

**UNIVERSIDADE DE CAXIAS DO SUL**  
**ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS**

**LAURA CECONI**

**EXPERIÊNCIA DO USUÁRIO EM PROGRESSIVE WEB APPS**

**CAXIAS DO SUL**

**2018**



**LAURA CECONI**

**EXPERIÊNCIA DO USUÁRIO EM PROGRESSIVE WEB APPS**

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientadora: Prof<sup>a</sup>. Dr<sup>a</sup>. Elisa Boff

**CAXIAS DO SUL**

**2018**



**LAURA CECONI**

**EXPERIÊNCIA DO USUÁRIO EM PROGRESSIVE WEB APPS**

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

**Aprovado(a) em:** \_\_\_\_/\_\_\_\_/\_\_\_\_

**Banca Examinadora**

---

Prof<sup>a</sup>. Dr<sup>a</sup>. Elisa Boff  
Universidade de Caxias do Sul

---

Prof<sup>a</sup>. Ms<sup>a</sup>. Iraci Cristina da Silveira de Carli  
Universidade de Caxias do Sul

---

Prof. Ms. Marcos Eduardo Casa  
Universidade de Caxias do Sul



## RESUMO

A popularização da *web* nas últimas décadas possibilitou a expansão das tecnologias que a compõem. Mais recentemente, surgiram os *Progressive Web Apps* (PWAs) que conseguem quase que completamente entregar as mesmas funcionalidades e experiências encontradas em aplicativos nativos para dispositivos móveis. PWAs, porém, apresentam algumas vantagens como a possibilidade de desenvolvimento de apenas um código, que pode ser executado em múltiplas plataformas e diferentes sistemas operacionais, uma vez que são executados pelos navegadores *web* dos dispositivos. Além disso, as aplicações *web* também passaram a ter a possibilidade de acesso mesmo *offline*, a partir da utilização de *Service Workers* que gerenciam a comunicação entre a aplicação e o servidor, com o intuito de fazer *cache* de dados importantes para a execução da aplicação. Este trabalho tem por objetivo estudar e elencar as características que envolvem os PWAs e quais abordagens precisam ser aplicadas em seu desenvolvimento para que seja possível entregar uma experiência inovadora e satisfatória para o usuário que o utilizar. Dessa forma, torna-se necessário também fazer o levantamento das diretrizes e princípios que envolvem o *design* de interação e a experiência do usuário em aplicações *web*. A fim de validar os estudos realizados, propõe-se o desenvolvimento de uma aplicação colaborativa que contemple as definições elencadas.

**Palavras-chave:** *Progressive Web Apps*, Experiência do Usuário, *Service Workers*, Desenvolvimento para Multiplataformas.





## ABSTRACT

The popularization of the web in the last decades made possible the expansion of the technologies that compose it. More recently, there have been Progressive Web Apps (PWAs) that can almost completely deliver the same functionality and experience found in native applications for mobile devices. PWAs, however, have some advantages such as the possibility of developing just one code, which can be executed on multiple platforms and different operating systems, as they are executed by the web browsers of the devices. In addition, web applications also have the possibility of offline access, by using Service Workers that manage the communication between the application and the server, in order to cache important data for the execution of the application. This work aims to study and list the characteristics that involve the PWAs and what approaches need to be applied in their development, so it can deliver an innovative and satisfactory experience for the user who uses it. To accomplish this, it is also necessary to survey the guidelines and principles that involve interaction design and user experience in web applications. In order to validate the studies carried out, it is proposed the development of a collaborative application that contemplates the definitions listed.

**Keywords:** *Progressive Web Apps*, User Experience, *Service Workers*, Multiplatform Development.



## LISTA DE ILUSTRAÇÕES

|   |    |
|---|----|
| Figura 1 – Representação do desenvolvimento <i>web</i> em camadas. ....   | 19 |
| Figura 2 – Demonstração da atuação dos <i>Service Workers</i> . ....  | 22 |
| Figura 3 – Combinação do <i>shell</i> da aplicação com o conteúdo encontrado em cache na renderização inicial. .... | 24 |
| Figura 4 – Exemplo de <i>Web App Manifest</i> . ....  | 26 |
| Figura 5 – Telas da aplicação desenvolvida com o Ionic Framework. ....  | 27 |
| Figura 6 – Telas da aplicação desenvolvida com o Polymer. ....  | 28 |
| Figura 7 – Elementos da experiência do usuário. ....  | 31 |
| Figura 8 – Demonstração de aplicação dos princípios de Gestalt. ....  | 37 |
| Figura 9 – Exemplos de aplicação dos sistemas de navegação. ....  | 39 |
| Figura 10 – Exemplo de <i>wireframe</i> . ....  | 41 |
| Figura 11 – Personas que representam o público-alvo da aplicação que está sendo proposta. ....                      | 48 |
| Figura 12 – Diagrama de Casos de Uso do projeto. ....   | 50 |
| Figura 13 – Estrutura organizacional e de navegação da aplicação. ....  | 50 |
| Figura 14 – Protótipo de tela referente ao UC01: Cadastrar novo usuário. ....                                       | 51 |
| Figura 15 – Protótipo de tela referente ao UC02: Editar dados do perfil. ....                                       | 51 |
| Figura 16 – Protótipo de tela referente ao UC03: Criar Diário de Viagem. ....                                       | 52 |
| Figura 17 – Protótipo de tela referente ao UC04: Adicionar um local no Diário de Viagem. ....                       | 52 |
| Figura 18 – Protótipo de tela referente ao UC05: Compartilhar Diário de Viagem. ....                                | 53 |
| Figura 19 – Protótipo de tela referente ao UC06: Solicitar recomendações de locais interessantes. ....              | 53 |
| Figura 20 – Protótipo de tela referente ao UC07: Sugerir Diário de Viagem. ....                                     | 54 |
| Figura 21 – Protótipo de tela referente ao UC08: Criar grupos. ....   | 54 |
| Figura 22 – Protótipo de tela referente ao UC09: Seguir e classificar um usuário. ....                              | 55 |
| Figura 23 – Arquitetura em três camadas. ....   | 56 |
| Figura 24 – Trecho de código em Angular. ....   | 57 |
| Figura 25 – Código das classes Diário de Viagens e Local de Interesse e suas manipulações. ....                     | 58 |
| Figura 26 – Diagrama de Classes do projeto. ....  | 59 |
| Figura 27 – Exemplo de solicitação de localização no navegador Chrome. ....   | 61 |

|   |    |
|---|----|
| Figura 28 – Demonstração do comportamento responsivo na aplicação Wanderlust. ....  | 62 |
| Figura 29 – Mensagens de alerta para a falta de conectividade. ....   | 62 |
| Figura 30 – Demonstração do certificado HTTPS no navegador Chrome. ....   | 64 |
| Figura 31 – Funcionalidade de “Adicionar à tela inicial” e <i>splashscreen</i> disponibilizados<br>pela inclusão do arquivo <i>manifest.json</i> . .... | 64 |
| Figura 32 – Demonstração da funcionalidade de notificações. ....  | 65 |
| Figura 33 – Elementos utilizados na aplicação para dar feedback ao usuário. ....  | 67 |
| Figura 34 – Resultado da avaliação <i>Performance/Desempenho</i> da ferramenta Lighthouse.<br>.....   | 68 |
| Figura 35 – Tela referente ao UC01: Cadastrar novo usuário. ....  | 79 |
| Figura 36 – Telas referentes ao UC02: Editar dados do perfil. ....  | 80 |
| Figura 37 – Telas referentes ao UC03: Criar Diário de Viagem. ....  | 81 |
| Figura 38 – Telas referentes ao UC04: Adicionar um local no Diário de Viagem. ....  | 82 |
| Figura 39 – Telas referentes ao UC05: Compartilhar Diário de Viagens. ....  | 83 |
| Figura 40 – Telas referentes ao UC06: Solicitar recomendações de locais interessantes. ....   | 84 |
| Figura 41 – Telas referentes ao UC07: Sugerir Diário de Viagem. ....  | 85 |
| Figura 42 – Telas referentes ao UC08: Criar grupos. ....  | 86 |
| Figura 43 – Telas referentes ao UC09: Seguir e classificar um usuário. ....   | 87 |

## LISTA DE QUADROS

|  |    |
|--|----|
| Quadro 1 – Comparação entre as tecnologias web, PWA, híbrido e nativo.....                       | 18 |
| Quadro 2 – Comparação entre desktops e dispositivos móveis.....                                  | 20 |
| Quadro 3 – Comparação das pontuações geradas pela avaliação do Lighthouse.....                   | 28 |
| Quadro 4 – Requisitos funcionais do projeto. ....  | 49 |
| Quadro 5 – Relação de URLs da aplicação. ....  | 66 |
| Quadro 6 – Avaliação da Página Inicial da aplicação, por meio da ferramenta Lighthouse.<br>..... | 69 |
| Quadro 7 – Avaliação da página <i>Feed</i> da aplicação, por meio da ferramenta Lighthouse. .... | 70 |
| Quadro 8 – UC01: Cadastrar novo usuário. ....  | 79 |
| Quadro 9 – UC02: Editar dados do perfil.....   | 80 |
| Quadro 10 – UC03: Criar Diário de Viagem.....  | 81 |
| Quadro 11 – UC04: Adicionar um local no Diário de Viagem. ....                                   | 82 |
| Quadro 12 – UC05: Compartilhar Diário de Viagem.....   | 83 |
| Quadro 13 – UC06: Solicitar recomendações de locais interessantes.....                           | 84 |
| Quadro 14 – UC07: Sugerir Diário de Viagem. ....   | 85 |
| Quadro 15 – UC08: Criar grupos. ....   | 86 |
| Quadro 16 – UC09: Seguir e classificar um usuário. ....  | 87 |



## LISTA DE SIGLAS

|       |                                     |
|-------|-------------------------------------|
| HTML  | HyperText Markup Language           |
| AJAX  | Asynchronous JavaScript e XML       |
| PWA   | Progressive Web App                 |
| CSS   | Cascading Style Sheets              |
| URL   | Uniform Resource Locator            |
| API   | Application Programming Interface   |
| NFC   | Near Field Communication            |
| RWD   | Responsive Web Design               |
| GPS   | Global Positioning System           |
| HD    | Hard Disk                           |
| HTTPS | Hyper Text Transfer Protocol Secure |
| JSON  | JavaScript Object Notation          |
| SEO   | Search Engine Optimization          |
| UX    | User Experience                     |
| UC    | User Case                           |





## SUMÁRIO

|              |   |           |
|--------------|---|-----------|
| <b>1</b>     | <b>INTRODUÇÃO</b> .....                     | <b>13</b> |
| 1.1          | PROBLEMA DE PESQUISA .....                  | 14        |
| 1.2          | QUESTÃO DE PESQUISA .....                   | 15        |
| 1.3          | OBJETIVOS .....                             | 15        |
| <b>1.3.1</b> | <b>Objetivo Geral</b> .....                 | <b>15</b> |
| <b>1.3.2</b> | <b>Objetivos Específicos</b> .....          | <b>15</b> |
| 1.4          | METODOLOGIA .....                           | 16        |
| 1.5          | ESTRUTURA DO TEXTO .....                    | 16        |
| <br>         |   |           |
| <b>2</b>     | <b>PROGRESSIVE WEB APPS</b> .....           | <b>17</b> |
| 2.1          | CARACTERÍSTICAS .....                       | 18        |
| <b>2.1.1</b> | <b>Progressividade</b> .....                | <b>19</b> |
| <b>2.1.2</b> | <b>Responsividade</b> .....                 | <b>20</b> |
| <b>2.1.3</b> | <b>Independência de conectividade</b> ..... | <b>21</b> |
| <b>2.1.4</b> | <b>Semelhança com aplicativos</b> .....     | <b>23</b> |
| <b>2.1.5</b> | <b>Sempre atualizados</b> .....             | <b>25</b> |
| <b>2.1.6</b> | <b>Segurança</b> .....                      | <b>25</b> |
| <b>2.1.7</b> | <b>Descobríveis e instaláveis</b> .....     | <b>25</b> |
| <b>2.1.8</b> | <b>Re-envolventes</b> .....                 | <b>26</b> |
| <b>2.1.9</b> | <b>Linkáveis</b> .....                      | <b>26</b> |
| 2.2          | FERRAMENTAS PARA DESENVOLVIMENTO .....      | 27        |
| 2.3          | LIMITAÇÕES .....                            | 29        |
| 2.4          | CONSIDERAÇÕES SOBRE O CAPÍTULO .....        | 29        |
| <br>         |   |           |
| <b>3</b>     | <b>EXPERIÊNCIA DO USUÁRIO NA WEB</b> .....  | <b>30</b> |
| 3.1          | ESTRATÉGIA .....                            | 31        |
| <b>3.1.1</b> | <b>Objetivos do <i>website</i></b> .....    | <b>31</b> |
| <b>3.1.2</b> | <b>Necessidades do usuário</b> .....        | <b>32</b> |
| 3.2          | ESCOPO .....                                | 33        |
| <b>3.2.1</b> | <b>Especificações funcionais</b> .....      | <b>33</b> |
| <b>3.2.2</b> | <b>Requisitos de conteúdo</b> .....         | <b>33</b> |
| 3.3          | ESTRUTURA .....                             | 33        |

|              |  |           |
|--------------|--|-----------|
| <b>3.3.1</b> | <b><i>Design de interação</i></b> .....                  | <b>33</b> |
| <b>3.3.2</b> | <b>Arquitetura da informação</b> .....                   | <b>34</b> |
| 3.3.2.1      | Sistemas de organização .....                            | 35        |
| 3.3.2.2      | Estruturas organizacionais .....                         | 35        |
| 3.3.2.3      | Vocabulários e metadados .....                           | 36        |
| 3.4          | <b>ESQUELETO</b> .....                                   | <b>36</b> |
| <b>3.4.1</b> | <b><i>Design de interface</i></b> .....                  | <b>37</b> |
| <b>3.4.2</b> | <b><i>Design de navegação</i></b> .....                  | <b>38</b> |
| 3.4.2.1      | Sistemas de rotulação.....                               | 38        |
| 3.4.2.2      | Sistemas de navegação.....                               | 39        |
| 3.4.2.3      | Sistemas de busca .....                                  | 40        |
| <b>3.4.3</b> | <b><i>Design da informação</i></b> .....                 | <b>40</b> |
| 3.5          | <b>SUPERFÍCIE</b> .....                                  | 41        |
| 3.6          | <b>A EXPERIÊNCIA DO USUÁRIO EM PWA</b> .....             | 43        |
| 3.7          | <b>CONSIDERAÇÕES SOBRE O CAPÍTULO</b> .....              | 45        |
| <b>4</b>     | <b>PROPOSTA DE IMPLEMENTAÇÃO DO PWA WANDERLUST</b> ..... | <b>46</b> |
| 4.1          | <b>SISTEMAS SEMELHANTES</b> .....                        | 46        |
| 4.2          | <b>MODELAGEM</b> .....                                   | 47        |
| <b>4.2.1</b> | <b>Estratégia</b> .....                                  | <b>47</b> |
| <b>4.2.2</b> | <b>Escopo</b> .....                                      | <b>48</b> |
| <b>4.2.3</b> | <b>Estrutura</b> .....                                   | <b>50</b> |
| <b>4.2.4</b> | <b>Esqueleto</b> .....                                   | <b>51</b> |
| <b>4.2.5</b> | <b>Superfície</b> .....                                  | <b>55</b> |
| 4.3          | <b>ARQUITETURA E TECNOLOGIAS</b> .....                   | 55        |
| <b>4.3.1</b> | <b>Camada de Apresentação</b> .....                      | <b>56</b> |
| <b>4.3.2</b> | <b>Camada de Negócio</b> .....                           | <b>57</b> |
| <b>4.3.3</b> | <b>Camada de Persistência</b> .....                      | <b>58</b> |
| <b>5</b>     | <b>VALIDAÇÃO DO PROJETO</b> .....                        | <b>60</b> |
| 5.1          | <b>CARACTERÍSTICAS DE PWA NA APLICAÇÃO</b> .....         | 60        |
| <b>5.1.1</b> | <b>Progressividade</b> .....                             | <b>60</b> |
| <b>5.1.2</b> | <b>Responsividade</b> .....                              | <b>61</b> |

|       |   |    |
|-------|---|----|
| 5.1.3 | Independência de conectividade .....  | 62 |
| 5.1.4 | Semelhança com aplicativos .....  | 63 |
| 5.1.5 | Sempre atualizados .....  | 63 |
| 5.1.6 | Segurança .....   | 63 |
| 5.1.7 | Descobríveis e instaláveis .....  | 63 |
| 5.1.8 | Re-envolventes .....  | 65 |
| 5.1.9 | Linkáveis .....   | 65 |
| 5.2   | ELEMENTOS DE UX NA APLICAÇÃO .....  | 66 |
| 5.2.1 | <i>Response</i> ou resposta .....   | 66 |
| 5.2.2 | <i>Animation</i> ou animação .....  | 66 |
| 5.2.3 | <i>Idle</i> ou momento ocioso .....   | 67 |
| 5.2.4 | <i>Load</i> ou carregamento .....   | 67 |
| 5.3   | AVALIAÇÃO DO LIGHTHOUSE .....   | 68 |
| 6     | CONSIDERAÇÕES FINAIS .....  | 71 |
|       | REFERÊNCIAS .....   | 73 |
|       | APÊNDICE A – QUESTIONÁRIO E RESULTADOS DA PESQUISA SOBRE O PÚBLICO-ALVO DA APLICAÇÃO PROPOSTA ..... | 75 |
|       | APÊNDICE B – CASO DE USO 01: CADASTRAR NOVO USUÁRIO .....   | 79 |
|       | APÊNDICE C – CASO DE USO 02: EDITAR DADOS DO PERFIL .....   | 80 |
|       | APÊNDICE D – CASO DE USO 03: CRIAR DIÁRIO DE VIAGEM .....   | 81 |
|       | APÊNDICE E – CASO DE USO 04: ADICIONAR UM LOCAL NO DIÁRIO DE VIAGEM .....                           | 82 |
|       | APÊNDICE F – CASO DE USO 05: COMPARTILHAR DIÁRIO DE VIAGEM .....                                    | 83 |

|  |           |
|--|-----------|
| <b>APÊNDICE G – CASO DE USO 06: SOLICITAR RECOMENDAÇÕES DE LOCAIS INTERESSANTES.....</b> | <b>84</b> |
| <b>APÊNDICE H – CASO DE USO 07: SUGERIR DIÁRIO DE VIAGEM.....</b>                        | <b>85</b> |
| <b>APÊNDICE I – CASO DE USO 08: CRIAR GRUPOS .....</b>                                   | <b>86</b> |
| <b>APÊNDICE J – CASO DE USO 09: SEGUIR E CLASSIFICAR UM USUÁRIO .....</b>                | <b>87</b> |

## 1 INTRODUÇÃO

A *web* está em constante evolução. Por isso, desde que a *internet* se tornou popularmente conhecida, muitas melhorias foram sendo introduzidas aos navegadores *web*. No início, as páginas HTML eram basicamente estáticas, com dados fixos. Com o passar do tempo, houve a necessidade da criação de conteúdos dinâmicos, surgindo a tecnologia AJAX, que possibilitou a troca de dados entre navegadores e servidores.

A outra grande evolução de *web* surgiu com a popularização dos dispositivos móveis, onde foi introduzida a filosofia *Mobile First*, que é explicada por Gardini (2015, p. 11):

A fim de se obter uma versão de *website* móvel adequada, devemos tratar a programação do comportamento nos dispositivos móveis como prioridade, e não mais como algo a ser deixado em segundo plano. E, apesar desta prática ainda não ser amplamente difundida, nota-se a existência de esforços para tentar mudar esta realidade. A concepção da filosofia *Mobile First*, por exemplo, alicerça-se exatamente nisso: a ideia é que o planejamento do *website* comece exatamente pelo dispositivo mais limitado (seja em tamanho de tela ou em recursos) e que o planejamento dos demais seja feito com base neste primeiro.

Apesar disso, muitos recursos dos *smartphones* - como geolocalização e notificações - não eram acessíveis pelos *browsers* dos dispositivos móveis, dando-se início ao desenvolvimento dos aplicativos nativos e suas disponibilizações nas lojas virtuais.

Considerados como o mais recente marco da evolução da *web*, os *Progressive Web Apps* - PWAs são aplicações que combinam o melhor da *web* e o melhor dos aplicativos (LEPAGE, 2018). São desenvolvidos utilizando HTML, CSS e JavaScript, e, progressivamente, vão recebendo novas funcionalidades. Ater (2017, p. 2 – tradução nossa), explica que: “PWAs começam como simples *websites*, mas à medida que o usuário os utilizam, eles acabam adquirindo novos poderes. Eles se transformam de um *website* para algo muito mais convencional, como um aplicativo nativo.”

No último ano, pôde-se notar um declínio na quantidade de *downloads* de aplicativos realizados nas lojas virtuais. Além disso, a maioria dos usuários não realiza sequer um novo *download* de aplicativo por mês, geralmente utilizando os mesmos 20 aplicativos ou menos. Isso demonstra que o interesse dos usuários em novos aplicativos começou a diminuir (COMSCORE, 2017). Neste contexto, pode-se citar que as aplicações *web* progressivas possuem uma vantagem sobre os aplicativos nativos, uma vez que eles possibilitam a adição de um ícone de acesso diretamente na tela inicial dos dispositivos e não necessitam de instalação.

A decisão pelo desenvolvimento de um PWA se deve, principalmente, pelo baixo custo e pelo menor tempo dedicado à sua produção, uma vez que é necessário apenas desenvolver um código enquanto que no projeto nativo é exigido um projeto para cada sistema operacional (MURAROLLI & GIROTTI, 2015 apud FREITAS et al., 2017).

Grandes empresas já estão tendo bons retornos com a utilização de aplicativos *web* progressivos. O Flipkart, maior empresa indiana de *e-commerce*, afirma que suas vendas aumentaram em 70% após transformar seu *website* em um PWA. Segundo a empresa, isso se deve principalmente ao fato de que a aplicação usa uma menor carga de dados, melhorando o desempenho de carregamento das informações; e também por ser possível manter estes dados *offline*, permitindo que, mesmo estando sem ou com uma má conexão à *internet*, a experiência do usuário continue sendo positiva (GOOGLE, 2018).

São evidentes os benefícios que as aplicações *web* progressivas trazem para organizações, desenvolvedores e usuários. Estudar seus conceitos é de grande importância, uma vez que novas tecnologias *web* surgem com grande frequência e por isso é necessário sempre estar à par destas novidades, para evitar prejuízos com desatualizações.

## 1.1 PROBLEMA DE PESQUISA

Atualmente observa-se uma grande utilização de aplicações de caráter colaborativo, tanto em *desktops*, quanto em dispositivos móveis. Waze<sup>1</sup>, Stack Exchange<sup>2</sup> e TripAdvisor<sup>3</sup>, são exemplos de sistemas construídos baseados no engajamento e interação dos usuários. Principalmente na área do Turismo é possível notar como sugestões e avaliações feitas por usuários têm grande relevância no momento de definição e planejamento de uma viagem. Devido a esses fatores, considera-se pertinente pesquisar e descobrir como um PWA pode melhorar a experiência de viajantes, desde o planejamento de uma viagem, até no momento em que o usuário deseja ter acesso às informações sobre o local onde se encontra, assim como no compartilhamento de suas experiências após a realização da viagem.

---

<sup>1</sup> <https://www.waze.com/pt-BR/>

<sup>2</sup> <https://www.stackexchange.com/>

<sup>3</sup> <https://www.tripadvisor.com.br/>

## 1.2 QUESTÃO DE PESQUISA

De que forma os aplicativos *web* progressivos podem aperfeiçoar a experiência de usuários em aplicações desenvolvidas para múltiplas plataformas?

## 1.3 OBJETIVOS

Os objetivos dividem-se em: geral e específicos.

### 1.3.1 Objetivo Geral

O presente trabalho propõe uma pesquisa para elencar quais as características, ferramentas e tecnologias fazem parte dos *Progressive Web Apps*, a fim de desenvolver uma aplicação voltada para a área do Turismo a partir deste estudo.

Esta aplicação terá caráter colaborativo, semelhante à uma rede social, onde o usuário deve ter a possibilidade de armazenar informações sobre suas viagens, fazer avaliações dos locais visitados e compartilhar estes dados com seus amigos ou seguidores.

A escolha pelo desenvolvimento desta aplicação se dá principalmente pelas funcionalidades oferecidas pelos PWAs, como acesso à geolocalização dos dispositivos e a possibilidade de acesso aos dados da aplicação mesmo estando *offline*.

### 1.3.2 Objetivos Específicos

- Conhecer a temática dos PWAs e sua aplicação no desenvolvimento de aplicativos;
- Estudar e analisar os fundamentos de experiência do usuário (*user experience*) para múltiplas plataformas;
- Estudar e analisar aplicações na área do Turismo a fim de projetar um PWA colaborativo para esta área;
- Estudar tecnologias para desenvolvimento de PWA;
- Analisar os resultados obtidos com o desenvolvimento proposto.

## 1.4 METODOLOGIA

O desenvolvimento deste trabalho será composto de quatro etapas:

- Primeira etapa: Realização do levantamento de referenciais teóricos, através de pesquisa bibliográfica, que contemplem assuntos como definições de *Progressive Web Apps*, desenvolvimento *web*, e de experiência do usuário (*user experience*);
- Segunda etapa: Avaliação das informações coletadas e projeto da aplicação voltada para a área do Turismo;
- Terceira etapa: Desenvolvimento da aplicação proposta;
- Quarta etapa: Avaliação do projeto desenvolvido utilizando os critérios definidos durante a primeira etapa.

## 1.5 ESTRUTURA DO TEXTO

Este trabalho está organizado como segue. O Capítulo 2 aborda um estudo acerca das características e tecnologias que compõem os *Progressive Web Apps*, além de citar suas limitações quanto à compatibilidade com diferentes navegadores *web*. Ainda neste Capítulo é feita uma comparação entre *frameworks* para desenvolvimento de PWAs, a fim de definir qual ferramenta deve ser utilizada no desenvolvimento da proposta de solução.

No Capítulo 3 são apresentados os elementos utilizados como base para o desenvolvimento de uma boa experiência para os usuários de *websites*, onde são citados princípios e padrões do *design*. Nesse contexto são elencadas também as melhores práticas de UX específicas para PWAs.

O Capítulo 4 apresenta um estudo sobre sistemas similares que serviram de subsídio para o levantamento de requisitos do sistema proposto, bem como as características da proposta de solução, sua arquitetura e tecnologias utilizadas.

No Capítulo 5 é feita a validação do projeto, onde é descrito como cada característica de PWA foi contemplada pela aplicação desenvolvida. Também são elencados quais elementos de UX estão presentes, além de uma avaliação que testa, entre outros aspectos, o desempenho da aplicação.

Por fim, no Capítulo 6, são apresentadas as considerações finais sobre o desenvolvimento deste trabalho, bem como sugestões de propostas para trabalhos futuros.



## 2 PROGRESSIVE WEB APPS

Os *Progressive Web Apps* (PWAs) são aplicações desenvolvidas a partir da utilização de tecnologias *web* e que são executadas nos navegadores de dispositivos de multiplataformas. Segundo Mozilla (2018, tradução nossa), “[...] PWAs são mais como uma nova filosofia para a criação de aplicativos *web*, envolvendo alguns padrões específicos, APIs e outras funcionalidades”.

O *website What Web Can Do Today*<sup>4</sup> apresenta todas as funcionalidades que estão disponíveis no navegador que está o acessando. Em dispositivos móveis, parte destas funcionalidades – como o envio de notificações, instalação e o funcionamento da aplicação mesmo *offline* – eram disponíveis somente a partir do desenvolvimento de aplicativos híbridos ou nativos. Com o avanço tecnológico dos navegadores, passou a ser possível o acesso a estas funcionalidades utilizando-se somente de tecnologias *web* (HTML e JavaScript) e inclusive levando-as também para ambientes *desktop*, caracterizando essa aplicação como um PWA.

PWAs são comumente comparados a aplicações *web*, aplicativos híbridos e aplicativos nativos. A seguir, é apresentada uma breve explicação sobre cada uma dessas tecnologias.

- Aplicações *web*: são aplicações executadas em navegadores. Podem ter conteúdos estáticos e/ou dinâmicos, porém não conseguem acessar funcionalidades disponíveis nos dispositivos. São acessados por uma URL utilizando um navegador;
- Aplicativos nativos: aplicativos desenvolvidos a partir da tecnologia específica do sistema operacional no qual será executado. Tem acesso à todas as funcionalidades disponíveis nos dispositivos, porém seu custo de desenvolvimento é considerado maior que das outras soluções por ser necessário que se escreva um código específico para cada sistema operacional, envolvendo maior quantidade de desenvolvedores e mais tempo de desenvolvimento. Pode ser encontrado e baixado em uma loja virtual;
- Aplicativos híbridos: aplicativos desenvolvidos a partir de tecnologias *web*, com comportamento muito semelhante ao de aplicativos nativos. A partir do desenvolvimento de um só código é possível gerar aplicativos para diversos sistemas operacionais que serão executados na *Web View* dos dispositivos.

---

<sup>4</sup> <https://whatwebcando.today/>

Também possui total acesso às funcionalidades dos dispositivos e podem ser encontrados nas lojas virtuais.

O Quadro 1 faz uma comparação entre as funcionalidades disponíveis para cada tipo de aplicação.

Quadro 1 – Comparação entre as tecnologias web, PWA, híbrido e nativo.

| <b>Funcionalidade</b>             | <b>Web</b> | <b>PWA</b> | <b>Híbrido</b> | <b>Nativo</b> |
|-----------------------------------|------------|------------|----------------|---------------|
| Responsividade                    | <b>Sim</b> | <b>Sim</b> | <b>Sim</b>     | <b>Sim</b>    |
| Funciona <i>offline</i>           | Não        | <b>Sim</b> | <b>Sim</b>     | <b>Sim</b>    |
| Atualizações rápidas              | <b>Sim</b> | <b>Sim</b> | Não            | Não           |
| Indexáveis pelos motores de busca | <b>Sim</b> | <b>Sim</b> | Não            | Não           |
| Notificações                      | Não        | <b>Sim</b> | <b>Sim</b>     | <b>Sim</b>    |
| Instalável                        | Não        | <b>Sim</b> | <b>Sim</b>     | <b>Sim</b>    |
| Multiplataforma                   | <b>Sim</b> | <b>Sim</b> | <b>Sim</b>     | Não           |
| Presença em lojas virtuais        | Não        | Não        | <b>Sim</b>     | <b>Sim</b>    |
| Acesso ao <i>hardware</i>         | Não        | Parcial    | <b>Sim</b>     | <b>Sim</b>    |

Fonte: Elaborado pela autora.

No Quadro 1 é possível notar que a funcionalidade “Acesso ao hardware” está definida como “parcial” para PWA. Isso se deve ao fato de que algumas funcionalidades disponibilizadas pelos dispositivos ainda não possuem APIs para *web*. Esse é o caso do NFC e dos sensores de proximidade, por exemplo. Porém nota-se um grande interesse por parte da comunidade em dar continuidade à disponibilização dessas funcionalidades. Vale ressaltar que funções mais utilizadas como a tecnologia Bluetooth, geolocalização e notificações já estão disponíveis para uso.

## 2.1 CARACTERÍSTICAS

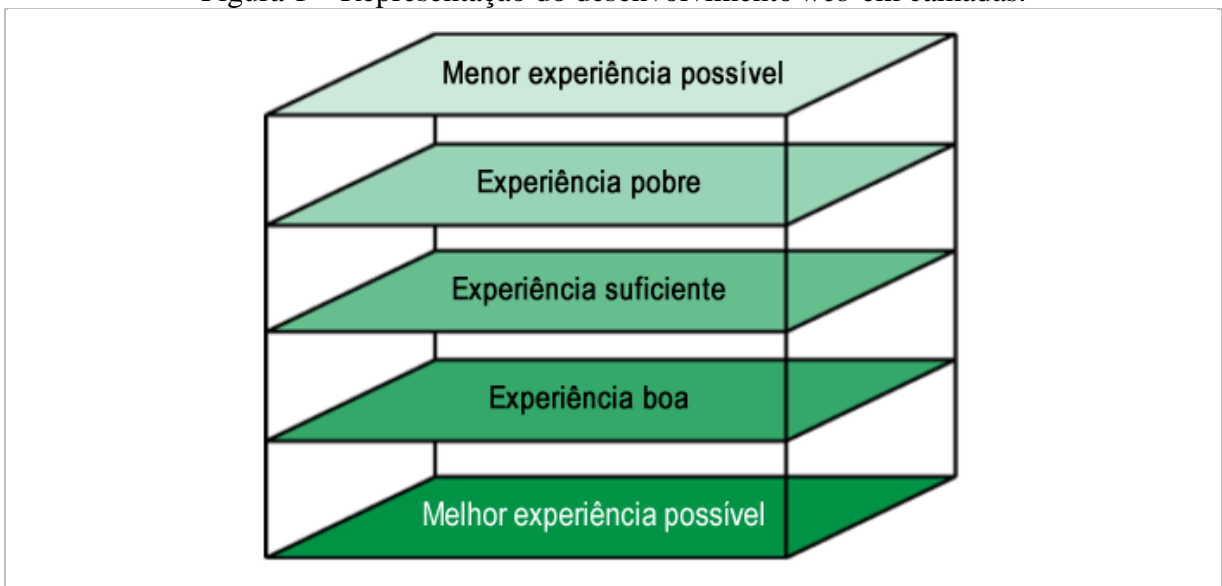
PWAs têm como foco principal entregar a melhor experiência possível para o usuário e possuem 10 características bastante marcantes. Eles devem ser: progressivos, responsivos,

independentes de conectividade, semelhantes à aplicativos, sempre atualizados, seguros, descobríveis, reenvolventes, instaláveis e linkáveis<sup>5</sup> (LEPAGE, 2018).

### 2.1.1 Progressividade

Consiste em aplicar o princípio de *progressive enhancement* (aprimoramento progressivo) no desenvolvimento do PWA. Fling (2010) destaca que *progressive enhancement* é a prática de, durante o desenvolvimento, aplicar as funcionalidades *web* em forma de camadas (conforme Figura 1), o que garante que a aplicação funcione para todos os usuários em qualquer navegador, independente das suas capacidades. Navegadores mais modernos, porém, são capazes de entregar uma quantidade maior de funcionalidades, enquanto que navegadores mais antigos apresentam algumas limitações.

Figura 1 – Representação do desenvolvimento *web* em camadas.



Fonte: Fling (2010, tradução nossa).

Segundo Ater (2017, p. 21, tradução nossa) esta prática

[...] não tem somente a vantagem de apresentar uma aplicação totalmente funcional para seus usuários, mas também torna seu site mais acessível para todos os públicos (incluindo aqueles que usam navegadores antigos ou celulares básicos) e permite que os mecanismos de busca indexem todo o seu conteúdo corretamente.

<sup>5</sup> Termos originais: *Progressive, responsive, connectivity independent, app-like, fresh, safe, discoverable, re-engageable, installable, linkable.*

Por exemplo, uma aplicação pode começar sendo desenvolvida com um conteúdo HTML simples, link e imagens. Para usuários que possuem navegadores com suporte à JavaScript podem ter algum conteúdo carregado via AJAX. Navegadores com suporte à *Service Workers* podem fazer *cache* deste conteúdo e disponibilizá-lo *offline* ao usuário.

### 2.1.2 Responsividade

O conceito *Responsive Web Design* começou a ser difundido em 2010 por Ethan Marcotte em um artigo no *blog* “*A List Apart*”, onde é dito que “o RWD nos permite projetar para o fluxo e o refluxo das coisas, adaptando o ambiente às variáveis de interação do usuário” (MARCOTTE, 2010). Nesse contexto pode-se dizer que o RWD visa estruturar as páginas *web* de acordo com o seu tamanho, orientação de tela, cores, entre outras variáveis que surgem durante a interação humano-interface.

Silva (2014, p. 35) também destaca que “[...] o conceito de *web design* responsivo na sua forma deve ser entendido como o *design* capaz de ‘responder’ às características do dispositivo ao qual é servido”. Ou seja, esta prática não deve ser projetada apenas para a adaptação do *layout* para diferentes tamanhos de telas, e sim modificar inclusive a experiência que o usuário deve ter ao utilizar a aplicação em diferentes dispositivos e plataformas. O Quadro 2 mostra uma comparação entre as diferentes experiências encontradas em *desktops* e dispositivos móveis.

Quadro 2 – Comparação entre desktops e dispositivos móveis.

(continua)

| <i>Desktop</i>                    | Dispositivos móveis   |
|-----------------------------------|---|
| Tela grande.                      | Telas pequenas, com diferentes dimensões, mas com substancial perda de área com relação ao <i>desktop</i> . |
| Teclado-padrão.                   | Teclado pequeno e sem <i>feedback</i> tátil nas versões <i>touch</i> .                                      |
| Mouse.                            | Dedo gordo ( <i>fat finger</i> ), sem cursor e <i>feedback</i> .  |
| Banda larga (baixo custo).        | 3G, 4G, limite de banda, custo alto e percentual considerável de usuários com acesso pré-pago.              |
| Energia abundante.                | Energia limitada, principalmente se os recursos como GPS estiverem ativos.                                  |
| Rede consistente.                 | Rede inconsistente.   |
| Alta capacidade de processamento. | Processamento limitado.   |

(conclusão)

| <i>Desktop</i>                                    | Dispositivos móveis  |
|---|--|
| Espaço de sobra no HD.                            | Limite de espaço de armazenamento.   |
| Uso em casa e/ou trabalho.                        | Uso em qualquer hora, em qualquer lugar.   |
| Ambiente calmo, confortável, seguro e controlado. | Em diferentes contextos, normalmente realizando mais de uma tarefa ao mesmo tempo, sem controle de variáveis como tempo, movimento, iluminação, som, atenção, etc. |

Fonte: Souza (2014).

Junto à definição de *web design* responsivo também difundiu-se o conceito *Mobile First*. Sua aplicabilidade consiste em projetar o *layout* de um *website* inicialmente com foco em dispositivos móveis, onde há maiores limitações, como espaço na tela, armazenamento e rede inconsistente. Segundo Soares (2014) estas limitações podem ser consideradas como pontos positivos nos projetos, pois assim é possível dar maior visibilidade aos conteúdos que realmente importam ao usuário que visita o *website*, além de minimizar o tráfego de dados. Soares (2014, p. 46) ainda ressalta que não só em dispositivos móveis a experiência é melhorada, mas também em aparelhos com telas maiores, como *desktops*: “[...] após a concepção e validação da solução móvel, se a equipe mantiver o mesmo foco e as mesmas prioridades, o resultado será uma solução *desktop* mais simples e com objetivos claros.”

### 2.1.3 Independência de conectividade

A maioria das aplicações *web* ainda é totalmente dependente de conectividade. Quando acessados, esses *websites* simplesmente não mostram nenhum conteúdo ao usuário, deixando-o completamente sem acesso. Caso a conexão com a *internet* esteja instável, o *website* demora a dar uma resposta ao usuário, geralmente permanecendo em uma tela branca, até que o navegador decida que o tempo de resposta atingiu seu limite e avisa o usuário que não é possível acessar naquele momento.

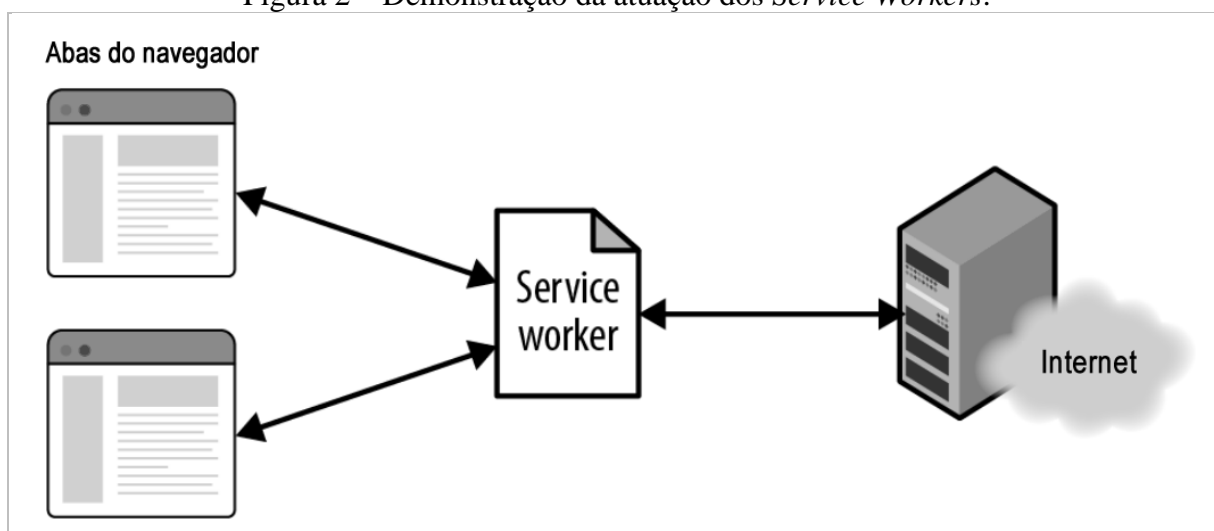
Com o intuito de melhorar a experiência dos usuários nestes casos, cunhou-se o termo *Offline First*. Assim como *Mobile First*, o conceito de *Offline First* é desenvolver a aplicação *web* partindo do cenário mais limitado, ou seja, sem nenhuma conectividade. Segundo Ater (2017), situações onde há falta de conectividade ainda são muito frequentes, e não parecem que

serão resolvidas nos próximos anos, por isso é necessário tratar essas condições não como falhas, mas como uma possível forma de se utilizar a aplicação.

Ater (2017) ainda explica que o tratamento da falta de conexão pode ser feito gradativamente. Quando há total falta de conexão, o usuário deve ser informado da impossibilidade da aplicação de executar certas funcionalidades, porém é importante que toda a ação que o usuário realizar no decorrer deste período, possa ser persistido ao servidor do *website* assim que o dispositivo estiver conectado.

Esta prática resume-se a fazer *cache* das páginas e dados acessados anteriormente pelo usuário. Navegadores mais modernos fazem uso de *Service Workers* que são *scripts* executados nos navegadores, que ao serem registrados nos *websites* passam a gerenciar as páginas a partir de eventos que ocorrem nelas.

Figura 2 – Demonstração da atuação dos *Service Workers*.



Fonte: Ater (2017, tradução nossa).

Na Figura 2 é possível notar que os *Service Workers* atuam em uma camada intermediária na comunicação entre o cliente e o servidor. Caso a conexão com a *internet* venha a falhar, o *Service Worker* é o responsável por servir os dados armazenados, se este tiver sido registrado. Em outro caso, as abas do navegador podem vir a ser fechadas, porém o *Service Worker* continua mantendo a conexão com o servidor, possibilitando, entre outras funcionalidades, o envio de *push notifications* e a não interrupção de requisições que ainda precisavam ser persistidas (ATER, 2017).

Outra funcionalidade disponibilizada pelos *Service Workers* quando a aplicação se encontra *offline* é a sincronização em segundo plano. Esta função permite que requisições

fiquem aguardando até que a aplicação tenha acesso à *internet* para concluir a requisição (ARCHIBALD, 2015).

Segundo Archibald (2014), existem diferentes padrões de armazenamento e disponibilização dos conteúdos. Cada um possui uma particularidade, e cabe à equipe responsável pelo projeto de uma aplicação *web* identificar qual padrão melhor se adequa aos requisitos do negócio.

- Somente *cache*: toda requisição feita para o recurso será buscada em *cache*. Caso o recurso não estiver em *cache*, a requisição irá falhar. Geralmente utiliza-se esse padrão para armazenar arquivos estáticos como logos, ícones e arquivos de folhas de estilo. É possível atualizar estes recursos renomeando-os e armazenando-os novamente.
- *Cache*, com *fallback* para rede: semelhante ao *somente cache*, onde busca-se o recurso primeiramente em *cache*, porém se o conteúdo não estiver disponível o *Service Worker* faz tentativas de buscar o recurso no servidor.
- Somente rede: este é o modelo clássico. O *website* tenta buscar os recursos necessários no servidor, porém se não houver conectividade, a requisição falha.
- Rede, com *fallback* para *cache*: sempre busca os recursos no servidor. Caso este venha a falhar, o *Service Worker* retorna a versão em *cache* dos recursos. Se os recursos também não forem encontrados em *cache*, a requisição falha.
- *Cache*, depois rede: ao acessar a aplicação, imediatamente é exibido todo o conteúdo armazenado em *cache*. Enquanto isso, é feita uma requisição para o servidor, em busca de novas versões do conteúdo. Se a versão que está o servidor for mais atual do que a que está em *cache*, é esta que deve ser disponibilizada para o usuário.
- *Fallback* genérico: exibe um retorno genérico caso não o conteúdo não está armazenado em *cache*, nem há conexão com *internet*. Pode ser utilizado juntamente a outros padrões de cache como último *fallback*.

#### 2.1.4 Semelhança com aplicativos

Para as aplicações que são executadas em dispositivos móveis, os usuários devem poder interagir e navegar de forma que se assemelhe a um aplicativo nativo. Para entregar essa experiência, é utilizada uma prática chamada arquitetura de *shell* (LEPAGE, 2018).

O *shell* da aplicação é composto pelos arquivos de HTML, CSS, JavaScript e imagens mínimos necessários para que o usuário tenha uma resposta rápida da interface. No primeiro acesso, esses arquivos precisam ser armazenados no *cache* local para que nos próximos acessos a apresentação da aplicação aconteça o mais rapidamente possível, independentemente de haver conexão com a *internet*. Dessa forma, a arquitetura de *shell* deixa toda a interface do usuário separada dos dados dinâmicos, que só serão carregados quando necessários (LEPAGE, 2018).

Ater (2017) por outro lado explica que mostrar a aplicação sem nenhum conteúdo também não é o ideal, e por isso ele sugere que sejam carregados os dados guardados em *cache*. Porém cabe à equipe responsável pelo projeto se questionar:

- O carregamento imediato de dados antigos que se encontram em *cache*, e logo depois a atualização dos mesmos a partir do servidor, prejudicará ou melhorará a experiência do usuário?
- Qual será o impacto no tempo de carregamento inicial da aplicação, caso haja necessidade de recuperar e apresentar este conteúdo?

A Figura 3 exemplifica como pode ser feita uma combinação entre o *shell* da aplicação e os dados armazenados em *cache*. É imediatamente apresentada ao usuário a interface mínima da aplicação e os dados antigos que estavam armazenados em *cache*, além de uma mensagem avisando ao usuário que estão sendo carregadas as mensagens mais recentes, vindas do servidor.

Figura 3 – Combinação do *shell* da aplicação com o conteúdo encontrado em *cache* na renderização inicial.



Fonte: Ater (2017, tradução nossa).



### 2.1.5 Sempre atualizados

Consiste em utilizar as funcionalidades dos *Service Workers* para que novas versões da aplicação sejam disponibilizadas para os usuários, sem que haja necessidade de tomar uma ação quanto a isso. Esta técnica se assemelha às formas de realizar *cache* dos conteúdos, citadas anteriormente.

A aplicação terá uma versão em *cache* e assim que for disponibilizada uma nova versão, o *Service Worker* deve gerenciá-la e apresentá-la ao usuário. Caso o usuário esteja utilizando o *website* no momento em que a atualização for disponibilizada, esta ficará em um estado de “esperando para ser ativada”. Enquanto o usuário continuar navegando pelo *website*, a versão que vai estar sendo acessada ainda é a antiga, ou seja, o *Service Worker* consegue executar diversas versões diferentes simultaneamente. Para que a nova versão possa ser acessada é necessário que a aba ou o navegador sejam fechados, dessa forma a versão antiga do *Service Worker* será descartada e a nova poderá ser utilizada (INSTANT, 2016).

### 2.1.6 Segurança

Em um ambiente de produção, para que seja possível registrar e utilizar um *Service Worker* é obrigatório que o *website* tenha um certificado de segurança HTTPS. É utilizado para que se estabeleça uma conexão segura entre o navegador e o servidor *web* e sua principal função é criptografar os dados que estão sendo transferidos durante essa comunicação, evitando invasões e modificações dos dados (STALLINGS, 2015).

### 2.1.7 Descobriáveis e instaláveis

Estas duas características possuem semelhanças entre si, por isso podem ser unificadas. A semelhança em questão se dá a partir da criação de um manifesto de aplicativo *web* (*Web App Manifest*), um arquivo JSON em que são definidas algumas informações da aplicação, de forma semelhante à Figura 4 (GAUNT; KINLAN, 2018).

Gaunt e Kinlan (2018) destacam que este arquivo deve possuir metadados que façam os motores de busca e os dispositivos reconhecerem que aquele *website* é um PWA. Além de terem todo seu conteúdo indexado, a criação deste manifesto ainda dá a possibilidade de:

- Determinar um nome e ícone único para a aplicação;

- Permitir a adição da aplicação na tela principal do dispositivo, como se este estivesse instalado no dispositivo;
- Gerenciar cores, *splashscreens*, orientação da tela e a forma como a aplicação deve se comportar ao ser aberta (tela cheia ou como uma página no navegador).

Figura 4 – Exemplo de *Web App Manifest*.

```
{
  "short_name": "AirHorner",
  "name": "Kinlan's AirHorner of Infamy",
  "icons": [
    {
      "src": "launcher-icon-1x.png",
      "type": "image/png",
      "sizes": "48x48"
    },
    {
      "src": "launcher-icon-2x.png",
      "type": "image/png",
      "sizes": "96x96"
    },
    {
      "src": "launcher-icon-4x.png",
      "type": "image/png",
      "sizes": "192x192"
    }
  ],
  "start_url": "index.html?launcher=true"
}
```

Fonte: Gaunt e Kinlan (2018).

### 2.1.8 Re-envolventes

Outra funcionalidade oferecida pelos *Service Workers* é o envio de notificações para os usuários, de forma a mantê-los utilizando a aplicação. As notificações podem ser geradas a partir de uma aplicação local que está sendo executada, ou pode ser enviada por um servidor, caso a aplicação não esteja aberta (GAUNT, 2018).

### 2.1.9 Linkáveis

Assim como *websites* simples, os PWAs também são executados diretamente no navegador dos dispositivos, gerando por padrão URLs. Cada página da aplicação deve ter uma URL única para que possam ser compartilhadas e acessadas por qualquer pessoa com acesso à *internet* (LEPAGE, 2018).

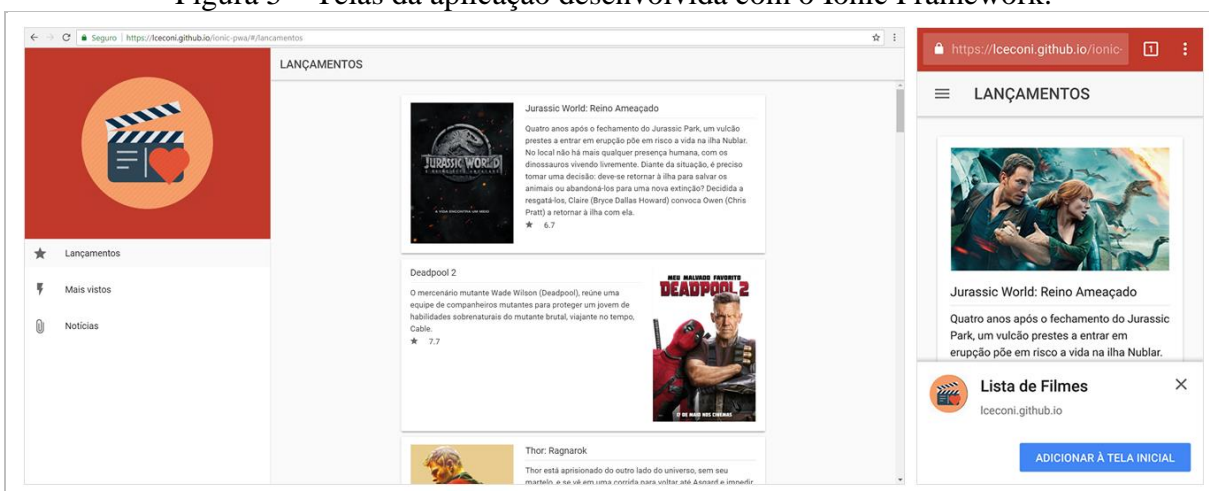
## 2.2 FERRAMENTAS PARA DESENVOLVIMENTO

Existem diversas ferramentas que auxiliam e agilizam o processo de desenvolvimento de um PWA. Nesta seção será feita uma comparação entre alguns *frameworks* utilizados para desenvolvimento de PWA; e a apresentação de ferramentas de suporte que serão utilizadas na etapa do desenvolvimento da aplicação.

Para o desenvolvimento da aplicação, optou-se por fazer uso de um *framework*. Atualmente existem diversos *frameworks* no mercado e cabe ao desenvolvedor decidir qual se adequa melhor à sua aplicação.

Dois *frameworks* foram comparados: o Ionic e o Polymer. Em ambos foi desenvolvida uma aplicação que lista os últimos lançamentos de filmes, a partir da utilização de uma API disponibilizada pelo *website The Movie Database* <sup>6</sup>. Além disso, deveria ser possível acessar a aplicação mesmo sem conexão à *Internet* e que a mesma pudesse ser instalada em dispositivos móveis. Nas Figuras 5 e 6 são apresentadas imagens das telas das aplicações desenvolvidas.

Figura 5 – Telas da aplicação desenvolvida com o Ionic Framework.



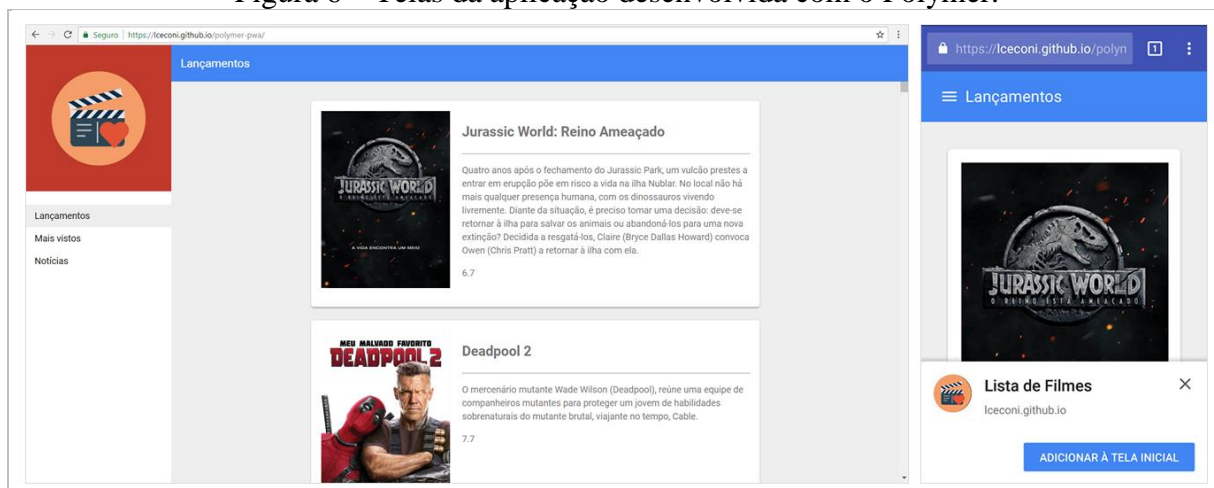
Fonte: Elaborado pela autora.

Para medir o desempenho das aplicações, executou-se a ferramenta Lighthouse. Esta ferramenta é utilizada para avaliação de aplicações web e PWA e pode ser obtida a partir da instalação da sua extensão para o Google Chrome, ou executando-a diretamente pelas Ferramentas de Desenvolvedor, também disponibilizadas nas últimas versões do Google Chrome. Quando ativado em um *website*, o Lighthouse executa diversos testes na página e gera

<sup>6</sup> <https://www.themoviedb.org/>

um relatório onde são pontuados alguns aspectos, além de explicar quais técnicas podem ser utilizadas para que estes pontos possam ser melhorados.

Figura 6 – Telas da aplicação desenvolvida com o Polymer.



Fonte: Elaborado pela autora.

Cada métrica possui um *checklist*<sup>7</sup> de configurações, valores e boas práticas que devem ser aplicados ao *website*. A pontuação gerada pela ferramenta baseia-se nestes *checklists*, onde cada item contemplado é somado à pontuação final da métrica. Pontuações de 0 a 44 são consideradas insuficientes; pontuações de 45 a 74 são suficientes; e pontuações de 75 a 100 são consideradas ótimas. O Quadro 3 apresenta os dados coletados a partir da execução do Lighthouse nas duas aplicações desenvolvidas.

Quadro 3 – Comparação das pontuações geradas pela avaliação do Lighthouse.

| Métricas          | Ionic          |               | Polymer        |               |
|-------------------|----------------|---------------|----------------|---------------|
|                   | <i>Desktop</i> | <i>Mobile</i> | <i>Desktop</i> | <i>Mobile</i> |
| Performance       | 97             | 48            | 100            | 72            |
| PWA               | 82             | 91            | 82             | 91            |
| Acessibilidade    | 80             | 80            | 100            | 100           |
| Melhores práticas | 100            | 100           | 94             | 94            |
| SEO               | 89             | 89            | 100            | 100           |

Fonte: Elaborado pela autora.

Nota: *Desktop* simula uma rede cabeada de *Internet*, e *mobile* simula uma rede 3G.

<sup>7</sup> Checklist completa: <https://developers.google.com/web/tools/lighthouse/?hl=pt-br>

A partir desta comparação é possível notar que o *framework* Polymer se mostra superior ao Ionic nos quesitos performance, acessibilidade e SEO. Outras vantagens ainda podem ser citadas para reforçar a utilização do Polymer, como o fato de o *framework* Ionic ainda não possuir uma boa documentação quanto à criação de PWAs e apenas possuir uma versão Beta de *boilerplate*. Desta forma, conclui-se que o *framework* que será utilizado no desenvolvimento da proposta de solução é o Polymer.

### 2.3 LIMITAÇÕES

Atualmente PWAs estão em total funcionamento em dispositivos Android e no navegador Google Chrome. Navegadores mais populares como Mozilla Firefox e Microsoft Edge apresentam as mesmas funcionalidades disponíveis no Google Chrome, exceto a sincronização em segundo plano. Apesar disso, ambos já declararam que a função está em desenvolvimento e será disponibilizada em breve.

Até o presente momento, porém, os dispositivos com o sistema operacional iOS e o navegador Safari ainda apresentam diversas limitações na execução de PWAs. Pode-se citar o envio de notificações e o armazenamento permanente de dados em cache como as funcionalidades mais importantes que ainda estão indisponíveis. Além disso, o processo de instalação (ou a função de adicionar a aplicação na tela inicial) demonstra algumas inconsistências (FIRTMAN, 2018).

### 2.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Ao estudar as características dos PWAs, pode-se perceber o crescimento das funcionalidades disponíveis nos navegadores a partir da utilização principalmente dos *Service Workers*. Sem dúvida eles fazem parte de uma grande evolução para a *web*, onde finalmente é possível entregar experiências melhores para os usuários, a partir de respostas mais rápidas, acesso *offline* e recebimento de notificações. Além disso, estudos e projetos continuam avançando para que PWAs sejam cada vez mais populares, o que ressalta a sua eficácia.

### 3 EXPERIÊNCIA DO USUÁRIO NA WEB

Como citado no capítulo anterior, PWAs tem como foco principal entregar a melhor experiência possível para o usuário. Segundo Norman e Nielsen (2018), o termo experiência do usuário, ou *user experience* (UX), se refere à toda forma de interação que o usuário final tem para com um produto ou serviço. Eles ainda ressaltam que:

[...] O primeiro requisito para uma experiência de usuário exemplar é atender às necessidades exatas do cliente, sem problemas ou incômodos. Em seguida vem a simplicidade e a elegância que resultam em produtos que são uma alegria para si próprios, uma alegria de usar.

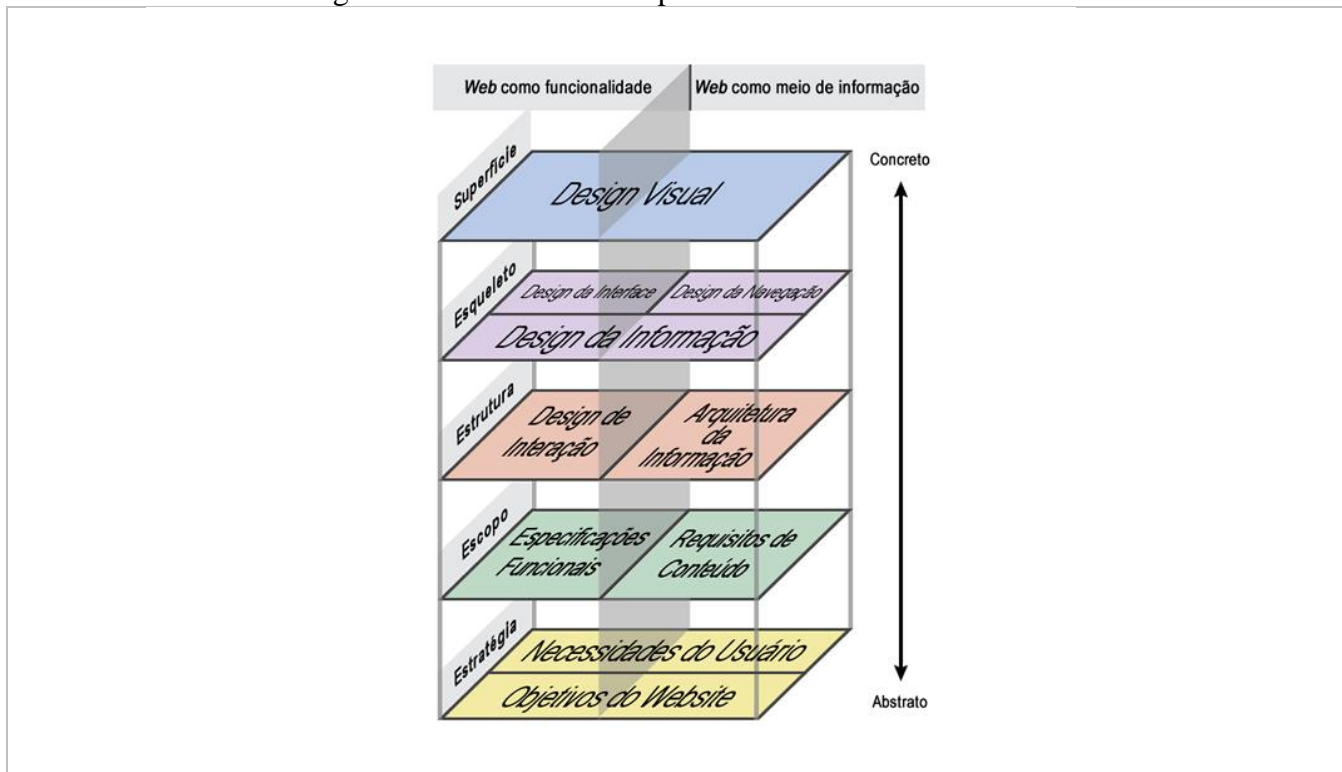
Garrett (2011) propõem um *framework* para auxiliar no processo de *design* da experiência do usuário de um produto, ou mais especificamente para projetos de *websites*. Nele são propostas 5 camadas<sup>8</sup> a serem seguidas: Estratégia, Escopo, Estrutura, Esqueleto e Superfície<sup>9</sup>; onde o primeiro tem um conceito mais abstrato, enquanto que o último, mais concreto. Cada plano depende do plano anterior, ou seja, escolhas feitas em um plano delimitam as decisões que podem ser feitas nos posteriores. Para melhor explicá-los, Garrett dividiu cada plano utilizando duas abordagens distintas: a *web* como funcionalidade e a *web* como meio de informação. A Figura 7 ilustra os elementos e suas divisões.

---

<sup>8</sup> Termo original: *Planes*.

<sup>9</sup> Termos originais: *Strategy, Scope, Structure, Skeleton e Surface*.

Figura 7 – Elementos da experiência do usuário.



Fonte: Adaptado de Garrett (2011, tradução nossa).

Neste capítulo será feita uma explicação destes elementos e suas aplicações em projetos de *websites*.

### 3.1 ESTRATÉGIA

A camada de Estratégia consiste em determinar quais são os objetivos que devem ser alcançados pelo *website* (objetivo do produto), e o que o usuário espera poder realizar ao navegar pelo *website* (necessidade do usuário). Garrett (2011, p. 37, tradução nossa) explica que: “A palavra-chave é especificidade. Quanto mais claramente pudermos articular o que queremos, e o que os outros querem de nós, mais precisamente podemos ajustar nossas escolhas para atingir esses objetivos.”

#### 3.1.1 Objetivos do *website*

Devem ser definidos a partir de análises sobre quais as metas que devem ser alcançadas com a criação do *website*. Geralmente essas abordagens são bastante amplas, primeiramente sendo definido como “uma forma de gerar lucro para a organização”. Porém elas precisam ser

mais específicas e limitadas, partindo da questão “para que serve o meu *website*?” (GARRETT, 2011).

Garrett (2011) ainda cita que se deve também determinar métricas, para que seja possível validar se o objetivo em questão está sendo alcançado ou não. Alguns exemplos de métricas válidas são:

- A quantidade de usuários que estão acessando o *website* ao longo de um determinado tempo;
- A quantidade de usuários que estão revisitando o *website*.

### 3.1.2 Necessidades do usuário

Para determinar as necessidades do usuário é preciso levar em consideração que as pessoas envolvidas na utilização de um produto podem não ser somente o usuário final. Um novo sistema em uma empresa, por exemplo, pode influenciar em processos de pessoas que não trabalham diretamente utilizando a aplicação. Estas pessoas são chamadas de *stakeholders*, e, apesar de não serem o público-alvo do produto, também devem ter suas necessidades contempladas (PREECE; ROGERS; SHARP, 2005).

Segundo Preece, Rogers e Sharp (2005, p. 192) as necessidades dos usuários podem ser determinadas a partir de algumas questões como: “[...] suas características e capacidades, o que estão tentando alcançar, como fazem isso atualmente e se atingiriam seus objetivos com mais eficiência caso recebessem algum tipo de suporte”. Existem diversos pontos que podem influenciar as necessidades dos usuários, que variam desde características físicas até questões culturais. Portanto é imprescindível a realização de pesquisas, entrevistas ou a organização de grupos de foco para coletar informações relevantes sobre o público que se está tentando atingir (PREECE; ROGERS; SHARP, 2005).

Outra forma de identificar as necessidades dos usuários é formulando personas. Segundo Benyon (2011, p. 33), personas “são representações concretas dos diferentes tipos de pessoas para as quais o sistema ou serviço está sendo projetado. Personas devem ter um nome, alguns antecedentes e, o que é mais importante, algumas metas e aspirações”. Para se obter um resultado significativo, é preciso criar diversas personas com características bem distintas, de forma que suas necessidades também sejam as mais variadas. Dessa forma é possível identificar como o produto em questão pode auxiliar esta persona a atingir seu objetivo. (BENYON, 2011).



## 3.2 ESCOPO

A camada de Escopo consiste em realizar o levantamento de requisitos do *website*, a partir das definições dos objetivos do produto e das necessidades do usuário, feitas na camada anterior. É dividida em dois tipos: as especificações funcionais e os requisitos de conteúdo (GARRETT, 2011).

### 3.2.1 Especificações funcionais

Conjunto de funcionalidades que devem ser incluídas no produto para que satisfaça as necessidades do usuário. Pode-se considerar que é nesta etapa que são levantados os requisitos funcionais e não-funcionais do projeto.

### 3.2.2 Requisitos de conteúdo

Definição sobre a abordagem que o *website* deve ter em relação ao conteúdo disponibilizado. É geralmente mais associada às áreas de comunicação e marketing, uma vez que depende de variáveis como:

- A maneira que deve ser feita a comunicação com os usuários;
- A frequência de atualização do conteúdo;
- Os responsáveis pela publicação do conteúdo;
- Políticas de uso de mídia, como imagens e vídeos;

## 3.3 ESTRUTURA

A camada da Estrutura é onde o projeto do *website* começa a tomar forma, sendo mais concreto e tangível do que as camadas anteriores. É composto de duas abordagens: o *design* de interação e a arquitetura da informação.

### 3.3.1 Design de interação

*Design* de interação tem como objetivo “[...] criar experiências que melhorem e estendam a maneira como as pessoas trabalham, se comunicam e interagem” (PREECE; ROGERS; SHARP, 2005, p. 28). Ou seja, seu foco principal é facilitar e auxiliar para que os

usuários atinjam seus objetivos. Para que esse processo possa ser bem definido, foram estabelecidas metas de usabilidade e metas decorrentes da experiência do usuário.

No quesito de usabilidade foram determinadas 6 metas:

- Eficácia: o *website* deve possibilitar que todas as suas funcionalidades sejam executadas de maneira eficiente;
- Eficiência: o formato do *website* deve ser simples e objetivo, de forma que o usuário possa ter um alto nível de produtividade após aprender a utilizá-lo;
- Segurança: consiste em prevenir que o usuário realize ações indesejáveis, ou disponibilizar formas de reverter erros;
- Utilidade: o *website* deve fornecer as funções necessárias para que os usuários possam realizar suas tarefas conforme a maneira que julgarem melhor;
- Capacidade de aprendizagem<sup>10</sup>: as funcionalidades do *website* devem ser fáceis de aprender. Preferencialmente não deve tomar um longo tempo para que o usuário entenda e saiba como utilizar as funções básicas;
- Capacidade de memorização<sup>11</sup>: após aprender como utilizar o *website*, deve ser fácil de lembrar como operá-lo, principalmente se o *website* não é utilizado com frequência.

Já no quesito de experiência do usuário, as metas apresentam-se de forma mais abstrata, tendo como objetivo o de mensurar o quão prazeroso é utilizar um *website*. Portanto é preciso projetar a aplicação para que ela seja: satisfatória, agradável, divertida, interessante, útil, motivadora, esteticamente apreciável, incentivadora de criatividade, compensadora, emocionalmente adequada. Vale ressaltar que as metas de usabilidade são fundamentais para o *design* de interação, enquanto que nem todas as metas decorrentes da experiência do usuário precisam ser contempladas.

### 3.3.2 Arquitetura da informação

A arquitetura da informação define a forma como o conteúdo disponibilizado no *website* deve ser classificado e organizado para facilitar o entendimento do usuário. Para esta abordagem é necessário entender o conceito de sistemas de organização, da estrutura da organização e de metadados e vocabulário no contexto de arquitetura da informação.

---

<sup>10</sup> Termo original: *Learnability*

<sup>11</sup> Termo original: *Memorability*

### 3.3.2.1 Sistemas de organização

Segundo Morville e Rosenfeld (2006), sistemas de organização referem-se à diferentes formas de classificar os conteúdos de um *website*. Apresentam-se de duas maneiras distintas: esquemas organizacionais exatos e esquemas ambíguos. Os esquemas organizacionais exatos são 3:

- Alfabético: utilizado na maioria dos casos, por ser de fácil compreensão. Servem como base para outros tipos de esquemas;
- Cronológico: mais utilizado para organização de arquivos, artigos ou agendas. Geralmente é necessário utilizar alguma outra forma de esquema em conjunto com esta, como pesquisar por palavra-chave.
- Geográfico: utilizado para assuntos referentes a viagens, política e meteorológicos. Frequentemente apresentado juntamente com mapas.

Os esquemas ambíguos são 5:

- Tópicos ou assuntos: para projetar uma organização desse tipo é necessário que estejam bem definidos quais são os objetivos do usuário ao utilizar o *website*;
- Tarefas: organiza os conteúdos a partir de ações que o usuário pode tomar, como por exemplo “Comprar” ou “Editar”;
- Público: específicos para algum tipo de usuário, por exemplo “Guia para estudantes”;
- Metáforas: são utilizadas para estender os conhecimentos dos usuários. Deve ser utilizado com cautela, pois nem sempre é possível existir uma definição exata sobre o que se está propondo;
- Híbrido: junção de vários esquemas. Deve ser bem estruturado para que não resulte em um esquema confuso.

### 3.3.2.2 Estruturas organizacionais

As estruturas organizacionais são definidas pensando no modo como os usuários podem navegar pelo *website*. A estrutura hierárquica, também conhecida como estrutura de cima para baixo, é baseada nas definições feitas na camada de estratégia. Na página raiz são apresentadas as categorias mais amplas do conteúdo e das funcionalidades necessárias para atingir os objetivos estratégicos. Depois essas categorias são divididas em subcategorias e assim

por diante. Esta é a abordagem mais utilizada nos projetos, uma vez que a relação hierárquica é de fácil compreensão para os usuários (GARRETT, 2011).

Já a estrutura de organização de banco de dados é conhecida como estrutura de baixo para cima. Também é dividida em categorias e subcategorias, porém sua abordagem é feita com base nas definições da camada de escopo, ou seja, nos requisitos do sistema. Tendo conhecimento de quais conteúdos serão disponibilizados, deve-se agrupá-los em categorias de “baixo nível” conforme suas semelhanças e depois inserindo-os em categorias de “alto nível” (GARRETT, 2011).

Há ainda a organização de hipertexto, que envolve dois tipos de componentes: a informação que vai ser conectada, e a conexão entre essas informações. Provê grande flexibilidade para a organização do conteúdo, porém pode ser confusa, uma vez que nem todos os usuários podem compreender da mesma forma as conexões estabelecidas. Recomenda-se utilizar primeiramente a estrutura hierárquica e complementar a organização utilizando o hipertexto onde for possível (MORVILLE; ROSENFELD, 2006).

### 3.3.2.3 Vocabulários e metadados

Manter um padrão no vocabulário do *website* também deve ser levando em consideração no projeto da organização. Para isso é necessário conhecer de que forma os usuários se comunicam e desenvolver um sistema de nomenclaturas para o *website*.

A partir da criação de uma nomenclatura padrão, ela poderá ser utilizada em conjunto com metadados. Metadados são “os dados sobre dados”, ou seja, dados sobre o conteúdo do *website*. Uma das formas mais utilizadas atualmente para a inclusão de metadados é a adição de *tags* nos artigos e páginas dos *websites*. As *tags* consistem em palavras-chave que agrupam conteúdos de mesmos assuntos, além de melhorar os resultados nos mecanismos de buscas (BENYON, 2011).

## 3.4 ESQUELETO

Na camada de Esqueleto é detalhada a forma como as funcionalidades devem ser apresentadas ao usuário, considerando a estrutura definida na camada anterior. Na abordagem de *web* como funcionalidade, o esqueleto é definido pelo *design* de interface. Já na abordagem da *web* como meio de informação, define-se o *design* de navegação. Comum às duas abordagens é apresentado o *design* de informação.

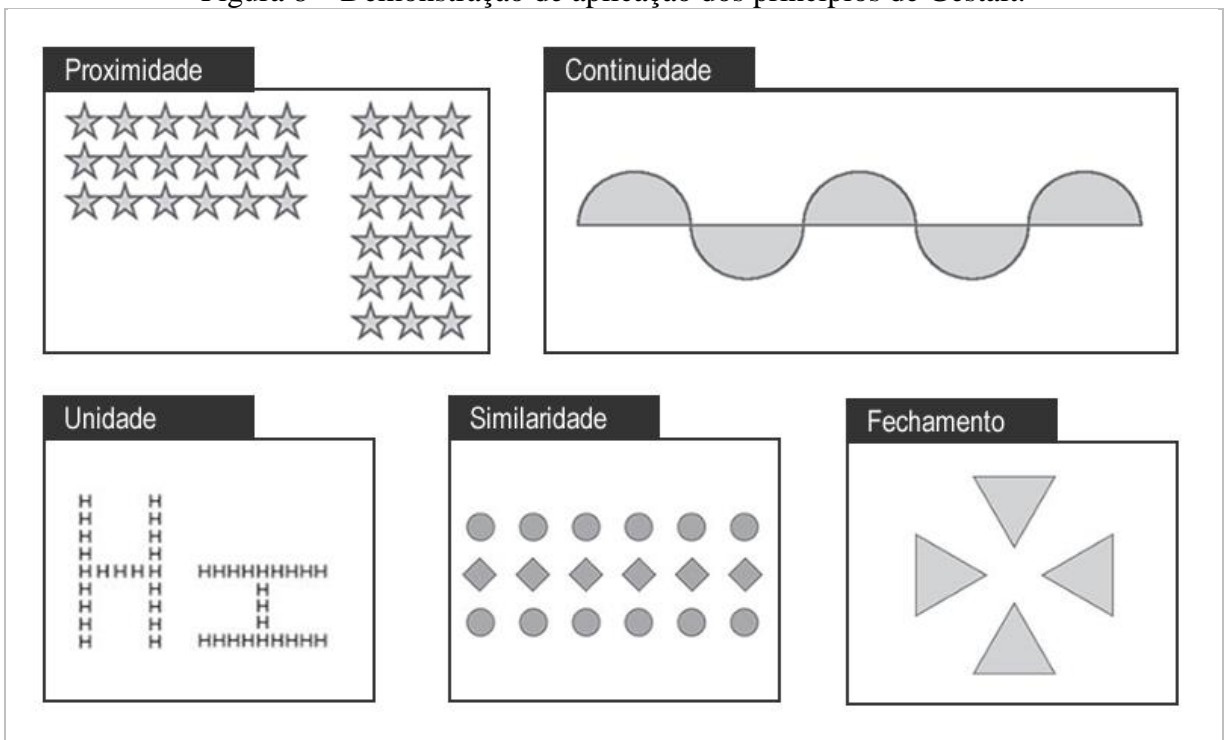
### 3.4.1 Design de interface

Na abordagem da *web* como funcionalidade, é definida a disposição dos elementos de interface, chamado de *design* de interface. O *design* de interface consiste em escolher e organizar os elementos apresentados na tela, de forma a possibilitar a interação do usuário com as suas funcionalidades. Geralmente preocupa-se com a posição de botões, menus e com o agrupamento de elementos que possuem alguma relação, como em campos de formulários (GARRET, 2011).

Nesse caso pode-se afirmar que nesta etapa são aplicados os princípios de percepção da Gestalt, que são explicados por Benyon (2010) e ilustrados na Figura 8:

- Proximidade: elementos que estão dispostos próximos um ao outro tendem a ser percebidos com semelhanças ou unidos;
- Continuidade: tendência a visualizar certos padrões entre elementos diferentes;
- Similaridade: organização dos elementos pelas suas semelhanças;
- Fechamento: tendência a visualizar imagens fechadas, mesmo que elas tenham aberturas;
- Unidade: relação de diversos elementos que formam um todo.

Figura 8 – Demonstração de aplicação dos princípios de Gestalt.



Fonte: Adaptado de Benyon (2010).

### 3.4.2 Design de navegação

Já na abordagem de *web* como meio de informação, apresenta-se o *design* de navegação, que serve para projetar a melhor forma de navegação entre os conteúdos do *website*. Morville e Rosenfeld (2006) destacam que um bom design de navegação deve conter três elementos de arquitetura da informação: sistema de rotulação, sistema de navegação e sistema de busca.

#### 3.4.2.1 Sistemas de rotulação

Rótulos são utilizados para representar coisas, que podem ou não possuir uma relação entre elas. É importante que seja determinado um padrão de nomenclatura no sistema de rotulação, de forma que o usuário passe a entendê-las claramente em qualquer contexto que estiver inserido. São encontradas de duas formas diferentes: rotulação textual e icônica. A rotulação textual é composta por:

- *Links* contextuais: são *links* incorporados em corpos de textos que contém maiores explicações ou detalhes sobre o contexto que está inserido;
- Cabeçalhos: também classificados como títulos e subtítulos de uma página. São utilizados como uma pequena prévia do conteúdo que está sendo apresentado em seguida ou como separador de diferentes categorias. Geralmente sua estrutura é hierárquica, onde o título no cabeçalho principal possui fonte maior que o subtítulo, e assim por diante;
- Opções de sistemas de navegação: são rótulos criados para integrarem os sistemas de navegação de um *website*, como no menu principal. Devem ser bem projetados para que a navegação seja clara;
- Termos de índice: exemplos de termos de índice são as *tags* e metadados relacionados às páginas. Colaboram para que as pesquisas por termos sejam mais precisas.

A rotulação icônica se refere à utilização de imagens para a representação de uma informação. É bastante utilizado para facilitar o entendimento de uma função específica e por serem de fácil reconhecimento visual. Deve ser utilizado com cautela, uma vez que esta é uma forma limitada de linguagem e sugere-se que, sempre que possível, esse tipo de rotulação esteja acompanhado de um texto, tornando-o mais claro possível.

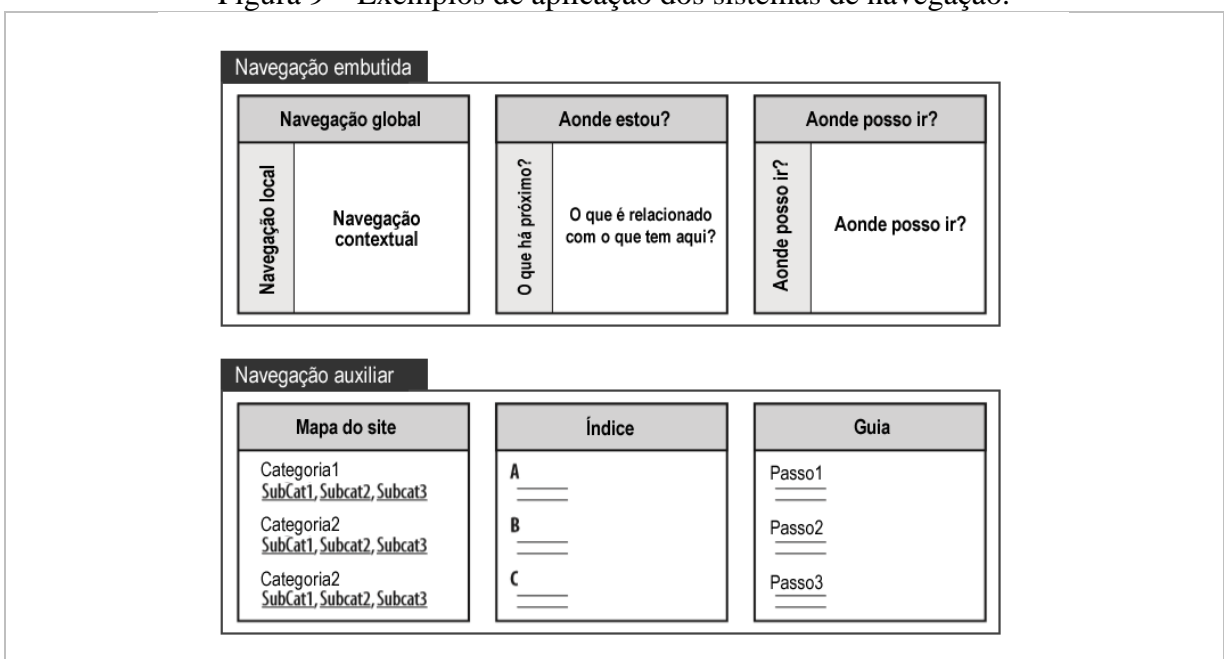
### 3.4.2.2 Sistemas de navegação

Os sistemas de navegação auxiliam os usuários a saber em qual parte do *website* eles se encontram naquele momento, e quais são as próximas possibilidades de navegação. São categorizados como sistemas de navegação embutidos e os auxiliares.

Sistemas de navegação embutidos são os que estão dispostos ao redor do *website* e mostram ao usuário em qual contexto ele se encontra e suas próximas opções. São encontrados em três áreas de um *website*: 1) a navegação global costuma ser composta sempre pelas mesmas categorias, além de um link para a página inicial, e fica disponível em todas as subpáginas do site; 2) a navegação local apresenta as subcategorias conforme a categoria acessada no menu global; e por fim, 3) a navegação contextual apresenta links relacionados ao conteúdo apresentado, inclusive os que já foram acessados (chamados de navegação migalha-de-pão ou *breadcrumb*).

Já os sistemas de navegação auxiliar são páginas específicas do *website* que listam as páginas que podem ser acessadas conforme algum tipo de categoria. Podem ser encontradas como um mapa do site, onde são listadas as categorias e subcategorias do *website*; em formato de índice, ordenados alfabeticamente, ou por algum outro esquema organizacional; ou em formato de guia, geralmente como um tutorial de utilização das funcionalidades disponíveis ou como um passo-a-passo. A Figura 9 apresenta exemplos de aplicação destes sistemas.

Figura 9 – Exemplos de aplicação dos sistemas de navegação.



Fonte: Adaptado de Morville e Rosenfeld (2006, tradução nossa).

### 3.4.2.3 Sistemas de busca

Nielsen (2000) aponta que a maioria dos usuários de *websites*, quando estão em busca de alguma informação específica, tem o costume de ir diretamente à caixa de pesquisa e escrever o termo desejado. Por isso o sistema de busca de um *website* deve ser muito bem planejado e apresentado, para que o usuário não fique frustrado com a sua utilização.

O sistema de busca de um *website* é considerado um sistema de navegação auxiliar, porém possui maior importância por ser a forma mais “poderosa” de navegação, uma vez que o usuário tem a possibilidade de expressar sua própria nomenclatura para diferentes assuntos, e espera encontrar o que procura.

Nielsen (2000) ainda cita alguns cuidados que devem ser levados em consideração quanto ao projeto de um sistema de busca:

- Os usuários podem sentir-se perdidos a qualquer momento navegando por um *website*, portanto é de suma importância que a caixa de pesquisa esteja sempre disponível em qualquer contexto;
- Buscas com escopo, ou relacionadas a um contexto específico, tendem a apresentar melhores resultados, porém deve-se identificar com clareza este tipo de busca para não confundir o usuário de que não se trata de uma busca global;
- Os resultados da busca devem ser apresentados de forma a mostrar quais páginas possuem aquele termo, ordenadas pelo nível de relevância e com uma breve descrição da página.

### 3.4.3 *Design da informação*

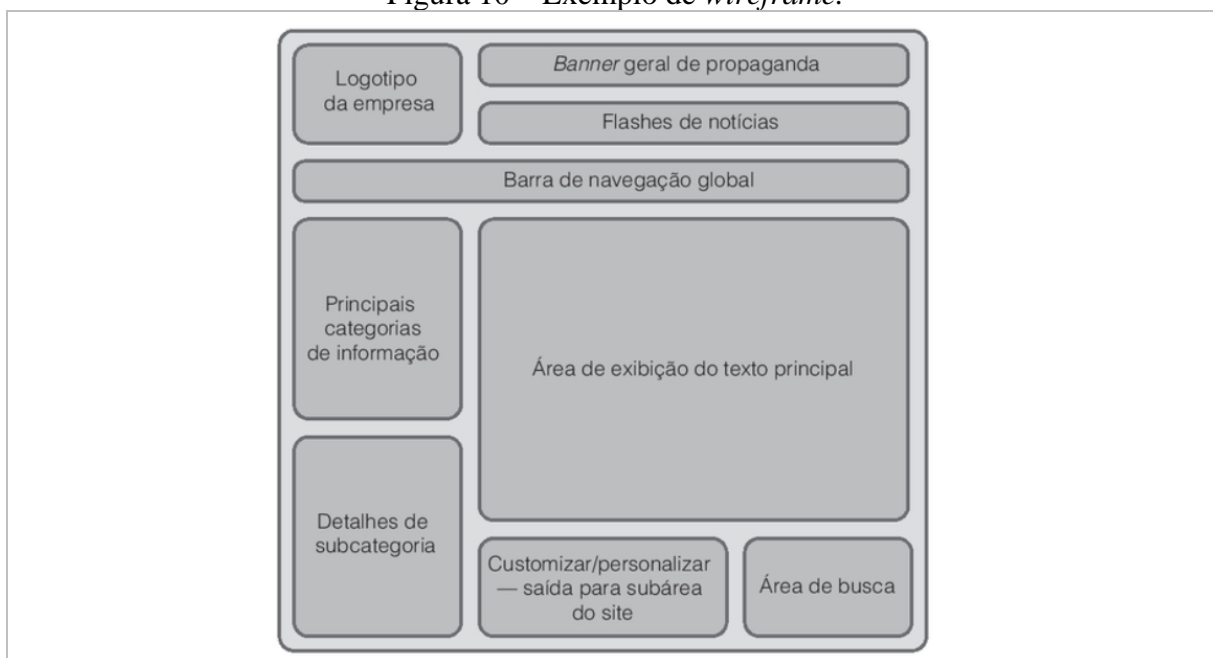
Comum às duas abordagens, tem-se o *design* da informação, que visa manter uma organização coesa dos elementos que compõem a interface apresentada para o usuário, sempre priorizando que a apresentação destes conteúdos seja a mais eficaz possível (GARRETT, 2011).

É nesta etapa também que define-se o *wireframe* do *website*. O *wireframe* é um protótipo de *layout*, composto basicamente por formas geométricas e linhas, sem definições de cores e conteúdo textual. É nele que são definidas as disposições dos elementos da página de acordo com sua importância. Garrett (2011, p. 129, tradução nossa) também explica que “[...] o diagrama de arquitetura que vimos no plano da estrutura é a grande visão do projeto; aqui no



plano do esqueleto, o *wireframe* é o documento detalhado que mostra exatamente como essa visão será preenchida”. A Figura 10 mostra um exemplo de *wireframe*.

Figura 10 – Exemplo de *wireframe*.



Fonte: Benyon (2011).

### 3.5 SUPERFÍCIE

Finalmente, no topo dos elementos de UX, tem-se a camada da Superfície. É aqui onde os usuários realmente visualizam a aplicação e interagem com ela a partir dos elementos gráficos, que devem estar visualmente adequados conforme os princípios do *design* visual.

Vale ressaltar que na obra de Garrett (2011) a camada da Superfície visa projetar o *design* sensorial do *website*. Porém na versão original do Diagrama dos Elementos de Experiência do Usuário de Garrett (2000), o foco desta camada é na aplicação do *design* visual, e é esta abordagem que será definida, uma vez que ela se adequa melhor ao projeto de *website* que será proposto com este trabalho.

O *design* visual baseia-se na aplicação dos princípios de *design*, desenvolvidos por Norman (2013). Estes princípios tratam da forma como o usuário interage com a interface visual da aplicação, considerando a facilidade de utilização, de aprendizagem, de adaptabilidade, entre outros.

- *Affordance*: refere-se à capacidade de que um objeto ou elemento possui de permitir que as pessoas saibam como utilizá-lo de forma intuitiva e de acordo com experiências vividas no cotidiano;
- *Signifiers*: é uma forma de comunicar qual a maneira mais adequada que se deve fazer o uso de um objeto ou funcionalidade, por meio de algum sinal, seja ele textual, com a utilização de imagens ou sons;
- Mapeamento: técnica utilizada para demonstrar relação entre dois objetos ou elementos distintos;
- Consistência: maneiras semelhantes de realizar tarefas parecidas;
- Visibilidade: os controles ou funcionalidades disponíveis precisam estar acessíveis para utilização;
- *Feedback*: forma de comunicação utilizada para dar uma resposta sobre a ocorrência de uma ação;
- Restrições: limitar a interação que pode ser feita em certos momentos;
- Modelo conceitual: explicação bastante resumida sobre o funcionamento de algum objeto ou funcionalidade.

Preece, Rogers e Sharp (2005) ainda citam os princípios de usabilidade, que possuem semelhanças aos de *design*, porém são mais utilizados para fins avaliativos dos *layouts*, chamados de heurísticas. Nielsen (1995) elenca dez princípios de usabilidade:

- Visibilidade do status do sistema: o sistema deve sempre informar o que está acontecendo no momento. Semelhante ao princípio do *feedback*;
- Compatibilidade do sistema com o mundo real: o sistema deve utilizar uma linguagem apropriada para o público que o utiliza;
- Controle do usuário e liberdade: os usuários precisam ter a liberdade de errar e desfazer esses erros de forma prática;
- Consistência e padrões: o sistema deve manter um padrão de nomenclatura e convenções para que o usuário não perca tempo tentando assemelhar termos diferentes, que significam a mesma coisa;
- Ajuda os usuários a reconhecer, diagnosticar e recuperar-se de erros: as mensagens de erro devem ser coesas e precisam auxiliar o usuário a entender como corrigi-lo;

- Prevenção de erros: tentar impedir que ocorram ações não desejadas pelo usuário. Sugere-se mostrar uma caixa de confirmação para realizar ações críticas;
- Reconhecimento ao invés de memorização: as opções e funcionalidades devem estar visíveis para o usuário de modo intuitivo, evitando que ele precise decorar onde e como o sistema funciona. Semelhante ao princípio da visibilidade;
- Flexibilidade e eficiência de uso: disponibilizar funções que usuários mais experientes podem usar para agilizar processos, enquanto que usuários inexperientes podem continuar usando normalmente;
- Estética e *design* minimalista: evitar utilizar muita informação para compor as páginas;
- Ajuda e documentação: disponibilização de guias e tutoriais com explicações mais detalhadas sobre as funcionalidades.

Ainda há no *design* visual a questão de como manter um apelo visual que identifique o produto e seja adequado para que o usuário se sinta confortável em navegar no *website*. A definição de uma paleta de cores que harmonize e auxilie na identificação da organização ao qual o *website* pertence é um dos pontos a serem considerados nesta etapa. A tipografia utilizada também precisa ser condizente com a marca da organização, porém deve-se estar ciente que em blocos de texto com mais quantidade de palavras é necessário optar por fontes que melhor se adequem à leitura (GARRETT, 2011).

### 3.6 A EXPERIÊNCIA DO USUÁRIO EM PWA

Os elementos de experiência do usuário estudados neste capítulo são adequados para qualquer aplicação *web*. Sendo assim, aplica-se também ao desenvolvimento de PWAs. Ater (2017) porém, especifica alguns padrões que melhoram a experiência do usuário para PWAs.

PWAs desejam passar confiança para seus usuários, da mesma forma que os aplicativos nativos. Para se alcançar isso é preciso saber gerenciar a comunicação entre a aplicação e o usuário. O usuário precisa estar ciente do que está acontecendo na aplicação, como os dados estão sendo utilizados e que tipos de permissões são necessárias para o funcionamento da mesma. Tudo isso deve ser transcrito e comunicado ao usuário, que se sentirá seguro ao saber sobre estas informações. A mesma abordagem serve também para momentos de falta de conexão com a internet, onde é preciso deixar explícito que a página e os dados que estão sendo

acessados naquele momento podem estar desatualizados, uma vez que os mesmos estavam armazenados em *cache*. Apelos visuais, como desabilitar botões ou modificar seu texto para alertar o usuário, também são formas interessantes de lidar com a falta de conectividade.

Outra abordagem que precisa ser estrategicamente projetada é a forma de captar usuários que estejam dispostos a cadastrarem-se para receber notificações da aplicação. O modelo padrão de solicitação de envio de notificações disponibilizadas pelos navegadores não é suficientemente claro quanto ao o que realmente faz e nem é esteticamente apropriado. Todos esses pontos reduzem as chances de os usuários efetivamente clicarem na permissão de cadastro de envio de notificação. Nesse caso sugere-se criar uma forma diferente de comunicação, contando com mensagens customizadas, que melhor expliquem quais os benefícios do recebimento de notificações, além de fazer com que esta sugestão apareça em um determinado momento, que capture a atenção do usuário para sua utilidade em um momento de necessidade.

Semelhante à forma de captação de usuários para o envio de notificações tem-se a caixa de permissão para adicionar a aplicação na tela inicial do dispositivo. Por padrão, o navegador decide quando mostrar esta caixa (levando em consideração o tempo de uso da aplicação), e por isso pode aparecer em algum momento não apropriado, levando o usuário a se frustrar. Uma forma de resolver isso é interceptando que a caixa apareça e só mostrá-la quando realmente for necessário, ou quando o próprio usuário demonstrar interesse.

Uma vez que a aplicação estiver disponível na tela inicial do dispositivo é necessário garantir que seu desempenho seja tão bom quanto os outros aplicativos nativos ali presentes. Para isso, foram definidas algumas diretrizes utilizadas para determinar o desempenho de uma aplicação. É o chamado RAIL, acrônimo de *response*, *animation*, *idle* e *load*:

- *Response* ou resposta: quando o usuário executa alguma ação – geralmente um clique em um botão – é preciso demonstrar de alguma forma que a ação foi percebida pela aplicação. Mostrar algum tipo de ícone com animação de carregamento é uma forma básica, porém bastante eficaz de lidar com essa situação;
- *Animation* ou animação: precisam estar de acordo com a percepção humana de movimento suave. Geralmente utilizado em transações de páginas, apresentação de dados e na rolagem da página;
- *Idle* ou momento ocioso: garantir que outros processos que estão sendo executados como forma de suporte à navegação não atrapalhem a performance

da funcionalidade que o usuário está utilizando no momento e utilizar os momentos ociosos para que isso seja feito;

- *Load* ou carregamento: tentar ao máximo que o carregamento dos dados da aplicação seja o mais rápido possível. Caso não for possível mostrar todos os dados de uma só vez, deve-se mostrá-los em partes de forma que o usuário tenha a percepção de um carregamento rápido.

### 3.7 CONSIDERAÇÕES SOBRE O CAPÍTULO

As abordagens citadas neste capítulo visam mostrar que uma boa experiência do usuário necessita que várias etapas sejam desenvolvidas, levando em consideração, principalmente, as necessidades dos usuários. Decisões acertadas desde as camadas iniciais facilitam o andamento do projeto entre todos os envolvidos, uma vez que definições explícitas tornam o trabalho em equipe muito mais integrado.

A preocupação com um bom desempenho das aplicações *web* também foi citado neste capítulo. Quanto mais rápido for o carregamento dos elementos da aplicação, maiores são as chances de o usuário continuar a navegar em um *website*. E esse deve ser um dos pontos a serem focados no desenvolvimento da aplicação: manter o usuário navegando no website de forma satisfatória e sem frustrá-lo.

## 4 PROPOSTA DE IMPLEMENTAÇÃO DO PWA WANDERLUST

Este trabalho visa o desenvolvimento de uma aplicação *web* progressiva em formato de rede social e com caráter colaborativo. A aplicação, denominada Wanderlust, voltada para a área do Turismo, tem como objetivo disponibilizar ferramentas – como Diário de Viagens e mapas – que permitam aos usuários a criação de roteiros de viagens com base nas recomendações e notas compartilhadas pelos seus amigos ou seguidores.

Primeiramente foi feita uma pesquisa de sistemas semelhantes, para definições de funcionalidades e levantamento de requisitos. Após, foi feita a modelagem do processo, com base no referencial teórico levantado neste trabalho, com foco na experiência do usuário.

### 4.1 SISTEMAS SEMELHANTES

Para auxiliar no levantamento dos requisitos do sistema, fez-se necessário pesquisar e analisar sistemas que tem como foco entregar serviços destinados ao Turismo e afins.

- TripAdvisor (<https://www.tripadvisor.com.br/>): é um *website* que oferece diversos tipos de serviços que englobam várias etapas de uma viagem. É possível “criar uma viagem” e adicionar locais e atividades de interesse do usuário, a fim de montar uma agenda com a programação das visitas. Pode-se também fazer reservas de voos e de hotéis, além de realizar a compra de pacotes de excursões e ingressos. Por fim, é possível avaliar os locais visitados, atribuindo uma nota e descrevendo como foi a experiência. Com isso o usuário acumula pontos que determinam o seu nível de colaborador, apresentando uma característica de gamificação;
- Guia mTrip (<https://www.mtrip.com/pt/guia-viagem/>): é um aplicativo gratuito disponível nas plataformas iOS e Android. Ele disponibiliza o download de guias turísticos das cidades cadastradas, tornando esses dados disponíveis mesmo offline. Para cada cidade pode-se navegar pelo guia, adicionar locais ao itinerário e criar um diário com fotos e descrições. Na sua versão paga são disponibilizadas ainda mais cidades para consulta, acesso *offline* aos mapas da cidade e acesso à ferramenta Trip Genius, que automaticamente monta um itinerário com base nas preferências do usuário;

- Foursquare (<https://pt.foursquare.com/>): é uma rede social disponível tanto em versão *web* como para aplicativos nas plataformas iOS, Android e Windows Phone. Nele é possível seguir pessoas de interesse do usuário, que compartilham dicas de locais para visitar, além de incluir diversas listas, como a relação dos locais mais bem avaliados nas proximidades, estabelecimentos abertos no momento e ranking de preços. Todos os locais são apontados em um mapa e as pesquisas podem ser realizadas a partir da identificação de uma cidade ou país.

## 4.2 MODELAGEM

A modelagem do projeto foi feita tendo como base o *framework* dos elementos de experiência do usuário de Garrett, explicado no capítulo 3 deste trabalho.

### 4.2.1 Estratégia

O objetivo da aplicação é disponibilizar aos usuários ferramentas que facilitem a busca de locais de interesse, a partir de recomendações feitas em grupos de amigos ou por pessoas influentes.

O público-alvo deste produto são pessoas interessadas em realizar viagens e que, a partir da recomendação de pessoas com interesses semelhantes, possam decidir com mais precisão quais locais se adequariam melhor aos seus interesses. Uma pesquisa realizada com uma amostra de 66 pessoas auxiliou na identificação dos diferentes perfis do público-alvo. O questionário utilizado para a pesquisa, bem como os resultados obtidos, podem ser visualizados no Apêndice A deste trabalho. As personas apresentadas na Figura 11 foram criadas com base nos resultados obtidos a partir desta pesquisa.

Figura 11 – Personas que representam o público-alvo da aplicação que está sendo proposta.



Fonte: Elaborado pela autora.

#### 4.2.2 Escopo

A partir da análise dos sistemas semelhantes e das necessidades das personas, foram levantados os requisitos funcionais apresentados no Quadro 4.



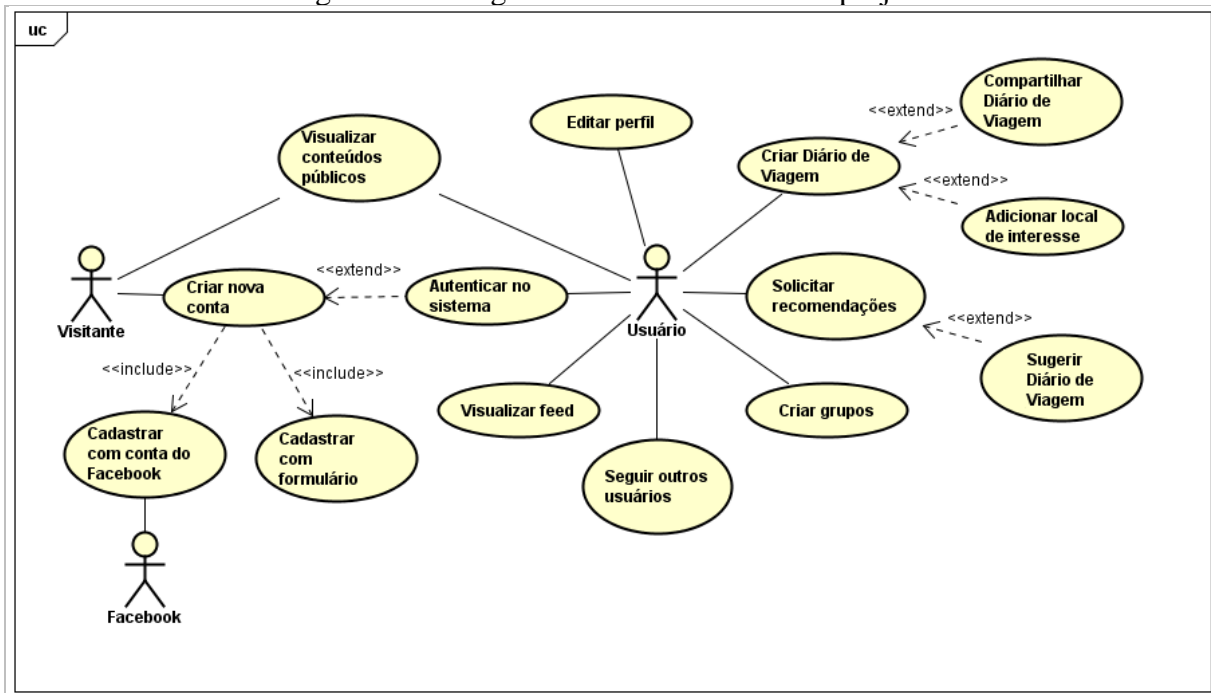
Quadro 4 – Requisitos funcionais do projeto.

|     |   |
|-----|---|
| R1  | Permitir que os usuários façam um cadastro para acessar o sistema.  |
| R2  | Permitir que o usuário tenha uma página descrevendo seu perfil.   |
| R3  | Permitir que usuários visitantes tenham acesso ao conteúdo público.   |
| R4  | Permitir que o usuário possa criar um Diário de Viagem pessoal, que agrupe locais de interesse.   |
| R5  | Permitir que o usuário adicione locais de interesse no Diário de Viagem, contendo o nome do local, descrição, indicação no mapa e fotos.                                |
| R6  | Permitir o compartilhamento das informações do Diário de Viagem com seguidores ou com grupos específicos de pessoas.  |
| R7  | Permitir que o usuário possa solicitar que outros usuários lhe enviem recomendações de locais interessantes de uma certa cidade ou de acordo com sua localização atual. |
| R8  | Permitir que os usuários sigam outros usuários, com o intuito de acompanhar suas atividades.  |
| R9  | Permitir a criação de grupos de usuários com interesses semelhantes.  |
| R10 | Permitir acesso <i>offline</i> a algumas funcionalidades da aplicação.  |
| R11 | Possuir uma página de <i>feed</i> , contendo as atualizações e atividades das pessoas e dos grupos que o usuário está seguindo.   |
| R12 | Permitir a classificação das pessoas que o usuário está seguindo. Exemplo: Amigo, Conhecido, Melhores Amigos.   |
| R13 | Usuário deve receber notificações quando houver um novo seguidor, novas solicitações ou recebimento de sugestões, ou quando for adicionado a um grupo.                  |

Fonte: Elaborado pela autora.

A Figura 12 apresenta o Diagrama de Caso de Uso do sistema, baseado nos requisitos funcionais elencados. O detalhamento de cada Caso de Uso, com a apresentação das telas desenvolvidas, pode ser visualizado nos Apêndices B, C, D, E, F, G, H, I e J.

Figura 12 – Diagrama de Casos de Uso do projeto.

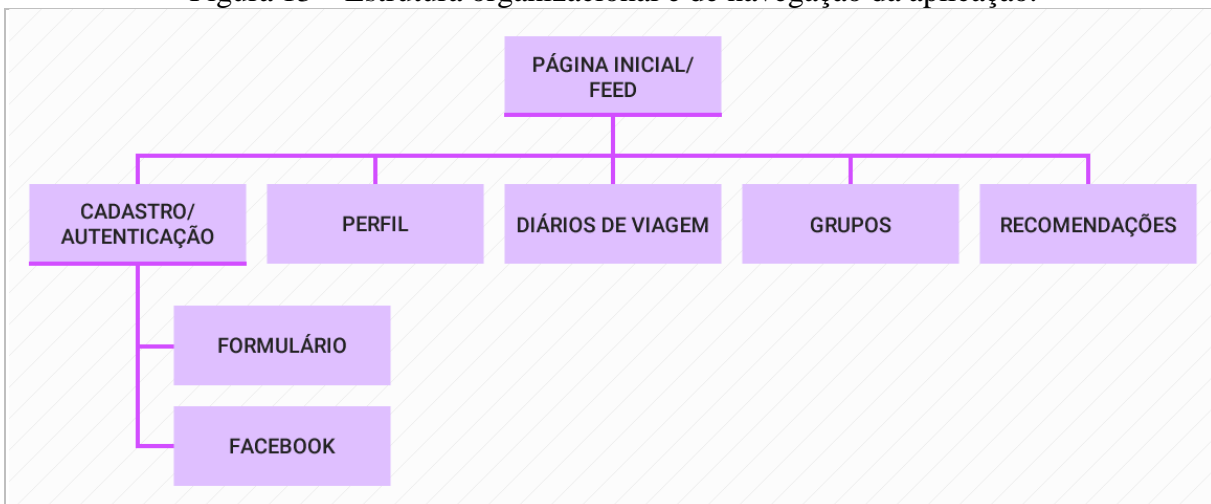


Fonte: Elaborado pela autora.

### 4.2.3 Estrutura

O sistema de navegação da aplicação é apresentado na Figura 13.

Figura 13 – Estrutura organizacional e de navegação da aplicação.

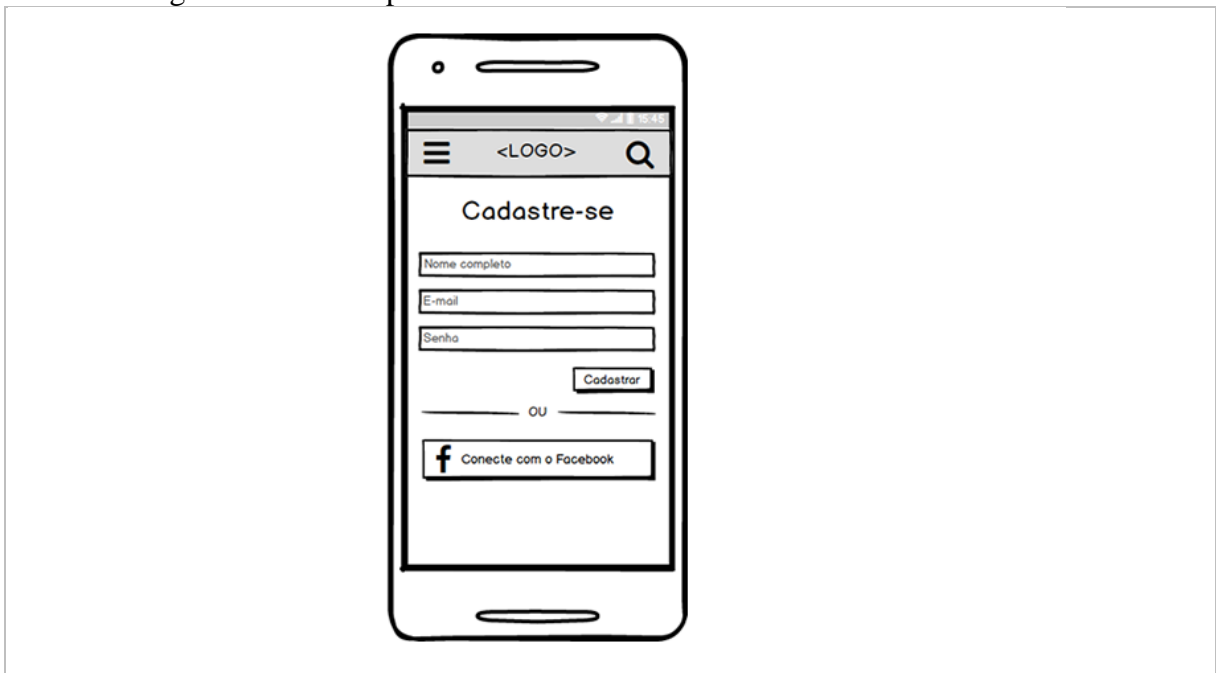


Fonte: Elaborado pela autora.

#### 4.2.4 Esqueleto

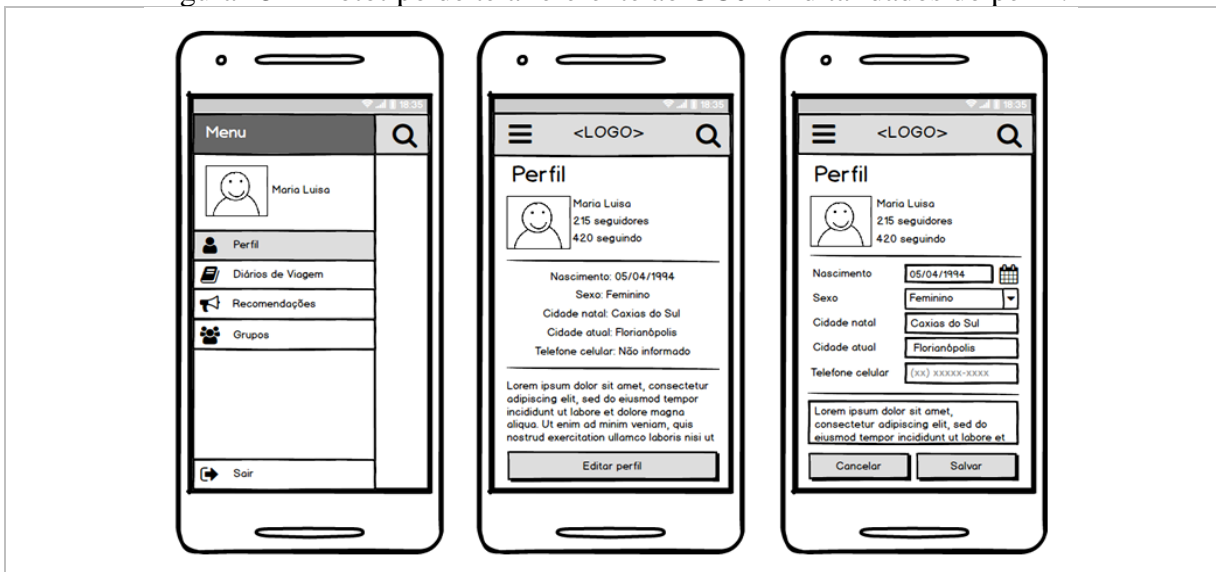
A seguir são apresentados os protótipos de tela (da Figura 14 à Figura 22), desenvolvidos a partir dos Diagramas de Casos de Uso estabelecidos para o projeto. O detalhamento dos Casos de Uso pode ser visualizado nos Apêndices B, C, D, E, F, G, H, I e J deste trabalho.

Figura 14 – Protótipo de tela referente ao UC01: Cadastrar novo usuário.



Fonte: Elaborado pela autora.

Figura 15 – Protótipo de tela referente ao UC02: Editar dados do perfil.



Fonte: Elaborado pela autora.

Figura 16 – Protótipo de tela referente ao UC03: Criar Diário de Viagem.



Fonte: Elaborado pela autora.

Figura 17 – Protótipo de tela referente ao UC04: Adicionar um local no Diário de Viagem.



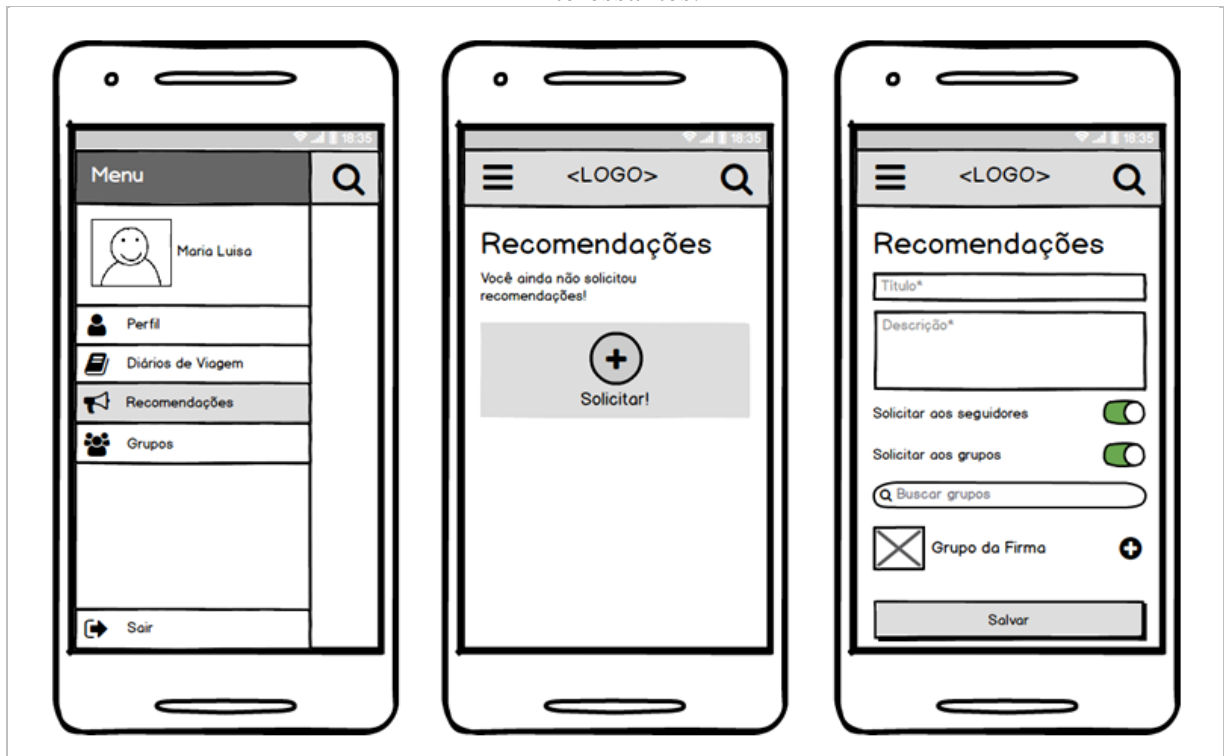
Fonte: Elaborado pela autora.

Figura 18 – Protótipo de tela referente ao UC05: Compartilhar Diário de Viagem.



Fonte: Elaborado pela autora.

Figura 19 – Protótipo de tela referente ao UC06: Solicitar recomendações de locais interessantes.



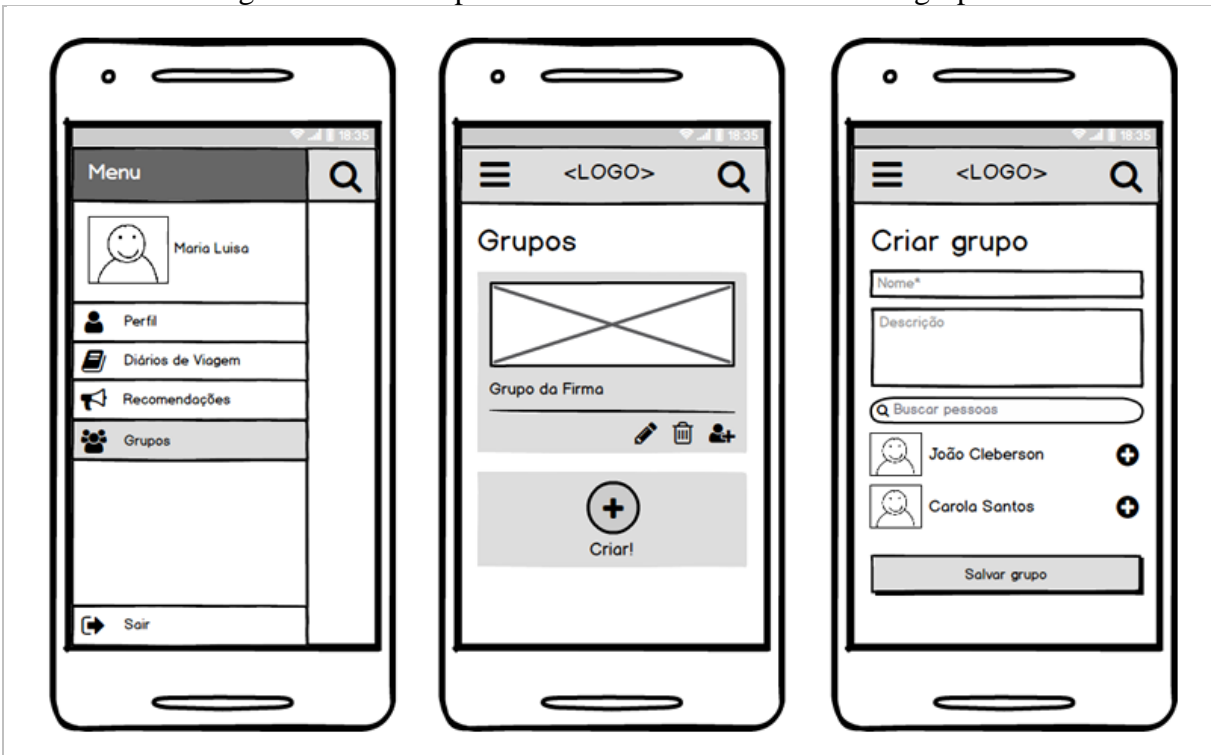
Fonte: Elaborado pela autora.

Figura 20 – Protótipo de tela referente ao UC07: Sugerir Diário de Viagem.



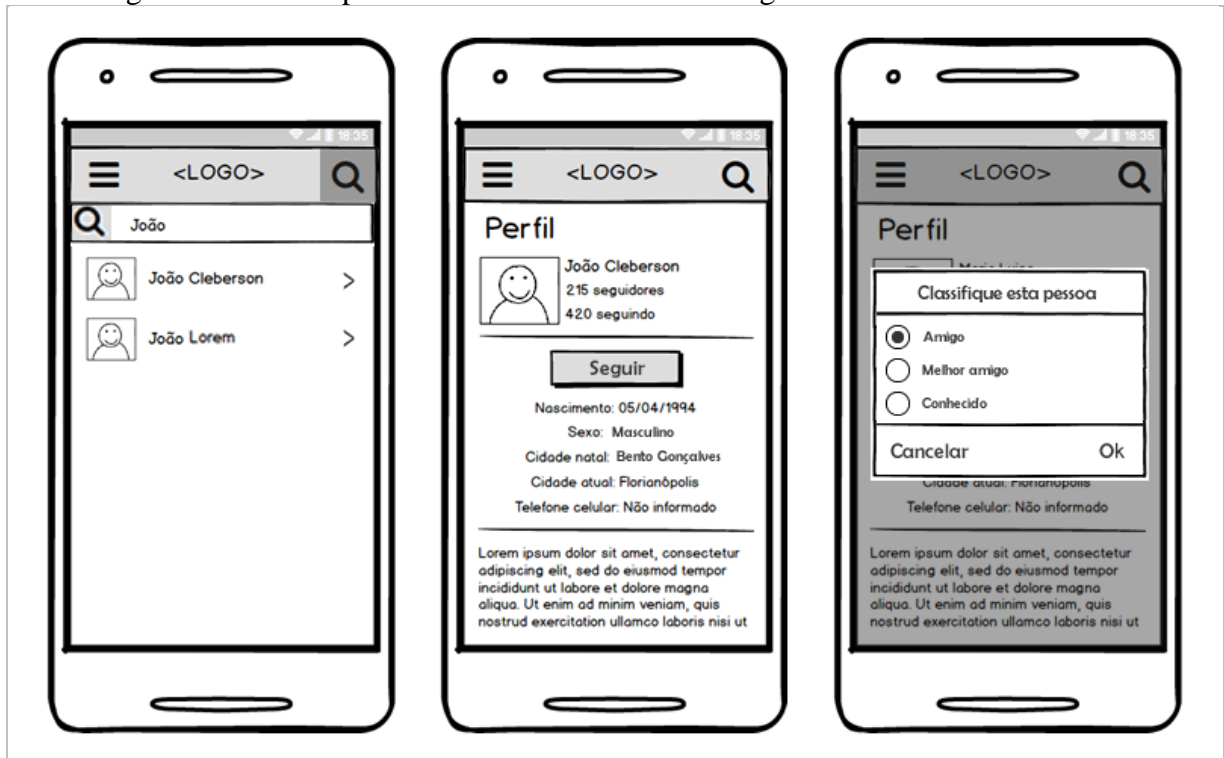
Fonte: Elaborado pela autora.

Figura 21 – Protótipo de tela referente ao UC08: Criar grupos.



Fonte: Elaborado pela autora.

Figura 22 – Protótipo de tela referente ao UC09: Seguir e classificar um usuário.



Fonte: Elaborado pela autora.

#### 4.2.5 Superfície

A Superfície do projeto será implementada a partir das definições do Material Design<sup>12</sup>, disponibilizado pelo Google. O Material Design fornece um guia com diretrizes padronizadas para elementos de interface, englobando questões como cores, formas de apresentar botões, animações, formatos, entre outros.

#### 4.3 ARQUITETURA E TECNOLOGIAS

A arquitetura de software desta aplicação é dividida em três camadas distintas e independentes que devem interagir entre si. A Figura 23 demonstra graficamente como é realizada esta interação, bem como os componentes e tecnologias utilizados em cada uma das camadas.

<sup>12</sup> <https://material.io/>

Figura 23 – Arquitetura em três camadas.



Fonte: Elaborado pela autora.

### 4.3.1 Camada de Apresentação

Na Camada de Apresentação, encontram-se os componentes que são executados nos dispositivos dos clientes. Nesta camada o usuário tem acesso a interface da aplicação, onde é feita a interação com a camada lógica (IBM, 2014).

No primeiro capítulo deste trabalho foi estabelecida a utilização do *framework* Polymer para construção da Camada de Apresentação. Porém recentemente houve o lançamento de uma nova versão do Ionic Framework, com foco em desenvolvimento de PWA e com uma documentação robusta. Dessa forma optou-se por modificar a ferramenta original para o desenvolvimento desta camada.

O Ionic Framework<sup>13</sup> é uma ferramenta de código aberto que disponibiliza diversos componentes de interface para o desenvolvimento de aplicações *web*, utilizando-se de tecnologias como HTML, CSS e JavaScript. Esses componentes possuem formas de interação

<sup>13</sup> <https://ionicframework.com/>



e animações que são voltados para entregar uma boa experiência para o usuário da aplicação (IONIC, 2018).

O Ionic possui integração com Angular<sup>14</sup>, que é “uma plataforma de aplicações web de código-fonte aberto e *front-end* baseado em TypeScript” (ANGULAR, 2018). A utilização do Angular em projetos Ionic serve principalmente para as injeções de dependência de bibliotecas – que são utilizadas nas requisições AJAX e para a sincronização do Google Maps, por exemplo – e para outras partes de interação como a navegação e gerenciamento de URLs. A Figura 24 exibe um trecho de código demonstrando a utilização do Angular para manipulação dos componentes do Ionic e demais interações.

### 4.3.2 Camada de Negócio

Na Camada de Negócio encontra-se toda a estrutura lógica da aplicação. Esta camada serve como uma “ponte” entre a interface da aplicação e o banco de dados, sendo ela a responsável por processar as requisições vindas do cliente, manipular os dados necessários e retorná-los (IBM, 2014).

Figura 24 – Trecho de código em Angular.

```

diarios.page.html
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>Diários de Viagem</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button (click)="criarNovoDiario()">
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
  <ion-grid>
    <ion-row *ngIf="!diarios?.length">
      <ion-col size-xs="12">
        
        <p class="text-center">
          Não encontramos nenhum Diário de Viagem para mostrar.
          Que tal criar um?
        </p>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size-md="4" size-xs="12" *ngFor="let diario of diarios">
        <ion-card class="card-item" (click)="acessarDiario(diario.id)">
          <ion-card-header>
            <ion-icon name="image"></ion-icon>
          </ion-card-header>
          <ion-card-content>
            <ion-card-title>{{ diario.titulo }}</ion-card-title>
          </ion-card-content>
        </ion-card>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>

diarios.page.ts
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AlertController } from '@ionic/angular';
import { DiarioService } from '../services/diario.service';

@Component({
  selector: 'app-diarios',
  templateUrl: './diarios.page.html',
  styleUrls: ['./diarios.page.scss'],
  providers: [DiarioService]
})
export class DiariosPage implements OnInit {

  constructor(
    public service: DiarioService,
    public alertController: AlertController,
    public router: Router
  ) {}

  public diarios: any;

  ngOnInit() {
    this.carregarDiarios();
  }

  public carregarDiarios() {
    this.service.getDiarios().then(response => {
      this.diarios = response;
    });
  }

  public acessarDiario(id) {
    this.router.navigate(['diarios', id]);
  }
}

```

Fonte: Elaborado pela autora.

<sup>14</sup> <https://angular.io/>

Utilizando o *framework* Django, escrito sobre a linguagem Python, foi criada uma API REST capaz de ser acessada pela Camada de Apresentação por meio de requisições AJAX.

Neste processo de desenvolvimento, utilizou-se a biblioteca Django Rest Framework<sup>15</sup>, que possui diversos métodos específicos para a criação de APIs REST. Entre eles destacam-se os métodos de autenticação, de validação dos dados e de tratamento de permissões.

### 4.3.3 Camada de Persistência

A Camada de Persistência tem a função de armazenar e indexar os dados gerados pela Camada de Negócios e outros demais dados utilizados como parametrização do sistema. O Django possui uma ferramenta de ORM, que auxilia no processo de gerenciamento do banco de dados. Esta técnica consiste em utilizar classes de programação orientada à objetos, onde a classe e suas propriedades tornam-se registros no banco de dados, não havendo a necessidade da utilização da linguagem SQL para o gerenciamento das mesmas. A Figura 25 mostra a criação de um modelo de classe e a sua manipulação, enquanto que a Figura 26 mostra o Diagrama de Classes gerado a partir destas definições.

Figura 25 – Código das classes Diário de Viagens e Local de Interesse e suas manipulações.

```

models.py
class Diário(models.Model):
    """
    Representação de um Diário de Viagens
    """
    autor = models.ForeignKey(Usuario,
                             related_name='diarios',
                             on_delete=models.CASCADE)
    titulo = models.CharField(max_length=100)

    def __unicode__(self):
        return self.titulo

class LocalDeInteresse(models.Model):
    """
    Representação de um Local de Interesse,
    que é associado a um Diário
    """
    diario = models.ForeignKey(Diário,
                              on_delete=models.CASCADE,
                              verbose_name='Diário',
                              related_name='locais_de_interesse')
    nome = models.CharField('Nome', max_length=100)
    descricao = models.TextField('Descrição', blank=True, null=True)
    latitude = models.FloatField('Latitude', blank=True, null=True)
    longitude = models.FloatField('Longitude', blank=True, null=True)
    foto = models.ImageField(upload_to='images',
                             blank=True,
                             null=True)

    def __unicode__(self):
        return self.nome

views.py
class DiárioViewSet(viewsets.ModelViewSet):
    """
    Viewset para criar um Diário de Viagem
    """
    queryset = Diário.objects
    list_serializer_class = DiárioSerializer
    detail_serializer_class = DetalheDiárioSerializer
    permission_classes = (permissions.IsAuthenticated, IsOwnerOrReadOnly)

    def perform_create(self, serializer):
        serializer.save(autor=self.request.user.usuario)

    def dispatch(self, request, pk=None):
        self.pk = pk
        return super(DiárioViewSet, self).dispatch(request, pk=pk)

    def get_queryset(self):
        queryset = super(DiárioViewSet, self).get_queryset()
        return queryset.filter(autor=self.request.user.usuario)

    def get_serializer_class(self):
        return self.list_serializer_class \
            if not self.pk else self.detail_serializer_class

    @action(methods=['POST'], detail=True, url_path='cria-local')
    def cria_local(self, request, pk=None):
        """
        Função para criar um Local de Interesse em um Diário
        """
        diario = self.get_object()
        serializer = LocalDeInteresseSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save(diario=diario)
            resposta = 'Local de Interesse criado com sucesso!'
            status_resposta = status.HTTP_200_OK
        else:
            resposta = serializer.errors
            status_resposta = status.HTTP_400_BAD_REQUEST

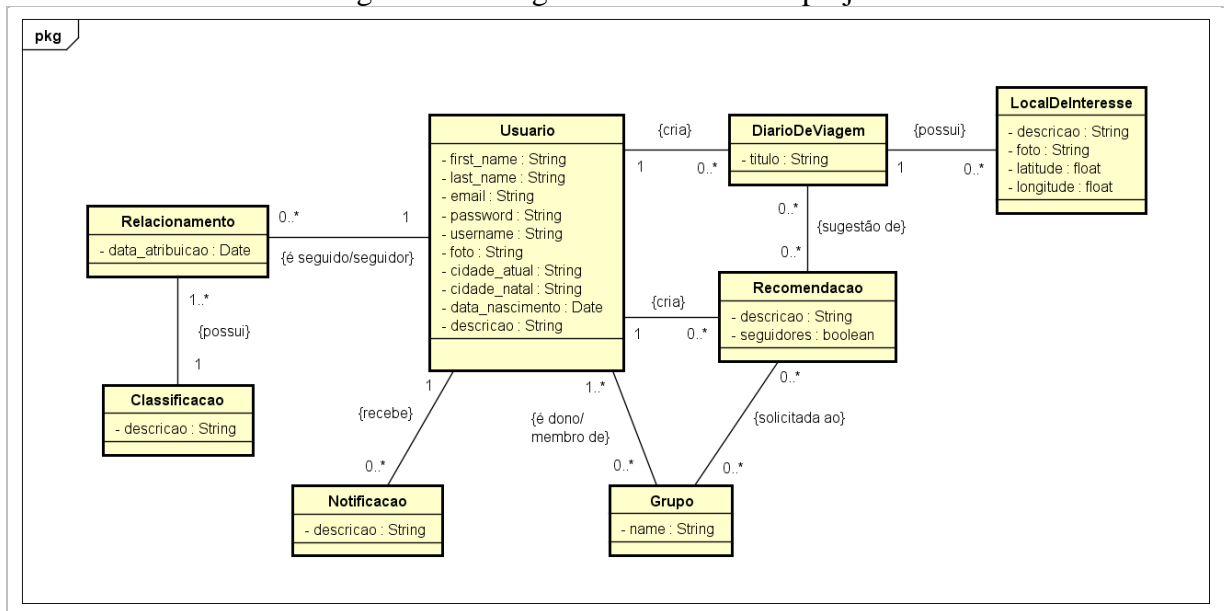
        return Response(resposta, status=status_resposta)

```

Fonte: Elaborado pela autora.

<sup>15</sup> <https://www.django-rest-framework.org/>

Figura 26 – Diagrama de Classes do projeto.



Fonte: Elaborado pela autora.

O Django permite integração com diversos bancos de dados. Devido a questões de desempenho e processamento de dados, optou-se por utilizar o banco MySQL ao invés do banco padrão do Django, o SQLite.

## 5 VALIDAÇÃO DO PROJETO

A aplicação Wanderlust está sendo servida no endereço <https://wanderlust-pwa.firebaseio.com>, e a API utilizada para o *back-end* está no endereço <https://lceconi.pythonanywhere.com/>. Os acessos para estes projetos estarão disponíveis para acesso até o final do ano de 2018.

Nesta seção será apresentada a validação do projeto, de acordo com as características de PWA elencadas no primeiro capítulo deste trabalho; com as características de UX voltadas para PWA na implementadas na aplicação; e com sua avaliação a partir da execução da ferramenta Lighthouse.

É necessário informar que optou-se por retirar do projeto o requisito “R3: Permitir que usuários visitantes tenham acesso ao conteúdo público”, com o intuito de diminuir a burocracia na criação e compartilhamento dos conteúdos, uma vez que seria necessário que o usuário deixasse explícito quais conteúdos gostaria de deixar público, toda vez que um novo conteúdo fosse criado.

### 5.1 CARACTERÍSTICAS DE PWA NA APLICAÇÃO

Todas as características de PWA foram contempladas pela aplicação. A seguir estão descritos como se deram os seus desenvolvimentos.

#### 5.1.1 Progressividade

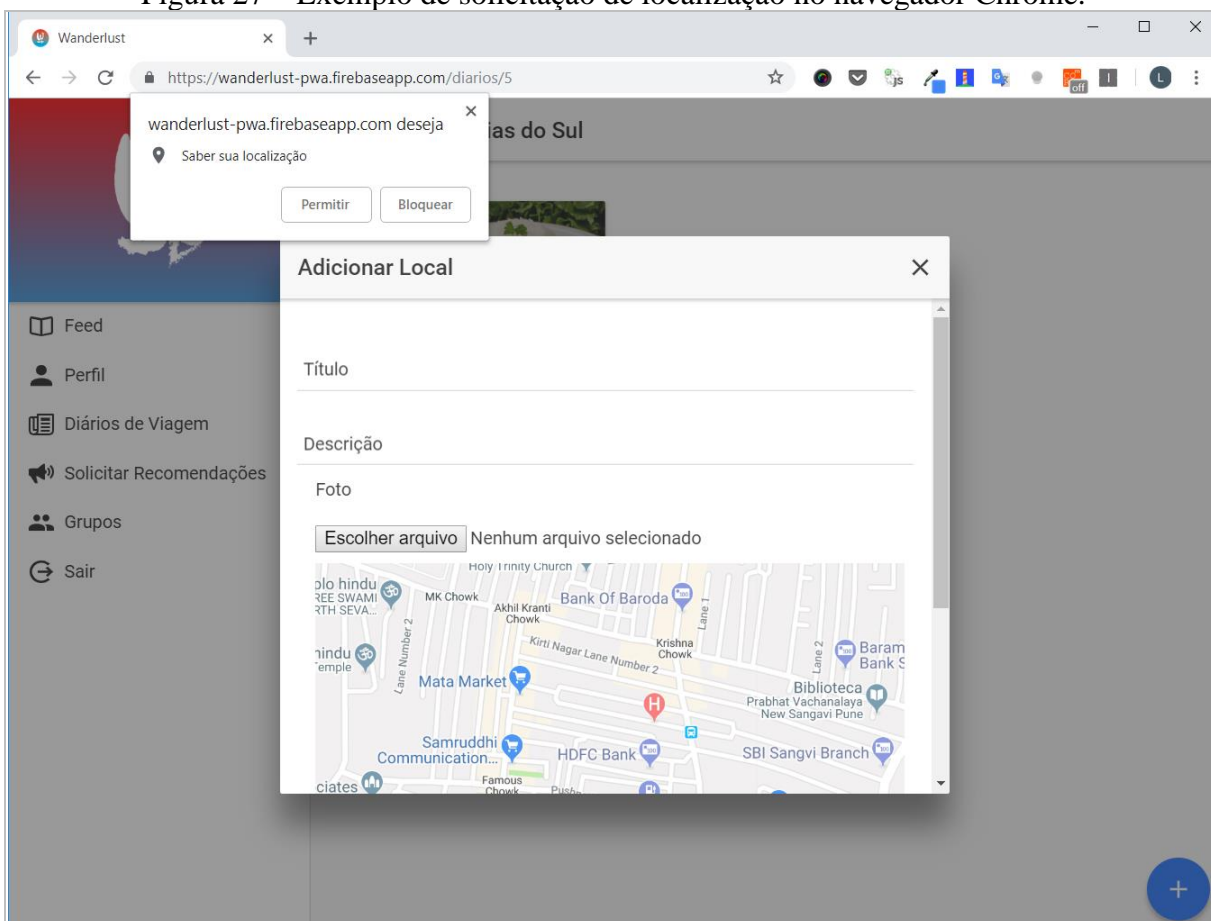
O Ionic Framework provê *scripts* em JavaScript chamados *polyfills* que adequam os métodos e estilos de CSS modernos para que sejam interpretados nos navegadores antigos. Apesar disso, algumas tecnologias dependentes de utilização dos *Service Workers* acabam não estando disponíveis, como o caso do funcionamento *offline* e das notificações.

Outra tecnologia que não está presente em navegadores antigos é a localização do dispositivo. A Figura 27 mostra que em navegadores mais modernos é solicitada a utilização da localização do dispositivo, fazendo com que o mapa seja renderizado próximo do local encontrado. Já nos dispositivos antigos o mapa sempre apresentará um local fixo.

É importante ressaltar que as funcionalidades da aplicação, como cadastros e consultas, podem ser executadas independente das tecnologias disponíveis no navegador. O não

funcionamento de certas tecnologias apenas impacta na facilidade de realizar certas ações, e não na perda de funcionalidades.

Figura 27 – Exemplo de solicitação de localização no navegador Chrome.



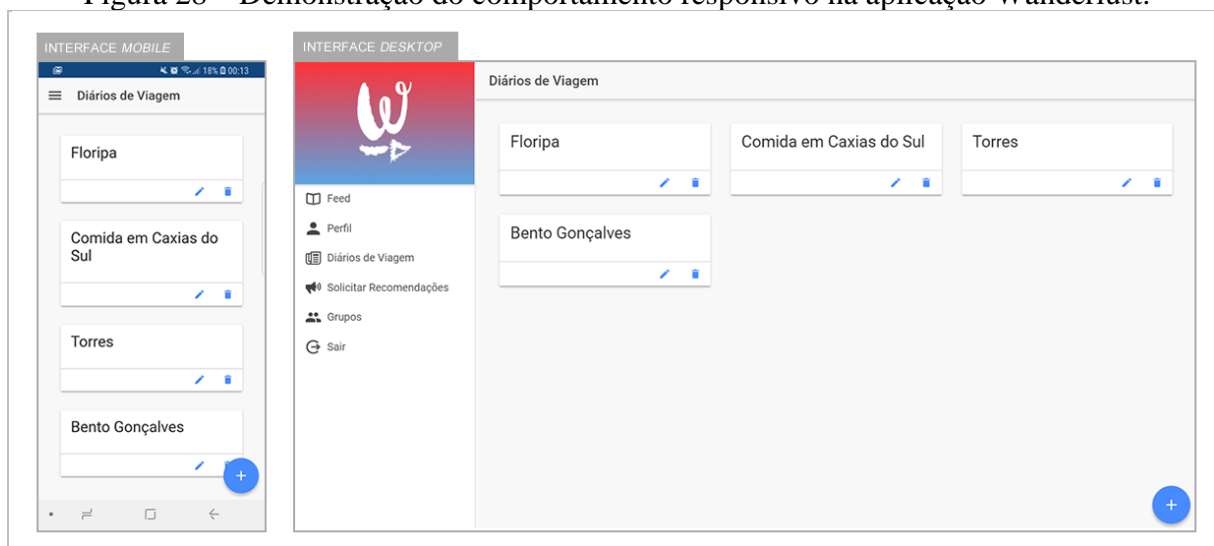
Fonte: Elaborado pela autora.

### 5.1.2 Responsividade

Para o desenvolvimento das interfaces responsivas, utilizou-se a biblioteca de CSS do Ionic Framework que disponibiliza estilos específicos para *grids*<sup>16</sup>. A aplicação inicialmente foi projetada para suprir as necessidades dos dispositivos menores (de acordo com a filosofia *Mobile First*), e a partir disso, adequou-se a interface para tamanhos de telas maiores. A Figura 28 demonstra o comportamento responsivo da aplicação.

<sup>16</sup> <https://beta.ionicframework.com/docs/layout/grid>

Figura 28 – Demonstração do comportamento responsivo na aplicação Wanderlust.

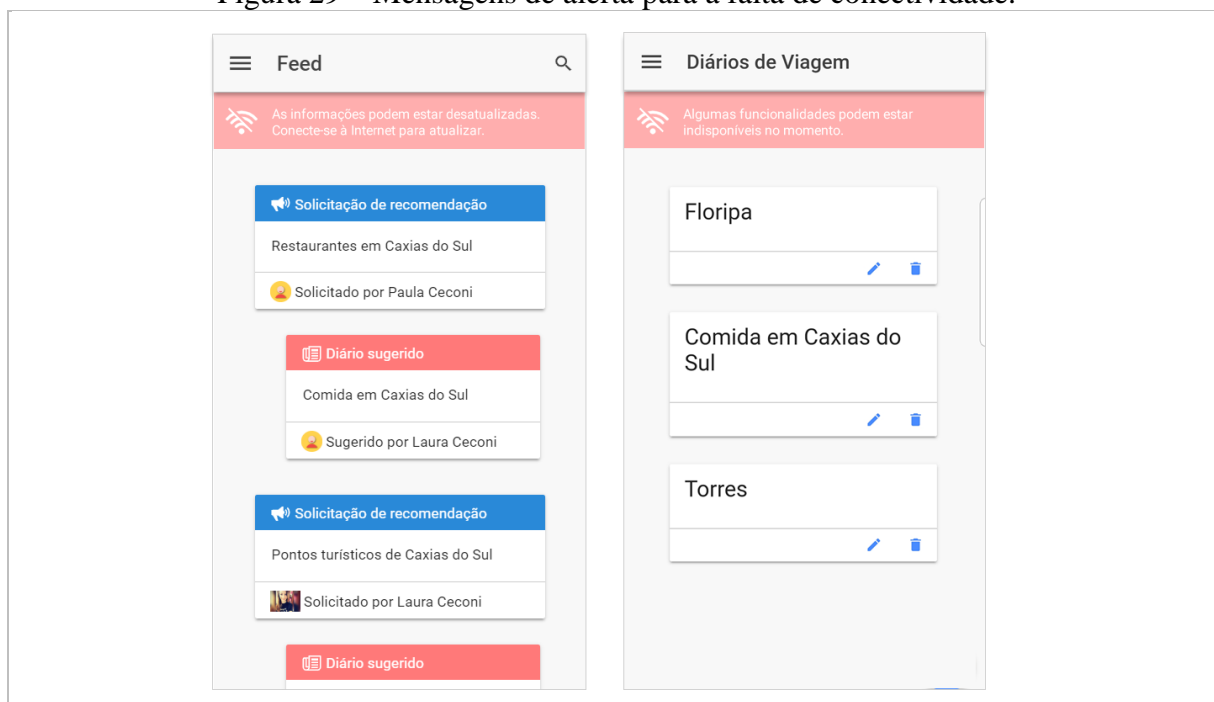


Fonte: Elaborado pela autora.

### 5.1.3 Independência de conectividade

A consulta às telas que já foram acessadas anteriormente fica disponível mesmo estando em modo *offline*. Funcionalidades que necessitam ser persistidas através da API ficam indisponíveis enquanto o dispositivo estiver *offline*, sendo exibida uma mensagem alertando o usuário sobre a limitação, conforme demonstrado na Figura 29.

Figura 29 – Mensagens de alerta para a falta de conectividade.



Fonte: Elaborado pela autora.

#### 5.1.4 Semelhança com aplicativos

A aplicação está desenvolvida sobre um *app shell* que faz com que a renderização dos elementos iniciais tenha um desempenho muito semelhante à de uma aplicação nativa, por não depender uma quantidade muito grande de processamento de códigos JavaScript e CSS. Além disso as transições de tela e efeitos de interação também se assemelham aos aplicativos nativos (na seção 5.2 deste trabalho serão abordados mais detalhes sobre os elementos de UX incluídos na aplicação).

#### 5.1.5 Sempre atualizados

O *script* de *Service Worker* implementado espera receber novas versões da aplicação e replicá-la aos demais usuários. Inclusive o Ionic Framework provê uma ferramenta que modifica o nome dos arquivos de JavaScript e CSS das páginas para cada nova atualização, garantindo que os navegadores entendam que aquele conteúdo foi modificado, e que não continue fazendo *cache* dos mesmos.

#### 5.1.6 Segurança

A aplicação está sendo servida pelo serviço de hospedagem do Firebase<sup>17</sup>, com certificado de segurança HTTPS. A Figura 30 mostra como o navegador Chrome apresenta este certificado.

#### 5.1.7 Descobríveis e instaláveis

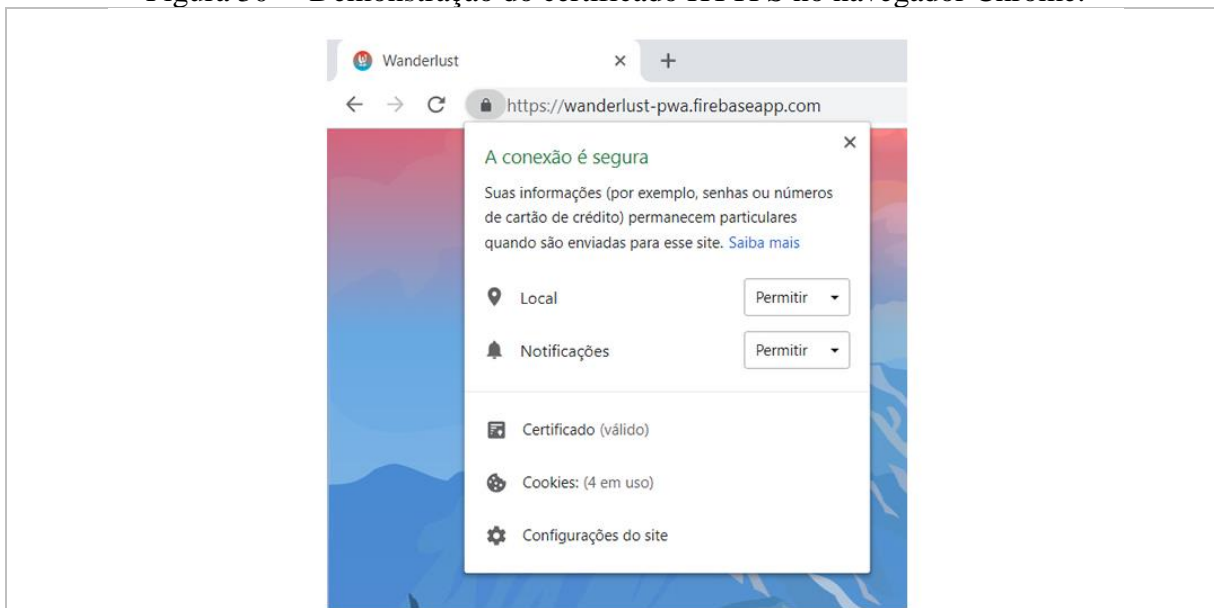
Para tornar a aplicação totalmente indexável pelos motores de busca, foi necessário incluir dois arquivos à raiz do projeto:

- *robots.txt*: neste arquivo estão contidas as URLs que devem ser indexadas ou não pelos motores de busca. Também faz referência ao mapa do site (aplicação), que contém todas as URLs do site.
- *sitemap.xml*: arquivo em XML que contém todas as URLs disponíveis para acesso pelo motor de busca.

---

<sup>17</sup> <https://firebase.google.com/>

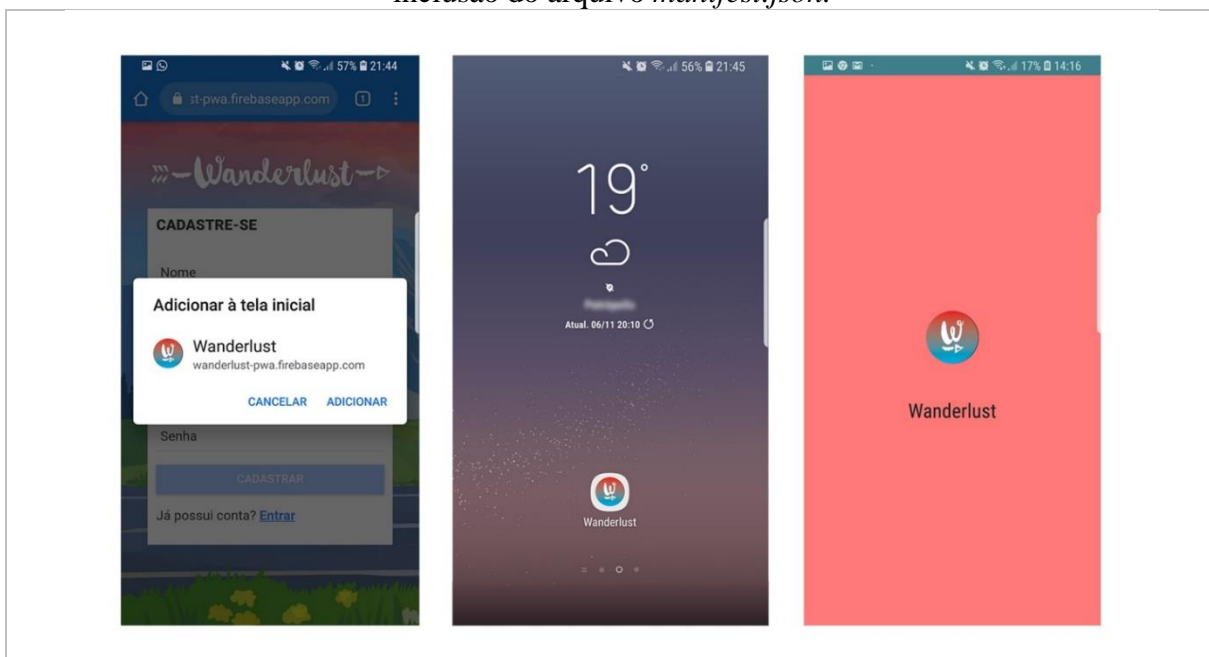
Figura 30 – Demonstração do certificado HTTPS no navegador Chrome.



Fonte: Elaborado pela autora.

No caso da aplicação Wanderlust somente uma URL foi indexada – a página inicial – uma vez que para acessar seu conteúdo é necessária a criação de um cadastro com e-mail e senha.

Também foi incluído o arquivo *manifest.json*, permitindo, assim, que a aplicação seja instalada no dispositivo e tenha um *splashscreen*. A Figura 31 apresenta estas funcionalidades.

Figura 31 – Funcionalidade de “Adicionar à tela inicial” e *splashscreen* disponibilizados pela inclusão do arquivo *manifest.json*.

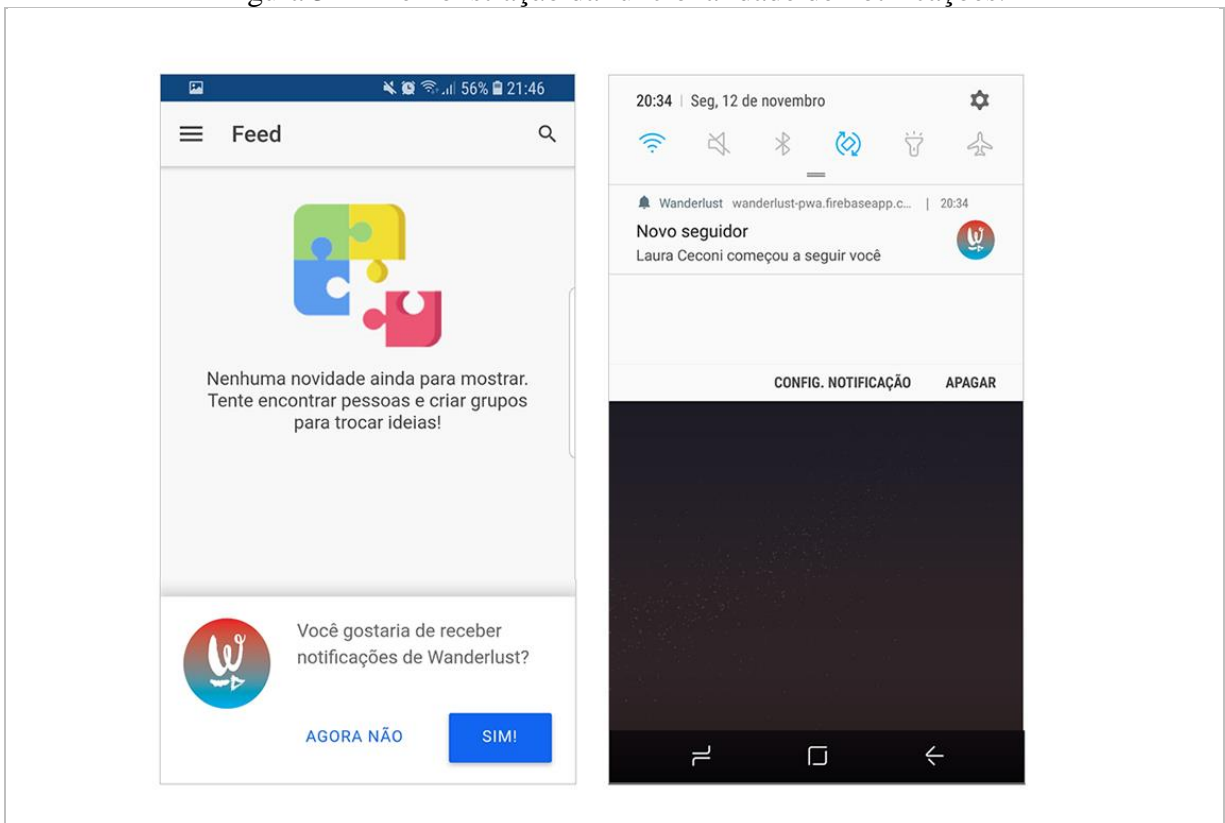
Fonte: Elaborado pela autora.



### 5.1.8 Re-envolventes

Para implementar as notificações foi utilizada a ferramenta OneSignal<sup>18</sup> que possibilita a criação e gerenciamento das notificações *web*. Além disso disponibiliza uma API, que foi integrado ao código Python da aplicação fazendo a geração e envio das notificações. A Figura 32 mostra o comportamento das notificações.

Figura 32 – Demonstração da funcionalidade de notificações.



Fonte: Elaborado pela autora.

### 5.1.9 Linkáveis

Cada página da aplicação possui uma URL única, facilitando o acesso às páginas.

<sup>18</sup> <https://onesignal.com/>

Quadro 5 – Relação de URLs da aplicação.

| Descrição da Página           | URL                             |
|-------------------------------|---------------------------------|
| Página inicial                | /                               |
| <i>Feed</i>                   | /feed                           |
| Perfil do usuário autenticado | /perfil                         |
| Perfil de outro usuário       | /perfil/:id                     |
| Diários                       | /diarios                        |
| Diário específico             | /diarios/:id_diario             |
| Local de Interesse            | /local/:id_local                |
| Solicitar Recomendações       | /recomendacoes                  |
| Solicitação de Recomendação   | /recomendacoes/:id_recomendacao |
| Grupos                        | /grupos                         |
| Grupo específico              | /grupos/:id_grupo               |

Fonte: Elaborado pela autora.

## 5.2 ELEMENTOS DE UX NA APLICAÇÃO

Nesta seção serão demonstradas como foram contempladas as diretrizes RAIL.

### 5.2.1 *Response* ou resposta

Cada interação do usuário que necessite comunicação com a API externa é representada por um *spinner*, demonstrando que os dados estão sendo processados e logo será retornada alguma informação, seja ela os dados que foram solicitados, ou uma mensagem de erro, representando o *feedback* para o usuário. Os elementos utilizados para representar o *feedback* podem ser vistos na Figura 33.

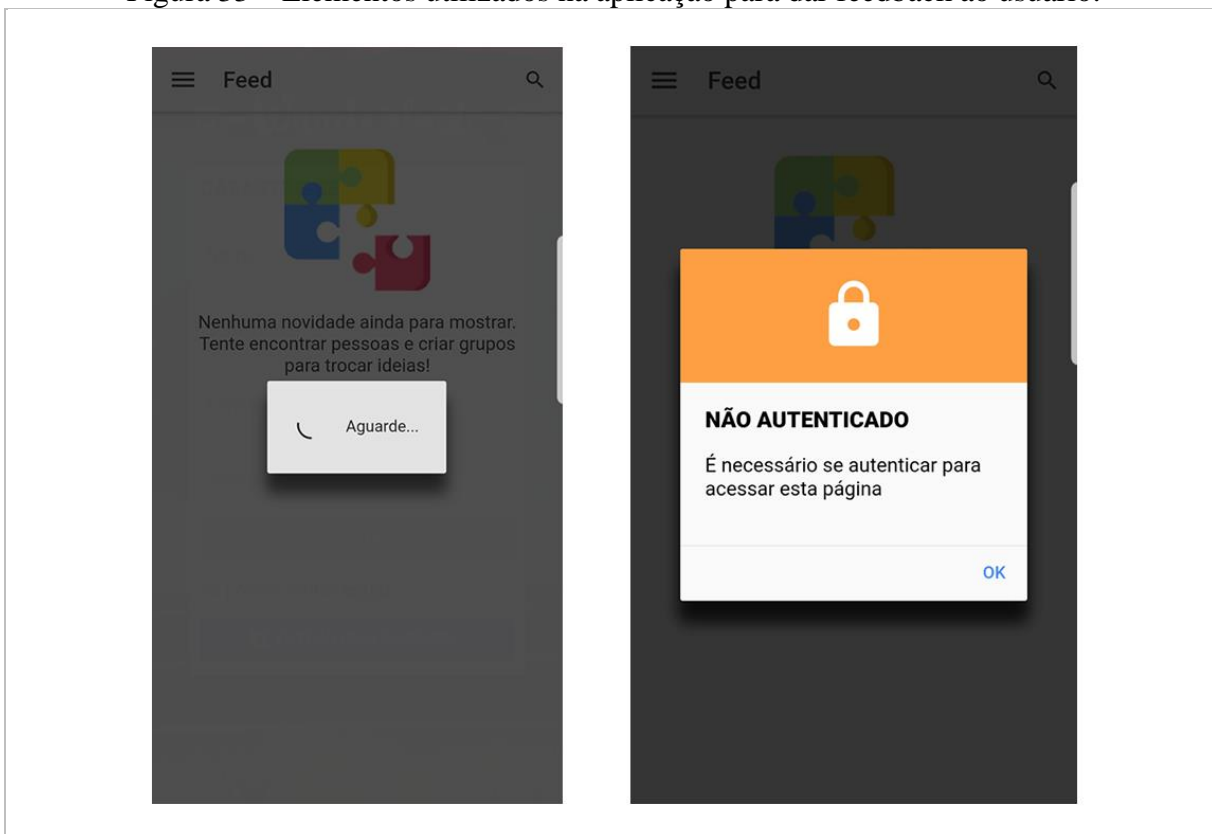
### 5.2.2 *Animation* ou animação

O Ionic Framework utiliza como base o manual sobre Movimento do Material Design<sup>19</sup> para as animações dos elementos. Na aplicação Wanderlust as animações são usadas para:

<sup>19</sup> <https://material.io/design/motion/understanding-motion.html>

transições de tela, abertura e fechamento do menu, apresentação de alertas/modals nos botões e nos formulários que possuem *labels* flutuantes.

Figura 33 – Elementos utilizados na aplicação para dar feedback ao usuário.



Fonte: Elaborado pela autora.

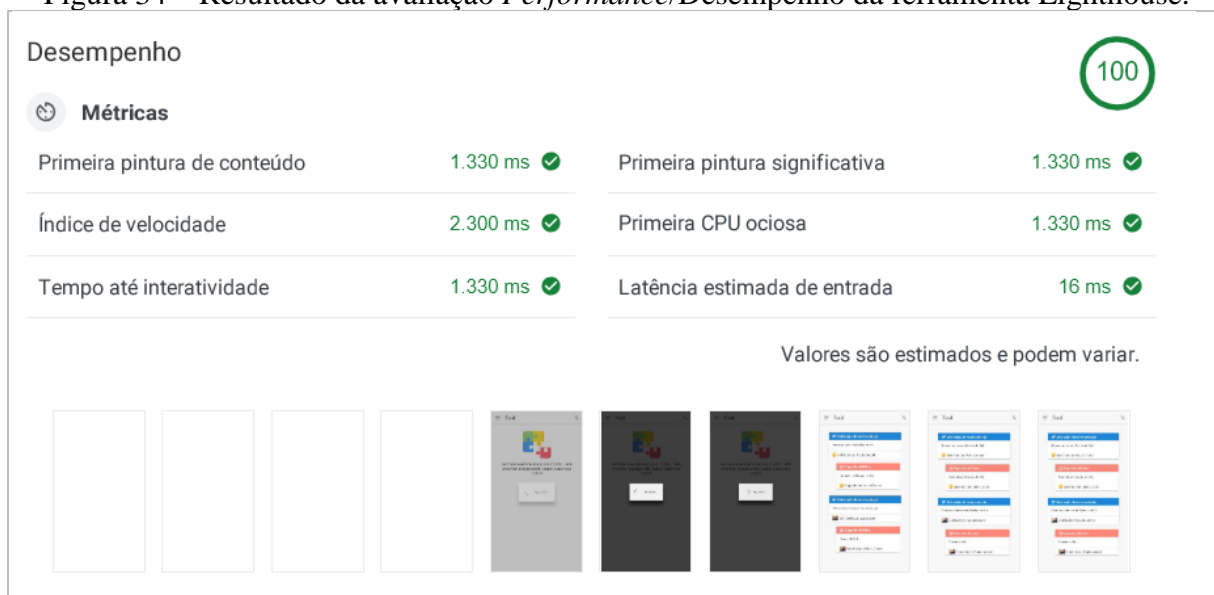
### 5.2.3 *Idle* ou momento ocioso

O método de carregamento de páginas e recursos da aplicação foi desenvolvido a partir de um método do Angular chamado *Lazy Loading*, o que permite carregar as páginas e componentes sob demanda, diminuindo o tempo de carregamento e a banda utilizada.

### 5.2.4 *Load* ou carregamento

Além da utilização do método de carregamento *Lazy Loading*, também foi feito o *cache* das requisições e dos componentes mais utilizados (como o *app shell*) para que a sensação de carregamento pareça mais rápida. Na Figura 34 é possível ver o resultado da avaliação *Performance* (ou desempenho) da ferramenta Lighthouse, que avalia se o tempo de carregamento da página está adequado para utilização da aplicação.

Figura 34 – Resultado da avaliação *Performance/Desempenho* da ferramenta Lighthouse.



Fonte: Adaptado de Lighthouse.

### 5.3 AVALIAÇÃO DO LIGHTHOUSE



A partir da execução da ferramenta Lighthouse na aplicação pôde-se reforçar a sua validação, principalmente nos quesitos de *Performance/Desempenho* e quanto aos requisitos para considera a aplicação um PWA. Reitera-se o esclarecimento de que esta avaliação é feita com base em uma *checklist*<sup>20</sup>, onde cada métrica contemplada pela aplicação recebe uma pontuação.

Nos Quadros 6 e 7 são apresentados os resultados obtidos a partir da execução da ferramenta na página inicial, e em uma das páginas internas (julgou-se que não seria necessário realizar a avaliação em todas as páginas da aplicação, uma vez que seus conteúdos são muito semelhantes, e por isso acabam apresentando resultados praticamente iguais).

A avaliação se dá a partir da simulação da aplicação em dispositivos *Desktop* (com rede de *Internet* cabeada) e em dispositivos móveis (com rede de Internet 3G). Após, são feitos comentários sobre a pontuação obtida.

<sup>20</sup> Checklist: <https://developers.google.com/web/tools/lighthouse/v3/scoring>

Quadro 6 – Avaliação da Página Inicial da aplicação, por meio da ferramenta Lighthouse.

|                          |  |
|--------------------------|--|
| <b>Tela</b>              | Página Inicial (cadastro e <i>login</i> ).   |
| <b>Avaliação Desktop</b> |  <p>Fonte: Adaptado de Lighthouse.</p>   |
| <b>Comentários</b>       | <ul style="list-style-type: none"> <li>• Desempenho – Todos os requisitos foram contemplados.</li> <li>• <i>Progressive Web App</i> – Não obteve pontuação máxima devido a uma falha<sup>21</sup> no cálculo do tamanho da janela do navegador, com relação ao conteúdo da aplicação.</li> <li>• Acessibilidade – Reporta a falta de um <i>label</i> para os <i>inputs</i> da página. Neste cenário não seria necessário, uma vez que cada <i>input</i> possui um <i>placeholder</i>.</li> <li>• Boas Práticas – Todos os requisitos foram contemplados.</li> <li>• SEO – Todos os requisitos foram contemplados.</li> </ul> |
| <b>Avaliação Mobile</b>  |  <p>Fonte: Adaptado de Lighthouse.</p>   |
| <b>Comentários</b>       | <ul style="list-style-type: none"> <li>• Desempenho – A quantidade de código em JavaScript que é carregado na página pode ser diminuído, sendo carregado somente o código que o usuário precisa utilizar.</li> <li>• <i>Progressive Web App</i> – Todos os requisitos foram contemplados.</li> <li>• Acessibilidade – Mesma situação da avaliação <i>Desktop</i>.</li> <li>• Boas Práticas – Todos os requisitos foram contemplados.</li> <li>• SEO – Todos os requisitos foram contemplados.</li> </ul>   |

<sup>21</sup> Fonte: <https://github.com/GoogleChrome/lighthouse/issues/5558>

Quadro 7 – Avaliação da página *Feed* da aplicação, por meio da ferramenta Lighthouse.

|                                    |   |
|------------------------------------|---|
| <b>Tela</b>                        | <i>Feed</i>   |
| <b>Avaliação</b><br><i>Desktop</i> | <p>Desempenho      Progressive Web App      Acessibilidade      Boas Práticas      SEO</p> <p>Fonte: Adaptado de Lighthouse.</p>  |
| <b>Comentários</b>                 | <ul style="list-style-type: none"> <li>• Desempenho – Todos os requisitos foram contemplados.</li> <li>• <i>Progressive Web App</i> – Mesma situação da avaliação da Página Inicial no <i>Desktop</i>, no Quadro 6.</li> <li>• Acessibilidade – Reporta a falta do atributo <i>alt</i> nos ícones da página e do atributo <i>value</i> para alguns botões (ambos os atributos são necessários para que os elementos sejam identificados por leitores de tela). Por uma limitação do Ionic não é possível incluir esses atributos nos componentes que estão sendo utilizados, e por isso seria necessário desenvolver um <i>workaround</i><sup>22</sup> para contemplar esse requisito.</li> <li>• Boas Práticas – Todos os requisitos foram contemplados.</li> <li>• SEO – Todos os requisitos foram contemplados.</li> </ul> |
| <b>Avaliação</b><br><i>Mobile</i>  | <p>Desempenho      Progressive Web App      Acessibilidade      Boas Práticas      SEO</p> <p>Fonte: Adaptado de Lighthouse.</p>  |
| <b>Comentários</b>                 | <ul style="list-style-type: none"> <li>• Desempenho – Mesma situação da avaliação da Página Inicial em <i>Mobile</i>, no Quadro 6.</li> <li>• <i>Progressive Web App</i> – Todos os requisitos foram contemplados.</li> <li>• Acessibilidade – Mesma situação da avaliação <i>Desktop</i>.</li> <li>• Boas Práticas – Todos os requisitos foram contemplados.</li> <li>• SEO – Todos os requisitos foram contemplados.</li> </ul>   |

<sup>22</sup> Desenvolver uma solução alternativa para corrigir uma limitação em um sistema.

## 6 CONSIDERAÇÕES FINAIS

O presente trabalho tinha como objetivo realizar estudos acerca da experiência do usuário em aplicações *web*, e mais especificamente em aplicações *web* progressivas, a fim de elencar quais elementos e técnicas seriam as mais relevantes para serem implantadas em uma aplicação que suprisse ambas as necessidades (ser um PWA, e oferecer uma boa experiência para o usuário).

A partir do referencial teórico levantado sobre *Progressive Web Apps* e *User Experience* pôde-se sugerir o desenvolvimento de uma aplicação, em formato de rede social, que possuísse funcionalidades características de PWA. Entre elas destacam-se a possibilidade de acesso *offline* aos conteúdos e o recebimento de notificações.

O desenvolvimento da aplicação se deu a partir da criação de dois projetos: uma interface cliente, e uma API – que também podem ser identificados como *front-end* e *back-end*, respectivamente. Enquanto que no *front-end* são apresentados elementos visuais, necessários para a interação do usuário com o sistema; no *back-end* são recebidas e processadas as requisições feitas por meio dessas interações.

Optou-se utilizar o Ionic Framework para o desenvolvimento do *front-end*, por oferecer uma grande quantidade de componentes de interface reutilizáveis e pelo grande engajamento da comunidade e da própria equipe do Ionic em sanar dúvidas, e de fácil acesso à sua documentação. Apesar disso é pertinente ressaltar que houve dificuldades no modo de desenvolvimento do Ionic, pois o processo que compila os arquivos e os renderiza no navegador requer um grande processamento do computador, deixando-o por muitas vezes lento ao ponto de precisar encerrar o processo e iniciá-lo novamente.

Já no *back-end* foi utilizado o Framework Django, escrito em Python, onde foi desenvolvida uma API REST, de fácil acesso pelo *front-end*. Esta ferramenta se mostrou adequada às necessidades do projeto, por também possuir uma comunidade ativa e boa documentação.

Os resultados obtidos pela validação da aplicação foram considerados satisfatórios. Com a exceção da retirada de um dos requisitos estabelecidos inicialmente, o projeto contemplou todas as expectativas funcionais. Com relação à experiência do usuário, pôde-se concluir que a aplicação também se mostra satisfatória, uma vez que está de acordo com as diretrizes RAIL. Além disso, a partir da avaliação realizada pela ferramenta Lighthouse, foi possível verificar que o desempenho da aplicação se mostrou excelente, mesmo em conexões

de *Internet 3G*, sendo um ponto positivo para a experiência dos usuários que fazem parte do público-alvo desta aplicação (pessoas que estão viajando, ou passeando, fora do alcance de uma *Wi-Fi*). Ainda com relação ao público-alvo, pode-se destacar também que as funcionalidades desenvolvidas supriram as necessidades estabelecidas na proposta de solução, como por exemplo a criação de Diários de Viagem que podem ser sugeridos nas solicitações de recomendações de outros usuários, e o fato de estas informações estarem persistidas e poderem ser acessadas a qualquer momento.

Para trabalhos futuros, sugere-se a inclusão de novas funcionalidades, como a categorização dos Locais de Interesse utilizando palavras-chave (por exemplo: restaurante, bar, festa, parque, entre outros), onde serviria também para a realização de pesquisas de locais interessantes. Outra funcionalidade pertinente seria a possibilidade de fazer *check-in* nos locais, sendo criado automaticamente um Local de Interesse, a partir dos dados encontrados na localização atual.

Por ser uma aplicação *web* progressiva ainda há a possibilidade do surgimento de novas funcionalidades que façam acesso ao *hardware*, de forma que, ao surgimento delas, faça-se uma análise sobre sua utilização e se seria pertinente adicioná-las à aplicação.



## REFERÊNCIAS

- ANGULAR. **What is Angular?** Disponível em <<https://angular.io/docs>>. Acesso em 16 de outubro de 2018.
- ARCHIBALD, J. **Introducing Background Sync**. 2015. Disponível em <<https://developers.google.com/web/updates/2015/12/background-sync>>. Acesso em 20 de junho de 2018.
- ARCHIBALD, J. **The offline cookbook**. 2014. Disponível em <<https://jakearchibald.com/2014/offline-cookbook/>>. Acesso em 09 de maio de 2018.
- ATER, T. **Building Progressive Web Apps**. Sebastopol: O'Reilly Media Inc., 2017.
- BENYON, D. **Interação humano-computador**. Pearson Prentice Hall, 2011.
- COMSCORE. **The 2017 U.S Mobile App Report**. 2017. Disponível em <<https://www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-2017-US-Mobile-App-Report>>. Acesso em 19 de março de 2018.
- FIRTMAN, M. **Progressive Web Apps on iOS are here**. 2018. Disponível em <<https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7>>. Acesso em 20 de junho de 2018.
- FLING, B. **Mobile design and development**. O'Reilly Media Inc., 2010.
- FREITAS, C. G. et.al. Desenvolvimento de Progressive Web Apps e Aplicações Nativas. **Revista Científica Multidisciplinar Núcleo do Conhecimento**, edição 07, ano 02, v. 02, p. 27-37, outubro de 2017.
- GARDINI, M. **Uma proposta de recomendações para design de interfaces web móveis**. 2015. Trabalho de Conclusão de Curso (Graduação) - Universidade de Caxias do Sul. Bacharelado em Ciência da Computação, 2015.
- GARRETT, J. J. **The Elements of User Experience**. 2000. Disponível em <<http://www.jjg.net/elements/pdf/elements.pdf>>. Acesso em 14 de maio de 2018.
- GARRETT, J. J. **The Elements of User Experience: User-Centered Design for the Web and Beyond, Second Edition**. New Riders, 2011.
- GAUNT, M. **Adicionar Notificações Push a um app da Web**. Disponível em <<https://developers.google.com/web/fundamentals/codelabs/push-notifications/>>. Acesso em 11 de maio de 2018.
- GAUNT, M.; KINLAN, P. **The Web App Manifest**. Disponível em <<https://developers.google.com/web/fundamentals/web-app-manifest/>>. Acesso em 11 de maio de 2018.

GOOGLE. **Flipkart triples time-on-site with Progressive Web App**. Disponível em <<https://developers.google.com/web/showcase/2016/flipkart>>. Acesso em 22 de março de 2018.

IBM. **Arquiteturas de Três Camadas**. Disponível em <[https://www.ibm.com/support/knowledgecenter/pt-br/SSAW57\\_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr\\_3-tier.html](https://www.ibm.com/support/knowledgecenter/pt-br/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr_3-tier.html)>. Acesso em 16 de outubro de 2018.

IONIC FRAMEWORK. **What is Ionic Framework?** Disponível em <<https://beta.ionicframework.com/docs/intro>>. Acesso em 16 de outubro de 2018.

INSTANT Loading: Building offline-first Progressive Web Apps - Google I/O 2016. [Mountain View]: **Google Chrome Developers**, 2016. (46 min.). Disponível em: <<https://www.youtube.com/watch?v=cmGr0RszHc8>> Acesso em 18 de maio 2018.

LEPAGE, P. **Seu primeiro Progressive Web App**. Disponível em <<https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=pt-br>> Acesso em 11 de março de 2018.

MARCOTTE, E. **Responsive Web Design**. Disponível em <<https://alistapart.com/article/responsive-web-design>>. Acesso em 22 de abril de 2018.

MORVILLE, P.; ROSENFELD, L. **Information Architecture for the World Wide Web**. 3. ed. Sebastopol: O'Reilly Media, Inc., 2006.

NIELSEN, J. **10 Usability Heuristics for User Interface Design**. 1995. Disponível em <<http://www.nngroup.com/articles/ten-usability-heuristics/>>. Acesso em 04 de junho de 2018.

NIELSEN, J. **Projetando Websites**. Elsevier, 2000.

NORMAN, D. **The Design Of Everyday Things**. Basic Books, 2013.

NORMAN, D.; NIELSEN, J. **The Definition of User Experience (UX)**. Disponível em <<https://www.nngroup.com/articles/definition-user-experience/>>. Acesso em 19 de maio de 2018.

PARKER, T. et al. **Designing With Progressive Enhancement: Building the Web that Works for Everyone**. New Riders, 2010.

PREECE, J.; ROGERS, Y.; SHARP, H. **Design de interação: além da interação homem-computador**. Bookman, 2005.

SILVA, M. S. **Web Design Responsivo**. Novatec, 2014.

SOARES, H. P. RWD segundo Horácio Pastor Soares. In: SILVA, M. S. **Web Design Responsivo**. Novatec, 2014, p. 43-47

STALLINGS, W. **Criptografia e segurança de redes**. Pearson, 2015.

**APÊNDICE A – QUESTIONÁRIO E RESULTADOS DA PESQUISA SOBRE O  
PÚBLICO-ALVO DA APLICAÇÃO PROPOSTA**

### **Interesse em viagens**

Todos os campos são obrigatórios.

#### **Qual a sua idade?**

17 ou menos  18 a 23  24 a 29  30 a 39  40 a 49  Acima de 50

---

#### **Qual seu sexo?**

Masculino  Feminino  Prefiro não dizer

---

#### **Qual sua renda média mensal?**

Até R\$1000,00  De R\$1000,00 a R\$2400,00  De R\$2500,00 a R\$3400,00  
 De R\$3500,00 a R\$5000,00  Acima de R\$5000,00  Prefiro não dizer

---

#### **Qual seu estado civil?**

Solteiro(a) ou namorando  Casado(a)  Divorciado(a) ou separado(a)  Viúvo(a)  
 Prefiro não dizer

---

#### **Você realizou alguma viagem nos últimos 6 meses?**

Marque uma ou mais alternativas.

Sim, a trabalho  Sim, a lazer  Não

---

#### **Como você costuma fazer suas viagens?**

Marque uma ou mais alternativas.

Sozinho(a)  Com amigos  Com família  Com grupos de excursão  
 Não costumo fazer viagens

---

#### **Como você busca recomendações de locais interessantes para suas viagens?**

Marque uma ou mais alternativas.

Peço recomendações aos meus amigos ou conhecidos  
 Pesquisa por recomendações na internet  
 Planejo minhas viagens com agências  
 Não busco recomendações

---

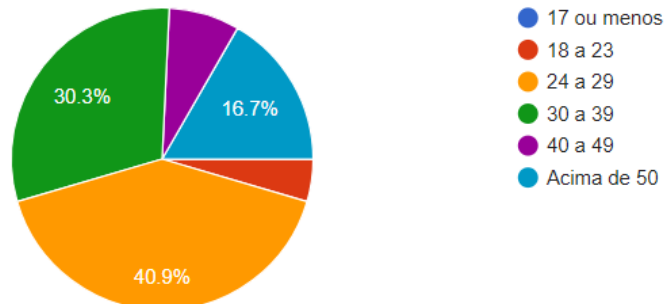
#### **Você já consultou algum tipo de aplicativo ou website para programar uma viagem? Se sim, qual(is)?**

Marque uma ou mais alternativas.

TripAdvisor  Guia mTrip  Tripit  Foursquare  Não consultei  
 Outro: \_\_\_\_\_

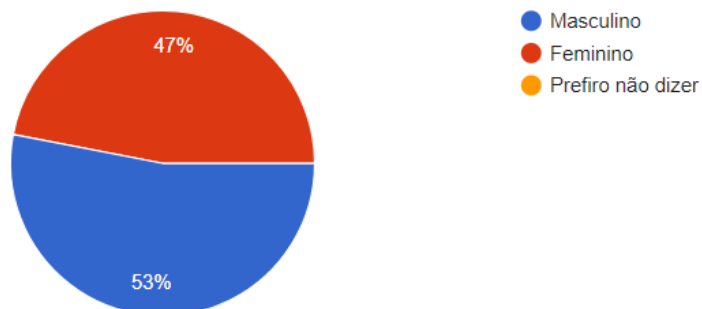
### Qual a sua idade?

66 responses



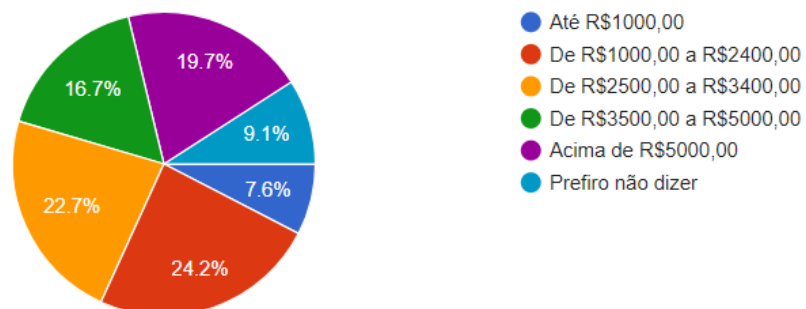
### Qual seu sexo?

66 responses



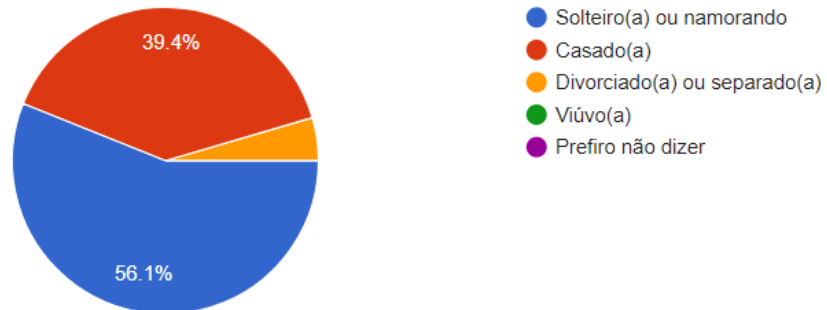
### Qual sua renda média mensal?

66 responses



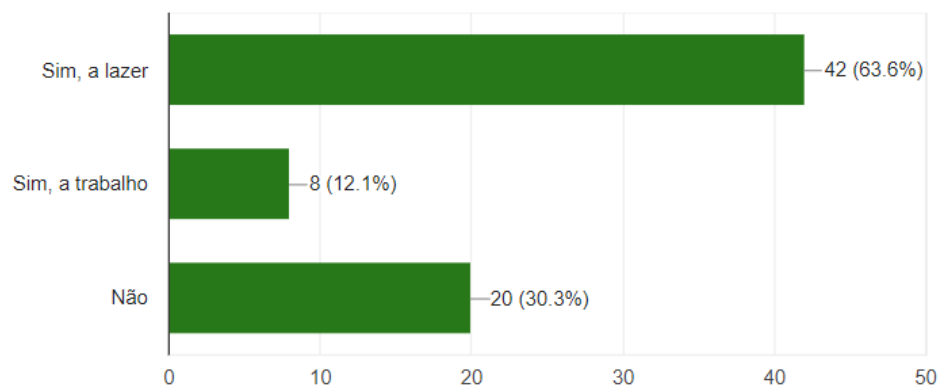
### Qual seu estado civil?

66 responses



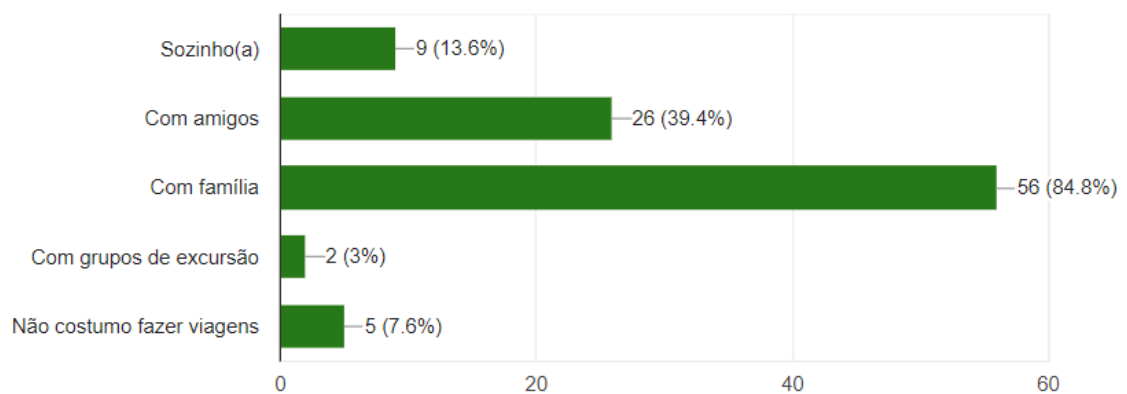
### Você realizou alguma viagem nos últimos 6 meses?

66 responses



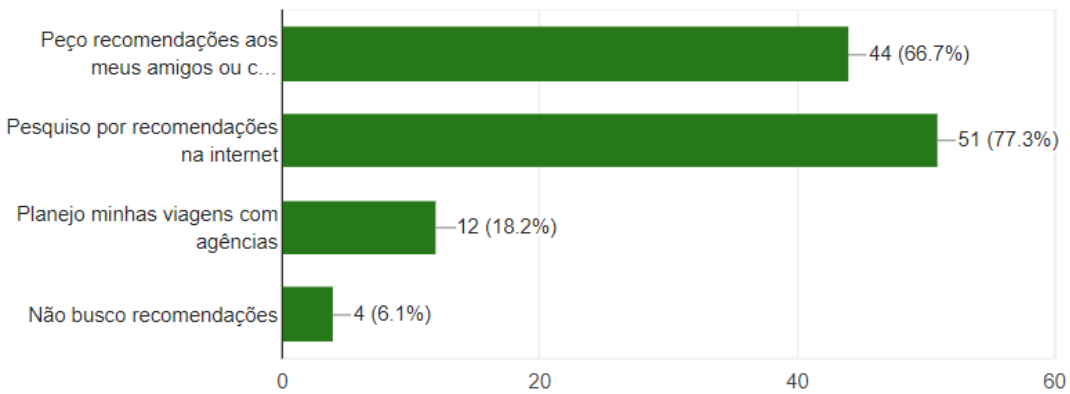
### Como você costuma fazer suas viagens?

66 responses



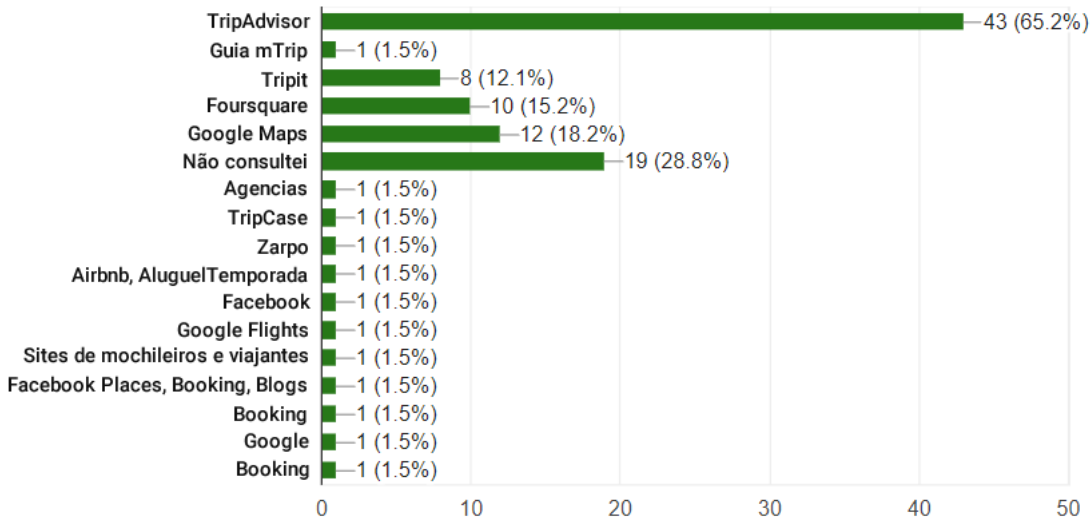
## Como você busca recomendações de locais interessantes para suas viagens?

66 responses



## Você já consultou algum tipo de aplicativo ou website para programar uma viagem? Se sim, qual(is)?

66 responses



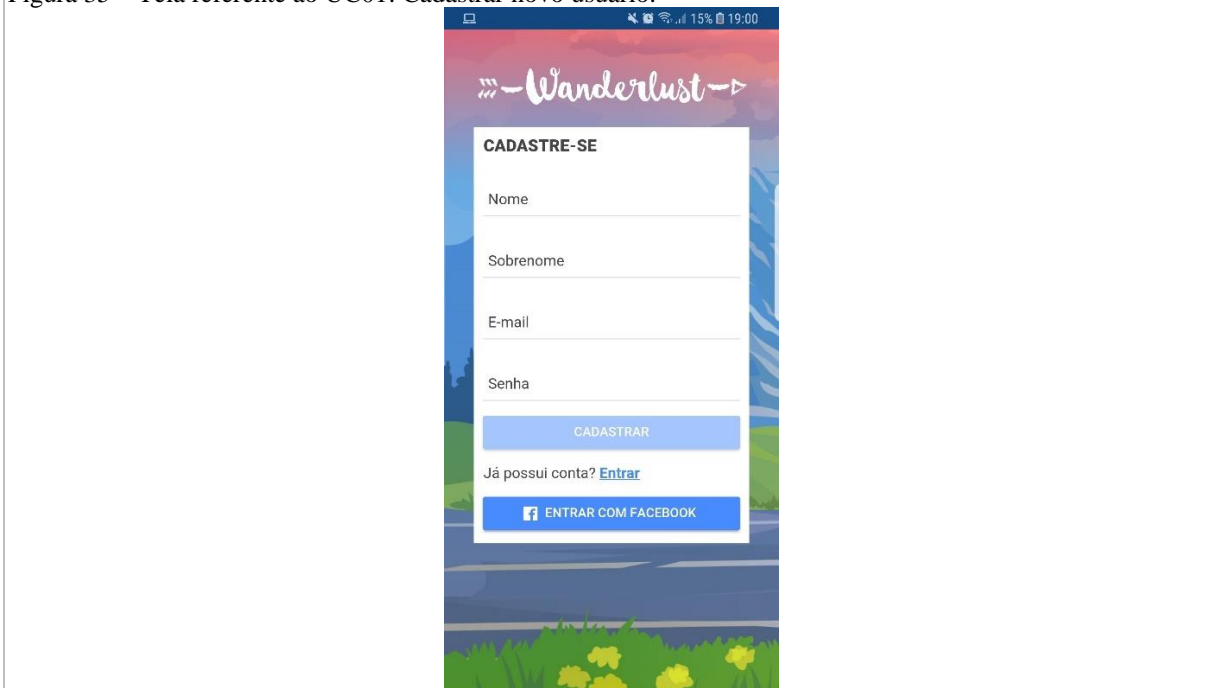
## APÊNDICE B – CASO DE USO 01: CADASTRAR NOVO USUÁRIO

Quadro 8 – UC01: Cadastrar novo usuário.

|                    |   |
|--------------------|---|
| Descrição          | O visitante deseja criar um cadastro, a partir do preenchimento de um formulário ou de sua conta do Facebook, para poder acessar o sistema.   |
| Atores             | Visitante, Facebook   |
| Pré-condições      | Nenhuma.  |
| Pós-condições      | Cadastro do usuário.  |
| Referência         | R1.   |
| Fluxo principal    | <p><b>1.</b> O usuário decide por fazer o cadastro via formulário ou via sua conta do Facebook</p> <p><b>1.1.</b> O usuário preenche o formulário de cadastro com e-mail, nome completo e senha. Todos os campos são obrigatórios.</p> <p><b>1.2a.</b> O usuário clica no botão “Conecte com o Facebook”.</p> <p><b>1.2b.</b> O sistema faz a comunicação com serviço do Facebook, que retorna os dados necessários para efetuar o cadastro.</p> <p><b>2.</b> Cadastro concluído.</p> |
| Fluxo alternativos | <p><b>1.1a.</b> Sistema verifica que e-mail já está cadastrado. Volta ao passo 1.</p> <p><b>1.1b.</b> Sistema verifica que um ou mais campos obrigatórios não foram preenchidos, ou possuem erros. Volta ao passo 1, com os dados mantidos.</p> <p><b>1.2.</b> Caso o usuário não estiver conectado ao Facebook, informa e-mail e senha.</p>  |

Fonte: Elaborado pela autora.

Figura 35 – Tela referente ao UC01: Cadastrar novo usuário.



Fonte: Elaborado pela autora.

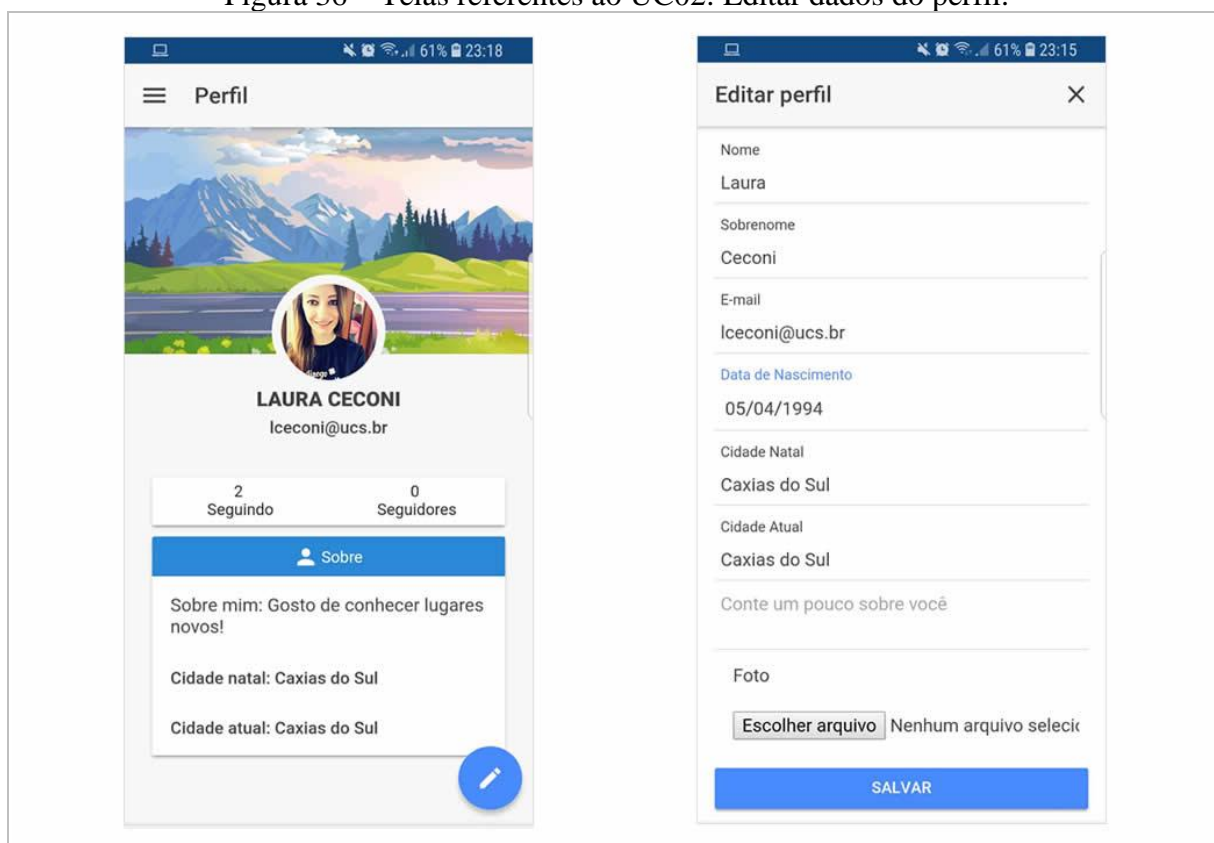
## APÊNDICE C – CASO DE USO 02: EDITAR DADOS DO PERFIL

Quadro 9 – UC02: Editar dados do perfil.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja editar os dados do seu perfil.   |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.  |
| Pós-condições      | Dados de perfil modificados e salvos.   |
| Referência         | R2.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário acessa a página de “Perfil”.</li> <li>2. O usuário clica no botão “Editar perfil”.</li> <li>3. O usuário edita/adiciona seus dados pessoais (data de nascimento, sexo, cidade e telefone), foto e descrição.</li> </ol> |
| Fluxo alternativos | Sem fluxos alternativos.  |

Fonte: Elaborado pela autora.

Figura 36 – Telas referentes ao UC02: Editar dados do perfil.



Fonte: Elaborado pela autora.



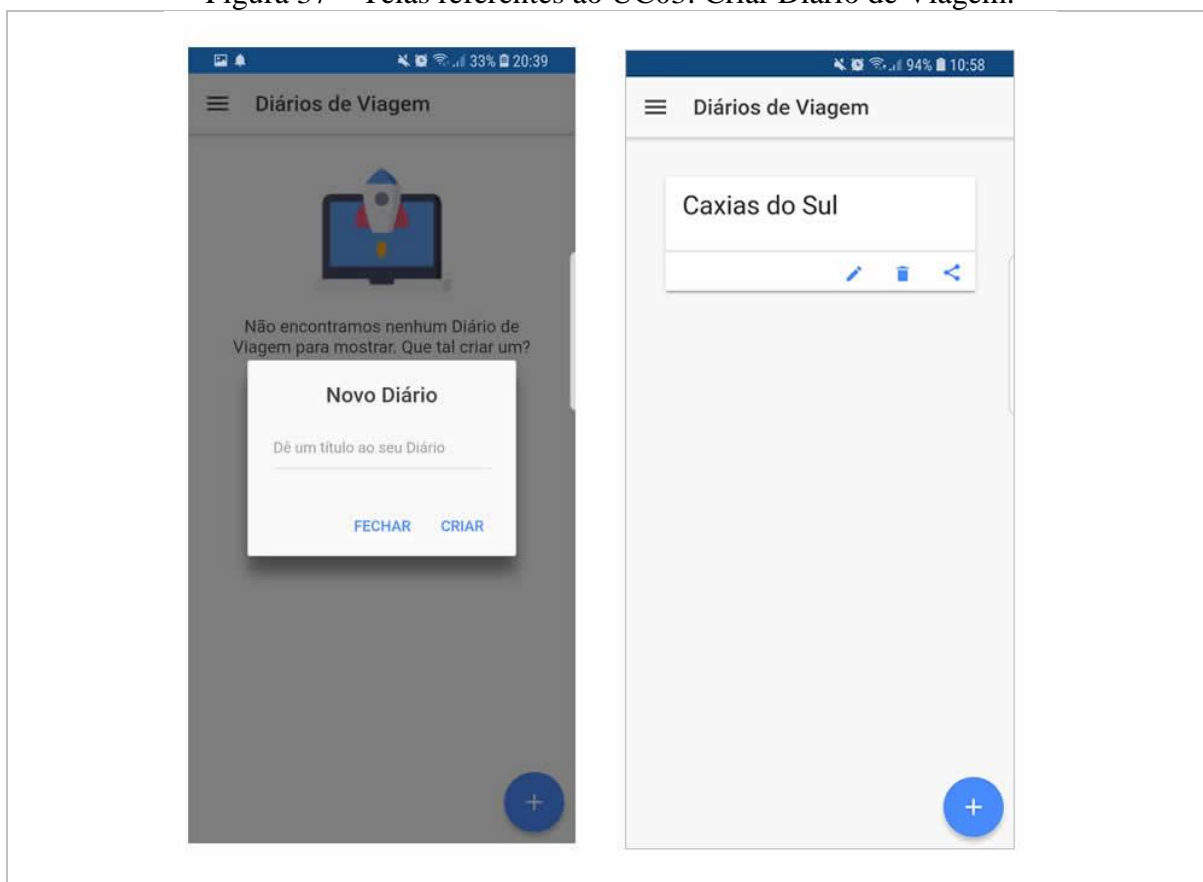
## APÊNDICE D – CASO DE USO 03: CRIAR DIÁRIO DE VIAGEM

Quadro 10 – UC03: Criar Diário de Viagem.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja criar um Diário de Viagem.   |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.  |
| Pós-condições      | Criação do Diário de Viagem.  |
| Referência         | R4.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário clica no botão “Diários de Viagem”.</li> <li>2. Na tela de listagem dos Diários de Viagem, o usuário clica em “Criar”.</li> <li>3. O usuário preenche o formulário com título e descrição, ambos obrigatórios.</li> <li>4. O usuário salva o formulário.</li> </ol> |
| Fluxo alternativos | <ol style="list-style-type: none"> <li>4. Sistema verifica que um ou mais campos obrigatórios não foram preenchidos, ou possuem erros. Volta ao passo 2, com os dados mantidos.</li> </ol>  |

Fonte: Elaborado pela autora.

Figura 37 – Telas referentes ao UC03: Criar Diário de Viagem.



Fonte: Elaborado pela autora.

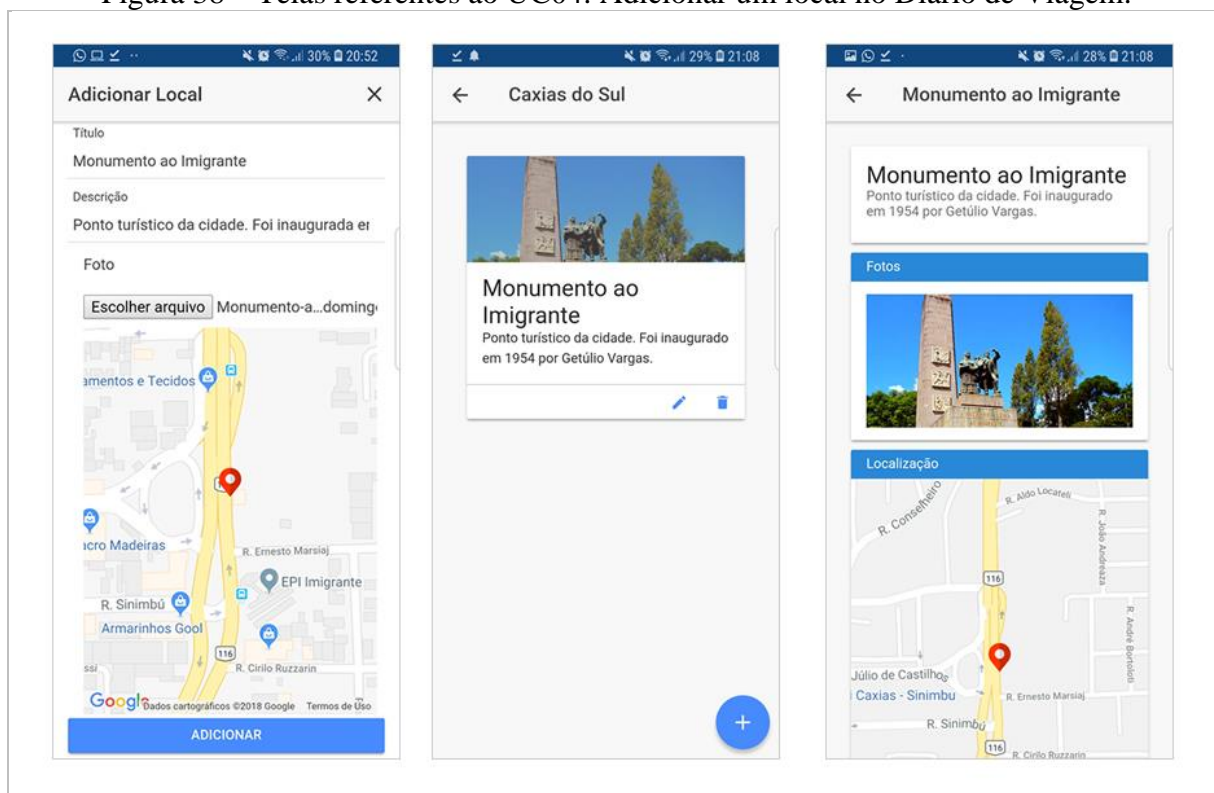
## APÊNDICE E – CASO DE USO 04: ADICIONAR UM LOCAL NO DIÁRIO DE VIAGEM

Quadro 11 – UC04: Adicionar um local no Diário de Viagem.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja adicionar um local no Diário de Viagem criado.   |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.<br>Ter criado um Diário de Viagem  |
| Pós-condições      | Inclusão do local no Diário de Viagem   |
| Referência         | R5.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário clica no botão “Diários de Viagem”.</li> <li>2. Na tela de listagem dos Diários de Viagem, o usuário clica no ícone de “mais”, no Diário que pretende adicionar o local.</li> <li>3. O usuário preenche o formulário com nome do local (obrigatório) e descrição, aponta a localização no mapa e adiciona fotos.</li> <li>4. O usuário salva o formulário.</li> </ol> |
| Fluxo alternativos | 4. Sistema verifica que o nome do local não foi preenchido. Volta ao passo 3, com os dados mantidos.  |

Fonte: Elaborado pela autora.

Figura 38 – Telas referentes ao UC04: Adicionar um local no Diário de Viagem.



Fonte: Elaborado pela autora.

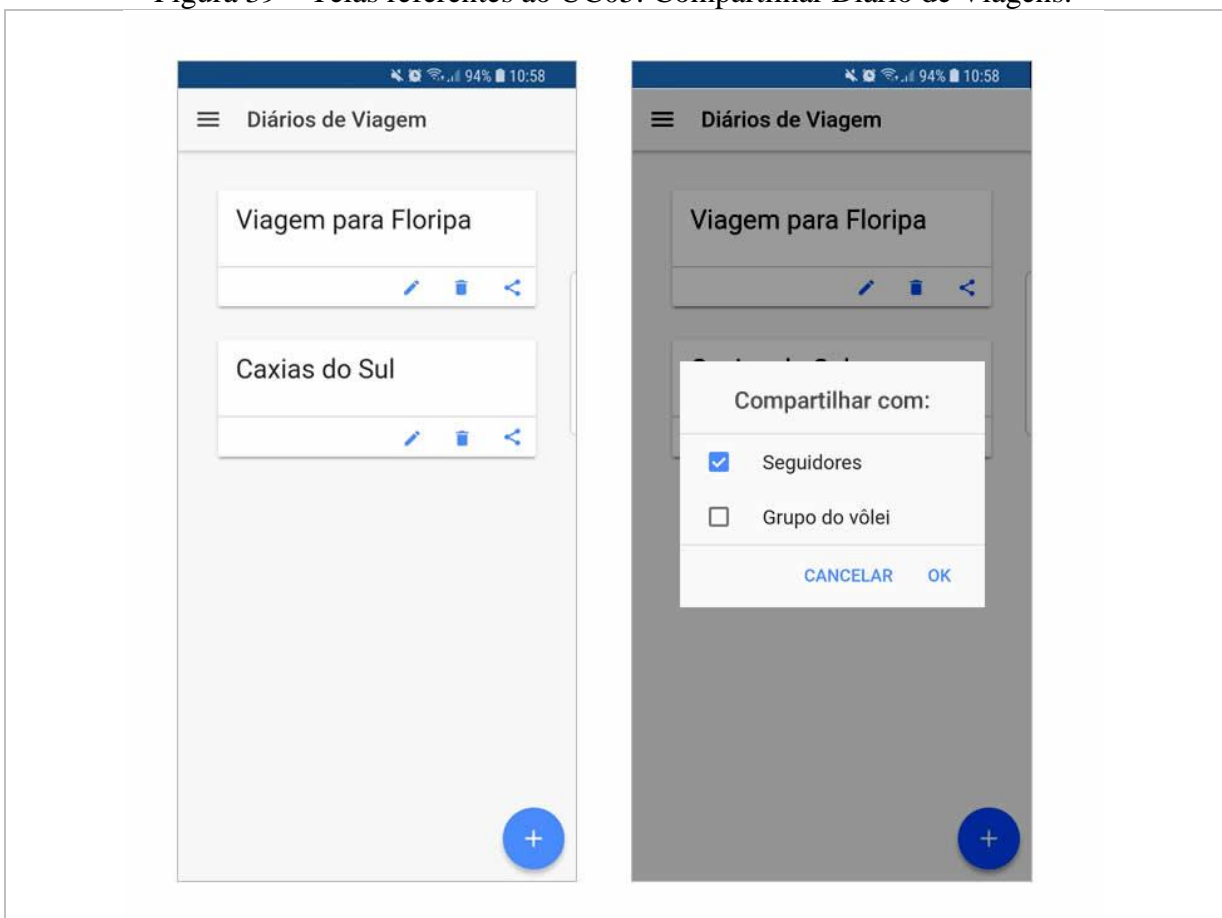
## APÊNDICE F – CASO DE USO 05: COMPARTILHAR DIÁRIO DE VIAGEM

Quadro 12 – UC05: Compartilhar Diário de Viagem.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja compartilhar um Diário de Viagem.  |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.<br>Ter criado um Diário de Viagem.   |
| Pós-condições      | Compartilhamento do Diário de Viagem.   |
| Referência         | R6.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário acessa seu Diário de Viagens.</li> <li>2. O usuário clica no ícone de compartilhamento do Diário de Viagem.</li> <li>3. O usuário seleciona para quem deseja compartilhar a informação (busca de outros usuários, para todos os seguidores ou grupos).</li> <li>4. O usuário salva a configuração de compartilhamento.</li> </ol> |
| Fluxo alternativos | Sem fluxos alternativos.  |

Fonte: Elaborado pela autora.

Figura 39 – Telas referentes ao UC05: Compartilhar Diário de Viagens.



Fonte: Elaborado pela autora.

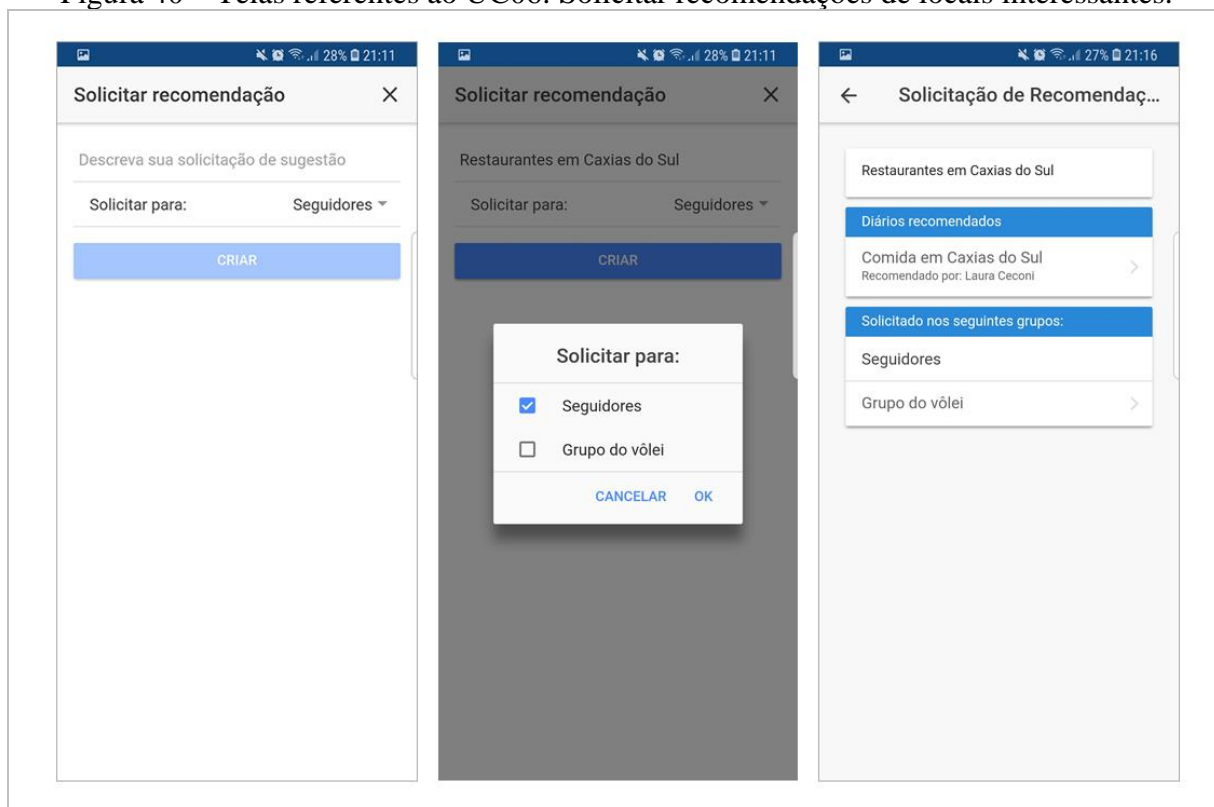
## APÊNDICE G – CASO DE USO 06: SOLICITAR RECOMENDAÇÕES DE LOCAIS INTERESSANTES

Quadro 13 – UC06: Solicitar recomendações de locais interessantes.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja solicitar aos seus seguidores, ou a grupos, recomendações de locais interessantes.   |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.  |
| Pós-condições      | Criação da solicitação. Recebimento de recomendações.   |
| Referência         | R7.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário clica no botão “Solicitar recomendações”.</li> <li>2. O usuário informa a cidade ou local atual.</li> </ol> |
| Fluxo alternativos | Sem fluxos alternativos.  |

Fonte: Elaborado pela autora.

Figura 40 – Telas referentes ao UC06: Solicitar recomendações de locais interessantes.



Fonte: Elaborado pela autora.

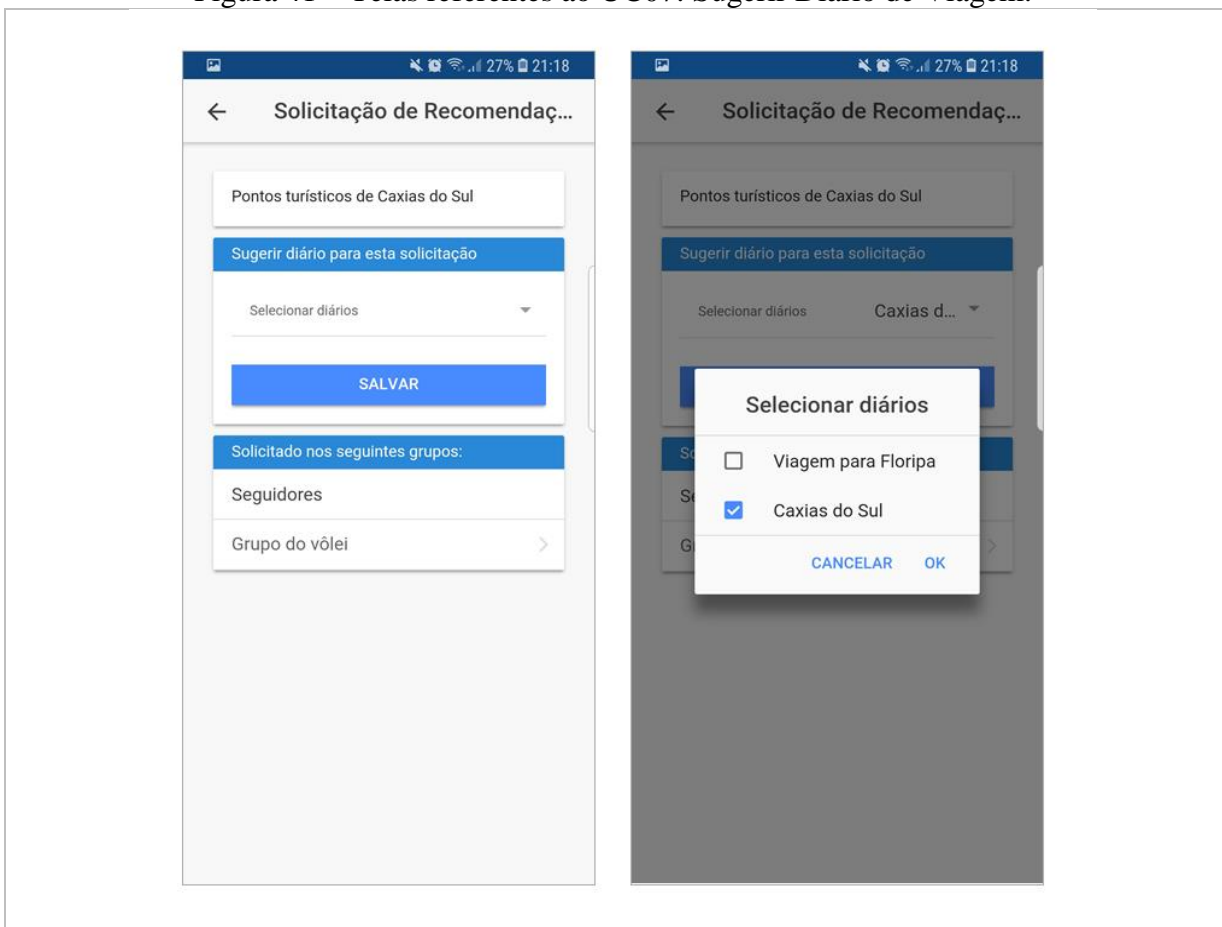
## APÊNDICE H – CASO DE USO 07: SUGERIR DIÁRIO DE VIAGEM

Quadro 14 – UC07: Sugerir Diário de Viagem.

|                    |  |
|--------------------|--|
| Descrição          | O usuário deseja sugerir um Diário de Viagem em uma solicitação de Recomendação  |
| Atores             | Usuário.   |
| Pré-condições      | Estar cadastrado e autenticado no sistema.<br>Ter criado um Diário de Viagem.  |
| Pós-condições      | Sugestão do Diário de Viagem.  |
| Referência         | R6.  |
| Fluxo principal    | <b>1.</b> O usuário acessa uma solicitação de Recomendação.<br><b>2.</b> Seleciona um dos seus diários já criados e clica em “Ok”. |
| Fluxo alternativos | Sem fluxos alternativos.   |

Fonte: Elaborado pela autora.

Figura 41 – Telas referentes ao UC07: Sugerir Diário de Viagem.



Fonte: Elaborado pela autora.

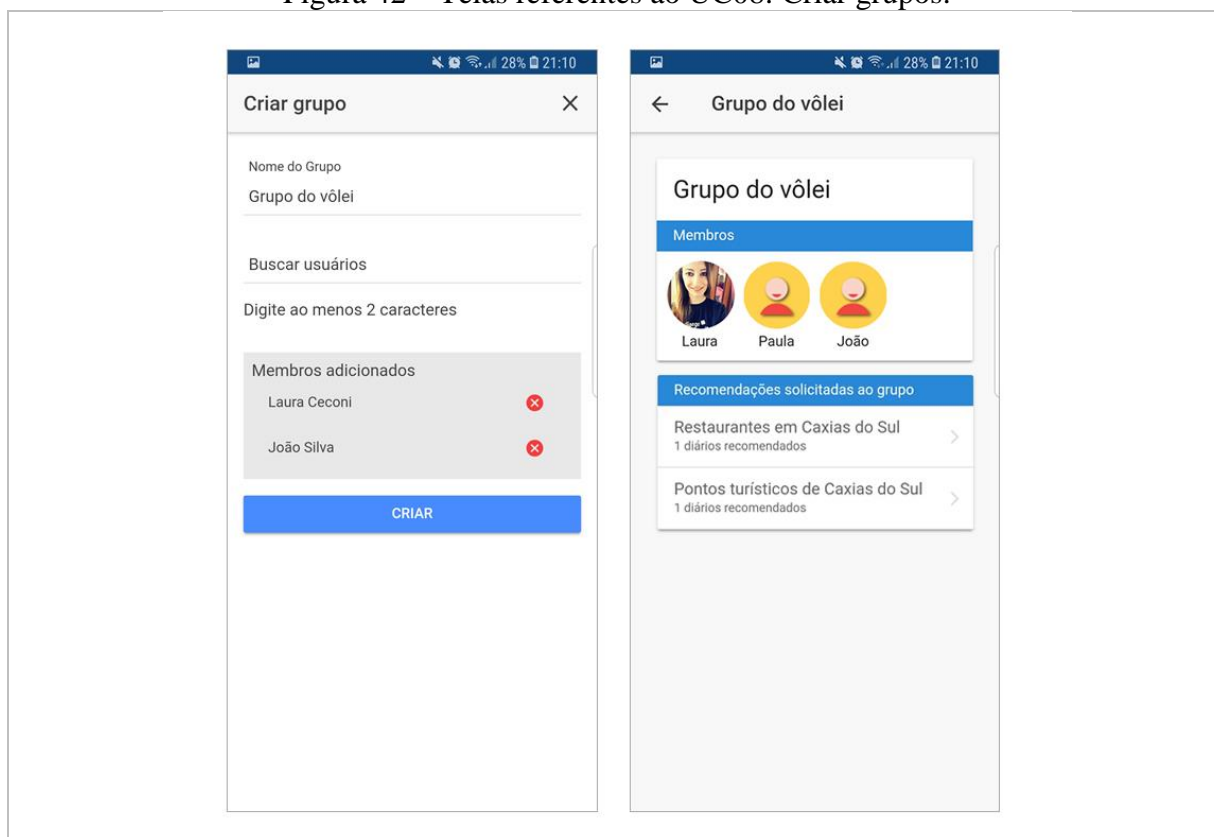
## APÊNDICE I – CASO DE USO 08: CRIAR GRUPOS

Quadro 15 – UC08: Criar grupos.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja criar um grupo, adicionando outros usuários com interesses em comum.   |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.  |
| Pós-condições      | Criação do grupo.   |
| Referência         | R9.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário clica no botão “Criar grupo”.</li> <li>2. O usuário preenche o nome e a descrição do grupo. O campo nome é obrigatório.</li> <li>3. O usuário busca e adiciona outros usuários ao grupo.</li> <li>4. O usuário salva o novo grupo.</li> </ol> |
| Fluxo alternativos | 4. Sistema verifica que o campo nome não foi informado. Retorna ao passo 2.   |

Fonte: Elaborado pela autora.

Figura 42 – Telas referentes ao UC08: Criar grupos.



Fonte: Elaborado pela autora.

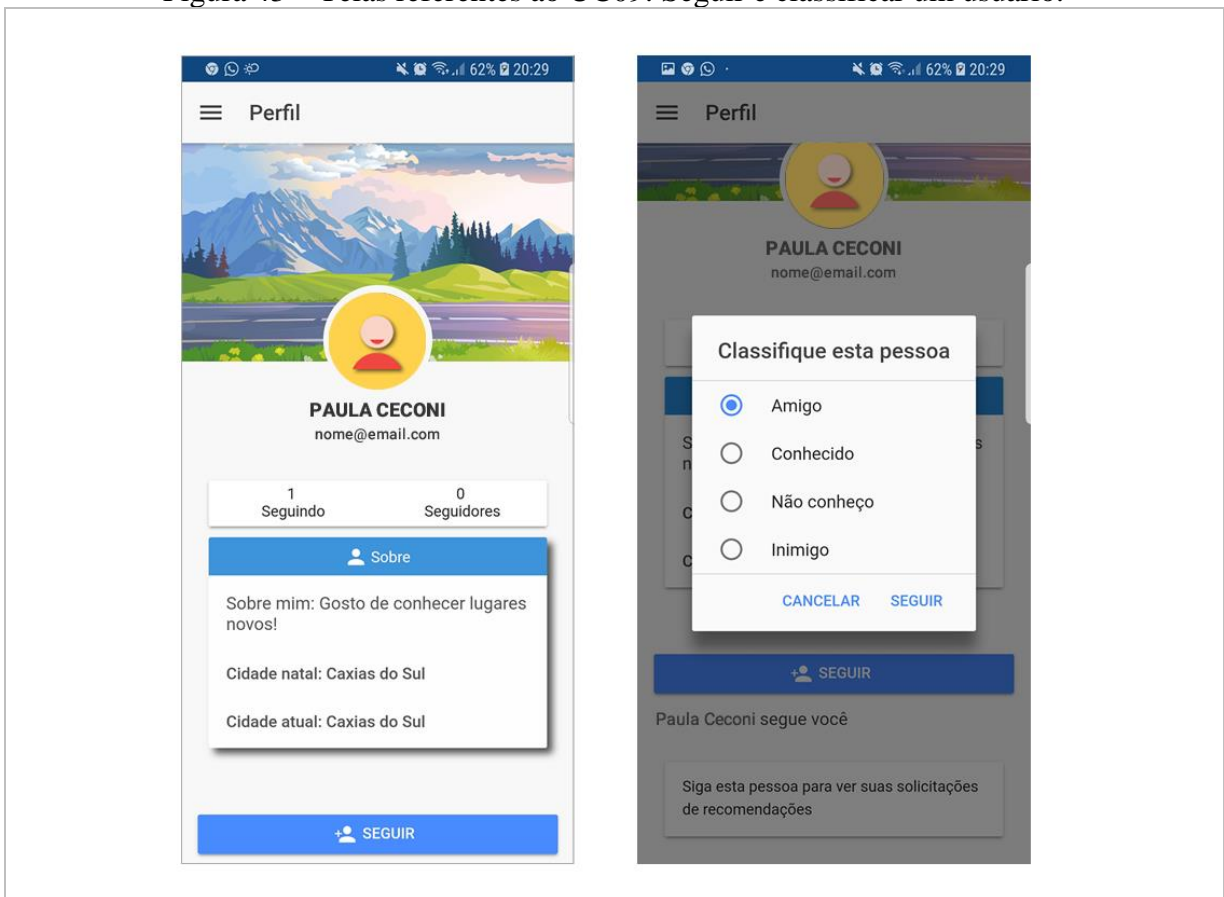
## APÊNDICE J – CASO DE USO 09: SEGUIR E CLASSIFICAR UM USUÁRIO

Quadro 16 – UC09: Seguir e classificar um usuário.

|                    |   |
|--------------------|---|
| Descrição          | O usuário deseja seguir outro usuário, no intuito de acompanhar suas atividades.  |
| Atores             | Usuário.  |
| Pré-condições      | Estar cadastrado e autenticado no sistema.  |
| Pós-condições      | Seguir outro usuário.   |
| Referência         | R9 e R12.   |
| Fluxo principal    | <ol style="list-style-type: none"> <li>1. O usuário busca pelo nome de um outro usuário.</li> <li>2. O usuário acessa o perfil encontrado.</li> <li>3. O usuário clica no botão “Seguir”.</li> <li>4. O usuário seleciona uma classificação para o este usuário.</li> </ol> |
| Fluxo alternativos | Sem fluxos alternativos.  |

Fonte: Elaborado pela autora.

Figura 43 – Telas referentes ao UC09: Seguir e classificar um usuário.



Fonte: Elaborado pela autora.