UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

Findr: Uma aplicação para compartilhamento de informações usando o framework Node.js

Jonatham Petzold de Souza dos Santos

Orientador
Pedro Nuno de Souza Moura

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2019

Catalogação informatizada pelo(a) autor(a)

Santos, Jonatham Petzold de Souza dos
S237 Findr: Uma aplicação para compartilhamento de informações usando o framework Node.js / Jonatham Petzold de Souza dos Santos. -- Rio de Janeiro, 2019.
38

Orientador: Pedro Nuno de Souza Moura.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2019.

1. Compartilhamento de informações. 2. Geolocalização. 3. Aplicativo mobile. I. Moura, Pedro Nuno de Souza, orient. II. Título.

Compartilhamento de informações - Findr

Jonatham Petzold de Souza dos Santos

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

`

Pedro Nuno de Souza Moura (UNIRIO)

`

Sean Wolfgand Matsui Siqueira (UNIRIO)

`

Jefferson Elbert Simões (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

JANEIRO DE 2019

Agradecimentos

Aos meus pais Vilma Moreira Braga e Sérgio Augusto Soares Leite, por me darem uma nova chance na vida, por me incentivarem e ensinarem a lutar pelos meus objetivos, além de me amarem incondicionalmente. Nada disso seria possível sem eles.

Ao meu professor, orientador e amigo Pedro Nuno de Souza Moura, por ter me acompanhado, dentro e fora da universidade, sempre me incentivando e estando presente, além de lecionar a mim matérias chaves do curso. Por ter despertado em mim o gosto pela computação, e por incentivar ativamente a minha continuidade na vida acadêmica. Muito obrigado.

Ao meu amigo Leonardo Meirelles e sua excelentíssima mãe Christine da Silva, por cuidarem do meu psicológico, e por me apoiarem de todas as formas durante parte crucial da minha formação.

À minha professora tutora Geiza Maria Hamazaki da Silva, pelo apoio durante a universidade e puxões de orelha ao longo do curso.

Ao meu amigo Luiz Paulo Carvalho, por me incentivar no início do curso.

À todos os professores e amigos que participaram da minha formação, muito obrigado.

RESUMO

Com o crescimento do uso de dispositivos móveis, bem como a necessidade de mantermo-nos sempre conectados, surgiu o estímulo para desenvolver o Findr, um sistema de compartilhamento de informações. Atualmente, embora existam aplicações que proporcionem tal compartilhamento, não existem aplicações que permitam compartilhamento com usuários próximos desconhecidos. Este trabalho se propõe a desenvolver uma aplicação que possibilite o compartilhamento de informações com usuários próximos, selecionados de acordo com a categoria desejada.

Palavras-chave: compartilhamento, geolocalização, informação, aplicativo.

ABSTRACT

With the growth of the use of mobile devices, as well as the need to keep us always connected, the stimulus to develop Findr, an information sharing system, was emerged. Currently, although there are applications that provide such information sharing, there are no applications that allow sharing with unknown nearby users. This work proposes to develop an application that allows the sharing of information with nearby users, selected according to the desired category.

Keywords: sharing, geolocation, information, app.

1 Introdução 1
1.1 Motivação 1
1.2 Objetivos 2
1.3 Organização do texto 2
2 Contexto 3
2.1 A evolução da troca de informações. 3
2.2 Modelo existente de compartilhamento de informações 3
2.3 Aplicações similares 4
2.4 A Proposta do Findr 6
3 Especificação do Sistema 7
3.1 Metodologia Waterfall 7
3.2 Requisitos do Sistema 8
3.3 Processos do Sistema 9
3.4 Casos de Uso 11
3.5 Modelo Conceitual 16
4 Desenvolvimento da Aplicação 18
4.1 Introdução à arquitetura do sistema 18
4.2 A arquitetura MVC2 18
4.3 Web Server e protocolo HTTP 19
4.4 Uma RESTful API 20
4.5 O Ionic e os Frameworks Híbridos 23
4.6 O AngularJS 25
4.7 O Node.js 27
4.8 O MongoDB 28
5 O aplicativo Findr 30
5.1 Introdução ao Sistema 30
5.2 O método de geolocalização 30

5.3 Recursos e telas 32
6 Conclusões 36
6.1 Considerações Finais 36
6.2 Trabalhos Futuros 36
Referências Bibliográficas 38

Índice de Figuras

Figura 1 - Número de usuários de Internet no mundo de 2005 a 2017 1
Figura 2 - Interface do aplicativo “Happn”. 5
Figura 3 - Interface da aplicação “AirDrop”. 5
Figura 4 - Processos do modelo Cascata. 7
Figura 5 - Comparação entre os processos do sistema convencional e pós Findr. 10
Figura 6 - Diagrama de Casos de Uso do sistema. 11
Figura 7 - Modelo conceitual do banco utilizado na aplicação. 16
Figura 8 - O modelo do banco de dados utilizado na aplicação. 17
Figura 9 - As tecnologias do aplicativo Findr 18
Figura 10 - Exemplo de função "delete" disponível no Web Server na aplicação 20
Figura 11 - O funcionamento do controller da aplicação. 21
Figura 12 - O funcionamento dos “Providers” na aplicação. 22
Figura 13 - A estruturação do Ionic. 24
Figura 14 - Uma função do sistema utilizando AngularJS. 25
Figura 15 - O Html utilizando o AngularJS. 26
Figura 16 - A conexão com o MongoDB. 27
Figura 17 - A conexão com o banco a partir da página Web do MongoDB. 29
Figura 18 - Função da biblioteca do Ionic para a busca de coordenadas. 30
Figura 19 - Função de distância a partir de coordenadas. 31
Figura 20 - Tela de login da aplicação. 32
Figura 21 - Tela principal da aplicação. 33
Figura 22 - Tela de configuração da aplicação. 34
Figura 23 - Tela de perfil da aplicação. 35

Índice de Tabelas

Tabela 1 - Lista de requisitos funcionais do sistema. 8
Tabela 2 - Lista de requisitos não funcionais do sistema. 9
Tabela 3 - Atores do caso de uso 1 da aplicação. 12
Tabela 4 - Atores do caso de uso 2 da aplicação. 13
Tabela 5 - Atores do caso de uso 3 da aplicação. 14
Tabela 6 - Atores do caso de uso 4 da aplicação. 15

1 Introdução

1.1 Motivação

No contexto da crescente globalização que experienciamos hoje, se faz cada vez mais necessário o uso de tecnologias capazes de prover, de forma rápida e simples, a troca de informações. A Internet aparece como principal meio por onde as informações são tanto consumidas quanto fornecidas. O gráfico da Figura 1 a seguir mostra a quantidade de pessoas no mundo com acesso à Internet:

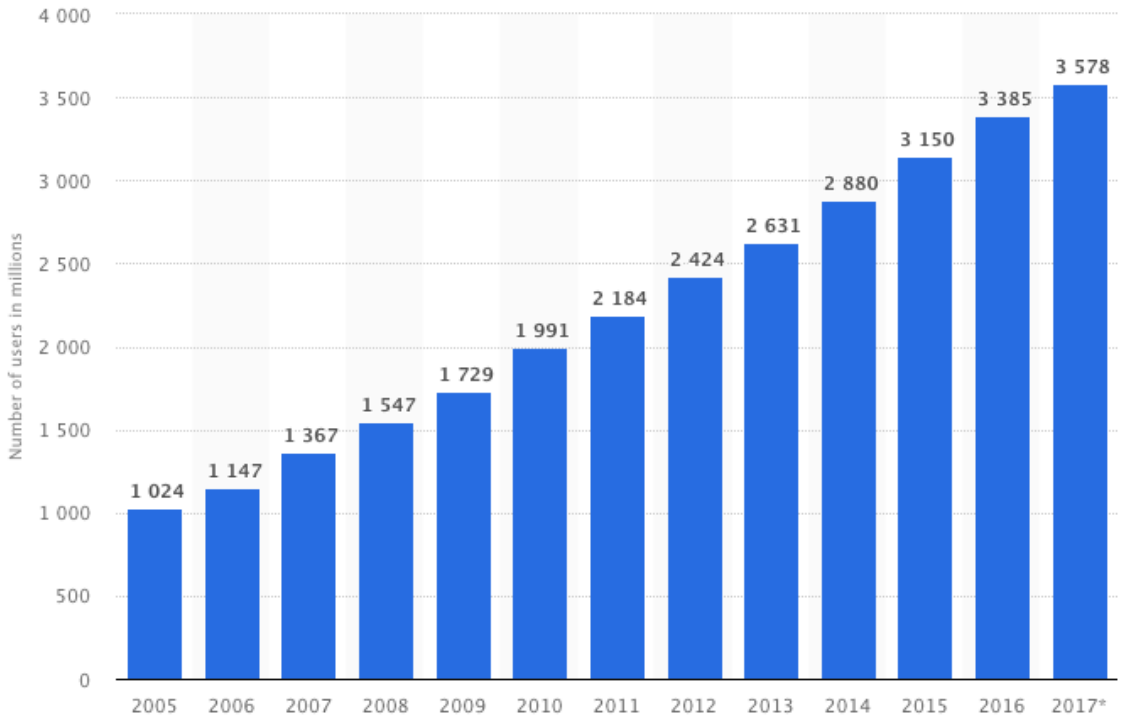

Figura 1 - Número de usuários de Internet no mundo de 2005 a 2017. Fonte: Statista

Nesse sentido, o celular deixou de ser um dispositivo apenas de ligações nos últimos anos e passou a ser um dispositivo de troca de informações e entretenimento, uma vez que a evolução dos *smartphones* (celulares com hardwares semelhantes aos de um computador) deu-se vertiginosamente. Nesse contexto de globalização e de um mundo interconectado, torna-se quase imprescindível seu uso.

Dessa forma, emergiu um amplo mercado a ser explorado nesta área, pois todos os

setores utilizam a tecnologia de alguma forma e, mesmo os que não utilizam como meio de trabalho, utilizam *smartphones* para auxílio pessoal. A busca por aplicativos que de alguma forma facilitem ou automatizem tarefas necessárias à rotina diária humana é sólida. Segundo Favretto (2012), em torno de 65% dos donos de *smartphones* utilizam diariamente aplicativos de localização, em que 15% aceitariam pagar por um serviço de boa qualidade.

O aplicativo Findr surgiu da ideia de organizar e agilizar a troca de informações básicas entre seus usuários, tais como número de telefone, perfil no Facebook, etc. Ao abrir o aplicativo, a ideia é que um usuário consiga ter acesso às informações que outros usuários disponibilizaram para os demais próximos.

Para tal, o sistema utiliza a biblioteca de geolocalização nativa dos celulares, que utilizam alguns métodos para localização pessoal, sendo o mais conhecido o GPS (Global Positioning System - Sistema de Posicionamento Global), o qual utiliza uma tecnologia de localização por satélite. O sinal emitido pelo celular é enviado e recebido por satélites diferentes, cada um com sua própria posição, que criam juntos um mapeamento triangular e definem a posição do usuário, mas sem grande precisão (Martins, 2009), o que será abordado nas limitações deste trabalho.

1.2 Objetivos

Este trabalho visa à criação de uma aplicação capaz de disponibilizar a troca de informações com usuários próximos, havendo local para configuração do mesmo, bem como a obtenção de informações de terceiros. Espera-se que os usuários sejam capazes de decidir qual informação desejam compartilhar e que também tenham acesso a informações que terceiros queiram compartilhar.

O estudo feito neste projeto também tem como objetivo exemplificar como ferramentas híbridas, isto é, que gerem sistemas para mais de uma plataforma, como o Ionic (será explicado a frente) podem ser empregadas na construção de aplicativos híbridos mais facilmente.

1.3 Organização do texto

Além da introdução, o trabalho será organizado da seguinte forma:
Capítulo II: Contexto atual: conteúdo referente ao cenário que se encontra a aplicação, bem como ressaltar a motivação do projeto.

Capítulo III: Especificação do sistema: análise prévia do sistema.
Capítulo IV: Arquitetura da aplicação: introdução às tecnologias utilizadas durante o projeto.
Capítulo V: O aplicativo Findr: conteúdo esclarecedor a respeito do funcionamento e funcionalidades do aplicativo.
Capítulo VI: Conclusões: conclusão final do aplicativo, dificuldades encontradas.

2 Contexto

2.1 A evolução da troca de informações.

O celular tornou-se o novo computador pessoal, graças à grande variedade de feitorias que disponibiliza. Quase tudo que fazemos no computador pessoal pode também ser feito no celular e, muitas vezes, o celular torna-se a opção principal por sua portabilidade, disponibilidade de Internet e geolocalização.

Esse é um dos fatores que, somado também à maior demanda de comunicação da globalização crescente, tem levado o número de pessoas com acesso à Internet a crescer vertiginosamente, tal como ressaltado na Figura 1. No Brasil, o número de pessoas que possuem acesso à dispositivos móveis aumentou em 147%, segundo o IBGE (2019).

Atualmente existem basicamente dois principais grandes sistemas operacionais de plataforma *mobile*: o iOS, disponibilizado pela empresa Apple, e o Android, disponibilizado pela Google (GOIS, 2017).

2.2 Modelo existente de compartilhamento de informações

No contexto atual, a necessidade de troca de informações entre indivíduos tem se tornado frequente, estimulando, portanto, a evolução dos meios existentes de comunicação, como redes sociais e outros. Estão disponíveis atualmente inúmeros métodos de compartilhamento de informações, sobretudo a partir da Internet, e da criação de redes sociais.

Os modelos de compartilhamento de informações atuais se baseiam, sobretudo, em redes sociais criadas com a difusão da Internet.

Apesar de ser possível o compartilhamento de informações, ainda não existe um meio

que permita a quem quiser acessar alguma informação específica e genérica sobre usuários fisicamente próximos. Não existem meios que permitam às pessoas compartilharem o que quiserem de forma que outras pessoas fisicamente próximas possam ter acesso ao que foi disponibilizado.

2.3 Aplicações similares

No mercado, existem algumas aplicações destinadas à troca de informações com usuários fisicamente próximos, tais como: a ferramenta nativa AirDrop, da empresa Apple (Apple, 2019), e o aplicativo Happn, da empresa FTW & Co Happn, (Happn, 2019).

Embora tais aplicativos tenham a mesma funcionalidade, a motivação deles é diferente. No caso do Happn, eles têm como função principal criar encontros entre pessoas com fins românticos, e não é permitido ao usuário que escolha as informações que deseja disponibilizar ao público.

No caso do AirDrop, o usuário pode enviar arquivos a outros usuários que estão fisicamente próximos, mas o usuário não tem a opção de procurar usuários dos quais deseja obter informação. Além disso, é possível compartilhar arquivos somente com outros dispositivos Apple, o que torna o aplicativo bastante restritivo. Nas Figuras 2 e 3, são exibidas imagens do funcionamento de tais aplicativos.

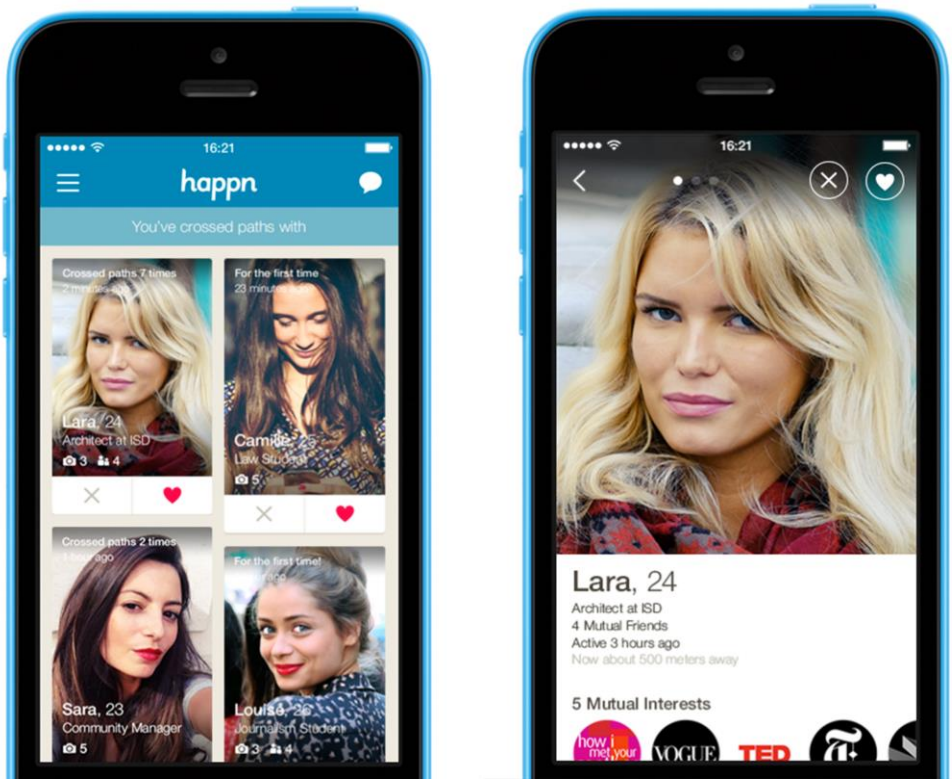

Figura 2 - Interface do aplicativo “Happn”. Fonte: www.happn.com

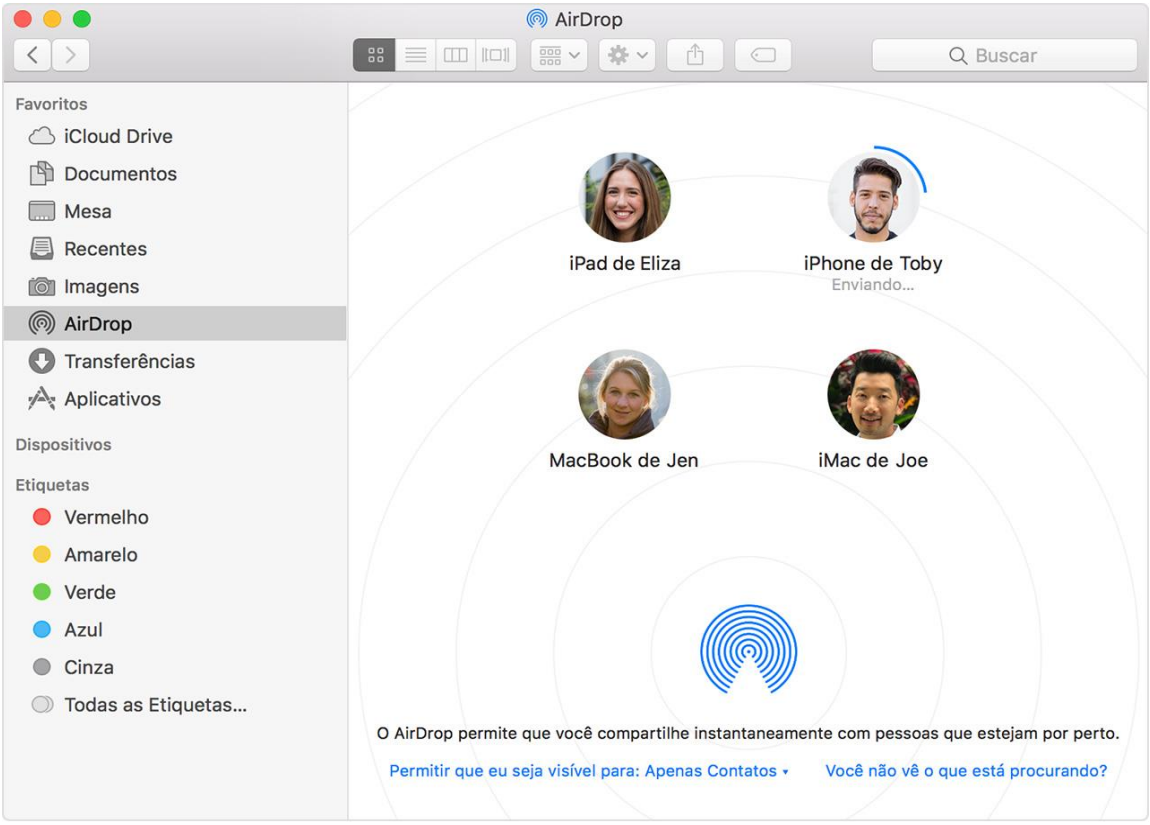

Figura 3 - Interface da aplicação “AirDrop”. Fonte: support.apple.com/pt-br/HT204144

2.4 A Proposta do Findr

Nesse contexto, o aplicativo Findr surge com a proposta de permitir que usuários busquem outros usuários fisicamente próximos com o objetivo de acessar as informações pessoais que esses disponibilizaram como textos e arquivos. Os usuários não mais serão passivos ao aplicativo. Serão, portanto, ativos, na medida em que poderão ativamente buscar informações sobre outros usuários.

A vantagem dele em relação aos demais disponíveis no mercado é a capacidade que tem de facilitar o acesso às informações de terceiros, uma vez que é necessário apenas que uma das partes queira obter a informação para a obtenção se concretizar.

A ideia é que duas pessoas que eventualmente tenham se encontrado fisicamente possam trocar informações, tais como o número de telefone ou uma mensagem, como um cartão de visitas. Para tal, basta que os usuários façam parte da rede social Facebook e comecem a trocar informações.

3 Especificação do Sistema

3.1 Metodologia Waterfall

Conhecido como a metodologia clássica de desenvolvimento de software, o modelo Waterfall de desenvolvimento tem esse nome pois foi baseado na forma sequencial de suas fases (Sommerville, 2011). Após o fim de uma fase, dá-se início à seguinte. Há um planejamento e criação de um escopo que antecede e prediz como será a execução de cada etapa.

Ainda segundo Sommerville (2011), o modelo cascata possui cinco fases principais: definição de requisitos; projeto de sistema e software; implementação e teste unitário; integração e teste de sistema; e operação e manutenção de software. O modelo é ilustrado na Figura 4.

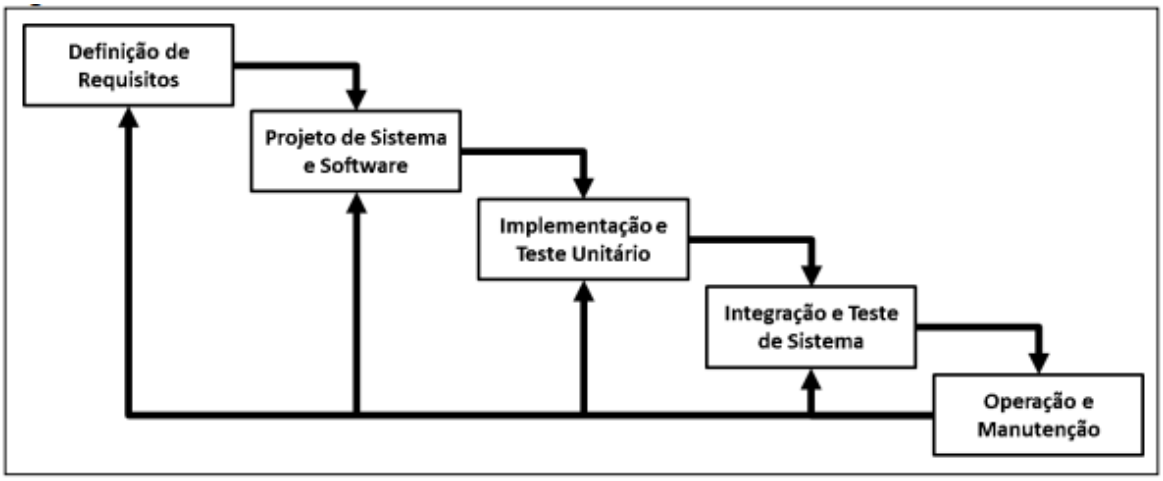

Figura 4 - Processos do modelo Cascata. Fonte: Sommerville, 2011

Neste sistema não foi utilizada a metodologia à risca, pois o sistema é um protótipo de sistema funcional, cuja principal função é exemplificar a arquitetura de software sugerida no trabalho. Porém, as fases de Definição de Requisitos, e Projeto de Sistema e Software foram abordadas no tópico 3.2 deste trabalho, pois referem-se à parte de análise da aplicação. A parte de implementação da aplicação é abordada ao longo da Seção 3.1.

3.2 Requisitos do Sistema

Os requisitos do sistema são bem especificados ao usarmos a metodologia de desenvolvimento cascata. Para esse sistema, são especificados os seguintes requisitos funcionais:

Requisito	**Nome**	**Descrição do Requisito**
RF0	Gerenciar controle de acesso	Na tela inicial, o aplicativo solicitará login ou cadastro do usuário. Na tela principal, o usuário poderá se deslogar.
RF1	Buscar usuários próximos	O sistema deve pertmitir a localização dos usuários em um raio de 100 metros.
RF2	Configurar informações a serem exibidas	O sistema deve permitir aos usuários escolherem quais informações serão exibidas aos demais usuários.
RF3	Visualizar informações de outros usuários	O sistema deve permitir a visualização de informações de outros usuários.
RF4	Redirecionar usuário	Dependendo do que o usuário selecionar nas informações que escolheu visualizar, ele poderá ser redirecionado a outro aplicativo.

Tabela 1 - Lista de requisitos funcionais do sistema.

E os seguintes requisitos não funcionais:

Requisito	**Nome**	**Descrição do Requisito**
RNF1	Uso do JavaScript	O aplicativo deverá basear-se em JavaScript (Node, Angular) como plataforma de desenvolvimento, sendo usada a IDE Android Studio.
RNF2	Armazenamento de informações	O aplicativo terá um armazenamento de informações em um banco de dados hospedado em um servidor.
RNF3	Uso do MongoDB	O aplicativo deverá utilizar o SGBD MongoDB.

Tabela 2 - Lista de requisitos não funcionais do sistema.

3.3 Processos do Sistema

Os processos do sistema ditam como o sistema vai reagir a cada estímulo externo e interno, quando disparados pelo usuário. Podem ser descritos de acordo com o diagrama de atividades exibidos na Figura 5. Na imagem, podemos citar os agentes “Informante”, responsável por disponibilizar seus dados pessoais aos usuários terceiros, “Solicitante”, responsável por acessar os dados do “Informante”, e o agente “Aplicativo”, presente apenas no segundo quadro, responsável por descrever o processo após o uso do aplicativo. A figura também faz uma comparação entre o processo convencional, que descreve o meio de troca de informações atual, com o processo obtido após a utilização do aplicativo, explicitando suas vantagens. É proveitoso observar que, com o sistema, o papel do informante na troca de informações é bem reduzido em relação ao convencional.

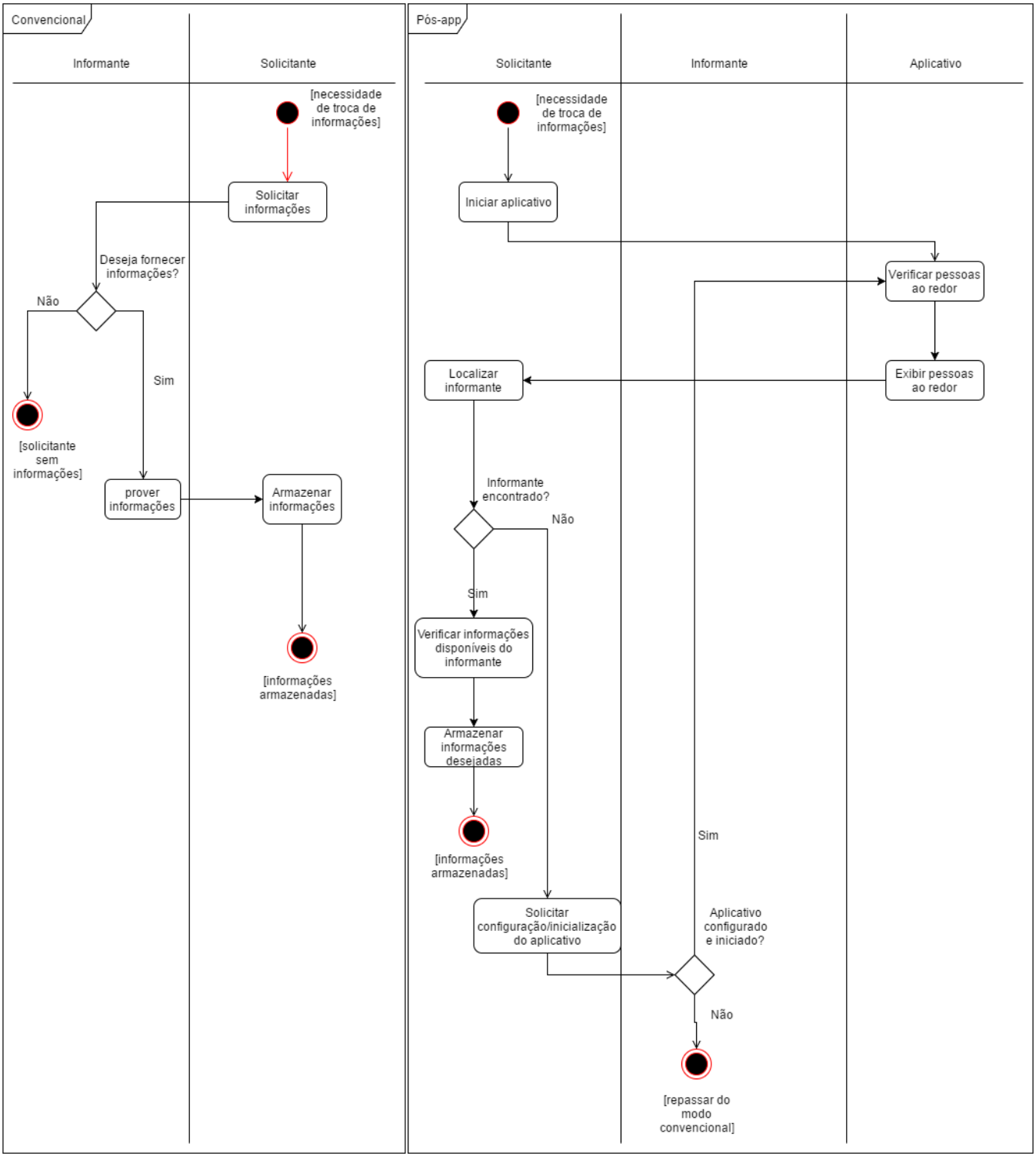

Figura 5 - Comparação entre os processos do sistema convencional e pós Findr.

3.4 Casos de Uso

Com base nos Requisitos Funcionais e no Diagrama de Atividades exibidos anteriormente, foi obtido o Diagrama de Casos de Uso, ilustrado na Figura 6 a seguir. Posteriormente, são apresentadas suas devidas descrições (somente os fluxos principais).

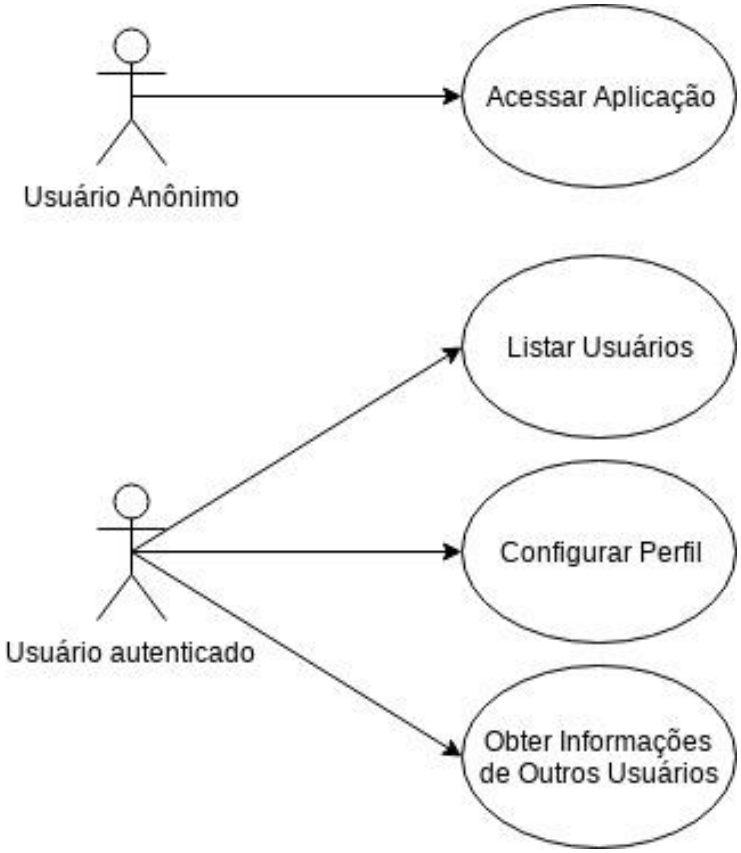

Figura 6 - Diagrama de Casos de Uso do sistema.

Caso de Uso 1 (CSU01) - Acessar aplicação

Descrição: Este caso de uso destina-se a descrever as funcionalidades que controlam o acesso à aplicação.

Ator	Responsabilidade
Usuário Anônimo	Responsável por iniciar a aplicação pela primeira vez.

Tabela 3 - Atores do caso de uso 1 da aplicação.

Pré-condições: Não se aplica.

Fluxo principal:
P1 Efetuar login.
P1.1 Esse fluxo se inicia quando o Usuário Anônimo decide iniciar o acesso à aplicação;
P1.2 O Usuário anônimo utiliza o botão de login na aplicação;
P1.3 O Sistema tem acesso aos dados do usuário disponíveis pelo login (nome e e-mail), e;
P1.4 O Usuário Anônimo se torna um Usuário Autenticado;
Fim do Fluxo Principal.

Caso de Uso 2 - Listar Usuários

Descrição: Este caso de uso destina-se a descrever as funcionalidades relacionadas à visualização do conteúdo exibido por terceiros por meio do App.

Ator	Responsabilidade
Usuário Autenticado	Responsável por iniciar a aplicação depois de feita a autenticação.

Tabela 4 - Atores do caso de uso 2 da aplicação

Pré-condições: O Usuário deve ter feito autenticação (CSU01)

Fluxo Principal:
P1 Listar usuários próximos.
P1.1 Esse fluxo se inicia quando o Usuário, após autenticado, decide iniciar a aplicação;
P1.2 O Usuário acessa a opção necessária para listar usuários;
P1.3 O sistema verifica se o serviço de localização está ligado;
P1.4 O sistema verifica se o serviço de localização está permitido;
P1.5 O Sistema lista usuários que tenham acessado o aplicativo em até quatro horas antes, em um raio de 100m.
Fim do Fluxo Principal.

Caso de Uso 3 - Configurar Perfil

Descrição: Este caso de uso destina-se a descrever as funcionalidades que controlam o que será exibido aos outros usuários por meio do App.

Ator	Responsabilidade
Usuário Autenticado	Responsável por iniciar a aplicação depois de feita a autenticação.

Tabela 5 - Atores do caso de uso 3 da aplicação.

Pré-condições: O Usuário deve ter feito autenticação (CSU01).

Fluxo principal:
P1 Exibir Informações.
P1.1 Esse fluxo se inicia quando o Usuário, após iniciar o acesso à aplicação, decide visualizar as suas informações que serão acessadas por terceiros;
P1.2 O Usuário acessa a opção necessária para acessar a tela de configuração;
P1.3 O Sistema apresenta as informações que o usuário já enviou ao Sistema;
Fim do Fluxo Principal.

Caso de Uso 4 - Obter Informações de Outros Usuários

Descrição: Este caso de uso destina-se a descrever as funcionalidades que controlam o que será exibido ao usuário principal quando o mesmo acessar o perfil de um outro usuário..

Ator	Responsabilidade
Usuário Autenticado	Responsável por iniciar a aplicação depois de feita a autenticação.

Tabela 6 - Atores do caso de uso 4 da aplicação.

Pré-condições: O Usuário deve ter feito autenticação (CSU01).

Fluxo principal:
P1 Exibir Informações de Outros Usuários.
P1.1 Esse fluxo se inicia quando o Usuário, após iniciar o acesso à aplicação, decide visualizar as suas informações que serão acessadas por outros usuários;
P1.2 O Usuário acessa a opção necessária para acessar a tela de configuração;
P1.3 O Sistema apresenta as informações que o usuário já enviou ao Sistema;
Fim do Fluxo Principal.

3.5 Modelo Conceitual

Os bancos de dados NoSQL como o MongoDB não suportam operações como “JOIN” dos bancos de dados sequenciais (MongoDB, 2018). Caso haja a necessidade de se referenciar algo, o mesmo deve ser referenciado ainda durante a inserção e modelo do banco. O modelo do banco é definido no próprio código, não sendo necessário que haja um modelo base que posteriormente se transformará em SQL (MongoDB, 2018). Tais modelos criam coleções (*collections*) no banco de dados, cada uma com os dados específicos do usuário. A Figura 8 exemplifica a implementação em código do modelo do banco. A Figura 7 mostra o Modelo conceitual do banco utilizado no sistema.

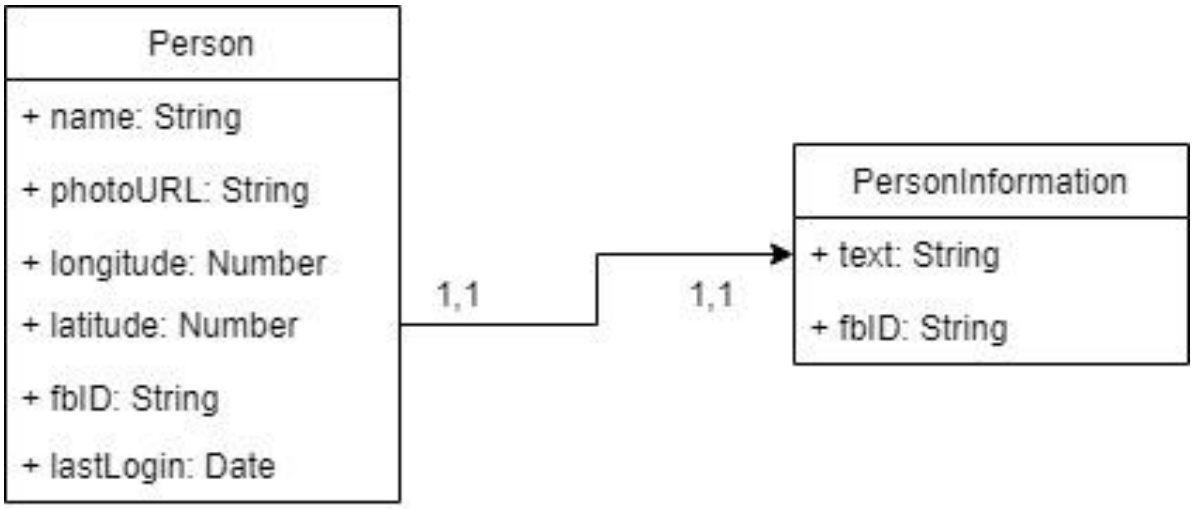

Figura 7 - Modelo conceitual do banco utilizado na aplicação.

```
var Person = mongoose.model('Person', {
    name: String,
    photoURL: String,
    longitude: Number,
    latitude: Number,
    fbID: String,
    lastLogin: Date,
    subtitle: String
}, 'people');

var PersonInformation = mongoose.model('PersonInformation', {
    subtitle: { type: String, default: null },
    text: { type: String, default: null },
    fbID: String
});
```

Figura 8 - O Modelo do banco de dados utilizado na aplicação.

Neste modelo, temos as coleções definidas utilizando a ferramenta explicada acima “Mongoose” (Mongoose, 2018), a qual nos permite criar modelos de coleções. Acima foram definidas as coleções “Person” e “PersonInformation”. Cada uma com seus atributos definidos e tipados ao lado. Ao inserirmos, por exemplo, um valor a um campo comum a ambos “fbID”, inserimos nos dois modelos ao mesmo tempo.

4 Desenvolvimento da Aplicação

4.1 Introdução à arquitetura do sistema

O sistema foi desenvolvido a partir de várias tecnologias, e utilizando a metodologia Cascata para desenvolvimento de software, que é descrita na Seção 3.1. Dentre as principais tecnologias, que serão explicadas da Seção 4.2 em diante, é possível citar: o Web Server e requisições HTTP, a arquitetura MVC2, o Ionic para o desenvolvimento do aplicativo; bem como todas que tangem ao lado cliente da aplicação: a ferramenta AngularJS, a linguagem de programação Node.js, e o banco de dados MongoDB. A Figura 9 ilustra a integração entre cada tecnologia.

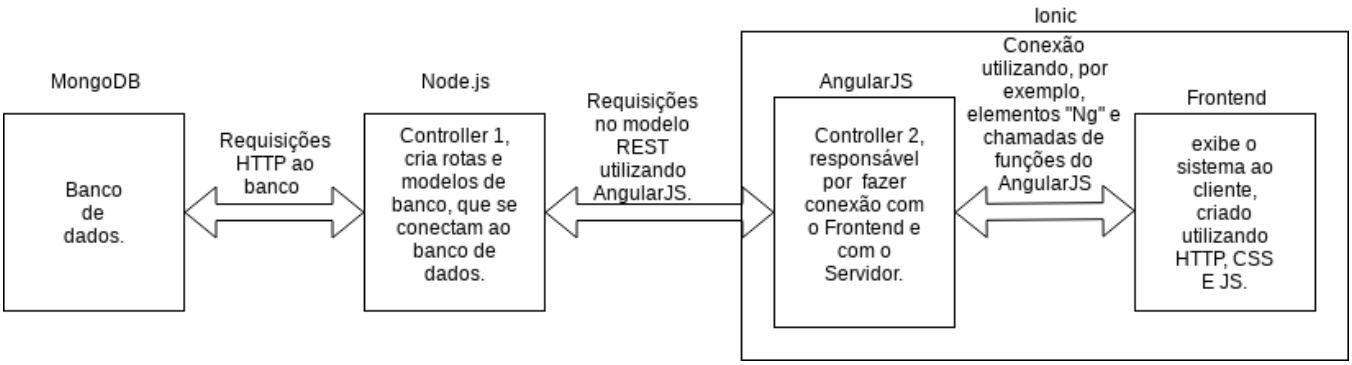

Figura 9 - As tecnologias do aplicativo Findr.

4.2 A arquitetura MVC2

A arquitetura MVC2 (Model-View-Controller 2) dita como o sistema se relaciona entre seu Modelo do banco de dados (Model), sua visualização no software (View), e a conexão entre ambos, chamada controlador (Controller). Surgiu após a configuração MVC clássica (Seshadri e Green, 2014) como uma forma de modularizar mais o sistema a ser desenvolvido.

Na arquitetura MVC clássica, o acesso dos usuários ao que é exibido na tela acontece diretamente através de múltiplos controladores da aplicação. No caso, o usuário acessa um controlador, que faz uma requisição direta ao servidor, e retorna todos os dados a serem exibidos na tela de uma vez, dificultando o trabalho com páginas dinâmicas (Seshadri e Green, 2014).

A arquitetura MVC2, não faz mais uso de múltiplos controladores de acesso único,

como era feito no modelo MVC clássico (Seshadri e Green, 2014). O que ocorre é a criação de duas formas de código que fazem o controle, mas somente havendo um controlador que segue o padrão de controladores do modelo MVC clássicos (Seshadri e Green, 2014). Há um controlador que faz a conexão com o banco na parte servidor da aplicação, cria e disponibiliza "rotas", que será explicada melhor na seção 4.8, mas que, basicamente, é uma URL criada pelo código para fazer modificações no banco de dados, que serão consumidas posteriormente pela parte View do sistema (Seshadri e Green, 2014), e um ou mais serviços (chamados "Providers") (Ionic Framework, 2018), responsáveis por consumir na parte cliente as rotas criadas na parte servidor da aplicação. As Figuras 11 e 12 mostram exemplos de controlador e provider utilizados no sistema, e que, respectivamente, criam e consomem uma rota.

4.3 Web Server e protocolo HTTP

O termo "Web Server" (servidor web) pode se referir tanto a um servidor físico que armazena as informações em hardware, quanto ao software que conecta ao servidor às aplicações, ou a ambos trabalhando juntos (Pereira, 2014).

No que se refere ao hardware, o servidor é um computador que armazena arquivos que compõem, sobretudo, os dados digitais das aplicações no geral. Ele está conectado à Internet, e pode ser acessado através do seu nome de domínio (DNS) (Pereira, 2014).

No que concerne ao software, o servidor web diz respeito aos componentes que gerenciam o acesso dos usuários aos arquivos que nele estão hospedados. Durante a execução deste trabalho, foi utilizado um servidor HTTP, que se baseia em requisições e respostas. No servidor geralmente ficam hospedados o banco de dados e *scripts* de acesso ao banco de dados (chamados controllers) (Pereira, 2016).

Durante uma consulta ao servidor, o usuário fará uma requisição ao servidor utilizando o protocolo HTTP. Quando a requisição (req) chegar ao servidor web correto, o servidor enviará como resposta (res) o documento requerido, também via HTTP (Pereira, 2016).

O protocolo HTTP, como dito anteriormente, permite ao usuário fazer operações de Criação (Create), Atualização (Update) utilizando o index "post", Leitura (Read) utilizando o index "get", e Deleção utilizando o index "delete".

No sistema, foi utilizado um servidor privado para hospedagem dos scripts (IBM's Bluemix), que acessaram o banco de dados disponibilizado pelo próprio MongoDB, que será explicado na Seção 3.8.

A Figura 10 a seguir ilustra um exemplo de como tais requisições foram utilizadas no sistema.

```
// delete a review
app.delete('/api/delete/:person_id', function (req, res) {

    Person.deleteOne({ _id: req.params.person_id }, (err) => {
        if (err) {
            res.send("deu erro ao deletar 1" + err);
        }
        else {
            res.send("nao deu erro algum ao deletar");
        }
    });
});
```

Figura 10 - Exemplo de função “delete” disponível no Web Server na aplicação

A figura acima mostra o funcionamento de uma função “delete” simples. A função recebe como requisição o id (identificador) a ser deletado, chamando a função “deleteOne” da biblioteca “Mongoose” (Mongoose, 2018) utilizada acima, e então é enviada de volta a resposta, que indica sucesso ou falha na requisição.

4.4 Uma RESTful API

REST é um acrônimo para “Representational State Transfer” (Pereira, 2016). Utiliza os padrões Web de arquitetura, descritos na Seção 4.3, e o protocolo HTTP (Pereira, 2016). Aplicações REST utilizam requisições HTTP para quatro operações nomeadas “CRUD”, que significam, respectivamente, “Create” (criar), “Read” (ler), “Update” (atualizar) e “Delete” (deletar) (Pereira, 2016). Tais operações são especificadas melhor na Seção 4.3, mas, basicamente, uma tecnologia que utiliza o padrão REST utiliza, sobretudo, métodos chamados “get”, “post”, “update”, e “delete” para a manutenção dessas operações básicas.

A grande vantagem do uso de aplicações REST se dá por a aplicação ser “Stateless” (sem estado). Ou seja, é uma aplicação cujas chamadas ao servidor são independentes e não podem ficar retidas a uma requisição (é menos suscetível a *bugs*). Além disso, por as aplicações serem sem estado, podem ser usadas em aplicações “Cloud” (nuvem). Os componentes sem

estado podem ser livremente modificados nas aplicações Cloud (Pereira, 2016).

```
// Models
var Person = mongoose.model('Person', {
    name: String,
    photoURL: String,
    longitude: Number,
    latitude: Number,
    fbID: String,
    lastLogin: Date
}, 'people');

var PersonInformation = mongoose.model('PersonInformation', {
    email: { type: String, default: null },
    phone: { type: String, default: null },
    text: { type: String, default: null },
    fbID: String
});

// Routes

// Get people
app.get('/api/getAllPeople', function (req, res) {

    //console.log(Person.collection.collectionName);
    // use mongoose to get all reviews in the database
    Person.find(function (err, people) {

        // if there is an error retrieving, send the error. nothing after res.send(err) will execute
        if (err)
            res.send(err)
        res.json(people); // return all reviews in JSON format
    });
});
```

Figura 11 - O funcionamento do controller da aplicação.

Na Figura 11 são gerados os modelos do banco, que serão explicados melhor no capítulo 4.8. Na figura 11 há o uso de uma rota (url criada pelo programa para fazer alterações no banco) disponibilizada no backend da aplicação. A função “get” da biblioteca “Express” (Express, 2018) utiliza a rota “getAllPeople” sem parâmetros. É procurado no modelo “Person” definido acima todos os registros de pessoas e, então, o script retorna os registros como resposta.

```
getPeople() {

  if (this.data) {
    return Promise.resolve(this.data);
  }

  return new Promise(resolve => {

    this.http.get('http://localhost:8082/api/getAllPeople')
      //this.http.get('https://findr3.mybluemix.net/api/getAllPeople')
      .map(res => res.json())
      .subscribe(data => {
        this.data = data;
        //console.log("REPOSTA AQUI " + JSON.stringify(this.data));
        resolve(this.data);
      });
  }).catch((err) => {
    console.log(err);
  });
}
```

Figura 12 - O funcionamento dos “Providers” na aplicação.

Na figura 12, é criada no Provider do sistema a função “getPeople”. A função utiliza um serviço HTTP presente nas próprias bibliotecas base do Ionic (Ionic Framework, 2018) que acessa a rota criada anteriormente (Figura 11), não envia parâmetros e espera a resposta do servidor. Assim que a resposta do servidor é obtida, a função termina e mostra os resultados. Caso haja erro, a função para e então imprime o erro.

4.5 O Ionic e os Frameworks Híbridos

O Ionic é um *framework* híbrido para desenvolvimento de aplicações para dispositivos móveis, que permite ao usuário a criação de aplicações híbridas (que podem ser convertidas para mais de uma plataforma) e de desenvolvimento facilitado (Ionic Framework, 2018). Foi utilizado no trabalho pois além de possibilitar a possível conversão do sistema para várias plataformas, possui uma biblioteca extensa.

O Ionic utiliza linguagens de programação genéricas baseadas em JavaScript, como Angular e TypeScript. Além disso, utiliza as linguagens clássicas de estruturação como HTML e CSS, para fazer suas páginas (Ionic Framework, 2018).

Utiliza o componente Cordova (Apache Cordova, 2018) para integração do sistema com recursos nativos dos dispositivos, o Angular, que é uma plataforma e framework para criação de interfaces de aplicação no lado cliente utilizando as linguagens de estruturação HTML (Prescott, 2015), CSS (Silva, 2012) e, sobretudo, a linguagem de programação TypeScript, que é uma máscara da linguagem de programação JavaScript com algumas modificações (Williamson, 2015). O Angular será explicado melhor posteriormente no tópico 3.1.6.

O Ionic tem sua própria estrutura de desenvolvimento, a qual permite separar todos os componentes que foram previamente explicados, e que serão utilizados pela aplicação como providers e telas (cada tela com seu próprio HTML, CSS e JavaScript) (Ionic Framework, 2018). A Figura 13 ilustra como o Ionic estrutura seus componentes.

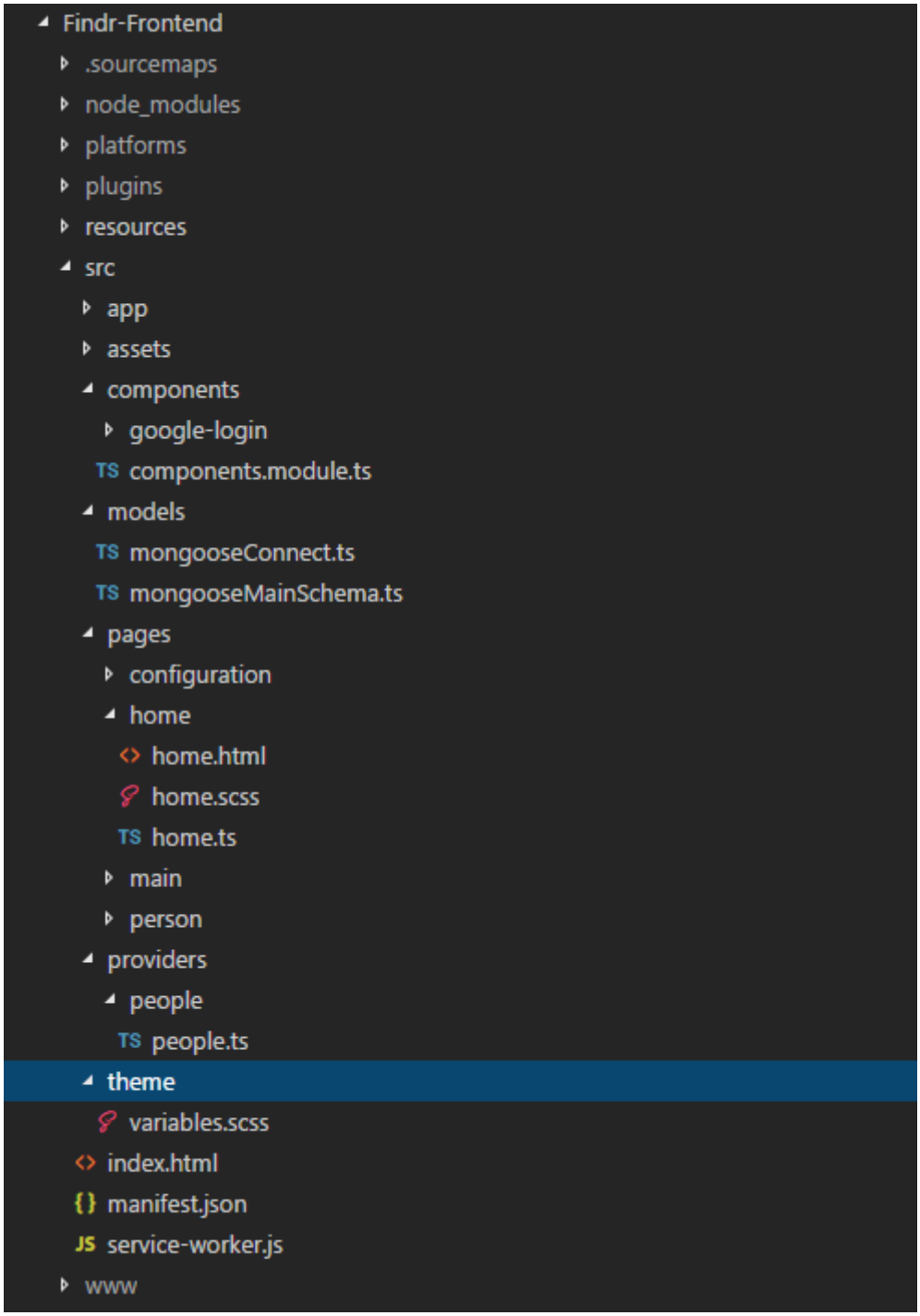

Figura 13 - A estruturação do Ionic (Ionic Framework, 2018).

Na figura 13, é possível visualizar que o Ionic divide os modelos usados (acesso ao banco) das páginas, e dos *providers*. Cada página possui sua pasta, com suas respectivas estruturações e códigos em HTML, SCSS (uma máscara da linguagem de estruturação CSS), e TypeScript (uma máscara de JavaScript) (Ionic Framework, 2018).

4.6 O AngularJS

O AngularJS, como dito acima, é um *framework* de código aberto que auxilia na execução de *single-page applications* (aplicações de página única). O AngularJS é responsável por todo o gerenciamento de dados que chegam e saem no lado cliente. Esse *framework* também auxilia o uso de dados dinâmicos (obtidos ou gerados em tempo de execução) no lado cliente (Williamson, 2015).

Durante o projeto, ele foi utilizado constantemente, visto que o Ionic (Seção 4.5) utiliza AngularJS em seu escopo. Com o AngularJS, foi possível criar *providers* (Seção 4.4) que se conectassem às rotas criadas no lado servidor, bem como criar as páginas da aplicação.

As Figura 14 ilustra exemplo de como o AngularJS utiliza dados em uma página dinâmica.

```
export class ConfigurationPage {
  subtitle: any;
  email: any;
  text: any;

  constructor(public navCtrl: NavController, public navParams: NavParams, public viewCtrl: ViewController) {
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad ConfigurationPage');
  }
  save(): void {

    let personInfo = {
      subtitle: this.subtitle,
      email: this.email,
      text: this.text
    };
    this.viewCtrl.dismiss(personInfo);
  }
}
```

Figura 14 - Uma função do sistema utilizando AngularJS.

Na figura 14 acima, a classe “ConfigurationPage” possui suas variáveis base “subtitle”, “email” e “text”, que serão lidas a partir do formulário HTML, como mostra a Figura 15 a seguir. Ao executar a função “save”, pressionando o botão “save” do HTML, as variáveis recebem o valor descrito no HTML e passam a ser exibidas com o novo valor.

```
<ion-content padding>
  <ion-list no-lines>
    <ion-item>
      <ion-item>
        <ion-label floating>E-mail</ion-label>
        <ion-input [(ngModel)]="email" type="text"></ion-input>
      </ion-item>
      <ion-label floating>Subtitle</ion-label>
      <ion-input [(ngModel)]="subtitle" type="text"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label floating>Text</ion-label>
      <ion-input [(ngModel)]="text" type="text"></ion-input>
    </ion-item>
  </ion-list>
  <button ion-button full color="secondary" (click)="save()">Save</button>
</ion-content>
```

Figura 15 - O HTML utilizando o AngularJS

A Figura 15 mostra a organização de cada componente da classe, listados na Figura 14. Os campos marcados com “ng” antes de cada palavra, destacam os componentes do AngularJS. Mais especificamente, os componentes marcados com “ng” exibem dinamicamente os dados da classe principal (Williamson, 2015), que foram descritos na Figura 14.

4.7 O Node.js

Node.js é um interpretador de código JavaScript, com ênfase na criação de código para o lado servidor. Ele facilita a comunicação com a maior parte dos frameworks de lado cliente *open-source* atuais (no caso com o AngularJS) pois também é baseado em JavaScript (Pereira, 2012). Foi utilizado no trabalho pois possui mecanismos que facilitam o trabalho de dados (sobretudo no formato JSON) provenientes do banco utilizado no trabalho (Express, 2018).

Ao longo do trabalho, foram criadas em Node.js rotas a partir de uma biblioteca para Node.js chamada “Express” (Express, 2018). O Node.js também ficou responsável por gerenciar a conexão entre as rotas criadas e suas conexões com o banco de dados.

A Figura 16 exibe um exemplo de conexão com o banco.

```
mongoose.connect('mongodb+srv://Findr:SENHAAQUI@cluster0-lm5sk.mongodb.net/test?retryWrites=true').catch(err => {
    console.log(err);
});

app.use(morgan('dev'));                                         // log every request to the console
app.use(bodyParser.urlencoded({ 'extended': 'true' }));            // parse application/x-www-form-urlencoded
app.use(bodyParser.json());                                     // parse application/json
app.use(bodyParser.json({ type: 'application/vnd.api+json' })); // parse application/vnd.api+json as json
app.use(methodOverride());
app.use(cors());

//maybe set up some headers
app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header('Access-Control-Allow-Methods', 'DELETE, PUT, GET'); // or get?
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    next();
});
```

Figura 16 - A conexão com o MongoDB.

A Figura 16 mostra a conexão utilizada com o banco, com o campo de senha alterado. A biblioteca “Mongoose” utiliza a função “connect” para se conectar, e a biblioteca “Express” utiliza os valores de header de um CRUD comum para acessar a aplicação e demonstrar as funções que serão utilizadas (Express, 2018).

4.8 O MongoDB

O MongoDB é um banco de dados orientado a documentos. Diferentemente dos bancos de dados relacionais, é um banco de dados NoSQL (Not Only SQL)(MongoDB, 2018). Suas principais características são possuir todas as suas informações importantes em um único documento, sendo livre de esquemas, e possuir identificadores únicos universais (MongoDB, 2018). Foi utilizado no trabalho pois além de ser NoSQL, disponibiliza um espaço de operações simples gratuito.

No trabalho, foi utilizado a biblioteca “Mongoose”, que facilita as operações básicas em um banco de dados, como consulta, escrita e atualização de documento. O Mongoose também facilita o modelo de dados da aplicação (Mongoose, 2018). Os modelos que são possíveis criar nele são similares aos de um banco de dados relacional, onde é possível especificar o tipo dos dados que entram. As consultas também utilizam métodos facilitados. O Mongoose, além disso, cria rotas (URLs) que se conectam com o banco de dados diretamente.

O espaço de banco de dados utilizado foi disponibilizado pelo próprio MongoDB, que permite usuários cadastrados utilizarem um espaço de dados limitado para armazenamento de informação. A Figura 17 ilustra a página de configuração da conexão com o MongoDB, em que é disponibilizada a URL de acesso.

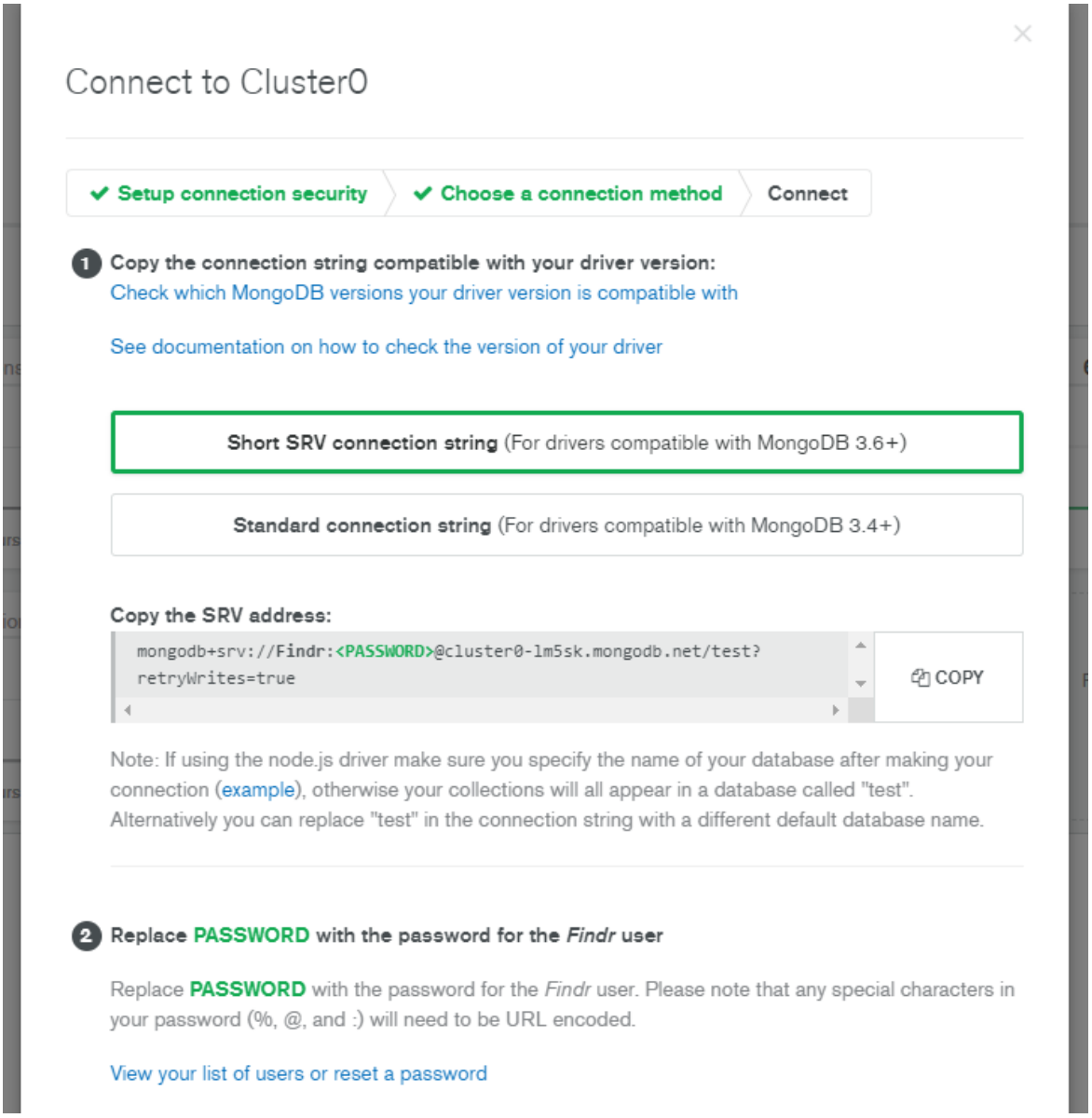

Figura 17 - A conexão com o banco a partir da página Web do MongoDB. Fonte: mongodb.com

5 O aplicativo Findr

5.1 Introdução ao Sistema

O Findr em si é um aplicativo que permite que uma pessoa faça login utilizando a rede social Facebook. Uma vez feito o login, o aplicativo envia sua localização para a base de dados do sistema, busca nessa base outros usuários fisicamente próximos, e exibe na tela uma lista contendo tais usuários, se utilizando apenas da Internet e do serviço de localização do *smartphone*, e exibindo o campo de subtítulo que o usuário disponibilizou. Há também outras duas telas: a de configuração pessoal, e a de visualização de perfil.

5.2 O método de geolocalização

O Sistema utiliza um método de geolocalização em 2D, a partir de uma função nativa do próprio banco de dados MongoDB, que procura os usuários próximos a partir de uma distância originalmente especificada de 100 metros, ao invés de trabalhar os dados no próprio smartphone.

O Ionic disponibiliza uma biblioteca que pode ser utilizada para obter a localização atual do usuário, assim como outras funções úteis. A Figura 18 ilustra uma função dessa biblioteca do Ionic. Por sua vez, o MongoDB disponibiliza outra função que torna possível trabalhar com as coordenadas obtidas, que é chamada "$near" (MongoDB, 2018). Essa função utiliza padrões de distância mínima e máxima, no caso ambas sendo 100 metros, recebendo coordenadas para retornar ao sistema uma lista com todos os usuários cujas coordenadas se encontram em uma distância especificada (100 metros).

```
ionViewWillLeave() {
  this.geolocation.getCurrentPosition().then(res => {
    this.person.latitude = res.coords.latitude;
    this.person.longitude = res.coords.longitude;
  }).catch(err => {
    console.log(err);
  });
}
```

Figura 18 - Função da biblioteca do Ionic para a busca de coordenadas. Fonte: Ionic Framework, 2018.

```
{
   <location field>: {
     $near: {
       $geometry: {
          type: "Point" ,
          coordinates: [ <longitude> , <latitude> ]
       },
       $maxDistance: <distance in meters>,
       $minDistance: <distance in meters>
     }
   }
}
```

Figura 19 - Função de distância a partir de coordenadas. Fonte: docs.mongodb.com/manual/reference/operator/query/near/

5.3 Recursos e telas

A seguir, uma introdução sobre as telas da aplicação e suas funcionalidades, com uma breve explicação sobre cada uma.

1. Tela de Login

A Figura 20 descreve a tela inicial do aplicativo, na qual o usuário pode realizar o acesso a partir da rede social Facebook. Ao clicar no *link*, é automaticamente redirecionado ao aplicativo, que permite o login a partir da sua conta na rede social. Após logado uma vez, o aplicativo passa a pular a tela de login em próximas vezes que o usuário acessar o aplicativo, a menos que o usuário escolha fazer o logout ou reinstalar a aplicação.

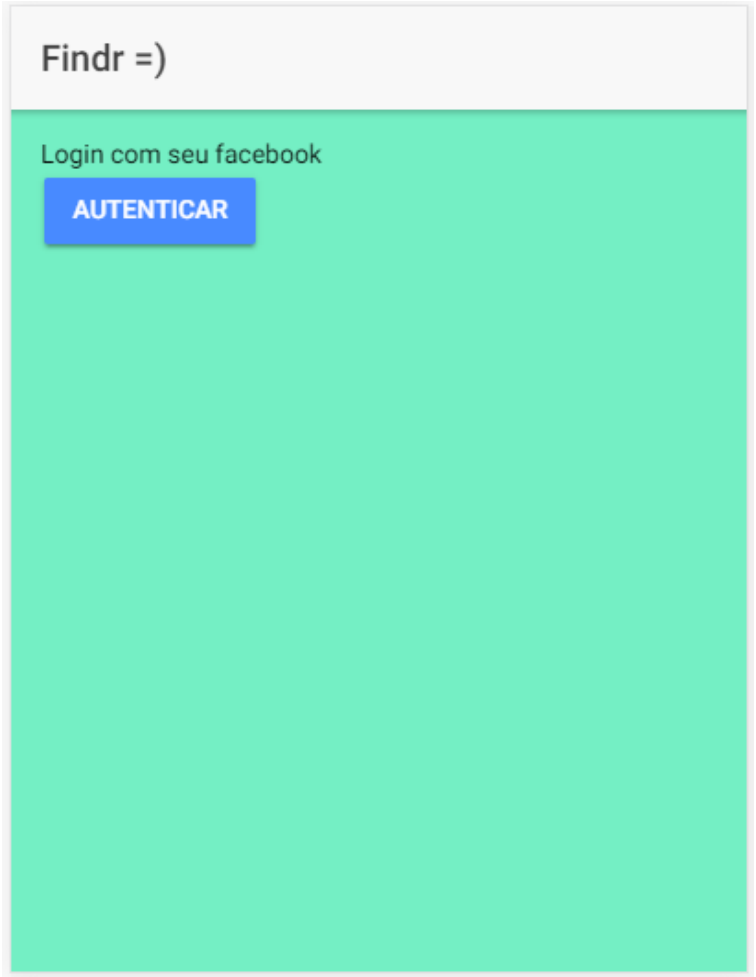

Figura 20 - Tela de login da aplicação.

2. Tela Principal

A Figura 21 descreve a tela principal da aplicação, acessada logo após o usuário ter feito login na tela acima. Ao acessar esta tela, é feita uma requisição ao servidor que retorna todos os usuários presentes em um raio de até 100 metros, com suas imagens de perfil e seus subtítulos escolhidos.

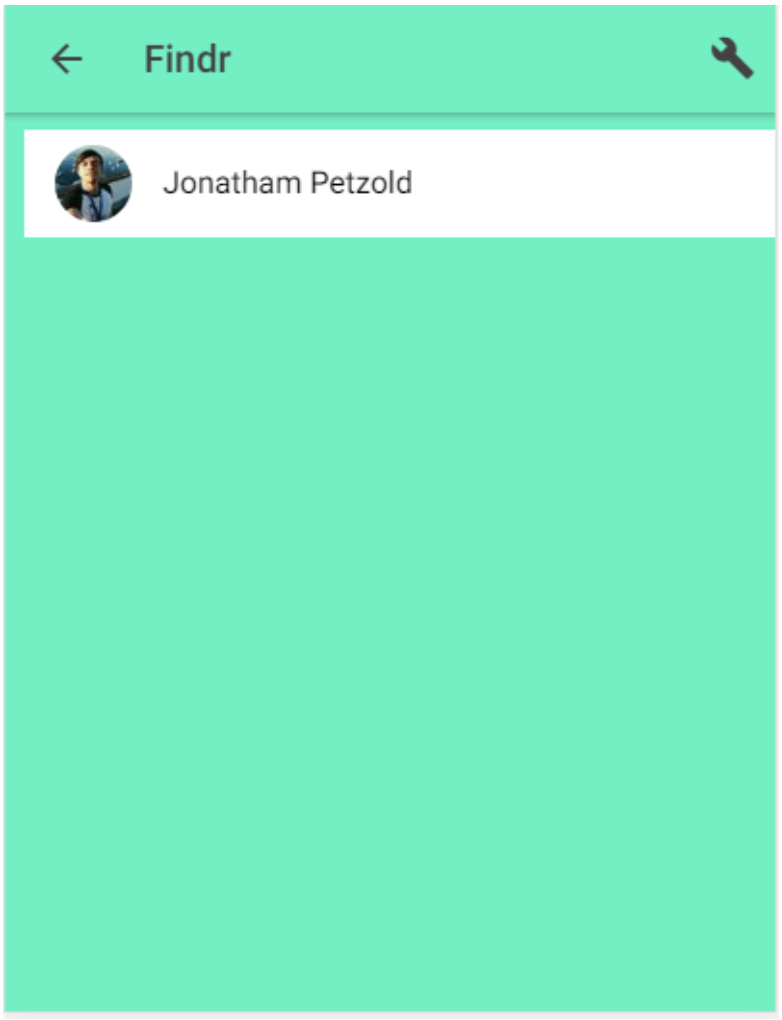

Figura 21 - Tela principal da aplicação.

3. Tela de Configuração

A Figura 22 descreve a tela é acessada quando o usuário clica no ícone de configuração no canto superior direito da tela principal. Nesta tela, o usuário pode digitar as informações que deseja compartilhar, e pode posteriormente salvar as informações clicando no botão de salvar, ou não salvar as informações ao clicar no botão de voltar à tela anterior.

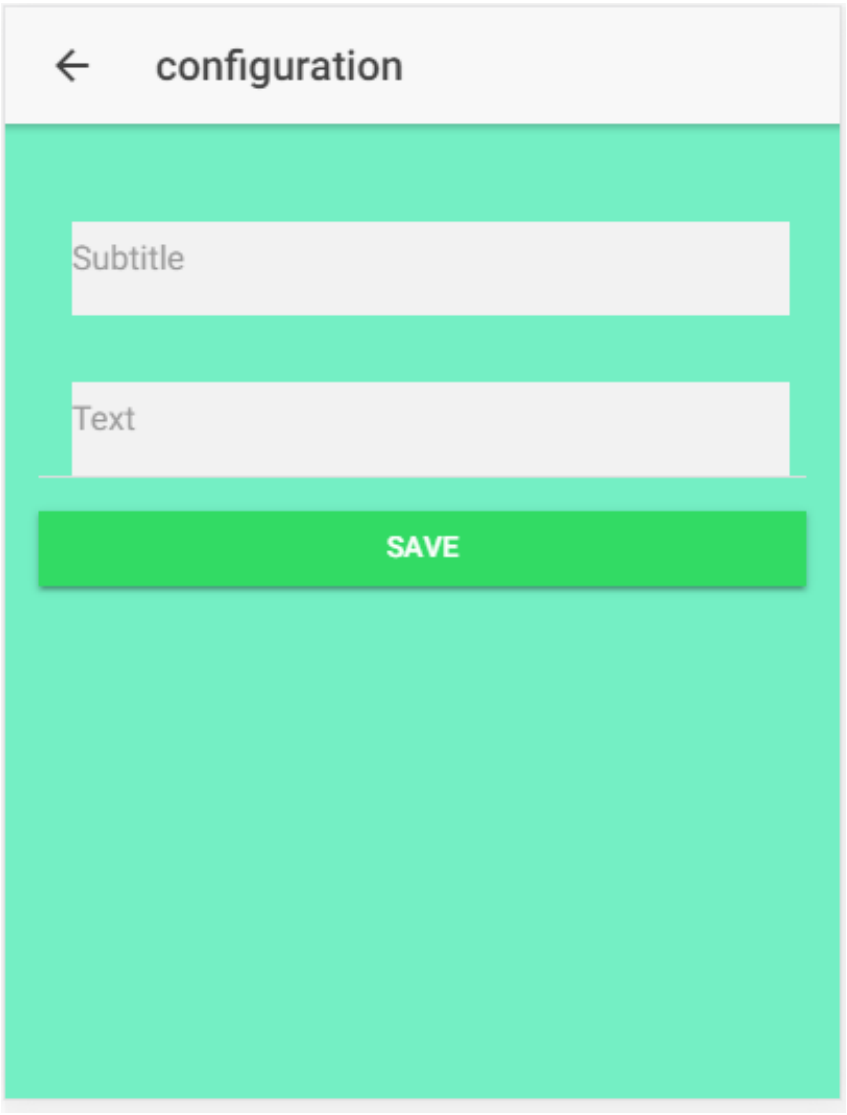

Figura 22 - Tela de configuração da aplicação.

4. Tela de Perfil

A Figura 23 descreve a tela é acessada quando o usuário clica em um dos perfis disponíveis na tela principal. Ao entrar na tela, é feita uma requisição ao banco, que retorna as informações que o usuário alvo disponibilizou em suas configurações. Nesta tela há também um botão sinalizado com “X” no canto superior direito, que fecha a tela e retorna à tela principal.

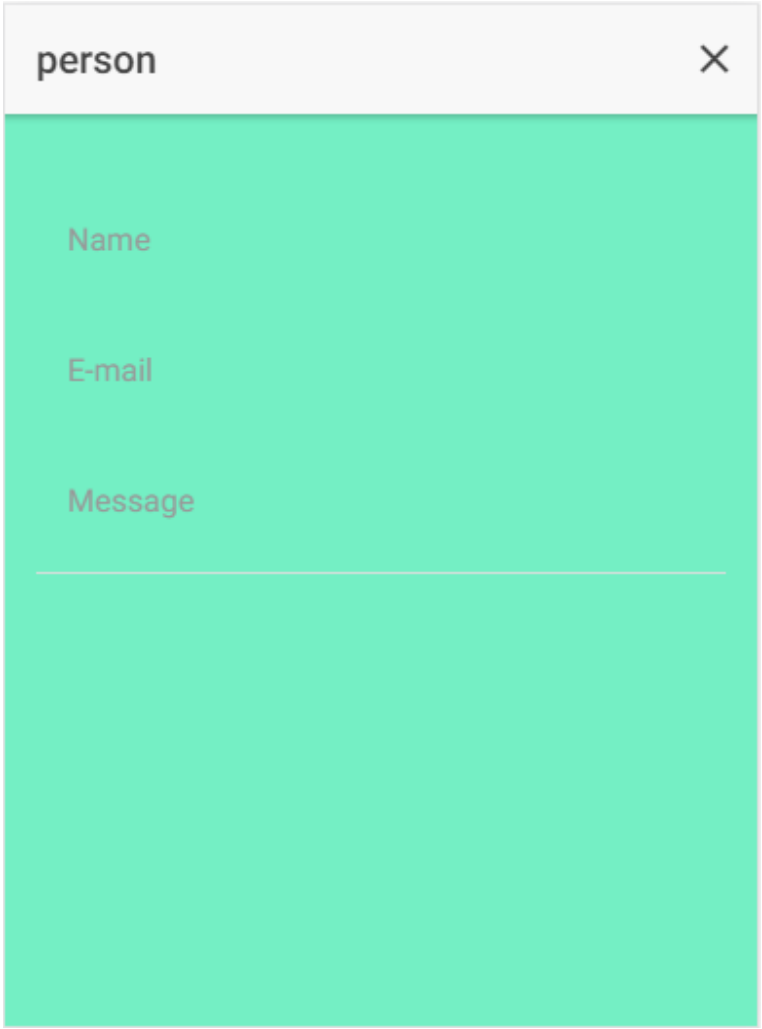

Figura 23 - Tela de perfil da aplicação.

6 Conclusões

6.1 Considerações Finais

Neste trabalho foi apresentado o aplicativo Findr. Disponibilizado para Android, ele auxilia usuários na troca de informações com demais usuários próximos. Para construir a aplicação, foi utilizado o framework Ionic, que permite a criação de telas a partir de códigos HTML, CSS e Javascript. Além disso, foi utilizado o modelo de desenvolvimento Cascata, e o padrão de arquitetura MVC2.

Ao fim do trabalho, foi possível observar como vantagens, que o aplicativo pode facilitar a troca de informações entre usuários de forma cotidiana, seja ao trocar números, seja ao trocar informações como links úteis. Dentre as vantagens das tecnologias utilizadas, é possível evidenciar a praticidade de se desenvolver um aplicativo utilizando *frameworks* híbridos.

Como desvantagens, o aplicativo solicita uso constante do hardware do celular, devido à geolocalização ativa. Além disso, o sistema de localização da aplicação não é preciso, o que torna algumas vezes problemática a troca de informações por usuários no limite de alcance da aplicação.

O servidor utilizado durante o trabalho (como a maioria dos servidores em larga escala) é pago e, portanto, o tráfego de informações salvas bem como o tráfego de dados pode ser limitado.

6.2 Trabalhos Futuros

Dentre as opções de trabalhos futuros relacionados ao projeto, é possível indicar:

- Estruturar o sistema para permitir o envio de arquivos por usuários, a fim de permitir não só que as informações de um usuário fique disponível a usuários próximos, como também seus arquivos;
- Estruturar o sistema de forma que seja possível criar uma solicitação de acesso a informações específicas;
- Disponibilizar o aplicativo em lojas virtuais como a Play Store do sistema

Android; e

- Disponibilizar o acesso de empresas ao aplicativo que também desejem compartilhar informações a clientes pelo aplicativo por restrições de proximidade.

Referências Bibliográficas

1. Apache Cordova. Página Inicial. Disponível em: https://cordova.apache.org/docs/en/5.1.1/guide/overview/. Acesso em out. 2018.

2. Apple. Página de suporte da aplicação AirDrop. Disponível em: https://support.apple.com/pt-br/HT204144. Acesso em jan. 2019.

3. Express. Documentação oficial. Disponível em: https://expressjs.com/. Acesso em nov. 2018

4. Gois, A. Ionic Framework: Construa aplicativos para todas as plataformas mobile. Casa do Código, 2017.

5. Happn. Página Inicial. Disponível em: https://www.happn.com/pt/. Acesso em jan. 2019.

6. IBGE. Página de Notícia. Disponível em: http://agenciabrasil.ebc.com.br/geral/noticia/2016-12/numero-de-pessoas-que-tem-celular-aumenta-147-em-dez-anos-diz-ibge. Acesso em jan. 2019.

7. Ionic Framework. Documentação oficial. Disponível em: https://ionicframework.com/docs/. Acesso em out. 2018

8. Martins, Elaine.Como funciona o GPS. TEC MUNDO. São Paulo. 10 Ago. 2009.

9. MongoDB. Documentação oficial. Disponível em: https://docs.mongodb.com/. Acesso em nov. 2018

10. Mongoose. Documentação oficial. Disponível em: https://mongoosejs.com/docs/api.html. Acesso em nov. 2018

11. Pereira, C. R. Aplicações web real-time com Node.JS. Casa do Código, 2014.

12. Pereira, C. R. Construindo APIs REST com Node.JS. Casa do Código, 2016.

13. Prescott, P. HTML 5. Tradução de Paulo Alexandre F. M. Torres. [S.l]: Babelcube Inc, 2015.

14. Seshadri, S.; GREEN, B. Desenvolvendo com AngularJS: AUMENTO DE PRODUTIVIDADE COM APLICAÇÕES WEB ESTRUTURADAS. São Paulo, SP: Novatec Editora, 2014.

15. Silva, M. S. CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. Primeira edição. São Paulo: Novatec Editora, 2012.

16. Sommerville, I. Engenharia de software. 9. ed. São Paulo, SP: Pearson Prentice Hall, 2011.

17. Williamson, K. Introdução ao AngularJS. São Paulo, SP: Novatec Editora, 2015.