UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UNIVERSIDADE VIRTUAL - UFC VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

FILIPE DOURADO FALCÃO

DESENVOLVIMENTO DO APLICATIVO TURISTANDO BEBERIBE UTILIZANDO REACT NATIVE

FORTALEZA

2022

FILIPE DOURADO FALCÃO

DESENVOLVIMENTO DO APLICATIVO TURISTANDO BEBERIBE UTILIZANDO REACT NATIVE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual - UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Windson Viana de Carvalho

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F163d Falcão, Filipe Dourado.
Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native / Filipe Dourado Falcão. – 2022.
67 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Prof. Dr. Windson Viana de Carvalho.

1. Cross-platform. 2. React Native. 3. Expo. 4. Turismo. I. Título.

CDD 302.23

FILIPE DOURADO FALCÃO

DESENVOLVIMENTO DO APLICATIVO TURISTANDO BEBERIBE UTILIZANDO REACT NATIVE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual - UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Windson Viana de Carvalho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará (UFC)

Prof. Me. Carlos Diego Andrade de Almeida
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

A Deus, por ter me dado força para chegar até aqui e pelas suas bençãos e misericórdias em minha vida.

Ao Prof. Dr. Windson Viana de Carvalho, pelos conselhos e orientações dados durante o desenvolvimento deste trabalho.

Aos meus pais, por toda a dedicação dada à minha criação e educação, fazendo tudo o que estava ao seu alcance para que eu pudesse chegar ao ensino superior.

À minha namorada, Viviane Macambira, pelo apoio e companhia durante grande parte da minha graduação, sempre me incentivando a não desistir e me dando suporte nos momentos mais difíceis.

Aos integrantes da equipe Ethos, João André, Lucca Macambira e Viviane Macambira, por tornarem possível a criação do produto apresentado neste trabalho. Foi um prazer ter trabalhado com vocês.

À Universidade Federal do Ceará e ao curso de Sistemas e Mídias Digitais, por abrirem as suas portas, dessa forma, me dando a oportunidade de ter acesso ao ensino superior.

RESUMO

Com a volta gradativa do setor turístico após o início da pandemia em 2020, em 2021, a equipe Ethos desenvolveu o aplicativo Turistando Beberibe, que tem como principal proposta oferecer um catálogo de experiências turísticas na região, uma vez que existe uma carência de aplicações voltadas ao turismo na cidade Beberibe/CE. Neste contexto, o presente relatório técnico tem como objetivo descrever todo o processo de criação do aplicativo, desde a sua conceituação até o seu desenvolvimento, destacando as características *cross-platform* da ferramenta de desenvolvimento usada, o React Native. Também estão presentes neste relatório uma breve conceituação da abordagem *cross-platform* de desenvolvimento e as principais etapas para a construção e distribuição de uma aplicação feita com React Native. No caso do Turistando Beberibe, foi usada a plataforma Expo. Ao final do relatório, é descrita uma avaliação de usabilidade do aplicativo utilizando o método SUS, que contou com seis participantes. Outrossim, descrevem-se aspectos técnicos do aplicativo gerado assim como as limitações decorrentes das abordagens e ferramentas de desenvolvimento escolhidas.

Palavras-chave: Cross-platform. React Native. Expo. Turismo.

ABSTRACT

With the gradual return of the tourist sector after the beginning of the pandemic in 2020, in 2021, the Ethos team developed the Turistando Beberibe application, whose main proposal is to offer a catalog of tourist experiences in the region, since there is a lack of applications aimed at tourism in the city Beberibe/CE. In this context, this technical report aims to describe the entire application creation process, from its conceptualization to its development, highlighting the cross-platform characteristics of the development tool used, React Native. Also present in this report are a brief conceptualization of the cross-platform development approach and the main steps for building and distributing an application made with React Native. In the case of Turistando Beberibe, the Expo platform was used. At the end of the report, a usability assessment of the generated application is carried out, through a survey using the SUS method, with six participants. Furthermore, technical aspects of the generated application are described, as well as the limitations resulting from the chosen development tools and approaches.

Keywords: Cross-platform. React Native. Expo. Tourism.

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

ADECE	Agência de Desenvolvimento do Estado do Ceará
ABIH-CE	Associação Brasileira da Indústria de Hotéis no Ceará
CNDL	Confederação Nacional de Dirigentes Lojistas
SPC	Serviço de Proteção ao Crédito
SMD	Sistemas e Mídias Digitais
UFC	Universidade Federal do Ceará
SO	sistema operacional
SUS	*System Usability Scale*
MVC	Model View Controller
DOM	Document Object Model
NPM	Node Package Manager
TCLE	Termo de Consentimento Livre e Esclarecido

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

O Brasil possui um vasto território com diversas paisagens, o que o torna um dos países com maior potencial turístico. Segundo o Ministério de Turismo, o Ceará é um dos destinos mais procurados do país, atraindo turistas de várias partes do Brasil e do mundo (ADECE, 2021). O estado conta com 573 quilômetros de litoral e tem dias ensolarados durante quase todo o ano, além de regiões serranas e diversas atrações culturais.

De acordo com a Agência de Desenvolvimento do Estado do Ceará (ADECE), a cidade de Beberibe é uma das mais procuradas para o turismo de lazer no estado (ADECE, 2021). A região é famosa por possuir um vasto litoral, contemplando belas praias, como a Praia de Morro Branco e a Praia das Fontes. Além disso, também fazem parte do seu cenário: falésias coloridas, monumentos naturais, dunas, fontes e a Lagoa de Uruaú.

Ultimamente, o turismo tem enfrentado uma grande retração por causa das condições impostas pela pandemia do COVID-19. Porém, com a chegada das vacinas, esse setor vem retomando as atividades aos poucos (VIECELI, 2021). No estado do Ceará não é diferente, segundo Régis Medeiros, presidente da Associação Brasileira da Indústria de Hotéis no Ceará (ABIH-CE), é esperada uma boa recuperação no setor ainda em 2021, devendo chegar à normalidade em 2022 (FCDLCE, 2021).

Por outro lado, um dos principais efeitos colaterais causados pelo isolamento social imposto pela pandemia foi o aumento do uso de *smartphones* por parte das pessoas. De acordo com um relatório disponibilizado em abril de 2021 pela empresa de monitoramento de aplicativos App Annie, o uso de *smartphones* no Brasil aumentou cerca de 35% no primeiro quadrimestre de 2021, quando comparado ao mesmo período em 2019 (KRISTIANTO, 2021), inclusive, aumentou o uso por pessoas que antes eram menos familiarizadas, como os idosos (PIRES; NUNES, 2020). Além disso, segundo a pesquisa Consumo online no Brasil, realizada pela Confederação Nacional de Dirigentes Lojistas (CNDL) e pelo Serviço de Proteção ao Crédito (SPC), 87% das compras online de produtos ou serviços realizadas em 2021 foram através de um *smartphone* (CNDL, 2021). Esta pesquisa também mostra que apesar dos sites ainda serem o principal meio utilizado pelos consumidores, os aplicativos estão ganhando cada vez mais espaço no mercado devido a ampliação do acesso à rede 4G.

A volta gradativa do setor turístico e o aumento do uso de *smartphones* juntos

reforçam a possibilidade de que aplicativos voltados para o turismo possam ser usados com maior frequência. Essas soluções podem ser usadas para o aluguel de estadias, para consultar informação sobre locais ou para a contratação de serviços, como passeios e trilhas.

1.2 Justificativa

Durante a disciplina de Projeto Integrado II do curso de Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará (UFC), a equipe Ethos[1] se interessou pelo tema de aplicações turísticas voltadas para cidades específicas. Apesar do grande potencial turístico da cidade de Beberibe, a equipe constatou por meio de pesquisas nas lojas de aplicativos dos dois principais sistemas operacionais de *smartphones*, Android e iOS, que não existe uma aplicação voltada especificamente para o turismo na cidade de Beberibe.

Isso posto, a equipe Ethos enxergou uma oportunidade de negócio promissora e resolveu explorá-la. Com o objetivo de suprir a carência encontrada, durante a disciplina de Projeto Integrado II, a equipe desenvolveu um aplicativo voltado para a contratação de passeios e experiências turísticas com o foco no município de Beberibe, chamado de Turistando Beberibe.

No entanto, apesar da boa oportunidade e do potencial encontrado, a equipe se deparou com algumas dificuldades para a produção do Turistando Beberibe. Com o objetivo de abranger a máxima quantidade de pessoas, foi decidido que o aplicativo seria disponível para as plataformas Android e iOS, ou seja, precisaria ser desenvolvido um aplicativo para cada sistema operacional (SO). Porém, com tempo, recursos e mão de obra limitados, a equipe esbarrou no seguinte problema: Como desenvolver um aplicativo para as plataformas Android e iOS com um baixo custo de produção?

Com o objetivo de contornar esse problema, foi vislumbrado o uso de abordagens *cross-platforms* como uma possível solução. O *cross-platform* é uma abordagem de desenvolvimento na qual permite a criação de uma única aplicação que funcione em diversas plataformas. O principal objetivo dessa metodologia é permitir a construção de um software que seja o mais próximo possível de um nativo, mas que possa ser distribuído para a maior quantidade possível de sistemas operacionais (XANTHOPOULOS; XINOGALOS, 2013).

Hoje em dia, existem diversas maneiras de se desenvolver aplicações *cross-platform*

[1] Membros da equipe Ethos: Filipe Dourado Falcão, João André França Mourão, Lucca Uriel Macambira Barbosa e Viviane Macambira de Oliveira

e ferramentas variadas, como Xamarin[2] e Qt[3], estão disponíveis para dar suporte a isso. Uma delas é o React Native, que é uma biblioteca Javascript criada pelo Facebook para a produção de aplicativos para as plataformas Android e iOS. Esta biblioteca usa conceitos e tecnologias do desenvolvimento web para o móvel, dessa forma, proporcionando uma baixa curva de aprendizado para programadores que estão familiarizados com o ambiente web, além de ser uma das ferramentas mais utilizadas no mercado (STATISTA, 2022), contando com o suporte do Facebook e com uma comunidade muito ativa. Por este motivo, React Native foi a ferramenta escolhida para a produção do Turistando Beberibe.

1.3 Objetivos geral e específicos

Levando em consideração o exposto até o momento, o objetivo principal deste relatório é descrever o processo de desenvolvimento do aplicativo Turistando Beberibe utilizando a biblioteca React Native. Os objetivos específicos são:

- Apresentar o React Native e seu ambiente de desenvolvimento;
- Mostrar as etapas de desenvolvimento de uma aplicação React Native com Expo;
- Testar e avaliar o aplicativo desenvolvido para as plataformas Android e iOS.

1.4 Estrutura do documento

A estrutura e organização deste trabalho foi baseada no TCC de Araujo (2019), sendo composto por mais seis capítulos. Na seção 2, são apresentadas as técnicas e métodos usados no desenvolvimento do Turistando Beberibe. No capítulo 3, são descritas algumas aplicações móveis de apoio ao turismo e também é conceituado o termo *cross-platform*, assim como as ferramentas React, React Native e Expo. A seção 4 apresenta detalhadamente o produto deste trabalho, o aplicativo Turistando Beberibe. Já no capítulo 5, são expostas as etapas de desenvolvimento da aplicação em questão. A etapa de teste e avaliação é descrita na seção 6. Por fim, concluindo o trabalho, existe o capítulo 7, no qual é feita uma síntese do que foi mostrado neste relatório e também são apresentadas algumas propostas de trabalhos futuros.

[2] Xamarin - https://dotnet.microsoft.com/en-us/apps/xamarin
[3] Qt - https://www.qt.io/

2 METODOLOGIA

2.1 Técnicas e Métodos adotados

O presente trabalho é um relatório descritivo sobre o desenvolvimento do aplicativo Turistando Beberibe durante a disciplina de Projeto Integrado II. No início do projeto, foi realizada uma pesquisa de natureza quali-quantitativa para verificar se a ideia do aplicativo teria potencial de mercado e se seria bem aceita pelo público-alvo, que basicamente são prestadores de serviços turísticos e viajantes, com foco específico na região de Beberibe/CE.

A pesquisa foi realizada com formulários online, um para o perfil de viajante, contando com 71 participações, e outro para o perfil de prestador de serviços, contando com 21 participações. Nosso principal objetivo foi apurar se nosso público-alvo possui celular com acesso à internet, costuma utilizar aplicativos, gosta de fazer passeios turísticos e tem interesse em um aplicativo nos moldes do Turistando Beberibe. Ao final da pesquisa, 80,3% dos viajantes disseram ter interesse pela proposta de aplicativo apresentada e 76,2% dos prestadores de serviços gostaram da ideia de uma aplicação voltada para a divulgação de seus trabalhos. O restante da pesquisa é melhor detalhado no capítulo 4.

Para guiar o desenvolvimento do aplicativo, a equipe adotou conceitos do *Design Thinking* e do SCRUM. O *Design Thinking* é uma metodologia de desenvolvimento iterativa que tem como objetivo transformar uma ideia em um produto finalizado (AMBROSE; HARRIS, 2011). O diferencial dessa abordagem é o foco no usuário do projeto a ser desenvolvido, pois sua premissa é transformar as necessidades, anseios e limitações dos usuários em soluções criativas e que agreguem valor ao produto e à empresa. Segundo Ambrose e Harris, o *Design Thinking* é composto por sete etapas, que são: definição, pesquisa, ideação, prototipação, seleção, implementação e aprendizagem (AMBROSE; HARRIS, 2011).

O SCRUM é uma metodologia ágil para gestão e planejamento de projetos de software (SOARES, 2004). Essa metodologia também é iterativa e se baseia em ciclos com duração de duas a quatro semanas, chamados de *sprints*. No início de cada *sprint*, é feita uma reunião de planejamento na qual as funcionalidades do produto em desenvolvimento são priorizadas e selecionadas para serem desenvolvidas durante o ciclo que irá se iniciar.

Assim como o *Design Thinking*, não seguimos o SCRUM à risca. Porém, aderimos ao modelo de *sprints*, separando e desenvolvendo funcionalidades do Turistando Beberibe em pequenos ciclos iterativos precedidos de reuniões de planejamento.

Na fase de avaliação do aplicativo realizada neste trabalho, foi feita uma avaliação de usabilidade utilizando o método *System Usability Scale* (SUS), com o intuito de verificar se a aplicação foi bem aceita pelos usuários, e uma outra análise que considerou aspectos técnicos do desenvolvimento, como tamanho final do aplicativo gerado e aproveitamento de código para ambas as plataformas.

A avaliação de usabilidade se refere a um conjunto de métodos baseados na avaliação e análise com o foco em aspectos de usabilidade da interface com o usuário (NASCIMENTO, 2006). O SUS é uma das ferramentas mais utilizadas para esse fim. Este método consiste em um questionário simples, contendo dez assertivas, com o objetivo de coletar e demonstrar uma visão geral sobre a usabilidade de um determinado produto e, ao mesmo tempo, avaliar os níveis de satisfação do usuário em relação à interface (SANTANA *et al.*, 2016).

2.2 Etapas do TCC

As etapas do desenvolvimento deste trabalho são detalhadas nesta seção. Como mostrado na Figura 1, os passos 1, 2 e 3 foram concluídos nos semestres anteriores, nas disciplinas de Projeto Integrado II e Projeto de Trabalho Final. Assim, as etapas 4 e 5 foram finalizadas durante o semestre atual.

Figura 1 – Etapas do TCC

Fonte: elaborado pelo autor

2.2.1 Análise e Desenvolvimento do aplicativo Turistando Beberibe

O aplicativo Turistando Beberibe foi desenvolvido na disciplina de Projeto Integrado II. Durante a disciplina, a equipe Ethos fez a análise da viabilidade do produto no mercado através de pesquisas com potenciais usuários. Também foram implementados protótipos para

guiar a escolha de qual tecnologia e proposta de design utilizar. Após essas etapas, se deu início ao desenvolvimento do aplicativo tentando seguir os conceitos do *Design Thinking* e do SCRUM.

2.2.2 Escrita da Introdução e Metodologia do TCC

A escrita deste trabalho foi iniciada durante a disciplina de Projeto de Trabalho Final. Nesse período foram decididos o tema e o formato do trabalho. Além da escrita dos capítulos de Introdução e Metodologia, também foi criado um cronograma para se ter como referência durante o restante do desenvolvimento do trabalho.

2.2.3 Escolha das metodologias para aplicação de testes no aplicativo

A escolha das metodologias para a avaliação do Turistando Beberibe também foi feita na disciplina de Projeto de Trabalho Final, especificamente durante o desenvolvimento desde capítulo. Decidiu-se realizar uma avaliação de usabilidade utilizando o método SUS e uma análise de aspectos técnicos do aplicativo gerado.

2.2.4 Aplicação e análise dos testes

A avaliação de usabilidade, utilizando o método SUS, e a avaliação de aspectos mais técnicos do aplicativo, assim com as suas respectivas análises, foram realizadas no decorrer da disciplina de TCC. A escolha prévia das metodologias ajudou bastante e acelerou esta fase.

2.2.5 Escrita dos demais capítulos do TCC

Com os capítulos de Introdução e Metodologia já escritos, restou o desenvolvimento dos demais. A escrita de algumas seções foram feitas paralelamente aos testes citados anteriormente. Ao final, foi feito um compilado com os resultados dos testes, dessa forma, possibilitando a sua escrita.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados conceitos diretamente ligados ao desenvolvimento do Turistando Beberibe. Primeiramente, é apresentada a abordagem de desenvolvimento *cross-platform*. Logo após, são apresentadas as principais ferramentas utilizadas no desenvolvimento do aplicativo, sendo as bibliotecas React e React Native e a plataforma Expo.

3.1 *Cross-platform*

O desenvolvimento *cross-platform* consiste em uma abordagem que permite a criação de uma aplicação que seja funcional e compatível em diferentes plataformas. Esse tipo de abordagem foi criada com o objetivo de tornar o desenvolvimento mais eficiente, evitando repetição de código (HEITKÖTTER *et al.*, 2012) e retirando a necessidade de se ter equipes de programadores especialistas em plataformas específicas, dessa forma, aumentando a produtividade e diminuindo o custo de produção.

O *cross-platform* permeia várias áreas da engenharia de software. No entanto, se tratando de desenvolvimento móvel, que é a realidade deste trabalho, um aplicativo *cross-platform* basicamente deve funcionar nas duas principais plataformas, Android e iOS (STATCOUNTER, 2022). O desafio dessa abordagem de desenvolvimento é encontrar pontos em comum entre os diferentes sistemas operacionais, para que dessa forma seja possível criar um software a partir de uma base de código única, possibilitando o aproveitamento e a utilização do máximo de funcionalidades que cada sistema e aparelho possa oferecer (HEITKÖTTER *et al.*, 2012).

Existem diversas formas de se desenvolver aplicativos *cross-platform*. Uma das técnicas mais usadas é criar sites que se adaptam a diferentes tamanhos de telas, ou seja, sites responsivos [1]. Utilizando linguagens de desenvolvimento web, como HTML, CSS e JavaScript, é possível fazer que um site tenha uma aparência comum em um computador, mas tenha características de um aplicativo quando for acessado a partir de um *smartphone* (DANIELSSON, 2016).

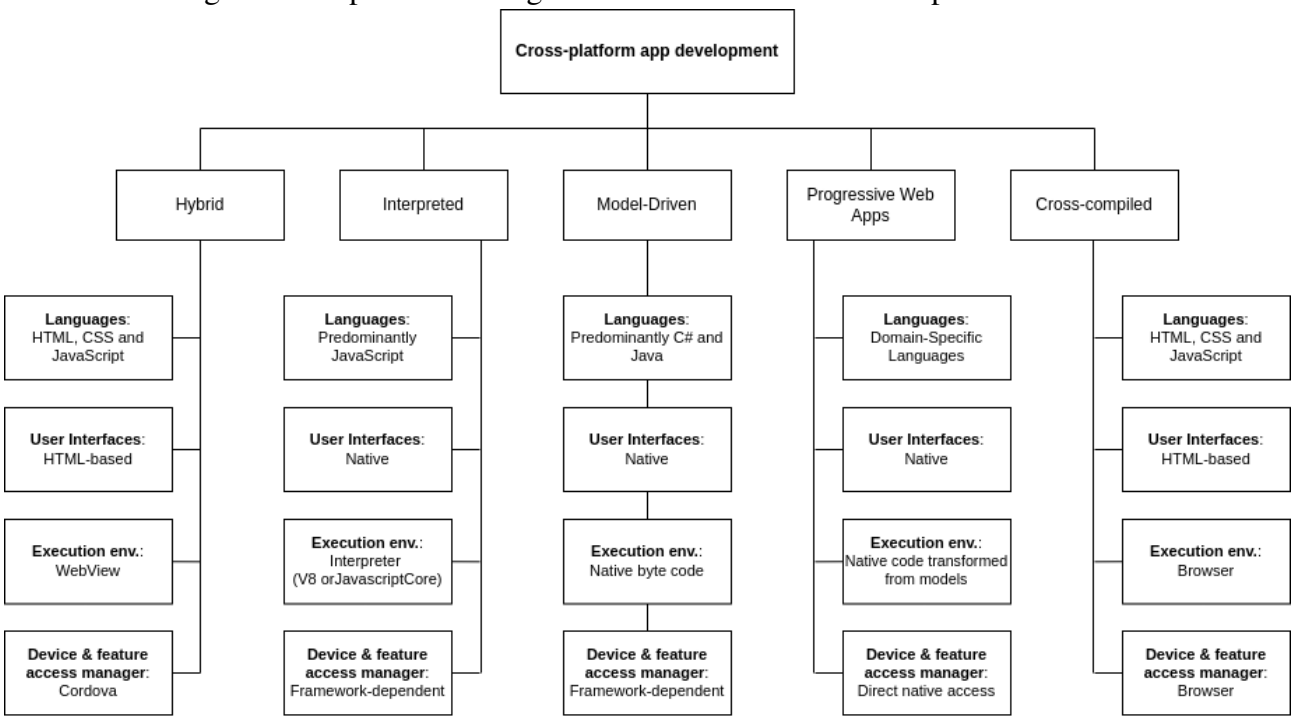
Outra técnica *cross-platform* reconhecida são os *frameworks* híbridos, como Cordova[2] e Ionic[3]. Esse método utiliza conceitos do desenvolvimento web e nativo para gerar uma aplicação que funcione em ambos sistemas operacionais. O aplicativo é criado com ferramentas

[1] Responsividade - https://blog.betrybe.com/tecnologia/responsividade/
[2] Apache Cordova - https://cordova.apache.org/
[3] Ionic Framework - https://ionicframework.com/

web, mas executado e exibido de forma nativa, por meio de uma WebView[4]. Apesar desta técnica possibilitar o uso de alguns recursos do dispositivo e ser bem menos custosa do que o desenvolvimento nativo, ela possui algumas limitações, principalmente em relação ao desempenho e à interface de usuário, e não consegue oferecer uma experiência parecida com as aplicações nativas (DANIELSSON, 2016).

Além das metodologias *cross-platform* citadas, existem diversas outras alternativas. Na Figura 2 são mostradas outras abordagens de desenvolvimento, destacando as linguagens de programação mais utilizadas no método, o ambiente de execução, a forma na qual a interface é exibida e como é feito o acesso às funcionalidades do dispositivo (BIØRN-HANSEN *et al.*, 2018).

Figura 2 – Tipos de abordagens de desenvolvimento cross-platform



Fonte: Biørn-Hansen *et al.* (2018)

3.2 React

Segundo Kumar e Singh (2016), o React é uma biblioteca de UI desenvolvida pelo Facebook para facilitar a criação de componentes de interface interativos, reusáveis e com estados dinâmicos (KUMAR; SINGH, 2016). Seu lançamento foi em 2013, no formato de código-aberto,

[4] WebView - https://developer.android.com/reference/android/webkit/WebView

com o objetivo principal de facilitar o desenvolvimento de interfaces complexas, com dados que podem sofrer alterações durante o seu tempo de uso (DANIELSSON, 2016). O React atua na parte de visualização da camada do Model View Controller (MVC)[5]. No entanto, uma aplicação React pode ser renderizada tanto no lado do cliente quanto do servidor (DANIELSSON, 2016).

Uma das principais características do React é a forma como ele lida com o Document Object Model (DOM). Segundo Danielsson (2016), o DOM é uma estrutura em forma de árvore que representa toda a estrutura de uma página web e seus estados usando elementos HTML (DANIELSSON, 2016). Ao invés de manipular o DOM diretamente, o que pode ser bastante custoso, a ideia por trás do React é criar uma cópia, chamada de Virtual DOM, e fazer as manipulações dos dados e atualizar os elementos de interface através dela (KUMAR; SINGH, 2016).

Outra particularidade marcante do React é a linguagem utilizada para gerar os componentes de interface, o JSX. Esta é uma extensão de sintaxe para o JavaScript, que tem uma estrutura muito semelhante a do HTML. Ao invés de separar o JavaScript e o HTML em arquivos distintos, como normalmente é feito, a ideia do React é separar e desacoplar a aplicação em pequenas unidades reutilizáveis, chamadas de componentes (REACTJS, 2021a).

3.3 React Native

O React Native é uma biblioteca também criada pelo Facebook, mas voltada para o desenvolvimento de aplicativos móveis. Por ser baseado no React, o React Native se beneficia de suas vantagens, como a componentização dos elementos de interface, o JSX e o Virtual DOM.

Sendo uma biblioteca para o desenvolvimento móvel *cross-platform*, é possível aproveitar a mesma base de código para fazer a distribuição da aplicação tanto para sistemas iOS quanto Android, além de outras plataformas, como: Android e Apple TV, Linux, macOS e Windows. No entanto, uma das características que diferencia o React Native de outras ferramentas *cross-platform*, como o Ionic, é ter a capacidade de gerar aplicações finais com componentes das plataformas nativas, não necessitando de uma WebView para poder exibir os elementos de interface do aplicativo (DANIELSSON, 2016). Essa característica faz com que aplicações feitas em React Native ofereçam uma melhor experiência aos usuários, além de proporcionar um melhor desempenho em relação a outras abordagens *cross-platform*. Isso é possível porque o React Native é executado em uma instância embutida do JavaScriptCore (iOS)

[5] MVC - https://www.rareparts.com/pdf/MVC.pdf

ou V8 (Android)[6] dentro das aplicações (DANIELSSON, 2016). Dessa forma, o React Native transforma os componentes JSX em componentes nativos (MALIK, 2021).

3.4 Expo

O Expo é um *framework* construído para apoiar o desenvolvimento de aplicações React em geral, mas é mais conhecido pelo suporte dado na criação de aplicativos móveis com React Native. Ele é formado por um conjunto de ferramentas e serviços criados com o foco nos ambientes do React Native e das plataformas nativas, que auxiliam no desenvolvimento, construção, implantação, testes e na execução de simuladores nos sistemas Android e iOS utilizando a mesma base de código (EXPO, 2021a).

O Expo funciona como uma camada de abstração para uma aplicação React Native. Ele elimina a necessidade do desenvolvedor ter de lidar com códigos das plataformas nativas, o que pode ocorrer quando se está utilizando apenas o React Native. Dessa forma, é preciso lidar somente com JavaScript no ambiente de desenvolvimento.

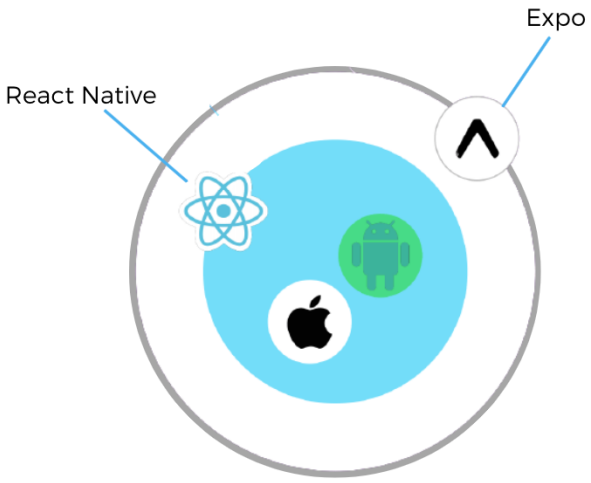
Também é fornecido um programa de linha de código, o Expo CLI, que é a principal interface de comunicação com o ambiente do Expo. Essa ferramenta retira a necessidade de se usar os programas XCode[7] e Android Studio[8] para executar uma aplicação e distribuí-la nas lojas das respectivas plataformas.

Apesar do Expo trazer uma série de benefícios, ele também traz consigo algumas limitações. Por estar em uma camada acima do React Native, como visto na Figura 3, algumas APIs ainda não estão disponíveis em seu ambiente. Por exemplo, não é possível acessar a API de Bluetooth do dispositivo. No entanto, a equipe de desenvolvimento do Expo está constantemente evoluindo a ferramenta[9], o que a torna uma boa opção para a criação de aplicativos móveis em conjunto com o React Native.

---

6 JavaScriptCore e V8 são *engines* usadas para interpretar e executar códigos Javascript.
7 XCode - https://developer.apple.com/xcode/
8 Android Studio - https://developer.android.com/studio
9 Lista com SDKs lançados do Expo - https://docs.expo.dev/workflow/upgrading-expo-sdk-walkthrough/

Figura 3 – Camadas de abstração do React Native e Expo



Fonte: elaborado pelo autor

4 TURISTANDO BEBERIBE

O aplicativo Turistando Beberibe, produto deste trabalho, é descrito de forma detalhada neste capítulo. É feita a apresentação do problema que o aplicativo busca sanar, bem como a sua proposta de solução. Também são mostrados os resultados da pesquisa de mercado realizada.

Por último, é apresentado um diagrama de casos de uso, seguido da explicação de cada caso, o esquema de arquitetura utilizado no aplicativo e as principais funcionalidades da aplicação, juntamente com suas principais telas.

4.1 Definição do problema

Como dito na introdução deste trabalho, a cidade de Beberibe possui um grande potencial turístico, tendo um vasto litoral que contempla praias belíssimas. A região também conta com várias atrações turísticas, como passeios de *buggy*, jangada, *jet-ski*, lancha, esqui aquático na Lagoa do Uruaú, além de estar muito próxima ao Rio Jaguaribe, um ótimo lugar para ser explorado pelos turistas.

No entanto, a equipe Ethos não encontrou nenhum site ou aplicativo específico para Beberibe que tenha como foco o turismo e a reserva de serviços do gênero. Alguns sites e redes sociais podem até ajudar nas recomendações de experiências e passeios, mas é uma busca descentralizada e os dados podem estar desatualizados.

Pensando nisso, a equipe decidiu propor soluções para minimizar essa carência encontrada no setor turístico da cidade de Beberibe.

4.2 Público-alvo

A proposta do aplicativo é ter dois públicos-alvo: prestadores de serviços turísticos e viajantes. O primeiro público é composto por pessoas que exercem profissões como: guias turísticos, professores de surf, motoristas de *buggy*, dentre outras, sem restrição de sexo ou idade. Já o público dos viajantes abrange principalmente pessoas jovens e adultas, de ambos os sexos, porém não exclui pessoas com idades mais avançadas.

4.3 Pesquisa mercadológica

De início, foi realizada uma pesquisa de mercado durante a disciplina de Projeto Integrado II. A pesquisa foi realizada usando formulários online e divulgada em grupos relacionados ao turismo no Facebook. Foram feitos dois formulários distintos, um para cada público-alvo. Foram coletadas 71 respostas de viajantes e 21 de prestadores de serviços.

O objetivo da pesquisa foi apurar se o público-alvo possui celular com acesso à internet, faz uso de aplicativos, gosta de explorar os destinos de suas viagens e tem interesse em anunciar seus serviços em um aplicativo nos moldes do Turistando Beberibe.

Na pesquisa dos viajantes, foi observado que mais da metade, cerca de 56,3%, utilizam de aplicativos para o planejamento de viagens e 88,7% planejam com antecedência o que fazer e visitar em viagens. Também foi apurado que 32,4% gostam e procuram por experiências e passeios e 64,8% tendem a também conhecer melhor o local de destino, dependendo das atrações turísticas. Levando em consideração o contexto de Beberibe, 93% disseram não conhecer ferramentas que disponibilizam o contato de prestadores de serviços na cidade. Por fim, 80,3% se interessaram pela proposta de aplicativo apresentada.

Mesmo com a tendência do público-alvo de viajantes considerar a realização de experiências e passeios turísticos uma opção, não tendo plena certeza se irá fazer ou não, foi enxergada uma chance de atrair esse público que está em dúvida através da divulgação do aplicativo, uma vez que a porcentagem de interessados no aplicativo foi alta. Com isso, é possível chegar a conclusão de que apesar de existirem outras ferramentas com foco em viagens, os possíveis concorrentes não atendem às expectativas e necessidades dos viajantes quando comparados à proposta do Turistando Beberibe.

Partindo para os resultados dos prestadores de serviços, foram obtidas respostas muito favoráveis à solução proposta. Todos os participantes disseram possuir um celular com acesso à internet e 81% costumam utilizar aplicativos no cotidiano, o que pode ter contribuído com a alta aceitação, cerca de 76,2%, de um aplicativo focado na divulgação de seus trabalhos.

Com os resultados das pesquisas favoráveis ao Turistando Beberibe, a equipe seguiu com a ideia e iniciou o desenvolvimento da aplicação.

4.4 Principais Concorrentes

Os principais concorrentes do Turistando Beberibe são aqueles que apresentam um produto similar ao que está sendo proposto: um aplicativo que auxilia no planejamento de viagem com o foco na oferta de serviços turísticos a pessoas que vão para Beberibe. Nesse contexto, não foram encontrados concorrentes diretos à aplicação proposta. No entanto, nas pesquisas sobre concorrência realizadas pela equipe, o TripAdvisor e o Airbnb foram considerados potenciais concorrentes por serem produtos já desenvolvidos e integrados no mercado para auxiliar no planejamento de viagens.

Na pesquisa realizada com o público-alvo, tanto com os viajantes quanto com os prestadores de serviço, também foram feitas algumas perguntas para sondar o uso de aplicações voltadas para o turismo. Por parte dos viajantes, as respostas confirmaram a pesquisa sobre concorrência realizada anteriormente. Os aplicativos mais utilizados para viagens foram TripAdvisor e Airbnb. Já para os prestadores de serviços, foi perguntado quais eram os meios que eles utilizavam para a divulgação dos seus trabalhos. A maioria apontou para grupos de Facebook, contas no Instagram e parcerias hotéis e pousadas.

Como o Facebook e o Instagram não são centralizados na proposta de viagens, foi decidido pesquisar e analisar somente os dois serviços anteriormente mencionados. Com isso, foram levantados pontos interessantes nessas aplicações que serviram de inspiração para algumas funcionalidades e regras do Turistando Beberibe, como mostrado a seguir.

No TripAdvisor, uma das maiores aplicações para contratação de serviços turísticos, existe um recurso de ranking de popularidade que os usuários podem avaliar estabelecimentos e comentar como foi a sua experiência naquele local, sendo muito útil para a visualização dos *feedbacks*, que auxiliam na escolha final. Esse recurso de popularidade já existe em outros aplicativos e também foi usado no desenvolvimento do Turistando Beberibe. Já no Airbnb, existe uma rede de usuários que são divididos entre anfitriões e inquilinos. Eles são avaliados uns pelos outros e vão criando uma reputação na rede. Todas as transações são feitas diretamente dentro do aplicativo, sendo cobrada uma taxa de 20% dos anfitriões que oferecerem as acomodações. Dessa forma, a equipe também adotou essa estratégia de cobrar uma taxa percentual relativa ao valor da experiência oferecida pelos prestadores de serviço, passando a ser a principal fonte de receita do aplicativo.

4.5 Príncipios do Turistando Beberibe

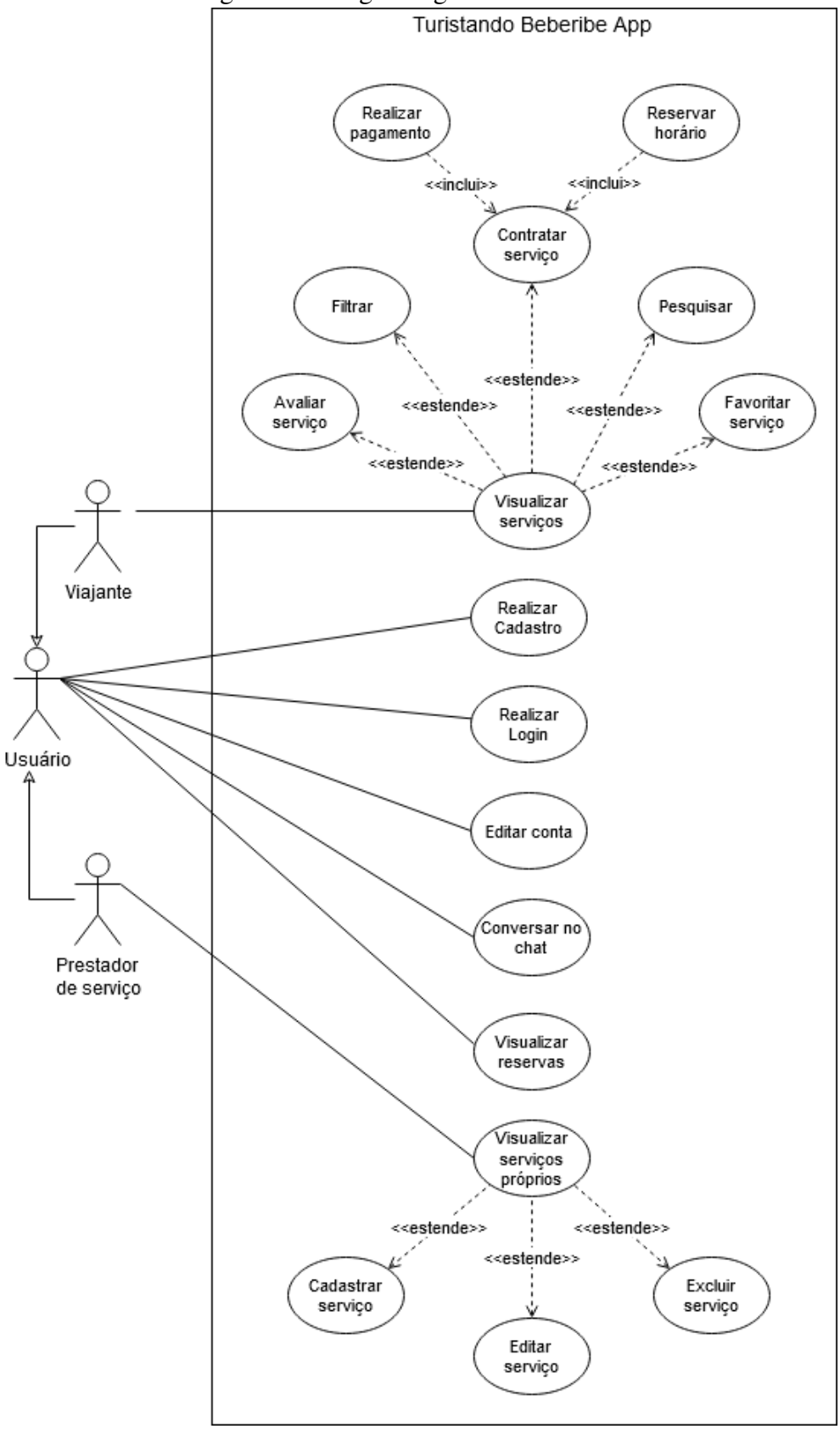
O Turistando Beberibe tem como proposta ser um aplicativo para contratação de serviços turísticos, como passeios e trilhas, focando no município de Beberibe. O propósito é mostrar aos viajantes as melhores experiências do local, eliminando qualquer dificuldade na hora de pesquisar e contratar esse tipo de serviço. O aplicativo também tem como objetivo beneficiar e dar uma maior visibilidade aos prestadores de serviços locais, sendo mais um meio de divulgação dos seus trabalhos e uma potencial fonte de novos clientes.

Apesar de existirem aplicações com algumas características semelhantes as do Turistando Beberibe, a sua proposta de valor e principal diferencial é oferecer experiências especificamente para a região de Beberibe, conectando diretamente os viajantes que estão interessados em passeios turísticos aos prestadores de serviço da região.

4.6 Funcionalidades e Casos de uso

Logo abaixo, na Figura 4, é apresentado o diagrama de casos de uso do Turistando Beberibe. Nele, é possível observar todas as funcionalidades do aplicativo de forma geral, assim como os atores que farão uso do mesmo. A seguir, são descritos os casos de uso presentes no diagrama.

Figura 4 – Diagrama geral de casos de uso



Fonte: equipe Ethos

4.6.1 Caso de uso: Realizar Login

Logo ao entrar no aplicativo, o usuário é apresentado à tela de login, na qual deve preencher os campos de e-mail e senha e clicar no botão Entrar. Após o login ser realizado com sucesso, o usuário é redirecionado para a tela inicial do aplicativo, que pode variar de acordo com o tipo de usuário (viajante ou prestador).

4.6.2 Caso de uso: Realizar Cadastro

O usuário poderá efetuar o login no aplicativo somente se estiver cadastrado na base. Para isso, existe um botão na tela de login do aplicativo que leva ao formulário de cadastro. Após preencher os seus dados corretamente e sinalizar o tipo de conta, sendo viajante ou prestador, o usuário é cadastrado e redirecionado para a tela inicial do aplicativo.

4.6.3 Caso de uso: Editar conta

Após estar logado, o usuário pode editar seus dados de perfil clicando em um botão disponível no menu de navegação do aplicativo.

4.6.4 Caso de uso: Conversar no chat

Caso o usuário do tipo viajante deseje conversar com o prestador de serviços para tirar dúvidas, existe um botão na tela do anúncio do serviço que leva a um chat. Todas as conversas ficão disponíveis para os dois usuários na aba de Conversas, que pode ser acessada em um botão no menu de navegação.

4.6.5 Caso de uso: Visualizar serviços

O viajante será apresentado a uma tela que exibe todos os serviços disponíveis no aplicativo. Esta tela conta com um filtro para o usuário selecionar as características dos serviços que mais se encaixam com o seu gosto. Clicando no botão de ver detalhes em um anúncio, pode-se visualizar todas as suas informações e também salvá-lo como favorito para facilitar o seu acesso. Por último, o usuário poderá escolher o dia, fazer a reserva do serviço e realizar o pagamento.

4.6.6 Caso de uso: Visualizar reservas

O viajante pode visualizar os serviços reservados a qualquer momento, dessa forma, podendo acompanhar os status de suas reservas, que podem ser: aguardando confirmação, confirmado, concluído ou rejeitado. Após a realização do serviço agendado, o botão de avaliação da experiência fica disponível.

Já os prestadores de serviço podem visualizar as reservas dos seus anúncios, tendo a opção de aceitar o agendamento ou não, caso seu dia já esteja cheio, por exemplo.

4.6.7 Caso de uso: Visualizar serviços próprios

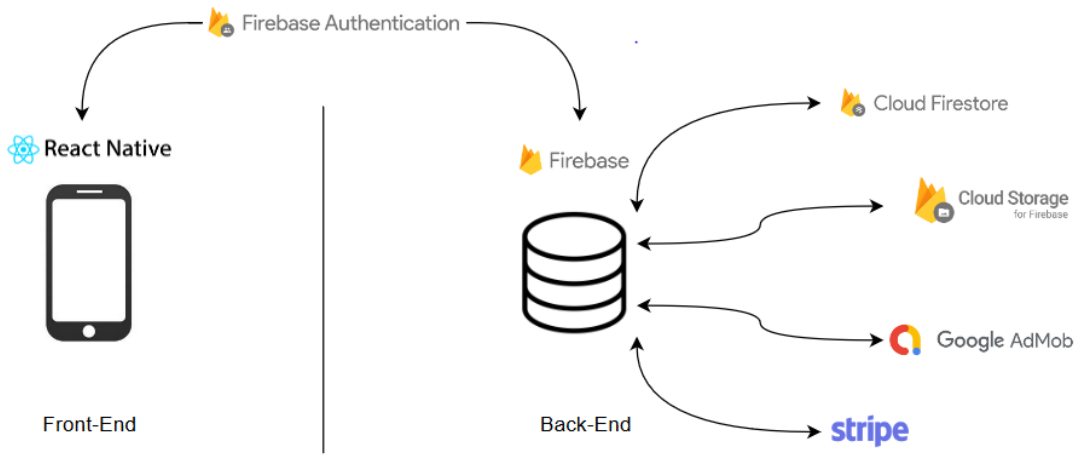
O usuário prestador de serviço pode visualizar seus serviços, cadastrar novos e editar ou excluir os existentes.

4.7 Visão Geral da Aplicação

Uma visão geral dos elementos da aplicação é mostrada na Figura 5. Nela, é possível ver como os componentes do sistema se comunicam, dando uma ideia geral de como a aplicação funciona.

A solução é composta por um aplicativo móvel desenvolvido em React Native, com versões para Android e iOS, e por um *back-end* feito utilizando o Firebase, no qual são feitos a autenticação, a persistência dos dados dos usuários e a integração com os serviços de exibição de anúncios e processamento de pagamentos.

Figura 5 – Visão geral dos elementos do sistema



Fonte: equipe Ethos

4.7.1 Aplicativo React Native

O *front-end* da solução é um aplicativo feito em React Native. Também é usado o Expo para acessar as APIs nativas do dispositivo sem precisar recorrer a código nativo, focando apenas no Javascript. Além disso, o Expo ajuda facilitando e acelerando testes, criação de *builds* e *deploys*.

4.7.2 Firebase

Todo o *back-end* é centralizado no Firebase[1], um serviço BaaS do Google. Diversos serviços e APIs dessa plataforma são utilizados, desde a autenticação até a integração com serviços de processamento de pagamentos.

4.7.3 Firebase Authentication

Para ter acesso ao aplicativo, o usuário tem que criar uma conta e efetuar o login. Para isso, foi usado o Firebase Authentication, que é uma API do Firebase para a autenticação de usuários. Ele oferece suporte à autenticação usando e-mail, número de telefone, redes sociais, como Google e Facebook, entre outros.

4.7.4 Firebase Cloud Firestore

O banco de dados Cloud Firestore foi escolhido para persistir os dados dos usuários. O motivo da escolha se deu porque o Cloud Firestore é robusto, conta com consultas rápidas e avançadas, tem uma boa escalabilidade e também permite sincronizar as informações em tempo real com todos os clientes conectados.

4.7.5 Firebase Cloud Storage

Para o armazenamento de arquivos dos usuários, como imagens e vídeos, é utilizado o Firebase Cloud Storage.

[1] Firebase - https://firebase.google.com/?hl=pt-br

4.7.6 Google AdMob

Uma das fontes de receita planejadas para o Turistando Beberibe é a exibição de anúncios. Para isso, decidiu-se usar o Google AdMob[2] integrado ao Firebase. O Admob disponibiliza anúncios de milhões de anunciantes do Google. As propagandas podem ser exibidas de diversas formas, como *banners*, anúncios ou vídeos.

4.7.7 Stripe

A principal fonte de receita prevista para o Turistando Beberibe é a taxa cobrada em cima dos valores das reservas feitas no aplicativo. A equipe escolheu usar a Stripe integrada ao Firebase para tornar isso possível. A Stripe[3] é uma plataforma que operacionaliza e realiza processos de pagamentos. Todo o processo de pagamento será delegado à Stripe, que oferece tanto o crédito quanto o débito como formas de pagamento, além de disponibilizar o histórico de pagamentos e a função de reembolso.

4.8 Interface da aplicação

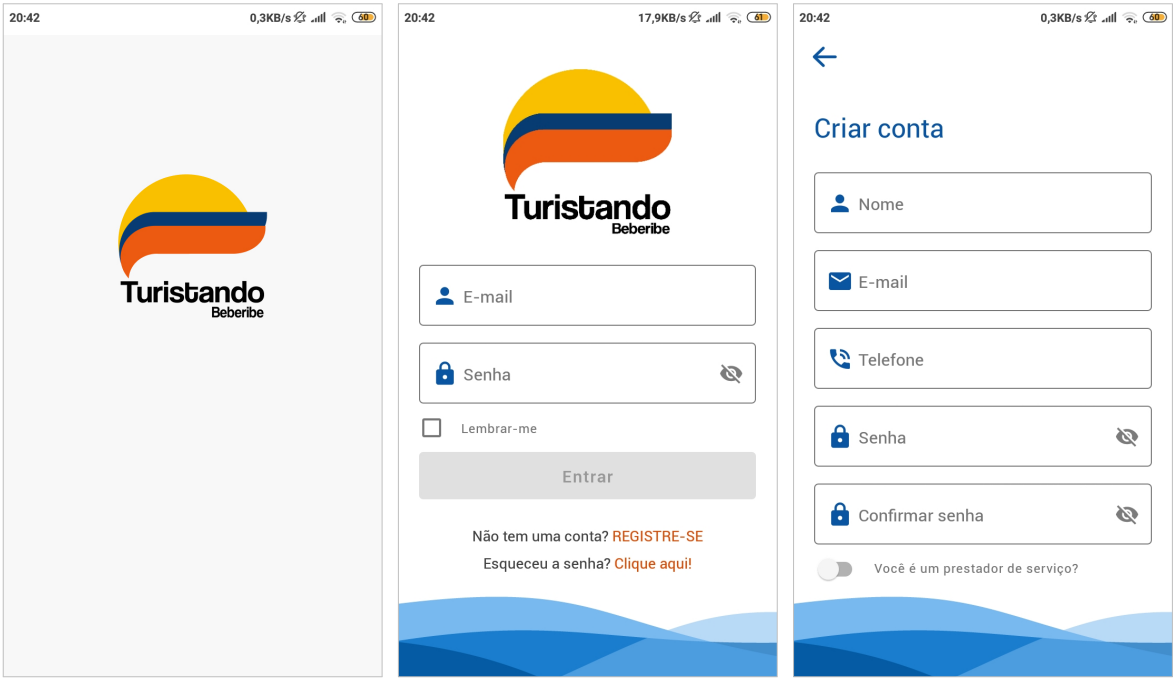
A seguir, são apresentadas as principais telas produzidas pela equipe Ethos para compor o aplicativo Turistando Beberibe. Para auxiliar na criação das telas, foram utilizadas recomendações e padrões do Material Design[4].

As telas iniciais da aplicação são mostradas na Figura 6. A primeira é a *Splash Screen*, apresentada durante a inicialização do aplicativo. Após o término do carregamento inicial, o usuário é levado para a tela de login. Caso ele não possua uma conta, basta clicar no botão “Registre-se” para ser direcionado para a tela de cadastro. Além desta tela pedir os dados cadastrais, ela também solicita ao usuário para informar se a sua conta será do tipo prestador de serviço, caso não seja, será automaticamente do tipo viajante.

---

[2] Google AdMob - https://admob.google.com/intl/pt-BR_br/home/
[3] Stripe - https://stripe.com/en-br
[4] Material Design - https://material.io/design

Figura 6 – Telas iniciais do Turistando Beberibe

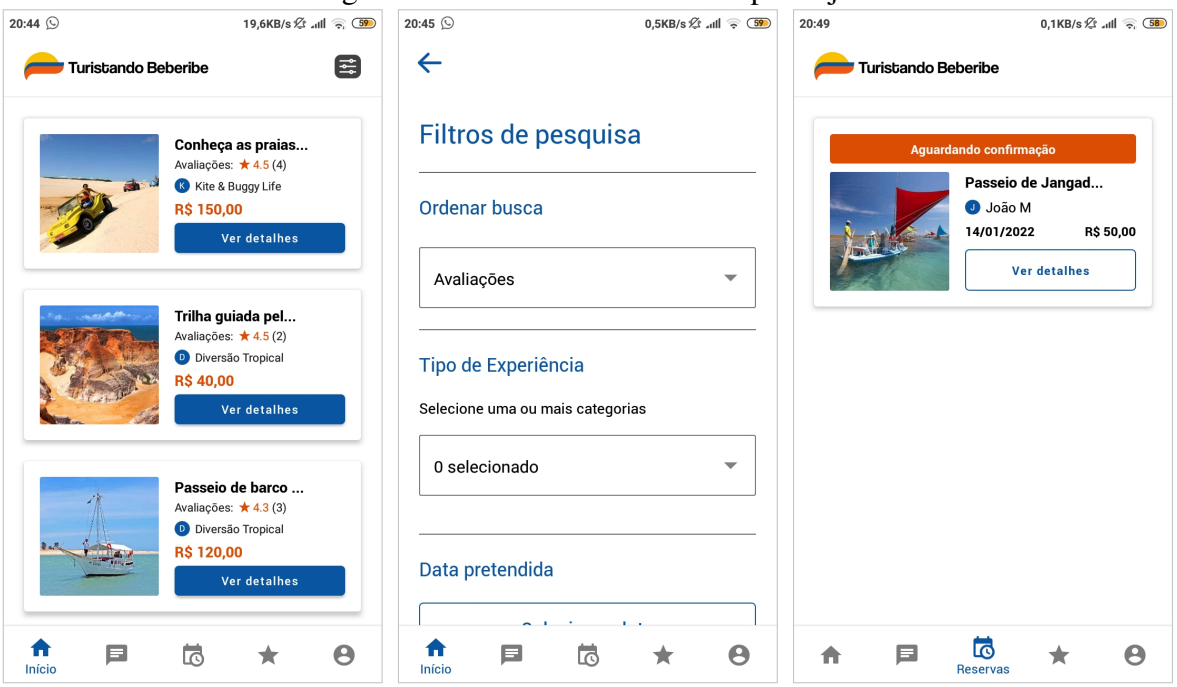


(a) *Splash screen* (b) Tela de login (c) Tela de cadastro

Fonte: imagens geradas a partir do aplicativo desenvolvido equipe Ethos

A Figura 7 mostra as telas exibidas para usuários do tipo viajante. A tela inicial, na qual se encontram os anúncios disponíveis no aplicativo, é apresentada na Figura 7a. Já a Figura 7b mostra o filtro de pesquisa que pode ser utilizado pelos viajantes para facilitar as suas buscas. Após a reserva de um serviço, o usuário pode acompanhar o status do seu agendamento na aba Reservas, que é exemplificada na Figura 7c.

Figura 7 – Telas dos usuários do tipo viajante

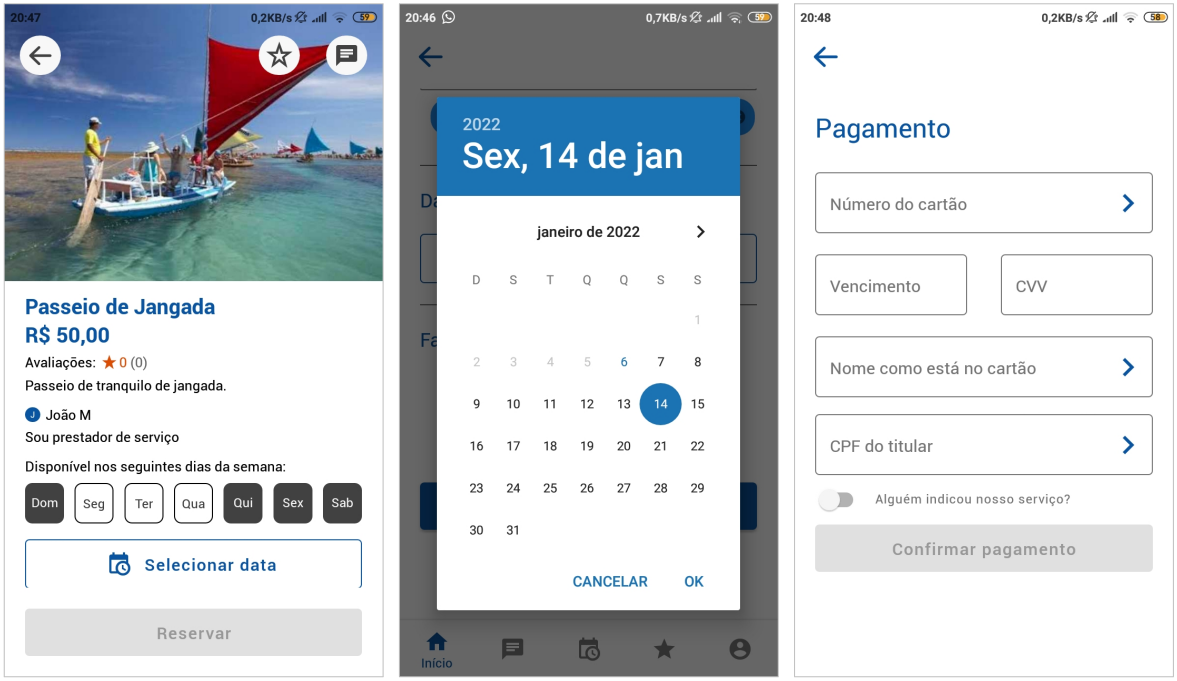


(a) Tela inicial viajantes (b) Filtro dos anúncios (c) Tela de reservas

Fonte: imagens geradas a partir do aplicativo desenvolvido equipe Ethos

O fluxo de reserva de uma experiência é apresentado na Figura 8. Neste fluxo o viajante pode obter informações mais detalhadas sobre o anúncio e o anunciante, ver as avaliações da experiência, escolher a data de agendamento e seguir para o pagamento.

Figura 8 – Telas do fluxo de reservas experiência

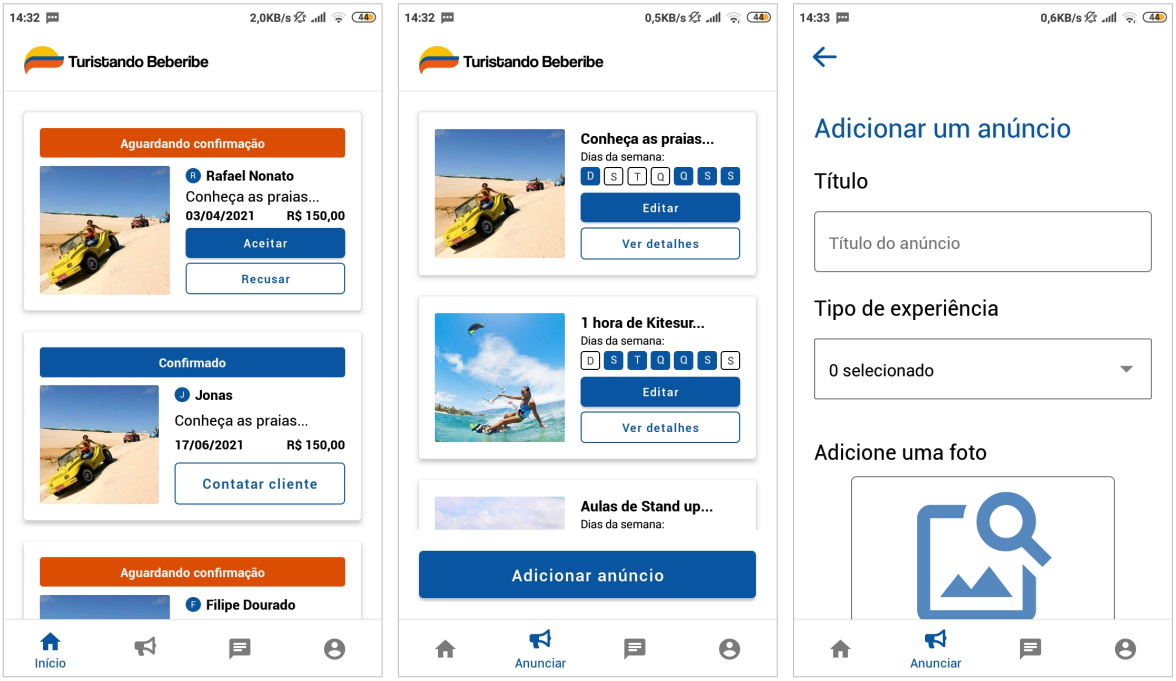


(a) Tela do anúncio (b) Calendário para escolher data (c) Tela de pagamento

Fonte: imagens geradas a partir do aplicativo desenvolvido equipe Ethos

Na Figura 9, são exibidas as principais telas disponíveis para os prestadores de serviço. A primeira mostra todas as solicitações de reservas para um serviço, que podem ser aceitas ou não. Já na Figura 9b, o usuário anunciante pode visualizar seus anúncios já criados, editar ou criar um novo. Estas duas últimas ações são realizadas da tela da Figura 9c.

Figura 9 – Telas dos usuários do tipo prestador de serviços



(a) Tela inicial prestador serviços (b) Anúncios criados (c) Cadastrar novo anúncio

Fonte: imagens geradas a partir do aplicativo desenvolvido equipe Ethos

5 DESENVOLVIMENTO DO APLICATIVO

Neste capítulo, é mostrado todo o processo de desenvolvimento do aplicativo Turistando Beberibe, desde as configurações iniciais, até o seu *build* e *deploy*. A ideia é apresentar de forma detalhada cada etapa, expondo as vantagens e limitações trazidas pelas ferramentas escolhidas.

Das funcionalidades e integrações previstas no capítulo anterior, não foi possível desenvolver o chat no aplicativo e integrar com o AdMob e Stripe, serviços de propaganda e pagamento, respectivamente. Isso aconteceu devido à complexidade dessas funcionalidades e à falta de tempo durante o desenvolvimento do projeto. No entanto, foram criadas telas para simular como seriam essas funcionalidades.

5.1 Início do projeto

A forma mais simples e rápida de se criar um aplicativo com React Native, segundo a sua própria documentação (REACTJS, 2021b), é através do Expo. Ele é indicado como a melhor alternativa quando não se tem uma equipe muito experiente no desenvolvimento móvel, pois consegue encapsular a aplicação, retirando a necessidade de configurações mais robustas, e ao mesmo tempo entrega a grande maioria dos recursos disponíveis no React Native. Essa opção foi considerada como a mais viável para o desenvolvimento do Turistando Beberibe, devido ao curto prazo de desenvolvimento e a limitações técnicas dos seus membros.

Para iniciar um projeto com o Expo, basta ter instalado o Node.js[1] e o Expo CLI. Com essas ferramentas, já é possível criar e executar uma aplicação. Também é possível visualizar o projeto e as alterações feitas durante o desenvolvimento em tempo real através de um emulador ou do aplicativo disponibilizado pelo próprio Expo, o Expo Go. Este aplicativo está disponível tanto para Android quanto para iOS. Nele é possível executar os projetos, sem a necessidade de gerar *builds* ou de instalação, em qualquer dispositivo, o que acelera muito a disponibilização das aplicações para testes.

5.1.1 Escolha do gerenciador de pacotes

Quando o Node.js é instalado, junto a ele também vem o Node Package Manager (NPM). Esta ferramenta de linha de código serve basicamente para o gerenciamento de pacotes

[1] Node.js - https://nodejs.org/en/

de projetos que funcionam no ambiente do Node.js. No entanto, apesar de ser o gerenciador de pacotes padrão do Node, o NPM começou a mostrar alguns problemas relacionados a desempenho e segurança (BALLERINI, 2021).

Com isso, o Facebook, com a ajuda de outras empresas mundialmente conhecidas, como o Google, criou o Yarn[2]. Esta ferramenta foi criada com a intenção de substituir o NPM, tendo a proposta de ser mais eficiente e segura (BALLERINI, 2021). Por este motivo, o Yarn foi adotado como o gerenciador de pacotes.

5.1.2 Criação e primeira execução do projeto

Escolhido o gerenciador de pacotes, o próximo passo é instalar o Expo-CLI no sistema. Para isso, primeiramente é necessário já ter o Node.js instalado, preferencialmente em sua versão estável mais recente. Logo após, basta executar o seguinte comando no terminal para instalar o Expo globalmente:

```
yarn global add expo-cli
```

Com o Expo e suas dependências instaladas, já é possível criar um projeto React Native com o comando abaixo:

```
expo init turistando_beberibe
```

É importante ressaltar que a parte do comando “turistando_beberibe” é o lugar onde deve ser colocado o nome do projeto.

Após a execução do comando acima, é apresentado o seguinte menu com opções de configurações iniciais do projeto:

```
    Choose a template: - Use arrow-keys. Return to submit.
    ----- Managed workflow -----
>   blank               a minimal app as clean as an empty canvas
    blank (TypeScript)  same as blank but with TypeScript configuration
    tabs (TypeScript)   several example screens and tabs using react-
       navigation and TypeScript
    ----- Bare workflow -----
```

[2] Yarn - https://yarnpkg.com/

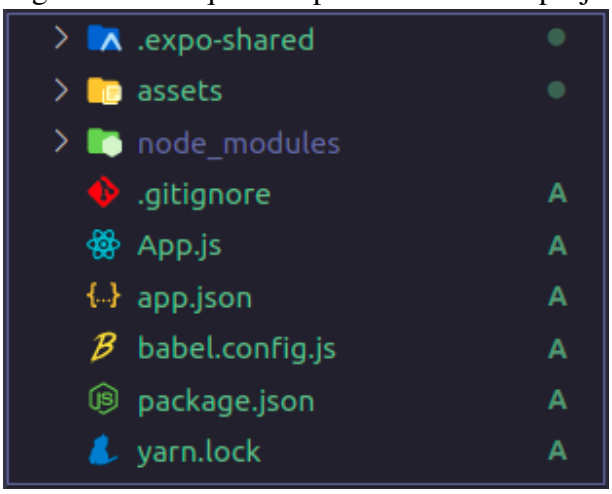
```
    minimal              bare and minimal, just the essentials to get you
        started
```

É esse um momento importante, pois pode impactar o resto do desenvolvimento da aplicação. Primeiramente, é necessário escolher o tipo de *workflow* (fluxo de trabalho), que pode ser *managed* ou *bare*. No *managed*, é necessário lidar somente com o JavaScript e o ambiente do Expo. Já o *bare* funciona basicamente como um projeto em React Native puro, no qual é possível lidar com toda a parte nativa do código. Por esse motivo, o Expo fica mais limitado no *bare workflow* (EXPO, 2021b).

Juntamente com a escolha do *workflow*, também é necessário escolher o *template*, que pode ser *blank* (em branco), com as configurações necessárias para usar JavaScript ou TypeScript, e *tabs*, que já conta com exemplos de telas.

A equipe decidiu usar o *workflow managed*, para usufruir de todos os benefícios do Expo, e o *template blank*, para iniciar o projeto somente com a configuração e dependências iniciais. Após essa escolha, o projeto é criado inicialmente com os arquivos e pastas mostrados na Figura 10.

Figura 10 – Arquivos e pastas iniciais do projeto



Fonte: elaborado pelo autor

Para executar a aplicação, é necessário ir para o diretório raiz e executar o comando abaixo:

```
expo start
```

Feito isso, o Expo iniciará um servidor de desenvolvimento local para o projeto e

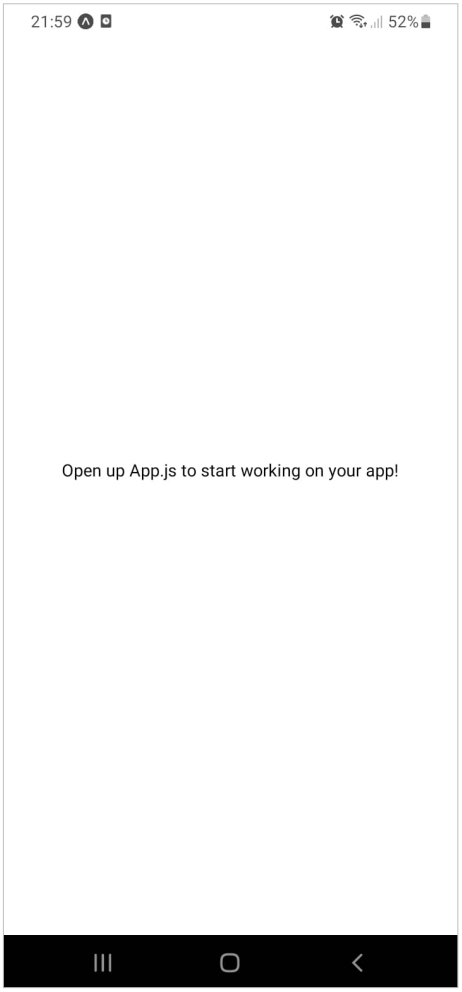
exibirá no terminal a interface mostrada na Figura 11. Através dessa interface é possível abrir os emuladores do Android e iOS, reiniciar o aplicativo, ver mensagens de aviso, erro e *debug*, entre outras opções. Além disso, é mostrado um *QR Code* para ser lido através do aplicativo Expo Go e se conectar ao projeto em tempo real. É importante ressaltar que o computador e o *smartphone* utilizados devem estar conectados à mesma rede para funcionar.

Figura 11 – Interface de interação com o servidor de desenvolvimento

Fonte: elaborado pelo autor

Ao ler o *QR Code* gerado ou executar um simulador, é possível acessar e visualizar a aplicação em tempo real, assim como as alterações que são feitas ao longo do desenvolvimento. Isso acontece porque o servidor fica observando as alterações realizadas no código e as reflete no aplicativo, atualizando somente os pontos da interface que foram modificados. Na Figura 12 é possível ver a tela inicial de um projeto Expo recém-criado com o *template blank*.

Figura 12 – Tela inicial de um projeto Expo



Fonte: elaborado pelo autor

5.1.3 Configuração do ESLint

O ESLint[3] é uma ferramenta para padronização de códigos JavaScript. Através dela é possível adotar padrões de código que todos os desenvolvedores do projeto devem seguir. Dessa forma, a aplicação fica mais consistente, menos suscetível a erros e mais fácil de ser mantida (ESLINT, 2021).

Para instalar o ESLint em qualquer aplicação JavaScript, é necessário ter o Node.js instalado, preferencialmente em sua versão mais recente, e o projeto também já precisa ter o arquivo *package.json* configurado. Como o Expo já gera este arquivo na criação do projeto, basta executar o comando a seguir para instalar o ESLint:

```
yarn add eslint --dev
```

[3] ESLint - https://eslint.org/

É necessário criar um arquivo de configuração após a instalação. Isso pode ser feito manualmente ou através do comando abaixo utilizando a *flag* - *-init*, que é a forma mais simples e rápida (ESLINT, 2021).

```
yarn run eslint --init
```

Depois da conclusão do comando acima, é gerado um arquivo *.eslintrc.{js,yml,json}* no diretório do projeto. Através deste arquivo é possível configurar os padrões e regras a serem adotados durante a escrita do código. Um exemplo dessas configurações pode ser vista do Código-fonte 1.

Código-fonte 1 – Exemplo do arquivo de configuração do ESLint

```
{
    "env": {
        "browser": true,
        "es2021": true
    },
    "extends": [
        "plugin:react/recommended",
        "airbnb"
    ],
    "parser": "babel-eslint",
    "plugins": [
        "react"
    ],
    "rules": {
      "linebreak-style": "off",
      "semi": "off"
    }
}
```

5.1.4 Gerenciamento de versão

Para o compartilhamento e gerenciamento das versões do código-fonte, decidiu-se utilizar o Github, que é uma das principais plataformas para este fim. Foi criado um repositório público para o projeto, sendo possível acessá-lo através do *link* (https://github.com/filipedf1997/turistando-app).

5.2 Sistema de rotas

Para um usuário conseguir utilizar o Turistando Beberibe, ele precisa se autenticar com uma conta válida. Sendo assim, o aplicativo foi dividido entre uma área pública e outra privada. A primeira área é composta pelas telas de login, cadastro e recuperação de senha. Já a parte privada possui todas as outras funcionalidades da aplicação.

O primeiro arquivo executado em projetos React Native, é o *App.js*. Este arquivo é responsável por inicializar e carregar configurações que vão estar disponíveis para todo o resto da aplicação, como tema e fontes. Além disso, por ser o ponto inicial, ele contém todas as rotas de telas disponíveis no sistema. Por o aplicativo possuir uma quantidade de telas considerável, sendo 27 no total, as rotas foram criadas em arquivos separados com o intuito de não sobrecarregar o *App.js*, tornando-o difícil de manter. Um exemplo do *App.js* é apresentado no Código-fonte 2.

Código-fonte 2 – Exemplo do arquivo App.js

```
1  import React from 'react'
2  import { Provider } from 'react-native-paper'
3  import BaseNavigator from './src/navigators/BaseNavigator'
4  import theme from './src/theme/theme'
5
6  export default function App() {
7    return (
8      <Provider theme={theme}>
9        <BaseNavigator />
10     </Provider>
11   )
12 }
```

5.2.1 Rotas públicas e privadas

Como dito anteriormente, o Turistando Beberibe possui uma área pública e outra privada. Essa diferenciação é feita com base na autenticação do usuário e está contida no componente *BaseNavigator*. Este componente é chamado dentro do *App.js*, como mostrado na Código-fonte 2, sendo o responsável por alternar entre a rota autenticada e a pública. Para isso, ele observa se o estado *userStore.token* está preenchido. O *token* de acesso ao aplicativo é atribuído a esse estado no momento da autenticação, dessa forma, fazendo com que o usuário tenha acesso a área privada da aplicação.

Além da distinção entre a área pública e privada, também existe a diferença de rotas de acordo com o perfil do usuário, podendo ser viajante ou prestador de serviços. Essa diferença é necessária porque as telas e funcionalidades disponíveis para cada tipo de usuário são distintas em parte.

Código-fonte 3 – Componente *BaseNavigator*

```
1  import React from 'react'
2  import { observer } from 'mobx-react'
3  import { NavigationContainer } from '@react-navigation/
       native'
4  import TravelerNavigator from './TravelerNavigator'
5  import ProviderNavigator from './ProviderNavigator'
6  import AuthNavigator from './AuthNavigator'
7  import { useStores } from '../hooks/useStores'
8
9  const BaseNavigator = observer(() => {
10   const { userStore } = useStores()
11
12   return (
13     <NavigationContainer>
14       {
15         userStore.token
16           ? (userStore.user.isProvider
17                 ? <ProviderNavigator />
```

```
18                    : <TravelerNavigator />)
19              : <AuthNavigator />
20          }
21        </NavigationContainer>
22      )
23  })
24
25  export default BaseNavigator
```

O Código-fonte 3 mostra que quando o estado *userStore.token* não está preenchido, a rota pública *AuthNavigator* é selecionada, caso contrário, é feita outra verificação para saber o tipo de usuário se baseando no estado *userStore.user.isProvider*. Se este estado for verdadeiro, a rota dos prestadores de serviços (*ProviderNavigator*) é selecionada, do contrário, a rota escolhida é a dos viajantes (*TravelerNavigator*).

5.3 Construção das telas

Seja em qual área do projeto for (pública, privada, viajantes e prestador de serviços), todas as telas seguem basicamente o mesmo padrão de construção. Essa prática de desenvolvimento foi adotada para que todos os membros envolvidos na programação do aplicativo fossem capazes de entender e modificar facilmente qualquer tela ou componente. Abaixo, no Código-fonte 4, é apresentada a estrutura básica de uma tela do Turistando Beberibe para exemplificar o padrão adotado.

Código-fonte 4 – Componente *ChangePassword*

```
1  import React from 'react'
2  import React, { useState } from 'react'
3  import { observer } from 'mobx-react'
4  import { StyleSheet } from 'react-native'
5  import { Text } from 'react-native-paper'
6  import { Container, Content, HeaderBar } from '../../
       components'
7  import MyAccountStore from './store/MyAccountStore'
```

```
8
9  const ChangePassword = observer(({ navigation }) => {
10   const [store] = useState(() => new MyAccountStore())
11
12   return (
13     <Container>
14       <HeaderBar onPress={navigation.goBack} />
15       <Content>
16         <Text style={styles.title}>
17           {`Ola ${store.name}`}
18         </Text>
19         ...
20       </Content>
21     </Container>
22   )
23 })
24
25 const styles = StyleSheet.create({
26   title: {
27     fontSize: 25,
28     marginVertical: 20,
29   }
30 })
31
32 export default ChangePassword
```

Através do código mostrado acima, é possível observar o funcionamento de um componente no React Native. As primeiras linhas são utilizadas para fazer as importações necessárias, que podem ser bibliotecas externas, estilos e até mesmo outros componentes. É importante ressaltar que em toda tela criada no React Native, sempre é preciso importar o React. Além disso, os recursos importados em um componente se referem somente a ele, dessa forma, cada um deve conter suas próprias importações.

Depois de importar os recursos necessários, é criado o componente em si, neste caso, o *ChangePassword*. Atualmente, a forma mais recomendada de se fazer um componente no React Native é através de uma função. Pode-se atribuir uma função a uma variável, igual ao exemplo apresentado no Código-fonte 4, ou implementar uma função JavaScript comum, com a palavra reservada *fuction*. Toda a lógica é escrita no corpo do componente funcional. Esta lógica pode ser composta pela implementação de estados e funções para manipulá-los ou tratar eventos de interface, métodos do ciclo de vida do componente, como o *useEffect*, e requisições para APIs. Após a lógica, a função deve retornar o componente, escrito em JSX, que será renderizado em tela.

Ainda no Código-fonte 4, é possível observar a criação de um objeto *StyleSheet*, que é responsável por armazenar os estilos do componente. Cada propriedade deste objeto equivale a uma classe CSS. Ao final do código, o componente é exportado para ficar disponível para o resto da aplicação.

5.3.1 Gerenciamento de estados

Os dados não estáticos de um componente ficam armazenados em *props* e *states*, citado anteriormente como estados. As *props* são passadas de um componente pai para o filho. Este, por sua vez, recebe suas *props* como um parâmetro da função implementada na criação do componente. Já os *states* são criados dentro de cada componente e geralmente são utilizados para armazenar valores que podem sofrer alterações durante o ciclo de vida de um componente, diferentemente das *props*, que armazenam valores normalmente não modificáveis.

Um dos desafios de uma aplicação *front-end*, incluindo o React Native, é o gerenciamento de estados. Como dito acima, componentes podem receber *props* de componentes superiores. Se uma árvore de componentes possuir vários ramos e for necessário compartilhar dados através de *props* de um para o outro, gerenciar e manter essa distribuição de dados pode se tornar uma tarefa muito complexa.

Para auxiliar com essa problemática, surgiram diversas ferramentas com o objetivo de simplificar e tornar o gerenciamento de estados de uma aplicação mais escalável. A biblioteca Mobx[4] foi a escolhida para auxiliar nessa questão.

Assim como outras bibliotecas de gerenciamento que seguem a arquitetura Flux[5], o

[4] Mobx - https://mobx.js.org/README.html
[5] Arquitetura Flux - https://facebook.github.io/flux/

Mobx trabalha com o conceito de *stores*. O principal objetivo de uma *store* é retirar toda a lógica e gerenciamento de estados de dentro dos componentes (MOBX, 2021). Dessa forma, o sistema fica mais desacoplado, reutilizável e testável, uma vez que os componentes ficam responsáveis somente pela renderização do que será mostrado em tela, enquanto as *stores* ficam encarregadas pela lógica.

Uma *store* Mobx é feita utilizando uma classe JavaScript, na qual as suas propriedades atuam como os *states* e seus métodos como as funções de um componentes. Para um componente poder usufruir de uma *store*, dessa forma podendo utilizar seus *states* e funções, ele deve ser envolto pela função especial *observer*, disponibilizada pelo Mobx. Usando essa função, o componente passa a observar as alterações sofridas pelos *states* e consegue renderizá-las na tela. É possível ver o uso do *observer* na linha 9 do Código-fonte 4.

Para utilizar uma *store*, basta importá-la e criar uma instância dela dentro do componente, como mostrado na linha 10 do Código-fonte 4. A partir desse momento, o componente pode acessar os *states* e funções da *store* através da instância criada, que funciona como um objeto JavaScript.

No Turistando Beberibe foi utilizado o conceito de *stores* globais e locais, em que as globais ficam disponíveis para todos os componentes da aplicação e as locais se restringem ao fluxo nas quais foram criadas, por exemplo, o fluxo de alterar os dados da conta. No Código-fonte 5, é apresentado um exemplo de uma *store*.

Código-fonte 5 – Exemplo de uma *store* Mobx

```
1  import { makeAutoObservable } from 'mobx'
2
3  class UserStore {
4    user = null
5    token = null
6
7    getUserInfo() {
8      return `/${this.user} #${this.token}`
9    }
10
11   constructor() {
```

```
12      makeAutoObservable(this)
13    }
14  }
15
16  export default UserStore
```

5.3.2 Componentes

Em React Native, desde uma tela inteira até um único botão são componentes. No entanto, para manter a aplicação mais escalável e fácil de ser mantida, é recomendado agrupar os elementos da tela em pequenos componentes, os quais podem ser reusados em outras partes. Para agrupá-los, foi criada uma pasta de nome *components* no projeto. Atualmente, esta pasta conta com 23 componentes variados, como: botões, caixas de texto, *cards* e cabeçalhos.

Na construção de alguns componentes, foi utilizada a biblioteca React Native Paper[6]. Esta biblioteca possui uma rica coleção de componentes já prontos, todos seguindo as recomendações do Material Design, que foi o padrão de design usado no Turistando Beberibe.

Dentre os componentes do aplicativo, é importante ressaltar dois, o *Container* e o *Content*. Eles servem como base para a construção de praticamente todas as telas da aplicação. O *Container* envolve todo o componente, incluindo o cabeçalho e o *Container*, que por sua vez abriga o conteúdo rolável da tela.

5.4 Firebase

Como citado na seção 4.7 sobre o Turistando Beberibe, o Firebase foi utilizado como o *back-end* do aplicativo. Dos vários serviços disponíveis no Firebase, foram usados os de autenticação, banco de dados e repositório de imagens.

Para utilizar os serviços, primeiro é necessário criar um projeto no Firebase utilizando uma conta Google. Depois, é preciso instalar o seu SDK na aplicação e fazer a sua configuração, fornecida pelo próprio Firebase. Esse processo é simples, como pode ser visto no Código-fonte 6.

Código-fonte 6 – Configuração do Firebase

[6] React Native Paper - https://callstack.github.io/react-native-paper/

```
import firebase from 'firebase'

const firebaseConfig = {
  apiKey: 'chave-da-api',
  authDomain: 'url-de-autenticacao',
  projectId: 'id-do-projeto',
  storageBucket: 'url-repositorio-imagens',
  messagingSenderId: 'id-de-mensageria',
  appId: 'id-da-aplicacao',
  measurementId: 'id-de-monitoramento',
}
firebase.initializeApp(firebaseConfig)

export default firebase

export const db = firebase.firestore()

export const storage = firebase.storage().ref()
```

Após criar o objeto de configuração *firebaseConfig*, a aplicação do Firebase é inicializada e exportada, dessa forma, ficando disponível para ser usada em todo o projeto. Também são criadas e exportadas as instâncias do bancos de dados e do repositório de imagens, *firebase.firestore()* e *firebase.storage()*, respectivamente.

5.4.1 Autenticação

O serviço de autenticação do Firebase oferece vários métodos de *login*, como: e-mail, número celular, Google, Facebook, dentre outros provedores externos. O método escolhido pela equipe foi o de e-mail, por ser muito utilizado em outras aplicações disponíveis no mercado. Esse serviço também oferece a possibilidade de recuperação de senha e verificação de e-mail após o cadastro. No entanto, apenas a primeira opção foi implementada no Turistando Beberibe.

O Firebase disponibiliza um painel de controle para o gerenciamento das contas

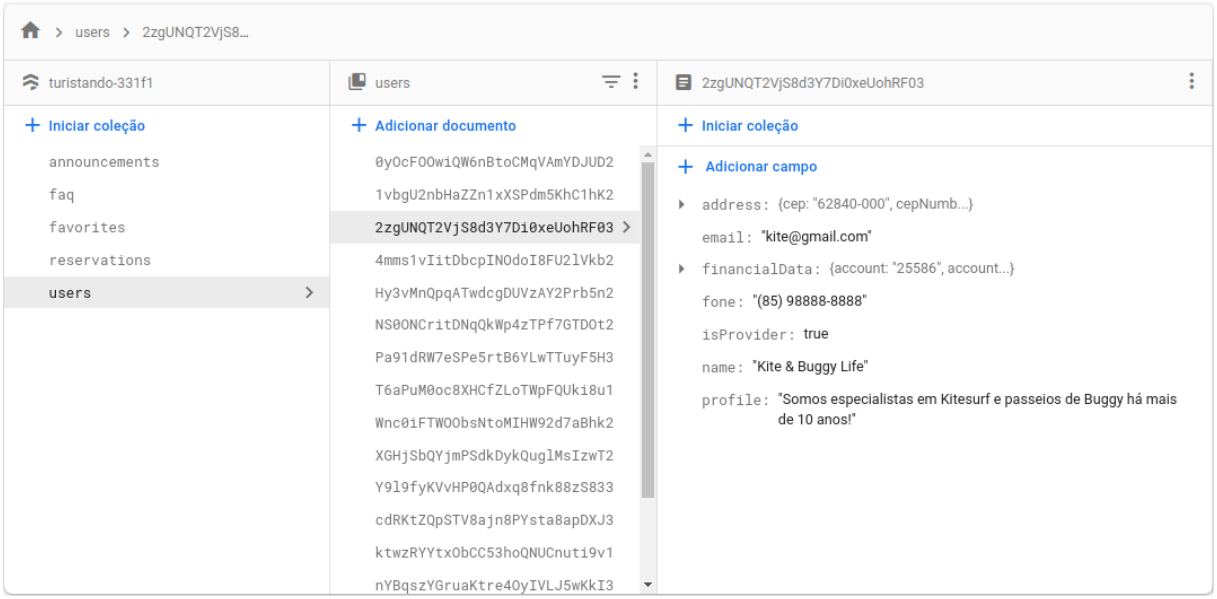
criadas no projeto. Através desse recurso, é possível criar, visualizar os dados, excluir ou desativar um usuário.

5.4.2 Banco de dados e Repositório de imagens

No Firebase existem dois bancos de dados, o Realtime e o Firestore. Foi decidido usar o Firestore por ele ser mais robusto e contar com mais recursos do que o Realtime. Além disso, ele possui um melhor desempenho, consultas mais avançadas e também é sincronizado em tempo real.

O Firestore é um banco de dados NoSQL que adota um modelo de coleções e documentos. As coleções funcionam como contêineres que armazenam documentos e são semelhantes a uma entidade em modelos de bancos relacionais. Cada coleção é composta por documentos, os quais são formados por propriedades chave-valor. Esses documentos suportam diversos tipos de dados, como: *strings*, números e objetos complexos (FIREBASE, 2021). O painel de gerenciamento do banco de dados do Turistando Beberibe é mostrado na Figura 13. Nele, é possível observar como funciona a estrutura e organização do Firestore. É importante ressaltar que os dados mostrados na imagem são fictícios, usados apenas para testes.

Figura 13 – Painel de gerenciamento do Firestore



Fonte: elaborado pelo autor

Já para o armazenamento de imagens, foi usado o Storage. O Firebase oferece esse serviço especificamente para o depósito de arquivos multimídia, como: imagens, vídeos e áudios.

No caso do Turistando Beberibe, apenas as imagens dos anúncios foram armazenadas, utilizando o formato de codificação Base64, sendo preciso criar apenas um repositório. Cada imagem foi nomeada com o id do seu respectivo anúncio para tornar possível a correta vinculação no momento da consulta.

Uma função que utiliza o Firestore e o Storage é mostrada no Código-fonte 7. Essa função está presente na *store* responsável pelo gerenciamento de anúncios, sendo encarregada pela criação de um novo.

Código-fonte 7 – Função responsável por criar um novo anúncio

```
async createAnnouncement() {
    try {
        this.isFetching = true

        const user = firebase.auth().currentUser
        this.announcement.ownerUID = user.uid
        this.announcement.ownerName = this.ownerName
        this.announcement.ownerDescription = this.
           ownerDescription
        const { id } = await db.collection('announcements')
           .add(this.announcement)
        await storage.child(`announcements/${id}`).put(this
           .photoBlob)

        this.isFetching = false
        return true
    } catch (error) {
        this.isFetching = false
        return false
    }
}
```

5.5 *Build* e *Deploy*

Por se tratar de uma aplicação *cross-platform*, é possível gerar dois *builds* diferentes, um para Android e outro para iOS, a partir do mesmo projeto. Isso pode ser feito utilizando o próprio Expo, pois ele permite gerar arquivos binários para serem submetidos nas lojas, dessa forma, disponibilizando o aplicativo ao público.

Na disciplina de Projeto Integrado II, a equipe gerou apenas o *build* do Android (*.apk*), para que a aplicação pudesse ser testada pelos professores. Naquele momento, não foi possível gerar o arquivo binário para iOS (*.ipa*), pois para isso, é necessário se autenticar com uma conta de desenvolvedor da Apple, podendo ser criada somente através do pagamento de US $99 anuais. Mesmo assim, a equipe conseguiu realizar testes em simuladores de iPhone através do aplicativo Expo Go.

Para tornar o presente trabalho mais abrangente, entendeu-se que seria necessário também gerar o *build* do iOS (*.ipa*). No entanto, apesar do professor e orientador deste relatório, Dr. Windson Viana de Carvalho, ter concedido uma conta de desenvolvedor Apple institucional, não foi possível gerar um arquivo binário *.ipa* do Turistando Beberibe por limitações técnicas do autor deste trabalho. Mesmo não sendo possível criar um *build* para o iOS, esse procedimento foi descrito para as duas plataformas.

No momento de escrita deste trabalho, ainda não foi realizado o *deploy* da aplicação nas lojas, mas o processo de como aconteceria também foi descrito.

5.5.1 Criação de builds

O processo de geração de *builds* através do Expo é similar para ambas as plataformas. É necessário possuir uma conta Expo e ter configurado corretamente o arquivo *app.json* do projeto, contendo no mínimo as seguintes informações: nome do aplicativo, ícone, versão, *url* para identificação da aplicação Javascript (*slug*), *package* e *bundleIdentifier*, que são *strings* de identificação dos aplicativos Android e iOS, respectivamente.

Após configurar o *app.json*, é possível gerar os *builds* através do seguinte comando:

```
expo build:android ou expo build:ios
```

Ao executar o comando acima, dependendo da plataforma escolhida, as opções apresentadas são diferentes. No caso do Android, é possível gerar um arquivo *.apk* ou *.aab*. O

.apk pode ser instalado diretamente em simuladores ou dispositivos físicos, sendo possível testar o aplicativo sem a intermediação do Expo Go. Além disso, esse tipo de arquivo também pode ser submetido para a loja do Google. No entanto, é recomendo enviar à loja um arquivo *.aab*, por ser melhor otimizado. Por isso, este tipo de *build* não pode ser instalado diretamente.

Semelhante ao processo ocorrido com o Android, duas opções são apresentadas na geração de *builds* para o iOS, sendo *arquive* (*.ipa*) e *simulator* (*.app*). O arquivo do tipo *simulator* serve apenas para ser instalado em simuladores, não podendo ser usado em dispositivos físicos, por questões de seguranças impostas pela Apple, nem ser submetido à loja. Para a última opção, é necessário gerar um *build .ipa*.

Após a escolha do tipo de arquivo, é necessário se autenticar com um conta de desenvolvedor Apple. Logo em seguida, deve-se criar os certificados de distribuição iOS. O Expo oferece a opção de gerar esses certificados automaticamente, sendo a opção mais recomendada, a não ser que a equipe de desenvolvimento já seja familiarizada com esse processo.

É importante frisar que o processo de *build* de ambas as plataformas acontece nos servidores do Expo. Ao final, os arquivos ficam disponíveis para download por 30 dias.

5.5.2 Deploy da aplicação

Para tornar o aplicativo disponível ao público, é necessário submeter os *builds* às lojas do Google e da Apple. Um *.apk* ou *.aab* pode ser enviado para o *Google Play* de qualquer sistema operacional, sendo necessário apenas a criação de um novo aplicativo no *Google Play Console*. Já o *build* de uma aplicação iOS pode ser submetido para a *Apple Store Connect* somente através de um *MacBook*, utilizando o *XCode* ou o *Transporter*. Ao ser enviado para a *Apple Store Connect*, o aplicativo fica disponível para ser testado através do *TestFlight*, uma aplicação iOS para testes, semelhante ao Expo Go.

Após a submissão, o aplicativo passa por uma avaliação de cada loja antes de ser liberado ao público. Esse processo costuma levar algumas horas, mas pode passar de um dia.

5.6 Limitações

Durante o desenvolvimento do Turistando Beberibe foram encontradas algumas dificuldades pela equipe. Já eram esperadas algumas limitações por ter sido usado o React Native juntamente com o Expo na criação da aplicação. O principal desafio relacionado ao aplicativo foi

achar uma biblioteca ou API para usar os calendários nativos dos dispositivos. Esse recurso foi necessário na funcionalidade de contratar um serviço, especificamente no momento da escolha da data. Não foi possível refletir os dias disponíveis do anúncio no calendário, impactando negativamente a usabilidade do aplicativo.

A equipe também se deparou com algumas limitações impostas pelo Firebase. A primeira foi no método de *login*. A ideia inicial era que o usuário pudesse entrar no aplicativo através de um nome de usuário criado por ele, porém, o serviço de autenticação do Firebase não disponibiliza essa opção. Dessa forma, o método de e-mail foi o escolhido. Além disso, apesar do banco de dados Firestore oferecer um bom sistema de consultas, ele não satisfez totalmente a filtragem dos anúncios presente no Turistando Beberibe. Mesmo podendo piorar o desempenho, foi preciso filtrar alguns dados no próprio aplicativo para finalizar essa funcionalidade.

6 ANÁLISE E AVALIAÇÃO DE USABILIDADE DO TURISTANDO BEBERIBE

Após o desenvolvimento do Turistando Beberibe, uma análise técnica do código aplicativo desenvolvido e avaliações de usabilidade foram realizadas. Os resultados desses processos são descritos a seguir.

6.1 Código

O React Native é muito similar ao React, utilizado para desenvolvimento web. Isso foi um grande facilitador para a equipe, uma vez que os membros não possuíam experiências sólidas com o desenvolvimento mobile. O uso do Expo também ajudou nesse aspecto, pois ele conseguiu abstrair uma parte considerável da configuração da aplicação, dessa forma, a equipe pôde se concentrar melhor na programação do aplicativo.

No geral, não foi necessário realizar grandes alterações para que a aplicação funcionasse corretamente no Android e iOS, este, através do Expo Go, visto que não foi possível realizar testes com o aplicativo instalado em um iPhone.

Foi preciso utilizar a API *Platform* do React Native em poucas ocasiões. Essa API informa em qual plataforma o aplicativo está executando, o que possibilita especificar comportamentos para cada tipo de ambiente. A API *Platform* foi utilizada para ajustar o espaçamento entre os componentes de interface e o teclado, quando ativo, de acordo com a plataforma. O Código-fonte 8 mostra como isso foi feito.

Código-fonte 8 – Uso da API *Platform* para ajustar o espaçamento entre os componentes e o teclado

```
<KeyboardAvoidingView
    behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
    {...keyboardAvoidingProps}
>
    ...
</KeyboardAvoidingView>
```

Da mesma forma, foi preciso ajustar o caminho das imagens (*uri*) obtidas para exibi-las corretamente no momento do cadastro de anúncios. Isso se fez necessário porque o iOS

adiciona à *uri* o prefixo *file://*. Um exemplo desse ajuste é apresentado no Código-fonte 9.

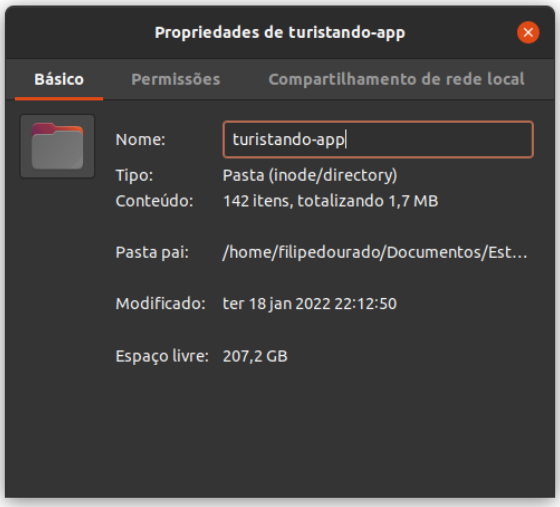
Código-fonte 9 – Uso da API *Platform* para ajustar a *uri* de imagens

```
1 const result = await ImagePicker.launchImageLibraryAsync
     ({...})
2 if (!result.cancelled) {
3     store.announcement.photo = Platform.OS === 'ios' ?
        result.uri.replace('file://', '') : result.uri
4 }
```

6.2 Arquivos gerados

Ao final do desenvolvimento do Turistando Beberibe, foram gerados 142 arquivos, ocupando 1,7 MB (*megabytes*) de armazenamento, podendo ser visto na Figura 14. A pasta *node_modules* foi desconsiderada na contabilização por conter uma grande quantidade de arquivos não necessários à aplicação. Dentre os arquivos gerados, destacam-se os componentes, no total de 23, e as telas, somando mais 27. Também existem arquivos de configurações, como os de rotas de navegação, e outros relacionados à identidade visual da aplicação, como imagens, ícones e fontes.

Figura 14 – Arquivos e pastas iniciais do projeto

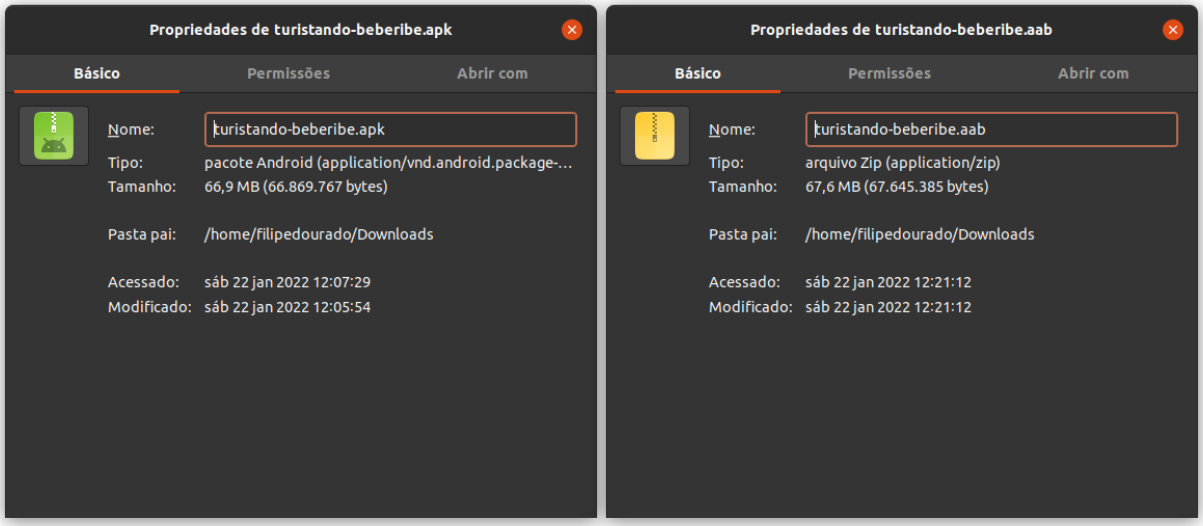


Fonte: elaborado pelo autor

6.3 *Builds*

Apesar de não ter sido possível gerar o arquivo binário *.ipa*, conseguiu-se gerar outros três *builds*, sendo dois para Android (*.apk* e *.aab*) e um para simuladores iOS (*.app*). Os arquivos gerados para Android tiveram tamanhos similares. O *.apk* ocupou 66,9 MB de armazenamento, enquanto o arquivo *.aab* ocupou 67,6 MB.

Figura 15 – Arquivos binários gerados para Android

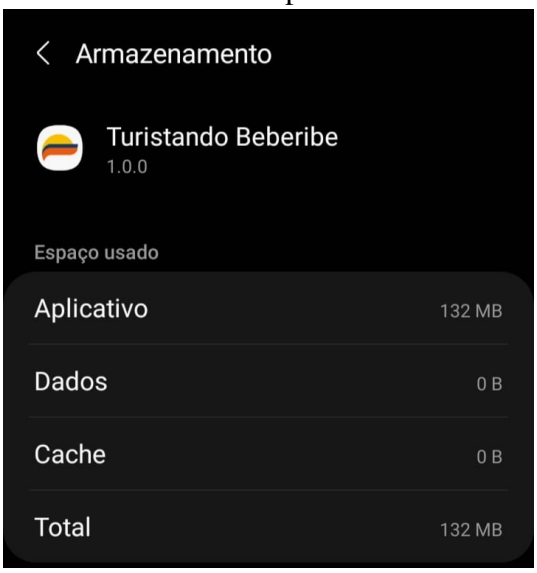


(a) Arquivo .apk (b) Arquivo .aab

Fonte: elaborado pelo autor

Após a instalação do *.apk* em um dispositivo, o aplicativo passou a ocupar 132 MB, praticamente o dobro do tamanho inicial. Esse aumento considerável se deve às dependências do Expo instaladas juntamente com a aplicação gerada.

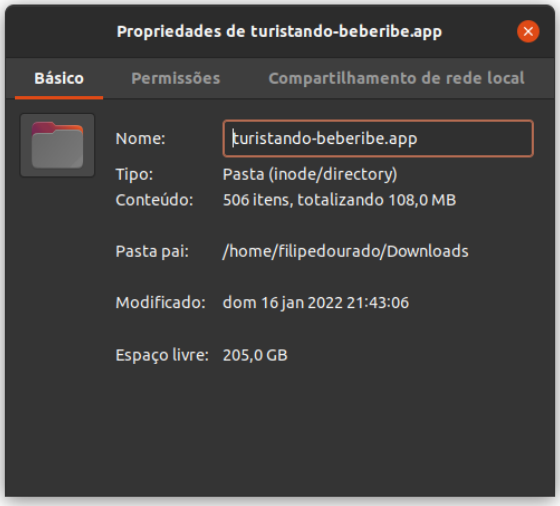
Figura 16 – Tamanho final do aplicativo instalado no Android



Fonte: elaborado pelo autor

O arquivo para simuladores iOS (*.app*) ocupou 108 MB de armazenamento. Não foi possível avaliar o espaço ocupado após a instalação, pois os simuladores não disponibilizam essa informação.

Figura 17 – Arquivo .app gerado para iOS



Fonte: elaborado pelo autor

6.4 Avaliação de Usabilidade

Uma pesquisa de usabilidade foi realizada, utilizando o método SUS, com o objetivo de avaliar a interface do Turistando Beberibe e verificar se ele foi bem aceito por potenciais usuários.

6.4.1 Estrutura da pesquisa

O questionário conta com dez questões, cada uma possuindo cinco opções de escolha, de acordo com a escala *Likert*: 1 (discordo plenamente), 2 (discordo), 3 (neutro), 4 (concordo) e 5 (concordo plenamente) (BROOKE *et al.*, 1996). O resultado final do SUS é obtido através do somatório de todas as respostas. Para as questões ímpares, subtrai-se 1 ponto da resposta do usuário. Já para as questões pares, deve-se subtrair a resposta de 5, ou seja, se a pontuação for 2, contabiliza-se 3. Após determinar todos os valores, é necessário somá-los e fazer a multiplicação do resultado por 2,5. Dessa forma, o montante final identifica o resultado do SUS, que deve variar de 0 a 100 (BROOKE *et al.*, 1996).

Os sistemas que conseguem atingir 90 pontos ou mais possuem a melhor usabilidade possível. Os que ficam entre 80 e 90 pontos têm a usabilidade classificada como excelente. Resultados entre 70 e 80 são tidos como bons, apesar de haver pequenas melhorias. Já pontuações entre 60 e 70 são consideradas *ok*, necessitando de melhorias consideráveis. Por fim, sistemas que não conseguem atingir pelo menos 60 pontos possuem um nível de usabilidade inaceitável (BANGOR *et al.*, 2009).

O SUS foi elaborado inicialmente na língua inglesa. No entanto, uma versão do questionário traduzida para o português foi utilizada neste trabalho, com o intuito de tornar a pesquisa mais acessível aos usuários. A Figura 18 apresenta as perguntas do SUS traduzidas para o português por Martins *et al.* (2015).

Figura 18 – Perguntas do SUS traduzidas para português

Original Item	Corresponding item in Portuguese
I think that I would like to use this system frequently.	Acho que gostaria de utilizar este produto com frequência.
I found the system unnecessarily complex.	Considerei o produto mais complexo do que necessário.
I thought the system was easy to use.	Achei o produto fácil de utilizar.
I think that I would need the support of a technical person to be able to use this system.	Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto.
I found the various functions in this system were well integrated.	Considerei que as várias funcionalidades deste produto estavam bem integradas.
I thought there was too much inconsistency in this system.	Achei que este produto tinha muitas inconsistências.
I would imagine that most people would learn to use this system very quickly.	Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto.
I found the system very cumbersome to use.	Considerei o produto muito complicado de utilizar.
I felt very confident using the system.	Senti-me muito confiante a utilizar este produto.
I needed to learn a lot of things before I could get going with this system.	Tive que aprender muito antes de conseguir lidar com este produto.

Fonte: Martins *et al.* (2015)

Além do questionário do SUS, foram feitas mais três perguntas para identificar os

perfis dos participantes da pesquisa, focando nos seguintes aspectos: faixa etária, familiaridade com o uso de *smartphone* e uso de aplicativos voltados ao turismo.

6.4.2 Participantes da pesquisa

Como mencionado anteriormente, o Turistando Beberibe possui dois públicos-alvo: viajantes e prestadores de serviços. No entanto, a pesquisa foi realizada apenas com pessoas com o perfil de viajante, dessa forma, a parte do aplicativo voltada aos prestadores de serviços não foi contemplada. Portanto, é fundamental que sejam realizadas pesquisas futuras com esse perfil de usuário com o intuito de complementar o presente trabalho.

A pesquisa contou com seis participantes e foi feita através de um questionário online. Cada pessoa foi convidada a participar de uma webconferência para realizar a avaliação.

6.4.3 Procedimento e Funcionalidades avaliadas

O foco da pesquisa foi avaliar as principais funcionalidades do aplicativo voltadas ao usuário do tipo viajante. As funções avaliadas foram: cadastro de uma nova conta, login na aplicação e filtragem e contratação de um serviço. Para que cada participante executasse os mesmos passos, foi dado um cenário de uso idêntico a todos (detalhado a seguir). Dessa forma, foi possível contemplar todas as funcionalidades citadas.

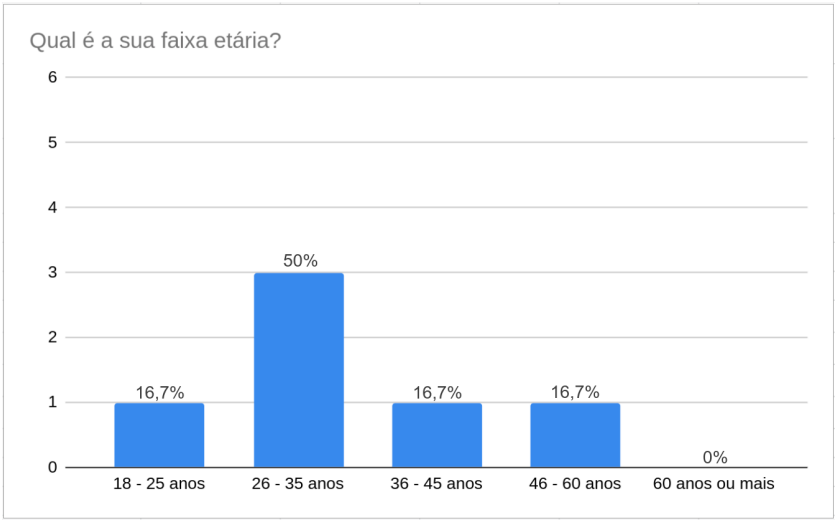
Cenário de uso: Após pesquisar na internet por plataformas que disponibilizam a contratação de serviços turísticos na cidade de Beberibe, você acaba encontrando o Turistando Beberibe e decide baixá-lo. Depois de realizar o cadastro na aplicação, você faz o login e começa a buscar por passeios de *buggy* entre os dias 28/01/2022 e 30/01/2022, período de sua estadia na cidade. Ao encontrar o anúncio que mais lhe agradar, faça a reserva e o pagamento do mesmo (com dados fictícios). Utilize os recursos do aplicativo que você achar necessário para cumprir os passos mencionados.

Primeiramente, o Termo de Consentimento Livre e Esclarecido (TCLE) contendo todas as condições da pesquisa foi apresentado aos usuários. Após o aceite do TCLE, a primeira parte do formulário, compreendendo as perguntas para identificação do perfil, foi respondida. Em seguida, os participantes testaram as funcionalidades da aplicação de acordo com o cenário de uso. O aplicativo foi utilizado através do Expo Go para que a experiência fosse a mesma independente da plataforma utilizada. Ao final, foi solicitado aos participantes que respondessem o questionário do SUS a partir da experiência obtida durante o uso da aplicação.

6.4.4 Resultados

Com o intuito de obter resultados mais variados, assim, enriquecendo a pesquisa, tentou-se selecionar pessoas com faixas etárias distintas. No entanto, dos seis participantes, 50% pertencem à faixa de 26-35 anos. Os outros 50% variaram entre 18-25 anos, 36-44 anos e 46-60 anos, com cada grupo representando 16,7% do total.

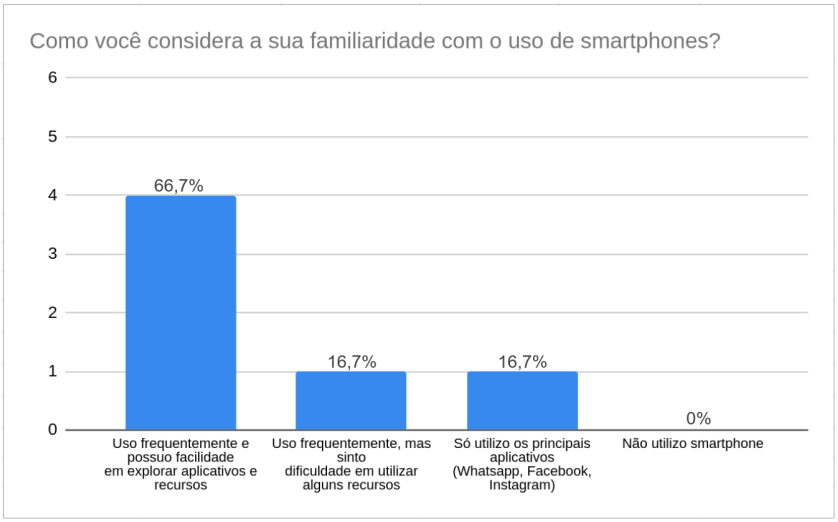
Figura 19 – Gráfico referente às respostas sobre faixa etária



Fonte: elaborado pelo autor

Todos os participantes afirmaram utilizar *smartphone* no seu dia a dia. A maioria, um total de 66,7%, disse usar *smartphone* frequentemente e ter facilidade em explorar aplicativos e recursos presentes no dispositivo. Outros 16,7% utilizam *smartphone* de forma frequente, mas sentem alguma dificuldade em explorar novos recursos. Os 16,7% restantes disseram utilizar basicamente aplicativos de mensagens e redes sociais, como Whatsapp e Facebook.

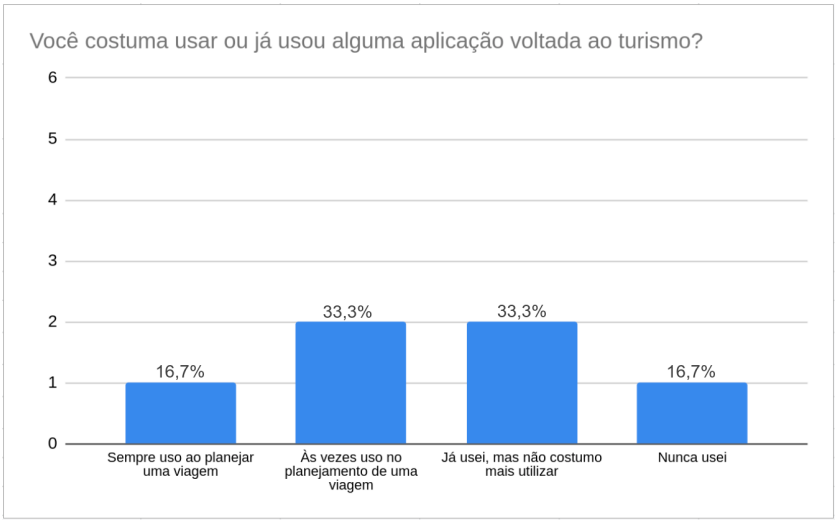
Figura 20 – Gráfico referente às respostas sobre uso de *smartphone*



Fonte: elaborado pelo autor

Em relação ao uso de aplicativos voltados ao turismo, 16,7% afirmaram sempre utilizar aplicações do tipo no planejamento de viagens. Já 33,3% disseram usar aplicativos turísticos algumas vezes quando viajam. Outros 33,3% já utilizaram, mas não utilizam mais por algum motivo não explicitado. Por fim, 16,7% nunca usaram uma aplicação do gênero.

Figura 21 – Gráfico referente às respostas sobre uso de aplicativos turísticos



Fonte: elaborado pelo autor

Depois de concluir os passos do cenário de uso citado anteriormente, os participantes responderam o questionário SUS. Um compilado de todas as respostas são apresentadas na Figura 22.

Figura 22 – Respostas dadas ao SUS

	Discordo totalmente	Discordo	Neutro	Concordo	Concordo totalmente
Acho que gostaria de utilizar este aplicativo com frequência	-	-	2 respostas	1 resposta	3 respostas
Considerei o aplicativo mais complexo do que o necessário	3 respostas	3 respostas	-	-	-
Achei o aplicativo fácil de utilizar	-	-	1 resposta	1 resposta	4 respostas
Acho que necessitaria de ajuda de um técnico para conseguir utilizar este aplicativo	3 respostas	2 respostas	1 resposta	-	-
Considerei que as várias funcionalidades deste aplicativo estavam bem integradas	-	-	1 resposta	1 resposta	4 respostas
Achei que este aplicativo tinha muitas inconsistências	3 respostas	2 respostas	1 resposta	-	-
Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este aplicativo	-	-	1 resposta	1 resposta	4 respostas
Considerei o aplicativo muito complicado de utilizar	4 respostas	2 respostas	-	-	-
Senti-me muito confiante a utilizar este aplicativo	-	-	1 resposta	3 respostas	2 respostas
Tive que aprender muito antes de conseguir lidar com este aplicativo	4 respostas	2 respostas	-	-	-

Fonte: elaborado pelo autor

O cálculo do SUS foi feito de acordo com as regras citadas na seção 6.4.1, chegando-se ao valor de 85,8. Esse resultado classifica a usabilidade da aplicação como excelente, de acordo com Bangor *et al.* (2009).

6.4.5 Considerações finais

Apesar dos recursos limitados e do curto prazo, a equipe Ethos conseguiu desenvolver uma aplicação com uma excelente usabilidade para o grupo de pessoas que participou da pesquisa. No entanto, entende-se que devido ao número limitado de participantes e da não cobertura da área do prestador de serviços, é essencial que sejam feitos estudos futuros com o intuito de complementar e atestar os resultados obtidos neste trabalho.

7 CONCLUSÃO

Este trabalho mostrou todo o processo de criação do aplicativo, desde a sua conceituação até o seu desenvolvimento, destacando as características *cross-platform* da biblioteca utilizada, o React Native. Os principais passos para se criar uma aplicação mobile React Native, através do Expo, foram apresentados ao longo deste relatório. Além disso, as etapas para se utilizar o Firebase, plataforma utilizada como *back-end* do Turistando Beberibe, também foram mostradas. No geral, observou-se que as ferramentas usadas no desenvolvimento do aplicativo facilitaram a sua criação, aumentando a produtividade da equipe.

Ao final, foi feita uma avaliação de usabilidade do aplicativo, através de uma pesquisa utilizando o método SUS, e outra levando em consideração aspectos técnicos. Esta última, sofreu restrições no momento da comparação dos arquivos binários devido à limitações técnicas na geração do *build* para a plataforma iOS, dessa forma, se faz necessário estudos futuros para uma melhor avaliação desse ponto. Quanto à avaliação de usabilidade, o aplicativo alcançou um resultado muito satisfatório, contando com uma boa aceitação por parte dos participantes.

Como trabalhos futuros, pretende-se: (1) finalizar e integrar as funcionalidades de chat e pagamento na aplicação; (2) fazer uma avaliação de usabilidade com prestadores de serviços; (3) realizar refatorações de código e melhorias de interface, juntamente com uma nova pesquisa para validar as alterações feitas; (4) gerar o *build* para ambas as plataforma e disponibilizar o aplicativo ao público.

REFERÊNCIAS

ADECE. **Turismo**. 2021. Disponível em: https://www.adece.ce.gov.br/setores-da-economia/turismo/. Acesso em: 07 jul. 2021.

AMBROSE, G.; HARRIS, P. **Design Th!nking**. Lausanne, Suíça: AVA Publishing SA, 2011.

ARAUJO, G. R. d. **Desenvolvimento cross-platform com react native: um estudo de caso do aplicativo naveg**, 2019. 69 f. Relatório Técnico (Graduação em Sistemas e Mídias Digitais) – Universidade Federal do Ceará, Fortaleza, 2019.

BALLERINI, R. **NPM vs Yarn**. 2021. Disponível em: https://www.alura.com.br/artigos/npm-vs-yarn. Acesso em: 09 jan. 2022.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Citeseer, v. 4, n. 3, p. 114–123, 2009.

BIØRN-HANSEN, A.; GRØNLI, T.-M.; GHINEA, G. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 5, p. 1–34, 2018.

BROOKE, J. *et al.* Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996.

CNDL. **Pesquisa Consumo online no Brasil**. 2021. Disponível em: https://materiais.cndl.org.br/pesquisa-consumo-online-no-brasil#rd-form-joq3m2m5. Acesso em: 22 fev. 2022.

DANIELSSON, W. React native application development. **Linköpings universitet, Swedia**, v. 10, n. 4, 2016.

ESLINT. **Getting Started with ESLint**. 2021. Disponível em: https://eslint.org/docs/user-guide/getting-started. Acesso em: 10 jan. 2022.

EXPO. **Introduction to Expo**. 2021. Disponível em: https://docs.expo.dev/. Acesso em: 28 dez. 2021.

EXPO. **Workflows**. 2021. Disponível em: https://docs.expo.dev/introduction/managed-vs-bare/. Acesso em: 09 jan. 2022.

FCDLCE. **Retomada do turismo deve ser impulsionada com grandes investimentos no Ceará**. 2021. Disponível em: https://fcdlce.org.br/retomada-do-turismo-deve-ser-impulsionada-com-grandes-investimentos-no-ceara/. Acesso em: 08 jul. 2021.

FIREBASE. **Cloud Firestore**. 2021. Disponível em: https://firebase.google.com/docs/firestore. Acesso em: 15 jan. 2022.

HEITKÖTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: SPRINGER. **International Conference on Web Information Systems and Technologies**. [*S. l.*], 2012. p. 120–138.

KRISTIANTO, D. **Winning the Attention War: Consumers in Nine Major Markets Now Spend More than Four Hours a Day in Apps**. 2021. Disponível em: https://www.appannie.com/en/insights/market-data/q1-2021-market-index. Acesso em: 08 jul. 2021.

KUMAR, A.; SINGH, R. K. Comparative analysis of angularjs and reactjs. **International Journal of Latest Trends in Engineering and Technology**, v. 7, n. 4, p. 225–227, 2016.

MALIK, K. Q. Appsheet vs react native: Evaluation of performance and development of android apps. 2021.

MARTINS, A. I.; ROSA, A. F.; QUEIRÓS, A.; SILVA, A.; ROCHA, N. P. European portuguese validation of the system usability scale (sus). **Procedia Computer Science**, Elsevier, v. 67, p. 293–300, 2015.

MOBX. **Defining data stores**. 2021. Disponível em: https://mobx.js.org/defining-data-stores.html. Acesso em: 12 jan. 2022.

NASCIMENTO, J. A. M. d. Usabilidade no contexto de gestores, desenvolvedores e usuários do website da biblioteca central da universidade de brasília. 2006.

PIRES, A. K.; NUNES, I. D. O uso de smartphones por idosos durante a pandemia do covid-19 no rn: um estudo exploratório. In: SBC. **Anais do XXVI Workshop de Informática na Escola**. [*S. l.*], 2020. p. 479–488.

REACTJS. **Introduzindo JSX**. 2021. Disponível em: https://pt-br.reactjs.org/docs/introducing-jsx.html. Acesso em: 27 dez. 2021.

REACTJS. **Setting up the development environment**. 2021. Disponível em: https://reactnative.dev/docs/environment-setup. Acesso em: 09 jan. 2022.

SANTANA, C. A.; ALCÂNTARA, R. A.; AVILA, B. T. Comparando métodos de avaliações de usabilidade, encontrabilidade e experiência do usuário. **ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO**, v. 17, 2016.

SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.

STATCOUNTER. **Mobile Operating System Market Share Worldwide**. 2022. Disponível em: https://gs.statcounter.com/os-market-share/mobile/worldwide. Acesso em: 02 fev. 2022.

STATISTA. **Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021**. 2022. Disponível em: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. Acesso em: 22 fev. 2022.

VIECELI, L. **Avanço da vacinação libera turismo e setor prevê retomada das viagens no fim do ano**. 2021. Disponível em: https://www1.folha.uol.com.br/mercado/2021/10/avanco-da-vacinacao-libera-turismo-e-setor-preve-retomada-das-viagens-no-fim-do-ano.shtml. Acesso em: 02 fev. 2022.

XANTHOPOULOS, S.; XINOGALOS, S. A comparative analysis of cross-platform development approaches for mobile applications. In: **Proceedings of the 6th Balkan Conference in Informatics**. [*S. l.*: *s. n.*], 2013. p. 213–220.

APÊNDICE A – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

Convidamos o(a) Sr.(a) para participar da presente avaliação de usabilidade sob responsabilidade do pesquisador Filipe Dourado Falcão.

A pesquisa tem por objetivo avaliar a usabilidade do aplicativo Turistando Beberibe, desenvolvido na disciplina de Projeto Integrado II do curso de Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará (UFC), sendo utilizado, atualmente, como produto do Trabalho de Conclusão de Curso (TCC) do pesquisador.

O Turistando Beberibe consiste em uma aplicação voltada à contratação de experiência turísticas, como passeios e trilhas, na cidade de Beberibe. Através do aplicativo é possível buscar por tais serviços e já fazer uma reserva e o seu respectivo pagamento. Também pode-se entrar em contato com os prestadores de serviços responsáveis pelas experiências para sanar eventuais dúvidas.

Primeiramente, enfatizamos que sua participação é voluntária, ou seja, a qualquer momento você pode recusar-se a participar da avaliação e até mesmo desistir de participar da pesquisa, sem gerar qualquer tipo de prejuízo. A sua participação consistirá em realizar algumas tarefas dentro do aplicativo e, ao assinar este termo, você automaticamente autoriza a disponibilização de suas respostas para compor a análise de dados.

Os resultados da pesquisa serão analisados e publicados, mas sua identidade será mantida em sigilo. Se depois de consentir em sua participação, o(a) Sr. (a) desistir de continuar participando, tem o direito e a liberdade de solicitar a exclusão de sua participação em qualquer fase da pesquisa. O(A) Sr. (a) não terá nenhuma despesa e também não receberá nenhuma remuneração.