Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Um aplicativo móvel multiplataforma para apoio ao registro de ocorrências atendidas por um grupo de bombeiros voluntários

Leonardo Barcelos Nardy Dias

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Euler Horta Marinho

Abril, 2021
João Monlevade–MG

Leonardo Barcelos Nardy Dias

Um aplicativo móvel multiplataforma para apoio ao registro de ocorrências atendidas por um grupo de bombeiros voluntários

Orientador: Euler Horta Marinho

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Abril de 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

D541a Dias, Leonardo Barcelos Nardy .
Um aplicativo móvel multiplataforma para apoio ao registro de ocorrências atendidas por um grupo de bombeiros voluntários. [manuscrito] / Leonardo Barcelos Nardy Dias. - 2021.
58 f.: il.: color., tab..

Orientador: Prof. Me. Euler Horta Marinho.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aplicativos móveis. 2. Bombeiros. 3. Sistemas de recuperação da informação - Web. 4. Voluntários no serviço social. I. Marinho, Euler Horta. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.41

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Leonardo Barcelos Nardy Dias

Um aplicativo móvel multiplataforma para apoio ao registro de ocorrências atendidas por um grupo de bombeiros voluntários

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 20 de abril de 2021

Membros da banca

Mestre - Euler Horta Marinho - Orientador (Universidade Federal de Ouro Preto)
Doutor - Diego Zuquim Guimarães Garcia - (Universidade Federal de Ouro Preto)
Mestra - Daniela Rodrigues Dias - (Doutoranda em Educação - Universidade Federal de Ouro Preto)

Euler Horta Marinho, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 08/05/2021

Documento assinado eletronicamente por **Euler Horta Marinho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 08/05/2021, às 22:33, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0168981** e o código CRC **AE1D9419**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado à minha família, em especial à minha mãe, por sempre me motivar e acreditar em mim, à minha namorada Mariana, por toda força e apoio durante os anos de graduação e a todos os meus amigos...

Agradecimentos

Agradeço primeiramente a Deus, por dar-me saúde e esperança para vencer meus obstáculos, à minha mãe, Maria, por todo exemplo de força, coragem, resiliência e batalhas que ela enfrentou para cuidar de mim e me mostrar como ser uma pessoa digna e nunca desistir dos meus sonhos. Ao meu pai Marino por seu apoio. À toda minha família pela demonstração de amor e por sempre acreditar em mim e me apoiar.

À minha namorada Mariana, por toda paciência e motivação durante o período de graduação, toda demonstração de amor e confiança foram fundamentais durante o caminho percorrido.

Aos meus eternos amigos, pelo companheirismo e todo apoio que me deram nos momentos mais difíceis.

À equipe da Informática Desenvolvimento da Bio Extratus, onde eu pude obter um grande desenvolvimento pessoal e profissional.

Ao Euler, pelos ensinamentos em sala de aula, por acreditar em mim e poder me auxiliar no desenvolvimento deste projeto.

Aos professores e professoras que tive durante a vivência na graduação, pelo ensino de qualidade que me proporcionaram.

"O correr da vida embrulha tudo. A vida é assim: esquenta e esfria, aperta e daí afrouxa, sossega e depois desinquieta. O que ela quer da gente é coragem..."

— Guimarães Rosa (1908 – 1967),

Resumo

Este trabalho tem como objetivo o desenvolvimento de um aplicativo para apoiar grupos de bombeiros voluntários que prestam serviços de resgate e utilizam um Sistema de Informação *Web* específico para apoiá-los. O aplicativo em questão foi desenvolvido buscando proporcionar uma utilização sem conexão disponível de forma confiável e que realize sincronia de dados em tempo real com o Sistema de Informação *Web* que atualmente é utilizado pelos bombeiros. O aplicativo desenvolvido segue uma abordagem multiplataforma, possibilitando o uso em diferentes tipos de dispositivos, independente das diversas plataformas atualmente no mercado e pode ser utilizado mesmo com conexões instáveis ou inexistentes durante o deslocamento do usuário. Durante o desenvolvimento do projeto, foram analisadas as ferramentas e técnicas disponíveis que apoiam a criação desse tipo de aplicativo. Por meio do aplicativo, é possível obter um ganho nas atividades relacionadas ao cadastro de boletins de ocorrências, otimizando assim o trabalho de registro de informações de tais grupos, que trazem um grande impacto social positivo para toda a população.

Palavras-chave: Aplicativo móvel. Multiplataforma. Bombeiros voluntários.

Abstract

This work proposes the development of an application to support groups of volunteer firefighters who provide rescue services and use a specific Web Information System to support them. The application was developed aiming to provide a connectionless use available reliably and that performs data synchronization in real time with the Web Information System that is currently used by the firefighters. The developed application follows a cross-platform approach, allowing the use in different types of devices regardless of the various platforms currently on the market and can be used even with unstable or non-existent connections during the user's movement. During the development of the project, the available tools and techniques that support the creation of this type of application were analyzed. The application allows obtaining a gain in activities related to the registration of accident reports, thus optimizing the work of recording information from such groups, which bring a great positive social impact for the entire population.

Key-words: Mobile application. Cross-platform. Volunteer firefighters.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PC Computador Pessoal

SI Sistemas de Informação

SisGera SisGera Sistema de Gerenciamento e Registro de Atividades

IDC *International Data Corporation*

CSS *Cascading Style Sheets*

PWA *Progressive Web App*

SQL *Structured Query Language*

API *Application Programming Interface*

iOS *iPhone Oerating System*

HTML *Hypertext Markup Language*

JSON *JavaScript Object Notation*

SDK Kit de Desenvolvimento de Software

MVC Model View Controller

Sumário

1 Introdução

Este trabalho tem como proposta o desenvolvimento de um suporte sem conexão e sincronia de dados de um Sistema de Informação que visa auxiliar na tarefa do registro de ocorrências que são atendidas por um grupo de bombeiros voluntários. O aplicativo desenvolvido é de tecnologia móvel multiplataforma, possibilitando o uso em diferentes tipos de dispositivos mesmo com conexões instáveis ou inexistentes durante o deslocamento do usuário. Durante o desenvolvimento do projeto, foram analisadas as ferramentas e técnicas disponíveis que apoiam a criação desse tipo de aplicativo, além da apresentação das etapas da criação do sistema, protótipos desenvolvidos e mais conceitos das áreas de Engenharia de Software, Sistemas *Web* e Móvel. Este trabalho representa uma extensão do projeto do (ARANTES, 2018) e (OLIVEIRA, 2018).

No trabalho de (ARANTES, 2018), foi realizado o desenvolvimento do projeto do SisGera Sistema de Gerenciamento e Registro de Atividades (SisGera), o qual consistia em um sistema para controle de ocorrências das corporações de bombeiros voluntários de São Domingos do Prata e Barão de Cocais. Porém, ao ser desenvolvido o sistema, novos requisitos e necessidades surgiram, motivando assim o desenvolvimento de uma nova versão do projeto SisGera, que foi desenvolvido por (OLIVEIRA, 2018), agregando a ele funcionalidades para gestão de atividades administrativas das corporações. Junto a todas as necessidades do sistema, uma que se destacava era a utilização do sistema em áreas sem conexão com uma rede e como os usuários poderiam interagir com o sistema de forma satisfatória independente de sua localização.

Foi criado um aplicativo que oferece mobilidade na sua utilização e é acessado por dispositivos móveis independente da variedade dos mesmos hoje disponíveis no mercado. O foco do aplicativo desenvolvido se baseia no momento de cadastro de ocorrências, visando um trabalho em locais que a conexão com a Internet seja instável ou mesmo inexistente. Atualmente a indisponibilidade de rede é um obstáculo a ser resolvido e a expectativa é que com o desenvolvimento de um aplicativo que ofereça suas funcionalidades sem conexão ajude os bombeiros voluntários e sirva de apoio ao SisGera. Como o desenvolvimento do aplicativo tem o foco direcionado ao trabalho conjunto com o SisGera, caso uma nova corporação passe a utilizar o Sistema *Web*, os bombeiros da corporação também poderão utilizar o aplicativo, o aplicativo não possui funcionalidades voltadas a cadastro de novos usuários ou permissões, portanto, o aplicativo criado é destinado para os bombeiros voluntários que utilizam o SisGera, independente de corporação.

1.1 Problema

A Associação de Bombeiros Voluntários de São Domingos do Prata, fundada em 28 de março de 2012, é uma instituição voluntária que presta serviços de resgate, salvamento e combate a incêndio, com plantão 24 horas por dia, todos os dias da semana, e está sediada na cidade de São Domingos do Prata/Minas Gerais. De maneira mais aprofundada, eles prestam atendimento a vítimas de trauma, busca e salvamento, ajudam em acidentes com compostos perigosos, e no combate e prevenção a incêndio. O atendimento prioriza as rodovias 381 e 262 na proximidades do município sede, Dionísio, São José do Goiabal, MGC120, LMG820, estradas vicinais de áreas rurais e outros locais nas imediações.

O Grupo de Resgate Voluntário de Emergência é uma entidade voluntária que também presta serviços de resgate para vítimas envolvidas em ocorrências de trauma e atendimento clínico, além de prevenção e combate a incêndios nos municípios de Barão de Cocais, Santa Bárbara, Catas Altas e o trecho da BR-381 até o trevo de Boa Vista.

Antigamente, as corporações realizavam todos os registros de forma manual, pois como prestam serviço voluntário, não contam com recursos para adesão de um sistema de informação comercial. Durante um atendimento prestado por grupos de bombeiros, muitas informações devem ser registradas de forma eficiente e confiável e, quando realizavam este processo de forma manual e escrita, os dados eram sujeitos a alguns problemas que poderiam comprometer a confiabilidade dos mesmos. Durante o desenvolvimento do projeto Sisgera no trabalho de (ARANTES, 2018), foram levantadas novas necessidades das corporações, sendo tais necessidades voltadas para as atividades administrativas, que resultaram no trabalho desenvolvido por (OLIVEIRA, 2018).

Com o desenvolvimento do Sistema de Informação para otimizar os processos realizados na instituição e auxiliar atendimentos que são prestados por equipes de bombeiros, as informações sobre as ocorrências devem ser armazenadas para consultas futuras e para as tomadas de decisões. Por isso, é de extrema importância o tratamento da disponibilidade sem conexão. As conexões com a Internet podem ser instáveis ou inexistentes dependendo de onde for o resgate do bombeiro voluntário, o que não pode o impedir de realizar o atendimento. Assim, o suporte ao modo sem conexão e o desempenho confiável são recursos que devem ser implementados visando uma sincronia correta ao banco de dados, que motivou o desenvolvimento deste trabalho.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um aplicativo móvel para apoio ao registro de ocorrências atendidas pelos bombeiros que utilizam o sistema *Web* SisGera.

1.2.2 Objetivos Específicos

Este trabalho tem como objetivos específicos:

- Revisar a literatura acerca de Sistemas de Informações Gerenciais, tecnologia móvel multiplataforma, Engenharia de *Software* e sistemas similares.
- Estudar e buscar tecnologias e ferramentas a serem utilizadas no trabalho relacionadas a *framework*, testes e disponibilidade sem conexão.
- Desenvolver protótipos para conhecer melhor as tecnologias analisadas e estudadas.
- Desenvolver aplicativo móvel que ofereça registros de boletins de ocorrência.
- Permitir que o aplicativo seja acessado por dispositivos independente da variedade dos mesmos.
- Disponibilizar uma sincronia sem conexão disponível para o aplicativo desenvolvido.
- Desenvolver testes para o aplicativo e também para serviços que o apoiam.

1.3 Organização do Trabalho

O restante deste trabalho é organizado como segue. O Capítulo 2 apresenta uma revisão da bibliografia, no qual são mencionados temas relacionados ao desenvolvimento deste trabalho. O Capítulo 3 descreve as escolhas tecnológicas e as abordagens utilizadas no processo de desenvolvimento do aplicativo móvel do SisGera, desde a criação de protótipos, desenvolvimento, até a validação das funcionalidades. O Capítulo 4 descreve e exibe as funcionalidades criadas e os resultados obtidos. O Capítulo 5 relata as considerações finais do projeto e propostas para eventuais trabalhos futuros.

2 Revisão bibliográfica

Este capítulo apresenta uma Revisão Bibliográfica sobre Sistemas de Informação, Aplicativos Móveis, Aplicativos Móveis multiplataforma, Sisgera e Teste de Software.

2.1 Sistemas de Informação (SI)

Segundo (LAUDON; LAUDON, 2011), um Sistema de Informação tecnicamente pode ser definido como um conjunto de elementos relacionados entre si que coletam ou recuperam, processam, armazenam e distribuem informações para o apoio na tomada de decisões, a coordenação e o controle de uma organização. Tais sistemas também auxiliam gerentes e trabalhadores na análise de problemas, visualização de assuntos complexos e na criação de novos produtos, sendo que contém informações sobre pessoas, locais e itens significativos para a organização ou o ambiente que a cerca.

Três atividades em um sistema de informação se destacam e geram as conclusões necessárias: entrada, processamento e saída. A entrada define a coleta de dados que podem ser obtidos do ambiente interno ou externo da organização; o processamento é a conversão desses dados obtidos para um contexto significativo, enquanto a saída é a transferência de informações que foram processadas aos membros da organização que as utilizarão de alguma forma.

Diversas organizações e grupos acadêmicos estão investindo tanto em novas tecnologias quanto em Sistemas de Informação para alcançar e alavancar bons resultados. De acordo com (LAUDON; LAUDON, 2011), buscando melhorar a eficiência e a qualidade das operações por meio de inovações e alternativas proporcionadas pelo uso de tecnologias e Sistemas de Informação, podemos alcançar uma excelência operacional.

2.2 Aplicativo Móvel

Segundo (B'FAR, 2004), Sistemas Computacionais Móveis são sistemas que podem ser movidos fisicamente com facilidade ou, cujas capacidades podem ser utilizadas enquanto eles estão sendo movidos. Como estes sistemas proveem tal mobilidade, eles normalmente oferecem recursos e características não encontradas em sistemas comuns, além de vantagens como:

- Possibilitar o acesso de dados em qualquer lugar e a qualquer momento;
- Armazenar dados local e/ou remoto, por meio de conexão com ou sem fio;

- Sincronização de dados com outros sistemas;
- Melhor experiência de usuário proporcionando-lhe um uso mais rápido e agradável, por possuir recursos gráficos e de interface;
- Diminuir custo de acesso (toda a parte da interface já se encontra instalada no celular), o que implica em um tráfego de dados também menor, para se acessar um determinado conteúdo da internet;
- Acessar recursos nativos do celular como: a câmera fotográfica, GPS, *bluetooth*, agendas telefônica, entre outros.

Muitos aplicativos estão hoje disponíveis em lojas e nos ajudam com diversas funcionalidades que são úteis na nossa vida, isto inclui, turismo, educação, saúde, entretenimento e clima. Os dois principais e maiores exemplos de lojas utilizadas atualmente são a Google Play Store [1] e App Store [2].

2.2.1 Dispositivos móveis e suas plataformas

Ao longo dos últimos anos, presenciamos um grande avanço na tecnologia móvel. De acordo com (MONTAN, 2017), os dispositivos móveis, em particular os *smartphones* e *tablets*, estão disponíveis em uma grande variedade de modelos e marcas, com processadores rápidos, amplo espaço de armazenamento, sensores e muitas funcionalidades. Como resultado, a utilização desses dispositivos e o mercado de aplicativos estão em crescimento e as lojas apresentam uma grande variedade de tipos de aplicações para os usuários.

De acordo com a última versão da pesquisa da *International Data Corporation* (IDC), publicada em 4 de Janeiro de 2021, os fornecedores mundiais de *smartphones* enviaram um total de 354.7 milhões de unidades apenas durante o terceiro trimestre de 2020. Embora grande parte dos holofotes sejam direcionados para a Apple e a Samsung que nos últimos anos costumam liderar o mercado, cada vez mais vemos diversidade a medida que mais fabricantes entram no mercado o tornando, cada vez mais competitivo. É o exemplo da Huawei e Xiaomi, que ficaram à frente da Apple na última pesquisa disponível até a publicação deste trabalho [3].

Apesar da diversidade citada acima sobre fabricantes de dispositivos móveis, quando buscamos analisar as plataformas atualmente utilizadas pelas marcas podemos observar que o Android é de longe a plataforma mais utilizada e comercializada. A Apple distribui seus dispositivos com *iPhone Oerating System* (iOS) e a maioria dos outros fabricantes utiliza o

[1] Disponível em: <https://play.google.com/store?hl=pt_BR&gl=US> Acessado em: 30/04/2021
[2] Disponível em: <https://www.apple.com/br/app-store/> Acessado em: 30/04/2021
[3] Disponível em: <https://www.idc.com/promo/smartphone-market-share/vendor> Acessado em: 30/04/2021

Android como plataforma base, uma pequena quantidade de dispositivos utilizaram outras plataformas como Windows Phone e BlackBerry OS nos últimos anos , porém, ao que tudo indica agora apenas o Android e o iOS estão sendo utilizados em novos lançamentos.

A Tabela 1 exibe diferenças no desenvolvimento das principais plataformas utilizadas Android e iOS, também é demonstrado o Windows Phone para evidenciar as diferenças de linguagem e ambiente de desenvolvimento.

Tabela 1 – Linguagens e suas Plataformas

Plataforma	**Linguagem**	**Ambiente**
Android	Java/Kotlin	Android Studio
iOS	Swift/Objective-C	Xcode
Windows Phone	C	Visual Studio

Fonte: Elaborada pelo autor(2019)

2.2.2 Desenvolvimento de Aplicativo Móvel

De acordo com (EL-KASSAS et al., 2015), o desenvolvimento de aplicações móveis é um caso especial de desenvolvimento de *software*, onde os desenvolvedores devem considerar aspectos diferentes como o ciclo de vida curto de desenvolvimento, a capacidade e recursos do dispositivo, mobilidade, especificações como tela, *design* e até mesmo a interação da interface do usuário do aplicativo.

Segundo (EL-KASSAS et al., 2015), o ciclo de desenvolvimento de um aplicativo móvel consiste em alguns passos, o primeiro deles é uma análise geral da ideia sobre o aplicativo, o segundo se trata de como é importante tratar o *design* de interface com o usuário, o terceiro aborda o desenvolvimento do aplicativo utilizando as ferramentas e linguagens de programação da plataforma escolhida. Um passo de grande importância é a realização de testes em dispositivos diferentes para observar a compatibilidade do aplicativo com os mesmos e finalmente a publicação do aplicativo na loja da plataforma escolhida. Depois de publicado, o aplicativo pode receber atualizações de correção ou novas funcionalidades diretamente da loja da plataforma escolhida pelo desenvolvedor. Para desenvolver o aplicativo visando múltiplas plataformas, o ciclo se repete em cada plataforma que seja escolhida previamente, porém, o primeiro passo é realizado somente uma vez.

Caso o desenvolvedor do aplicativo deseje construir o mesmo aplicativo para diferentes plataformas, alguns equipamentos serão necessários como: um Computador Pessoal (PC) ou um Mac para desenvolvimento para o ambientes iOS, ferramentas de desenvolvimento das plataformas escolhidas, pelo menos um dispositivo *smart* e um

tablet de cada plataforma para realização de testes. No caso do Android, é interessante a realização de testes em variados dispositivos, pois suas distribuições podem variar conforme o fabricante. De acordo com (WEI; LIU; CHEUNG, 2016) a plataforma Android tem um grande problema com a sua fragmentação entre diferentes dispositivos, devido à sua natureza aberta, um grande número de fabricantes por exemplo Samsung, Lg, Xiaomi e Huawey optam por desenvolver seus dispositivos móveis personalizando os sistemas Android originais e isso causa desafios sem precedentes para os desenvolvedores de aplicativos, pois existem mais de 10 versões principais do Android rodando em milhares de modelos de dispositivos distintos, e é impraticável para os desenvolvedores testar totalmente a compatibilidade de seus aplicativos em tais dispositivos.

2.2.3 Desenvolvimento Nativo

Os aplicativos nativos são desenvolvidos utilizando as ferramentas e uma linguagem de programação que são fornecidas pela própria plataforma móvel que foi escolhida. Como o aplicativo é feito exclusivamente para a plataforma, em geral, só é possível adquirir ou instalar o aplicativo no ambiente de loja da plataforma escolhida. Versões para Android por exemplo, são desenvolvidas utilizando as linguagens Java ou Kotlin, enquanto que versões para iOS são implementadas utilizando as linguagens Swift ou Objective-C.

Estamos acostumados a lembrar primeiro de uma abordagem nativa quando estamos pensando em desenvolver um aplicativo, pois, estamos acostumados de uma forma bem fácil, baixar e instalar aplicativos das respectivas lojas da plataforma. A Tabela 2 exibe algumas vantagens e desvantagens relacionadas ao desenvolvimento de um aplicativo nativo.

Tabela 2 – Desenvolvimento Nativo.

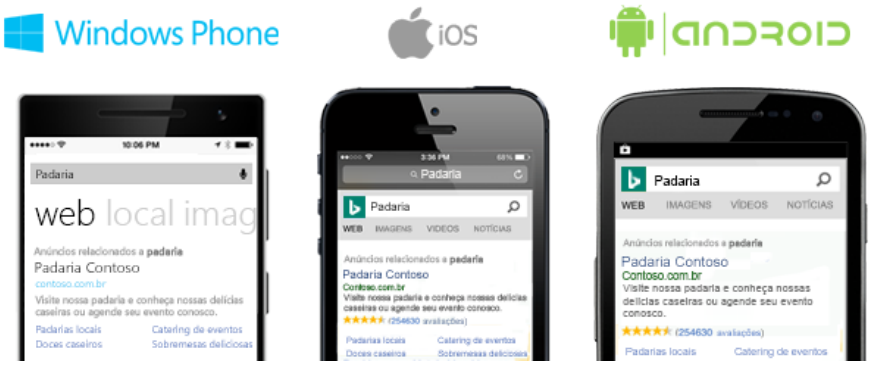
Vantagens	**Desvantagens**
Entrega alta performance	O desenvolvimento pode se tornar caro
Possui acesso total a todos os recursos do dispositivo	São mais complexas de se desenvolver
Pode ser publicada em lojas de aplicativos online	Necessário desenvolvimento independente de plataformas
Aparência nativa da interface com o usuário	Várias versões desenvolvidas para assegurar compatibilidade

Fonte: (EL-KASSAS et al., 2015)

2.3 Aplicativo Móvel Multiplataforma

O desenvolvimento de aplicativos móveis se tornou um desafio diante da diversidade de plataformas (Android, iOS, Windows Phone, outras) que utilizam linguagens de programação e ambientes de desenvolvimento diferentes. Do ponto de vista do desempenho, a melhor forma de desenvolver um aplicativo para um dispositivo móvel é a utilização direta dos recursos oferecidos pelos desenvolvedores do dispositivo (desenvolvimento nativo), contudo, criar uma versão para cada plataforma aumenta o custo de desenvolvimento e manutenção. Tal aumento de custo é mantido durante toda a vida do sistema, sempre que uma nova versão for disponibilizada. Neste contexto, surge a possibilidade de desenvolvimento híbrido que possibilita o reaproveitamento do código na geração de aplicações para diferentes plataformas móveis, com a consequente diminuição no custo de desenvolvimento e manutenção. A Figura 1 apresenta um navegador sendo utilizado realizando a mesma pesquisa em três plataformas diferentes.

Figura 1 – Aplicações em diferentes plataformas



Fonte: Elaborado pelo autor (2019)

2.3.1 Desenvolvimento de Aplicativo Móvel Multiplataforma

De acordo com (EL-KASSAS et al., 2015), o conceito de uma abordagem multiplataforma é desenvolver um aplicativo e executá-lo em qualquer sistema operacional móvel. Comparado ao desenvolvimento de aplicativos nativos, podemos dizer que a abordagem multiplataforma está chegando cada vez mais consolidada para nos ajudar a construir aplicativos de forma mais rápida e menos custosa, tanto em termos de valores monetários quanto em relação à curva de aprendizagem.

Segundo (MARINHO; RESENDE, 2015), os requisitos do aplicativo são o ponto principal em uma escolha de desenvolvimento de aplicativos móveis nativos ou multiplataforma. Algumas classes de aplicativo por exemplo jogos, têm demandas específicas

de experiência do usuário (por exemplo, aparência e comportamento) que podem ser dissociadas de considerações de plataforma específicas. Para a tomada de decisão, as organizações devem levar em consideração diversos fatores como as opções de mercado, as habilidades de desenvolvimento e as características do projeto. Na Figura 2 podemos observar várias características de cada tipo de desenvolvimento que aliada aos requisitos podem ajudar no processo de tomada de decisão.

Neste trabalho, foram estudadas as taxonomias sugeridas por (EL-KASSAS et al., 2015) e (BIORN-HANSEN; GRONLI; GHINEA, 2018). O foco relacionado ao desenvolvimento do aplicativo será na abordagem Híbrida e tipo de Aplicativo *Web* Progressivo. Porém, serão brevemente citadas as abordagens de desenvolvimento multiplataforma: Interpretado e Compilado, que também possuem atualmente *frameworks* populares que as seguem.

Figura 2 – Características

Característica	Nativa	Web	Multiplataforma
Acesso ao Dispositivo	Total	Limitado	Total (Com plugins)
Performance	Alta	Média a Alta	Média a Alta
Linguagem de Desenvolvimento	Plataforma Específica	HTML, CSS, Javascript	HTML, CSS, Javascript
Suporte Multiplataforma	Não	Sim	Sim
Experiência de Usuário	Alta	Média a Alta	Média a Alta
Reúso de Código	Não	Sim	Sim

Fonte: Adaptado de <https://ionicframework.com/> Acessado em: 30/04/2021

2.3.2 Aplicação *Web* Regular

De acordo com (EL-KASSAS et al., 2015), as aplicações *Web* para dispositivos móveis são desenvolvidas usando tecnologias *Web* disponíveis como *Hypertext Markup Language* (HTML) versão 5, Javascript e *Cascading Style Sheets* (CSS). Estas aplicações não são instaladas através da loja da plataforma que estiver em uso, porém, elas podem ser acessadas através de uma URL inserida em um navegador compatível.

A aplicação *Web* não é necessariamente uma aplicação móvel, ela pode ser vista como um *site* que é desenvolvido para atuar como um aplicativo e são indicadas caso o projeto do interessado não tenha funcionalidades complexas ou o mesmo tenha pouco orçamento para investir no desenvolvimento da aplicação. Um detalhe que atualmente pode pesar no desenvolvimento de uma aplicação *Web* regular para uso em dispositivos móveis é que a aplicação não terá um ícone visível na tela do celular do usuário, não passando confiança para o mesmo. Estas aplicações não são capazes de trabalhar sem conexão, dependendo completamente de estar conectada a rede para uso das funcionalidades desenvolvidas.

2.3.2.1 Aplicação *Web* Progressiva

Atualmente, quando uma abordagem *Web* é escolhida para um aplicativo móvel, possuímos a opção de escolher um desenvolvimento progressivo para seguir como conceito. Ela é um tipo de aplicação *Web* regular, porém, com consideráveis melhorias que trazem uma experiência de uso mais completa para o usuário.

Segundo (GRONER, 2018), uma *Progressive Web App* (PWA), ou Aplicação *Web* Progressiva pode ser considerada como um grande passo para evolução da *Web*. Uma aplicação *Web* progressiva é aquela que implementa um conjunto de técnicas para oferecer a melhor experiência na *Web* e, progressivamente, oferecer funcionalidades que antes só estavam disponíveis nativamente, seja no *desktop* ou em dispositivos móveis. Uma PWA pode ser vista também como uma evolução híbrida entre as páginas da *Web* regulares e um aplicativo móvel. Segundo informações da Google [4], uma PWA deve ter as seguintes características:

- Progressiva: para qualquer usuário, independentemente do navegador utilizado;
- Responsiva: ajustável em qualquer tamanho de tela (computador pessoal, móvel, *tablet*);
- Instalável: é possível adicionar um ícone na tela principal do *smartphone*, tablet ou como um aplicativo do navegador;
- Segura: uso somente com o protocolo HTTPS
- Independente de conectividade: funciona sem conexão de rede.

A seguir na Tabela 3 podemos ver algumas vantagens e desvantagens relacionadas ao desenvolvimento de uma PWA.

[4] Disponível em: <https://developers.google.com/web/updates/2015/12/getting-started-pwa> Acessado em: 30/04/2021

Tabela 3 – Aplicação *Web* Progressiva.

Vantagens	**Desvantagens**
Fácil de aprender e desenvolver usando as tecnologias disponíveis	Não possui acesso a todos recursos do dispositivo
Utilizando os navegadores *Web* pode ser executado em diferentes plataformas	Perde visibilidade pois não esta disponível nas lojas online
Desenvolvimento possui baixo custo	Baixa / Média performance pois são executadas nos navegadores

Fonte: (EL-KASSAS et al., 2015)

2.3.3 Híbrida

Um aplicativo híbrido, pode ser visto como o desenvolvimento que combina a abordagem nativa e a abordagem *Web* para buscar um meio termo entre elas. Como uma versão nativa, ele pode incorporar recursos do sistema operacional e, como uma aplicação *Web*, pode usar tecnologias como HTML5, CSS e Javascript por exemplo. Isso é possível, pois as ferramentas que implementam esse método criam um projeto nativo, instanciam um navegador por meio de um componente denominado *Webview* e executam o código do aplicativo no navegador.

Atualmente, a abordagem de aplicativo híbrido está sendo muito utilizada, mas linguagens de desenvolvimento estão aparecendo com novas ideias e ferramentas para incentivar a utilização da mesma. Estão sendo considerados alguns fatores no momento de desenvolver ou estudar um aplicativo móvel, como as aplicações híbridas podem ser desenvolvidas em um único repositório, a adição de novas funcionalidades a cada nova versão lançada é realizada de forma mais fácil e rápida. Desenvolvedores cada vez mais visam trabalhar com metodologias ágeis de desenvolvimento e, no momento de considerar opções, é importante considerar trabalhar com um desenvolvimento que pode transformar uma ideia em realidade de forma eficiente, rápida e econômica.

De acordo com uma pesquisa do *Stack Overflow* realizada em 2020 [5] com desenvolvedores profissionais, menos de 7% de todos desenvolvedores citaram Swift, Kotlin ou Objective-C como linguagens de desenvolvimento familiares. Em contrapartida, desenvolvedores *Web* compuseram quase 70% dos entrevistados. De acordo com esta pesquisa, podemos concluir que a comunidade de desenvolvedores *Web* é aproximadamente 10 vezes maior do que a de desenvolvedores móveis nativos, atualmente muitos programadores tem conhecimento utilizando HTML, Css e Javascript.

Como dito acima, os aplicativos híbridos tem vantagens interessantes. Por outro

[5] Disponível em: <https://insights.stackoverflow.com/survey/2020#most-popular-technologies> Acessado em: 30/04/2021

lado, como são um meio termo, a performance do aplicativo híbrido ainda está relacionada com a velocidade do navegador do usuário. Portanto, ele não consegue ser executado com o mesmo desempenho de um aplicativo nativo. As ferramentas mais populares que utilizam essa abordagem são PhoneGap e Ionic Framework.

A seguir na Tabela 4 podemos visualizar algumas vantagens e desvantagens relacionadas a abordagem Híbrida, em seguida são demonstradas duas abordagens de desenvolvimento com alocação de equipes e plataformas alvo, a Figura 3 mostra uma arbordagem tradicional de desenvolvimento e a Figura 4 mostra uma abordagem Híbrida de desenvolvimento utilizando o Ionic.

Tabela 4 – Aplicativo Híbrido.

Vantagens	**Desvantagens**
Aprendizado é mais fácil e o desenvolvimento é mais rápido	Tem menos performance do que um aplicativo nativo
Exige menos manutenção	A interface não será idêntica a um aplicativo nativo, sendo necessário esforço para adaptação
Pode ser publicada nas lojas de aplicações *online*	A segurança tende a ser mais difícil de controlar

Fonte: (EL-KASSAS et al., 2015)

Figura 3 – Abordagem Tradicional de Desenvolvimento

Fonte: Adaptado 30/04/2021 de <https://ionicframework.com/books/hybrid-vs-native>

Figura 4 – Abordagem Utilizando o Ionic de Desenvolvimento

Fonte: Adaptado 30/04/2021 de <https://ionicframework.com/books/hybrid-vs-native>

2.3.4 Interpretado

Em contraste com a abordagem híbrida, que reutiliza o mecanismo do navegador do dispositivo, o método interpretado permite o uso de linguagens genéricas como JavaScript por meio de interpretadores nos dispositivos. As ferramentas conseguem acessar as APIs nativas (e, consequentemente, as funcionalidades nativas, como notificações, câmera, entre outras) por meio de uma camada de abstração.

2.3.5 Compilado

Abordagens baseadas em compilação (chamados de compiladores cruzados ou trans) visam a reutilização de um aplicativo nativo mapeando o aplicativo de entrada para uma representação de destino. A principal diferença entre a abordagem compilada e as abordagens mencionadas acima é que o código produzido na linguagem de programação escolhida pelo *framework* é compilado para código binário executável nativo de cada plataforma alvo, sem necessidade de camadas intermediárias ou utilização do navegador como nos métodos anteriores. O acesso às funcionalidades nativas nesse caso é feito por meio do SDK do *framework*. Por ser compilado para código binário nativo, a maior vantagem desse método sobre os anteriores é a performance (LATIF; LAKHRISSI; ES-SBAI, 2016).

A seguir na Tabela 5 podemos visualizar as abordagens citadas acima e os *frameworks* associados que são mais utilizados atualmente por cada uma.

Tabela 5 – Abordagens de Desenvolvimento e *Frameworks* Associados Mais Utilizados.

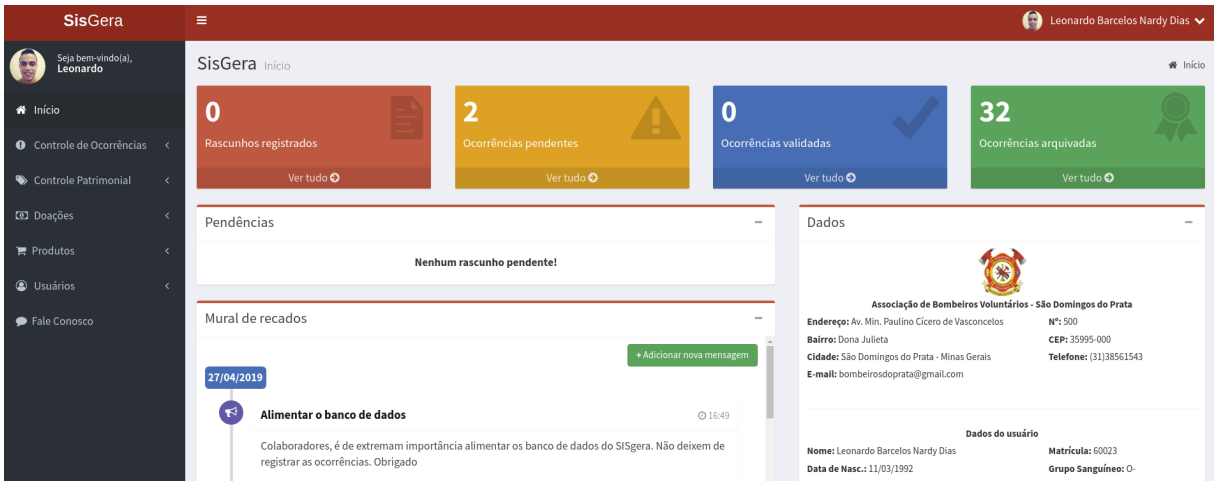
Híbrido	Interpretado	Compilado	*Web* Progressivo
Ionic Framework PhoneGap Cordova Capacitor	React Native NativeScript Titanium Appcelerator	Xamarin Flutter	Ionic Framework Vue.js Angular React.js

Fonte: Adaptado de (BIORN-HANSEN; GRONLI; GHINEA, 2018)

2.4 SisGera

Nesta seção, serão apresentadas algumas informações do Sistema SisGera, atualmente utilizado, e que motivou o desenvolvimento deste trabalho. Como citado na introdução, o SisGera é uma aplicação *Web* que foi desenvolvida como parte de dois trabalhos de conclusão de curso (ARANTES, 2018) e (OLIVEIRA, 2018). Este sistema visa atender as necessidades de um grupo de bombeiros voluntários para otimizar seus processos e gerenciar melhor sua corporação. A seguir na Figura 5 podemos ver a tela inicial do SisGera.

Figura 5 – Tela inicial do SisGera

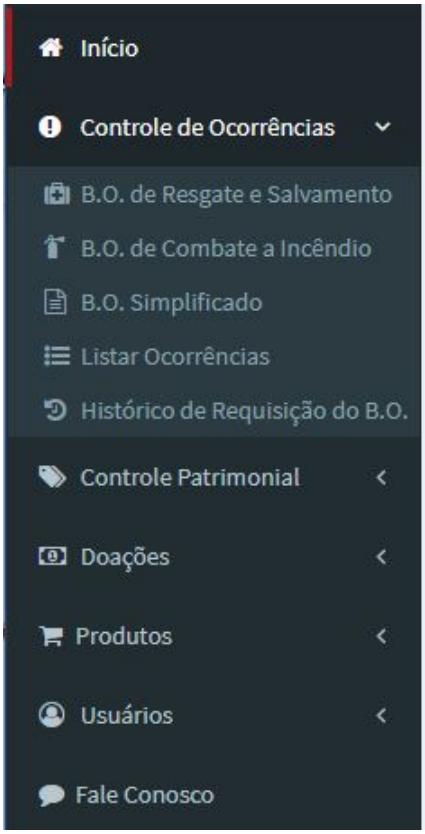


Fonte: <https://www.sisgera.com.br/> Acessado em: 30/04/2021

O foco do desenvolvimento do trabalho de (ARANTES, 2018) foi nos requisitos voltados ao registro do boletim de ocorrência, além da implementação inicial do projeto junto com suas dependências gerais. Nesse trabalho foram criados os ambientes para cadastro de boletins de ocorrência, que serão utilizados como base para o desenvolvimento do projeto atual, já que o foco do aplicativo atual é realizar o cadastro destes boletins e sincronizá-los com o servidor.

(OLIVEIRA, 2018) teve seu foco direcionado a gerência dos patrimônios da corporação. Nesse trabalho, foram implementadas novas funcionalidades que visam apoiar as atividades administrativas. Dessa forma, a corporação pode ter um melhor controle dos dados e informações para auxiliar na tomada de decisão voltadas a questões administrativas. Importante também destacar uma nova funcionalidade implementada na segunda parte do trabalho, que é a geração e validação do boletim de ocorrência para o usuário externo. Na Figura 6 podemos ver o menu geral do SisGera que contém os ambientes citados acima.

Figura 6 – Menu Geral e Ocorrências do SisGera



Fonte: <https://www.sisgera.com.br/> Acessado em: 30/04/2021

O Sistema SisGera vem sendo utilizado em sua versão *Web*, estando hospedado em um servidor comercial de forma gratuita, e pode ser acessado a partir do endereço <www.sisgera.com.br>.

2.5 Teste de *Software*

O enorme avanço tecnológico impactou nas mais variadas áreas, inclusive no desenvolvimento de aplicações móveis. Embora tenham surgido diversas tecnologias de modo a facilitar e otimizar o desenvolvimento, a realização de testes ainda não é algo simples de ser realizado. Em particular, no caso das aplicações móveis, testes podem ser ainda mais complexos, quando comparado ao caso de aplicações de outros domínios, devido às características das aplicações móveis.

De acordo com (ZHANG; ADIPAT, 2005), dispositivos móveis são utilizados para acessar serviços de informação por meio de uma estrutura descentralizada, que deve estar disponível independentemente da localização física do usuário e a qualquer momento. Novas funcionalidades e recursos tais como: câmera, televisão, áudio, voz, texto, sensores dentre outros, tem acarretado novas demandas por novas aplicações e com a grande variedade do mercado, cada dispositivo novo pode apresentar uma particularidade diferente.

Podemos exemplificar algumas limitações que existem em relação às características dos aplicativos móveis e que podem impactar as atividades e desenvolvimento de testes:

- Tela: existem restrições relacionados ao tamanho e à baixa resolução da tela. Um tamanho de tela muito pequeno, pode fazer com que páginas ou recursos não fiquem amigáveis aos olhos do usuário. Por mais que recursos responsivos estejam sendo utilizados por desenvolvedores, é sempre importante avaliar a usabilidade do aplicativo;

- Bateria: atualmente, mesmo com todo avanço tecnológico, a vida da bateria dos dispositivos móveis ainda é bastante limitada e seu consumo pode variar muito dependendo da tecnologia que foi escolhida para o desenvolvimento do aplicativo. Em certos dispositivos ou mesmo em dispositivos mais antigos, a bateria pode ser um obstáculo;

- Largura de Banda: a largura de banda está altamente relacionada com o desenvolvimento deste trabalho, que tem como um dos objetivos o funcionamento do aplicativo mesmo que a rede seja instável ou inexistente. Em muitos locais, a rede pode impedir um serviço ou funcionalidade de executar como esperado. A potência do sinal ou a velocidade da transferência de dados baixas é uma realidade ainda presente em vários locais;

- Capacidade de Armazenamento: a capacidade de armazenamento teve alguns obstáculos removidos com a chegada dos serviços em nuvem, que podem ajudar muito o usuário a guardar seus dados e não ocupar espaço local. A capacidade de armazenamento varia de dispositivo para dispositivo, e contando que nem todo usuário utiliza as vantagens da nuvem, caso um dispositivo não possua um espaço livre para

armazenar dados, é um fator limitante para inúmeras aplicações, até mesmo pra a sincronia de dados que está sendo desenvolvida no trabalho presente.

Um fato que é sempre importante lembrar, principalmente em aplicações móveis, é que todos os testes escritos não podem substituir testes reais em um dispositivo, ainda mais com a quantidade significativa de dispositivos atualmente no mercado, como citado acima. O aplicativo sempre parecerá e se comportará um pouco diferente no dispositivo, portanto, é importante nunca confiar completamente apenas na escrita de testes e verificar se um usuário ou mesmo um testador trabalhe com o aplicativo desenvolvido para garantir que o mesmo não execute funcionalidades quebradas.

2.5.1 Teste de unidade

O teste de unidade é um tipo de teste no qual são testados os componentes do programa como métodos, serviços e classes de objeto, de uma forma isolada do resto do sistema. O foco do teste de unidade é a verificação da menor unidade de projeto de um *software* (PRESSMAN, 2011). Ao se testar as classes de objeto, devem ser projetados testes que forneçam uma cobertura de todas as características do objeto. Como os testes de unidade exercitam o código isoladamente, eles são rápidos, robustos e permitem um alto grau de cobertura do código.

2.5.2 Teste de ponta a ponta

O teste de ponta a ponta é usado para verificar se um aplicativo funciona como um todo e geralmente inclui uma conexão com dados ativos. Enquanto os testes de unidade se concentram em unidades de código isoladas e, portanto, permitem o teste de baixo nível da lógica do aplicativo, os testes de ponta a ponta tentam descobrir problemas que ocorrem quando essas unidades individuais são usadas juntas [6].

Os testes de ponta a ponta existem em nosso próprio aplicativo no projeto, à parte do código do próprio aplicativo principal. A maioria dos testes ponta a ponta opera automatizando interações comuns do usuário com o aplicativo e examinando o documento e seus elementos para determinar os resultados dessas interações.

[6] Disponível em: <https://ionicframework.com/docs/angular/testing> Acessado em: 30/04/2021

3 Desenvolvimento

Este capítulo apresenta as abordagens e tecnologias utilizadas para a estruturação do projeto e desenvolvimento das funcionalidades, que foram pesquisadas durante o decorrer do trabalho e escolhidas mediantes experiências com as mesmas que são citadas a seguir:

3.1 Escolha tecnológica

Esta seção apresenta as tecnologias escolhidas para o desenvolvimento das funcionalidades do projeto.

3.1.1 *Ionic Framework*

O Ionic é um *framework* de código aberto que é voltado para o desenvolvimento de aplicações móveis multiplataforma. Quando foi lançado em 2012, o uso de tecnologias *Web* para desenvolvimento móvel estava dando seus primeiros passos. O Ionic é hoje o *framework* mais popular para desenvolvimento multi plataforma. Alguns pontos devem ser destacados: [1]

- Utilitário de linha de comando: facilita o início, a criação, a execução e a emulação de aplicações;
- Ionic Lab: podemos construir e testar as versões para iOS e Android lado a lado;
- Compilador ao vivo: atualiza instantaneamente aplicativos com alterações de código, mesmo quando executados diretamente em dispositivos.

3.1.1.1 Comparando com outros *frameworks*

React Native e Flutter são *frameworks* que permitem criar aplicativos móveis para iOS e Android sem a necessidade de aprender ObjectiveC ou Java. Estes foram citados na Tabela 5 do capítulo 2.

Flutter é um Kit de Desenvolvimento de Software (SDK) e um *framework* para o Dart, que é uma linguagem de programação desenvolvida pelo Google. A ideia por trás dele é que deve ser escrito um código que pode ser compilado para código nativo e que o mesmo também é executado no dispositivo. Já o React Native é um *framework* desenvolvido pelo Facebook. Ele usa Javascript e a biblioteca React para permitir a criação de boas interfaces

[1] Disponível em: <https://ionicframework.com/about> Acessado em: 30/04/2021

de usuário composta pelos seus componentes. Ele não usa marcações HTML mas, sim, um conjunto de componentes pré-criados que serão compilados para código nativo.

Um aplicativo desenvolvido no Ionic se comporta como um aplicativo nativo. Isso é feito criando uma aplicação Web utilizando HTML, Javascript, e CSS que será envolvida por uma camada nativa de um aplicativo hospedando uma *webview*. Como o desenvolvimento é bem semelhante a uma página *Web*, o Ionic é mais fácil aprendizagem para desenvolvedores *Web*, um dos motivos que reforçaram a escolha do autor. A partir do Ionic 4, ele é basicamente um conjunto enorme de componentes que podemos utilizar.

Cada abordagem tem seus benefícios e desvantagens, que geralmente se resumem a debates sobre desempenho, compartilhamento de código, customização, documentação e portabilidade.

3.1.2 *Apache Cordova*

O Apache Cordova é uma estrutura de desenvolvimento móvel de código aberto. Ele permite que os desenvolvedores usem tecnologias *Web* como HTML5, CSS3 e JavaScript para o desenvolvimento de plataformas. Os aplicativos são executados em invólucros direcionados a cada plataforma e contam com ligações de API compatíveis com os padrões para acessar os recursos de cada dispositivo, como sensores, dados e também status da rede [2], que será utilizado para desenvolvimento deste trabalho.

De uma forma mais simplificada, o Cordova envolve o código nativo (Swift, Objective-C, Java) em uma interface Javascript. Ele é quem faz essa ponte para o código funcionar de forma adequada. O Apache Cordova é recomendado quando:

- Um desenvolvedor de aplicativos móveis e deseja estender um aplicativo em mais de uma plataforma, sem ter que reimplementá-lo com o idioma e o conjunto de ferramentas de cada plataforma.
- Um desenvolvedor de aplicações *Web* que deseja implantar um aplicativo da *Web* empacotado para distribuição em vários portais de uma loja de aplicativos.
- Um desenvolvedor de aplicativos móveis interessado em misturar componentes de aplicativos nativos com um *WebView* (janela especial do navegador) que possa acessar APIs no nível do dispositivo ou se desejar desenvolver uma interface de *plugin* entre componentes nativos e *WebView*.

[2] Disponível em: <https://cordova.apache.org/docs/en/10.x/guide/overview/> Acessado em: 30/04/2021

3.1.3 *Capacitor*

O Capacitor é uma opção para o desenvolvimento hibrido, criada pelo mesmo time do Ionic. O Capacitor pode ser visualizado como um navegador para aplicativos da *Web* modernos que desbloqueia toda a funcionalidade nativa de cada plataforma por meio de APIs consistentes. Com o Capacitor, pode ser utilizado um conjunto de APIs independente de plataforma, em vez de gerenciar várias APIS para cada plataforma de destino.

Neste trabalho, optou-se pelo Capacitor para utilização de APIs modernas que não estavam disponíveis quando o Apache Cordova foi criado em 2009. Atualmente, mais desenvolvedores tem usado o Capacitor[3] . Quando um projeto Ionic é iniciado no terminal, para iniciar um novo aplicativo é sugerida a utilização do Capacitor.

3.1.4 *Angular*

HTML é adequada para declarar documentos estáticos, mas falha quando usada para declarar visualizações dinâmicas em aplicativos da *Web*. O Angular permite que se estenda o vocabulário HTML para o aplicativo. O ambiente resultante é extraordinariamente expressivo, legível e rápido de desenvolver. [4]

Angular é um cliente, o que significa que é executado no navegador e não no servidor. Ele utiliza o typescript que é um superconjunto do Javascript, para que os desenvolvedores tenham uma maneira mais conveniente de desenvolver aplicações Angular. No final, obteremos código Javascript mesmo que não esse código não tenha sido escrito diretamente.

Ele é um *framework*, de forma que existem um conjunto de regras, ferramentas e auxiliares de utilidade que podem ser utilizados em código Javascript. Por exemplo, enviar solicitações HTTP, roteamento, gerenciar estado, renderizar páginas e conteúdos diferentes, de acordo com a navegação do usuário no aplicativo. Tudo isso seria possível sem a utilização do Angular ou sem um *framework* parecido com o Angular. Mas utilizando um *framework*, o desenvolvimento é simplificado.

[3] Disponível em: <https://ionic.io/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development> Acessado em: 30/04/2021

[4] Disponível em: <https://angular.io/> Acessado em: 30/04/2021

3.1.5 Trabalho conjunto dos *frameworks*

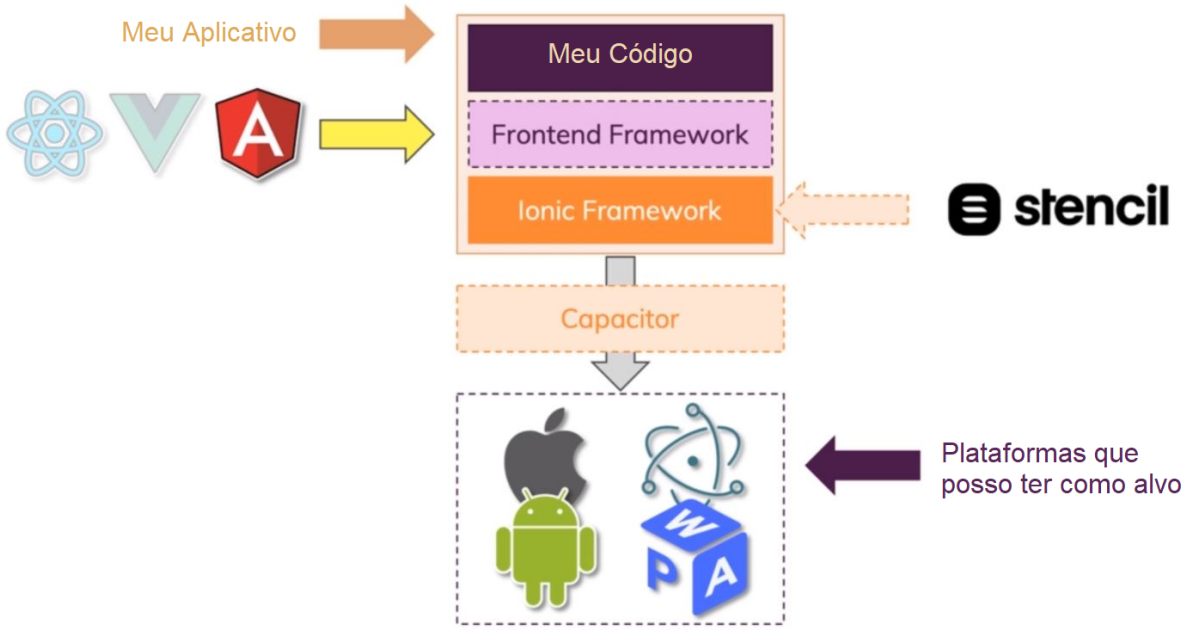
O Ionic e o Angular trabalham juntos a partir do pacote *@ionic/angular*. Ele é um pacote que se envolve em torno do conjunto de componentes do Ionic. Assim, ele integra todos os componentes Ionic e, também, todos os componentes da *Web* que podem ser utilizados, agrupando-os em um pacote ou módulo específico Angular. Isso é feito porque torna mais fácil o uso dos componentes e também é mais eficiente para utilizar os controladores em geral. Dessa forma, é fornecido uma ponte que facilita a utilização dos componentes.

O uso do Ionic e Angular pode ser visualizado como um projeto Angular, onde apenas é adicionado o Ionic no módulo da aplicação. O Ionic realiza uma configuração extra no arquivo `angular.json`, permitindo a utilização de duas estruturas ou bibliotecas.

Na Figura 7, podemos ver como se completam os *frameworks* que foram escolhidos como tecnologia para o desenvolvimento deste trabalho. Foi escolhido o *framework* front-end Angular. Entretanto, podemos ver que atualmente o Ionic trabalha com mais alternativas como React, Vue e até o Javascript puro. Em seguida, é utilizado o *framework* Ionic para que todo esse conjunto de componentes *Web* do Ionic obtenha elementos agradáveis e ricos adicionados ao aplicativo automaticamente ou com pouco esforço. Estes componentes *Web* do Ionic são desenvolvidos utilizando a ferramenta Stencil, porém não é preciso utilizar o Stencil para desenvolver com o Ionic. No final, diferentes tipos de aplicativos podem ser criados com base em uma única base de código e isso é feito utilizando Capacitor ou Cordova. Conforme mencionado anteriormente, foi selecionado o Capacitor, que atua como uma ponte entre o código da *Web*, o código Javascript e a plataforma nativa. Assim, a partir do código Javascript, podem ser acionadas certas funções que, no final, invocam o código nativo do dispositivo com o objetivo de, por exemplo, abrir a câmera, obter a localização do usuário ou exibir uma notificação.

Um outro ponto que podemos observar na Figura 7 é que, dentre as possíveis plataformas-alvos utilizando o Ionic, se encontram os aplicativos para computadores de mesa. Isso é possível com a utilização do *framework* Electron.

Figura 7 – Meu aplicativo



Fonte: Adaptado de: Maximilian Schwarzmüller, *Build Native iOS Android as well as Progressive Web Apps with Angular, Capacitor and the Ionic Framework*. Acessado em: 30/04/2021

3.2 Protótipos

Foram desenvolvidos dois protótipos durante a fase de planejamento do projeto, com o objetivo de adquirir mais conhecimento principalmente prático, mas também documentações existentes de ferramentas que seriam capazes de auxiliar o desenvolvimento da aplicação móvel do SisGera.

3.2.1 *PWA* Bloco de Notas

Foi desenvolvido um aplicativo que realizava apenas um cadastro de um formulário simples para realização de testes. O aplicativo se baseava no funcionamento de um bloco de notas e suas funcionalidades eram apenas incluir e excluir registros. Apesar da simplicidade, este aplicativo auxiliou na compreensão dos fatores relacionados à conectividade e responsividade. A Figura 8 exibe o simples aplicativo de teste.

3.2.1.1 Obstáculos e Conclusão

Já era de conhecimento que o aplicativo não possuía acesso aos recursos do dispositivo. Porém, o motivo que levou a ser descartada como tecnologia a ser adotada para o

Figura 8 – Protótipo - PWA

Fonte: Elaborado pelo autor (2019)

desenvolvimento do aplicativo Sisgera foi a sua limitação quando testado em dispositivos que utilizam o iOS.

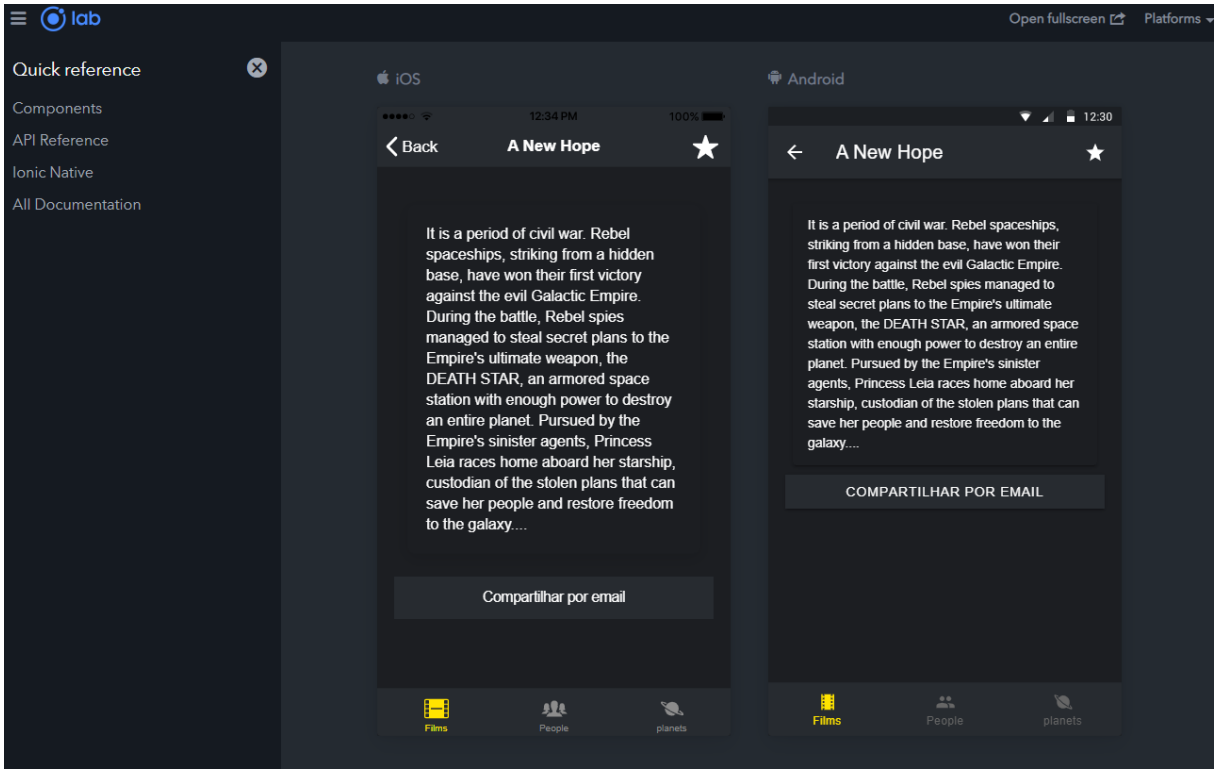
3.2.2 *Aplicação Ionic consumindo Api REST*

Outro protótipo foi desenvolvido utilizando o Ionic em sua versão 3. O foco do aplicativo era como o *framework* podia se integrar com uma API restfull e adquirir um conhecimento geral do *framework* e suas funcionalidades. Ao ter um primeiro contato com o Ionic, foi obtida uma boa experiência. Desde a primeira vez que o aplicativo é compilado, já nos chama a atenção de como sua interface é agradável e intuitiva para acompanhar o desenvolvimento do aplicativo. Outros aprendizados que foram adquiridos neste protótipo, foram a utilização do componente de envio de e-mail e também a utilização do localstorage. A Figura 9 exibe o aplicativo rodando no Ionic Lab sendo utilizada em sua versão para iOS e Android.

3.2.2.1 Obstáculos e Conclusão

O maior obstáculo encontrado foi o trabalho envolvendo Angular no qual o autor não possuía conhecimento. Como o Ionic possui uma boa documentação de sua estrutura, foi possível ir realizando pesquisas e implementações que trouxeram bons resultados e motivaram a escolha como *framework* a ser utilizado para o desenvolvimento do trabalho.

Figura 9 – Protótipo - Ionic consumindo Api



Fonte: Elaborado pelo autor (2019)

3.3 Processo de Desenvolvimento

Nesta seção, será apresentado o processo de desenvolvimento do aplicativo multiplataforma SisGera, desde a elaboração das histórias de usuário e modelagem do banco de dados, até as questões relacionadas a sincronia, conexão e testes.

3.3.1 Histórias de usuário

A especificação dos requisitos das novas funcionalidades do projeto SisGera voltadas ao aplicativo foi realizada utilizando a abordagem de histórias de usuário. Para a especificação das histórias de usuário, foi utilizada a sintaxe sugerida por (COHN, 2004) na qual uma história de usuário deve ter:

- **Como um...** (Quem) (Papel, ator)
- **eu quero...** (O que) (Funcionalidade a ser desenvolvida)
- **de modo que...** (Por que) (Benefício a ser obtido)

Abaixo são apresentadas as histórias de usuário das novas funcionalidades que foram incluídas no aplicativo do SisGera:

Sincronizar Boletim.
Como um usuário interno,
eu quero sincronizar um boletim,
de modo que seja possível validá-lo e seguir seu fluxo normalmente no SisGera *Web*.

Verificar Conexão.
Como um usuário interno,
eu quero observar a conexão de meu dispositivo em tempo real na tela de sincronia,
de modo que possa me informar e realizar uma decisão de sincronia confiável.

Verificar Boletins Sincronizados.
Como um usuário interno,
eu quero visualizar os boletins que foram sincronizados a partir de meu dispositivo,
de modo que possa manter um histórico temporário que me auxilie no controle de minhas ações.

3.3.2 Sincronização Sem Conexão

Para que uma aplicação execute todas as suas funcionalidades propostas de uma forma coesa, podemos considerar que a aplicação necessite utilizar informações que estão atualizadas de acordo com os acessos anteriores de um usuário na aplicação, independente de local ou dispositivo usado. Um local comum onde os dados podem ser centralizados e armazenados será necessário.

Um desafio importante no desenvolvimento deste trabalho é relacionado ao tratamento da aplicação, levando em conta possíveis estados de disponibilidade da rede. Para uma aplicação que se encontra sem disponibilidade de conexão, algumas abordagens podem ser identificadas:

- Interrupção completa da aplicação.
- Utilização parcial da aplicação, de forma que algumas funcionalidades podem ou não estar disponíveis, dependendo de seu contexto.
- Utilização completa da aplicação, de maneira que todas as funcionalidades funcionam normalmente, mesmo sem rede disponível.

Analisando o contexto acima, o objetivo do trabalho é seguir a abordagem semelhante a utilização completa das funcionalidades. A aplicação quando sem conexão

disponível continuará sendo usada normalmente. Porém, assim que estiver com acesso a rede novamente, deve se comunicar com um serviço no qual sincroniza todas as informações que se encontram pendentes e que ainda não foram enviadas para o servidor, seja pela falta de conexão com a rede ou instabilidade com a mesma.

3.3.3 API

Para que se possa realizar comunicações com um servidor foi necessário o desenvolvimento de uma *Application Programming Interface* (API). Acessar o banco de dados diretamente do aplicativo Ionic seria realmente inseguro, pois seria necessário inserir as credenciais do banco de dados no código do aplicativo. Assim, pessoas sem autorização poderiam ter acesso as credenciais e definitivamente isso não é o ideal. Portanto, o aplicativo vai se conectar em uma API exposta em um servidor.

A API aceita solicitações HTTP, que são manipuladas de forma a executar algum código do servidor. Por sua vez, o servidor interage com o banco de dados. Dessa forma, a interação se realiza somente com a API para o envio de solicitações e recebimento de respostas. Essas solicitações e respostas são trocadas no formato *JavaScript Object Notation* (JSON), que permite simplificar o tratamento das mesmas.

A API pode ser desenvolvida de forma a personalizar o próprio *back-end* ou utilizando um serviço que já é gerenciado, por exemplo, o Firebase. como o SisGera já trabalha com um banco de dados relacional *Structured Query Language* (SQL) e o Firebase utiliza um banco de dados NoSQL, que não é relacional, foi decidido desenvolvimento de uma API personalizada.

Atualmente, existem vários *frameworks* que possibilitariam a criação de uma API. No entanto, devido ao projeto *Web* do SisGera ter sido desenvolvido pelo *framework* Laravel[5] e pelo mesmo disponibilizar boas funcionalidades que auxiliam na criação de uma API, este foi escolhido como o *framework* ideal para esta necessidade.

O Laravel é um *framework* de código aberto para desenvolvimento *Web* o qual utiliza o padrão arquitetural Model View Controller (MVC). Para implementação de uma API, o Laravel nos permite trabalhar de forma otimizada com autenticação, roteamento, e diversas ferramentas que auxiliam a criar uma robusta aplicação. A Figura 10 apresenta o sistema de rotas que o Laravel nos permite utilizar. Com ele, podem ser filtradas as requisições a partir de definições e regras especificadas. Podemos ver que também é verificado se o usuário está autenticado na API.

A Figura 11 exibe a utilização de uma *resource* do Laravel, ela funciona como uma camada de tratamento e conversão para as respostas JSON que são expostas pela API. As *resources* permitem que transformemos facilmente, modelos e coleções em JSON.

[5] Disponível em: <https://laravel.com/> Acessado em: 30/04/2021

Figura 10 – Sistema de Rotas

```
Route::middleware(['auth:api','user'])->group(function () {
    Route::get('/users', 'Api\UsuariosController@index');
    Route::get('/users/{usuario}', 'Api\UsuariosController@show');
    Route::post('/boletimS', 'Api\BOSimplificadoController@store');
});
```

Fonte: Elaborado pelo Autor (2020)

Figura 11 – Resource

```
Route::prefix('Api')->group(function () {
    Route::resources([
        'users' => 'UsuariosController',
        'boletimS' => 'BOSimplificadoController',
        'boletimR' => 'BOResgatecontrollerController',
        'boletimI' => 'BOIncendioController',
    ]);
});
```

Fonte: Elaborado pelo Autor (2020)

3.3.4 Armazenamento

Serão destacados dois métodos que foram utilizados para auxiliar no armazenamento dos dados no dispositivo e auxiliaram o desenvolvimento do trabalho.

Local Storage: Pode ser utilizado o método de armazenamento de dados interno/externo, quando é preciso armazenar dados no sistema de arquivos do telefone que não exija recursos de banco de dados relacional. O armazenamento de dados interno/externo garante o armazenamento imediato dos dados e é simples de usar. Os dados armazenados, usando o método de armazenamento interno, são exclusivos da aplicação. Assim, se o aplicativo for desinstalado, os dados serão removidos do dispositivo.

SQLite database: Tanto o Android quando o iOS suportam bancos de dados SQLite. o SQLite funciona bem em telefones e fornece aos aplicativos a velocidade e a potência de um banco de dados relacional com todos os recursos. O SQLite usa um único arquivo para armazenar dados. Este armazenamento por si só não pode resolver o lado da sincronização e

da resolução de conflitos. No entanto, é uma opção fácil de usar para enfileirar e armazenar os dados em cache.

Neste trabalho, foi integrado o plugin SQLite do Capacitor que nos permite construir um aplicativo que em certos casos primeiro são carregados alguns dados iniciais de formato JSON e, em seguida, nos permite trabalhar com esses dados dentro do aplicativo. O recurso mais interessante deste plugin é que ele nos permite importar dados em JSON e com isso podemos realizar um bom trabalho com a API desenvolvida, utilizamos este recurso para nos auxiliar no processo de login sem conexão disponível dos usuários pois assim conseguimos manter os registros e permissões dos mesmos atualizados.

A título de exemplo, a Figura 12 mostra um pouco da lógica para armazenar o nome do banco de dados dentro do armazenamento do Capacitor e, também, manter o controle se já foram sincronizados os dados iniciais. Portanto ou é iniciado o download e a importação ou, se já o foi feito antes, abrir o banco de dados diretamente. A Figura 13 exibe um pouco da lógica da importação. O arquivo JSON é baixado desde o início. É verificado se é um arquivo válido para a importação e, então, é importado todos os dados do banco de dados SQLite. Como em certos momentos devem ser sincronizados novos dados, pode ser criada uma tabela de sincronização com o *plugin*. A tabela terá um valor *timestamp* para auxiliar a sincronização atual. Dessa forma, é possível observar as alterações que aconteceram desde a última sincronização. Depois de sincronizar os dados novamente, o aplicativo pode ser utilizado normalmente, mesmo sem conexão disponível.

Figura 12 – Preparando Base

```
private async prepararBase() {
  const bdPreparado = await Storage.get({ key: BD_CHAVE_CONFIG });

  if (!bdPreparado.value) {
    this.downloadUsers();
  } else {
    this.bdNome = (await Storage.get({ key: BD_CHAVE_NOME })).value;
    await CapacitorSQLite.open({ database: this.bdNome });
    this.bdPronto.next(true);
  }
}
```

Fonte: Elaborado pelo Autor (2021)

Figura 13 – Trabalho com Usuários

```
private downloadUsers(atualizar = false) {
  this.http.get('http://127.0.0.1:8000/api/v1/users').subscribe(async (jsonExport: JsonSQLite) => {
    const jsonstring = JSON.stringify(jsonExport);
    const valido = await CapacitorSQLite.isJsonValid({ jsonstring });

    if (valido.result) {
      this.bdNome = jsonExport.database;
      await Storage.set({ key: BD_CHAVE_NOME, value: this.bdNome });
      await CapacitorSQLite.importFromJson({ jsonstring });
      await Storage.set({ key: BD_CHAVE_CONFIG, value: '1' });

      // Mudanças offline
      if (!atualizar) {
        await CapacitorSQLite.criarTabelaSincronia();
      } else {
        await CapacitorSQLite.setDataSincronia({ datasinc: '' + new Date().getTime() })
      }
      this.bdPronto.next(true);
    }
  });
}
```

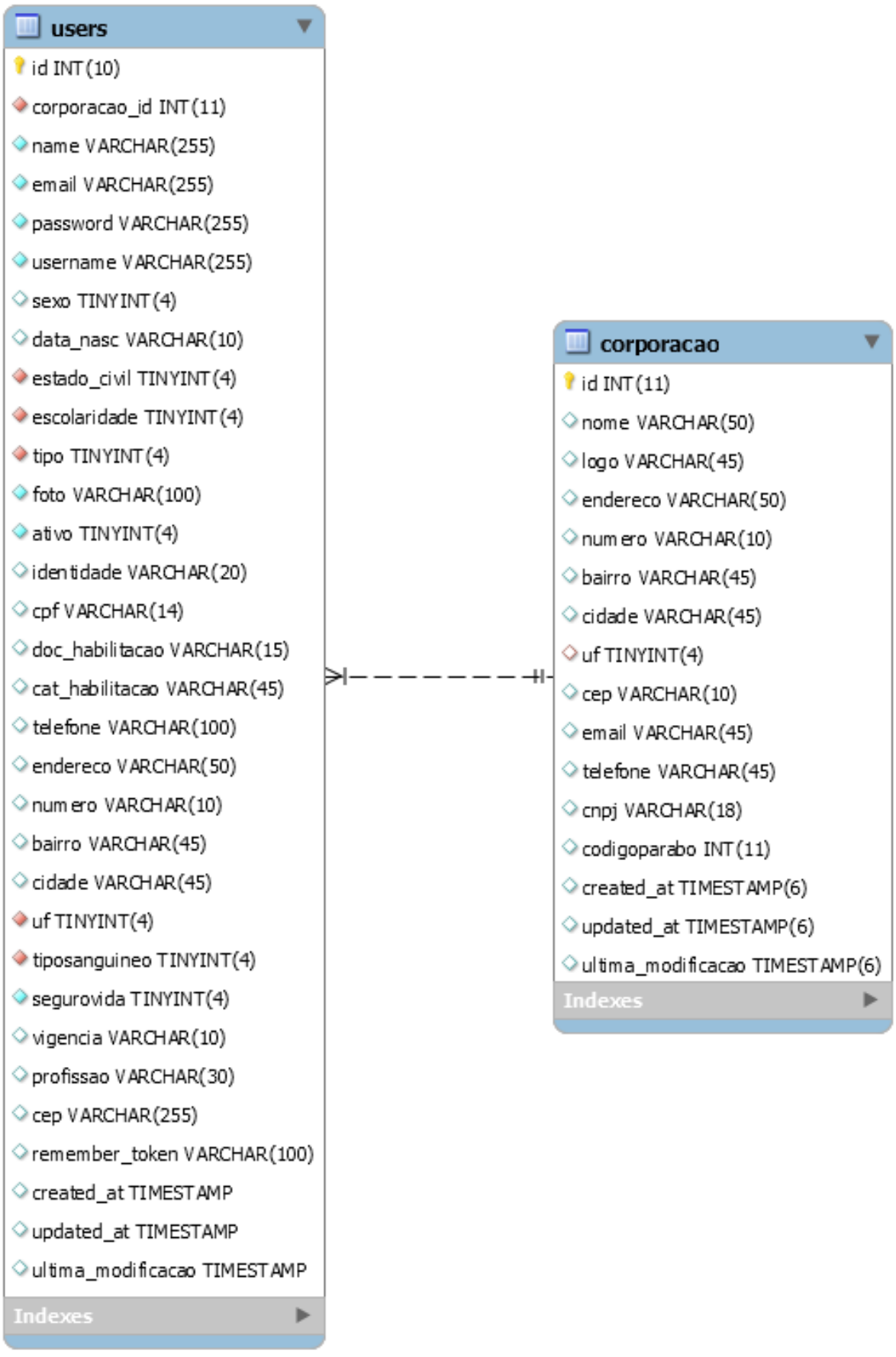
Fonte: Elaborado pelo Autor (2021)

3.3.4.1 Modelagem do banco de dados

Como mencionado acima, duas alternativas relacionadas ao armazenamento foram utilizadas no desenvolvimento do trabalho. Para cuidar dos usuários que acessam o aplicativo mesmo sem conexão disponível e as corporações que têm suas informações em destaque, foi utilizado o banco de dados relacional SQLite [6]. A Figura 14 mostra o diagrama Entidade Relacionamento referente ao usuário e a corporação que existem no aplicativo SisGera, para trabalhar com as duas tabelas relacionais e para tratar a sincronia. A única mudança em relação às tabelas criadas em (ARANTES, 2018) foi o acréscimo da coluna última modificação. Isso se deve ao fato de que a exportação para JSON incluirá as tabelas, índices ou dados que foram modificados após a data de sincronização.

[6] Disponível em: <https://www.sqlite.org/index.html> Acessado em: 30/04/2021

Figura 14 – Diagrama Entidade-Relacionamento para o usuario e corporação.



Fonte: Elaborado pelo Autor (2021)

3.3.5 Conexão e Sincronia

Em primeiro lugar, é preciso saber se existe conexão de rede. Para isso, foi desenvolvido um serviço que nos informa sobre todas as mudanças que acontecem na rede. Um serviço no ambiente Ionic pode ser visto como uma classe que pode atuar como um intermediário entre componentes ou como um armazenamento de dados para outros componentes, auxiliando no estado dos dados em todo o aplicativo ou até mesmo interagir com outros serviços. Criando um serviço para verificar o status da conexão, também evita gastar muito espaço nas páginas que tratam um formulário.

A Figura 15 exibe um observável que nos informa sobre todas as mudanças que acontecem na rede. Se uma conexão estiver disponível, é indicado ao usuário em tempo real na tela do dispositivo, além de exibir um alerta que indica o usuário se a conexão se manteve. Essas informações podem ser vistas na Figura 16. Para uma sincronização de dados mais confiável, sabendo da instabilidade das redes disponíveis nas estradas, o botão para sincronizar os boletins só fica disponível a partir de uma rede 3g, 4g ou Wi-fi.

Figura 15 – Observável do Estado da Conexão

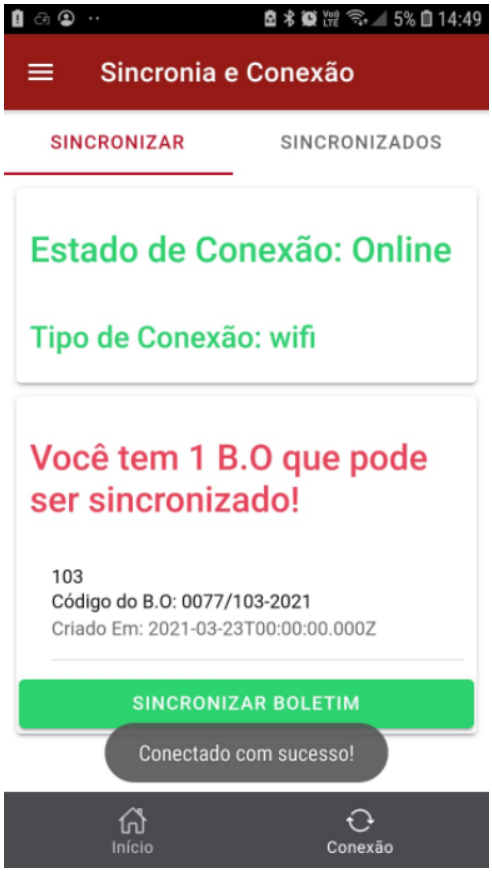
```
this.conexaoListener = Network.addListener('networkStatusChange', (status) => {
  console.log("Estado da conexao", status);
  this.conexaoStatus = status;
  if(status.connected){
    this._statusConexao.pipe(take(1)).subscribe(conexao => {
      this._statusConexao.next(ConnectionStatus.Online);
    });
    ref.exibirToast("Conectado com sucesso!")
  }else{
    this._statusConexao.pipe(take(1)).subscribe(conexao => {
      this._statusConexao.next(ConnectionStatus.Offline);
    });
    ref.exibirToast("Internet desconectada!")
  }
});

this.conexaoStatus = await Network.getStatus();
}
```

Fonte: Elaborado pelo Autor (2021)

Em seguida, é necessário armazenar as solicitações feitas durante o tempo em que o dispositivo estiver sem conexão. Em certos casos quando o dispositivo voltar a ter conexão, não é preciso obter certos dados novamente, por exemplo, como os dados do usuário. Entretanto, como o aplicativo envia requisições para inclusão de novos dados, é preciso manter o controle. A maior dificuldade encontrada nesse processo é devida ao

Figura 16 – Conexão com Wi-fi



Fonte: Elaborado pelo Autor (2021)

fato de que não é armazenada apenas a solicitação, mas também deve ser executada a operação localmente, de forma antecipada. Portanto, quando os boletins são sincronizados, é utilizado o *Local Storage*.

Utilizando o *Local Storage*, caso a conexão esteja disponível, é enviado o formulário cadastrado para ser salvo. Porém, caso a conexão não esteja disponível, o formulário e a requisição são salvos e esperam até que sejam enviados à API, quando o usuário selecionar o botão de sincronia ou quando o dispositivo encontrar uma conexão Wi-fi.

Conforme mencionado anteriormente, os boletins também podem ser sincronizados automaticamente. Porém, este caso será habilitado apenas quando o dispositivo estiver conectado à uma rede Wi-fi, evitando possíveis problemas de instabilidades da conexão. Essa tarefa é realizada pela função `observaEvento` como é exibido na Figura 17. Ela será chamada sempre que o usuário estiver online novamente em uma rede Wi-fi. Basicamente, é obtido o array do armazenamento e são enviadas todas as requisições.

Figura 17 – Envio Automático

```
observaEvento(): Observable<any> {
  let ref = this;
  return from(this._storage.get(CHAVE_REQ_BD)).pipe(
    switchMap(storedOperations => {
      let formulario = JSON.parse(storedOperations);
      if (formulario && formulario.length > 0) {
        return this.enviarRequisicoes(formulario).pipe(
          finalize(() => {
            ref. exibirToast("Boletins Sincronizados via Wi-fi!")
            this._storage.remove(CHAVE_REQ_BD);
          })
        );
      }else {
          //Não existe formulário para sincronizar
          return of(false);
      }
    })
  );
}
```

Fonte: Elaborado pelo Autor (2021)

3.4 Testes

3.4.1 API - Postman

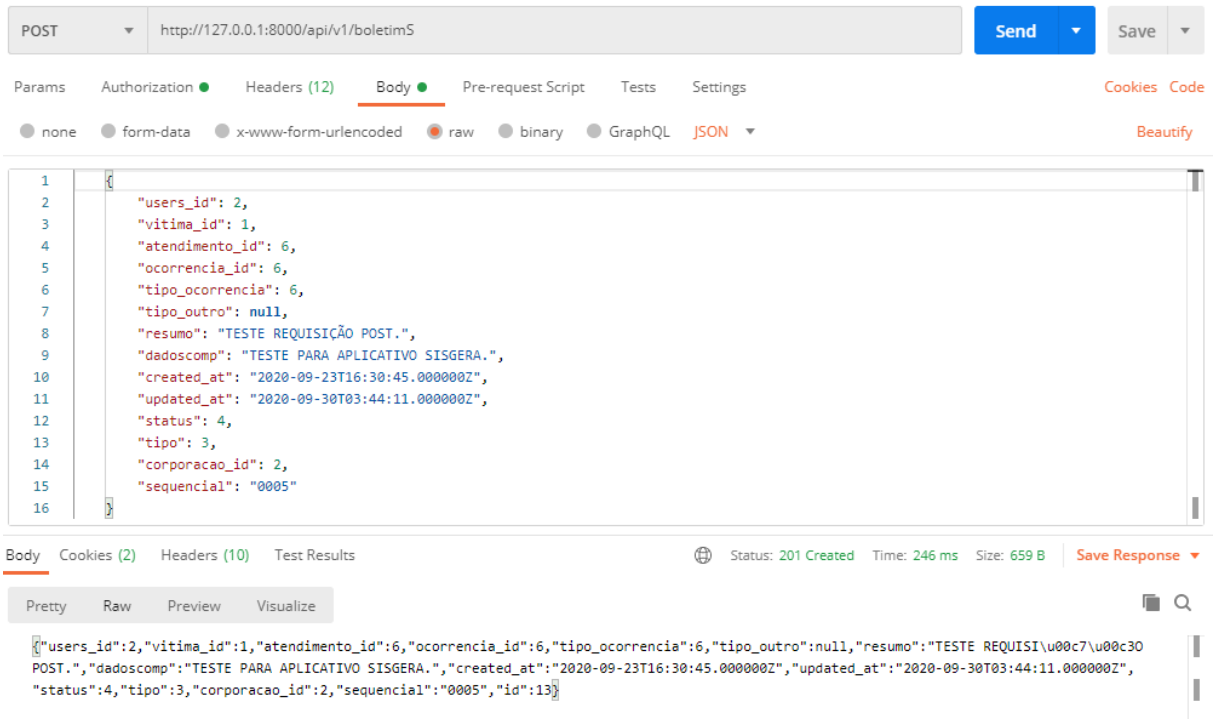
Durante o desenvolvimento do aplicativo e da API, foram encontradas necessidades de testar as rotas e obter auxílio para as requisições desenvolvidas. A melhor forma encontrada para tais situações foi a utilização do *software* Postman [7].

O Postman consiste em uma ferramenta de apoio ao desenvolvimento de APIs. Os recursos que o Postman oferece simplificam cada etapa da construção de uma API e nos ajudam a testar e desenvolver APIs em uma interface bastante simples e intuitiva. Quando realizamos nossas requisições, o Postman nos ajuda com as respostas enviadas pela API e as exibe visualmente de forma muito agradável e de fácil compreensão.

[7] Disponível em: <https://www.postman.com/> Acessado em: 30/04/2021

A Figura 18 nos apresenta a utilização do Postman para realizar um teste de inclusão de um boletim simplificado através do método de comunicação POST. Podemos visualizar o conteúdo do corpo da requisição, além do código de status da respostas HTTP informado, que no caso é 201 *Created*. que nos indica que a requisição foi bem sucedida e um novo recurso foi criado como resultado.

Figura 18 – Requisição POST sendo testada com o Postman



Fonte: Elaborado pelo Autor (2021)

3.4.2 Aplicativo

Para o desenvolvimento de testes para o aplicativo foram utilizados dois *framworks* para aplicações que utilizam Javascript, principalmente o Angular. O Jasmine[8] e o Karma[9].

O Jasmine é um *framework* que é utilizado independente de um navegador *Web* e não precisa de outras bibliotecas para funcionar. Com o Jasmine, foram desenvolvidos alguns testes de unidade. Ele possui uma sintaxe limpa para que o teste possa ser desenvolvido de forma simples. A Figura 19 exibe alguns dos casos de teste escritos para o aplicativo. Foram escritos testes para os serviços, páginas, dependências, elementos das *views* e também chamadas assíncronas.

[8] Disponível em: <https://jasmine.github.io/> Acessado em: 30/04/2021
[9] Disponível em: <https://karma-runner.github.io/1.0/index.html> Acessado em: 30/04/2021

Figura 19 – Testes escritos utilizando o Jasmine

```
it('Serviço Existe?', function(){
  expect(SincroniaService).toBeDefined();
});

it('Método Existe?', function(){
  expect(SincroniaService.sincronizarBoletim()).toBeDefined();
});

it('Deve Retornar um Array de Formulário', () => {
  expect(Array.isArray(_formularios)).toBeTruthy();
});

it('Deve conter o título correto', () => {
  let de = fixture.debugElement.query(By.css('ion-title'));
  let elemento = de.nativeElement;
  expect(elemento.textContent).toContain('Todos');
});
```

Fonte: Elaborado pelo Autor (2021)

Já o Karma, é um teste *runner* feito para o Angular. O principal objetivo do Karma é possibilitar a execução dos testes em diversos navegadores *Web* com um único comando. Assim, com os testes escritos utilizando a sintaxe do Jasmine, é possível utilizar o Karma para auxiliar no processo de execução dos testes.

4 Resultados

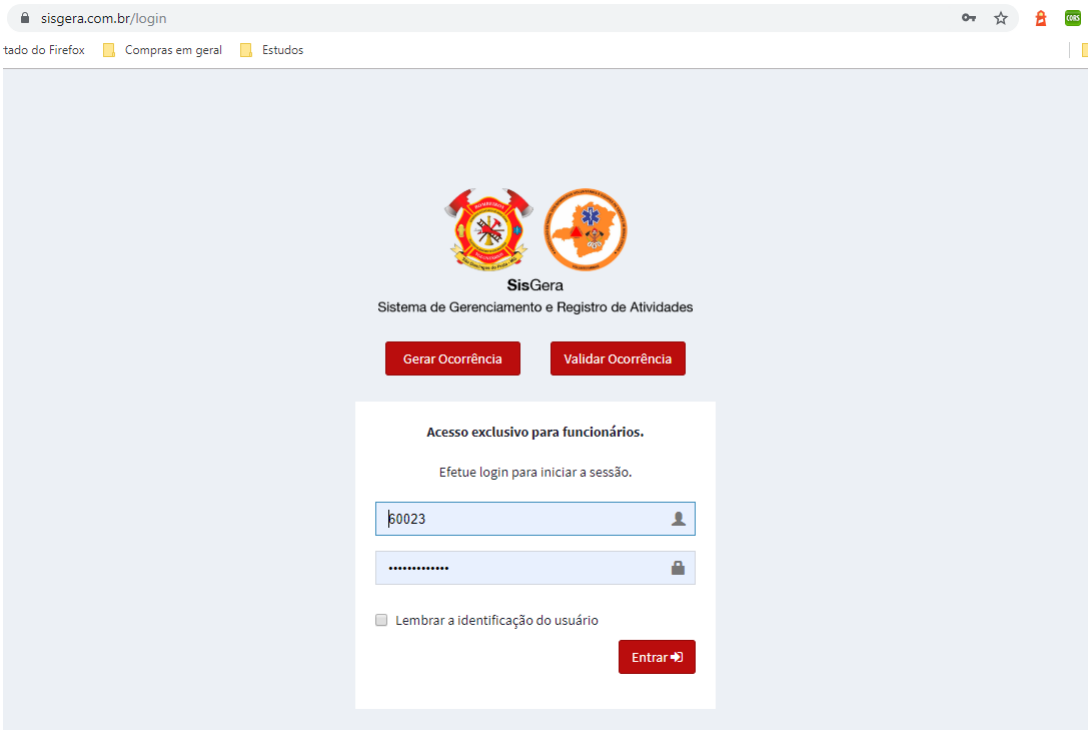
Esta seção apresenta a aplicação móvel desenvolvida para o projeto SisGera, onde é feita uma breve descrição dos ambientes do aplicativo, bem como um exemplo visual da tela e suas funcionalidades.

4.1 Apresentação das telas

4.1.1 Acesso ao sistema

O aplicativo móvel do sistema SisGera foi desenvolvido baseado no sistema *Web* que os bombeiros já estão adaptados e treinados a utilizar. Portanto, existe uma semelhança visível entre as aplicações o que faz com que o usuário, desde o momento do login já se sinta familiarizado com o sistema *Web* principal, incluindo o layout, menus e alertas. O primeiro contato do usuário com o aplicativo se dá pela tela de login. A Figura 20 e a Figura 21 exibem a tela em ambas aplicações para exibir a semelhança citada acima.

Figura 20 – Tela acesso do SisGera *Web*



Fonte: Elaborado pelo autor (2021)

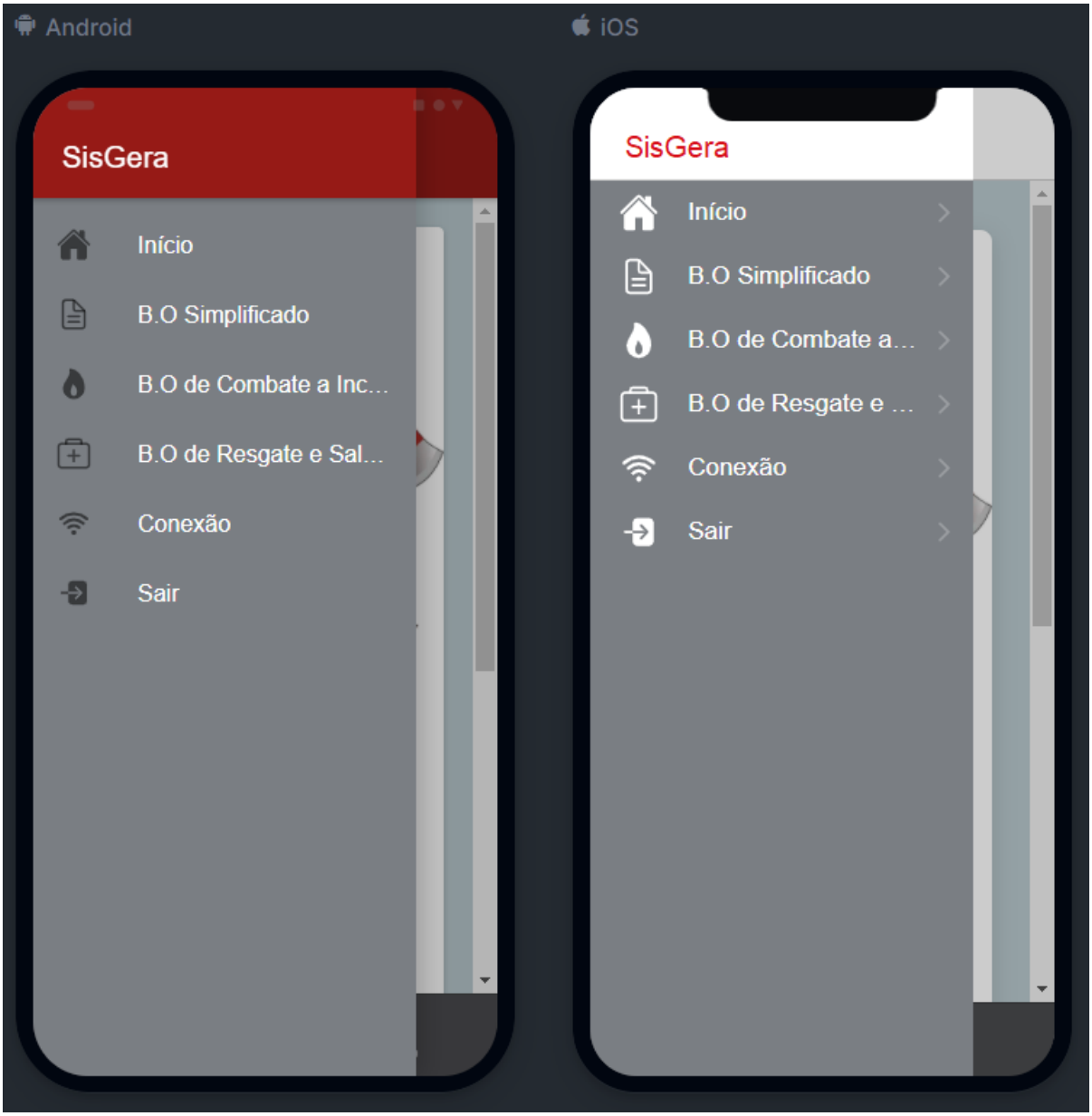
Figura 21 – Tela acesso do SisGera App

Fonte: Elaborado pelo autor (2021)

4.1.2 Menu principal

O aplicativo não possui todos os ambientes e menus que estão disponíveis atualmente na versão *Web*. Como o foco é o cadastro dos boletins de ocorrência, o menu lateral é exibido de forma mais simples e direta. O usuário tem como opção acessar um dos três tipos de boletim: o simplificado, o de resgate/salvamento e o de combate a incêndio, que por enquanto ainda não é utilizado pelas corporações. Além dos boletins, os usuários podem também verificar o status de sua conexão e realizar a sincronia de dados acessando a opção de conexão. Além das funcionalidades citadas acima que são mais específicas, o menu lateral do aplicativo conta com ações básicas como retornar para a tela de início ou sair do sistema. Na Figura 22, podemos visualizar o menu sendo utilizado tanto no Android quanto no iOS. Para utilizar o menu, basta deslizar o dedo na tela ou pressionar o ícone correspondente ao mesmo no canto superior esquerdo da tela.

Figura 22 – Menu inicial do aplicativo no Android e iOS



Fonte: Elaborado pelo autor (2021)

4.1.3 Tela de Início

A tela inicial do aplicativo é uma apresentação onde o usuário pode verificar seus dados após efetuar acesso no aplicativo, tanto dados pessoais do usuário logado, como os da corporação que ele atualmente representa. Junto à tela inicial do aplicativo, é possível visualizar uma barra inferior que contém dois ícones. Um dos ícones contém a ação de direcionar a própria tela inicial e o outro ícone direciona o usuário para a tela de sincronia, proporcionando uma navegação fluida entre as duas telas do aplicativo. A Figura 23 exibe a tela inicial que, por conter muitas informações, foi desenvolvida também de forma

responsiva, bem como todas as telas do aplicativo.

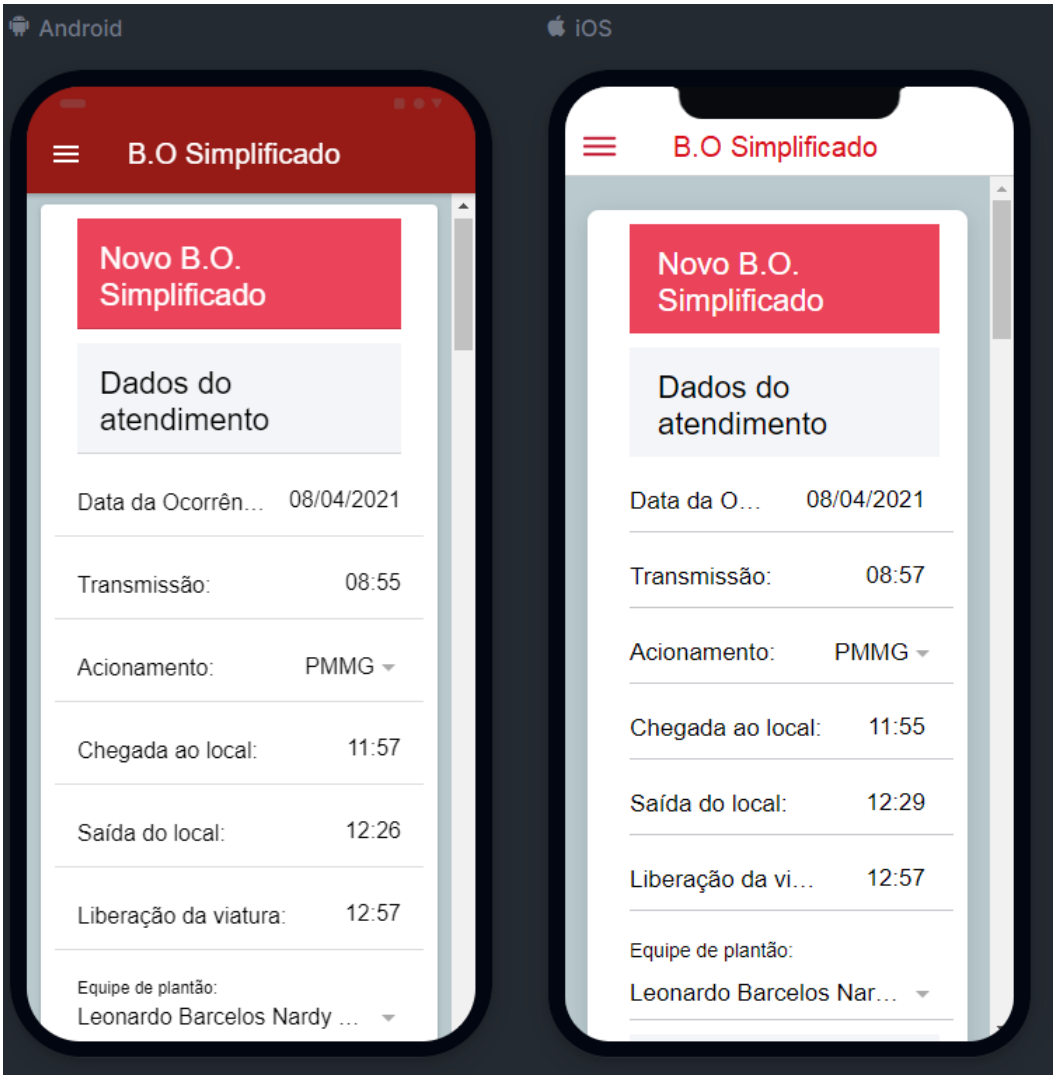
Figura 23 – Tela inicial do aplicativo

Fonte: Elaborado pelo autor (2021)

4.1.4 Registro de B.O

A funcionalidade para registro de boletins de ocorrência é totalmente baseada no sistema *Web* do SisGera. Os formulários são semelhantes em questão de layout e validação dos campos no momento da inclusão. O formulário do boletim no momento de inclusão verifica a conexão e também a qualidade da rede disponível. Caso a qualidade seja satisfatória, envia os dados diretamente para o servidor. Em caso contrário, os dados são salvos na memória local do dispositivo e podem ser sincronizados posteriormente na tela de conexão e sincronia. A Figura 24 demonstra um exemplo de inclusão de um boletim simplificado no Android e no iOS.

Figura 24 – Novo boletim de ocorrência

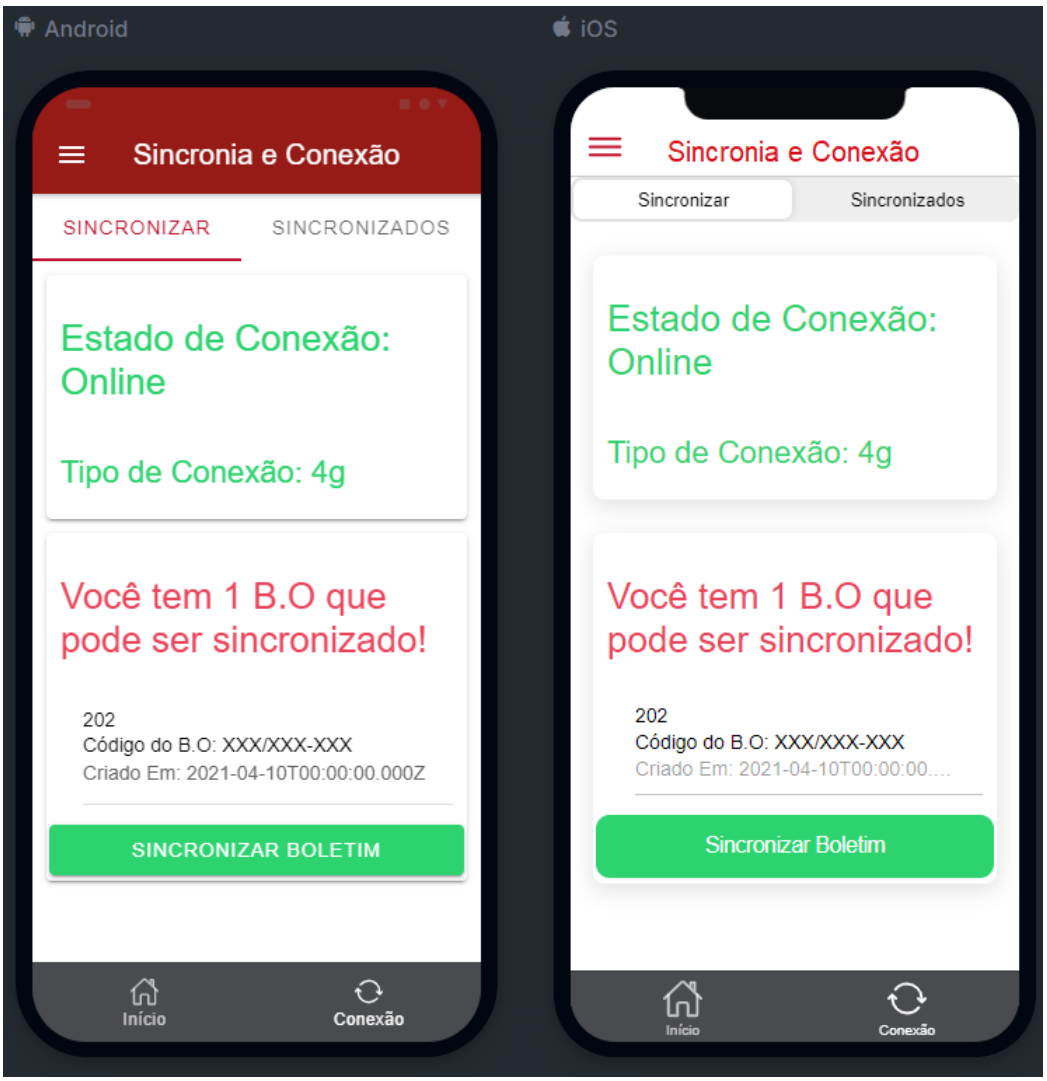


Fonte: Elaborado pelo autor (2021)

4.1.5 Conexão e sincronia

A funcionalidade para conexão e sincronia tem como objetivo exibir a atual situação da conexão do dispositivo com a rede de Internet. Ela também disponibiliza o botão para os formulários armazenados serem sincronizados com o servidor por meio da API. Para a realização de um tráfego mais confiável das requisições, a conexão do dispositivo deverá no mínimo possuir uma rede 3G estável. Isso garante que os dados serão transportados corretamente. O aplicativo também permite a sincronização automática, mas apenas quando o dispositivo estiver conectado a uma rede Wi-fi. Os formulários que estão aguardando serem sincronizados são exibidos também na tela permitindo que o usuário tenha um controle mais confiável de suas requisições. A Figura 25 demonstra a tela de conexão e sincronia do aplicativo nas plataformas Android e iOS.

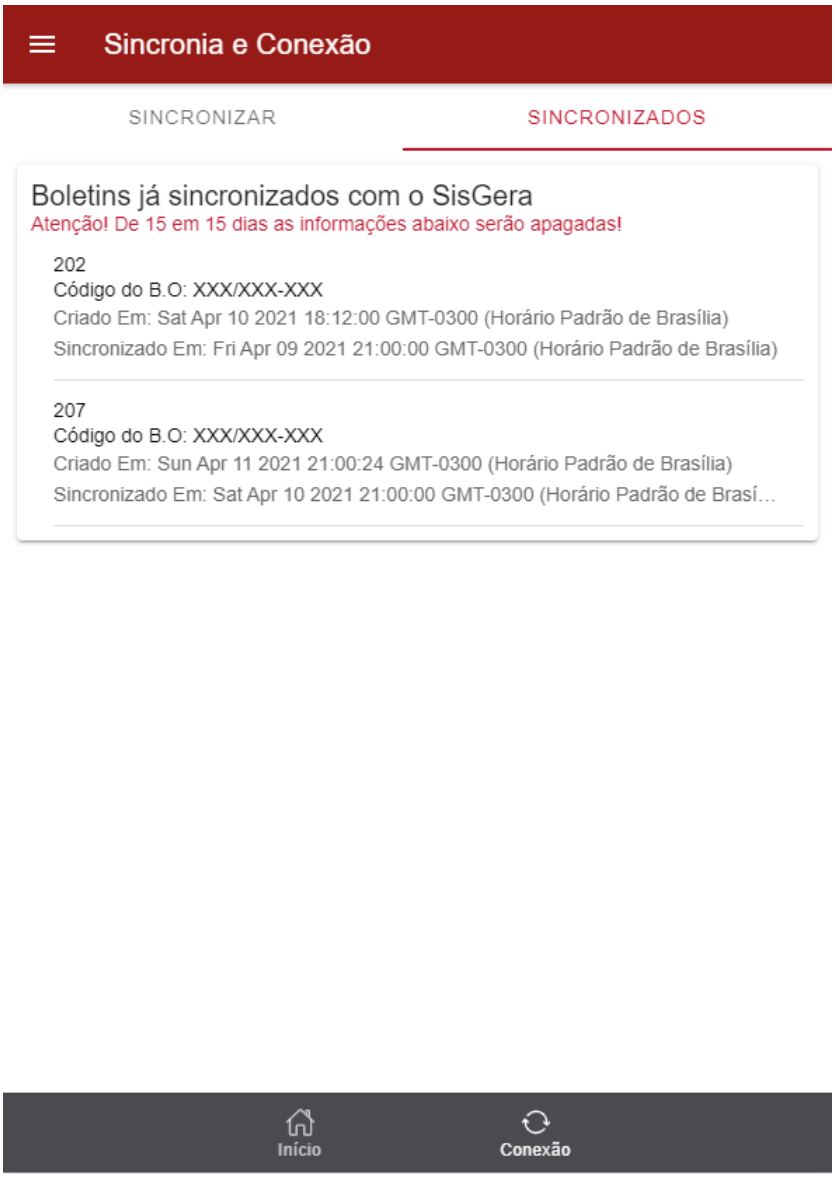
Figura 25 – Sincronia e Conexão



Fonte: Elaborado pelo autor (2021)

Os boletins que são sincronizados com o servidor com sucesso também podem ser visualizados, selecionando a aba "Sincronizados". Eles permanecem disponíveis no aplicativo durante 15 dias para que o usuário possa consultar e organizar suas ações. Após os 15 dias, por questões de segurança, os dados serão apagados do histórico.

Figura 26 – Sincronizados



Fonte: Elaborado pelo autor (2021)

5 Considerações Finais

Este trabalho apresentou o desenvolvimento de uma aplicativo móvel multiplataforma para apoio ao sistema Web SisGera, que é um sistema utilizado atualmente por bombeiros voluntários que prestam serviços de resgate, salvamento e combate a incêndio. O aplicativo em questão foi desenvolvido buscando proporcionar uma utilização sem conexão disponível de forma confiável, permitindo com que os bombeiros a utilizem mesmo sem conexão com a rede de dados ou mesmo com uma conexão instável. É esperado um grande impacto positivo na rotina dos bombeiros com o aplicativo, que será utilizado em conjunto com a versão Web. O aplicativo foi desenvolvido para ser facilmente utilizado por quem já utiliza a versão Web, já que o *layout*, responsividade e interação com usuário são semelhantes.

O trabalho iniciou com uma revisão da literatura acerca dos assuntos pertinentes, como dispositivos móveis e suas plataformas. Foram elucidados os tipos de desenvolvimento multiplataforma, assim como o desenvolvimento nativo tradicional, analisando algumas vantagens e desvantagens de cada abordagem. Foram apresentados o sistema Web do SisGera e, também, conceitos sobre o teste de software.

Durante o desenvolvimento do trabalho, foram encontrados problemas relacionados a questões sobre sincronia e sobre suporte sem conexão disponível. Os boletins, por exemplo, estavam sendo eliminados do telefone quando o mesmo se encontrava com pouco espaço de armazenamento disponível. Quando era realizada uma limpeza geral do dispositivo para obter mais espaço, os dados armazenados para serem sincronizados posteriormente eram eliminados do dispositivo. Após pesquisas sobre os obstáculos encontrados e com a realização de mais testes, um novo *plugin* foi utilizado pra trabalhar com o armazenamento local do aplicativo, de forma que este problema não acontece mais.

Também no desenvolvimento do trabalho, foram apresentadas as tecnologias e abordagens utilizadas para a criação das funcionalidades, bem como as técnicas utilizadas para desenvolvimento geral do aplicativo e como elas se integram. Testes foram utilizados para a validação das funcionalidades desenvolvidas e da API. Com base na análise realizada no desenvolvimento das funcionalidades, foram levantados alguns pontos como propostas para trabalhos futuros, que são apresentados a seguir:

- Desenvolvimento de funcionalidade visando um controle de acesso biométrico;
- Desenvolvimento de funcionalidade para a aplicação trabalhar com mapas e executar funções baseadas em localização;
- Testes e avaliações de usabilidade com os usuários do sistema.

- Migração do *plugin Local Storage* que controla os formulários de boletim armazenados para o *plugin* SQLite.

Referências

ARANTES, V. M. Um sistema de informação para apoio ao registro de ocorrências atendidas por grupos de bombeiros voluntários. João Monlevade, MG, 2018. Disponível em: <https://goo.gl/Yu2SFe>. Citado 5 vezes nas páginas 14, 15, 27, 28 e 42.

B'FAR, R. *Mobile Computing Principles*. 9. ed. Cambridge University Press: [s.n.], 2004. Citado na página 17.

BIORN-HANSEN, A.; GRONLI, T.-M.; GHINEA, G. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. New York, NY, USA, 2018. Citado 2 vezes nas páginas 22 e 27.

COHN, M. *User Stories Applied: For Agile Software Development*. [S.l.: s.n.], 2004. Citado na página 37.

EL-KASSAS et al. Taxonomy of cross-platform mobile applications development approaches. Cairo, Egypt, 2015. Disponível em: <https://doi.org/10.1016/j.asej.2015.08.004>. Citado 6 vezes nas páginas 19, 20, 21, 22, 24 e 25.

GRONER, L. *Progressive Web Apps: Conheça Parte do Processo de evolução da web*. 26. ed. São Paulo: [s.n.], 2018. Citado na página 23.

LATIF; LAKHRISSI; ES-SBAI. *Cross platform approach for mobile application development: A survey*. [S.l.: s.n.], 2016. Citado na página 26.

LAUDON, K.; LAUDON, J. *Sistemas de Informações Gerenciais*. 9. ed. São Paulo: [s.n.], 2011. Citado na página 17.

MARINHO, E.; RESENDE, R. Native and multiple targeted mobile applications. 2015. Citado na página 21.

MONTAN, J. dos S. *Avaliação de plataformas híbridas para desenvolvimento de aplicações para o android*. 1. ed. Cambridge University Press: [s.n.], 2017. Citado na página 18.

OLIVEIRA, S. S. M. Sistema de informação para controlede materiais e doações aos bombeiros voluntários. João Monlevade, MG, 2018. Disponível em: <https://goo.gl/Yu2SFe>. Citado 4 vezes nas páginas 14, 15, 27 e 28.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional*. 7. ed. Porto Alegre: [s.n.], 2011. Citado na página 30.

WEI, L.; LIU, Y.; CHEUNG, S.-C. Taming android fragmentation: Characterizing and detecting compatibility issues for android apps. New York, NY, USA, 2016. Citado na página 20.

ZHANG, D.; ADIPAT, B. Challenges, methodologies, and issues in the usability testing of mobile applications. 2005. Citado na página 29.