Desenvolvimento de um aplicativo de relacionamento baseado na geolocalização

Jonas Prunzel[1], Alessandro Mainardi de Oliveira[1]

[1]Curso de Ciência da Computação – Universidade Franciscana (UFN)
CEP 97010-032 – Santa Maria – RS – Brasil

{j.prunzel, alessandroandre}@ufn.edu.br

Abstract. *The present article shows the proposal of an application, with the purpose of facilitating the interaction of people geographically close to each other. The same will be developed for the Android platform. The methodology used for development was Feature-Driven Development. For the development of the application will be used the programming language Kotlin along with the database Firebase which is responsible for the storage of the data. Thus, the project aims to increase the chances of people socializing with new people.*

Resumo. *O presente artigo mostra a proposta de um aplicativo, com o objetivo de facilitar a interação de pessoas geograficamente perto uma das outras. O mesmo foi desenvolvido para a plataforma Android. A metodologia utilizada para o desenvolvimento foi a Feature-Driven Development. Para o desenvolvimento do aplicativo foi utilizada a linguagem de programação Kotlin, juntamente com o banco de dados Firebase, que é responsável pelo armazenamento dos dados. Assim, o projeto tem o propósito de aumentar as chances de pessoas interagir com novas pessoas.*

1. Introdução

A maioria das pessoas já passaram por situações em que se sentiram envergonhadas ou desconfortáveis ao longo de suas vidas. A reação de sentir medo quando estamos diante de uma situação de interação com pessoas ou situações novas é comum. A diferença é que para as pessoas mais tímidas, esse sentimento não pode ser simplesmente deixado de lado [Ferrari 2019]. Uma maneira de diminuir esse desconforto é através de aplicativos para *smartphones*.

Os aplicativos de relacionamentos encontrados hoje oferecem informações dos usuários como fotos, idade, descrição, estilo musical entre outros. Tendo o objetivo de facilitar a identificação de quem pode ter mais afinidade entre si sem precisar conversar pessoalmente. Desta maneira são aumentadas as chances de encontrar indivíduos com gostos comuns entre si. Muitas vezes mesmo depois de ter conversado com alguém encontrado nos aplicativos, os usuários não se conhecem pessoalmente por muitos motivos, como não morar perto, não ter horários livres em comum ou até mesmo perder o interesse por já ter passado um tempo e a pessoa não se encontra mais na mesma disposição emocional que estava no momento em que conheceu a pessoa no aplicativo.

Com o objetivo de facilitar o encontro entre pessoas, o aplicativo criado neste projeto, é uma ferramenta capaz de proporcionar um encontro em poucos minutos. Com a exata localização de cada pessoa e os mesmos estando próximos, é maior a chance de que os dois saiam do aplicativo para o encontro real. Em eventos com festas, feiras e shows, devido à grande quantidade de pessoas, é difícil identificar quem está disposto a socializar com um desconhecido.

1.1. Objetivo Geral

O objetivo desse trabalho é desenvolver um aplicativo para a plataforma Android que consiga mostrar a localização dos usuários, para que ele possa identificar quem também está disponível em sua proximidade, após um usuário aceitar a solicitação do outro os dois podem conversar em um *chat* privado entre eles.

1.2. Objetivos Específicos

Os objetivos específicos mostram o que foi necessário para a pesquisa e estudo do presente trabalho.

- Empregar a linguagem de programação Kotlin na implementação do aplicativo.
- Utilizar o banco de dados Firebase para armazenar os dados na nuvem.
- Implementar o projeto conforme os requisitos para atender as suas funcionalidades.
- Validar o aplicativo.

2. Referencial Teórico

Esta seção apresenta o referencial teórico do trabalho, cujo objetivo é criar um plano de sustentação argumentativo sobre o tema a ser abordado, dando embasamento e servindo como comparação em relação aos resultados a serem obtidos a partir do trabalho desenvolvido.

2.1. Tecnologias

Esta subseção apresenta as tecnologias computacionais necessárias para o desenvolvimento do aplicativo proposto.

2.1.1. Google Maps

O Google Maps foi desenvolvido pela empresa Google, o seu objetivo é mostrar a localização do usuário através de um mapa. A API (*Application Programming Interface*) tem como principal função automatizar o acesso aos servidores do Google Maps, *download* de dados, exibição de mapas e resposta a gestos de mapas. Marcadores, polígonos e sobreposições podem ser adicionados ao mapa através de chamadas de API, a visualização do usuário em uma área específica também pode ser alterada [Google 2019].

Na aplicação, o Google Maps tem como função auxiliar na localização, tanto no cadastro dos eventos como publicação, e os usuários podem visualizar o endereço em um fragmento de mapa.

2.1.2 Android

O Android fornece uma estrutura de aplicativo avançada que permite criar aplicativos e jogos inovadores para dispositivos móveis em um ambiente de linguagem Kotlin [Android 2020].

O aplicativo precisa acessar alguns recursos disponíveis no dispositivo. A plataforma Android tem como característica a indiferença entre os aplicativos integrados e os aplicativos criados com o SDK (Software Development Kit), proporcionando o acesso a recursos do dispositivo [Ableson et al. 2012].

Seu objetivo principal é criar uma plataforma onde desenvolvedores, operadoras e fabricantes possam inserir suas ideias e inovações resultando em um produto que realmente aprimora a experiência do usuário [Android 2020].

2.1.3 Firebase

O Firebase é um SGBD (Sistemas de Gestão de Base de Dados) que possui funções para a elaboração de aplicativos *mobile* e *web* através de ferramentas e infraestruturas que tem o objetivo de ajudar os desenvolvedores a construir bons aplicativos. Por ser um banco de dados não relacional, dispensa e elaboração de um esquema antes de sua implementação, pois todas as informações ficam agrupadas em um único registro [Firebase 2019].

Em relação a proteção de dados, o Firebase está adaptado ao Regulamento geral de proteção de dados da União Europeia, a qual impõe obrigações aos controladores e processadores de dados. Também conta com certificados de acordo com os principais padrões de privacidade e segurança [Firebase 2020].

2.1.4 Kotlin

Foi escolhida esta linguagem pois tem total compatibilidade com a plataforma Android Studio. Disponibiliza recursos como refatoração e depuração do código, o que auxilia muito na identificação de erros no projeto [Kotlin 2020].

2.2. Trabalhos correlatos

Nesta parte é possível observar trabalhos com ideias e características parecidas com o projeto, que tem o objetivo de agregar conhecimento contribuindo com uma melhor elaboração e organização do aplicativo.

2.2.1. Aplicativo de relacionamento Tinder

O Tinder foi criado em 2012, é um aplicativo de relacionamento baseado em geolocalização. O usuário cria seu perfil pessoal podendo adicionar fotos, música preferida, local de trabalho, instituição de ensino e uma descrição. Ele permite que os usuários se comuniquem através dos chamados *matchs*, que é quando dois usuários demostram interesse simultâneo deslizando a tela para a direita no perfil selecionado, caso não tenha interesse basta deslizar a tela para esquerda [Tinder 2019].

2.2.2. Aplicativo de relacionamento Happn

O Happn tem o mesmo objetivo do Tinder por também ser um aplicativo de relacionamento, porém tem uma maneira diferente de atingi-lo. Ele se baseia pelo GPS (*Global Positioning System*), mostrando apenas perfis que passaram perto um do outro em determinado momento. A localização exata não é informada, apenas a proximidade do lugar em comum [Happn 2019].

2.2.3. Aplicativo Snapchat

O Snapchat é um aplicativo de troca de mensagens, fotos e vídeos que são apagadas após a visualização, além disso ele possui um mapa mostrando a localização da pessoa que é sempre atualizada enquanto o seu dispositivo estiver conectado à internet. Para ter acesso a localização de outros usuários é necessário mandar uma solicitação de amizade, tendo em vista que o Snapchat é mais voltado para uma rede social [Snapchat 2019].

2.2.4 Considerações sobre os trabalhos correlatos

O diferencial do aplicativo é descobrir novas pessoas pois todos os usuários serão exibidos no mapa, focando mais em dar a oportunidade de se conhecerem quando utilizarem o aplicativo. Em relação ao Tinder a diferença é o mapa com a localização dos usuários, o Tinder mostra apenas a distância em que um usuário está do outro. Para utilizar o aplicativo apresentado neste trabalho, o usuário deverá aceitar um termo se responsabilizando em mostrar sua exata localização.

3. Metodologia

Nesta seção é apresentada a metodologia utilizada para a realização do trabalho, com o objetivo de mostrar os fluxos, diagramas, requisitos e as principais funcionalidades do aplicativo.

3.1. *Feature-Driven Development* (FDD)

Começou por ser concebido por Peter Coad e seus colegas, e mais tarde Stephen Palmer e John Felsing estenderam e melhoraram o processo orientado a objetos que pode ser aplicado a projetos de software de tamanho moderado e grande [Tomás 2009].

O FDD coloca mais ênfase em diretrizes e técnicas de gestão de projeto do que muitos outros métodos ágeis [Pressman 2006]. Segundo Barbosa et al. (2011), o FDD constitui-se basicamente de cinco processos, que definem a maneira de trabalho adotada pela metodologia: Desenvolver um Modelo Abrangente, Construir uma Lista de Funcionalidades, Planejar por Funcionalidades, Projetar por Funcionalidades e Construir por Funcionalidades.

3.1.1. Desenvolver um modelo abrangente

Modelo Abrangente é responsável pelo estudo sobre o domínio do negócio e pela definição do escopo do projeto [Silva et. Al 2009]. A Figura 1 apresenta um Diagrama de Objetos, que mostra em tempo de execução, os objetos do *software* e seus inter-relacionamentos.

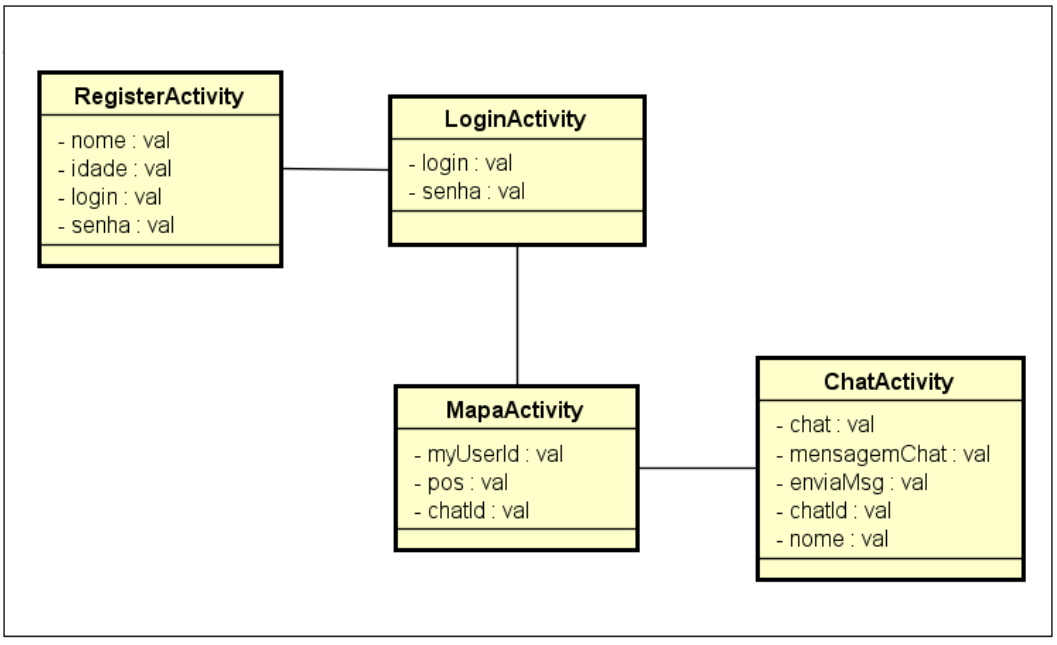

Figura 1. Diagrama de Objetos

A tela inicial do projeto apresenta o login do usuário, caso ele ainda não tenha se registrado, tem a opção de se registrar informando o seu login, nome, idade e senha. A Figura 2 mostra as telas de interação entre as pessoas, a Tela 1 é mostrada após o usuário fazer o *login*, onde exibe o mapa com a localização dos outros usuários cadastrados, sendo identificados pelo desenho de um pingo vermelho, com seu nome e idade identificados em cima, no projeto futuro o desenho do pingo vermelho vai ser substituído pela foto do usuário. A Tela 2 é composta pelo *chat*, o qual apresenta-se depois que um usuário mandar uma solicitação que deve ser aceita pelo outro usuário.

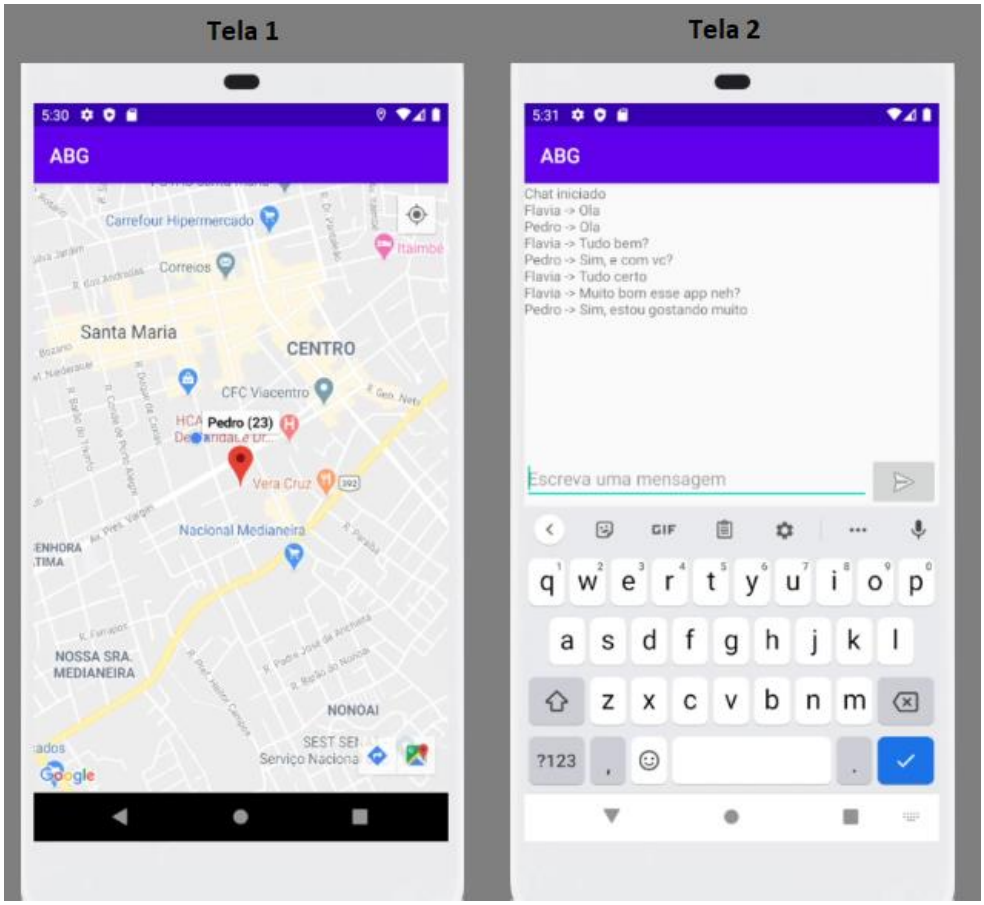

Figura 2. Telas de Interação

3.1.2. Construção da lista de funcionalidades

Segundo Silva todas as funcionalidades necessárias ao cumprimento das necessidades do cliente são levantadas na lista de funcionalidades [Silva 2009]. Apresenta-se o Quadro 1 de Requisitos Funcionais (RF).

Quadro 1. Requisitos funcionais do sistema

Requisitos Funcionais		
Funcionalidade	Descrição	Complexidade
RF01: Cadastrar usuário	Deve permitir cadastrar usuário.	Baixo
RF02: Logar usuário	O aplicativo deve permitir o login do usuário.	Baixo
RF03: Mostrar localização dos outros usuários	O aplicativo deve mostrar a localização no mapa dos usuários cadastrados.	Médio
RF04: Enviar solicitação de conversa	O aplicativo deve permitir enviar uma solicitação de conversa para outro usuário ativo no mapa.	Médio
RF04.1: Enviar retorno da solicitação de conversa	O aplicativo deve enviar um retorno mostrando se o usuário aceitou a solicitação de conversa.	Baixo
RF05: Abrir um canal de comunicação entre os usuários	Se a solicitação de conversa for aceita o aplicativo deve abrir um canal para os usuários enviar mensagens.	Médio

Apresenta-se no Quadro 2 os Requisitos Não Funcionais do sistema (RNF):

Quadro 2. Requisitos não funcionais do sistema

Requisitos Não Funcionais	
Funcionalidade	Descrição
RNF01	O aplicativo deverá ser desenvolvido para plataforma Android.
RNF02	O aplicativo deverá ser desenvolvido na linguagem de programação Kotlin.
RNF03	As informações de localização deverão ser armazenadas na nuvem, através do banco de dados Firebase.
RNF04	O aplicativo deverá fornecer um registro para que usuários sejam denunciados devido a atitutes com desconfirmidades as leis.

3.1.3. Planejar por Funcionalidades

Esta fase tem como objetivo construir um plano de desenvolvimento baseado no conjunto de funcionalidades. Essas atividades são consideradas em conjunto após serem feitos diversos refinamentos [Silva 2016].

Tabela3. Estimativa do tempo de implementação das funcionalidades

Código	Funcionalidade	Tempo (Horas)
RF01	Cadastrar usuário	10
RF02	Logar usuário	5
RF03	Mostrar localização dos outros usuários	15
RF04	Enviar solicitação de conversa	20
RF04.1	Enviar retorno da solicitação de conversa	10
RF05	Abrir um canal de comunicação entre os usuários	40

Na Figura 3 é apresentado o Diagrama de Casos de Uso, onde o mesmo especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator [Booch, Rumbaugh e Jacobson 2000].

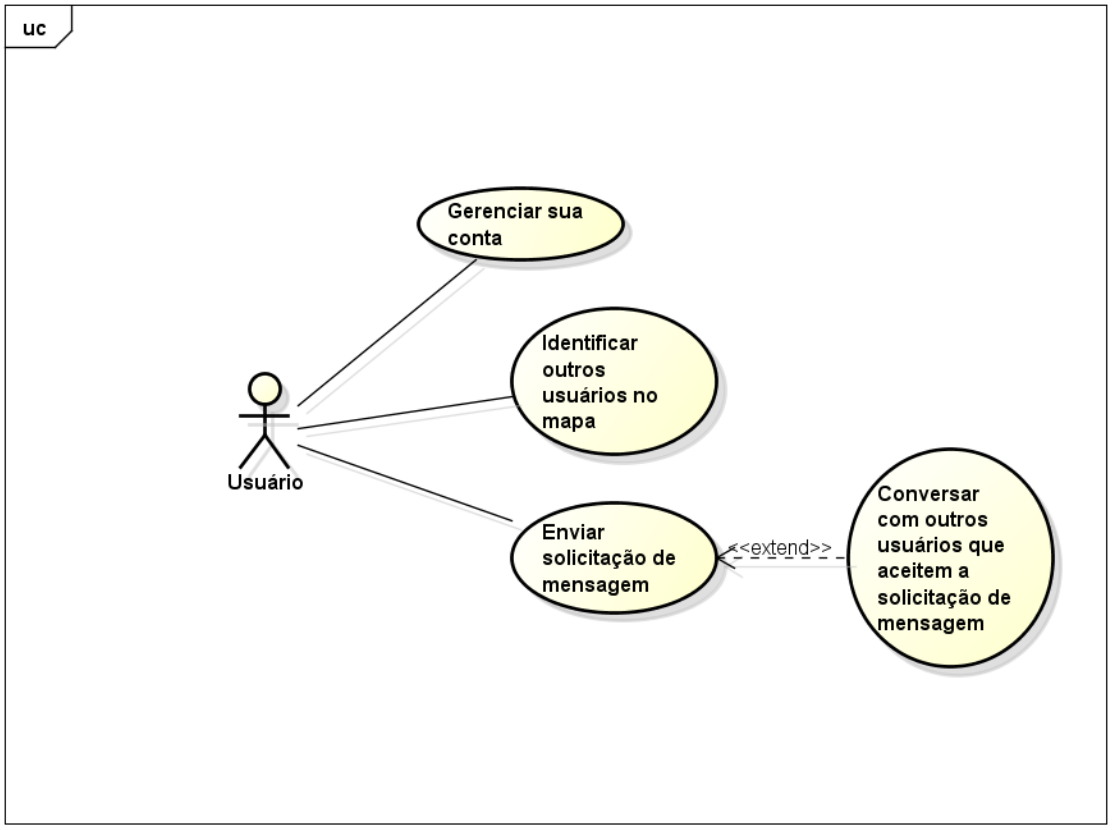

Figura 3. Diagrama de Caso de Uso

3.1.4. Projetar por Funcionalidades

Nesta secção são definidas as atividades para cada funcionalidade do sistema. Para representá-las foi feito um Diagrama de Atividade. Este diagrama é essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra [Booch, Rumbaugh e Jacobson 2000]. Na Figura 4 pode ser observado o Diagrama de atividade.

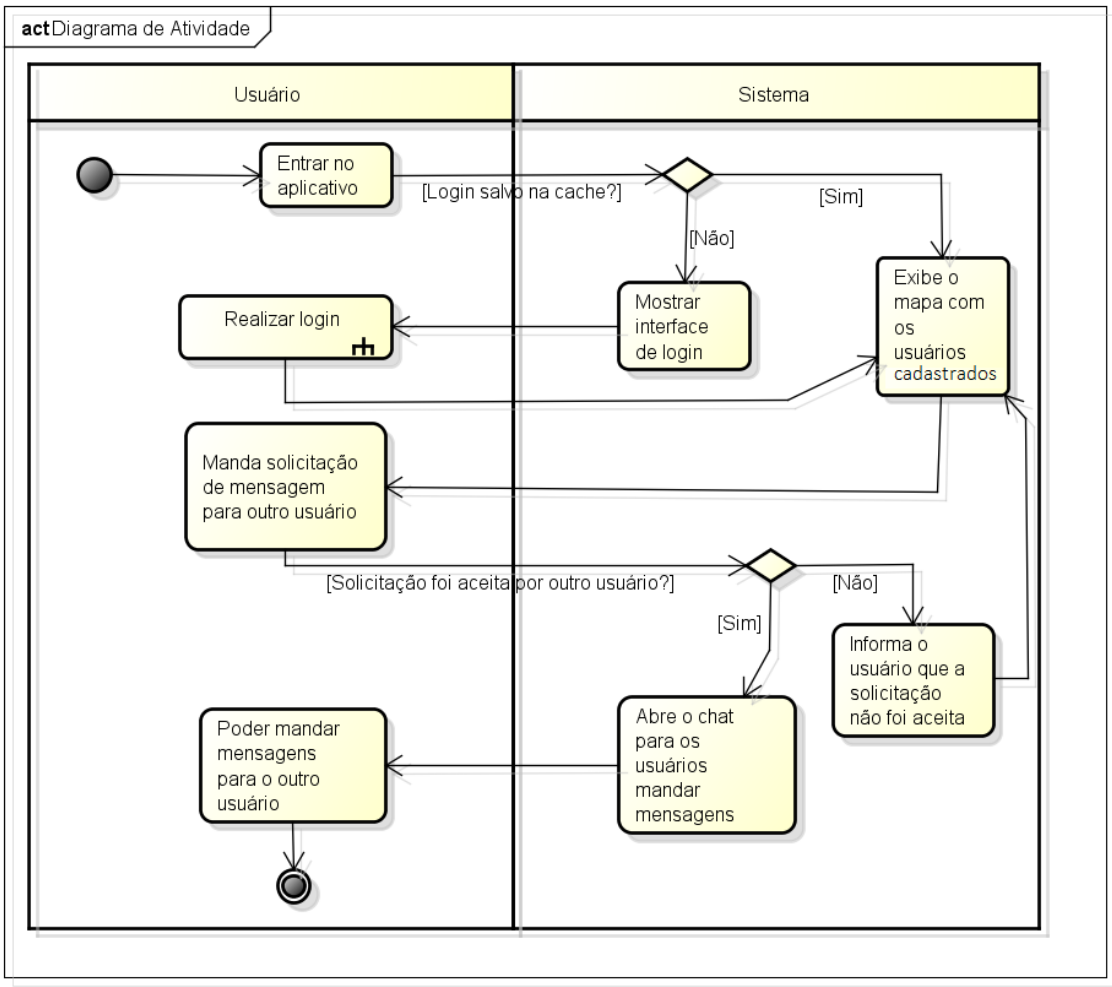

Figura 4. Diagrama de Atividade do fluxo geral do aplicativo

O subprocesso Realizar login, é responsável pelo cadastro do usuário, caso já seja cadastrado e as informações estejam salvas, o aplicativo mostra diretamente o mapa com os usuários ativos no mesmo. Se ainda não for cadastrado é permitido realizar o cadastro.

O processo Mandar solicitação de mensagem para outros usuário, mostra a atividade necessária para que dois usuários possam se comunicar, com a solicitação aceita, é criado um *chat* composto apenas pelo usuário que mandou a solicitação e o que aceitou. A Tela 1 da Figura 5 mostra a solicitação enviada por Flavia, enquanto a Tela 2 mostra a resposta do Pedro.

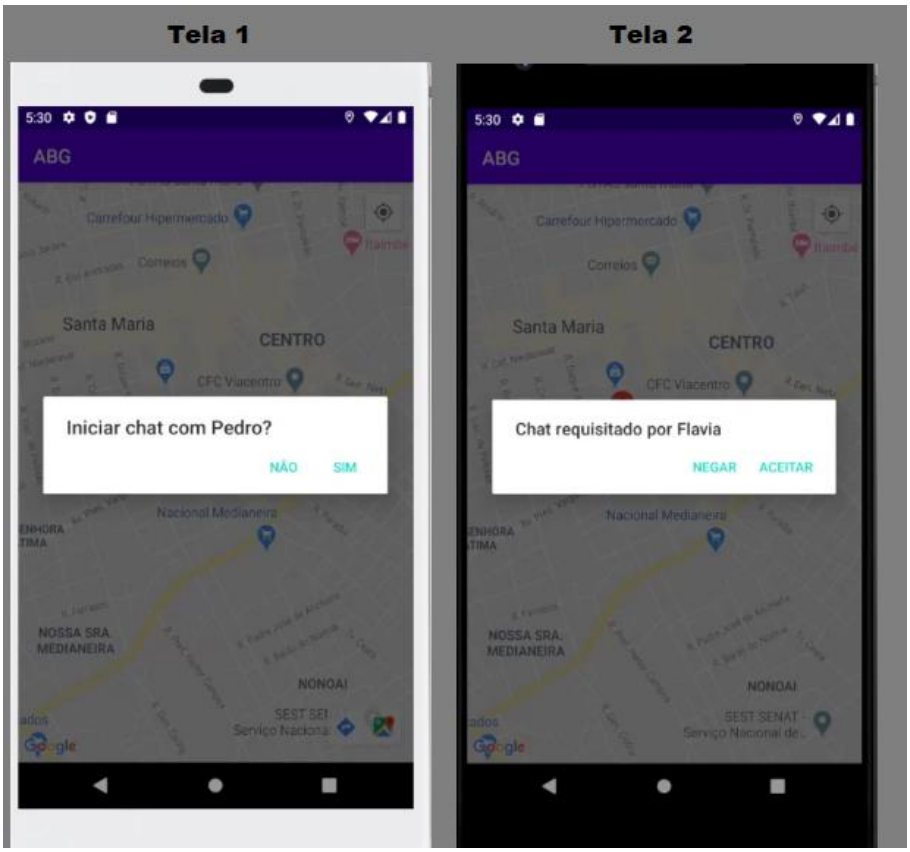

Figura 5. Solicitação para enviar mensagem

3.1.5. Construir por Funcionalidades

Nesta etapa os códigos são criados conforme o que foi analisado e elaborado nas etapas anteriores. É criado um produto onde é possível fazer a validação dos requisitos solicitados [Sbrocco e Macedo 2012].

No banco de dados orientado a documentos os dados são encapsulados em pares de chave-valor. Cada documento tem um identificador exclusivo dentro da sua coleção de documentos. Os documentos não têm nenhuma estrutura definida, por conta disso não é preciso definir um esquema de dados [Wanzeller 2013]. Foram criadas quatro coleções para trabalhar com os dados do projeto. A Figura 6 mostra um esquema em que é possível ver a relação entra as coleções do banco de dados. No documento da coleção “usuario” ficam salvas as informações do usuário como o seu nome, idade e coordenadas geográficas, além do seu “id” que é relacionado com o “id” da coleção “authentication” que é responsável por fazer o *login* no aplicativo. Quando o usuário envia uma solicitação para outro usuário é criado um documento na coleção “*call*”, onde fica salvo o “id” dos dois usuários e o “id” do *chat* em que eles devem entrar. No documento da coleção “*chat*” ficam salvas as mensagens trocadas pelos usuários. Quando o *chat* é fechado no aplicativo as mensagens são apagadas.

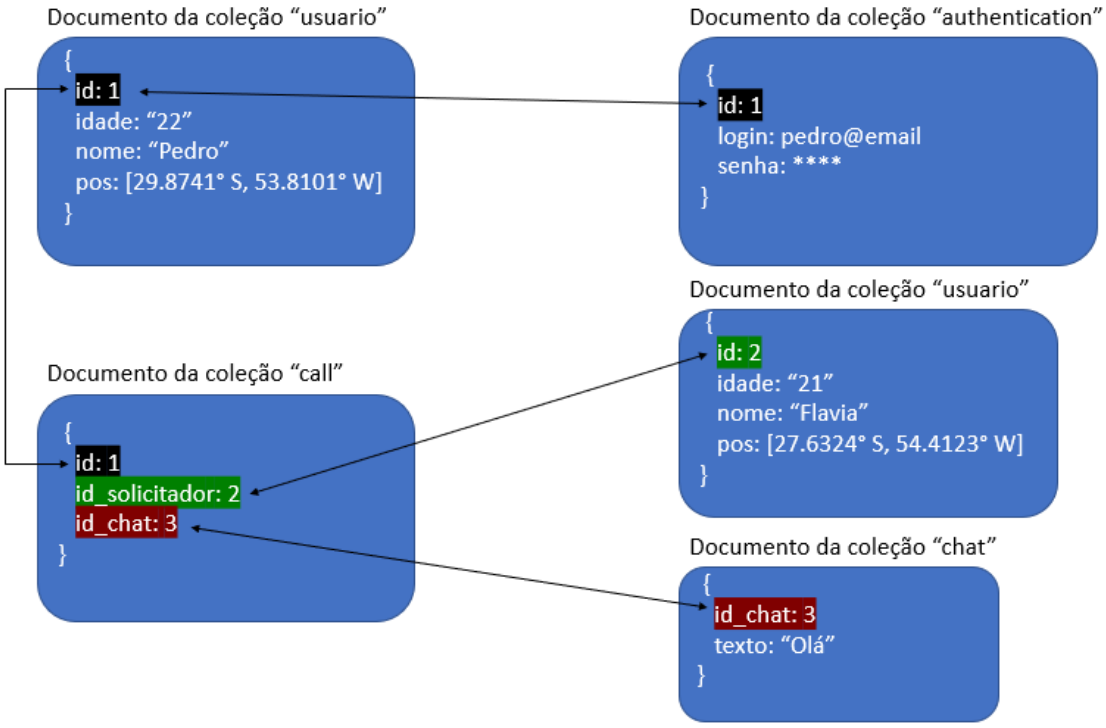

Figura 6. Relações Entre Coleções

No começo da implementação foi alterada a linguagem de programação de Java para Kotlin, pois esta possui uma sintaxe que ajuda a simplificar o código. Para cadastro e login do usuário foi utilizado uma funcionalidade de autenticação do Firebase.

A Figura 7 apresenta um trecho do código que mostra uma parte da implementação do mapa do aplicativo. Quando o mapa é inicializado através da API do Google, é adicionado no mapa a localização do usuário, o método *requestPermissions* pede permissão ao usuário para dar acesso a sua localização, com o acesso permitido a propriedade *isMyLocationEnable* mostra a localização atual do usuário. Ao mesmo tempo é percorrido a coleção usuario do banco de dados onde o método *addMaker* adiciona a localização dos outros usuários no mapa.

```
if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
    == PackageManager.PERMISSION_GRANTED) {
    googleMap.isMyLocationEnabled = true
} else {
    ActivityCompat.requestPermissions( activity: this, arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), REQ_CODE)
}

Firebase.firestore.collection( collectionPath: "usuario").get().addOnCompleteListener { res ->
    if (res.isSuccessful) {
        for (doc in res.result!!) {
            if (doc.id == myUserId) continue
            val nome = doc.data["nome"] as String
            val idade = doc.data["idade"] as String
            val pos = doc.data["pos"] as GeoPoint

            val marker = googleMap.addMarker(MarkerOptions().position(LatLng(pos.latitude, pos.longitude)).title("$nome ($idade)"))
            marker.tag = doc;
        }
    }
}
```

Figura 7. Trecho do código para implementação do mapa

Outra parte importante da implementação é a comunicação entre os usuários, para começar, um usuário manda uma solicitação de conversa, neste momento é criado um campo na coleção *chat*, onde fica salvo apenas a última mensagem enviada. A coleção *chat* possui uma identificação (ID), é através dela que os usuários entram no mesmo *chat*, a coleção *call* é responsável por armazenar o nome da pessoa que requisitou a conversa e o ID do *chat* como é mostrado na Figura 8.

```
Firebase.firestore.collection( collectionPath: "call").document(myUserId).addSnapshotListener { doc, err ->
    doc?.data?.let { it: (Mutable)Map<String!, Any!>
        val chatId = it["chat"] as String?
        val nome = it["nome"] as String?

        if(chatId != null && nome != null) {
            val alert = AlertDialog.Builder( context: this)
            alert.setTitle("Chat requisitado por $nome")

            alert.setPositiveButton( text: "Aceitar") { _, _ ->
                val telaChat = Intent( packageContext: this, ChatActivity::class.java)
                telaChat.putExtra( name: "chat", chatId)
                telaChat.putExtra( name: "nome", nome)

                startActivity(telaChat)
            }
```

Figura 8. Trecho do código para implementação do *chat*

3.1.5.1 Validação

Esta é uma versão beta do aplicativo onde foram feitos testes com dois usuários ativos, em que foi alcançado o objetivo de comunicação entre os usuários sem falha. Também foi testado a funcionalidade de autenticação do Firebase, onde foram conferidas se somente as informações de *login* cadastradas permitiam acesso a conta do usuário. Para um projeto futuro é interessante adicionar algumas funcionalidades, por exemplo, adicionar fotos do usuário e sua descrição, para que interesses em comum sejam identificados entre as pessoas. Além disso, mostrar sua localização apenas quando ativo no aplicativo e concordar com os termos de uso, onde constara como requisito, ser maior de idade e estar consciente de que sua real localização será mostrada para os demais usuários ativos.

4. Conclusão e Trabalhos Futuros

Com a análise de aplicativos de relacionamento, foi descoberto que muitas pessoas têm dificuldade em se relacionar com desconhecidos. Com base nesse fato, foi elaborado um aplicativo *mobile* na plataforma Android, que pode facilitar a comunicação entre seus usuários com base na sua geolocalização.

Para atingir esse objetivo, no decorrer do trabalho, efetuou-se um breve estudo sobre o desconforto que algumas pessoas sentem ao conversar com alguém desconhecido. Foram pesquisados trabalhos correlatos com a finalidade de encontrar um diferencial que ainda não exista no mercado. Também foi necessário obter os conhecimentos sobre as tecnologias utilizadas. Com base no conhecimento adquirido, foi possível realizar conforme o planejado no referencial teórico as etapas do projeto.

O trabalho teve como sustentação o banco de dados Firebase, na qual foram exploradas algumas funções de sua biblioteca e plataforma, como a vinculação de campos

entre suas coleções e a segurança sobre os dados, garantida pela adaptação ao Regulamento geral de proteção de dados da União Europeia. A API do Google mapas também merece destaque, por viabilizar um mapa de qualidade excepcional, além de um bom suporte para esclarecer o funcionamento de suas funções.

O próximo passo é viabilizar o lançamento do aplicativo no mercado, para isso é necessário adicionar mais algumas atribuições, como fotos para os usuários, diminuir a abrangência do mapa, melhorar o seu *design* e adicionar os termos para o uso do aplicativo.

5. Referências Bibliográficas

Ableson, Frank; King, Chris; SEN, Robi. (2012) “Android em ação”. Elsevier Brasil.

Android Developers. (2020) “Desenvolver apps para Android com o Kotlin”, https://developer.android.com/kotlin. Acesso em: junho de 2020.

Android Developers. (2020) “Conheça o Android Studio”, https://developer.android.com/studio/intro. Acesso em: junho de 2020.

Barbosa, Antonio et al. (2014) “Metodologia Ágil: Feature Driven Development”. Disponível em: http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_22.pdf. Acesso em: maio de 2019.

Booch, Grady; Rumbaugh, James; Jacobson, Ivar. (2000) “UML: guia do usuário”. Rio de Janeiro, RJ: Campus.

Ferrari, Juliana Spinelli. (2019) "Timidez"; Brasil Escola. Disponível em: https://brasilescola.uol.com.br/psicologia/timidez.htm. Acesso em: julho de 2019.

Firebase Database. (2019) “Firebase Realtime Database”, https://firebase.google.com/docs/database/. Acesso em: maio de 2019.

Firebase Suport. (2020) “Privacidade e segurança no Firebase”, https://firebase.google.com/support/privacy?hl=pt-br/. Acesso em: junho de 2020.

Google Developers. SDK do Google Maps. (2019) https://developers.google.com/maps/documentation/android-sdk/intro?hl=pt-br. Acesso em: maio de 2019.

Kotlin developers. (2020) "Primeiros passos com o Kotlin no Android". Disponível em: <https://developer.android.com/kotlin/get-started>. Acesso em: junho de 2020

Happn. (2019) “Tutorial”, https://www.happn.com/en/#app-concept. Acesso em: março de 2019.

Pressman, Roger S. (2006) “Engenharia de software”. Ed. Rio de Janeiro, RJ: McGraw Hill.

Sbrocco, J. H., e Macedo, P. C. (2012) “Metodologias Ágeis: engenharia de software sob Medida” (1 ed.). São Paulo, Brazil: Érica.

Silva, Alessandra Galvão da. (2016) “A importância dos métodos ágeis na engenharia de software”, https://app.uff.br/riuff/bitstream/1/5488/1/TCC_ALESSANDRA_GALVAO_DA_SILVA%20%281%29.pdf. Acesso em: abril de 2019.

Silva, F. G.; Hoentsch, Sandra CP; Silva, Leila. (2009) “Uma análise das Metodologias Ágeis FDD e Scrum sob a Perspectiva do Modelo de Qualidade MPS”. BR. Scientia Plena, v. 5, n. 12.

Snapchat. (2019) “Crie o seu”, https://www.snapchat.com/l/pt-br/create. Acesso em: abril de 2019.

Tinder. (2019) “O que é o Tinder?”, https://www.help.tinder.com/hc/pt-br/categories/115000755686-Guia-para-o-Tinder-. Acesso em: março de 2019.

Tomás, Mário, R.S. (2009) “IET Working Papers Series”. Disponível em:<https://run.unl.pt/bitstream/10362/2003/1/WPSeries_09_2009Tomas.pdf>. Acesso em 2019.

Wanzeller, D. A. P. (2013) “Investigando o Uso de Banco de Dados Orientados a Documentos para Gerenciar Informações da Administração Pública.” Universidade de Brasília. Brasilia.