **segurança de dados**
- centralização excessiva do poder digital
- **criptografia**, proporcionando aos usuários controle sobre os próprios dados

Breve histórico

Um resumo...



Arquitetura Web e características das Aplicações Web

Arquitetura Web

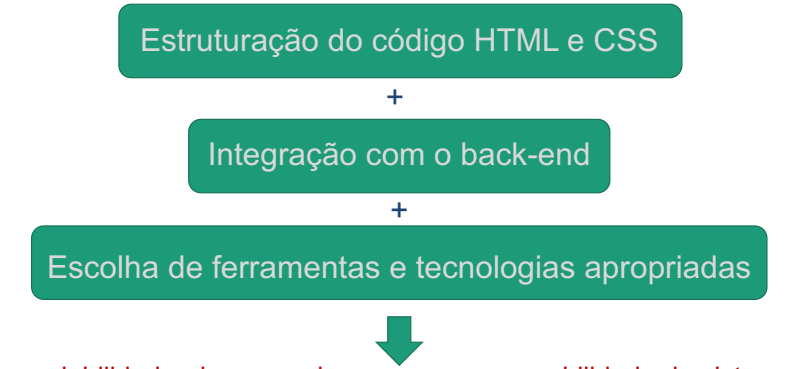
Conjunto de **padrões**, **princípios, técnicas** e **boas práticas** utilizados para **projetar e desenvolver sistemas e aplicações web**.

Arquitetura Web

Responsável por definir:

- a estrutura
- os componentes da aplicação
- as regras
- os protocolos de comunicação entre esses componentes

Arquitetura Web



Arquitetura Web

Características importantes:

- **Escalabilidade**: ser capaz de lidar com grande volume de tráfego
- **Flexibilidade**: ser flexível o suficiente para acomodar mudanças e atualizações futuras

Arquitetura Web

Características importantes (cont.):

- **Segurança**: deve ser segura para proteger os dados dos usuários e a integridade da aplicação (mecanismos de autenticação, autorização e criptografia de dados)

Arquitetura Web

Características importantes (cont.):

- **Performance**: deve ser otimizada para oferecer um desempenho rápido e eficiente (uso de cache, otimização de consultas ao banco de dados, otimização de imagens e arquivos estáticos)

Arquiteturas mais comuns

- **Arquitetura de três camadas (Cliente-servidor)**
- **Arquitetura orientada a serviços (SOA)**
- **Arquitetura baseada em microsserviços**

Arquiteturas de três camadas (Cliente-servidor)

Divide a aplicação em três camadas:

- **camada de apresentação (front-end)**
- **camada de negócios (back-end)**
- **camada de dados (banco de dados)**

Cada camada é responsável por uma parte específica da aplicação, permitindo melhor definição de responsabilidades.

Arquiteturas de três camadas (Cliente-servidor)

Camada de apresentação (front-end)

- responsável por apresentar a interface do usuário e interagir com o usuário
- construída com tecnologias como HTML, CSS e JavaScript
- solicita informações da camada de negócios e exibe os resultados ao usuário

Arquiteturas de três camadas (Cliente-servidor)

Camada de negócios (back-end)

- responsável pela lógica de negócios da aplicação
- recebe solicitações da camada de apresentação, processa as informações e retorna uma resposta
- construída com uma linguagem de programação do lado do servidor (PHP, Python, Java) e se comunica com o banco de dados por meio da camada de dados.

Arquiteturas de três camadas (Cliente-servidor)

Camada de dados (banco de dados)

- responsável por armazenar e recuperar dados da aplicação
- construída usando um sistema de gerenciamento de banco de dados (MySQL ou Oracle)
- se comunica com a camada de negócios para fornecer os dados necessários para a aplicação

Arquitetura Orientada a Serviços
(SOA - Service-Oriented Architecture)

Divide a aplicação em **serviços independentes**, que podem ser acessados por outros serviços ou aplicativos. **Cada serviço é responsável por uma tarefa específica** e pode ser atualizado ou substituído sem afetar outros serviços.

Arquitetura Orientada a Serviços
(SOA - Service-Oriented Architecture)

Características relevantes

- serviço que pode ser acessado por outros componentes ou aplicações por meio de interfaces bem definidas
- independentes de qualquer plataforma específica, linguagem de programação ou SO (integração em diferentes sistemas heterogêneos)

Arquitetura Orientada a Serviços
(SOA - Service-Oriented Architecture)

Características relevantes (cont.)

- enfatiza a integração e a interoperabilidade entre diferentes sistemas e aplicações (**soluções mais flexíveis, escaláveis e economicamente viáveis**)
- complexa e exige esforço significativo de design e implementação

Arquitetura baseada em microsserviços

Essa arquitetura é uma variação da SOA, em que a aplicação é dividida em **microsserviços independentes que se comunicam entre si por meio de APIs**. Cada microsserviço é responsável por uma funcionalidade específica e pode ser escalado independentemente.

Arquitetura baseada em microsserviços

Características relevantes

- cada microsserviço se comunica com outros microsserviços por meio de interfaces bem definidas e padronizadas
- permite que diferentes equipes de desenvolvimento trabalhem em diferentes serviços independentes (escalabilidade e flexibilidade)

Arquitetura baseada em microsserviços

Características relevantes (cont.)

- microserviços podem ser facilmente substituídos ou atualizados sem afetar o restante do sistema
- permite uma maior tolerância a falhas
- permite que cada microserviço seja desenvolvido com a linguagem de programação e tecnologia mais adequadas para a funcionalidade específica

Arquitetura baseada em microsserviços

Características relevantes (cont.)

- complexidade na orquestração e gerenciamento dos serviços, para garantir a consistência dos dados em diferentes serviços e a necessidade de uma infraestrutura de monitoramento e gerenciamento mais robusta

Aplicações Web

Uma aplicação Web pode ser caracterizada por três dimensões do projeto:

- **estrutural (ou conceitual)**
- **navegacional**
- **apresentação**

Aplicações Web

A dimensão estrutural define a organização das informações a serem tratadas pela aplicação e os seus relacionamentos

A dimensão navegacional define como as informações serão acessadas através da aplicação

A dimensão de apresentação define como as informações e o acesso a essas informações serão apresentados ao usuário da aplicação.

Aplicações Web

Características gerais

- Acesso através do navegador
- Armazenamento de dados em nuvem/remoto
- Interatividade
- Multiplataforma

Aplicações Web

Características gerais (cont.)

- Atualizações automáticas (sem necessidade do usuário)
- Flexibilidade (personalização/adaptação)
- Segurança
- Conexão com outras aplicações

Aplicações Web

Aplicações Web x Sistemas Tradicionais

A engenharia de um sistema para Web envolve, além dos aspectos da engenharia de sistemas convencionais, aspectos relevantes apenas para esse tipo de sistema.

Aplicações Web

Aplicações Web x Web site

Uma aplicação Web enfatiza principalmente os **aspectos relacionados à aplicabilidade e funcionalidade** enquanto um **Web Site tem ênfase na apresentação, aparência e navegação**.

Visão geral de Protocolos

Protocolos – Visão geral

Como estamos trabalhando com Sistemas Web, especificamente o desenvolvimento, precisamos estudar os principais protocolos de rede que estão associados diretamente ao nosso objetivo.

Protocolos – Visão geral

Protocolos de rede:

todas as atividades de comunicação na Internet são governadas por protocolos

Protocolos definem o **formato**, **ordem das msgs enviadas e recebidas** pelas entidades da rede, e **ações tomadas quando da transmissão ou recepção dessas mensagens**

Conceito de Protocolos

Um protocolo é uma convenção que controla e possibilita **conexão, comunicação, transferência de dados** entre dois sistemas computacionais.

Pode ser definido como:

as regras que governam a sintaxe, semântica e sincronização da comunicação.

Conceito de Protocolos

Responsáveis por coletar os dados transmitidos pela rede e dividí-los em pequenos pedaços, que são chamados de **pacotes**.

Cada pacote carrega em si informações de **endereçamento de origem e destino**.

Elementos-chave de Protocolos

Elementos-chave que definem os protocolos de rede:

- **sintaxe:** representa o formato dos dados e a ordem pela qual eles são apresentados;
- **semântica:** refere-se ao significado de cada conjunto sintático que dá sentido à mensagem enviada;
- **timing:** define uma velocidade aceitável de transmissão dos pacotes.

Protocolos de Internet

Existem vários **protocolos de internet** importantes:

- TCP/IP
- HTTP/HTTPS
- FTP
- DNS
- SMTP/POP3/IMAP
- ICMP

Protocolos de Internet

TCP/IP - ***Protocolo de Controle de Transmissão/Protocolo Internet***

- Protocolo padrão da Internet, usado para transmitir dados entre dispositivos em redes de computadores.
- Divide os dados em pacotes e os envia de um dispositivo para outro, verificando se todos os pacotes chegaram corretamente.

Protocolos de Internet

HTTP/HTTPS - ***Protocolo de Transferência de Hipertexto***

- Usado para transferir dados da Web entre um servidor e um navegador.
- Também usado para solicitar e receber páginas da Web e outros recursos da Internet.

Protocolos de Internet

FTP - *Protocolo de Transferência de Arquivos*

- Usado para transferir arquivos entre dispositivos em uma rede de computadores.
- Amplamente usado para transferir arquivos grandes, como imagens, vídeos e arquivos de áudio.

Protocolos de Internet

DNS - *Sistema de Nomes de Domínio*

- Usado para traduzir nomes de domínio em endereços IP.
- Permite que os usuários da Internet acessem sites e recursos da Web usando nomes de domínio em vez de endereços IP numéricos.

Protocolos de Internet

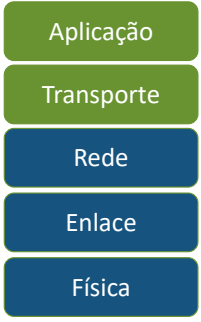
SMTP/POP3/IMAP - ***Protocolo de Transferência de Correio Simples***

- Usado para enviar e receber e-mails entre dispositivos em uma rede de computadores.
- Permite que os usuários enviem e recebam e-mails usando programas de e-mail.

Protocolos de Rede

Divididos em **cinco camadas** para tornar a comunicação mais eficiente e confiável:

- **Aplicação**
- **Transporte**
- Rede
- Enlace
- Física



Protocolos de Transporte e Protocolos de Aplicação

Transporte

Os protocolos utilizados nesta camada são **TCP e UDP.**

Protocolos de Transporte

Responsável por fornecer uma **comunicação confiável e eficiente entre aplicativos em diferentes dispositivos**.

Garante que os **dados sejam transmitidos sem erros**, em **ordem correta** e com uma **velocidade adequada**.

Protocolo TCP

Transmission Control Protocol

É um protocolo base da internet, complementado pelo ***Internet Protocol*** **(IP)**.

É um tipo bastante versátil e robusto de protocolo, fazendo com que seja adequado para grandes redes, como a **rede mundial de computadores** (**World Wide Web**).

Protocolo TCP

A função principal do **TCP** é verificar se os **dados** que circulam entre os dispositivos de uma rede **são enviados de forma correta e na sequência apropriada**.

Protocolo TCP

Verifica a **confiabilidade dos dados**, garantindo que eles sejam enviados na ordem correta e verificando possíveis erros nos pacotes de dados que fluem entre os usuários e dispositivos conectados à rede.

Protocolo TCP

Aplicações práticas

Navegação na web

quando você acessa um site através do seu navegador, o TCP é usado para estabelecer uma conexão com o servidor web e garantir que todos os dados sejam transmitidos com segurança e sem erros.

Protocolo TCP

Aplicações práticas

E-mail

ao enviar e receber e-mails, o TCP é usado para estabelecer uma conexão entre o servidor de e-mail e o cliente de e-mail, garantindo que todos os dados sejam transmitidos com segurança e sem erros.

Protocolo TCP

Aplicações práticas

Transferência de arquivos

ao fazer o download de um arquivo da internet ou enviar um arquivo através de um serviço de compartilhamento de arquivos, o TCP é usado para garantir que todos os dados sejam transmitidos com segurança e sem erros.

Protocolo TCP

Aplicações práticas

Conexão remota

ao fazer uma conexão remota com outro computador, o TCP é usado para garantir que todos os dados sejam transmitidos com segurança e sem erros.

Protocolo TCP

Aplicações práticas

Streaming de vídeo

ao assistir a um vídeo online, o TCP é usado para garantir que todos os dados sejam transmitidos com segurança e sem interrupções.

Protocolo UDP

Tipo mais simples na comparação com o TCP.

Permite que a aplicação envie um datagrama em um pacote IPv4 ou um IPv6 para determinado destino, **sem garantias** de que o **pacote de dados chegue ao destino da forma correta**.

Protocolo UDP

IPv4 (Internet Protocol version 4) **e IPv6** (Internet Protocol version 6) permitem que os dispositivos se comuniquem na internet.

IPv4 é o protocolo de internet mais antigo e ainda amplamente utilizado atualmente.
IPv6 é a versão mais recente do protocolo de internet.

Protocolo UDP

Esse tipo de protocolo, **não é confiável e não oferece o nível de proteção e verificação dos dados** transmitidos entre os usuários e dispositivos de uma determinada rede.

Protocolo UDP

UDP é um registro indivisível, voltado à transmissão de bytes sem um começo e sem um fim determinado. É utilizado quando a **velocidade da transmissão de dados é priorizada em detrimento da segurança desses dados**.

Protocolo UDP

Envia os dados **sem verificar a confiabilidade**, garantindo garantindo que sejam enviados com alta velocidade, sem interrupções.

Protocolo UDP

Aplicações práticas

Jogos online

o UDP é usado para transmitir dados em tempo real entre o servidor e o cliente, garantindo uma experiência de jogo mais rápida e responsiva.

Protocolo UDP

Aplicações práticas

Transmissão de vídeo ao vivo

ao transmitir vídeo ao vivo pela internet, o UDP é usado para garantir que o fluxo de dados seja transmitido em tempo real, sem atrasos ou interrupções.

Protocolo UDP

Aplicações práticas

Aplicativos de voz sobre IP

em aplicativos de voz sobre IP, como o Skype ou o Zoom, o UDP é usado para transmitir a voz em tempo real, garantindo uma comunicação mais fluida e sem interrupções.

Protocolo UDP

Aplicações práticas

Aplicativos de IoT

em dispositivos conectados à internet, como sensores, dispositivos de rastreamento ou de monitoramento, o UDP é usado para transmitir dados em tempo real, como informações de localização ou de temperatura.

Protocolo UDP

Aplicações práticas

Serviços de DNS (Domain Name System)

o UDP é usado para enviar consultas de resolução de nomes de domínio (DNS) a servidores DNS, permitindo que o usuário acesse sites da internet com facilidade.

Protocolo TPC x UDP

Principais diferenças:

- O protocolo TCP preza pela confiabilidade agregando em seu *header bits* de controle de fluxo e recebimento, o UDP dispensa esses bits de controle.
- O TCP é orientado para a conexão através do seu ”reconhecimento” e o UDP não, visto que *não é criada uma conexão para ele, só o envio direto de dados.*

Protocolo TPC x UDP

	TCP	UDP
Confiabilidade	Alto	Baixo
Rapidez	Baixo	Alto
Método de transferência	Pacotes entregues em sequência	Pacotes entregues em um fluxo
Detecção e correção de erros	Sim	Não
Reconhecimento	Sim	Apenas a verificação

Protocolo TPC x UDP

Vantagens do protocolo UDP

- O protocolo UDP é **rápido** em relação ao TCP
- Não causa sobrecarga
- Pode oferecer suporte à comunicação de um ponto a outro ponto e de um ponto a vários outros pontos.

A desvantagem do UDP você já sabe: não existe comunicação entre o emissor e o receptor (se os pacotes não chegarem completos, eles não são enviados)

Protocolo TPC x UDP

Como saber qual o melhor protocolo?

Quem define é o desenvolvedor, pois cada um desses protocolos tem suas vantagens e desvantagens (como já mencionado) de acordo com determinadas funções e objetivos.

Protocolo TPC x UDP

Como saber qual o melhor protocolo?

1. Temos aplicações que exigem a **robustez do TCP**, outras precisam de **velocidade e dinamismo do UDP**.
2. Se sua rede tiver um **bom nível de segurança** e você souber evitar vários comportamentos de risco, **qualquer um desses protocolos** será bem utilizado pelo seu sistema.

Protocolo TPC x UDP

Como saber qual o melhor protocolo?

Como base no que você viu nos slides anteriores, será capaz de responder a pergunta a seguir e definir o protocolo a ser utilizado quando do desenvolvimento da sua aplicação Web:

Qual a finalidade da minha solução?

Protocolos de Aplicação

E composto por uma variedade de protocolos: **HTTP, FTP, SMTP, DNS, SSH, TLS**, entre outros.

Protocolo de Aplicação

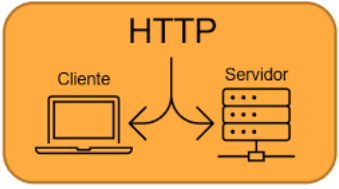
É responsável pela **comunicação entre aplicativos em diferentes dispositivos de rede**.

Define **formatos de mensagens** e **regras de comunicação** que os aplicativos devem seguir para se comunicarem uns com os outros.

Protocolo de Aplicação

Responsável também por **estabelecer uma comunicação confiável entre os aplicativos de origem e destino**, garantindo que os dados sejam entregues na ordem correta e sem erros.

Protocolo HTTP



Hypertext Transfer Protocol

Usado para **transferência de dados na web**.

Define como os dados devem ser transmitidos e formatados para permitir a comunicação entre os clientes (navegadores) e os servidores da web.

Protocolo HTTP

Hypertext Transfer Protocol

É um protocolo de solicitação-resposta, onde o **cliente envia uma solicitação para o servidor e este responde com uma mensagem** contendo os dados solicitados.

Protocolo HTTPs

Hyper Text Transfer Protocol Secure

Constitui um aplicativo do HTTP sobre uma **camada adicional de segurança** (usando TLS) e permite a transmissão de dados em uma **conexão criptografada através de certificados digitais.**

Protocolo HTTPs

Como funciona?

HTTPS **criptografa uma ligação através de um certificado digital** SSL (*Secure Sockets Layer*), usando HTTP sobre SSL, permitindo uma **ligação segura** entre cliente e servidor.

Protocolo HTTPs

Como funciona? (cont.)

SSL é um **sistema de certificado digital** que usa chaves para criptografar os dados entre cliente e servidor. Este certificado garante ao visitante que o site que está acessando é seguro e que os seus dados estão protegidos.

Protocolo HTTPs

Como implementar?

1) **Primeiramente deve-se adquirir um Certificado SSL.** Esses certificados criam conexões seguras e criptografadas entre um navegador de Internet e um servidor.

2) **Depois todas as ligações ao site serão feitas de forma segura por HTTPS.**

Protocolos HTTP x HTTPs

HTTP

http://www.dominio.com
password: rtu123

Password não encriptada
Hacker vê "rtu123".

HTTPS

https://www.dominio.com
password: rtu123

Password encriptada
Hacker vê "o09lpt"

Protocolo HTTP

Aplicações práticas

Navegação na web

Protocolo principal utilizado pelos navegadores web para acessar páginas da web e outros recursos na Internet.

Protocolo HTTP

Aplicações práticas

Aplicações móveis

Muitas aplicações móveis utilizam o HTTP para se comunicar com servidores na Internet.

Ex.: aplicativos de redes sociais, de comércio eletrônico e de notícias, entre outros.

Protocolo HTTP

Aplicações práticas

APIs (*Application Programming Interfaces*)

Muitas APIs permitem que diferentes sistemas e aplicativos se comuniquem também utilizando o HTTP como protocolo de comunicação.

Ex.: enviar e receber dados entre diferentes sistemas.

Protocolo HTTP

Aplicações práticas

Transferência de arquivos

Não é o protocolo mais eficiente para transferência de arquivos grandes, mas é frequentemente utilizado para transferir arquivos menores.

Ex.: imagens e arquivos de áudio menores.

Protocolo HTTP

Aplicações práticas

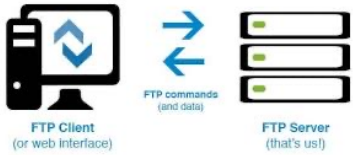
Streaming de vídeo

Permite que os usuários acessem o conteúdo de vídeo em seus navegadores ou aplicativos sem a necessidade de instalar software adicional.

HTTP é utilizado juntamente com outros protocolos para permitir a transmissão de vídeo em tempo real.

Protocolo FTP

File Transfer Protocol



Utilizado para transferência de arquivos entre computadores em uma rede (**upload e o download de arquivos entre um cliente e um servidor FTP**).

Define como os dados devem ser transferidos, incluindo autenticação, permissões de acesso e controle de conexão.

Protocolo FTP

Aplicações práticas

Hospedagem de sites

Muitas empresas de hospedagem de sites utilizam o FTP para permitir que seus clientes enviem arquivos para seus servidores web.

Ex.: os clientes podem usar clientes FTP para enviar seus arquivos.

Protocolo FTP

Aplicações práticas

Compartilhamento de arquivos

FTP é frequentemente utilizado para compartilhar arquivos entre diferentes usuários em uma rede. Isso inclui redes locais e a Internet.

Ex.: um usuário pode disponibilizar um arquivo para download através de um servidor FTP.

Protocolo FTP

Aplicações práticas

Backup de dados

FTP é muitas vezes utilizado para realizar backups de dados importantes.

Ex.: os usuários podem enviar seus arquivos para um servidor FTP remoto para garantir que seus dados estejam seguros e acessíveis de qualquer lugar.

Protocolo FTP

Aplicações práticas

Transferência de arquivos grandes:

FTP é uma opção viável para transferência de arquivos grandes, já que é projetado para lidar com arquivos de tamanho variado. Ele permite a transferência de arquivos de forma rápida e eficiente, mesmo em conexões com largura de banda limitada.

Protocolo FTP

Aplicações práticas

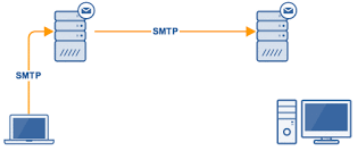
Atualização de software

Muitas empresas de software utilizam o FTP para distribuir atualizações de software para seus usuários.

Ex.: usuários podem baixar as atualizações diretamente do servidor FTP da empresa e instalar as atualizações em seus computadores.

Protocolo SMTP

Simple Mail Transfer Protocol



Utilizado para **transferência (roteamento e pela entrega) de e-mails** na Internet.

Define como as mensagens de e-mail devem ser transmitidas de um servidor de e-mail para outro.

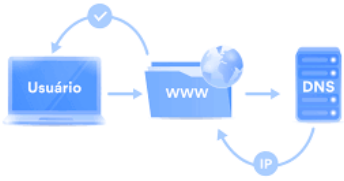
Protocolo SMTP

Aplicações práticas

- **Envio de e-mails pessoais**
- **Comunicação empresarial**
- **Marketing por e-mail**
- **Notificações automáticas**
- **Segurança**

Protocolo DNS

Domain Name System



Utilizado para **“tradução” (resolver) de nomes de domínio em endereços IP**.

É responsável por **localizar o servidor que hospeda o site associado ao nome de domínio**.

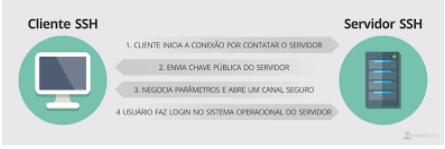
Protocolo DNS

Aplicações práticas

- **Navegação na web**
- **E-mail**
- **Aplicações em nuvem** (balancear carga)
- **Segurança** (verificar a autenticidade dos servidores)

Protocolo SSH

Secure Schell



Utilizado para **comunicação segura entre computadores em uma rede** (utiliza criptografia para garantir que as informações não possam ser interceptadas por terceiros). Permite que os usuários se conectem a um servidor de forma segura e **realizem ações em um terminal remoto**.

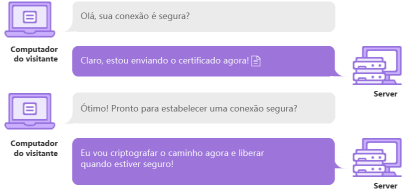
Protocolo SSH

Aplicações práticas

- **Acesso remoto seguro**
- **Transferência de arquivos segura**
- **Backup remoto**
- **Gerenciamento de servidores**
- **Acesso a bancos de dados**

Protocolo TLS

Transport Layer Security



Utilizado para **criptografar a comunicação entre dois computadores em uma rede**.

Garante a **segurança das transações realizadas em sites**, como compras online e acesso a serviços bancários.

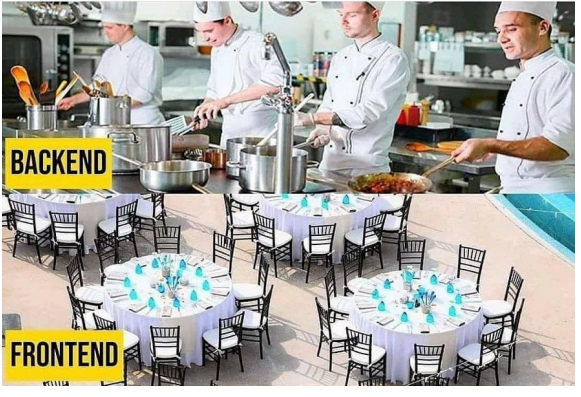
Protocolo SSH

Aplicações práticas

- **Comércio eletrônico**
- **E-mail seguro**
- **VPN (*Virtual Private Network*)**
- **Autenticação do servidor**
- **Aplicativos móveis**
- **Redes sociais**

2 – Front End e Back End

Front end & Back end



Front end

Front end é a parte de um sistema we ou aplicativo que o usuário interage diretamente.

É responsável pela apresentação visual e interação com o usuário.

Front-end

A implementação do front-end é o processo de desenvolvimento da interface do usuário, incluindo o design, a codificação e a integração de todos os elementos que fazem parte dessa interface.

Front-end

O desenvolvimento front-end envolve tanto habilidades técnicas quanto criativas.

É necessário entender as tecnologias e ferramentas usadas na implementação, mas também é importante ter um senso estético e de design para criar interfaces atraentes e intuitivas para os usuários.

Front-end

Cuidado para não confundir o Front-end com o Design!!

O profissional que define as cores, as formas e o estilo de um site ou de um aplicativo é o designer.

O programador Front-end será o responsável em codificar e dar vida a essa arte criada pelo designer.

Front-end

A implementação do Front-end envolve várias tecnologias e ferramentas:

- HTML
- CSS
- JavaScript e;
- frameworks front-end

HTML

O HTML é a linguagem usada para **criar a estrutura básica do sistema web ou aplicativo**

As *tags* do HTML indicam ao navegador como renderizar o conteúdo

HTML

Algumas das tags mais comuns incluem:

<html>, <head>, <title>, <body>, <header>, <footer>, <nav>, <section>, <article>, , <p>, <img>, <a> e <form>

Vale ressaltar que é importante criar uma estrutura semântica e acessível para o seu projeto

CSS

O CSS é a linguagem usada para **definir o estilo visual do sistema web ou aplicativo**

É importante criar um estilo consistente e coerente para a interface, usando cores, fontes, layout e animações apropriadas para o conteúdo e objetivo do projeto

CSS

É importante também garantir que a interface seja responsiva, adaptando-se a diferentes tamanhos de tela e dispositivos

Para isso, pode ser útil usar técnicas como *media queries* e *grid systems*

CSS

Exemplo básico de CSS para adicionar estilo a um elemento HTML

Veja o seguinte código HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de CSS</title>
</head>
<body>
    <h1>Bem-vindo ao meu site!</h1>
    <p>Este é um exemplo de como adicionar estilo a um
elemento HTML usando CSS.</p>
</body>
</html>
```

CSS

Para adicionar estilo ao nosso HTML, vamos criar um arquivo CSS separado e vincular ao nosso HTML usando a tag link no cabeçalho (head) do documento. O arquivo CSS pode ser chamado de "style.css" e deve ser salvo na mesma pasta que o arquivo HTML

```
h1 {
  color: blue;
  font-size: 36px;
}

p {
  font-size: 20px;
  line-height: 1.5;
  margin-top: 20px;
}
```

CSS

Depois de salvar o arquivo CSS, precisamos vincular ao nosso arquivo HTML. Para isso, adicionamos a seguinte linha ao cabeçalho do nosso HTML:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

CSS

Agora, o título será azul e terá um tamanho de fonte de 36 pixels, enquanto o parágrafo terá um tamanho de fonte de 20 pixels e um espaço de linha de 1.5, além de ter um espaço de 20 pixels acima do elemento. O arquivo HTML ficaria assim:

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <h1>Bem-vindo ao meu site!</h1>
    <p>Este é um exemplo de como adicionar estilo a um
elemento HTML usando CSS.</p>
</body>
</html>
```

Java Script

O JavaScript é uma linguagem de programação usada para criar interatividade na interface do usuário

Algumas das funcionalidades que podem ser criadas com JavaScript incluem animações, validações de formulários e interações com o usuário

Java Script

Exemplo - código JavaScript que pode ser adicionado a uma página HTML:

```
<!DOCTYPE html>
<html>
<head>
	<title>Exemplo de JavaScript</title>
</head>
<body>
	<h1>Clique no botão abaixo</h1>
	<button onclick="mudarTexto()">Clique aqui</button>
	<p id="mensagem">A mensagem aparecerá aqui</p>

	<script>
		function mudarTexto() {
			document.getElementById("mensagem").innerHTML = "Olá, mundo!";
		}
	</script>
</body>
</html>
```

Java Script

Temos um botão que, quando clicado, altera o texto de um elemento HTML para "Olá, mundo:

- ***onclick="mudarTexto()"*:** adiciona um evento ao botão para executar a função mudarTexto() quando o botão é clicado.
- ***function mudarTexto() { ... }*:** define a função mudarTexto() que será chamada quando o botão é clicado.
- ***document.getElementById("mensagem").innerHTML*:** seleciona o elemento com o ID "mensagem" e altera seu conteúdo interno para o novo texto.

Dessa forma, quando o usuário clica no botão, o texto do elemento com o ID "mensagem" é alterado para "Olá, mundo!".

Ferramentas e Frameworks

Existem várias ferramentas e frameworks que podem ser usados para facilitar o desenvolvimento front-end

O React é um exemplo de framework front-end que permite criar interfaces complexas e reativas usando componentes reutilizáveis

Ferramentas e Frameworks

O Vue.js é outro framework front-end que permite criar interfaces flexíveis e escaláveis usando componentes e diretivas

O Sass é uma ferramenta de pré-processamento de CSS que permite escrever estilos mais eficientes e modulares

Back-end

Back-end é a parte do sistema web ou aplicativo que fica no servidor e cuida da **lógica de negócio, o armazenamento de dados e a segurança**

A implementação do Back-end envolve várias tecnologias e ferramentas, incluindo **linguagens de programação, bancos de dados e frameworks.**

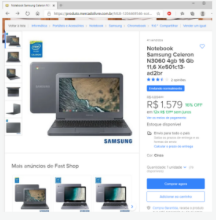
Back-end

Back-end é o conjunto de elementos que estão por trás da interface de uma aplicação:

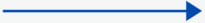
- seus sistemas
- banco de dados
- toda parte de segurança de dados
- envio e recebimento de informações
- armazenamento
- entre outros.

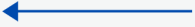
Back-end - funcionamento

Página web
(Front-end)



O **Front-end** envia o pedido de compra para o **Back-end**





O **Back-end** valida a compra do usuário e envia a resposta para o **Front-end**

Back-end

Valor da compra: 50,00
Limite disponível: 100,00
Compra: Aprovada

Linguagens de Programação

Existem várias linguagens de programação que podem ser usadas para criar o Back-end, incluindo:

- PHP
- Python
- Ruby
- Node.js
- entre outras

Linguagens de Programação

Cada linguagem tem suas próprias vantagens e desvantagens, e a escolha da linguagem dependerá das necessidades específicas do projeto

É importante conhecer bem a linguagem escolhida e suas bibliotecas e frameworks associados

Banco de Dados

Os bancos de dados são usados para armazenar e gerenciar os dados do aplicativo

Existem vários tipos de bancos de dados, incluindo bancos de dados relacionais (como MySQL e PostgreSQL) e bancos de dados NoSQL

É importante escolher o tipo de banco de dados adequado para o projeto e criar uma estrutura de banco de dados eficiente e escalável

Frameworks Beck-end

Os frameworks Back-end são ferramentas que ajudam a criar rapidamente aplicativos Web robustos e escaláveis

Existem vários frameworks Back-end populares: Laravel (para PHP), Django (para Python), Ruby on Rails (para Ruby) e Express.js (para Node.js)

Cada framework tem suas próprias convenções e recursos, e é importante escolher o framework certo para o projeto

Segurança

É importante implementar medidas de segurança adequadas, como **autenticação e autorização de usuários, criptografia de dados e prevenção de ataques**

APIs e Back-end

Quando falamos de back-end é importante relacionarmos o conceito de APIs

As **APIs** são usadas para **conectar diferentes sistemas e permitir a troca de informações entre eles de maneira fácil e eficiente**

APIs e Back-end

- Uma API é uma interface de programação de aplicativos que permite a comunicação entre diferentes sistemas

- No contexto do back-end, as APIs são usadas para expor as funcionalidades e dados de um sistema para uso externo

APIs e Back-end

As APIs fornecem uma maneira padronizada para que os desenvolvedores possam acessar as informações e funcionalidades de um sistema, sem precisar conhecer os detalhes internos do sistema

As APIs são implementadas como *endpoints* HTTP ou HTTPs que são acessados por clientes externos, como aplicativos ou outros sistemas

APIs e Back-end

Os *endpoints* são mapeados para funções e rotas específicas no código do back-end, que processam as solicitações e enviam as respostas correspondentes

As APIs podem ser projetadas para permitir diferentes tipos de solicitações para acessar e modificar dados

Tipos de APIs

Alguns tipos de APIs:

- **APIs da Web:** usadas para acessar informações e serviços da Web
- **APIs de serviços:** usadas para acessar informações e funcionalidades de sistemas específicos, como bancos de dados e sistemas de gerenciamento de conteúdo

Tipos de APIs (cont.)

- **APIs de plataforma:** usadas para acessar as funcionalidades de uma plataforma específica, como a API do Facebook ou a API do Twitter

Cada tipo de API tem suas próprias especificidades e requisitos, e é importante escolher a API certa para o projeto

Segurança nas APIs

- As APIs devem ser projetadas com autenticação e autorização adequadas para proteger os dados e as funcionalidades do sistema

- É importante limitar o número de solicitações feitas para evitar sobrecarregar o servidor da API e proteger contra ataques de negação de serviço (DDoS)

Exemplo de aplicação – *front-end/back-end*

Você está acessando uma loja virtual

Ao efetuar a compra no cartão de crédito, você aguarda a mensagem do pagamento.

Após alguns instantes, recebe a informação de que a compra foi feita e deve aguardar o pedido chegar no endereço cadastrado.

Exemplo de aplicação (cont.)

Todas as **informações visíveis** da página da loja virtual, como: a disposição das imagens e o design, são **desenvolvidas pelo front-end**.

Ao efetuar o pagamento do cartão, **informações são enviadas do navegador para a empresa** de cartão de crédito para autenticar e validar a compra, cumprindo as regras de segurança (back-end)

Exemplo de aplicação (cont.)

3 – Tipos de Sistemas Web: SPA, MPA e PWA

Tipos de Sistemas Web

Quais são os tipos de de sistemas web?

- **SPA (Single Page Application)**
- **MPA (Multi Page Application)**
- **Progressive WebApps (PWA)**

SPA (Single Page Application)

SPA (Single Page Application)

é um modelo de desenvolvimento de aplicações web que consiste em **uma única página web que se atualiza dinamicamente**, permitindo a construção de uma interface de usuário rica e responsiva.

SPA (Single Page Application)

Ao invés de carregar múltiplas páginas, o **SPA carrega todo o conteúdo necessário em uma única página** (experiência mais rápida e fluída para o usuário).

SPA (Single Page Application)

O carregamento do conteúdo é feito de forma assíncrona, através de chamadas AJAX (Asynchronous JavaScript and XML), o que evita a necessidade de recarregar a página inteira a cada interação do usuário.

É possível criar interfaces mais interativas e dinâmicas, que respondem de forma rápida às ações do usuário.

SPA (Single Page Application)

O SPA é amplamente utilizado em aplicações web modernas, como **plataformas de e-commerce, aplicativos de redes sociais, ferramentas de produtividade e jogos online**, etc.

Seu uso permite que os desenvolvedores criem aplicações web mais rápidas, escaláveis e fáceis de manter, proporcionando uma melhor experiência para os usuários.

SPA (Single Page Application)

Por que escolher SPA (Single Page Application)?

1.Experiência de usuário mais fluida e rápida:
O SPA evitando a necessidade de carregar várias páginas.

2. Redução da carga do servidor:
Com o SPA, apenas o conteúdo necessário é carregado de forma assíncrona, através de chamadas AJAX, evitando que a página inteira seja recarregada, reduzindo a carga no servidor.

SPA (Single Page Application)

Por que escolher SPA (Single Page Application)?

3. Melhor escalabilidade e manutenibilidade:
Como o SPA é construído em torno de um único código base, ele é mais fácil de ser mantido e escalado em comparação com uma aplicação web de várias páginas. Além disso, o uso de frameworks e bibliotecas, como o Angular e o React, simplifica o processo de desenvolvimento e reduz a complexidade do código.

SPA (Single Page Application)

Por que escolher SPA (Single Page Application)?

4. Maior interatividade e personalização:
É possível criar interfaces de usuário mais interativas e personalizadas, permitindo que os usuários realizem ações em tempo real sem precisar recarregar a página, proporcionando uma experiência mais agradável e personalizada para o usuário.

SPA (Single Page Application)

Benefícios do uso:

1.Velocidade:
Os SPAs são conhecidos por sua velocidade, pois só carregam uma vez e depois só atualizam as partes necessárias da página.

2. Melhor Experiência:
Por causa da velocidade e da natureza interativa dos SPAs, os usuários têm uma experiência melhor e mais responsiva.

SPA (Single Page Application)

Benefícios do uso:

3. Desenvolvimento mais fácil:
É necessário lidar com apenas uma página. O código é organizado e escalável, o que ajuda a manter o projeto gerenciável.

4. Arquitetura modular:
Os SPAs são projetados para serem altamente modularizados. Cada seção da página pode ser dividida em seus próprios componentes, trabalhar independentemente.

SPA (Single Page Application)

Benefícios do uso:

5. Acesso off-line:
Como a maioria dos SPAs utiliza recursos como caches, eles podem continuar funcionando mesmo sem conexão com a internet.

6. Maior segurança:
Os SPAs são mais seguros porque a maioria da lógica do aplicativo é executada no cliente e não no servidor, reduzindo a vulnerabilidade a ataques de segurança.

SPA (Single Page Application)

Benefícios do uso:

7. Facilidade de manutenção:
Como os SPAs têm uma arquitetura modular, a manutenção é muito mais fácil. Pequenas atualizações podem ser feitas sem afetar todo o aplicativo, tornando a manutenção do aplicativo mais fácil e econômica.

SPA (Single Page Application)

Como funciona:

Se concentra em **uma única página HTML** e usa **JavaScript e AJAX** para atualizar dinamicamente o conteúdo da página sem precisar recarregá-la completamente.

Quando um usuário navega em uma SPA, todo o código JavaScript é carregado no navegador na primeira visita e, em seguida, as interações do usuário são tratadas no lado do cliente, sem precisar fazer solicitações adicionais ao servidor.

SPA (Single Page Application)

Como funciona:

O SPA usa uma rotação de URL para renderizar o conteúdo na página sem carregar uma nova página.
Quando o usuário clica em um link, o JavaScript intercepta o evento de clique e usa o histórico do navegador para alterar a URL na barra de endereços do navegador, mas a página não é atualizada.

O JavaScript busca os dados necessários do servidor usando uma chamada AJAX e atualiza apenas a parte da página que precisa ser atualizada. Isso fornece uma experiência de usuário mais rápida e responsiva.

SPA (Single Page Application)

Como funciona:

Os SPAs são geralmente construídos com estruturas de aplicativos JavaScript, como React, Angular ou Vue.js, que fornecem funcionalidades para renderizar o conteúdo dinamicamente e gerenciar o estado do aplicativo.

Além disso, os SPAs geralmente usam APIs RESTful para se comunicar com um servidor back-end e buscar dados em tempo real para exibir na página.

SPA (Single Page Application)

Arquitetura:

1.View: camada de apresentação que contém todos os elementos visuais que são exibidos na tela. É responsável por apresentar os dados do modelo para o usuário e responder às interações do usuário.

2.Modelo: camada de dados que contém a lógica de negócios e os dados do aplicativo. É responsável por armazenar, recuperar e atualizar os dados que são exibidos na view.

SPA (Single Page Application)

Arquitetura:

3. Controlador: camada que controla as interações do usuário e atualiza o modelo e a view em resposta a essas interações. Ele atua como um intermediário entre a view e o modelo.

4. Serviços: responsáveis por fazer chamadas ao servidor para buscar dados em tempo real e fornecer recursos adicionais ao aplicativo.

SPA (Single Page Application)

Arquitetura:

5. Roteamento: usado para navegar entre diferentes partes do aplicativo sem precisar carregar uma nova página, feito usando uma biblioteca de roteamento que intercepta as alterações na URL e atualiza a view correspondente.

6. Gerenciador de estado: responsável por manter o estado do aplicativo e garantir que ele seja consistente em toda a aplicação, podendo ser implementado usando uma biblioteca de gerenciamento de estado como o Redux.

SPA (Single Page Application)

Arquitetura:

7. Bibliotecas/frameworks: são usados para fornecer funcionalidades adicionais ao aplicativo, como renderização do lado do servidor, internacionalização, animações e muito mais.

SPA (Single Page Application)

Tecnologias usadas para desenvolvimento:

- HTML
- CSS
- JavaScript
- Frameworks JavaScript
- Bibliotecas JavaScript
- AJAX
- APIs RESTful
- JSON
- entre outros

SPA (Single Page Application)

Exemplo de aplicação: Gmail

Construído com HTML5, CSS3 e JavaScript, e usa a arquitetura de SPA para fornecer uma experiência de usuário rápida e responsiva.

SPA (Single Page Application)

Exemplo de aplicação: Gmail

Ao acessar o Gmail, você é levado a uma única página da web, onde todas as interações acontecem.
Quando você clica em uma mensagem de e-mail, por exemplo, a página não é recarregada, mas a interface do usuário é atualizada dinamicamente para exibir a mensagem selecionada. O Gmail usa AJAX para fazer chamadas assíncronas ao servidor e atualizar a interface do usuário em tempo real.

MPA (Multi Page Application)

MPA (Multi Page Application)

Modelo de desenvolvimento de aplicativos da web no qual **cada página é carregada como uma página separada**, com sua própria URL, e as ações do usuário geralmente exigem o carregamento de uma nova página.

São **várias páginas HTML independentes**, cada uma com sua própria estrutura e conteúdo.

MPA (Multi Page Application)

Cada página da aplicação web geralmente contém sua própria lógica de negócios, scripts JavaScript e folhas de estilo CSS, e é carregada em um novo contexto do navegador, resultando em uma experiência do usuário menos suave e ágil do que a de um SPA.

MPA (Multi Page Application)

No modelo MPA, a comunicação entre as páginas geralmente ocorre por meio de solicitações HTTP, com cada página sendo carregada do servidor em resposta a uma solicitação do usuário.

Pode resultar em tempos de carregamento mais lentos e maior consumo de largura de banda do que um SPA, que geralmente requer apenas uma solicitação HTTP inicial e pode atualizar o conteúdo dinamicamente em resposta às ações do usuário.

MPA (Multi Page Application)

Tecnologias usadas para desenvolvimento:

- HTML
- CSS
- JavaScript
- Frameworks e bibliotecas front-end
- Servidores web
- Banco de dados
- Linguagens de programação back-end

MPA (Multi Page Application)

Vantagens:

- **SEO*-friendly:** MPAs são mais amigáveis aos motores de busca, pois cada página é carregada individualmente, permitindo que os mecanismos de busca rastreiem e indexem cada página separadamente.

- **Manutenção mais fácil:** Como cada página é independente, as atualizações e manutenção podem ser realizadas em uma página sem afetar o funcionamento de outras páginas.

**Otimização de Mecanismos de busca*

MPA (Multi Page Application)

Vantagens:

- **Rápido tempo de carregamento:** MPAs carregam rapidamente porque apenas o conteúdo necessário é carregado em cada página.
- **Boa usabilidade:** Os usuários podem facilmente navegar e entender a estrutura do site, pois cada página tem seu próprio propósito e funcionalidade específica.

MPA (Multi Page Application)

Desvantagens:

- **Transições mais lentas:** A navegação entre páginas pode ser mais lenta em uma MPA do que em uma SPA, pois cada página precisa ser carregada separadamente.

- **Experiência de usuário inconsistente:** Como cada página tem sua própria interface do usuário e funcionalidade, pode haver uma falta de consistência na experiência do usuário.

MPA (Multi Page Application)

Desvantagens:

- **Requisições de servidor:** Cada vez que uma nova página é carregada, uma nova requisição de servidor é feita, o que pode aumentar a carga do servidor.
- **Dificuldade em gerenciar o estado do aplicativo:** O estado do aplicativo pode ser mais difícil de gerenciar em uma MPA do que em uma SPA, pois cada página tem seu próprio estado.

MPA (Multi Page Application)

Arquitetura:

Baseada em um modelo de desenvolvimento clássico do lado do servidor, em que **cada página é carregada a partir do servidor em resposta a uma requisição do cliente**.

Front-end (cliente):
A camada do cliente em uma MPA é responsável por renderizar o HTML, CSS e JavaScript no navegador do usuário. Ele se comunica com o servidor para solicitar e receber as páginas necessárias para exibição no navegador.

MPA (Multi Page Application)

Arquitetura:

Back-end (servidor):
A camada do servidor em uma MPA é responsável por receber as requisições do cliente, processá-las e retornar as páginas HTML correspondentes. O servidor também gerencia o acesso ao banco de dados e fornece funcionalidades como autenticação e autorização.

Banco de dados:
O banco de dados armazena e gerencia os dados do aplicativo, incluindo informações de usuário, conteúdo e configurações do aplicativo. O servidor se comunica com o banco de dados para recuperar e atualizar as informações necessárias.

MPA (Multi Page Application)

Arquitetura:

Camada de middleware:
A camada de middleware é responsável por intermediar a comunicação entre o front-end e o back-end. Ele fornece funcionalidades como validação de entrada, autenticação e autorização, gerenciamento de sessão e cache.

MPA (Multi Page Application)

Exemplo de aplicação: Amazon

Cada página do site é uma página separada, com sua própria URL única, conteúdo e funcionalidade. Quando o usuário clica em um link para navegar para outra página, a página é carregada separadamente do servidor e exibida no navegador do usuário.

Na página inicial, os usuários podem navegar por categorias de produtos, como eletrônicos, roupas, livros e muito mais. Cada categoria tem sua própria página, com sua própria lista de produtos, filtros de pesquisa e outras funcionalidades específicas.

MPA (Multi Page Application)

Estratégias para otimização de performance:

- **Minimizar o tamanho dos recursos:** Reduza o tamanho dos arquivos HTML, CSS e JavaScript, removendo código desnecessário e comentários. Use técnicas de compressão, como Gzip, para reduzir ainda mais o tamanho dos arquivos.

- **Cache de conteúdo:** Use caching para armazenar em cache os recursos estáticos do site, como imagens, CSS e JavaScript. Isso reduz o número de solicitações ao servidor, melhorando o tempo de carregamento das páginas.

MPA (Multi Page Application)

Estratégias para otimização de performance:

- **Otimização de imagens:** Reduza o tamanho das imagens, usando formatos compactados como JPEG e PNG, em vez de formatos pesados, como BMP ou TIFF. Reduza a qualidade das imagens para um nível aceitável sem comprometer a usabilidade do site.

- **Carregamento assíncrono de recursos:** Use técnicas como lazy loading para carregar recursos sob demanda, melhorando o tempo de carregamento da página inicial.

MPA (Multi Page Application)

Estratégias para otimização de performance:

- **Use um CDN (servidor de entrega de conteúdo):** Um CDN ajuda a melhorar o tempo de carregamento das páginas, distribuindo o conteúdo do site em vários servidores em todo o mundo.

- **Minimizar as solicitações HTTP:** Reduza o número de solicitações HTTP para o servidor, combinando arquivos CSS e JavaScript, reduzindo o número de imagens e usando técnicas como sprites de imagem.

MPA (Multi Page Application)

Estratégias para otimização de performance:

- **Otimização de banco de dados:** Use técnicas como indexação e normalização de banco de dados para melhorar o tempo de resposta do banco de dados; ferramentas de monitoramento para identificar consultas lentas e otimizá-las para melhorar o desempenho do banco de dados.

- **Monitoramento de desempenho:** Use ferramentas de monitoramento para acompanhar o desempenho do site, identificar gargalos e corrigir problemas..

MPA (Multi Page Application)

Vantagens das MPAs em relação às SPAs:

- **Melhor SEO:** Como cada página em uma MPA tem sua própria URL, é mais fácil para os motores de busca indexarem e classificarem as páginas individuais.

- **Fácil manutenção:** cada página é uma entidade independente, o que pode facilitar a manutenção e a atualização do site, o que pode tornar a depuração e a correção de bugs mais fáceis.

MPA (Multi Page Application)

Vantagens das MPAs em relação às SPAs:

- **Carregamento mais rápido:** o tempo de carregamento pode ser mais rápido em comparação com um SPA, especialmente para usuários com conexões mais lentas ou dispositivos mais antigos.

- **Compatibilidade com navegadores mais antigos:** Como as MPAs dependem de solicitações HTTP para carregar cada página, elas tendem a ser mais compatíveis com navegadores mais antigos e dispositivos móveis

MPA (Multi Page Application)

Vantagens das MPAs em relação às SPAs:

- **Menor complexidade:** Em geral, as MPAs tendem a ser menos complexas do que as SPAs, o que pode tornar o desenvolvimento mais fácil para equipes menores ou projetos menos complexos.

PWA (Progressive WebApps)

Progressive Web Apps (PWA)

São **aplicativos web que oferecem uma experiência de usuário semelhante à de um aplicativo móvel nativo**, mas que **são executados em um navegador da web**.

Combinação entre um “site” e um aplicativo móvel.

Progressive Web Apps

Combinam o melhor dos dois mundos: a **acessibilidade** e a **facilidade de desenvolvimento** de um sistema web com a **funcionalidade e a capacidade de engajamento** de um aplicativo móvel.

Progressive Web Apps

Princípios que uma aplicação web deve seguir para ser considerada um PWA:

- Poder ser **encontrada em mecanismos de busca**
- Ser **instalável**
- Ser **compartilhável** por meio de uma URL
- **Funcionar offline** ou com conexão instável

Progressive Web Apps

- Usar **aprimoramento progressivo** (permitir que todos acessem o conteúdo básico da aplicação, oferecendo uma versão aprimorada para usuários com dispositivos mais modernos ou conexão melhor)
- **Enviar notificações** quando houver novidades
- Ser **responsiva**, adaptando-se aos diferentes dispositivos, como smartphone, tablet e computador
- Ser **segura**

Progressive Web Apps

Características

Confiabilidade

PWAs são confiáveis e funcionam independentemente da qualidade da conexão de internet ou de outros fatores externos.

Podem armazenar conteúdo em cache para acesso offline, reduzindo a dependência da internet.

São executadas em um ambiente seguro e confiável, pois precisam ser hospedadas em HTTPS.

Progressive Web Apps

Engajamento

PWAs oferecem recursos de engajamento, como notificações push, que permitem que os desenvolvedores se comuniquem com os usuários mesmo quando o aplicativo não está em uso. Os usuários podem optar por receber notificações personalizadas, o que ajuda a melhorar a fidelidade do usuário.

Progressive Web Apps

Navegação suave

As PWAs oferecem uma navegação suave e uma experiência de usuário fluida - os aplicativos são projetados para se adaptar aos diferentes dispositivos e tamanhos de tela.

Utilizam tecnologias como o Web App Manifest para especificar as configurações de exibição do aplicativo e a API Service Worker para gerenciar o cache do aplicativo, o que melhora a velocidade de carregamento e a responsividade do aplicativo.

Progressive Web Apps

Instalação fácil

PWAs são fáceis de instalar em dispositivos móveis, pois não requerem download na loja de aplicativos. Podem ser adicionados diretamente à tela inicial do dispositivo, com um ícone que parece um aplicativo nativo.

Progressive Web Apps

Compatibilidade

PWAs são compatíveis com uma ampla variedade de dispositivos e sistemas operacionais, pois são baseados em tecnologias web padrão, como HTML, CSS e JavaScript. Assim, os desenvolvedores podem criar um único aplicativo que funciona em vários dispositivos.

Progressive Web Apps

Métricas de desempenho

As PWAs permitem que os desenvolvedores monitorem o desempenho do aplicativo usando métricas como o tempo de carregamento, o tempo de resposta e a taxa de rejeição, permitindo que os desenvolvedores ajustem o aplicativo para melhorar a experiência do usuário.

Progressive Web Apps

Benefícios para o desenvolvimento de aplicativos:

Desenvolvimento mais rápido e fácil

PWAs são desenvolvidos com tecnologias como HTML, CSS e JavaScript, o que os torna mais fáceis e rápidos de desenvolver em comparação com aplicativos nativos.

Custos reduzidos

Como PWAs são desenvolvidos com tecnologias web, eles podem ser desenvolvidos com menos recursos do que aplicativos nativos, o que pode reduzir significativamente os custos de desenvolvimento.

Progressive Web Apps

Benefícios para o desenvolvimento de aplicativos:

Fácil manutenção

PWAs são atualizados automaticamente, assim os desenvolvedores podem manter o aplicativo atualizado com facilidade e sem precisar se preocupar com a compatibilidade com diferentes versões de sistema operacional.

Multiplataforma

PWAs são compatíveis com diferentes plataformas e dispositivos, o que significa que os desenvolvedores não precisam criar diferentes versões de um aplicativo para cada plataforma.

Progressive Web Apps

Benefícios para o desenvolvimento de aplicativos:

Melhor experiência do usuário

PWAs são responsivos e oferecem uma experiência de usuário semelhante à de um aplicativo nativo. Também podem ser usados offline.

Fácil distribuição

PWAs podem ser distribuídos facilmente por meio da web, sem precisar passar por lojas de aplicativos ou processos de aprovação.

Progressive Web Apps

Service Workers

Uma das principais tecnologias que tornam os PWAs possíveis.

São scripts em segundo plano que executam funções sem a necessidade de interação do usuário e permitem que PWAs funcionem offline, além de fornecer outras funcionalidades avançadas, como notificações push.

Progressive Web Apps

Service Workers (cont.)

Operam como um intermediário entre o aplicativo web e a rede, interceptando solicitações de rede e permitindo que os aplicativos respondam com conteúdo armazenado em cache, mesmo quando não há conexão com a internet. Quando a conexão com a internet é restabelecida, sincronizam as atualizações com o servidor.

Progressive Web Apps

Service Workers (cont.)

São armazenados em cache pelo navegador e podem ser acionados a qualquer momento, mesmo quando o aplicativo web não está em execução. Assim, podem ser usados para executar ações em segundo plano, como atualizar conteúdo sem a necessidade de uma interação do usuário.

Progressive Web Apps

Service Workers (cont.)

São escritos em JavaScript e precisam ser registrados para serem executados no navegador. Trabalham em um modelo de eventos, com eventos sendo emitidos quando ocorrem solicitações de rede, notificações push e outras ações relevantes.

Progressive Web Apps

Principais aplicações de Service Workers

- Cache de recursos
- Atualizações em segundo plano
- Notificações push
- Gerenciamento de erros
- Sincronização em segundo plano

Progressive Web Apps

O Manifesto da Web em PWA

É um arquivo JSON que contém informações sobre a aplicação web progressiva, como: *nome, ícone, cor de tema, orientação de tela, entre outras.*

Usado para configurar a experiência do usuário ao instalar e utilizar a PWA em dispositivos móveis.

Progressive Web Apps

O Manifesto da Web em PWA (cont.)

É importante para que a PWA possa ser adicionada à tela inicial do dispositivo e se comportar de forma semelhante a um aplicativo nativo, como a possibilidade de funcionar offline e enviar notificações push.

É parte integrante das diretrizes da Web App Manifest da W3C, que busca padronizar o desenvolvimento de aplicativos web progressivos e facilitar a interação com o usuário em diferentes dispositivos.

Progressive Web Apps

Exemplos práticos:

Twitter Lite: permite aos usuários acessar o Twitter em dispositivos móveis com conexões lentas ou limitadas. Ele carrega rapidamente e tem um design simples e responsivo.

Starbucks: permite que os usuários comprem e paguem por bebidas usando seu dispositivo móvel. Também permite que os usuários encontrem lojas próximas, vejam o menu e personalizem suas bebidas.

Progressive Web Apps

Exemplos práticos:

Uber: permite que os usuários solicitem uma viagem, acompanhem o progresso do motorista e paguem pela viagem, tudo dentro do aplicativo.

Pinterest: permite que os usuários salvem e pesquisem ideias de maneira rápida e fácil. Ele também funciona offline, permitindo que os usuários acessem suas ideias salvas mesmo quando não estão conectados à internet.

Progressive Web Apps

Adoção de PWA, exemplos:

Microsoft: A Microsoft está adotando PWAs em seus próprios produtos, como o Microsoft Teams e o Outlook.com. A empresa também criou um conjunto de APIs para permitir que os desenvolvedores criem PWAs para a Microsoft Store.

PUCRS online uol edtech