UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOÃO EDUARDO KERNITSKEI

MARCOS VINICIUS LABRES DE OLIVEIRA

DESENVOLVIMENTO DE UM APLICATIVO PARA GERENCIAMENTO DE EVENTOS PARA ORGANIZAÇÕES RELIGIOSAS DA REGIÃO DOS CAMPOS GERAIS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2019

MARCOS VINICIUS LABRES DE OLIVEIRA

JOÃO EDUARDO KERNITSKEI

DESENVOLVIMENTO DE UM APLICATIVO PARA GERENCIAMENTO DE EVENTOS PARA ORGANIZAÇÕES RELIGIOSAS DA REGIÃO DOS CAMPOS GERAIS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática / Coordenação do Curso Superior em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof.ª Drª Simone de Almeida

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO PARA GERENCIAMENTO DE EVENTOS PARA ORGANIZAÇÕES RELIGIOSAS DA REGIÃO DOS CAMPOS GERAIS

por

JOÃO EDUARDO KERNITSKEI
MARCOS VINICIUS LABRES DE OLIVEIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 29 de maio de 2019 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof(a)./Dra. Simone de Almeida
Orientadora

_____
Prof./Dr. Diego Roberto Antunes
Membro titular

_____
Prof./Dr. Richard Duarte Ribeiro
Membro titular

_____
Prof./MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof./Dr. André Pinz Borges
Coordenador de curso

RESUMO

KERNITSKEI, João Eduardo, OLIVEIRA, Marcos Vinicius Labres de, **Desenvolvimento de um aplicativo para gerenciamento de eventos para organizações religiosas da região dos campos gerais**. 2019. 49 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

Este trabalho apresenta uma estrutura necessária para o desenvolvimento de um aplicativo móvel, o qual objetiva gerenciar eventos de uma organização religiosa dos campos gerais. Para o desenvolvimento foi utilizado o banco de dados NoSql Firebase Realtime Database, e para o desenvolvimento do aplicativo foi utilizado o framework Ionic, juntamente com o AngularJS e com o Apache Cordova. Para reduzir o tempo de desenvolvimento e sintetizar a documentação necessária do sistema, foi adotada a tecnologia de desenvolvimento ágil XP (*Extreme programming*).

Palavras-chave: Religião. Ionic. Firebase. XP. Rede Social. Aplicativo Móvel.

ABSTRACT

KERNITSKEI, João Eduardo, OLIVEIRA, Marcos Vinicius Labres de, **Development of an event management application for religious organizations in the general field region**. 2019. 49 p. Course Conclusion Work (Technology in Analysis and Development of Systems) - Federal Technological University of Paraná. Ponta Grossa, 2019.

This work presents a necessary framework for the development of a mobile application, which aims to manage events of a religious organization from the general fields. For development, the NoSql Firebase Realtime Database was used, and for the application development the Ionic framework was used along with AngularJS and Apache Cordova. To reduce the development time and synthesize the necessary documentation of the system, it adopted the technology of agile development XP (Extreme programming).

Keywords: Religion. Ionic. Firebase. XP. Social network. Mobile Application.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Cada vez mais as organizações têm feito uso das redes sociais com o objetivo de divulgar suas atividades, eventos, e/ou acontecimentos a toda sociedade, de uma forma fácil e de baixo custo. Segundo Telles (2010), as redes sociais são uma ótima maneira de divulgar uma marca.

Da mesma forma, as organizações religiosas têm encontrado nas redes sociais uma forma de disponibilizar conteúdos e informações, com o objetivo de elevar o número de adeptos a religião, principalmente o público jovem (SILVA E NALINI, 2015).

De acordo com o IBGE (2010), o último censo realizado sobre o número de evangélicos no Brasil, mostra que esse número subiu de 26,2 milhões para 42,3 milhões, representando 2,2 por cento da população. O mesmo censo apontou que a religião evangélica tem maior aceitação entre os jovens e adolescentes.

De forma a centralizar o conteúdo relacionado as redes sociais discutidas neste trabalho, serão abordados os dados relacionados a rede social *Facebook*, que hoje comporta mais de 2,13 milhões de usuários, segundo a versão eletrônica do jornal Estadão (ESTADÃO, 2018).

Devido ao grande número de usuários, o conteúdo disponibilizado no *Facebook* é oriundo de diversas vertentes e dos mais diversos temas, que por sua vez, acaba por gerar conflitos frente ao conteúdo disponibilizado pelas organizações religiosas. Segundo o *site* BBC (2015), os assuntos mais divulgados e discutidos no *Facebook* estão relacionados aos escândalos políticos, eventos de música não religiosas, casamento igualitário, entre outras.

Este trabalho tem como proposta apresentar a estrutura de um aplicativo, que atenderá as necessidades de uma organização religiosa localizada na região dos campos gerais, a qual envolverá 37 igrejas.

O aplicativo desenvolvido aplica tecnologias Web, utilizando várias ferramentas, com destaque ao uso do *framework* Ionic, tecnologia de código aberto que utiliza em sua estrutura linguagens de desenvolvimento Web, tais como HTML 5, CSS e *JavaScript*, podendo ser interpretado por um navegador ou então gerar aplicativos móveis para diferentes plataformas, com o auxílio do Apache Cordova, tecnologia também de código aberto (SILVA, 2016).

O aplicativo resultante dessa estrutura fornece a organização uma opção para disponibilizar e divulgar suas atividades, por meio de uma ferramenta específica ao contexto religioso, de forma simples e similar ao fornecido pelas redes sociais. Cada igreja irá gerenciar um perfil solicitado pelo administrador do aplicativo. Cada usuário comum pode acessar todos os eventos divulgados no aplicativo, pode também gerenciar sua conta e customiza-la conforme a sua necessidade.

Como citado anteriormente, o *Facebook* é utilizado como principal referência, o qual dispõe de funcionalidades de grande utilidade, porém que se destina a divulgação de conteúdos de cunho religioso, não sendo necessário criar grupos isolados, garantindo assim a facilidade de uso por parte do público alvo.

1.1. OBJETIVOS

São apresentados a seguir o objetivo geral do trabalho assim como os objetivos específicos.

1.1.1. Objetivo Geral

Desenvolver um sistema que atenda as especificidades do contexto religioso, similar ao de uma rede social, desenvolvido dentro das tecnologias Web.

1.1.2. Objetivos Específicos

Juntamente ao objetivo geral foram destacados alguns objetivos específicos, como segue:

- Fornecer um aplicativo móvel nas plataformas Android e IOS que possa ser acessado de qualquer dispositivo móvel;
- Permitir que usuários comuns possam criar contas de forma a acessar as informações disponibilizadas pelas igrejas, dessa forma conhecendo em detalhes os eventos de cada igreja pertencente a organização religiosa em questão;

- Disponibilizar eventos realizados pelas igrejas pertencentes à organização religiosa, em forma de publicações, além de fornecer funcionalidades relacionadas aos eventos publicados de modo a atrair os usuários;
- Padronizar a forma com que as igrejas pertencentes à organização divulgam seus eventos, centralizando o material que atualmente é disponibilizado no *Facebook* em um sistema pertencente à organização, de modo a gerenciá-los conforme a necessidade de cada igreja. Cada uma terá controle sobre as suas informações e eventos criados por ela.

1.2. JUSTIFICATIVA

O conteúdo divulgado pelas organizações religiosas dentro das mídias sociais tem sido prejudicado com material divergente como pornografia, consumo de entorpecentes, ideologia de gênero, corrupção política, entre outras.

As organizações religiosas têm sido alvo de críticas e manifestações contrárias, como mostra a notícia publicada no *site* G1 (2015), onde o movimento LGBT (lésbicas, gays, bissexuais e transexuais) utiliza de símbolos religiosos de forma irônica e provocativa. Também no *site* G1 (2011), outra publicação discute sobre a acusação feita pelo ministério público sobre possível lavagem de dinheiro em uma das maiores igrejas do país. As opiniões sobre as apurações de tais publicações não são relevantes neste trabalho, e não é o objetivo discuti-las aqui, mas o fato que estas publicações podem se misturar ao conteúdo religioso divulgado pelas igrejas dentro da mídia social *Facebook*, devido a estrutura disponibilizada pela mídia social em questão.

Desta forma, os eventos divulgados pela organização religiosa, se misturam em meio ao conteúdo de outros usuários, ou outras organizações. As divergências geradas pelos diversos assuntos, oriundos de diversas fontes, muitas vezes conflitam com assuntos de contexto religioso, abrindo questionamentos e críticas, gerando oposição de assuntos diversos, e desviando o objetivo da organização religiosa em divulgar seus eventos e atividades. A estrutura oferecida pelo *Facebook* fornece aos seus usuários a possibilidade de gerenciar grupos, onde é possível

restringir que conflitos de contexto aconteçam, porém, os acessos às informações de como criar os grupos se tornam indiretas, aumentando a navegabilidade do sistema e diminuindo consequentemente a facilidade de uso.

1.3. METODOLOGIA

Para o desenvolvimento do aplicativo proposto por este trabalho, foi aplicada a metodologia de desenvolvimento ágil *Extreme Programming* (XP), que segundo Teles (2014), tem como finalidade simplificar a busca por atingir o objetivo do sistema. O desenvolvimento total de trabalho foi dividido em três principais partes: levantamento aproximado do número de postagens realizadas pelas igrejas pertencentes à organização religiosa em questão, levantamento dos requisitos para a elaboração da documentação do sistema e apresentação das tecnologias necessárias, com base no trabalho elaborado na análise de requisitos.

A primeira fase possibilitou conhecer com qual frequência as igrejas realizam postagens na rede social atualmente. Esse levantamento foi realizado por meio de monitoramento nos perfis de cada igreja dentro da mídia social *Facebook*. Com base nos números levantados, deu início à segunda fase, que será realizar o levantamento dos requisitos com a estrutura necessária e com as funcionalidades utilizadas na ferramenta em uso, requeridas à divulgação e gerenciamento do perfil de cada igreja. Essas informações direcionaram o desenvolvimento da documentação e diagramas UML (Linguagem de Modelagem Unificada).

Com base na documentação, seguiu-se a terceira fase, a apresentação das tecnologias que foram utilizadas para o desenvolvimento do aplicativo, onde foi utilizado do *framework* Ionic, que segundo Silva (2016) é um dos *frameworks* para desenvolvimento de aplicativos híbridos mais utilizados no mundo. Aplicativos híbridos referem-se à aplicativos que utilizam tecnologias Web e nativas, podendo ser utilizados em diferentes plataformas.

Para que o Ionic possa gerenciar suas aplicações e acessar os recursos nativos de um *smartphone*, ele faz uso do Apache Cordova, *framework* de código aberto, que é responsável pelo acesso as diferentes APIs (Interface de Programação de Aplicativos) nativas e também gerar os aplicativos móveis (CORDOVA, 2015).

Nesta fase também será apresentada a utilização do *Firebase Firestore*, um banco de dados NoSql hospedado na nuvem, que sincroniza os dados em tempo real (FIREBASE, 2018).

1.4. RESULTADOS

O trabalho forneceu uma ferramenta que contém apenas as funcionalidades necessárias às organizações religiosas, que deve atingir as expectativas dos usuários a serem alcançados pelas igrejas, devendo apenas fazer o uso das mídias sociais para realizar a divulgação e redirecionar o usuário para o sistema em questão, até que o mesmo tenha alcançado o seu público alvo.

Fazer com que os eventos de cada igreja alcancem as pessoas pertencentes às igrejas e também pessoas que possam vir a fazer parte desta por meio do uso do aplicativo.

1.5. ESTRUTURA DO TRABALHO

O presente trabalho foi dividido em 4 capítulos, sendo este o primeiro, o qual contém toda a introdução do trabalho, os objetivos, a justificativa para o seu desenvolvimento, a metodologia aplicada e os resultados esperados ao fim do trabalho. No capítulo 2 estão contidos o referencial teórico, que contém todas as fontes utilizadas como referência no desenvolvimento do trabalho; e apresenta o conceito de um aplicativo gerenciador de eventos e todas as tecnologias utilizadas no seu desenvolvimento. O capítulo 3 contém o desenvolvimento, o qual apresenta a estrutura do aplicativo, demonstrando todas as suas funcionalidades exemplificadas através de imagens das suas telas. O capítulo 4 apresenta as considerações finais, que contém a conclusão e os trabalhos futuros, seguido das referências utilizadas no trabalho.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta todo o material utilizado como fonte de consulta para a elaboração deste trabalho. Com base neste referencial, são apresentados os conceitos de um aplicativo gerenciador de eventos, assim como as ferramentas que foram utilizadas para o desenvolvimento da aplicação.

2.1. GERENCIADOR DE VENTOS

A principal característica de um gerenciador de eventos é a fácil e rápida navegação pelo sistema, por isso a predominância na tela é a imagem ou o banner de divulgação. Assim, o usuário é capaz de encontrar mais eficientemente o evento desejado, ou descobrir outros de seu interesse.

O gerenciamento de um evento se faz por vários processos que estão associados às pessoas da administração do evento. Um portal de eventos tem a responsabilidade de contemplar as necessidades destes processos e oferecer suporte a organização do evento para facilitar a divulgação do mesmo.

As atividades na criação do evento, no que diz respeito ao cadastro de uma nova postagem, são simples e intuitivos aos administradores. Para manter um padrão, o preenchimento das informações e de mídia do evento, se dá por meio de um formulário que abrange os principais dados a serem divulgados.

No caso do aplicativo, estes dados cadastrados podem ser usados para acessar recursos nativos do aparelho, como por exemplo para fazer uma chamada telefônica ao responsável do evento, por meio do número que foi disponibilizado na postagem.

Ainda na utilização dos recursos nativos, é possível beneficiar-se de outros meios do dispositivo móvel, desde a funcionalidade de organização oferecidos pela agenda, até o uso da geolocalização.

Vaz (2008) enfatiza o valor da informação como diferencial competitivo para a escolha de ações relativo às necessidades de seus clientes, destacando a tecnologia da informação como ferramenta de importância neste meio, tendo a capacidade de transformar dados em informações preciosas.

2.2. IONIC

O Ionic é uma ferramenta eficiente que possibilita o uso da flexibilidade de elementos do desenvolvimento Web, como as tecnologias HTML (*HyperText Markup Language*), *JavaScript* e CSS (*Cascading Style Sheets*) juntamente com funções nativas, pois “emula as diretrizes da interface do usuário do aplicativo *mobile* utilizando SDKs (*Software Development Kit)* nativos” (IONIC, 2018, nossa tradução).

Esta ferramenta possui licença gratuita e de código aberto, licenciada pelo MIT (*Massachusetts Institute of Technology*), o que garante que “Ele sempre permanecerá livre para se usar, apoiado por uma enorme comunidade mundial” (IONIC, 2018, nossa tradução).

Segundo Bradley (2013), este *framework* possui um *design* similar ao iOS, o sistema operacional móvel da Apple que todavia destaca que o desenvolvedor tem a liberdade de alterar os modelos conforme necessitar, pois os estilos são projetados em CSS.

A escolha de um *framework* de interface de usuário, como no caso do Ionic, possibilita o desenvolvimento de uma aplicação bem estruturada e com um modelo similar a um aplicativo nativo. Com o uso do *framework* Ionic foi desenvolvida uma aplicação padronizada e semelhante ao de um aplicativo puramente nativo.

Alguns componentes fornecidos pelo *framework* Ionic necessitam do Angular e do Apache Cordova, por isso é preciso explorar como essas ferramentas funcionam. Isso será feito nas seções seguintes

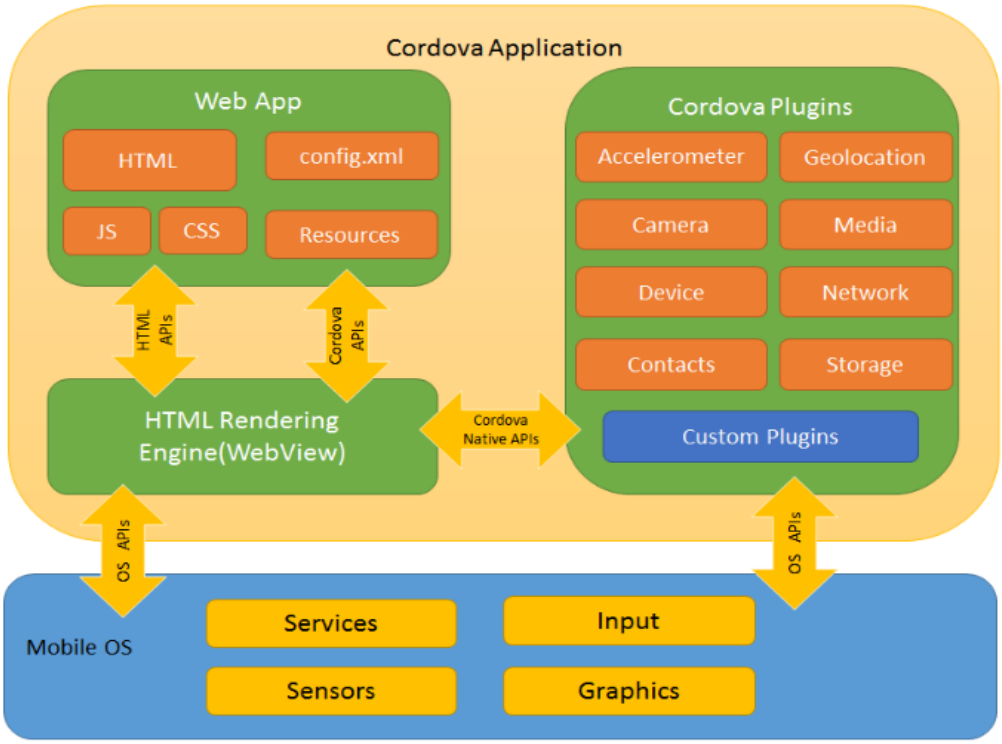
2.2.1. Apache Cordova

O *framework* Apache Cordova é uma estrutura de desenvolvimento para aplicativos móveis que possui licença gratuita e código aberto. “Ele permite que se utilize tecnologias padrão da Web (HTML5, CSS3 e *JavaScript*) para desenvolvimento em várias plataformas” (CORDOVA, 2018, nossa tradução), podendo utilizar o mesmo código para variadas plataformas, evitando assim a necessidade de compilação na linguagem de desenvolvimento nativo.

Para o acesso aos recursos nativos, faz-se o uso de “ligações de API compatíveis com os padrões para acessar os recursos de cada dispositivo, como sensores, dados, *status* da rede etc.” (CORDOVA, 2018, nossa tradução).

Pode-se perceber melhor na Figura 1 como o Apache Cordova *Framework* estabelece a comunicação entre as tecnologias Web até o sistema operacional do dispositivo móvel.

Figura 1 - Arquitetura da Aplicação Cordova



Fonte: Apache Cordova (2018)

Para gerar os aplicativos e assim testá-los, utilizou-se a interface de linha de comando do Cordova, o Cordova-CLI. Esta ferramenta possibilita a adição de *plug-ins* ao aplicativo e, além disso, abstrai funcionalidades de *scripts* e permite gerar binários para várias plataformas de uma vez só.

Os *plug-ins* são parte essencial para o projeto, pois “fornecem uma interface para que o Cordova e os componentes nativos se comuniquem entre si e vinculem-se às APIs de dispositivos padrão” (CORDOVA, 2018, nossa tradução), permitindo assim que possa ser invocado o código nativo a partir do *JavaScript*. Pode-se definir um *plug-in* Cordova como “um conjunto de arquivos que em união acrescentam ou

aperfeiçoam as capacidades de uma aplicação Cordova” (WARGO, 2013, nossa tradução).

Um aplicativo construído com o Apache Cordova é implementado como uma página Web, exemplificado no conjunto de Web App na Figura 1, a qual é empacotada em um *container* nativo, para que possa ser executada por meio do *WebView*.

A definição de WebView, pode se dar como “um componente de uma aplicação nativa que é utilizado para interpretar o conteúdo Web dentro de uma janela de um aplicativo” (WARGO, 2013, nossa tradução).

Deste modo, com todos os benefícios apresentados, o Apache Cordova se mostra como um facilitador no desenvolvimento de aplicativos para plataformas móveis, tendo em vista que este permite o acesso aos recursos nativos e também pode gerar um aplicativo para variadas plataformas a partir de um único código-fonte.

2.2.2. Angular

O Angular é um *framework* de licença gratuita e de código aberto, destinado a criação de aplicações *Web*. Segundo Freeman (2014), este *framework* é mantido pelo Google e tem uma grande comunidade mundial, visto que “contêm alguns dos melhores pontos do desenvolvimento do lado servidor e utiliza-os para melhorar o HTML do navegador, tornando uma base que torna simples e fácil a criação de aplicações custosas” (FREEMAN, 2014, p.3, nossa tradução).

Com o Angular é possível utilizar o HTML como uma linguagem de marcação, e ampliá-lo “adicionando novos elementos, atributos, classes…” (FREEMAN, 2014, p.19, nossa tradução).

As aplicações em Angular seguem o padrão estrutural de *Model-View-Controller* (MVC), o que sugere a separação “dos dados (*model*), a lógica que opera nos dados (*controller*) e os elementos HTML usados para mostrar os dados (*view*)“ (FREEMAN, 2014, p.47, nossa tradução).

Um dos aspectos positivos do *framework* é a capacidade de manipular as requisições que são feitas para o servidor, automatizando várias tarefas que teriam

que ser pensadas pelo desenvolvedor sem a utilização da ferramenta. Por esta razão “o objetivo do Angular é trazer ferramentas e recursos que estão disponíveis somente no lado do servidor para o cliente Web” (FREEMAN, 2014, p.45, nossa tradução).

Assim, pela necessidade do Ionic de utilizar o Angular e também por todas as vantagens apresentadas, foi utilizada esta ferramenta para o desenvolvimento da aplicação.

2.3. FIREBASE

O Firebase (2018) é uma plataforma da Google disponível para a criação de aplicativos móveis que fornece ferramentas e uma infraestrutura adequada para a utilização do desenvolvedor. É possível utilizar alguns serviços sem pagar nada, no entanto, se a demanda de acesso aumentar, a organização oferece planos de acordo com a escala de uso dos serviços.

A plataforma oferece diversos serviços, mas os que foram adotados para este trabalho foram o de banco de dados (*Firebase Firestore*), o de autenticação (*Firebase Authentication*) e o de armazenamento (*Firebase Storage*).

O *Firebase Clound Firestore* permite que todos os usuários do aplicativo recebam as atualizações ao mesmo tempo, pois ele “é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados” (FIREBASE, 2018).

Outra funcionalidade interessante dessa plataforma é o *Firebase Authentication*, que automatiza o processo de conexão do usuário ao aplicativo e oferece diferentes modos de autenticação. “Ele oferece suporte à autenticação por meio de senhas, números de telefone e provedores de identidade federados como Google, Facebook, Twitter e muito mais.” (FIREBASE, 2018). Como complemento, foi usado o *Cloud Storage* para o armazenamento das mídias do usuário.

O Firebase oferece aos seus usuários um console que permite visualizar e manipular todos os dados adicionados por meio de aplicações, de modo a possibilitar ao desenvolvedor gerenciar todos os dados armazenados.

Este console comporta toda a estrutura do Firebase, além de fornecer gráficos relacionados ao uso dos recursos disponíveis, o que é de suma importância, tendo em vista que a gratuidade dos recursos é limitada.

Figura 2 - Gráfico de consumo de armazenamento

Fonte: Console Firebase

Por todos os atrativos que o Firebase oferta, ele foi escolhido para ser a camada de persistência da aplicação desenvolvida.

2.4. EXTREME PROGRAMMING (XP)

O desenvolvimento ágil de *software* (Agile *software development*) ou método ágil é uma expressão que define um conjunto de metodologias utilizadas no desenvolvimento de *software*. As metodologias que fazem parte do conceito de desenvolvimento ágil, tal como qualquer metodologia de *software*, providenciam uma estrutura conceitual para reger projetos de engenharia de *software* (COHEN, 2004).

Os métodos ágeis tentam minimizar o risco pelo desenvolvimento do *software* em curtos períodos, chamados de iteração, os quais gastam tipicamente menos de uma semana. Cada iteração é como um projeto de *software* em miniatura de seu próprio, e inclui todas as tarefas necessárias para implantar o mini incremento da nova funcionalidade: planejamento, análise de requisitos, projeto, codificação, teste e documentação (COHEN, 2004).

Enquanto em um processo convencional, cada iteração não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades, um projeto de *software* ágil busca a capacidade de implantar uma nova versão do *software* ao fim de cada iteração, etapa a qual a equipe responsável reavalia as prioridades do projeto (BECK, 1999).

Métodos ágeis enfatizam comunicações em tempo real, preferencialmente frente a frente, a documentos escritos. Combinado com a comunicação face a face, métodos ágeis produzem pouca documentação em comparação a outros métodos, sendo este um dos pontos que podem ser considerados negativos. É recomendada a produção de documentação que realmente será útil (COHEN, 2004).

A XP é uma metodologia de desenvolvimento ágil que permite criar projetos de *softwares* em um menor período de tempo, e tem como objetivo o desenvolvimento de sistemas com maior qualidade (PRESNER, JUNIOR, 2014). Segundo Teles (2014), a XP possuí alguns valores que definem a estrutura da metodologia.

2.4.1. Simplicidade

Segundo Teles (2014), a simplicidade consiste em aplicar todo o esforço somente no resultado esperado por cada funcionalidade desenvolvida, seja ela constatada no início do projeto ou durante o seu desenvolvimento, de forma que a mesma atenda às necessidades esperadas pelo cliente, limitando a equipe de desenvolvimento a não ultrapassar o objetivo do projeto.

É comum constatar funcionalidades com resultados incrementados quando comparados com os resultados definidos no pré-projeto, em que o cliente expressa suas necessidades. Os incrementos não apontados pelo cliente são considerados excessivos e desnecessários, visto que tal incremento pode futuramente ser utilizado pelo cliente ou nunca ser utilizado, resultando em um trabalho sem utilidade (TELES, 2014).

2.4.2. Coragem

A aplicabilidade da coragem no desenvolvimento de um sistema consiste em utilizar de procedimentos que por muitas vezes são questionados por especialistas da área, procedimentos que visam a agilidade e eficiência de uma equipe de desenvolvimento (TELES, 2014).

Frente aos avanços nas áreas de desenvolvimento de interfaces gráficas, e levando em consideração que existem à disposição ferramentas que fornecem inúmeras informações aos usuários finais, desenvolvendo uma *interface* de forma que sua aparência seja simples, pode exigir coragem por parte dos desenvolvedores (PINHEIRO, 2016).

Outro fato que requer coragem é a questão de abrir mão de documentações excessivas. Pode-se dizer que este ponto se torna essencial quando o objetivo do projeto engloba pouco tempo de desenvolvimento. Muitos documentos gerados no pré-projeto, são utilizados para levar ao conhecimento de toda a equipe as necessidades do cliente, porém um dos valores da XP é a comunicação entre os membros da equipe, que por sua vez prioriza que as modificações e adições do projeto sejam comunicadas ao cliente e a todos os membros da equipe de forma presencial (técnica descrita no tópico comunicação).

Tais documentos são deixados de lado, unificados em apenas documentos relevantes ao desenvolvimento do projeto (TELES, 2014). Para que tais documentos não sejam utilizados, é necessário que a equipe esteja comprometida em revisar toda nova funcionalidade desenvolvida, aplicando todos os testes necessários para que o sistema tenha maior qualidade.

2.4.3. Comunicação

Segundo Teles (2014), a metodologia XP relaciona o valor da comunicação com a forma de relatar o andamento do projeto ao cliente, canalizando a informação de forma mais eficaz do que a comunicação escrita por meio de documentações extensas.

Conforme a metodologia XP, a comunicação face a face define de forma mais clara as ideias em relação ao projeto, permitindo maior percepção sobre os

detalhes de cada funcionalidade e diminuindo a distância entre os membros da equipe de desenvolvimento. “A forma de se transmitir uma ideia exerce uma grande influência na compreensão correta da mesma” (TELES, 2014).

Quando duas pessoas estão frente a frente, alguns elementos podem facilitar o entendimento das informações transmitidas de ambas as partes. Esses elementos podem ser resumidos em expressões faciais, gestos, postura, palavras verbalizadas e tom de voz. Uma comunicação a distância elimina grande parte desses elementos, por exemplo um telefonema dificulta o entendimento das informações transmitidas (TELES, 2014).

2.4.4. Feedback

Umas das atividades mais difíceis no pré-projeto é identificar as necessidades do usuário em relação ao projeto final (TELES, 2014). A XP prioriza pequenos ciclos de desenvolvimento, que envolvem em cada ciclo o *feedback* ao cliente, para que ele possa evidenciar sua necessidade atendida por uma funcionalidade do *software*. É importante que cada ciclo seja curto, de forma que o cliente possa obter os resultados esperados por cada funcionalidade.

É possível também levar ao conhecimento do usuário as dificuldades técnicas que uma funcionalidade pode apresentar, podendo, por meio dos *feedback’s*, discutir novas possiblidades de alcançar o objetivo proposto no pré-projeto. As eventuais falhas detectadas pelo cliente nesta fase do projeto, possibilitam a equipe de desenvolvimento aplicar menores esforços em sua correção, devido ao escopo reduzido de cada ciclo, tornando as correções mais baratas (TELLES, 2014).

3. DESENVOLVIMENTO

Este capítulo apresenta o aplicativo desenvolvido, atendendo a proposta inicial do trabalho, e encontra-se dividido em seis seções. A seção 3.1 apresenta a modelagem do aplicativo desenvolvido. A seção 3.2 descreve os recursos utilizados para o armazenamento dos dados. A seção 3.3 discorre sobre as principais funcionalidades desenvolvidas. A seção 3.4 apresenta os eventos disponíveis e sua forma de gerenciamento.

3.1. ESTRUTURA

A seguir é apresentada estrutura do aplicativo, a forma com que as funcionalidades são apresentadas aos usuários, e como essas funcionalidades serão acessadas pelos diferentes tipos de usuários disponíveis no aplicativo.

3.1.1. Usuários

O aplicativo proposto neste trabalho tem como função gerenciar eventos criados por várias igrejas que fazem parte de uma organização religiosa. Cada igreja é responsável por adicionar os eventos no aplicativo, para que outros usuários, aqui chamados de “usuário comum” possam ter acesso aos eventos. Porém, os usuários comuns não possuem acesso para adicionar eventos. Tanto para a igreja como para o usuário comum, a forma de acessar o aplicativo é por meio de uma conta que é criada no primeiro acesso. A forma com que a igreja e o usuário comum criam essa conta é a mesma, e são solicitadas as mesmas informações, porém, no ato da criação da conta o aplicativo permite ao usuário solicitar a funcionalidade de criação de eventos.

Todas as solicitações são listadas para o administrador do sistema, que é o responsável por coordenar a organização religiosa. A única maneira de se tornar um administrador é diretamente pelo console do Firebase, que é a base de dados do aplicativo, descrito no Capítulo 2, seção 2.3 Firebase. Para que o administrador possa analisar a solicitação, o aplicativo informa o nome do solicitante e a igreja a

qual ele pertence, dados obrigatórios no ato da criação da conta. A partir disso, o administrador avalia a aprovação da solicitação ou a sua recusa. Caso a solicitação seja aprovada e liberada pelo administrador, o usuário solicitante passa a ser um “gerenciador de eventos”.

Cada igreja possui vários departamentos, que por sua vez podem organizar eventos e que podem ser adicionados no aplicativo. Cada departamento possui um responsável, devido a isso, o aplicativo possibilita que mais de um usuário por igreja se torne um gerenciador, tendo em vista que o número de departamentos pode variar de uma igreja para outra.

A maneira com que um usuário pode criar uma conta no aplicativo e solicitar acesso de gerenciador de eventos está descrita com detalhes na subseção 3.3.1. Dessa forma, é possível constatar que o aplicativo possui três tipos de usuários, sendo eles:

- Administrador: O administrador possui acesso a todas as funcionalidades do aplicativo, e o que o diferencia dos outros tipos de usuários é a possibilidade de liberar acesso a outros usuários para criar eventos, onde tal acesso pode ser solicitado no ato da criação da conta;
- Gerenciador de eventos: O usuário gerenciador possui a funcionalidade principal de criar eventos a partir da sua conta. Como descrito anteriormente, só é possível se tornar um usuário gerenciador após a liberação do usuário administrador;
- Usuário comum: O usuário comum possui menos funcionalidades que os demais, podendo apenas criar uma conta que, o permite visualizar todos os eventos criados e utilizar as funcionalidades do evento.

3.1.2. Requisitos

A seguir são apresentados os requisitos funcionais e não funcionais que foram contemplados pelo aplicativo.

3.1.2.1. Funcionais e não funcionais

Nesta seção são apresentados os requisitos funcionais do aplicativo e as principais funcionalidades que são disponibilizadas para os usuários. A lista de requisitos funcionais é descrita a seguir:

a) O aplicativo deve possibilitar aos usuários a criação de uma conta para que todos possam acessar o aplicativo pela primeira vez;
b) O aplicativo deve permitir a autenticação dos usuários por meio de e-mail e senha, informados na criação da conta;
c) Para os usuários gerenciadores de evento, o aplicativo deve fornecer a possibilidade de criar eventos;
d) O aplicativo deve permitir ao usuário adicionar imagens para personalizar sua tela de perfil.

Os requisitos não funcionais são relacionados ao desempenho, confiabilidade, segurança e manutenção do aplicativo, são características do sistema que acontecem durante a sua execução, e, na maioria das vezes, tais comportamentos não são visíveis aos usuários. Os requisitos não funcionais do aplicativo são:

a) Manter o usuário autenticado, mesmo quando o aplicativo é desligado;
b) Atualização dos dados em todos os dispositivos conectados em tempo de execução, sem necessidade de atualizar a página;
c) Fácil navegação devido ao uso de componentes apropriados ao uso em dispositivos móveis;
d) Alto desempenho ao acesso dos dados por se tratar de um banco de dados NoSql;
e) Conexão automática do usuário após a criação da conta;
f) O sistema de criptografia dos dados de autenticação de forma a proteger o usuário.

3.1.3. Caso de Uso

Nesta seção são apresentados os relacionamentos entre os atores (usuários) e as funcionalidades do aplicativo. Esses relacionamentos são

representados por meio de um diagrama gráfico, permitindo assim visualizar a interação de cada usuário com o aplicativo.

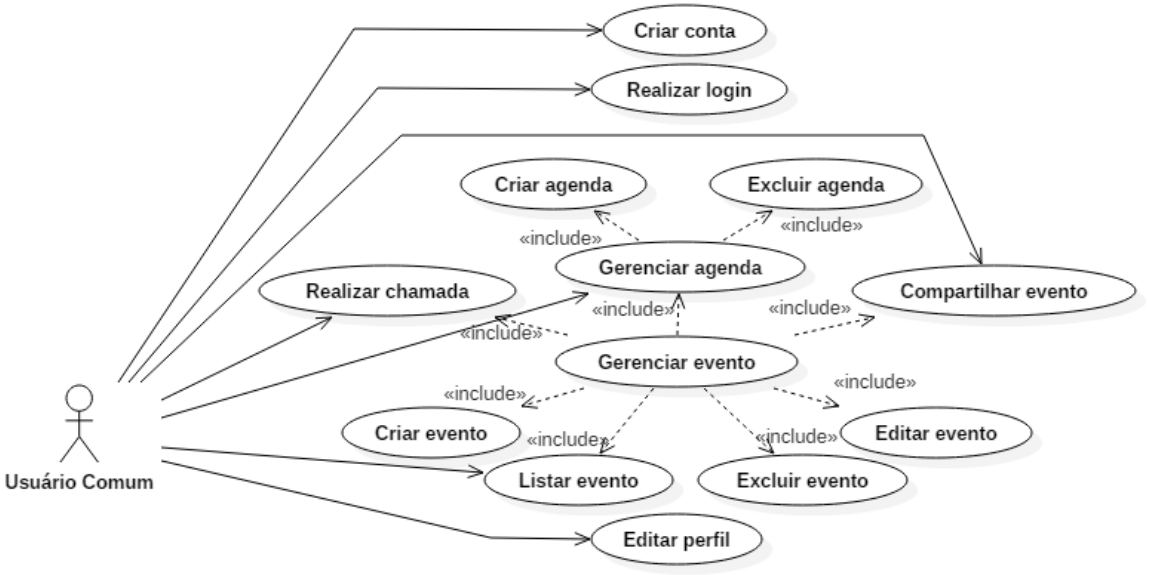
Para melhor visualização, o caso de uso foi dividido em duas partes. A primeira parte apresenta os atores administrador e gerenciador, a segunda parte apresenta o ator usuário comum. A Figura 3 representa o diagrama de caso de uso dos atores administrador e gerenciador

Figura 3 - Caso de Uso Atores Administrador e Gerenciador

Fonte: Autoria Própria

A Figura 4 representa o diagrama de caso de uso do ator usuário comum.

Figura 4 - Diagrama Caso de Uso Ator Usuário Comum



Fonte: Autoria Própria

Os diagramas representados nas Figuras 3 e 4 apresentam a interação dos usuários com o aplicativo. Na Figura 3 é possível destacar que o ator administrador possui acessos a algumas funcionalidades que os outros atores não possuem, assim como o ator gerenciador possui acesso a funcionalidades que o ator usuário comum não possui.

3.2. BASE DE DADOS

Nesta seção é apresentada a estrutura utilizada para o armazenamento dos dados do aplicativo, assim como suas características de acesso e manipulação.

A tecnologia de armazenamento de dados utilizada no aplicativo proposto neste projeto foi o *Firebase.* Como descrito na seção 2.3, o *Firebase* é um banco de dados *NoSql,* que tem como principal característica, proporcionar ao desenvolvedor uma maior capacidade de armazenamento e velocidade. Outro motivo para o uso do *Firebase*, é a facilidade em utilizar as suas várias funcionalidades com poucas linhas de código, pois as suas funções são bastante compactas e intuitivas.

O uso do *Firebase* em projetos Ionic é sugerido na própria documentação do *framework*, onde o desenvolvedor tem a sua disposição todos os passos necessários para fazer uso da tecnologia, que vão desde a sua instalação até as suas principais funções.

Como o *Firebase* é um banco de dados orientado a documentos, os principais objetos definidos no levantamento dos requisitos do sistema, são organizados em coleções, o que em uma comparação rápida com um banco de dados relacional, seriam chamados de entidades. Cada uma das coleções possibilita ao desenvolvedor criar subcoleções, ou coleções filhas. Desta forma pode-se concluir que o *Firebase* suporta um documento dentro de outro.

Devido a estrutura descrita acima, não se faz necessário apresentar diagramas de modelagem de dados, pois não é possível descrever relacionamentos ou cardinalidades. Sendo assim será apresentada a estrutura desenvolvida para o aplicativo descrito neste projeto, que possui ao total 6 coleções, organizadas em:

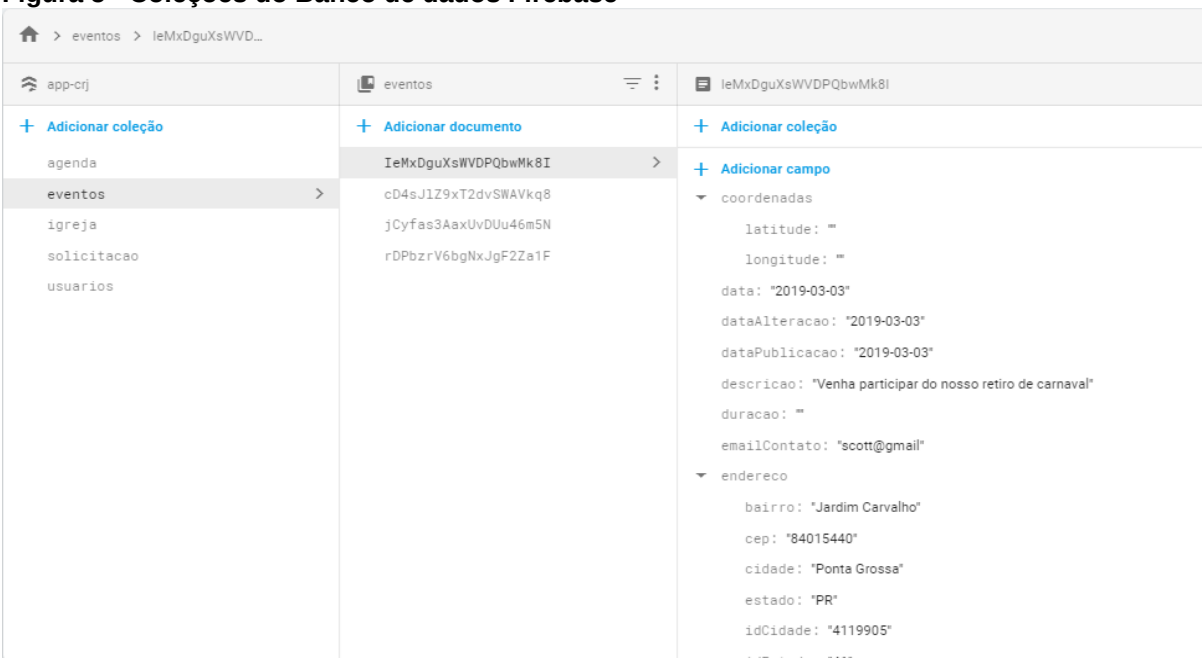
- Agenda;
- Eventos;
- Igreja;
- Solicitação;
- Usuários.

Os documentos armazenados nas coleções podem ser acessados a partir de um código identificador, que é definido automaticamente pelo *Firebase* quando um novo documento é adicionado ao banco de dados. Este código é composto de letras maiúsculas, minúsculas e números.

A única coleção que possui documentos aninhados é a coleção “Eventos”, que armazena os dados de latitude e longitude dentro do campo “coordenadas”, e armazena dentro do campo “endereço”, os dados da localização do evento, tais como: rua, bairro, cidade, estado, entre outros.

Na Figura 5 é apresentada a forma como estão organizadas as coleções do banco de dados do aplicativo, assim como são armazenados os dados aninhados dentro da coleção evento.

Figura 5 - Coleções do Banco de dados Firebase



Fonte: Firebase (2018)

Para incluir o *Firebase* ao projeto, foi utilizado um *provider,* que consiste em um conjunto de funcionalidades que são compartilhadas entre várias páginas.

O *Firebase* fornece um objeto que contém as chaves de API para que o projeto possa se conectar com o banco de dados, o qual precisa ser inserido diretamente no construtor da classe “FirebaseProvider”.

Figura 6 - Chaves de Conexão Firebase

```
constructor() {
  let config = {
    apiKey: "xxxxxxxxxxxxxxxxx-xxxxx",
    authDomain: "xxxx.xxxxxxxxxx.xxx",
    databaseURL: "xxxxx://xxx-xxx.xxxxxxxx.xxx",
    projectId: "xxx-xxx",
    storageBucket: "xxx-xxx.xxxxxxx.xxx",
    messagingSenderId: "xxxxxxxxxx"
  };
  firebase.initializeApp(config);
}
```

Fonte: Autoria Própria

3.3. FUNCIONALIDADES

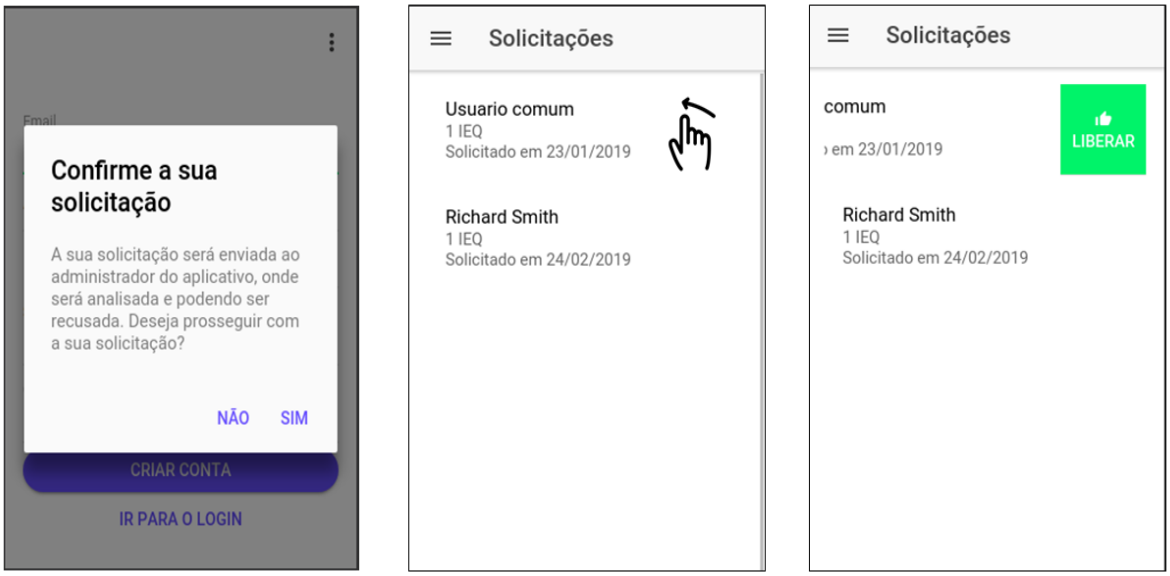
O aplicativo gerenciador de eventos possui diversas funcionalidades que possibilitam ao usuário interagir de forma intuitiva com todos os eventos divulgados pelas igrejas pertencentes a organização religiosa. A seguir são apresentadas de forma detalhada cada uma das funcionalidades contidas no aplicativo.

3.3.1. Criar Conta

Para criar uma conta no aplicativo, o usuário deve fornecer algumas informações solicitadas. Os dados informados pelo usuário são armazenados na coleção “usuários”, porém como informado na subseção 3.3.2, os dados de *e-mail* e senha são armazenados primeiramente na área de autenticação do *Firebase*.

Ao criar a conta, o usuário pode solicitar ao administrador do sistema a funcionalidade de “gerenciador”, que permite criar eventos. A solicitação, conforme ilustrado na Figura 7, é listada ao administrador para que o mesmo possa avaliá-la, para que então ele possa aceitá-la ou recusá-la, onde o administrador desliza o item da lista de solicitações para um lado ou para o outro, conforme a sua avaliação.

Figura 7 - Telas de solicitação



Fonte: Autoria Própria

A facilidade no uso de tal recurso, é fornecida por meio do componente de interface do Ionic*, ion-item-sliding.* O administrador interage com a funcionalidade pressionando em um botão, o qual altera um campo nomeado “gerenciador”, do tipo booleano da coleção “usuários”, que pode ser *true* (verdadeiro) no caso em que o administrador aceite a solicitação ou *false* (falso) no caso em que o administrador recuse a solicitação.

Para realizar a criação da conta, o aplicativo segue uma, salvando os dados de *e-mail* e senha, por meio do método “CreateUserWithEmailAndPassword”. Após isso o aplicativo armazena os dados informados pelo usuário no *Firebase* por meio do método “Add”, informando a coleção “usuários”.

Figura 8 - Salvar Dados da Conta

```
await this.firebase.db().collection('usuarios').add({

  uid: user.user.uid,
  nome: this.usuario.nome,
  telefone: this.usuario.telefone,
  fotoPerfil: this.usuario.fotoPerfil,
  fotoCapa: this.usuario.fotoCapa,
  bio: this.usuario.bio,
  email: this.usuario.email,
  igreja: this.igreja,
  administrador: false,
  gerenciador:false

}).then((data) => {
  this.idUsuario = data.id;
}).catch((error) => {

  this.notificacao.adicionarMensagemErro('Erro ao realizar cadastro!',
  this.notificacao.BOTTOM).mostrarNotificacoes();
});
```

Fonte: Autoria Própria

3.3.2. Autenticação

Para fornecer maior segurança, o aplicativo realiza a autenticação de todos os seus usuários, utilizando os recursos necessários para que cada usuário possa acessar o aplicativo com seus dados de autenticação.

Para fazer uso deste recurso, primeiramente é necessário escolher um método de *login*, que pode ser realizado diretamente no console do *Firebase*. Neste projeto foi utilizado o método “e-mail e senha”**.**

Para realizar a autenticação do usuário, foi necessário utilizar o método “signInWithEmailAndPassword” fornecido pelo objeto “Auth”, descrito no “provider” “FirebaseProvider” como demonstra a Figura 9.

Figura 9 - Método de Autenticação

```
await this.firebase.auth().signInWithEmailAndPassword(this.logar.email, this.logar.senha);
```

Fonte: Autoria Própria

Para que o usuário não precise realizar o “login” sempre que for usar o aplicativo, foi utilizado o método “OnAuthStateChanged”, que consiste em verificar se o usuário já está conectado, para que então o mesmo possa acessar o aplicativo diretamente. Para que o usuário possa se desconectar do sistema, o mesmo deverá clicar em “sair”, função localizada no menu do usuário. Para desconectar o usuário foi utilizado o método “signOut”.

Os métodos de autenticação são fornecidos na tela “login”, juntamente de outras funcionalidades, como Criar Conta e também “Recuperar Senha”. Na Figura 10 é possível identificar as funcionalidades disponibilizadas na tela de *login* do aplicativo.

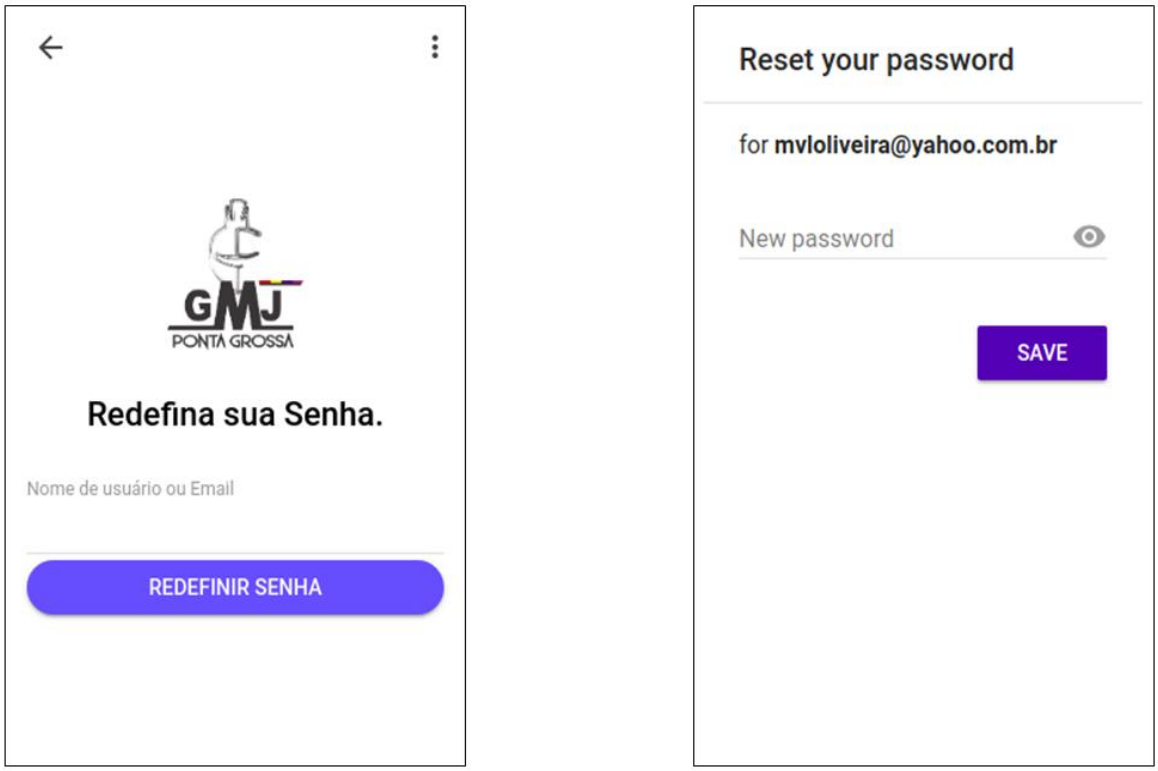
Figura 10 - Tela de login

Fonte: Autoria Própria

3.3.3. Redefinir Senha

Para que o usuário possa entrar no sistema quando não se lembre da senha cadastrada, é possível solicitar a redefinição da senha em uma função específica, onde foi utilizado o método *SendPasswordResetEmail.* Este método de redefinição de senha é disponibilizado pelo *Firebase*, o qual consiste em informar o *e-mail* cadastrado para que o *Firebase* envie um *link* de redirecionamento. Para acessar o *link* o usuário deve entrar em sua conta de *e-mail* e clicar no *link*. O usuário será então direcionado a um website para redefinir a sua senha de acesso. Após a confirmação da senha o usuário deve retornar ao aplicativo para realizar o seu acesso novamente, como mostra a Figura 11:

Figura 11 - Telas de redefinição de senha



Fonte: Autoria Própria

3.3.4. Conta

O aplicativo disponibiliza ao usuário a funcionalidade de gerenciamento da conta, onde ficam armazenadas as informações cadastrais, além de permitir inserir fotos de perfil e de capa, de modo a personalizar a área da sua conta.

As imagens podem ser inseridas a partir da biblioteca do dispositivo ou a partir da câmera. Tais funcionalidades são disponibilizadas pelo Ionic*,* o qual utiliza recursos nativos do dispositivo. O *plug-in* responsável por fornecer as funcionalidades de acesso a câmera ou a biblioteca é o *Camera.* Para acessar esses recursos é necessário fornecer algumas informações de modo a configurar os atributos das imagens que serão utilizadas no aplicativo. Esses atributos são adicionados em um objeto, que por sua vez é passado por parâmetro pelo método *getPicture.*

No Quadro 1 podem ser vistos os parâmetros utilizados pelo *plug-in* e entender o significado de cada um e sua respectiva aplicação.

Quadro 1 – Parâmetros da câmera

Propriedade do objeto	Tipo	Descrição
Quality	Numeral	Define a qualidade da imagem no intervalo de 0 a 100. O padrão é 50
Destination Type	Numeral	Define o formato do valor de retorno. O padrão é FILE_URI que retorna a imagem como *string* codificada em base64.
SourceType	Numeral	Define qual será a fonte da imagem, podendo ser da biblioteca ou da câmera.
allowEdit	Booleano	Permite a edição rápida da imagem antes de sua utilização (não utilizado).
encodingType	Numeral	Define a extensão da imagem. O padrão é *JPEG (Joint Photographic Experts Group)*
targetWidth	Numeral	Define a largura da imagem em *pixels.*
targetHeight	Numeral	Define a altura da imagem em *pixels.*
mediaType	Numeral	Define o tipo de mídia retornada. Os tipos podem ser fotos, vídeos ou os dois tipos.
correctOrientation	Booleano	Permite girar a imagem para corrigir a sua orientação em relação ao dispositivo (não utilizado).
saveToPhotoAlbum	Booleano	Permite salvar a foto na biblioteca do dispositivo.
cameraDirection	Numeral	Permite escolher entre a câmera frontal ou traseira do dispositivo.

Fonte: Ionic (2018)

As imagens adicionadas na conta do usuário são armazenadas no *Firebase Storage.* Como descrito anteriormente as imagens são fornecidas pelo *plug-in* em formato de texto, e para serem adicionadas ao *storage* elas são convertidas em

imagem. A partir do método *getDownload* é gerada uma url, a qual é armazenada na base de dados do *Firebase*, na coleção “usuários”.

Figura 12 - Armazenamento de imagens no Firebase

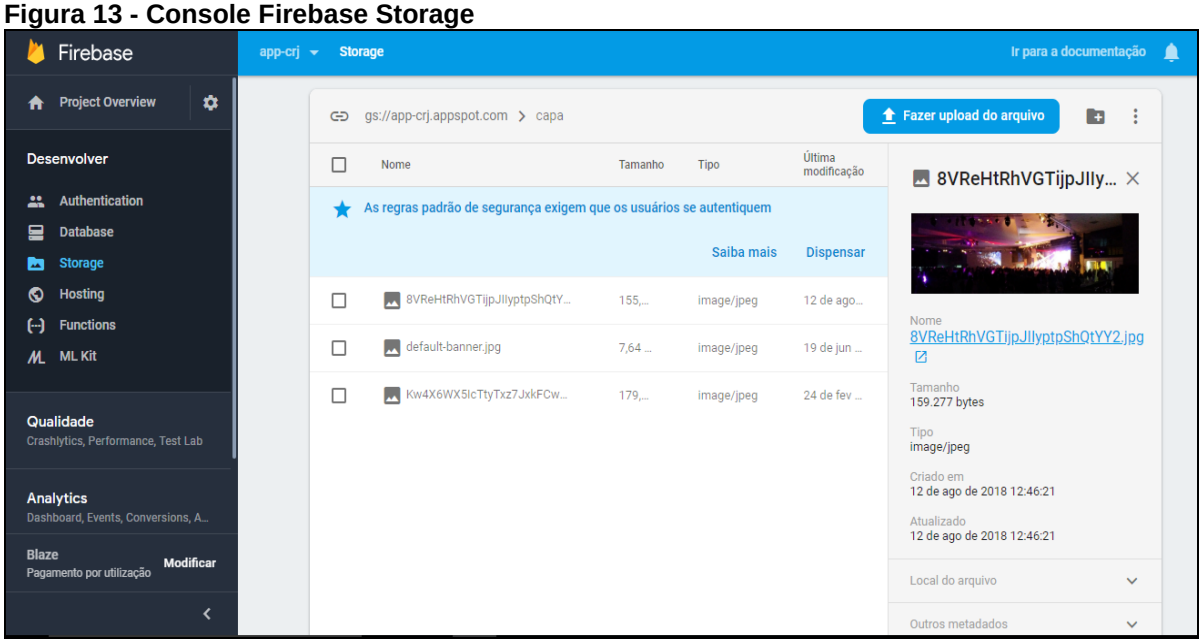
```
if (this.foto.capa.length > 0) {
  let storageRef = this.firebase.storage().ref('capa/' + this.usuario.get().uid + '.jpg');
  await storageRef.putString(this.foto.capa, 'base64', { contentType: 'image/jpeg' });

  storageRef.getDownloadURL().then((url) => {
    this.dados.fotoCapa = url;
  })
```

Fonte: Autoria Própria

No console do *Firebase* é possível acessar todas as imagens inseridas através do aplicativo. As imagens podem ser organizadas em pastas, de acordo com a necessidade do desenvolvedor, além de fornecer as informações de nome do arquivo, tamanho, tipo, data de criação e atualização. Também é possível fazer o *download* das imagens armazenadas e *upload* de novas imagens diretamente pelo console.

Figura 13 - Console Firebase Storage

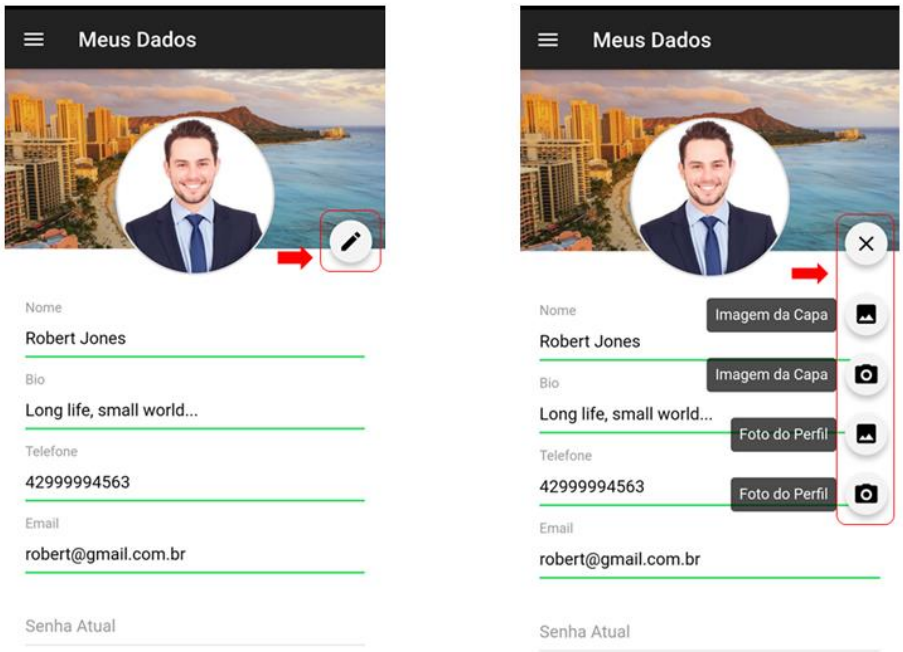


Fonte: Firebase (2018)

A possibilidade de adicionar imagens à conta, tem como finalidade customizar a área do usuário conforme o seu gosto. Além disso, a imagem de perfil do usuário é exibida em outras áreas do aplicativo, como por exemplo, no cabeçalho de cada evento, quando listados na tela principal do aplicativo, como também no menu principal do aplicativo.

Para que usuário possa alterar as imagens de perfil e de capa da sua conta, foi utilizado o componente *ion-fab-list*, que consiste em um *container* com vários botões, que aparecem ao usuário quando clicado em um botão principal. Para definir em que direção o *container* irá abrir, o componente possui os atributos *start, end, top* e *bottom.*

Figura 14 - Componente *ion-fab-list*



Fonte: Autoria Própria

Além da possibilidade de o usuário alterar as imagens adicionadas à sua conta, todos os outros dados também podem ser editados conforme sua necessidade. Quando o usuário se torna um gerenciador de eventos, além de customizar a área da sua conta, ele também pode criar eventos relacionados ao seu perfil, que serão listados na tela principal do sistema.

3.4. EVENTOS

Os eventos no aplicativo são criados pelos gerenciadores do sistema, os quais são responsáveis por divulgar as atividades de cada igreja pertencente a organização religiosa. Os eventos devem conter todas as informações solicitadas pelo aplicativo no ato de sua criação.

O gerenciador pode inserir uma imagem do evento, a qual pode conter outras informações que não são solicitadas pelo aplicativo, mas que podem ser úteis aos outros usuários. Uma das principais funcionalidades do aplicativo e listar todos os eventos organizados em um mapa. Essa funcionalidade está descrita na subseção 3.4. Tal funcionalidade possui em sua estrutura o uso de recursos nativos do dispositivo, assim também como as funcionalidades a seguir.

3.4.1. Agenda

De modo a organizar os eventos que sejam de interesse do usuário, o aplicativo disponibiliza duas funções de agendamento de eventos.

Primeiramente, o usuário tem a opção de armazenar os eventos em uma agenda interna do aplicativo, a qual armazena os dados do evento em uma coleção do *Firebase* chamada “agenda”. Na tela principal do aplicativo, o usuário pode acessar os eventos agendados buscando-os a partir da sua data de realização, onde o usuário filtra os eventos por período.

Para realizar o armazenamento dos eventos, a agenda interna do aplicativo recebe os *IDs* de usuário e evento. Essas informações são armazenadas na coleção e podem ser excluídas a qualquer momento pelo usuário.

O aplicativo também disponibiliza uma agenda que utiliza recursos nativos do dispositivo. Para fazer uso de tal funcionalidade, o *Ionic* utiliza um *plug-in* chamado *calendar,* no qual os eventos podem ser agendados e acessados pela agenda do dispositivo.

No momento em que o usuário acessa o evento, ele tem a opção de agendar o evento na agenda interna do aplicativo, o que também oferece a opção de adicionar o evento na agenda do dispositivo. O aplicativo armazena as informações do evento por parâmetros, por meio do método *createEvent*.

Figura 15 - Salvar evento na agenda do dispositivo

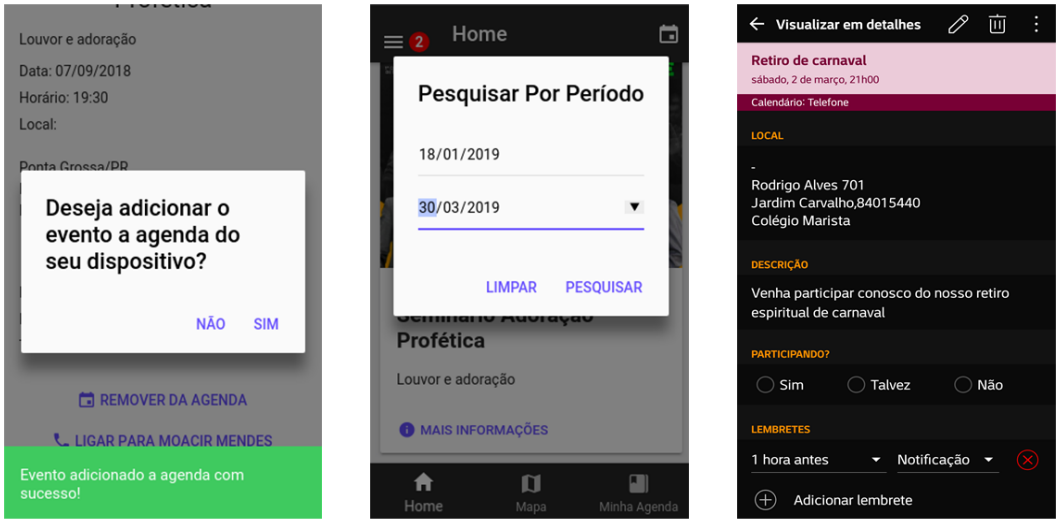
```
adicionarAgendaCelular() {
  this.alertCtrl.create({
    title: 'Deseja adicionar o evento a agenda do seu dispositivo?',
    buttons: [
      {
        text: 'Não',
        role: 'nao',
        handler: data => {}
      },
      {
        text: 'Sim',
        handler: data => {
          let dataInicial = new Date(this.evento.data);
          let dataFinal = new Date(this.evento.data);

          this.calendar.createEvent(this.evento.nome, this.evento.endereco.cidade + ' - '
          + this.evento.endereco.estado + '\n' + this.evento.endereco.rua + ' '
          + this.evento.endereco.numero + '\n' + this.evento.endereco.bairro + ','
          + this.evento.endereco.cep + '\n' + this.evento.endereco.referencia,
          this.evento.descricao, dataInicial, dataFinal);
        }
      }
    ]
  }).present();
}
```

Fonte: Autoria Própria

Para que o usuário possa receber notificações da agenda, a mesma precisa estar previamente configurada, assim como também a forma como o dispositivo irá disponibilizar a visualização do evento na agenda.

Figura 16 - Telas das agendas do aplicativo



Fonte: Autoria Própria

As duas funções possibilitam a remoção dos eventos agendados, excluindo os dados armazenados, sejam da memória interna do dispositivo com uso do recurso nativo, ou da base de dados do aplicativo.

3.4.2. Ligar para Contato

Outra função de grande utilidade disponível no aplicativo é a possibilidade de o usuário ligar para o responsável pelo evento. O número do telefone é informado no ato da criação do evento, junto do DDD (Discagem Direta a Distância).

Para fazer uso de tal funcionalidade, o Ionic utiliza outro recurso nativo do dispositivo, que é acionado a partir do *plug-in* “CallNumber”, utilizando um método de mesmo nome, onde o número do telefone é passado por parâmetro. A partir disso, o recurso nativo se responsabiliza por toda a funcionalidade da chamada telefônica.

3.4.3. Contato WhatsApp

Outra forma de o usuário entrar em contato com o responsável pelo evento é através do aplicativo “WhatsApp”. O Ionic utiliza uma *URL* que passa como parâmetro o telefone cadastrado no evento.

Quando a URL é acessada, o dispositivo verifica se o aplicativo “WhatsApp” já está instalado no dispositivo, e o acessa para realizar a funcionalidade.

3.4.4. Compartilhar

O aplicativo também fornece a função de compartilhar os eventos com as redes sociais “WhatsApp” e “Twitter”, que tem como finalidade divulgar os eventos com outras pessoas que não possuam acesso ao aplicativo gerenciador de eventos.

O *plug-in* responsável por fornecer a funcionalidade é o *socialSharing,* o qual possui o método *canShareVia,* que recebe como parâmetros o nome da rede social e o conteúdo a ser compartilhado.

3.4.5. Mapa de Eventos

Uma das principais funcionalidades disponibilizadas pela aplicação é o mapa de eventos, que consiste em organizar os eventos registrados no mapa com base em sua localização, informada no ato da sua criação. Para o desenvolvimento de tal funcionalidade foi utilizado o SDK (*Software Development Kit*) nativo (Android ou IOS).

Ao utilizar a API (*Application Programming Interface*) nativa, o grande ganho é o fato de que muitas informações referentes ao mapa já estão armazenadas no dispositivo, necessitando apenas atualizar as novas informações a serem solicitadas. Desta forma, é possível ter ganhos em termos de desempenho, devido ao menor número de requisições ao servidor do *Google* (que fornece as informações do mapa) em relação ao SDK *JavaScript*.

A ideia principal em utilizar o recurso na aplicação é de listar todos os eventos no mapa, de forma a direcionar o usuário, em relação a localização de cada evento, possibilitando-o executar funções disponibilizadas com o *Google Maps*, como por exemplo definir uma trajetória a partir de sua localização.

A cada novo evento criado, é necessário que o usuário informe o endereço completo (dados obrigatórios) com as demais informações necessárias ao cadastro de um novo evento. A lista de cidades e estados são fornecidos por uma API de localidades e as demais informações são descritas pelo usuário.

Os dados de endereço informados são repassados em formato de texto para uma API responsável por converter o endereço para coordenadas geográficas.

3.4.5.1. NativeGeocoder

Quando um evento é criado, todas as informações relacionadas ao endereço são concatenadas em um único texto, no qual cada informação é separada por um espaço. O texto concatenado é passado para o *NativeGeocoderForwardResult,* propriedade do *NativeGeocoder* responsável por converter o endereço em coordenadas geográficas.

Para carregar os dados de todos os eventos armazenados no sistema, é realizada uma busca na coleção eventos, ordenada por data. Assim como na aba eventos, o mapa de eventos também permite realizar filtros por data.

Para cada evento listado, um marcador é adicionado para que possa destacar a posição do evento no mapa. A API *GoogleMaps* possui uma propriedade chamada *AddMarkerSync,* que é responsável por adicionar todos os atributos necessários ao marcador. Resumidamente os atributos são: Título, ícone, animação e posição (contém as informações da posição geográfica, obtidas pelo endereço informado no ato da criação do evento).

Os marcadores disponibilizam eventos de interface, para que o usuário possa interagir de forma a obter informações adicionais inseridas ao marcador. Para o desenvolvimento do sistema proposto foi utilizado o evento de *Marker_Click.*

O evento *Click* foi utilizado de forma a disponibilizar ao usuário os detalhes sobre o evento relacionado ao marcador. O usuário ao clicar sobre o marcador de um evento, envia as informações sobre o mesmo por parâmetro para um modal, que é instanciado no momento do *click* do usuário.

Figura 17 - Listar eventos no mapa

```
carregarEventos() {
  this.unsubscribe();
  let ref = this.firebase.db().collection('eventos').orderBy('data', 'desc');
  this.unsubscribe = ref.onSnapshot((snapshot) => {
    this.eventos.forEach((marker) => {
      if(marker != null && marker != undefined)
        marker.remove();});
    this.eventos = [];
    snapshot.docs.forEach(doc => {
      let evento = { id: doc.id, ...doc.data()};
      this.criarMarcador(evento);
    });
  });
}
async criarMarcador(evento){
  if (evento.coordenadas.latitude && evento.coordenadas.longitude) {
    let marker = await this.map.addMarkerSync({
      title: evento.nome,
      icon: 'red',
      animation: 'DROP',
      position: {
        lat: parseFloat(evento.coordenadas.latitude),
        lng: parseFloat(evento.coordenadas.longitude)
      }
    });
    marker.on(GoogleMapsEvent.MARKER_CLICK).subscribe(() => {
      let eventoModal = this.modal.create("EventoPage", { evento: evento, abrirPerfil: true });
      eventoModal.present();
    });
    this.eventos.push(marker);
  }
}
```

Fonte: Autoria Própria

Com o auxílio da API *GeoLocation* e sua propriedade *GetCurrentPosition*, as coordenadas geográficas, formadas pelos dados da latitude e longitude do dispositivo são informadas ao mapa.

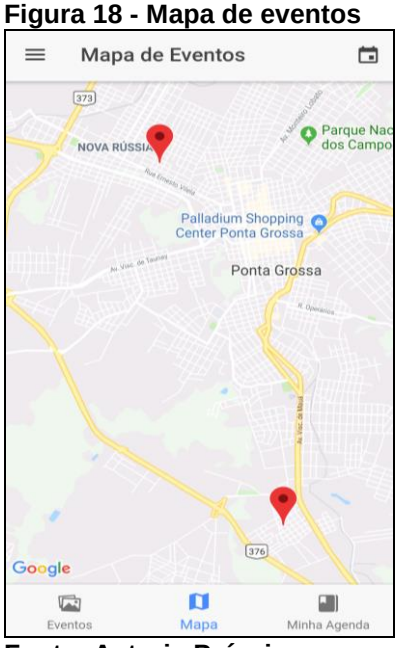
Após obter as coordenadas locais do dispositivo, é necessário iniciar o mapa. Para isso é necessário informar alguns atributos essenciais, sendo eles:

- *MapType:* Tipo do mapa relacionado ao tipo de imagem disponibilizado pela API *GoogleMaps.*
- *Zoom:* O *zoom* deve ser configurado conforme a necessidade do sistema e a sua área de cobertura.
- *Target:* Para esse atributo são informadas as coordenadas obtidas pela *API GeoLocation.*

A chamada das funções responsáveis por obter as coordenadas do usuário e iniciar o mapa devem ser realizadas no método construtor da classe “Mapa”. A disposição dos eventos no mapa e da localização do usuário são resultantes das funcionalidades descritas acima.

Além das funcionalidades já listadas, a *API* do GoogleMaps disponibiliza outras funcionalidades, como por exemplo traçar a trajetória da posição do usuário até um evento selecionado no mapa. Para ter acesso a essas funcionalidades não é necessário realizar implementações adicionais.

Figura 18 - Mapa de eventos



Fonte: Autoria Própria

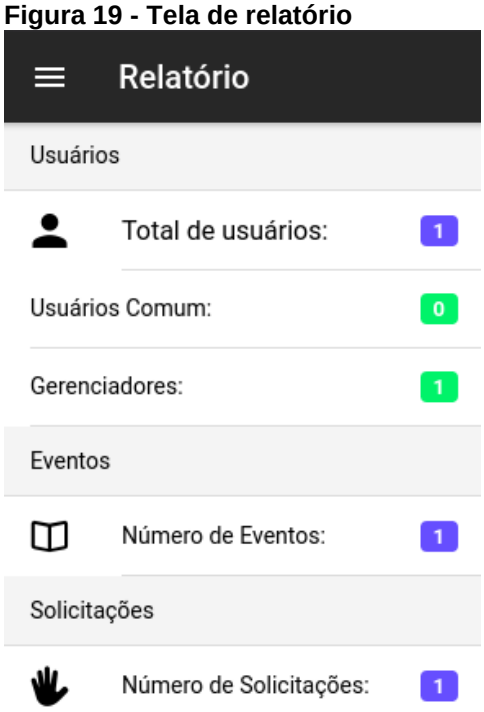
3.4.6. Relatório

O administrador do aplicativo tem a sua disposição um relatório resumido em apenas uma tela, que o informa dados compilados a partir da base de dados da aplicação. As informações disponíveis no relatório são:

- Total de usuários, que também se divide em usuários comuns e gerenciadores;
- Número de eventos;
- Número de solicitações.

Para realizar a coleta das informações que compõem o relatório, foram utilizados contadores, variáveis que são incrementadas pelo método *foreach,* que percorre os dados retornados em uma busca no banco de dados. A Figura 19 apresenta a tela do relatório descrita nesta seção.

Figura 19 - Tela de relatório



Fonte: Autoria Própria

3.5. BUILD

O *build* é a construção de um arquivo resultante da compilação de todas as classes de projeto. O Ionic fornece em sua documentação todos os passos para realizar o *build* do projeto, os comandos que necessitar ser executados em linha de comando diretamente pelo terminal do sistema operacional. Os dois comandos abaixo são utilizados para realizar a compilação do projeto, que no caso deste projeto, foi compilado para o sistema operacional Android:

- *ionic cordova run android* – para testar o aplicativo em modo de teste
- *ionic cordova build android –release –prod* – compilação em modo de produção

Após executar os comandos, o arquivo será armazenado no caminho abaixo, dentro do projeto:

- /platforms/android/app/build/outputs/apk/release/

4. CONSIDERAÇÕES FINAIS

O objetivo principal proposto neste trabalho foi alcançado com êxito. Foi desenvolvido um aplicativo gerenciador de eventos baseado nos requisitos necessários para que uma organização religiosa possa divulgar seus eventos.

O desenvolvimento do aplicativo foi simplificado devido ao uso da metodologia ágil XP, o que proporcionou a interação entre os desenvolvedores, o atendimento a prazos, sem abrir mão das principais documentações.

Com a estrutura simples do aplicativo, é fácil criar uma conta e ter acesso aos eventos criados por outros usuários, além de ter a possibilidade de poder criar novos eventos, após a liberação do administrador em resposta a uma solicitação prévia.

As tecnologias aplicadas no desenvolvimento do aplicativo fornecem as mais variadas funcionalidades contidas nos aplicativos mais utilizados na atualidade, como a possibilidade de compartilhar o conteúdo do aplicativo com outros aplicativos de comunicação, além de proporcionar segurança com a autenticação do *Firebase*, *design* inovador, proporcionado pelo Ionic, e o fácil uso dos recursos nativos, como o GPS por exemplo, por meio do Apache Cordova. O conteúdo das documentações do *Firebase* e do Ionic*,* que estão à disposição na Internet, de uma forma geral são bastante claras, o que auxiliou para o uso correto das diversas funcionalidades utilizadas no desenvolvimento do aplicativo.

A metodologia *Extreme Programming* nos auxiliou principalmente pela prática de desenvolvimento em par, a qual nos proporcionou realizar o desenvolvimento do projeto em duas, mas em um mesmo computador. Essa prática proporciona a resolução de problemas em tempo de desenvolvimento, não tendo necessidade da realização de documentações extensas que visam a comunicação de uma equipe de desenvolvimento. O projeto teve todo seu desenvolvimento realizado em par, sendo assim, não foi necessário fazer uso de ferramentas de controle de versão.

4.1. TRABALHOS FUTUROS

Sabendo das várias utilidades do aplicativo desenvolvido, é possível implementar muitas melhorias para um trabalho futuro.

A proposta é oferecer ao usuário um sistema de comunicação com outros usuários do aplicativo, onde possibilitaria uma maior interação, e assim maior interesse na participação dos eventos divulgados. Muitos aplicativos oferecem tal funcionalidade, porém o objetivo é oferecer ao usuário uma grande sala de bate-papo, somente com pessoas pertencentes da organização religiosa, partindo do ponto em que o aplicativo será utilizado somente por membros das igrejas da organização. Devido a isso, não há necessidade de criação de grupos, possibilitando o acesso a funcionalidade de forma simples e rápida.

REFERÊNCIAS

AGÊNCIA DE NOTÍCIAS DO IBGE. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2013-agencia-de-noticias/releases/14244-asi-censo-2010-numero-de-catolicos-cai-e-aumenta-o-de-evangelicos-espiritas-e-sem-religiao.html>. Acesso em 27 Mai. 2018.

AGÊNCIA DE NOTÍCIAS DO IBGE, 2012, Censo 2010 Disponível em:<https://agenciadenoticias.ibge.gov.br/agencia-noticias/2013-agencia-de noticias/releases/14244-asi-censo-2010-numero-de-catolicos-cai-e-aumenta-o-de-evangelicos-espiritas-e-sem-religiao.html>. Acesso em 18 Mai. 2018.

BBC. Os 10 assuntos mais discutidos no facebook em 2015. Disponível em: http://www.bbc.com/portuguese/noticias/2015/12/151211_10_assuntos_facebook_rm .Acesso em 20 Set. 2017.

BECK, K. **Extreme Programming Explained**: Embrace Change. Boston, MA: Addison-Wesley. 1999.

BRADLEY, Adam. Where does the Ionic Framework fit in? 2013. Disponível em: <http://ionicframework.com/blog/where-does-the-ionic-framework-fit-in>. Acesso em 18 Jun. 2018.

COHEN, D., LINDVALL, M., COSTA, P. An introduction to agile methods. In Advances in Computers. New York: **Elsevier Science**. 2004. p. 1-66.

CORDOVA. Overview. Disponível em: <http://cordova.apache.org/docs/en/4.0.0/guide_overview_index.md.html#Overview> . Acesso em 28 Mai. 2018.

CORDOVA: Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em 18 Jun. 2018.

SILVA, Cristiomar; NALINI, Lauro Eugênio Guimarães**. Religião e mídias sociais: A disseminação do discurso religioso no facebook**, Goiás: PUC, 2016.

ESTADÃO: Disponível em <https://link.estadao.com.br/noticias/empresas,facebook-chega-a-2-13-bilhoes-de-usuarios-em-todo-o-mundo,70002173062>. Acesso em 27 Mai. 2018.

FACEBOOK. Criar grupos no facebook. Disponível em: <https://www.facebook.com/help/167970719931213?helpref=about_content data de>. Acesso em 28 Mai. 2018.

FIREBASE: Disponível em: <https://firebase.google.com/docs/database/?hl=pt-br>. Acesso em 28 Mai. 2018.

FIREBASE: Disponível em: <https://firebase.google.com/docs/auth>. Acesso em 18 Jun. 2018.

FREEMAN, Adam. **Pro AngularJS**. [S.I.]: Apress, 2014.

G1: Disponível em: <http://g1.globo.com/sao-paulo/noticia/2015/06/veja-transexual-crucificada-e-outras-polemicas-com-simbolos-cristaos.html>. Acesso em 28 Mai. 2018.

G1: Disponível em: <http://g1.globo.com/sao-paulo/noticia/2011/09/mpf-denuncia-bispo-edir-macedo-sob-acusacao-de-lavagem-de-dinheiro.html>. Acesso em 18 Mai. 2018.

IONIC: Disponível em: >https://ionicframework.com/framework>. Acesso em 18 Jun. 2018.

LIMA, Camila. O imparcial. Confiram quais são as redes sociais mais usadas no Brasil. Disponível em: https://oimparcial.com.br/noticias/2017/08/confira-quais-sao-as-redes-sociais-mais-usadas-no-brasil/. Acesso em: 21 Set. 2017.

PINHEIRO, Allan Petterson da Silva. **UX Design Introduzido no Desenvolvimento de interfaces Gráficas**. Brasília – DF, 2016.

PRESNER, Diego Henrique; JUNIOR, Elson Luís dos Santos. **Estudo sobre metodologias ágeis de desenvolvimento aplicando a metodologia Extreme Programming em uma aplicação web**. Ponta Grossa, Paraná – Brasil. 2014.

SILVA, José Fábio Rogerio. **#1 Ionic Framework Essencial**. 2016.

SILVA, Maurício Samy. **Bootstrap 3.3.5**, São Paulo: novatec, 2015.

TELES, Vinícius Manhães. **Extreme Programming**, São Paulo: novatec, 2014.

TELES, André. **A revolução das mídias sociais,** São Paulo**:** Editora M.Books, 2017.

VAZ, Giovana Aparecida et.al. **Sistemas de Informações Gerenciais**: A importância da utilização do sistema Beta dentro dos processos decisórios e gerenciais da empresa Transportadora Alfa: Um estudo de caso. Anais doo 4 o. Encontro de Engenharia e tecnologia dos Campos Gerais. Ponta Grossa, Paraná – Brasil. 25 a 29 de agosto de 2008.

WARGO, John M. **Apache Cordova 3 Programming**. [S.I.]: Pearson Education, 2013.