FitDolphin: Aplicativo para controle de atividades físicas para plataforma Android

João Pedro Santos Vidiri [1]
Daniela Marques [1]

[1]Instituto Fereral de São Paulo (IFSP)
Avenida Thereza Ana Cecon Breda, s/n - Vila São Pedro
Hortolândia-SP - Brasil - Cep: 13183-25

Abstract. *Some fitness centers uses paper forms to assist users annotate theirs exercises. The paper where this notes are taken degrades easily in the fitness center environment, in the other hand there is a crescent access of Brazilian people to smartphones that can be used in those ambients. This paper propose the development of an application for the Android platform based on the paper forms of fitness centers, it centralizes the control of the fitness activities that the user did, differentiated by the customizability of the training routines.*

Resumo. *Algumas academias utilizam fichas de papel para acompanhamento do aluno, de seu desempenho e anotação de exercícios realizados. O papel em que é escrita essa ficha se degrada rapidamente no ambiente de academia. Aliado a isso, há um crescente acesso dos brasileiros a smartphones que podem ser utilizados nesses ambientes. Este trabalho mostra o desenvolvimento de um aplicativo para a plataforma Android baseado nos requisitos dessas fichas de acompanhamento, com o intuito de controle das atividades físicas realizados pelo usuário, com o diferêncial de ser customizável para o treino que o usuário recebeu de um profissional.*

1. Introdução e Motivação

No mundo moderno a prática de atividades físicas se mostra uma opção para a melhora na saúde [Um 2004] [Haitao 2011], a diminuição dos riscos de se contrair uma doença crônica [Booth FW 2012], além de a prática de diminuir o risco de obesidade [Donnelly 2013].

Tratado como um tema prioritário pelo governo brasileiro [Rocha 2017b], a melhora da qualidade de vida da população brasileira tem se demonstrado um desafio, já que há um aumento ano a ano dos índices de obesidade e sedentarismo na população [Rocha 2017a].

Na contramão desses índices, há um crescimento de 51% no uso de aplicativos voltados para exercícios e dietas [Saifi 2017]. Os aplicativos focados em atividades físicas tentam ajudar o usuário a praticar esportes, como corrida ou bicicleta, e mostrar os locais por onde o usuário passou e qual a média de velocidade. As grandes empresas de tecnologia tem aplicativos para o controle das práticas de exercícios, como o *Google Fit* (Figura 1) e o *Apple Health*.

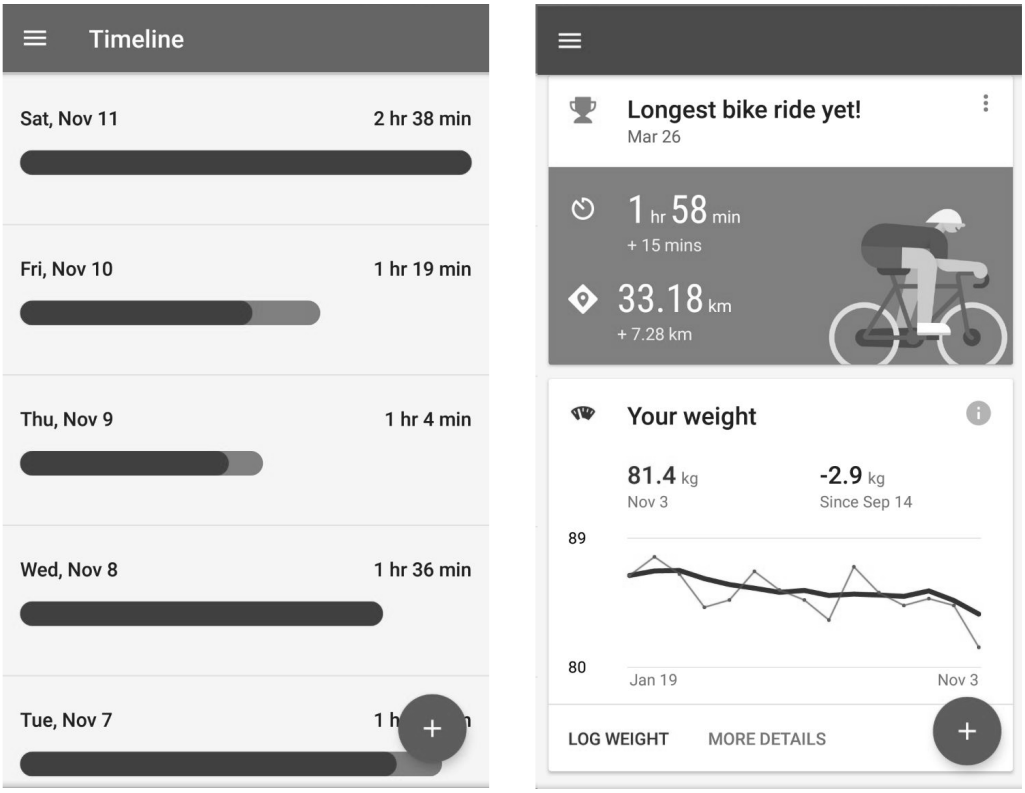

Figura 1. Reprodução aplicativo Google Fit

Com o uso cada vez mais difundido dos smartphones pela sociedade em todas as faixas etárias, torna-se natural que o *smartphone* acompanhe a prática de exercícios.

Este trabalho apresenta o desenvolvimento de um aplicativo para a monitoração de atividades fisicas, além de ser customizável para o usuário, permitindo o cadastro de exercícios e rotinas de treinamento.

Este artigo está dividido da seguinte maneira: na Seção 2 são identificados os trabalhos correlatos, a Seção 3 mostra os diferenciais do aplicativo, a Seção 4 mostra o referêncial teórico, a Seção 5 descreve a metodologia utilizada, a Seção 6 apresenta o trabalho realizado e a Seção 7 conclui o trabalho.

2. Trabalhos Correlatos

Existem duas categorias de aplicativos existentes para monitoramento na *play store*, encontrados durante as pesquisas, há os *indoor* que são aplicativos que te auxiliam em exercicios em ambiente controlado, na academia ou sua própria casa, e os *outdoor* que são aplicativos de *tracking*, ou seja fazem um mapeamento das suas atividades fisicas na rua, como bicicleta ou corrida. Dentro dessas categorias existem aplicativos que apenas anotam suas atividades e os que te entregam treinos programados para serem feitos.

Esta Seção mostra alguns aplicativos existentes que serviram como base de pesquisa para a proposta desse trabalho, suas principais funcionalidades e contrasta os recursos já existentes no mercado com os propostos para o aplicativo deste artigo.

2.1. Google FIT

Aplicativo desenvolvido pela Google para o acompanhamento de atividades físicas ao ar livre. Possui integração com o sistema operacional *Android* e utiliza dados de geolocalização para gerar acompanhamento diário de suas atividades, sem que seja preciso intervenção direta do usuário [Google 2018b].

2.2. MyFitnessPal

Aplicativo de controle de calorias diárias do usuário. Utiliza a inclusão de dados feita pelo próprio usuário por meio de formulários no aplicativo para contar as calorias da dieta e as calorias gastas na execução de exercícios [MyFitnessPal 2018].

2.3. Smart Fit Coach

Aplicativo de treinos da rede de academias *SmartFit* Brasil, o usuário tem a opção de criar um treino baseado em perfis pré-definidos, com ajuda de um formulário inteligente que tem um formato de *chatBot* com perguntas e respostas, que simulam um aplicativo de mensagens [de Ginástica S/A 2018].

3. Diferenças do aplicativo FitDolphin com relação aos concorrentes.

Os aplicativos que fazem o mapeamento dos exercícios dos usuários, em suma não são customizáveis ou fazem o usuário cadastrar os exercícios após a realização dos mesmos. O aplicativo proposto é um centro de atividades do usuário com o diferêncial da customização dos treinos que já tem prontos e servirá para anotar e executar esses exercícios.

Baseado em *feedbacks* dos usuários na *google play store* um dos problemas do aplicativo da *SmartFit* é a impossibilidade de customizar seu treino, já que os treinos são pré definidos e não podem ser alterados segundo a vontade do usuário. O aplicativo proposto foi feito para permitir uma customização dos treinos por parte do usuário, assim o mesmo não perde o treino já criado pelo seu *personal trainer* ou professor de academia.

Na maioria dos aplicativos, o usuário deve cadastrar o seu histórico de exercícios de academia. Nesta proposta de trabalho o usuário será auxiliado na realização dos exercícios e o histórico é gerado automáticamente após a realização de cada rotina de exercícios cadastrada e configurada para o dia de execução.

4. Referêncial Teórico

Nesta Seção são apresentados os referênciais teóricos utilizados para o desenvolvimento deste trabalho.

4.1. Modelo de desenvolvimento

O objetivo dos modelos de desenvolvimento é ajudar a trazer regras para o desenvolvimento do software, a fim de facilitar a coordenação das equipes [Munassar and Govardhan 2010].

Um modelo do tipo prescritivo pode ser utilizado quando se conhece razoavelmente os requisitos iniciais, mas se tem um escopo geral ainda grande e indefinido. O modelo de processo incremental, foca em uma disponibilização rápida de um conjunto funcional ao usuário e após a entrega inicial seu refinamento, ele é útil também quando

não há pessoal necessário para a implementação total do sistema no prazo estabelecido para o projeto [Munassar and Govardhan 2010].

Os incrementos são versões selecionadas do produto final, mas essas versões possuem capacidade para atender ao usuário e também oferecem uma plataforma para a avaliação [Munassar and Govardhan 2010].

4.2. Arquitetura

A arquitetura de um programa ou sistema computacional é a estrutura ou estruturas do sistema, as quais englobam elementos de software, as propriedades visíveis destes elementos, e a relação entre eles [Len Bass 2003].

Primeiramente a arquitetura é uma abstração do sistema, e portanto deixa de fora algumas partes do software, assim, é interessante notar que ela consegue encaixar em qualquer sistema, já que todos eles podem ser representados por elementos e suas relaçoes [Len Bass 2003].

O desenvolvimento de arquitetura pode ser feito sem padrão, ou utilizar modelos de mercado como o padrão *MVC (Model-View-Controller)* [Fowler 2003].

4.2.1. *MVC*

O padrão de arquitetura *MVC* é usado primariamente para criação de *Graphic User Interfaces* (GUI), a premissa é modularizar os três principais aspectos das GUI: o Modelo de dados (*Model*), a representação deles para o usuário (*View*) e a interface entre o modelo de dados e a representação gráfica dos dados (*Controller*). Separar o sistema nesses três componentes significa que cada um deles seja o mais independente possível dos outros e alterações feitas em um, não afetam alterações nos outros, assim permitindo a atualização da interface sem que seja necessário alterar os modelos de dados e os controladores [Fowler 2003].

Geralmente o *Model* é implementado primeiro, o modelo de dados tem duas atribuições principais: guardar e manipular alterações nos dados. O primeiro é basicamente encontrar como armazenar os dados, e o segundo engloba funções para executar, identificar e repassar as alterações de dados, permitindo a interface se adaptar de maneira consistente aos dados [Fowler 2003].

Uma vez feito o modelo de dados, já se pode construir as visualizações para os dados, isso é responsabilidade da *View*. Ela identifica as alterações dos dados por meio dos métodos criados no modelo e os apresenta aos usuários. Duas perguntas importantes a se pensar quando se esta desenvolvendo a *View* são: (i) "O que eu devo fazer quando há mudança nos dados?"e (ii) "Como eu disponibilizo isso ao usuário?"[Fowler 2003].

O restante da interface, que não são apresentados ao usuário, são responsabilidade do *Controller*, basicamente a *View* são dados estáticos, e quando há um evento de alteração por parte do usuário é necessário que o *Controller* receba essa requisição, processe e a encaminhe para o *Model* para armazenamento e atualização de estados [Fowler 2003].

4.2.2. Requisitos de Software

Segundo Sommerville [SOMMERVILLE 2011], "Requisitos de sistema são descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento". No entanto, ainda há incosistências no uso do termo "requisito", pois dependendo do tempo de cada projeto é necessário ter requisitos mais abstratos, ou mais específicos para uma definição de solução do problema em si.

Outro conceito importante é referente a (i) requisitos funcionais, aqueles que definem um serviço do sistema e como deve proceder o processamento dos dados desse serviço; e (ii) requisitos não funcionais, que abrangem requisitos de produto, requisitos externos ou requisitos da organização.

4.3. Plataformas e ferramentas de desenvolvimento

O aplicativo funciona no sistema operacional Android [Google 2018a], é o sistema móvel mais utilizado no Brasil e cresce consistentemente nos últimos anos [Sales 2016]. Para teste do aplicativo será utilizado a ferramenta de emulação Genymotion [Genymobile 2018] e o *Android Debug Bridge (ADB)* para o *debug* no *smartphone* do autor.

O desenvolvimento do código fonte será na ferramenta Android Studio [Google 2017] fornecido pela Google, utilizando a linguagem de programação JAVA [Corporation 2018b].

Há uma integração da aplicação *mobile* com um *webservice rest* [Zhao and Doshi 2009] que funciona em estrutura Apache [Foundation. 2018] e os *endpoints rest* serão escritos em linguagem PHP [Group 2018], com o uso do *Slim Framework* [Allen and the Slim Framework Team 2018], a ferramenta de escrita do código para esta parte do aplicativo será o Sublime-Text [Ltd 2018], os dados serão tranferidos no formato *JSON* [Json.org 2018].

O sistema operacional para a execução dos programas de desenvolvimento, testes e *debug* será o Ubuntu [Ltd. 2018].

4.4. Modelagem de dados

Um banco de dados é uma coleção de dados relacionados para manutenção dos dados de uma organização, assim sendo definido uma vez e acessado por diversos usuários. Seguindo [HEUSER 1998], "um modelo de (banco de) dados é uma descrição dos tipos de informações que estão armazenadas em um banco de dados".

No armazenamento local *Android* foi escolhido o banco de dados *ObjectBox* [Limited 2018a] por ser uma maneira simples e rápida de guardar objetos, assim facilitando o desenvolvimento das operações de *CRUD (Create, Read, Update and Delete)* no aplicativo. Também foi desenvolvido um *webservice* com banco de dados relacional *MySql* [Corporation 2017], foi escolhido este tipo de banco por sua natural integração com a linguágem PHP [Group 2018].

4.5. Prototipação

O *design* do sistema deve ser iniciado após a elucidação dos requisitos. O *design* deve ser definido em ciclos de *design*, quando se cria o conceito, avalia-se esse conceito e por fim

se faz o *redesign*. Após a definição conceitual do *design*, se pode criar protótipos que são ferramentas para a interação inicial e validação do *design* [PREECE 2013].

Os protótipos por sua vez podem ser divididos entre de baixa fidelidade, aqueles sem muito apreço pela forma e que não se parecem muito com o produto final e os de alta fidelidade, esses mais parecidos com o produto final, são mais caros e difíceis de fazer [PREECE 2013].

5. Resultados

O desenvolvimento se deu durante o período de janeiro de 2018 e julho de 2018, abaixo o resultado obtido do trabalho.

A interação do usuário se dará por meio de aparelho móvel, a *interface* do aplicativo vai se basear no padrão de desenvolvimento do *Android*, o *Material Design* [Google 2018c].

5.1. Elucidação de requisitos

A elucidação dos requisitos se deu por meio da ficha da acâdemia (Figura 2) e na experiência de academia do aluno, baseado nisso se chegou aos seguintes casos de uso (Figura 3) para o aplicativo.

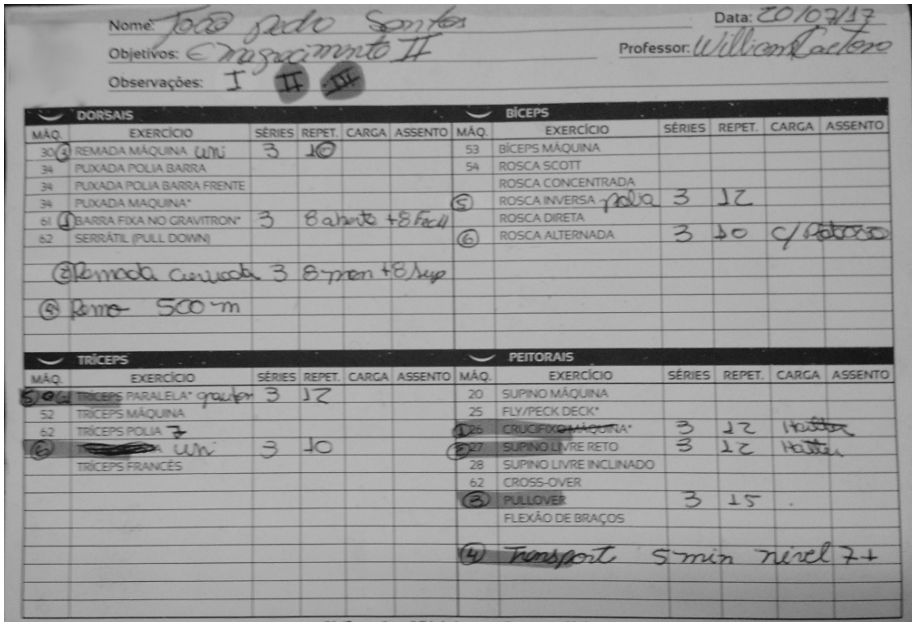

Figura 2. Reprodução Ficha de academia

1. Manter exercícios: o usuário ordena seus exercícios conforme deseja. Também é permitido que o usuário altere a ordem dos exercícios assim como edite os detalhes dos exercícios de maneira rápida.
2. Manter rotinas: o usuário organiza seus exercícios em rotinas. Desta maneira, será possível ter treinos diferenciados para cada dia da semana ou para cada membro da musculatura (por exemplo: pernas, tronco, etc.).
3. Manter conta do usuário: o usuário edita as informações da sua conta, nome, email, objetivo, entre outros dados.

4. Manter objetivo: o usuário define um objetivo para suas rotinas de exercícios para motivar e ter foco ao realizá-los.
5. Iniciar rotina de exercícios: o usuário inicia uma rotina de exercícios cadastrada anteriormente, ao fim do mesmo sumarizando qual a porcentagem dos treinos foi realizada pelo usuário.
6. Enviar *feedback*: uma maneira fácil e prática de entrar em contato com o desenvolvedor do aplicativo e que pode ou não ser de forma anônima.

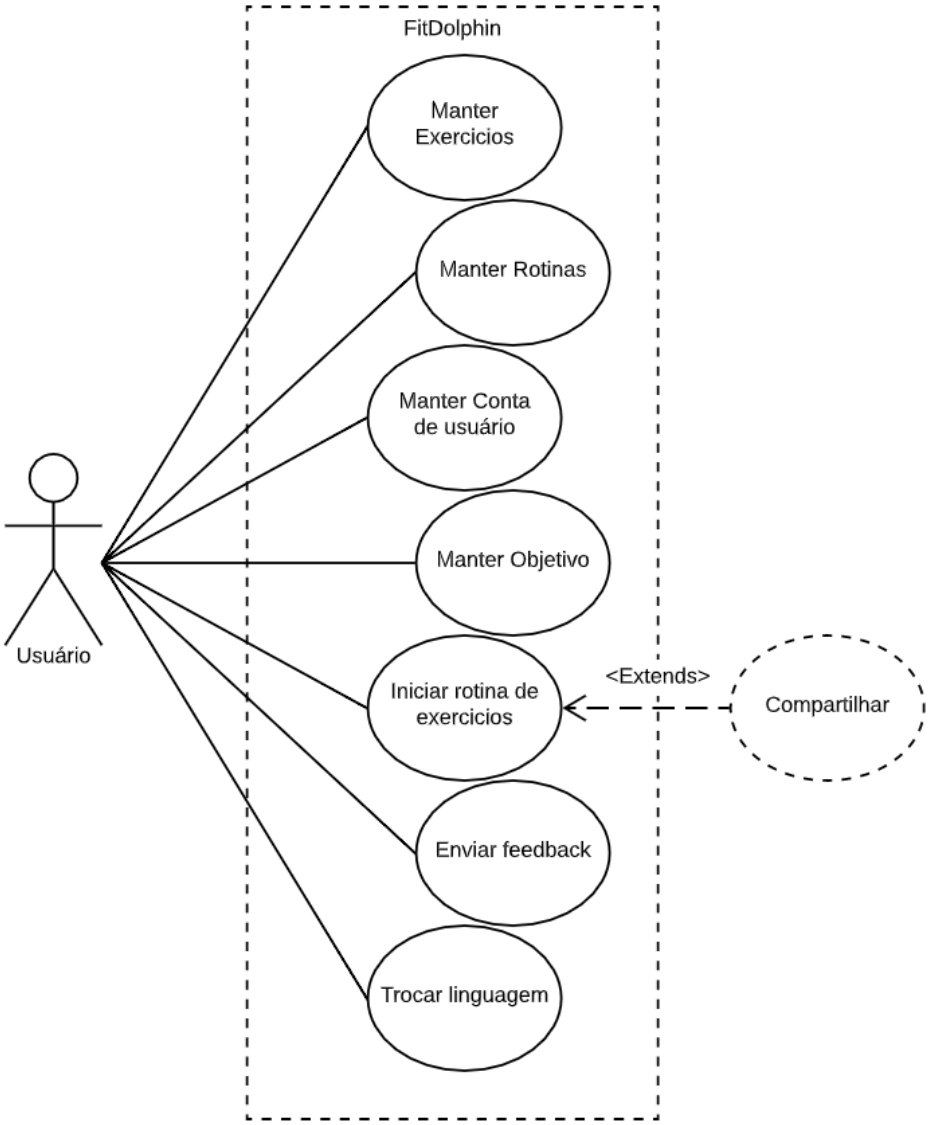

Figura 3. Casos de uso do Aplicativo FitDolphin

5.2. Criação de banco de dados

O banco de dados foi criado com base na ficha técnica da academia (Figura 2) e nos casos de uso identificados. A Figura 4 mostra Diagrama Fisico do banco elaborado.

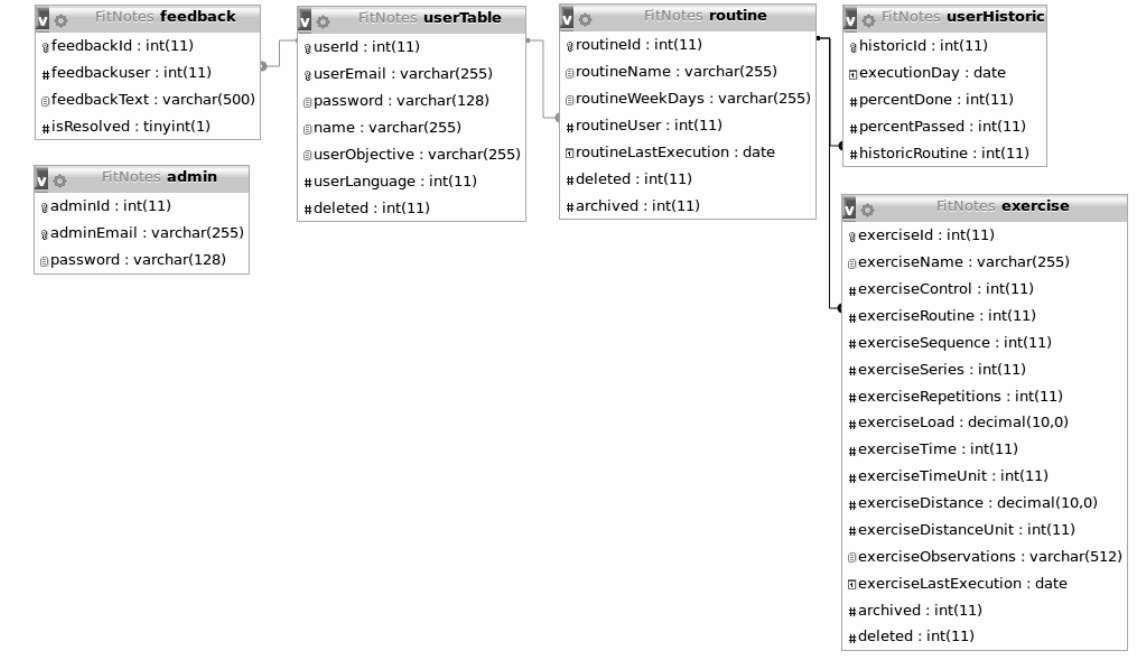

Figura 4. Diagrama Entidade-Relacionamento

O autor também utilizou o banco de dados *Object Box* para criar as entidades no Android, o seu desenvolvimento é ágil, basta criar as classes representando os objetos que se deseja salvar no banco de dados e utilizar uma *Annotation* [Corporation 2018a] para que o módulo do *object box* identifique o objeto e, automaticamente durante o *build* o modúlo identifique as classes anotadas e crie toda a estrutura de *CRUD* para a mesma [Limited 2018b], as entidades do *object box* foram criadas com base nas mesmas entidades do banco de dados relacional (MySQL).

A 5 mostra a arquitetura do aplicativo, onde é possível verificar o banco de dados local para o uso de maneira *offline* e o banco de dados relacional acessado via *webservice* para os usuários que usarem o sistema logados. Ressalta-se que não há sincronização entre as bases de dados.

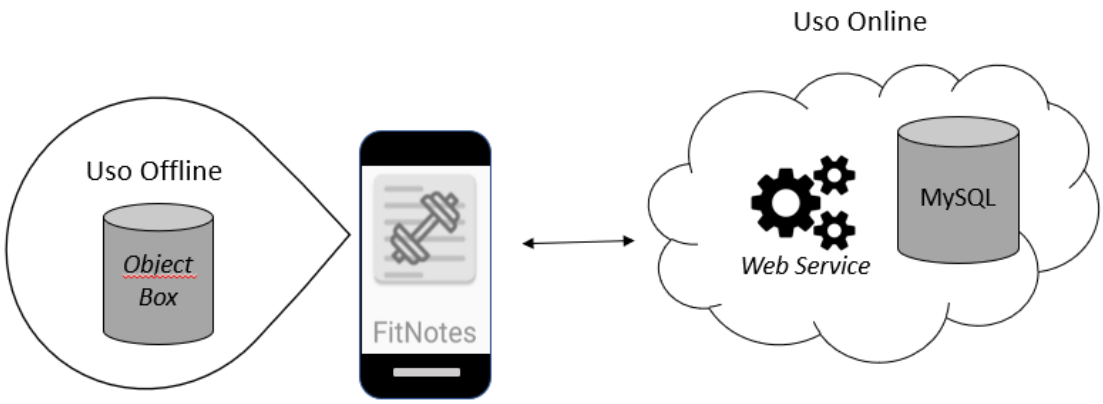

Figura 5. Arquitetura do FitDolphin

5.3. Entidades do banco de dados

1. *exercise*: é a tabela principal do banco de dados, em que são armazenados dados de exercícios cadastrados pelo usuário.
2. *routine*: são armazenados os dados referente as rotinas do usuário.
3. *feedback*: são armazenados os *feedbacks* dos usuários, para posterior avaliação pelo time técnico.
4. *userTable*: são armazenados os dados de usuário nesta tabela. É utilizada para que seja possível identificar os usuários de cada rotina de maneira única.
5. *userHistoric*: é armazenado todo o histórico do usuário através do tempo (atributo data e hora). Baseado nessas informações serão montados gráficos (inclusive de desempenho do usuário) e relatórios que fazem parte da interface da aplicação.
6. *admin*: é utilizada para que os usuários que administram o banco de dados tenham dados de acesso.

5.4. Definição de interface

A interface foi definida com base no *Material Design* da google [Google 2018c], foram montados *mock-ups* em papel para que o autor pudesse identificar as interfaces não intuitivas do aplicativo (lista de exercícios, por exemplo) e a paleta de cores.

5.5. Desenvolvimento do aplicativo e *webservice*

Após a fase de elucidação de requisitos e definição do banco de dados foi feito o planejamento dos casos de uso em incrementos para serem implementados. A Tabela 1 apresenta os casos de uso planejados em cada incremento (*Sprints*).

Tabela 1. Incrementos x Casos de Uso

Incrementos	Casos de Uso
***Sprint* 1**	Manter Conta do Usuário
***Sprint* 1**	Manter Objetivo
***Sprint* 2**	Manter Rotinas
***Sprint* 3**	Manter Exercícios
***Sprint* 4**	Iniciar Rotinas de Exercícios

Todos os incrementos seguiram o mesmo processo de desenvolvimento em duas etapas, primeiramente foi criado o *webservice* e depois sua *interface*.

O *webservice* inclui os métodos para fazer o cadastramento, atualização e exclusão dos dados e um método que verifica se os dados enviados estão cadastrados ou não no banco de dados. Todos os métodos recebem os dados em formato *JSON* e retornam o resultado da requisição no mesmo formato.

Os *JSON* de resposta consistem em dois objetos, um *status* que pode assumir o valor *error* ou *success*, e um *responseBody* que pode conter tanto uma lista de objetos quando um objeto de erro.

```
{
    "statusCode":"error",
    "responseBody":
        "{
            "error":"Rotina nao possui exercicios.",
            "errorId":1
        }"
}
```


Figura 6. Json representando um erro no na listagem de exercícios.

Um exemplo desse comportamento é encontrado na Figura 6, o *status* com valor *error* representa que houve algum erro no processamento da requisição, neste caso uma descrição mais detalhada do erro, com descrição e id é enviada no objeto *responseBody*.


```
{
   "statusCode":"success",
   "responseBody":[
      {
         "exerciseId":"2",
         "exerciseName":"Esteira",
         "exerciseControl":"1",
         "exerciseRoutine":"2",
         "exerciseSequence":"0",
         "exerciseSeries":"3",
         "exerciseRepetitions":"3",
         "exerciseLoad":"0",
         "exerciseTime":"10",
         "exerciseTimeUnit":"1",
         "exerciseDistance":"0",
         "exerciseDistanceUnit":"0",
         "exerciseObservations":"corrida de 9km\\\/h",
         "exerciseLastExecution":null,
         "archived":"0",
         "deleted":"0"
      }
   ]
}
```


Figura 7. Json representando uma listagem de exercícios.

Na Figura 7 o *status* com valor *success* representa que a requisição foi bem sucedida e uma lista dos objetos de resposta é enviada no objeto *responseBody*.

O aplicativo envia requisições ao *webservice* dependendo da opção selecionada pelo usuário. A tela de *Login* do aplicativo permite a entrada de um usuário cadastrado ou como "visitante", sem necessidade de cadastro e autenticação. Pela opção "visitante"os dados são armazenados localmente e caso realize o *login* é utilizado os dados do *webservice*. A Figura 8a apresenta a tela de acesso ao sistema.

Todo o processo de armazenamento local ou remoto são transparentes ao usuário, sem mudança de interface por estar *offline* ou *online* no sistema.

5.5.1. *Sprint 1: Registro de usuário e manter objetivo*

Na *Sprint 1* foram desenvolvidos os casos de uso Manter Registro de Usuário e Manter Objetivo. O cadastro de usuário e de objetivo são unificados em uma mesma tela, é a primeira que o usuário utiliza quando se cadastra no aplicativo, nela são pedidos os dados de nome, email, objetivo e senha para acesso, a Figura 8b mostra a tela de registro do aplicativo.

Quando o usuário se registra, os dados são enviados ao *webservice* e o usuário é direcionado para a tela de login, caso ocorra um erro no cadastro é mostrada uma mensagem e o usuário é convidado a corrigi-los.

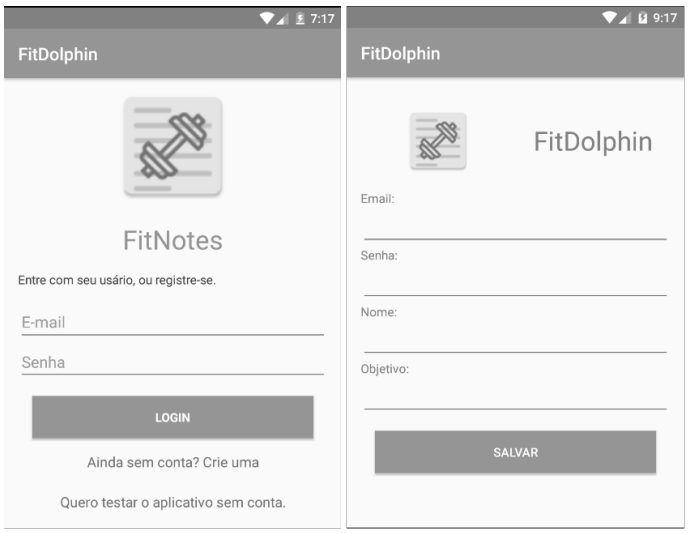

Figura 8. Tela de *Login* e Cadastro de Usuário do FitDolphin

5.5.2. *Sprint* 2: Manter rotinas

Na *Sprint 2* foi desenvolvido o caso de uso Manter Rotinas. A Figura 9a mostra o resultado do Cadastro de Rotinas, é possível criar uma rotina e selecionar quais dias da semana irá realizar essa rotina.

A listagem de rotinas está representada na Figura 9b, que mostra todas as rotinas de um determinado usuário, na mesma interface são mostrados os dias da semana em que a rotina deve ser realizada.

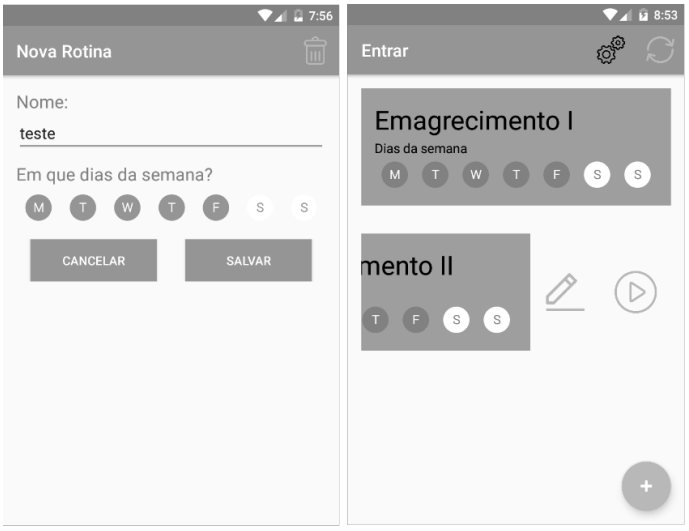

Figura 9. Tela de Cadastro de Rotinas

5.5.3. *Sprint* 3: Manter exercícios

Na *Sprint 3* foi desenvolvido o caso de uso Manter exercícios. A Figura 10 mostra o resultado do Cadastro de exercícios.

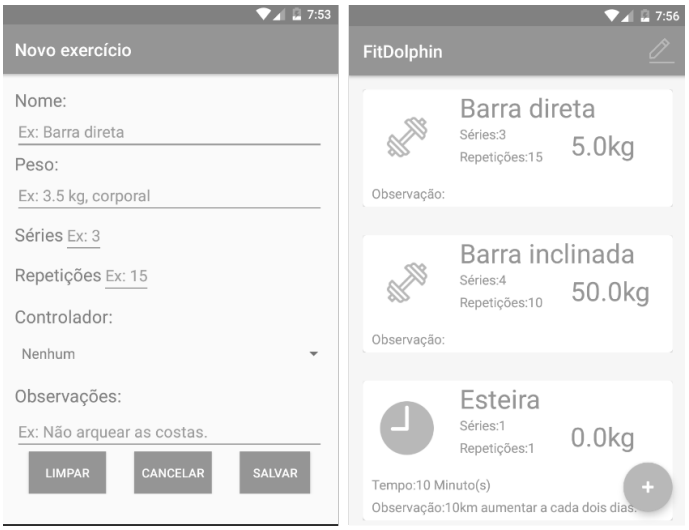

Figura 10. Tela de Cadastro de Exercícios

5.5.4. *Sprint* 4: Execução de exercícios

O *Sprint* 4 seguiu com a implementação para que o usuário pudesse realizar a execução dos exercícios (Figura 11a), o usuário seleciona se cada exercício foi executado ou não e após a realização da rotina completa é exibida uma tela com informações sobre a porcentagem dos exercícios cumpridos (Figura 11b).

Figura 11. Tela de Execução de Exercícios

6. Conclusão

Neste artigo foi demonstrado o processo de desenvolvimento de um aplicativo para o controle e compartilhamento dos dados de atividades físicas de um usuário, o aplicativo foi desenvolvido para a plataforma Android, com uso de *webservice* implantado em PHP.

O aplicativo foi implementado conforme a expectativa e requisitos apresentados neste artigo já que é possível para o usuário final realizar o cadastro e monitoração das atividades físicas realizadas. Foi feita uma abordagem de exercício genérico, que facilita para o usuário escolher a melhor forma de controlar os exercícios a realizá-los, além de ser uma ótima ferramenta para o lembrete de quais exercícios o usuário precisa realizar em determinado dia.

Um maneira de evolução para o aplicativo proposto seria uma possível integração com relógios de atividade, que são ferramentas para facilitar a criação de um histórico de atividades diárias e não apenas dos exercícios realizados na academia. Outro ponto seria o desenvolvimento da aplicação para multiplataformas, assim atingindo um número maior de usuários uma versão *web* poderia levar a experiência de controle para outros dispositivos, como computadores. O compartilhamento das atividades se mostra um passo interessante, já que o *feedback* social, se mostra uma importante ferramenta para a continuidade dos exercícios, poderia também ser implementado um caminho para a sincronização dos dados locais e do *webservice* do aplicativo, assim permitindo que o usuário utilize enquanto estiver sem conexão a rede e ainda assim integrar esses dados mais tarde com a sua conta.

Os conhecimentos em desenvolvimento de uma aplicação móvel são um importante diferencial para o mercado de trabalho atual, dado que a migração para o *mobile* está acontecendo com rapidez e nem sempre os desenvolvedores saem do meio acadêmico preparados para o desenvolvimento de aplicações ricas em recursos e confiáveis para o usuário final.

Se mostra necessário o uso consciente dos recursos disponibilizados pela tecnolo-

gia para a realização e facilitação do dia a dia, dado que as atividades físicas são importantes instrumentos de saúde e auto-estima, e esse é o intuito do aplicativo desenvolvido neste artigo. O aplicativo não substitui em nenhum momento o suporte e monitoramento dos profissionais de saúde para os usuários.

7. Referências

Referências

Allen, J. L. A. S. R. and the Slim Framework Team (2018). Slim framework. https://www.slimframework.com/. [Online; Acessado em 15 de Maio de 2018].

Booth FW, Roberts CK, L. M. (2012). Lack of exercise is a major cause of chronic diseases.

Corporation, O. (2017). My sql. https://www.mysql.com/. [Online; Acessado em 15 de novembro de 2017].

Corporation, O. (2018a). Annotations. https://docs.oracle.com/javase/1.5.0/docs/guide/language/annotations.html. [Online; Acessado em 20 de junho de 2018].

Corporation, O. (2018b). Java. https://java.com/en/. [Online; Acessado em 19 de novembro de 2017].

de Ginástica S/A, S. R. A. (2018). Smart fit coach. https://play.google.com/store/apps/details?id=com.eokoe.smartfitcoach. [Online; Acessado em 26 de Setembro de 2018].

Donnelly, J. E. e. a. (2013). Aerobic exercise alone results in clinically significant weight loss for men and women: Midwest exercise trial-2.

Foundation., T. A. S. (2018). Apache. https://httpd.apache.org/. [Online; Acessado em 15 de Maio de 2018].

Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA, USA.

Genymobile (2018). Genymotion desktop. https://www.genymotion.com/desktop/. [Online; Acessado em 08 de Agosto de 2018].

Google (2017). Android studio. https://developer.android.com/studio/. [Online; Acessado em 15 de Maio de 2018].

Google (2018a). Android. https://www.android.com/. [Online; Acessado em 15 de Maio de 2018].

Google (2018b). Google fit. https://www.google.com/fit/. [Online; Acessado em 26 de Setembro de 2018].

Google (2018c). Material design. https://material.io/design/. [Online; Acessado em 15 de Maio de 2018].

Group, T. P. (2018). Php. http://www.php.net/. [Online; Acessado em 15 de Maio de 2018].

Haitao, H. (2011). Research on the relationship between physical exercise, physical health and physical self-confidence of university students. In *2011 International Conference on Future Computer Science and Education*, pages 485–488.

HEUSER, C. A. (1998). *Projeto de Banco de Dados*. Ed. Sagra, quarta edição.

Json.org (2018). Json. http://www.json.org/. [Online; Acessado em 08 de Agosto de 2018].

Len Bass, Paul Clements, R. K. (2003). *Software Architecture in Practice, Second Edition*. Addison Wesley.

Limited, O. (2018a). Objectbox. http://objectbox.io/. [Online; Acessado em 15 de Maio de 2018].

Limited, O. (2018b). Objectbox. http://objectbox.io/documentation/introduction/. [Online; Acessado em 15 de Maio de 2018].

Ltd., C. (2018). Ubuntu desktop for developers. https://www.ubuntu.com/desktop/developers. [Online; Acessado em 08 de Agosto de 2018].

Ltd, S. H. P. (2018). sublime text. https://www.sublimetext.com/. [Online; Acessado em 08 de Agosto de 2018].

Munassar, N. M. A. and Govardhan, A. (2010). A comparison between five models of software engineering.

MyFitnessPal, I. (2018). Myfitnesspal. https://www.myfitnesspal.com/. [Online; Acessado em 26 de Setembro de 2018].

PREECE, J. e. a. (2013). Bookman, terceira edição.

Rocha, G. (2017a). Em dez anos, obesidade cresce 60% no brasil e colabora para maior prevalência de hipertensão e diabetes. http://portalsaude.saude.gov.br/index.php/cidadao/principal/agencia-saude/28108-em-dez-anos-obesidade-cresce-60-no-brasil-e-colabora-para-maior [Online; Acessado em 19 de outubro de 2017].

Rocha, G. (2017b). Em evento internacional, brasil assume metas para frear o crescimento da obesidade. http://portalsaude.saude.gov.br/index.php/cidadao/principal/agencia-saude/27804-em-evento-internacional-brasil-assume-metas-para-frear-o-cresci [Online; Acessado em 19 de outubro de 2017].

Saifi, R. (2017). The 2017 mobile app market: Statistics, trends, and analysis. https://www.business2community.com/mobile-apps/2017-mobile-app-market-statistics-trends-analysis-01750346. [Online; Acessado em 19 de outubro de 2017].

Sales, R. (2016). Acesso à internet cresce no país puxada por smartphones, diz ibge. http://www.valor.com.br/empresas/4815696/acesso-internet-cresce-no-pais-puxada-por-smartphones-diz-ibge. [Online; Acessado em 19 de outubro de 2017].

SOMMERVILLE, I. (2011). *Engenharia de Software*. Pearson, nona edição.

Um, S.-H. (2004). The effect of a regular exercise on mental health on aged people. In *Proceedings. The 8th Russian-Korean International Symposium on Science and Technology, 2004. KORUS 2004.*, volume 3, pages 284–286 vol. 3.

Zhao, H. and Doshi, P. (2009). Towards automated restful web service composition. In *2009 IEEE International Conference on Web Services*, pages 189–196.