UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

HERMES TESSARO DE MELLO AFFONSO

Uma Aplicação Mobile para Sistematização de Fichas de Alunos de Academias

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Profa. Dra. Renata Galante

Porto Alegre
2020

“Vale a pena sonhar com isso: todos nós humanos em harmonia conectada de pulsões criativas, alforriados do trabalho mecânico pelas máquinas, não libertos do corpo, mas libertos no corpo, não mais predadores universais da criação, mas hiperlúcidos guardas-parque de Gaia. Quem não entender, que pingue mais uma gota.”

— ALDOUS HUXLEY

AGRADECIMENTOS

Agradeço à minha família e especialmente a meus pais Ivete e Cleumar, por possibilitarem tudo que conquistei até hoje e por me darem o apoio necessário; à Danielly Cruz, por ter me dado tanto carinho e amor e por ser uma pessoa maravilhosa. Também agradeço à Thais Dias de Quadros, a melhor amiga que alguém poderia pedir, que sempre me apoiou e ajudou, além de ter passado as tardes conversando comigo enquanto eu escrevia este trabalho. A meus amigos Gabriel Pacicco, Rafael Allegretti e Matheus Michel, por me proporcionarem tantas risadas, festas, diversões e apoio quando precisei. À minha orientadora, Profa. Dra. Renata Galante, que sempre foi honesta em suas correções e solícita em responder quaisquer dúvidas que eu tivesse.

Agradeço também ao Grêmio Football Porto Alegrense, que naufragou em todas as competições que disputou esse ano e me permitiu ter mais horas sóbrias e sem futebol para dedicar a este trabalho.

RESUMO

Muitas academias de musculação ainda utilizam fichas em papel para manter os dados de seus alunos e para criar rotinas de treino para estes. Além de ser ecologicamente prejudicial, o papel pode ser perdido ou arruinado com facilidade, e também dificulta a alteração das informações ali contidas. Tendo isso em vista, esse trabalho propõe uma aplicação mobile para a sistematização e gerenciamento dos alunos e de suas rotinas de treino de forma fácil e rápida, mantendo as informações salvas em mais de um banco de dados - no caso, um banco de dados local para cada instância do aplicativo e um no servidor. A aplicação torna o compartilhamento de informações entre professores mais fácil, permitindo que estes exponham suas ideias variáveis para melhor aproveitamento dos aparelhos da academia e de eficiência para diferentes rotinas de treino. Experimentos de usabilidade com usuários indicam que a aplicação é fácil de usar, com funções bem integradas e que há demanda para ele.

Palavras-chave: Academia. Musculação. Aplicativo. Android. Sistematização. Gerenciamento.

ABSTRACT

Many bodybuilding gyms still use paper files to keep the data of their students and to create workout routines for them. Besides being harmful to the environment, paper may be easily lost or ruined, while also making it difficult to alter the information it contains. With that in sight, this essay proposes a mobile application for the systematization and management of students and their workout routines in a faster and easier way, keeping the data saved in more than one database - a local database for every instance of the application and one in the server. It also aims to make the sharing of data between instructors easier, allowing them to expose their different ideas for better use of the gym equipments and for efficiency to different workout routines. Trials made with users indicate that the application is easy to use, has well integrated functions and that there is a demand for it.

Keywords: Gym. Workout. Bodybuilding. Application. Android. Management. Systematization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API Application Program Interface

MVC Model-View-Controller

UI User Interface

SUS System Usability Scale

IDE Integral Development Environment

SUMÁRIO

1 INTRODUÇÃO

Existem muitos aplicativos no mercado com o intuito de salvar rotinas de treino, exercícios e dados relativos à musculação. No entanto, a maioria é destinada ao usuário final, ou seja, ao aluno de musculação. Há inclusive aplicativos com a intenção de substituir o professor de musculação, usando a presunção de que este estaria obsoleto frente à tecnologia disponível atualmente. No entanto, especialistas alertam contra os riscos de treinar sem orientação de um profissional competente[1]. O principal risco é a ocorrência de lesões por não haver um atendimento personalizado[2], que entenda as necessidades e limitações do aluno. Somando-se a estes fatos, os profissionais da área da computação têm de considerar sua responsabilidade social e ética ao desenvolver tecnologias que podem tornar profissões obsoletas. Antes do desenvolvedor perguntar se consegue fazer algo, ele deve se perguntar se deve fazê-lo.

Com isto em mente, a aplicação aqui descrita tem como proposta auxiliar o profissional de educação física a proporcionar um atendimento mais personalizado e rápido para seus alunos, sem a pretensão de substituir este professor. Outro ponto que a aplicação almeja cumprir é na economia de papel, por substituir as fichas utilizadas normalmente. O objetivo deste aplicativo, enfim, é servir como uma ferramenta mobile para o profissional de educação física - a ser usado em smartphones ou tablets - , onde ele pode cadastrar e editar alunos, rotinas de treino e exercícios de maneira mais rápida e fácil. Também foi idealizada a construção de um website com as mesmas funções do aplicativo, e que também serve de servidor para receber, organizar e enviar os dados compartilhados para todas as instâncias do aplicativo. Um requisito feito foi que o aplicativo mantivesse os dados salvos num banco de dados local, para se manter funcional caso não houvesse uma conexão com internet disponível. Uma *Application Programming Interface* (API) foi desenvolvida utilizando PHP e o *framework* Yii para realizar esta troca de informações entre servidor e aplicações. Neste servidor, também há uma versão web da aplicação, projetada pelo autor deste trabalho e por Matheus Schilling Michel - que a implementou. As funcionalidades presentes na versão mobile são as mesmas que estão disponíveis na versão web.

Este trabalho está dividido em 7 capítulos. O capítulo 2 apresenta o arcabouço teórico utilizado para realizar a aplicação, assim como seus detalhes. O capítulo 3 com-

[1] https://gauchazh.clicrbs.com.br/saude/conteudo-publicitario/2018/02/conheca-os-riscos-de-fazer-exercicios-fisicos-por-conta-propria-cjdex0qwf007b01n3u6a0c7bh.html (acesso em dezembro de 2019)

[2] https://oniquenutrition.com/blog/aplicativos-fitness/ (acesso em dezembro de 2019)

para esta aplicação com outras semelhantes, denotando vantagens e desvantagens de uma em relação às outras. O capítulo 4 versa sobre a metodologia usada para desenvolver a aplicação, discutindo a arquitetura do sistema, requisitos e o projeto de banco de dados. O capítulo 5 fornece um panorama geral sobre a aplicação, incluindo seu funcionamento e interfaces da aplicação, e o capítulo 6 demonstra a metodologia utilizada para realizar experimentos de usabilidade da aplicação, além de apresentar os resultados destes. O capítulo 7 apresenta uma conclusão sobre o trabalho realizado, e o Apêndice A contém o questionário enviado aos usuários para que estes avaliassem a aplicação.

2 TECNOLOGIAS UTILIZADAS

Este capítulo tem como propósito apresentar as tecnologias e conceitos utilizados no desenvolvimento do projeto.

2.1 Model-View-Controller (MVC)

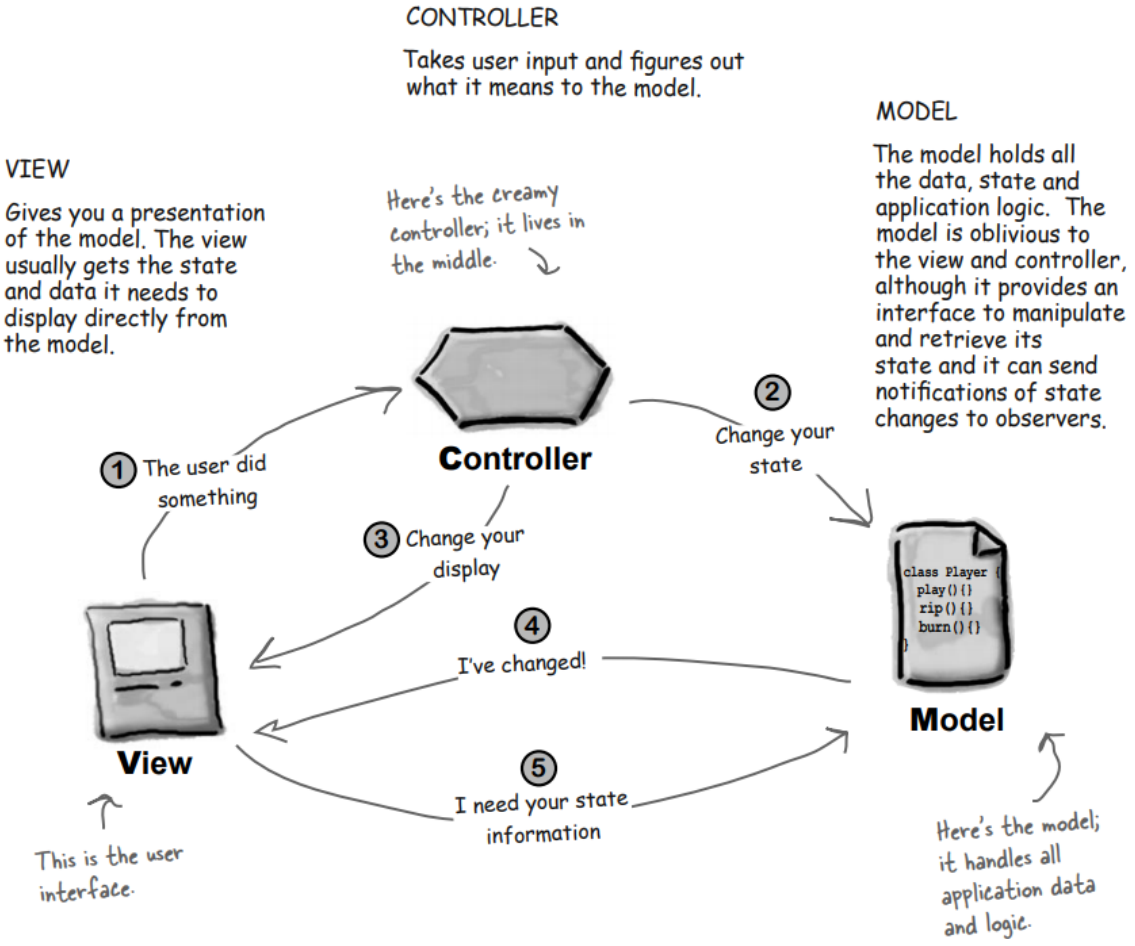
De acordo com Krasner e Pope (1988), o Model-View-Controller (MVC) aplica uma modularização que tem várias vantagens além de tornar a vida do desenvolvedor mais simples. Esta modularização é uma separação em três vias dos componentes da aplicação, sendo estes componentes (1) as partes que representam o modelo do domínio da aplicação, (2) a maneira que o modelo é apresentado para o usuário do sistema e (3) a maneira que o usuário interage com ele.

No MVC, a classe responsável pelas operações com o domínio da aplicação é chamada de *Model*; a responsável pela apresentação do estado da aplicação é a View, e a responsável pela interação do usuário com o *Model* e a *View* é chamada de *Controller*. De acordo com Subramanium e Hunt (2006), este padrão é eficiente, pois permite uma maior coesão - as classes no *Model* contêm um tipo de funcionalidade, as no *Controller* contêm outra, e as na *View* se interessam apenas pela interação com o usuário. Tal coesão também afeta a reusabilidade dos componentes.

A Figura 2.1 demonstra a estrutura do MVC. O usuário utiliza os comandos dispostos na *View* - botões, campos de texto, etc - para enviar instruções ao *Controller*, a fim de receber as informações que gostaria. O *Controller* interpreta os comandos recebidos e manipula o conjunto de dados da forma necessária e, a seguir, seleciona a *View* mais adequada para apresentar os dados requisitados. Dependendo da estrutura criada, tanto o *Model* quanto o *Controller* podem atualizar a *View* com os dados que, então, é apresentada ao usuário.

O desenvolvimento para Android se mostra ideal para a utilização do MVC, pois por padrão já se separa os layouts (a *View*) do resto do código, o que facilita para o programador no momento de estruturar o seu projeto. A utilização de Java e/ou Kotlin como linguagem de programação também facilita, pois ambas são orientadas a objetos, significando que tanto os *Models* quanto os *Controllers* podem ser criados como classes, e as interações com a *View* podem ser implementadas como interfaces.

Figura 2.1: Diagrama do MVC



Fonte: FREEMAN, Eric et al. Head first design patterns. "O'Reilly Media, Inc.", 2008.

2.2 Kotlin

A linguagem de programação escolhida para este projeto foi Kotlin[1], em detrimento da escolha tida como clássica para projetos em Android, o Java. Kotlin é uma linguagem de programação fortemente tipada criada pela JetBrains com o intuito de substituir o Java, que estava causando problemas no desenvolvimento do Ambiente Integral de Desenvolvimento (*Integral Development Environment* - IDE) própria da JetBrains. Em 2017 a Google anunciou que o Kotlin se tornara a linguagem oficial de programação Android.

Neste contexto, Kotlin apresenta várias vantagens. De acordo com Panchal e Patel (2017), estas são a interoperabilidade com Java, a limpeza e concisidade do código, e o

[1]https://kotlinlang.org/ (acesso em dezembro de 2019)

fato de ter uma prática de *Null Safety*[2] imbuída diretamente na linguagem.

2.3 SQLite

O SQLite[3] é uma biblioteca, escrita na linguagem C, que implementa um banco de dados SQL. O código da biblioteca é *open-source*, e - de acordo com o ranking do db-engines.com -, o SQLite é o 11º modelo de banco de dados mais utilizado no mundo. Ao contrário da maioria dos modelos de banco de dados, o SQLite é uma biblioteca compacta (cerca de 600 KB), e não possui um processo separado de servidor. Todas as operações são feitas diretamente no disco e o formato do arquivo da base de dados é aceito em vários tipos de plataforma. Por estes motivos, o SQLite foi diretamente integrado no Android e oferece mais recursos que a outra base de dados local que o Android possui, o *SharedPreferences*.

Na implementação do projeto, foi criada uma classe auxiliar para gerenciar todas as operações de criação, atualização, remoção e leitura (CRUD) do banco de dados - Figura 2.2. Esta classe implementa a classe nativa do Android SQLiteOpenHelper, que tem funções específicas para manipular o SQLite. A partir disto, é fácil fazer operações no banco usando a classe auxiliar, como fica demonstrado na Figura 2.3.

Figura 2.2: Declaração da classe auxiliar DatabaseHelper

```
class DatabaseHelper(context:Context?): SQLiteOpenHelper(context, DB_NAME,  factory: null, DB_VERSION) {
```

Fonte: Elaborada pelo autor.

Figura 2.3: Operação de recuperação do ID de um Aluno por seu nome.

```
245     fun getIdAlunoByName(name: String): Int {
246         val db : SQLiteDatabase!  = this.readableDatabase
247         val selectQuery = "SELECT * FROM $TABLE_ALUNO WHERE $KEY_NAME = '$name';"
248         val c: Cursor = db.rawQuery(selectQuery,  selectionArgs: null)
249         c.moveToLast()
250         return c.getInt(c.getColumnIndex(ID_ALUNO))
251     }
```

Fonte: Elaborada pelo autor.

[2]Null Safety é um sistema de tipagem que almeja eliminar exceções e erros onde o programa tenta executar uma operação em um valor nulo. No Kotlin, os objetos não podem ser nuláveis, a não ser que o programador explicite isto. Fonte: https://kotlinlang.org/docs/reference/null-safety.html (acesso em dezembro de 2019)

[3]https://www.sqlite.org/about.html (acesso em dezembro de 2019)

2.4 Git

De acordo com Laster (LASTER, Brent. Professional Git. John Wiley '|&' Sons, 2016), o Git é uma ferramenta de versionamento de código que simplifica em muito o desenvolvimento. Git permite a usuários criarem, usarem e mudarem de ramificações (i.e. diferentes desenvolvimentos que o código pode seguir dentro de um repositório, desde sua concepção) de forma rápida e fácil. É implementado com uma arquitetura leve e poderosa, dando espaço para que ocorram experimentações e refinamento de mudanças locais em um ambiente isolado antes do compartilhamento destas mudanças com outrém. Em resumo, possibilita a usuários iniciantes que foquem mais no conteúdo do que no gerenciamento do código e provê usuários mais avançados com a habilidade de salvar, editar e compartilhar mudanças em qualquer nível de detalhe. Neste projeto também foi utilizada a ferramenta GitKraken[4], uma *Graphical User Interface* (GUI) para Git, tornando a visualização e alteração das ramificações ainda mais simples.

2.5 Retrofit

Criado pela Square, Retrofit[5] é uma API para Android que emula um cliente HTTP, utilizando o padrão *Representational State Transfer* (REST). A API possibilita a troca de informações no formato JSON entre uma aplicação mobile e um servidor que contenha um webservice. Como o Retrofit implementa a biblioteca OkHttp, ele também aumenta o nível de abstração do processo – evitando que o desenvolvedor precise fazer as requisições HTTP.

O Retrofit utiliza, por padrão, a biblioteca GSON para serialização dos dados. De acordo com o site do desenvolvedor, o GSON pode ser utilizado para converter objetos Java em sua representação JSON e vice versa. Uma grande vantagem do GSON sobre as outras bibliotecas que convertem objetos Java em representação JSON é que esta não exige que sejam feitas anotações dentro das classes, tornando o código mais modularizado.

[4]https://www.gitkraken.com/ (acesso em dezembro de 2019)
[5]https://square.github.io/retrofit/ (acesso em dezembro de 2019)

2.6 Expandable Text-View

Em uma das telas, foi utilizado um *Expandable Text-View* - algo como 'Caixa de texto expansível'. Como esta ferramenda de *User Interface* (UI) não está disponível de forma nativa no Android, foi utilizada uma biblioteca *open-source* disponível no Github[6], sob autoria de Hakob Astvatsatryan.

2.7 Date Input Mask

Na tela de criação de aluno, foi utilizada uma máscara de entrada de data, forçando com que o usuário escreva uma data de nascimento no formato dd/MM/yyyy. O código para implementar a máscara foi tirado de uma postagem no StackOverflow[7], de autoria do usuário Ícaro Mota.

[6]https://github.com/hakobast/DropdownTextView (acesso em dezembro de 2019)
[7]https://stackoverflow.com/a/46653229/11805717 (acesso em dezembro de 2019)

3 TRABALHOS RELACIONADOS

Existem vários aplicativos disponíveis no mercado para auxiliar treinos de musculação. Ao buscar por 'musculação' na loja de apps da Google, encontra-se cerca de 250 aplicativos. Retirando aqueles que são apenas jogos ou direcionados à alimentação, ainda resta um número expressivo de apps que focam apenas em rotinas de treino. Neste capítulo, é dado um panorama de soluções similares à apresentada neste trabalho, sendo comparadas ao final. As soluções aqui apresentadas foram selecionadas por possuirem funcionalidades similares a este trabalho, além de possuírem as mais altas avaliações na loja de aplicativos.

3.1 Resumo de Aplicativos Semelhantes

3.1.1 Gym WP

O Gym WP[1] é um app para Android que visa o progresso do usuário na musculação. Possui opções para que o usuário salve informações pessoais suas - como medidas corporais, peso, altura, etc - e para que este visualize e registre rotinas de treino, tanto criadas por si mesmo quanto pré prontas pelo aplicativo. A aplicação também possui GIFs demonstrando como executar corretamente os exercícios, e gráficos e estatísticas relatando o progresso do usuário.

3.1.2 GymRun

O aplicativo GymRun[2], assim como o Gym WP, é para Android e foca em oferecer ao usuário uma maneira de gerenciar suas rotinas de treino e exercícios disponíveis. Também oferece análise gráfica e estatística das sessões de treinos, além da possibilidade de compartilhar os dados com outros apps, como Google Fit, S Health, Instagram, Facebook e Twitter.

[1] https://play.google.com/store/apps/details?id=com.lealApps.pedro.gymWorkoutPlan&hl=pt_BR (acesso em dezembro de 2019)

[2] https://play.google.com/store/apps/details?id=com.imperon.android.gymapp&hl=pt_BR (acesso em dezembro de 2019)

3.1.3 App Trainer

O App Trainer[3] é um serviço fornecido para academias que foca em oferecer suporte aos alunos, agregar receita adicional para a academia, auxiliar os instrutores, prospectar novas matrículas e fornecer um canal de comunicação entre a academia e o aluno. A empresa fornece um aplicativo a ser utilizado pelos instrutores e alunos, além de um site de gerenciamento, onde o usuário pode alimentar e gerenciar o aplicativo.

3.2 Análise Comparativa

Nesta subseção, é feita uma análise de diferentes itens de relevância dentro da proposta dos aplicativos, comparando estes com o aplicativo implementado nesse trabalho - que no decorrer desta seção, será chamado de 'Presente Aplicação'.

A Tabela 3.1 mostra as plataformas em que cada aplicação está disponível. Nota-se que nenhuma aplicação está disponível para Desktop, que o App Trainer é a única disponível para iOS e que as únicas com versão web são a Presente Aplicação e o App Trainer. Ambas também possuem integração do aplicativo com o website.

Tabela 3.1: Plataformas nas quais a aplicação está disponível

	GymRun	Gym WP	App Trainer	Presente Aplicação
Web	Não	Não	Sim	Sim
Android	Sim	Sim	Sim	Sim
iOS	Não	Não	Sim	Não
Desktop	Não	Não	Não	Não

Fonte: O Autor

A Tabela 3.2 mostra as funcionalidades disponíveis em cada aplicação. Funcionalidades básicas para um aplicativo deste tipo, como gerenciamento de rotinas de treino, estão disponíveis em todas as aplicações, como é de se esperar. Nota-se que, com exceção da Presente Aplicação, nenhum outro aplicativo - dos que foram escolhidos para esta análise - possui opção de gerenciamento de mais de um aluno, demonstrando que estes aplicativos têm seu desenvolvimento pensado tendo em vista que o usuário final seja o aluno da academia, e não o instrutor, ao contrário do objetivo da Presente Aplicação. Outras funções requisitadas, tais como o funcionamento offline e a sincronização com servidor, estão implementadas na Presente Aplicação, e além dela, apenas o App Trainer

[3] http://apptrainer.com.br/aplicativo-para-academia.aspx (acesso em dezembro de 2019)

possui essas duas funções. As funções que a Presente Aplicação não apresenta são Gráfico Evolutivo, Histórico de Frequência e Exportação de Dados. As duas primeiras são previstas para próximas versões da aplicação.

Tabela 3.2: Funcionalidades

	GymRun	Gym WP	App Trainer	Presente Aplicação
Base de dados de Exercícios	Sim	Sim	Sim	Sim
Gerenciamento de Rotinas de Treino	Sim	Sim	Sim	Sim
Funcionamento Offline	Sim	Não	Sim	Sim
Sincronização com Servidor	Não	Não	Sim	Sim
Gráfico Evolutivo	Sim	Não	Não	Não
Histórico de Frequência	Sim	Sim	Sim	Não
Exportar Dados	Sim	Não	Não	Não
Gerenciamento de mais de um aluno	Não	Não	Não	Sim

Fonte: O Autor

4 METODOLOGIA DE DESENVOLVIMENTO

O objetivo deste capítulo é explicar como se deu o desenvolvimento do trabalho, apresentando uma visão geral das etapas do processo como um todo, e focando em aspectos específicos e relevantes do trabalho, como requisitos, arquitetura e projeto de banco de dados.

4.1 Visão Geral

A primeira etapa de desenvolvimento constistiu no levantamento de requisitos e no estudo da melhor maneira de implementar a aplicação. Um levantamento de informações e requisitos foi realizado com um proprietário de academia e com professores de musculação, para melhor entender o contexto no qual a aplicação seria utilizada. Depois, uma esquematização de telas foi feita no papel afim de visualizar melhor as transições da aplicação e as necessidades de design.

A segunda etapa consistiu em desenvolver a estrutura básica da aplicação como um todo, construindo as principais atividades, modelos e controladores da aplicação. Nesta etapa também muito se estudou sobre as possibilidades de implementação de interfaces e comportamentos no Android, como Fragmentos, Navigation Drawer e Navigation Bar[1].

A terceira etapa focou na modelagem e implementação do banco de dados local, utilizando o SQLite. Tendo sido definidas as relações entre entidades do banco, bastou implementar os métodos para que estas entidades fossem salvas e recuperadas do banco.

A quarta etapa teve como foco o desenvolvimento da API e do Web Service para recuperar e sincronizar as informações salvas no banco de dados do servidor. Para a API foi utilizado o Retrofit, e a implementação do Web Service se deu utilizando o framework PHP Yii, que já havia sido utilizado para desenvolver a versão web da aplicação.

[1] https://material.io/design/

4.2 Requisitos

4.2.1 Requisitos Funcionais

Logo no início do projeto foi deixado claro quais seriam os requistos buscados. Considerando que apenas os professores da academia teriam acesso aos equipamentos que contém o sistema, tais funcionalidades foram requisitadas:

- Criação, visualização, edição e remoção de alunos.
- Criação, visualização, edição e remoção de treinos para cada aluno.
- Criação, visualização, edição e remoção de exercícios de musculação, e de seus respectivos grupos musculares.

Conforme a aplicação foi sendo desenvolvida, novas necessidades e requisitos foram sendo apresentados. Os alunos estariam associados ao professor, mas cada professor poderia ver todos os alunos - e seus respectivos treinos - matriculados. Para facilitar a experiência do usuário e evitar ao máximo erros gramaticais, foi decidido que não se poderia adicionar exercícios de mesmo nome no banco de dados, e quando um treino fosse criado, uma lista dos exercícios disponíveis seria mostrada.

4.2.2 Requisitos Não Funcionais

Houve dois pedidos grandes de requisitos não funcionais; um era que fosse desenvolvida uma versão web da aplicação, com as mesmas funcionalidade supracitadas, e também possuindo cadastro, visualização, remoção e edição de professores. O site funcionaria como um servidor, mantendo sincronizados e salvos os dados coletados por todas as instâncias da aplicação. Outra requisição feita era que o dispositivo em que o aplicativo estivesse sendo executado não precisasse estar sempre online, e que o fato de estar offline não impedisse a execução correta da aplicação.

Para que ambas as requisições fossem atendidas, se mostrou necessário implementar um banco de dados local em cada instância da aplicação - utilizando o SQLite - , e que uma API fosse desenvolvida para que os dados das aplicações pudessem ser enviados de e para o banco de dados do servidor.

4.3 Arquitetura do Sistema

Para o desenvolvimento da aplicação foi utilizado o Android Studio[2], IDE oficial de desenvolvimento visando o sistema Android. A IDE foi desenvolvida pela Google em parceria com a JetBrains, e possui as ferramentas necessárias para a criação de qualquer aplicativo Android, como um editor específico para construção de telas, emulador de dispositivo com Android para testar a aplicação, compilador, etc. Kotlin é a linguagem recomendada pelo Google para programação Android, mas a IDE também aceita Java e C++. O Android Studio também separa por padrão os arquivos XML (extensão dos layouts das telas) do resto do código, tornando mais fácil e lógica a utilização do padrão de arquitetura *Model-View-Controller*, considerando-se que o layout seja visto como a *View*.

Por ser mais recente e mais concisa do que Java, além de recomendada pelo fornecedor do Android Studio, Kotlin foi a linguagem escolhida para o desenvolvimento da aplicação.

Cada entidade do banco de dados foi mapeada para uma classe do padrão Model, e a biblioteca usada para implementar este banco foi a SQLite. Para manipular as informações do banco de dados, linguagem SQL simples foi utilizada.

A biblioteca utilizada para fazer as requisições web para o servidor foi a Retrofit. Ela é de fácil utilização e já contém um *parser* JSON integrado, o GSON.

Também foi utilizada uma biblioteca que torna possível a implementação de caixas de texto expansíveis, que funcionam da seguinte maneira: ao clicar em uma barra colorida, esta barra se expande e apresenta o texto que ela contém. Outro exemplo será dado com figuras - no capítulo 5.

4.4 Projeto do Banco de Dados

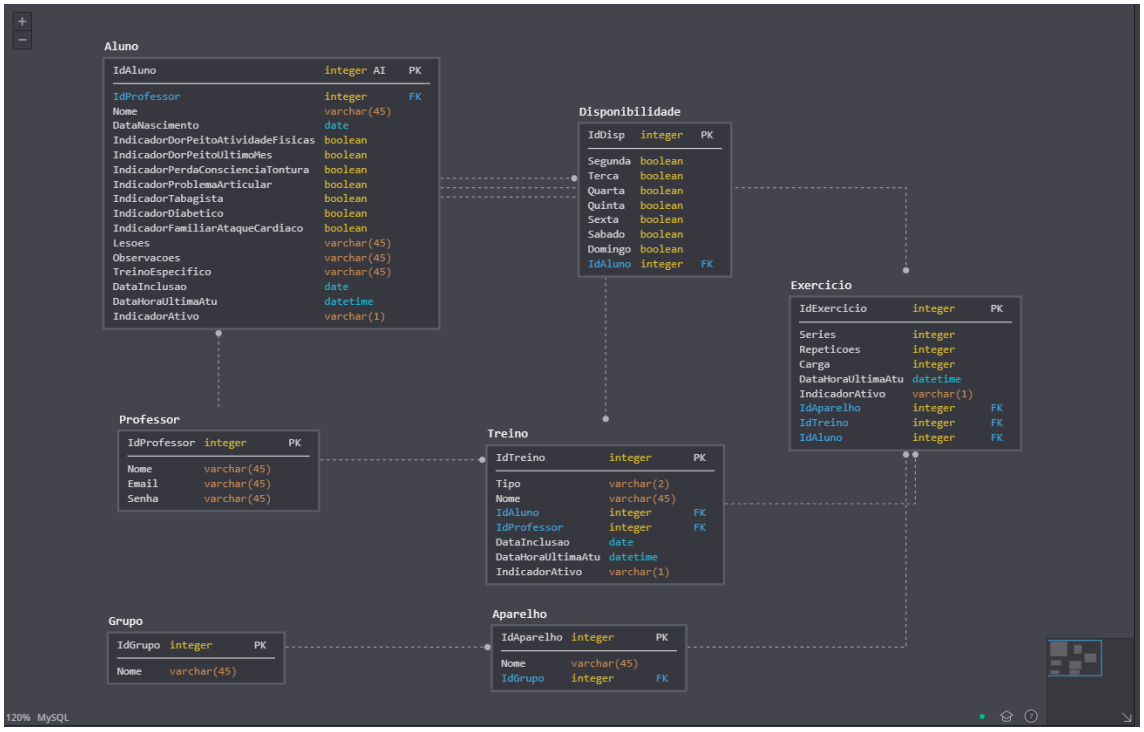
O banco de dados foi projetado e modelado tendo em mente a melhor maneira de se cumprir as funcionalidades requisitadas. A Figura 4.1 foi gerada utilizando a ferramenta SQLDBM[3], e mostra as relações entre as tabelas, bem como as colunas de cada uma. Cada modelo possui uma tabela, sendo elas as seguintes:

- Aluno - armazena as informações sobre cada aluno criado.

[2]https://developer.android.com/studio (acesso em dezembro de 2019)
[3]https://sqldbm.com/ (acesso em dezembro de 2019)

- Disponibilidade - tabela que salva a disponibilidade semanal de cada aluno.
- Professor - responsável por armazenar as informações dos professores, e só pode ter um registro criado ao receber este pela API.
- Grupo - armazena informações sobre grupos musculares. Não pode possuir entradas de mesmo nome. Esta tabela é utilizada para criar e selecionar os exercícios no momento de criar uma rotina de treino.
- Aparelho - armazena nome e grupo muscular (para o qual ele visa o fortalecimento) dos aparelhos disponíveis na academia.
- Exercicio - armazena os exercícios propostos pelo professor, contendo informações como número de séries e repetições, id da rotina de treino e id do aluno para o qual este está sendo criado.
- Treino - armazena informações relativas às rotinas de treino criadas para os alunos.

Figura 4.1: Visualização do Banco de Dados



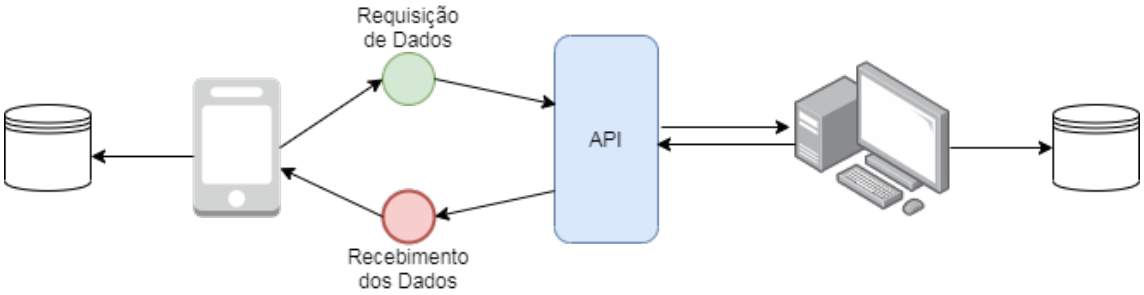
Fonte: Gerado pelo autor utilizando a ferramenta SQLDBM

Existem algumas colunas comuns a mais de uma tabela. IndicadorAtivo serve para verificar se aquele registro deve ser considerado ativo ou não. Para que houvesse possibilidade de um registro ser suprimido para recuperá-lo posteriormente, esta coluna foi criada. DataHoraUltimaAtu também foi criado para verificar a hora da última atualização, e facilitar a sincronização com a API.

4.5 Implementação dos Serviços Relativos à Sincronização de Dados

A Figura 4.2 mostra as comunicações realizadas entre aplicativo, API, servidor e bancos de dados para total sincronização dos dados utilizados.

Figura 4.2: Diagrama mostrando a possível movimentação de dados do sistema

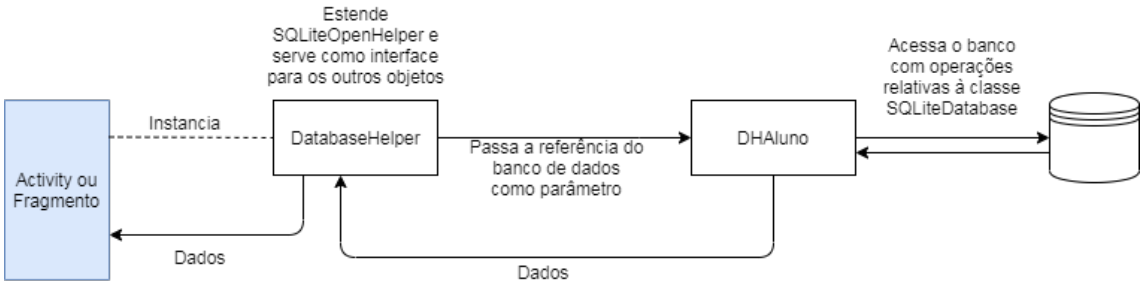


Fonte: Gerado pelo autor utilizando a ferramenta draw.io

4.5.1 Aplicativo e Banco de Dados Local

A Figura 4.3 mostra como acontece o acesso ao banco de dados local do aplicativo a partir de alguma Atividade ou Fragmento. Ela instancia um objeto de uma classe chamada DatabaseHelper, que é responsável por organizar as requisições ao banco de dados. Este objeto passa a referência do banco como parâmetro para uma classe contendo os métodos relativos ao modelo procurado - no caso da Figura 4.3, o modelo é aluno, logo a classe é DHAluno. Esta, finalmente, acessa o banco de dados utilizando operações da classe da qual DHAluno estende, que é a SQLiteOpenHelper.

Figura 4.3: Diagrama mostrando a comunicação entre Atividade ou Fragmento e o banco local

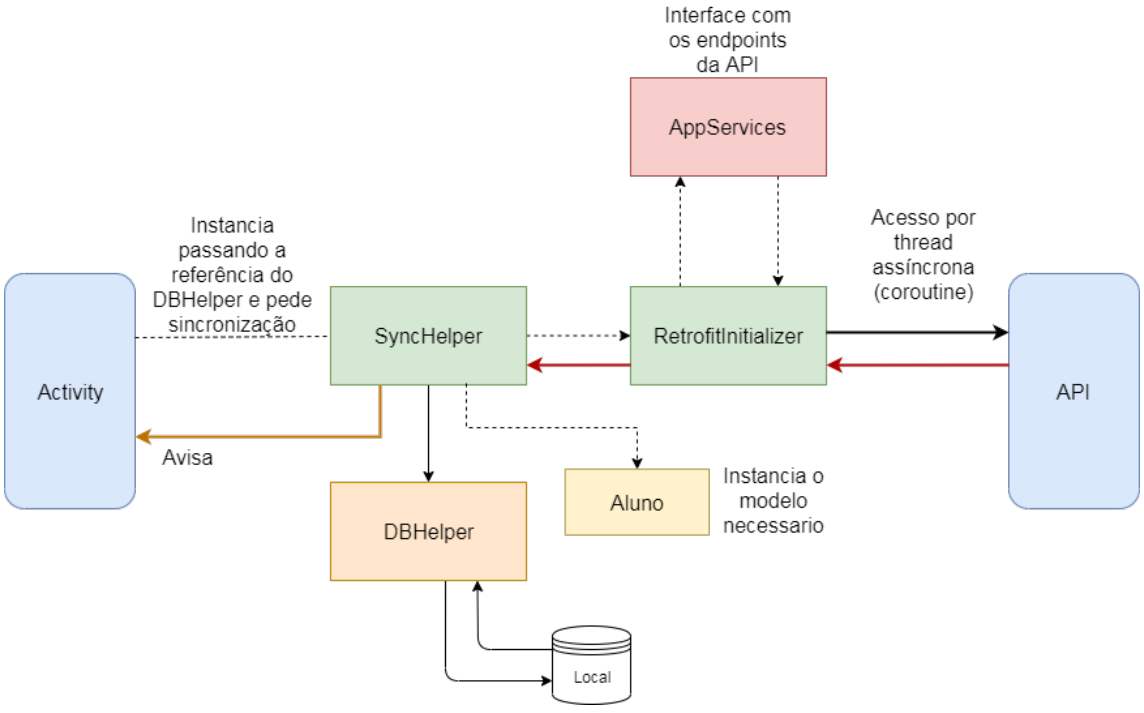


Fonte: Gerado pelo autor utilizando a ferramenta draw.io

4.5.2 Aplicativo e API

A Figura 4.4 mostra como o aplicativo age quando é necessário um acesso aos dados do servidor, para sincronização dos bancos de dados. A atividade instancia um objeto da classe SyncHelper passando como parâmetro a referência do banco de dados, e manda para esta classe a requisição de sincronização. A SyncHelper fica responsável por instanciar tanto o modelo necessário para serialização dos dados quanto as classes responsáveis por fazer a requisição web - a RetrofitInitializer - e o acesso ao banco de dados local - a DBHelper. A RetrofitInitializer recebe a requisição e a avalia, procurando então, na interface AppServices, o endereço web correto para enviar a requisição. Esta classe então acessa a API por meio de uma corotina, impedindo que o processo trave a interface de usuário.

Figura 4.4: Diagrama mostrando como o aplicativo lida com requisições para a API



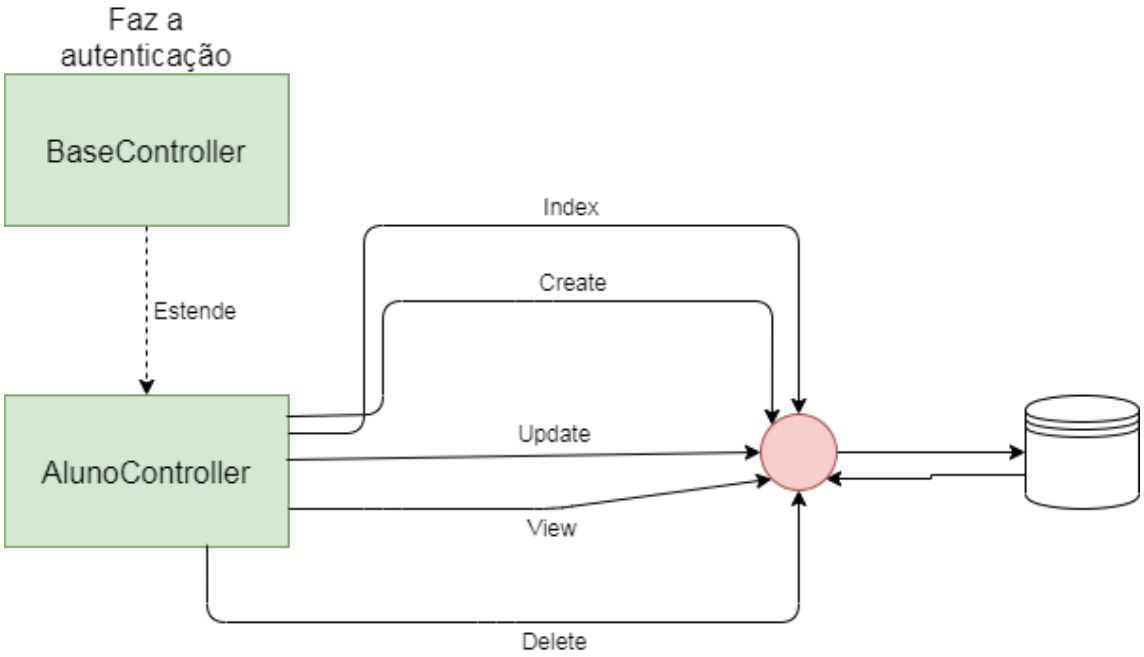
Fonte: Gerado pelo autor utilizando a ferramenta draw.io

4.5.3 API e Servidor

A Figura 4.5 ilustra como o servidor lida com requisições ao banco de dados. A classe baseada no modelo - neste exemplo, AlunoController - estende a classe BaseController, que é implementada pelo framework Yii. O Yii divide as requisições ao banco de

dados em cinco categorias: index (para listar os objetos), create (para inserir um registro no banco), update (para atualizar um registro), view (visualizar apenas um registro) e delete (para excluir um registro).

Figura 4.5: Diagrama mostrando o acesso ao banco de dados no servidor



Fonte: Gerado pelo autor utilizando a ferramenta draw.io

5 APLICAÇÃO MOBILE PARA GERENCIAMENTO DE ALUNOS E TREINOS EM ACADEMIAS

O objetivo deste capítulo é descrever as características e funcionalidades da aplicação, descrevendo a maneira como ela foi desenvolvida e apresentando um panorama geral da aplicação e de suas funcionalidades.

5.1 Visão Geral

A aplicação descrita aqui tem como objetivo auxiliar professores que trabalham em academias de musculação a gerenciarem os dados de seus alunos, bem como de suas rotinas de treino. Além disso, os professores inserem e modificam informações sobre os aparelhos disponíveis na academia, juntamente com as possibilidades de exercícios a serem realizados nestes. A aplicação é destinada para a plataforma Android, visando a utilização da mesma em equipamentos mobile, e implementa um banco de dados local para manutenção de dados entre sessões.

Uma versão web da aplicação foi desenvolvida por Matheus Schilling Michel, com o objetivo de centralizar as informações e de prover uma base de coerência para os dados. Como o objetivo é que várias instâncias da aplicação sejam utilizadas simultaneamente, o site age como servidor, provendo e recebendo informações novas sobre os dados para que os dispositivos estejam sincronizados. O WebService foi apenas parcialmente implementado, possibilitando que apenas professores, alunos e rotinas de treino tenham registros sendo enviados de e para o servidor.

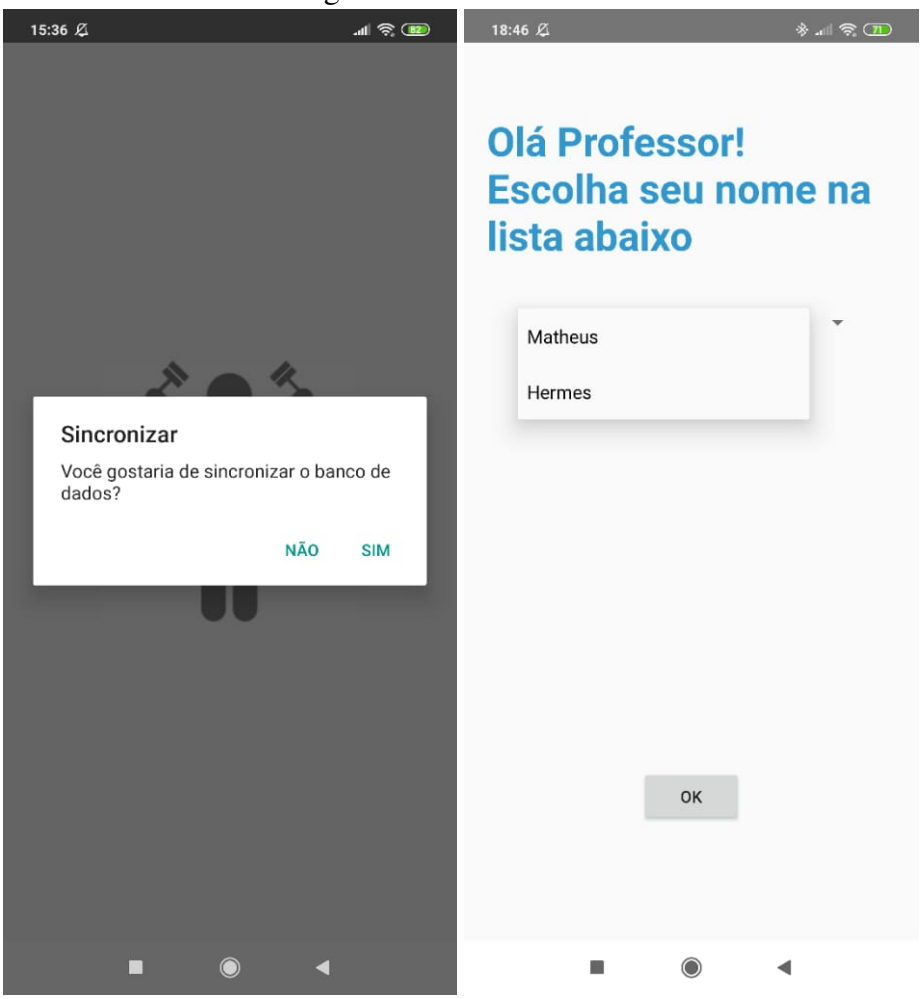
5.2 Funcionamento

A fim de demonstrar as diferentes funcionalidades da referida aplicação, serão utilizadas imagens demonstrativas da interface.

5.2.1 Sincronização Inicial e Tela de Boas Vindas

As Figuras 5.1a e 5.1b mostram as duas primeiras telas ao qual o usuário é apresentado.

Figura 5.1: Telas Iniciais



(a) Diálogo para Sincronização (b) Tela de Escolha

Fonte: Gerado pelo autor

Na primeira tela, uma janela de diálogo surge para perguntar se o usuário gostaria de realizar uma sincronização de dados com o servidor. Ao clicar 'Não', o aplicativo utiliza apenas os dados salvos localmente. Se a opção 'Sim' for escolhida, a aplicação enviará uma requisição para o servidor, que responderá com os dados salvos em seu banco, ao que o aplicativo realizará um *parse* JSON e irá, enfim, salvar estes dados em seu próprio banco local.

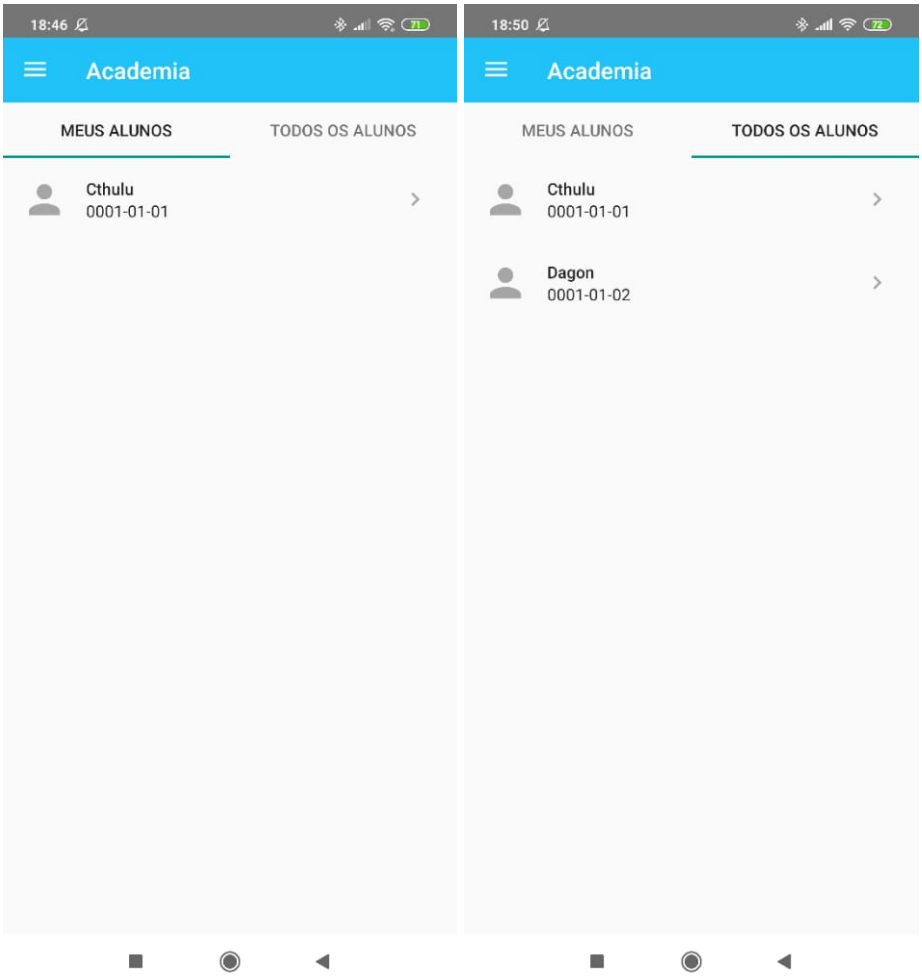
Na segunda tela é pedido que o professor (no caso, o usuário) escolha seu nome dentre aqueles de uma lista. Esta lista contém os dados salvos no banco ou que foram

trazidos do servidor. Importante denotar que, na primeira execução do aplicativo, é obrigatório que seja realizada a sincronização, ou não haverá itens nesta lista. A entidade professor pode ser criada apenas na versão web.

5.2.2 Lista de Alunos

As Figuras 5.2a e 5.2b mostram as visões respectivas das listas de alunos.

Figura 5.2: Listas de alunos



(a) Visão da lista de alunos do usuário (b) Visão da lista com todos os alunos

Fonte: Gerado pelo autor

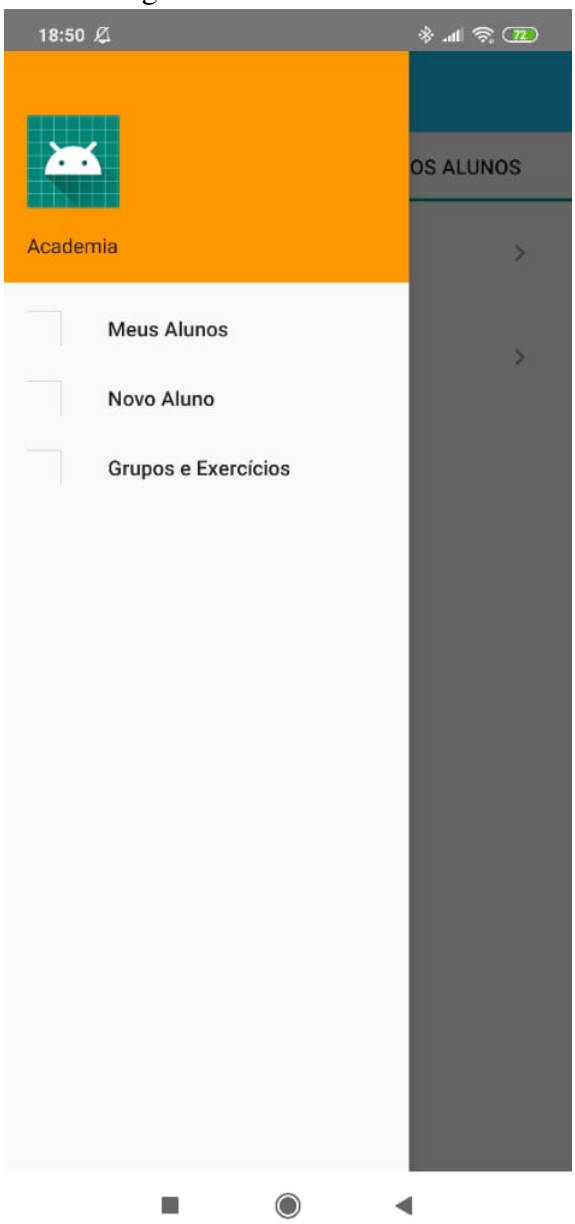
A primeira tela mostra apenas a lista de alunos do professor que está utilizando o aplicativo. Ao clicar na aba 'Todos os alunos', a aplicação mostra a lista de todos os usuários cadastrados no banco de dados. É possível ao usuário ver os alunos dos outros professores, pois assim foi requisitado, já que é comum alunos irem à academia fora do horário de trabalho de seu professor, sendo necessário que outro profissional possa

visualizar as rotinas de treino deste aluno.

5.2.3 Menu

A Figura 5.3 mostra o que acontece quando o ícone de hambúrguer no canto superior esquerdo da tela é clicado.

Figura 5.3: Visão do Menu



Fonte: Gerado pelo autor

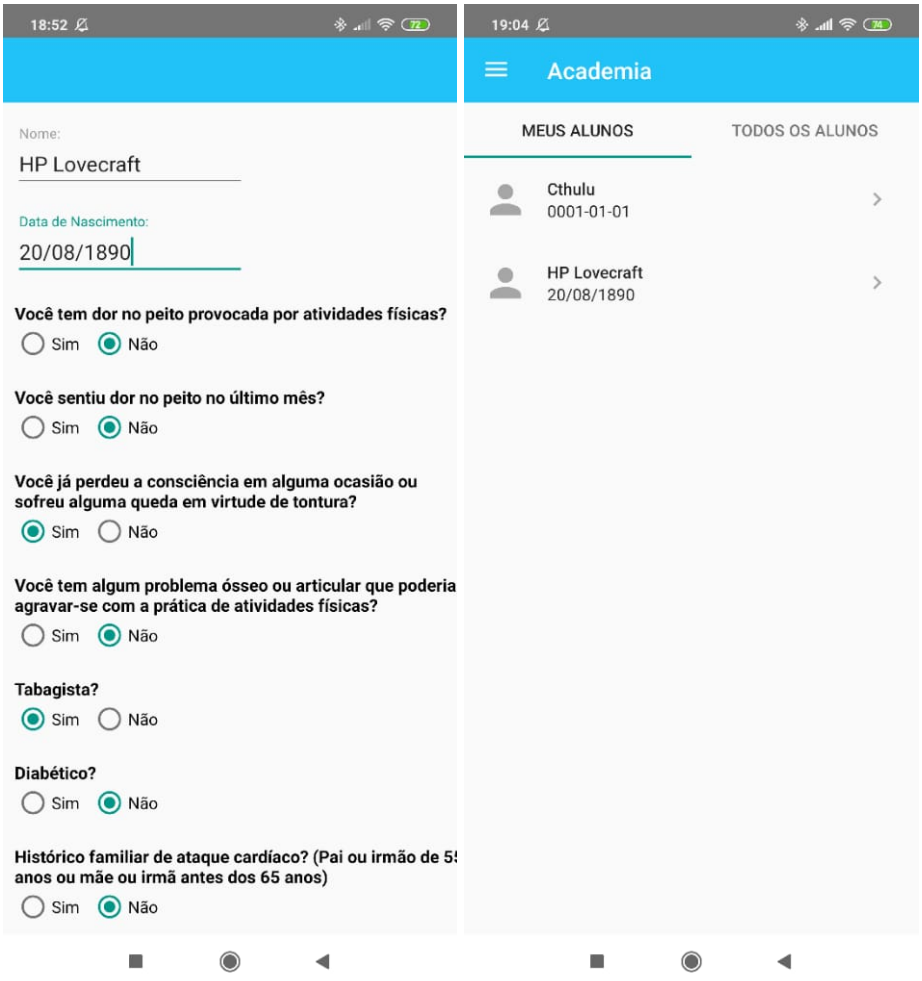
No caso, um Navigation Drawer aparece para mostrar o Menu com as seguintes opções:

- Meus Alunos - leva para as telas supracitadas.
- Novo Aluno - leva para a tela de cadastro de aluno.
- Grupos e Exercícios - a qual leva para a tela de visualização de Exercícios e dos respectivos Grupos musculares os quais eles fortalecem.

5.2.4 Cadastro de Novo Aluno

As Figuras 5.4a e 5.4b mostram as interfaces de cadastro de aluno e sua consequente adição à lista.

Figura 5.4: Cadastro



(a) Trecho da interface de cadastro de alunos

(b) Visão da lista de alunos do usuário atualizada

Fonte: Gerado pelo autor

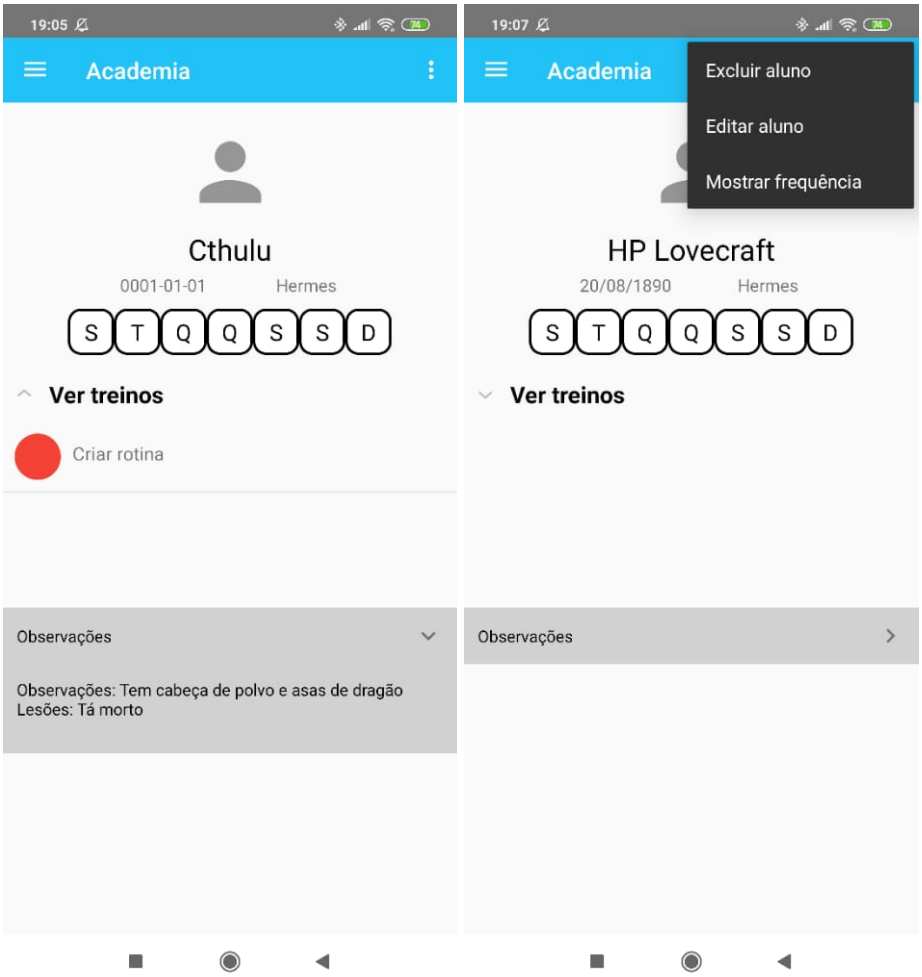
A Figura 5.4a mostra um trecho da tela de cadastro de novo aluno, com seus campos já preenchidos. O professor insere as informações e clica em salvar. Se algum

campo necessário não for preenchido - como nome ou data de nascimento -, a aplicação não salva o registro e pede que o usuário preencha tal campo. A Figura 5.4b ilustra a lista de alunos já atualizada com o aluno recentemente cadastrado.

5.2.5 Detalhes do Aluno

Nas Figuras 5.5a e 5.5b, apresenta-se a tela de detalhes do aluno.

Figura 5.5: Detalhes do Aluno



(a) Detalhe do Aluno (b) Menu relativo ao aluno

Fonte: Gerado pelo autor

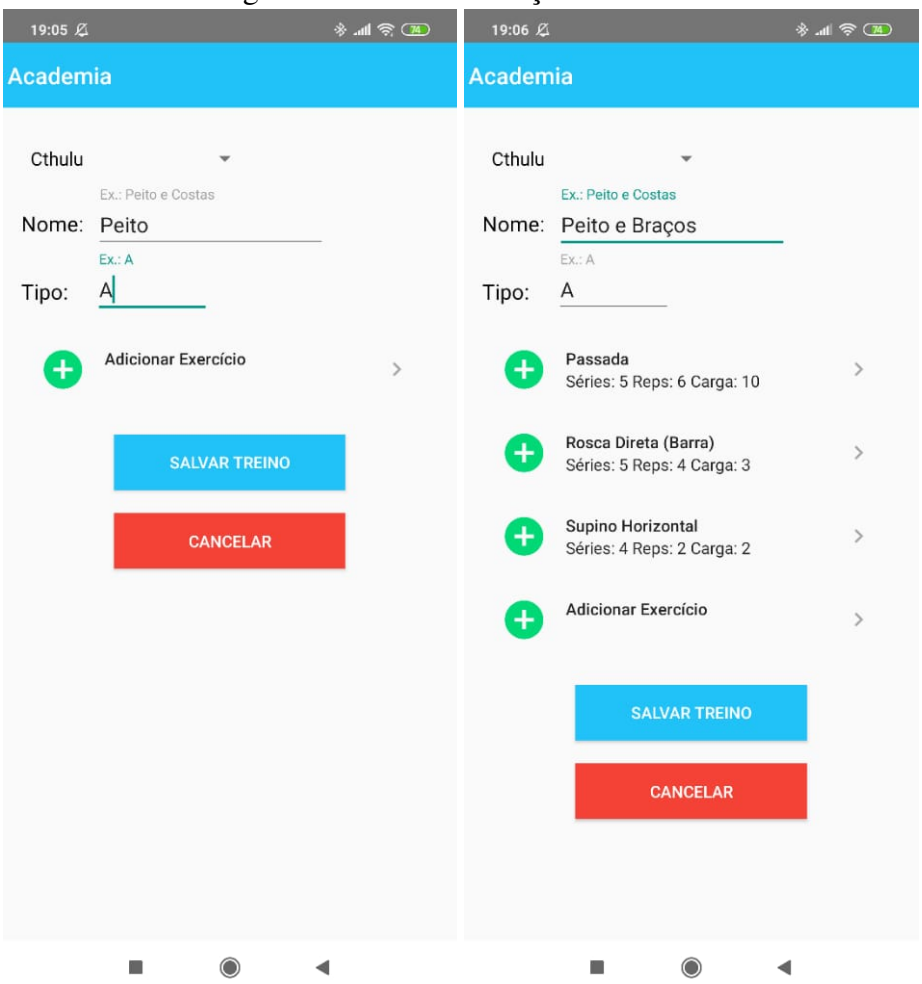
A Figura 5.5a ilustra a tela que surge quando se clica em algum aluno na lista da Figura 5.2a. Nesta tela é possível ver nome do aluno, data de nascimento, professor responsável, disponibilidade (os dias da semana em que o aluno pode ir à academia estarão marcados em verde, em próximas iterações), rotinas de treino, observações e lesões. O lugar em que aparecem as observações e lesões é uma caixa de texto expansível, que teve

seu funcionamento explicado no capítulo 4. O texto Ver Treinos é, na realidade, uma lista que mostra as rotinas de treino daquele aluno e um item chamado ‘criar rotina’, que ao ser clicado, leva para a tela de criação de treino. No canto superior direito há um botão que, ao ser clicado, abre um menu com opções referentes àquele aluno, como demonstrado na Figura 5.5b. As opções do menu são as seguintes:

- Excluir aluno - um diálogo aparece perguntando se o usuário tem certeza que deseja excluir este aluno. Se a opção escolhida for não, o diálogo desaparece. Se for sim, o usuário volta para a tela com os seus alunos.
- Editar aluno - o usuário é levado para a mesma tela de cadastro de aluno, porém com os campos preenchidos com os dados do aluno em questão, onde podem ser alterados.
- Mostrar frequência - esta função não está implementada, porém no futuro espera-se que um calendário seja mostrado com os dias que o aluno compareceu à academia e qual rotina de treino ele completou.

5.2.6 Criação de Treino

Figura 5.6: Tela de Criação de Rotina



(a) Tela de criação de treino sem exercícios

(b) Tela de criação de treino com exercícios adicionados

Fonte: Gerado pelo autor

Clicando no botão de Criar Rotina, uma nova tela aparece para a criação do treino, como nota-se na Figura 5.6a. Ali o professor vê o nome do aluno em uma lista *dropdown*, podendo trocar o aluno para o qual a rotina é criada. Também pode dar um nome para a rotina, como por exemplo os grupos musculares no qual esta vai focar. O campo 'Tipo' é um identificador que é utilizado na lista de treinos posteriormente. Para adicionar exercícios, o professor deve clicar no item correspondente. A Figura 5.6b mostra a mesma tela, porém com três exercícios já adicionados.

Figura 5.7: Detalhe do Aluno com Rotina

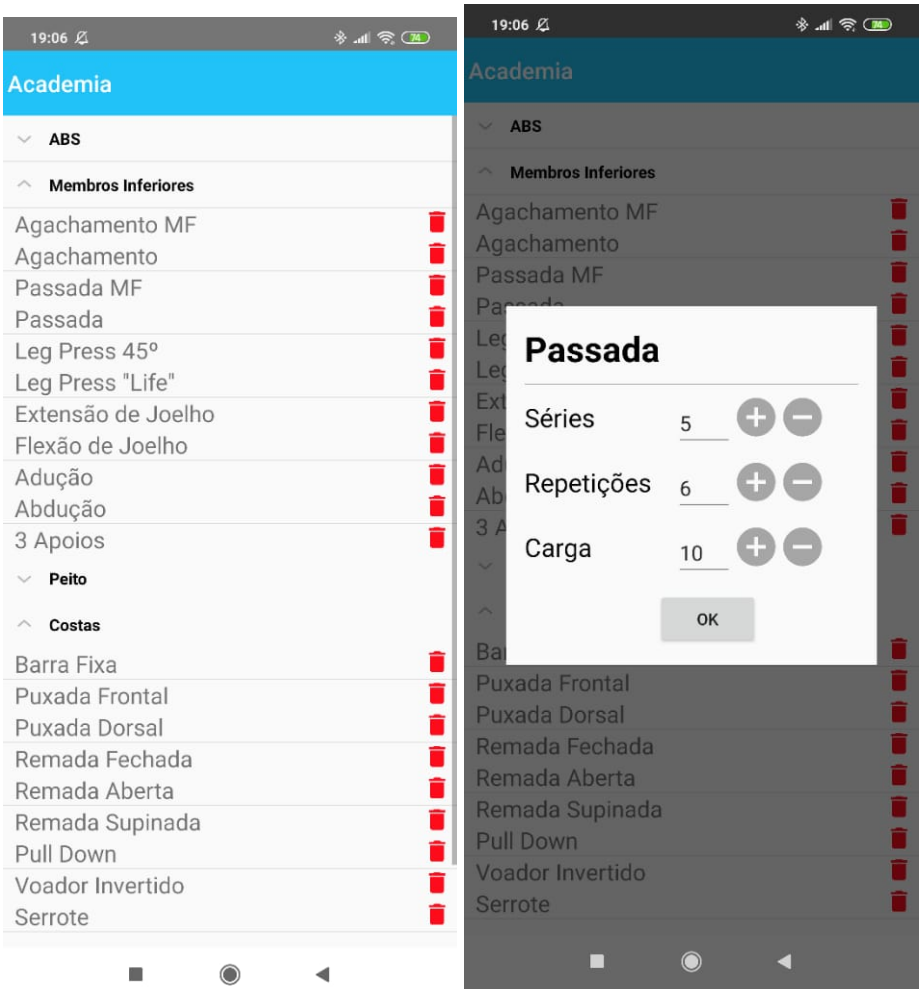
Fonte: Gerado pelo autor

Na Figura 5.7, tem-se a tela de detalhe do aluno com um treino adicionado. O identificador - no caso 'A' - fica no meio do círculo colorido para melhor visualização e será utilizado em futuras iterações da aplicação para identificar qual rotina de treino foi realizada em certo dia.

5.2.7 Adição de Exercício

Ao clicar no item ‘Adicionar Exercício’, na tela de criação de treino, o aplicativo leva para a tela mostrada pela Figura 5.8a.

Figura 5.8: Tela de Adição de Exercícios



(a) Lista expansível de exercícios

(b) Diálogo para definir parâmetros do exercício

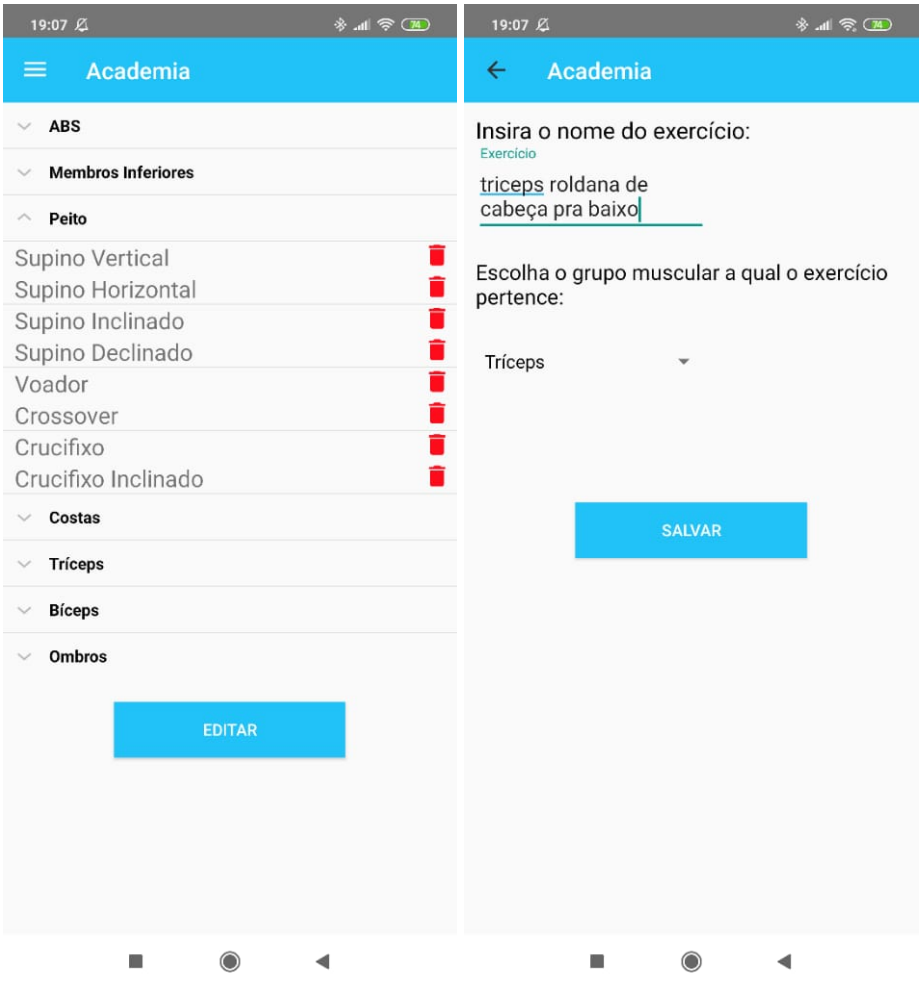
Fonte: Gerado pelo autor

Na primeira tela, tem-se várias listas expansíveis, cujos títulos são os nomes dos grupos musculares, e cujos itens são os exercícios que fortalecem tais grupos. Ao clicar no nome do exercício, um diálogo surge - como demonstrado pela Figura 5.8b - onde o usuário pode determinar quantas séries daquele exercício o aluno fará, assim como com qual carga e quantas repetições por série. Clicando em OK, a aplicação retorna para a tela de criação de treino, porém com o exercício adicionado na lista. Na lista de exercícios, caso o usuário clique no ícone de lixeira, um diálogo vai surgir perguntando se o usuário deseja excluir aquele exercício.

5.2.8 Grupos e Exercícios

Ao se clicar na terceira opção do menu principal, ‘Grupos e Exercícios’, a próxima tela surge, como demonstrada na Figura 5.9a. Esta tela mostra os grupos e exercícios cadastrados no banco de dados, com a opção de cadastrar um novo exercício (ao clicar no botão Editar). Também tem-se a opção de excluir o exercício, clicando no ícone vermelho de lixeira, fazendo com que surja um diálogo na tela perguntando se o usuário deseja realmente excluir o exercício.

Figura 5.9: Lista de Exercícios



(a) Lista de Grupos e Exercícios (b) Tela de cadastro de Exercício

Fonte: Gerado pelo autor

Na Figura 5.9b há a tela de cadastro de exercício, onde é requisitado o nome deste, assim como o grupo muscular ao qual ele pertence.

5.3 Limitações

Como o foco do desenvolvimento desta aplicação foi em torná-la o mais robusta possível, não houve tempo o suficiente para implementar todos os requisitos funcionais nesta primeira iteração do sistema. Um exemplo é a opção de o professor deixar várias rotinas de treino semiabertos, como em abas, para que ele possa visualizá-los de maneira rápida, sem precisar fazer o fluxo lista de alunos -> selecionar aluno -> ver rotinas de treino -> selecionar rotina.

Outro requisito faltando é a opção de mostrar um calendário para cada aluno, contendo os dias que o aluno foi à academia e qual rotina de treino foi realizada naquele dia.

Algo a ser desenvolvido para o futuro é a possibilidade de se incluir vídeos demonstrativos dos exercícios na plataforma, além de uma maneira de visualizar o histórico de dados e treinos dos alunos.

6 EXPERIMENTOS DE USABILIDADE

Este capítulo descreve o teste de usabilidade que foi realizado com usuários.

Para o teste do aplicativo, 10 usuários foram convidados pessoalmente, sendo 5 deles profissionais da área de educação física. O objetivo foi captar usuários de diferentes faixas etárias que tenham tido o costume ou não de utilizar aplicativos de musculação. Após realizar o teste, os usuários foram convidados a responder um questionário para verificar a Escala de Usabilidade do Sistema (SUS, na sigla em inglês). Este foi um método criado por John Brooke em 1986 para avaliar a eficiência, efetividade e satisfação do usuário em 10 perguntas. Cada pergunta é então respondida usando a escala Likert (LIKERT, 1932), onde o usuário deve selecionar uma opção de 1 a 5, sendo 1 'Discordo totalmente' e 5 'Concordo totalmente'. A variabilidade de aplicações da Escala de Usabilidade do Sistema e o fato de ser um teste cientificamente apurado - porém não demasiadamente longo - tornou o método uma referência para a indústria e extremamente popular.

6.1 Protocolo

Para a avaliação deste trabalho, foi pedido para os usuários que usassem o aplicativo simulando serem instrutores de academia. O usuário realizava algumas ações básicas, como cadastrar um novo aluno, criar um exercício novo, criar uma nova rotina de treinos para este aluno e visualizá-la. Logo após, é requisitado ao usuário que ele responda o questionário - que se encontra no Apêndice A - o qual divide-se em motivação da pesquisa, informações demográficas e questionário de Escala de Usabilidade do Sistema.

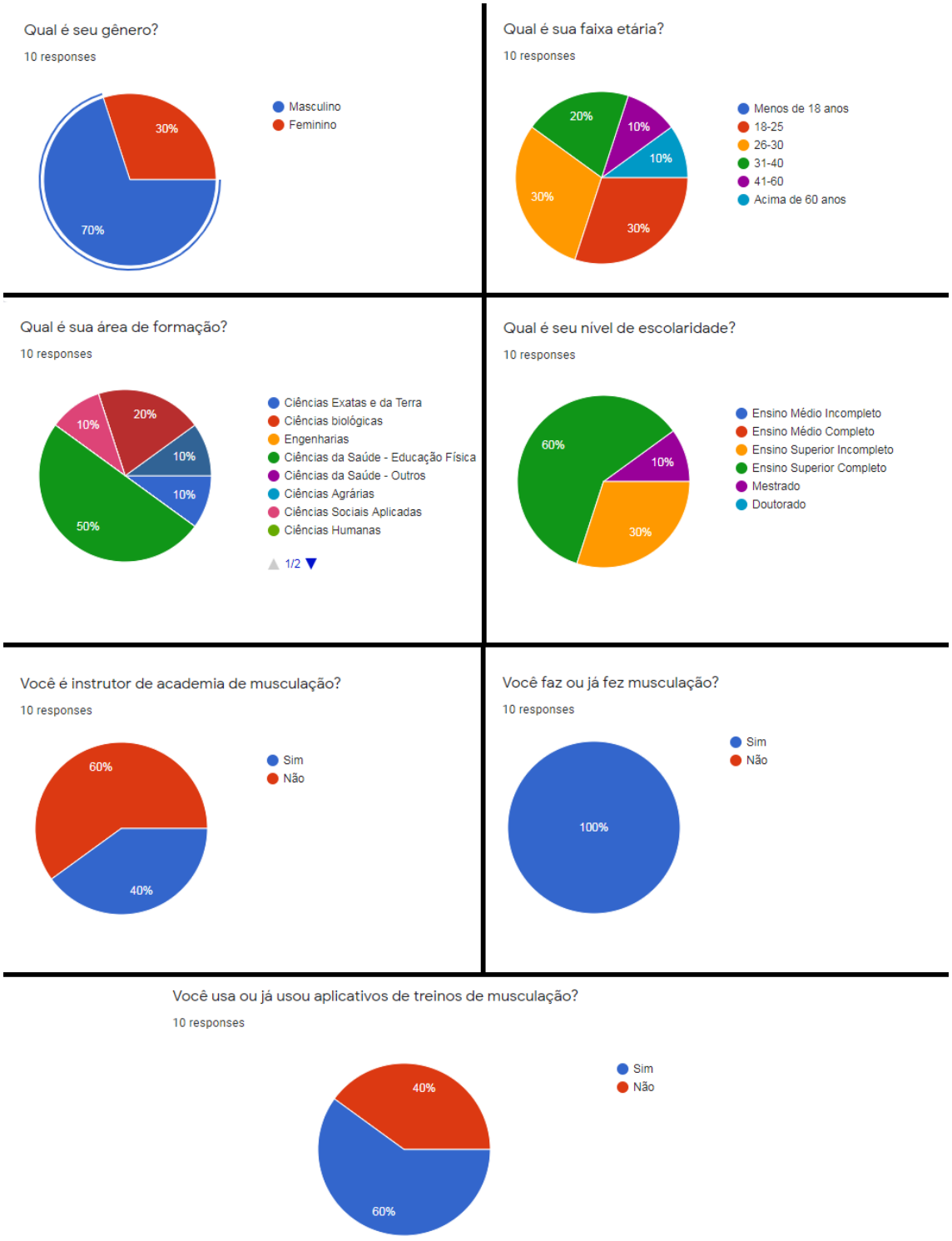
6.2 Participantes

O questionário foi respondido por 10 usuários, sendo que 5 destes têm como formação a área de Educação Física. Como o aplicativo é destinado aos instrutores de academia, é importante que boa parte dos usuários de teste sejam o público alvo. As idades dos usuários foram bem variadas, sendo que 30% têm entre 18 e 25 anos, 30% têm entre 26 e 30 anos, 20% estão entre 31 a 40 e apenas 10% estão entre 41 e 60, assim como apenas 10% possuem mais de 60 anos. A única faixa etária que não foi contemplada entre os usuários de teste foi abaixo de 18 anos. Quanto ao nível de escolaridade, todos os usuá-

rios tiveram pelo menos Ensino Superior Incompleto, sendo que 70% já são graduados em alguma área.

Algumas perguntas foram feitas na intenção de descobrir a relação do usuário com o ambiente de academia e musculação. Todos os usuários marcaram que fazem ou já fizeram musculação e 40% marcaram que são instrutores de academias. No entanto, apesar de todos já terem experiência com musculação, apenas 60% marcaram que usam ou já usaram aplicativos destinados ao treino de musculação. Os resultados sumarizados sobre o questionário de informações demográficas se encontram na Figura 6.1.

Figura 6.1: Resultado Sumarizado das Perguntas Pessoais



Fonte: Gerado pelo autor

6.3 Resultados

De acordo com Brooke[1], para calcular a pontuação do teste de usabilidade, deve-se realizar o seguinte procedimento:

- Para as respostas ímpares, subtrair 1 da pontuação escolhida pelo usuário.
- Para as respostas pares, subtrair a pontuação escolhida de 5.
- Somar os valores das perguntas e multiplicar por 2.5

Realizando estes três passos, se tem o resultado do teste, que vai de 0 a 100. O trabalho aqui apresentado teve uma ótima pontuação, sendo 91.5 a média das respostas dos 10 usuários. Entre os usuários que são da área de Educação Física, a média é até maior, sendo esta de 94. Ambos os resultados estão acima da média e dentro da classificação A (acima de 80.3), considerada a melhor classificação possível.

O resultado do questionário SUS encontra-se na Tabela 6.1.

[1]BROOKE, John et al. SUS-A quick and dirty usability scale. Usability evaluation in industry, v. 189, n. 194, p. 4-7, 1996.

Tabela 6.1: Resultados do Questionário

Questão	Enunciado	Concorda	Neutro	Discorda
Q1	Eu gostaria de utilizar esse aplicativo com frequência	80%	20%	0%
Q2	Eu acho o aplicativo desnecessariamente complexo	10%	10%	80%
Q3	Eu achei o aplicativo fácil de usar	100%	0%	0%
Q4	Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo	0%	0%	100%
Q5	Eu acho que as funções do aplicativo estão bem integradas (ex.: clicar no aluno para ver os detalhes deste, navegar pelo app, etc)	80%	10%	10%
Q6	Eu acho que o aplicativo apresenta muita inconsistência	0%	0%	100%
Q7	Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente	100%	0%	0%
Q8	Eu achei o aplicativo atrapalhado de usar	0%	0%	100%
Q9	Eu me senti confiante usando o aplicativo	100%	0%	0%
Q10	Eu precisei aprender várias coisas novas antes de usar o aplicativo	0%	0%	100%

Fonte: O Autor

O teste de usabilidade foi desenvolvido de uma maneira que as questões ímpares são de teor positivo em relação à aplicação - sendo assim, desejável que as respostas tendam a ‘Concordo’ nestas opções - e as questões pares são negativas e, portanto, rendem

uma pontuação maior se tiverem uma maioria de respostas tendendo a ‘Discordo’. Nota-se, pelos resultados da Tabela 6.1, que as respostas das questões atingem o desejado. Ao menos 80% dos usuários gostariam de utilizar o aplicativo com frequência, e todos os usuários testados concordaram que o aplicativo é fácil de usar e que se sentiram confiantes usando-o. A única questão que teve alguma divergência - mesmo que pequena - foi quanto à qualidade da integração das funções do aplicativo. Quanto às questões pares, nota-se que a única que teve uma divergência - também pequena - foi quanto à complexidade do aplicativo, e se o usuário achava isto desnecessário. No tocante às outras perguntas, é possível deduzir que os usuários não precisaram aprender coisas novas antes de usar o aplicativo, acham que ele é consistente e não acham que precisariam de ajuda de alguém com conhecimentos técnicos.

7 CONCLUSÕES

Este trabalho teve como intenção apresentar uma aplicação destinada ao instrutor de musculação em academias, sendo útil para este e para os alunos deste. Ela auxilia o instrutor no controle e gerenciamento dos seus alunos e em suas rotinas de treino, agilizando e facilitando sua função, porém sem excluir o importante papel que o instrutor tem no processo de fortalecimento do aluno. A aplicação começou a ser desenvolvida no segundo semestre de 2019, coletando requisitos a partir de pessoas que estão inseridas no meio de academias de musculação. A aplicação faz parte de um sistema que compreende um banco de dados principal em um servidor e uma página web, esta desenvolvida por Matheus Schilling Michel. Nos próximos meses é previsto que todo o sistema esteja totalmente funcional e em produção. Os resultados dos experimentos foram satisfatórios e indicam que o aplicativo é fácil de usar, que existe uma demanda para ele - pois 80% dos usuários disseram que o usariam com frequência -, que é consistente e de fácil aprendizado.

7.1 Trabalhos Futuros

Para o futuro, seria interessante que a aplicação implementasse gráficos evolutivos sobre os dados dos alunos, como peso corporal, métricas e carga dos exercícios. Uma funcionalidade que está nos planos de implementação é um calendário que mostra a frequência com a qual o aluno foi à academia e quais rotinas de treino ele realizou nestes dias. Outra funcionalidade ainda não implementada neste trabalho foi mostrar, na tela de detalhes do aluno, a disponibilidade semanal dele de ir à academia. Outras aplicações possíveis poderiam ser implementadas, como um sistema para integrar as informações do aplicativo com informações de uma Pulseira Inteligente.

8 APÊNDICE A - QUESTIONÁRIO APLICADO AOS USUÁRIOS

Neste apêndice encontra-se o questionário aplicado via web aos usuários.

Avaliação da Aplicação de Sistematização de Fichas

Este questionário faz parte de um trabalho de conclusão do curso de Ciência da Computação do INF-UFRGS e busca coletar opiniões sobre a Aplicação de Sistematização de Fichas de Academia, uma aplicação que almeja substituir a utilização de fichas de papel da parte de instrutores de academias de musculação, possibilitando ao usuário gerenciar alunos e suas rotinas de treino. O objetivo desta pesquisa é avaliar a usabilidade da aplicação. O tempo aproximado deste questionário é de 10 minutos e os resultados serão agrupados de forma anônima.

Perfil do Usuário

1. Qual é seu gênero?
   - Masculino
   - Feminino
2. Qual é sua faixa etária?
   - Menos de 18 anos
   - 18-25
   - 26-30
   - 31-40
   - 41-50
   - Acima de 60 anos
3. Qual é seu nível de escolaridade?
   - Ensino Médio Incompleto
   - Ensino Médio Completo
   - Ensino Superior Incompleto
   - Ensino Superior Completo

- Mestrado
- Doutorado

4. Qual é sua área de formação?
    - Ciências Exatas e da Terra
    - Ciências biológicas
    - Engenharias
    - Ciências da Saúde - Educação Física
    - Ciências da Saúde - Outros
    - Ciências Agrárias
    - Ciências Sociais Aplicadas
    - Ciências Humanas
    - Linguística, Letras e Artes
    - Outro
5. Você é instrutor de academia de musculação?
    - Sim
    - Não
6. Você faz ou já fez musculação?
    - Sim
    - Não
7. Você usa ou já usou aplicativos de treinos de musculação?
    - Sim
    - Não

Perguntas relativas ao Aplicativo

1. Eu gostaria de utilizar esse aplicativo com frequência

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

2. Eu acho o aplicativo desnecessariamente complexo

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

3. Eu achei o aplicativo fácil de usar

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

5. Eu acho que as funções do aplicativo estão bem integradas (ex.: clicar no aluno para ver os detalhes deste, navegar pelo app, etc)

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

6. Eu acho que o aplicativo apresenta muita inconsistência

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

7. Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

8. Eu achei o aplicativo atrapalhado de usar

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

9. Eu me senti confiante usando o aplicativo

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

10. Eu precisei aprender várias coisas novas antes de usar o aplicativo

	1	2	3	4	5	
Discordo plenamente	○	○	○	○	○	Concordo plenamente

REFERÊNCIAS

G.E. KRASNER, S.T. POPE. 'A Cookbook for Using the Model-ViewController User Interface Paradigm in Smalltalk-80'. Joop, August/September, 1988

PANCHAL, Ronak K.; PATEL, Mr Akshay K. A comparative study: Java Vs kotlin Programming in Android. International journal of Innovative Trends in Engineering 'I&' Research, v. 2, n. 9, 2017.

LASTER, Brent. Professional Git. John Wiley 'I&' Sons, 2016.

PÖLHS, M.; PEITEK, N. Retrofit: Love Working with APIs on Android, 2017.

SUBRAMANIAM, Venkat; HUNT, Andy. Practices of an agile developer: Working in the real world. Pragmatic Bookshelf, 2006.

FREEMAN, Eric et al. Head first design patterns. "O'Reilly Media, Inc.", 2008.

BROOKE, John et al. SUS-A quick and dirty usability scale. Usability evaluation in industry, v. 189, n. 194, p. 4-7, 1996.

LIKERT, Rensis. A technique for the measurement of attitudes. Archives of psychology, 1932.

https://db-engines.com/en/ranking

https://github.com/google/gson

https://gauchazh.clicrbs.com.br/saude/conteudo-publicitario/2018/02/conheca-os-riscos-de-fazer-exercicios-fisicos-por-conta-propria-cjdex0qwf007b01n3u6a0c7bh.html

https://oniquenutrition.com/blog/aplicativos-fitness/

https://kotlinlang.org/

https://www.sqlite.org/about.html

https://square.github.io/retrofit/

https://github.com/hakobast/DropdownTextView

https://developer.android.com/studio