FELIPE RAMOS PALMUTI

DESENVOLVIMENTO DE TELAS DE UM APLICATIVO MÓVEL PARA USUÁRIOS FINAIS DE UMA PLATAFORMA DE STREAMING/ESTUDOS

LAVRAS - MG
2021

FELIPE RAMOS PALMUTI

DESENVOLVIMENTO DE TELAS DE UM APLICATIVO MÓVEL PARA USUÁRIOS FINAIS DE UMA PLATAFORMA DE STREAMING/ESTUDOS

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. André Vital Saúde
Orientador

LAVRAS - MG
2021

RESUMO

Este trabalho é um relatório de estágio realizado em uma empresa cujo principal desafio é disponibilizar conteúdos de diferentes tipos e de temáticas diversas, a um nicho específico de pessoas. Dado esse problema, o objetivo do estágio foi desenvolver e refatorar telas e interfaces gráficas específicas para um aplicativo móvel, permitindo o consumo e a interação com os conteúdos por parte dos usuários. Para alcançar esse objetivo, foi utilizada a tecnologia React Native na implementação das telas, assim como outras bibliotecas responsáveis por criar efeitos visuais específicos, gerenciar o estado global da aplicação e estabelecer conexão com as outras *API's* mantidas pela mesma empresa. Como resultado, as telas almejadas foram concluídas e colocadas em produção como novas versões do aplicativo. O processo como um todo estimulou o aprendizado de novas tecnologias de desenvolvimento, a revisitação de conceitos estudados ao longo do curso de Ciência da Computação e o trabalho em equipe.

Palavras-chave: React Native. Aplicação móvel. Front-end.

ABSTRACT

This work is an internship report carried out in a company whose main challenge is to provide content of different types and with different themes, to a specific niche of people. Given this problem, the objective of the internship was to develop and refactor specific screens and graphical interfaces for a mobile application, allowing users to consume and interact with the content. To achieve this goal, React Native technology was used in the implementation of the screens, as well as other libraries responsible for creating specific visual effects, managing the global state of the application and establishing a connection with the other APIs maintained by the same company. As a result, the targeted screens have been completed and put into production as new versions of the application. The process as a whole stimulated the learning of new development technologies, the revisiting of concepts studied throughout the Computer Science course and teamwork.

Keywords: React Native. Mobile application. Front-end.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O estágio foi realizado em uma empresa chamada Zeester durante o período de um ano, onde o principal desafio enfrentado foi a necessidade de disponibilizar conteúdos de diversos tipos a usuários finais de um cliente. A organização e a plataforma são detalhadas no capítulo 2. Para solucionar o problema mencionado, foi desenvolvida uma plataforma de *streaming*/estudos denominada Blesss. Ela permite que os usuários não só consumam e interajam com seus conteúdos disponibilizados, mas que também possam conhecer as personalidades que protagonizam tais conteúdos.

Essa plataforma é formada por cinco sistemas que são gerenciados por subequipes diferentes dentro da empresa, porém que operam em conjunto para garantir seu funcionamento como um todo. São eles:

- O servidor - armazena o banco de dados e as *API's* que servem aos outros sistemas, além de hospedar todas aplicações web da empresa.
- A aplicação web principal - serve de interface para que o usuário final consuma e interaja com os conteúdos oferecidos pela plataforma.
- A aplicação web auxiliar - é de uso interno da empresa e garante funcionalidades, como por exemplo, o gerenciamento das assinaturas dos usuários e uma interface para que sejam postados os vídeos da plataforma.
- O aplicativo Android - consiste em uma versão mobile da aplicação web principal, desenvolvida para dispositivos Android e que pode ser baixada na Google Play.
- O aplicativo IOS - consiste no mesmo aplicativo desenvolvido para dispositivos Android, porém com adaptações exigidas pela plataforma e que pode ser baixado no serviço TestFlight da Apple.

Durante o período desse estágio, o objetivo foi desenvolver e refatorar telas e funcionalidades dos aplicativos Android e IOS, de maneira que ambos fossem mantidos atualizados em relação a aplicação web principal. Além disso, era objetivo testar e realizar o *deploy* dos resultados exclusivamente para a aplicação Android, uma vez que esses passos eram realizados por outro integrante da equipe, no caso da aplicação IOS.

Para tal, a tecnologia React Native foi utilizada, garantindo que com a escrita de um só código, as duas aplicações pudessem ser evoluídas simultaneamente. Além dessa tenologia, foram utilizadas bibliotecas que auxiliaram no processo de criação de animações visuais em algumas telas específicas, no gerenciamento e na persistência dos dados da aplicação e na realização de requisições para as *API's* da plataforma.

Como resultado, as telas foram concluídas com sucesso e lançadas nos aplicativos como novas versões. Todo o percurso do estágio exigiu o contante estudo e aprofundamento das tecnologias mencionadas, a fim de garantir a obtenção dos melhores resultados possíveis.

2 ORGANIZAÇÃO

A organização onde o estágio foi realizado é chamada Zeester e está localizada na cidade de Lavras, MG. Seu objetivo é desenvolver soluções tecnológicas para terceiros, com intuito de alavancar seus respectivos negócios. Atualmente, a principal plataforma mantida e desenvolvida pela empresa é denominada Blesss e tem como objetivo fornecer conteúdos como vídeos individuais, *playlists* de vídeos, lives, artigos e livros aos seus usuários finais.

Além de consumir e interagir com os conteúdos oferecidos pela plataforma, o usuário pode obter informações a respeito dos chamados preletores. Eles normalmente estão relacionados aos conteúdos, por ministrarem aqueles do tipo áudio-visual ou por serem os autores daqueles do tipo textual.

Atualmente a plataforma conta com mais de 100 séries de vídeo, estas que proporcionam mais de 1000 horas de estudos. Além disso, conta com a parceria de mais de 120 preletores, que contribuem na produção dos conteúdos.

Para ilustrar o alcance da plataforma, vale ressaltar que, durante o período do estágio, haviam aproximadamente 2700 downloads ativos do aplicativo versão Android.

2.1 Estrutura de equipes

Os funcionários que compõem a organização podem ser divididas em quatro sub equipes:

- A liderança - responsável por gerenciar a empresa e tomar decisões a respeito do futuro da plataforma. Com base nessas decisões, são criadas atividades com objetivo de expandir ou refatorar elementos específicos da plataforma.

- A equipe de design - responsável por analisar os objetivos propostos pela liderança e criar layouts de qualidade em termos de usabilidade, para o usuário final, respeitando as características do sistema para o qual irá se desenvolver (web ou mobile).

- A equipe de desenvolvedores - os desenvolvedores *back-end* são responsáveis por configurar o servidor, estruturar *API's* e modelar o banco de dados, de acordo com as atividades propostas. Já os desenvolvedores *front-end* são responsáveis pela programação das interfaces gráficas que serão utilizadas diretamente pelos usuários, com base nos layouts desenvolvidos pela equipe de design. Além disso, se encarregam de interligar as interfaces com as *API's* desenvolvidas pela equipe *back-end*.

- A equipe de marketing/comunicação - são responsáveis pela produção de todo conteúdo audiovisual, em colaboração com os chamados preletores. Realizam a divulgação da plataforma de maneira geral, através do gerenciamento das redes sociais e mantêm contato próximo com usuários finais, principalmente no recebimento de *feedbacks*.

Dado o contexto das equipes nas quais os funcionários se distribuem, é importante destacar que a função desempenhada durante todo o período do estágio foi a de desenvolvedor *front-end*.

3 REFERENCIAL

A plataforma desenvolvida utiliza do modelo arquitetural cliente servidor, que consiste na divisão do processamento da informação em duas diferentes aplicações. As aplicações clientes executam nos dispositivos do usuário final e são responsáveis por provê-lo interfaces usáveis para controlar os recursos disponíveis. Além disso, é função das aplicações clientes estabelecer comunicação com o servidor, através de requisições a suas *API's* (CANAL TI, 2018).

O servidor, por sua vez, é responsável pelo armazenamento do banco de dados, por executar processamentos de dados e por responder às requisições a ele feitas.

Dado o contexto arquitetural, este capítulo irá explorar as tecnologias que estão por trás do funcionamento do aplicativo móvel e de sua comunicação com as *API's* do servidor.

3.1 JavaScript

Consiste em uma linguagem de programação multi paradigma e de tipagem dinâmica fraca, que inicialmente se tornou conhecida por ser interpretada por navegadores. Isso significa que ao receber arquivos HTML com *scripts* dessa linguagem, o navegador é capaz de executá-los no lado do cliente (MOZILLA, 2021).

3.2 Node.js

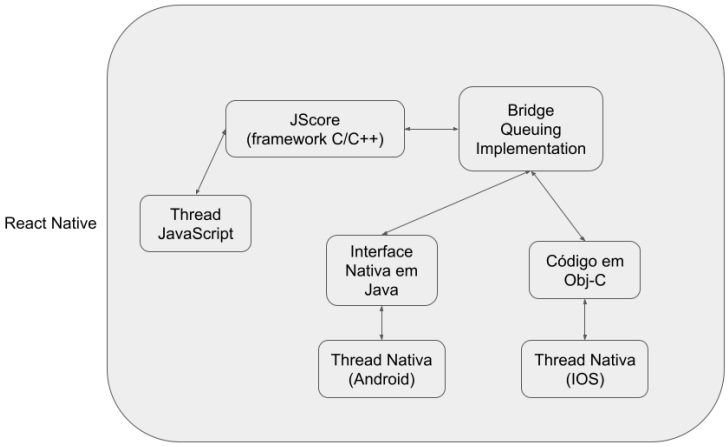
Esta tecnologia consiste no meio para se executar a linguagem JavaScript através do terminal, o que proporcionou sua utilização também nos servidores. A iniciativa de seu desenvolvimento surgiu como consequência da alta performance obtida pelo interpretador de JavaScript utilizado no Google Chrome, o V8 (OPUS SOFTAWARE, 2018).

3.3 React Native

Consiste em uma biblioteca da linguagem JavaScript que é utilizada no desenvolvimento de aplicações mobile destinadas às plataformas Android e IOS (REACT NATIVE, 2021). Uma aplicação React Native utiliza de um *framework* chamado JScore, que foi implementado em C/C++ e que é responsável por interpretar códigos JavaScript em um processo executado no dispositivo móvel (REACT NATIVE GUIDE, 2018). Essa interpretação resulta em mensagens no formato JSON, que são repassadas para a interface nativa existente, de acordo com o sistema operacional do dispositivo (Figura 1).

O repasse, por sua vez, é responsabilidade de uma implementação de fila que faz papel de ponte entre a *thread* JavaScript e a *thread* nativa. A comunicação entre ambas *threads* é do tipo bidirecional, garantindo que a *thread* nativa possa receber informações

Figura 1: Funcionamento do React Native em dispositivos móveis



a respeito do que deverá ser renderizado e comunicar a *thread* JavaScript a respeito de eventos disparados.

Na figura 2, é apresentado um código simples que configura uma tela através do React Native. Nela é possível localizar a seção de importações, a função "clickMe"que renderiza um alerta na tela, o método "render"que utiliza o padrão JSX para construir um componente visual e o objeto "styles"que garante o componente seja estilizado como um quadrado vermelho.

Esse exemplo basicamente renderiza um quadrado vermelho na tela, que ao ser tocado, utiliza a função "clickMe"como *callback* para exibir um alerta na tela com o texto "Hi!".

Figura 2: Código exemplo em React Native

```
import React from "react"
import { View, Text, StyleSheet, TouchableOpacity, Alert } from "react-native"

class HelloThere extends React.Component {
  clickMe = () => {
    Alert.alert("Hi!")
  }

  render() {
    return (
      <TouchableOpacity onPress={this.clickMe}>
        <View style={styles.box}>
          <Text>Hello! Please click me.</Text>
        </View>
      </TouchableOpacity>
    );
  }
}

var styles = StyleSheet.create({
  box: {
    borderColor: 'red',
    backgroundColor: '#fff',
    borderWidth: 1,
    padding: 10,
    width: 100,
    height: 100,
  },
});
```

3.4 Animated (React Native)

É uma *API* do React Native que não exige instalação de bibliotecas externas ao projeto. Fornece métodos para criar e manipular "valores animados", os quais podem ser vinculados a eventos diversos, como os disparados por botões ou por contêineres de rolagem. Além disso, permite a criação de animações complexas a partir da composição de outras mais simples, controlando atributos como cor, opacidade e tamanho (REACT NATIVE, 2021).

3.5 Redux

O Redux é uma das bibliotecas que podem ser instaladas em projetos React Native, porém merece destaque por ser responsável pelo gerenciamento do estado global da aplicação. Inspirado na arquitetura Flux, que consiste na implementação de *actions*, *reducers* e de um estado global, o Redux garante a centralização do armazenamento, acesso e atualização do estado da aplicação. Além disso, facilita o processo de armazenamento local dos dados no dispositivo físico, com a ajuda de bibliotecas auxiliares (REDUX, 2021).

3.6 Axios

Consiste em uma biblioteca externa que pode ser instalada em projetos React Native, a fim de permitir realização de requisições HTTP seguindo o padrão REST (NPM, 2021):

- *GET* - obtenção de dados.
- *POST* - envio de novos dados.
- *PUT/PATCH* - envio de alterações para um dado específico.
- *DELETE* - remoção de um dado específico.

3.7 Git e GitLab

Ferramentas utilizadas por equipes de desenvolvimento com propósito de gerenciar o versionamento do sistema. Permitem a contribuição simultânea e organizada de várias pessoas no desenvolvimento de um mesmo sistema, de maneira a construir um histórico de trabalho. Os contribuintes de um repositório realizam *commits* em suas respectivas *branchs* para salvar o progresso do trabalho e realizam *merges* entre essas e a *branch master* (a *branch* principal), quando terminam uma atividade específica (GIT, 2021). Dessa forma, é criado um fluxo único, organizado e transparente de trabalho, de maneira a facilitar o rastreamento de possíveis *bugs* e a restauração de um *commit* específico, caso necessário.

3.8 Scrum

Consiste em uma metodologia ágil utilizada na gestão e no planejamento de projetos em equipe (TREASY, 2016). Envolve conceitos como:

- *Product Owner* - pessoa que propõe as atividades a serem desenvolvidas para o sistema.
- *Backlog* - espaço onde são arquivadas as atividades que serão desenvolvidas futuramente.
- *Sprints* - períodos de trabalho. Ex: uma *sprint* pode ser definida como um período de uma semana, significando que as atividades propostas e a ela vinculadas, deverão ser concluídas nesse tempo.
- *Scrum Master* - o líder da equipe durante o período de uma *sprint*.
- Reuniões - estas que podem ser periódicas, acompanhando a frequência das *sprints*, ou mesmo diárias.

4 METODOLOGIA

O trabalho realizado consistiu no desenvolvimento de algumas telas e componentes visuais para o aplicativo móvel da plataforma Blesss. Para alcançar esse resultado foi utilizada a tecnologia React Native e suas bibliotecas, além das *IDE's* Visual Studio Code e Android Studio.

As seções de 4.1 a 4.3 apresentam a dinâmica da empresa e o ciclo de vida do processo de desenvolvimento de funcionalidades para a plataforma, inclusive para o aplicativo móvel.

4.1 Ciclo de vida das atividades

Em intervalos de aproximadamente um mês são realizadas reuniões entre a liderança e os líderes de cada subequipe, cujo objetivo é definir o futuro da plataforma a longo prazo e quais atividades devem ser executadas para que esse objetivo seja alcançado, assim como sua ordem de prioridade.

Simultaneamente, cada subequipe tem como rotina a execução de uma reunião semanal, em que cada líder é responsável por repassar as atividades previstas a sua respectiva subequipe. Essas reuniões também tem como objetivo definir as atividades a serem executadas por cada integrante a curto prazo, no caso, no período de uma semana (duração da *sprint*). Além da definição de novas atividades, é reservado um tempo para que cada integrante dê um *feedback* à equipe sobre as atividades da *sprint* anterior.

É importante ressaltar que os padrões de atuação adotados e que regem o ciclo de vida da realização das atividades pela equipe foram inspirados na metodologia ágil *scrum*.

4.2 Ciclo de vida das atividades no aplicativo

Para compor o conjunto de atividades que serão realizadas no aplicativo em uma *sprint*, sempre são selecionadas aquelas que não possuem outras não finalizadas como suas dependências. Essas dependências normalmente são as atividades da equipe de design e as dos desenvolvedores *back-end*.

Durante e após a implementação das funcionalidades relacionadas às atividades da *sprint*, elas são testadas manualmente. Uma vez concluído, o produto final é submetido à avaliação da liderança e são feitos ajustes de acordo com seu *feedback*.

Feitos os ajustes, a versão do aplicativo está pronta para o lançamento. Isso significa que a partir do projeto React Native é necessário gerar um *APK* (instalador do aplicativo para Android), este que deverá ser submetido no console da Google Play, de onde será distribuída a nova versão para todas as instalações do aplicativo.

Vale destacar que para gerenciar o versionamento dos código dos sistemas foi utilizado Git e para o armazenamento dos respectivos repositórios, GitLab. Para cada *feature*

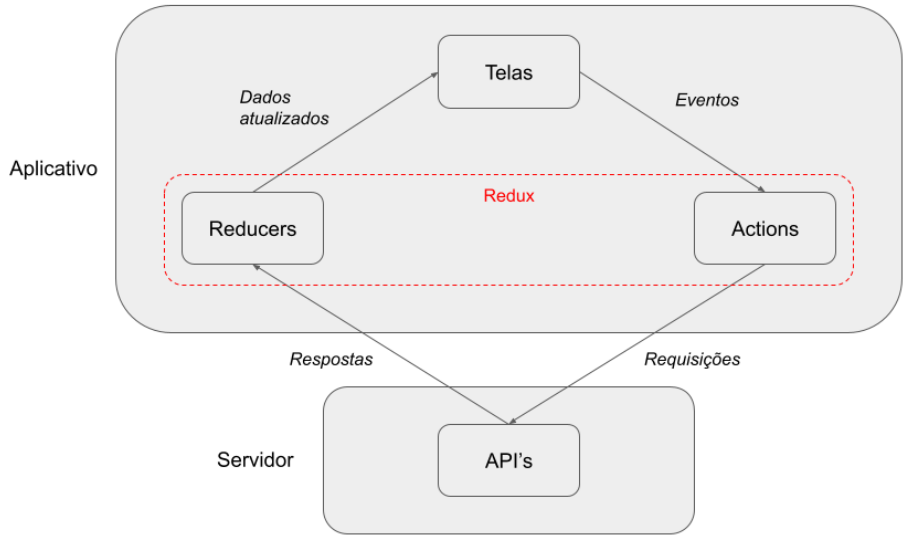
desenvolvida foram criadas *branchs* específicas, estas que foram unificadas à *branch* chamada "developer"ao fim de seus desenvolvimentos. Por fim, quando uma versão estável do sistema era alcançada na *branch* "developer", esta também passava por um processo de *merge* para a *branch* "master".

4.3 Uso das tecnologias no contexto do aplicativo

Como o aplicativo é desenvolvido utilizando a tecnologia React Native, as telas são estruturadas utilizando a notação JSX (JavaScript XML) e a folha de estilos própria da tecnologia. Além disso, toda lógica adicional intrínseca às telas é escrita utilizando a linguagem JavaScript pura.

Para realizar a integração do aplicativo com as *API's* do servidor é utilizada a biblioteca Axios, que permite a realização de requisições HTTP utilizando o padrão REST. A realização de cada requisição é vinculada a uma *action* diferente, seguindo a arquitetura do Redux. Isso significa que em qualquer ponto do código pode-se programar que algum evento dispare uma ou mais *actions* e que essas, por sua vez, realizem requisições ao servidor (Figura 3).

Figura 3: Arquitetura entre o aplicativo móvel e o servidor



Eventos como o acesso a telas específicas normalmente disparam *actions* relacionadas a requisições do tipo *GET*, para que sejam obtidas informações a respeito da tela que será renderizada. Por outro lado, requisições dos tipos *POST*, *PUT* e *DELETE* normalmente estão associadas a *actions* disparadas por eventos relacionados a interações específicas do usuário, como "curtir"um conteúdo ou excluir um comentário realizado.

Os dados obtidos como resposta às requisições feitas ao servidor são armazenados nos chamados *reducers*, com o propósito de tornar seu acesso global pela aplicação. Além

disso, essa prática torna mais organizado o processo de escrita dos dados no armazenamento local do dispositivo móvel.

5 RESULTADOS

Neste capítulo, serão apresentadas as telas e componentes visuais que foram desenvolvidos, respeitando o fluxo existente no próprio aplicativo. São as telas: a tela do preletor, a tela da série e a tela do artigo. E são os componentes: o *player* de vídeo e a seção de comentários.

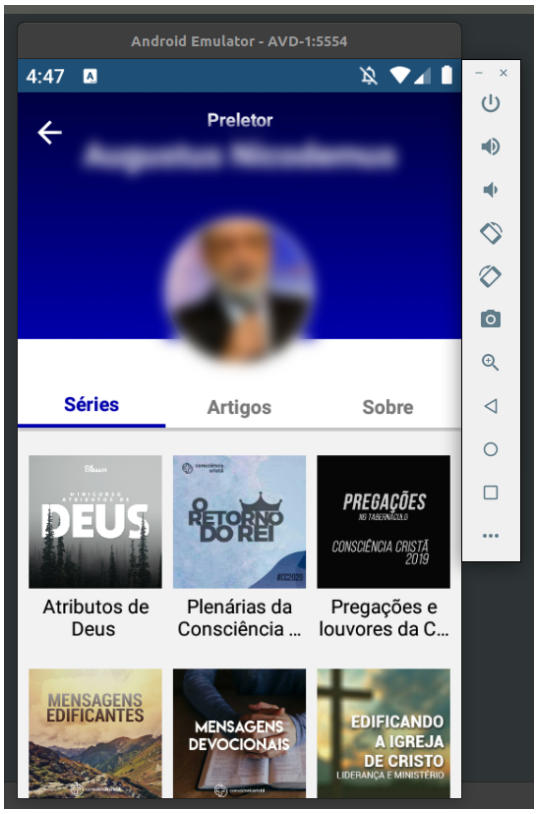
Nas seções seguintes serão apresentados *prints* para demonstrar os resultados obtidos pelas animações, que foram implementadas.

5.1 Tela do preletor

Os ditos preletores são personalidades com formações em diferentes áreas do conhecimento que geram conteúdos de diferentes tipos em parceria com a plataforma. Essa tela, portanto, tem como objetivo apresentar informações a respeito do preletor e de sua carreira, e principalmente exibir os conteúdos disponíveis que são de sua autoria ou que com ele se relacionam em algum nível.

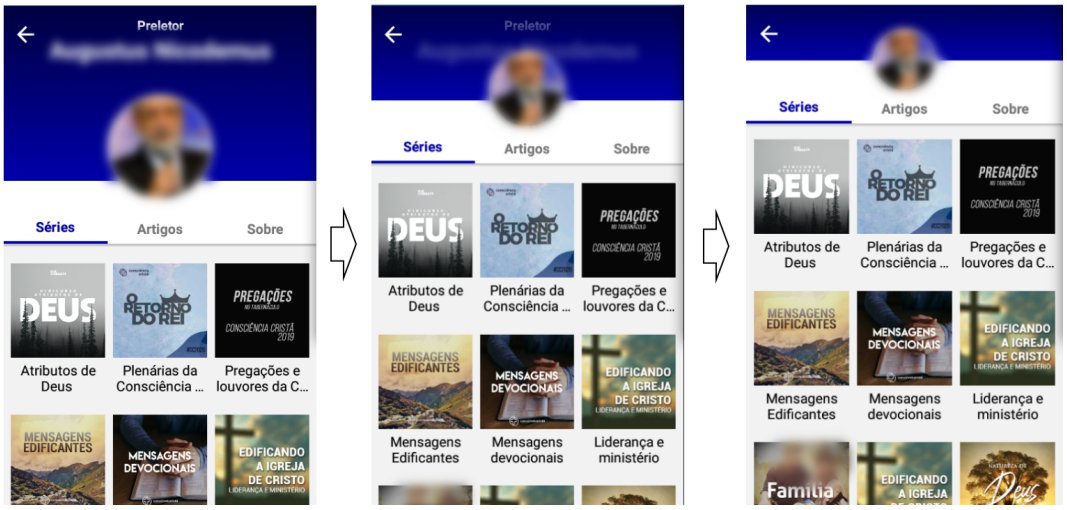
Nesta tela as abas são exibidas de acordo com os tipos de conteúdos existentes e relacionados ao preletor em questão. Ao tocar a representação de qualquer conteúdo, o usuário será redirecionado à página de visualização do mesmo, segundo o seu tipo (Figuras 4 e 5).

Figura 4: Tela do preletor



É importante ressaltar a animação customizada de encolhimento do cabeçalho, que funciona em sincronia com o evento de rolagem da tela. O seu desenvolvimento adicionou grande complexidade à tela, por exigir conhecimento a respeito da utilização de recursos da biblioteca Animated (Figura 5).

Figura 5: Animação da tela do preletor



5.2 Tela de série (playlist)

As séries são um conjunto de vídeos sequenciais que contemplam uma temática específica. A tela da série permite a visualização de suas informações primárias, como título, descrição, imagem ilustrativa e o progresso do usuário. Além disso, oferece ao usuário as ações de compartilhamento e de continuar assistindo-a do vídeo onde parou.

Na lista de vídeos, o usuário consegue visualizar o título, a *thumbnail* e o status (concluído ou não) de cada um. Ao tocar a visualização de um vídeo, o usuário é redirecionado para a página a ele relacionada, onde poderá assisti-lo e interagir com ele (Figuras 6 e 7).

Figura 6: Tela de série

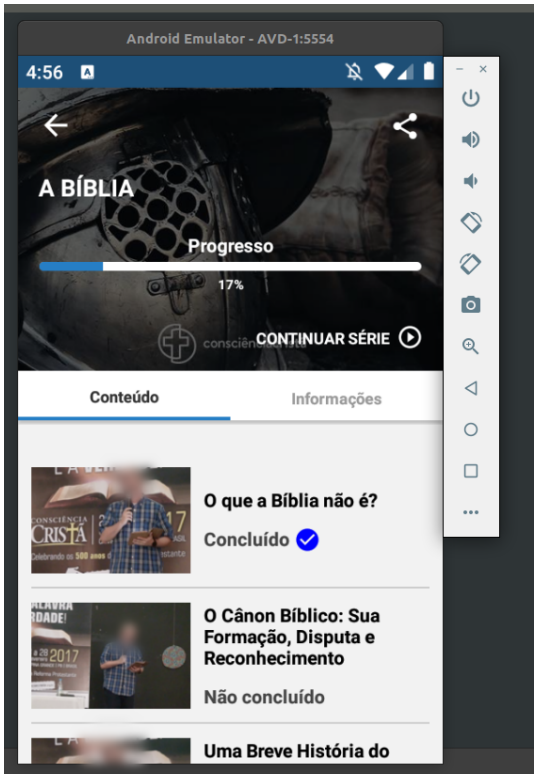
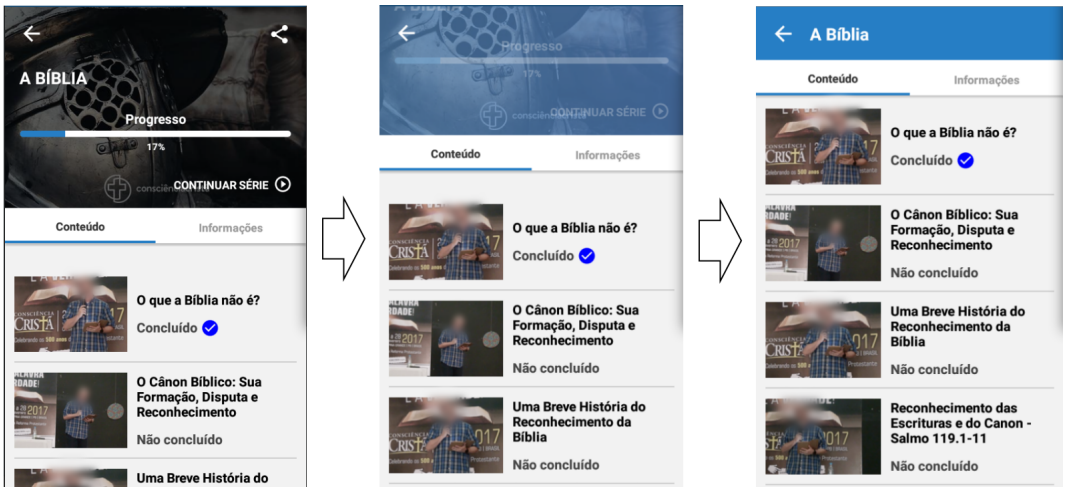


Figura 7: Animação da tela de série



5.3 Componente player de vídeo

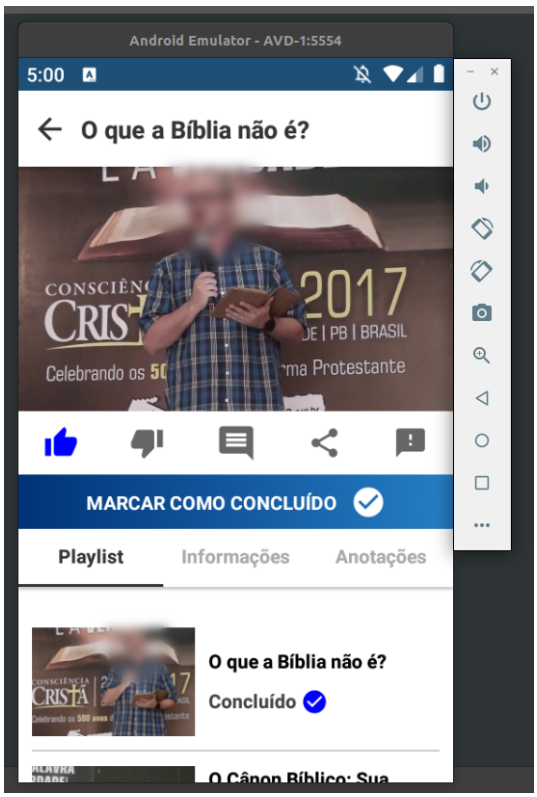
O componente *player* de vídeo é o utilizado em todas as telas do aplicativo que reproduzem vídeos comuns ou transmissões ao vivo (lives). Na sua construção foi necessária a adição da biblioteca react-native-video ao projeto, que é responsável por receber a URL do vídeo como parâmetro e renderizar a imagem e o áudio, de acordo com o tempo corrente.

Por outro lado, toda interface de interação com o vídeo foi construída a partir dos componentes padrões do React Native, sem a utilização de uma biblioteca externa.

5.3.1 Interface de interação

É composta por dois estados. No primeiro, a interface permanece transparente, permitindo a visualização completa da imagem do vídeo (Figura 8). No outro, a interface borra levemente a imagem e exibe todos os controles relacionados ao progresso do vídeo (Figura 9). Os estados podem ser alternados de acordo com o toque do usuário.

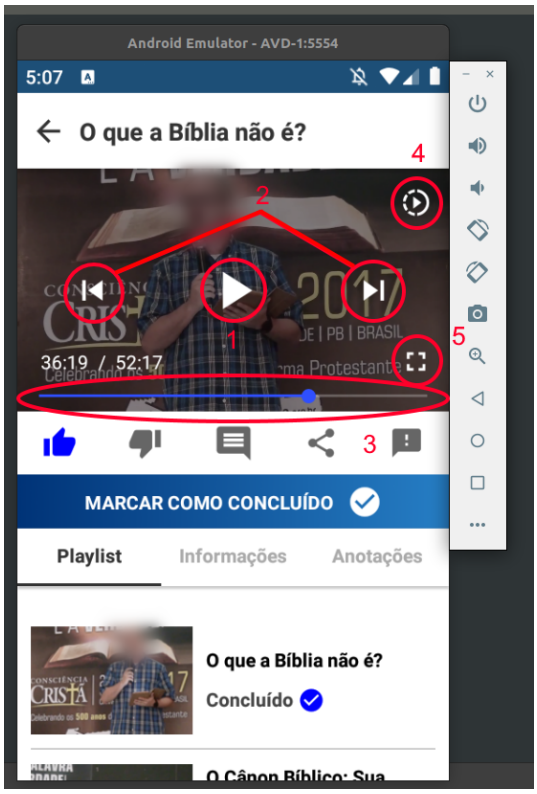
Figura 8: Player de vídeo com interface oculta



Os controles da interface de interação são (Figuras 9 a 11):

1. Reproduzir/pausar vídeo.
2. Retroceder ao vídeo anterior e avançar ao próximo vídeo: são exibidos somente quando o vídeo pertence a uma *playlist*.

Figura 9: Player de vídeo com interface visível



3. Barra de progresso controlável.

4. Controle de velocidade da reprodução: abre um modal com as opções de velocidade disponíveis (Figura 10).

5. Botão de tela cheia: permite que o usuário alterne entre os modos de tela comum ou tela cheia.

6. Botão de “Próxima página”: é exibido somente nos segundos finais do vídeo para permitir ao usuário um avanço ágil ao próximo vídeo (Figura 11).

Figura 10: Modal para seleção da velocidade de reprodução

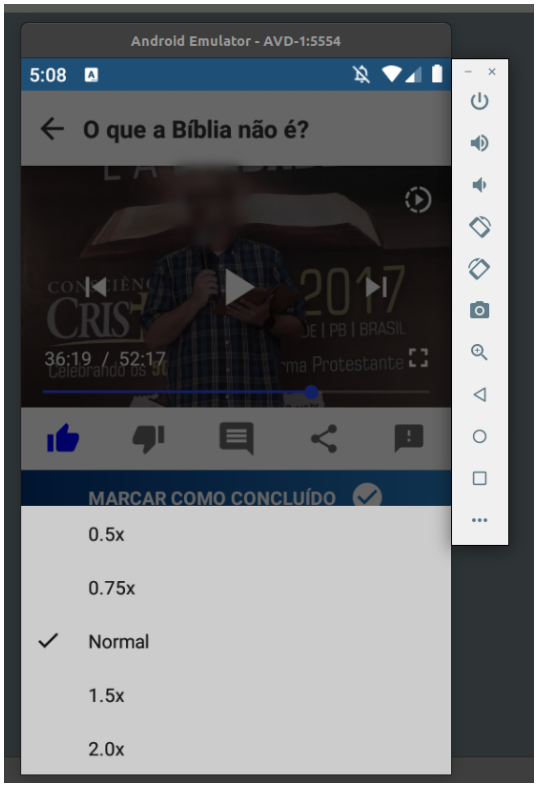
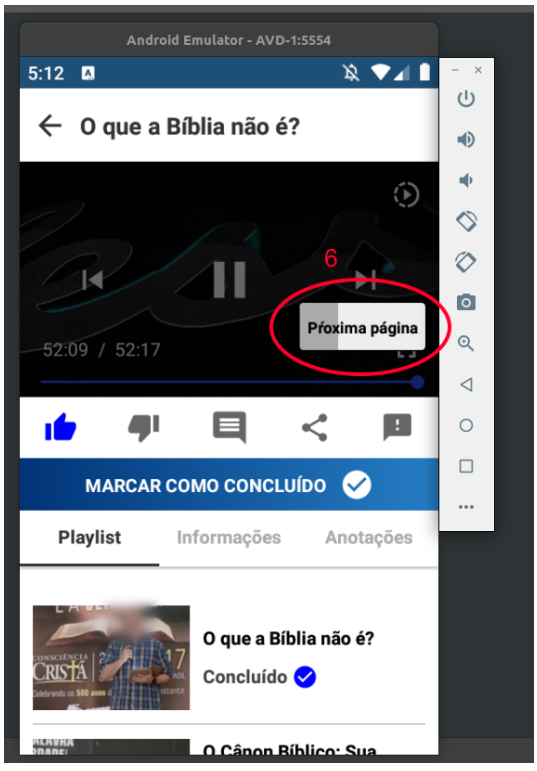


Figura 11: Botão de *próxima página*



5.3.2 Modo live

A resposta do servidor à requisição do vídeo informa não só a URL deste, mas também um booleano com propósito de informar se o vídeo é uma live. Para esses casos, foi implementado o modo live no *player*, em que as informações a respeito do tempo do vídeo são substituídas pela etiqueta “Ao vivo” junto a um círculo vermelho (Figura 12).

Caso o usuário esteja assistindo a live atrasada, é exibido um valor negativo indicando o quanto ele está atrasado em relação ao tempo corrente da live e um botão que o permite sincronizar-se com a transmissão (Figura 13).

Figura 12: Player de live no tempo corrente da transmissão

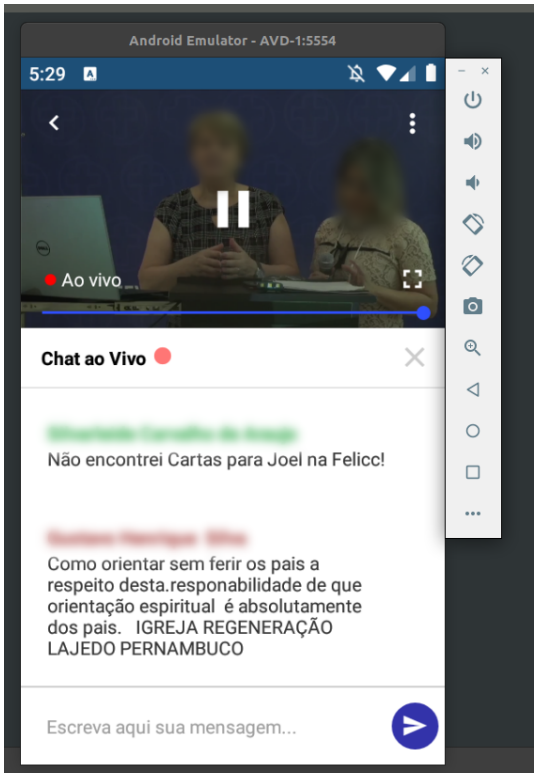
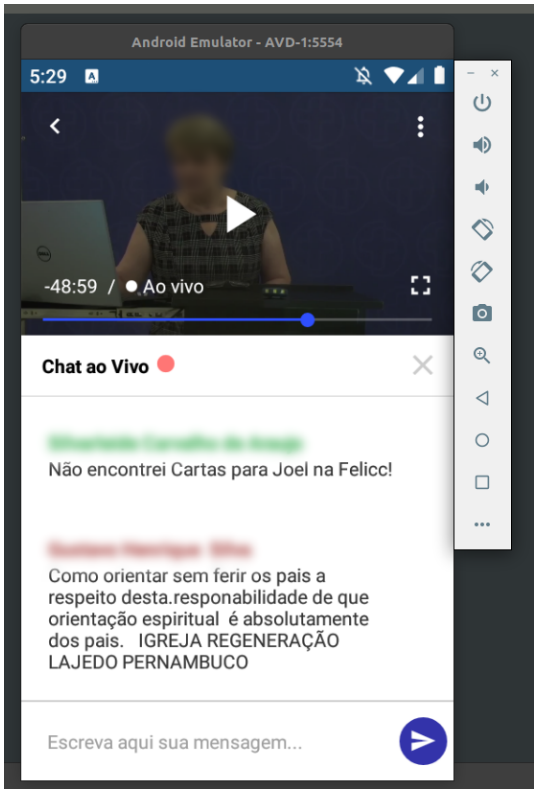


Figura 13: Player de live atrasado em relação ao tempo corrente da transmissão



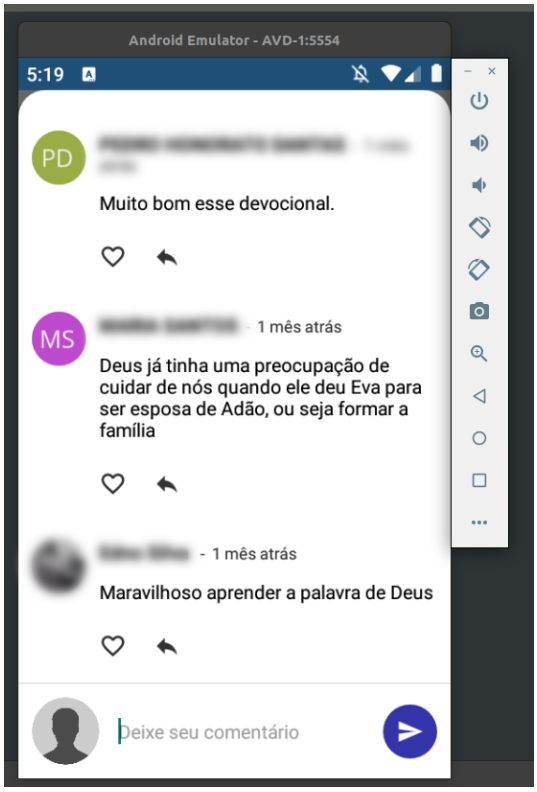
5.4 Seção de comentários

A seção de comentários representa o meio pelo qual os usuários conseguem interagir com os conteúdos e entre si, deixando seus pontos de vista a respeito dos assuntos tratados ou verbalizando uma avaliação.

O usuário pode escrever e enviar seu comentário de forma avulsa, pode responder a outro comentário existente ou pode marcar comentários que gostou com uma avaliação positiva (símbolo de coração).

As telas que permitem o usuário deixar um comentário possuem o ícone do tipo “balão de fala” (Figura 8). Quando tocado, um modal é aberto, contendo a seção de comentários (Figura 14).

Figura 14: Modal com a seção de comentários



5.5 Tela de artigo

Esta é a tela responsável pela exibição de artigos escritos por preletores e oferecidos à leitura por parte do usuário final. Além do conteúdo textual do artigo, são exibidos *cards* contendo links para telas de preletores e de outros conteúdos oferecidos pela plataforma que estão tematicamente relacionados.

É importante destacar que o cabeçalho dessa tela possui o comportamento de se esconder conforme a tela desce e de ser mostrado novamente quando a tela sobe, independentemente do ponto em que estiver. Além disso, em sua base, existe uma barra em vermelho que mostra o progresso da leitura (Figuras 15 a 17).

Por fim, na parte mais inferior da tela é exibida uma seção de avaliação e compartilhamento, onde o usuário pode interagir com o conteúdo (Figura 17).

Figura 15: Parte superior da página de artigo

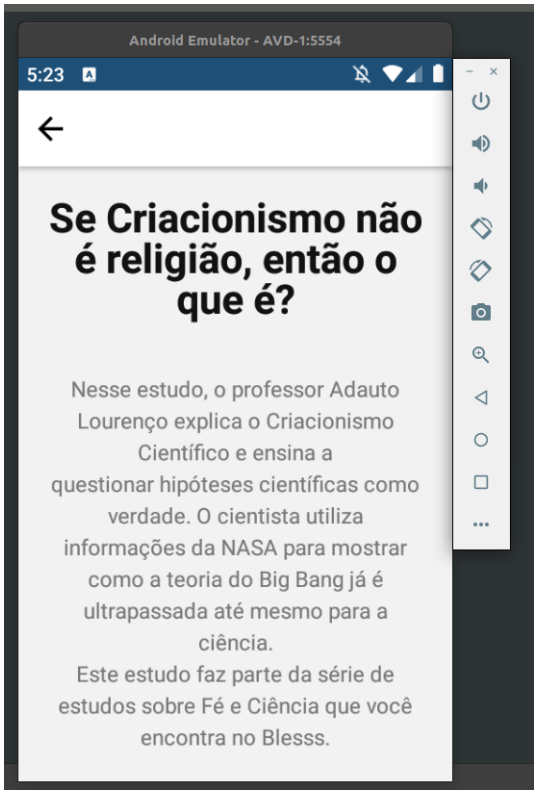


Figura 16: Parte intermediária da página de artigo

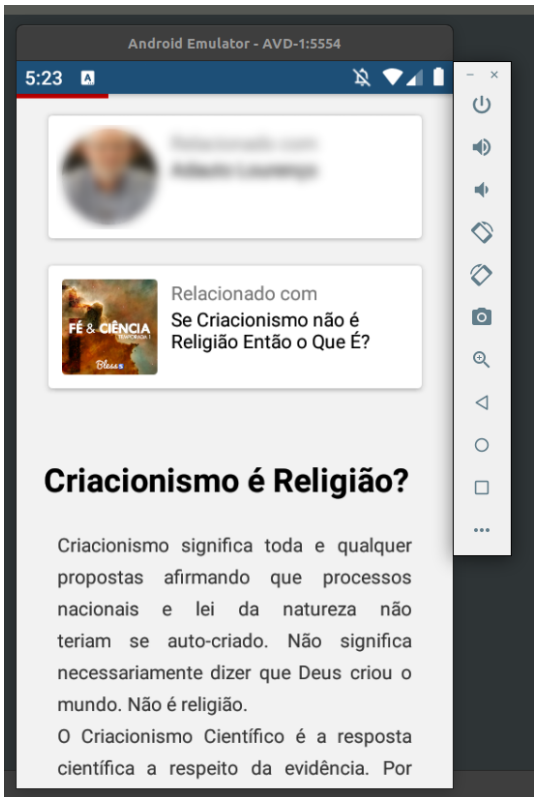
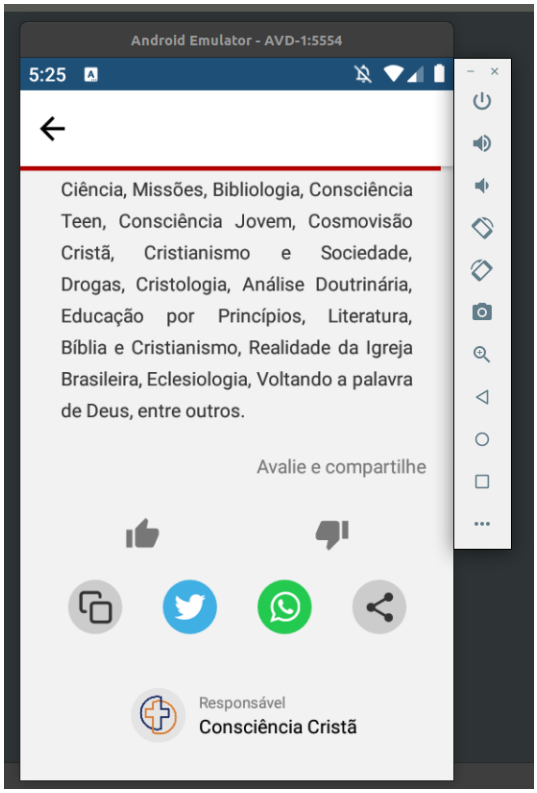


Figura 17: Parte inferior da página de artigo



6 CONCLUSÃO

As atividades realizadas no estágio envolveram os desenvolvimento de telas e funcionalidades para aplicações móveis, como solução para o desafio de disponibilizar conteúdos diversos aos usuários finais de uma plataforma. Todo o processo que envolveu a realização dessas atividades foi responsável pela criação de noções a respeito do ciclo de vida de desenvolvimento de uma aplicação. Essas noções englobam não somente o contexto técnico, que diz respeito às tecnologias que foram necessárias para a execução desse ciclo, mas também o contexto metodológico, que diz respeito à organização da equipe frente aos objetivos a serem alcançados.

É importante destacar que os resultados ressaltados no capítulo 5 foram obtidos exclusivamente pelo trabalho do autor deste relatório. Essa afirmação tem base no fato que, apesar das telas terem sido desenvolvidas inicialmente na plataforma Android, as adaptações exigidas para seu aproveitamento na plataforma IOS foram mínimas e realizadas antes do processo automático de geração dos códigos nativos.

Dentre os tópicos estudados nas disciplinas do curso de Ciência da Computação e que foram amplamente utilizados durante o estágio, se destacam:

- Algoritmos e estruturas de dados - na criação da lógica intrínseca às telas, para o gerenciamento da informação exibida.
- Conceitos de engenharia e arquitetura de software - em práticas organizadas de planejamento, implementação, versionamento e teste dos sistemas.
- Conceitos de sistemas distribuídos - no estabelecimento de comunicação entre os sistemas que compõe a plataforma.
- Programação WEB/Android - na utilização de tecnologias e *frameworks* apresentados nas disciplinas.

Dentre conhecimentos técnicos que foram desenvolvidos é importante ressaltar:

- Técnicas para implementação de layouts diversos (incluindo animações).
- Técnicas para armazenamento local de informações e gerenciamento de estado de uma aplicação.
- Comunicação com *API's*.
- Geração de novas versões e realização do *deploy* da aplicação.

Conclui-se que o estágio foi de fundamental importância para que assuntos estudados ao longo do curso fossem revisitados sob uma ótica mais prática e aprofundada. Também contribuiu para a expansão do leque de tecnologias conhecidas pelo estagiário.

7 REFERÊNCIAS

CANAL TI. **Arquitetura cliente-servidor**. 2018. Disponível em: <https://www .canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor> Acesso em: 1 de abr. de 2021.

GIT. **Branching and Merging**. 2021. Disponível em: <https://git-scm.com/about> Acesso em: 1 de abr. de 2021.

MOZILLA. **JavaScript Tutorials**, 2021. Disponível em: <https://developer.mozilla.org /pt-BR/docs/Web/JavaScript> Acesso em: 1 de abr. de 2021.

NPM. **Axios**. 2021. Disponível em: <https://www.npmjs.com/package/axios> Acesso em: 1 de abr. de 2021.

OPUS SOFTAWARE. **Node.js – O que é, como funciona e quais as vantagens**. 2018. Disponível em: <https://www.opus-software.com.br /node-js> Acesso em: 1 de abr. de 2021.

REACT NATIVE. **Animated**. 2021. Disponível em: <https://reactnative.dev/docs /animated> Acesso em: 1 de abr. de 2021.

REACT NATIVE. **React Native**. 2021. Disponível em: <https://reactnative.dev/docs /tutorial> Acesso em: 1 de abr. de 2021.

REACT NATIVE GUIDE. **React Native Internals**. 2018. Disponível em: <https://www .reactnative.guide/3-react-native-internals/3.1-react-native-internals.html> Acesso em: 1 de abr. de 2021.

REDUX. **Redux Essentials, Part 1: Redux Overview and Concepts**. 2021. Disponível em: <https://redux.js.org/tutorials/essentials/part-1-overview-concepts> Acesso em: 1 de abr. de 2021.

TREASY. **Tudo sobre Metodologia Scrum**. 2016. Disponível em: <https://www.treasy .com.br/blog/scrum> Acesso em: 1 de abr. de 2021.