**1ª Fase: CODIFICANDO**

**criar códigos no Visual Studio Code;**

**executar no navegador;**

**testes;**

**controle de versão pelo GitHub;**

**visualizar no navegador pelo ‘pages’ do GitHub;**

**criar o ‘atalho’do app mobile.**

1. Criar o arquivo manifest.json, na raiz do diretório, com metadados para um Progressive Web App (PWA), incluindo nome, descrição, ícone, tema de cor e outras informações relevantes:

{

"name": "Meu App PWA",

"short\_name": "Meu App",

"description": "Um aplicativo web progressivo (PWA) incrível!",

"icons": [

{

"src": "icons/icon-72x72.png",

"sizes": "72x72",

"type": "image/png"

},

{

"src": "icons/icon-96x96.png",

"sizes": "96x96",

"type": "image/png"

},

{

"src": "icons/icon-128x128.png",

"sizes": "128x128",

"type": "image/png"

},

{

"src": "icons/icon-144x144.png",

"sizes": "144x144",

"type": "image/png"

},

{

"src": "icons/icon-152x152.png",

"sizes": "152x152",

"type": "image/png"

},

{

"src": "icons/icon-192x192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "icons/icon-384x384.png",

"sizes": "384x384",

"type": "image/png"

},

{

"src": "icons/icon-512x512.png",

"sizes": "512x512",

"type": "image/png"

}

],

"theme\_color": "#317EFB",

"background\_color": "#FFFFFF",

"display": "standalone",

"orientation": "portrait"

}

Acima o trecho código, abaixo a explicação:

- `"name"`: O nome do aplicativo.

- `"short\_name"`: Um nome curto para ser exibido em espaços reduzidos, como a tela inicial de um dispositivo móvel.

- `"description"`: Uma descrição breve do aplicativo.

- `"icons"`: Uma matriz de objetos representando os ícones do aplicativo em diferentes tamanhos e formatos. Certifique-se de ter os ícones correspondentes disponíveis no diretório especificado.

- `"theme\_color"`: A cor do tema do aplicativo, usada pela barra de status do dispositivo e no estilo do cabeçalho da interface do usuário do navegador.

- `"background\_color"`: A cor de fundo da tela de inicialização do aplicativo.

- `"display"`: Define como o aplicativo é exibido na janela do navegador. O valor `standalone` indica que o aplicativo deve ser aberto em sua própria janela, sem elementos do navegador.

- `"orientation"`: A orientação preferida da tela do aplicativo. Neste exemplo, está definido como `portrait` (retrato).

Certifique-se de personalizar esses valores de acordo com as necessidades específicas do seu aplicativo. Além disso, substitua os ícones pelos seus próprios ícones nos tamanhos e formatos apropriados.

1. Criar o código para registrar um Service Worker, na raiz do diretório (service-worker.js), pois, é uma etapa crucial para o Progressive Web App (PWA). Aqui está código:

// service-worker.js

// Evento de instalação do Service Worker

self.addEventListener('install', event => {

console.log('Service Worker instalado');

});

O evento de instalação do Service Worker foi configurado corretamente. Quando o Service Worker for instalado, a mensagem "Service Worker instalado" será registrada no console do navegador. Isso confirma que o Service Worker foi instalado com sucesso e está pronto para ser ativado.

1. Trecho do código no index.html do "Service Worker" para ser implementado o app mobile.

<!DOCTYPE html>

<!-- index.html -->

<script>

if ('serviceWorker' in navigator) {

window.addEventListener('load', () => {

navigator.serviceWorker.register('/service-worker.js')

.then(registration => {

console.log('Service Worker registrado com sucesso:', registration);

})

.catch(error => {

console.log('Falha ao registrar o Service Worker:', error);

});

});

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Resumo:

- O service worker #975 foi ativado e está em execução.

- Uma mensagem de teste de notificação push foi enviada a partir das DevTools, com a tag "test-tag-from-devtools".

- Uma operação de sincronização (sync) foi realizada.

- Uma sincronização periódica também foi realizada.

1. Para tornar o aplicativo móvel responsivo para qualquer smartphone e tablet, você pode seguir estas diretrizes:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Layout Flexível: Use layouts flexíveis em vez de layouts fixos. Isso permitirá que o conteúdo se ajuste dinamicamente ao tamanho da tela do dispositivo.

Viewport Meta Tag: Certifique-se de incluir a tag de viewport meta no cabeçalho HTML para garantir que o layout se adapte corretamente a diferentes tamanhos de tela.

A tag `<meta name="viewport" content="width=device-width, initial-scale=1.0">` é uma importante diretiva HTML usada para controlar a escala e a largura da tela em dispositivos móveis. Aqui está o que cada parte dessa tag faz:

- `name="viewport"`: Este atributo indica que estamos configurando as propriedades de visualização do navegador.

- `content="width=device-width, initial-scale=1.0"`: Aqui, definimos duas propriedades:

- `width=device-width`: Isso define a largura do viewport para ser igual à largura do dispositivo, garantindo que o conteúdo se ajuste ao tamanho da tela.

- `initial-scale=1.0`: Define a escala inicial da página. Um valor de 1.0 significa que a página será exibida na escala original sem zoom.

Essa tag é essencial para garantir que o layout da página seja responsivo e se adapte corretamente a diferentes tamanhos de tela em dispositivos móveis. Sem isso, os navegadores móveis podem tentar mostrar a página em uma escala inadequada, resultando em uma experiência de usuário ruim.

1. Substituir o arquivo ‘stylesheet.css’ para o estilo css responsivo de smartphones e tablets, assim:

(ANTIGO)Stylesheet.css

body{

font: 15px/1.5 'Lato black', Arial, Helvetica, Sans-serif;

padding: 0;

margin: 0;

background-color: #f2f2f7;

}

/\* Global \*\*/

.container{

width: 100%;

margin: 20px;

overflow: hidden;

}

ul{

margin: 0;

padding: 0;

}

…

}

#newsletter form input[type="email"], .quote input, .quote textarea{

width: 100%;

margin-bottom: 5px;

}

}

“NOVO” STYLESHEET.CSS

/\* Estilos para dispositivos com largura de tela menor ou igual a 768px \*/

@media screen and (max-width: 768px) {

/\* Estilos aqui \*/

}

/\* Estilos para dispositivos com largura de tela entre 769px e 1024px \*/

@media screen and (min-width: 769px) and (max-width: 1024px) {

/\* Estilos aqui \*/

}

/\* Outras Media Queries conforme necessário \*/

Isso significa que os estilos dentro deste bloco serão aplicados apenas a dispositivos com uma largura de tela máxima de 768 pixels. Você pode adicionar estilos específicos aqui para ajustar a aparência do seu site em dispositivos menores, como smartphones.

Os estilos dentro deste bloco serão aplicados a dispositivos com largura de tela mínima de 769 pixels e largura de tela máxima de 1024 pixels. Esta faixa de largura de tela pode incluir tablets e dispositivos com telas um pouco maiores. Aqui, você pode adicionar estilos para otimizar a aparência do seu site nesses dispositivos específicos.

1. "Imagens Responsivas".
2. Para fazer com que o Progressive Web App (PWA) seja aberto em tela cheia, sem a barra de endereço e outras distrações do navegador, você pode seguir estes passos:

Adicionar Meta Tags ao Cabe√ßalho HTML: Certifique-se de adicionar as seguintes meta tags ao cabe√ßalho HTML do seu PWA para habilitar o modo de tela cheia:

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">

A primeira meta tag indica ao iOS que o aplicativo pode ser aberto em tela cheia, enquanto a segunda define o estilo da barra de status quando o aplicativo estiver em uso.

Adicionar Ícones da Tela de Início:

Certifique-se de incluir ícones da tela de início no manifesto do seu PWA. Isso garante que o sistema operacional iOS saiba quais ícones usar ao adicionar o aplicativo à tela inicial.

Atualizar Manifesto do Aplicativo:

No arquivo manifest.json do seu PWA, verifique se todos os parâmetros estão configurados corretamente, incluindo o nome do aplicativo, descrição, ícones, cores de tema e outras configurações relevantes.

Remover a Barra de Endereço no Safari:

Quando o PWA é aberto no Safari, a barra de endereço ainda pode estar visível. Para ocultá-la, você pode adicionar o seguinte código CSS ao seu arquivo de estilo:

/\* Esconda a barra de endereço no Safari \*/

body {

padding-top: constant(safe-area-inset-top); /\* iOS 11.0 \*/

padding-top: env(safe-area-inset-top); /\* iOS 11.2+ \*/

}

Este código CSS aproveita as variáveis de ambiente do CSS para ajustar o preenchimento do corpo para evitar que a barra de endereço do Safari cubra o conteúdo do PWA.

Adicionar à Tela Inicial:

Após implementar essas mudanças, remova o atalho existente do PWA da tela inicial do seu iPhone e adicione-o novamente. Isso garantirá que as configurações atualizadas sejam aplicadas corretamente.

Para esconder a barra de endereço nos navegadores Safari (iOS), Chrome (Android) e Safari (macOS), você pode utilizar o seguinte código CSS:

/\* Esconder a barra de endereço nos navegadores móveis \*/

@media screen and (max-device-width: 480px) {

/\* Safari (iOS) e Chrome (Android) \*/

html {

overflow: auto;

-webkit-overflow-scrolling: touch;

}

body {

height: 100%;

overflow: auto;

}

.hide-address-bar {

height: auto;

}

}

/\* Esconder a barra de endereço no Safari (macOS) \*/

@media not all and (min-resolution:.001dpcm) {

@supports (-webkit-appearance:none) {

.hide-address-bar {

padding-top: constant(safe-area-inset-top); /\* iOS 11.0 \*/

padding-top: env(safe-area-inset-top); /\* iOS 11.2+ \*/

}

}

}

Este código CSS ajusta o comportamento da barra de endereço em dispositivos móveis e no Safari do macOS.

Lembre-se de adicionar a classe hide-address-bar aos elementos da página onde deseja esconder a barra de endereço, geralmente no elemento body. Por exemplo:

<body class="hide-address-bar">

<!-- Conteúdo da página aqui -->

</body>

Ao utilizar esse CSS, a barra de endereço será ocultada nos navegadores móveis e no Safari do macOS, proporcionando uma experiência mais imersiva ao usuário.

Para tornar os ícones dentro das páginas HTML responsivos, você pode usar unidades de medida relativas, como porcentagem ou em, em vez de valores fixos de largura e altura. Isso permitirá que os ícones se ajustem de acordo com o tamanho da tela do dispositivo.

Aqui estão algumas diretrizes gerais que você pode seguir para tornar os ícones responsivos:

Use unidades de medida relativas: Em vez de definir tamanhos fixos para os ícones, como px, use porcentagens ou em para definir largura e altura. Por exemplo:

.icon {

width: 10%;

height: auto;

}

Utilize sprites de ícones: Agrupe vários ícones em uma única imagem e, em seguida, posicione-os usando CSS. Isso reduzirá o número de solicitações de imagem e melhorará o desempenho.

Considere o tamanho do toque: Lembre-se de que os dispositivos móveis têm telas menores e os usuários geralmente usam os dedos para interagir. Certifique-se de que os ícones tenham um tamanho adequado para serem facilmente tocados.

Teste em vários dispositivos: Verifique se os ícones se ajustam corretamente em diferentes tamanhos de tela e dispositivos móveis. Você pode usar ferramentas de emulação de dispositivo em navegadores da web para facilitar esse processo.

disso, é importante garantir que os elementos que contêm os ícones também sejam responsivos. Certifique-se de que eles usem layouts flexíveis ou grades CSS para se ajustar conforme necessário.

Criando ‘atalho’ pelo navegador chrome e pelo iphone:

A screenshot of a computer

Description automatically generated

Inserir vídeo do iphone, teste bem sucedido.

**2ª Fase: IMPLEMENTAÇÃO**

**No Azure, criar e implementar app mobile com banco de dados. Logo, precisamos saber qual melhor: app híbrido ou app pwa, qual melhor?**

**Integração (deploy contínuo) do GitHub Actions para o Azure.**

**⁠⁠Executar o sistema (app mobile) pelo Azure.**

**Melhorar as páginas de banco de dados relacional: ‘produtos’ e ‘cadastro de clientes’.**

**Salvar, recuperar tabelas de dados.**

**⁠⁠Análise de dados.**