Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

UM APLICATIVO MÓVEL PARA PROFISSIONAIS DE EDUCAÇÃO FÍSICA

João Vitor Silva Flores

Prof. Dr. Nilton Cézar de Paula (Orientador)

Dourados - MS
2020

UM APLICATIVO MÓVEL PARA PROFISSIONAIS DE EDUCAÇÃO FÍSICA

João Vitor Silva Flores

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por João Vitor Silva Flores e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 15 de dezembro de 2020

Prof. Dr. Nilton Cézar de Paula

F657a Flores, João Vitor Silva
Um aplicativo móvel para profissionais de educação física/ João Vitor Silva Flores. – Dourados, MS: UEMS, 2020.
224p.

Monografia (Graduação) – Ciência da Computação – Universidade Estadual de Mato Grosso do Sul, 2020.
Orientador: Prof. Dr. Nilton Cézar de Paula

1. Android 2. Personal trainers 3. Geolocalização I. Paula, Nilton Cézar de II. Título

CDD 23. ed. – 005.43

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

UM APLICATIVO MÓVEL PARA PROFISSIONAIS DE EDUCAÇÃO FÍSICA

João Vitor Silva Flores

Dezembro de 2020

Banca Examinadora:

Prof. Dr. Nilton Cézar de Paula (Presidente)
Área de Computação – UEMS

Profa. Ma. Adriana Betânia de Paula Molgora
Área de Computação – UEMS

Prof. Dr. Fabrício Sérgio de Paula
Área de Computação – UEMS

A minha tia Marinalva Correia da Silva (in memorian) e a minha bisavó Maria de Lourdes Silva (in memorian), cujo apoio sempre foi imprescindível, mas que não tiveram a chance de celebrar essa conquista a qual elas fizeram parte.

AGRADECIMENTOS

Agradeço a Deus pela minha vida e pelo suporte nos momentos difíceis durante essa graduação.

Agradeço ao meu orientador pela oportunidade e pelo auxílio durante esse trabalho.

Agradeço a minha família, namorada e todos que me apoiaram e partilharam comigo todos os momentos.

Agradeço a todos os professores que me passaram lições que vão além daquilo que estava na grade curricular, em especial os professores André Chastel Lima, Antonio Aparecido Zanfolim e Rubens Barbosa Filho.

RESUMO

Devido a busca pela vida saudável, os profissionais de educação física têm recebido clientes que almejam mais saúde física e mental. Seja para facilitar os clientes na busca por esses profissionais ou fazer com que esses profissionais tenham mais visibilidade, o objetivo deste projeto é integrar esses dois tipos de pessoas, para que ambos tenham uma ótima experiência. Tudo isso aliado a uma plataforma móvel que está entre as mais utilizadas no mundo, a Android. O aplicativo para profissionais de educação física, denominado ConnectFit, tem como objetivo facilitar a busca por esses profissionais da parte dos clientes, ao mesmo tempo que traz mais visibilidade para os profissionais entre os seus potenciais clientes ao seu redor.

Palavras-chave: Android; *personal trainers*; geolocalização.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Com a busca pela vida saudável através da prática de exercícios, vem também a necessidade de se contratar profissionais especializados para auxiliar nessas ações, os chamados *personal trainers* (PORTAL_ED_FISICA, 2020). Atualmente, os *personal trainers* trabalham com os clientes de diversas formas, sendo as principais fisicamente, via *e-mail* ou por aplicativos móveis já existentes. Mas a forma dos clientes conseguirem encontrar e escolher entre esses profissionais é pouco produtiva e há a necessidade de agrupar todos esses profissionais dentro de uma única plataforma.

Atualmente, dentre as plataformas móveis já existentes, a Android é a mais popular. O sistema operacional para dispositivos móveis desenvolvido pela Google está presente em 74,25% dos aparelhos no mundo (STATCOUNTER, 2020). Tal popularidade é consequência de uma arquitetura robusta e principalmente por disponibilizar o código fonte da base da sua plataforma ao público (AOSP, 2020).

O desafio então consiste em usar um plataforma conhecida para agrupar os serviços oferecidos pelos *personal trainers*.

1.1 Objetivos

Este trabalho tem como objetivo principal desenvolver um aplicativo móvel que agrupa os *personal trainers*, a fim de auxiliar os clientes na busca desses profissionais, para então contratar os seus serviços.

1.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Estudo da plataforma Android;
- Estudo da teoria envolvendo o minimundo educação física;

- Pesquisar sobre os sistemas já existentes, bem como as suas principais funcionalidades e principalmente as suas limitações;
- Estudo de serviços de banco de dados na nuvem;
- Estudo da *Application Programming Interface* (API) do Google Maps;
- Implementar um aplicativo móvel que permita o contato entre *personal trainers* e seus clientes.

1.2 Justificativa e Motivação

Utilizar das novas tecnologias como a geolocalização e computação móvel para auxiliar as pessoas a contratarem *personal trainers*, tendo em vista que o modelo atual de encontrar e contratar esses profissionais seja pouco produtivo. Desta forma, a motivação é desenvolver um aplicativo que seja usado por pessoas em busca de contratar esses *personal trainers*, e que acompanhe o usuário desde a busca pelo profissional até o recebimento dos seus serviços.

1.3 Metodologia

Para os estudos sobre a plataforma Android, os serviços de banco de dados na nuvem, a geolocalização e a teoria envolvendo o minimundo educação física foram consultados livros, artigos na Internet, dissertações e teses.

A lista de aplicativos semelhantes existentes foram buscados dentro da loja oficial de aplicativos para o Android (GOOGLE_PLAY, 2020), e posteriormente foi feita uma busca por mais informações de cada aplicativo na Internet.

A implementação do aplicativo se deu utilizando o Android Studio, um *Integrated Development Environment* (IDE) fornecido pela Google (ANDROID_STUDIO, 2020). Dentro do ambiente de desenvolvimento, foi escolhida a API level 23, que trabalha com as versões a partir da 6.0 no Android.

1.4 Organização do Texto

O **Capítulo 1** apresenta a introdução deste trabalho, os seus objetivos, justificativa e a sua metodologia.

O **Capítulo 2** apresentará o referencial teórico envolvendo a plataforma Android, o minimundo educação física, os aplicativos já existentes, os serviços de banco de dados na nuvem e a API do Google Maps.

O **Capítulo 3** mostrará o desenvolvimento do aplicativo ConnectFit, bem como o seu funcionamento.

No **Capítulo 4** serão apresentadas as telas do aplicativo ConnectFit já finalizado, assim como os testes feitos visando verificar se os objetivos foram atingidos.

O **Capítulo 5** apresenta as conclusões finais do trabalho, bem como algumas melhorias para versões futuras do aplicativo.

No **Apêndice A** será apresentada a instalação do Android Studio.

No **Apêndice B** será apresentada a configuração do serviço Firebase.

No **Apêndice C** será apresentada a configuração da API Google Maps.

No **Apêndice D** será apresentado o conteúdo na íntegra das principais classes do aplicativo.

2. REFERENCIAL TEÓRICO

Neste capítulo será abordada toda a teoria envolvida na criação do aplicativo, como o sistema operacional Android e algumas das tecnologias utilizadas, como a API do Google Maps. Também será comentado sobre o minimundo educação física e a teoria que foi utilizada para a elaboração do aplicativo.

2.1 Android

Android é um sistema operacional para dispositivos móveis atualmente desenvolvido pela Google. Inicialmente, fundado em 2003 pela Android Inc, o sistema inicialmente tinha como planejamento melhorar os sistemas operacionais das câmeras digitais da época. Anos depois, em 2005, o Android foi então adquirido pela Google, que concordou em manter os principais desenvolvedores. Por ser baseado no *kernel* Linux, o sistema ainda poderia ser oferecido gratuitamente para fabricantes de aparelhos móveis (ANDROID_HIST, 2020).

Em 2007, a Google deu outro grande passo, ao liderar a *Open Handset Alliance* (OHA), uma aliança entre as grandes empresas na área da telefonia, *hardware* e *software*, como HTC, Motorola, Qualcomm e Texas Instruments (ANDROID_HIST, 2020). Com um grupo de 84 empresas, a OHA cumpriu o seu objetivo principal ao desenvolver a versão 1.0 do Android, uma plataforma para dispositivos móveis que seja completa, aberta e gratuita (OHA, 2020).

2.1.1 Arquitetura do Android

Baseado no Linux, o sistema operacional Android é dividido em vários componentes, sendo os principais componentes: kernel do Linux, camada de abstração de *hardware*, Android Runtime (ART), bibliotecas C/C++ nativas, estrutura da Java API e aplicativos do sistema (PLATFORM_ARCH, 2020). A **Figura 1** ilustra os principais componentes citados.

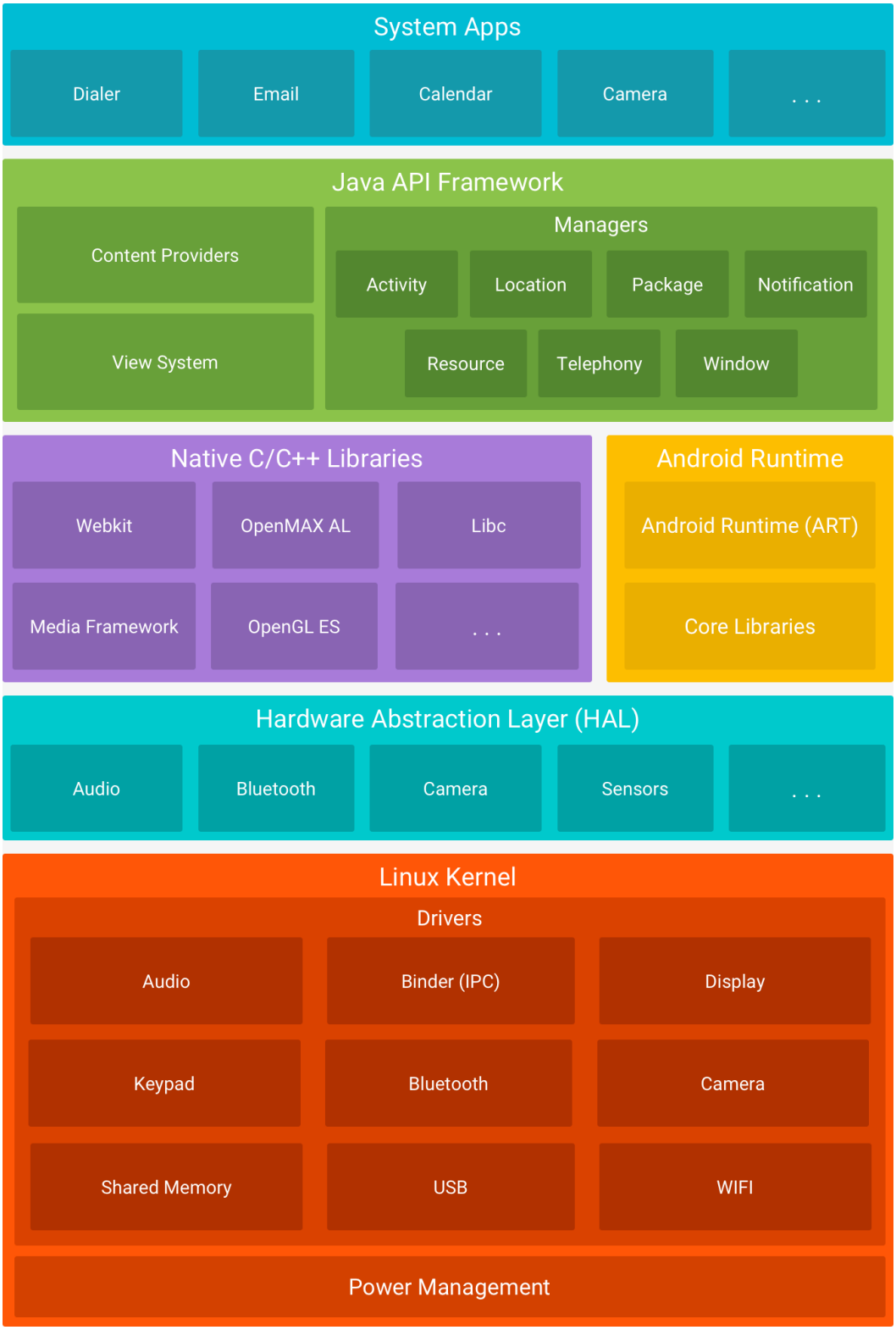

Figura 1 – Principais componentes Android (PLATFORM_ARCH, 2020)

Na base do sistema operacional Android está o *kernel* do Linux. Também chamado de núcleo, o *kernel* trabalha como uma ponte entre a camada física e a

pilha de *software* do Android (ANNUZZI, 2014). Dentre as tarefas que o *kernel* realiza estão: gerenciamento de memória e de processos, permissões dos aplicativos e também os componentes físicos como a tela, câmera, e armazenamento (ANNUZZI, 2014).

Acima do *kernel* está a camada de abstração de *hardware*. Também chamada de *Hardware Abstraction Layer* (HAL), a HAL é formada principalmente de módulos de biblioteca, fornecendo uma interface para componentes de hardware, como a câmera ou Bluetooth (PLATFORM_ARCH, 2020).

Em seguida, temos o Android Runtime. Ele é responsável por permitir que cada aplicação seja executada em um processo independente, dentro da sua própria máquina virtual, também chamada de *Virtual Machine* (VM). Baseada na máquina virtual Java, a máquina virtual Dalvik do Android tem um gasto de memória menor, maior otimização e pode ter várias instâncias executando ao mesmo tempo dentro de um dispositivo (ANNUZZI, 2014).

Tem-se também as bibliotecas C/C++ nativas. Elas são utilizadas por vários componentes e serviços principais do Android. Sua utilidade está em fornecer funcionalidades como acesso ao OpenGL ES para o desenho e manipulação de gráficos com 2 ou 3 dimensões no aplicativo, por exemplo (PLATFORM_ARCH, 2020).

Em seguida temos a estrutura da Java API. Nela, estão presentes todos os recursos do sistema operacional Android, através de APIs programadas na linguagem Java. Esses recursos podem ser utilizados pelo desenvolvedor para simplificar a implementação do seu aplicativo, pois já estão implementados e podem ser reutilizados. Dentre esses recursos que tanto o sistema quanto o desenvolvedor tem acesso estão: um gerenciador de notificação, que permite a exibição de alertas personalizados na barra de status; um gerenciador de recursos, que fornece acesso a recursos como *strings* e arquivos de *layout*; um sistema de visualização, que contém elementos visuais como botões, listas e caixas de texto (PLATFORM_ARCH, 2020).

Por último, estão os aplicativos do sistema. Dentre eles estão inclusos tanto os aplicativos do próprio Android, como e-mail e navegador de *internet*, como também os aplicativos de terceiros, que o próprio usuário instala (PLATFORM_ARCH, 2020).

2.2 API Google Maps

O Google Maps é um serviço desenvolvido pela Google, que dentre as principais funcionalidades, oferece uma visão por satélite e mapas das ruas de todo o mundo. A sua API atualmente se dividiu em várias, já que o Google Maps cresceu de tal forma que possui diversas funcionalidades, como mapas, lugares e rotas. Atualmente, devido as diferentes APIs que o sistema oferece, API Google Maps agora se chama Google Maps Platform (MAPS_PLAT, 2020).

Antes de entender como a API do Google Maps funciona, é necessário entender o que é uma API. A sua utilidade está em servir como uma ponte na comunicação entre o seu software e um outro serviço, sem precisar saber em detalhes como esse outro serviço foi implementado (RED_HAT_API, 2020). Essa comunicação se dá por regras e protocolos, que se usados de forma correta, permitem que o serviço que deseja ser utilizado responda a sua solicitação de forma adequada.

Outra principal utilidade de uma API, desta vez para o serviço que fornece essa ferramenta, é oferecer acesso aos seus dados sem comprometer a sua segurança, pois não oferece acesso direto de terceiros ao seu banco de dados. Além disso, é também uma forma de monetizar o acesso a esses dados, como ocorre com a API Google Maps.

Por ter várias APIs, a escolhida para este projeto foi a Maps *Software Development Kit* (SDK) para Android (MAPS_SDK, 2020). Com esta API é possível adicionar mapas do Google Maps na própria aplicação Android. A API é responsável por se comunicar com os servidores do Google Maps, obter dados, mostrar mapas e fazer com que os usuários consigam interagir com esses mapas (MAPS_SDK,

2020). Além disso, a API também permite outras interações como adições de pontos customizados no mapa, que no próximo capítulo será utilizada para a adição da localização de *personal trainers* no mapa.

Como exemplo de funcionamento da API Google Maps usaremos o aplicativo 99 (99, 2020). O 99 é um aplicativo de transporte individual que utiliza a API Google Maps para conectar passageiros e motoristas.

Ao ser inicializado, o aplicativo 99 coleta a localização do usuário e aguarda para que ele selecione o ponto de chegada no mapa fornecido pelo Google Maps. Com o ponto de chegada estabelecido, o Google Maps então encontra a melhor rota entre o ponto inicial e o ponto de chegada, e mostra essa rota ao usuário. Após a rota estabelecida, o aplicativo encontra o motorista disponível mais próximo e aguarda a confirmação do usuário para que o transporte seja iniciado. A **Figura 2** ilustra esse funcionamento na forma de 3 capturas de tela, sendo a primeira e segunda captura a posição inicial, e a terceira captura de tela o ponto de chegada, juntamente com a opção de confirmar o pedido.

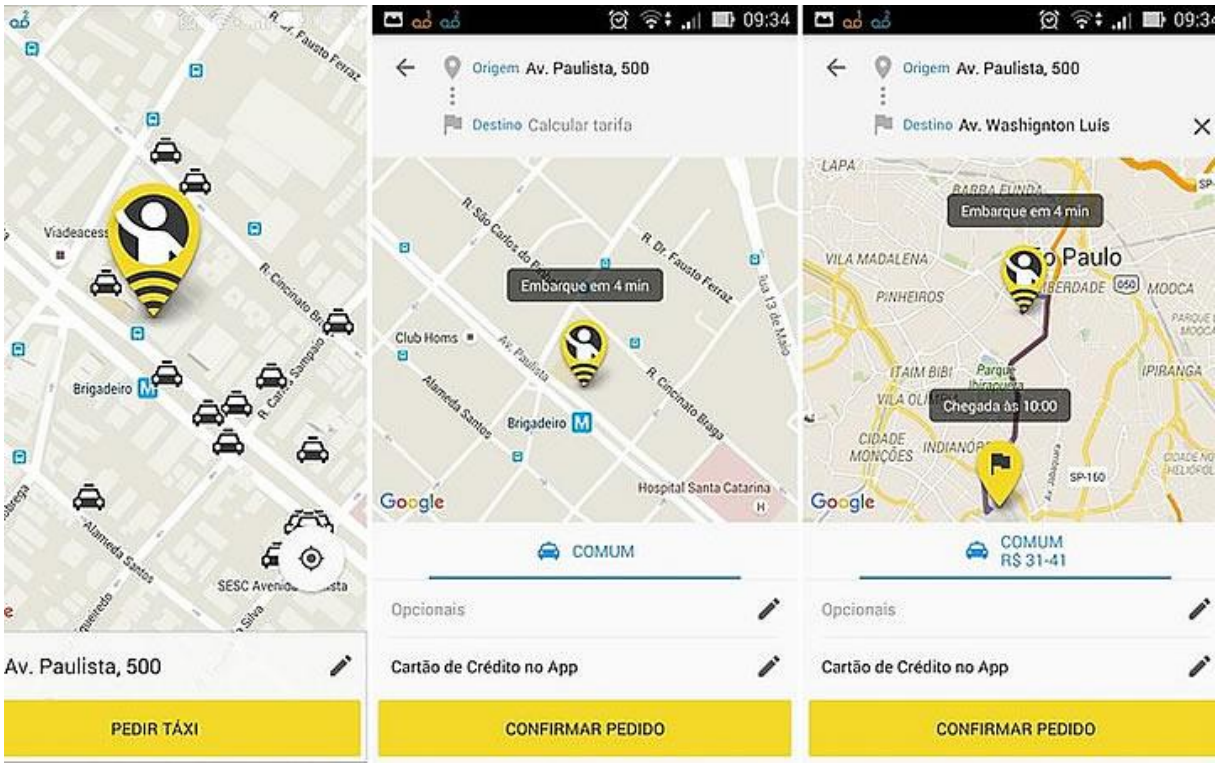

Figura 2 – Aplicativo 99 (99_FUNCIONAMENTO, 2020)

2.3 Serviços de Banco de Dados na Nuvem

Para armazenar os dados dos usuários e do aplicativo, decidiu-se por usar um banco de dados na nuvem, na forma de serviço. Também chamado de *Database-as-a-Service* (DBaaS), um DBaaS é um serviço de banco de dados oferecido por um provedor, normalmente na forma de assinatura baseada em taxas (ORACLE_CLOUD_DB, 2020).

Um banco de dados na nuvem quer dizer que ele está localizado em um servidor externo e é acessado através da Internet, ao contrário de um banco de dados local (CLOUDFLARE, 2020). Desta forma, um DBaaS elimina os custos de se manter um servidor, já que o responsável por manter esse servidor é o provedor desse DBaaS. A **Figura 3** ilustra um exemplo de serviço na nuvem.

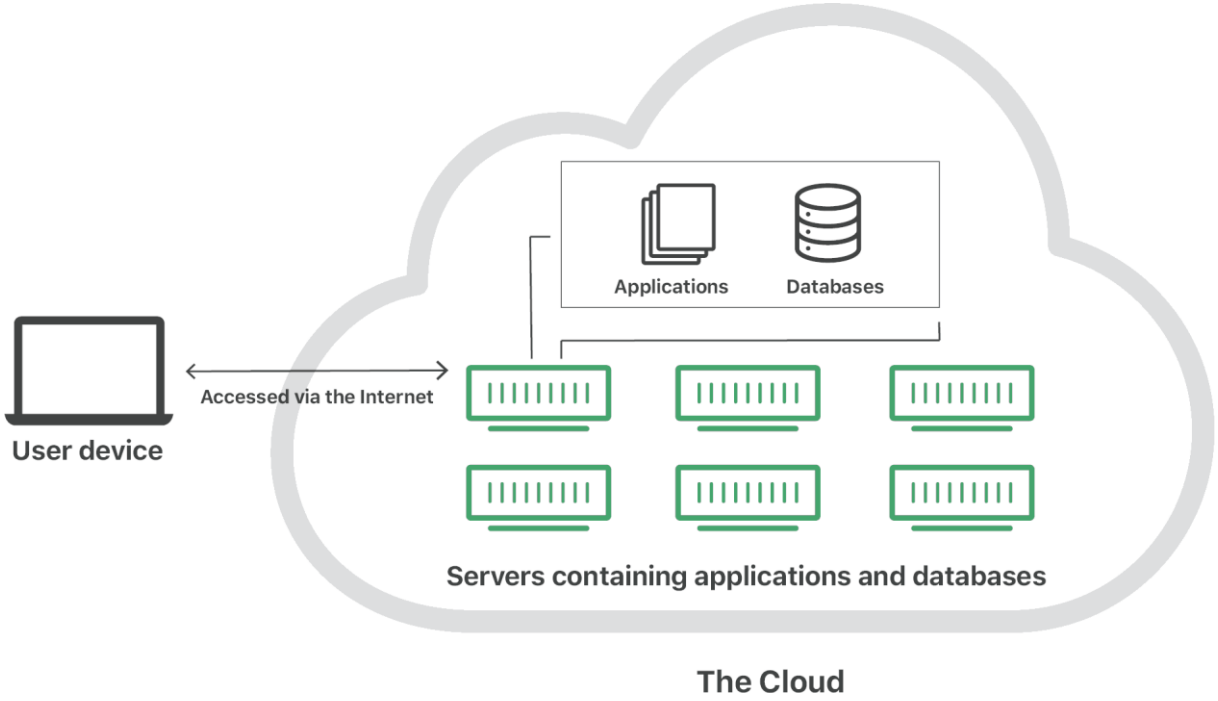

Figura 3 – Serviço na nuvem (CLOUDFLARE, 2020)

Com uma descrição de um banco de dados na nuvem e como ele funciona, é necessário aprofundar e descobrir como a computação na nuvem funciona no geral.

Como veremos adiante, a computação na nuvem abrange mais do que o modelo DBaaS.

2.3.1 Computação na Nuvem

Os bancos de dados, e a computação na nuvem num geral, funcionam por meio da virtualização. A criação de máquinas virtuais permite que vários sistemas operacionais sejam simulados ao mesmo tempo, dentro de uma única máquina física a qual fornece os recursos como memória e processamento. Isso permite que cada máquina virtual se comporte como um servidor físico individual, podendo ter ou não contato com as outras máquinas virtuais (VMWARE, 2020). Desta forma, as prestadoras de serviços podem criar várias máquinas virtuais e oferecer os seus recursos para os mais variados clientes (CLOUDFLARE, 2020).

De modo mais abrangente, um DBaaS é apenas um dos tipos de serviços do modelo de computação na nuvem. Apesar das máquinas virtuais descritas acima também serem utilizadas como servidores de bancos de dados, os principais modelos de serviço oferecidos pela computação na nuvem são: *Infrastructure-as-a-Service* (IaaS), *Platform-as-a-Service* (PaaS) e *Software-as-a-Service* (SaaS) (CLOUDFLARE, 2020). A **Figura 4** ilustra esses três modelos de serviço na nuvem.

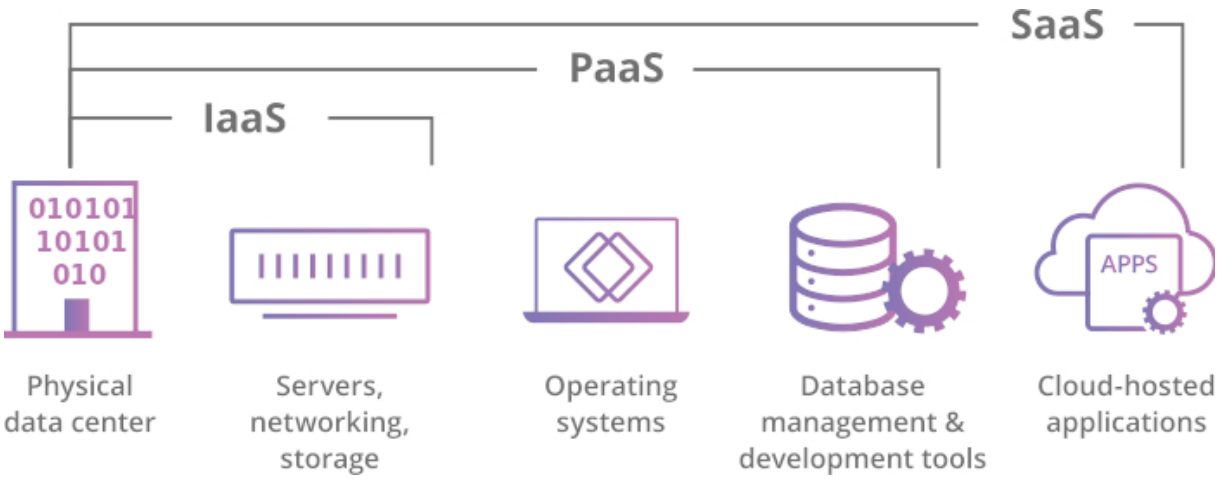

Figura 4 – Modelos de serviço na nuvem (CLOUDFLARE, 2020)

Um IaaS oferece serviços básicos de infraestrutura, acessados através da Internet. Dentre os serviços estão incluídos *hardware* e sistemas operacionais. Desta forma, o cliente tem acesso total a um servidor, por exemplo, com o poder de adaptar esse servidor para as suas necessidades. O propósito de um IaaS é fazer com que os clientes tenham controle total tanto da utilização quanto da configuração do serviço contratado (ERL, 2013). Com isso, a responsabilidade de administrar os recursos contratados são do cliente, já que os recursos oferecidos pelo provedor de um IaaS não vêm pré-configurados (ERL, 2013). A **Figura 5** ilustra um exemplo de serviço IaaS.

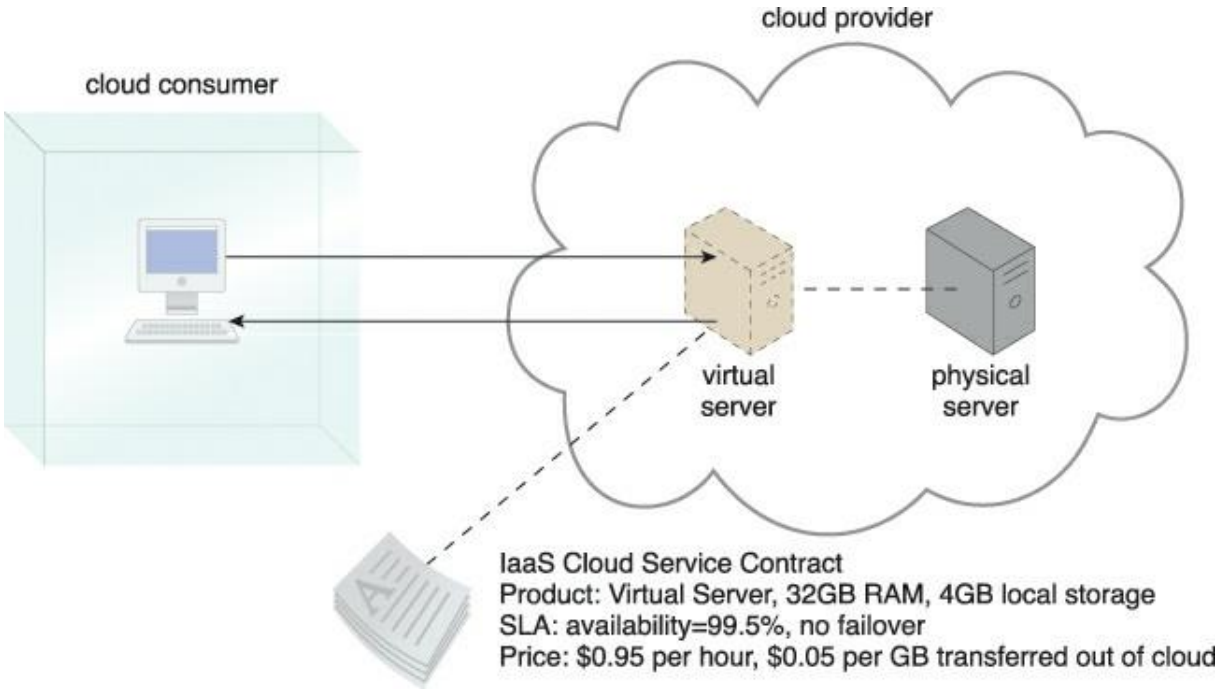

Figura 5 – Exemplo de serviço IaaS (ERL, 2013)

Uma versão à frente do IaaS é o PaaS. O modelo PaaS representa um ambiente já configurado com os principais recursos necessários. Recursos como sistemas operacionais, *firewall* e armazenamento já estão incluídos e configurados no PaaS. O seu propósito é fazer com que o cliente gerencie apenas o próprio serviço ou aplicativo, e deixe a responsabilidade da infraestrutura por conta do provedor PaaS (AZURE_PAAS, 2020). A **Figura 6** ilustra um exemplo de serviço PaaS.

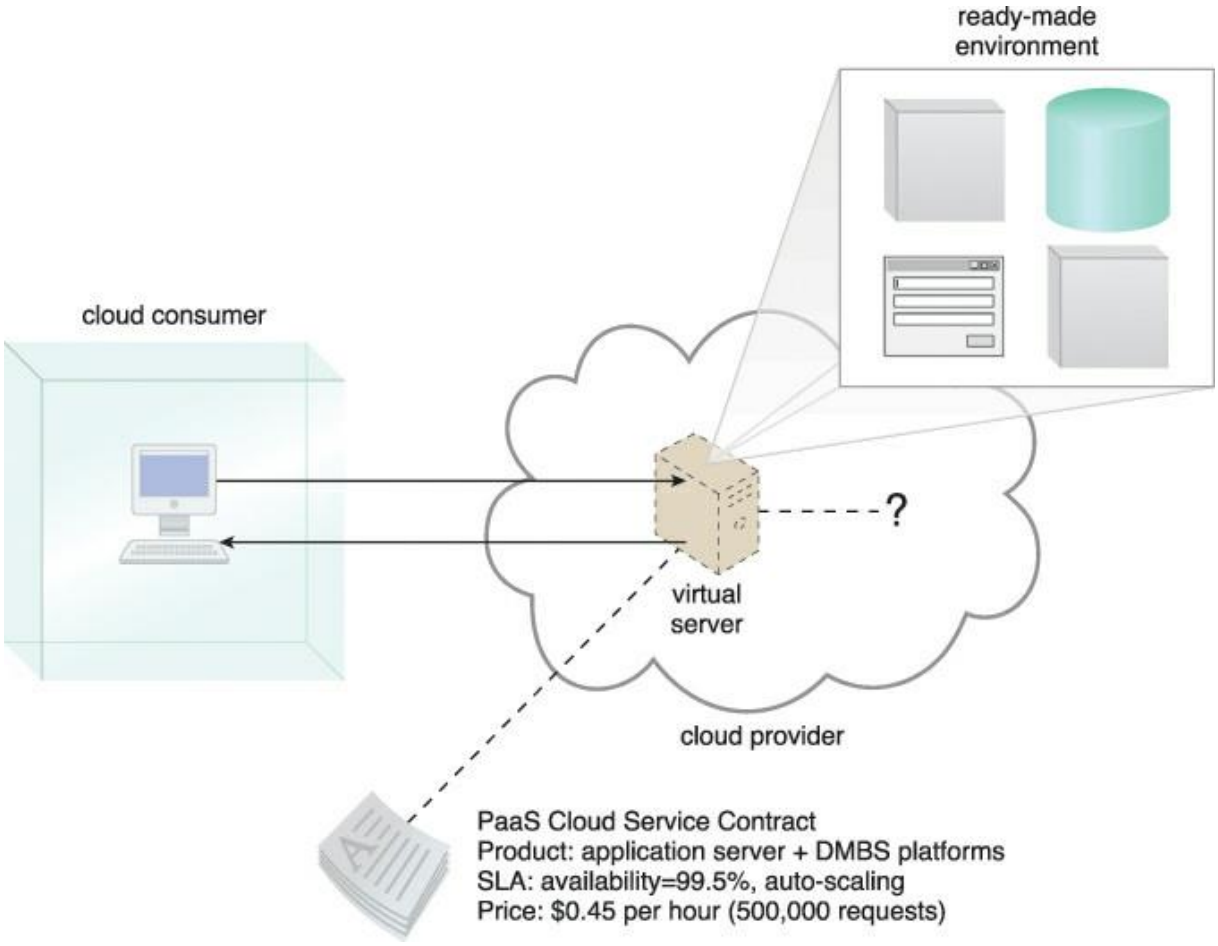

Figura 6 – Exemplo de serviço PaaS (ERL, 2013)

Por último, temos o modelo SaaS. Esse modelo constitui de *softwares* hospedados na nuvem, os quais são oferecidos como um produto. Também chamados de *softwares* sob demanda, eles são mantidos pelo provedor do serviço, que tem a responsabilidade de gerenciar a aplicação, mantendo a segurança e estrutura de dados (SF_SAAS, 2020). Dentre os exemplos de SaaS estão incluídos a rede social Linkedin (LINKEDIN, 2020), o serviço de *e-mails* Gmail (GMAIL, 2020) e o serviço de armazenamento Dropbox (DROPBOX, 2020). A **Figura 7** ilustra um exemplo de serviço SaaS.

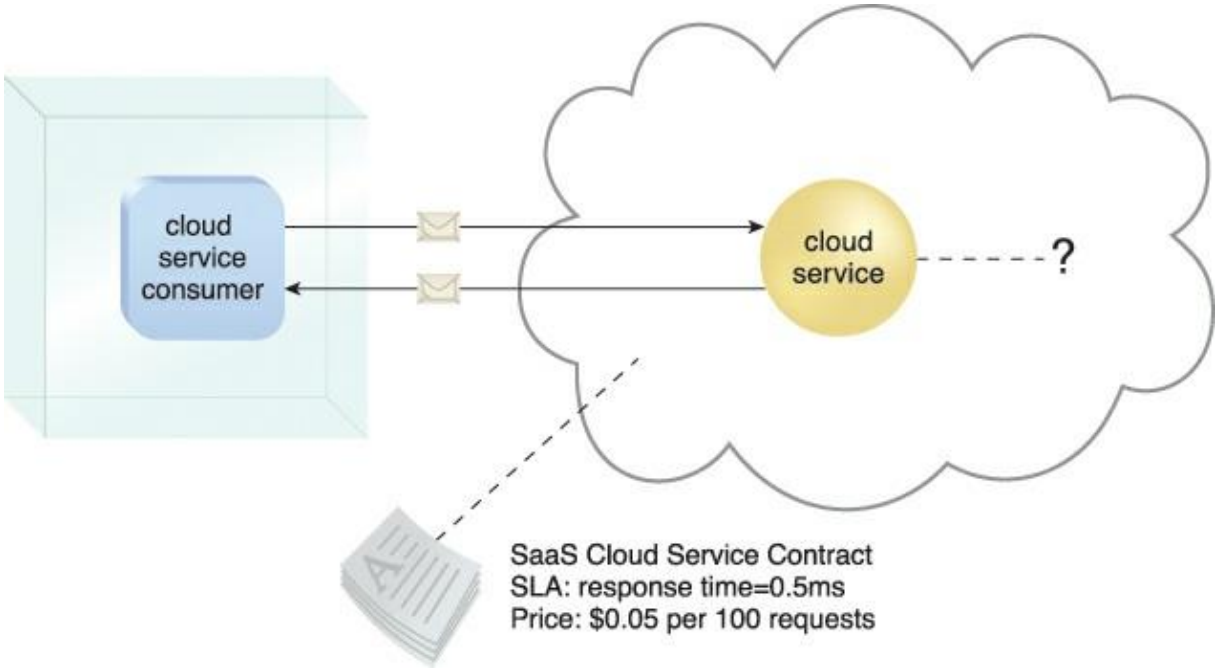

Figura 7 – Exemplo de serviço SaaS (ERL, 2013)

2.3.2 Principais Serviços de Banco de Dados na Nuvem

O serviço escolhido para este projeto foi o Firebase (FIREBASE, 2020). O Firebase é um dos serviços disponíveis que suprem a necessidade de um DBaaS no projeto. Há diversas alternativas ao Firebase, mas essencialmente todas possuem as mesmas funcionalidades, o que faz com que a decisão seja principalmente baseada no preço ou qualidade do serviço. As maiores e principais alternativas ao Firebase estão descritas a seguir.

- Heroku: o Heroku (HEROKU, 2020) é uma plataforma baseada na nuvem que permite desenvolvedores publicar e gerenciar as suas aplicações. Além disso, oferece a opção de ampliar os recursos usados pela aplicação caso tenha um aumento na demanda, por exemplo. Seus planos variam de gratuitos, para projetos pessoais, até planos avançados de no mínimo 250 dólares mensais.
- AWS Amplify: fornecido pela Amazon, o AWS Amplify (AWS, 2020) é uma das opções mais robustas. Suas ferramentas e serviços incluem autenticação, APIs, armazenamento e estatísticas da aplicação. Seus

planos funcionam na forma de pagar apenas pelo que usar, conforme a demanda.

- Parse: uma das opções gratuitas e de código aberto, o Parse (PARSE, 2020) possui as mesmas funcionalidades básicas de um serviço na nuvem, como APIs e estatísticas de aplicação. O que difere dos outros serviços é a sua necessidade de ser implementado pelo desenvolvedor, ou seja, o próprio desenvolvedor deve ser responsável pela infraestrutura.
- Kinvey: sendo a sua maior desvantagem o seu custo, o Kinvey (KINVEY, 2020) oferece flexibilidade para os desenvolvedores na hora de configurar e gerenciar as suas aplicações. Possui diversas funcionalidades que a computação na nuvem pode proporcionar e é uma das opções mais populares.
- Firebase: a alternativa escolhida, principalmente por ser distribuída pela mesma empresa do Android, o Firebase é oferecido pela Google. O Firebase conta com banco de dados, autenticação simplificada, armazenamento na nuvem e hospedagem do aplicativo, caso necessário. Para este projeto, foi utilizada apenas a autenticação por *e-mail*, o banco de dados fornecido e o armazenamento.

2.4 Minimundo Educação Física

O propósito deste projeto é oferecer um aplicativo que conecte pessoas com *personal trainers*. A função do *personal trainer* é oferecer atendimento personalizado para os seus clientes no que diz respeito ao condicionamento físico, de acordo com as necessidades e limitações do cliente (PERSONAL_QUAL, 2020).

Antes de oferecer os seus serviços, o profissional deve ser licenciado pelo conselho regional de educação física (PERSONAL_QUAL, 2020). Essa licença é adquirida através do curso de bacharel em Educação Física, que ao final da graduação permite ao aluno um registro no Conselho Regional de Educação Física (CREF) (GUIA_ED_FISICA, 2020).

Com o seu registro feito, o *personal trainer* está legalmente apto para atuar na sua profissão, com a opção de trabalhar em academias ou oferecer atendimento individual, por exemplo. O foco do aplicativo é justamente auxiliar esses profissionais no seu atendimento individual. Como atendimento individual, tem-se não só o treinamento físico dos clientes, como também o seu cotidiano, rotina, hábitos alimentares e horários de sono (BIEHL_BOSSLE, 2008). O cliente, ao delegar essas responsabilidades para o profissional, permite que a sua qualidade de vida seja gerenciada pelo *personal trainer*.

2.4.1 Escopo do Trabalho do Profissional Dentro do Aplicativo

Neste projeto, o tipo de serviço oferecido pelos *personal trainers* dentro do aplicativo é na forma de treinamento físico, ou também chamadas de sessões de treino. Cada sessão de treino fica a critério do profissional responsável, mas normalmente ela é constituída de um aquecimento, a fase do treinamento e o seu final que pode ou não incluir o *cool down*, uma forma de diminuir a alta velocidade dos batimentos cardíacos ocasionadas pelo exercício.

Essencialmente, cada sessão de treino é composta por exercícios escolhidos pelo profissional. Cada exercício é realizado pelo cliente na sequência indicada e por um número de séries e repetições especificadas. A **Figura 8** ilustra um desses possíveis exercícios, o agachamento.

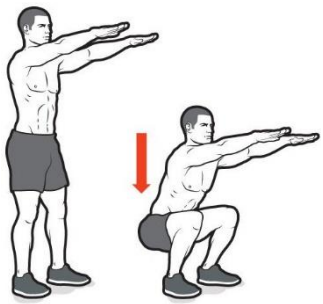

Figura 8 – Exemplo de exercício agachamento (ARMORED_FARMER, 2020)

Para compreender os termos usados acima, usaremos o agachamento ilustrado acima como exemplo. Uma repetição de um exercício é o movimento executado uma vez. Uma série são várias repetições executadas em sequência e de forma contínua. Um exercício pode conter várias séries, com tempo de descanso ou não entre elas. Uma sessão de treino é composta de um ou mais exercícios.

Ao serem contratados, os *personal trainers* seguirão esse formato de sessão de treinos dentro do aplicativo. O cliente então irá retornar para o profissional os dados da sessão realizada, como quantas repetições de um exercício conseguiu fazer, por exemplo. Desta forma, o *personal trainer* pode usar o seu critério de avaliação e desenvolver as próximas sessões. Esse retorno do cliente é essencial pois faz com que o profissional acompanhe a evolução do seu cliente e aprimore as suas futuras sessões de treino.

2.5 Aplicativos Existentes

Antes do desenvolvimento do aplicativo ConnectFit, foi feita uma pesquisa sobre os aplicativos semelhantes existentes no mercado. Obteve-se uma lista desses aplicativos, bem como as suas principais funcionalidades e limitações.

- BTFIT (2015): criado apenas para a utilização dos clientes, o BTFIT permite o aluno acessar programas de treinamento feitos por um *personal trainer* e aulas coletivas, de acordo com as necessidades do cliente (BTFIT, 2020). Todos os programas de treinamento e aulas coletivas são feitas por profissionais contratados pela própria BTFIT, não permitindo o cadastro de *personal trainers*, apenas clientes.

  Limitações: não permite cadastro de *personal trainers*, isso faz com que os únicos profissionais disponíveis sejam os da própria empresa por trás do aplicativo.
- GoTrainer: disponível para os habitantes de Singapura, permite o cliente buscar por *personal trainers* em Singapura e entrar em contato com eles. (GOTRAINER, 2020). A busca por profissionais pode ser filtrada pelo

gênero do *personal trainer*, tipo de treino, objetivos desejados, localização e preço.

Limitações: feito apenas para os habitantes de uma região e não permite que se faça uma busca por profissionais na sua própria região. Além disso, não há versão para dispositivos móveis.

- Mobitrainer: feito tanto para clientes quanto *personal trainers*, o aplicativo permite o gerenciamento de clientes por parte do profissional, incluindo sessões de treino (MOBITRAINER, 2020). O profissional deve fazer o cadastro no site e logo estará disponível para utilizar o aplicativo. A partir daí, os clientes que utilizam o aplicativo têm acesso as suas sessões de treino feitas pelo profissional contratado, além de poderem trocar mensagens de texto. Além disso, ambos os usuários tem acesso ao seu histórico de treino e também a uma biblioteca de exercícios com imagens e vídeos de suas execuções.

  Limitações: cliente não tem acesso a lista de profissionais disponíveis. Além disso, o aplicativo é “privado”. O único modo do cliente fazer o cadastro no aplicativo e contratar um treinador é se o próprio treinador enviar um convite para o e-mail do cliente.
- MFIT Personal: semelhante ao Mobitrainer, foca no gerenciamento de clientes pelo profissional. Permite a criação de sessões de treino e armazenamento de arquivos (MFIT, 2020). Os clientes possuem acesso as sessões de treino recebidas pelo profissional para que possam realizá-los e caso queira, também tem acesso a uma biblioteca de exercícios como referência. O profissional ainda consegue receber pagamentos pelo próprio aplicativo, por meio de cartão de crédito ou boleto bancário.

  Limitações: cliente não tem acesso a lista de profissionais disponíveis. Além disso, o aplicativo é “privado”. O único modo do cliente fazer o cadastro no aplicativo e contratar um treinador é se o próprio treinador enviar um convite para o e-mail do cliente.
- Nexur: semelhante ao Mobitrainer, foca no gerenciamento de clientes pelo profissional. Permite a criação de sessões de treino e armazenamento de

arquivos bem como a opção do profissional oferecer o seu próprio aplicativo com seu nome para os alunos (NEXUR, 2020). Os clientes possuem acesso as sessões de treino recebidas pelo profissional para que possam realizá-los, bem como uma biblioteca de exercícios. O aplicativo ainda oferece gráficos de acompanhamento de resultados e evolução de medidas, agendamento de aulas e histórico de treinos. Além disso, o profissional ainda pode criar avaliações físicas baseadas nos modelos prontos que o aplicativo oferece e também pode receber pagamentos pelo aplicativo.

Limitações: cliente não tem acesso a lista de profissionais disponíveis. Além disso, o aplicativo é “privado”. O único modo do cliente fazer o cadastro no aplicativo e contratar um treinador é se o próprio treinador enviar um convite para o e-mail do cliente.

- Tecnofit: foca no gerenciamento de clientes, o Tecnofit vai além de atender os *personal trainers*. O aplicativo atende também a empresas na área *fitness*, como as academias e estúdios de dança. Dentre as suas funcionalidades estão: possui a possibilidade de receber pagamentos pelo aplicativo, bem como emitir nota fiscal eletrônica; possui acompanhamento dos treinos dos clientes, indicador de contratos e indicador de metas por vendedor ou faturamento; enquetes e pesquisas personalizadas; controle financeiro, como fluxo de caixa, contas a pagar ou receber, plano de contas e abertura e fechamento de caixa; aplicativo para o aluno, afim de ele poder acompanhar os treinos, avaliações físicas e também reservar vagas em aulas especiais (TECNOFIT, 2020).

  Limitações: cliente não tem acesso a lista de profissionais disponíveis. Além disso, o aplicativo é “privado”. O único modo do cliente fazer o cadastro no aplicativo e contratar um treinador é se o próprio treinador enviar um convite para o e-mail do cliente.
- Trei.no: foca no gerenciamento de clientes pelo profissional. O aplicativo vai além e permite o gerenciamento por estúdios e academias. Permite a criação de sessões de treino customizadas e também permite a customização do aplicativo caso o profissional queira distribuir o aplicativo

com o próprio nome e aparência (TREI_NO, 2020). Além disso, o aplicativo possui um módulo de impressão que permite imprimir os treinos em impressoras de cupom, presentes em várias academias. O aplicativo conta também com uma biblioteca de exercícios para referência e reservas de aulas. Os clientes possuem acesso as sessões de treino recebidas pelo profissional para que possam realizá-los e também podem receber todas as postagens que o profissional contratado fizer.

Limitações: cliente não tem acesso a lista de profissionais disponíveis. Além disso, o aplicativo é “privado”. O único modo do cliente fazer o cadastro no aplicativo e contratar um treinador é se o próprio treinador enviar um convite para o e-mail do cliente.

3. DESENVOLVIMENTO

Este capítulo é responsável por detalhar o projeto de desenvolvimento do aplicativo ConnectFit, voltado para os *personal trainers* que desejam ser mais facilmente encontrado pelos seus clientes. Será descrito desde o funcionamento da aplicação, até a sua parte estrutural. Posteriormente, as principais classes do projeto serão citadas e descritas brevemente, junto com o diagrama de caso de uso e o esquema do banco de dados.

3.1 Funcionamento

A busca pela melhora da qualidade de vida faz com que a procura por profissionais capacitados seja necessária. Seja um profissional de Educação Física para cuidar da performance física ou um médico para cuidar dos problemas de saúde, por exemplo. Assim como na busca pelo melhor médico que atenda as suas necessidades, a busca pelo melhor profissional de educação física que atenda as suas próprias necessidades se faz tão importante quanto, já que o seu trabalho terá grandes impactos na sua vida.

Com isso em mente, foi proposto o desenvolvimento de uma solução que permita esse contato entre a população geral e profissionais de Educação Física, mais especificamente os *personal trainers*.

O aplicativo ConnectFit consiste nesse contato entre os clientes e os *personal trainers*. Ao iniciar o aplicativo pela primeira vez, é necessário informar os dados para realizar o *login*. Caso o usuário não esteja cadastrado, ele deve informar seu nome, *e-mail*, senha e também se é um cliente ou um *personal trainer*. Esses dois tipos de contas são as mesmas até certo ponto, pois cada tipo de conta, cliente ou *personal trainer*, possuem as próprias funcionalidades exclusivas.

Caso a conta cadastrada seja a de um *personal trainer*, ele pode editar dados de sua conta, como foto, cidade, estado, um breve parágrafo sobre sua pessoa, o

seu registro no CREF, que como explicado no Capítulo 2, é o seu número de registro no Conselho Regional de Educação Física. O *personal trainer* também pode atualizar a sua localização, caso deseje, para que os outros clientes nas proximidades o encontrem. Além disso, cada *personal trainer* pode visualizar uma biblioteca de exercícios fornecidas pelo próprio aplicativo, que contém descrições e *links* para a execução de cada exercício. Além disso, cada *personal trainer* pode visualizar os contratos pendentes que requerem aprovação do próprio profissional e também os contratos ativos com cada cliente.

Também é possível para o *personal trainer* trocar mensagens de texto com os seus clientes, sejam eles clientes ativos ou não, e também atribuir treinos para os seus clientes com quem ele tem contrato ativo.

Caso a conta cadastrada seja a de um cliente, ele também pode atualizar seus dados, como foto, altura, peso, cidade e estado. Além disso, o cliente também tem acesso a mesma biblioteca de exercícios que o *personal trainer*, a fim de que ele tenha uma referência na hora de executar os exercícios passados pelo profissional contratado.

O cliente tem uma funcionalidade exclusiva que é buscar por *personal trainers*. A busca pode ser feita pelo nome do profissional, ou pode fazer uma busca baseada na sua localização. A busca pela localização coleta a localização do cliente, e mostra os profissionais mais próximos a ele, em um mapa fornecido pela Google. Ao selecionar um *personal trainer* na busca, o cliente pode então visualizar o seu perfil e ver se aquele profissional é o mais adequado para ele.

O cliente então pode contratar um profissional e também enviar mensagens de texto para ele. Ao contratar e o *personal trainer* aceitar o contrato, o cliente então estará disponível para receber os treinos passados por aquele profissional. Após a realização de cada treino, o cliente pode adicionar detalhes a ele, marcar o treino como concluído e enviar o treino de volta ao profissional, que irá visualizar e dar o *feedback* necessário.

Por fim, caso o *personal trainer* ou cliente deseja encerrar o contrato, essa opção está disponível para ambos. Ao escolher essa opção o contrato entre os dois se encerra, bem como a possibilidade de enviar e receber os serviços prestados.

Para que essa aplicação funcione, além do aplicativo Android, há também o Firebase, que é o serviço de banco de dados na nuvem escolhido. A comunicação entre o aplicativo Android e o Firebase é feita via internet, pelo protocolo *Hyper Text Transfer Protocol Secure* (HTTPS). O banco de dados utilizado pelo Firebase é chamado Firebase Realtime Database (FIREBASE_DATABASE, 2020).

3.2 Ambiente de Desenvolvimento

No desenvolvimento do aplicativo Android ConnectFit, foi necessário primeiro configurar o ambiente de desenvolvimento Android. O SDK Android, que inclui o Android Studio (ANDROID_STUDIO, 2020), possui todas as ferramentas necessárias para o desenvolvimento de uma aplicação Android, desde a sua interface até o código necessário. Como linguagem de programação foi escolhida a linguagem Java (JAVA, 2020), que é totalmente suportada pelo SDK Android.

O Android Studio, ferramenta utilizada para o desenvolvimento do aplicativo Android ConnectFit, teve a sua versão 4.0.2 como a escolhida. Fornecida pela mesma empresa responsável pelo Android, a Google, o Android Studio conta com uma vasta gama de funcionalidades. Dentre elas estão: editor gráfico de interface, permitindo criar interfaces sem trabalhar com códigos; um editor de código inteligente, com funções de autocompletar e importação rápida de bibliotecas; Gradle, um sistema de compilação flexível e que permite várias opções para a compilação de vários dispositivos, por exemplo (ANDROID_STUDIO, 2020).

3.3 Descrição da Aplicação

O aplicativo, sempre que é iniciado pelo usuário, faz uma rápida verificação se há uma conta autenticada existente naquele aparelho. Para isso, utilizamos um método fornecido pelo Firebase de nome *getCurrentUser*. Este método verifica se há um usuário autenticado. Caso exista, o aplicativo mostra a tela principal. Senão, o aplicativo mostra a tela de *login*.

3.3.1 Login

A tela de login contém os campos de *e-mail* e senha, para que sejam fornecidos pelo usuário. Além disso, também há um *link* que redireciona para a tela para recuperar a senha do usuário, e um *link* que redireciona para a tela de cadastro de usuário, ambas as telas explicadas nas próximas seções. Com os campos de *e-mail* e senha preenchidos pelo usuário, o usuário clica em *login* e esses dois campos são enviados para o Firebase, para que realize a autenticação. A comunicação entre o aplicativo e o Firebase para fazer o *login* está ilustrada na **Figura 9**.

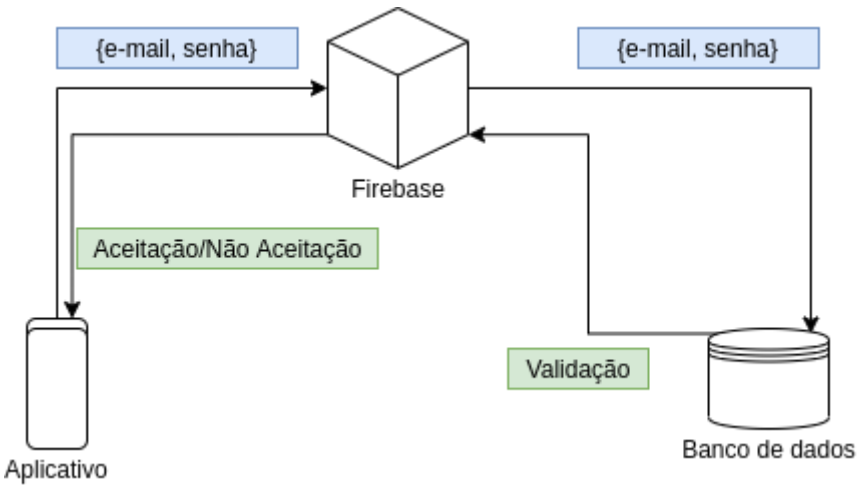

Figura 9 – Processo de login entre o aplicativo e o Firebase

Com o retorno da solicitação, o aplicativo decide se ele redireciona para a tela principal do aplicativo caso o *login* tenha sido um sucesso, ou se mantém na tela de *login* caso o procedimento tenha falhado.

3.3.2 Recuperação da Senha

A tela para a recuperação da senha serve para que o usuário que tenha perdido a sua senha, possa cadastrar uma nova senha para a sua conta. Para isso, o usuário deve inserir o seu *e-mail* cadastrado, e clicar no botão de enviar para fazer a solicitação ao Firebase. A comunicação entre o aplicativo e o Firebase para fazer essa solicitação para recuperar a senha está ilustrada na **Figura 10**.

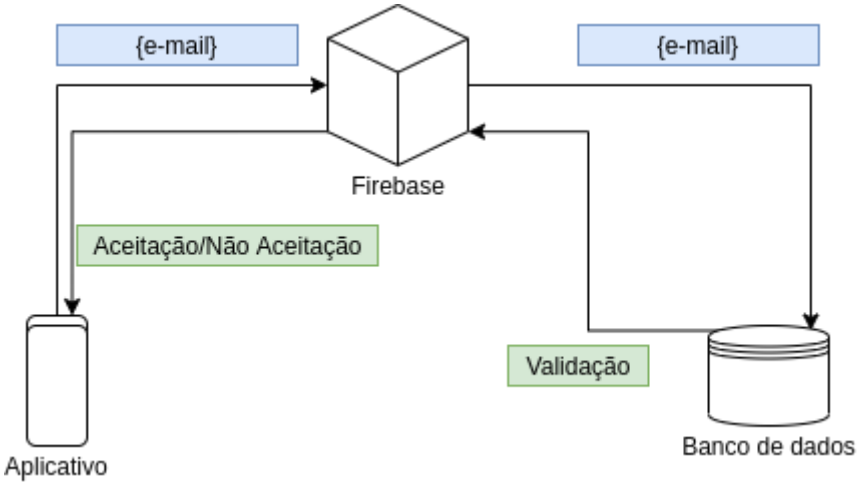

Figura 10 – Requisição para a recuperação da senha

Ao enviar o *e-mail* que deseja recuperar a senha, o Firebase verifica se é um *e-mail* válido, gera um link para o usuário cadastrar uma nova senha, e envia esse *link* para o *e-mail* que o usuário utilizou ao fazer a solicitação acima. Com esse *link* o usuário poderá então cadastrar uma nova senha para a sua conta, e fazer o *login* normalmente.

3.3.3 Cadastro de Usuários

Caso o usuário não tenha uma conta no aplicativo, ele tem a opção de cadastrar uma nova conta. Na tela de *login*, há um *link* onde é possível redirecionar para uma tela de cadastro em que o usuário será requisitado um nome, *e-mail*, senha e o tipo de conta. Ao clicar para enviar, o Firebase primeiro recebe apenas o *e-mail* e a senha, e retorna se houve sucesso ou falha na operação. A comunicação

entre o aplicativo e o Firebase para fazer essa solicitação para cadastrar um *e-mail* e senha está ilustrada na **Figura 11**.

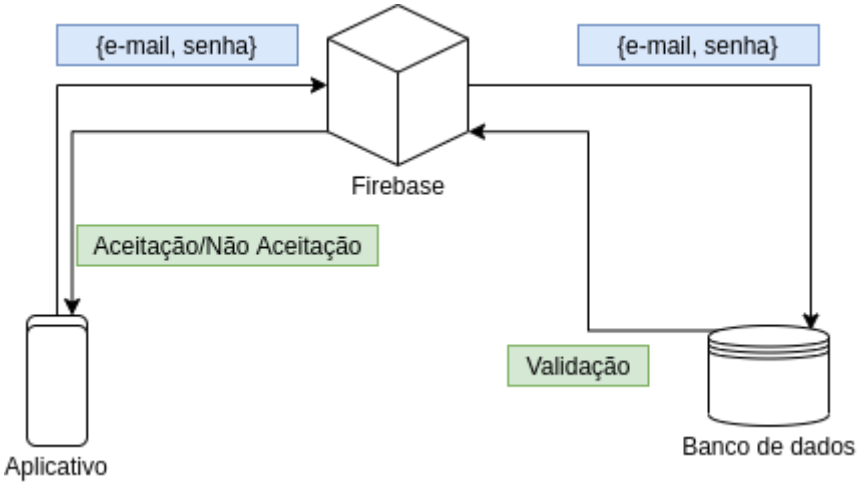

Figura 11 – Envio do e-mail e senha para cadastro

Caso houver sucesso no cadastro, esse usuário criado é automaticamente autenticado no aparelho do usuário. Em seguida, o aplicativo envia os outros campos da tela de cadastro e mais alguns atributos em branco, que serão editados pelo usuário apenas posteriormente. São eles o *id* do usuário, nome do usuário, nome do usuário sem acentos, tipo de conta, *link* para imagem de perfil, altura do usuário, peso do usuário, parágrafo sobre o usuário, número do registro CREF, cidade e estado do usuário. Alguns desses atributos serão utilizados apenas pelo tipo de conta cliente, e outros serão utilizados apenas pelo tipo de conta profissional. A comunicação entre o aplicativo e o Firebase para fazer essa solicitação para cadastrar os atributos citados anteriormente está ilustrada na **Figura 11**. Na **Figura 12**, compilamos os atributos que acabamos de citar anteriormente e chamamos apenas de “Dados do usuário compilados”.

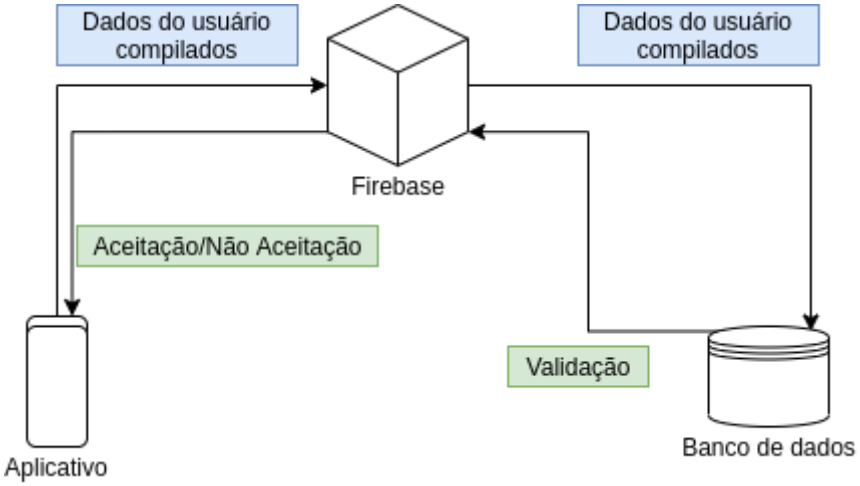

Figura 12 – Envio dos dados secundários para cadastro

Após a validação do cadastro acima, o usuário é redirecionado para a tela de edição de conta, onde ele pode inserir mais dados sobre ele, se quiser, como a cidade e estado onde mora.

3.3.4 Edição de Conta

Nessa tela é possível editar alguns atributos da conta do usuário. São eles: foto de perfil; CREF, que é o número de registro do profissional de Educação Física; sobre, que é um breve texto disponível para as contas de *personal trainer*, onde ele escreve sobre a sua experiência profissional; altura e peso, atributos disponíveis para os clientes; cidade e estado onde mora o cliente ou profissional; atualização da localização, para os profissionais que desejam ser encontrados pela busca por localização.

Para escolher uma nova foto de perfil, basta o usuário clicar na foto atual na tela de edição de conta. Ao clicar sobre a foto na tela de edição de conta, o usuário pode selecionar uma foto armazenada na galeria do celular, e enviar ao Firebase, que armazena essa foto no seu servidor. O aplicativo então armazena o *link* para essa foto no banco de dados. A **Figura 13** ilustra esse procedimento.

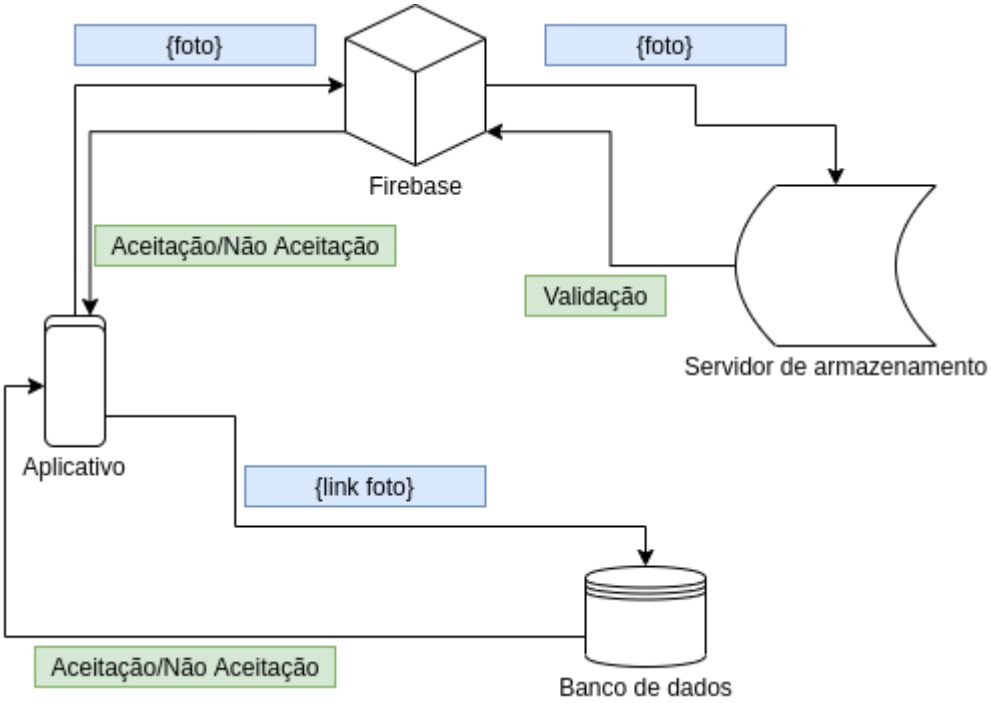

Figura 13 – Upload da foto de perfil

Para os outros campos, basta o usuário preencher e clicar no botão para salvar. O aplicativo então compila esses atributos editados pelo usuário e envia ao Firebase, que se encarregará de atualizar o banco de dados com os novos valores. Esse procedimento é realizado da mesma forma exemplificada na **Figura 12**, apenas com campos diferentes desta vez.

Caso o envio desses dados tenha sido um sucesso, redireciona o usuário para a tela principal do aplicativo. Nela, há 6 *links* na forma de botões que redirecionam para diferentes telas do aplicativo. Alguns desses *links* são exclusivos para o *personal trainer* e outros apenas para os clientes. Os links são: edição de conta (esta seção), profissionais disponíveis, contratos pendentes, contratos ativos, exercícios e chats. Cada *link* será explicado a seguir.

3.3.5 Profissionais Disponíveis

Disponível apenas para os clientes, essa seção permite que o usuário busque por profissionais. Inicialmente, o aplicativo mostra a lista de todos os profissionais

cadastrados. Para isso, o aplicativo primeiro faz uma solicitação ao Firebase por todos os usuários do seu banco de dados. A **Figura 14** ilustra a solicitação que o aplicativo faz ao Firebase pelos usuários.

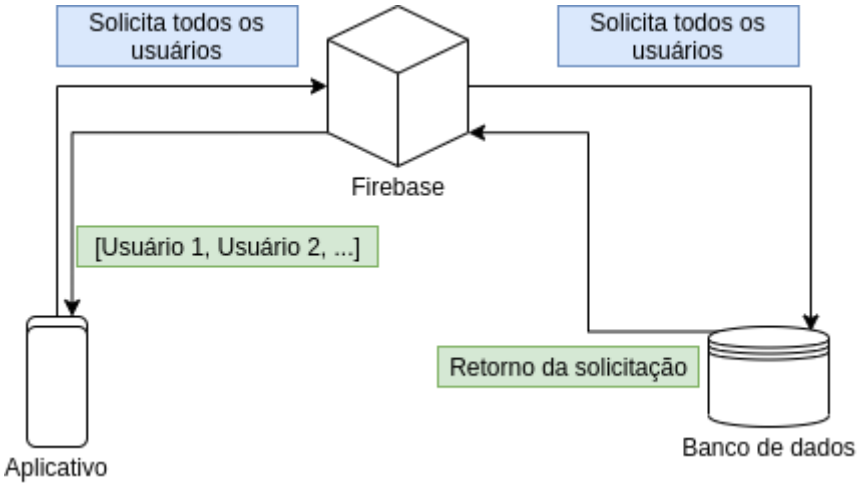

Figura 14 – Solicitação ao Firebase por todos os usuários cadastrados

Com a lista de todos os usuários, o aplicativo então filtra esses usuários e seleciona apenas os usuários do tipo *personal trainers*. O aplicativo então mostra esses profissionais na tela.

3.3.6 Busca de Profissionais Disponíveis pelo Nome

Caso o cliente deseja buscar pelo nome do profissional, há uma barra de pesquisa disponível logo acima da lista de profissionais descrita na seção anterior. Nela, o usuário insere um nome e o aplicativo automaticamente coleta esse nome e faz novamente uma busca pelos usuários, mas dessa vez com o nome inserido pelo cliente. A **Figura 15** ilustra essa solicitação ao Firebase pelos usuários cujos nomes começam com o termo de pesquisa inserido pelo cliente.

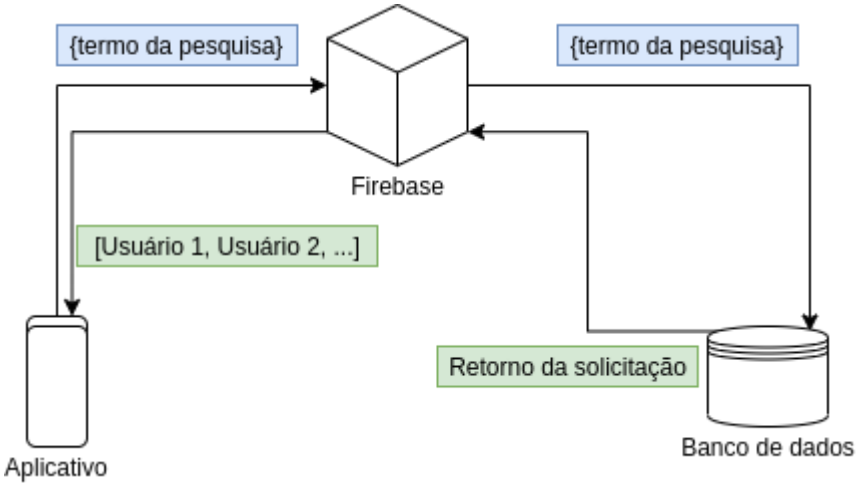

Figura 15 – Pesquisa de usuários pelo nome

Essa solicitação retorna uma lista com todos os usuários cujo nome condizem com o termo de pesquisa inserido pelo usuário. Com essa lista de usuários, o processo do aplicativo de mostrar os usuários é o mesmo. O aplicativo filtra todos os usuários que são profissionais e mostra esses usuários na tela.

3.3.7 Busca de Profissionais Disponíveis pela Localização

Há ainda a opção de fazer uma busca por profissionais geograficamente próximos ao cliente. Nessa busca, primeiramente com a API do Google Maps devidamente configurada, como será mostrado no apêndice C, faz-se uma solicitação pelo mapa. Os servidores do Google Maps então retornam esse mapa, que será mostrado na tela e posteriormente povoado com a localização dos *personal trainers*.

Com o mapa pronto, o aplicativo faz uma solicitação ao Firebase por todos os *personal trainers*, como ilustrado na **Figura 14**. Após o aplicativo armazenar todos esses usuários em uma lista, o aplicativo coleta a localização do usuário autenticado. Faz-se uma verificação caso o usuário esteja com a localização ligada e em seguida o aplicativo faz essa coleta. Com todos os profissionais armazenados em uma lista e a localização do cliente autenticado, o aplicativo mostra todos esses pontos no mapa. Primeiramente, coloca-se a localização do cliente no mapa. Em

seguida, a lista de profissionais é percorrida e verificada caso esses profissionais tenham a localização cadastrada. Caso tenham, insere esse ponto no mapa.

3.3.8 Perfil do Usuário

Em qualquer das buscas especificadas na seção anterior, ao clicar em qualquer dos profissionais, o usuário é redirecionado para a sua tela de perfil correspondente. Nela, o aplicativo faz uma solicitação ao Firebase pelos dados do usuário clicado, utilizando o seu *id*. A **Figura 16** ilustra a requisição pelo *id* do usuário.

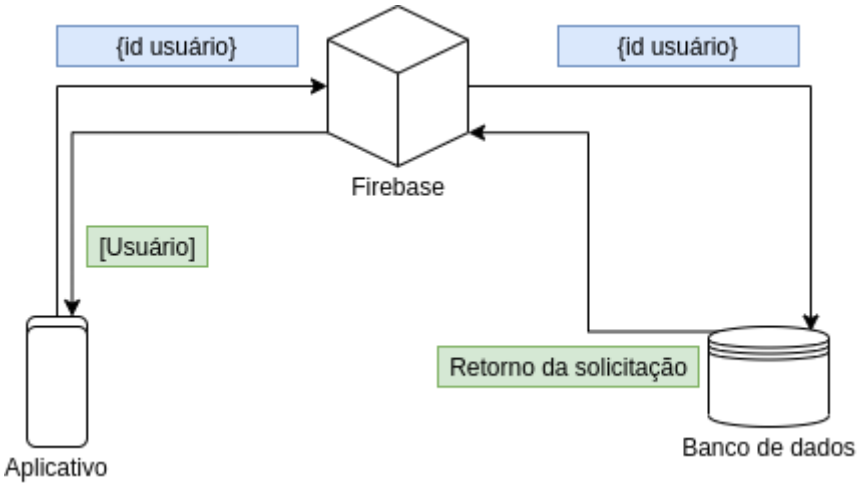

Figura 16 – Pesquisa por usuário específico

Com os dados do usuário retornados, o aplicativo mostra esses dados do profissional na tela. Nessa tela do perfil do profissional, o cliente além de poder conferir detalhes da conta que está visitando, também pode contratar/dispensar o profissional, enviar mensagem de texto e conferir os treinos passados por ele, caso o tenha contratado.

3.3.9 Contratos Pendentes

Apesar de ambos os tipos de contas terem a opção de encerrar o contrato a qualquer momento, o contrato para ser iniciado depende da aprovação do profissional. O cliente ao escolher contratar aquele profissional, ainda necessitará da aprovação dele. Até lá, o contrato ficará marcado como pendente.

Essa é uma tela disponível para ambos os tipos de conta. Nela, o aplicativo solicita ao Firebase por todos os contratos com status de pendente para aquele usuário autenticado. O aplicativo então armazena essa lista de contratos e mostra os usuários com quem o usuário autenticado tem contrato pendente. Caso o usuário autenticado seja um cliente, o aplicativo apenas mostra que o contrato está pendente, já que a mudança de pendente para ativo depende do profissional. O cliente pode apenas clicar em qualquer um dos perfis que aparecerem na tela e ele será redirecionado para o perfil daquele profissional.

Caso o usuário autenticado seja um *personal trainer*, o aplicativo mostrará a lista de clientes que o desejam contratar e aguardam aprovação. Ao clicar em qualquer dos perfis na tela, o profissional é redirecionado para a tela de perfil daquele cliente, que funciona da mesma forma que a tela de perfil do profissional descrita anteriormente. Nesse caso, o profissional pode aceitar a solicitação do cliente apertando no botão de aceitar, e só então o contrato é iniciado. Para que ele seja iniciado, o aplicativo envia uma confirmação com o *id* do usuário cliente, *id* do profissional e o *status* de ativo para o Firebase, que atualiza o contrato no banco de dados e muda o seu *status* para ativo. A **Figura 17** ilustra o procedimento de mudar um contrato de *status*.

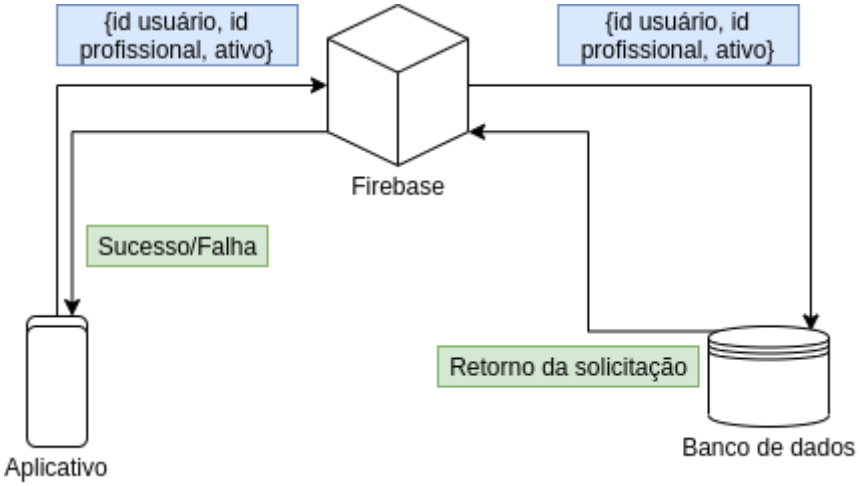

Figura 17 – Mudança de status de um contrato

3.3.10 Contratos Ativos

Essa tela funciona da mesma maneira que a tela de contratos pendentes explicada na seção anterior. Mas em vez de solicitar os contratos com *status* pendentes, o aplicativo solicita ao Firebase os contratos com *status* ativo. Para isso, o aplicativo manda junto com a solicitação o *id* do usuário autenticado. A **Figura 18** ilustra essa solicitação por todos os contratos com o *id* fornecido.

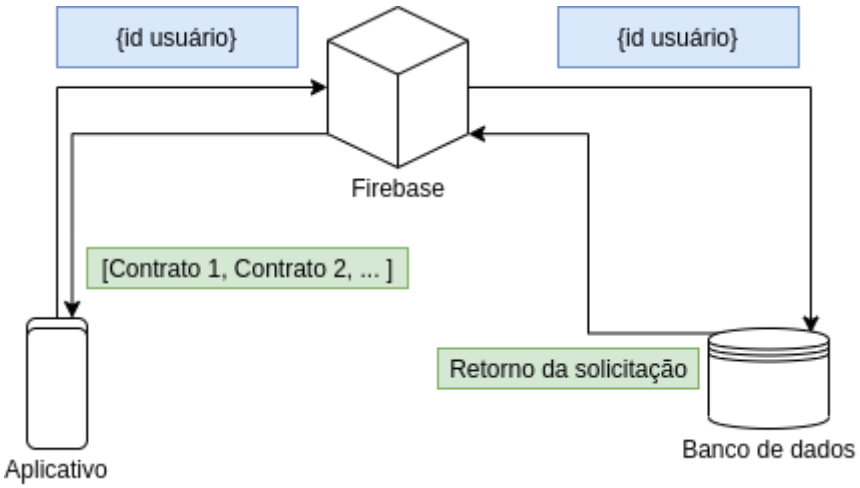

Figura 18 – Pesquisa por contratos do usuário

Com a lista de contratos para aquele usuário, o aplicativo então filtrará e selecionará apenas os contratos ativos para serem mostrados ao usuário.

3.3.11 Visualização de Treinos

Com o contrato ativo, ambas as partes podem visualizar os treinos feitos por aquele *personal trainer*. Para isso, o usuário deve clicar em um dos usuários com quem tem o contrato ativo, e o botão para visualizar os treinos estará disponível. O usuário é redirecionado para a tela com os treinos referentes àquele contrato. A solicitação do aplicativo pelos treinos ao Firebase é feita no mesmo modelo das solicitações anteriores. Mas ao invés de pedir pelos contratos, o aplicativo pede pelos treinos, e passa o *id* do profissional e o *id* do cliente. O Firebase faz a busca no banco de dados e retorna todos os treinos com esses usuários.

Caso a conta que esteja visualizando os treinos seja de um cliente, ele poderá apenas visualizar e editar os treinos disponíveis. Caso a conta seja a de um profissional, além das funcionalidades citadas anteriormente ele também pode criar novos treinos para o seu cliente.

3.3.12 Adição de Treinos

Para que um profissional adicione um novo treino para o seu cliente, o mesmo deve clicar no botão de adicionar e então será redirecionado para a tela onde poderá inserir nome, data e os detalhes do treino, além de uma opção para marcar o treino como pendente ou concluído. A **Figura 19** ilustra o envio dessas informações de treino do aplicativo para o Firebase, que guardará essas informações no banco de dados.

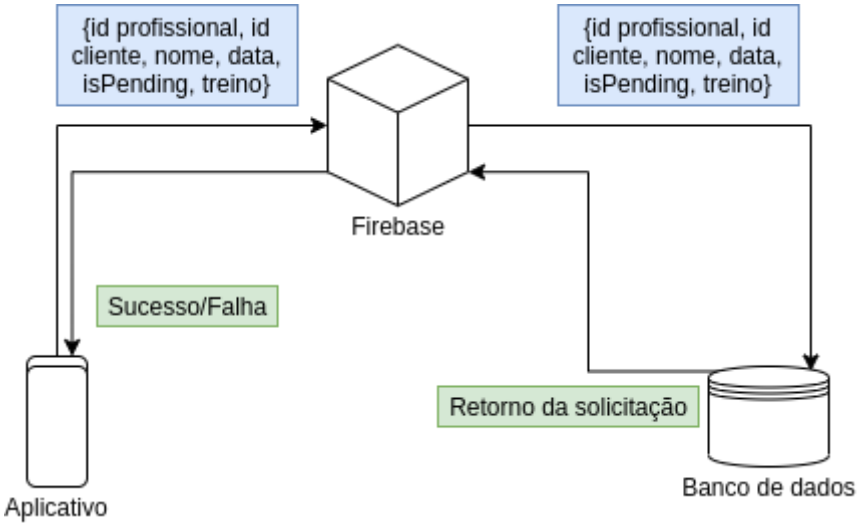

Figura 19 – Envio de um treino específico para o Firebase

Após criado o treino, tanto o profissional quanto o cliente podem visualizar e também editar os treinos, caso necessário. A tela de edição do treino é a mesma da tela de adição do treino. Mas antes de enviar a solicitação, o aplicativo faz uma rápida verificação para saber se é uma solicitação para editar o treino ou não. Com isso o aplicativo sabe se deve inserir um treino novo ou atualizar um treino já existente, que é o caso da edição do treino.

3.3.13 Exercícios

Essa é uma lista de exercícios inseridas manualmente no Firebase e que é acessada pelo aplicativo. A solicitação feita pelo aplicativo ao Firebase é a mesma descrita nas figuras anteriores. O aplicativo faz uma solicitação por todos os exercícios e então esses exercícios são mostrados ao usuário pelo aplicativo. Ao clicar em qualquer um dos exercícios o usuário é redirecionado para uma tela onde o aplicativo faz uma solicitação por dados daquele exercício, passando o *id* daquele exercício, onde retornará o nome, descrição e *link* para o vídeo da execução daquele exercício. Com esses dados, o aplicativo então mostra esses dados. Essas duas solicitações feitas são feitas da mesma forma que as solicitações descritas nas seções anteriores, mudando apenas os parâmetros enviados pelo aplicativo.

3.3.14 Chats

Essa funcionalidade é responsável pela troca de mensagens entre clientes e profissionais. A tela responsável pelos *chats* mostra todos os *chats* que o usuário tem com seus clientes, caso seja um *personal trainer*, ou mostra todos os *chats* com os *personal trainers*, caso seja um cliente. Para isso, o aplicativo faz uma solicitação ao Firebase e passa o *id* do usuário autenticado. O Firebase então retorna todos os *chats* com aquele usuário presente. A **Figura 20** ilustra essa solicitação.

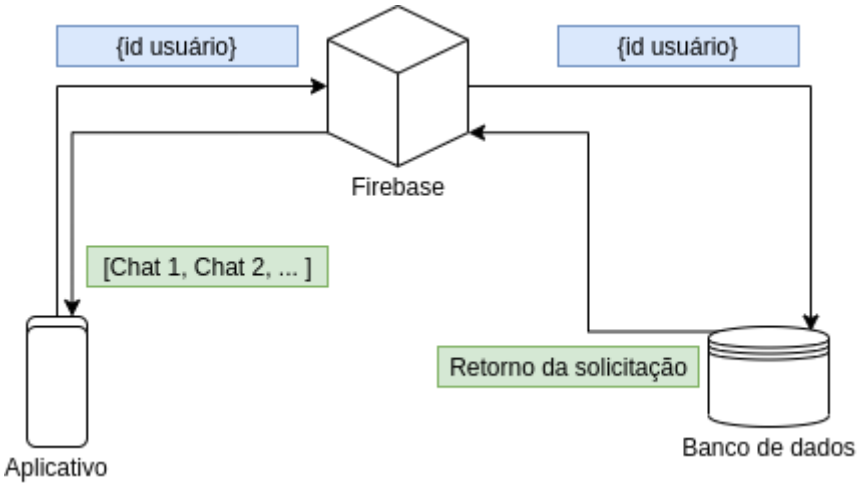

Figura 20 – Pesquisa por todas as conversas de um usuário

Com a lista de *chats* em mãos, o aplicativo então faz uma solicitação ao Firebase por todos os usuários. Esse passo é necessário para cruzar as informações da lista de *chats* com a lista de usuários. Como a lista de *chats* contém só o *id* dos usuários envolvidos, utilizaremos esses *ids* para coletarmos mais informações desses usuários através dessa última solicitação. Com os dados dos usuários salvos, o aplicativo então mostra cada usuário com quem o usuário autenticado trocou mensagens. Ao clicar em qualquer um dos usuários, o usuário é redirecionado para a tela de troca de mensagens com aquele usuário.

3.3.15 Troca de Mensagens

Para iniciar um *chat* o cliente pode visitar o perfil de qualquer *personal trainer*, das diversas maneiras explicadas acima, e clicar no botão de *chats*. O cliente então será redirecionado para a tela de troca de mensagens com aquele profissional. Um detalhe importante é que como a busca por usuários no aplicativo é feita somente pelos clientes, um *personal trainer* não pode pesquisar por todos os clientes cadastrados no aplicativo, e portanto não pode iniciar uma conversa. A iniciativa vem do cliente, que ao buscar por profissionais pelo nome ou localização poderá iniciar uma conversa.

A tela de troca de mensagens contém um campo onde o usuário pode digitar a sua mensagem e um botão para enviar a mensagem. Logo acima há a lista de todas as mensagens enviadas. O aplicativo faz uma solicitação ao Firebase e passa o id do usuário com quem é feita aquela conversa. O Firebase retorna os dados daquele usuário e o aplicativo usa essas informações para mostrar a foto e nome do usuário.

Ao digitar e enviar uma mensagem o aplicativo envia o *id* do remetente, o *id* do destinatário e o conteúdo da mensagem para o Firebase, que se encarregará de salvar a mensagem. A **Figura 21** ilustra o envio da mensagem do aplicativo para o Firebase, que se encarregará de salvar essa mensagem no banco de dados.

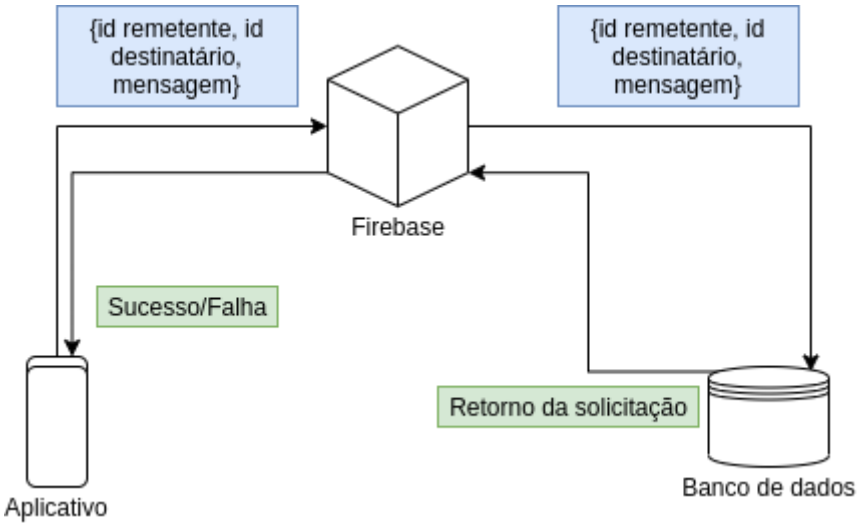

Figura 21 – Envio de uma mensagem para o Firebase

Após enviar a mensagem, o aplicativo verifica se esse usuário destinatário já estava presente na lista de *chats* antes. Caso não esteja, ele adiciona na lista de *chats* o *id* desse usuário, para que o mesmo apareça naquela lista de *chats* descrita no início desta seção.

Feito isso, o aplicativo então solicita ao Firebase para que o mesmo notifique o usuário destinatário seja notificado quando receber a mensagem. Essa solicitação é feita utilizando *tokens* gerados pelo Firebase. O aplicativo é responsável apenas por enviar a mensagem que será utilizada na notificação e o Firebase se encarregará de alertar o usuário destinatário através de uma notificação na barra de tarefas.

As mensagens mostradas na tela são recebidas do Firebase, através de uma solicitação. Essa solicitação é feita da mesma maneira que as anteriores: solicita todas as mensagens que contém os dois *ids* dos usuários presentes na conversa.

3.4 Principais Classes do Aplicativo

As principais classes e métodos que compõem a aplicação Android são descritas a seguir.

LoginUserActivity	Classe responsável pelo *login* dos usuários
ResetPasswordActivity	Classe responsável por ajudar o usuário a resetar a sua senha
RegisterUserActivity	Classe responsável por registrar o usuário no Firebase
MainActivity	Classe principal do aplicativo, após o usuário ter autenticado
AccountClientSettingsActivity	Classe responsável por editar as configurações da conta tipo cliente

AccountProSettingsActivity	Classe responsável por editar as configurações da conta tipo profissional
ChatsActivity	Classe responsável por mostrar a lista de *chats* com os outros usuários
ClientProfileActivity	Classe responsável por mostrar o perfil do cliente para o *personal trainer*
ContractsActivity	Classe responsável por mostrar os contratos ativos ou pendentes
EditWorkoutActivity	Classe responsável por editar um treino
ExercisesActivity	Classe responsável por mostrar todos os exercícios armazenados
LocationSearchActivity	Classe responsável por fazer a busca pela localização e mostrar no mapa
LookUpExerciseActivity	Classe responsável por mostrar detalhes de um exercício selecionado
LookUpWorkoutActivity	Classe responsável por mostrar detalhes de um treino selecionado
MessageActivity	Classe responsável por mostrar a tela de troca de mensagens com outro usuário
ProfessionalProfileActivity	Classe responsável por mostrar o perfil do *personal trainer* para o cliente
WorkoutsActivity	Classe responsável por mostrar todos os treinos disponíveis para aquele usuário
ExerciseAdapter	Classe responsável por receber uma lista com exercícios e mostrá-los na tela
MessageAdapter	Classe responsável por receber uma lista de mensagens e mostrá-las na tela
UserAdapter	Classe responsável por receber uma lista de usuários e mostrá-los na tela

WorkoutAdapter	Classe responsável por receber uma lista de treinos e mostrá-los na tela
AvailableProfessionals	Classe responsável por mostrar ao cliente a lista de *personal trainers* disponíveis
Chat	Classe responsável por armazenar o conteúdo de uma mensagem
Chatlist	Classe responsável por armazenar dados de um item de uma lista de *chats*
Contract	Classe responsável por armazenar dados de um contrato
Exercise	Classe responsável por armazenar dados de um exercício
User	Classe responsável por armazenar dados de um usuário
Workout	Classe responsável por armazenar dados de um treino
FirebaseMessaging	Classe responsável por mostrar notificações recebidas pelo Firebase
AndroidManifest	Arquivo XML que armazena informações essenciais sobre o aplicativo Android

Tabela 1 – Principais classes do aplicativo ConnectFit

Cada classe de nome terminado em Activity, está relacionada a uma tela do aplicativo. Como por exemplo a classe LoginUserActivity está relacionada a tela de *login* do usuário. Todas essas classes tem como base a classe AppCompatActivity do Android. Uma Activity é responsável por manusear os elementos visuais daquela tela que a pertence.

Ainda utilizando a tela de *login* como exemplo, os campos de *e-mail* e senha, assim como o botão para o *login* são declarados dentro do método *onCreate*, para

que possam ser manuseados. É nesse método *onCreate* de cada Activity em que os elementos visuais são declarados e manipulados. Os comportamentos sobre o que se deve fazer quando se clica em um botão também são definidos dentro desta função *onCreate*.

É dentro deste método *onCreate* que a maioria das solicitações do aplicativo para o Firebase são feitas também, como por exemplo para mostrar todos os contratos ativos. Os métodos utilizados para fazer as solicitações para o Firebase são o *getInstance* que retorna a instância atual do Firebase e *getReference* que retorna o endereço do item solicitado no banco de dados.

Algumas Activities contém métodos extras, como é o caso da classe LocationSearchActivity. Esta classe é responsável por mostrar a lista de profissionais disponíveis no mapa, utilizando a API Google Maps. Isso é feito através do método *onMapReady*, que é chamada quando a API Google Maps fornece o mapa para o aplicativo. Só então o aplicativo pode utilizar os métodos fornecidos pela API Google Maps para colocar os marcadores com a localização dos profissionais através do método *addMarker*.

As classes terminadas em Adapter são responsáveis por utilizar uma lista de objetos, como por exemplo contratos, e mostrá-los na tela para o usuário. Para isso, cada adapter utiliza o método *onBindViewHolder*, que pega cada elemento da lista de objetos fornecida e insere na tela para o usuário.

As classes como Chat, Chatlist e Workout são as chamadas classe modelo. Elas contém apenas atributos e métodos *get* para acessar esses valores. Como por exemplo a classe Contract que contém atributos como o *id* e *status*, além de métodos *get* para que o aplicativo possa acessar esses atributos. Essas classes são utilizadas sempre que o aplicativo faz uma solicitação para o Firebase. Se o aplicativo faz uma solicitação ao banco de dados do Firebase por contratos, então utilizaremos a classe Contract para criar objetos onde serão armazenados esses contratos retornados pelo banco de dados.

3.5 Banco de Dados

O Firebase utiliza um modelo de banco de dados chamado NoSQL. Nele, o usuário pode armazenar dados e sincronizar em tempo real. O modelo de NoSQL fornecido pelo Firebase é o Documento. Nele, os dados são representados na forma de um objeto JSON. O banco de dados utilizado pelo Firebase não exige a criação de tabelas e colunas antecipadamente, como um banco de dados relacional. Isso permite que as tabelas e colunas sejam criadas conforme os dados são inseridos. A **Figura 22** ilustra o esquema do banco de dados utilizado.

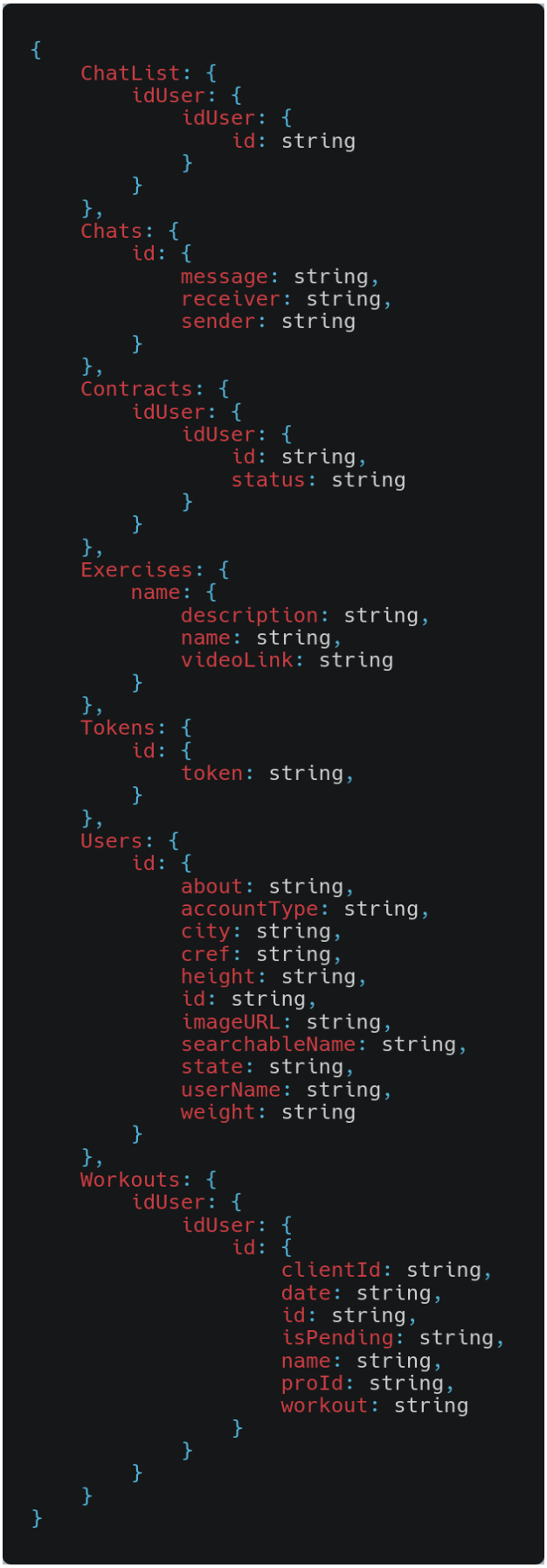

Figura 22 – Esquema do banco de dados do aplicativo ConnectFit

Baseado na figura acima, o item *Chatlist* se refere a uma conversa entre dois usuários. Ele contém o *id* desses dois usuários, e um *id* para a *Chatlist*. Os *ids* dos dois usuários utilizamos para identificar entre quais usuários há uma conversa. Se por exemplo o usuário A trocou mensagens com o usuário B, criamos um item dentro da *Chatlist* que armazena os *ids* destes dois usuários para sabermos que há uma conversa entre eles. Também adicionamos um identificador único para cada *Chatlist*, como citado no início deste parágrafo, para fins de identificação de cada *Chatlist* apenas.

O item *Chats* se refere a cada mensagem enviada. Nele, armazenamos cada mensagem enviada pelos usuários conta com um *id* para essa mensagem, o *id* do remetente, o *id* do destinatário, e o conteúdo da mensagem. Estas mensagens são utilizadas na tela de troca de mensagens entre os usuários. Como exemplo, o aplicativo irá solicitar por todas as mensagens entre dois usuários, os quais o aplicativo fornecerá os seus devidos *ids*. Com essas mensagens o aplicativo mostrará o seu conteúdo.

O item *Contracts* se refere a cada contrato. Cada contrato armazenado no banco de dados contém o *id* de ambos os usuários envolvidos, um *id* para o contrato, e o seu *status*, caso seja ativo ou pendente. Quando algum usuário contrata ou aceita o contrato, no caso do profissional, é o campo *status* daquele contrato que é mudado. Caso um dos usuários queira encerrar um contrato, o aplicativo então remove o contrato do banco de dados, juntamente com os seus campos.

O item *Exercises* contém os exercícios. Estes são adicionados diretamente utilizando o site do Firebase. Nele, cada exercício armazenado recebe um nome para o exercício, a sua descrição e um *link* para um vídeo da execução deste exercício. Esses exercícios são acessados pelo aplicativo e não são editados, são apenas mostrados para o usuário, como uma forma de referência para a execução de cada exercício.

O item *Tokens* é utilizado e gerido pelo Firebase para enviar notificações.

O item *Users* contém os usuários do aplicativo. Nele, cada usuário armazenado tem o seu *id* único, seu nome, cidade, estado, CREF, sobre, tipo de conta, altura, peso e *link* para imagem de perfil. Todos os campos menos o *id* e o tipo de conta podem ser alterados pelo usuário na tela de edição de conta.

O item *Workouts* é responsável por armazenar todos os treinos. Nele, cada treino armazenado recebe o *id* do profissional que passou o treino, o *id* do cliente para quem ele passou o treino, uma data para a realização deste treino, um *id* único para o treino, *status* de pendente ou completado do treino e o treino em si, que é escrito pelo profissional. Apesar de inicialmente adicionado pelo profissional, o cliente que recebeu algum treino pode editá-lo pelo aplicativo, adicionando *feedback* caso necessário e também marcando o treino como completado.

4. INTERFACES DO APLICATIVO

Este capítulo é dedicado a mostrar os testes do aplicativo ConnectFit que foram feitos em ambiente de desenvolvimento, a fim de determinar se o projeto atendeu aos objetivos propostos. Para os testes, foram utilizados dois dispositivos móveis reais com o sistema operacional Android. Para o funcionamento do aplicativo, inicialmente foram criadas 4 contas dentro do próprio aplicativo. Duas contas do tipo cliente e duas contas do tipo profissional.

Ao iniciar o aplicativo pela primeira vez, o usuário se depara com a tela de *login*, como ilustra a **Figura 23**. Nela o usuário deve informar *e-mail* e senha e clicar no botão ENTRAR para realizar o *login*.

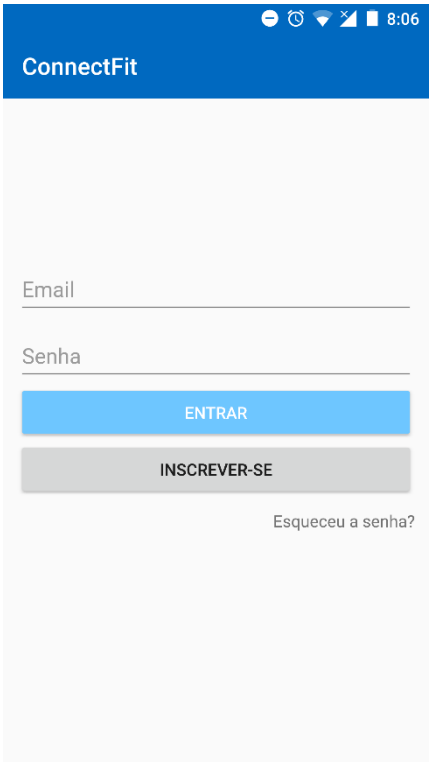

Figura 23 – Tela de login do aplicativo

Caso o usuário queira fazer o cadastro, ele deve clicar no botão INSCREVER-SE na tela de *login*, onde será redirecionado para uma tela onde deverá informar seus dados, como ilustra a **Figura 24**.

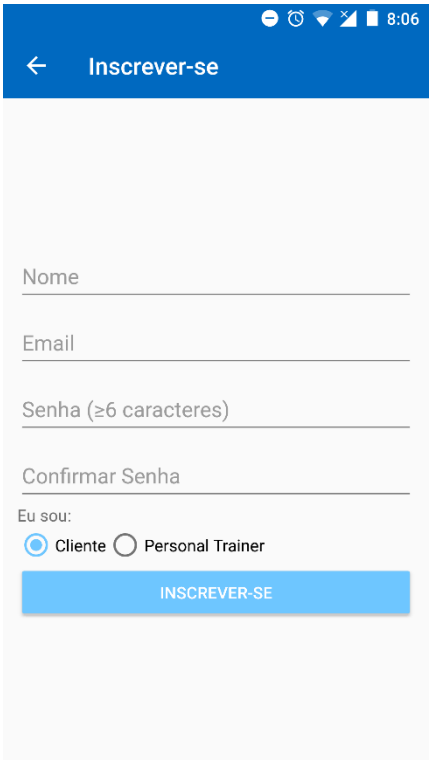

Figura 24 – Tela de cadastro do aplicativo

Caso o usuário queira recuperar a sua senha, ele deve clicar no *link* abaixo do botão INSCREVER-SE na tela de *login*, onde será redirecionado para uma tela onde informará o seu *e-mail*. A **Figura 25** representa essa tela de recuperação de senha.

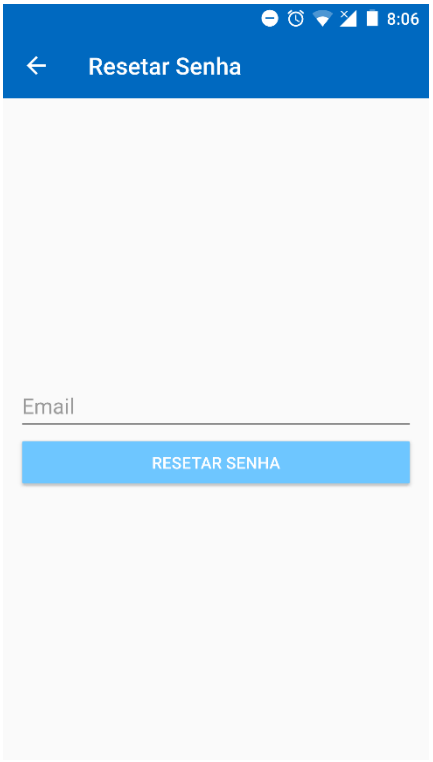

Figura 25 – Tela de recuperação da senha do usuário

Após a validação do cadastro, o usuário é redirecionado para a tela de edição de conta, onde poderá inserir novas informações sobre ele mesmo, conforme mostra a **Figura 26** para a conta do tipo cliente e a **Figura 27** para a conta do tipo profissional.

Figura 26 – Tela de edição de conta do tipo cliente

Figura 27 – Tela de edição de conta do tipo profissional

Ao clicar em salvar o usuário é redirecionado para tela principal. Nela, o usuário tem acesso aos links para as outras principais funcionalidades do aplicativo. A **Figura 28** ilustra a tela principal para a conta do tipo cliente, com a conta criada inicialmente como exemplo. A única diferença para a conta do tipo profissional é que ela possui um link para a tela de busca por profissionais.

Figura 28 – Tela principal para a conta do tipo cliente

O cliente então pode clicar no botão PROFISSIONAIS DISPONÍVEIS onde será redirecionado para a tela com a lista de profissionais cadastrados no aplicativo, conforme ilustra a **Figura 29**.

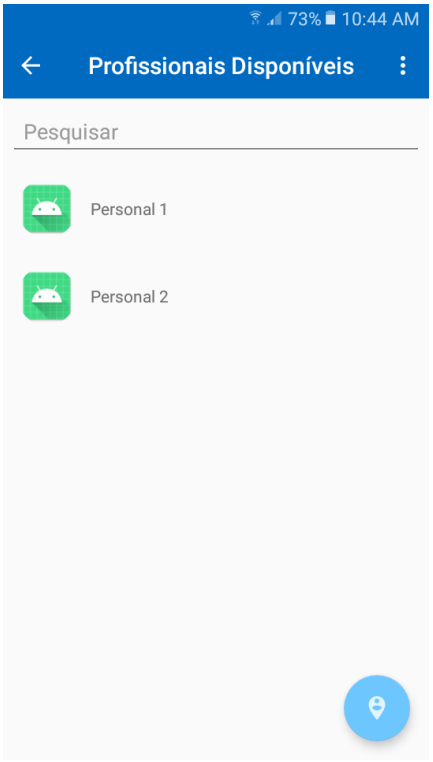

Figura 29 – Tela com lista dos profissionais cadastrados no aplicativo

O cliente também pode usar a barra de pesquisa, para pesquisar por um profissional pelo seu nome. A **Figura 30** ilustra um exemplo do resultado da pesquisa feita pelo usuário.

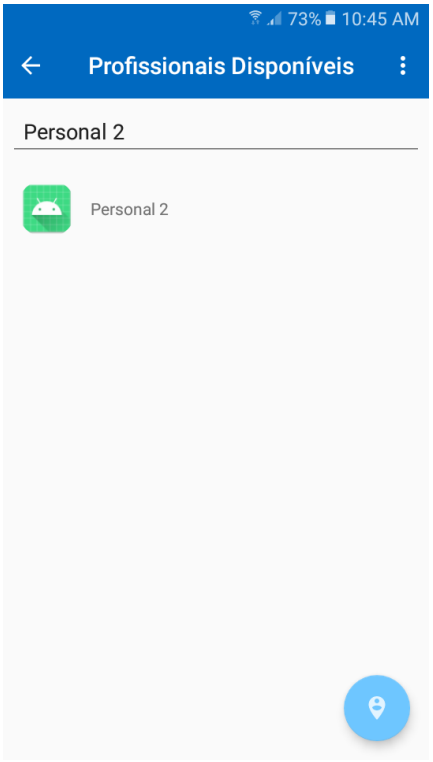

Figura 30 – Tela com resultado da busca por nome do profissional

O cliente também pode clicar no botão azul no canto inferior direito da tela de profissionais disponíveis. Ao fazer isso, o usuário é redirecionado para a busca por profissionais ao seu redor, através da localização coletada pelo aparelho celular do usuário. A **Figura 31** ilustra a tela com os profissionais disponíveis ao redor do cliente. A localização do cliente está marcada em vermelho, e a localização dos profissionais está marcada em azul.

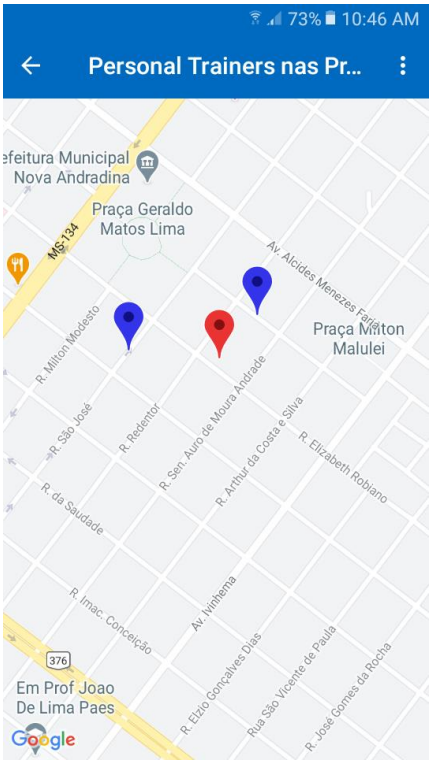

Figura 31 – Tela com mapa com os profissionais disponíveis

Ao clicar em um dos marcadores azuis, o usuário tem acesso ao nome daquele profissional. E ao clicar nesse nome, o usuário é redirecionado para a tela com detalhes sobre aquele profissional. A **Figura 32** ilustra essa tela com detalhes do profissional selecionado, como exemplo.

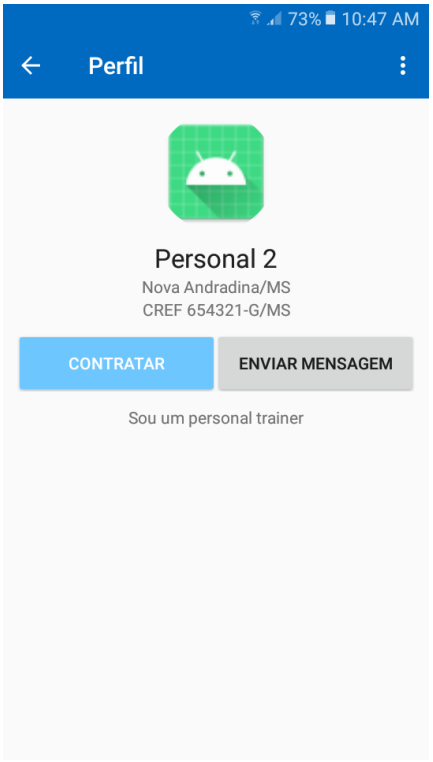

Figura 32 – Tela com detalhes da conta do profissional selecionado

Nessa tela o cliente tem acesso a informações do profissional a ser contratado. Ao clicar em no botão CONTRATAR, o contrato passa a ter o *status* de pendente no banco de dados. Ao clicar no botão ENVIAR MENSAGEM, o cliente é redirecionado para a tela de troca de mensagens com aquele profissional. A **Figura 33** ilustra essa tela de troca de mensagens. A figura abaixo já contém algumas mensagens enviadas como exemplo. Para enviar novas mensagens o usuário deve digitar no campo no canto inferior da tela e clicar no botão no canto inferior.

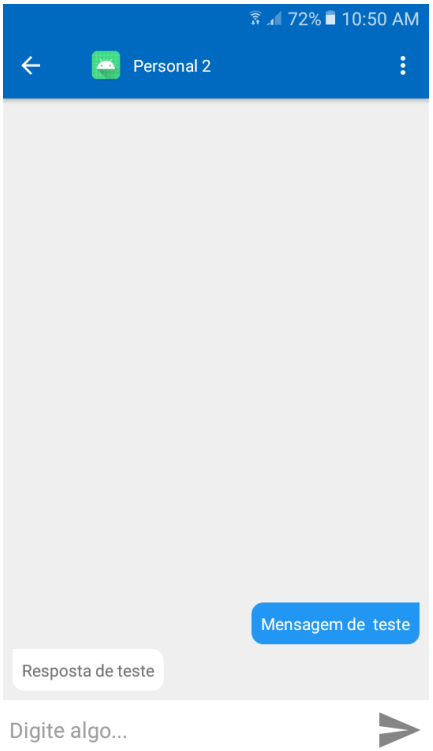

Figura 33 – Tela de troca de mensagens entre dois usuários

Sempre ao enviar uma mensagem, o usuário destinatário recebe uma notificação informando que recebeu uma mensagem. A **Figura 34** ilustra essa notificação recebida.

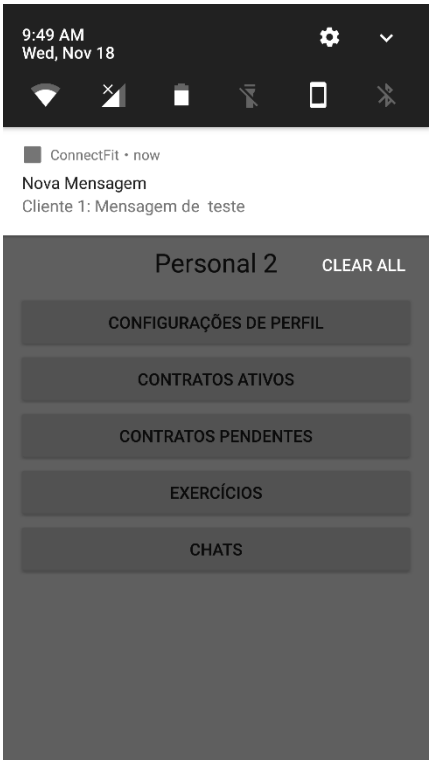

Figura 34 – Notificação mostrada na barra de tarefas do aparelho

Voltando a tela principal, temos o botão CONFIGURAÇÕES DE PERFIL que redireciona para a tela de edição de conta descrita anteriormente neste capítulo. Logo abaixo desse botão temos o botão CONTRATOS ATIVOS e CONTRATOS PENDENTES. Ao clicar no botão CONTRATOS PENDENTES, o usuário é redirecionado para uma tela onde há uma lista com os clientes aguardando aprovação para iniciar o contrato, caso o usuário autenticado seja um profissional, ou uma lista com os profissionais que ainda não aprovaram o contrato para ser iniciado, caso o usuário autenticado seja um cliente. A **Figura 35** exemplifica uma lista de contratos pendentes para um usuário do tipo cliente.

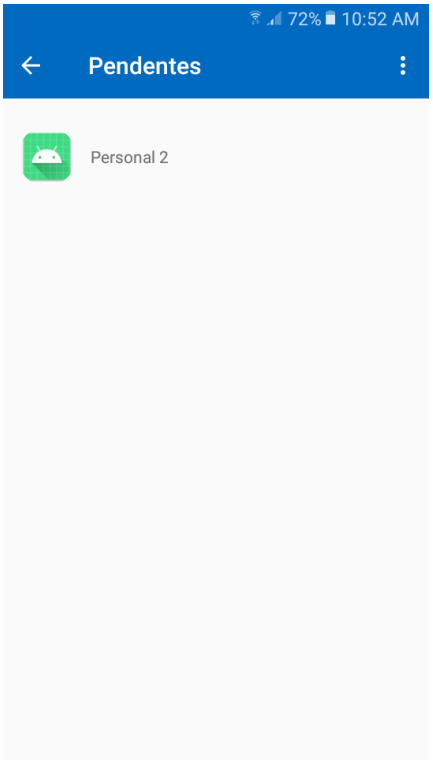

Figura 35 – Tela com lista de contratos pendentes

O profissional, ao clicar em um dos clientes na sua lista de contratos pendentes, é redirecionado para uma tela com dados do perfil daquele cliente, onde poderá aceitar ou não o contrato. A **Figura 36** ilustra essa tela com os dados do cliente.

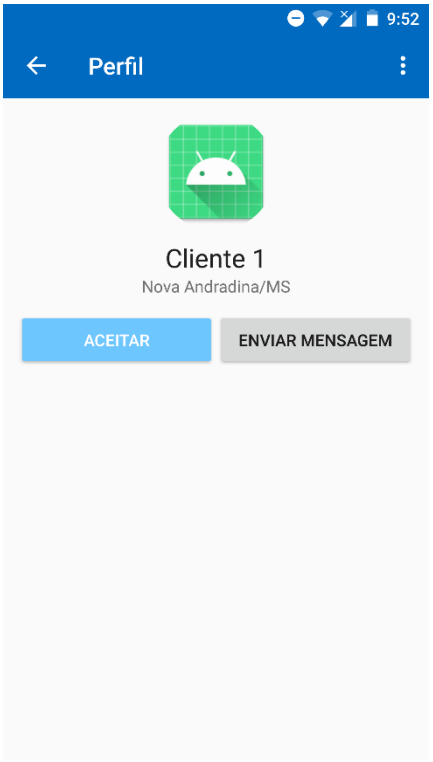

Figura 36 – Tela com dados do perfil do cliente selecionado

O profissional, ao clicar no botão ACEITAR, faz com o que o contrato ganhe o *status* de ativo, e esse botão se transforma no botão ENCERRAR, caso o profissional queira encerrar o contrato.

Na tela principal, ao clicar no botão CONTRATOS ATIVOS, o usuário pode então ver os contratos que foram aprovados pelo profissional e estão em atividade. A **Figura 37** ilustra a lista de contratos ativos.

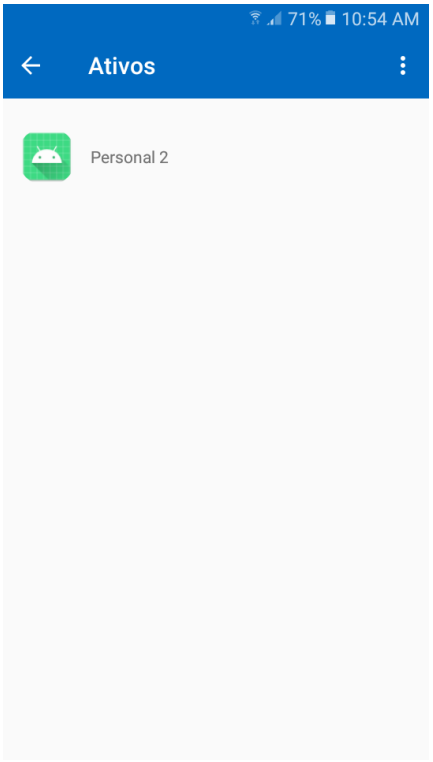

Figura 37 – Tela com lista de contratos ativos

O seu funcionamento é o mesmo da tela com os contratos pendentes. Ao clicar em qualquer usuário da lista, o aplicativo redireciona para a tela com dados da conta daquele usuário selecionado. Desta vez, como o contrato está ativo, há um novo botão na tela de perfil, conforme ilustrado na **Figura 38**.

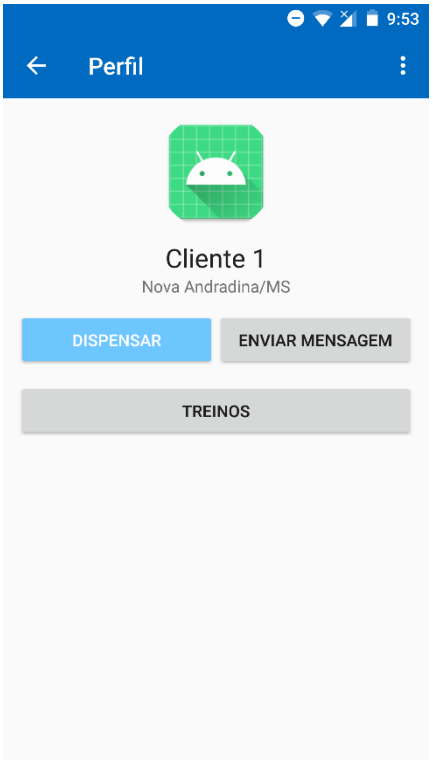

Figura 38 – Tela de perfil do usuário selecionado com contrato ativo

Ao clicar no botão TREINOS, o usuário autenticado é redirecionado para a lista de treinos com aquele usuário selecionado. A **Figura 39** exemplifica essa tela de treinos, já com um treino inserido como exemplo.

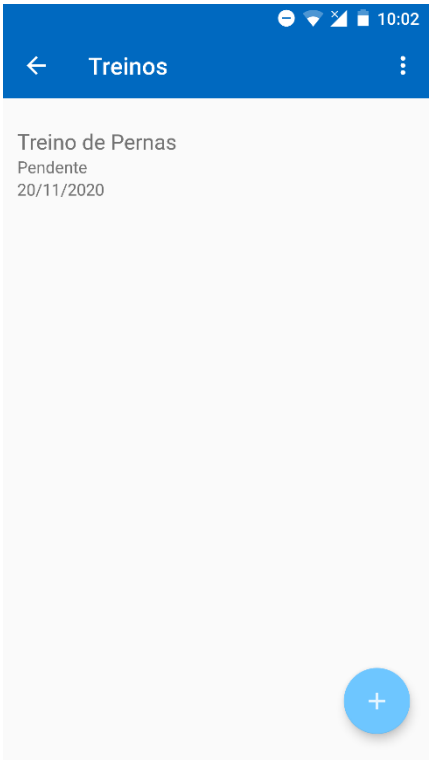

Figura 39 – Tela com a lista de treinos do usuário

Caso o usuário autenticado seja a de um profissional, como é o caso da figura acima, há o botão azul no canto inferior direito, que tem a função de redirecionar para a tela de adicionar um novo treino para o seu cliente. A **Figura 40** ilustra essa tela de adição de um novo treino para o cliente.

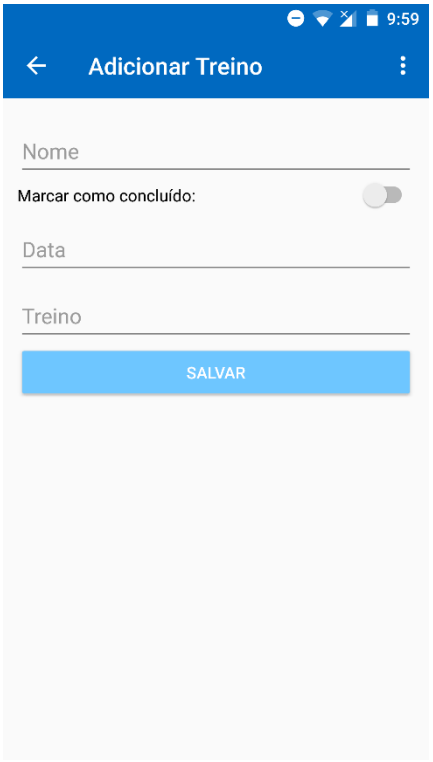

Figura 40 – Tela de adição de um novo treino

Preencher os campos e clicar no botão SALVAR, o novo treino é adicionado à lista de treinos ilustrada na **Figura 39**. O usuário, seja cliente ou profissional, ao clicar em um dos treinos da lista, é redirecionado para uma tela com os detalhes daquele treino. A **Figura 41** exemplifica a tela com detalhes de um dos treinos adicionados.

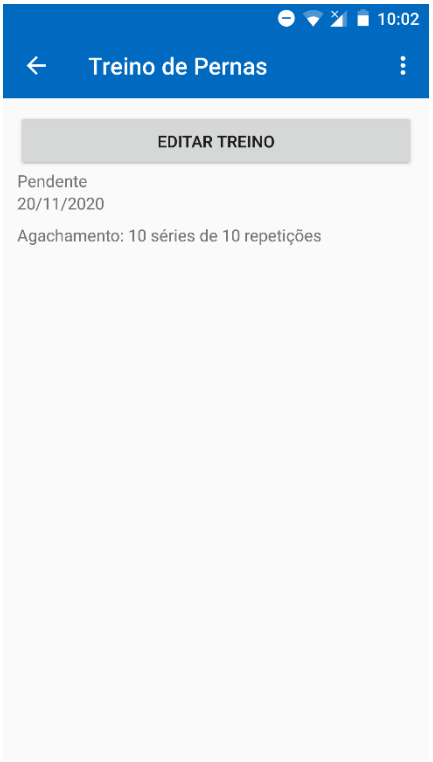

Figura 41 – Tela com detalhes de um dos treinos selecionados

Ao clicar no botão EDITAR TREINO, o usuário é redirecionado para a tela de edição de treino, onde tanto o cliente quanto o profissional podem adicionar detalhes ou marcar o treino como concluído, no caso do cliente. A **Figura 42** ilustra essa tela de edição de treino.

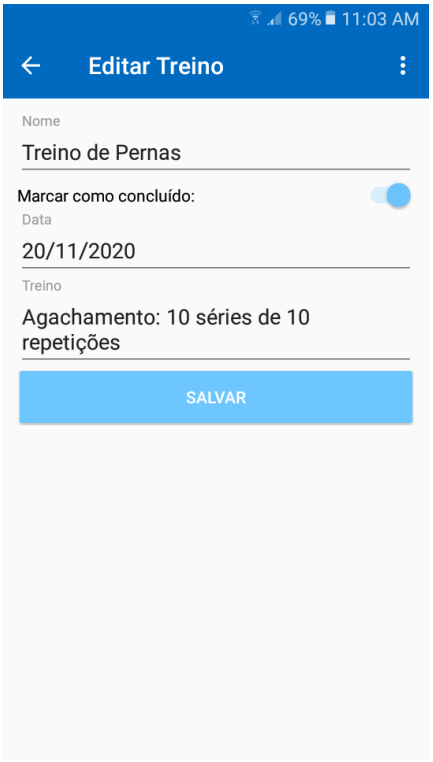

Figura 42 – Tela de edição de treino

Ao clicar no botão SALVAR, aquele treino então é atualizado, e passará a mostrar as novas informações sempre que for requisitado.

Voltando a tela principal novamente, ainda temos dois botões a serem abordados. O botão EXERCÍCIOS, ao ser clicado, redireciona para a tela com a lista de exercícios armazenadas no banco de dados. A **Figura 43** ilustra essa tela com a lista de exercícios.

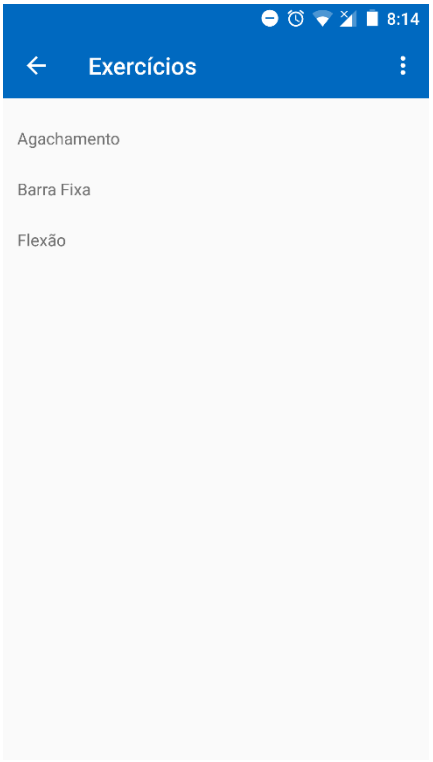

Figura 43 – Tela com lista de exercícios

Ao clicar em qualquer um desses exercícios, o usuário é redirecionado para uma tela com detalhes daquele exercício selecionado. A **Figura 44** ilustra essa tela com detalhes do exercício selecionado.

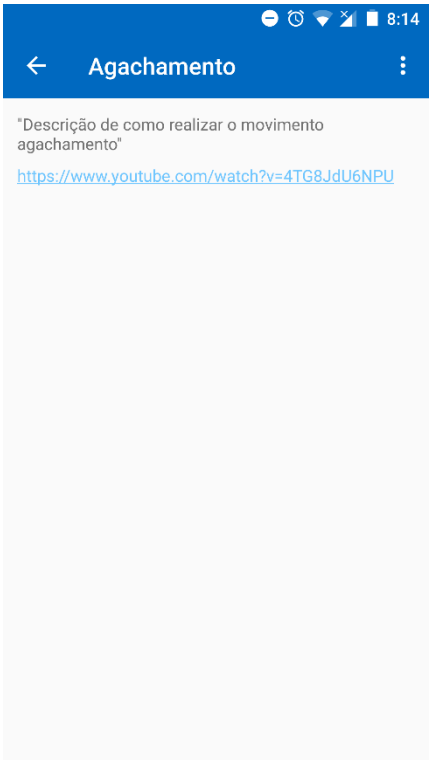

Figura 44 – Tela com descrição do exercício selecionado

Voltando a tela principal, o último botão a ser abordado, o botão CHATS ao ser clicado redireciona para a tela com todos os usuários os quais o usuário autenticado trocou mensagens. Essa tela é exemplificada na **Figura 45** abaixo.

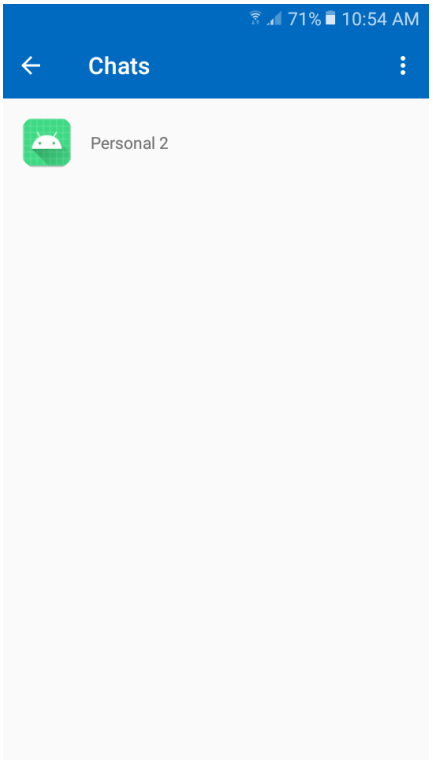

Figura 45 – Tela com lista de usuários os quais foram trocadas mensagens com o usuário autenticado

Ao clicar em qualquer um desses usuários da lista, o usuário é redirecionado para a tela de troca de mensagens com aquele usuário selecionado, tela a qual já foi ilustrada em figuras anteriores.

Por último, algo que está presente em quase todas as telas é o botão de menu. Este botão está localizado no canto superior direito do aplicativo, no formato de três pontos brancos. Esse botão ao ser clicado oferece links para as principais telas do aplicativo. O resultado ao clicar nesse botão é exemplificado na **Figura 46** abaixo.

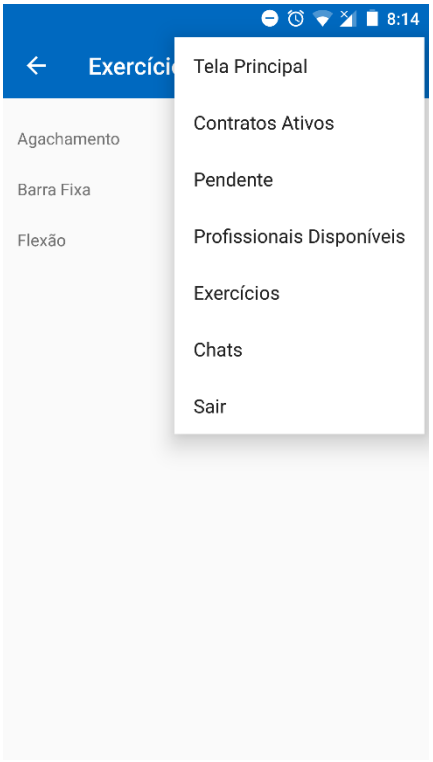

Figura 46 – Botão de menu ao ser clicado

5. CONCLUSÃO

Com a busca pela vida saudável, aqueles que desejam melhorar a sua qualidade de vida através da prática de exercícios físicos devem buscar a ajuda de profissionais. Com a vasta gama de profissionais de educação física disponíveis hoje, fica difícil avaliar e até mesmo encontrar profissionais qualificados sem fazer uma extensa busca pelas redes sociais ou utilizar buscadores como o Google, por exemplo. Foi justamente pensando nisso, de buscar por profissionais qualificados, de preferência os mais próximos geograficamente, que este projeto foi proposto e desenvolvido.

Através do estudo e pesquisa realizados neste trabalho, foi verificado o poder de trazer projetos simples ou complexos para a realidade que a plataforma Android possui. Isso, junto com a aplicação de novas tecnologias como serviços de banco de dados na nuvem e também a integração de mapas, mostra que a plataforma Android tem se expandido para acompanhar a necessidade de solucionar os problemas atuais. Utilizando essas tecnologias, foi possível desenvolver o aplicativo ConnectFit apresentado neste trabalho, que trouxe uma forma das pessoas buscarem por profissionais de educação física de forma facilitada e objetiva. Com isso, ajudamos tanto os clientes quanto os profissionais, que agora possuem mais visibilidade, já que os seus futuros clientes que moram ao seu redor podem encontrá-lo de maneira fácil pelo mapa.

5.1 Trabalhos Futuros

Apresento aqui possíveis funcionalidades que poderiam ser implementadas no aplicativo, a fim de melhorá-lo.

- Cadastro no aplicativo através das contas nas redes sociais: Esta funcionalidade está disponível dentro do próprio Firebase e permite ser ativada em poucos passos;

- Sistema de pagamento dentro do próprio aplicativo: Pode ser feito integrando sistemas de pagamento dentro do aplicativo, como o Google Wallet;
- Possibilidade dos usuários criarem álbuns de fotos: Fazer com que os usuários possam fazer o *upload* de fotos e criar uma tela onde essas fotos possam ser vistas;
- Ambiente de treinos com interface mais intuitiva: Fazer com que o usuário economize tempo digitando ao permitir que ele escolha e adicione exercícios diretamente da lista de exercícios já existente, com alguns cliques e sem a necessidade de digitar o nome de cada exercício manualmente;
- Gráficos da evolução física dos clientes: Acompanhar a mudança de peso do cliente e colocar a sua evolução em um gráfico;
- Agendamento de aulas por parte do cliente: Fazer com que o cliente tenha a possibilidade de reservar em um calendário aulas presenciais com aquele profissional;

REFERÊNCIAS BIBLIOGRÁFICAS

99. 99. Disponível em <https://99app.com/>. Acesso em 20/11/2020 às 11h45min.

99_FUNCIONAMENTO. Passo-a-passo do 99 táxis. Disponível em <https://www.transportal.com.br/noticias/informacoes-uteis/passo-a-passo-do-99-taxi/>. Acesso em 20/11/2020 às 11h54min.

ANDROID_HIST. The history of Android. Disponível em <https://www.androidauthority.com/history-android-os-name-789433/>. Acesso em 20/11/2020 às 11h02min.

ANDROID_STUDIO. Android Studio. Disponível em <https://developer.android.com/studio/>. Acesso em 20/11/2020 às 11h02min.

ANNUZZI, Joseph, Jr. et al. Introduction to Android application development. 4ª edição, 2014.

AOSP. Android Open Source Project. Disponível em <https://source.android.com/>. Acesso em 20/11/2020 às 11h02min.

ARMORED_FARMER. Bodyweight Squats. Disponível em <https://armoredfarmer.com/2019/05/26/body-weight-squats/>. Acesso em 20/11/2020 às 11h36min.

AWS. AWS Amplify. Disponível em <https://aws.amazon.com/pt/amplify/>. Acesso em 20/11/2020 às 11h14min.

AZURE_PAAS. Plataforma como serviço. Disponível em <https://azure.microsoft.com/pt-br/overview/what-is-paas/>. Acesso em 20/11/2020 às 11h07min.

BIEHL BOSSLE, Cibele. O personal trainer e o cuidado de si: uma perspectiva de mediação profissional. Movimento [em linha], Vol. 14 (1), 187-198, 2008. ISSN 0104-754X. Disponível em: <https://www.redalyc.org/articulo.oa?id=115316019009>. Acesso em 20/11/2020 às 11h34min.

BTFIT. BTFIT. Disponível em <https://bt.fit/pt/>. Acesso em 20/11/2020 às 11h37min.

CLOUDFLARE. Cloud Definition. Disponível em <https://www.cloudflare.com/learning/cloud/what-is-the-cloud/>. Acesso em 20/11/2020 às 11h06min.

DROPBOX. Dropbox. Disponível em <https://www.dropbox.com/>. Acesso em 20/11/2020 às 11h08min.

ERL, Thomas et al. Cloud Computing: Concepts, Technology & Architecture. 1ª edição, 2013.

FIREBASE. Firebase. Disponível em <https://firebase.google.com/>. Acesso em 20/11/2020 às 11h11min.

FIREBASE_DATABASE. Firebase Database. Disponível em <https://firebase.google.com/docs/database>. Acesso em 20/11/2020 às 11h56min.

GMAIL. Gmail. Disponível em <https://www.gmail.com/>. Acesso em 20/11/2020 às 11h07min.

GOOGLE_PLAY. Google Play. Disponível em <https://play.google.com/>. Acesso em 20/11/2020 às 11h02min.

GOTRAINER. GoTrainer. Disponível em <https://www.gotrainer.sg/>. Acesso em 20/11/2020 às 11h38min.

GUIA_ED_FISICA. Guia do Estudante. Disponível em <https://guiadoestudante.abril.com.br/profissoes/educacao-fisica/>. Acesso em 20/11/2020 às 11h20min.

HEROKU. Heroku. Disponível em <https://www.heroku.com/>. Acesso em 20/11/2020 às 11h12min.

JAVA. Java. Disponível em <https://www.java.com/en/>. Acesso em 20/11/2020 às 11h55min.

KINVEY. Kinvey. Disponível em <https://www.progress.com/kinvey/>. Acesso em 20/11/2020 às 11h17min.

LINKEDIN. LinkedIn. Disponível em <https://www.linkedin.com/>. Acesso em 20/11/2020 às 11h07min.

MAPS_PLAT. Google Maps Platform. Disponível em <https://developers.google.com/maps/documentation/>. Acesso em 20/11/2020 às 11h04min.

MAPS_SDK. Maps SDK for Android. Disponível em <https://developers.google.com/maps/documentation/android-sdk/overview/>. Acesso em 20/11/2020 às 11h06min.

MFIT. MFIT. Disponível em <https://www.mfitpersonal.com.br/>. Acesso em 20/11/2020 às 11h42min.

MOBITRAINER. Mobitrainer. Disponível em <https://mobitrainer.com.br/>. Acesso em 20/11/2020 às 11h42min.

NEXUR. Nexur. Disponível em <https://aplicativonexur.com.br/>. Acesso em 20/11/2020 às 11h42min.

OHA. Open Handset Alliance. Disponível em <https://www.openhandsetalliance.com/oha_faq.html>. Acesso em 20/11/2020 às 11h03min.

ORACLE_CLOUD_DB. O que É um Banco de Dados em Nuvem?. Disponível em <https://www.oracle.com/br/database/what-is-a-cloud-database/>. Acesso em 20/11/2020 às 11h06min.

PARSE. Parse. Disponível em <https://parseplatform.org/>. Acesso em 20/11/2020 às 11h16min.

PERSONAL_QUAL. Como ser um personal trainer qualificado. Disponível em <https://www.educacaofisica.com.br/fitness2/personal-training2/como-ser-um-personal-trainer-qualificado/>. Acesso em 20/11/2020 às 11h17min.

PLATFORM_ARCH. Platform Architecture. Disponível em <https://developer.android.com/guide/platform/>. Acesso em 20/11/2020 às 11h03min.

PORTAL_ED_FISICA. Portal Educação Física. Disponível em <https://www.educacaofisica.com.br/fitness2/personal-training2/conheca-os-desafios-e-tendencias-da-carreira-de-personal-trainer/>. Acesso em 20/09/2020 às 10h55min.

RED_HAT_API. O que é API?. Disponível em <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces/>. Acesso em 20/11/2020 às 11h05min.

SF_SAAS. SaaS. Disponível em <https://www.salesforce.com/br/saas/>. Acesso em 20/11/2020 às 11h07min.

STATCOUNTER. Mobile Operating System Market Share Worldwide. Disponível em <https://gs.statcounter.com/os-market-share/mobile/worldwide/>. Acesso em 20/11/2020 às 10h57min.

TECNOFIT. Tecnofit. Disponível em <https://www.tecnofit.com.br/>. Acesso em 20/11/2020 às 11h45min.

TREI_NO. Trei.no. Disponível em <https://trei.no/>. Acesso em 20/11/2020 às 11h45min.

VMWARE. Máquina virtual. Disponível em <https://www.vmware.com/br/topics/glossary/content/virtual-machine.html/>. Acesso em 20/11/2020 às 11h06m.

APÊNDICE A – Instalação e configuração do Android Studio

Este apêndice é um guia para a instalação do Android Studio no sistema operacional Ubuntu, que foi o sistema onde o aplicativo foi desenvolvido. A instalação é simples e requer apenas alguns comandos no terminal.

Com o terminal aberto, executa-se 3 comandos em sequência para adicionar o repositório que contém a versão mais atualizada do Android Studio, um comando para atualizar a lista de repositórios e finalmente um comando para instalar o Android Studio. A **Figura 47** ilustra quais são esses três comandos.

```
sudo add-apt-repository ppa:maarten-fonville/android-studio &&
sudo apt-get update &&
sudo apt-get install android-studio
```

Figura 47 – Comandos de instalação do Android Studio

Com o Android Studio instalado e aberto pela primeira vez, o usuário necessita apenas seguir os passos que o Android Studio orienta. A **Figura 48** ilustra a primeira janela que aparece quando o usuário inicia o Android Studio pela primeira vez. O usuário deve selecionar a última opção e clicar no botão OK.

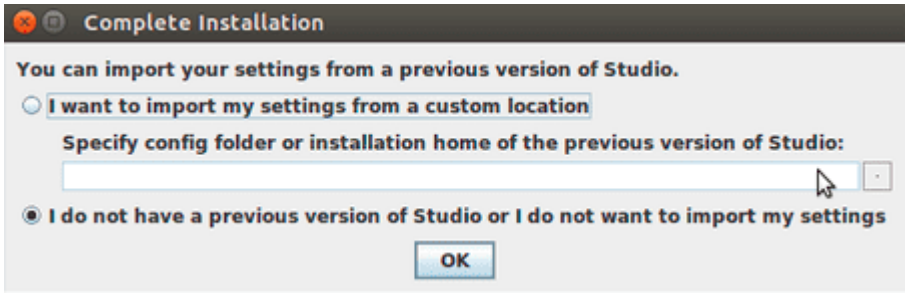

Figura 48 – Tela com a primeira janela após a instalação do Android Studio

O usuário então é recebido com a tela de configuração inicial do Android Studio, como ilustra a **Figura 48**. O usuário deve apenas clicar no botão Next que o Android Studio fará toda a configuração sozinho.

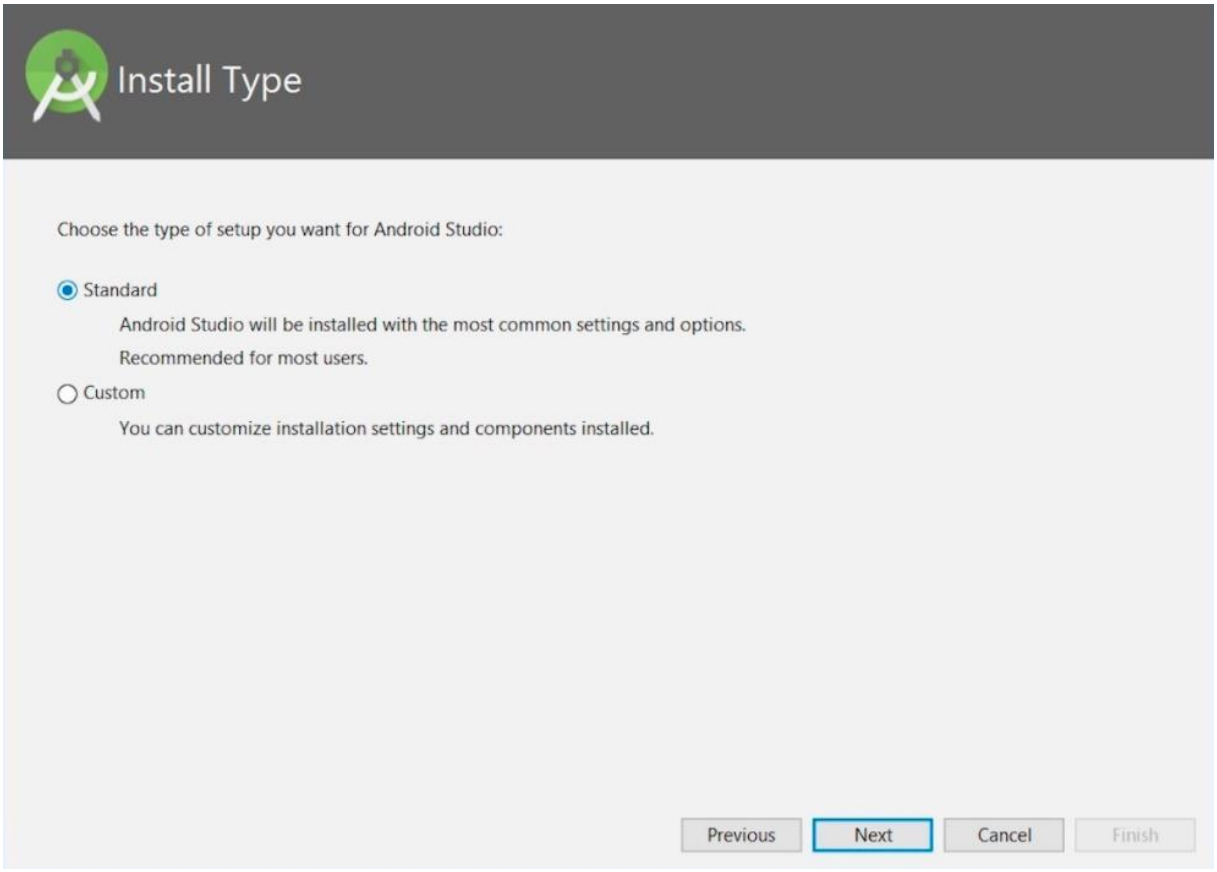

Figura 49 – Tela de configuração inicial do Android Studio

Após clicar no botão Next em cada passo, deve-se clicar no botão Finish e o Android Studio estará pronto para uso.

APÊNDICE B – Configuração do Firebase

Este apêndice mostrará o passo-a-passo para configurar o Firebase com um projeto do Android Studio. Ele se baseia nas instruções oficiais disponibilizadas pela documentação do próprio Firebase em https://firebase.google.com/docs/android/setup?hl=pt-br.

Passo 1: Primeiro deve-se acessar o console do Firebase através do link https://console.firebase.google.com e em seguida clicar no botão de Criar um projeto.

Passo 2: Inserir um nome para o projeto e marcar a caixa para aceitar os termos do Firebase para então clicar em continuar, conforme mostra a **Figura 50**.

Figura 50 – Tela de criação de um projeto no Firebase

Passo 3: Na próxima tela deve-se clicar apenas em continuar e na última tela deve marcar todas as caixas e clicar em Criar projeto, conforme ilustra a **Figura 51**.

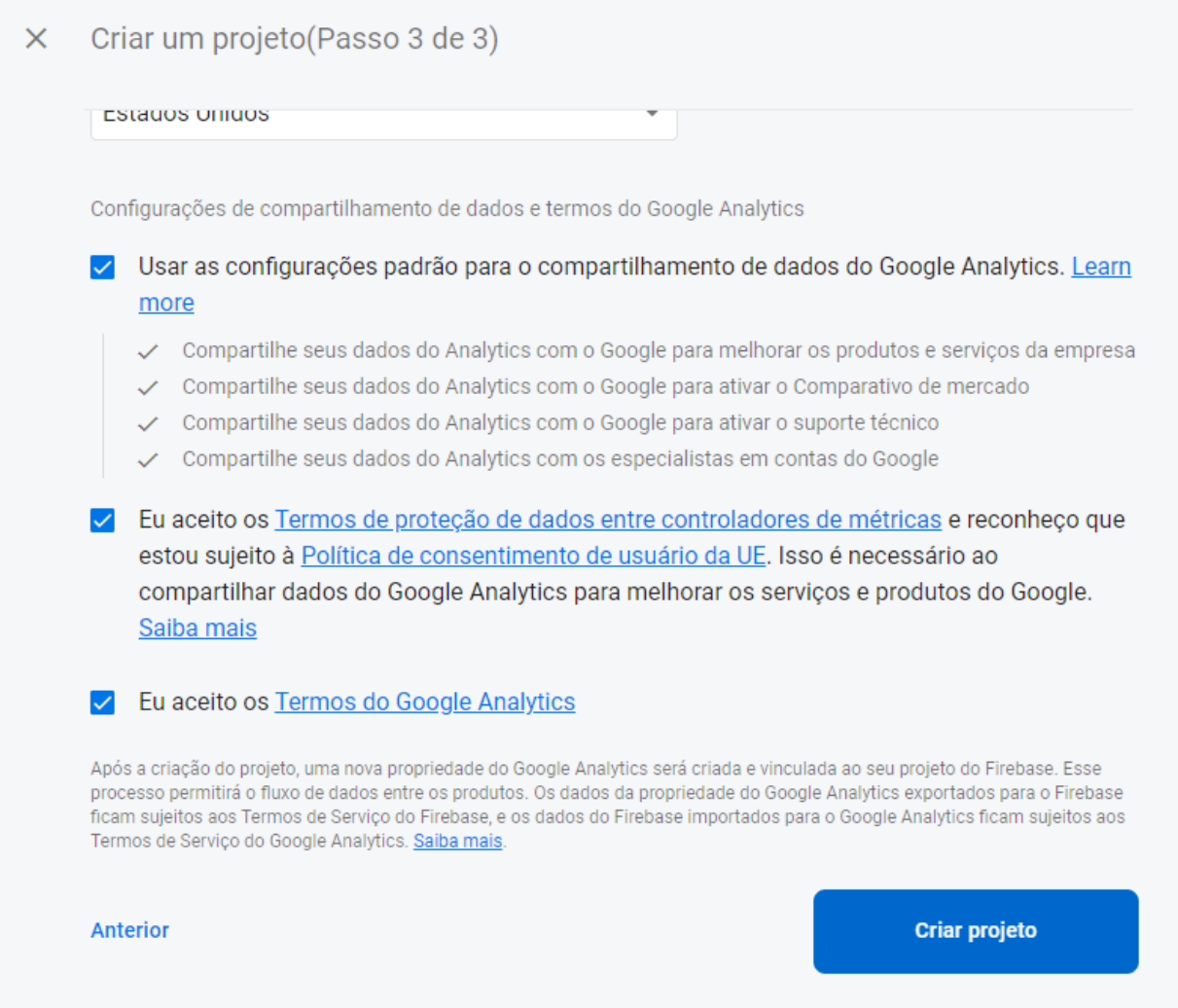

Figura 51 – Tela com configurações para a criação de um projeto no Firebase

Passo 4: Após o site do Firebase redirecionar para a tela do projeto que acabara de ser criado, deve-se clicar no ícone do Android onde o Firebase mostrará os passos para adicioná-lo a um projeto Android.

Passo 5: O usuário deve inserir o nome do pacote do Android, conforme ilustra a **Figura 52**.

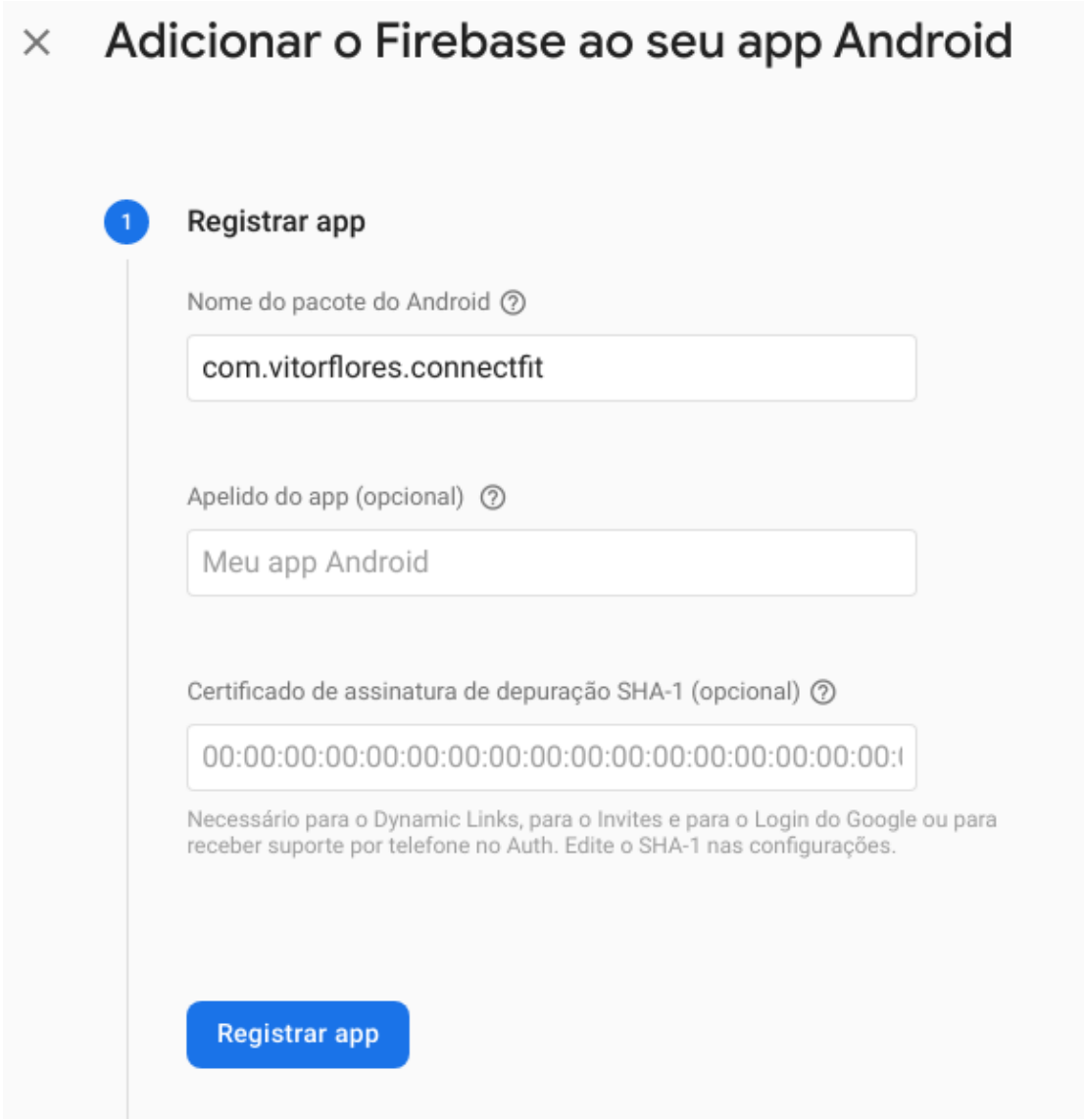

Figura 52 – Tela para registrar o aplicativo

Passo 6: Após clicar em Registrar app na figura acima, o usuário deve fazer o *download* do arquivo google-services.json e move-lo para o diretório raiz do módulo do app Android, como indica a **Figura 53**.

Figura 53 – Tela com instruções para adicionar arquivo do Firebase para o aplicativo Android

Passo 7: Após clicar no botão Próxima na figura acima, o usuário deve abrir com o Android Studio o arquivo build.gradle dentro do diretório raiz do projeto e adicionar as linhas de código indicadas pelos comentários em rosa nas seções indicadas na **Figura 54**.

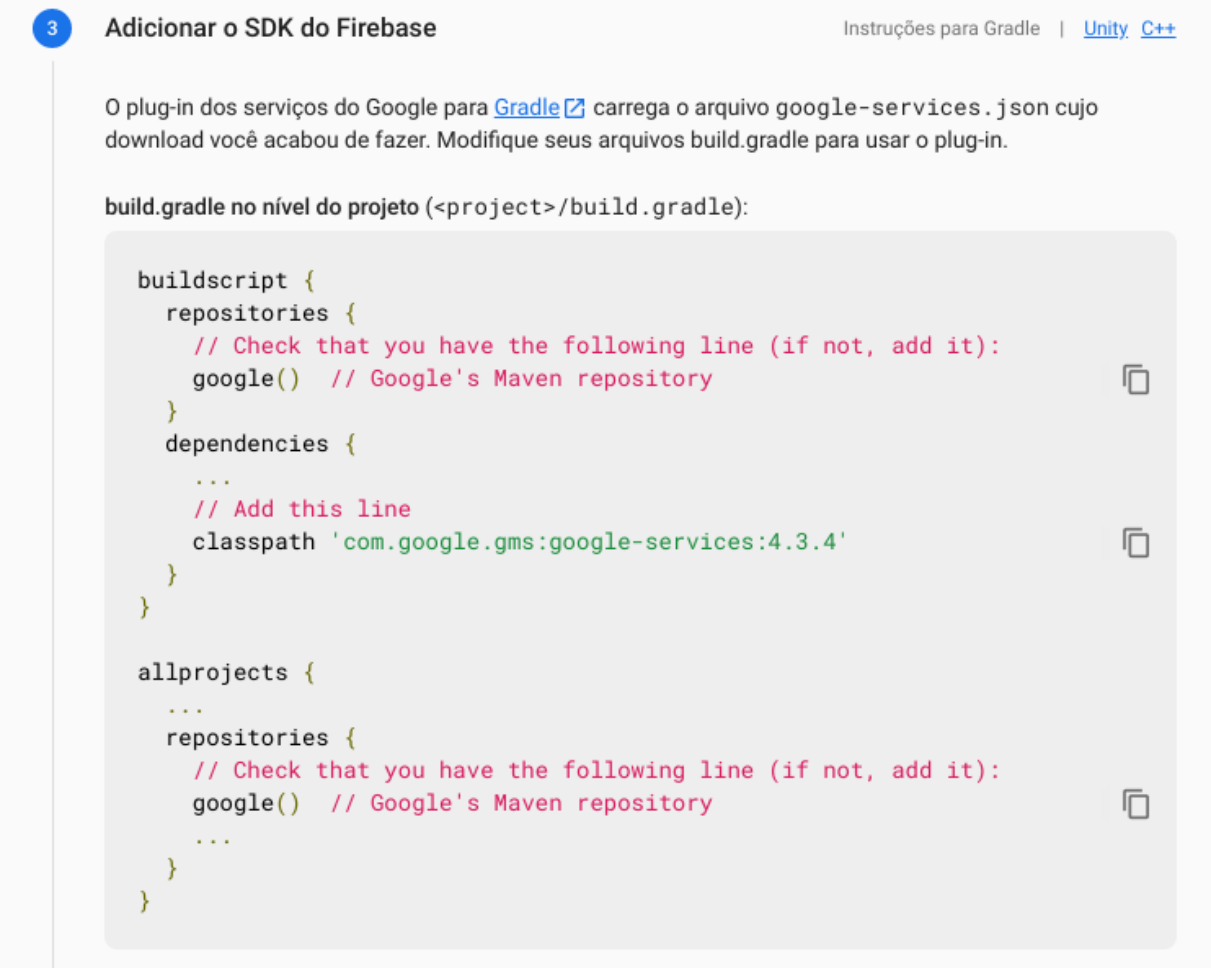

Figura 54 – Tela para adicionar o SDK do Firebase

Passo 8: Em seguida, o usuário deve abrir com o Android Studio o arquivo build.gradle dentro do diretório *app*, localizado no diretório raiz do projeto e adicionar as linhas de código indicadas pelos comentários em rosa nas seções indicadas na **Figura 55** e por fim clicar em *sync now* dentro do Android Studio.

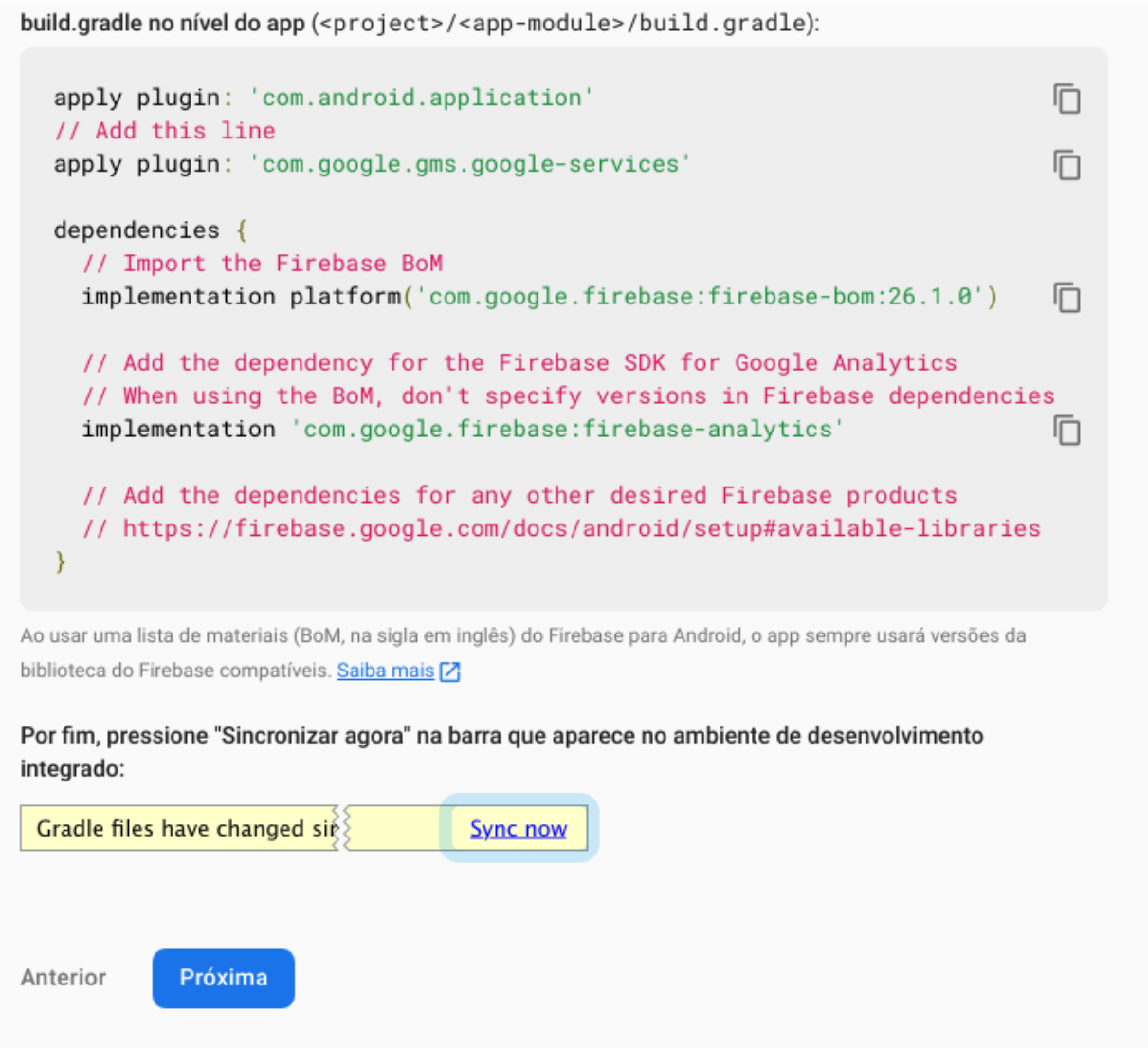

Figura 55 – Tela adicional para adicionar o SDK do Firebase

Passo 9: Após clicar no botão Próxima ilustrado na figura acima, o usuário deve então clicar no botão Continuar no console, onde será redirecionado para o console da aplicação do Firebase criada.

Passo 10: No menu à esquerda, o usuário deve clicar em Authentication para ser redirecionado para outra página.

Passo 11: O usuário deve então clicar no botão Primeiros passos, onde será redirecionado para uma nova página.

Passo 12: O usuário deve então clicar em E-mail/senha e então ativar, como mostra a **Figura 56**.

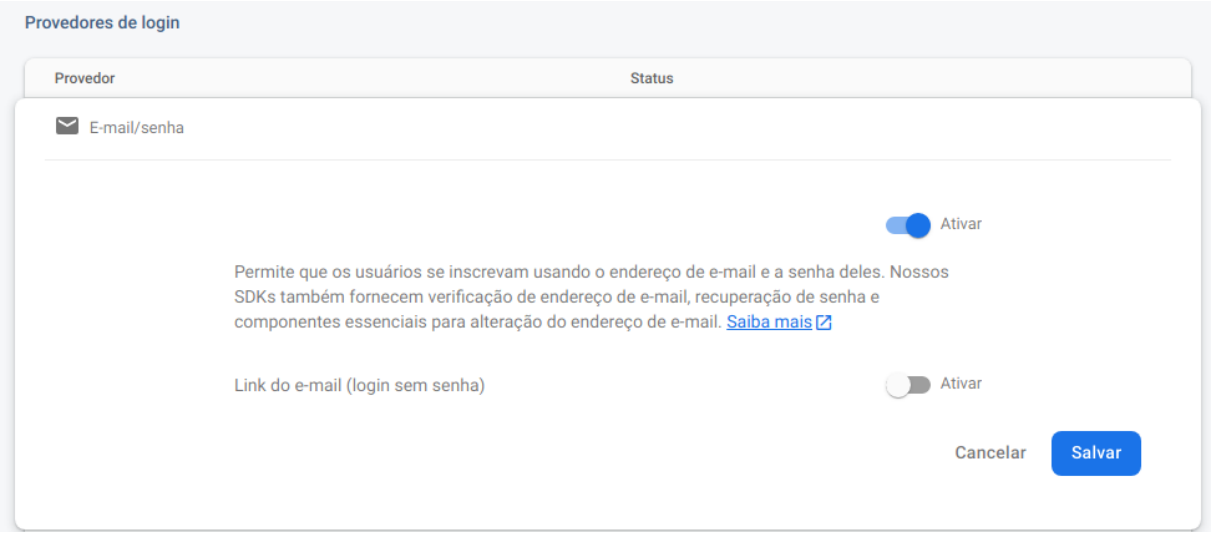

Figura 56 – Tela para ativar a opção de login com E-mail/senha

Passo 13: Após clicar em Salvar na figura acima, o usuário deve clicar em Realtime Database no menu à esquerda.

Passo 14: Em seguida o usuário deve clicar no botão Criar banco de dados, em seguida deve clicar no botão Iniciar modo de teste e por último no botão ativar.

Passo 15: O usuário então deve clicar no botão com três pontos na vertical, ao lado dos botões de + e -, como exemplifica a **Figura 57**.

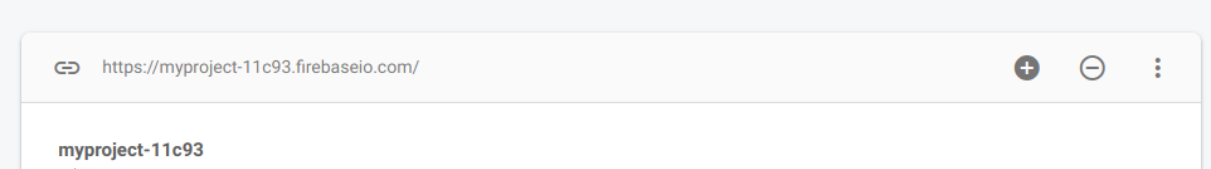

Figura 57 – Tela para editar o banco de dados

Passo 16: Após clicar no botão indicado acima, o usuário deve então clicar na opção Importar o JSON e em seguida selecionar o arquivo exercícios.json, presente no Apêndice D deste trabalho.

Passo 17: O usuário deve clicar em Storage, no menu à esquerda. Em seguida clicar no botão Primeiros passos, depois no botão Próxima e em seguida no botão Concluir.

Passo 18: O usuário então deve clicar na aba Rules e substituir todo o texto com as regras com o texto mostrado na **Figura 58**.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if true;
    }
  }
}
```

Figura 58 – Tela com as regras para a função de armazenamento do Firebase

Passo 19: Por último, o usuário deve clicar no botão concluir e o Firebase estará completamente configurado para o uso do aplicativo ConnectFit.

APÊNDICE C – Configuração da API Google Maps

Este apêndice mostrará a configuração da API Google Maps em um projeto Android. Ele se baseia nas instruções oficiais disponibilizadas pela documentação da própria API Google Maps em https://developers.google.com/maps/documentation/android-sdk/config?hl=pt-br.

Passo 1: Adicionar com.google.android.gms:play-services-location:17.1.0 na seção de dependências do arquivo build.gladle do aplicativo Android.

Passo 2: Acessar o link https://cloud.google.com/console/google/maps-apis/overview?hl=pt-br.

Passo 3: Clicar na lista suspensa do projeto e selecionar o projeto Android.

Passo 4: Clicar no botão de menu e depois escolher APIs e serviços > Credenciais.

Passo 5: Na página Credenciais, clicar em Criar credenciais > Chave de API, onde uma caixa de diálogo aparecerá com a chave recém-criada, conforme mostra a **Figura 59**.

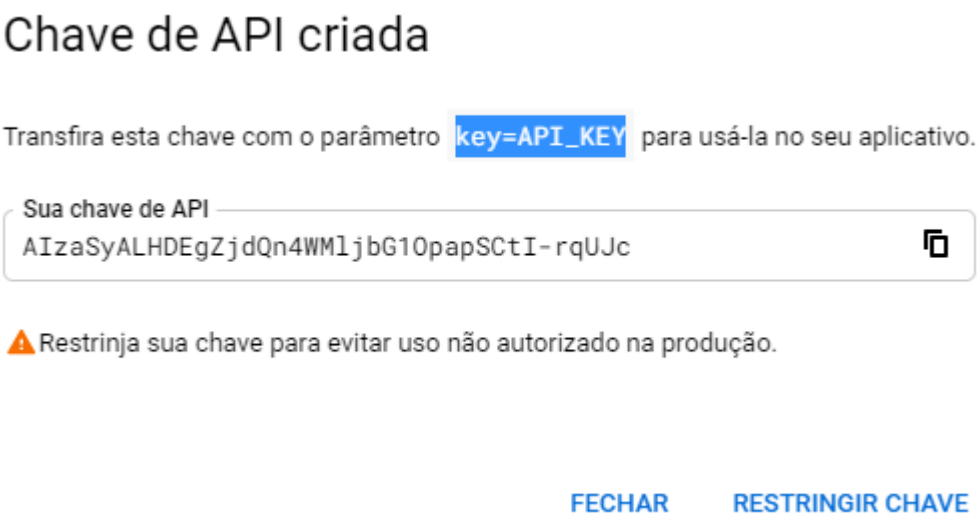

Figura 59 – Tela com chave de API criada

Passo 6: Copiar a chave gerada acima e insira no arquivo AndroidManifest.xml do Android dentro da tag <meta-data />, conforme mostra a **Figura 60**, substituindo o texto YOUR_API_KEY pela chave gerada anteriormente. A tag <meta-data /> deve ser inserida antes do fechamento da tag <application />.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY"/>
```

Figura 60 – Tela de inserção da chave de API no projeto Android

Passo 7: A API Google Maps então estará pronta para ser usada no aplicativo Android.

APÊNDICE D – Principais arquivos do projeto ConnectFit

Classe ExerciseAdapter:

```
package com.vitorflores.connectfit.Adapter;

import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Client.ProfessionalProfileActivity;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.LookUpExerciseActivity;
import com.vitorflores.connectfit.MessageActivity;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Professional.ClientProfileActivity;
import com.vitorflores.connectfit.R;

import java.util.List;

public class ExerciseAdapter extends RecyclerView.Adapter<ExerciseAdapter.ViewHolder> {

    private Context mContext;
    private List<Exercise> mExercises;
```

```
    public ExerciseAdapter(Context mContext, List<Exercise> mExercises) {
        this.mContext = mContext;
        this.mExercises = mExercises;
    }

    @NonNull
    @Override
    public ExerciseAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(mContext).inflate(R.layout.item_exercise, parent, false);
        return new ExerciseAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ExerciseAdapter.ViewHolder holder, int position) {
        final Exercise exercise = mExercises.get(position);

        holder.name.setText(exercise.getName());
        holder.itemView.setOnClickListener(view -> {

            Intent intent = new Intent(mContext, LookUpExerciseActivity.class);
            intent.putExtra("name", mExercises.get(position).getName());
            mContext.startActivity(intent);
        });
    }

    @Override
    public int getItemCount() {
        return mExercises.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView name;

        public ViewHolder(View itemView) {
            super(itemView);

            name = itemView.findViewById(R.id.textExerciseName);
        }
    }
}
```

Classe MessageAdapter:

```
package com.vitorflores.connectfit.Adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.vitorflores.connectfit.Model.Chat;
import com.vitorflores.connectfit.R;

import java.util.List;

public class MessageAdapter extends RecyclerView.Adapter<MessageAdapter.ViewHolder> {

    public static final int MSG_TYPE_LEFT = 0;
    public static final int MSG_TYPE_RIGHT = 1;
    private Context mContext;
    private List<Chat> mChat;

    FirebaseUser mFirebaseUser;

    public MessageAdapter(Context mContext, List<Chat> mChat) {
        this.mChat = mChat;
        this.mContext = mContext;
    }

    // Show the message with the right styling
    @NonNull
    @Override
    public MessageAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view;

        if (viewType == MSG_TYPE_RIGHT) {
            view = LayoutInflater.from(mContext).inflate(R.layout.chat_item_right, parent, false);
        } else {
            view = LayoutInflater.from(mContext).inflate(R.layout.chat_item_left, parent, false);
        }
```

```
        return new MessageAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MessageAdapter.ViewHolder holder, int position) {
        Chat chat = mChat.get(position);
        holder.textMessage.setText(chat.getMessage());
    }

    @Override
    public int getItemCount() {
        return mChat.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView textMessage;

        public ViewHolder(View itemView) {
            super(itemView);

            textMessage = itemView.findViewById(R.id.textMessage);
        }
    }

    // Decide if the message was sent by the logged in user
    @Override
    public int getItemViewType(int position) {
        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        if (mChat.get(position).getSender().equals(mFirebaseUser.getUid())) {
            return MSG_TYPE_RIGHT;
        } else {
            return MSG_TYPE_LEFT;
        }
    }
}
```

Classe UserAdapter:

```
package com.vitorflores.connectfit.Adapter;

import android.content.Context;
```

```
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Client.ProfessionalProfileActivity;
import com.vitorflores.connectfit.MessageActivity;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Professional.ClientProfileActivity;
import com.vitorflores.connectfit.R;

import java.util.List;

// Adapter to show users on a list
public class UserAdapter extends RecyclerView.Adapter<UserAdapter.ViewHolder> {

    private Context mContext;
    private User currentUser;
    private List<User> mUsers;

    public UserAdapter(Context mContext, List<User> mUsers) {
        this.mContext = mContext;
        this.mUsers = mUsers;
    }


    // Show a user in the list
    @NonNull
```

```
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(mContext).inflate(R.layout.item_user, parent, false);
        return new UserAdapter.ViewHolder(view);
    }

    // Set the profile image and username of the profile
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        final User user = mUsers.get(position);

        holder.username.setText(user.getUserName());

        if (user.getImageURL().equals("default")) {
            holder.profile_image.setImageResource(R.mipmap.ic_launcher);
        } else {
            Glide.with(mContext).load(user.getImageURL()).into(holder.profile_image);
        }

        // If you click on a single item, redirect to the correct profile
        holder.itemView.setOnClickListener(view -> {
            FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
            DatabaseReference reference =
FirebaseDatabase.getInstance().getReference("Users").child(firebaseUser.getUid());
            //Intent intent;

            reference.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    Intent intent;
                    currentUser = dataSnapshot.getValue(User.class);

                    assert currentUser != null;
                    if (mContext instanceof AvailableProfessionals
                            || (currentUser.getAccountType() != null
                            && !currentUser.getAccountType().equals("pro")
                            && mContext instanceof ContractsActivity)) {
                        intent = new Intent(mContext, ProfessionalProfileActivity.class);
                    } else if (currentUser.getAccountType() != null
                            && currentUser.getAccountType().equals("pro")
                            && mContext instanceof ContractsActivity) {
                        intent = new Intent(mContext, ClientProfileActivity.class);
                    } else {
```

```
                    intent = new Intent(mContext, MessageActivity.class);
                }

                intent.putExtra("id", user.getId());
                mContext.startActivity(intent);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });


    });
  }

  @Override
  public int getItemCount() {
    return mUsers.size();
  }

  public class ViewHolder extends RecyclerView.ViewHolder {
    public TextView username;
    public ImageView profile_image;

    public ViewHolder(View itemView) {
      super(itemView);

      username = itemView.findViewById(R.id.textProfileName);
      profile_image = itemView.findViewById(R.id.imgProfileAvatar);
    }
  }
}
```

Classe WorkoutAdapter:

```
package com.vitorflores.connectfit.Adapter;

import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.vitorflores.connectfit.LookUpExerciseActivity;
import com.vitorflores.connectfit.LookUpWorkoutActivity;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.Workout;
import com.vitorflores.connectfit.R;

import java.util.List;

public class WorkoutAdapter extends RecyclerView.Adapter<WorkoutAdapter.ViewHolder> {

    private Context mContext;
    private List<Workout> mWorkouts;

    public WorkoutAdapter(Context mContext, List<Workout> mWorkouts) {
        this.mContext = mContext;
        this.mWorkouts = mWorkouts;
    }

    @NonNull
    @Override
    public WorkoutAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(mContext).inflate(R.layout.item_workout, parent, false);
        return new WorkoutAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull WorkoutAdapter.ViewHolder holder, int position) {
        final Workout workout = mWorkouts.get(position);

        holder.name.setText(workout.getName());
        Log.d("teste", "Status: " + workout.getIsPending());
        if (workout.getIsPending()) {
            holder.status.setText("Pendente");
        } else {
            holder.status.setText("Concluído");
        }
        holder.date.setText(workout.getDate());
        holder.itemView.setOnClickListener(view -> {
```

```
            Intent intent = new Intent(mContext, LookUpWorkoutActivity.class);
            intent.putExtra("name", mWorkouts.get(position).getName());
            intent.putExtra("proId", mWorkouts.get(position).getProId());
            intent.putExtra("clientId", mWorkouts.get(position).getClientId());
            intent.putExtra("id", mWorkouts.get(position).getId());
            mContext.startActivity(intent);
        });
    }

    @Override
    public int getItemCount() {
        return mWorkouts.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView name, status, date;

        public ViewHolder(View itemView) {
            super(itemView);

            name = itemView.findViewById(R.id.textWorkoutName);
            status = itemView.findViewById(R.id.textStatus);
            date = itemView.findViewById(R.id.textDate);
        }
    }
}
```

Classe AvailableProfessionals:

```
package com.vitorflores.connectfit.Client;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.UserAdapter;
import com.vitorflores.connectfit.ChatsActivity;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.ExercisesActivity;
import com.vitorflores.connectfit.LocationSearchActivity;
import com.vitorflores.connectfit.LoginUserActivity;
import com.vitorflores.connectfit.MainActivity;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.R;

import org.apache.commons.lang3.StringUtils;

import java.util.ArrayList;
import java.util.List;

public class AvailableProfessionals extends AppCompatActivity {

    // Visual elements
    FloatingActionButton mBtnLocationSearch;
    EditText mFieldSearch;
    private RecyclerView listAvailableProfessionals;
    private UserAdapter userAdapter;

    private List<User> mProfessionals;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_available_professionals);

// Toolbar settings
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setTitle(R.string.bar_title_available_professionals);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
toolbar.setNavigationOnClickListener(view -> finish());

// Visual elements
mBtnLocationSearch = findViewById(R.id.btnLocationSearch);

// Go to the maps activity, to find a personal trainer near you
mBtnLocationSearch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(AvailableProfessionals.this, LocationSearchActivity.class);
        startActivity(intent);
        //finish();
    }
});

// Watch for characters added on the search field
mFieldSearch = findViewById(R.id.fieldSearch);
mFieldSearch.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
        String searchTerm = StringUtils.stripAccents(charSequence.toString().toLowerCase());
        searchUsers(searchTerm);
    }

    @Override
    public void afterTextChanged(Editable editable) {
        /*Intent intent = new Intent(AvailableProfessionals.this, LocationSearchActivity.class);
        startActivity(intent);
        //finish();*/
    }
});
```

```
        // Show the list of available professionals
        listAvailableProfessionals = findViewById(R.id.listAvailableProfessionals);
        //recyclerView.setHasFixedSize(true);
        listAvailableProfessionals.setLayoutManager(new LinearLayoutManager(AvailableProfessionals.this));

        mProfessionals = new ArrayList<>();
        readProfessionals();
    }

    private void searchUsers(String searchTerm) {
        final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        // Search for usernames starting with the text on the search field
        Query query = FirebaseDatabase.getInstance().getReference("Users").orderByChild("searchableName")
                .startAt(searchTerm)
                .endAt(searchTerm + "\uf8ff");

        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                // Show the search results
                mProfessionals.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    User user = snapshot.getValue(User.class);

                    assert user != null;
                    assert firebaseUser != null;
                    if (user.getId() != null && !user.getId().equals(firebaseUser.getUid()) &&
user.getAccountType().equals("pro")) {
                        mProfessionals.add(user);
                    }
                }

                userAdapter = new UserAdapter(AvailableProfessionals.this, mProfessionals);
                listAvailableProfessionals.setAdapter(userAdapter);

            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
```

```
        });
    }


    // Make a list of the available professionals and show it to the user
    private void readProfessionals() {
        final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference("Users");

        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                // Show all the professionals if the searchbar is empty
                if (mFieldSearch.getText().toString().equals("")) {
                    mProfessionals.clear();

                    // Loop through the "pro" users
                    for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                        User user = snapshot.getValue(User.class);

                        assert user != null;
                        assert firebaseUser != null;
                        if (user.getId() != null && !user.getId().equals(firebaseUser.getUid()) &&
user.getAccountType().equals("pro")) {
                            mProfessionals.add(user);
                        }
                    }

                    // Take the professionals list and use an adapter to show on the screen
                    userAdapter = new UserAdapter(AvailableProfessionals.this, mProfessionals);
                    listAvailableProfessionals.setAdapter(userAdapter);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }

    // Create the options menu on the toolbar
    @Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(AvailableProfessionals.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(AvailableProfessionals.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(AvailableProfessionals.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(AvailableProfessionals.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
```

```
            case R.id.exercises:
                startActivity(new Intent(AvailableProfessionals.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(AvailableProfessionals.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(AvailableProfessionals.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe ProfessionalProfileActivity:

```
package com.vitorflores.connectfit.Client;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.ScrollView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.ChatsActivity;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.ExercisesActivity;
import com.vitorflores.connectfit.LoginUserActivity;
import com.vitorflores.connectfit.MainActivity;
import com.vitorflores.connectfit.MessageActivity;
import com.vitorflores.connectfit.Model.Contract;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Professional.ClientProfileActivity;
import com.vitorflores.connectfit.R;
import com.vitorflores.connectfit.WorkoutsActivity;

import de.hdodenhof.circleimageview.CircleImageView;

public class ProfessionalProfileActivity extends AppCompatActivity {

    Button mBtnHireProfessional, mBtnMessageProfessional, mBtnWorkouts;
    CircleImageView mImgProfileAvatarView;
    TextView mTextProfileName, mTextProfileLocation, mTextProfileCref, mTextProfileAbout;

    ProgressBar mProgressBar;
    ScrollView mLayoutContainer;

    // Firebase Auth
    FirebaseUser mFirebaseUser;

    private User user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_professional_profile);

        // Toolbar
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_profile);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

```
toolbar.setNavigationOnClickListener(view -> finish());

// Don't show the data on the screen until everything is loaded
mLayoutContainer = findViewById(R.id.layoutContainer);
mLayoutContainer.setVisibility(View.GONE);

// Visual elements
mProgressBar = findViewById(R.id.progressBar);
mBtnHireProfessional = findViewById(R.id.btnHireProfessional);
mBtnMessageProfessional = findViewById(R.id.btnMessageProfessional);
mBtnWorkouts = findViewById(R.id.btnWorkouts);
mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
mTextProfileName = findViewById(R.id.textProfileName);
mTextProfileLocation = findViewById(R.id.textProfileLocation);
mTextProfileCref = findViewById(R.id.textProfileCref);
mTextProfileAbout = findViewById(R.id.textProfileAbout);

// Firebase user
mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

final Intent intent = getIntent();
String proId = intent.getStringExtra("id");
DatabaseReference usersReference = FirebaseDatabase.getInstance().getReference("Users").child(proId);

// Show the user info on the screen
usersReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        user = dataSnapshot.getValue(User.class);

        // Hide the progress bar and show the profile info to the user
        mProgressBar.setVisibility(View.GONE);
        mLayoutContainer.setVisibility(View.VISIBLE);

        // Show the profile picture of the user
        if (user.getImageURL() != null && user.getImageURL().equals("default")) {
            mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher);
        } else {
            Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
        }

        // Show the profile name
        mTextProfileName.setText(user.getUserName());
```

```
            // Show the location of the user
            mTextProfileLocation.setVisibility(View.GONE);
            if (user.getCity() != null && user.getState() != null) {
                if (!user.getCity().equals("") && !user.getState().equals("")) {
                    String location = user.getCity() + "/" + user.getState();
                    mTextProfileLocation.setVisibility(View.VISIBLE);
                    mTextProfileLocation.setText(location);
                }
            }

            // Show the cref of the user
            mTextProfileCref.setVisibility(View.GONE);
            if (user.getCref() != null) {
                if (!user.getCref().equals("")) {
                    String cref = "CREF " + user.getCref();
                    mTextProfileCref.setVisibility(View.VISIBLE);
                    mTextProfileCref.setText(cref);
                }
            }

            // Show the 'about me' section
            mTextProfileAbout.setText(user.getAbout());

            // Show the hire button
            mBtnHireProfessional.setVisibility(View.GONE);
            DatabaseReference contractsReference =
FirebaseDatabase.getInstance().getReference("Contracts").child(mFirebaseUser.getUid()).child(proId);
            contractsReference.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    if (dataSnapshot.exists()) {
                        Contract contract = dataSnapshot.getValue(Contract.class);

                        assert contract != null;
                        // If the contract is pending, disable the 'hire' button
                        if (contract.getStatus() != null && contract.getStatus().equals("pending")) {
                            mBtnHireProfessional.setText(R.string.action_pending);
                            mBtnHireProfessional.setClickable(false);
                            mBtnHireProfessional.setEnabled(false);
                        } else {
                            // If the contract is active, show the button to end the contract
                            mBtnHireProfessional.setText(R.string.action_dismiss);
```

```
                    // Show the 'workouts' button
                    mBtnWorkouts.setVisibility(View.VISIBLE);
                }
            }

            // If there's no contract yet, just shows the default 'hire' button
            mBtnHireProfessional.setVisibility(View.VISIBLE);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {}
});

mBtnHireProfessional.setOnClickListener(view -> {
    DatabaseReference contractsReference = FirebaseDatabase.getInstance().getReference("Contracts").child(mFirebaseUser.getUid()).child(proId);
    contractsReference.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            // If there's an active contract and you click to end the contract:
            if (dataSnapshot.exists()) {
                contractsReference.removeValue();
                mBtnHireProfessional.setText(R.string.action_hire);
            } else {
                // Send a request to start a contract
                contractsReference.child("id").setValue(proId);
                contractsReference.child("status").setValue("pending");
                mBtnHireProfessional.setText(R.string.action_pending);
            }

            contractsReference.removeEventListener(this);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });

    // The same as above, but on the professional's side
```

```
        DatabaseReference contractsReference2 = FirebaseDatabase.getInstance().getReference("Contracts").child(proId).child(mFirebaseUser.getUid());
        contractsReference2.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    contractsReference2.removeValue();
                } else {
                    contractsReference2.child("id").setValue(mFirebaseUser.getUid());
                    contractsReference2.child("status").setValue("pending");
                }

                contractsReference2.removeEventListener(this);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });
    });

    mBtnMessageProfessional.setOnClickListener(view -> {
        // Redirects to the chat screen with the user
        Intent intent2 = new Intent(ProfessionalProfileActivity.this, MessageActivity.class);
        intent2.putExtra("id", intent.getStringExtra("id"));
        startActivity(intent2);
    });

    mBtnWorkouts.setOnClickListener(view -> {
        // Redirects to the user workouts
        Intent intent2 = new Intent(ProfessionalProfileActivity.this, WorkoutsActivity.class);
        intent2.putExtra("proId", intent.getStringExtra("id"));
        intent2.putExtra("clientId", mFirebaseUser.getUid());
        startActivity(intent2);
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
```

```
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(ProfessionalProfileActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(ProfessionalProfileActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(ProfessionalProfileActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(ProfessionalProfileActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(ProfessionalProfileActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
```

```
            case R.id.chats:
                startActivity(new Intent(ProfessionalProfileActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(ProfessionalProfileActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe AccountClientSettingsActivity:

```
package com.vitorflores.connectfit.Client;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.Continuation;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.StorageTask;
import com.google.firebase.storage.UploadTask;
import com.vitorflores.connectfit.ChatsActivity;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.ExercisesActivity;
import com.vitorflores.connectfit.LoginUserActivity;
import com.vitorflores.connectfit.MainActivity;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Professional.ClientProfileActivity;
import com.vitorflores.connectfit.R;

import java.util.HashMap;

import de.hdodenhof.circleimageview.CircleImageView;

public class AccountClientSettingsActivity extends AppCompatActivity {

    // Visual elements
    private CircleImageView mImgProfileAvatarView;
    private TextView mTextHeight, mTextWeight, mTextCity, mTextState;
    private Button mBtnSave;

    // Firebase Authentication
    private FirebaseUser mFirebaseUser;
    private DatabaseReference mUsersReference;

    // Image upload
    StorageReference storageReference;
    private static final int IMAGE_REQUEST = 1;
    private Uri mImageUri;
    private StorageTask mUploadTask;

    User user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_account_client_settings);

// Toolbar
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setTitle(R.string.bar_title_profile_settings);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
toolbar.setNavigationOnClickListener(view -> finish());

// Visual elements
mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
mTextHeight = findViewById(R.id.fieldHeight);
mTextWeight = findViewById(R.id.fieldWeight);
mTextCity = findViewById(R.id.fieldCity);
mTextState = findViewById(R.id.fieldState);
mBtnSave = findViewById(R.id.btnSave);

// Firebase Auth
mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
mUsersReference = FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());

// Image upload
storageReference = FirebaseStorage.getInstance().getReference("uploads");

// Fill the fields with the current values
mUsersReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        user = dataSnapshot.getValue(User.class);

        assert user != null;
        if (user.getImageURL() != null && user.getImageURL().equals("default")) {
            mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher_round);
        } else {
            Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
        }

        mTextHeight.setText(user.getHeight());
        mTextWeight.setText(user.getWeight());
        mTextCity.setText(user.getCity());
        mTextState.setText(user.getState());
    }
```

```
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });

    // Change the profile image
    mImgProfileAvatarView.setOnClickListener(view -> openImage());

    // Update the account details
    mBtnSave.setOnClickListener(v -> {
        final String height = mTextHeight.getText().toString();
        final String weight = mTextWeight.getText().toString();
        final String city = mTextCity.getText().toString();
        final String state = mTextState.getText().toString();

        mUsersReference =
FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());

        // Combine the field values into one hashmap
        HashMap<String, java.lang.Object> hashMap = new HashMap<>();
        hashMap.put("height", height);
        hashMap.put("weight", weight);
        hashMap.put("city", city);
        hashMap.put("state", state);

        // Update the account details and redirects to the main screen
        mUsersReference.updateChildren(hashMap).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Intent intent = new Intent(AccountClientSettingsActivity.this, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText(AccountClientSettingsActivity.this, "Error", Toast.LENGTH_SHORT).show();
            }
        });
    });
}

// Open the screen to choose a new image from the gallery
private void openImage() {
```

```
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(intent, IMAGE_REQUEST);
}

// Get the extension of a file
private String getFileExtension(Uri uri) {
    ContentResolver contentResolver = getApplicationContext().getContentResolver();
    MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();

    return mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri));
}

// Upload the new image to the firebase server
private void uploadImage() {
    // Progress dialog while the image is uploading
    final ProgressDialog pd = new ProgressDialog(AccountClientSettingsActivity.this);
    pd.setMessage("Uploading");
    pd.show();

    // Upload the image
    if (mImageUri != null) {
        final StorageReference fileReference = storageReference.child(System.currentTimeMillis()
                + "." + getFileExtension(mImageUri));

        mUploadTask = fileReference.putFile(mImageUri);
        // If the app failed to upload, show the error message
        // If the image was successful uploaded, update the profile picture with the new image
        mUploadTask.continueWithTask((Continuation<UploadTask.TaskSnapshot, Task<Uri>>) task -> {
            if (!task.isSuccessful()) {
                throw task.getException();
            }

            return fileReference.getDownloadUrl();
        }).addOnCompleteListener(task -> {
            // If the image was successful uploaded, update the profile picture with the new image
            if (task.isSuccessful()) {
                String uri = task.getResult().toString();

                mUsersReference                                                    =
FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());
                HashMap<String, java.lang.Object> map = new HashMap<>();
```

```
                map.put("imageURL", uri);
                mUsersReference.updateChildren(map);
            } else {
                Toast.makeText(AccountClientSettingsActivity.this, "Error", Toast.LENGTH_SHORT).show();
            }

            pd.dismiss();
        }).addOnFailureListener(e -> {
            // If the app failed to upload, show the error message
            Toast.makeText(AccountClientSettingsActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
            pd.dismiss();
        });
    } else {
        Toast.makeText(AccountClientSettingsActivity.this, "No image selected", Toast.LENGTH_SHORT).show();
    }
}

// Wait for the user to choose the new image, after that, call the method to upload it to the server
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == IMAGE_REQUEST && resultCode == RESULT_OK
        && data != null && data.getData() != null) {
        mImageUri = data.getData();

        if (mUploadTask != null && mUploadTask.isInProgress()) {
            Toast.makeText(AccountClientSettingsActivity.this, "Upload in progress",
Toast.LENGTH_SHORT).show();
        } else {
            uploadImage();
        }
    }
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
```

```
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(AccountClientSettingsActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(AccountClientSettingsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(AccountClientSettingsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(AccountClientSettingsActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(AccountClientSettingsActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
```

```
            startActivity(new Intent(AccountClientSettingsActivity.this, ChatsActivity.class));
            finish();
            return true;
        // Log out
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(AccountClientSettingsActivity.this, LoginUserActivity.class));
            finish();
            return true;
    }

    return false;
  }
}
```

Classe Chat:

```
package com.vitorflores.connectfit.Model;

// This class is responsible to store a single message sent/received
// It contains the sender of the message, the receiver and its content
public class Chat {

    private String sender;
    private String receiver;
    private String message;

    public Chat(String sender, String receiver, String message) {
        this.sender = sender;
        this.receiver = receiver;
        this.message = message;
    }

    public Chat() {}

    public String getSender() {
        return sender;
    }

    public void setSender(String sender) {
        this.sender = sender;
    }
```

```
    public String getReceiver() {
        return receiver;
    }

    public void setReceiver(String receiver) {
        this.receiver = receiver;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Classe Chatlist:

```
package com.vitorflores.connectfit.Model;

// This class is responsible to store the ids of the people you had a chat with
// Each instance of this class store the other person's id
public class Chatlist {

    public String id;

    public Chatlist(String id) {
        this.id = id;
    }

    public Chatlist() {}

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```

Classe Contract:

```
package com.vitorflores.connectfit.Model;

public class Contract {
    private String id;
    private String status;

    public Contract(String id, String status) {
        this.id = id;
        this.status = status;
    }

    public Contract() {}

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

Classe Exercise:

```
package com.vitorflores.connectfit.Model;

public class Exercise {
    private String name;
    private String description;
    private String videoLink;

    public Exercise(String name, String description, String videoLink) {
        this.name = name;
        this.description = description;
        this.videoLink = videoLink;
```

```
    }

    public Exercise() {}

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getVideoLink() {
        return videoLink;
    }

    public void setVideoLink(String videoLink) {
        this.videoLink = videoLink;
    }
}
```

Classe User:

```
package com.vitorflores.connectfit.Model;

// This class is responsible to store the users info, like his id and name
public class User {
    private String id;
    private String userName;
    private String searchableName;
    private String accountType;
    private String imageURL;
    private String city;
    private String state;
    private String height;
```

```
    private String weight;
    private String about;
    private String cref;
    private Double latitude;
    private Double longitude;

    public User() {}

    public User(String id, String userName, String searchableName, String accountType, String imageURL, String city, String state, String height, String weight, String about, String cref) {
        this.id = id;
        this.userName = userName;
        this.searchableName = searchableName;
        this.accountType = accountType;
        this.imageURL = imageURL;
        this.city = city;
        this.state = state;
        this.height = height;
        this.weight = weight;
        this.about = about;
        this.cref = cref;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getSearchableName() {
        return searchableName;
    }
```

```
public void setSearchableName(String searchableName) {
    this.searchableName = searchableName;
}

public String getAccountType() {
    return accountType;
}

public void setAccountType(String accountType) {
    this.accountType = accountType;
}

public String getImageURL() {
    return imageURL;
}

public void setImageURL(String imageURL) {
    this.imageURL = imageURL;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

public String getState() {
    return state;
}

public void setState(String state) {
    this.state = state;
}

public String getHeight() {
    return height;
}

public void setHeight(String height) {
    this.height = height;
}
```

```
    public String getWeight() {
        return weight;
    }

    public void setWeight(String weight) {
        this.weight = weight;
    }

    public String getAbout() {
        return about;
    }

    public void setAbout(String about) {
        this.about = about;
    }

    public String getCref() {
        return cref;
    }

    public void setCref(String cref) {
        this.cref = cref;
    }

    public Double getLatitude() {
        return latitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    public Double getLongitude() {
        return longitude;
    }

    public void setLongitude(Double longitude) {
        this.longitude = longitude;
    }
}
```

Classe Workout:

```
package com.vitorflores.connectfit.Model;

import java.util.Date;
import java.util.List;

public class Workout {
    private String id;
    private String proId;
    private String clientId;
    private String name;
    private String date;
    private boolean isPending;
    private String workout;

    public Workout(String id, String proId, String clientId, String name, String date, boolean isPending, String workout) {
        this.id = id;
        this.proId = proId;
        this.clientId = clientId;
        this.name = name;
        this.date = date;
        this.isPending = isPending;
        this.workout = workout;
    }

    public Workout() {}

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getProId() {
        return proId;
    }

    public void setProId(String proId) {
        this.proId = proId;
    }
```

```
    public String getClientId() {
        return clientId;
    }

    public void setClientId(String clientId) {
        this.clientId = clientId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public boolean getIsPending() {
        return isPending;
    }

    public void setIsPending(boolean pending) {
        isPending = pending;
    }

    public String getWorkout() {
        return workout;
    }

    public void setWorkout(String workout) {
        this.workout = workout;
    }
}
```

Classe FirebaseMessaging:

```
package com.vitorflores.connectfit.Notifications;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;
import com.vitorflores.connectfit.MessageActivity;

public class FirebaseMessaging extends FirebaseMessagingService {

    @Override
    public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);

        String sent = remoteMessage.getData().get("sent");
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        // Set the correct way of showing the notification, based on the android version
        if (firebaseUser != null && sent.equals(firebaseUser.getUid())) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                sendOreoNotification(remoteMessage);
            } else {
                sendNotification(remoteMessage);
            }
        }
    }
```

```
    // Notifications for Android Oreo and above
    private void sendOreoNotification(RemoteMessage remoteMessage) {
        // Get the notification data
        String user = remoteMessage.getData().get("user");
        String icon = remoteMessage.getData().get("icon");
        String title = remoteMessage.getData().get("title");
        String body = remoteMessage.getData().get("body");

        // Build the notification
        RemoteMessage.Notification notification = remoteMessage.getNotification();
        int j = Integer.parseInt(user.replaceAll("[\\D]", ""));
        Intent intent = new Intent(this, MessageActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("id", user);
        intent.putExtras(bundle);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, j, intent, PendingIntent.FLAG_ONE_SHOT);

        Uri defaultSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        OreoNotification oreoNotification = new OreoNotification(this);
        Notification.Builder builder = oreoNotification.getOreoNotification(title, body, pendingIntent, defaultSound,
icon);

        int i = 0;

        if (j > 0) {
            i = j;
        }

        // Show the notification
        oreoNotification.getManager().notify(i, builder.build());
    }

    // Notifications for android versions below Oreo (8)
    private void sendNotification(RemoteMessage remoteMessage) {
        String user = remoteMessage.getData().get("user");
        String icon = remoteMessage.getData().get("icon");
        String title = remoteMessage.getData().get("title");
        String body = remoteMessage.getData().get("body");

        RemoteMessage.Notification notification = remoteMessage.getNotification();
        int j = Integer.parseInt(user.replaceAll("[\\D]", ""));
```

```
        Intent intent = new Intent(this, MessageActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("id", user);
        intent.putExtras(bundle);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, j, intent, PendingIntent.FLAG_ONE_SHOT);

        Uri defaultSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
                .setSmallIcon(Integer.parseInt(icon))
                .setContentTitle(title)
                .setContentText(body)
                .setAutoCancel(true)
                .setSound(defaultSound)
                .setContentIntent(pendingIntent);
        NotificationManager notifManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        int i = 0;
        if (j > 0) {
            i = j;
        }

        // Show the notification
        notifManager.notify(i, builder.build());
    }
}
```

Classe AccountProSettingsActivity:

```
package com.vitorflores.connectfit.Professional;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.accounts.Account;
import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Looper;
import android.provider.Settings;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.tasks.Continuation;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.StorageTask;
import com.google.firebase.storage.UploadTask;
import com.vitorflores.connectfit.ChatsActivity;
```

```
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.ExercisesActivity;
import com.vitorflores.connectfit.LoginUserActivity;
import com.vitorflores.connectfit.MainActivity;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.R;
import com.vitorflores.connectfit.WorkoutsActivity;

import java.util.HashMap;

import de.hdodenhof.circleimageview.CircleImageView;

public class AccountProSettingsActivity extends AppCompatActivity {

    // Visual elements
    private CircleImageView mImgProfileAvatarView;
    private TextView mTextCref, mTextAbout, mTextCity, mTextState;
    private Button mBtnSave;
    private Switch mSwitchUpdateLocation;

    // Firebase Authentication
    private FirebaseUser mFirebaseUser;
    private DatabaseReference mUsersReference;

    // Image upload
    StorageReference storageReference;
    private static final int IMAGE_REQUEST = 1;
    private Uri mImageUri;
    private StorageTask mUploadTask;

    // Location
    FusedLocationProviderClient fusedLocationProviderClient;

    User user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_account_pro_settings);

        // Toolbar
        Toolbar toolbar = findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_profile_settings);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Visual elements
        mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
        mTextAbout = findViewById(R.id.fieldAbout);
        mTextCref = findViewById(R.id.fieldCref);
        mTextCity = findViewById(R.id.fieldCity);
        mTextState = findViewById(R.id.fieldState);
        mSwitchUpdateLocation = findViewById(R.id.switchUpdateLocation);
        mBtnSave = findViewById(R.id.btnSave);

        // Firebase Auth
        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        mUsersReference = FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());

        // Image upload
        storageReference = FirebaseStorage.getInstance().getReference("uploads");

        // Location
        fusedLocationProviderClient                                                                    =
LocationServices.getFusedLocationProviderClient(AccountProSettingsActivity.this);

        // Fill the fields with the current values
        mUsersReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                user = dataSnapshot.getValue(User.class);

                assert user != null;
                if (user.getImageURL() != null && user.getImageURL().equals("default")) {
                    mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher);
                } else {
                    Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
                }

                mTextAbout.setText(user.getAbout());
                mTextCref.setText(user.getCref());
                mTextCity.setText(user.getCity());
                mTextState.setText(user.getState());
            }
```

```
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });

    // Change the profile image
    mImgProfileAvatarView.setOnClickListener(view -> openImage());

    // Update the account details
    mBtnSave.setOnClickListener(v -> {
        final String about = mTextAbout.getText().toString();
        final String cref = mTextCref.getText().toString();
        final String city = mTextCity.getText().toString();
        final String state = mTextState.getText().toString();
        final boolean updateLocation = mSwitchUpdateLocation.isChecked();

        mUsersReference =
FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());

        // Combine the field values into one hashmap
        HashMap<String, java.lang.Object> hashMap = new HashMap<>();

        // Update location
        if (updateLocation) {
            // Check condition
            if (ActivityCompat.checkSelfPermission(AccountProSettingsActivity.this,
                Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(AccountProSettingsActivity.this,
                Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

                getCurrentLocation();

            } else {
                // If permission denied, request permission
                //ActivityCompat.requestPermissions(AccountProSettingsActivity.this,
                //                                  new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,Manifest.permission.ACCESS_COARSE_LOCATION},
100);
                Toast.makeText(AccountProSettingsActivity.this, "Ative a localização nas permissões do aplicativo",
Toast.LENGTH_SHORT).show();
            }
```

```
        }

        hashMap.put("about", about);
        hashMap.put("cref", cref);
        hashMap.put("city", city);
        hashMap.put("state", state);



        // Update the account details and redirects to the main screen
        updateUserInfo(hashMap);
    });
}

/*@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    // check condition
    if (requestCode == 100 && grantResults.length > 0 && (grantResults[0] + grantResults[1] == PackageManager.PERMISSION_GRANTED)) {
        getCurrentLocation();
    } else {
        // Permission denied
        Toast.makeText(AccountProSettingsActivity.this, "Permissão negada para acessar localização", Toast.LENGTH_SHORT).show();
    }
}*/

@SuppressLint("MissingPermission")
private void getCurrentLocation() {
    LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

    if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
        || locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {

        fusedLocationProviderClient.getLastLocation().addOnCompleteListener(new OnCompleteListener<Location>() {
            @Override
            public void onComplete(@NonNull Task<Location> task) {
                Location location = task.getResult();

                if (location != null) {
                    // Set latitude and longitude
                    HashMap<String, java.lang.Object> hashMap = new HashMap<>();
                    hashMap.put("latitude", location.getLatitude());
```

```
                hashMap.put("longitude", location.getLongitude());

                updateUserLocation(hashMap);
            } else {
                LocationRequest locationRequest = new LocationRequest()
                        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
                        .setInterval(10000)
                        .setFastestInterval(1000)
                        .setNumUpdates(1);

                LocationCallback locationCallback = new LocationCallback() {
                    @Override
                    public void onLocationResult(LocationResult locationResult) {
                        Location location1 = locationResult.getLastLocation();
                        // Set lat and long
                        HashMap<String, java.lang.Object> hashMap = new HashMap<>();
                        hashMap.put("latitude", location1.getLatitude());
                        hashMap.put("longitude", location1.getLongitude());

                        updateUserLocation(hashMap);
                    }
                };

                fusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback,
Looper.myLooper());
            }
          }
        });
    } else {
        // When location service is disabled
        //startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS)
        //    .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK));
        Toast.makeText(AccountProSettingsActivity.this, "Ative a localização nas configurações do Android",
Toast.LENGTH_SHORT).show();
    }
  }

  private void updateUserLocation(HashMap<String, java.lang.Object> hashMap) {
    mUsersReference.updateChildren(hashMap).addOnCompleteListener(task1 -> {
        if (!task1.isSuccessful()) {
            Toast.makeText(AccountProSettingsActivity.this, "Erro ao atualizar localização",
Toast.LENGTH_SHORT).show();
        }
    });
```

```
}

private void updateUserInfo(HashMap<String, java.lang.Object> hashMap) {
    mUsersReference.updateChildren(hashMap).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            Intent intent = new Intent(AccountProSettingsActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            finish();
        } else {
            Toast.makeText(AccountProSettingsActivity.this, "Error", Toast.LENGTH_SHORT).show();
        }
    });
}

// Open the screen to choose a new image from the gallery
private void openImage() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(intent, IMAGE_REQUEST);
}

// Get the extension of a file
private String getFileExtension(Uri uri) {
    ContentResolver contentResolver = getApplicationContext().getContentResolver();
    MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();

    return mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri));
}

// Upload the new image to the firebase server
private void uploadImage() {
    // Progress dialog while the image is uploading
    final ProgressDialog pd = new ProgressDialog(AccountProSettingsActivity.this);
    pd.setMessage("Uploading");
    pd.show();

    // Upload the image
    if (mImageUri != null) {
        final StorageReference fileReference = storageReference.child(System.currentTimeMillis()
                + "." + getFileExtension(mImageUri));
```

```
            mUploadTask = fileReference.putFile(mImageUri);
            // If the app failed to upload, show the error message
            // If the image was successful uploaded, update the profile picture with the new image
            mUploadTask.continueWithTask(new Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
                @Override
                public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task) throws Exception {
                    if (!task.isSuccessful()) {
                        throw task.getException();
                    }

                    return fileReference.getDownloadUrl();
                }
            }).addOnCompleteListener(task -> {
                // If the image was successful uploaded, update the profile picture with the new image
                if (task.isSuccessful()) {
                    String uri = task.getResult().toString();

                    mUsersReference                                                                    =
FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());
                    HashMap<String, java.lang.Object> map = new HashMap<>();
                    map.put("imageURL", uri);
                    mUsersReference.updateChildren(map);
                } else {
                    Toast.makeText(AccountProSettingsActivity.this, "Error", Toast.LENGTH_SHORT).show();
                }

                pd.dismiss();
            }).addOnFailureListener(e -> {
                // If the app failed to upload, show the error message
                Toast.makeText(AccountProSettingsActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
                pd.dismiss();
            });
        } else {
            Toast.makeText(AccountProSettingsActivity.this, "No image selected", Toast.LENGTH_SHORT).show();
        }
    }

    // Wait for the user to choose the new image, after that, call the method to upload it to the server
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == IMAGE_REQUEST && resultCode == RESULT_OK
```

```
            && data != null && data.getData() != null) {
        mImageUri = data.getData();

        if (mUploadTask != null && mUploadTask.isInProgress()) {
            Toast.makeText(AccountProSettingsActivity.this, "Upload in progress", Toast.LENGTH_SHORT).show();
        } else {
            uploadImage();
        }
    }
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(AccountProSettingsActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(AccountProSettingsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
```

```
            // Pending contracts
            case R.id.pendingContracts:
                intent = new Intent(AccountProSettingsActivity.this, ContractsActivity.class);
                intent.putExtra("typeContracts", "pending");
                startActivity(intent);
                finish();
                return true;
            // Available Professionals Screen
            case R.id.availableProfessionals:
                startActivity(new Intent(AccountProSettingsActivity.this, AvailableProfessionals.class));
                finish();
                return true;
            // Exercises Screen
            case R.id.exercises:
                startActivity(new Intent(AccountProSettingsActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(AccountProSettingsActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(AccountProSettingsActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe ClientProfileActivity:

```
package com.vitorflores.connectfit.Professional;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
```

```
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.ScrollView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.ChatsActivity;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.ContractsActivity;
import com.vitorflores.connectfit.ExercisesActivity;
import com.vitorflores.connectfit.LoginUserActivity;
import com.vitorflores.connectfit.MainActivity;
import com.vitorflores.connectfit.MessageActivity;
import com.vitorflores.connectfit.Model.Contract;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.R;
import com.vitorflores.connectfit.WorkoutsActivity;

import de.hdodenhof.circleimageview.CircleImageView;

public class ClientProfileActivity extends AppCompatActivity {

    Button mBtnHireProfessional, mBtnMessageProfessional, mBtnWorkouts;
    CircleImageView mImgProfileAvatarView;
    TextView mTextProfileName, mTextProfileLocation;

    ProgressBar mProgressBar;
    ScrollView mLayoutContainer;

    // Firebase Auth
```

```
    FirebaseUser mFirebaseUser;

    private User user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client_profile);

        // Toolbar
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_profile);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Don't show the data on the screen until everything is loaded
        mLayoutContainer = findViewById(R.id.layoutContainer);
        mLayoutContainer.setVisibility(View.GONE);

        // Visual elements
        mProgressBar = findViewById(R.id.progressBar);
        mBtnHireProfessional = findViewById(R.id.btnHireProfessional);
        mBtnMessageProfessional = findViewById(R.id.btnMessageProfessional);
        mBtnWorkouts = findViewById(R.id.btnWorkouts);
        mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
        mTextProfileName = findViewById(R.id.textProfileName);
        mTextProfileLocation = findViewById(R.id.textProfileLocation);

        // Firebase user
        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        final Intent intent = getIntent();
        String clientId = intent.getStringExtra("id");
        DatabaseReference                                    usersReference                                    =
FirebaseDatabase.getInstance().getReference("Users").child(clientId);



        // Show the user info on the screen
        usersReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                user = dataSnapshot.getValue(User.class);
```

```
            // Hide the progress bar and show the profile info to the user
            mProgressBar.setVisibility(View.GONE);
            mLayoutContainer.setVisibility(View.VISIBLE);

            // Show the profile picture of the user
            if (user.getImageURL() != null && user.getImageURL().equals("default")) {
                mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher);
            } else {
                Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
            }

            // Show the profile name
            mTextProfileName.setText(user.getUserName());

            // Show the location of the user
            mTextProfileLocation.setVisibility(View.GONE);
            if (user.getCity() != null && user.getState() != null) {
                if (!user.getCity().equals("") && !user.getState().equals("")) {
                    String location = user.getCity() + "/" + user.getState();
                    mTextProfileLocation.setVisibility(View.VISIBLE);
                    mTextProfileLocation.setText(location);
                }
            }

            // Show the hire button
            mBtnHireProfessional.setVisibility(View.GONE);
            DatabaseReference contractsReference =
FirebaseDatabase.getInstance().getReference("Contracts").child(mFirebaseUser.getUid()).child(clientId);
            contractsReference.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    if (dataSnapshot.exists()) {
                        Contract contract = dataSnapshot.getValue(Contract.class);

                        assert contract != null;
                        // If the contract is pending, disable the 'hire' button
                        if (contract.getStatus() != null && contract.getStatus().equals("active")) {
                            mBtnHireProfessional.setText(R.string.action_dismiss);
                            // Show the 'workouts' button
                            mBtnWorkouts.setVisibility(View.VISIBLE);
                        }
                    }
```

```
                // If there's no contract yet, just shows the default 'accept' button
                mBtnHireProfessional.setVisibility(View.VISIBLE);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {}
});


    mBtnHireProfessional.setOnClickListener(view -> {
        DatabaseReference contractsReference = FirebaseDatabase.getInstance().getReference("Contracts").child(mFirebaseUser.getUid()).child(clientId);
        DatabaseReference contractsReference2 = FirebaseDatabase.getInstance().getReference("Contracts").child(clientId).child(mFirebaseUser.getUid());
        contractsReference.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                contractsReference.removeEventListener(this);

                // If there's a pending contract and you click to accept it
                if (dataSnapshot.exists()) {
                    Contract contract = dataSnapshot.getValue(Contract.class);

                    assert contract != null;
                    if (contract.getStatus() != null && contract.getStatus().equals("pending")) {
                        contractsReference.child("status").setValue("active");
                        contractsReference2.child("status").setValue("active");
                        mBtnHireProfessional.setText(R.string.action_dismiss);
                    } else {
                        contractsReference.removeValue();
                        contractsReference2.removeValue();
                        finish();
                    }
                }
            }

            @Override
```

```
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });

        // The same as above, but on the client's side

        /*contractsReference2.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    contractsReference2.removeValue();
                } else {
                    contractsReference2.child("id").setValue(mFirebaseUser.getUid());
                    contractsReference2.child("status").setValue("active");
                }

                contractsReference2.removeEventListener(this);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });*/


    });

    mBtnMessageProfessional.setOnClickListener(view -> {
        // Redirects to the chat screen with the user
        Intent intent2 = new Intent(ClientProfileActivity.this, MessageActivity.class);
        intent2.putExtra("id", intent.getStringExtra("id"));
        startActivity(intent2);
    });

    mBtnWorkouts.setOnClickListener(view -> {
        // Redirects to the user workouts
        Intent intent3 = new Intent(ClientProfileActivity.this, WorkoutsActivity.class);
        intent3.putExtra("proId", mFirebaseUser.getUid());
        intent3.putExtra("clientId", intent.getStringExtra("id"));
        startActivity(intent3);
    });
}

// Create the options menu on the toolbar
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(ClientProfileActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(ClientProfileActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(ClientProfileActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(ClientProfileActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
```

```
            case R.id.exercises:
                startActivity(new Intent(ClientProfileActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(ClientProfileActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(ClientProfileActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe ChatsActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
```

```
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.iid.FirebaseInstanceId;
import com.vitorflores.connectfit.Adapter.UserAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Chatlist;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Notifications.Token;

import java.util.ArrayList;
import java.util.List;

public class ChatsActivity extends AppCompatActivity {

    // Visual elements
    private RecyclerView chatsLayout;

    // Firebase Authentication
    FirebaseUser mFirebaseUser;
    DatabaseReference mUsersReference, mChatlistReference;

    private UserAdapter userAdapter;
    private List<User> mUsers;
    private List<Chatlist> usersList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chats);

        // Toolbar
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_chats);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Prepare the container before showing the active chats in it
        chatsLayout = findViewById(R.id.chatsLayout);
        chatsLayout.setHasFixedSize(true);
        chatsLayout.setLayoutManager(new LinearLayoutManager(ChatsActivity.this));

        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
```

```
        usersList = new ArrayList<>();

        // If you had a chat with a user, add his userid to a list
        mChatlistReference = FirebaseDatabase.getInstance().getReference("Chatlist").child(mFirebaseUser.getUid());
        mChatlistReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                usersList.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Chatlist chatlist = snapshot.getValue(Chatlist.class);
                    usersList.add(chatlist);
                }

                // Take the list of userids and show their names on the screen
                showChats();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });

        updateToken(FirebaseInstanceId.getInstance().getToken());
    }

    private void updateToken(String token) {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference("Tokens");
        Token token1 = new Token(token);
        reference.child(mFirebaseUser.getUid()).setValue(token1);
    }

    // Show active chats
    private void showChats() {
        mUsers = new ArrayList<>();

        // Loop through the users and select the ones who have their id on your chatlist
        // After that, add the user and his info to a list
        mUsersReference = FirebaseDatabase.getInstance().getReference("Users");
        mUsersReference.addValueEventListener(new ValueEventListener() {
            @Override
```

```
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mUsers.clear();

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                User user = snapshot.getValue(User.class);

                for (Chatlist chatlist : usersList) {
                    assert user != null;
                    if (user.getId().equals(chatlist.getId())) {
                        mUsers.add(user);
                    }
                }
            }

            // Take this list of users and their info and show them on the screen
            userAdapter = new UserAdapter(ChatsActivity.this, mUsers);
            chatsLayout.setAdapter(userAdapter);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
```

```
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(ChatsActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(ChatsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(ChatsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(ChatsActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(ChatsActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
            startActivity(new Intent(ChatsActivity.this, ChatsActivity.class));
            finish();
            return true;
        // Log out
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(ChatsActivity.this, LoginUserActivity.class));
            finish();
            return true;
```

```
        }

        return false;
    }
}
```

Classe ContractsActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.RelativeLayout;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.UserAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Contract;
import com.vitorflores.connectfit.Model.User;

import java.util.ArrayList;
import java.util.List;

public class ContractsActivity extends AppCompatActivity {

    // Visual elements
    RelativeLayout mContractsContainer, mPendingContractsContainer, mActiveContractsContainer;
    RecyclerView mContractsLayout, mPendingContractsLayout, mActiveContractsLayout;
```

```
// Firebase Authentication
FirebaseUser mFirebaseUser;
DatabaseReference mUsersReference, mContractsReference;

private UserAdapter userAdapter, pendingUserAdapter, activeUserAdapter;
private List<User> mUsers, mPendingUsers, mActiveUsers;
private List<Contract> usersList;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_contracts);

    Intent intent = getIntent();
    final String typeContracts = intent.getStringExtra("typeContracts");

    // Toolbar
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    if (typeContracts.equals("active")) {
        getSupportActionBar().setTitle(R.string.section_title_active);
    } else {
        getSupportActionBar().setTitle(R.string.section_title_pending);
    }

    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    toolbar.setNavigationOnClickListener(view -> finish());

    // Visual elements
    mContractsContainer = findViewById(R.id.contractsContainer);
    mContractsLayout = findViewById(R.id.contractsLayout);
    mContractsLayout.setHasFixedSize(true);
    mContractsLayout.setLayoutManager(new LinearLayoutManager(ContractsActivity.this));

    // Firebase Authentication
    mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

    usersList = new ArrayList<>();

    // If you have a contract (pending or active), add his userid to a list
    mContractsReference                                                    =
FirebaseDatabase.getInstance().getReference("Contracts").child(mFirebaseUser.getUid());
```

```
    mContractsReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            usersList.clear();

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                Contract contract = snapshot.getValue(Contract.class);

                assert contract != null;
                if (contract.getStatus() != null && contract.getStatus().equals(typeContracts)) {
                    usersList.add(contract);
                }
            }

            // Take the list of userids and show their names on the screen
            showContracts();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}

private void showContracts() {
    mUsers = new ArrayList<>();

    // Loop through the users and select the ones who have their id on your contracts list
    // After that, add the user and his info to a list
    mUsersReference = FirebaseDatabase.getInstance().getReference("Users");
    mUsersReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mUsers.clear();

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                User user = snapshot.getValue(User.class);

                for (Contract contract : usersList) {
                    assert user != null;
                    assert contract != null;
                    if (user.getId() != null && contract.getId() != null && user.getId().equals(contract.getId())) {
                        mUsers.add(user);
                    }
```

```
            }
        }

        // Take this list of users and their info and show them on the screen
        if (!mUsers.isEmpty()) {
            userAdapter = new UserAdapter(ContractsActivity.this, mUsers);
            mContractsLayout.setAdapter(userAdapter);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {}
  });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
  final String accountType = accountPrefs.getString("accountType", null);

  if (accountType.equals("pro")) {
    getMenuInflater().inflate(R.menu.menu_pro, menu);
  } else {
    getMenuInflater().inflate(R.menu.menu_client, menu);
  }

  return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
  Intent intent;
  switch (item.getItemId()) {
    // Main Screen
    case R.id.mainScreen:
      startActivity(new Intent(ContractsActivity.this, MainActivity.class));
      finish();
      return true;
    // Active contracts
    case R.id.activeContracts:
      intent = new Intent(ContractsActivity.this, ContractsActivity.class);
```

```
                intent.putExtra("typeContracts", "active");
                startActivity(intent);
                finish();
                return true;
            // Pending contracts
            case R.id.pendingContracts:
                intent = new Intent(ContractsActivity.this, ContractsActivity.class);
                intent.putExtra("typeContracts", "pending");
                startActivity(intent);
                finish();
                return true;
            // Available Professionals Screen
            case R.id.availableProfessionals:
                startActivity(new Intent(ContractsActivity.this, AvailableProfessionals.class));
                finish();
                return true;
            // Exercises Screen
            case R.id.exercises:
                startActivity(new Intent(ContractsActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(ContractsActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(ContractsActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe EditWorkoutActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.Workout;

import java.util.HashMap;

public class EditWorkoutActivity extends AppCompatActivity {

    // Visual elements
    private EditText mFieldName, mFieldDate, mFieldWorkout;
    private Switch mFieldFinished;
    private Button mBtnSave;

    // Firebase Auth
    FirebaseUser mFirebaseUser;
    DatabaseReference mReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_workout);
        SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
        final String accountType = accountPrefs.getString("accountType", null);
```

```
        Intent intent = getIntent();
        final String workoutId = intent.getStringExtra("workoutId");
        final String proId = intent.getStringExtra("proId");
        final String clientId = intent.getStringExtra("clientId");

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        if (workoutId != null) {
            getSupportActionBar().setTitle(R.string.bar_title_edit_workout);
        } else {
            getSupportActionBar().setTitle(R.string.bar_title_add_workout);
        }

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Visual elements
        mFieldName = findViewById(R.id.fieldName);
        mFieldFinished = findViewById(R.id.switchFinished);
        mFieldDate = findViewById(R.id.fieldDate);
        mFieldWorkout = findViewById(R.id.fieldWorkout);
        mBtnSave = findViewById(R.id.btnSave);

        // If edit workout screen
        if (workoutId != null && clientId != null) {
            mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
            if (accountType.equals("pro")) {
                mReference =
FirebaseDatabase.getInstance().getReference("Workouts").child(mFirebaseUser.getUid()).child(clientId).child(wor
koutId);
            } else {
                mReference =
FirebaseDatabase.getInstance().getReference("Workouts").child(proId).child(mFirebaseUser.getUid()).child(worko
utId);
            }

            mReference.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    Workout workout = dataSnapshot.getValue(Workout.class);

                    assert workout != null;
                    if (workout != null) {
```

```
                mFieldName.setText(workout.getName());
                if (!workout.getIsPending()) {
                    mFieldFinished.setChecked(true);
                }
                mFieldDate.setText(workout.getDate());
                mFieldWorkout.setText(workout.getWorkout());
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
  }

  mBtnSave.setOnClickListener(view -> {
    final String name = mFieldName.getText().toString();

    if (!name.equals("")) {
      if (workoutId != null) {
        saveWorkout(true, proId, clientId, workoutId);
      } else {
        saveWorkout(false, proId, clientId, workoutId);
      }
    } else {
      Toast.makeText(EditWorkoutActivity.this, "Erro ao salvar treino", Toast.LENGTH_SHORT).show();
    }
  });
}

private void saveWorkout(boolean editMode, String proId, String clientId, String workoutId) {
  SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
  final String accountType = accountPrefs.getString("accountType", null);
  final String name = mFieldName.getText().toString();
  final boolean isFinishedChecked = mFieldFinished.isChecked();
  final String date = mFieldDate.getText().toString();
  final String workout = mFieldWorkout.getText().toString();
  FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();


  assert firebaseUser != null;
```

```
HashMap<String, Object> hashMap = new HashMap<>();
if (accountType.equals("pro")) {
    hashMap.put("proId", firebaseUser.getUid());
} else {
    hashMap.put("proId", proId);
}

hashMap.put("clientId", clientId);
hashMap.put("name", name);
if (isFinishedChecked) {
    hashMap.put("isPending", false);
} else {
    hashMap.put("isPending", true);
}
hashMap.put("date", date);
hashMap.put("workout", workout);

if (editMode) {
    if (accountType.equals("pro")) {
        mReference = FirebaseDatabase.getInstance().getReference("Workouts").child(firebaseUser.getUid()).child(clientId).child(workoutId);
    } else {
        mReference = FirebaseDatabase.getInstance().getReference("Workouts").child(proId).child(firebaseUser.getUid()).child(workoutId);
    }
    mReference.updateChildren(hashMap).addOnCompleteListener(task1 -> {
        if (task1.isSuccessful()) {
            Toast.makeText(EditWorkoutActivity.this, "Treino salvo com sucesso", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(EditWorkoutActivity.this, WorkoutsActivity.class);
            intent.putExtra("proId", proId);
            intent.putExtra("clientId", clientId);
            startActivity(intent);
            finish();
        } else {
            Toast.makeText(EditWorkoutActivity.this, "Erro ao salvar treino", Toast.LENGTH_SHORT).show();
        }
    });
} else {
    if (accountType.equals("pro")) {
        mReference = FirebaseDatabase.getInstance().getReference("Workouts").child(firebaseUser.getUid()).child(clientId);
    } else {
```

```
            mReference = FirebaseDatabase.getInstance().getReference("Workouts").child(proId).child(firebaseUser.getUid());
        }
        final String id = mReference.push().getKey();
        hashMap.put("id", id);
        mReference.child(id).setValue(hashMap).addOnCompleteListener(task1 -> {
            if (task1.isSuccessful()) {
                Toast.makeText(EditWorkoutActivity.this, "Treino salvo com sucesso", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(EditWorkoutActivity.this, WorkoutsActivity.class);
                intent.putExtra("proId", proId);
                intent.putExtra("clientId", clientId);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText(EditWorkoutActivity.this, "Erro ao salvar treino", Toast.LENGTH_SHORT).show();
            }
        });
    }
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
```

```
            startActivity(new Intent(EditWorkoutActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(EditWorkoutActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(EditWorkoutActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(EditWorkoutActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(EditWorkoutActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
            startActivity(new Intent(EditWorkoutActivity.this, ChatsActivity.class));
            finish();
            return true;
        // Log out
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(EditWorkoutActivity.this, LoginUserActivity.class));
            finish();
            return true;
      }

      return false;
    }
}
```

Classe ExercisesActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.ExerciseAdapter;
import com.vitorflores.connectfit.Adapter.UserAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Contract;
import com.vitorflores.connectfit.Model.Exercise;

import java.util.ArrayList;
import java.util.List;

public class ExercisesActivity extends AppCompatActivity {

    // Visual elements
    RelativeLayout mExercisesContainer;
    RecyclerView mExercisesLayout;
```

```
FirebaseUser mFirebaseUser;
DatabaseReference mExercisesReference;

private List<Exercise> exercisesList;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_exercises);

    // Toolbar settings
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle(R.string.bar_title_exercises);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    toolbar.setNavigationOnClickListener(view -> finish());

    // Visual elements
    mExercisesContainer = findViewById(R.id.exercisesContainer);
    mExercisesLayout = findViewById(R.id.exercisesLayout);
    mExercisesLayout.setHasFixedSize(true);
    mExercisesLayout.setLayoutManager(new LinearLayoutManager(ExercisesActivity.this));

    // Firebase Authentication
    mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

    exercisesList = new ArrayList<>();

    // If you have added an exercise, add its id to a list
    mExercisesReference = FirebaseDatabase.getInstance().getReference("Exercises");
    mExercisesReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            exercisesList.clear();

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                Exercise exercise = snapshot.getValue(Exercise.class);

                assert exercise != null;
                exercisesList.add(exercise);
            }

            // Take the list of ids and show its names on the screen
```

```
            if (!exercisesList.isEmpty()) {
                ExerciseAdapter exerciseAdapter = new ExerciseAdapter(ExercisesActivity.this, exercisesList);
                mExercisesLayout.setAdapter(exerciseAdapter);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(ExercisesActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(ExercisesActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
```

```
            // Pending contracts
            case R.id.pendingContracts:
                intent = new Intent(ExercisesActivity.this, ContractsActivity.class);
                intent.putExtra("typeContracts", "pending");
                startActivity(intent);
                finish();
                return true;
            // Available Professionals Screen
            case R.id.availableProfessionals:
                startActivity(new Intent(ExercisesActivity.this, AvailableProfessionals.class));
                finish();
                return true;
            // Exercises Screen
            case R.id.exercises:
                startActivity(new Intent(ExercisesActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(ExercisesActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(ExercisesActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe LocationSearchActivity:

```
package com.vitorflores.connectfit;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
```

```
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentActivity;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.UserAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Client.ProfessionalProfileActivity;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Professional.AccountProSettingsActivity;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Objects;

public class LocationSearchActivity extends AppCompatActivity implements OnMapReadyCallback {

    private List<User> mProfessionals;
    private GoogleMap mMap;

    // Location
    FusedLocationProviderClient fusedLocationProviderClient;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_location_search);

        // Location
        fusedLocationProviderClient                                                    =
LocationServices.getFusedLocationProviderClient(LocationSearchActivity.this);

        // Toolbar settings
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_nearby_professionals);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    // Make a list of the available professionals and show it to the user
    private void readProfessionals() {
        final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference("Users");

        reference.addValueEventListener(new ValueEventListener() {
            @Override
```

```
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                mProfessionals.clear();

                // Loop through the "pro" users
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    User user = snapshot.getValue(User.class);

                    assert user != null;
                    assert firebaseUser != null;
                    if (user.getId() != null && !user.getId().equals(firebaseUser.getUid()) &&
user.getAccountType().equals("pro")) {
                        mProfessionals.add(user);
                    }
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mProfessionals = new ArrayList<>();
        final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        DatabaseReference reference =
FirebaseDatabase.getInstance().getReference("Users").child(firebaseUser.getUid());

        mMap = googleMap;

        // Make a list of all the professionals
        readProfessionals();
```

```
        // Show your location and the location of the professionals on the map
        // Check condition
        if (ActivityCompat.checkSelfPermission(LocationSearchActivity.this,
                Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(LocationSearchActivity.this,
                Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED)
{


            getCurrentLocationAndShowMap();



        } else {
            // If permission denied, request permission
            Toast.makeText(LocationSearchActivity.this, "Ative a localização nas permissões do aplicativo",
Toast.LENGTH_SHORT).show();
            finish();
        }
    }


    @SuppressLint("MissingPermission")
    private void getCurrentLocationAndShowMap() {
        LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);


        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
                || locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {


            fusedLocationProviderClient.getLastLocation().addOnCompleteListener(new
OnCompleteListener<Location>() {
                @Override
                public void onComplete(@NonNull Task<Location> task) {
                    Location location = task.getResult();


                    if (location != null) {
                        // Set latitude and longitude
                        LatLng currentLocation = new LatLng(location.getLatitude(), location.getLongitude());


                        showMapWithMarkers(currentLocation);
                    } else {
                        LocationRequest locationRequest = new LocationRequest()
                                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
                                .setInterval(10000)
                                .setFastestInterval(1000)
                                .setNumUpdates(1);
```

```
                LocationCallback locationCallback = new LocationCallback() {
                    @Override
                    public void onLocationResult(LocationResult locationResult) {
                        Location location1 = locationResult.getLastLocation();
                        // Set lat and long
                        LatLng currentLocation1 = new LatLng(location1.getLatitude(), location1.getLongitude());
                        showMapWithMarkers(currentLocation1);
                    }
                };

                fusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback,
Looper.myLooper());
            }
        }
    });
} else {
    // When location service is disabled
    Toast.makeText(LocationSearchActivity.this, "Ative a localização nas configurações do Android",
Toast.LENGTH_SHORT).show();
    finish();
}
}

private void showMapWithMarkers(LatLng currentLocation) {
    // Add a marker in your location and move the camera
    mMap.clear();
    Marker marker = mMap.addMarker(new MarkerOptions()
            .position(currentLocation)
            .title("Minha Localização")
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED)));
    marker.setTag(FirebaseAuth.getInstance().getCurrentUser().getUid());

    // Loop through the professionals list and add their markers if they have location enabled
    for (User user : mProfessionals) {
        if (user.getLatitude() != null && user.getLongitude() != null) {
            marker = mMap.addMarker(new MarkerOptions()
                    .position(new LatLng(user.getLatitude(), user.getLongitude()))
                    .title(user.getUserName())
                    .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE)));
            marker.setTag(user.getId());
        }
    }
```

```
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLocation, 14));

    // If the user click on a name, redirect to the professional profile
    mMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {

        @Override
        public void onInfoWindowClick(Marker marker) {
            String currentUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();
            if (!Objects.requireNonNull(marker.getTag()).toString().equals(currentUserId)) {
                Intent intent = new Intent(LocationSearchActivity.this, ProfessionalProfileActivity.class);
                intent.putExtra("id", marker.getTag().toString());
                startActivity(intent);
            }
        }
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(LocationSearchActivity.this, MainActivity.class));
            finish();
            return true;
```

```
            // Active contracts
            case R.id.activeContracts:
                intent = new Intent(LocationSearchActivity.this, ContractsActivity.class);
                intent.putExtra("typeContracts", "active");
                startActivity(intent);
                finish();
                return true;
            // Pending contracts
            case R.id.pendingContracts:
                intent = new Intent(LocationSearchActivity.this, ContractsActivity.class);
                intent.putExtra("typeContracts", "pending");
                startActivity(intent);
                finish();
                return true;
            // Available Professionals Screen
            case R.id.availableProfessionals:
                startActivity(new Intent(LocationSearchActivity.this, AvailableProfessionals.class));
                finish();
                return true;
            // Exercises Screen
            case R.id.exercises:
                startActivity(new Intent(LocationSearchActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(LocationSearchActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(LocationSearchActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe LoginUserActivity:

```
package com.vitorflores.connectfit;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Model.User;

public class LoginUserActivity extends AppCompatActivity {

    private Button mBtnLogin, mBtnRegister;
    private EditText mEmailView, mPasswordView;
    private TextView mForgotPasswordView;

    private FirebaseAuth mAuth;
    private FirebaseUser mFirebaseUser;

    @Override
    protected void onStart() {
        super.onStart();

        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        // Check if logged in
```

```
        if (mFirebaseUser != null) {
            Intent intent = new Intent(LoginUserActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            finish();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_user);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.app_name);

        mBtnLogin = findViewById(R.id.btnLogin);
        mBtnRegister = findViewById(R.id.btnRegister);
        mEmailView = findViewById(R.id.fieldEmail);
        mPasswordView = findViewById(R.id.fieldPassword);
        mForgotPasswordView = findViewById(R.id.btnForgotPassword);

        mAuth = FirebaseAuth.getInstance();

        mBtnLogin.setOnClickListener(v -> {
            String email = mEmailView.getText().toString().toLowerCase();
            String password = mPasswordView.getText().toString();

            if (TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
                Toast.makeText(LoginUserActivity.this, R.string.system_message_empty_fields,
Toast.LENGTH_SHORT).show();
            } else {
                mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        // Save the account type on a sharedPreference
                        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
                        DatabaseReference usersReference =
FirebaseDatabase.getInstance().getReference("Users").child(firebaseUser.getUid());
                        usersReference.addValueEventListener(new ValueEventListener() {
                            @Override
                            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                                User user = dataSnapshot.getValue(User.class);
```

```
                    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
                    SharedPreferences.Editor editor;

                    assert user != null;
                    if (user.getAccountType().equals("pro")) {
                        editor = accountPrefs.edit();
                        editor.putString("accountType", "pro");
                        editor.commit();
                    } else {
                        editor = accountPrefs.edit();
                        editor.putString("accountType", "client");
                        editor.commit();
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {}
            });

            // Redirect to the main screen
            Intent intent = new Intent(LoginUserActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            finish();
        } else {
            Toast.makeText(LoginUserActivity.this, R.string.system_message_login_error,
Toast.LENGTH_SHORT).show();
        }
    });
  }
});

mBtnRegister.setOnClickListener(v -> {
    Intent intent = new Intent(LoginUserActivity.this, RegisterUserActivity.class);
    startActivity(intent);
    //finish();
});

mForgotPasswordView.setOnClickListener(view -> startActivity(new Intent(LoginUserActivity.this,
ResetPasswordActivity.class)));
  }
}
```

Classe LookUpExerciseActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.ExerciseAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.User;

public class LookUpExerciseActivity extends AppCompatActivity {

    // Visual Elements
    Button mBtnEditExercise;
    TextView mTextDescription, mTextYtLink;

    FirebaseUser mFirebaseUser;
    DatabaseReference mReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_look_up_exercise);

    Intent intent = getIntent();
    final String name = intent.getStringExtra("name");

    // Toolbar settings
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle(name);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    toolbar.setNavigationOnClickListener(view -> finish());

    mBtnEditExercise = findViewById(R.id.btnEditExercise);
    mTextDescription = findViewById(R.id.textDescription);
    mTextYtLink = findViewById(R.id.textYtLink);

    mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    assert name != null;
    mReference = FirebaseDatabase.getInstance().getReference("Exercises").child(name);
    mReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            Exercise exercise = dataSnapshot.getValue(Exercise.class);

            assert exercise != null;
            mTextDescription.setText(exercise.getDescription());
            mTextYtLink.setText(exercise.getVideoLink());
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
```

```
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(LookUpExerciseActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(LookUpExerciseActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(LookUpExerciseActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(LookUpExerciseActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(LookUpExerciseActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
            startActivity(new Intent(LookUpExerciseActivity.this, ChatsActivity.class));
```

```
            finish();
            return true;
        // Log out
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(LookUpExerciseActivity.this, LoginUserActivity.class));
            finish();
            return true;
    }

    return false;
  }
}
```

Classe LookUpWorkoutActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Model.Workout;
```

```
public class LookUpWorkoutActivity extends AppCompatActivity {

    // Visual Elements
    Button mBtnEditWorkout;
    TextView mTextStatus, mTextDate, mTextWorkout;

    FirebaseUser mFirebaseUser;
    DatabaseReference mReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_look_up_workout);
        SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
        final String accountType = accountPrefs.getString("accountType", null);

        Intent intent = getIntent();
        final String name = intent.getStringExtra("name");
        final String proId = intent.getStringExtra("proId");
        final String clientId = intent.getStringExtra("clientId");
        final String workoutId = intent.getStringExtra("id");

        // Toolbar settings
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(name);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        mBtnEditWorkout = findViewById(R.id.btnEditWorkout);
        mTextStatus = findViewById(R.id.textStatus);
        mTextDate = findViewById(R.id.textDate);
        mTextWorkout = findViewById(R.id.textWorkout);

        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        // If you have workouts, add its id to a list
        if (accountType.equals("pro")) {
            mReference                                                                                    =
FirebaseDatabase.getInstance().getReference("Workouts").child(mFirebaseUser.getUid()).child(clientId).child(wor
koutId);
        } else {
```

```
            mReference = FirebaseDatabase.getInstance().getReference("Workouts").child(proId).child(mFirebaseUser.getUid()).child(workoutId);
        }
        assert workoutId != null;

        mReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Workout workout = dataSnapshot.getValue(Workout.class);

                assert workout != null;
                mTextDate.setText(workout.getDate());
                mTextWorkout.setText(workout.getWorkout());

                if (workout.getIsPending()) {
                    mTextStatus.setText("Pendente");
                } else {
                    mTextStatus.setText("Concluído");
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });

        mBtnEditWorkout.setOnClickListener(view -> {
            Intent intent1 = new Intent(LookUpWorkoutActivity.this, EditWorkoutActivity.class);
            intent1.putExtra("proId", proId);
            intent1.putExtra("clientId", clientId);
            intent1.putExtra("workoutId", workoutId);
            startActivity(intent1);
        });
    }

    // Create the options menu on the toolbar
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
        final String accountType = accountPrefs.getString("accountType", null);

        if (accountType.equals("pro")) {
            getMenuInflater().inflate(R.menu.menu_pro, menu);
```

```
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(LookUpWorkoutActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(LookUpWorkoutActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(LookUpWorkoutActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(LookUpWorkoutActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(LookUpWorkoutActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
```

```
            startActivity(new Intent(LookUpWorkoutActivity.this, ChatsActivity.class));
            finish();
            return true;
        // Log out
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(LookUpWorkoutActivity.this, LoginUserActivity.class));
            finish();
            return true;
    }

    return false;
  }
}
```

Classe MainActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Client.AccountClientSettingsActivity;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.User;
```

```
import com.vitorflores.connectfit.Professional.AccountProSettingsActivity;

import de.hdodenhof.circleimageview.CircleImageView;

public class MainActivity extends AppCompatActivity {

    Button mBtnProfileSettings, mBtnActiveContracts, mBtnPendingContracts, mBtnProfessionals, mBtnExercises,
mBtnChats;
    CircleImageView mImgProfileAvatarView;
    TextView mTextProfileName;
    User user;

    // Firebase Authentication
    private FirebaseAuth mAuth;
    private FirebaseUser mFirebaseUser;
    private DatabaseReference mUsersReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.app_name);

        final Boolean isProAccount = false;

        mBtnProfileSettings = findViewById(R.id.btnProfileSettings);
        mBtnActiveContracts = findViewById(R.id.btnActiveContracts);
        mBtnPendingContracts = findViewById(R.id.btnPendingContracts);
        mBtnProfessionals = findViewById(R.id.btnProfessionals);
        mBtnExercises = findViewById(R.id.btnExercises);
        mBtnChats = findViewById(R.id.btnChats);
        mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
        mTextProfileName = findViewById(R.id.textProfileName);

        mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        mUsersReference = FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());

        mUsersReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
```

```
            user = dataSnapshot.getValue(User.class);

            assert user != null;
            if (user.getImageURL() != null && user.getImageURL().equals("default")) {
                mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher);
            } else {
                Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
            }

            mTextProfileName.setText(user.getUserName());

            // Show the correct buttons based on the type of the account
            mBtnProfileSettings.setVisibility(View.VISIBLE);
            mBtnActiveContracts.setVisibility(View.VISIBLE);
            mBtnPendingContracts.setVisibility(View.VISIBLE);
            mBtnChats.setVisibility(View.VISIBLE);
            mBtnExercises.setVisibility(View.VISIBLE);

            if (!user.getAccountType().equals("pro")) {
                mBtnProfessionals.setVisibility(View.VISIBLE);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });

    mBtnProfileSettings.setOnClickListener(view -> {
        Intent intent;

        if (user.getAccountType().equals("pro")) {
            intent = new Intent(MainActivity.this, AccountProSettingsActivity.class);
        } else {
            intent = new Intent(MainActivity.this, AccountClientSettingsActivity.class);
        }

        //intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    });

    mBtnActiveContracts.setOnClickListener(view -> {
```

```
        Intent intent = new Intent(MainActivity.this, ContractsActivity.class);
        intent.putExtra("typeContracts", "active");
        startActivity(intent);
        //finish();
    });

    mBtnPendingContracts.setOnClickListener(view -> {
        Intent intent = new Intent(MainActivity.this, ContractsActivity.class);
        intent.putExtra("typeContracts", "pending");
        startActivity(intent);
    });

    mBtnProfessionals.setOnClickListener(view -> {
        Intent intent = new Intent(MainActivity.this, AvailableProfessionals.class);
        startActivity(intent);
        //finish();
    });

    mBtnExercises.setOnClickListener(view -> {
        Intent intent = new Intent(MainActivity.this, ExercisesActivity.class);
        startActivity(intent);
    });

    mBtnChats.setOnClickListener(view -> {
        Intent intent = new Intent(MainActivity.this, ChatsActivity.class);
        startActivity(intent);
        //finish();
    });
}

// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu, menu);
    return true;
}

// If the logout button is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        // Log out
        case R.id.logout:
```

```
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(MainActivity.this, LoginUserActivity.class));
            finish();
            return true;
        }

        return false;
    }
}
```

Classe MessageActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.MessageAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Fragments.APIService;
import com.vitorflores.connectfit.Model.Chat;
```

```
import com.vitorflores.connectfit.Model.User;
import com.vitorflores.connectfit.Notifications.Client;
import com.vitorflores.connectfit.Notifications.Data;
import com.vitorflores.connectfit.Notifications.Response;
import com.vitorflores.connectfit.Notifications.Sender;
import com.vitorflores.connectfit.Notifications.Token;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;
import retrofit2.Call;
import retrofit2.Callback;

public class MessageActivity extends AppCompatActivity {

    // Visual elements
    CircleImageView mImgProfileAvatarView;
    EditText mFieldSend;
    ImageButton mBtnSend;
    TextView mTextProfileName;
    User user;

    MessageAdapter messageAdapter;
    List<Chat> mChat;
    RecyclerView messagesLayout;

    // Firebase Authentication
    private FirebaseUser mFirebaseUser;
    private DatabaseReference mUsersReference;

    Intent intent;
    String receiverId;

    // Notifications
    APIService apiService;
    boolean notify = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
```

```
// Toolbar
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setTitle("");
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
toolbar.setNavigationOnClickListener(view -> finish());

// Establish a communication with the fcm server
apiService = Client.getClient("https://fcm.googleapis.com/").create(APIService.class);

// Visual Elements
mImgProfileAvatarView = findViewById(R.id.imgProfileAvatar);
mTextProfileName = findViewById(R.id.textProfileName);
mFieldSend = findViewById(R.id.fieldSend);
mBtnSend = findViewById(R.id.btnSend);

messagesLayout = findViewById(R.id.messagesLayout);
messagesLayout.setHasFixedSize(true);
LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getApplicationContext());
linearLayoutManager.setStackFromEnd(true);
messagesLayout.setLayoutManager(linearLayoutManager);

intent = getIntent();
receiverId = intent.getStringExtra("id");
mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

// Send a message
mBtnSend.setOnClickListener(view -> {
    notify = true;
    String message = mFieldSend.getText().toString();

    if (!message.equals("")) {
        sendMessage(mFirebaseUser.getUid(), receiverId, message);
    } else {
        Toast.makeText(MessageActivity.this, "Error", Toast.LENGTH_SHORT).show();
    }

    mFieldSend.setText("");
});

assert receiverId != null;
mUsersReference = FirebaseDatabase.getInstance().getReference("Users").child(receiverId);
```

```
        // Puts the profile picture and name of the receiver in the toolbar
        mUsersReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                user = dataSnapshot.getValue(User.class);

                assert user != null;
                if (user.getImageURL().equals("default")) {
                    mImgProfileAvatarView.setImageResource(R.mipmap.ic_launcher);
                } else {
                    Glide.with(getApplicationContext()).load(user.getImageURL()).into(mImgProfileAvatarView);
                }

                mTextProfileName.setText(user.getUserName());

                // Show the messages with the receiver on the screen
                readMessages(mFirebaseUser.getUid(), receiverId);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }

    // Send a message
    private void sendMessage(String sender, String receiver, String message) {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference();

        HashMap<String, java.lang.Object> hashMap = new HashMap<>();
        hashMap.put("sender", sender);
        hashMap.put("receiver", receiver);
        hashMap.put("message", message);

        reference.child("Chats").push().setValue(hashMap);

        // If you send a message to a user, keep their id saved on your chatlist so you can know who you had a chat
with
        final DatabaseReference chatSenderReference = FirebaseDatabase.getInstance().getReference("Chatlist")
                .child(mFirebaseUser.getUid())
                .child(receiverId);
```

```
chatSenderReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (!dataSnapshot.exists()) {
            chatSenderReference.child("id").setValue(receiverId);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});

// Do the same thing as above but with your id under the receiver's chatlist
final DatabaseReference chatReceiverReference = FirebaseDatabase.getInstance().getReference("Chatlist")
        .child(receiverId)
        .child(mFirebaseUser.getUid());

chatReceiverReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (!dataSnapshot.exists()) {
            chatReceiverReference.child("id").setValue(mFirebaseUser.getUid());
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});

// After you sent a message, send a notification to the receiver
final String msg = message;
mUsersReference = FirebaseDatabase.getInstance().getReference("Users").child(mFirebaseUser.getUid());
mUsersReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        User user = dataSnapshot.getValue(User.class);

        assert user != null;
```

```
                if (notify) {
                    sendNotification(receiver, user.getUserName(), msg);
                }

                notify = false;
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }

    // Send a notification to the receiver, after you sent him a message
    private void sendNotification(String receiver, final String username, String message) {
        DatabaseReference tokens = FirebaseDatabase.getInstance().getReference("Tokens");
        Query query = tokens.orderByKey().equalTo(receiver);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Token token = snapshot.getValue(Token.class);
                    Data data = new Data(mFirebaseUser.getUid(), R.mipmap.ic_launcher, username + ": " + message,
"Nova Mensagem", receiverId);

                    assert token != null;
                    Sender sender = new Sender(data, token.getToken());

                    apiService.sendNotification(sender)
                            .enqueue(new Callback<Response>() {
                                @Override
                                public void onResponse(Call<Response> call, retrofit2.Response<Response> response) {
                                    if (response.code() == 200) {
                                        assert response.body() != null;
                                        if (response.body().success != 1) {
                                            Toast.makeText(MessageActivity.this, "Error", Toast.LENGTH_SHORT).show();
                                        }
                                    }
                                }

                                @Override
                                public void onFailure(Call<Response> call, Throwable t) {
```

```
                    }
                });
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}

// Show the messages on the screen
private void readMessages(final String myId, final String ReceiverId) {
    mChat = new ArrayList<>();

    mUsersReference = FirebaseDatabase.getInstance().getReference("Chats");
    mUsersReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mChat.clear();

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                Chat chat = snapshot.getValue(Chat.class);

                assert chat != null;
                if ((chat.getReceiver().equals(myId) && chat.getSender().equals(ReceiverId)) ||
                    (chat.getReceiver().equals(ReceiverId) && chat.getSender().equals(myId))) {
                    mChat.add(chat);
                }

                // Take the messages list and show all the messages with the other person on the screen
                messageAdapter = new MessageAdapter(MessageActivity.this, mChat);
                messagesLayout.setAdapter(messageAdapter);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}
```

```
// Create the options menu on the toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    if (accountType.equals("pro")) {
        getMenuInflater().inflate(R.menu.menu_pro, menu);
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(MessageActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(MessageActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(MessageActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(MessageActivity.this, AvailableProfessionals.class));
            finish();
```

```
                return true;
            // Exercises Screen
            case R.id.exercises:
                startActivity(new Intent(MessageActivity.this, ExercisesActivity.class));
                finish();
                return true;
            // Chats Screen
            case R.id.chats:
                startActivity(new Intent(MessageActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(MessageActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Classe RegisterUserActivity:

```
package com.vitorflores.connectfit;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
```

```
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.vitorflores.connectfit.Client.AccountClientSettingsActivity;
import com.vitorflores.connectfit.Professional.AccountProSettingsActivity;

import org.apache.commons.lang3.StringUtils;

import java.util.HashMap;

public class RegisterUserActivity extends AppCompatActivity {

    // Visual Elements
    private EditText mUserNameView, mEmailView, mPasswordView, mConfirmPasswordView;
    private RadioButton mAccountTypeClient, mAccountTypePro;
    private Button mBtnRegister;

    // Firebase Authentication
    private FirebaseAuth mAuth;

    private DatabaseReference mReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register_user);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.action_register_account);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        mUserNameView = findViewById(R.id.fieldName);
        mEmailView = findViewById(R.id.fieldEmail);
        mPasswordView = findViewById(R.id.fieldPassword);
        mConfirmPasswordView = findViewById(R.id.fieldConfirmPassword);
        mAccountTypeClient = findViewById(R.id.rbAccountTypeClient);
        mAccountTypePro = findViewById(R.id.rbAccountTypePro);
        mBtnRegister = findViewById(R.id.btnRegister);
        mAuth = FirebaseAuth.getInstance();
```

```
        mBtnRegister.setOnClickListener(v -> {
            final String name = mUserNameView.getText().toString();
            final String email = mEmailView.getText().toString().toLowerCase();
            final String password = mPasswordView.getText().toString();
            final String confirmPassword = mConfirmPasswordView.getText().toString();

            if (!name.equals("") && !email.equals("") && !password.equals("") && !confirmPassword.equals("") &&
password.equals(confirmPassword)) {
                createFirebaseUser();
            } else {
                Toast.makeText(RegisterUserActivity.this,                R.string.system_message_register_error,
Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void createFirebaseUser() {
        final String name = mUserNameView.getText().toString();
        final String email = mEmailView.getText().toString().toLowerCase();
        final boolean isPro;
        final String password = mPasswordView.getText().toString();

        if (mAccountTypePro.isChecked()) {
            isPro = true;
        } else {
            isPro = false;
        }

        // Register an user
        mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                FirebaseUser firebaseUser = mAuth.getCurrentUser();
                assert firebaseUser != null;
                String userid = firebaseUser.getUid();

                mReference = FirebaseDatabase.getInstance().getReference("Users").child(userid);

                HashMap<String, String> hashMap = new HashMap<>();
                hashMap.put("id", userid);
                hashMap.put("userName", name);
                String searchTerm = StringUtils.stripAccents(name.toLowerCase());
                hashMap.put("searchableName", searchTerm);
```

```
hashMap.put("imageURL", "default");
hashMap.put("height", "");
hashMap.put("weight", "");
hashMap.put("about", "");
hashMap.put("cref", "");
hashMap.put("city", "");
hashMap.put("state", "");

if (isPro) {
    hashMap.put("accountType", "pro");
} else {
    hashMap.put("accountType", "client");
}


// Redirects to the 'edit user' screen
mReference.setValue(hashMap).addOnCompleteListener(task1 -> {
    if (task1.isSuccessful()) {
        mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task11 -> {
            if (task11.isSuccessful()) {
                Intent intent;
                SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
                SharedPreferences.Editor editor;

                if (isPro) {
                    intent = new Intent(RegisterUserActivity.this, AccountProSettingsActivity.class);
                    // Save the account type on a sharedPref
                    editor = accountPrefs.edit();
                    editor.putString("accountType", "pro");
                    editor.commit();
                } else {
                    intent = new Intent(RegisterUserActivity.this, AccountClientSettingsActivity.class);
                    // Save the account type on a sharedPref
                    editor = accountPrefs.edit();
                    editor.putString("accountType", "client");
                    editor.commit();
                }

                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                finish();
            } else {
```

```
                        Toast.makeText(RegisterUserActivity.this, R.string.system_message_login_error,
Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
    } else {
        Toast.makeText(RegisterUserActivity.this, R.string.system_message_register_error,
Toast.LENGTH_SHORT).show();
    }
  });
 }
}
```

Classe ResetPasswordActivity:

```
package com.vitorflores.connectfit;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.vitorflores.connectfit.Client.AccountClientSettingsActivity;

public class ResetPasswordActivity extends AppCompatActivity {

    // Visual elements
    private Button mBtnResetPassword;
    private EditText mFieldEmail;

    // Firebase Authentication
    FirebaseAuth mFirebaseAuth;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reset_password);

        // Toolbar
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.bar_title_reset_password);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(view -> finish());

        // Visual elements
        mBtnResetPassword = findViewById(R.id.btnResetPassword);
        mFieldEmail = findViewById(R.id.fieldEmail);

        // Firebase auth
        mFirebaseAuth = FirebaseAuth.getInstance();

        mBtnResetPassword.setOnClickListener(view -> {
            String email = mFieldEmail.getText().toString().toLowerCase();

            if (email.equals("")) {
                Toast.makeText(ResetPasswordActivity.this, R.string.system_message_empty_fields,
Toast.LENGTH_SHORT).show();
            } else {
                mFirebaseAuth.sendPasswordResetEmail(email).addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        Toast.makeText(ResetPasswordActivity.this, R.string.system_message_check_your_email,
Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(ResetPasswordActivity.this, LoginUserActivity.class));
                    } else {
                        String error = task.getException().getMessage();
                        Toast.makeText(ResetPasswordActivity.this, email, Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
    }
}
```

Classe WorkoutsActivity:

```
package com.vitorflores.connectfit;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vitorflores.connectfit.Adapter.ExerciseAdapter;
import com.vitorflores.connectfit.Adapter.WorkoutAdapter;
import com.vitorflores.connectfit.Client.AvailableProfessionals;
import com.vitorflores.connectfit.Model.Exercise;
import com.vitorflores.connectfit.Model.Workout;

import java.util.ArrayList;
import java.util.List;

public class WorkoutsActivity extends AppCompatActivity {

    // Visual elements
    RelativeLayout mWorkoutsContainer;
    RecyclerView mWorkoutsLayout;
    FloatingActionButton mBtnAddWorkout;

    FirebaseUser mFirebaseUser;
    DatabaseReference mWorkoutsReference;
```

```
private List<Workout> workoutsList;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_workouts);

    Intent prevIntent = getIntent();
    final String proId = prevIntent.getStringExtra("proId");
    final String clientId = prevIntent.getStringExtra("clientId");
    SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
    final String accountType = accountPrefs.getString("accountType", null);

    // Toolbar settings
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle(R.string.bar_title_workouts);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    toolbar.setNavigationOnClickListener(view -> finish());

    // Visual elements
    mWorkoutsContainer = findViewById(R.id.workoutsContainer);
    mWorkoutsLayout = findViewById(R.id.workoutsLayout);
    mWorkoutsLayout.setHasFixedSize(true);
    mWorkoutsLayout.setLayoutManager(new LinearLayoutManager(WorkoutsActivity.this));
    mBtnAddWorkout = findViewById(R.id.btnAddWorkout);

    // Firebase Authentication
    mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

    if (accountType.equals("pro")) {
        mBtnAddWorkout.setVisibility(View.VISIBLE);
    }

    mBtnAddWorkout.setOnClickListener(view -> {
        Intent intent = new Intent(WorkoutsActivity.this, EditWorkoutActivity.class);
        intent.putExtra("clientId", clientId);
        startActivity(intent);
    });

    workoutsList = new ArrayList<>();

    // If you have workouts, add its id to a list
```

```
        assert mFirebaseUser != null;
        assert proId != null;
        assert clientId != null;
        if (accountType.equals("pro")) {
            mWorkoutsReference = FirebaseDatabase.getInstance().getReference("Workouts").child(mFirebaseUser.getUid()).child(clientId);
        } else {
            mWorkoutsReference = FirebaseDatabase.getInstance().getReference("Workouts").child(proId).child(mFirebaseUser.getUid());
        }

        mWorkoutsReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                workoutsList.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Workout workout = snapshot.getValue(Workout.class);

                    assert workout != null;
                    workoutsList.add(workout);
                }

                // Take the list of ids and show its names on the screen
                if (!workoutsList.isEmpty()) {
                    WorkoutAdapter workoutAdapter = new WorkoutAdapter(WorkoutsActivity.this, workoutsList);
                    mWorkoutsLayout.setAdapter(workoutAdapter);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}
        });
    }

    // Create the options menu on the toolbar
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        SharedPreferences accountPrefs = getSharedPreferences("accountPrefs", 0);
        final String accountType = accountPrefs.getString("accountType", null);

        if (accountType.equals("pro")) {
            getMenuInflater().inflate(R.menu.menu_pro, menu);
```

```
    } else {
        getMenuInflater().inflate(R.menu.menu_client, menu);
    }

    return true;
}

// If an option on menu is clicked
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent;
    switch (item.getItemId()) {
        // Main Screen
        case R.id.mainScreen:
            startActivity(new Intent(WorkoutsActivity.this, MainActivity.class));
            finish();
            return true;
        // Active contracts
        case R.id.activeContracts:
            intent = new Intent(WorkoutsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "active");
            startActivity(intent);
            finish();
            return true;
        // Pending contracts
        case R.id.pendingContracts:
            intent = new Intent(WorkoutsActivity.this, ContractsActivity.class);
            intent.putExtra("typeContracts", "pending");
            startActivity(intent);
            finish();
            return true;
        // Available Professionals Screen
        case R.id.availableProfessionals:
            startActivity(new Intent(WorkoutsActivity.this, AvailableProfessionals.class));
            finish();
            return true;
        // Exercises Screen
        case R.id.exercises:
            startActivity(new Intent(WorkoutsActivity.this, ExercisesActivity.class));
            finish();
            return true;
        // Chats Screen
        case R.id.chats:
```

```
                startActivity(new Intent(WorkoutsActivity.this, ChatsActivity.class));
                finish();
                return true;
            // Log out
            case R.id.logout:
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(WorkoutsActivity.this, LoginUserActivity.class));
                finish();
                return true;
        }

        return false;
    }
}
```

Arquivo exercícios.json

```
{
  "Exercises" : {
    "Agachamento" : {
      "description" : "\"Descrição de como realizar o movimento agachamento\"",
      "name" : "Agachamento",
      "videoLink" : "https://www.youtube.com/watch?v=4TG8JdU6NPU"
    },
    "Barra Fixa" : {
      "description" : "\"Descrição de como realizar o movimento barra fixa\"",
      "name" : "Barra Fixa",
      "videoLink" : "https://www.youtube.com/watch?v=JX_YM7Bp26U"
    },
    "Flexão" : {
      "description" : "\"Descrição de como realizar o movimento flexão\"",
      "name" : "Flexão",
      "videoLink" : "https://www.youtube.com/watch?v=F9FC_KBsLpY"
    }
  }
}
```