UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUILHERME WALTRICKE FREITAS

UTILIZAÇÃO DA LINGUAGEM SWIFT EM DISPOSITIVOS MÓVEIS BASEADOS NO SISTEMA OPERACIONAL IOS PARA AVALIAÇÃO DA COMPOSIÇÃO CORPORAL

CRICÍUMA

2014

GUILHERME WALTRICKE FREITAS

UTILIZAÇÃO DA LINGUAGEM SWIFT EM DISPOSITIVOS MÓVEIS BASEADOS NO SISTEMA OPERACIONAL IOS PARA AVALIAÇÃO DA COMPOSIÇÃO CORPORAL

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. MSc. Luciano Antunes

Co-orientador: Prof. Dr. Joni Marcio Farias

CRICIÚMA

2014

GUILHERME WALTRICKE FREITAS

A UTILIZAÇÃO DA LINGUAGEM SWIFT EM DISPOSITIVOS MÓVEIS BASEADOS NO SISTEMA OPERACIONAL IOS PARA AVALIAÇÃO DA COMPOSIÇÃO CORPORAL

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Dispositivos Móveis.

Criciúma, 28 de novembro de 2014.

BANCA EXAMINADORA

Prof. MSc. Luciano Antunes - (UNESC) - Orientador

Prof. Dr. Joni Marcio Farias - (UNESC) – Co-orientador

Profa. MSc. Ana Claudia Garcia Barbosa - (UNESC)

Prof. MSc. Gustavo Bisognin - (UNESC)

À minha família, que em todos os momentos me apoiou nesta longa caminhada com muita paciência e dedicação sem cessar.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me iluminado neste longo caminho que com toda sua sabedoria me deu saúde e forças para continuar e superar todos os momentos difíceis que passei.

A minha família que me apoiou ao longo desta jornada, que sempre acreditou no meu potencial e acreditou em min em todos os momentos.

As pessoas que direta ou indiretamente fizeram parte da minha formação, que passaram e sempre deram incentivos positivos.

Ao meu orientador Luciano pela paciência e suporte que teve durante o desenvolvimento deste trabalho.

“A maior invenção do mundo não é a minha tecnologia! É a morte! Pois através dela, o velho sempre dará lugar para o novo!”

Steve Jobs

RESUMO

Dispositivos móveis estão cada vez mais difundidos na cultura popular, impulsionados pelo alto consumo de smartphones e tablets. Com o grande avanço nesse setor é possível ter acesso a dispositivos móveis com alto poder de processamento e armazenamento, proporcionando ao desenvolvedor migrar de aplicações antes restritas a computadores de mesa para a realidade móvel. Uma das fabricantes pioneiras neste nicho de tecnologia é a Apple, seus dispositivos são cobiçados mundialmente por terem um nível de qualidade diferenciado e fornecerem ao desenvolvedor um poder de integração único entre o sistema operacional e seu hardware. Diante disto, o trabalho apresenta o desenvolvimento de um aplicativo para realização de uma avaliação da composição corporal baseada na plataforma móvel iOS utilizando a linguagem de Programação Swift. O aplicativo desenvolvido proporciona ao profissional da educação física realizar uma avaliação física de seus alunos, disponibilizando um ambiente onde é possível cadastrar alunos, avaliações e consultar relatórios sobre as avaliações realizadas. O desenvolvimento realizado foi efetuado utilizando a linguagem de programação Swift de alto nível, disponibilizada pela Apple que é fornecida como alternativa a linguagem Objective-C, juntamente com outras tecnologias como SQLite que foi utilizado como plataforma de armazenamento não volátil e Core Data que manipula a serialização dos dados. Por fim, em testes realizados com auxílio de profissionais da educação física foi possível verificar que são adequados os resultados da aplicação desenvolvida.

Palavras-chave: Sistema Operacional iOS, Swift, Xcode, Composição Corporal, Aplicativos móveis.

ABSTRACT

Mobile devices are increasingly pervasive in popular culture, boosting the high consumption of smartphones and tablets. With the breakthrough in this sector, it is possible to have access to mobile devices with high processing power and storage, providing the developer to migrate to applications previously restricted to desktop computers to mobile reality. One of the pioneer manufacturers in this technology niche is Apple; its devices are coveted worldwide for having differentiated levels of quality and provide the developer with a power integration between the operating system and its hardware. Therefore, the paper presents the development of an application to perform a body composition evaluation based on the iOS mobile platform using the Swift programming language. The developed device provides the physical education professional to perform a physical evaluation of its students by providing an environment where it is possible to register students, evaluations and consult reports on the evaluations. The development was done using the high-level Swift programming language, available from Apple, which is provided as an alternative to Objective-C language, along with other technologies such as SQLite, which was used as nonvolatile storage platform, and Core Data that handles data serialization. Finally, in tests conducted with the help of physical education professional it was verified that the results are suitable from the developed application.

Keywords: Operational System iOS, Swift Language, Xcode, Body Composition, Mobile Applications.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BI	*Business Intelligence*
DC	Densidade Corporal
FDD	*Feature Driven Development*
GNU	*General Public License*
G	Gordura
H	*Headers*
IDE	*Integrated Development Environment*
LLVM	*Low Level Virtual Machine*
MCM	Massa Corporal Magra
MG	Massa Gorda
MVC	*Model View Controller*
OPEN GL	*Open Graphics Library Embedded System*
PME	Pequenas e Médias Empresas
SQL	*Structured Query Language*
SDK	*Standard Development Kit*
UNESC	Universidade do Extremo Sul Catarinense
WWDC	*World Wide Development Conference*
XML	*Extensible Markup Language*
XP	*Extreme Programming*
3G	Terceira Geração
4G	Quarta Geração

SUMÁRIO

1 INTRODUÇÃO

Vive-se em uma época onde tudo evolui rapidamente e o que impulsiona está evolução sem dúvida é a tecnologia. Todas as profissões necessitam de tecnologia, tudo ao nosso redor expira e inspira tecnologia, está inspiração se dá pelo fato de existir muitas áreas que necessitam ainda do emprego das últimas tecnologias lançadas para seu nicho de mercado. Fator de criação de uma solução inovadora para um nicho é de facilitar e auxiliar determinado profissional em sua área, assim criando os propósitos de pesquisadores da computação.

Educação Física é uma área de extrema importância na sociedade, o profissional executando suas funções possuí responsabilidades importantes, pois aplicam seus conhecimentos para promover a saúde em diversos fatores. Pode-se citar dezenas de problemas de saúde decorrentes da falta de atividade física como a obesidade e falta de massa magra. O profissional pode auxiliar no controle da obesidade, na composição corporal, aptidão física dentre outros. Atualmente tem aumentado a procura pela prática regular da atividade física, com orientação do profissional para realizar esta atividade. Diante disso o profissional tende a partir à procura de novas tecnologias que o auxiliam a realizar a melhor prescrição e avaliação de seu aluno.

Telefones inteligentes conhecidos como smartphones e tablets estão em constante crescimento, impulsionado por este crescimento os dispositivos móveis da Apple são uma realidade na vida de grande parte dos consumidores deste tipo de tecnologia.

Com o forte crescimento da plataforma Apple iOS utilizada por dispositivos móveis da Apple, desenvolvedores de aplicações móveis estão cada vez suscetíveis a sanar as necessidades das pessoas com o uso de tecnologias, fornecendo mobilidade. A mobilidade uma necessidade profissional, poder acessar e modificar seus arquivos em qualquer lugar que esteja, e com este projeto mobile pode-se alcançar a esta mobilidade.

Durante vinte anos a linguagem de programação adotada no desenvolvimento móvel para a plataforma Apple foi o Objective-C, pensando no desenvolvedor, afim de facilitar o desenvolvimento para sua plataforma, a linguagem de programação Swift foi lançada em 2014 e foi adotada neste projeto. O Swift

possuí características de uma linguagem de alto nível e traz ao profissional desenvolvedor uma alternativa a linguagem de baixo nível antes utilizada, o Objective-C.

Desta forma o projeto desenvolvido é uma aplicação móvel que utiliza a linguagem de programação Swift focada no sistema operacional Apple iOS, para auxiliar o profissional da educação física a realizar o acompanhamento na prática de atividade física, consumando o emprego de suas pertinências em determinado indivíduo de forma única e de fácil utilização. Com está ferramenta o profissional vai ter maior praticidade, mobilidade e precisão nas avaliações em qualquer lugar.

1.1 OBJETIVO GERAL

Desenvolver um aplicativo utilizando a linguagem de programação Swift para dispositivos móveis baseados em sistema operacional iOS.

1.2 OBJETIVOS ESPECIFÍCOS

Os objetivos específicos dessa pesquisa consistem em:

a) analisar os métodos de avaliação físico-funcional realizados pelos profissionais da educação física;
b) compreender o funcionamento do sistema operacional móvel iOS;
c) compreender o ambiente de desenvolvimento Xcode;
d) desenvolver um aplicativo para avaliação físico-funcional em dispositivos iOS;
e) realizar simulações e testes com o aplicativo.

1.3 JUSTIFICATIVA

Computação móvel é o paradigma no qual usuários podem carregar seus dispositivos computacionais de forma móvel e manter certa conectividade a outras máquinas (COULOURIS; DOLLIMORE; KINDBERG, 2005). Pode-se dizer que tal paradigma é utilizado por dispositivos móveis, esses dispositivos se comparados aos

de computadores comuns possuem recursos limitados, porém tal limitação é compensada pela sua praticidade envolvida com seu tamanho reduzido. A crescente utilização destes dispositivos se deve ao fato de que seus usuários desejam acessar e alterar suas informações em qualquer horário, de qualquer lugar (LEE et al, 2002, tradução nossa).

O crescente mercado de tecnologia e serviços móveis e os novos modelos de negócios envolvendo mobilidade têm evoluído rapidamente na última década (FALCHUK; LOEB, 2007). Parte desse crescimento se deve a existência de situações e/ou locais em que a tecnologia ainda não está acessível, quer seja pelo alto custo ou pela ausência de infraestrutura. Os dispositivos móveis aparecem então para atender estas necessidades, disponibilizando a tecnologia onde ela for necessária (SALOMÃO, 2007).

Com a ascensão da área da tecnologia, o de desenvolvimento para dispositivos móveis, e os aparelhos como iPhone e iPad são cobiçados por usuários, empresas e desenvolvedores, devido a sua grande facilidade de uso e a ótima interatividade com o usuário, além de diversos recursos que permitem ao desenvolvedor criar aplicações ricas e integradas a uma série de serviços (LECHETA, 2012).

Diante do fato de o desenvolvimento ser direcionado para aplicativos móveis da linha Apple iPhone/iPad surge a necessidade da implementação ser para o sistema operacional móvel iOS.

O sistema operacional iOS da Apple se destaca como uma das principais plataformas de desenvolvimento móvel do mercado, uma das características do iOS é sua única e exclusiva integração com o hardware da Apple (LECHETA, 2012).

Para se desenvolver aplicativos para o sistema operacional iOS é necessário se apropriar da ferramenta de desenvolvimento fornecida pela Apple chamada Xcode, sua base de desenvolvimento é feita utilizando padrões *Model View Controller* (MVC) com apoio da linguagem de programação Objective C. Esta linguagem foi baseada em uma outra linguagem chamada SmallTalk-80, e toda sua estrutura foi é herdada da linguagem de programação C (KOCHAN, 2012, tradução nossa).

O desenvolvimento de um aplicativo para dispositivo móvel para avaliação físico-funcional proporcionará ao profissional da Educação Física uma resposta rápida e eficiente no processo de avaliação.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está subdivido em sete capítulos, No capítulo 1 uma breve descrição sobre o trabalho desenvolvido, onde é possível tomar conhecimento do objetivo geral e dos objetivos específicos como também saber qual é a justificativa para a realização deste trabalho.

O segundo capítulo aborda uma visão de como o sistema operacional iOS funciona, explicando desde sua estrutura com as *layers* do sistema até como funciona a integração com suas APIs.

No terceiro capítulo são apresentados as tecnologias de desenvolvimento do aplicativo proposto. É possível neste capítulo tomar conhecimento dos modelos de engenharia de software adotados, IDE de desenvolvimento, linguagens de programação, persistência de dados, padrões de desenvolvimento e padrões de design aplicativos no trabalho.

O quarto capítulo aborda a área da avaliação física, é possível conhecer todo o levantamento teórico a respeito dos procedimentos avaliativos aplicados pelos profissionais da educação física. Métodos de avaliações, medidas antropométricas, protocolos avaliativos podem ser conhecidos neste capítulo.

A descrição de trabalhos correlatos que possuem relacionamento com este trabalho estão no quinto capítulo.

No sexto capítulo o desenvolvimento do aplicativo para avaliação física-funcional pode ser conhecido. A metodologia que foi aplicada no desenvolvimento e seus fatores, foram abordados juntamente com os resultados aqui obtidos no trabalho.

O último capítulo deste trabalho, sete, fornece as conclusões de todo o desenvolvimento realizado, desde conhecimentos obtidos até as dificuldades encontradas no trabalho. Pode-se também ao último parágrafo conhecer as propostas de trabalhos futuros a serem realizados.

2 SISTEMA OPERACIONAL IOS

iOS é o nome dado pela Apple Inc. para seu sistema operacional móvel criado em 2007, este sistema também já foi conhecido pelo nome de iPhone OS porém foi alterado para iOS devido ao seu uso estendido para outros dispositivos da Apple como iPad, iPod Touch e recentemente para a segunda geração da AppleTV. Diferentemente de outras fabricantes de software a Apple não licencia o seu sistema operacional para instalação em dispositivos de outros fabricantes (NEUBURG, 2014, tradução nossa).

A tecnologia aplicada no iOS teve como principal característica sua interface com usuário que permite multi-toques (*multi-touch*) entregando ao usuário uma experiência diferenciada com combinações variadas de gestos em tela. Outra característica que enriquece o sistema operacional móvel da Apple é sua única e exclusiva integração com seu hardware, famosa por desenvolver bons conjuntos de hardware e software, o que garante ao sistema um ótimo desempenho geral (LECHETA, 2012).

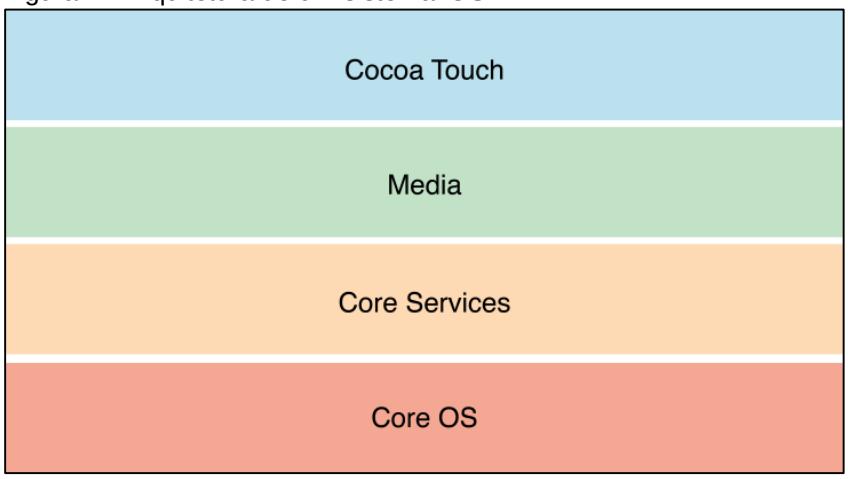
iOS se destaca como uma das principais plataformas de desenvolvimento móvel do mercado, pois é uma das preferidas pelos usuários e desenvolvedores. Está preferência inicia-se por um conjunto de fatores que começa pelo lado do usuário que possui uma loja virtual que ultrapassa um milhão de aplicativos (LECHETA, 2012).

No lado oposto o desenvolvedor que dispõem do Software *Development Kit* (SDK), que trata-se de um poderoso conjunto de ferramentas fornecidas pela fabricante para auxiliar os desenvolvedores no desenvolvimento de aplicativos. O SDK do iOS possui diversas aplicações inclusas, entre elas o Xcode, iOS Simulator e o *Instruments* (APPLE, 2014a, tradução nossa).

2.1 ESTRUTURA

O iOS possui uma arquitetura dividida em quatro camadas que são conhecidas normalmente como *layers*, conforme pode ser observado na figura 1.

Figura 1 - Arquitetura de um sistema iOS



Fonte: Chandra (2013).

2.1.1 Cocoa touch

A camada Cocoa touch é o topo das camadas do sistema, pois é a camada que fornece a maior integração com o usuário. Cocoa touch é uma UI *Framework* para construção de aplicativos que rodam no sistema operacional móvel iOS, Através dessa camada é possível o acesso a várias tecnologias aplicadas no sistema operacional como a manipulação baseada em toques, os recursos de multi-tarefas, notificações em *push* e muitos outros recursos de alto nível com o usuário (APPLE, 2014b, tradução nossa).

2.1.2 Media

Na camada de Media estão as tecnologias aplicadas a funções multimídia. Esta camada fornece os *frameworks* para trabalhar com gráficos, áudio e vídeos no iOS, incluindo a biblioteca gráfica *Open Graphics Library Embedded System* mais conhecida como OpenGL ES 3 (APPLE, 2014b, tradução nossa).

2.1.3 Core services

A Camada Core Services possuí diversos frameworks que as aplicações para o iOS utilizam de forma indireta, pode-se exemplificar o suporte que esta camada fornece na utilização da biblioteca SQLite que permite as aplicações

desenvolvidas realizem o armazenamento de dados com estrutura SQL. O core services também permite o suporte a *Extensible Markup Language* conhecida como XML através do *framework Foundation* (APPLE, 2014b, tradução nossa).

2.1.4 Core OS

A camada Core OS é a de mais baixo nível da hierarquia devido ao fato de todas as outras camadas são construídas a partir dela. Seu uso a partir das aplicações é raro diante de que ela é recomendada apenas em situações que se é necessário lidar diretamente com segurança ou com a comunicação com algum hardware externo.

Alguns dos frameworks importantes dessa camada são: Accelerate Framework (realiza cálculos de álgebra linear, cálculos para processar imagens, entre outros); Core Bluetooth (permite a integração entre os dispositivos utilizando a tecnologia bluetooth) e o External Accessory Framework (realiza a comunicação do dispositivo com acessórios externos que não sejam bluetooth).

Core OS contém a biblioteca chamada LibSystem, utilizada para acessar recursos utilizados pelo kernel do sistema iOS, entre eles estão recursos do sistema de sockets de rede e o sistema de acesso de arquivos (APPLE, 2014b, tradução nossa).

2.2 APIS

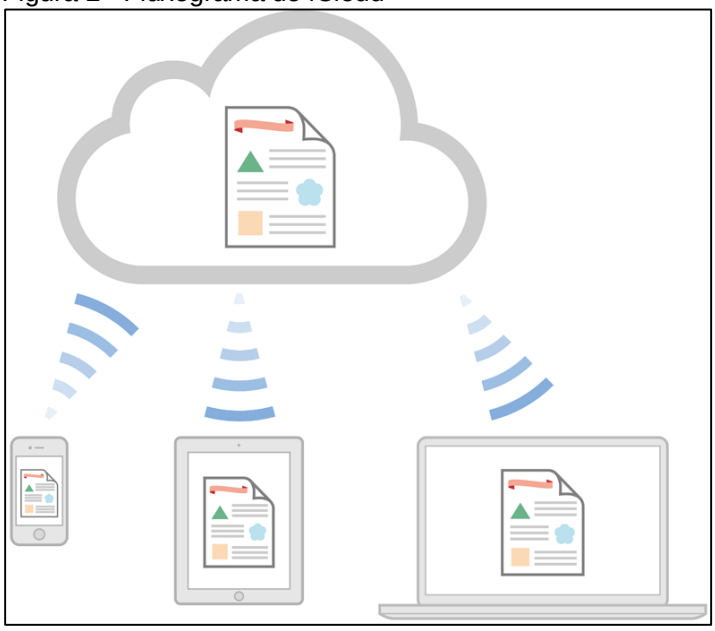
Uma *Application Programming Interface* (API) especifica como um software se comporta, que nada mais é do que um conjunto de *frameworks* necessários para que determinada aplicação funcione e se comporte em conjunto com o seu software (APPLE, 2014, tradução nossa).

Existe uma gama abrangente de APIs disponíveis para utilização em conjunto com o iOS como: *Twitter*, *Facebook*, *Game Center*, *iCloud* dentre outros.

2.2.1 Apple iCloud

iCloud é um serviço grátis fornecido pela Apple, que permite ao usuário acessar e compartilhar seu conteúdo pessoal com todos seus dispositivos como iPhone, iPad, Macbook entre outros através de um serviço de armazenamento online na nuvem. Este serviço pode ser considerado como uma ferramenta de sincronização baseada na nuvem, já que os aplicativos se comunicam entre si, sendo possível o acesso e armazenamento das informações geradas pela mesma aplicação em outros dispositivos conforme ilustra a figura 2. O iCloud fornece aos desenvolvedores de aplicativos mecanismos com transparência de como salvar dados nos servidores da Apple com o mínimo de esforço possível sendo possível salvar todos as informações do aplicativo de forma automática (MARK, 2013, tradução nossa).

Figura 2 - Fluxograma do iCloud



Fonte: Apple (2014c).

3 TECNOLOGIAS DE DESENVOLVIMENTO

Para realização de um desenvolvimento de um software é necessário conhecimento em diversas áreas, um software pode ser subdividido em tarefas distintas, para o projeto proposto essas áreas e conhecimentos são divididas em: engenharia de software, SDK para desenvolvimento, linguagem de desenvolvimento Objective-C, linguagem de desenvolvimento Swift, persistência de dados, padrão de desenvolvimento e padrões de design.

3.1 ENGENHARIA DE SOFTWARE

A existência da aflição para a criação de um software é normal nos dias de hoje, a descoordenação e o despreparo com o planejamento são repetidos em diversos projetos iniciados (PRESSMAN, 1995).

Segundo Sommerville (2003) a engenharia de software é uma disciplina da engenharia que se ocupa de todas as camadas de um software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após a execução do mesmo.

Engenharia de software tem uma abrangência sobre três elementos fundamentais são eles: Métodos, Ferramentas e Procedimentos. Quando os elementos são agrupados e utilizados de forma conjunta em um projeto eles possibilitam ao gerente o controle do processo de desenvolvimento do software a base para criação de um software com alta qualidade, e é claro respeitando a produtividade (SOMMERVILLE, 2003).

3.1.2 A engenharia de software voltada a dispositivos móveis

Um estudo em formato de artigo realizado por Wasserman (2010, tradução nossa) destacou grandes diferenças que deve-se levar em consideração quando se está efetuando um desenvolvimento para aplicações móveis.

Com a explosão de dispositivos móveis, este nicho de desenvolvimento de software também cresceu de forma inimaginável, e a partir disto surge a

necessidade de se aplicar também o conhecimento da engenharia de software para projetos para si desenvolvidos.

Em muitos aspectos o desenvolvimento de aplicações moveis são simulares aos de outras aplicações convencionais, Funções comuns incluindo integração com tipos de hardware diferentes, os tradicionais assuntos ligados a segurança, performance, manutenção e limitações de espaço. De certo modo, aplicações móveis possuem alguns requerimentos adicionais que são menos normalmente encontrados em projetos tradicionais de desenvolvimento, eles são:

a) potencial de integração com outras aplicações: diferente de outros tipos de desenvolvimento, em dispositivos moveis há disponibilidade de aplicações de variadas fontes, que possibilitam uma integração entre eles;
b) aplicações Nativas ou Hibridas: podem invocar do dispositivo e serviços sobre a rede do telefone (Edge/3G/4G) como também via *web browser* (navegador) que vão afetar o uso de data (informações) e *display* do aparelho.
c) famílias de hardware: em desenvolvimento móvel um software deve rodar em variados versões de software e hardware, está preocupação se encaixa entre as que mais levam atenção quando se é iniciado um projeto.
d) *user interface* (UI): Existe a necessidade de preocupação extrema com a experiência de uso de um software móvel, uma aplicação móvel deve compartilhar de elementos que são sugeridos pelas *guidelines* do fabricante do dispositivo móvel.
e) complexibilidade do teste: Enquanto aplicações comuns podem ser testadas diretamente na tradicional maneira via computador, dependendo da aplicação há necessidades de outros tipos de teste, especialmente para softwares que se utilizam da rede, assim há uma grande preocupação com uso de rede do aparelho.
f) consumo de bateria: uma aplicação tem muitos assuntos que podem afetar o uso energia e consequentemente reduzir a vida da bateria e aparelho.

3.2 SDK PARA DESENVOLVIMENTO APPLE XCODE

Conforme mencionado no capítulo anterior, a Apple disponibiliza para os desenvolvedores de aplicativos um framework completo de desenvolvimento contendo ferramentas muito úteis elas são: Xcode, iOS Simulator e Instruments. Todas as ferramentas do SDK de desenvolvimento são gratuitas e exclusivas para o sistema operacional MAC OS X da Apple.

O Xcode é nome dado a uma *Integrated Development Environment* (IDE) que fornece um completo ambiente para o desenvolvimento de aplicações nativas para o iOS. A IDE de desenvolvimento da Apple é abrangente possuindo várias ferramentas como um editor de código fonte, um editor de interface gráficas (interface builder), ferramentas de debug, gerenciador de repositórios para o código fonte dentre outras (LECHETA, 2012).

iOS Simulator é a ferramenta de teste do desenvolvedor, a Apple disponibiliza ela de modo completo onde o desenvolvedor pode realizar os testes do seu aplicativo direto no computador sem a necessidade de um dispositivo externo. O iOS Simulator possui todas as características relevantes dos dispositivos que rodam o iOS que inclui reconhecimento de gestos, rotacionamento do dispositivo nas quatro posições possíveis, dentro outros (LECHETA, 2012).

A ferramenta Instruments é utilizada para analisar o desempenho da aplicação desenvolvida, ela analisa tanto a aplicação rodando no iOS Simulator como também no dispositivo externo. As informações recolhidas por ele são desde utilização de memória, leitura de disco, atividade de escrita, uso da rede e muitos outros dados (LECHETA, 2012).

3.3 LINGUAGEM DE DESENVOLVIMENTO OBJECTIVE C

Dennis Ritchie foi o pioneiro na programação em C nos laboratórios da AT&T em meados de 1970. Porém a linguagem não ganhou muita popularidade e

suporte fora dos portões dos laboratórios da AT&T, Isto porque compiladores em C não são realmente produtivos para aplicações comerciais (KOCHAN, 2012, tradução nossa).

Alguns anos após em meados de 1980, Brad J. Cox criou a linguagem Objective-C, a linguagem foi baseada em uma outra linguagem chamada de SmallTalk-80. Objective-C foi desenhado para o topo das linguagens C, extensões foram adicionadas ao C para criar está nova linguagem que tem disponível Objetos (*objects*) para serem criados e manipulados *(*NEUBURG, 2014*,* tradução nossa)*.*

A empresa NeXT licenciou o Objective-C em 1988 e adicionou ela a *Free Software Foundation's GNU development environment*, tornando-se assim uma licença de uso público e qualquer usuário pode usá-la. Após mais de uma década em 1996 a Apple compra a NeXT e assim possuí os direitos e bibliotecas do Objective-C, tudo isto planejando o lançamento de seu sistema operacional MAC OS X e alguns anos depois o lançamento do primeiro celular inteligente o iPhone com sistema operacional iOS também baseado na sua linguagem própria o Objective-C (PILONE; PILONE, 2014, tradução nossa).

Diante do fato do Objective-C ser de propriedade da Apple ela assim prepara todos seus dispositivos para usarem sua tecnologia, o desenvolvedor deve se apropriar do conhecimento no Objective-C para assim desenvolver seus aplicativos tanto para iOS como para OS X que é o sistema operacional para computadores da Apple.

3.4 LINGUAGEM DE DESENVOLVIMENTO SWIFT

Todos os anos a Apple apresenta seu evento especial focado em novos softwares e tecnologias para desenvolvedores o *Worldwide Developers Conference* (WWDC), em junho de 2014 uma grande novidade foi apresentada por Tim Cook atual CEO da companhia, a nova linguagem de desenvolvimento Swift.

Swift é a nova linguagem de programação para desenvolvimento de aplicativos para iOS e OS X, ela foi desenvolvida pela própria Apple para substituir o Objective-C, a nova linguagem foi resultado de últimas pesquisas em linguagens de programação, que foi combinada com décadas de experiência na criação de aplicações para iOS e OS X. Swift adota uma programação segura e moderna que

torna o desenvolvimento de aplicativos mais fácil, mais flexível, e mais divertido. Objetivo é entregar ao desenvolvedor uma experiência de desenvolvimento única, para isto é empregado técnicas de linguagens de alto nível. Ela é exemplificada como o Objective-C sem o termo C, se tornando assim uma linguagem altamente ligada a objetos com modelos e sintaxes modernas (APPLE INC, 2014a, tradução nossa).

Rápida e poderosa é o conceito sobre Swift isto é baseado que todo código feito com a linguagem é construído de forma rápida usando a alta performance do compilador LLVM, o código é transformado e otimizado em código nativo (APPLE, 2014d, tradução nossa).

Sintaxe é um dos pontos fortes do Swift, suas novidades são a ausência da necessidade cabeçalhos conhecidos como *headers* (.h) na sua criação, o gerenciamento de memória é feito de maneira automática que é diferente do Objective-C e não existe a necessidade também da utilização do ponto-e-vírgula ao final de cada comando. Conforme a figura 4 exibe um código em Objetive-C para concatenar duas *strings*, e na figura 5 exibe o mesmo código em Swift que faz a concatenação de maneira totalmente reduzida, de fácil entendimento e formando em ambos a mensagem "Olá UNESC". Pode-se notar que em Swift é possível omitir qual tipo primitivo é uma variável, sendo assim o compilador entende qual é o tipo e o define da melhor maneira possível (APPLE INC, 2014a, tradução nossa).

Figura 4 - Um código feito com Objective-C

```
NSString *str = @"Olá";
str = [str stringByAppendingString:@" UNESC!"];
```

Fonte: Do autor.

Figura 5 - Um código feito com Swift

```
var str = "Olá"
str += " UNESC"
```

Fonte: Do Autor

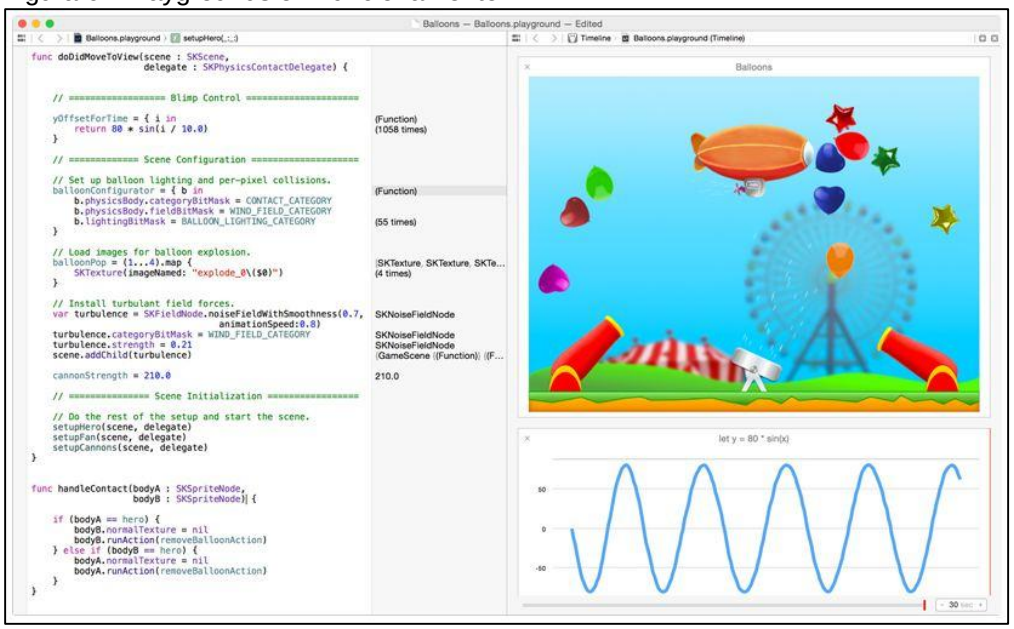
Segundo a Apple o Swift foi concebido para atender as modernidades e auxiliar o desenvolvedor nas suas funções, porém ainda existirá o suporte ao Objective-C em sua IDE, o Xcode. Haverá possibilidade por parte do desenvolvedor

de se utilizar as duas linguagens dependendo do que for necessário em seu projeto, e isto é totalmente permitido com o Xcode6 (APPLE, 2014d, tradução nossa).

3.4.1 Playgrounds

Playground permite escrever códigos com Swift ser uma tarefa fácil. Ao escrever cada linha de código o resultado aparece em tempo real para o desenvolvedor, de forma imediata. Se o código está demorando para rodar, está iniciando uma estrutura de repetição, o desenvolvedor pode visualizar o progresso com o playground. Ele exibe variáveis em gráficos, desenhos de cada etapa que compõem uma *view* e ainda suporta a APIs como SpriteKit que é utilizada na criação de jogos de forma rápida. Na figura 6 podemos ver um exemplo do funcionando do Playground, onde ao lado esquerdo é o código fonte da aplicação e do lado direito é possível visualizar os acontecimentos do código em tempo real, sem a necessidade de compilar o projeto (APPLE, 2014d, tradução nossa).

Figura 6 - Playgrounds em funcionamento



Fonte: Apple (2014e).

3.5 PERSISTÊNCIA DE DADOS

Segundo Mark et al (2013, tradução nossa) grande parte das aplicações possuem a necessidade de ler e reter informações geradas por si mesma, nove em cada dez aplicações tem salvo algum tipo de informação de forma persistente, armazenamento não volátil sobrevive a um *restart* do sistema e por isso se torna de fundamental a sua importância. Está realidade é vivenciada por grande parte dos usuários, seguindo o fato de que toda vez que a aplicação é lançada, ela aparece exatamente da mesma forma que foi encerrada na última utilização.

Uma quantidade expressiva de diferentes mecanismos estão disponíveis para persistir dados em um dispositivo iOS, os mais comum usados são: *Property lists*, *Object archives* (arquivamento) e o SQLite (LECHETA, 2012).

3.5.1 Property lists

Property lists são baseados em uma hierarquia simples de dados. Os itens com dados em uma property list são limitados a alguns tipos de dados, Apenas alguns tipos primitivos de dados e outros containers. Tipos primitivos aceitos são *strings, numbers, binary data, dates*, e *boolean*. Já os containers são arrays e dictionaries que são os principais coleções de dados, identificados por valores e por chaves respectivamente, lista completa de tipos está ilustrada na figura 7. Os containers podem conter outros containers como também tipos de dados primitivos. Portanto pode-se ter um *array* de *dictionaries* ou um dictionary de arrays como também de tipos primitivos. Um objeto chamado *root* em uma property list é o topo da hierarquia, em muitos casos é um array ou um dictionary. De certo modo um objeto root em uma property list não tem que possuir um array ou dictionary, este objeto pode conter apenas uma *string*, *data*, ou outros tipos de dados primitivos (APPLE, 2010, tradução nossa).

Figura 7 - Tipos primitivos de dados para uma property list

Abstract type	XML element	Cocoa class	Core Foundation type
array	<array>	NSArray	CFArray (CFArrayRef)
dictionary	<dict>	NSDictionary	CFDictionary (CFDictionaryRef)
string	<string>	NSString	CFString (CFStringRef)
data	<data>	NSData	CFData (CFDataRef)
date	<date>	NSDate	CFDate (CFDateRef)
number - integer	<integer>	NSNumber (intValue)	CFNumber (CFNumberRef, integer value)
number - floating point	<real>	NSNumber (floatValue)	CFNumber (CFNumberRef, floating-point value)
Boolean	<true/> or <false/>	NSNumber (boolValue == YES or boolValue == NO)	CFBoolean (CFBooleanRef ; kCFBooleanTrue or kCFBooleanFalse)

Fonte: Apple (2010).

3.5.2 Objective Archives

Objective archives provem de uma técnica que converte objetos e valores em uma arquitetura independente de cadeia de bytes que preserva a identidade do relacionamento entre os objetos e valores. Esta técnica de arquivamento de objetos trabalha facilmente com escrita complexa de objetos em arquivos e então lê eles novamente (MARK et al, 2013, tradução nossa).

Pode ser arquivado Objetos do Objective-C, arrays, structures, e strings. Não se pode se pode arquivar os tipos de objetos que variam ao redor da aplicação como *union*, *void*, *function pointers* e uma longa cadeia de pointers (APPLE, 2012, tradução nossa).

3.5.3 SQLite

SQLite é um banco de dados simples e popular entre dispositivos móveis, é um projeto de código-fonte aberto, É implementado em linguagem C que o torna

um banco de dados rápido com performance poderosa, e justamente devido a estes fatores o fez lhe tornar tão dizimado em dispositivos móveis (LECHETA, 2013).

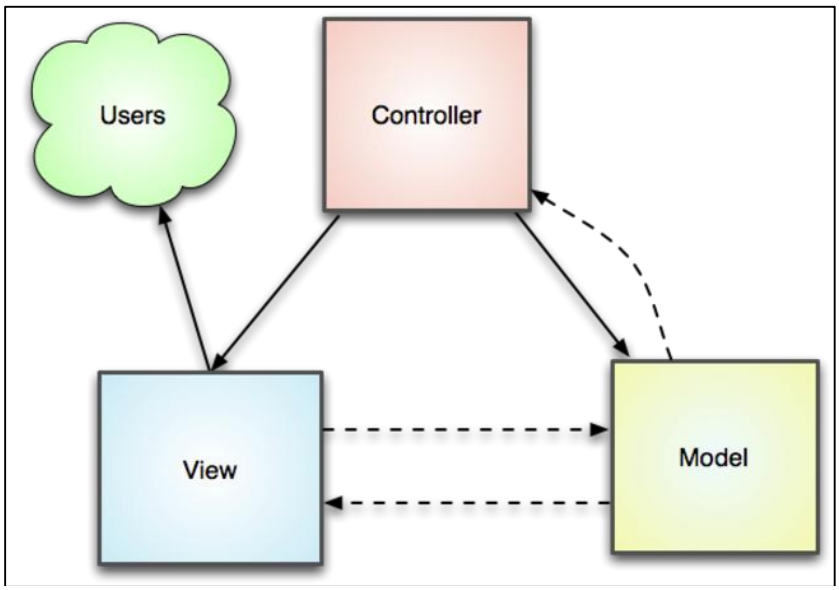
IOS inclui a biblioteca SQLite3 por padrão disponível para ser utilizada em suas aplicações (APPLE, 2014b, tradução nossa). SQLite3 é eficiente em armazenar e recuperar dados como também é capaz de fazer passos complexos de armazenamento de dados, com mais desempenho maior do que é possível utilizando os mesmos passos com objetos como *archives*. Como por exemplo uma aplicação necessita calcular a soma de um campo em particular em todos os objetos na aplicação, utilizando-se de objetos é necessário carregar todos os objetos em memória e calcula-los, se apropriando do conhecimento em SQLite3 é possível fazer isto sem carregar todos os objetos em memória, SQLite3 possuí ferramentas que podem somar estes dados de forma mais rápida e sem ter que carregar todos os objetos assim não gastando memória do dispositivo (MARK et al, 2013, tradução nossa).

Structured Query Language conhecida como SQL é uma linguagem padrão utilizada para interagir em bancos de dados relacionais, está linguagem é utilizada pelo SQLite3.

3.6 MODEL VIEW CONTROLLER (MVC)

O padrão de design de projetos Model-View-Controller é uma técnica que se apropria do conhecimento em orientação a objetos, que auxilia na separação de uma aplicação ou ainda apenas um pedaço da interface da aplicação em três partes: o *model* (modelo), a *view* (interface) e o *controller* (controlador) formando o MVC conforme exemplificado na figura 8 (DOOLEY, 2011, tradução nossa).

Figura 8 - Modelo de funcionamento de um MVC



Fonte: Dooley (2011).

O *model* (modelo) gerencia um ou mais elementos de informação, ele responde a queries sobre estados, e responde a instruções de mudança de estado da aplicação. O modelo sabe como a aplicação supostamente tem que fazer e como fazer, isto é a principal estrutura computacional de uma arquitetura MVC.

A *view* ou comumente chamada de interface é a área retangular que é mostrada na tela para o usuário e é responsável por apresentar todas as informações através de uma combinação de gráficos e textos. A *view* não pode saber nada sobre como a aplicação está se comportando, está parte é de responsabilidade do *controller* (controlador) outra parte do MVC. A interface apenas se encarrega de pegar os dados para serem exibidos do controlador e mostra-los na tela para o usuário (DOOLEY, 2011, tradução nossa).

O *controller* (controlador) é a parte que interpreta as entradas de mouse e teclado do usuário, essas ações efetuadas pelo usuário são enviadas para o *model* (modelo) e ou para a *view* com os efeitos propriamente alterados (DOOLEY, 2011, tradução nossa).

3.7 FRONT END

Pessoas possuem a tendência a não se importar com a experiência de navegação em um aplicativo ao menos quando navegação não corresponde a expectativas delas. O trabalho de criar padrões de design em um aplicativo é

implementar a navegação do mesmo em um caminho que suporta a estrutura e o propósito do aplicativo sem chamar a atenção para si próprio (APPLE, 2014e, tradução nossa).

3.7.1 Framework UIKit

Segundo Apple (2013) quase todos aplicativos iOS usam no mínimo alguns dos componentes de UI (User Interface) fornecidos pela UIKit Framework. Conhecendo os nomes, funcionamento e capacidades desses componentes fornecidos capacita o desenvolvedor a criar um design intuitivo e que auxilia nas decisões a serem tomadas.

Os elementos de UI fornecidos pelo UIKit framework são quatro grandes categorias (APPLE, 2013):

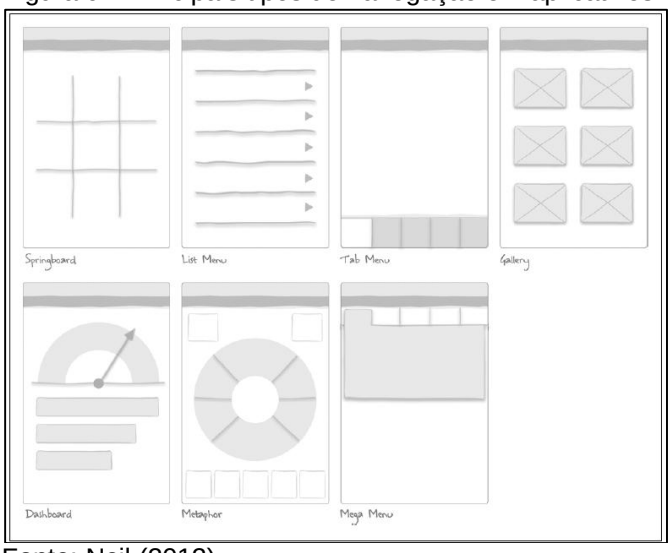
a) *bars*: *bars* contem informação contextual que diz ao usuário onde eles estão e controla a ajuda ao usuário na navegação ou na inicialização de ações;
b) *content views*: *contente views* possuem conteúdo especifico que pode ser ativado a partir de comportamentos como *scrolling* (scroll), *insertion* (inserção), *deletation* (deletar), e *rearrangement* (realinhamento) dos itens;
c) *controls*: *controls* tem o poder de realizar a performance de ações ou mostrar informações;
d) *temporary views*: as *views* temporárias aparecem rapidamente para dar importantes informações ou escolhas adicionais e funcionais.

3.7.2 Padrões de navegação primários

Uma boa navegação como um bom design são invisíveis, uma navegação tem que atrair o sentimento intuitivo e tornar está tarefa fácil de realizar para quaisquer funções. Existem dezenas de opções para navegação dentro de um aplicativo móvel, como foco de boas práticas pode-se separar em sete tipos de

navegação são eles: spring board, list menu, tab menu, gallery, dash board, metaphor, mega menu, como ilustra a figura 9 (NEIL, 2012).

Figura 9 - Principais tipos de navegação em aplicativos móveis



Fonte: Neil (2012).

Springboard é um padrão de adotado por quase todas as plataformas móveis para telas de largadas geralmente chamadas de *launchpad*. O springboard é caracterizado por ser a tela de largada para todos os menus da aplicação, nele é possível ter um conjunto de menus de forma organizada e com rápido acesso, para isto ele adota a formação em grid podendo variar de acordo com a resolução do dispositivo as mais comumente usadas são de 3x3, 2x3, 2x2 e 1x2 (NEIL, 2012).

Listmenu funciona de forma similar ao springboard sua diferença é que cada linha possui um menu único. Existem diversas variações de list menus que geralmente são versões personalizadas que trazem sub grupos (NEIL, 2012).

Tabmenu trabalha de forma diferente em cada sistema operacional móvel, cada fabricante possui um modelo mas seu propósito permanece que é de entregar ao usuário a possibilidade de transição entre telas de forma rápida e de manter ambas com conteúdo (NEIL, 2012).

Gallery é um padrão de navegação onde a tela é apresentada ao usuário dividida em pedaços individuais de conteúdo. O conteúdo é usualmente formados de grids ou *slideshow* que possuem fotos, artigos e até mesmo produtos (NEIL, 2012).

Dashboard fornece ao usuário o poderoso acesso a indicadores, cada métrica pode ser alterada para extrair mais informação. Comumente utilizado por aplicações de cunho financeiro, análise de vendas e marketing (NEIL, 2012).

3.7.3 Boas práticas de design para iOS

A Apple disponibiliza para o desenvolvedor o iOS *Guide User Interface* (GUI) é o guia de desenvolvimento de interfaces para usuários, que auxilia a criar padrões de design para aplicativos que reforçam a importância de padronização e boas práticas. É frequentemente dito por pessoas que gastam mais que um ou dois minutos avaliando um novo aplicativo que é necessário que o mesmo inicie instantaneamente. Quando a inicialização acontece em um curto período exibindo um conteúdo completo imediatamente, é possível aumentar o interesse de novos usuários e entregar a todos uma experiência de uso superior. Outro fator que deve-se evitar quando possível são *splash screens* ou outras maneiras de startup que não sejam automáticas, o melhor caminho é quando um usuário pode iniciar imediatamente o uso do aplicativo (APPLE, 2014e, tradução nossa).

4 AVALIAÇÃO DA COMPOSIÇÃO CORPORAL

A Educação Física faz parte de um grupo com outras áreas da saúde, que possuí um papel extremamente importante no aspecto sócio educativo, criando um estilo de vida saudável e elevando a qualidade de vida, independentemente da idade, sexo, nível socioeconômico ou condição funcional da pessoa (NAHAS, 2006).

Segundo Nahas (2006) uma atitude regular e positiva em relação a atividade física pode ser influenciável em relação quando se tem um melhor conhecimento sobre os benefícios que ela fornece.

Avaliar fatores relacionados a pratica da atividade física tem sido um ponto forte da atenção dos profissionais desta área. Este fato é justificável diante do importante papel do profissional de educação física em avaliar indicadores biológicos, comportamentais e socioculturais que apresentam relação direta e indireta com atividade física que possui complexidade comparável à diversidade e à dificuldade que lhes são inerentes (GUEDES, 2006).

A composição corporal de um indivíduo é um ponto importante para traçar o perfil de saúde e de aptidão para realizar quaisquer exercícios físicos. De um lado existe um indivíduo com falta de gordura corporal, isto apresenta um grande risco a

sua saúde pois o corpo tem a necessidade de certas quantidades de gorduras para realizar suas funções fisiológicas normais. Em outro lado temos também as pessoas que sofrem de obesidade que é um problema sério de saúde, e que resulta na diminuição drástica da expectativa de vida, e esta diminuição ocorre devido aos problemas que a obesidade proporciona ao indivíduo como: doença arterial coronariana, hipertensão, diabete tipo II, doença pulmonar obstrutiva, osteoartrite e alguns tipos de câncer (HEYWARD, 2004).

Segundo Guedes (2006) pode-se conceituar a composição corporal em dividir o peso corporal total em seus diferentes componentes como ossos, músculos, gorduras, água dentre outros e esta divisão resulta em um aglomerado de informações valiosas sobre o comportamento de indicadores ligados ao crescimento físico e aos programas de controle do peso corporal, assim é possível intervir com programas nutricionais e de prática de exercícios físicos.

Profissionais da educação desempenhando suas funções precisam tomar diversas decisões sobre prescrição e orientação da pratica de exercícios físicos, porém isto se torna crítico partindo da ideia de saber o que e como avaliar exige habilidades e conhecimentos específicos cada vez mais complexos. Fatores relacionados à avaliação, procedimentos e recursos que profissionais se utilizam na educação física vem se alterando de forma extremamente rápida influenciados pelo desenvolvimento cientifico e tecnológico observados neste nicho, que por sua vez é terminado por diferentes tendências acadêmicas e profissionais (GUEDES, 2006).

Observa-se, atualmente o crescente interesse pela prática regular da atividade física. O primeiro passo para os iniciantes nessa pratica pode ser a realização da avaliação físico-funcional que é realizada por um profissional da educação física.

A avaliação física é um processo pelo qual utilizando medidas, pode-se subjetiva e objetivamente permitir visualizar a realidade do trabalho que se desenvolve, criando condições para que se entenda o grupo e situa-se um indivíduo dentro deste grupo (ROCHA, 1996).

A avaliação física permitirá avaliar os principais componentes estruturais do corpo humano, cujo enfoque no meio cientifico tem que concentrado na determinação de massa gorda e da massa magra (CARVALHO, 1999).

4.1 MÉTODOS DE AVALIAÇÕES

Para análise da composição corporal podem-se empregar três tipos de técnicas e procedimentos são eles: direto, indireto e duplamente indireto.

4.1.1 Procedimentos diretos

Procedimentos diretos são aqueles em que o avaliador conquista informações no formato in vitro em um laboratório de vários tipos diferentes de tecidos do corpo realizando uma dissecação macroscópica ou até mesmo uma extração lipídica. Este procedimento possuí uma precisão de alto nível em contrapartida o procedimento direto implica em incisões no corpo, o que limita a sua utilização em laboratórios com tecnologia avançada e a utilização de cadáveres de humanos. Portanto a importância dos procedimentos diretos é fundamental para os outros tipos de procedimentos indiretos e duplamente indiretos ligados ao fator de fornecer suporte teórico e técnico (GUEDES, 2006).

4.1.2 Procedimentos indiretos

Dentro dos procedimentos indiretos pode-se separar o mesmo em três grupos de técnicas de medição: bioquímica, de imagem e de densitometria.

Dentre as técnicas bioquímicas temos a hidrometria, a espectrometria de raios gama, a ativação de nêutrons e a excreção de creatinina. Porém os procedimentos bioquímicos possuem a necessidade de uma infraestrutura e custo financeiro dos equipamentos muito elevados. Entre as técnicas de imagem a absortometria radiológica de dupla energia, radiologia convencional, a ultrassonografia, a tomografia computadorizada e a ressonância magnética. Por último a técnica densitométricas que são a pesagem hidrostática que é muito conhecida como pesagem subàquática e a pletismografia (GUEDES, 2006).

4.1.3 Procedimentos duplamente indiretos

Embora os procedimentos indiretos sejam mais precisos e rigorosos eles são de difícil aplicação prática e por isto são geralmente aplicados em investigações científicas e na necessidade da validação dos procedimentos duplamente indiretos. Quando cita-se procedimentos duplamente indiretos não é suposto pelo nome duplamente indireto que eles são mais rigorosos que o procedimento indireto, ao contrário no procedimento duplamente indireto o nível de rigorosidade é menor que o direto e assim de fácil aplicação prática. Pensando no lado da precisão do procedimento o mesmo possuí elevado nível de precisão apesar de menos rigoroso e isto fica explícito quando o aplicador leva em consideração determinados cuidados na aplicação (GUEDES, 2006).

Quando se pensa em métodos utilizados nos procedimentos duplamente indiretos existe o destaque para dois tipos de métodos o de bioimpedância elétrica e a antropometria, estes métodos são os mais comumente usados (HEYWARD, 2004).

4.2 MEDIDAS ANTROPOMETRICAS

Segundo Guedes (2006) pode-se definir antropometria como conjunto de técnicas para se medir dimensões do corpo humano e é utilizada na avaliação do crescimento físico, está avaliação se apropria dos procedimentos indiretos e duplamente indiretos para serem realizados. A antropometria visa padronizar os métodos de avaliação para serem estudados e utilizados mundialmente, estes padrões envolvem técnicas desde aplicação à equipamentos de medição. Dentro da antropometria existem algumas áreas de estudo mas para fim de estudo deste trabalho pode-se destacar em três importantes como alturas e comprimentos, dobras cutâneas e perímetros.

4.2.1 Alturas e comprimentos

Quando surge à necessidade de realizar procedimentos antropométricos um fato importante é a utilização de estaturas, comprimentos e alturas no acompanhamento do crescimento e desenvolvimento do indivíduo. Pode-se iniciar com a medição do peso do indivíduo que deve ser feito com uma balança aferida pelo INMETRO com precisão de 100gr, o avaliado deve ficar em pé, na posição

ereta assim o avaliador obtém em quilos o peso do avaliado. Para realização de alturas de um indivíduo existem três tipos de equipamentos disponíveis como estadiômetro, paquímetro e fita métrica ambos devem ter precisão de 1mm, por convenção a medição deve sempre ser feita ao lado direito (PETROSKI, 2003).

4.2.2 Dobras cutâneas

Dobras cutâneas são as espessuras correspondentes a camada dupla de pele e de tecido subcutâneo destacados em pontos específicos do corpo conforme a figura 10 e também são chamadas de: coxa, axilar média, abdominal, panturrilha, peitoral, tricipital, supra-iliaca e subescapular conforme ilustrado na figura 10. As medidas cutâneas são utilizadas para estimar a composição corporal, para mensuração cutânea é utilizado o instrumento conhecido como adipômetro ilustrado na figura 11 (GUEDES, 2004).

Figura 10 - Locais para retirada de dobras cutâneas

Fonte: Machado (2012).

Figura 11 - Adipômetro

Fonte: Cescorf (2012).

4.2.2 Perímetros

Dentro da antropometria existe a necessidade da mensuração dos perímetros corporais, tal necessidade surge para ser possível a mensuração do crescimento corporal como também de índices de estado nutricional e dos níveis de gordura em conjunto com outras técnicas como dobras cutâneas e ou de forma isolada. Para obter das medidas dos perímetros é necessário se utilizar de uma fita métrica flexível com a precisão de um milímetro. Pode-se dizer que os perímetros dos segmentos corporais são correspondentes ás circunferências, segundo a figura 12 ilustra o local da mensuração mais comum na realização de uma avaliação física (PETROSKI, 2003).

Figura 12 - Locais para retirada de perímetros

Fonte: Avaliação Física Online (2004).

4.3 PROTOCOLOS AVALIATIVOS

Após um conjunto de informações adquiridas de como e o que é necessário para realizar uma avaliação física o que falta é obter o conhecimento dos protocolos mais conhecidos como formulas para se chegar ao objetivo da avaliação, que é possuir dados de um indivíduo como: Densidade Corporal (DC), Massa de Gordura (MG ou G) e Massa Corporal Magra (MCM).

4.3.1 Densidade corporal

As equações para estimativa de DC foram desenvolvidos em dois tipos podem ser em regressão lineares quanto se existe a necessidade da aplicação em uma população especifica ou quadráticos quando se pode generalizados. Existem mais de 100 equações especificas para diferentes populações feitos com regressões lineares, porém estas aplicações se tornam complicadas pois classificar pessoas em uma determinada população pode nem sempre ser de fácil aplicação. Pensando nisso a partir do modelo quadrático, Jackson e seus colaboradores (Jackson e Pollock, 1978; Jackson, Pollock e Ward, 1980) realizaram o desenvolvimento de equações generalizadas que podem ser aplicadas a uma grande população como idades de 18 a 60 anos. Sendo assim a vantagem do uso das equações de Jackson é a possibilidade de usar uma única equação, em vezes de várias para estimar precisamente a porcentagem de DC dos clientes (HEYWARD, 2004).

Baseados na procura de realizar uma avaliação com precisão de 7 dobras cutâneas retiradas das seguintes regiões: peitoral, coxa, abdômen, tríceps, subescapular, supra-iliaca e axilar formando o $\sum 7DC$. As seguintes fórmulas foram desenvolvidas por Jackson e Pollock (1978) para os subgrupo populacionais abaixo:

Homens brancos ou negros, atletas entre 18 a 55 anos de idade:

$$DC = 1{,}112 - [0{,}00043499 * \left(\sum 7DC\right) + \ 0{,}00000055 * \left(\sum 7DC\right)^2 - \ [0{,}0002882(idade)]]$$

Mulheres negras ou brancas entre 18 a 55 anos de idade:

$$DC = 1{,}0970 - [0{,}00046971 * \left(\sum 7DC\right) + 0{,}00000056 * \left(\sum 7DC\right)^2 - [0{,}00012828 * (idade)]]$$

Adaptando também a preferência de outros avaliadores Jackson & Pollock adaptaram suas equações quadráticas para profissionais que preferem pela adoção de apenas três dobras cutâneas porém aqui existe diferenciação dos locais de subtrair as dobras por sexo.

Homens negros ou brancos, atletas entre 18 e 55 anos de idade onde $\sum 3DC$ é o somatório das seguintes dobras cutâneas: tríceps, subescapular e peitoral.

$$DC = 1{,}1125025 - [0{,}0013125 * \left(\sum 3DC\right) + 0{,}0000055 * \left(\sum 3DC\right)^2 - [0{,}0002440 * (idade)]]$$

Mulheres negras ou brancas, entre 18 e 55 anos de idade onde $\sum 3DC$ é o somatório das seguintes dobras cutâneas: tríceps, abdominal e supra-ilíaca.

$$DC = 1{,}089733 - [0{,}0009245 * \left(\sum 3DC\right) + 0{,}0000025 * \left(\sum 3DC\right)^2 - [0{,}0000979 * (idade)]]$$

4.3.2 Separação de informações corporais

Segundo Petroski (2003) o objetivo da avaliação física quanto a resultados deve ser a quantidade de massa magra e gorda corporal em quilos. E para isto foi desenvolvido equações que permitem obter os resultados aguardados, estas formulas se apropriam de passos anteriores efetuados.

Obtenção da massa gorda em porcentagem é feita a partir da formula de Siri (2002).

$$\%MG = [(4{,}95/DC) - 4{,}50] * 100$$

Após obter a porcentagem de massa gorda é possível então calcular em quilos a quantidade de gordura absoluta, que seria a quantidade somente de gordura que o indivíduo possuí no corpo.

$$Gordura\ absoluta = Peso\ corporal * (\%MG/100)$$

Por fim para subtrair a última informação para fins de cálculos de um usuário é a massa magra e ela pode ser adquirira utilizando a seguinte formula:

$$Massa\ Magra\ =\ Peso\ Corporal\ -\ Gordura\ Absoluta$$

Com o total de massa magra do indivíduo pode-se também obter o peso ideal segundo Siri (2002) utilizando a seguintes formulas:

$$Peso\ Ideal\ (Homens) =\ Massa\ Magra/0.85$$
$$Peso\ Ideal\ (Mulheres) =\ Massa\ Magra/0.75$$

Segundo Heyward (2004) realizado todos os cálculos e obter todas as informações necessárias para fim de conhecimento de um indivíduo, uma informação válida é a classificação do mesmo em relação a sua porcentagem de gordura corporal, e isto é, definir qual em qual grupo o indivíduo está incluído e qual a chance do mesmo desenvolver problemas de saúde.

Tabela 1 - Classificação de pessoas baseadas na porcentagem de gordura

Classificação	Homens	Mulheres

Magro	< 8%	< 13%
Ótimo	8-15%	13-23%
Sobrepeso	16-20%	24-27%
Gordo	21-24%	28-33%
Obeso	> 24%	> 33%

Fonte: Lohman (1992).

5 TRABALHOS CORRELATOS

Áreas tecnológicas móveis são muito exploradas nos dias de hoje devido à grande procura por soluções inovadoras neste nicho de aplicação. Abaixo são descritos trabalhos e pesquisas relacionados ao presente trabalho proposto.

5.1 TAGARELA: APLICATIVO PARA COMUNICAÇÃO ALTERNATIVA

Aplicativo para comunicação alternativa para a plataforma iOS, trabalho de conclusão de curso, desenvolvido por Alan Felipe Cardozo Fabeni, na Universidade Regional de Blumenau no Curso de Ciência da Computação (FABENI, 2012).

O aplicativo foi desenvolvido como um trabalho de conclusão de curso para a universidade regional de Blumenau e tem base no apoio e auxilio para profissionais fonoaudiólogos, pacientes e o tutor destes pacientes a se comunicarem com um aplicativo para a plataforma iOS. O aplicativo se utiliza de planos de atividades desenvolvidos por fonoaudiólogos, com as técnicas de Comunicação Alternativa e Aumentativa (CAA) e com tecnologias presentes na plataforma iOS, como iCloud que realiza o armazenamento de informações na nuvem e o Core Data que é um poderoso framework de manipulação de mídias na plataforma iOS (FABENI, 2012).

5.2 DESENVOLVIMENTO DE UMA APLICAÇÃO BUSINESS INTELLIGENCE PARA IOS

Desenvolvimento de um aplicação de *Business Intelligence* (BI) para plataforma iOS realizado por Ana Luísa Meireles Caldeira na universidade de trás-os-montes e alto douro para obtenção do grau de mestre (CALDEIRA, 2006).

Business Intelligence é um de suma importância para as organizações, com ela é possível compreender os dados que as próprias organizações geram em sua atividade o que eleva as estratégias de negócios. Focado em Pequenas e Médias Empresas (PME) o *Business Intelligence* é aplicado neste trabalho. Se apropriando das técnicas de BI o projeto desenvolve um aplicativo para plataforma

iOS onde é possível auxiliar os gestores de PMEs gerir os processos de seus negócios, com ferramentas com tecnologias avançadas do iOS, o armazenamento aplicado neste projeto foi de *Data Warehouse* muito difundido em necessidades de armazenamento não-volátil (CALDEIRA, 2006).

5.3 DISSEMINAÇÃO DO USO DE APLICATIVOS MÓVEIS NA ATENÇÃO Á SAÚDE

Estudo com objetivo de avaliar a disseminação e o impacto do uso de aplicativos móveis para saúde e o bem-estar realizado por Karoline da S. Bonome, Camila C. D. Santo, Cristiana S. Prado, Fernando S. Souza e Ivan T. Pisa para o XIII Congresso Brasileiro em Informática em Saúde (BONOME at al, 2012).

Brasil bate recordes nos últimos anos em vendas de *smartphones* conhecidos como telefones inteligentes e com isso aumenta cada vez o comércio de aplicações móveis como já explanado anteriormente. Diante disso o trabalho realizado por Bonome at al (2012) tem como objetivo avaliar qual é o impacto dos aplicativos móveis para a saúde e bem-estar em usuários por meio da construção de uma lista com os cinquenta aplicativos para saúde e bem-estar em português brasileiros retirados das lojas Google Play para plataforma Android, AppStore para plataforma iOS e Marketplace da plataforma Windows Phone. A realização da verificação de disseminação e popularidade dos aplicativos foi baseado na multiplicação das notas computadas pelo avaliadores e a quantidade de avaliadores de cada aplicativo, retirou-se então a lista com os cinco melhores aplicativos de cada plataforma e assim foram avaliados suas recomendações. Assim foi concluído que existe uma grande aceitação por parte deste público móvel para aplicativos que os iram auxiliar em sua saúde e bem-estar.

5.4 USO DO IOS COMO FERRAMENTA DE INTERAÇÃO DO CLIENTE COM O AMBIENTE DE UM RESTAURANTE: IRESTAURANT

Desenvolvimento de um aplicativo para plataforma iOS para interação do cliente com um restaurante. Trabalho de conclusão de Curso de Ciência da Computação, desenvolvido por Silva at al, na Universidade Anhembi Morumbi (SILVA at al, 2010).

A utilização de dispositivos a partir de clientes afim de interagir com negócios é uma realidade nos dias de hoje. O trabalho proposto por Silva at al (2010) tem como objetivo avaliar os aplicativos disponíveis no nicho de interação do cliente com restaurantes e desenvolver um que atenda a necessidade do mesmo. O estudo por eles permitiu analisar falhas como falta de informação sobre os produtos do estabelecimento, tempo de atendimento alto e falta de iteração com o ambiente. Com os dados necessários iniciou-se o desenvolvimento do aplicativo que é focado na plataforma móvel Apple iOS, com um principal objetivo tornar o processo de pedido mais rápido como também melhorar a qualidade do atendimento e experiência do cliente com o produto, lhe entregando novas iterações com o restaurante.

5.5 DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE REFERÊNCIA SOBRE VACINAÇÃO NO BRASIL

Aplicativo desenvolvido utilizando o framework PhoneGap que auxilia como referência dados sobre vacinação no brasil. Artigo publicado na revista *Journal of Health Informatics* (JHI), desenvolvido por Costa e Oliveira (2011).

É subentendido que o uso da tecnologia aplicada a saúde é fundamental nos dias de hoje, intervenções da computação são indispensáveis neste nicho. Se apropriando da base de dados fornecida pelo Ministério da Saúde do Brasil, do Programa Nacional de Imunização, foi desenvolvido um aplicativo que auxilia o profissionais da saúde na consulta de informações sobre vacinações. PhoneGap é um framework que usa as linguagens de programação HTML, CSS e JavaScript para desenvolver aplicativos híbridos que são compilados em suas determinadas plataformas e podem ser utilizados em diferentes sistemas operacionais móveis como: Android, IOS, e Windows Phone e foi utilizado neste trabalho. Os resultados obtidos com o desenvolvimento foram satisfatórios após questionário respondido por usuários da aplicação (COSTA; OLIVEIRA, 2012).

6 A UTILIZAÇÃO DA LINGUAGEM SWIFT EM DISPOSITIVOS MÓVEIS BASEADOS NO SISTEMA OPERACIONAL IOS PARA AVALIAÇÃO DA COMPOSIÇÃO CORPORAL

Diante do grande avanço tecnológico na computação móvel e das dificuldades encontradas por profissionais da educação física em realizar uma avaliação da composição corporal, foi desenvolvido um aplicativo para dispositivos móveis baseados no sistema operacional iOS da Apple que auxilie o profissional no emprego de suas atividades. Este desenvolvimento foi realizado com os conhecimentos adquiridos no levantamento bibliográfico efetuado neste trabalho.

6.1 METODOLOGIA

Para realização do projeto foi realizado uma reunião com o profissional da educação física e co-orientador Prof. Dr. Joni Marcio Farias com o objetivo de adquirir informações necessárias para o desenvolvimento do aplicado. O co-orientador forneceu os requisitos necessários e o material bibliográfico a ser pesquisado pelo orientado.

Posteriormente com a ideia do aplicativo formada e os requisitos obtidos, foi realizado o levantamento bibliográfico para ampliar o embasamento científico para viabilizar este trabalho.

Em um próximo passo do trabalho foi possível realizar a análise de requisitos com seus diagramas afim de fixar o funcionamento do software e seu comportamento junto ao usuário.

Como o desenvolvimento do projeto foi planejado para o sistema operacional iOS, realizou-se um estudo aprofundado para compreender a forma de desenvolvimento de aplicativos para esta plataforma. Este estudo foi iniciado levantando as tecnologias que seriam utilizadas como: IDE de desenvolvimento Xcode; banco de dados SQLite; experiência de usuário (UX); e o foco principal deste trabalho, que é a linguagem de programação Swift

Por fim com a quantidade de informações obtidas foi possível iniciar a implementação do aplicativo e a realização de testes.

6.1.1 Requisitos do aplicativo

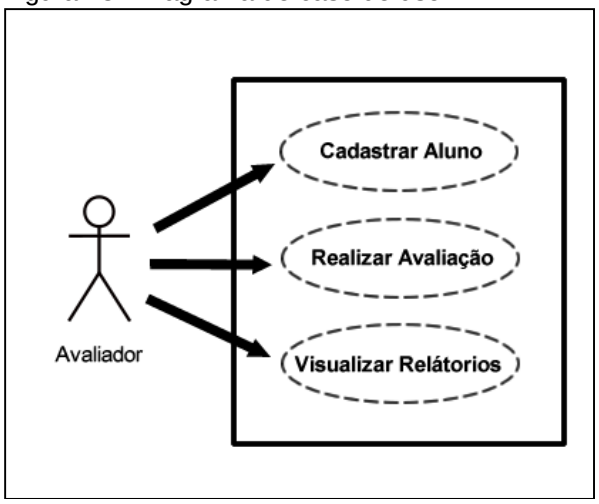
Os requisitos foram levantados durante a pesquisa e classificados em Requisito Funcional (RF) e Requisito Não-Funcional (RNF), os requisitos são listados abaixo:

a) o sistema realizará um cadastro de alunos (RF);
b) o sistema permitirá uma pesquisa de alunos cadastrados(RF);
c) o sistema realizará uma avaliação física-funcional (RF);
d) o sistema permitirá uma pesquisa de avaliações realizadas (RF);
e) será alertado ao avaliador alunos a serem reavaliados (RF);
f) o aplicativo deverá ser implementada na linguagem de programação Swift (RNF);
g) toda informação não volátil será armazenada em banco de dados relacional SQLite (RNF);
h) o aplicativo deverá utilizar o ambiente de desenvolvimento Xcode para ser implementado (RNF).

6.1.2 Especificações

A especificação do trabalho realizado foi desenvolvida utilizando a notação *Unfied Modeling Language* (UML) sendo possível gerar o diagrama de caso de uso como pode ser observado na figura 13, para geração desses diagramas foi utilizado a ferramenta livre *Astah Community*.

Figura 13 - Diagrama de caso de uso



Fonte: Do autor

6.1.2.1 Cadastrar aluno

Este caso de uso permite ao avaliador o registro de um aluno no sistema. O aplicativo coloca à disposição o cadastro de informações como: nome, data de nascimento, sexo, endereço completo, cidade, estado, telefone residencial, telefone celular e e-mail.

6.1.2.2 Realizar avaliação

Este caso de uso disponibiliza ao avaliador realizar a avaliação física de um aluno e registrá-la ao sistema. Se apropriando de uma informação do aluno anteriormente cadastrada, o sistema disponibiliza opções duas opções de avaliação física-funcional: Composição Corporal e Perímetros.

6.1.2.3 Visualizar Relatórios

Este caso de uso possibilita ao avaliador visualizar os alunos registrados e filtrá-los para poder exibir a lista das avaliações que o mesmo possui. Como forma de resultado para relatório são exibidas informações como: data da avaliação, peso, altura, dobras cutâneas, perímetros, gordura corporal, massa magra, objetivo do peso, peso ideal, índice de massa corporal e a classificação segundo Lohman.

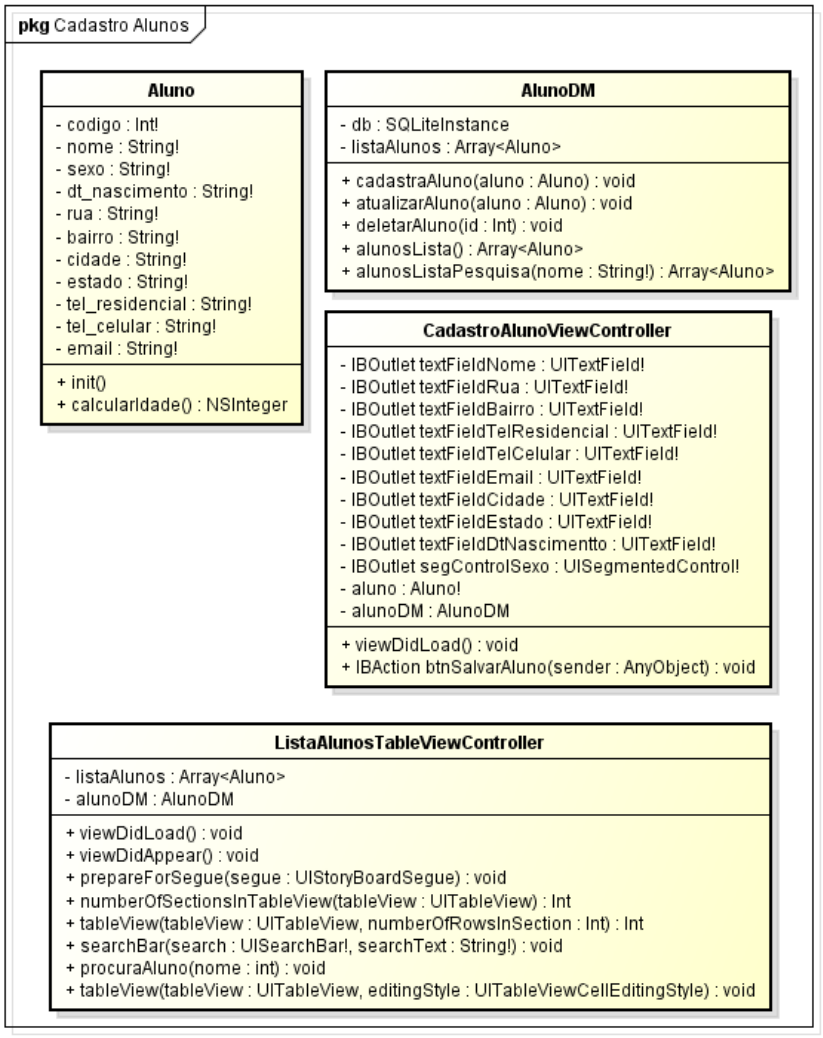
6.1.3 Diagrama de pacotes e classes

Esta seção tem o objetivo de facilitar o entendimento de como as classes estão organizadas. Como convenção de desenvolvedores do mundo inteiro e auxiliador de organização, o projeto é dividido em pacotes (*packages*) e suas dependências (classes).

6.1.3.1 Pacote aluno

As classes que constituem o pacote Aluno conforme ilustrada na figura 14, são responsáveis por lidar com a manipulação de informação de um aluno no aplicativo. Esta manipulação pode ser feita como pesquisa, cadastro e alteração de um aluno.

Figura 14 - Diagrama do pacote cadastro alunos



Fonte: Do autor

A classe *Aluno* é responsável por armazenar a instancia de um aluno em tempo de execução do aplicativo, seu principal objetivo é de auxiliar a manipulação de alunos em todo o aplicativo. Pode ser observado que ela possuí um método construtor chamado de *init()* para ser instanciada e manipulada, juntamente outro

método chamado de *calcularIdade()* que utiliza da variável local *dt_nascimento* para realizar um cálculo e retorna a idade do aluno no formato primitivo *NSInteger.*

A classe *AlunoDM* se apropria da API de manipulação de banco de dados SQLite. Chamada de DM ao seu final com relação a *Data Module* uma convenção escolhida pelo desenvolvedor para denominar classes que tem apenas como objetivo a manipulação de dados do banco de dados. Aqui é manipulada toda informação do banco, então para manipular determinado aluno a classe possuí funções como: *cadastrarAluno* que registra um aluno; *atualizarAluno* que atualiza um aluno já cadastrado no banco; *deletarAluno* realiza a exclusão de um determinado aluno do banco de dados*; alunosLista* que faz a listagem de todos os alunos cadastrados e *alunosListaPesquisa* que cria uma lista de alunos a partir de um determinado nome informado em um campo de texto.

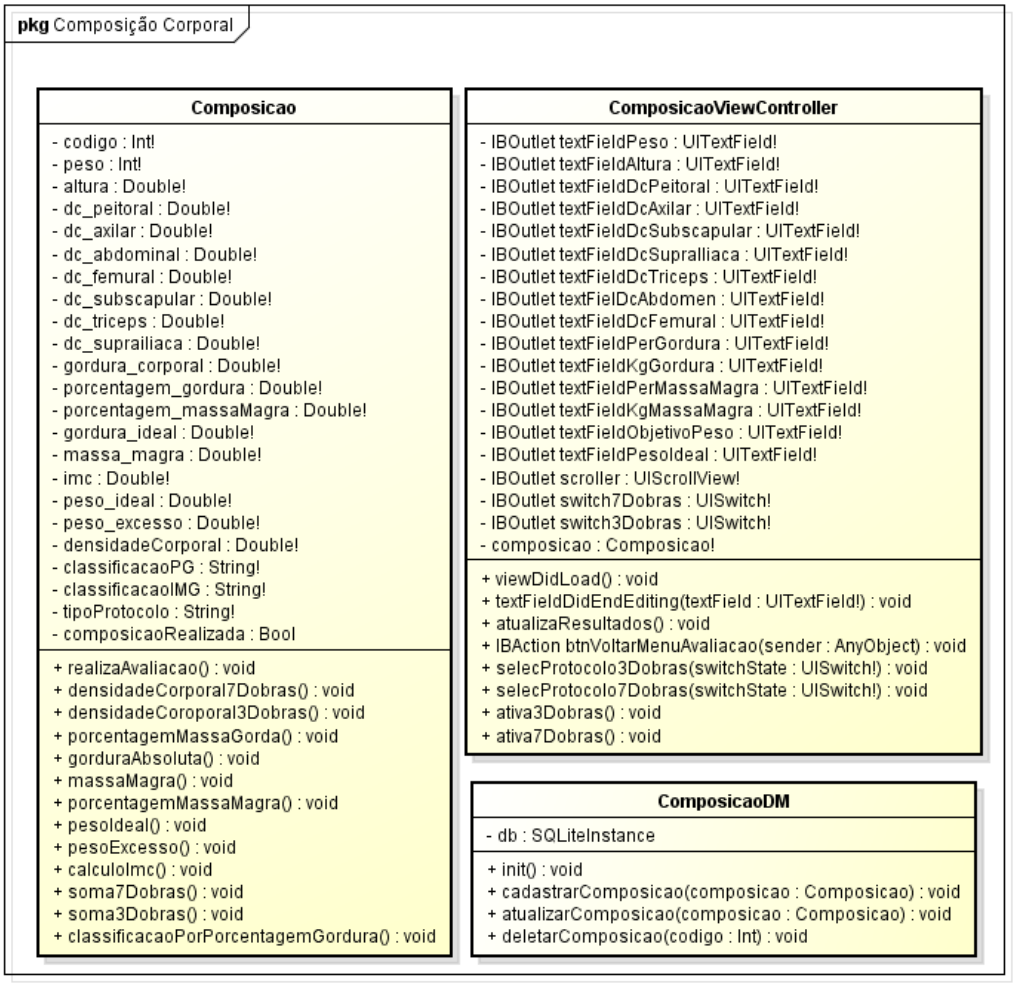
A classe *CadastroAlunoViewController* herda de uma outra classe primitiva do Swift chamada UIViewController como informação implícita no nome UI está classe tem como único objetivo manipular a interface do usuário, fazendo a ligação com as outras classes como Aluno, *AlunoDM*. Como padrão MVC as classes devem ser separadas para manter a integridade do projeto e assim estão feitas no decorrer do projeto. As variáveis são de armazenamento do conteúdo a ser transferido para as classes e manipulação que são chamadas de *IBOutlets*, e temos como função uma ação chamada *IBActionbtnSalvarAluno* que é vinculada com a classe *AlunoDM* para realizar o armazenamento do mesmo no banco de dados.

E a última classe que faz parte deste pacote é chamada de *ListaAlunoTableViewController,* está classe herda características da *UITableViewController* outra classe primitiva do Swift que disponibiliza funções de tabelas e estruturas de listagem de informações para usuário na interface. Aqui são efetuadas a lista de alunos cadastrados no sistema e armazenados em um array. Como o propósito é a listagem de alunos, é possível para o avaliador efetuar uma pesquisa de usuários cadastrados também, essa pesquisa utiliza técnicas de *delegates* que são *links* que permitem ao *controller* saber quando um usuário digita ou manipula algo na *interface* em tempo real, sendo assim funções como a *searchBar()* pode ser invocada e ainda manipular uma pesquisa em tempo real e exibida ao usuário.

6.1.3.2 Pacote composição corporal

As classes que aqui constituem o pacote composição corporal, ilustradas na figura 15, são os responsáveis por manipular as informações relativas a Composição Corporal.

Figura 15 - Diagrama do pacote composição corporal



Fonte: Do autor

A classe *Composicao* é a responsável pela manipulação em tempo de execução de uma instância de uma composição corporal. É possível além de armazenar informações da composição corporal como também se apropriar de funções importantes que geram resultados pertinentes do tipo de avaliação como

cálculos de densidade corporal, porcentagens de massa magra e massa gorda, peso ideal, peso em excesso, IMC e classificação de Lohman.

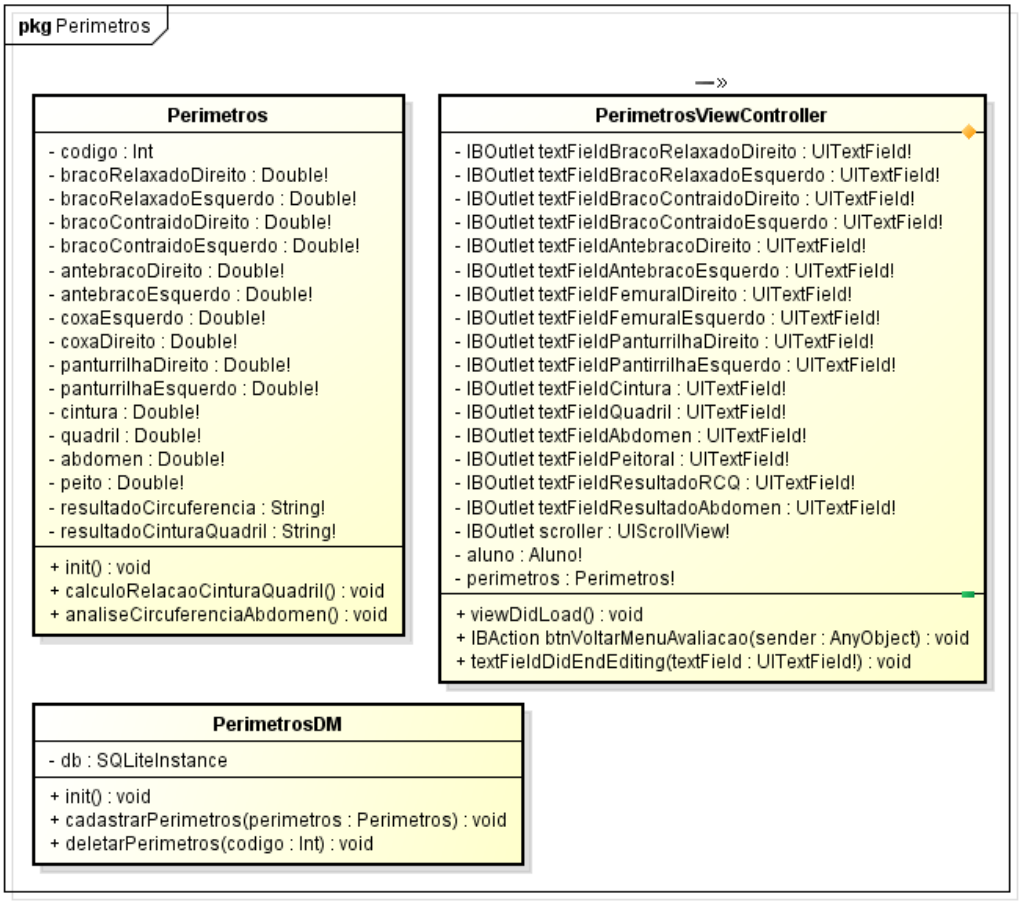
A classe *ComposicaoDM* é classe de manipulação de informação com o banco de dados, está classe funciona como API de comunicação para ler informações, cadastrá-las e até mesmo excluir do banco de dados.

Por fim a última classe do pacote é *ComposicaoViewController*, está classe possui herança também da classe *UIViewController* sendo assim aplicada a fins de manipulação de interface com o usuário.

6.1.3.3 Pacote perímetros

As classes do pacote Perimetros conforme ilustrado na figura 16, são as responsáveis pelo controle dos perimetros da aplicação, com elas é possível manipular os perimetros de uma avaliação física.

Figura 16 - Diagrama do pacote perímetros



Fonte: Do autor

A classe *Perimetros* é a responsável pela manipulação da informação dos perimetros do aplicativo em tempo de execução, ela possuí variáveis de alocação dos dados de cada perímetro, e também funções como *calculoRelacaoCinturaQuadril()* e *analiseCircuferenciaAbdomen()* que realizam os cálculos de risco de problemas cardíacos.

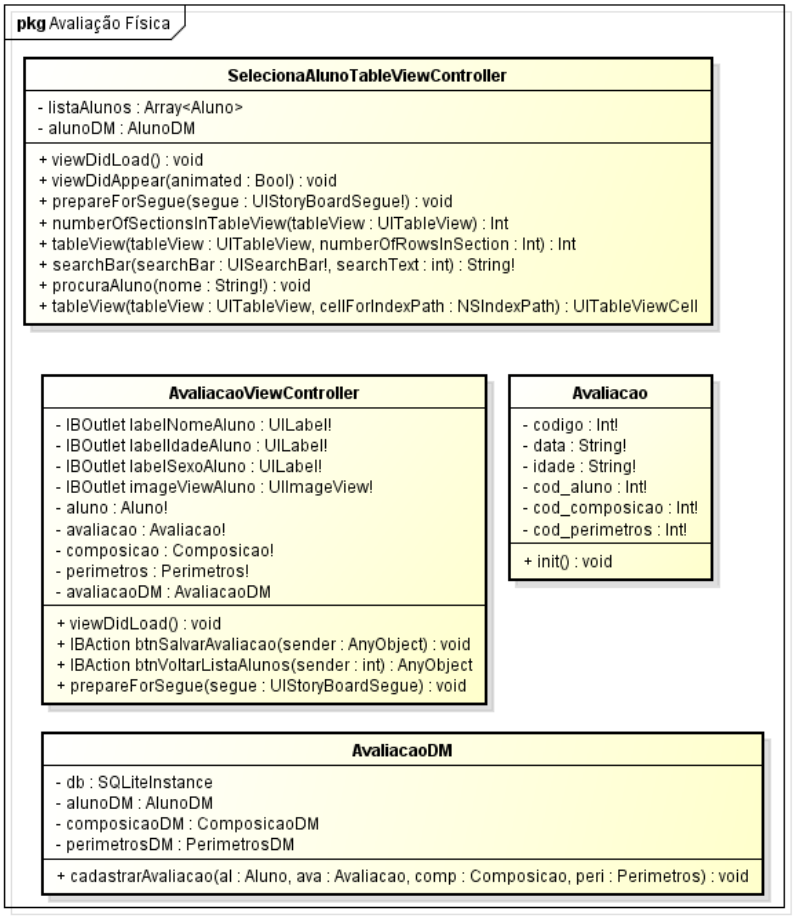
A classe *PerimetrosDM* é uma classe data module como explicado anteriormente que realiza a função de API do aplicativo com a camada Core Data onde fica o banco de dados. Aqui toda informação a ser gravada e excluída de forma não volátil é efetuada.

Para finalizar o pacote a classe *PerimetrosViewController* é uma classe que cuida da parte de interface com usuário e seu eventos, ela é fruto da herança da classe UIViewController que fornece a manipulação e integração com os componentes gráficos da aplicação.

6.1.3.4 Pacote avaliação física

As classes que estruturam este pacote tem como objetivo fornecer base ao aplicativo para efetuar uma avaliação física completa, estas classes estão ilustradas na figura 17. Uma avaliação física é composta de diferentes protocolos avaliativos, aqui foram aplicados: composição corporal e perímetros. O pacote avaliação física agrupa estas informações.

Figura 17 - Diagrama do pacote avaliação física



Fonte: Do autor

Avaliacao é a classe responsável por armazenar em tempo de execução as informações sobre uma avaliação física-funcional. Cada *Avaliacao* instanciada armazena dados como código de uma composição corporal, código de um perímetro, código do aluno avaliado como também dados de data e horário da avaliação.

AvaliacaoDM é a API que foi criada para trabalhar diretamente com informações contidas no banco de dados. Nesta classe toda informação que necessite ser cadastrada no banco de dados deve passar.

SelecionaAlunoTableViewController é uma herança da classe *UITableViewController,* sua principal característica é manipulação de dados com o

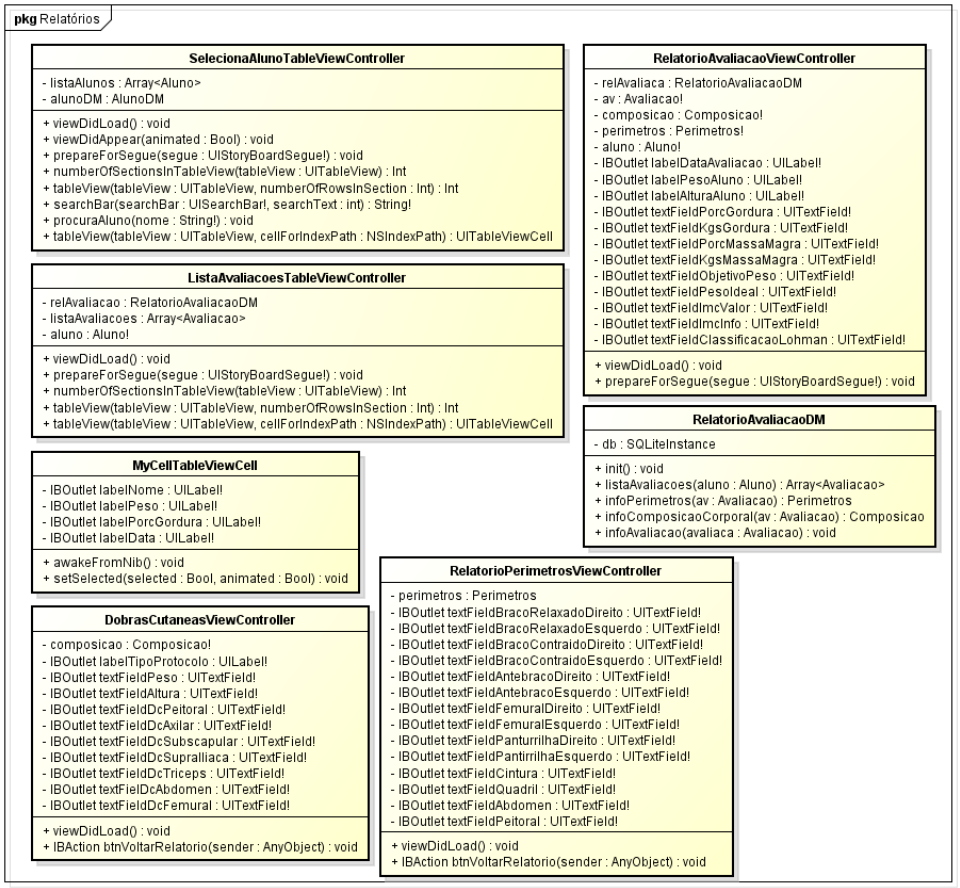
usuário por meio de sua interface. Este tipo de classe manipula dados em forma de tabelas e permite a geração e edição de dados na mesma, para isto ela possui uma lista chamada *listaAlunos* do tipo *Array<Alunos>* que possui todos os alunos cadastrados no banco de dados para exibição na tela. Todas as demais funções tem como objetivo a montagem da interface do aplicativo.

Por fim a classe *AvaliacaoViewController* é a responsável pelo menu principal da avaliação física do aplicativo, ela é uma herança de outra classe com poder visual UIViewController fornecendo ferramentas e objetos para manipular a interface do usuário. Está classe concentra e exibe ao usuário, dados sobre a avaliação a ser realizada, e a partir do componente chamado UIButton com os menus da avaliação é possível chamar a função *prepareForSegue()* que interpreta qual é o destino da avaliação ser instanciada, *PerimetrosViewController()* ou *ComposicaoViewController()* para enfim entregar ao usuário o menu escolhido.

6.1.3.5 Pacote relatórios

O pacote relatórios é o responsável por agrupar todas as classes que auxiliam o aplicativo na exibição de relatórios de avaliações físicas realizadas. Aqui um avaliador pode à partir de um aluno listar todas as avaliações físicas realizadas no indivíduo. A estrutura das classes pode ser observada na figura 18.

Figura 18 - Diagrama do pacote relatórios



Fonte: Do autor

A classe *RelatorioAvaliacaoDM* é utilizada como API de comunicação com o banco de dados, com ela é possível resgatar dados do banco para transmitir para as classes responsáveis pelo escopo de visão da aplicação. As principais funções são *listaAvaliacoes()* onde todas as avaliações de um determinado aluno são listadas e *infoPerimetros()*, *infoComposicao()*, *infoAvaliacao()* que retornam instancias de seus determinados tipos que formam um aglomerado de dados a serem montados e exibidos.

SelecionaAlunoTableViewController é uma classe que herda as características de uma *UITableViewContoller*, seus métodos implementados são para a montagem da tableView que é a tabela com todos os alunos cadastrados. É possível pesquisar um aluno com o componente *UISearchBar* em conjunto com a função implementada *procuraAluno()* e assim com o aluno selecionado dar continuidade no fluxo da aplicação.

A classe *MyCellTableViewCell* herda suas características também de outra classe primitiva do iOS chamada *UITableViewCell*, sua função é ser usada pela sua classe pai *UITableViewController* na criação de uma tabela, por padrão

esta classe já vem implementada dentro da *UITableViewCell* porém neste trabalho surgiu a necessidade de uma célula da tabela ser personalizada com os dados da avaliação, sendo assim está classe foi criada.

ListaAvaliacoesTableViewController possui como já implícito em seu nome a herança da classe de interface com o usuário *UITableViewController* e tem como seu objetivo a listagem de forma visual de todas as avaliações realizadas pelo usuário que foi selecionado na outra view chamada *SelecionaAlunoTableViewController*. Para montagem dos dados customizados na *tableView* foi necessário criar uma *customviewcell* que seria uma célula customizada que foi desenvolvida na classe *MyCellTableView*, assim pode-se empregar uma tabela personalizada e automática ao aplicativo. Com a implementação dos métodos da tabela é possível clicar em uma avaliação e então será feita a exibição dos dados em uma outra *view*.

RelatorioAvaliacaoViewController herda as propriedades da classe de visão *UIViewController*. Sua responsabilidade no pacote é de mostrar ao usuário os resultados de uma avaliação física realizada e escolhida como relatório. Seu armazenamento é quase todo constituído por *IBOutlets* que são utilizadas para exibir dados na interface. Uma função importante a se destacar é o *prepareForSegue()* onde a partir da escolhe de dois *UIButtons* sendo um deles de Composição Corporal e outro de Perímetros decidira onde o usuário deseja ir e instancia a classe para exibição.

RelatorioPerimetrosViewController é resultante da herança também da classe *UIViewController* e seu objetivo é de auxiliar a classe *RelatorioAvaliacaoViewController* a exibir os dados da avaliação para o usuário. A interface criada e exibida para o usuário são com as informações dos perímetros da avaliação.

DobrasCutaneasViewController é utilizada também como apoio a classe *RelatorioAvaliacaoViewController*, possuí a herança de outra classe *UIViewController* pois trabalha no escopo de visão da aplicação. Aqui são exibidas as informações das dobras cutâneas do usuário efetuadas no dia de sua avaliação física.

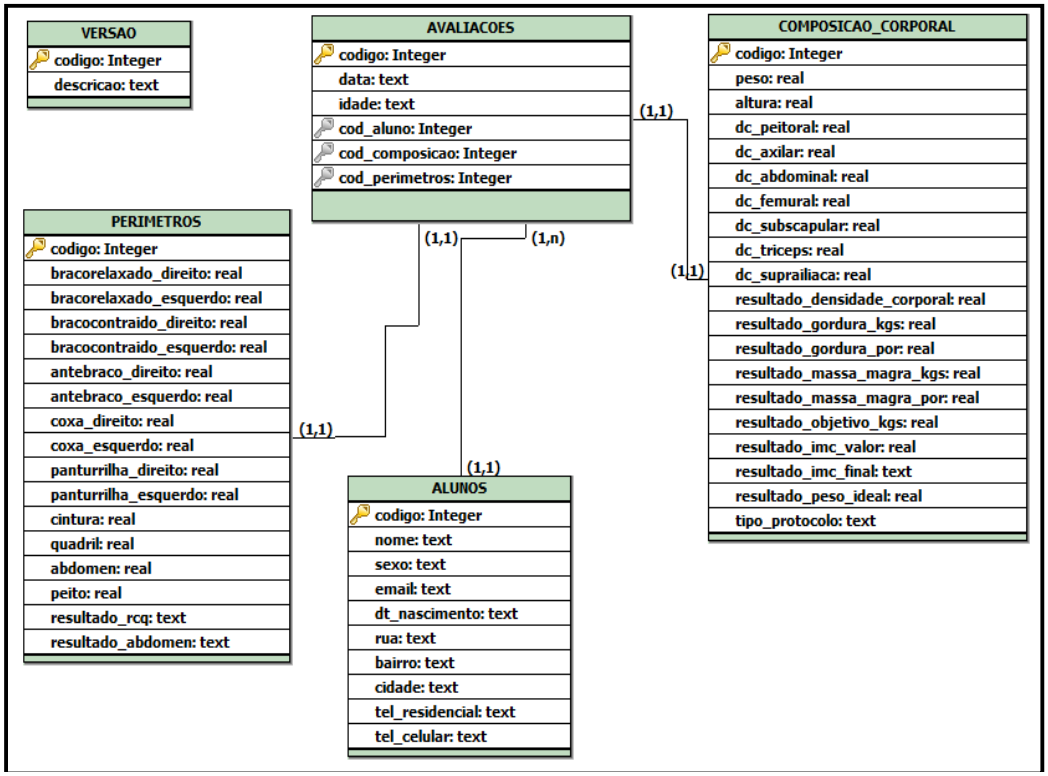
6.1.4 Diagrama de entidades do banco de dados

Procurando realizar a persistência das informações geradas pelo aplicativo de forma não volátil surge a necessidade um banco de dados, e diante desta necessidade foi adotado para realização deste projeto o SQLite.

SQLite trabalha na camada Core Data do iOS e por isto é disponibilizado juntamente com um framework de comunicação entre a aplicação e o banco de dados. A versão escolhida do framework foi a última disponibilizada pela Apple o SQLite 3.0.0.

Após a criação dos diagramas de caso de uso e de pacotes com suas classes é necessário criar a base de dados com as tabelas e suas colunas para o armazenamento destas informações. O modelo lógico pode ser observado na figura 19 e foi gerado a partir da ferramenta brModelo.

Figura 19 - Diagrama de entidades do banco de dados



Fonte: Do autor

A tabela *versao* é responsável por guardar a versão do aplicativo, com esta tabela é possível controlar a versão do aplicativo e efetuar alterações de atualizações de maneira controlada.

Já à tabela *alunos* é utilizada para armazenar todos os alunos cadastrados no aplicativo.

As informações armazenadas na tabela *perimetros* concentra todos os dados dos perímetros de uma avaliação física juntamente com os resultados obtidos da avaliação.

Tabela *composicao_corporal* é utilizada para realizar o armazenamento de todos os dados de dobras cutâneas e resultados da composição corporal de uma avaliação física.

Por fim para armazenar as informações das avaliações temos a tabela *avaliacoes* que é responsável por registrar uma avaliação física realizada, nela é possível registrar o aluno que efetuou a avaliação, a data, código dos perimetros e da composição corporal.

6.1.5 Implementação e funcionamento

Para realizar a implementação do aplicativo foi utilizado a linguagem de programação Swift que trabalha em conjunto com sua IDE de desenvolvimento o Xcode na sua versão 6.0.1 juntamente com o sistema operacional iOS 8. Para testes e simulações foi utilizado o iOS Simulator.

6.1.5.1 Launcher da aplicação

Os aplicativos para iOS possuem por padrão o processo de *launch* que é o processo de carregamento de informações do aplicativo na memória do dispositivo com a preparação do mesmo para uso. Ao decorrer do *launch* é exibido ao usuário uma tela de carregamento do software, chamada *splash screen* que possuí a logo do aplicativo. Porém o importante desta fase funcionamento é a classe padrão chamada *AppDelegate.swift* que é carregada em memória.

AppDelegate realiza a configuração da aplicação, desde quais dispositivos serão suportados pelo aplicação, até mesmo configurações avançadas de comunicação com o Core Data. No aplicativo utilizou-se esta classe para realizar a criação do banco de dados. Ao iniciar para o primeiro uso é feita a verificação de versões a fim de detectar possíveis mudanças e realizá-las ao iniciar o aplicativo. O método *application* da classe *AppDelegate* foi sobrecarregado para criar uma nova instancia de conexão com o banco conforme a figura 20.

Figura 20 - Função principal da classe AppDelegate.swift

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject:
    AnyObject]?) -> Bool {
    // Override point for customization after application launch.

    //inicia conexao SQL;
    let db = SQLiteDB.sharedInstance()
```

Fonte: Do autor

Com a instância de conexão com o banco armazenado em uma constante que pode ser observada com a palavra reservada *let db* (figura 20), os próximos passos são verificar se existe a tabela criada no banco, se a mesma não existir ele cria, conforme pode ser observado na figura 21.

Figura 21 - Criando tabelas no banco de dados

```
// Crias tabelas se as mesmas não existirem no banco (primeiro uso)
// tabela alunos;
db.execute("create table if not exists alunos (codigo integer not null primary key autoincrement,
    nome text, sexo text, email text, dt_nascimento text, rua text, bairro text, cidade text,
    estado text, tel_residencial text, tel_celular text)")

//tabela perimetros;
db.execute("create table if not exists perimetros(codigo integer not null primary key
    autoincrement, bracorelaxado_direito real, bracorelaxado_esquerdo real,
    bracocontraido_direito real, bracocontraido_esquerdo real, antebraco_direito real,
    antebraco_esquerdo real, coxa_direito real, coxa_esquerdo real, panturrilha_direito real,
    panturrilha_esquerdo real, cintura real, quadril real, abdomen real, peito real,
    resultado_rcq text, resultado_abdomen text)")
```

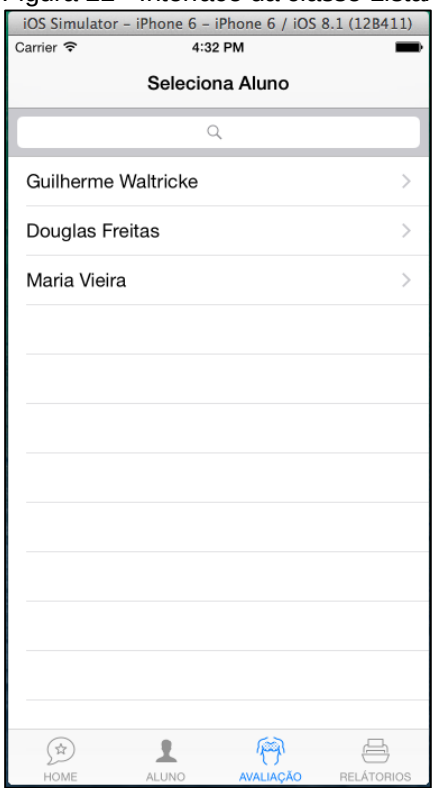
Fonte: Do autor

O processo de criação das tabelas continua o mesmo até verificar todas as tabelas necessárias do aplicativo no banco de dados, assim que toda está verificação é efetuada enfim o usuário é direcionado para a tela principal de uso do aplicativo.

6.1.5.2 Aluno

O menu ALUNO da aplicação é responsável pelo gerenciamento de informações de um aluno no fluxo da aplicação. Em sua primeira tela por padrão o aplicativo exibe a classe *ListaAlunosTableViewController* conforme a figura 22, está tela tem como objetivo listar todos os alunos cadastrados no sistema a partir das funções *viewDidAppear()* que lista todos os alunos do banco de dados e os coloca em uma lista chamada *listaAlunos*. Como pode ser observado o componente *SearchBar* está disponível para ser utilizado, o mesmo com auxílio do método *searchBarSearchButton* busca em tempo real no banco usuários que o usuário deseja. Existindo a necessidade de editar um dos alunos cadastrados, basta clicar sobre o nome do mesmo na tableView e assim irá para a tela de edição.

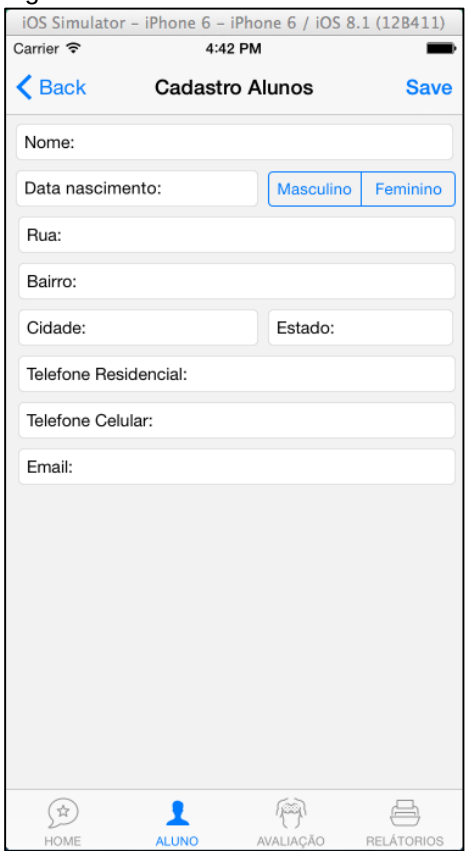
Figura 22 - Interface da classe ListaAlunosTableViewController



Fonte: Do autor

Para prosseguir, o usuário deve clicar sobre o botão Novo disponível acima do menu ou clicar sobre o nome do usuário disponível na *tableView* assim ele será redirecionado para a tela *Cadastro de Alunos* que é formada pela classe *CadastroAlunoViewController* que pode ser observada na figura 23.

Figura 23 - Interface da classe CadastroAlunoViewController



Fonte: Do autor

A tela de cadastro de alunos disponibiliza para o usuário o cadastro de diversos atributos, e estes atributos são todos armazenados em variáveis do tipo *IBOutlets* e entregues ao método *IBAction btnSalvarAluno*, que é responsável por armazenar um aluno em uma instância da classe *Aluno* e entregar ela a API de comunicação com o banco de dados chamada *AlunoDM*. O processo de cadastrar um aluno pode ser observado na figura 24.

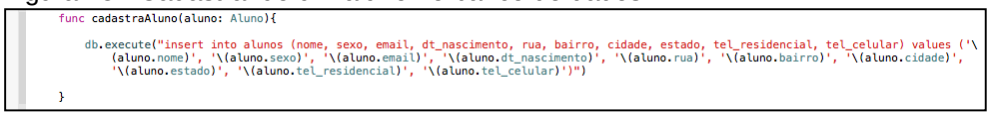
Figura 24 - Função de cadastro do aluno

```
@IBAction func btnSalvarAluno(sender: AnyObject) {

    //Armazenar os dados dos Outlets nas variaveis locais;
    nome = textFieldNome.text
    sexo = segControlSexo.selectedSegmentIndex == 0 ? "Masculino" : "Feminino"
    dt_nascimento = textFieldDtNascimento.text
    rua = textFieldRua.text
    bairro = textFieldBairro.text
    cidade = textFieldCidade.text
    estado = textFieldEstado.text
    tel_residencial = textFieldTelResidencial.text
    tel_celular = textFieldTelCelular.text
    email = textFieldEmail.text

    //Opcao para NOVO aluno;
    if (aluno == nil){

        //Instancia um novo aluno;
        aluno = Aluno(nome: nome, sexo: sexo, dt_nascimento: dt_nascimento, rua: rua, bairro: bairro, cidade: cidade, estado: estado,
            tel_residencial: tel_residencial, tel_celular: tel_celular, email: email)

        //Verifica se instancionou corretamente e o cadastra;
        if (aluno != nil) {
            alunoDM.cadastraAluno(aluno)
        }

        //Alerta de cadastro do aluno
        let alert: UIAlertView = UIAlertView(title: "SUCESSO", message: "Seu cadastro foi feito com sucesso", delegate: self,
            cancelButtonTitle: "OK!")
        alert.show()
```

Fonte: Do autor

A utilização da API de comunicação com o banco de dados *AlunoDM* que é executada pode ser observada na figura 25.

Figura 25 - Cadastrando um aluno no banco de dados



```
func cadastraAluno(aluno: Aluno){

    db.execute("insert into alunos (nome, sexo, email, dt_nascimento, rua, bairro, cidade, estado, tel_residencial, tel_celular) values ('\
        (aluno.nome)', '\(aluno.sexo)', '\(aluno.email)', '\(aluno.dt_nascimento)', '\(aluno.rua)', '\(aluno.bairro)', '\(aluno.cidade)',
        '\(aluno.estado)', '\(aluno.tel_residencial)', '\(aluno.tel_celular)')")

}
```

Fonte: Do autor

Após o cadastro ou edição de um aluno o sistema retorna sempre para a *viewController* padrão (*root*), que neste caso é o *ListaAlunosViewController.*

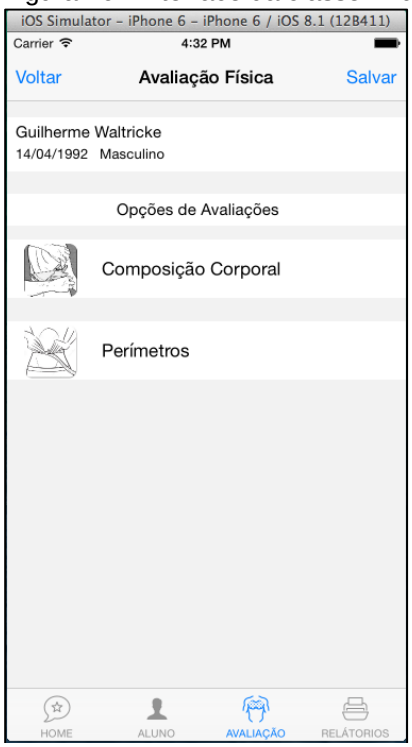
6.1.5.3 Avaliação

Menu avaliação é a área do aplicativo onde concentra-se os procedimentos de realização de uma avaliação física da composição corporal, aqui todos os tipos de avaliação são feitos e armazenados no banco de dados.

Sua composição inicial é parecida com a do cadastro de alunos, para se realizar uma avaliação física é necessário possuir um aluno, diante disto o procedimento padrão é sempre cadastrar o aluno a se efetuar avaliação física antes no sistema. Após ter o aluno cadastrado no aplicativo é possível se dar continuidade na avaliação, o sistema executa a tela de seleção do aluno a ser avaliado, a classe responsável por este procedimento é a *SelecionaAlunoTableViewController* e o seu funcionamento é semelhante ao da figura 22.

Uma segunda classe é chamada para dar continuidade ao fluxo da aplicação, após o usuário selecionar o aluno a ser avaliado clicando sobre seu nome na *tableView* de listagem de alunos, a classe a ser executada é a *AvaliacaoViewController.* Está classe é a menu central da avaliação aqui se concentra a central do fluxo de realização de uma avaliação física, sua tela pode ser observada na figura 26.

Figura 26 - Interface da classe AvaliacaoViewController



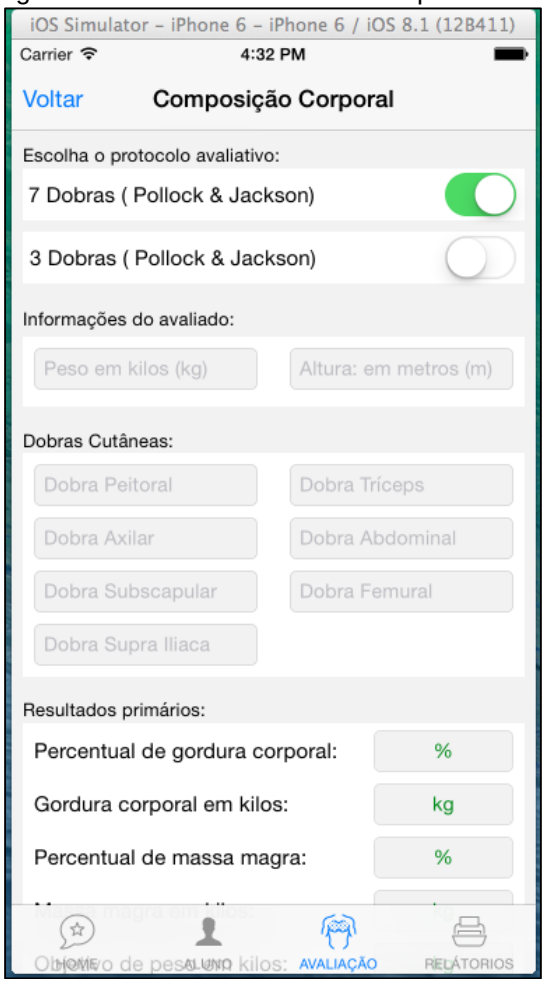
Fonte: Do autor

O funcionamento do menu de avaliações que possui a classe *AvaliacaoViewController* como controlador, é baseado em armazenar instâncias de classes de suas respectivas necessidades, assim concentrando o fluxo de armazenamento de informações em si mesmo. Uma avaliação física é composta por alguns tipos de avaliações, neste projeto o foco foi em dois tipos de avaliações a composição corporal e perímetros (PETROSKI, 2002). Os menus com os tipos de avaliações são dispostos na tela com seus respectivos nomes em dois botões do tipo *UIButton*.

O menu *Composição Corporal* é onde acontece o preenchimento dos dados de uma avaliação da composição corporal de um aluno, a classe responsável

pela tela é a *ComposicaoViewController,* sua interface pode ser observada na figura 27.

Figura 27 - Interface da classe ComposicaoViewController.swift



Fonte: Do autor

Seu funcionamento é iniciado pela escolha do tipo de protocolo a ser aplicado, com o auxílio do componente *UISlider* o avaliador escolhe uma das duas opções. Sempre que uma opção é selecionada a outra é automaticamente tirada de foco através do *selector selecProtocolo* implementado. É necessário preencher todos os campos de textos *UITextFields* para realizar a avaliação, ao editar o último campo, a função *textFieldDidEndEditing* que pode ser observada na figura 28 é executada através de uma trigger da linguagem Swift que é chamada de *Delegate*, nesta função todos os campos da tela são atribuídos a uma instancia da classe *Composicao* que é responsável pelo gerenciamento de uma composição corporal

em tempo de execução, e exibe na mesma tela na área de resultados primários alguns dados do aluno para uma rápida análise do avaliador.

Figura 28 - Função disparada pelo delegate do *UITextField*

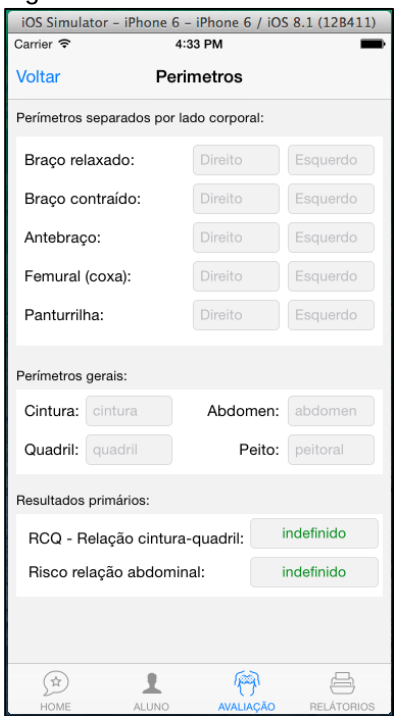
```
func textFieldDidEndEditing(textField: UITextField!) {   //delegate method

    //Verifica qual tipo Protocolo que ele vai utilizar;
    if switch7dobras.on {

        //Verificação de todos os valores necessários para os calculos;
        if ( (!textFieldPeso.text.isEmpty) && (!textFieldAltura.text.isEmpty) && (!textFieldDcTriceps.text.isEmpty) && (!
            textFieldDcSubscapular.text.isEmpty) && (!textFieldDcPeitoral.text.isEmpty) && (!textFieldDcAbdomen.text.isEmpty) && (!
            textFieldDcAxilar.text.isEmpty) && (!textFieldDcSupraIliaca.text.isEmpty) && (!textFieldDcFemural.text.isEmpty) ){

            //Carrega os dados para a classe de Composicao Corporal;
            composicao.peso = Double((textFieldPeso.text as NSString).doubleValue)
            composicao.altura = Double((textFieldAltura.text as NSString).doubleValue)
            composicao.dc_tricipis = Double((textFieldDcTriceps.text as NSString).doubleValue)
            composicao.dc_subscapular = Double((textFieldDcSubscapular.text as NSString).doubleValue)
            composicao.dc_suprailiaca = Double((textFieldDcSupraIliaca.text as NSString).doubleValue)
            composicao.dc_peitoral = Double((textFieldDcPeitoral.text as NSString).doubleValue)
            composicao.dc_abdominal = Double((textFieldDcAbdomen.text as NSString).doubleValue)
            composicao.dc_axiliar = Double((textFieldDcAxilar.text as NSString).doubleValue)
            composicao.dc_femural = Double((textFieldDcFemural.text as NSString).doubleValue)

            composicao.realizaAvaliacao() //Realiza a avaliação passando o tipo de 7 dobras;
            atualizaResultados() //Atualiza os valores dos resultados na tela;

        }
```

Fonte: Do autor

Depois de preenchido os dados da composição corporal, o usuário deve clicar em *"Voltar"* na barra de navegação para ir para o menu da *Avaliação*, ao fazer este procedimento a classe *ComposicaoViewController* devolve à classe *AvaliacaoViewController* uma instância da classe de armazenamento das informações da composição corporal, a classe *Composicao* totalmente popularizada e com os dados preenchidos.

Um segundo tipo de avaliação física também disponível é o menu *Perimetros*, o mesmo fica abaixo do *Composição Corporal* como foi mostrado na figura 26 e também é um componente UIButton. Ao menu ser selecionado o usuário é direcionado para a classe *PerimetrosViewController*, que é a classe que controla a interface a ser exibida, a mesma pode ser visualizada na ilustração da figura 29.

Figura 29 - Interface da classe *PerimetrosViewController*



Fonte: Do autor

Esta função da aplicação permite ao avaliador armazenar os dados relacionados aos perímetros de um aluno. O mesmo tem a sua disposição diversos tipos de perímetros e alguns quando se tem a necessidade possuem lados como esquerdo e direito. Quando o usuário preenche os campos de *Cintura* e *Quadril* uma *Delegate* é ativada através do método *textFieldDidEndEditing* e efetua o cálculo da função *calculoRelacaoCinturaQuadril()* que pode ser observada na figura 30. O mesmo procedimento é efetuado para o cálculo do risco de relação abdominal, quando o mesmo é preenchido a *Delegate* é ativada e executa a função *analiseCircuferenciaAbdomen()*.

Figura 30 - Função do cálculo da relação cintura-quadril

```
func calculoRelacaoCinturaQuadril(){
    var alunoIdade = aluno.calcularIdade()
    let rcq = cintura / quadril

    if aluno.sexo == "Masculino" {
        //Classificação de Risco para Homens;

        switch alunoIdade {
        case 20...29:
            if (rcq < 0.83){
                resultadoCinturaQuadril = "Baixo"
            }else if ( (rcq >= 0.83) && (rcq <= 0.88) ) {
                resultadoCinturaQuadril = "Moderado"
            }else if ( (rcq >= 0.89) && (rcq <= 0.94) ) {
                resultadoCinturaQuadril = "Alto"
            }else if (rcq > 0.94) {
                resultadoCinturaQuadril = "Muito Alto"
            }

        case 30...39:
            if (rcq < 0.84){
                resultadoCinturaQuadril = "Baixo"
            }else if ( (rcq >= 0.84) && (rcq <= 0.91) ) {
                resultadoCinturaQuadril = "Moderado"
            }else if ( (rcq >= 0.92) && (rcq <= 0.96) ) {
                resultadoCinturaQuadril = "Alto"
            }else if (rcq > 0.96) {
                resultadoCinturaQuadril = "Muito Alto"
            }

        case 40...49:
```

Fonte: Do autor

Com o fluxo de trabalho dentro do aplicativo, o botão voltar está disponível também nesta parte de perimetros do trabalho, e ao ser clicado devolve uma instância de *Perimetros* totalmente carregada de dados e devolve ao menu principal da avaliação física. Dentro de uma avaliação física o avaliador pode optar por não realizar algum tempo de avaliação, como avaliar um aluno apenas pela composição corporal e não pelos perímetros, mas segundo profissionais da área normalmente ambos são preenchidos.

Salvando uma avaliação é o último passo para ser realizado, cadastrando no banco de dados os procedimentos realizados pelo avaliador. Como explanado anteriormente a classe *AvaliacaoViewController* possuí as instâncias de todas as classes envolvidas no procedimento de uma avaliação física, sendo assim a sua tarefa é chamar o método *btnSalvarAvaliacao()* que executa a sua classe Data Module que é a API que efetua a comunicação com o banco de dados. A função da API *AvaliacaoDM* pode ser observada na figura 31.

Figura 31 - Função de cadastro de uma avaliação física

```
func cadastrarAvaliacao(aluno: Aluno, avaliacao: Avaliacao, composicao: Composicao, perimetros: Perimetros){
    //Salvando dados em variaveis locais para manipulação;

    //Tratando a data;
    var formatter: NSDateFormatter = NSDateFormatter()
    formatter.dateFormat = "dd-MM-yyyy"
    let stringDate: String = formatter.stringFromDate(NSDate())
    avaliacao.data = stringDate

    avaliacao.idade = String(aluno.calcularIdade())

    avaliacao.cod_aluno = aluno.codigo

    //Cadastra a COMPOSICAO;
    composicaoDM.cadastrarComposicacao(composicao)

    //Cadastra os Perimetros;
    perimetrosDM.cadastrarPerimetros(perimetros)

    //codigo da composicao
    if let data = db?.query("SELECT MAX (codigo) AS total FROM composicao_corporal") {
        let row = data[0]
        avaliacao.cod_composicao = row["total"]?.asInt()
    }

    //codigo dos perimetros;
    if let data = db?.query("SELECT MAX (codigo) AS total FROM perimetros"){
        let row = data[0]
        avaliacao.cod_perimetros = row["total"]?.asInt()
    }


    //CADASTRA a composicao;
    db.execute("INSERT INTO avaliacoes (data, idade, cod_aluno, cod_composicao, codigo_perimetros) values ( '\(avaliacao.data)', '\
        (avaliacao.idade)', '\(avaliacao.cod_aluno)', '\(avaliacao.cod_composicao)', '\(avaliacao.cod_perimetros)' )")
```

Fonte: Do autor

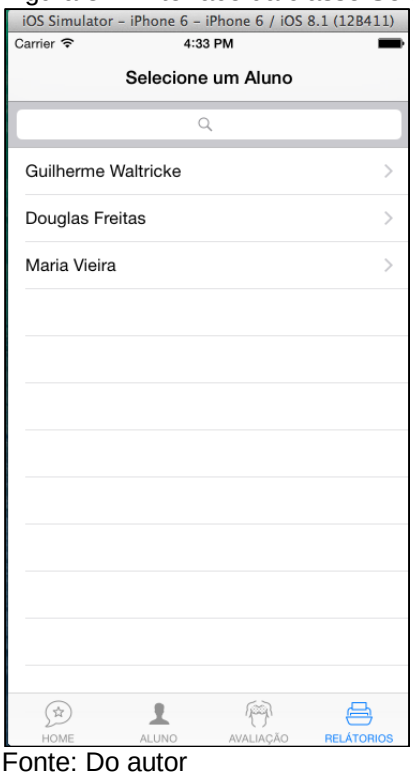
Assim que a avaliação física é salva, a mensagem de sucesso é exibida ao usuário e todos os dados já estão armazenados no banco de dados.

6.1.5.4 Relatórios

Menu *RELATÓRIOS* é a parte do aplicativo que entregará ao usuário resultados de todas as avaliações já realizadas. Aqui dados de cada avaliação são visualizados detalhadamente com todas as informações para o avaliador.

A primeira tela a ser visualizada ao acessar o menu de relatórios é a interface da classe *SelecionaAlunoTableViewController* que pode ser visualizada na figura 32 que tem como seu objetivo listar todos os alunos cadastrados no banco de dados e exibi-los em uma *tableView* para o usuário. Nesta tela é possível pesquisar um aluno cadastrado utilizando o componente *UISearchBar*. Desta forma quando é realizado uma pesquisa pelo nome do aluno, basta clicar no nome para prosseguir.

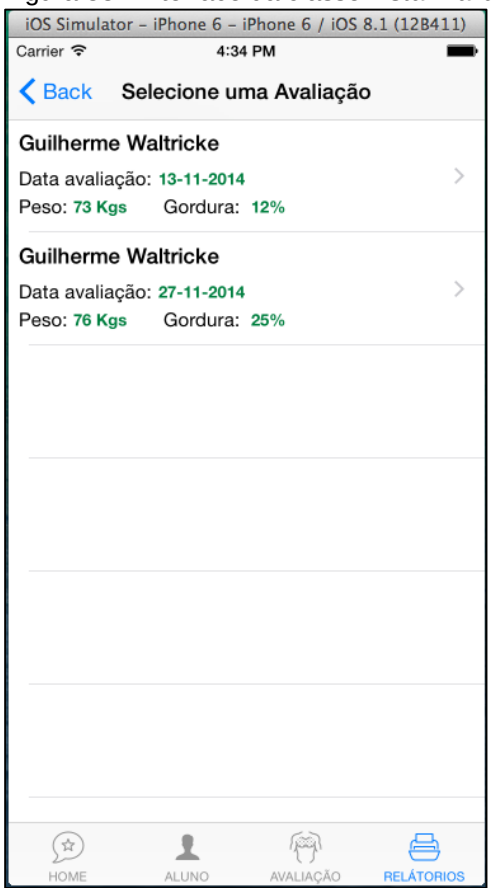
Figura 32 - Interface da classe SelecionaAlunoTableViewController.swift



Fonte: Do autor

Após a seleção do aluno, a próxima interface a ser exibida é a listagem de todas as avaliações efetuadas pelo aluno selecionado. A classe responsável pela listagem das avaliações é *ListaAvaliacoesTableViewController* que pode ser também visualizada na figura 33, está classe possui uma célula personalizada diferente das outras aqui neste trabalho exibidas, a célula personalizada faz parte da classe *MyCellTableViewCell* que permite mostrar *cells* da *tableView* de forma personalizada. Cada célula exibe a data, o peso e a porcentagem de gordura corporal do aluno na data da avaliação, então o avaliador visualiza qual o mesmo deseja exibir mais detalhes e clica sobre a mesma para partir para o relatório.

Figura 33 - Interface da classe *ListaAvaliacoesTableViewController*



Fonte: Do autor

Ao clicar sobre uma célula da tableView o usuário é redirecionado para a interface da classe *RelatorioAvaliacaoViewController* que é responsável por exibir todos os resultados de forma detalhada da avaliação física realizada sua interface pode ser visualizada na figura 34.

Conforme Guedes (2006) um avaliador físico tem que obter como resultado dados como: gordura corporal, massa magra, objetivo do peso, peso ideal, IMC, relação da circunferência quadril, dobras cutâneas e perímetros, esses dados são mostrados na tela como é possível visualizar na figura 34.

Figura 34 - Interface da classe RelatorioAvaliacaoViewController.swift

Fonte: Do autor

Os dados a serem exibidos são carregados a partir de lógica baseada na instância da classe *Avaliacao* que é enviada pelo *ListaAvaliacoesTableViewController*, pode-se então com auxílio da classe *RelatorioAvaliacaoDM* buscar todas as informações da avaliação física e exibir na tela utilizando o método *viewDidLoad()* que é disparado sempre quando a interface vai ser montada. O processo de carregar os *IBOutlets* e buscar as informações são exibidas na figura 35.

Figura 35 - Função de viewDidLoad()

```
override func viewDidLoad() {
    super.viewDidLoad()

    scroller.contentSize = CGSize(width: 320, height: 800)
    scroller.scrollEnabled = true

    if av != nil {
        //Setando as classes com suas determinadas informações;
        composicao = relAvaliacao.infoComposicaoCorporal(self.av)
        perimetros = relAvaliacao.infoPerimetros(self.av)

        //Dados de informação do aluno;
        //Formatar a data de forma correta (depois)
        labelDataAvaliacao.text = av.data

        labelPesoAluno.text = NSString(format: "%2.f Kgs", composicao.peso)
        labelAlturaAluno.text = NSString(format: "%0.2f Mts", composicao.altura)


        //Setando os valores de resultado da avaliação;
        textFieldKgsGordura.text = NSString(format: "%2.2f Kgs", composicao.gordura_corporal)
        textFieldPorcGordura.text = NSString(format: "%2.2f%%", composicao.porcentagem_gordura)
        textFieldPorcMassaMagra.text = NSString(format: "%2.2f%%", composicao.porcentagem_massaMagra)
        textFieldKgsMassaMagra.text = NSString(format: "%2.2f Kgs", composicao.massa_magra)

        textFieldObjetivoPeso.text = NSString(format: "%2.2f Kgs", composicao.peso_excesso)
        textFieldPesoIdeal.text = NSString(format: "%2.2f Kgs", composicao.peso_ideal)
        textFieldImcValor.text = NSString(format: "%2.2f", composicao.imc)
        textFieldImcInfo.text = composicao.classificacaoIMC

        //Pegando a classificação de Lohman
        composicao.aluno = aluno
        composicao.classificacaoPorPorcentagemGordura()
        textFieldClassificacaoLohman.text = composicao.classificacaoPG

    }
}
```

Fonte: Do autor

Como forma de auxiliar e melhorar a experiência de usuário ao visualizar os dados, foram separados os dados em mais duas telas auxiliares que são chamadas de *Dobras cutâneas* e *Perímetros*. Como são dados específicos e de grande quantidade de informação eles foram separados e são exibidos ao usuário ao clicar sobre seus botões exibidos na tela principal de avaliação. Ao ser executado o botão a função prepareForSegue() é executada e envia para a *ViewController* de cada uma das opções e envia a instância de classe da avaliação para que seja possível carregar os dados do banco e montá-los para o usuário.

A classe *DobrasCutaneasViewController* exibe as dobras cutâneas do aluno avaliado ela pode ser visualizada na figura 36. A classe recebe por parâmetro uma instância do tipo Composicao e assim pode acessar todos seus atributos e exibir na tela também com auxílio de *IBOutlets.*

Figura 36 - Interface da classe DobrasCutaneasViewController.swift

Fonte: Do autor

A classe *RelatorioPerimetrosViewController* exibe todos os perímetros cadastrados no momento da avaliação para o usuário, a interface pode ser observada na figura 37. Ao ser executada está classe recebe sempre uma instância de classe *Perimetros* assim pode carregar todos os valores e exibir.

Figura 37 - Interface da classe RelatorioPerimetrosViewController.Swift

Fonte: Do Autor

6.2 RESULTADOS OBTIDOS

Com a pesquisa desenvolvida neste trabalho, pôde-se obter um aplicativo para plataforma móvel iOS para avaliação da composição corporal que poderá auxiliar os profissionais de educação física. Sua utilização é simplificada ao máximo

para fornecer a qualquer usuário o mínimo de esforço possível no uso, e este resultado só é possível seguindo as boas práticas de experiência de usuário que foram aqui aplicadas.

Com o aplicativo o profissional da educação física poderá realizar uma avaliação da composição corporal de seu aluno em qualquer lugar em qualquer horário, basta ter o aplicativo em seu dispositivo instalado juntamente com os equipamentos de avaliação física necessários, diferentemente do que normalmente é utilizado, um software de mesa instalado em um computador geralmente em local fixo.

O funcionamento da aplicação foi avaliado em conjunto com profissionais da educação física e o co-orientador para trazer maior credibilidade aos resultados dos dados fornecidos pela aplicação.

A harmonia entre uma fundamentação teórica embasada e uma ferramenta de desenvolvimento correta o Swift, pode proporcionar como resultado a entrega do projeto inicial proposto o aplicativo que será disponibilizado para todos os profissionais da área na loja oficial da Apple a Apple Store. Um grande conhecimento foi obtido abrindo um novo caminho a ser percorrido neste nicho de aprendizado que é o desenvolvimento móvel para plataformas Apple iOS.

Como conclusão desta etapa, o aprendizado da linguagem Swift proporcionou o certificado de desenvolvedor ao pesquisador fornecido pela Apple, formalizando o resultado final do trabalho.

7 CONCLUSÃO

Este trabalho apresentou o aplicativo para avaliação da composição corporal, desenvolvido com a linguagem de programação Swift focada na plataforma móvel iOS. O aplicativo desenvolvido permite que um profissional da educação física realize uma avaliação física-funcional em seu aluno com seu dispositivo móvel.

Com o desenvolvimento desse projeto todos os objetivos explanados foram alcançados. Pode-se compreender o funcionamento da plataforma iOS, suas principais funcionalidades e características únicas de desenvolvimento. A integração da linguagem Swift com a plataforma iOS demonstrou-se perfeita, a linguagem adotada por padrão em desenvolvimentos iOS era o Objective-C como foi mostrado durante o trabalho, porém com o lançamento da linguagem Swift o desenvolvimento se tornou ainda melhor e escalável.

O Swift proporciona uma linguagem de alto nível para se desenvolver aplicações móveis, diferentemente de aplicações desenvolvidas com o Objective-C, pois nessa linguagem era necessário trabalhar com endereços de memória e possuir conhecimentos sobre ponteiros pois trata-se de uma linguagem de baixo nível.

Na realização da pesquisa também foi possível identificar e analisar os métodos de avaliação físico-funcionais para a construção do aplicativo.

Com o aplicativo desenvolvido foram realizados testes afim de corroborar com os resultados da fundamentação teórica, obtendo como positivos os resultados destes testes. Assim o aplicativo desenvolvido proporciona melhorias na forma pela qual os profissionais da Educação Física realizam as avaliações físicas.

Para fins acadêmicos pode-se aqui fornecer para o cunho educacional um trabalho com um tema atual e pouco explorado em trabalhos acadêmicos que é a linguagem Swift e o desenvolvimento móvel não hibrido para plataforma iOS. No cunho comunitário foi possível entregar uma aplicação funcional para a população que irá auxiliá-los em sua jornada.

Por ser uma linguagem relativamente nova lançada em julho de 2014 comparadas a outras linguagens como o Java lançado em 1995 o Swift passou por modificações que impactaram no projeto em seu desenvolvimento. Como iniciou-se o trabalho utilizando a versão beta do *Xcode with Swift* o projeto sofreu modificações constantes, até sair a versão final *Gold Master* do *Xcode 6* em Setembro de 2014.

Ao lançamento da versão final da linguagem, o projeto desenvolvido com suas características da versão beta foi totalmente incompatível com a versão final, necessitando assim o recomeço do desenvolvimento prático do trabalho, porém já com uma versão estável e com uma bibliografia mais ampla, o desenvolvimento do aplicativo para este trabalho foi possível de ser realizado.

Por fim, através de todo conhecimento adquirido, novos projetos podem ser desenvolvidos e novos caminhos podem se abrir. Por ser uma área em constante crescimento, ter a oportunidade de trabalhar com uma tecnologia recente e aprender um conteúdo ainda não aplicado dentro da universidade proporcionou uma experiência diferenciada no projeto.

Sugerem-se as seguintes extensões para trabalhos futuros:

a) sincronizar de dados na nuvem para armazenamento do banco de dados, utilizando webservices com *JSON*;
b) implementar notificações da aplicação em conjunto com API da Apple disponível no iOS 8;
c) utilizar bibliotecas gráficas nativas do Swift para exibir relatórios avaliativos de alunos;
d) criar um cadastro de avaliadores, para uso de dispositivos em local comum entre vários profissionais;
e) adicionar a avaliação física o tipo de avaliação postural, baseado em captura de imagens e leitura da imagem;
f) adicionar a avaliação física o tipo de avaliação de anamnese.

REFERÊNCIAS

APPLE. **iOSDev Center**. 2014a. Disponível em: <https://developer.apple.com/devcenter/ios/>. Acesso em: 13 março 2014.

_____. **Cocoa Touch Frameworks**. 2014b. Disponível em: <https://developer.apple.com/technologies/ios/cocoa-touch.html>. Acesso em: 19 março 2014.

_____. **iCloud for Developers**. 2014c. Disponível em: <https://developer.apple.com/icloud/>. Acesso em: 5 abril 2014.

_____. **Introducing Swift**. 2014d. Disponível em: <https://developer.apple.com/swift/>. Acesso em: 10 junho 2014.

_____. **iOS Human Interface Guidelines**. 2014e. Disponível em: <https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/>. Acesso em: 11 junho 2014.

_____. **Data Management in iOS**. 2014f. Disponível em: <https://developer.apple.com/technologies/ios/data-management.html>. Acesso em: 6 maio 2014.

_____.**Archives and Serializations Programming Guide**. 2012. Disponível em: <https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/Archiving/Archiving.html>. Acesso em: 10 abril 2014.

_____. **Introduction to Property Lists**. 2010. Disponível em: <https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html>. Acesso em: 10 abril 2014.

_____. **UIKit User Interface Catalog**. 2013. Disponível em: <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/

UIKitUICatalog/>. Acesso em: 21 maio 2014.

APPLE INC. **The Swift Programming Language**: Book 1 Swift Programming Series. 2014a. 500 p.

_____. **Using Swift with Cocoa and Objective-C**: Book 2, Swift Programming Series. 2014b. 100 p.

AVALIAÇÃO FÍSICA ONLINE. **Perímetros Corporais**. 2004. Disponível em < http://www.avaliacaofisica.com.br/si/site/021206>. Acesso em: 21 maio 2014.

BELL, Douglas. **Software Engineering for Students**.4th ed. Dorset Press, 2000. 447 p.

BONOME at al. **Disseminação do uso de aplicativos móveis na atenção à saúde**. São Paulo: São Paulo, 2012. Disponível em: <http://www.sbis.org.br/cbis2012/arquivos/807.pdf>. Acesso em 23 junho 2014.

CALDEIRA, Ana L. M. **Desenvolvimento de uma aplicação Businnes Intelligence parai OS**. Lisboa: Portugal, 2006. Disponível em: <https://repositorio.utad.pt/bitstream/10348/3026/1/msc_almcaldeira.pdf>. Acesso em 23 junho 2014.

CESCORF. **Plicômetro Clínico Tradicional**. 2012. Disponível em: <http://www.cescorf.com.br/portfolioentry/plicometro-clinico-tradicional-3/>. Acesso em: 21 de maio 2014.

COHN, Mike. **Succeding with Agile**. Pearson Education, 2010.503 p.

COSTA, Francielly M. R.; OLVEIRA Thiago R. **Desenvolvimento de aplicativo móvel de referência sobre vacinação no brasil**. Brasil, 2012. Journal of Health Informatics (JHI). Acesso em: 21 outubro 2014.

Coulouris, Dollimore e Kindberg 2005 COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos**: Conceitos e Projeto. Inglaterra: Bookman, 2005.

DOOLEY, John. **Software development and Professional Practice**. New York: Springer Science+Business Media, 2011. 255 p.

FABENI, Alan F. C. **Tagarela**: Aplicativo para comunicação alternativa no iOS. Blumenau, 2012. Disponível em: <https://app.box.com/s/x3jok9s7q4bv4g5g9cp3>. Acesso em 20 junho 2014.

FARINATTI, Paulo de Tarso V.; MONTEIRO, Walace David. **Fisiologia e avaliação funcional**. 2º edição. Rio de Janeiro: Sprint, 1992. 302 p.

Falchuk e Loeb 2007 FALCHUK, B.; LOEB, S. **Mobile healthcare**: Technologies and architectures. In: IEEE/NIH Life Science Systems and Applications Workshop. [S.1.]: LISSA, 2007. p. 180-183.

GUEDES, Dartagnan Pinto; GUEDES, Joana Elisabete Ribeiro Pinto. **Manual prático para avaliação física**. Barueri: Manole, 2006. 484 p.

HEYWARD,Vivian H. **Avaliação física e prescrição do exercício**: técnicas avançadas. Porto Alegre: Artmed, Tradução Márcia Dornelles, 4º Edição. 2004. 319 p.

HOROVITZ, Alex et al. **More iOS 6 Development**. New York: Springer Science+Business, 2012. 542 p.

KOCHAN, Stephen G. **Programming in Objective-C**.4th ed. Personal Tech Group, 2012. 562 p.

LECHETA, Ricardo R. **Desenvolvendo para iPhone e iPad**.2º edição. São Paulo: Novatec Editora, 2012. 749 p.

Lee et al. 2002 LEE, B.-Y.et al. **Data synchronization protocol in mobile computing environment using sync ml**. In: Conference on High Speed Networks and Multimedia Communications 5th IEEE International. [S.l.: s.n.], 2002. p. 133-146.

Lohman, T. G. **Advances in Body Composition Assessment**: CurrentIssuesinExercise Science. 1992.Monograph 3. Champaign, Illinois: HumanKineticsPublishers.

MACHADO, Dalmo Roberto Lopes. **Composição Corporal**: Antropometria I. 2012. Disponível em: <http://sistemas.eeferp.usp.br/myron/arquivos/3396411/867fb79974eed4593fa0708aef4acba3.pdf>. Acesso em: 16 maio 2014.

MACWORLD, Vendas **de Tablets vão superar os notebooks e netbooks em 2013**, Disponível em: <http://macworldbrasil.uol.com.br/noticias/2013/05/29/vendas-de-tablets-vao-superar-notebooks-e-netbooks-em-2013-diz-idc/>. Acesso em: 10 setembro 2013.

MARK, David at al. **Beginning iOS 6 Development**: Exploring the iOS SDK. New York: Springer Science+Business Media, 2013. 750 p.

MATHEWS, Donald K. **Medida e Avaliação Física**. Rio de Janeiro: Inter americana, 1980. 452 p.

NAHAS, Markus Vinícius. **Atividade física, saúde e qualidade de vida**: conceitos e sugestões para um estilo de vida ativo.4º edição. Londrina: Madiograf, 2006. 282 p.

NEIL, Theresa. **Mobile Design Pattern Gallery**. Sebastopol,CA: O'Reilly, 2012. 278 p.

NEUBURG, Matt. **iOS7Programming Fundamentals**. Sebastopol,CA: O'Reilly, 2014. 422 p.

PETROSKI, Edio Luiz. **Antropometria**: técnicas e padronizações. Porto Alegre: E.L. Petroski, 2003. 160 p.

PILONE, Dan et al. **Head First iPhone and iPad Development**. Sebastopol,CA: O'Reilly, 2014. 365 p.

JACKSON, A. S.; POLLOK, M. L. **Prediction accuracy of body density**: lean body weight, and total body volume equations. MedSci Sports. 1977;9(4):197-201.

PRESSMAN, Roger S. **Engenharia de Software**. tradução da 3º edição. São Paulo: MacGraw-Hill, 1995. 720 p.

PRESSMAN, Roger S. **Software Engineering**: a practitioner's approach. 5th ed. New York: McGraw-Hill, 2001. 743 p.

ROCHA, Paulo Eduardo Carnaval Pereira. **Medidas e Avaliação em Ciências do Esporte**.2º Edição. Rio de Janeiro: Sprint Editora, 1997. 161 p.

SALOMÃO, P. L. **Utilização do computador de mão integrado à telefonia celular no atendimento médico**: desenvolvimento de sistema e avaliação. Tese (Doutorado) | Universidade Federal de São Paulo, 2007.

Silva at al. **Uso do iOS como ferramenta de interação do cliente com o ambiente de um restaurante**. São Paulo, 2010. Disponível em: <http://engenharia.anhembi.br/tcc-10/si-03.pdf>. Acesso em 20 outubro 2014.

SOMMERVILLE, Ian. **Engenharia de Software**.6º Edição. Pearson Education, 2003.792 p.

TURNER, James. **Developing Enterprise iOS Applications**. Sebastopol, CA: O'Reilly, 2012. 114 p.

WASSERMAN, Anthony I. **Software Enginnering Issues for Mobile Application Development**. Moffett Field, CA. 2010. Disponível em: <http://repository.cmu.edu/silicon_valley/40/>. Acesso em 21 de abril 2014.

APÊNDICE A - ARTIGO

A Utilização da Linguagem Swift em Dispositivos Móveis Baseados no Sistema Operacional iOS para Avaliação da Composição Corporal

Guilherme W. Freitas[1], Luciano Antunes[1], Joni Marcio Farias[1]

[1]Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 88806-000 – Criciúma – SC – Brasil

guiwaltricke@gmail.com

Resumo. *Dispositivos móveis estão cada vez mais difundidos na cultura popular, impulsionados pelo alto consumo de smartphones e tablets. Com o grande avanço nesse setor é possível ter o acesso a dispositivos móveis com alto poder de processamento e armazenamento, proporcionando ao desenvolvedor migrar de aplicações antes restritas a computadores de mesa para realidade móvel. Diante disto o trabalho apresenta o desenvolvimento de um aplicativo para realização de uma avaliação da composição corporal na plataforma móvel iOS utilizando a linguagem de programação Swift. Por fim em testes é possível verificar os resultados como positivos.*

Abstract. *Mobile devices are increasingly pervasive in popular culture, boosting the high consumption of smartphones and tablets. With the breakthrough in this sector, it is possible to have access to mobile devices with high processing power and storage, providing the developer to migrate to applications previously restricted to desktop computers to mobile reality. Therefore, the paper presents the development of an application to perform a body composition evaluation based on the iOS mobile platform using the Swift programming language. Finally, in tests conducted with the help of physical education professional it was verified that the results are suitable from the developed application.*

1. Sistema Operacional iOS

iOS é o nome dado pela Apple Inc. para seu sistema operacional móvel criado em 2007, este sistema também já foi conhecido pelo nome de iPhone OS porém foi alterado para iOS devido ao seu uso estendido para outros dispositivos da Apple como iPad, iPod Touch e recentemente para a segunda geração da AppleTV. Diferentemente de outras fabricantes de software a Apple não licencia o seu sistema operacional para instalação em dispositivos de outros fabricantes.

A tecnologia aplicada no iOS teve como principal característica sua interface com usuário que permite multi-toques (multi-touch) entregando ao usuário uma experiência diferenciada com combinações variadas de gestos em tela. Outra característica que enriquece o sistema operacional móvel da Apple é sua única e exclusiva integração com seu hardware, famosa por desenvolver bons conjuntos de hardware e software, o que garante ao sistema um ótimo desempenho geral.

iOS se destaca como uma das principais plataformas de desenvolvimento móvel do mercado, pois é uma das preferidas pelos usuários e desenvolvedores. Está preferência inicia-se por um conjunto de fatores que começa pelo lado do usuário que possui uma loja virtual que ultrapassa um milhão de aplicativos.

No lado oposto o desenvolvedor que dispõem do Software *Development Kit* (SDK), que trata-se de um poderoso conjunto de ferramentas fornecidas pela fabricante para auxiliar os desenvolvedores no desenvolvimento de aplicativos. O SDK do iOS possui diversas aplicações inclusas, entre elas o Xcode, iOS Simulator e o *Instruments.*

1.1 Estrutura

O iOS possui uma arquitetura dividida em quatro camadas que são conhecidas normalmente como layers elas são: Cocoa Touch, Media, Core Services e Core OS.

A camada Cocoa touch é o topo das camadas do sistema, pois é a camada que fornece a maior integração com o usuário. Na camada de Media estão as tecnologias aplicadas a funções multimídia. A Camada Core Services possuí diversos frameworks que as aplicações para o iOS utilizam de forma indireta, pode-se exemplificar o suporte que esta camada fornece na utilização da biblioteca SQLite que permite as aplicações desenvolvidas realizem o armazenamento de dados com estrutura SQL. Por fim a camada Core OS é a de mais baixo nível da hierarquia devido ao fato de todas as outras camadas são construídas a partir dela. Seu uso a partir das aplicações é raro diante de que ela é recomendada apenas em situações que se é necessário lidar diretamente com segurança ou com a comunicação com algum hardware externo.

2. Tecnologias de Desenvolvimento

Para realização de um desenvolvimento de um software é necessário conhecimento em diversas áreas, um software pode ser subdividido em tarefas distintas, para o projeto proposto essas áreas e conhecimentos são divididas em: engenharia de software, SDK para desenvolvimento, linguagem de desenvolvimento Objective-C, linguagem de desenvolvimento Swift, persistência de dados, padrão de desenvolvimento e padrões de design.

2.1 SDK para Desenvolvimento Apple Xcode

Conforme mencionado no capítulo anterior, a Apple disponibiliza para os desenvolvedores de aplicativos um framework completo de desenvolvimento contendo ferramentas muito úteis elas são: Xcode, iOS Simulator e Instruments. Todas as ferramentas do SDK de desenvolvimento são gratuitas e exclusivas para o sistema operacional MAC OS X da Apple.

O Xcode é nome dado a uma *Integrated Development Environment* (IDE) que fornece um completo ambiente para o desenvolvimento de aplicações nativas para o iOS. iOS Simulator é a ferramenta de teste do desenvolvedor, a Apple disponibiliza ela de modo completo onde o desenvolvedor pode realizar os testes do seu aplicativo direto no computador sem a necessidade de um dispositivo externo. A ferramenta Instruments é utilizada para analisar o desempenho da aplicação desenvolvida, ela analisa tanto a aplicação rodando no iOS Simulator como também no dispositivo externo.

2.2 Linguagem de Desenvolvimento Objective-C

Dennis Ritchie foi o pioneiro na programação em C nos laboratórios da AT&T em meados de 1970. Porém a linguagem não ganhou muita popularidade e suporte fora dos portões dos laboratórios da AT&T, Isto porque compiladores em C não são realmente produtivos para aplicações comerciais.

Alguns anos após em meados de 1980, Brad J. Cox criou a linguagem Objective-C, a linguagem foi baseada em uma outra linguagem chamada de SmallTalk-80. Objective-C foi desenhado para o topo das linguagens C, extensões foram adicionadas ao C para criar está nova linguagem que tem disponível Objetos (objects) para serem criados e manipulados.

A empresa NeXT licenciou o Objective-C em 1988 e adicionou ela a *Free Software Foundation's GNU development environment*, tornando-se assim uma licença de uso público e qualquer usuário pode usá-la. Após mais de uma década em 1996 a Apple compra a NeXT e assim possuí os direitos e bibliotecas do Objective-C, tudo isto planejando o lançamento de seu sistema operacional MAC OS X e alguns anos depois o lançamento do primeiro celular inteligente o iPhone com sistema operacional iOS também baseado na sua linguagem própria o Objective-C

2.3 Linguagem de Desenvolvimento Swift

Todos os anos a Apple apresenta seu evento especial focado em novos softwares e tecnologias para desenvolvedores o Worldwide Developers Conference (WWDC), em junho de 2014 uma grande novidade foi apresentada por Tim Cook atual CEO da companhia, a nova linguagem de desenvolvimento Swift.

Swift é a nova linguagem de programação para desenvolvimento de aplicativos para iOS e OS X, ela foi desenvolvida pela própria Apple para substituir o Objective-C, a nova linguagem foi resultado de últimas pesquisas em linguagens de programação, que foi combinada com décadas de experiência na criação de aplicações para iOS e OS X. Swift adota uma programação segura e moderna que torna o desenvolvimento de aplicativos mais fácil, mais flexível, e mais divertido. Objetivo é entregar ao desenvolvedor uma experiência de desenvolvimento única, para isto é empregado técnicas de linguagens de alto nível. Ela é exemplificada como o Objective-C sem o termo C, se tornando assim uma linguagem altamente ligada a objetos com modelos e sintaxes modernas.

Rápida e poderosa é o conceito sobre Swift isto é baseado que todo código feito com a linguagem é construído de forma rápida usando a alta performance do compilador LLVM, o código é transformado e otimizado em código nativo.

Sintaxe é um dos pontos fortes do Swift, suas novidades são a ausência da necessidade cabeçalhos conhecidos como headers (.h) na sua criação, o gerenciamento de memória é feito de maneira automática que é diferente do Objective-C e não existe a necessidade também da utilização do ponto-e-vírgula ao final de cada comando.

2.4 Persistência de Dados

Grande parte das aplicações possuem a necessidade de ler e reter informações geradas por si mesma, nove em cada dez aplicações tem salvo algum tipo de informação de forma persistente, armazenamento não volátil sobrevive a um restart do sistema e por isso se torna de fundamental a sua importância. Está realidade é vivenciada por grande parte dos usuários, seguindo o fato de que toda vez que a aplicação é lançada, ela aparece exatamente da mesma forma que foi encerrada na última utilização.

SQLite é um banco de dados simples e popular entre dispositivos móveis, é um projeto de código-fonte aberto, É implementado em linguagem C que o torna um banco de dados rápido com performance poderosa, e justamente devido a estes fatores o fez lhe tornar tão dizimado em dispositivos móveis.

iOS inclui a biblioteca SQLite3 por padrão disponível para ser utilizada em suas aplicações (APPLE, 2014b, tradução nossa). SQLite3 é eficiente em armazenar e recuperar dados como também é capaz de fazer passos complexos de armazenamento de dados, com mais desempenho maior do que é possível utilizando os mesmos passos com objetos como *archives*. Como por exemplo uma aplicação necessita calcular a soma de um campo em particular em todos os objetos na aplicação, utilizando-se de objetos é necessário carregar todos os objetos em memória e calcula-los, se apropriando do conhecimento em SQLite3 é possível fazer isto sem carregar todos os objetos em memória, SQLite3 possuí ferramentas que podem somar estes dados de forma mais rápida e sem ter que carregar todos os objetos assim não gastando memória do dispositivo.

2.5 Model View Controller (MVC)

O padrão de design de projetos Model-View-Controller é uma técnica que se apropria do conhecimento em orientação a objetos, que auxilia na separação de uma aplicação ou ainda apenas um pedaço da interface da aplicação em três partes: o model (modelo), a view (interface) e o controller (controlador) formando o MVC.

O model (modelo) gerencia um ou mais elementos de informação, ele responde a queries sobre estados, e responde a instruções de mudança de estado da aplicação. O modelo sabe como a aplicação supostamente tem que fazer e como fazer, isto é a principal estrutura computacional de uma arquitetura MVC.

A view ou comumente chamada de interface é a área retangular que é mostrada na tela para o usuário e é responsável por apresentar todas as informações através de uma combinação de gráficos e textos. A view não pode saber nada sobre como a aplicação está se comportando, está parte é de responsabilidade do controller (controlador) outra parte do MVC. A interface apenas se encarrega de pegar os dados para serem exibidos do controlador e mostra-los na tela para o usuário.

O controller (controlador) é a parte que interpreta as entradas de mouse e teclado do usuário, essas ações efetuadas pelo usuário são enviadas para o model (modelo) e ou para a view com os efeitos propriamente alterados.

2.6 Front End

Pessoas possuem a tendência a não se importar com a experiência de navegação em um aplicativo ao menos quando navegação não corresponde a expectativas delas. O trabalho de criar padrões de design em um aplicativo é implementar a navegação do mesmo em um caminho que suporta a estrutura e o propósito do aplicativo sem chamar a atenção para si próprio.

Segundo Apple quase todos aplicativos iOS usam no mínimo alguns dos componentes de UI (User Interface) fornecidos pela UIKit Framework. Conhecendo os nomes, funcionamento e capacidades desses componentes fornecidos capacita o desenvolvedor a criar um design intuitivo e que auxilia nas decisões a serem tomadas.

Uma boa navegação como um bom design são invisíveis, uma navegação tem que atrair o sentimento intuitivo e tornar está tarefa fácil de realizar para quaisquer funções. Existem dezenas de opções para navegação dentro de um aplicativo móvel, como foco de boas práticas pode-se separar em sete tipos de navegação são eles: spring board, list menu, tab menu, gallery, dash board, metaphor e mega menu.

A Apple disponibiliza para o desenvolvedor o iOS Guide User Interface (GUI) é o guia de desenvolvimento de interfaces para usuários, que auxilia a criar padrões de design para aplicativos que reforçam a importância de padronização e boas práticas.

3. Avaliação da Composição Corporal

A Educação Física faz parte de um grupo com outras áreas da saúde, que possuí um papel extremamente importante no aspecto sócio educativo, criando um estilo de vida saudável e elevando a qualidade de vida, independentemente da idade, sexo, nível socioeconômico ou condição funcional da pessoa.

Avaliar fatores relacionados a pratica da atividade física tem sido um ponto forte da atenção dos profissionais desta área. Este fato é justificável diante do importante papel do profissional de educação física em avaliar indicadores biológicos, comportamentais e socioculturais que apresentam relação direta e indireta com atividade física que possui complexidade comparável à diversidade e à dificuldade que lhes são inerentes.

Segundo Guedes pode-se conceituar a composição corporal em dividir o peso corporal total em seus diferentes componentes como ossos, músculos, gorduras, água dentre outros e esta divisão resulta em um aglomerado de informações valiosas sobre o comportamento de indicadores ligados ao crescimento físico e aos programas de controle do peso corporal, assim é possível intervir com programas nutricionais e de prática de exercícios físicos.

A avaliação física é um processo pelo qual utilizando medidas, pode-se subjetiva e objetivamente permitir visualizar a realidade do trabalho que se desenvolve, criando condições para que se entenda o grupo e situa-se um indivíduo dentro deste grupo. A avaliação física permitirá avaliar os principais componentes estruturais do corpo humano, cujo enfoque no meio cientifico tem que concentrado na determinação de massa gorda e da massa magra.

Segundo Guedes pode-se definir antropometria como conjunto de técnicas para se medir dimensões do corpo humano e é utilizada na avaliação do crescimento físico, está avaliação se apropria dos procedimentos indiretos e duplamente indiretos para serem realizados. Dentro da antropometria existem algumas áreas de estudo mas para fim de estudo deste trabalho pode-se destacar em três importantes como alturas e comprimentos, dobras cutâneas e perímetros.

Dobras cutâneas são as espessuras correspondentes a camada dupla de pele e de tecido subcutâneo destacados em pontos específicos do corpo e também são chamadas de: coxa, axilar média, abdominal, panturrilha, peitoral, tricipital, supra-iliaca e subescapular. As medidas cutâneas são utilizadas para estimar a composição corporal, para mensuração cutânea é utilizado o instrumento conhecido como adipômetro.

Após um conjunto de informações adquiridas de como e o que é necessário para realizar uma avaliação física da composição corporal, o que falta é obter o conhecimento dos protocolos mais conhecidos como formulas para se chegar ao objetivo da avaliação, que é possuir dados de um indivíduo como: Densidade Corporal (DC), Massa de Gordura (MG ou G) e Massa Corporal Magra (MCM).

Baseados na procura de realizar uma avaliação com precisão de 7 dobras cutâneas retiradas das seguintes regiões: peitoral, coxa, abdômen, tríceps, subescapular, supra-iliaca e axilar formando o $\sum 7DC$. As seguintes fórmulas foram desenvolvidas por Jackson e Pollock (1978) para os subgrupo populacionais abaixo:

Homens brancos ou negros, atletas entre 18 a 55 anos de idade:

$$DC = 1{,}112 - [0{,}00043499 * \left(\sum 7DC\right) + 0{,}00000055 * \left(\sum 7DC\right)^2 - [0{,}0002882(idade)]]$$

Mulheres negras ou brancas entre 18 a 55 anos de idade:

$$DC = 1{,}0970 - [0{,}00046971 * \left(\sum 7DC\right) + 0{,}00000056 * \left(\sum 7DC\right)^2 - [0{,}00012828 * (idade)]]$$

Adaptando também a preferência de outros avaliadores Jackson & Pollock adaptaram suas equações quadráticas para profissionais que preferem pela adoção de apenas três dobras cutâneas porém aqui existe diferenciação dos locais de subtrair as dobras por sexo.

Homens negros ou brancos, atletas entre 18 e 55 anos de idade onde $\sum 3DC$ é o somatório das seguintes dobras cutâneas: tríceps, subescapular e peitoral.

$$DC = 1{,}1125025 - [0{,}0013125 * \left(\sum 3DC\right) + 0{,}0000055 * \left(\sum 3DC\right)^2 - [0{,}0002440 * (idade)]]$$

Mulheres negras ou brancas, entre 18 e 55 anos de idade onde $\sum 3DC$ é o somatório das seguintes dobras cutâneas: tríceps, abdominal e supra-ilíaca.

$$DC = 1{,}089733 - [0{,}0009245 * \left(\sum 3DC\right) + 0{,}0000025 * \left(\sum 3DC\right)^2 - [0{,}0000979 * (idade)]]$$

4. A Utilização da Linguagem Swift em Dispositivos Móveis Baseados no Sistema Operacional iOS para Avaliação da Composição Corporal

Diante do grande avanço tecnológico na computação móvel e das dificuldades encontradas por profissionais da educação física em realizar uma avaliação física-funcional, foi desenvolvido um aplicativo para dispositivos móveis baseados no sistema operacional iOS da Apple que auxilie o profissional no emprego de suas atividades. Este desenvolvimento foi realizado com os conhecimentos adquiridos no levantamento bibliográfico efetuado neste trabalho.

4.1. Metodologia

Para realização do projeto foi realizado uma reunião com o profissional da educação física e co-orientador Prof. Dr. Joni Marcio Farias com o objetivo de adquirir informações necessárias para o desenvolvimento do aplicado. O co-orientador forneceu os requisitos necessários e o material bibliográfico a ser pesquisado pelo orientado.

Posteriormente com a ideia do aplicativo formada e os requisitos obtidos, foi realizado o levantamento bibliográfico para ampliar o embasamento científico para viabilizar este trabalho. Em um próximo passo do trabalho foi possível realizar a análise de requisitos com seus diagramas afim de fixar o funcionamento do software e seu comportamento junto ao usuário.

Como o desenvolvimento do projeto foi planejado para o sistema operacional iOS, realizou-se um estudo aprofundado para compreender a forma de desenvolvimento de aplicativos para esta plataforma. Este estudo foi iniciado levantando as tecnologias que seriam utilizadas como: IDE de desenvolvimento Xcode; banco de dados SQLite; experiência de usuário (UX); e o foco principal deste trabalho, que é a linguagem de programação Swift

Por fim com a quantidade de informações obtidas foi possível iniciar a implementação do aplicativo e a realização de testes.

4.2. Requisitos

Os requisitos foram levantados durante a pesquisa e classificados em Requisito Funcional (RF) e Requisito Não-Funcional (RNF), os requisitos são listados abaixo:

a) o sistema realizará um cadastro de alunos (RF);

b) o sistema permitirá uma pesquisa de alunos cadastrados(RF);

c) o sistema realizará uma avaliação física-funcional (RF);

d) o sistema permitirá uma pesquisa de avaliações realizadas (RF);

e) será alertado ao avaliador alunos a serem reavaliados (RF);

f) o aplicativo deverá ser implementada na linguagem de programação Swift (RNF);

g) toda informação não volátil será armazenada em banco de dados relacional SQLite (RNF);

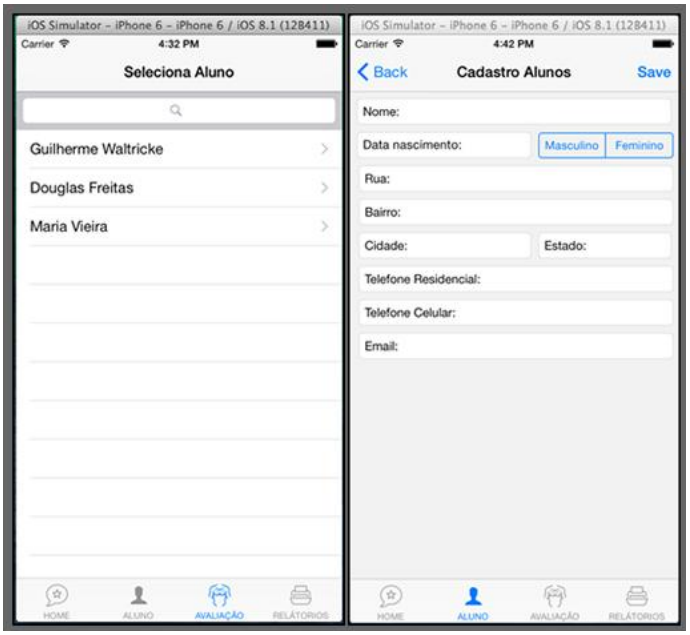
h) o aplicativo deverá utilizar o ambiente de desenvolvimento Xcode para ser implementado (RNF).

4.3. Principais Funções da Aplicação

Para realizar a implementação do aplicativo foi utilizado a linguagem de programação Swift que trabalha em conjunto com sua IDE de desenvolvimento o Xcode na sua versão 6.0.1 juntamente com o sistema operacional iOS 8. Para testes e simulações foi utilizado o iOS Simulator.

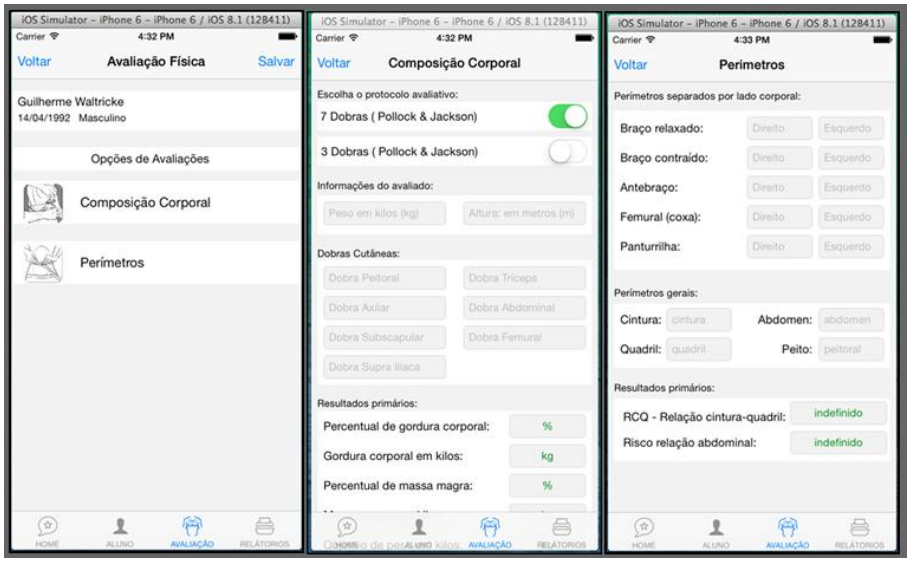
Procurando realizar a persistência das informações geradas pelo aplicativo de forma não volátil surge a necessidade um banco de dados, e diante desta necessidade foi adotado para realização deste projeto o SQLite.

O menu *ALUNO* da aplicação é responsável pelo gerenciamento de informações de um aluno no fluxo do sistema, um aluno pode ser visualizado em uma lista de usuários cadastrados como também ser pesquisado e por fim adicionado ao banco de dados como ilustrado na Figura 1.



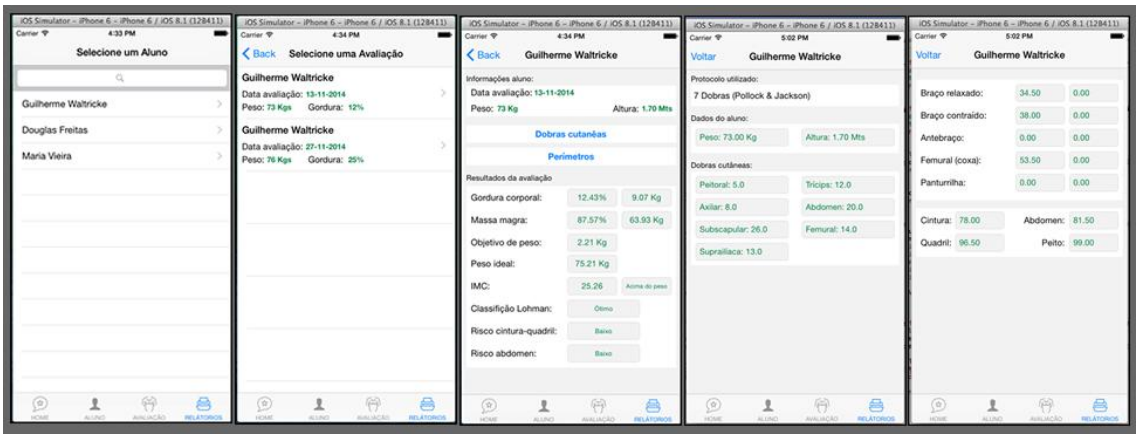
2 Figura 1 – Telas do menu *ALUNO.*

Menu *AVALIAÇÃO* é a área do aplicativo onde concentra-se os procedimentos de realização de uma avaliação física da composição corporal, aqui todos os tipos de avaliação são feitos e armazenados no banco de dados, suas telas podem ser visualizadas na Figura 2.



3 Figura 2 – Telas do menu *AVALIAÇÃO* do aplicativo.

Menu RELATÓRIOS é a parte do aplicativo que entregará ao usuário resultados de todas as avaliações já realizadas. Aqui dados de cada avaliação são visualizados detalhadamente com todas as informações para o avaliador. A interface de seu menu pode ser visualizado na Figura 3.



4 Figura 3 - Seleção de interfaces do menu *RELÁTORIOS*.

5. Resultados Obtidos

Com a pesquisa desenvolvida neste trabalho, pôde-se obter um aplicativo para plataforma móvel iOS para avaliação da composição corporal que poderá auxiliar os profissionais de educação física. Sua utilização é simplificada ao máximo para fornecer a qualquer usuário o mínimo de esforço possível no uso, e este resultado só é possível seguindo as boas práticas de experiência de usuário que foram aqui aplicadas.

Com o aplicativo o profissional da educação física poderá realizar uma avaliação da composição corporal de seu aluno em qualquer lugar em qualquer horário, basta ter o aplicativo em seu dispositivo instalado juntamente com os equipamentos de avaliação física necessários, diferentemente do que normalmente é utilizado, um software de mesa instalado em um computador geralmente em local fixo.

O funcionamento da aplicação foi avaliado em conjunto com profissionais da educação física e o co-orientador para trazer maior credibilidade aos resultados dos dados fornecidos pela aplicação.

A harmonia entre uma fundamentação teórica embasada e uma ferramenta de desenvolvimento correta o Swift, pode proporcionar como resultado a entrega do projeto inicial proposto o aplicativo que será disponibilizado para todos os profissionais da área na loja oficial da Apple a Apple Store.

Referências

APPLE INC. The Swift Programming Language: Book 1 Swift Programming Series. 2014. 500 p.

APPLE. Introducing Swift. 2014d. Disponível em: <https://developer.apple.com/swift/>. Acesso em: 10 junho 2014.

BONOME at al. Disseminação do uso de aplicativos móveis na atenção à saúde. São Paulo: São Paulo, 2012. Disponível em: <http://www.sbis.org.br/cbis2012/arquivos/807.pdf>. Acesso em 23 junho 2014.

GUEDES, Dartagnan Pinto; GUEDES, Joana Elisabete Ribeiro Pinto. Manual prático para avaliação física. Barueri: Manole, 2006. 484 p.

LECHETA, Ricardo R. Desenvolvendo para iPhone e iPad.2º edição. São Paulo: Novatec Editora, 2012. 749 p.

NEIL, Theresa. Mobile Design Pattern Gallery. Sebastopol,CA: O'Reilly, 2012. 278 p.

MARK, David at al. Beginning iOS 6 Development: Exploring the iOS SDK. New York: Springer Science+Business Media, 2013. 750 p.

PETROSKI, Edio Luiz. Antropometria: técnicas e padronizações. Porto Alegre: E.L. Petroski, 2003. 160 p.

JACKSON, A. S.; POLLOK, M. L. Prediction accuracy of body density: lean body weight, and total body volume equations. MedSci Sports. 1977;9(4):197-201.