UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE SOFTWARE

LUIZ GUILHERME DEVIDE SPIRITO

APLICAÇÃO PWA

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2018

LUIZ GUILHERME DEVIDE SPIRITO

APLICAÇÃO PWA

Trabalho de Conclusão de Curso apresentada na Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de bacharel em Engenharia de Software

Orientador: Diogo Cezar Teixeira Batista

CORNÉLIO PROCÓPIO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Diretoria de Graduação e Educação Profissional
Programa de Graduação em Engenharia de Software
Engenharia de Software

TERMO DE APROVAÇÃO

Aplicação PWA

por

Luiz Guilherme Devide Spirito

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro em Engenharia de Software" e aprovado em sua forma final pelo Programa de Graduação em Engenharia de Software da Universidade Tecnológica Federal do Paraná.

Cornélio Procópio, XX/XX/XXXX

Prof. Titulação, Nome Professor Orientador
Universidade Tecnológica Federal do Paraná

Prof. Titulação, Nome professor membro da banca
Universidade Tecnológica Federal do Paraná

Prof. Titulação, Nome professor membro da banca
Universidade Tecnológica Federal do Paraná

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso"

RESUMO

DEVIDE SPIRITO, Luiz Guilherme. Aplicação PWA. 20 f. Trabalho de Conclusão de Curso – Engenharia de Software, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Texto do resumo (máximo de 500 palavras).

Palavras-chave:

ABSTRACT

DEVIDE SPIRITO, Luiz Guilherme. Title in English. 20 f. Trabalho de Conclusão de Curso – Engenharia de Software, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Abstract ... (maximum of 500 words).

Keywords:

SUMÁRIO

1 CONTEXTO

Em 1990, Tim Berners-Lee desenvolveu o primeiro navegador web da história, o WorldWideWeb (BERNERS-LEE, 1996). Feito que no final desse mesmo ano se uniu a outras criações de Lee, como o servidor e as primeiras páginas web, marcando assim a história como o primeiro serviço publicado na internet.

Nos anos seguintes, com o anúncio feito pela CERN, em 1993, de que a internet seria liberada para todos (CERN, 2018), foi o estopim para que o mundo visse todo o potencial tecnológico que ali existia. Dessa maneira novas tecnologias passaram a serem criadas para facilitar todo esse processo, sendo as principais delas o CSS, em 1996, e o JavaScript em 1995.

Com a união do HTML, CSS e JavaScript temos a base dos sistemas Webs usados até hoje, porém de formas mais avançadas.

Diferente de outras linguagens, o HTML é uma linguagem de marcação (W3C, 2018b). Isso quer dizer que com ela podemos marcar elementos de nosso código, para assim definir o que deve ser mostrado na página. Porém a forma que os elementos são mostrados na página é de uma maneira muito simples, então para isso temos o CSS para nos auxiliar.

O CSS é formado por um conjunto de regras formando uma folha de estilo que irá alterar a apresentação do nosso HTML (W3C, 2018a). Porém temos algumas limitações usando apenas o CSS, como a falta de interação com o usuário e incapacibilidade de se fazer alterações dinâmicas.

Como o próprio nome diz, JavaScript é uma linguagem baseada em scripts, sendo utilizada pelo lado do cliente. Isso se deve ao fato de ser executada diretamente no browser, não tendo acesso ao servidor. Com ela podemos fazer alterações dinâmicas na nossa página, validar campos entre outras funcionalidades.

1.1 DELIMITAÇÃO DO TEMA

Com a facilidade do desenvolvimento o web, começamos a ver seu uso sendo implementado em qualquer tipo de objeto utilizado, fenômeno conhecido como a internet das coisas. Ideia que apoia o fato de ser possível acessar a internet através de qualquer dispositivo. Dessa forma o principal alvo foram os telefones celulares.

Esse foco nos aparelhos móveis acabou acarretando na criação dos smartphones, aparelhos multifuncionais. E para melhorá los ainda mais, era necessário o desenvolvimento de novas aplicações exclusivas para os mobiles.

Atualmente existem dois sistemas operacionais mais utilizados e mobiles, o Android e o iOS, exclusivo da Apple. Nos dois é possível através de uma loja virtual, fazer os downloads dos apps mais recentes. Aplicativos esses que geralmente são criados de forma nativa, isso quer dizer que são desenvolvidos na linguagem nativa do sistema operacional.

O desenvolvimento nativo é uma técnica muito utilizada, porém com grandes desvantagens em relação, por exemplo, ao desenvolvimento híbrido. Pois no nativo será necessário uma equipe para o desenvolvimento do app para apenas um sistema operacional, tornando assim o desenvolvimento muito custoso (MADUREIRA, 2017).

Por esses motivos uma nova tecnologia vem criando muita força no mercado. Esse novo tipo de desenvolvimento web é conhecido como PWA, ou Progressive Web Apps. Onde com essa tecnologia é possível desenvolver a sua aplicação web, fazendo a funcionar tanto como um site, e como um aplicativo mobile.

Tecnologia, essa conhecida por ter algumas características obrigatórias (DEVELOPERS, 2018):

- Progressivo: Deve funcionar em qualquer navegador utilizado pelo usuário.
- Responsivo: Deve se adequar aos formatos da tela sempre. Mesmo quando aberto em um desktop, celular ou tablet.
- Offline: O sistema deve estar sempre disponível, mesmo que de forma parcial.
- Instalável: Ao abrir a página web no navegador do celular, é preciso mostrar uma mensagem de instalação do aplicativo para o usuário.

Ao se cumprir todos esses requisitos, temos uma aplicação web, que pode ser vista como um site em seu navegador, ou pode ser baixada e utilizada como um aplicativo em seu dispositivo

móvel.

1.2 PROBLEMA

Em 2016 haviam aproximadamente 8.05 milhões de estudantes matriculados em cursos ofertados por 2.407 instituições de ensino superior. Número que se comparado ao ano de 2006, mostra um aumento de 62.8% da taxa de matrículas (BRASIL, 2017).

Esse grande número de universitários, em relação ao pouco número de instituições nos mostra que esses ambientes precisam ser muito grande, para comportar a alta quantidade de estudantes.

Portanto, o estudante acaba por não ter informação de todos os eventos que estão acontecendo ao seu redor, pelo simples motivo de não conhecer aquele seu ambiente completamente.

Dessa forma, a aplicação visa dar a oportunidade, para o nosso principal usuário, o estudante universitário, de saber de tudo o que está acontecendo ao seu redor de uma forma bem simples. Acessando o site, ou usando o aplicativo, ele poderá saber em tempo real se há algum evento em sua volta.

Além do fato de poder descobrir novos eventos, será possível realizar a criação deles também. Atraindo mais pessoas para uma área de seu interesse. Como por exemplo, a apresentação de um TCC, onde pessoas que se interessam por aquele assunto poderão saber que ela está acontecendo. Algo que geralmente não é muito divulgado.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

O principal objetivo desse trabalho será o desenvolvimento de um site, usando PWA (Progressive Web App) para isso. Esse site poderá ser usado como um aplicativo em dispositivos móveis. Sendo sua principal função, a de permitir que o usuário tenha conhecimento de eventos que estão acontecendo ao seu redor. Sua primeira versão será funcional apenas no campus de Cornélio Procópio, da Universidade Tecnológica Federal do Paraná.

1.3.2 OBJETIVOS ESPECÍFICOS

- Disponibilizar o mapa, através do Google Maps, com a localização atual do usuário, mostrando assim os eventos ao seu redor. Esse mapa, assim como todo sistema, estará

em apenas uma página.

- Ter o seu desenvolvimento em PWA irá garantir uma maior flexibilidade para o usuário. Tendo em vista que o foco do projeto é ajudar os estudantes a terem informações de todos os eventos acontecendo no interior do campus. Portanto o acesso poderá ser feito através de um notebook, ser baixado como um aplicativo, e até ter elementos carregados de forma offline.

- O layout da aplicação deverá ser idêntico em todas as plataformas utilizadas, se comportando sempre da mesma forma.

1.4 ORGANIZAÇÃO DO TEXTO

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PWA

Dados comparados entre os anos de 2015 e 2016, mostram que o número de downloads de aplicativos móveis diminui em 20%. Esse fato, somado ao dado de que o usuário gasta 80% do seu tempo em apenas 5 aplicativos (LIMA, 2017), mostra que esse é um mercado muito concorrido, e já dominado por grandes empresas.

Devido a esses números algumas empresas se sentem receosas ao desenvolver uma aplicação. Principalmente se ela for feita de forma nativa, onde será necessário o uso de pelo menos duas equipes de desenvolvimento, uma para sistemas Android, e outra para sistemas iOS.

Uma aplicação nativa traz algumas desvantagens, como um custo de desenvolvimento maior, e até mesmo uma incerteza sobre a aprovação da mesma em ser publicadas nas principais lojas de aplicativos (MADUREIRA, 2017).

É nesse momento que surge a opção do PWA, um modelo de desenvolvimento que combina o melhor do web, com o melhor dos aplicativos. Bastando assim apenas o desenvolvimento de tipo web, para ser ter um aplicativo funcional, assim como um site.

Mas para ser considerado um PWA, este aplicativo precisa obedecer algumas regras, que são (JUSTEN, 2017):

1. Progressivo: Deve funcionar para todo e qualquer tipo de usuário, independente do navegador utilizado. Um exemplo disso é o site do NY Times, que é um PWA, e funciona em qualquer navegador.

2. Responsive: O layout de sua página deve se comportar de forma funcional, independente do dispositivo usado. Como um celular, tablet, notebook.

3. Independente de conectividade: Uma aplicação PWA deve ter pelo menos uma funcionalidade disponível de forma offline, e aprimorada para funcionar em redes de baixa

qualidade. Um exemplo para isso é um site de notícias, onde mesmo estando offline, o usuário poderia ter acesso as noticias previamente carregas em seu último acesso.

4. Semelhante a aplicativos: Necessário que se pareça com aplicativos nativos, com interações e até emissões de notificações. Isso só será possível pelo fato de ser compilado no modelo shell de aplicativo.
5. Atual: Deve-se manter a aplicação sempre atualizada.
6. Seguro: A sua página deve ser no formato HTTPS, mantendo assim a sua segurança.
7. Descobrível: Em seus manifestos e service worker, deverá ser definido que ele pode ser identificável como um aplicativo, permitindo assim a sua instalação.
8. Reenvolvente: Deve permitir o uso de notificações push.
9. Instalável: Permite que o usuário instale a aplicação, porém sem ser preciso acessar alguma loja de aplicativos, como a Play Store ou a App Store.
10. Linkável: Pode ser acessado por uma URL, podendo assim ser acessado via navegador, e aplicação. Quando solicitada, a instalação deve ser executada de forma simples.

Apesar de todas essas regras serem necessárias, algumas delas serão essenciais no desenvolvimento da aplicação. Começando pela responsividade, pois em uma aplicação SPA(Single Page Application), em que o principal foco é um mapa, o modo em que a página é mostrada em diferentes tipos de tela faz uma grande diferença.

Outro grande foco será a parte de instalação, onde o maior o objetivo é que a aplicação depois de instalada seja extremamente parecida com uma aplicação nativa.

Para facilitar que esses objetivos sejam alcançados, existirá um arquivo chamado manifest no projeto.

2.2 MANIFEST

Um dos principais arquivos existentes em uma aplicação PWA é o manifest.json, pois através dele o seu browser irá reconhecer o site como um PWA e dando assim a opção de instalação para o usuário (JUSTEN, 2017).

O seu formato de arquivo é um JSON (JavaScript Object Notation), tipo muito usado para a transmissão de texto devido a sua simplicidade e portabilidade em relação ao JavaScript (CROCKFORD, 2006).

Abaixo podemos ver um exemplo de um arquivo manifest.json, e quais suas principais regras (KINLAN, 2018):

Figura 6: Exemplo de um arquivo manifest

```
{
  "name": "PWA Aplication",
  "short_name": "A-PWA",
  "theme_color": "#09ff1a",
  "background_color": "#000000",
  "display": "fullscreen",
  "scope": "/",
  "start_url": "/pwa-aplication",
  "lang": "pt-BR",
  "orientation": "any",
  "icons": [
    {
      "src": "/assets/img/icons/icone.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Fonte: Autoria Propria

1. name: Esse será o nome da aplicação.
2. short-name: O nome que irá aparecer no ícone do aplicativo.
3. theme-color: Define a cor tema da aplicação, como por exemplo a cor da barra de ferramentas.
4. background-color: Cor de fundo da aplicação, item obrigatório.
5. display: Define de que modo a aplicação apresentada na tela.
6. starturl: Com ela é possível saber se a página foi aberta via app.
7. lang: Define em que língua o aplicativo será utilizada.

8. orientation: Define se a aplicação será utilizada em telas na vertical ou na horizontal.
9. icons: Define as dimensões da imagem a ser usada com ícone. Essa imagem deve ser do tipo PNG.

Após a criação do manifest, é necessário o adicionar em seu projeto. Para isso é necessário fazer a sua chamado no head do projeto, utilizando a tag link.

2.3 SERVICE WORKER

O Service Worker é um arquivo java script, executado de forma paralela ao navegador. Sendo muito utilizado para o tratamento de solicitações de redes e gerenciamento de cache (GAUNT, 2018). Desse modo, temos um maior controle da experiência offline que o usuário terá.

Alguns detalhes são essenciais para se manter o seu funcionamento de forma correta. Como o fato de um Service Worker funcionar como uma thread separada do browser, não conseguindo ter acesso ao DOM(Modelo de Objetos de Documentos). E para evitar uma duplicação do Service Worker é necessário que o arquivo tenha sempre o mesmo nome, e fique sempre no mesmo lugar (JUSTEN, 2017).

Com isso em mente, o primeiro passo será verificação de que o navegador utilizado suporta o Service Worker. Atualmente navegadores como Chrome, FireFox e Opera se mostraram bastante receptivos a esta tecnologia. Porém alguns, como o Safari e Edge, não mostraram estar totalmente preparados para ela (GAUNT, 2018).

O funcionamento de um Service Worker é realizado através de fases, que se comportam como o ciclo de vida dele. Esse ciclo é formado por 6 eventos, que são:

- Install: Este evento só é chamado na primeira vez em que o Service Worker é registrado, porém caso haja alguma atualização no arquivo, ele será executado novamente.

  Nesse exemplo podemos observar que é atribuído um nome a variável cache, e pra ele é passado o horário em que o site foi gerado. Desse modo teremos a informação de quando o site foi atualizado. Isso permite que o Service Worker atualize o cache.

- Activate: É executado apenas uma vez quando uma nova versão do Service Worker for instalada, e nenhuma outra versão antiga estiver rodando.

  Um de seus principais que podemos ver na imagem, é o de excluir arquivos antigos. Dessa forma garantimos que nosso usuário nunca estará vendo informações antigas.

Figura 7: Exemplo do evento install

```
const staticCacheName = 'luiz-devide-2018-05-08-12-35';
this.addEventListener("install", event => {
    this.skipWaiting();
    event.waitUntil(
        caches.open(staticCacheName)
            .then(cache => {
                return cache.addAll(filesToCache);
            })
    )
});
```

Fonte: Autoria Propria

Figura 8: Exemplo do evento activate

```
this.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames
                    .filter(cacheName => (cacheName.startsWith('luiz-devide')))
                    .filter(cacheName => (cacheName !== staticCacheName))
                    .map(cacheName => caches.delete(cacheName))
            );
        })
    );
});
```

Fonte: Autoria Propria

- Fetch: Este evento é executado toda vez que a página for requisitada. Sendo um dos principais pela velocidade de carregamento de um conteúdo específico. Pois ele irá verificar se um arquivo já existe na cache, e caso não exista você poderá redirecionar o usuário

para uma outra página, podendo ser até offline.

- Message: Este evento é executado em situações específicas. Geralmente é uma mensagem criado pelo cliente, e lida pelo Service Worker.

Figura 9: Exemplo do evento message

```
//Enviando a mensagem para o cliente
client.postMessage({
    msg: "Exemplo de mensagem",
    url: event.request.url
});
//Recebendo a mensagem
navigator.serviceWorker.addEventListener('message', event => {
    console.log(event.data.msg, event.data.url);
});
```

Fonte: Autoria Propria

Nesse caso o cliente utilizou um método chamado postMessage() para enviar a mensagem para o Service Worker.

- Sync: Este evento será executado sempre que necessário. Sua principal função será de sincronizar uma página, mesmo que o usuário não tenha internet para isso no momento. Portanto ele ficará tentando fazer o seu carregamento até que a página esteja disponível, e quando isso acontecer poderá ser enviado uma notificação para o usuário. Um exemplo disso, são as publicações offline do facebook. Onde mesmo sem conexão o usuário pode realizar uma postagem. E ao se conectar a uma rede, essa publicação estará online.

- Push: Este evento é executado sempre que uma notificação for solicitada. É através dele que podemos criar notificações, tanto em dispositivos móveis, tanto em sistemas como o próprio Windows.

Desses eventos, teremos três de forma essencial na aplicação. Que são eles: Install, Activate e o Fetch.

2.4 TRABALHOS RELACIONADOS

3 PROPOSTA

No desenvolvimento web pode ser separado em duas partes. A primeira parte sendo a do front-end, e a segunda do back-end. Pode se dizer que o front-end é a parte na qual o usuário tem acesso, a interface de seu sistema, e suas funções. Já o back-end é o que controla as regras de negócio do sistema e o banco de dados. Por esse motivo existem tecnologias distintas para cada uma dessas partes.

3.1 TECNOLOGIAS E FERRAMENTAS

Portanto, iremos utilizar HTML5, CSS3 e JavaScript para o desenvolvimento da aplicação.

O HTML5 será utilizado na construção da página. É com ele que definimos a estrutura do site, através de seu sistema de tags. Dessa forma teremos a base de nossa página, porém de forma totalmente crua, sem nenhum estilização.

É nesse momento em que o CSS3 será utilizado. Ele será o responsável pela estilização de nossa página. Pois através dele podemos alterar fontes, definir cores, e alterar a posição de cada elemento de nossa página. Porém o CSS3 para por aí, abrindo espaço para o uso de JavaScript.

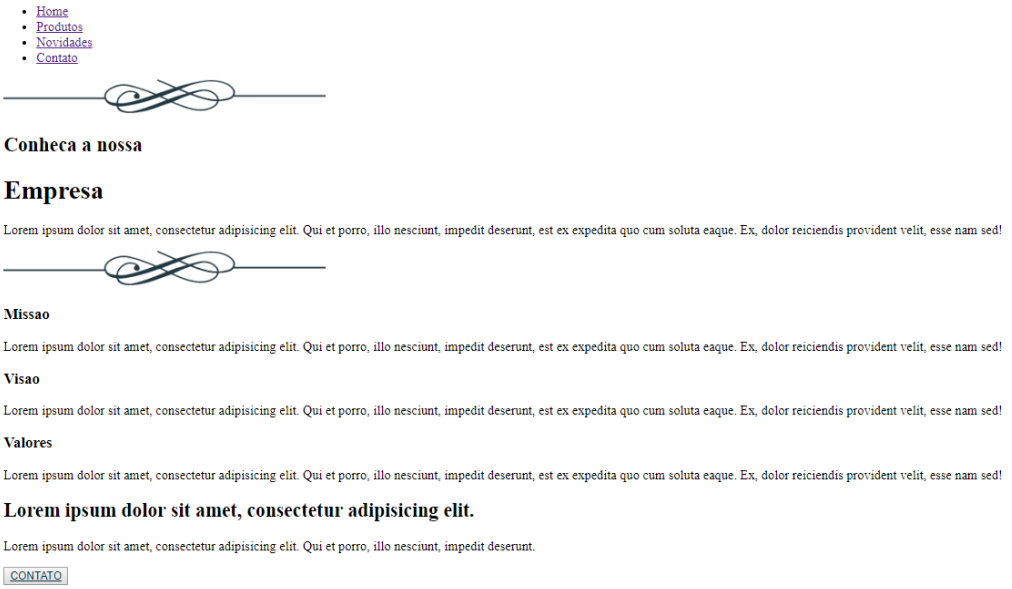
FIGURAS 10 e 11

O JavaScript nos permite implementar funcionalidades em nossa página web. Um exemplo disso será o de realizar o rolamento da página ao se clicar em uma opção do menu, ou de realizar um efeito ao se abrir o próprio menu.

3.2 DESENVOLVIMENTO

O primeiro passo do desenvolvimento será o de construir o front-end da aplicação. Nessa fase teremos a interface do sistema. Por se tratar de um SPA, a sua construção será bem minimalista. Contendo apenas um campo de busca, um menu de opções, um botão para

Figura 11: Exemplo de uma pagina apenas com o HTML5, sem o uso do CSS3



Fonte: Autoria Propria

Figura 12: Exemplo de uma pagina apenas com o HTML5 e CSS3

Fonte: Autoria Propria

adicionar um evento, tudo isso sobre o layout do mapa. Esse mapa será formado utilizando a API do Google Maps.

IMAGEM DE UM PROTÓTIPO DE TELA

Com o campo de busca será possível procurar endereços, tendo acesso a eventos próximos a ele. Já no menu teremos algumas informações do usuários, locais recentes e a

opção de criar um evento. Outro modo de se criar o evento e clicando no botão rápido existente.

O mapa a ser usado irá sempre acompanhar o usuário, mostrando todos os eventos ao seu redor. Esse mapa será no formato 2D, e irá ocupar a tela inteira. A localização do usuário será captada através de seu GPS, que compartilhará essa informação com a API do Google Maps, mostrando assim a sua localização.

Quando os eventos disponíveis aparecem no mapa, será possível selecionar ele. Isso mostrará os dados de quem criou o evento, em que horário ele será realizado, e em que local ou sala ele irá ocorrer. Essas mesmas informações serão necessárias quando o usuário criar o seu evento.

Na etapa inicial do projeto, iremos cobrir apenas o campus de Cornélio Procópio da Universidade Tecnológica Federal do Paraná.

REFERÊNCIAS

BERNERS-LEE, T. **The World Wide Web: Past, Present and Future**. 1996. Disponível em: <https://www.w3.org/People/Berners-Lee/1996/ppf.html>.

BRASIL, G. do. **Ensino superior tem 8,05 milhões de alunos matriculados em 2016**. 2017. Disponível em: <http://www.brasil.gov.br/educacao/2017/08/ensino-superior-tem-8-05-milhoes-de-alunos-matriculados-em-2016>.

CERN. **The birth of the web**. 2018. Disponível em: <https://home.cern/topics/birth-web>.

CROCKFORD, D. The application/json Media Type for JavaScript Object Notation (JSON). p. 1–10, 2006. Disponível em: <https://tools.ietf.org/pdf/rfc4627.pdf>.

DEVELOPERS, G. **Progressive Web Apps**. 2018. Disponível em: <https://developers.google.com/web/progressive-web-apps/>.

GAUNT, M. **Service Workers: uma Introdução**. 2018. Disponível em: <https://developers.google.com/web/fundamentals/primers/service-workers/?hl=pt-br>.

JUSTEN, W. **Como fazer seu site funcionar offline com PWA**. 2017. Disponível em: <https://willianjusten.com.br/como-fazer-seu-site-funcionar-offline-com-pwa/>.

KINLAN, M. G. P. **O manifesto do aplicativo web**. 2018. Disponível em: <https://developers.google.com/web/fundamentals/web-app-manifest/?hl=pt-br>.

LIMA, M. **Introdução aos Progressive Web Apps**. 2017. Disponível em: <https://tableless.com.br/introducao-aos-progressive-web-apps/>.

MADUREIRA, D. **Aplicativo nativo, web App ou aplicativo híbrido?** 2017. Disponível em: <https://usemobile.com.br/aplicativo-nativo-web-hibrido/>.

W3C. **Cascading Style Sheets**. 2018. Disponível em: <https://www.w3.org/Style/CSS/Overview.en.html>.

W3C. **HTML 5.2**. 2018. Disponível em: <https://www.w3.org/TR/html52/introduction.html#scope>.