UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO PARA IPHONE E IPAD PARA ACESSO A INFORMAÇÕES MÉDICAS EM UM HOSPITAL PERVASIVO NO ÂMBITO DO PROJETO CLINICSPACES

TRABALHO DE GRADUAÇÃO

Gustavo Pereira Guerra

Santa Maria, RS, Brasil

2010

DESENVOLVIMENTO DE UM APLICATIVO PARA IPHONE E IPAD PARA ACESSO A INFORMAÇÕES MÉDICAS EM UM HOSPITAL PERVASIVO NO ÂMBITO DO PROJETO CLINICSPACES

por

Gustavo Pereira Guerra

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Prof Giovani Rubert Librelotto

Trabalho de Graduação N. 308

Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UM APLICATIVO PARA IPHONE E IPAD PARA ACESSO A INFORMAÇÕES MÉDICAS EM UM HOSPITAL PERVASIVO NO ÂMBITO DO PROJETO CLINICSPACES

elaborado por
Gustavo Pereira Guerra

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

COMISSÃO EXAMINADORA:

Prof Giovani Rubert Librelotto
(Presidente/Orientador)

Profª Giliane Bernardi (UFSM)

Profª Marcia Pasin (UFSM)

Santa Maria, 09 de Dezembro de 2010.

"The cake is a lie."
— DO JOGO PORTAL

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, Milton e Vilma, ao meu irmão, Alex os quais sempre me apoiaram e incentivaram. Ao professor Giovani pela oportunidade deste trabalho e as professoras Marcia e Giliane por partiparem da banca de avaliação. Aos colegas de curso de Ciência da Computação; Árton, Breno, Ivan, Dewes pelos trabalhos conjuntos e atividades extra-classe. Por fim, agradeço a todos que de uma forma ou outra, contribuíram para a minha formação acadêmica ou fizeram parte dela.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM APLICATIVO PARA IPHONE E IPAD PARA ACESSO A INFORMAÇÕES MÉDICAS EM UM HOSPITAL PERVASIVO NO ÂMBITO DO PROJETO CLINICSPACES

Autor: Gustavo Pereira Guerra
Orientador: Prof Giovani Rubert Librelotto
Local e data da defesa: Santa Maria, 09 de Dezembro de 2010.

O objetivo desse trabalho consiste em criar um aplicativo na área de saúde que sirva de modelo para uma futura análise na criação de interfaces de usuário para dispositivos móveis, a serem utilizadas no projeto ClinicSpace. Esse aplicativo deverá permitir o acesso a informações referentes a pacientes cadastrados em um sistema de carater médico e também permitir o acesso, edição e exclusão de compromissos de um médico em sua agenda. Para o armazenamento de informações médicas utilizou-se o serviço Google Health, que usa como EHR para organizar seus dados o padrão Continuity of Care Record (CCR). Para o armazenamento de compromissos foi usado o serviço Google Calendar. Os dispositivos móveis escolhidos para serem utilizados na implementação, foram o iPhone e o iPad, smartphone e tablet da Apple. Aplicativos para estas plataformas são desenvolvidos utilizando-se a linguagem de programação Objective-C em conjunto com as ferramentas disponibilizadas pelo kit de desenvolvimento de software da Apple.

Palavras-chave: Aplicações móveis; iPhone; iPad; Google Health; objetive C; ambiente hospitalar; acessibilidade.

ABSTRACT

Trabalho de Graduação
Undergraduate Program in Computer Science
Universidade Federal de Santa Maria

DEVELOPMENT OF AN APPLICATION FOR IPHONE AND IPAD TO ACCESS MEDICAL INFORMATION IN A PERVASIVE HOSPITAL IN THE AMBIT OF CLINICSPACE'S PROJECT

Author: Gustavo Pereira Guerra
Advisor: Prof Giovani Rubert Librelotto

The purpose of this work is to create a medical application that serve as a model for a future analysis on the creation of user interfaces for mobile devices, which will be used on ClinicSpace project. This applicative should allow access to information of patients registered in a medical system as well, allow acess, creation, updating and exclusion of events to a medic calendar. For the data storage of medical information was used the Google Health service. It uses as EHR system the CCR standard. For the data storage of the calendar was used the Google Calendar Service.
The mobile devices chosen to store the applicative developed are the Apple's iPhone and iPad. Applications for this platform are created using the Objective-C programming language and the tools of the Apple SDK.

Keywords: Mobile Applications, iPhone, iPad, GoogleHealth, Objetive C, hospital environment, accessibility.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI	American National Standards Institute
API	Application Programming Interface
ASTM	American Society for Testing and Materials
CCR	Continuity of Care Record
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
EHR	Electronic Health Record
HL7	Health Level Seven
HTTP	Hypertext Transfer Protocol
ID	Identification
iOS	iPhone Operating System
JME	Java Micro Edition
MacOS	Mac Operating System
MIDP	Mobile Information Device Profile
MVC	Model-View-Controller
PDA	Personal Digital Assistants
SDK	Software Development Kit
SMS	Short Message Service
UIKit	User Interface Kit
URL	Uniform Resource Locator
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A computação Pervasiva (WEISER, 1991) tem por objetivo tornar a computação imperceptivel ao usuário, fazendo com que este foque apenas na realização de suas atividades e não na utilização dos recursos e aplicações computacionais. Para isso, ela prevê a comunicação entre dispositivos, recursos e aplicações do ambiente de forma pró-ativa para auxílio ao usuário, sem que este tenha que se preocupar com instalação ou configuração. A proposta de computação pervasiva vem ao encontro das necessidades da área da saúde, onde os profissionais não podem perder tempo de atendimento ao paciente na utilização de sistemas convencionais, pois necessitam de informações relevantes em tempo real para tomada de decisões.

A maioria dos sistemas atuais de registro eletrônico de informações do paciente não são desenvolvidos tendo como foco a visão do profissional da saúde (médicos e clínicos) e sim na abordagem de gerência e administração (VICENTINI; MACHADO; AUGUSTIN, 2009). Por outro lado o Grupo de Sistemas de Computação Móvel da UFSM, através do projeto ClinicSpace: auxílio às tarefas clínicas em um ambiente hospitalar do futuro baseado em tecnologias da Computação Ubíqua/Pervasiva, vem prototipando ferramentas que visam auxiliar a gerência de informações dentro de hospitais (FERREIRA et al., 2009). Estas ferramentas devem ser inseridas no contexto móvel para uma maior aceitação dos usuários finais e, nesse sentido, este trabalho propõe o estudo de métodos que contribuam para a criação destas interfaces e aplicativos.

Dentre os dispositivos móveis existentes atualmente, os mais desejados por usuários são o *smartphone* iPhone(APPLE, 2010a) e o *tablet* iPad da Apple(APPLE, 2010b). Devido a suas características e funcionalidades inovadoras, hoje são populares também entre os desenvolvedores. Por essas razões esses dispostivos serão a plataforma alvo do aplicativo implementado neste trabalho.

1.1 Objetivos

1.1.1 Objetivo Geral

Este projeto possui como objetivo geral estudo e uso de métodos existentes na criação de interfaces visando facilitar a Interação Homem-Computador (HUMAN-COMPUTER INTERACTION, 2010) presente em dispositivos móveis de última geração, criando um protótipo de aplicativo que venha a ser implementado futuramente em um ambiente hospitalar.

1.1.2 Objetivos Específicos

Como objetivos específicos podem ser citados:

- Análise e aplicação das técnicas de design e criação de interfaces para dois dispositivos móveis específicos, o Apple iPad e o Apple iPhone, propondo a criação de uma interface intuitiva para o usuário final;
- Desenvolvimento de uma aplicação para iPhone e iPad focada na área de gerenciamento hospitalar que acesse informações de pacientes em um registro eletrônico de pacientes (EHR) e que permita ao médico agendar compromissos, através do projeto desenvolvido pelo Grupo de Sistemas de Computação Móvel da UFSM (Gmob), intitulado ClinicSpaces, dando sequência ao trabalho desenvolvido anteriormente (APEL, 2009), onde foi criado um Web Service para integrar o ClinicSpace com o Google Health, através de uma aplicação para iPhone;
- Estudo de padrões de registro eletrônico de pacientes;
- Integração da aplicação desenvolvida aos *web services* Calendar e Health do Google.

1.2 Contribuição do trabalho

O trabalho tem como objetivo criar uma aplicação que de acesso rápido a informações hospitalares à profissionais da saúde. Integrando os dispositivos móveis dentro do contexto do hospital, possibilitando uma maior familiaridade e auxílio a estes profissionais no uso destas ferramentas.

1.3 Estrutura do texto

Este texto está estruturado da seguinte forma: O Capítulo 2 apresenta a revisão bibliografica do trabalho, detalhando quais são as plataformas alvo do trabalho, quais os padrões de registro eletrônico de pacientes, linguagem de programação e ferramentas usadas e exemplos de aplicativos. O Capítulo 3 descreve a modelagem, requisitos e implementação da aplicação desenvolvida. No Capítulo 4 é descrito uma avaliação da aplicação desenvolvida, e por fim, no Capítulo 5 são realizadas as conclusões finais e possíveis trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Nos últimos anos, a popularidade dos dispositivos móveis tem crescido a ponto de tornarem-se acessórios comuns e totalmente integrados ao dia-a-dia do consumidor. Suas funcionalidades também têm crescido exponencialmente, como também a miniaturização e a redução de peso desses dispositivos. Hoje em dia, é comum para a maioria dos dispositivos móveis incluírem funções de telefone, agenda de compromissos, alarme, câmera entre outros. Ao mesmo tempo, tem se tornado cada vez mais importante aprender a como avaliar seu uso, bem como desenvolver interfaces para estas novas funcionalidades (EISENSTEIN; VANDERDONCKT; PUERTA, 2000).

Convém citar que a computação móvel aumentou dramaticamente a dificuldade e complexidade de desenvolvimento de Interfaces de Usuário (UI) pelo fato de forçar os desenvolvedores a acomodar uma grande variedade de dispositivos e contextos de uso. Uma simples mudança nos requisitos da interface pode induzir diversas modificações importantes no código final de uma aplicação (LEE; GRICE, 2004).

Apesar das dificuldades envolvidas na criação de interfaces para dispositivos móveis, a sua mobilidade e a facilidade de acesso a informações apresenta uma grande aportunidade para a integração desses dispositivos no dia-a-dia de muitos ambientes de trabalho, em específico a hospitalar.

Visando criar um aplicativo para essa área. Este capítulo esclarecer os principais conceitos explanados no capítulo de introdução e as ferramentas usadas no objetivo desse trabalho.

2.1 Programação para iPhone e iPad

Nesta seção serão apresentados os dispostivos que são o alvo do aplicativo desenvolvido neste trabalho, bem como as ferramentas e a linguagem de programação usados no

desenvolvimento do aplicativo.

2.1.1 iPhone

O iPhone é o *smartphone* desenvolvido pela Apple (APPLE, 2010a). Está em sua quarta geração e possui funções como *player* de mp3, câmera, acelerômetros e tela multitoque. Por possuir estes recursos inovadores, desenvolver aplicativos para iPhone é um grande desafio, pois exige a criatividade do programador no uso desses recursos, como também a necessidade de fazer aplicativos úteis e inovadores para agradar aos usuários desta plataforma.

A criação de aplicações para este dispositivo exige o conhecimento da plataforma de desenvolvimento, pois utiliza linguagens e ferramentas diferentes das usadas normalmente para a programação em Windows e Linux, e outros dispositivos móveis que utilizam, na maioria a plataforma JME. O iPhone usa como sistema operacional o iOS (APPLE, 2010c) que é o sistema usado pelos dispositivos móveis da Apple, como o iPad e iPod Touch. A programação para esses dispositivos exige o conhecimento do *framework* Cocoa Touch, que fornece uma camada de abstração do sistema permitindo ao programador usar todos os recursos de multimídia e as funções de toque e acelerômetros desses dispositivos.

Para a criação de aplicativos para iPhone a Apple disponibiliza um kit de desenvolvimento de *software* (SDK, somente para MacOS X) que é baixado do portal do desenvolvedor iOS (APPLE, 2010d). Neste portal é possível encontrar a documentação do funcionamento do sistema e exemplos de como utilizar a API (somente em inglês). Por ser um dispositivo popular, hoje o iPhone possui grande número de livros que ensinam como criar aplicativos para esta plataforma.

A instalação e compra de aplicativos gratuitos e pagos para sistemas que usam iOS é feito pelo portal *App Store* da Apple.

2.1.2 iPad

O iPad é o dispositivo em forma de *tablet* desenvolvido pela Apple (APPLE, 2010b). Foi lançado em 2010 e é uma mistura de *notebook* e *smartphone*. Possui uma tela de 9,5 polegadas e recursos como internet sem fio, *Bluetooth*, acelerômetros e tela multitoque.

Utiliza o mesmo sistema operacional do iPhone, o iOS, sendo assim o desenvolvimento de aplicativos é dado de forma semelhante. Como utilizam o mesmo sistema,

aplicativos desenvolvidos para iPhone rodam no iPad utilizando o espaço de tela utilizado pelo iPhone ou esticando a tela do aplicativo para preencher a tela. Mas o ideal é criar aplicativos especialmente para iPad, fazendo assim o uso das vantagens que o dispositivo possui como a tela maior e os recursos de melhores hardware em relação ao iPhone.

2.1.3 Linguagem Objective-C

Objective-C é a linguagem de programação usada na programação para iPhone e iPad. É uma linguagem de programação reflexiva orientada a objetos que não implementa chamadas de métodos mas troca de mensagens. A linguagem é um *superset* da linguagem ANSI C (ANSI, 2010). Ou seja, qualquer código em C pode ser compilado em um compilador do Object-C. Atualmente, a linguagem está na versão 2.1. A sintaxe da linguagem foi criada baseada em Smalltalk (SMALLTALK, 2010), linguagem criada nos anos 70 como intuito de sustentar o "novo mundo"da computação exemplificado pela "simbiose homem-computador"(KAY, 2010). Em Objective-C tudo é objeto. Desde inteiros, *strings* e vetores. Todos herdam características da classe padrão da linguagem, a classe NSObject.

As figuras 2.1 e 2.2 mostram uma comparação entre as chamadas de métodos em C e Objective-C.

```
1. Fraction *frac = alloc();
2.
3. // definindo valores
4. setNumerator(frac,1);
5. setDenominator(frac,3);
```

Figura 2.1: Chamada de métodos em C.

```
1. Fraction *frac = [[Fraction alloc] init];
2.
3. // definindo valores
4. [frac setNumerator: 1]; //instância método e valor
5. [frac setDenominator: 3];
```

Figura 2.2: Troca de mensagem em Objective-C.

Outras particularidades são os tipos adicionados e diferenças adotadas pela linguagem em relação a C:

- #import: diretiva que equivale ao #include de C, mas não necessita adicionar guar-

diões (#ifndef #define #endif) para evitar a multi-definição da interface.

- id: ponteiro para um objeto qualquer. Permite a criação de objetos dinamicamente tipados.
- nil: equivale a NULL de C. É um ponteiro para objetos nulos.
- YES e NO: equivalentes a TRUE e FALSE em C. São os tipos *booleanos*.

Em Objective-C também existe o *header file* (.h), arquivo onde é declarado os protótipos do que será implementado na classe. As classes são chamadas de @*interface*, diferentemente do nome *class* usadas em outras linguagens. Na interface é onde declaram-se variáveis do objeto e seus métodos; estáticos, (iniciados com um + durante sua declaração) ou da instância (declarados com um -). A implementação da classe é feita em um arquivo com a extensão (.m) diferentemente do arquivo (.c) usado na linguagem C.

A figura 2.3 mostra um exemplo de implementação de uma classe e seus métodos em Objective-C.

```
1. // Programa Cão
2. // aqui comecam as diretrizes do pre-processador
3. #import <stdio.h>
4. #import <objc/Object.h>
5.
6. // aqui comeca a definicao das interfaces
7. @interface Cao: Object
8. -(void) au_au;
9. @end
10.
11. // aqui comecam as implementacoes dos métodos
12. @implementation Cao
13.
14. -(void) au_au {
15.     printf ("Au Au!\n");
16. }
17. @end
18.
19. // aqui comeca o programa
20. int main( int argc, const char *argv[] ) {
21.         Cao *meuCao;
22.     meuCao = [Cao alloc];
23.     meuCao = [meuCao init];
24.     [meuCao au_au];
25.     [meuCao free];
26.     return 0;
27. }
```

Figura 2.3: Exemplo de código em Objective-C.

2.1.4 Apple Software Development Kit

Para o desenvolvimento de aplicativos para o MacOS, iPhone e iPad a Apple disponibiliza um kit de desenvolvimento que pode ser baixado pelo portal do desenvolvedor da Apple(APPLE, 2010d). O kit, além de conter todos os *frameworks* necessários para criação de aplicativos para MacOS e iOS, possui as ferramentas: *Xcode*, é o ambiente de desenvolvimento integrado, *Interface Builder*, é o aplicativo para criação de interfaces gráficas, e o *iPhone Simulator*, para testar aplicações desenvolvidas para iPhone ou iPad.

Nas subseções seguintes são detalhadas as funções de cada um destes aplicativos.

2.1.4.1 Xcode

O *Xcode* é o ambiente de desenvolvimento integrado para aplicações oficial da Apple. Nele é possível criar aplicações para MacOS, iPhone e iPad, iPod, como também aplicações web em Java. O ambiente possui funcionalidades como debbuger, refatoração, auto completar código e suporte a programas de versionamento. O *Xcode* utiliza como compilador uma versão modificada do GNU Compiler Collection (GCC) para que seja possível gerar binários para o *hardware* dos computadores Apple e para os processadores ARMs utilizados nos dispositivos iPhone e iPad.

No Xcode é onde se localizam todos os arquivos relacionados ao projeto como: os arquivos de código (.h e .m), os de interface (.xib .nib) e os *resources* que podem ser arquivos como imagens ou sons.

A figura 2.4 mostra uma tela do aplicativo rodando em um sistema MacOs.

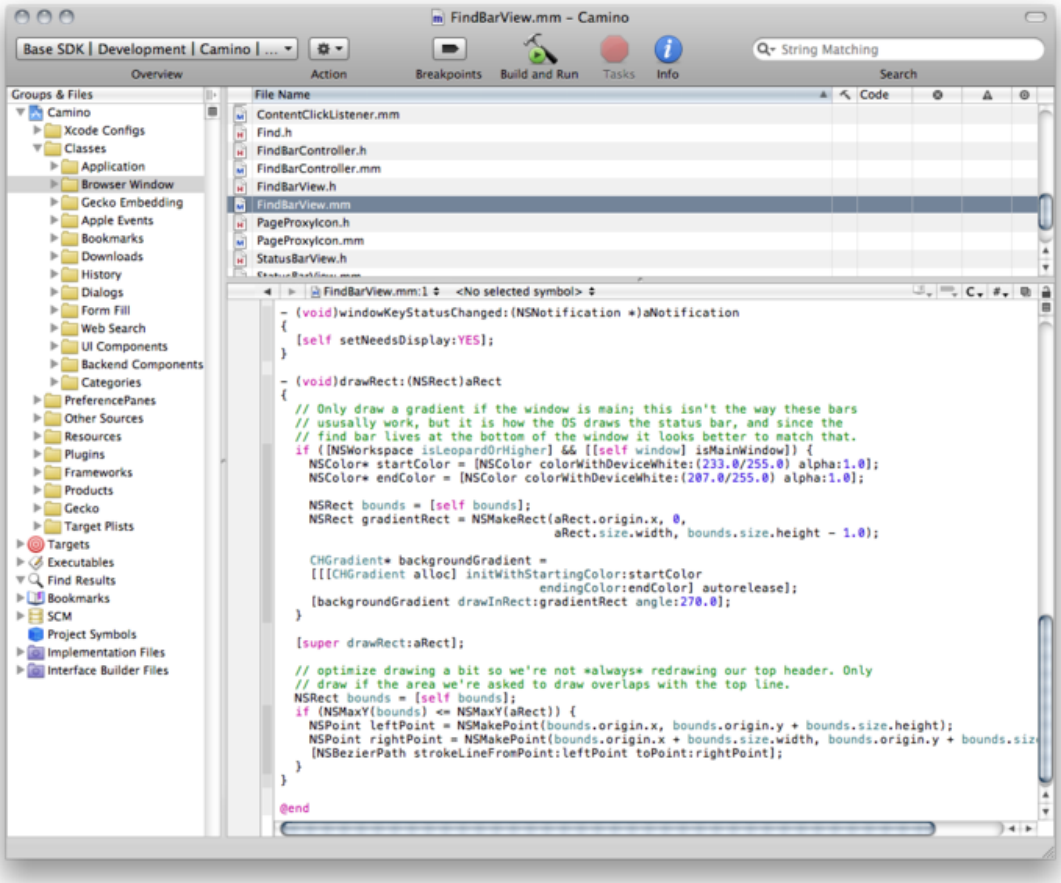

Figura 2.4: Tela do Xcode.

2.1.4.2 Interface Builder

O *Interface Builder* é o programa o qual se cria as interfaces gráficas dos aplicativos. Nele é possivel gerenciar e adicionar componentes visuais como *textfields*, *labels*, botões, entre outros que estarão presentes em uma tela do aplicativo. No *Interface Builder* é onde são feitas as ligações entre os arquivos de interface (.xib .nib) e as classes criadas no *Xcode*. Além de ter que referênciar a classe, o programador precisa definir no programa qual a variável de um objeto criado em código será responsável pelos eventos da interface.

Para fazer essas ligações o desenvolvedor clica no componente já adicionado a *view* e arrasta o mouse em direção ao objeto *File's Owner* que é onde fica definido qual é a classe base usada para a interface. Se existe um ou mais componentes da mesma classe o programa mostra quais são as opções para conectar. Essa forma de ligação se mostra muito útil, não precisando definir em código qual componente está relacionado a que variável da classe.

A figura 2.5 mostra como funciona a ligação e a figura 2.6 mostra uma tela do aplicativo em execução.

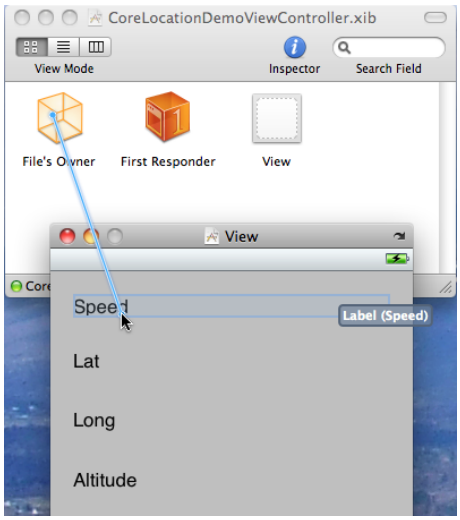

Figura 2.5: Exemplo de ligação de um componete da *view* com o código.

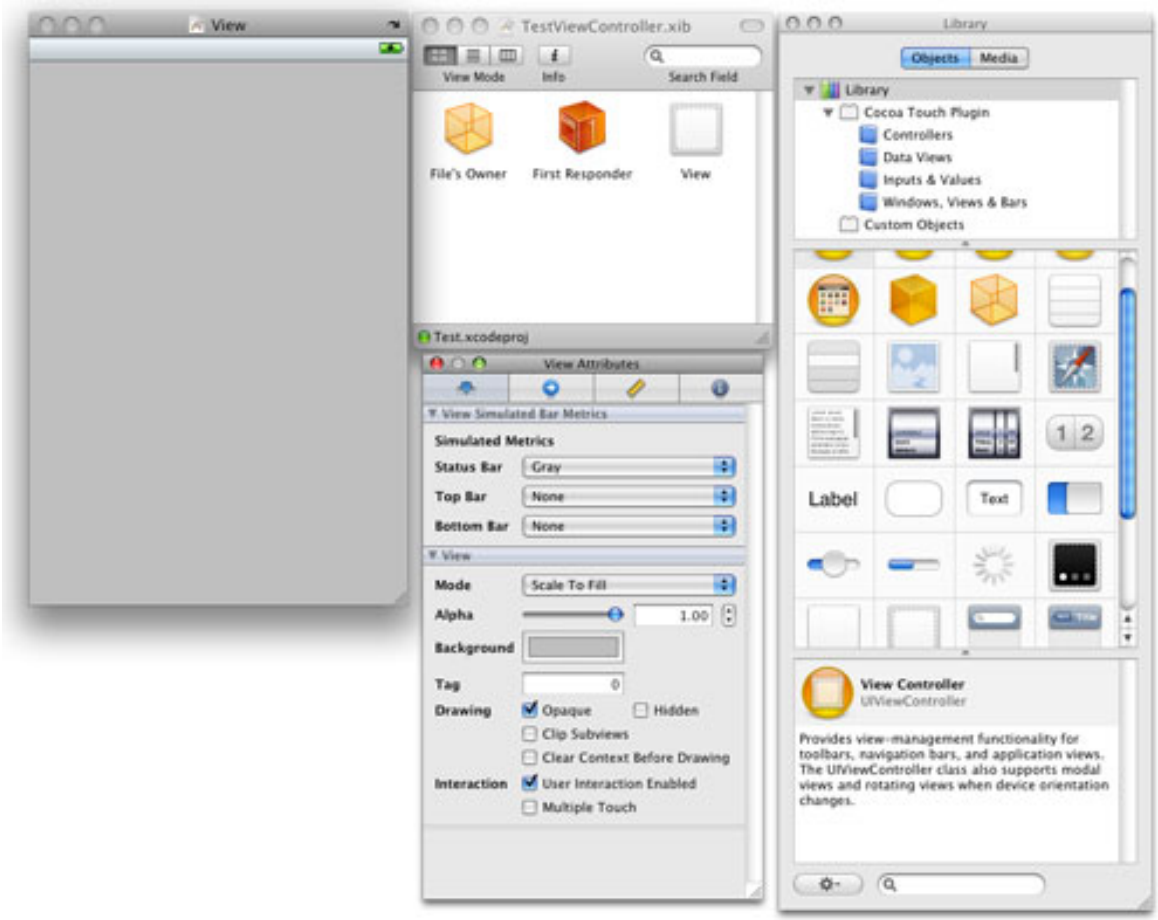

Figura 2.6: Janelas do Interface Builder.

2.1.4.3 *iPhone Simulator*

O *iPhone Simulator* é o programa da SDK onde são testados aplicativos desenvolvidos para iPhone, iPad e iPod Touch. Nele é possível saber como será o comportamento do aplicativo antes dele ser instalado em um desses dispositivos.

A figura 2.7 apresenta uma tela do aplicativo.

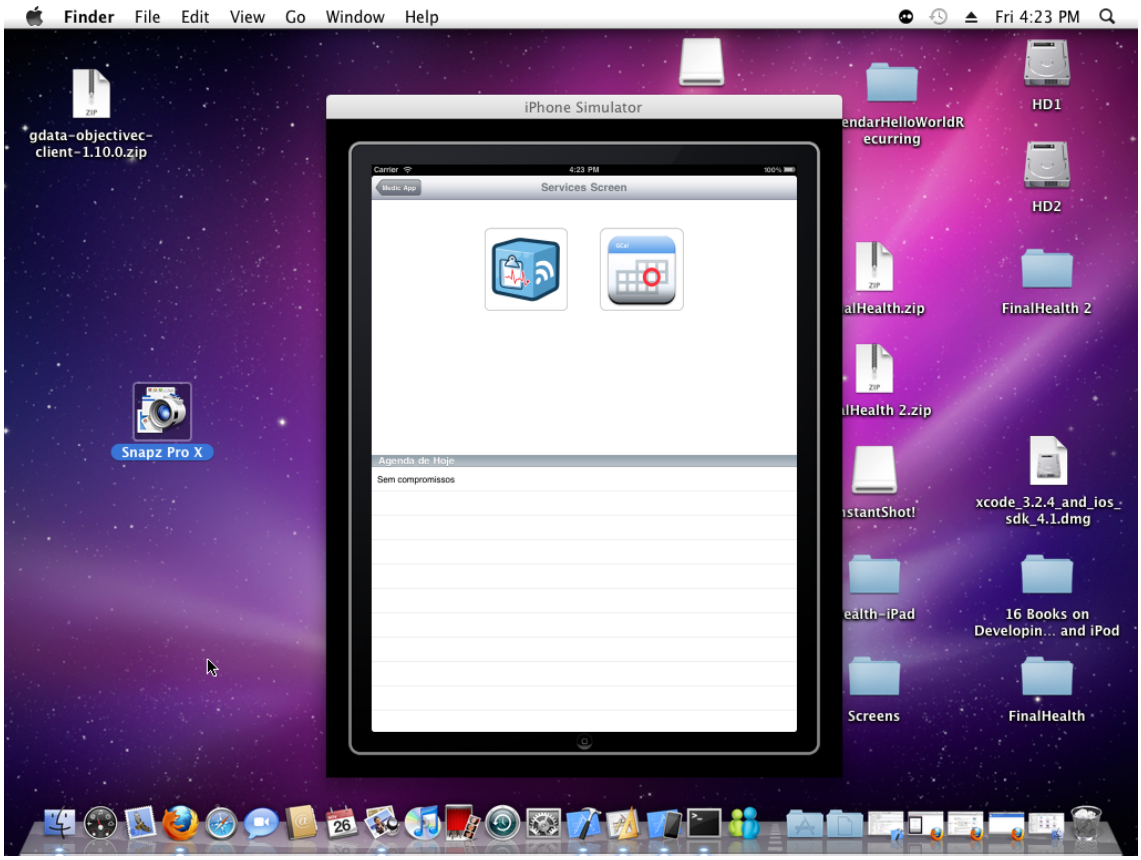

Figura 2.7: iPhone Simulator executando o aplicativo desenvolvido para iPad.

2.2 O Projeto ClinicSpace

Tendo como motivação auxiliar a associação da área médica com a computação móvel, o projeto ClinicSpace, desenvolvido pelo Grupo de Sistemas de Computação Móvel (Gmob, UFSM), propõe a utilização de tecnologias ubíqua e pervasiva para tornar a computação imperceptível e de fácil acesso ao usuário. O projeto tem como objetivo a construção de ferramentas intuitivas e personalizadas que auxilie o profissional de saúde na realização de suas atividades do dia-a-dia e não na administração de um hospital.

Atualmente, os clínicos são os responsáveis por captar informações do ambiente e introduzi-las nos sistemas informatizados, gerando uma série de problemas bem conhe-

cidos, destacando-se o alto grau de rejeição dos sistemas computacionais pelos clínicos em ambientes hospitalates (SILVA; AUGUSTIN, 2009). A introdução de tecnologias de acesso móvel visa mudar este comportamento e dar uma maior conforto ao clínico na realização de suas tarefas.

Durante a elaboração do projeto ClinicSpaces foram analisados um conjunto mínimo de tarefas clínicas publicados na pesquisa de Laerum e Faxvaag (LAERUM; FAXVAAG, 2004), o qual foi realizada com profissionais da saúde (principalmente médicos) para a identificação e definição das principais tarefas clínicas executadas pelos profissionais desta área. Este estudo propõs um conjuto de vinte e quatro atividades realizadas em ambientes hospitalares, dos quais onze foram destacadas por médicos consultados pelo projeto ClinicSpace como tarefas mínimas executadas com maior frequência no dia-a-dia de trabalho de um clínico (SILVA; AUGUSTIN, 2009). Estas tarefas estão listadas abaixo:

1. Revisar os problemas do Paciente;
2. Procurar informações específicas no registro do paciente;
3. Obter o resultado de novos testes e investigações;
4. Adicionar notas diárias sobre condições do paciente;
5. Requisitar análises laboratoriais;
6. Requisitar exames de vídeo/imagem (raio-x, tomografia, etc);
7. Obter resultados laboratoriais;
8. Obter resultados de exames de vídeo/imagem;
9. Requisitar tratamento;
10. Escrever prescrições médicas;
11. Registrar códigos para diagnóstico de procedimentos executados.

Visando auxiliar as tarefas 1, 2, 3 e 4, este trabalho propõe a criação de um aplicativo para dispositivo móvel que permite o acesso aos dados de pacientes disponível em um serviço de EHR (Electronic Health Record). Para fins de implementação deste trabalho,

foi usado como EHR e armazenamento dos dados o serviço Google Health do Google, o qual usa como padrão de EHR o *Continuity of Care Record* (CCR). O objetivo do aplicativo é que ele sirva de análise para futuras aplicações que venham a ser usados pelo projeto ClinicSpace e seu EHR.

A Figura 2.8 mostra a arquitetura do projeto ClinicSpace e seus módulos. O módulo 11, Interface do Usuário é o escopo onde encaixam-se os aplicativos de acesso móvel. A Figura 2.8 também apresenta o EHR do projeto: Pervasive Healthcare System (pEHS).

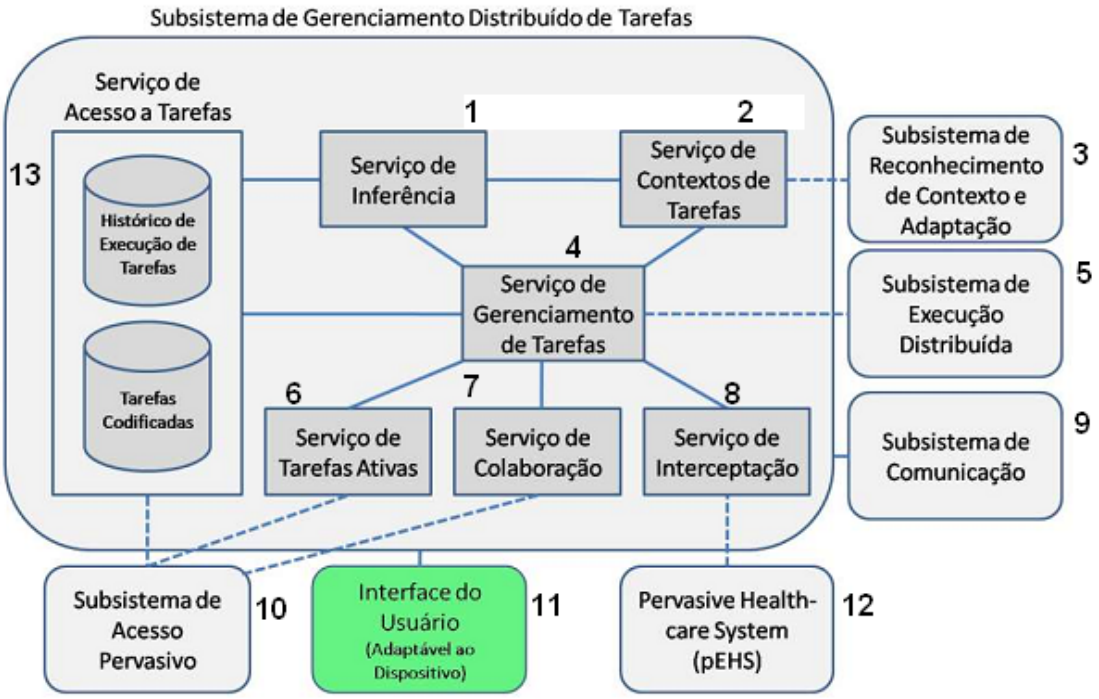

Figura 2.8: Arquitetura do projeto ClinicSpace.

2.3 Registro eletrônico de pacientes

O Registro eletrônico de pacientes também conhecido como EHR (*Electronic Health Record*) é um meio eletrônico de armazenar e trocar informações sobre saúde de pacientes ou de uma população. Nesse registro é possível acompanhar informações adicionadas a cada visita médica ao longo da vida de um paciente. O registro pode conter informações como alergias, patologias, doenças já tratadas, tratamento em progresso, vacinas, imagens de tomografias, raio-x entre outras informações de caráter médico. Com isso o registro eletrônico de pacientes tem a capacidade de concentrar informação e automaticamente gerar relatórios clínicos completos sobre pacientes auxiliando assim médicos a tomar suas decisões baseados em evidencias e também garantindo uma maior agilidade

em seu trabalho.

Atualmente existem vários padrões de Registro eletrônico de pacientes. Os formatos mais usados atualmente são os padrões criados pelo Instituto *Health Level* 7 (HL7, 2010) e o *Continuity of Care Record* (ASTM, 2003) criado pela *American Society for Testing and Materials* (ASTM). Visando integrar esse padrões o ASTM e o HL7 criaram em hamornia o padrão *Continuity of Care Document* (CONTINUITY OF CARE DOCUMENT: CHANGING THE LANDSCAPE OF HEALTHCARE INFORMATION EXCHANGE, 2010) o qual consiste em uma integração entre padrão o HL7 *Clinical Document Architecture* (HL7, 2004) e o ASTM *Continuity of Care Record*. Com o estudo desse novo padrão, foi avaliado que não seria viável no momento a criação de um parser para a conversão entre os formatos CCR e CDA, pois a conversão exigia o conhecimento de todos os códigos de doenças usados pelos dois formatos. Por isso esta parte foi retirada do objetivo do trabalho apresentado em seminário anterior.

A próxima subseção apresenta o padrão CCR, que é usado pelo Google Health, um dos serviços usados pela aplicação desenvolvida.

2.3.1 CCR

O *Continuity of Care Record* (CCR) (ASTM, 2003) é um padrão de registro eletrônico de pacientes. Foi criado em 2003 pela ASTM (*American Society for Testing And Materials*) com o objetivo de facilitar a troca de informações médicas entre sistemas, clínicas e hospitais. O registro possui um conjunto dos mais relevantes dados administrativos, demográficos e informações clínicas sobre a saúde do paciente, cobrindo um ou mais encontros médicos, contendo as informações médicas necessárias para a continuidade de um tratamento em outros centros médicos.

O padrão define um formato XML onde usa-se um vocabulário de códigos de vários padrões médicos para a definição de doenças, alergias entre outras informações, oferecendo uma interface comum de comunicação entre os sistemas. O arquivo é dividido em três componentes principais, que são:

1. Header: identifica o formato do registro e o tempo em que ele foi criado. Possui os campos: <CCRDocumentObjectID>, <Language>, <Version>, <DateTime>e <Patient>.

2. Body: contêm os detalhes do histórico médico. É composto por: <FunctionalSta-

tus>, <Problems>, <SocialHistory>, <Alerts>, <Medications>, <Immunizations>, <VitalSigns>, <Results> e <Procedures>.

3. Footer: consiste das informações demográficas sobre o paciente, como nome, gênero e data de nascimento. É representado por um único campo <Actor> que possui subcampos para informar os dados demográficos citados acima.

Na figura 2.9 há uma representação simplificada da árvore XML do padrão CCR usado pelo Google Health.

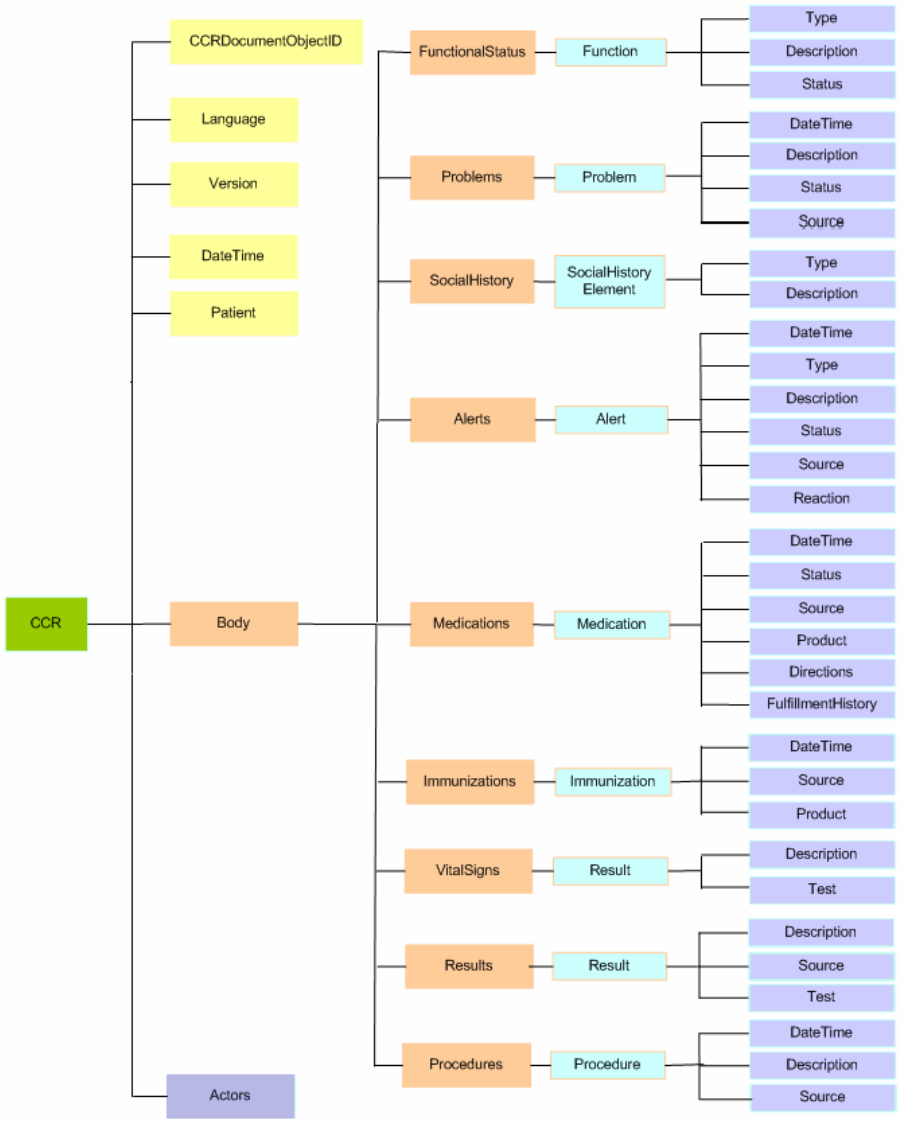

Figura 2.9: Árvore do padrão CCR usado pela API do Google Health.

2.3.2 Google Health

Lançado em 2008, o GoogleHealth (GOOGLE HEALTH,2010) é um serviço do Google onde as pessoas podem gerenciar e armazenar informações médicas próprias e de

outras pessoas desejadas. A ferramenta possibilita que o usuário compartilhe suas informações com outras pessoas ou também com médicos, facilitando assim o acesso do mesmo as informações arquivadas, e consequentemente, na realização de um diagnóstico mais rápido e preciso. O serviço se baseia no modelo de armazenamento de informações médicas chamado Continuity of Care Record (CCR) que já foi explicado na seção anterior.

Nesta ferramenta do Google é possível armazenar dados como histórico médico completo e dados clínicos gerais, como alergias, medicamentos em uso, cirurgias já realizadas, resultados de exames, vacinas,etc.. Para auxiliar o usuário, já se encontram cadastrados no serviço diversas informações referentes às citadas acima, como por exemplo, ao inserir ao perfil que a pessoa tenha bronquite, é possível visualizar clicando em *overview* os sintomas, tratamento, causas, diagnóstico, prevenção, complicações da patologia.

Figura 2.10: Interface web de descrição de patologias do Google Health.

O sistema também verifica como é a interação entre medicamentos em uso, alergias cadastradas e patologias existentes, caso exista alguma contra indicação, a mesma é exibida em detalhes. Por fim, permite ao usuário o *upload* de arquivos com imagens digitais, facilitando a inserção de resultados de exames como, raios-x ou qualquer tipo de exame gráfico.

2.3.2.1 API Google Health

O Google disponibiliza uma API para desenvolvimento de possíveis aplicações clientes interessadas em acessar e modificar o conteúdo encontrado no serviço Google Health.

Estas APIs de dados oferecem um protocolo simples e padrão para ler e gravar dados no serviço web. A API está disponível para várias linguagens como: python (PYTHON, 2010), java (ORACLE, 2010) e também objective-C.

Por manipular informações de caráter médico, a API do Google Health obriga que o desenvolvimento seja realizado em uma *sandbox*, ou seja, um ambiente de testes, chamado H9 (GOOGLE HEALTH - H9, 2010).

Esta *sandbox* não possui tantas restrições para a realização de qualquer operação quanto a versão oficial. As diferenças entre as duas plataformas podem ser visualizadas na tabela 2.1. As aplicações desenvolvidas não poderão interagir com a plataforma real do Google Health antes de serem testadas dentro do H9. Tanto no H9 quanto diretamente no Google Health, o acesso a dados requer autorização.

Tabela 2.1: Tabela comparando o *Sandbox* Google H9 e o Google Health.

Critério	H9 Sandbox	Google Health
Url de Acesso	`https://www.google.com/h9`	`https://www.google.com/health`
As requisições da API precisam ser assinadas digitalmente?	Não	Sim
Onde deve ser requisitado um token AuthSub?	`https://www.google.com/h9/authsub`	`https://www.google.com/health/authsub`
Posso utilizar `http://localhost` como o parametro next do AuthStub?	Sim	Não
Qual o valor do parametro scope do AuthStub?	`https://www.google.com/h9/feeds/`	`https://www.google.com/health/feeds/`
Qual o nome do service para o login do cliente?	weaver	health

A troca de informações entre aplicativo cliente e o servidor de dados do Google Health é realizada através de requisições HTTP.

2.3.3 Google Calendar

Google Calendar (GOOGLE, 2010a) é uma aplicação web que serve como agenda de compromissos. É um serviço criado em 2006 pelo Google (GOOGLE, 2010b). Permite

a inclusão e organização de eventos, sendo possível que o usuário seja notificado dos seus compromisso via e-mail ou pop-ups no aplicativo na web. O Google disponibiliza uma API onde um programa cliente pode criar novos eventos, editar ou excluir eventos existentes, e consultar eventos que correspondam a critérios específicos.

2.3.3.1 API Google Calendar

Na API de dados disponibilizada pelo Google para o Calendar, diferentemente da API do Google Health, o serviço não impõe restrições quanto ao uso do serviço principal. Aplicações clientes podem acessar e usar os dados da mesma forma do serviço disponibilizado pela web. Como os dois serviços (Calendar e Health) usam a mesma biblioteca. Na próxima seção será apresentado o seu funcionamento.

2.3.4 Biblioteca Objective-C da API Google

Para a comunicação entre aplicações clientes e seus serviços, o Google disponibiliza uma biblioteca em objective-C para o acesso aos seus dados. A biblioteca consiste em de estruturas e métodos que fazem requisições em http e que retornam dados em forma de feeds em XML. Através dessas estruturas é possível acessar os dados retornados de uma requisição.

Exemplo de um requisição em objective-C:

```
ticket = [myCalendarService fetchFeedWithURL:feedURL
                                    delegate:self
                           didFinishSelector:@selector(agendaSearchTicket:finishedWithFeed:error:)];
```

Para usar a biblioteca o programador deve estar ciente da hierarquia e funções dos dados determinada pela API.

Abaixo é demonstrado como é a hierarquia desta API:

- Service: estrutura responsável por armazenar o login e senha do usuário.
- Ticket: estrutura responsável por uma requisição.
- Feed: estrutura que armazena uma lista de calendários ou perfis usados pelo usuário, contém uma lista de *Entries*.
- Entry: é o calendario ou perfil de uma única pessoa. Possui uma lista de *Events*.
- Event: armazena um compromisso do calendar.

Através destas estruturas é possivel o acesso e manipulação do dados obtidos em uma requisição. No momento a API de dados do Google não permite a criação de novos perfis de pacientes através de aplicações clientes no Google Health, apenas via *browser*.

2.4 Trabalhos Relacionados

Nessa seção são apresentados dois exemplos de aplicativos desenvolvidos para o dispositivo móvel iPhone, usados na área médica. Estes aplicativos possuem funções semelhantes ao aplicativo implementado neste trabalho, como também outros serviços. Para o armazenamento dos dados estes aplicativos usam soluções próprias diferentemente dos serviços gratuitos e do padrão de EHR (CCR) usados nesta implementação.

2.4.1 Aplicativo para iPhone usado no Hospital Israelita Albert Einstein

O hospital mais moderno da America Latina, o Albert Einstein em São Paulo utiliza com sucesso uma aplicação médica para iPhone chamada Einstein Mobile. O sistema do Einstein Mobile inclui acesso a informações de pacientes (resultados de exames, passagens, prescrições de medicamentos e contatos), cadastros médicos (CRM, especialidade e contatos), recurso de comunicação entre pacientes e médicos do corpo clínico do Einstein (via SMS, *e-mail* ou ligações), visualização da agenda cirúrgica e consulta integrada ao CID 10 (Classificação Internacional de Doenças).

O programa está disponível gratuitamente para download em: `http://itunes.apple.com/br/app/einstein-mobile/id337461205?mt=8`. Mas apenas médicos do hospital tem acesso as informações médicas no aplicativo. Na figura 2.11 é apresentada uma tela do aplicativo.

Figura 2.11: Tela do aplicativo Einstein Mobile.

2.4.2 TDO Mobile

O TDO Mobile é um aplicativo móvel proprietário criado pela TDO (The Digital Office for Endodontists) (TDO, 2010). Funciona em integração com TDO *Practice Management Software* que é um sistema de gerenciamento de dados clínicos para endodentistas. Possui as seguintes funcionalidades: Lista de pacientes, agenda, lista de médicos de referência, uso de medicamentos, visualização de relatórios e imagens.

O aplicativo pode ser baixado gratuitamente em: `http://itunes.apple.com/br/app/tdo-mobile/id309099931?mt=8#`.

Na figura 2.12 são apresentadas telas do aplicativo.

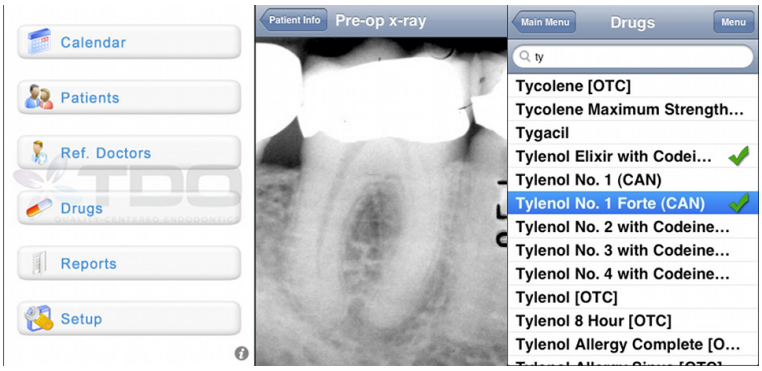

Figura 2.12: Telas do aplicativo TDO Mobile.

3 MODELAGEM

As seções seguintes mostram os requisitos e a modelagem de dados definidos no desenvolvimento do aplicativo para iPhone e iPad escolhido para estudo de caso deste trabalho.

3.1 Levantamento de Requisitos

Os requisitos são objetivos estabelecidos por clientes e usuários que definem as propriedades de um sistema. O conjunto de requisitos define a condição e a capacidade necessária que um software deve possuir para que o usuário possa resolver um problema ou objetivo. Tendo esse conceito, abaixo são definidos os objetivos e requisitos presentes na aplicação.

A aplicação tem como objetivo geral o acesso a informações de pacientes e da agenda médica de uma forma simples para o usuário. Visando esse objetivo geral, o aplicativo tem como objetivos específicos os seguintes itens:

- Permitir o acesso às informações através de uma conta do Google;
- Acesso a lista de pacientes;
- Acesso a detalhes de cuidados médicos de cada paciente;
- Consultar agenda de compromissos;
- Possibilitar a adição, alteração e exclusão de compromissos na agenda.

Partindo desses objetivos, os seguintes requisitos foram estudados e definidos para o funcionamento ideal da aplicação em um ambiente hospitalar.

- O usuário deverá ter acesso a internet através do seu dispositivo móvel;

- O usuário deve possuir uma conta no Google é ja ter acessado e aceitado as condições dos serviços Health e Calendar do Google via *browser*;
- Possibilitar que o usuário acesse o serviço de qualquer lugar através do seu dispositivo móvel.

Definido os requisitos e objetivos da aplicação, partiu-se para a modelagem de dados do trabalho.

3.2 Modelagem de Dados do Aplicativo

Antes de começar a programação para os dispositivos que usam o sistema iOS o desenvolvedor deve conhecer bem o funcionamento e os componentes oferecidos pela *framework Cocoa Touch* para a criação do seu projeto. O funcionamento dos aplicativos para esse sistema é baseado no controle de telas (*views*) e interrupções de sistema. Cabe ao programador a criação e gerenciamento dos dados representados, como também o tratamento das chamadas de sistema enquanto o programa está em execução.

A figura 3.1 mostra o ciclo de vida de um aplicativo para iOS segundo o guia de referência da Apple (APPLE, 2010e), mostrando a execução desde o princípio e algumas chamadas de sistema pré-definidas na arquitetura como:

1. *application:didFinishLauchingWithOptions*: onde chama-se a primeira tela do aplicativo;
2. *applicationWillResignActive*: ação a ser tomada quando a aplicação ficará em estado inativo ou quando o programa será finalizado;
3. *applicationDidEnterBackground*: ação a ser tomada antes do aplicativo ser executado em *background*.

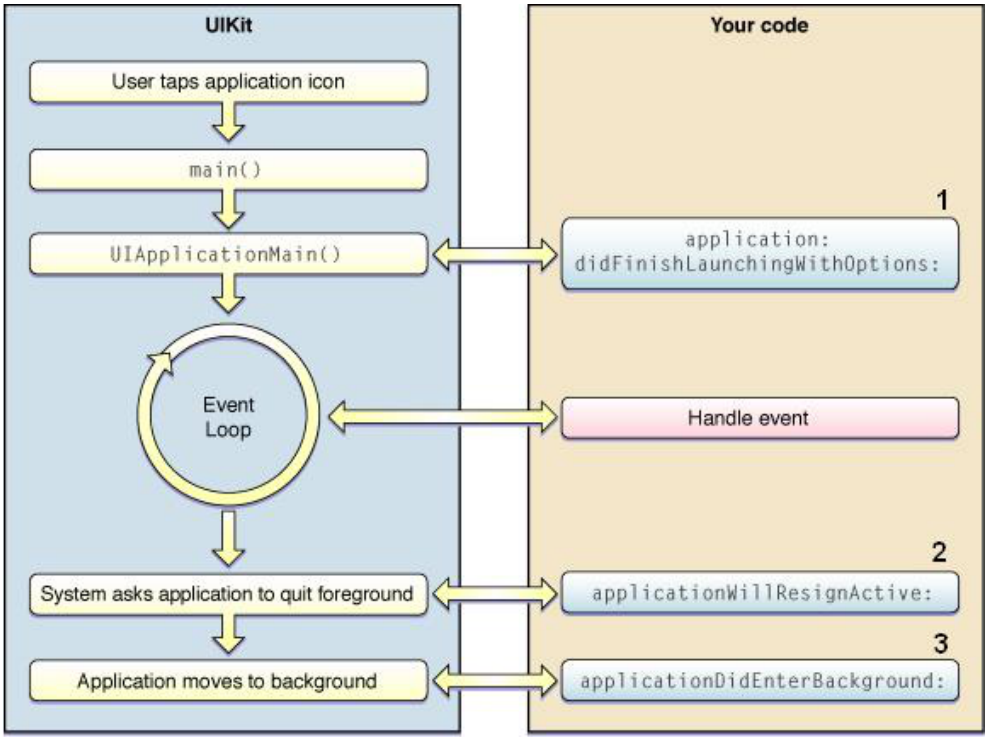

Figura 3.1: Ciclo de vida de um aplicativo no sistema iOS (Apple).

A programação para esta plataforma foca no uso da arquitetura de *software Model View Controller* (MVC) (XEROX-PARC, 1978-79) que apresenta um modelo de desenvolvimento onde separa-se os os arquivos de interface (*view*), o código que controla as ações de tela (*controller*) e os objetos de dados (*Model*). Essa arquitetura permite o desenvolvimento, teste e manutenção de código de forma mais independente.

Usando MVC no desenvolvimento, a criação de uma interface no *Interface Builder* exige uma classe derivada de uma classe do *framework* chamada *view controller*, que vai ser responsável pelas ações e eventos de componentes como; botões, campos de textos, entre outros, usados na tela.

Como o aplicativo desenvolvido usa a API de dados e armazenamento dos serviços do Google a modelagem de dados não necessitou ser projetada para o trabalho.

A SDK disponibiliza várias classes para o controle de telas, abaixo são apresentadas as mais importantes e as que foram usadas pelo aplicativo.

- *Navigation Controller*: É usada na criação de aplicativos com hierarquia de telas. Ela é responsável pelo gerenciamento e navegação entre as telas de um conjunto de *views* usadas pelo aplicativo. Quando usada é inserida uma barra no topo da tela do aplicativo, onde o usuário pode inserir um nome para a tela atual, botões para

avançar ou voltar a uma tela, entre outras funcionalidades desejadas. A troca de telas usando essa estrutura é feita através dos métodos *push* e *pop* da classe;

- *View Controller*: É usada na criação de interfaces genéricas. Permitindo a inserção de qualquer componente criado ou disponibilizado pela SDK;
- *Tab Bar Controller*: É usado para gerenciar um conjunto de telas de um único arquivo de interface. As telas são acessadas através de ícones em uma barra localizada no inferior da *view*;
- *Table View Controller*: É usado para mostrar dados em forma de tabela. O programador é responsável pela personalização da tabela e suas células como também pela a codificação dos métodos pré-definidos pela classe para a inserção dos dados;
- *Image Picker Controller*: É usado para a exibição e navegação em arquivos de vídeos e imagens.

Apresentado estes conceitos, as classes necessárias para a implementação das telas e controladores do aplicativo estão descritas abaixo e a figura 3.2 apresenta o diagrama de classes do trabalho desenvolvido.

1. *AppDelegate*: é criada pelo projeto do *Xcode* possui um objeto *navegation controller* o qual adiciona e chama a primeira tela do aplicativo no método *application:didFinishLauchingWithOptions*;
2. *RootViewController*: seu escopo é criado automaticamente pelo projeto do *Xcode* para controlar a primeira tela do aplicativo. A classe herda as propriedades de uma *view controller*. É responsável pela tela de *login* do usuário. Possui os campos de *e-mail* e senha, além de um botão para realização do acesso. É responsável por adicionar e chamar a tela de serviços no controlador de navegação do aplicativo;
3. *AppSelectionView*: classe derivada de uma *view controller*. É responsável pela tela de seleção. Que contém dois botões para a chamada das telas de serviços; Google Health e Calendar. Também apresenta no inferior da tela uma tabela com os compromissos do dia;

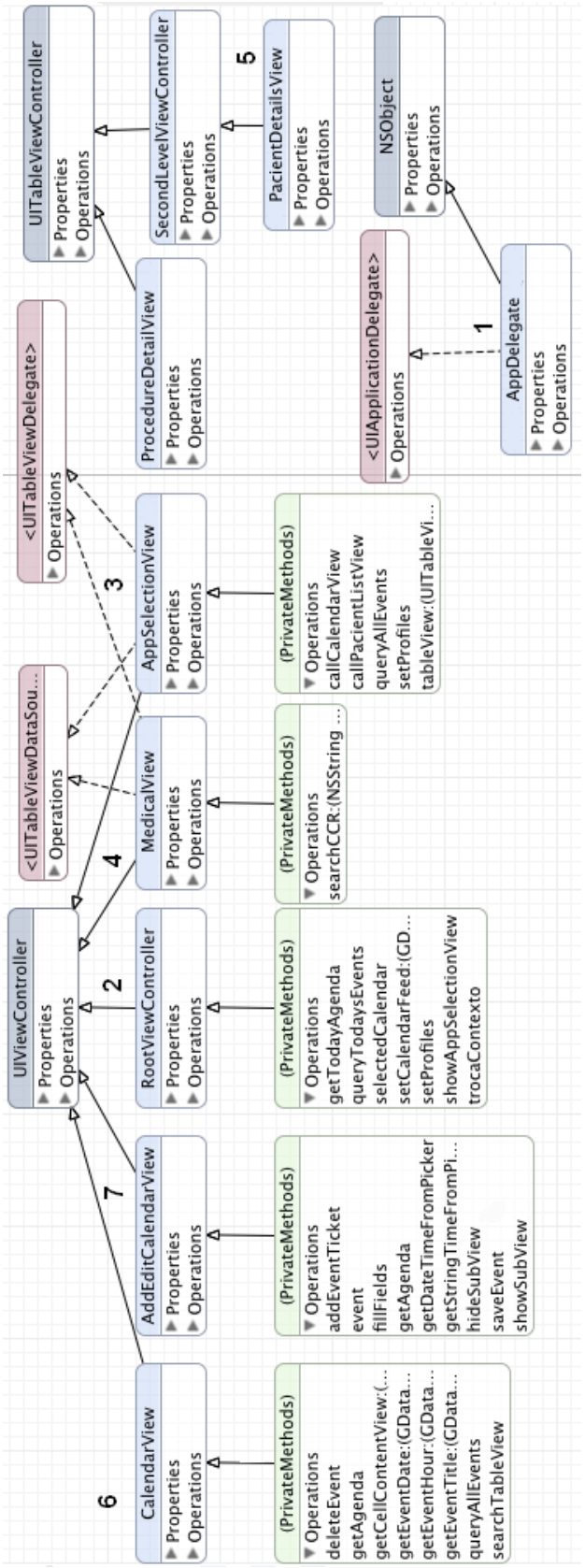

Figura 3.2: Diagrama de classes do projeto.

4. *MedicalView*: classe derivada de uma *table view controller*. Responsável pela requisição e apresentação da lista de pacientes em forma tabular. Ao clicar no nome de um paciente chama a tela com os detalhes médicos do mesmo;

5. *PacientDetailsView*: classe herda propriedades de uma *table view controller*. Apresenta os detalhes médicos do paciente no formato CCR na sua *view* principal. Clicando em uma informação médica, apresenta os detalhes na *view* secundária;

6. *CalendarView*: classe derivada de uma *view controller*. Efetua a requisição e apresentação de todos os compromissos do médico. Nesta tela é possível deletar eventos e chamar a tela de criação e edição de compromissos;

7. *AddEditCalendarView*: classe derivada de uma *view controller*. Tem como função a criação/edição de eventos. Possui duas *views*: a principal para escrever o título do evento, e dois botões para definir a data de inicial e final os quais chamam uma segunda view que apresenta um componente para seleção de datas;

Como todas as estas classes fazem requisições aos serviços do Google, elas possuem uma variável do tipo GDataService para armazer os dados de *e-mail* e senha do usuário. Na figura 3.3 é apresentada a hierarquia das telas definidas para o aplicativo.

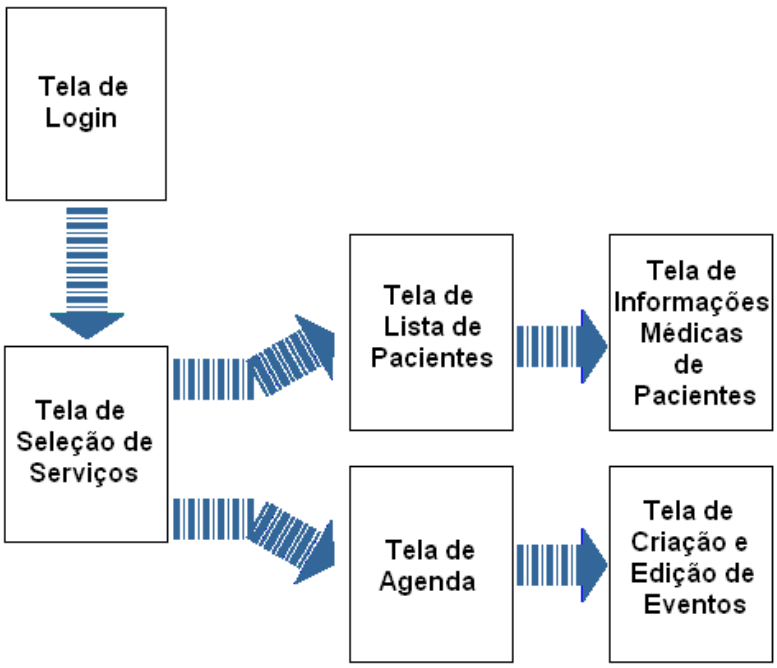

Figura 3.3: Hierarquia de telas do aplicativo projetado.

3.3 Implementação

Definidos os modelo de dados e os requisitos, esta sessão detalha a implementação do aplicativo.

O desenvolvimento da aplicação foi dado pela ordem das telas apresentadas ao usuário. A primeira foi a tela de autenticação de usuário e sua respectiva classe controladora (*RootViewController*). Nesta foram definidos métodos para o controle do teclado auxiliar apresentado durante o preenchimento dos campos de *e-mail* e senha, e um método responsável pela ação a ser tomada quando o botão de *login* é acionado. Nesse método é chamado outro método responsável pela requisição dos calendários do usuário. Essa requisição era responsável pela validação da autenticação. Caso a autenticação fosse válida a aplicação avança a próxima tela informando uma mensagem de *Login* Efetuado e salvando os dados de *e-mail* e senha no escopo do aplicativo. Caso contrário, era mostrada uma mensagem de erro para o usuário (*Login* Inválido).

Primeiramente a idéia era desenvolver classes apenas para os tipos de requisições necessárias. No caso uma para os serviços Google Health e outra para o Google Calendar. Mas durante o desenvolvimento e o conhecimento do funcionamento da API de dados do Google essa modularização foi deixada de lado. Pois cada requisição exigia uma *callback* personalizada para o tipo de dados requerido. Além de que cada requisição gerava a criação de uma *thread*. Dependendo do tempo que a requisição levava, as telas eram carregadas sem apresentação de dados na tela, apenas em memória. Então para resolver esse problema. Quando a requisição acabava e chamava a *callback* definida para tratar os dados, ela era responsável por chamar o método que atualizava os dados na tela. Então ficou definido que cada classe responsável pelo controle de uma interface, fosse responsável pela requisição dos dados necessários ao seu contexto e apresenta-se uma animação gráfica em tela enquanto a emphcallback não fosse executada.

Ciente deste problema foi criada a segunda tela do aplicativo e sua respectiva classe controladora (*AppSelectionView*). Ela contém dois botões os quais o usuário pode escolher o acesso dos serviços Google Health e Google Calendar, além da apresentação dos compromissos do dia. Essa classe é responsável pela requisição da lista de pacientes em caso do botão Health for clicado e da Agenda caso o botão do Calendar. Caso um destes botões fosse clicado a lista de pacientes ou o calendário do usuário eram salvos através de métodos *set* nas classes responsável por controlador as telas filhas.

A próxima *interface* e classe controladora (*MedicalView*) implementada foi a de pacientes. A classe é responsável por mostrar a lista de pacientes cadastros no Google Health e por requisitar os detalhes médicos (documento XML com padrão CCR) de um paciente quando clicado no seu nome e chamar a tela de registros médicos.

A classe *PacientDetailsView* foi a seguinte implementada. Ela é responsavel pela tela que mostra o registro médico do paciente e que faz o parser do XML do padrão CCR. Essa tela possui duas *views*; a primeira mostrando a lista de informações médicas: Notas (*notices*), Detalhes do paciente (*profile details*), condições (*conditions*), alergias (*allergies*), medicamentos (*medications*), procedimentos (*procedures*), resultados de exames (*test results*) e Imunizações (*immunizations*).

Clicando em cada um destes itens são exibidos os dados específicos dos cuidados médicos do paciente.

A quinta classe controladora (*CalendarView*) e *interface* a ser implementada foi a de agenda. Esta tela é responsável pela apresentação dos compromissos do usuário em forma tabular. Ela contém os botões de adicionar, editar e excluir compromissos em uma barra inferior. Quando clicado, o botão adicionar chama a próxima tela do aplicativo que gerencia a criação e edição de um novo evento. O botão excluir apaga um compromisso da agenda e atualiza os dados da tela. O botão editar chama a mesma tela do botão adicionar, mandando a instância do evento a ser modificado para a classe filha, diferentemente do adicionar que manda um ponteiro *nil*. Por fim, a tela mostra mensagens de alerta caso o usuário clique no botão deletar e editar sem ter selecionado um evento. A tela também possui um campo de busca, para a procura de um compromisso desejado.

A ultima tela e classe controladora (*AddEditCalendarView*) a ser desenvolvida foi a de criação e edição de compromissos da agenda. Possui duas *views*. A principal tem um campo de texto para o título do evento e dois botões para definir a data inicial e final do mesmo. Quando um desses botões é clicado um método da classe chama a segunda *view* que fica sobreposta a principal, onde o usuário seleciona uma data. Quando terminada a edição, o botão de salvar é responsável por chamar o método que cria/edita o evento no serviço do Google.

Para uma navegação mais eficiente entre telas segundo o guia de interface-humano da Apple (APPLE, 2010f), foi usado um *navigation controller* visando melhorar interação de troca telas no aplicativo. Como também métodos responsáveis por mudar a orientação

dos dados apresentados em tela baseado na orientação do dispositivo. A figura 4.3 mostra um exemplo. Por fim na criação das interfaces para iPad foram usadas fontes e botões maiores para uma melhor visualização das informações.

4 EXEMPLO DE USO

Nesta seção é apresentado através da exibição de telas do aplicativo a descrição de tarefas na visão do usuário.

4.1 Execução, Login e Tela de Serviços

O usuário começa a interação com o aplicativo clicando no seu ícone no iPhone ou iPad. A aplicação começa a execução com a exibição da tela principal de *login* do usuário. Nela deve-se preecher os campos de *e-mail* e senha. Após a execução desse passo caso o *login* foi efetuado com sucesso apresenta-se a tela de seleção de serviços; Google Health e Google Calendar, como também a exibição de compromissos do dia caso o usuário possui alguma. Caso aconteça algum erro durante o *login* o aplicativo mostra um alerta de *login* inválido sobre a a tela principal. A figura 4.1 mostra as telas destes passos.

Figura 4.1: Tela de *login* e de serviços do aplicativo no iPhone.

4.2 Acessando a dados de Pacientes

Selecionando o serviço Google Health na tela de aplicativos, o programa mostra os pacientes cadastrados na conta do médico em forma tabular. Clicando no nome de um paciente mostra-se a tela com os registros médicos no padrão CCR onde é possivel ver as informações mais detalhadas sobre cada registro clicando no dado desejado. As figuras 4.2 e 4.3 apresentam a tela de pacientes e a de informações médicas do paciente.

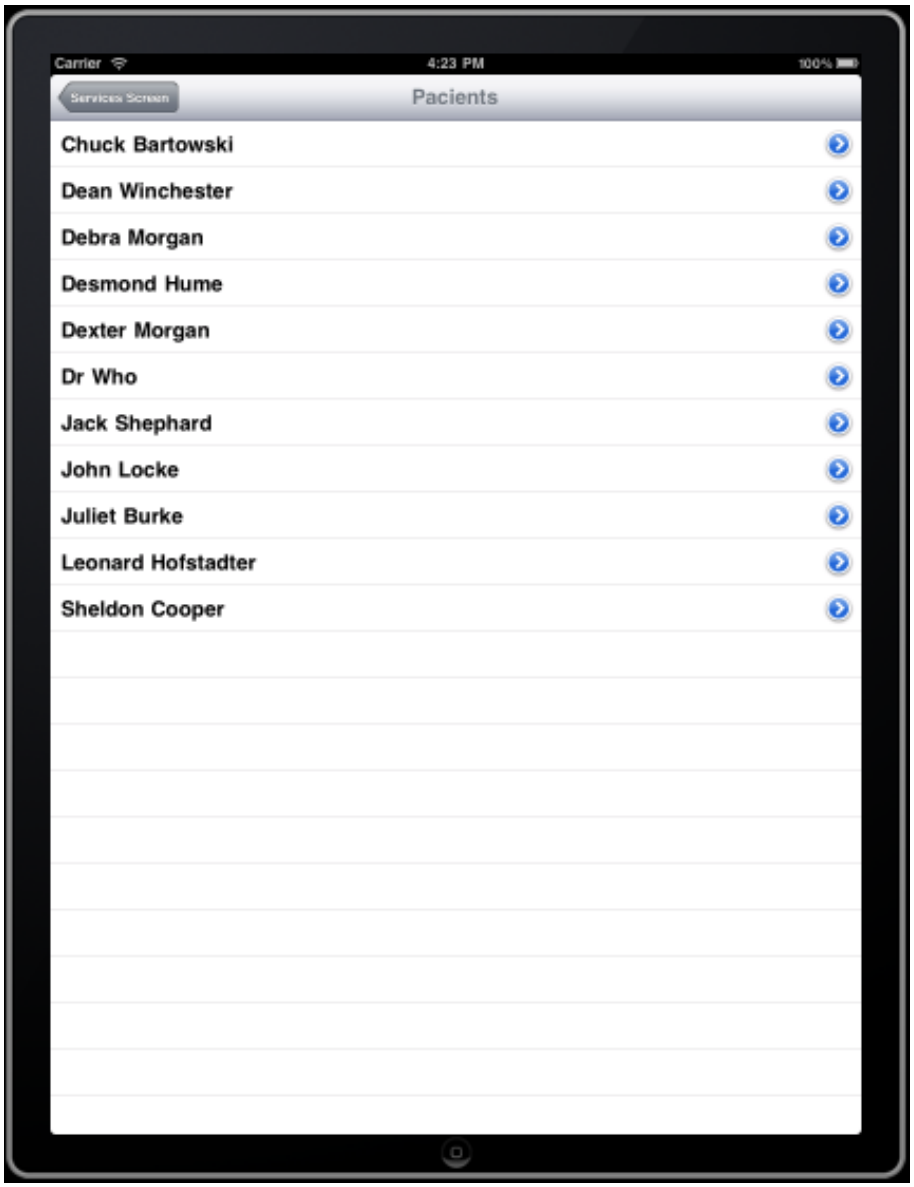

Figura 4.2: Tela de seleção de pacientes no iPad.

Figura 4.3: Tela de seleção de informações médicas no iPad.

4.3 Edição, Exclusão e criação de Compromissos na Agenda

Voltando a tela de seleção de serviço, o usuário pode ver todos compromissos que possui na agenda como também adicionar, editar e deletar eventos clicando nos botões da *tool bar* situados no inferior da *view*. A tela possui abaixo da *navigation bar* um campo de busca para a procura de eventos, onde o usuário pode buscar evento(s) que contenham a palavra desejada. Na figura 4.4 mostra-se a agenda de compromissos, a realização de uma busca de eventos com a palavra *chuck* e a alteração/edição de um evento.

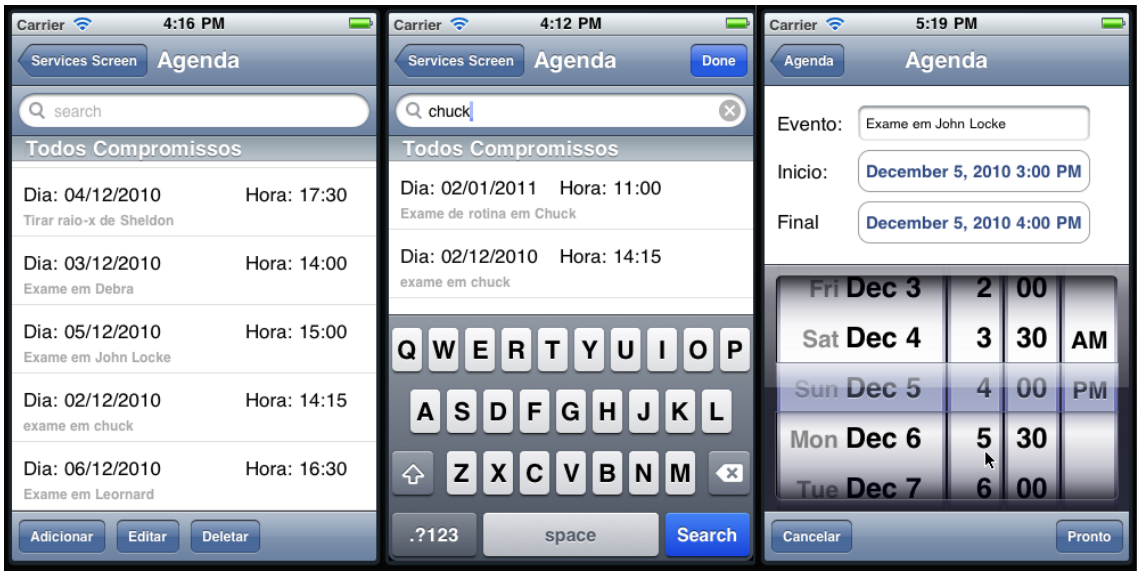

Figura 4.4: Tela da agenda com todos compromissos, realização de uma busca e tela de criação/edição de eventos no iPhone.

5 CONCLUSÃO

Neste trabalho foram abordados os conceitos para o desenvolvimento de aplicativos para dois dispositivos móveis da Apple; o iPad e iPhone. Apresentou-se a modelagem e implementação de um aplicativo para acesso a informações médicas, o qual utilizou o serviço Google Health para armazenamento de dados médicos de pacientes, que usa como registro eletrônico de pacientes o padrão CCR e o Google Calendar para armazenamento de compromissos.

Frente aos desafios de conhecer uma tecnologia nova e alguns imprevistos para configurar o ambiente de desenvolvimento, o trabalho implementado atingiu seu objetivo. O aplicativo implementado apresentou a facilidade de acesso e a gerencia de informações médicas através dos dispositivos usados neste trabalho.

A mobilidade, o acesso a internet e o uso das funções multitoque, demonstram o quanto esta tecnologia pode agregar às atividades do dia-a-dia dos profissionais que trabalham em ambiente hospitalares. Também a integração de funções em um mesmo aplicativo mostrou-se muito útil.

Espera-se que o trabalho desenvolvido sirva em uma futura análise na criação de interfaces para dispositivos móveis criados dentro do projeto ClinicSpaces.

Ficam como sugestões para trabalhos futuros a adição das seguintes funcionalidades:

- Comunicação com outros padrões de registro eletrônico de pacientes;
- Consulta a dados sobres doenças e patologias;
- Portar o aplicativo para outros sistemas abertos de computação móvel. Como por exemplo: o sistema Android do Google.

REFERÊNCIAS

ANSI. **C**. Disponível em: http://en.wikipedia.org/wiki/ANSIC/. Acesso em: Dezembro de 2010.

APEL, B. G. Desenvolvimento de um aplicativo para iPhone integrado ao googlehealth como ferramenta de auxilio ao clinicspace. , [S.l.], 2009.

APPLE. **iPhone**. Disponível em: http://www.apple.com/iPhone/. Acesso em: Dezembro de 2010.

APPLE. **iPad**. Disponível em: http://www.apple.com/iPad/. Acesso em: Dezembro de 2010.

APPLE. **Apple iOS**. Disponível em: http://www.apple.com/ios/. Acesso em: Dezembro de 2010.

APPLE. **Apple SDK**. Disponível em: http://developer.apple.com/devcenter/ios/index.action. Acesso em: Dezembro de 2010.

APPLE. **iOS Application Programming Guide**. Disponível em: http://developer.apple.com/library/ios/navigation/. Acesso em: Dezembro de 2010.

APPLE. **iOS Human Interface Guidelines**. Disponível em: http://developer.apple.com/library/safari/#documentation/UserExperience/Conceptual/MobileHIG/UEBestPractices/UEBestPractices.html. Acesso em: Dezembro de 2010.

ASTM. **ASTM E2369 - 05e1 Standard Specification for Continuity of Care Record**

(CCR). Disponível em: http://www.astm.org/Standards/E2369.htm/. Acesso em: Dezembro de 2010.

Continuity of Care Document: changing the landscape of healthcare information exchange. **Corepoint Health**, [S.l.], 2010.

EISENSTEIN, J.; VANDERDONCKT, J.; PUERTA, A. Adapting to Mobile Contexts with User-Interface Modeling. **in RedWhale Software Corporation, Palo Alto, CA**, [S.l.], 2000.

FERREIRA, G.; AUGUSTIN, I.; LIBRELOTTO, G. R.; SILVA, F. L. da; YAMIN, A. C. Middleware for management of end-user programming of clinical activities in a pervasive environment. **Workshop on Middleware for Ubiquitous and Pervasive Systems**, [S.l.], 2009.

GOOGLE. **Google Calendar**. Disponível em: http://www.google.com/calendar. Acesso em: Dezembro de 2010.

GOOGLE. **Google**. Disponível em: http://www.google.com/. Acesso em: Dezembro de 2010.

HL7. **Health Level 7 Clinical Document Architecture**. Disponível em: http://hl7.org/library/Committees/structure/CDA.ReleaseTwo.CommitteeBallot03.Aug.2004.zip. Acesso em: Dezembro de 2010.

HL7. **Health Level 7**. Disponível em: http://www.hl7.org/. Acesso em: Dezembro de 2010.

HUMAN-COMPUTER INTERACTION, A. S. C. for. **Definition of HCI**. Disponível em: http://old.sigchi.org/cdg/cdg2.html. Acesso em: Dezembro de 2010.

KAY, A. C. **The Early History of Smalltalk**. Disponível em: http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html. Acesso em: Dezembro de 2010.

LAERUM, H.; FAXVAAG, A. Task-oriented evaluation of electronic medical records systems: development and validation of a questionnaire for physicians. **BMC Medical Informatics and Decision Making**, [S.l.], 2004.

LEE, K. B.; GRICE, R. A. Developing a New Usability Testing Method for mobile devices. **Div. of Mobile Commun., Telecommun. Network, Samsung Electron. Co., Ltd., South Korea**, [S.l.], 2004.

ORACLE. **Java**. Disponível em: `http://java.com/`. Acesso em: Dezembro de 2010.

PYTHON. **Python**. Disponível em: `http://www.python.org/`. Acesso em: Dezembro de 2010.

SILVA, F. L. da; AUGUSTIN, I. **ClinicSpace**: modelagem de uma ferramenta piloto para definição de tarefas clínicas em um ambiente de computação pervasiva baseado em tarefas e direcionado ao usuário-final. 2009. Dissertação (Mestrado) — Universidade Federal de Santa Maria, Santa Maria, RS.

SMALLTALK. **Smalltalk**. Disponível em: `http://www.smalltalk.org/`. Acesso em: Dezembro de 2010.

TDO. **TDO Mobile**. Disponível em: `http://www.tdo4endo.com/OurProducts/OurProductsMobile.aspx`. Acesso em: Dezembro de 2010.

VICENTINI, C. F.; MACHADO, A.; AUGUSTIN, I. Requisitos de um Registro Eletrônico de Saúde Ubíquo. **SIRC**, [S.l.], 2009.

WEISER, M. The Computer for the 21st Century. **Scientific American**, [S.l.], 1991.

XEROX-PARC. **MVC**. Disponível em: `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`. Acesso em: Dezembro de 2010.