Roteiro da Apresentação:

Noções do Desenvolvimento Web

Elaborado por **Thiago** (Marcuth).

Revisado por Francisco Jr. e Equipe Técnica.

2024

Introdução

Olá, pessoas, boa noite!

Eu sou Thiago, um estudante de informática e além disso um programador. Atualmente meu foco está no Desenvolvimento Web.

Hoje, irei partilhar com vocês um pouco do que eu sei e outras coisas que eu andei pesquisando sobre o tema. Espero que gostem e aprendam algo com este trabalho.

Peço que permaneçam em silêncio e prestem atenção para que não fiquem perdidos. Se tiverem qualquer dúvida, por favor, levantem a mão para que eu reserve o tempo para ela.

Também quero dizer que ao final de cada tópico haverá algumas questões com um brinde para quem acertar a resposta certa. Funcionará da seguinte forma: eu irei ler a pergunta, as alternativas e após isso quem souber a resposta vai erguer a mão, caso acerte receberá um brinde.

Apenas a ponto de curiosidade, há alguém aqui com experiências em desenvolvimento web?

- **Sim:** *Contar quantos tem, em seguida dizer "*massa!*"
- **Não:** *Dizer "*bom, tá todo mundo nivelado então…*"

Uma breve história da Web

Uma das coisas que eu aprendi nesses anos de estudo é que devemos começar a estudar pela história de algo, para compreender melhor as motivações por trás do surgimento daquilo em questão e por isso começaremos por esta breve história e explicação básica de cada tecnologia.

O nascimento da Web

A Internet como a conhecemos foi diretamente impactada por *Timothy John Berners-Lee*, um físico britânico, cientista da computação e professor do *MIT*. Ele é o criador da *World Wide Web*, tendo feito a primeira proposta para sua criação em 12 de março de 1989.

A base de tudo da Web

Na década de 1990, *Berners-Lee* propôs a criação de algumas tecnologias que foram essenciais para a construção da web como a conhecemos. Sendo elas o *HTML*, o *HTTP* e a *URI*.

HTML

Tim Berners-Lee desenvolveu o *HTML* (HyperText Markup Language) como uma linguagem de marcação para criar documentos que pudessem ser exibidos em navegadores web. A ideia era permitir a criação de documentos com links interativos (hipertexto) que pudessem ser facilmente compartilhados e acessados através da internet.

A primeira especificação formal do *HTML* foi publicada em 1993 pelo *IETF* (Internet Engineering Task Force) como um documento chamado "HyperText Markup Language (HTML)" (RFC 1866). Desde então, o *HTML* passou por várias revisões e atualizações, com novas versões introduzindo recursos adicionais e melhorias na linguagem.

A necessidade de uma linguagem padronizada para criar documentos hipertexto era fundamental para o crescimento e sucesso da *World Wide Web*. O *HTML* permitiu que os criadores de conteúdo da web estruturassem seus documentos de uma maneira que fosse compreensível pelos navegadores web, permitindo, assim, a criação de páginas da web interativas e navegáveis.

HTTP

O *HTTP* (HyperText Transfer Protocol) ou *Protocolo de Transferência de HiperTexto* foi desenvolvido como parte do projeto *World Wide Web*, que buscava criar um sistema de informações distribuídas baseado em hipertexto.

O *HTTP* é um protocolo de comunicação utilizado para transferir dados na *World Wide Web*. Ele permite que os clientes (geralmente navegadores da web) solicitem recursos, como páginas da web, de servidores web e exibam esses recursos em um formato compreensível para os usuários.

O *HTTP* foi projetado para ser simples e flexível, permitindo a transferência de uma variedade de tipos de dados, incluindo texto, imagens, áudio, vídeo e outros recursos da web. Ele opera no modelo de *cliente-servidor*, onde um cliente faz uma solicitação a um servidor e o servidor responde com os dados solicitados.

O surgimento do *HTTP* foi impulsionado pela necessidade de uma maneira padronizada de acessar e compartilhar informações na embrionária *World Wide Web*. Com o tempo, o *HTTP* passou por várias revisões e atualizações para melhorar seu desempenho, segurança e eficiência, com as versões mais recentes sendo HTTP/1.1 e HTTP/2.

URI

A proposta inicial para a *URI* (Uniform Resource Identifier) foi feita em 1994, durante o desenvolvimento do *HTTP/1.0*. A necessidade de um sistema de identificação uniforme surgiu da crescente complexidade da internet e da necessidade de identificar recursos de forma consistente.

A especificação inicial para *URIs* foi publicada em 1994, no *RFC 1630*, intitulado "Universal Resource Identifiers in WWW". Desde então, a tecnologia *URI* tem sido fundamental para a identificação de recursos na web e em outros sistemas distribuídos.

A necessidade de uma forma padronizada de identificar recursos na internet era fundamental para o funcionamento eficaz da *World Wide Web*. Com a proliferação de diferentes tipos de recursos e protocolos, era importante ter um sistema que permitisse aos usuários e aplicativos acessar recursos de forma consistente, independentemente de sua localização ou natureza. *URIs* fornecem uma maneira única e globalmente compreensível

de identificar recursos na internet, o que é fundamental para a interoperabilidade e usabilidade da web.

Outras criações de Berners-Lee

E não para por aí! *Tim Berners-Lee* continuou a criar mais tecnologias como o primeiro navegador web, o primeiro servidor e o primeiro website!

Vamos conhecer mais detalhes sobre cada criação dele

Primeiro navegador web

Em 1990 *Berners-Lee* criou o primeiro navegador web chamado *WorldWideWeb* que mais tarde foi renomeado para *Nexus* para evitar a confusão entre o software (navegador) e a *World Wide Web*.

Além da funcionalidade de navegar por documentos da web, o navegador *WorldWideWeb* também tinha a funcionalidade de editar código *HTML*.

Berners-Lee fez todas essas criações e as executou em um computador *NeXT* no *CERN* (Organização Europeia para a Pesquisa Nuclear).

Primeiro servidor

Não satisfeito, *Berners-Lee* também criou o primeiro servidor web, o *CERN httpd* (abreviação para HyperText Transfer Protocol daemon).

Primeiro website

E para fechar com chave de ouro, *Tim Berners-Lee* fez o primeiro website, construído no *CERN* e foi posto online em 6 de agosto de 1991. “Info.cern.ch” foi o endereço do primeiro website e servidor web da história. O primeiro endereço de página web foi *http://info.cern.ch/hypertext/WWW/TheProject.html*, centrada em informações sobre o projeto *WWW*. Visitantes poderiam aprender mais sobre o hipertexto, detalhes técnicos para a criação de sua própria página web e até mesmo uma explicação sobre como pesquisar na *Web* para obter informações. Não há imagens da tela dessa página original e, em todo caso, alterações foram feitas diariamente com a informação disponível na página *WWW* quando o projeto desenvolveu-se.

Pode-se encontrar uma cópia mais tardia de 1992 no website do *World Wide Web Consortium*. Havia uma explicação sobre o que a *World Wide Web* era e como alguém poderia usar um navegador e configurar um servidor web.

Evolução da Web

A *Web* como a conhecemos não é da mesma forma que antes, ela sofreu atualizações e mudanças significativas na forma de como a consumimos e os recursos oferecidos por ela.

Para dividir esses períodos de grandes mudanças na Web temos os termos *Web 1.0*, *Web 2.0* e *Web 3.0*.

Web 1.0

A *Web 1.0* é a primeira fase da internet. Ela é caracterizada por sites estáticos, que oferecem informações aos usuários, mas não permitem a interação ou colaboração. Em resumo, os usuários eram meros espectadores, sem a capacidade de contribuir com conteúdo.

A tecnologia predominante era o HTML, e os sites projetados para visualização em computadores desktop. O principal objetivo dos sites da *Web 1.0* era fornecer informações.

Web 2.0

A *Web 2.0* é a segunda fase da internet. Ela é caracterizada por uma maior interatividade e colaboração entre os usuários. Os sites se tornaram mais dinâmicos e interativos, permitindo aos usuários contribuir com conteúdo e interagir com outros usuários. As redes sociais e os sites de compartilhamento de conteúdo se tornaram populares. A tecnologia predominante na *Web 2.0* é o *JavaScript*, que permite a criação de sites dinâmicos. Os sites da *Web 2.0* são projetados para serem acessados em dispositivos móveis e desktop.

No início do ano 2000, a internet ganhou recursos importantes, isso permitiu que os usuários não ficassem limitados a apenas visualizar informações, mas pudessem também interagir com as informações, postando comentários ou criando conteúdo dentro de comunidades.

Nesse período, surgiram plataformas de blogs, como o Blogger, comprado pelo *Google* em 2003, mesmo ano em que o *WordPress* nasceu.

Web 3.0

A *Web 3.0* é a *Web* marcada pela descentralização pois, diferentemente da computação em nuvem tradicional da *Web 2.0* ela não é necessariamente feita pelas empresas, contando também com a colaboração dos usuários.

Se isso te fez lembrar de blockchains, saiba que é sobre isso mesmo. Para ficar de melhor entendimento temos alguns exemplos:

- **Torrent:** A tecnologia Torrent exemplifica a descentralização da *Web 3.0*, permitindo que usuários compartilhem e distribuam arquivos diretamente entre si, em vez de depender de servidores centralizados. Isso ilustra a colaboração entre os usuários para compartilhar recursos digitais de forma eficiente e descentralizada, refletindo os princípios fundamentais da *Web 3.0*.

- **Bitcoin:** A tecnologia *Bitcoin* é outra manifestação da descentralização na *Web 3.*0. Ela introduziu a ideia de uma moeda digital descentralizada, baseada em uma rede *peer-to-peer* (*P2P*) e operando sem a necessidade de uma autoridade central, como um banco central. Ao contrário das moedas fiduciárias tradicionais, onde as transações são processadas e verificadas por intermediários centralizados, como bancos, o *Bitcoin* utiliza um livro-razão distribuído chamado blockchain.

Referências

- https://chat.openai.com/c/260ff158-f17f-4c54-a5fd-22603e044c0a
- https://pt.wikipedia.org/wiki/Tim_Berners-Lee
- https://g.co/gemini/share/fa13690319e0
- https://medium.com/mctw-tdw/p%C3%A1ginas-web-uma-breve-hist%C3%B3ria-bc7c432fcac7
- https://en.wikipedia.org/wiki/WorldWideWeb
- https://blog.culte.com.br/evolucao-da-internet-da-web-1-0-a-web-3-0/
- https://youtu.be/WjL--R00GPQ?si=Im6Gvg101iLTNZlp
- https://chat.openai.com/c/13decd0d-9d9a-429f-8aad-cc0ac95bf08d

Questões

1. Quem foi o criador da *World Wide Web*?
    a. Elon Musk
    b. Bjarne Stroustrup
    c. Timothy John Berners-Lee
    d. Mark Zuckerberg

2. Além de criar a *World Wide Web* ele também criou algumas tecnologias. Quais são?
    a. HTTP, URI e HTML
    b. HTML, CSS e JavaScript
    c. HTTP, CSS e HTML
    d. TCP/IP, HTTP e HTML

3. No que a *Web 3.0* se destaca?
    a. Dinamicidade de páginas
    b. Centralização de recursos computacionais
    c. Descentralização de recursos computacionais
    d. Maior interatividade para o usuário

Frontend, Backend & FullStack

Frontend

O que é?

Também conhecido como *Client-side* (ou lado do cliente) o *Frontend* é a parte de uma aplicação web que é executada no navegador, é a parte que a interface do usuário é definida. No *Frontend* podemos fazer coisas como criar elementos para representar e expor funcionalidades de nosso servidor *Backend*.

Não apenas expor funcionalidades do *Backend* mas podemos também desenvolver ferramentas que não necessitam de qualquer tipo de autenticação ou segurança pois tudo o que é enviado para o navegador o usuário se mal intencionado ele pode alterar os códigos para burlar algo. Um bom exemplo disso são os bloqueadores de anúncios, que são scripts que removem elementos *HTML* que são categorizados como anúncios por ele.

No entanto, é importante ressaltar que, embora o *Frontend* possa ser vulnerável a manipulações por parte do usuário, isso não significa que todas as funcionalidades ou medidas de segurança devem ser implementadas exclusivamente no *Backend*. Uma abordagem de segurança completa e eficaz deve considerar tanto o *Frontend* quanto o *Backend*, implementando medidas adequadas em ambas as camadas.

No atual *Frontend* temos 3 linguagens para desenvolver páginas web. Utilizamos elas para definir estrutura, aparência e funcionalidade. Vamos começar a entrar em detalhes pelo *HTML*:

HTML

O *HTML: HyperText Markup Language* (ou em português: *Linguagem de Marcação de HiperTexto*) é o bloco de construção mais básico da web. Ele define o significado e estrutura de uma página web utilizando tags.

O termo "HiperTexto" refere-se aos links que conectam páginas da Web entre si, seja dentro de um único site ou entre sites. Links são um aspecto fundamental da web. Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras pessoas, você se torna um participante ativo na *World Wide Web*.

Marcação? Tags? Que raios é isto?!

O HTML usa "Marcação" para anotar texto, imagem, vídeos e outros conteúdos para exibição em um navegador da Web.

Um elemento *HTML* é separado de outro texto em um documento por "tags", que consistem no nome do elemento entre "<" e ">". O nome de um elemento dentro de uma tag é insensível a maiúsculas e minúsculas. Isto é, pode ser escrito em maiúsculas, minúsculas ou um mistura. Por exemplo, a tag <*title*> pode ser escrita como <*Title*>, <*TITLE*> ou de qualquer outra forma.

Olá mundo!

Para melhor entendimento preparei um pequeno arquivo de HTML de exemplo, e irei explicar para vocês toda a estrutura do documento.

Um exemplo mais real

Bom, para ficar algo mais real resolvi preparar um projetinho um pouco mais estruturado e com textos, links, imagens, vídeos e muitas outras coisas.

Errr… achei meio feio… Tem como melhorar..?

Bom, que bom que notaram isso (assim espero) e eu vos digo, dá sim! Para fazer isso precisamos conhecer nossa próxima linguagem, a *CSS*!

CSS

A *CSS: Cascading Style Sheets* (ou em português: *Folhas de Estilo em Cascata*) é uma linguagem de estilização usada para descrever a apresentação de um documento escrito em *HTML* ou em *XML* (incluindo várias linguagens em *XML* como *SVG*, *MathML* ou

XHTML). A *CSS* descreve como elementos são mostrados na tela, no papel, na fala ou em outras mídias.

Estilos em cascata?

A cascata é o algoritmo para resolver conflitos em que várias regras *CSS* se aplicam a um elemento *HTML*.

A cascata atribui um conjunto de regras para aplicar o peso e a ordem de atribuição do estilo em um elemento. Para isso usa conceitos como: importância, especificidade e a posição em que foi declarada. Caso haja um conflito de regras, o seu navegador pode executar até quatro passos:

1. Analisa todas as regras.
2. Vê o nível de importância e a origem do elemento.
3. Ele vai comparar a especificidade das declarações com mesmo nível de importância.
4. Caso tenha um empate no grau da especificidade, ele olha a ordem de declaração, a que for declarada por último será selecionado.

Bom, aproveita e muda essa fonte…

Com essas noções em mente acredito que estamos mais que prontos para começar a aplicar estilos em nossa página e ver como ela vai ficar…

Responsividade e compatibilidade

Uma das coisas que devemos nos atentar quando desenvolvemos uma interface para uma aplicação web é a responsividade. Devemos testar como nossa aplicação está se comportando em diferentes navegadores e diferentes resoluções de tela, se tivermos qualquer problema de responsividade deveremos utilizar *media queries* da *CSS* para definir *breakpoints* e mudar a forma que são organizados e estilizados alguns elementos *HTML*.

Tá bom, ficou até que legal, mas ainda é só um documento

De fato, ainda é só um documento sem interatividade apesar de estar devidamente estilizado, para criar interatividade nesse nosso projeto vamos conhecer a nossa terceira linguagem da tríade…

JavaScript

O *JavaScript* (às vezes abreviado para *JS*) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, é conhecida por ser a linguagem de script para páginas Web, mas é usada também em vários outros ambientes sem ser dentro de um navegador, tais como runtimes como *Node.js*, *Bun* e *Deno* e também outras plataformas como *Apache CouchDB* e *Adobe Acrobat*. O *JavaScript* é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional).

Linguagem de programação?!

Diferentemente do *HTML* e *CSS,* o *JavaScript* é uma linguagem de programação pois executa funções e permite a criação de algoritmos, realização de operações matemáticas, interação com o usuário e muitas outras coisas.

DOM

O *DOM: Document Object Model* (ou em português: *Modelo de Objeto de Documento*) é uma interface de programação para os documentos *HTML* e *XML*. Representa a página de forma que o *JavaScript* possa alterar a estrutura do documento, alterar o estilo e conteúdo. O *DOM* representa o documento com nós e objetos, dessa forma, as linguagens de programação podem se conectar à página.

Olá mundo!

Que tal um exemplo de como fazer um “Olá mundo!” com *JavaScript*? Vamos lá…

Faz uma calculadora aí!

Exemplo de página

Backend

O que é?

Também conhecido como *Server-side* (ou lado do servidor) o *Backend* é todo o ecossistema de uma aplicação web que é executada por baixo dos panos e não vai para o navegador, incluem Web APIs, DBMS (*Database Manager System*) e outros. É aqui onde fica toda a parte lógica do negócio, sistema de pagamentos, login e registro de usuários e etc.

Linguagens de programação

Diferentemente do *Frontend* onde podemos utilizar apenas o *JavaScript* de forma nativa como linguagem de programação, no *Backend* podemos utilizar qualquer linguagem de programação que tenha a capacidade de lidar com requisições *HTTP* de forma decente, dentre elas podemos listar algumas:

JavaScript com Node.js

O *JavaScript* apesar de não ter sido feito para poder ser executado fora do navegador a comunidade *open-source* deu um jeito de adaptar o motor *JavaScript* do *Chrome* o chamado *V8* para executar *JavaScript* de forma que se comportasse como qualquer outra linguagem de programação, graças a isso podemos construir aplicações *Backend* com JS.

PHP

O *PHP: Hypertext Preprocessor* (ou em português: *Pré-processador de HiperTexto*) é uma linguagem de programação feita para gerar páginas web dinamicamente, pode ser utilizada como um script que faz operações e retorna uma página ou como um servidor web, que é a abordagem da maioria das linguagens.

Web API

Uma *Web API* é uma interface de programação de aplicação para um servidor. É utilizada para intermediar operações como recuperar, escrever e excluir dados específicos em um banco de dados ou fazer processamento de mídias em caso de um serviço de compressão de imagens.

É uma forma de expor funcionalidades por meio de rotas *HTTP*, servindo como parte de um ecossistema de uma aplicação ou como um *SaaS* (*Software as a Service*).

Mostra uma aí! (exemplo)

Bom, eu acredito que apenas teoria nunca é o suficiente, e precisamos de um pouco de prática para entendermos melhor os conceitos, por isso preparei um exemplo…

FullStack

O que é?

O termo *Fullstack* não é utilizado para definir uma área específica, mas sim, um termo para definir um profissional que tem competências tanto na área do *Frontend* quanto na área do *Backend*, ou seja, ele deve ser capaz de lidar com a criação de interface de usuário e validação no *Frontend* quanto lidar com operações no banco de dados, segurança e autenticação no *Backend*.

Exemplo de aplicação

Para melhor entendimento também preparei um exemplo de aplicação desenvolvida com competências de um profissional *FullStack*.

Referências

- https://chat.openai.com/c/2b66426d-9b27-4398-840f-7facabec51cb
- https://developer.mozilla.org/pt-BR/docs/Web/HTML
- https://developer.mozilla.org/pt-BR/docs/Web/CSS
- https://web.dev/learn/css/the-cascade?hl=pt
- https://medium.com/jaguaribetech/efeito-cascata-no-css-c55bda0a2ed4
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
- https://chat.openai.com/c/3df80cc4-b1e7-472e-9865-4bb610eca2f1
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document_Object_Model/Introduction
- https://pt.wikipedia.org/wiki/Web_API

Questões

1. O *HTML*, *CSS* e o *JavaScript* são respectivamente linguagens de:
    a. Estilização, marcação e programação
    b. Marcação, estilização e notação de objetos
    c. Marcação, estilização e programação
    d. Programação, estilização e marcação

2. Se quisermos executar alguma ação quando um botão for clicado qual linguagem devemos utilizar?
    a. HTML
    b. CSS
    c. JavaScript
    d. PHP

3. Qual a divisão da aplicação web que é responsável pelo registro de usuários no banco de dados?
    a. Frontend
    b. FullStack
    c. Machine Learning
    d. Backend

Modelos de construção de Páginas Web

Como visto anteriormente, a web teve de evoluir com o tempo, trazendo novidades baseada em novas demandas que foram surgindo, mais interatividade, maior velocidade e disponibilidade de conteúdos.

Para desenvolver essas aplicações foram criados alguns modelos de como criamos nossas páginas web, vamos ver mais detalhes sobre cada um:

Páginas Estáticas

Este modelo de criação de páginas surgiu na *Web 1.0*, tem a finalidade de entregar conteúdos estáticos e que não tenham qualquer personalização, um conteúdo simples como páginas de apresentação de um negócio ou portfólios.

Como dito anteriormente, não há qualquer tipo de personalização, ou seja, todos os usuários que acessarem o endereço em questão da página receberão um mesmo conteúdo. Não há como fazer áreas protegidas com autenticação nesse modelo de construção de páginas e muito menos criar um negócio online como um *e-commerce*.

Por outro lado, o custo para se manter uma página estática no ar é muito mais barato que qualquer alternativa a seguir, pois ela já está pronta, não requer qualquer processamento em servidor e não precisa solicitar dados extras para uma API. Arquivos estáticos podem ser entregues por uma *CDN* (Content Network Delivery) que além de estar disponível globalmente e ter uma alta disponibilidade ela é uma alternativa muito mais barata que manter um servidor fazendo o mesmo.

Páginas Dinâmicas

As Páginas Dinâmicas foram as páginas que revolucionaram a *Web* como a conhecemos, tanto que foram elas que marcaram o fim da *Web 1.0* e o início da *Web 2.0*.

Elas permitiram que houvesse personalização de páginas para cada usuário diferente, deixando que o servidor computasse uma página por inteiro a depender de algum fator específico ou uma combinação de fatores.

Foi por conta delas que surgiram diversos serviços e funcionalidades como fóruns, webmails, painéis administrativos, e-commerces, redes sociais, buscadores de páginas, chats e etc.

Porém, as Páginas Dinâmicas não são uma “bala de prata”, e tivemos que inventar um novo modelo para se construir páginas web, isso para contornar coisas como aplicações que precisavam de uma alta taxa de atualização em tempo real na interface do usuário.

Imagine ter que esperar o servidor retornar uma nova página do zero a cada nova interação como um clique em um botão? Se isso é ruim hoje, imagine na época que tínhamos internet discada e navegadores horríveis como o *Internet Explorer*. Por isso elas surgiram, as famosas *SPAs*!

SPAs

SPA é uma sigla para *Single-Page Application* na qual como o próprio nome sugere é uma aplicação de uma página só. Você pode estar pensando “mas ué, e se eu quiser que meu site tenha mais de uma página?!”, calma lá meu *pequeno gafanhoto*, com as SPAs você controla todo o conteúdo que é visível em tela pelo *Client-side* via *JavaScript*!

No modelo de *SPAs* usamos requisições *HTTP* para buscar dados dinamicamente por baixo dos panos e atualizar apenas pedaços da nossa página sem mesmo precisar efetuar um reload.

SPAs são ótimas para redes sociais, painéis administrativos, editores de mídias como vídeo, fotos e áudios e também para e-commerces que buscam uma dinamicidade maior.

Por outro lado, *SPAs* não são indicadas para quem tem blogs, fóruns, ou algum negócio que precise aparecer em pesquisas de mecanismos de buscas como o *Google*. Isso porque como a aplicação “hidrata” a página com dados via *Client-side* os bots acabam recebendo apenas uma página em branco já que eles não têm a capacidade de executar *JavaScript*.

Referências

- https://youtu.be/EW7m2WIvFgQ?si=IQLcGkq9rJo5p16_
- https://chat.openai.com/c/9912a654-fc1d-4198-88d1-6bf47a88257b

Questões

1. Para desenvolver um site de uma rede social qual seria o melhor modelo?
    a. Páginas Estáticas
    b. Páginas Dinâmicas
    c. SPAs
    d. HTML puro

2. Se eu quiser fazer um site de chat em tempo real, qual o melhor modelo para isso?
    a. Páginas Estáticas
    b. Páginas Dinâmicas
    c. SPAs
    d. HTML puro

3. Caso eu queira desenvolver um blog que seja bem indexado em motores de busca, qual alternativa eu devo utilizar?
    a. Páginas Estáticas
    b. Páginas Dinâmicas
    c. SPAs
    d. HTML puro

Hospedagem e Domínios

Hospedagem

Agora que você já construiu sua aplicação web é a hora de colocá-la no ar para que outras pessoas também possam acessá-la. Para isso precisaremos de um serviço de hospedagem para que execute nosso *Backend* e/ou sirva nossos arquivos estáticos do *Frontend*.

Tipos de hospedagem

Existem diversos tipos de hospedagem disponíveis, cada um com suas vantagens e desvantagens. A escolha do tipo ideal dependerá do contexto e necessidade do seu projeto.

Hospedagem compartilhada

Uma Hospedagem Compartilhada é ideal para projetos pequenos e iniciantes. Você divide um servidor com outros usuários, o que torna o serviço mais barato. No entanto, o desempenho pode ser impactado pelo uso dos outros usuários.

Servidor virtual privado (VPS)

Uma *VPS* oferece mais recursos e controle do que a hospedagem compartilhada. Você terá um servidor virtual dedicado, mas ainda estará compartilhando o hardware físico com outros usuários.

Servidor dedicado

Um Servidor Dedicado é ideal para projetos grandes e que exigem alto desempenho. Você terá um servidor físico inteiro à sua disposição, o que garante mais recursos e controle.

Cloud hosting

Um Cloud Hosting oferece escalabilidade e flexibilidade. Você pode aumentar ou diminuir os recursos do seu servidor de acordo com a demanda.

Domínio

Com sua aplicação online, você precisa de um endereço (um domínio) pelo qual as pessoas possam encontrá-la. Se você optou por hospedar seu site em um serviço como a *Netlify*, por exemplo, você provavelmente recebeu um subdomínio personalizado gratuitamente. Este subdomínio é uma extensão do domínio principal do serviço de hospedagem.

Os domínios têm uma estrutura específica, como **NOME**.**EXTENSÃO**.**CÓDIGO_DO_PAÍS**. Por exemplo, se sua aplicação está hospedada na *Netlify*, seu subdomínio personalizado poderia ser algo como *nome-do-site.netlify.app*, onde "netlify.app" é o domínio da *Netlify*. Outro exemplo disso é o *Google*, onde o domínio principal dele para o Brasil é *google.com.br, onde* “google” é o nome de domínio, “com” é a extensão e “br” é o código do país.

Também existem casos específicos onde domínios têm apenas **NOME**.**CÓDIGO_DO_PAÍS**. Um exemplo disso é o *Registro.br* que é um site de registro de domínios onde o endereço dele é apenas “registro.br”. Também existem plataformas que optam por utilizar isso sem mesmo residir nesse país, temos por exemplo o *VSCO* que o endereço dele é “vsco.co” onde “co” é o código de país da Colômbia.

Referências

- https://www.youtube.com/watch?v=hfA64sK5ujk&t=154s
- https://chat.openai.com/c/4b564439-2706-44ab-8729-558d1717af27
- https://g.co/gemini/share/a9837d26707e

Questões

1. Se eu tenho uma *Web API* e quero expor ela na internet, do que eu precisarei?
    a. Um domínio
    b. Uma conta no Google
    c. Uma hospedagem
    d. Um certificado SSL

2. Caso eu não goste do endereço que eu recebi de uma plataforma de hospedagem e eu queria um totalmente personalizado, eu devo comprar o quê?
    a. Um novo serviço de hospedagem
    b. Um domínio
    c. Um certificado SSL
    d. Um serviço de CDN

3. No endereço “www.tabnews.com.br”, qual é a identificação correta de cada parte dele?
    a. Domínio, extensão, subdomínio e código de país
    b. Subdomínio, domínio, código de país e extensão
    c. Subdomínio, domínio, extensão e código de país
    d. Domínio, subdomínio, extensão e código de país

Como começar a estudar sobre

Curso em Vídeo

Curso de HTML & CSS

Módulo 1/5: Começa aqui o novo @CursoemVideo de HTML5 e CSS3

Módulo 2/5: O que vamos aprender no módulo 02? - @Curso em Vídeo HTML5 + CSS3

Módulo 3/5: O que vamos aprender no módulo 3? - @Curso em Vídeo HTML5 + CSS3

Módulo 4/5: O que vamos aprender no módulo 4? - @Curso em Vídeo HTML5 + CSS3

Módulo 5/5: Ainda não disponível!

Curso de JavaScript: Curso de JavaScript - Teaser

CFBCursos

Curso de HTML: Codificação básica - Curso de HTML Completo e Profissional #01

Curso de Canvas: Introdução ao Canvas HTML e Javascript [Canvas] - Curso de Canvas - Aula 01

Curso de CSS3: Curso de CSS3 #01 - Inserindo CSS na página HTML

Curso de JavaScript: Novo Curso de Javascript Completo, Profissional e Moderno - Curso de Javascript Moderno - Aula 01

W3Schools

HTML: https://www.w3schools.com/html/default.asp

CSS: https://www.w3schools.com/css/default.asp

JavaScript: https://www.w3schools.com/js/default.asp

Livros

- JavaScript: O Guia Definitivo
- Aplicações web real-time com Node.js
- Desenvolvendo Websites com PHP: Aprenda a Criar Websites Dinâmicos e Interativos com PHP e Bancos de Dados
- Construindo Aplicações web com PHP e MySQL

Conclusão

Bom pessoal, alguém tem alguma dúvida final ou quer acrescentar algo?

Então, essa foi a apresentação, espero que tenham aprendido alguma coisa de bom hoje com essa e as outras apresentações, bons estudos e uma excelente noite à todos!

Agradecimentos

Revisão de conteúdo

- Thiago M.
- Francisco Jr.
- Equipe Técnica

Codificação de exemplos

- Thiago M.

Menção honrosa ao **Rafael Fonseca** por me aconselhar sobre os conteúdos .