

**UNIVERSIDADE REGIONAL DE BLUMENAU**  
**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**  
**CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO**

**TAGARELA: APLICATIVO PARA COMUNICAÇÃO  
ALTERNATIVA NO IOS**

**ALAN FILIPE CARDOZO FABENI**

**BLUMENAU  
2012**

**2012/2-01**

**ALAN FILIPE CARDOZO FABENI**

**TAGARELA: APLICATIVO PARA COMUNICAÇÃO  
ALTERNATIVA NO IOS**

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, Mestre - Orientador

**BLUMENAU  
2012**

**2012/2-01**

# **TAGARELA: APLICATIVO PARA COMUNICAÇÃO**

## **ALTERNATIVA NO IOS**

Por

**ALAN FILIPE CARDOZO FABENI**

Trabalho aprovado para obtenção dos créditos  
na disciplina de Trabalho de Conclusão de  
Curso II, pela banca examinadora formada  
por:

---

Presidente: Prof. Dalton Solano dos Reis, Mestre – Orientador, FURB

---

Membro: Prof. Alexander Roberto Valdameri, Mestre – FURB

---

Membro: Prof. Aurélio Faustino Hoppe, Mestre – FURB

Blumenau, 10 de dezembro de 2012

Dedico este trabalho à toda a minha família e todos os meus amigos, especialmente aqueles que estiveram comigo em todos estes anos de graduação.

## **AGRADECIMENTOS**

À minha família, pelo apoio e incentivo ao longo de todo o curso.

Ao professor Dalton Solano dos Reis, pela orientação e apoio no desenvolvimento deste trabalho.

Ao fonoaudiólogo Rodrigo França, pela colaboração no desenvolvimento deste trabalho.

Ao professor Mauro Marcelo Mattos, pela oportunidade em participar do projeto de Gestão de Saúde Pública através do desenvolvimento deste trabalho.

Aos meus amigos Jackson Krause, Jonathan Mauricenz e Thiago Pradi que foram companheiros durante todo o curso e com certeza contribuíram para a aquisição dos conhecimentos necessários para o desenvolvimento deste trabalho.

Nunca tenha certeza de nada, pois a sabedoria  
começa com a dúvida.

Sigmund Freud

## **RESUMO**

Este trabalho apresenta o desenvolvimento de um aplicativo para comunicação alternativa para a plataforma iOS. O aplicativo tem como principal objetivo criar um ambiente aonde o fonoaudiólogo, o seu paciente e o tutor deste paciente possam interagir de forma que haja uma evolução na capacidade de comunicação do paciente. Tudo isto através de planos de atividades elaborados pelo fonoaudiólogo em conjunto com o tutor do paciente. Estes planos tem como objetivo estimular a capacidade de comunicação do paciente através da utilização dos recursos multimídias presentes na plataforma iOS. Durante a evolução deste trabalho, recursos exclusivos da plataforma iOS, como o iCloud e o *Core Data* são apresentados através de conceitos e utilização dos mesmos.

Palavras-chave: iOS. Comunicação alternativa. Tecnologia assistiva. Multimídia móvel.

## **ABSTRACT**

This paper presents the development of a prototype application for alternative communication in the iOS platform. The prototype's main objective is to create an environment where the speech therapist, the patient and the tutor of this patient can interact in a way that there is an evolution in the patient's communication skills. All this through activity plans developed by the speech therapist in conjunction with the patient's tutor. These plans aims to stimulate the patient's ability to communicate through the use of multimedia resources present on the iOS platform. During the evolution of this work, unique features of iOS platform, as iCloud and Core Data are presented by concepts and their use.

Key-words: iOS. Alternative communication. Assistive technology. Mobile multimedia.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Combinação de duas Blisswords para a formação de uma nova Blissword .....	23
Figura 2 – Símbolos PIC .....	23
Figura 3 – Símbolos PCS.....	24
Figura 4 – Visão geral das camadas da plataforma iOS .....	25
Figura 5 – Aplicativo: Alexicom AAC.....	33
Figura 6 – Ferramenta: prancha livre de comunicação.....	34
Figura 7 – Ferramenta: teclado virtual livre .....	34
Quadro 1 – Características dos trabalhos correlatos.....	35
Figura 8 – Diagrama de casos de uso do ator fonoaudiólogo.....	38
Figura 9 – Diagrama de casos de uso do ator tutor .....	39
Figura 10 – Diagrama de caso de uso do ator paciente .....	39
Figura 11 – Diagrama de pacotes .....	46
Figura 12 – Pacote Patient .....	47
Figura 13 – Pacote Category.....	49
Figura 14 – Pacote Symbol.....	51
Figura 15 – Pacote Tutor .....	53
Figura 16 – Pacote Plan.....	54
Figura 17 – Pacote Util.....	56
Figura 18 – Pacote AppDelegate.....	57
Figura 19 – Diagrama de sequência Criar símbolos.....	59
Figura 20 – Diagrama de sequência Criar planos.....	61
Figura 21 – Diagrama de atividades do aplicativo .....	62
Figura 22 – Diagrama de classes das entidades relativas ao fonoaudiólogo .....	64
Figura 23 – Diagrama de classes das entidades relativas ao tutor e ao paciente .....	65
Quadro 2 – Primeira etapa do método de criação do arquivo JSON.....	67
Quadro 3 – Segunda etapa do método de criação do arquivo JSON.....	68
Quadro 4 – Terceira etapa do método de criação do arquivo JSON .....	68
Figura 24 – Estrutura do arquivo .psyb.....	69
Quadro 5 – Primeira etapa do método de gravação de áudio .....	69
Quadro 6 – Segunda etapa do método de gravação de áudio .....	70

Quadro 7 – Terceira etapa do método de gravação de áudio.....	70
Quadro 8 – Conexão com o iCloud .....	72
Figura 25 – Criação de pacientes.....	73
Figura 26 – Criação de tutores.....	74
Figura 27 – Criação de categorias de símbolos .....	74
Figura 28 – Criação de símbolos .....	75
Figura 29 – Criação de planos .....	76
Figura 30 – Tela para selecionar os símbolos do plano.....	77
Figura 31 – Tela de envio de e-mails com o plano anexado .....	77
Figura 32 – E-mail enviado pelo fonoaudiólogo com as opções para importar o plano .....	78
Figura 33 – Lista dos tutores importados e lista dos planos associados ao tutor escolhido .....	78
Figura 34 – Símbolos que compõem o plano escolhido .....	79
Figura 35 – Cenário do sistema .....	80
Figura 36 – Tempo de inserção dos dados em microssegundos.....	82
Figura 37 – Tempo de recuperação dos dados em microssegundos.....	83
Quadro 9 – Comparação do trabalho desenvolvido com os trabalhos correlatos.....	84
Quadro 10 – Caso de uso Criar paciente .....	90
Quadro 11 – Caso de uso Criar tutor .....	91
Quadro 12 – Caso de uso Criar categoria de símbolos .....	92
Quadro 13 – Caso de uso Criar símbolos .....	94
Quadro 14 – Caso de uso Criar planos .....	95
Quadro 15 – Caso de uso Enviar planos.....	96
Quadro 16 – Caso de uso Criar histórico de observações.....	96
Quadro 17 – Caso de uso Enviar histórico de observações .....	97
Quadro 18 – Caso de uso Importar planos.....	97
Quadro 19 – Caso de uso Escolher tutor .....	97
Quadro 20 – Caso de uso Escolher plano .....	98
Quadro 21 – Caso de uso Interagir com os símbolos.....	98
Quadro 22 – Caso de uso Enviar observações sobre o paciente para o fonoaudiólogo .....	98
Quadro 23 – Caso de uso Visualizar histórico de uso dos símbolos.....	99
Quadro 24 – Caso de uso Enviar histórico de uso dos símbolos para o fonoaudiólogo .....	99

<b>Quadro 25 – Caso de uso Importar arquivo com histórico de uso dos símbolos enviados pelo tutor.....</b>	<b>99</b>
<b>Quadro 26 – Caso de uso Visualizar histórico de uso dos símbolos enviados pelo tutor.....</b>	<b>100</b>
<b>Quadro 27 – Caso de uso Importar arquivo com observações do paciente enviadas pelo tutor.....</b>	<b>100</b>
<b>Quadro 28 – Caso de uso Visualizar observações do paciente enviadas pelo tutor .....</b>	<b>100</b>
<b>Figura 38 – Criação de um novo App ID.....</b>	<b>103</b>
<b>Figura 39 – Habilitação do App ID para o iCloud.....</b>	<b>103</b>
<b>Figura 40 – Configuração dos <i>Entitlements</i> .....</b>	<b>104</b>
<b>Figura 41 – Plano de atividades manual .....</b>	<b>105</b>
<b>Quadro 29 – Histórico de observações da paciente Giovanna com o fonoaudiólogo Rodrigo França .....</b>	<b>106</b>

## **LISTA DE SIGLAS**

AAC - *Advanced Audio Coding*

ALAC - *Apple Lossless Audio Codec*

CAA – Comunicação Alternativa e Aumentativa

CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico

CPU – *Central Processing Unit*

GPL – *General Public License*

GPU – *Graphics Processing Unit*

IDE – *Integrated Development Environment*

JSON - JavaScript Object Notation

LPCM - *Linear Pulse-Code Modulation*

OpenAL – *Open Audio Library*

OpenGL ES – *Open Graphics Library for Embedded Systems*

PCS – *Picture Communication Symbols*

PDF – *Portable Document Format*

PIC – *Pictogram Ideogram Communication*

PUCPR - Pontifícia Universidade Católica do Paraná

RGB - *Red Green Blue*

SDK – *Software Development Kit*

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

UML - *Unified Modeling Language*

# SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>16</b>
1.1 OBJETIVOS DO TRABALHO .....	17
1.2 ESTRUTURA DO TRABALHO .....	17
<b>2 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>19</b>
2.1 TECNOLOGIA ASSISTIVA.....	19
2.1.1 Avaliação e implementação da tecnologia assistiva .....	20
2.1.2 Modalidades de tecnologia assistiva .....	20
2.2 COMUNICAÇÃO ALTERNATIVA E AUMENTATIVA.....	21
2.2.1 Sistemas de comunicação alternativa e aumentativa .....	21
2.2.1.1 Blissymbolics.....	22
2.2.1.2 Pictogram Ideogram Communications .....	23
2.2.1.3 Picture Communication Systems .....	24
2.3 PLATAFORMA IOS.....	24
2.3.1 Camada <i>Cocoa Touch</i> .....	25
2.3.2 Camada <i>Media</i> .....	26
2.3.3 Camada <i>Core Services</i> .....	26
2.3.4 Camada <i>Core OS</i> .....	27
2.3.5 iCloud.....	28
2.3.6 Desenvolvimento para iOS .....	28
2.4 RECURSOS MULTIMÍDIA DA PLATAFORMA IOS .....	29
2.4.1 Tecnologias de áudio.....	29
2.4.2 Tecnologias gráficas.....	30
2.4.3 Tecnologias de vídeo.....	31
2.5 TRABALHOS CORRELATOS .....	32
2.5.1 Alexicom AAC.....	32
2.5.2 Projeto AMPLISOFT .....	33
2.5.3 Comparação entre os trabalhos correlatos.....	35
<b>3 DESENVOLVIMENTO.....</b>	<b>36</b>
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	36
3.2 ESPECIFICAÇÃO .....	37
3.2.1 Casos de uso .....	38

3.2.1.1 Criar paciente .....	39
3.2.1.2 Criar tutor .....	40
3.2.1.3 Criar categorias de símbolos .....	40
3.2.1.4 Criar símbolos .....	40
3.2.1.5 Criar planos .....	41
3.2.1.6 Enviar planos .....	41
3.2.1.7 Criar histórico de observações .....	41
3.2.1.8 Enviar histórico de observações .....	42
3.2.1.9 Importar planos .....	42
3.2.1.10 Escolher tutor .....	42
3.2.1.11 Escolher plano .....	42
3.2.1.12 Interagir com os símbolos .....	43
3.2.1.13 Enviar observações sobre o paciente para o fonoaudiólogo .....	43
3.2.1.14 Visualizar histórico de uso dos símbolos .....	43
3.2.1.15 Enviar histórico de uso dos símbolos para o fonoaudiólogo .....	44
3.2.1.16 Importar arquivo com histórico de uso dos símbolos enviados pelo tutor .....	44
3.2.1.17 Visualizar histórico de uso dos símbolos enviados pelo tutor .....	44
3.2.1.18 Importar arquivo com observações do paciente enviados pelo tutor .....	44
3.2.1.19 Visualizar observações do paciente enviadas pelo tutor .....	45
3.2.2 Diagrama de pacotes .....	45
3.2.2.1 Pacote Patient .....	47
3.2.2.2 Pacote Category .....	49
3.2.2.3 Pacote Symbol .....	50
3.2.2.4 Pacote Tutor .....	52
3.2.2.5 Pacote Plan .....	54
3.2.2.6 Pacote Util .....	55
3.2.2.7 Pacote AppDelegate .....	56
3.2.2.8 Pacote Core Data Classes .....	57
3.2.3 Diagramas de sequência .....	57
3.2.3.1 Diagrama de sequência Criar símbolos .....	57
3.2.3.2 Diagrama de sequência Criar planos .....	60
3.2.4 Diagramas de atividades .....	62
3.2.5 Diagrama de classes das entidades do aplicativo .....	63

<b>3.3 IMPLEMENTAÇÃO .....</b>	<b>66</b>
3.3.1 Técnicas e ferramentas utilizadas.....	66
3.3.2 Criação do arquivo JSON com os dados do plano .....	66
3.3.3 Gravação de áudio.....	69
3.3.4 iCloud.....	70
3.3.5 Operacionalidade da implementação .....	72
3.3.5.1 Criação de pacientes .....	72
3.3.5.2 Criação de tutores .....	73
3.3.5.3 Criação de categorias de símbolos.....	74
3.3.5.4 Criação de símbolos.....	75
3.3.5.5 Criação de planos.....	76
3.3.5.6 Importação e utilização de um plano .....	77
<b>3.4 RESULTADOS E DISCUSSÃO .....</b>	<b>79</b>
3.4.1 Análise do roteiro de implementação de um recurso de tecnologia assistiva para elaboração dos requisitos do aplicativo.....	80
3.4.2 Desempenho do <i>Core Data</i> .....	82
3.4.3 Comparativo entre o trabalho desenvolvido e os trabalhos correlatos.....	83
<b>4 CONCLUSÕES .....</b>	<b>85</b>
<b>4.1 EXTENSÕES .....</b>	<b>86</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>87</b>
<b>APÊNDICE A – Descrição detalhada do casos de uso .....</b>	<b>89</b>
<b>APÊNDICE B – Descrição das entidades do aplicativo .....</b>	<b>101</b>
<b>APÊNDICE C – Configuração do iCloud .....</b>	<b>103</b>
<b>APÊNDICE D – Plano de atividades manual .....</b>	<b>105</b>
<b>APÊNDICE E – Histórico de observações da paciente Giovanna com o fonoaudiólogo Rodrigo França</b>	<b>106</b>



## 1 INTRODUÇÃO

A constituição federal garante igualdade a todos os indivíduos, portanto nenhuma pessoa com necessidade especial deve ser impedida de trabalhar ou estudar (BRASIL, 1988). Hoje em dia, é possível notar um número crescente destes indivíduos incluindo-se na sociedade, tanto no mercado de trabalho quanto no ambiente escolar. O fato de que estas pessoas precisam de recursos e serviços especiais para superar suas necessidades e atingir seus objetivos de se adequar aos padrões da sociedade não pode mais ser ignorado.

Um dos grandes desafios enfrentados pelas pessoas com necessidades especiais é a questão da comunicação. No ambiente profissional e educacional, comunicar-se é um fator determinante para atingir sucesso na respectiva área. De acordo com Schirmer et al. (2007, p. 57), “Desde o momento em que o ser humano diz suas primeiras palavras, a linguagem facilita o encontro de desejos, necessidades, interação social, acesso às informações e conhecimento sobre o complexo mundo em que vive.”

Desta forma é necessário que as pessoas impossibilitadas de se comunicarem através da forma habitual utilizem meios alternativos de comunicação para que consigam se expressar. O meio alternativo pode ser o uso de linguagens de sinais ou até mesmo um dispositivo computacional com síntese de voz. Com o avanço da tecnologia de dispositivos móveis, qualquer pessoa pode ter ao seu alcance um dispositivo que o auxilie nesta tarefa de comunicar-se de forma eficiente com os indivíduos ao seu redor. Os softwares para comunicação alternativa já existentes não se beneficiam de todos os recursos novos que os dispositivos móveis atuais oferecem.

Decorrente destes fatos é possível verificar que existe uma demanda para a criação de ferramentas para estes dispositivos, que auxilie os deficientes a se comunicarem de forma eficiente. Observa-se que está ferramenta será criada de uma forma genérica, permitindo que atenda uma ampla gama de deficiências relacionadas a questão da comunicação. Cabe ao fonoaudiólogo do paciente, customizar a ferramenta para atender uma necessidade específica.

Diante do exposto, é realizado o desenvolvimento de um aplicativo para comunicação alternativa para pessoas com necessidades especiais, utilizando os recursos multimídia presentes no sistema móvel iOS, como áudio e imagens para promover uma maior imersão e interação com o usuário. Além da utilização dos recursos multimídia, este trabalho realiza o desenvolvimento de um ambiente que permite a troca de mensagens e atividades entre as pessoas envolvidas no processo de comunicação da pessoa com necessidades especiais.

O trabalho desenvolvido utiliza uma abordagem qualitativa para a investigação de seus resultados. Neste tipo de abordagem, é necessário aprofundar-se na compreensão dos fenômenos estudados – ações dos indivíduos, grupos ou organizações em seu ambiente e contexto social – interpretando-os segundo a perspectiva dos participantes da situação enfocada, sem se preocupar com representatividade numérica, generalizações estatísticas e relações lineares de causa e efeito (BRAATZ, 2012).

## 1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um aplicativo de comunicação alternativa utilizando os recursos multimídia da plataforma iOS 6.

Os objetivos específicos do trabalho são:

- a) propiciar um ambiente que auxilie a comunicação alternativa a pessoas com necessidades especiais e crianças em fase de alfabetização;
- b) proporcionar imersão ao usuário do aplicativo utilizando os diversos *frameworks* de áudio e imagem da plataforma iOS 6;
- c) especificar um ambiente que possa ser customizado de forma a atender necessidades específicas de cada paciente e a troca de mensagens e atividades entre as pessoas envolvidas.

## 1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos, sendo que no primeiro capítulo é apresentado uma introdução ao tema abordado, os objetivos e a estrutura deste trabalho.

O capítulo dois contém a fundamentação teórica necessária para permitir um melhor entendimento sobre este trabalho.

O capítulo três apresenta o desenvolvimento do aplicativo, contemplando também os seus requisitos e a especificação contendo os casos de uso, diagramas de pacotes, de atividades, de sequência e diagramas de classes. Neste capítulo são apresentadas também as ferramentas utilizadas na implementação. Por fim, são apresentados os resultados e discussão.

O quarto capítulo refere-se às conclusões do presente trabalho e sugestões para trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 trata sobre os conceitos relacionados a tecnologia assistiva. A seção 2.2 procura esclarecer sobre a comunicação alternativa e aumentativa. A seção 2.3 contém uma introdução sobre a plataforma iOS, incluindo os componentes do sistema operacional e os aspectos básicos do desenvolvimento de aplicações. Na seção 2.4 são apresentados os principais componentes multimídia da plataforma iOS. Por fim, a seção 2.5 apresenta dois trabalhos correlatos ao trabalho desenvolvido.

### 2.1 TECNOLOGIA ASSISTIVA

Tecnologia assistiva é uma área do conhecimento que reúne produtos, recursos, serviços e técnicas que tem uma única função em comum: proporcionar ou estender habilidades para pessoas com necessidades especiais (PELOSI, 2011).

Esta área do conhecimento tem como principal objetivo promover maior independência para pessoas com necessidades especiais. Além disso, promove também melhorias na qualidade de vida, mobilidade e integração com a sociedade.

Para o emprego das tecnologias assistivas são utilizados recursos e serviços. Recursos são itens ou equipamentos que são utilizados para manter ou melhorar as capacidades das pessoas com necessidades. Serviços são aqueles prestados de forma profissional, visando o emprego dos recursos como tecnologia para o tratamento adequado.

Os recursos podem variar de uma simples bengala para ajudar um deficiente físico até um sofisticado sistema computacional que utiliza recursos multimídia para auxiliar na alfabetização de crianças autistas (SARTORETTO; BERSCH, 2007).

Os recursos de tecnologia assistiva devem ser utilizados como alternativa para que o deficiente físico realize o que deseja ou o que precisa. Para isso, é necessário encontrar uma estratégia para que este recurso aumente suas capacidades de ação e interação com a sociedade.

Os serviços geralmente envolvem diversas áreas, entre elas: fisioterapia, terapia ocupacional, fonoaudiologia, educação, psicologia, enfermagem, medicina, engenharia, arquitetura, *design*, entre outras.

Os serviços de tecnologia assistiva devem envolver profundamente o usuário da tecnologia e sua família, bem como os profissionais das diversas áreas, de forma que a tecnologia se adeque completamente as necessidades específicas do usuário.

### 2.1.1 Avaliação e implementação da tecnologia assistiva

Para avaliar a necessidade de implementação de um recurso de tecnologia assistiva, o *Center on Disabilities* da *California State University* de Northridge, elaborou um roteiro que ajuda a identificar todas as fases do processo de implementação de um recurso.

O primeiro passo deste roteiro é identificar as necessidades do futuro usuário, de forma a entender suas rotinas e suas necessidades. A partir deste levantamento, é necessário estabelecer metas que deverão ser atingidas após o uso do recurso tecnológico, a fim de estabelecer um resultado satisfatório que deve ser alcançado. Após isto, inicia-se a fase de testes com o recurso, esta fase é essencial no processo de implementação, pois é nela que o usuário terá o primeiro contato com o recurso que poderá ser utilizado. Se as metas definidas anteriormente foram atingidas após a fase de testes, o profissional especialista deve recomendar o uso do recurso tecnológico e realizar um acompanhamento periódico do usuário com o recurso. Caso contrário, a utilização do equipamento deve ser revista para atender as metas propostas (SCHIRMER et al., 2007).

### 2.1.2 Modalidades de tecnologia assistiva

A tecnologia assistiva se organiza em diversas especialidades. Essa organização por especialidades contribui para o desenvolvimento de pesquisas, recursos, especializações profissionais e organização de serviços. Algumas das especialidades da tecnologia assistiva são:

- a) auxílios para a vida diária e vida prática;
- b) Comunicação Alternativa e Aumentativa (CAA);
- c) recursos de acessibilidade ao computador;
- d) adequação postural;
- e) auxílio de mobilidade;
- f) adaptação de veículos.

## 2.2 COMUNICAÇÃO ALTERNATIVA E AUMENTATIVA

A CAA é uma das áreas da tecnologia assistiva responsável por auxiliar os deficientes físicos a se comunicarem de forma eficiente com o restante da sociedade. Esta área da tecnologia assistiva atende principalmente pessoas sem fala ou escrita funcional ou em desfasagem entre sua necessidade comunicativa e sua habilidade em falar e/ou escrever (SCHIRMER et al., 2007).

Desta forma, a CAA pode ser considerada como uma área da prática clínica e educacional, que tem como objetivo principal compensar a incapacidade do indivíduo com problemas de comunicação através de recursos e serviços que se propõe ao auxílio. Ela tem como objetivo tornar o indivíduo com problemas de comunicação o mais independente possível, de forma que o mesmo possa ampliar sua participação na sociedade, através da interação com os outros, seja na escola, no trabalho ou em sua casa.

A comunicação é considerada alternativa quando o indivíduo utiliza um sistema de CAA para se comunicar ao invés da fala, devido a impossibilidade de articular ou produzir sons adequadamente. A comunicação aumentativa é quando o indivíduo utiliza um outro meio de comunicação para complementar ou compensar deficiências que a fala apresenta, mas sem substituí-la totalmente (SCHIRMER et al., 2007).

### 2.2.1 Sistemas de comunicação alternativa e aumentativa

Sistemas de comunicação alternativa e aumentativa são os recursos, as estratégias e as técnicas que apoiam os modos de comunicação existentes ou substituem a fala.

Um dos recursos mais utilizados na área de comunicação alternativa e aumentativa são as pranchas de comunicação, devido a sua facilidade de uso e as possibilidades de comunicação que oferece ao usuário deficiente.

As pranchas de comunicação são objetos utilizados para transmitir mensagens através de símbolos gráficos contidos nelas. Os símbolos devem representar algum conceito ou objeto do mundo real. Desta forma, o usuário da prancha pode comunicar-se com outras pessoas simplesmente apontando o dedo para a imagem na prancha. Cada símbolo contido na prancha deve ter algum significado relevante, para que seja possível o usuário formar sentenças visuais complexas, combinando os símbolos para expressar seus desejos e vontades. O

vocabulário de símbolos de uma prancha de comunicação é escolhido de acordo com as necessidades de seu usuário, portanto as pranchas são personalizadas (SCHIRMER et al., 2007).

Os símbolos utilizados nas pranchas de comunicação podem ser classificados em símbolos pictográficos, símbolos arbitrários e símbolos ideográficos.

Os símbolos pictográficos são símbolos em que seus desenhos assemelham-se com aquilo que deseja-se simbolizar. Os símbolos arbitrários são símbolos em que seus desenhos não tem relação pictográfica entre a forma e aquilo que desejam simbolizar, ou seja, não são semelhantes à forma correspondente. Os símbolos ideográficos são símbolos em que o desenho representa uma ideia, esse tipo de símbolo tem o objetivo de criar uma associação gráfica entre o símbolo e o conceito que ele representa (SCHIRMER et al., 2007).

No momento de projetar uma prancha de comunicação, é necessário classificar os símbolos em categorias de acordo a com a sua função semântica. As categorias de classificação mais comuns são: social, pronomes pessoais, verbos, substantivos, adjetivos e advérbios e miscelânea. Além disso, é recomendado que os símbolos sejam organizados de forma que os símbolos de mesma categoria estejam próximos e que a disposição destes símbolos estejam de acordo com a ordem frasal, ou seja, a prancha deve estar organizada da melhor maneira possível para que o usuário consiga formar sentenças de forma simplificada e que não haja muito esforço para achar o símbolo necessário.

A grande vantagem em utilizar pranchas de comunicação na CAA, é que os símbolos podem ser modificados e personalizados de acordo com as necessidades do usuário. Desta forma, se um sistema de símbolos não atender adequadamente o usuário, nada impede que o profissional especialista busque um símbolo alternativo em outros sistemas de símbolos para atender as necessidade específicas de seu paciente.

Existem vários sistemas de símbolos gráficos que são conhecidos e utilizados internacionalmente. Entre eles se destacam o Blissymbolics, o *Pictogram Ideogram Communication* (PIC) e o *Picture Communication Symbols* (PCS).

#### 2.2.1.1 Blissymbolics

O sistema Blissymbolics é um conjunto de símbolos, na maioria ideográficos, criado por Charles K. Bliss para facilitar a comunicação internacional entre as pessoas. Sua primeira aplicação na área de CAA ocorreu em 1971.

Atualmente o sistema conta com 4.500 símbolos padronizados pela organização internacional *Blissymbolics Communication International*. Estes símbolos, que também podem ser chamados de Blisswords são baseados nos significados das palavras e por este motivo podem ser combinados em infinitas maneiras (BLISSYMBOLICS, 2012). Para a confecção dos símbolos, são utilizadas formas simples devido a sua facilidade e rapidez de desenho. Devido a capacidade de representação de conceitos concretos e abstratos, os símbolos Blissymbolics podem ser aplicados em crianças e adultos. Na Figura 1 é possível visualizar a combinação de duas Blisswords para a formação de uma nova Blissword.



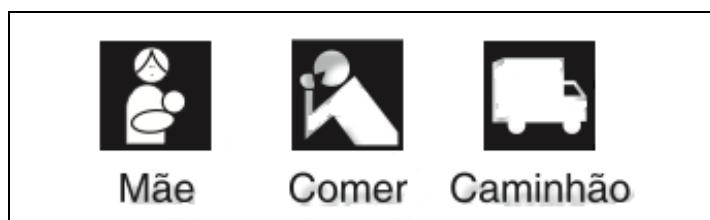
Fonte: Schirmer et al. (2007).

Figura 1 – Combinação de duas Blisswords para a formação de uma nova Blissword

#### 2.2.1.2 Pictogram Ideogram Communications

O PIC é um sistema de símbolos pictográficos criado em 1980 por Subah Maharaj. Este sistema de símbolos foi criado para estimular a comunicação entre indivíduos com algum tipo de deficiência e que não tenham manifestado grandes avanços utilizando o sistema Blissymbolics.

O PIC é um sistema composto por centenas de símbolos, divididos em categorias semânticas e desenhados em branco sob um fundo preto. Devido ao seu tipo de desenho, o PIC é indicado principalmente para crianças que apresentam dificuldades na visão, visto que o branco no preto cria um maior contraste (CRESER COMUNICANDO, 2011). Apesar de serem visualmente fáceis de serem reconhecidos, o PIC é um sistema menos versátil e mais limitado que os outros, pois seus símbolos não são combináveis. Na Figura 2 é possível visualizar alguns símbolos PIC.



Fonte: Schirmer et al. (2007).

Figura 2 – Símbolos PIC

### 2.2.1.3 Picture Communication Systems

O PCS é um sistema de símbolos pictográficos criados em 1980 pela fonoaudióloga Roxanna Mayer Johnson. Este sistema de símbolos é um dos mais utilizados no mundo e possui aproximadamente 8 mil símbolos que representam uma grande variedade de vocabulário (SCHIRMER et al., 2007). Os símbolos do PCS são de fácil reconhecimento e por isso são utilizados principalmente por crianças ou indivíduos que apresentam dificuldades em compreender representações mais abstratas. Devido a grande quantidade de símbolos deste sistema, alguns dos símbolos são desenhados parcialmente para customização do mesmo pelo usuário. Além disto, alguns símbolos possuem versões alternativas que são mais abstratos ou menos abstratos dependendo das capacidades do usuário que o utiliza.

O PCS é o sistema de símbolos mais utilizado no Brasil, pois trata-se de um sistema aberto que pode adaptar-se a questões regionais, culturais e pessoais do usuário. Desta forma, a versão brasileira deste sistema possui símbolos característicos da nossa história e cultura nacional. Na Figura 3 é possível visualizar alguns símbolos PCS.



Fonte: Schirmer et al. (2007).

Figura 3 – Símbolos PCS

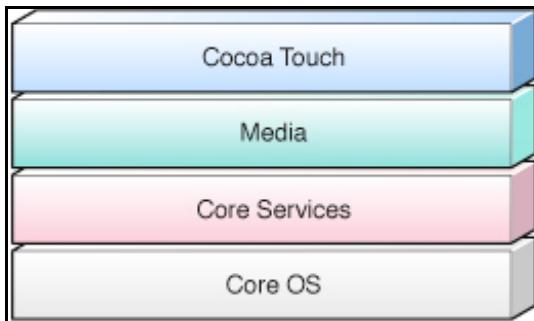
## 2.3 PLATAFORMA IOS

O iOS é um sistema operacional móvel da Apple, desenvolvido inicialmente para o iPhone mas agora presente em grande parte dos dispositivos móveis da empresa (iPhone, iPad e iPod Touch). Sua principal característica é a sua interface multi-toque, que permite um manuseio do dispositivo através de gestos variados.

Para desenvolver para o iOS é necessário o uso do iOS *Software Development Kit* (SDK), que trata-se de um conjunto de ferramentas disponibilizadas pela Apple para auxiliar os desenvolvedores. O SDK do iOS é composto por diversas aplicações, entre elas o Xcode, o

iOS Simulator e o Instruments (APPLE INC, 2011).

A arquitetura do iOS é composta por 4 camadas, conforme pode ser visto na Figura 4.



Fonte: Apple Inc (2011).

Figura 4 – Visão geral das camadas da plataforma iOS

### 2.3.1 Camada *Cocoa Touch*

A camada *Cocoa Touch* contém os principais *frameworks* necessários para a construção de aplicações iOS. Esta camada fornece os *frameworks* necessários para a construção da estrutura básica de qualquer aplicação iOS através de várias tecnologias, como o recurso de multitarefa, a entrada baseada em toque, notificações *push*, suporte a documentos e muitos outros recursos de alto nível.

O suporte a documentos é um recurso introduzido no iOS 5 e permite que as aplicações manipulem dados associados aos documentos do usuário. Este suporte permite implementar aplicações que armazenam os documentos do usuário no iCloud. Além disso, é fornecido também acesso assíncrono de leitura e escrita de dados, armazenamento seguro, gravação automática, suporte a detecção de conflitos no iCloud, entre outras funcionalidades.

A multitarefa é um recurso introduzido no iOS 4. Com este recurso, as aplicações desenvolvidas não são finalizadas quando o usuário troca de uma aplicação para outra, ao invés disso, a aplicação que estava rodando anteriormente roda em *background* enquanto a nova aplicação toma o seu lugar de execução. Para preservar a bateria do dispositivo, as aplicações que estão rodando em *background* são colocadas em estado de suspensão pelo sistema operacional, desta forma a aplicação continua em memória mas não executa nenhuma instrução. Este comportamento pode ser customizado pelo desenvolvedor da aplicação, uma aplicação que consiste em um *player* de música, por exemplo, pode continuar a execução da música em *background* enquanto o usuário do dispositivo utiliza uma outra aplicação em primeiro plano.

As notificações *push* são um recurso introduzido no iOS 3. Este recurso permite que os usuários sejam alertados sobre novas informações relacionadas à aplicação mesmo quando a aplicação não está executando. Utilizando este recurso, os desenvolvedores podem enviar notificações de textos e alertas audiovisuais ao usuário.

Introduzidas nas primeiras versões do iOS, o reconhecimento de gestos permite adicionar o reconhecimento de gestos variados na aplicação desenvolvida. Entre os gestos que podem ser reconhecidos estão o gesto de toque, de pinça, de arrastar, de rotacionar e o gesto de *swipe*.

### 2.3.2 Camada *Media*

Na camada *Media* estão as tecnologias multimídia. Esta camada fornece os *frameworks* para trabalhar com áudio, gráficos e vídeos no iOS, incluindo a biblioteca gráfica *Open Graphics Library for Embedded Systems* (OpenGL ES) (APPLE INC, 2011).

### 2.3.3 Camada *Core Services*

A camada *Core Services* contém os *frameworks* fundamentais que todas as aplicações do sistema iOS utilizam indiretamente. Esta camada é responsável pelo gerenciamento de documentos do iCloud, pela tecnologia *Grand Central Dispatch* (GCD), pelo suporte a *eXtensible Markup Language* (XML) e também pelo armazenamento de dados através da biblioteca SQLite.

O armazenamento de documentos no iCloud é um recurso introduzido no iOS 5 e que permite que a aplicação desenvolvida escreva documentos e dados em uma localização central na nuvem e que acesse todos esses itens em todos os dispositivos do usuário. Isso significa que o usuário poderá visualizar e editar esses documentos em qualquer dispositivo sem precisar sincronizar ou transferir o documento novamente.

A tecnologia GCD foi introduzida no iOS 4 e permite que as aplicações desenvolvidas manuseiem *tasks*, que é uma forma de programação assíncrona utilizada como alternativa às *threads*.

A biblioteca SQLite permite que as aplicações desenvolvidas façam armazenamento de dados em estruturas SQL. Com a utilização dessa biblioteca, é possível fazer com que a

aplicação crie base de dados locais e gerencie as tabelas e registros dessa base de dados de forma rápida e otimizada.

O suporte básico a documentos XML é feito através de classes do *framework Foundation*. O suporte mais avançado, incluindo a manipulação, o *parse* e a escrita em documentos é feito através de bibliotecas *open source* disponíveis junto ao *frameworks* padrões do sistema operacional.

#### 2.3.4 Camada *Core OS*

A camada *Core OS* contém as funcionalidades de baixo nível em que todas as outras camadas do sistema operacional iOS são construídas. Dificilmente os *frameworks* desta camada são utilizados diretamente pelas aplicações, o uso de seus *frameworks* é recomendado apenas em situações em que a aplicação precisa lidar diretamente com a segurança ou com a comunicação com algum *hardware* externo. Alguns dos *frameworks* desta camada são: *Accelerate Framework*, *Core Bluetooth*, *External Accessory Framework* e o *System*.

O *Accelerate Framework* é um *framework* introduzido no iOS 4 e que contém classes utilizadas para realizar cálculos de álgebra linear, cálculos para processamento de imagens, entre outros. A grande vantagem de utilizar este *framework* ao invés de fazer os cálculos manualmente, é que os cálculos deste *framework* são otimizados para todas as configurações de *hardware* presentes nos dispositivos que rodam o iOS.

O *Core Bluetooth* é um *framework* que permite que as aplicações desenvolvidas interajam com dispositivos *bluetooths* externos. Este *framework* permite o *scan* de dispositivos e também fornece métodos de conexão, desconexão e de leitura e gravação de atributos relacionados ao dispositivo conectado.

O *External Accessory Framework* é um *framework* utilizado para realizar a comunicação com acessórios externos que não sejam dispositivos *bluetooth*. Este *framework* fornece métodos para retornar informações sobre o acessório conectado e também métodos para iniciar a comunicação entre o dispositivo do usuário e o acessório externo. Após iniciada a comunicação, o *framework* fornece métodos para manipular o acessório utilizando os comandos que ele suporta.

Além destes *frameworks* a camada *Core OS* contém uma biblioteca chamada *LibSystem*, utilizada para acessar recursos utilizados pelo *kernel* do sistema iOS. Entre estes recursos estão o sistema de *sockets* de rede e o sistema de acesso de arquivos.

### 2.3.5 iCloud

O iCloud é um serviço de armazenamento em nuvem desenvolvido pela Apple. O iCloud permite que os usuários do serviço armazenem arquivos de músicas, fotos, aplicativos, documentos e *backups* na nuvem. Desta forma, todo este conteúdo pode ser acessado de qualquer aparelho. Ele pode ser considerado como uma ferramenta de sincronização baseada na nuvem, já que os aplicativos passam a conversar entre si, podendo acessar e armazenar as informações geradas pela mesma aplicação em outros dispositivos.

Para os desenvolvedores a Apple disponibiliza a iCloud *Application Programming Interface* (API), que nada mais é do que um conjunto de *frameworks* necessários para que as aplicações desenvolvidas armazenem seus dados na nuvem. Com isso, os desenvolvedores podem sincronizar os dados da mesma aplicação em diversos dispositivos (APPLE INC, 2012b).

### 2.3.6 Desenvolvimento para iOS

Para desenvolver aplicações para o iOS, a Apple disponibiliza um conjunto completo de ferramentas para auxiliar os desenvolvedores. Entre estas ferramentas estão o Xcode, o iOS Simulator e o Instruments.

O Xcode é uma *Integrated Development Environment* (IDE) que fornece um ambiente completo para o desenvolvimento de aplicações nativas para iOS escritas na linguagem Objective-C. O Xcode inclui um editor de código fontes, um editor de interface gráficas e muitos outros recursos, como um gerenciador de repositórios de código fonte e ferramentas de *debug*.

O iOS Simulator é um ambiente utilizado para testar as aplicações desenvolvida para iOS. Este ambiente funciona como um simulador da plataforma iOS, aonde todas as características relevantes dos dispositivos iPhone, iPad e iPod Touch estão presentes. Este simulador conta com diversos recursos que simulam algumas características específicas do *hardware* dos dispositivos, incluindo o reconhecimento de gestos, simulação de ligações a receber, rotacionamento do dispositivo nas quatro direções possíveis, entre outros.

O Instruments é uma ferramenta utilizada para analisar o desempenho da aplicação desenvolvida enquanto ela roda no próprio dispositivo ou no iOS Simulator. Com o

Instruments é possível reunir diversas informações da aplicação em execução, incluindo a utilização de memória, atividade de escrita e leitura de disco, uso da rede, entre outros. Desta forma é possível analisar diferentes aspectos da aplicação e analisar o seu desempenho a fim de identificar pontos de melhoria.

Todas estas ferramentas estão disponíveis gratuitamente no iOS SDK e são exclusivas para o sistema operacional Mac OS X. Estas ferramentas são atualizadas anualmente a cada nova versão do iOS lançada.

## 2.4 RECURSOS MULTIMÍDIA DA PLATAFORMA IOS

A camada *Media* fornece todo os recursos para a construção de aplicações ricas em conteúdo e apelo visual, que são características importantes em qualquer aplicativo para iOS. Esta camada é formada por diversos *frameworks* multimídias responsáveis pela manipulação direta de áudio, imagem e vídeo. Os *frameworks* oferecidos são de alto nível e baixo nível, de forma que os *frameworks* de alto nível são mais fáceis de utilizar porém oferecem menos recursos e os *frameworks* de baixo nível são mais complexos porém oferecem mais flexibilidade e recursos de utilização.

### 2.4.1 Tecnologias de áudio

O iOS fornece diversos *frameworks* para lidar com a parte de áudio, incluindo a reprodução e gravação de conteúdo.

Os *frameworks* que lidam com a parte de áudio são: *AV Foundation*, *Core Audio*, *Media Player* e *Open Audio Library* (OpenAL) (APPLE INC, 2011).

O *AV Foundation* é um *framework* de alto nível que contém classes em Objective-C que facilitam a reprodução e gravação de conteúdo em áudio. Este *framework* pode ser utilizado para reproduzir arquivos ou sons em memória de qualquer duração. É possível utilizar este *framework* para reproduzir múltiplos sons simultaneamente e controlar vários aspectos de cada som reproduzido.

O *Core Audio* é um *framework* de baixo nível que fornece classes em C para a manipulação de áudio estéreo. Com este *framework* é possível gerar, gravar, mixar e

reproduzir áudio nas aplicações. Além disto, o *framework Core Audio* permite o gerenciamento de *buffers* e a reprodução de áudio com múltiplos canais ou conteúdo transmitido via *streaming*.

O *framework Media Player* é de alto nível e fornece acesso fácil a biblioteca de músicas do usuário. Este *framework* é utilizado principalmente para a reprodução de faixas e *playlists* da biblioteca musical do usuário do dispositivo móvel.

O *framework OpenAL* fornece acesso a interfaces multi-plataformas para reproduzir áudio tridimensional. Este *framework* é utilizado principalmente em jogos que requerem distribuição de áudio em cada elemento no mundo 3D do jogo.

Todos os *frameworks* suportam os formatos de áudio *Advanced Audio Coding* (AAC), *Apple Lossless Audio Codec* (ALAC), *Linear Pulse-Code Modulation* (LPCM), entre outros.

#### 2.4.2 Tecnologias gráficas

Na plataforma iOS a forma mais simples de enriquecer graficamente uma aplicação é utilizar imagens pré-renderizadas em conjunto com os componentes gráficos padrões do sistema operacional. Porém, há situações aonde é necessário ir além e utilizar *frameworks* específicos para a renderização e manipulação do conteúdo gráfico.

Os *frameworks* necessários para lidar com conteúdo gráfico na plataforma iOS são: *Core Graphics*, *Core Animation*, *Core Image*, *Image I/O*, *Assets Library*, *Core Text* e *OpenGL ES* (APPLE INC, 2011).

O *framework Core Graphics*, também conhecido como *Quartz*, é um *framework* de baixo nível indicado para desenhos baseados em vetores. Este *framework* fornece os recursos para renderização sem serrilhados, manipulação de gradientes, cores e documentos em *Portable Document Format* (PDF).

O *Core Animation* é um *framework* de alto nível, utilizado para a criação de animações complexas e efeitos visuais. Este *framework* é integrado em diversas partes do iOS, de forma que as animações utilizadas pelo sistema operacional possam ser utilizadas e customizadas na própria aplicação.

O *Core Image* é um *framework* de alto nível que fornece uma série de filtros para manipulação de vídeos e imagens estáticas. É possível utilizar estes filtros para operações variadas, que vão desde a correção de fotos à detecção de faces. A grande vantagem na utilização destes filtros é que as operações realizadas sob a imagem não a alteram

diretamente. Além disto, este *framework* utiliza o poder de processamento da *Graphics Processing Unit* (GPU) e da *Central Processing Unit* (CPU) para que as operações realizadas sejam rápidas e eficientes (APPLE INC, 2011).

O *Image I/O* é um *framework* que fornece acesso a interfaces necessárias para a leitura e escrita de diversos formatos de imagem. Este *framework* é utilizado principalmente para acessar os meta-dados da imagem, devido a sua alta eficiência de leitura e acesso.

O *framework Assets Library* é um *framework* de alto nível que fornece fácil acesso às fotos e vídeos da biblioteca pessoal do usuário do dispositivo móvel. Para todos os *frameworks* descritos acima, o iOS suporta oficialmente os formatos: .png, .jpeg, .jpg e .bmp.

O *Core Text* é um *framework* de baixo nível que contém classes em C necessárias para manipulação avançada de texto. Este *framework* disponibiliza funções necessárias para a renderização personalizada de texto e estilo de fontes, necessárias para aplicações que requerem manuseio avançado de texto exibido na tela.

O *framework OpenGL ES* fornece ferramentas para desenho 2D e 3D. É um *framework* de alto nível, baseado em C e é utilizado principalmente em jogos. Este *framework* trabalha diretamente com o *hardware* do dispositivo para fornecer *frame rates* elevados nas aplicações no qual é utilizado.

#### 2.4.3 Tecnologias de vídeo

O iOS fornece diversos *frameworks* necessários para lidar com vídeos, incluindo a reprodução e gravação de conteúdo. Os *frameworks* que lidam com vídeos no iOS são: *Media Player* e *Core Media* (APPLE INC, 2011).

O *framework Media Player* é um *framework* de alto nível indicado para a reprodução de vídeos na aplicação. Este *framework* apenas facilita a reprodução vídeos, portanto não é indicado para a manipulação do conteúdo (ZDZIARSKY, 2009, p. 349).

O *framework Core Media* é um *framework* de baixo nível que fornece funções avançadas de criação, reprodução e manipulação de conteúdo exibido em vídeo. Os *frameworks* de vídeo suportam os formatos: .mov, .mp4, .m4v e .3gp.

## 2.5 TRABALHOS CORRELATOS

Existem diversas ferramentas comerciais e acadêmicas que tem como objetivo auxiliar portadores de necessidades especiais ou crianças em fase de aprendizado através do uso da comunicação alternativa. Dentre estas, foi selecionada a ferramenta comercial Alexicom AAC e o projeto acadêmico AMPLISOFT.

### 2.5.1 Alexicom AAC

O Alexicom AAC é um aplicativo gratuito, desenvolvido para o sistema operacional iOS que tem a função de transformar o dispositivo móvel em um aparelho de comunicação alternativa. Este aplicativo tem como foco principal crianças com algum tipo de necessidade especial e que necessitam de uma forma simples de se comunicar (APPLE INC, 2012a).

O aplicativo consiste em uma coleção de categorias, divididas por idade e necessidade. Cada uma destas categorias conta com imagens representando as necessidades básicas de uma pessoa. O aplicativo conta com o recurso de síntese de voz, para que quando o usuário tocar em uma imagem, o aplicativo reproduza em voz alta o que ela representa (ALEXICOM, 2011).

Cada categoria pode ter subcategorias. Por exemplo, ao clicar na categoria “*I'm hungry*” (Estou com fome), o aplicativo leva o usuário a uma outra categoria com imagens de alimentos, possibilitando o usuário a formar uma sentença visual, indicando que está com fome ou que precisa de tal alimento.

Além das categorias pré-definidas do aplicativo, existe também a possibilidade do usuário adicionar novas categorias de imagens. Este procedimento é realizado no *site* da empresa desenvolvedora do aplicativo. O usuário deve fazer um cadastro no *site* e então fazer o *upload* das imagens e do que elas representam. Após fazer o *upload*, é feita a sincronização entre as novas categorias criadas e o dispositivo móvel.

O aplicativo pode ser visto na Figura 5, onde são exibidas algumas das categorias de imagens e os botões que fazem o controle do áudio a ser falado.



Fonte: Apple Inc (2012a).

Figura 5 – Aplicativo: Alexicom AAC

## 2.5.2 Projeto AMPLISOFT

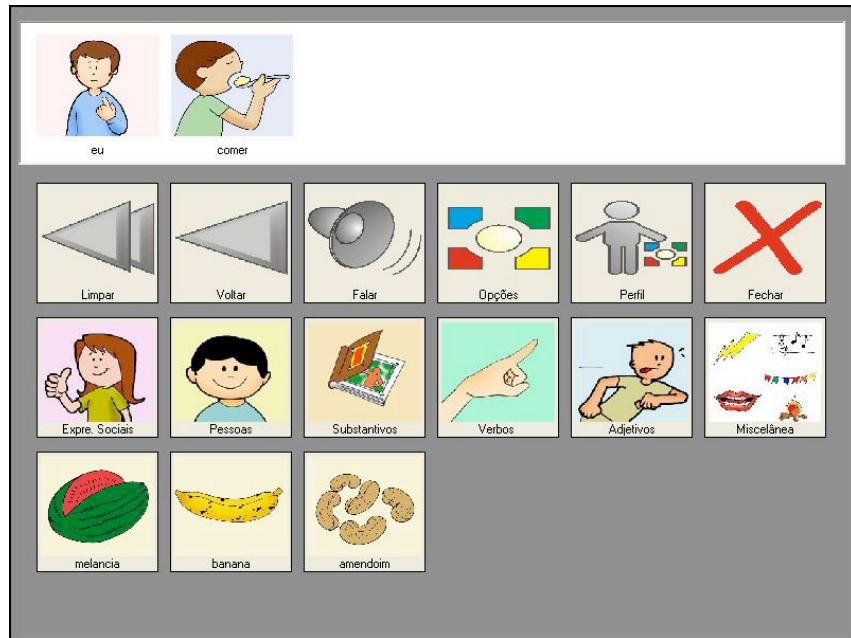
O projeto AMPLISOFT é um projeto acadêmico, iniciado em dezembro de 2003 com recursos do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)<sup>1</sup> e apoio da Pontifícia Universidade Católica do Paraná (PUCPR). Este projeto é composto por aplicativos que possuem licenças de software livre, como a *General Public License* (GPL) e que tem como objetivo geral proporcionar uma alternativa barata e simples de comunicação alternativa para pessoas com algum tipo de deficiência motora ou dificuldade em se comunicar.

A prancha livre de comunicação é um aplicativo indicado para pessoas não alfabetizadas, que ainda estão se alfabetizando ou pessoas que tem dificuldades em se comunicar através da fala (NOHAMA et al., 2010).

O aplicativo consiste em um conjunto de imagens que contém um significado próprio. Estas imagens podem ser combinadas entre si para que o usuário da prancha forme sentenças através delas, expressando suas vontades.

A ferramenta pode ser visualizada na Figura 6, aonde é possível verificar a existência das imagens e da combinação entre elas (eu + comer).

<sup>1</sup> Até 1971, Conselho Nacional de Pesquisa, cuja sigla se manteve.



Fonte: Nohama et al. (2010).

Figura 6 – Ferramenta: prancha livre de comunicação

O teclado virtual livre é um outro aplicativo desenvolvido para usuários da prancha livre de comunicação que estão em processo de alfabetização e que já são capazes de formar palavras através de um teclado alfanumérico.

Além de servir para facilitar a comunicação, este aplicativo pode ser utilizado para fazer a edição de texto por pessoas com deficiência. Suas principais características são as teclas largas e acessíveis e o seu recurso de predição de palavras, que auxilia os usuários no momento de escolher as palavras certas para redigirem um texto (NOHAMA et al., 2010).

A ferramenta pode ser visualizada na figura 7, aonde é possível verificar o editor de texto e o recurso de predição de palavras.



Fonte: Nohama et al. (2010).

Figura 7 – Ferramenta: teclado virtual livre

### 2.5.3 Comparação entre os trabalhos correlatos

O Quadro 1 mostra de forma resumida as principais características dos trabalhos analisados, tendo como base critérios considerados importantes em um aplicativo de comunicação alternativa. As informações foram dispostas em colunas, representando na vertical os trabalhos analisados e as linhas apresentam as características de cada trabalho, indicando semelhanças e/ou diferenças.

<b>características / trabalhos correlatos</b>	<b>Projeto AMPLISOFT</b>	<b>Alexicom AAC</b>
criação de símbolos personalizados	x	x
associação de áudio aos símbolos	x	x
possibilidade de impressão das pranchas	x	
criação de símbolos via plataforma web		x

Quadro 1 – Características dos trabalhos correlatos

Considerando a análise dos trabalhos correlatos, pode-se verificar que as ferramentas contam com os recursos necessários para o apoio à indivíduos com problemas de comunicação, porém nenhuma das ferramentas estudadas utilizam todos os recursos multimídia presentes no iOS. Também, percebe-se que estas ferramentas estudadas não contam com um apelo visual satisfatório, o que é um fator determinante para atrair as crianças que possam vir a utilizar a ferramenta. Outra característica importante, não implementada nos trabalhos estudados, é a possibilidade da troca de mensagens entre os usuários envolvidos no processo de comunicação do paciente, recurso importante, pois facilita o acompanhamento do processo de comunicação do mesmo.

Neste trabalho, tem-se por objetivo a implementação de um aplicativo que utilize os recursos multimídia da plataforma iOS para propiciar um ambiente que auxilie a comunicação alternativa a pessoas com necessidades especiais e crianças em fase de alfabetização. Este aplicativo deve ser criado de forma genérica, de forma que possa atender as necessidades específicas de cada paciente que utiliza o aplicativo, através da troca de mensagens e atividades entre as pessoas que estão envolvidas no processo de comunicação do paciente.

### 3 DESENVOLVIMENTO

Neste capítulo são abordadas as etapas de desenvolvimento do aplicativo. A primeira seção apresenta os principais requisitos funcionais e não-funcionais do problema trabalhado. A segunda seção descreve a especificação do aplicativo através de diagramas da *Unified Modeling Language* (UML). A terceira seção apresenta os detalhes da implementação do aplicativo, incluindo os trechos de códigos mais relevantes ao funcionamento do aplicativo e exemplos de uso do aplicativo. Por fim, a quarta seção aborda os resultados deste trabalho.

#### 3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O aplicativo para comunicação alternativa deverá:

- a) permitir que o fonoaudiólogo crie pacientes (Requisito Funcional – RF);
- b) permitir que o fonoaudiólogo crie tutores (RF);
- c) permitir que o fonoaudiólogo crie categorias de símbolos (RF);
- d) permitir que o fonoaudiólogo crie símbolos (RF);
- e) permitir que o fonoaudiólogo crie planos (RF);
- f) permitir que o fonoaudiólogo envie os planos criados para os pacientes (RF);
- g) permitir que o fonoaudiólogo crie históricos de observações do paciente (RF);
- h) permitir que o fonoaudiólogo envie os históricos de observações do paciente via e-mail (RF);
- i) permitir que o fonoaudiólogo gere um PDF com o histórico de observações do paciente (RF);
- j) permitir que o fonoaudiólogo faça a importação do arquivo com o histórico de uso dos símbolos do paciente enviados pelo tutor (RF);
- k) permitir que o fonoaudiólogo visualize o histórico de uso dos símbolos do paciente (RF);
- l) permitir que o fonoaudiólogo faça a importação do arquivo com as observações do paciente enviadas pelo tutor (RF);
- m) permitir que o fonoaudiólogo visualize as observações do paciente enviadas pelo tutor (RF);

- n) permitir que o tutor faça a importação dos planos criados pelo fonoaudiólogo via e-mail (RF);
- o) permitir que o tutor que está auxiliando o paciente faça a escolha de qual tutor deseja seguir os planos (RF);
- p) permitir que o tutor que está auxiliando o paciente faça a escolha de qual plano do tutor escolhido deseja seguir (RF);
- q) permitir que o tutor do paciente envie observações sobre o paciente para o fonoaudiólogo (RF);
- r) permitir que o tutor visualize o histórico de uso dos símbolos (RF);
- s) permitir que o tutor envie informações sobre o histórico de uso dos símbolos para o fonoaudiólogo (RF);
- t) permitir que o paciente interaja com os símbolos através do toque (RF);
- u) rodar no sistema operacional iOS 5 e em suas versões superiores (Requisito Não Funcional – RNF);
- v) ser implementado utilizando a linguagem de programação Objective-C (RNF);
- w) ser implementado utilizando o ambiente de desenvolvimento Xcode (RNF);
- x) seguir os guias de interface visual da Apple (RNF);
- y) realizar a troca de informações entre o fonoaudiólogo e o tutor via arquivos em formato JSON (RNF);
- z) manter os dados do aplicativo na nuvem (iCloud) (RNF).

### 3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando os diagramas da UML. Os casos de uso são apresentados através de um diagrama de casos de uso, seguidos da descrição do cenário de cada caso de uso. Em seguida, as classes do aplicativo são apresentadas através de um diagrama de pacotes. Como forma de facilitar a implementação, dois casos de uso foram representados através de diagramas de sequência. Complementando a especificação com diagramas UML, a utilização completa do aplicativo pelo fonoaudiólogo, tutor e paciente é demonstrada através de um diagrama de atividades. Por fim, as entidades do *Core Data* são demonstradas através de dois diagramas de classe. As ferramentas Astah Professional e Visual Paradigm foram as ferramentas responsáveis pela modelagem e geração destes diagramas.

### 3.2.1 Casos de uso

Nesta seção serão descritos os casos de uso do aplicativo. A partir dos requisitos criados para o aplicativo foram criados 20 casos de uso. Estes casos de uso tem como objetivo organizar os requisitos em funcionalidades que possam ser executadas de forma simples pelos usuários.

Conforme apresenta a Figura 8, o primeiro ator do aplicativo é o fonoaudiólogo que tem como responsabilidade gerenciar e elaborar os planos que serão executados pelo paciente. A Figura 9 apresenta os casos de uso do tutor, o segundo ator que é responsável por auxiliar e acompanhar o paciente na utilização dos planos enviados pelo fonoaudiólogo. Por fim, a Figura 10 apresenta o caso de uso do paciente, o terceiro ator que tem como responsabilidade interagir com símbolos dos planos criados pelo fonoaudiólogo.

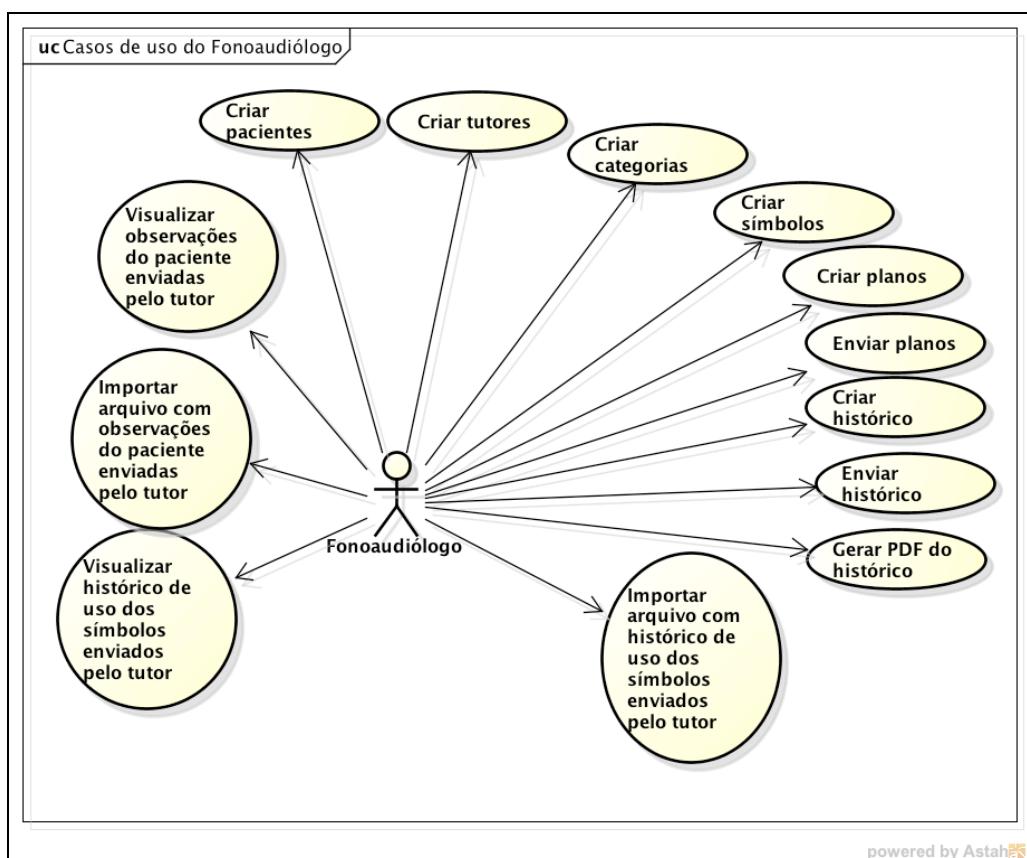


Figura 8 – Diagrama de casos de uso do ator fonoaudiólogo

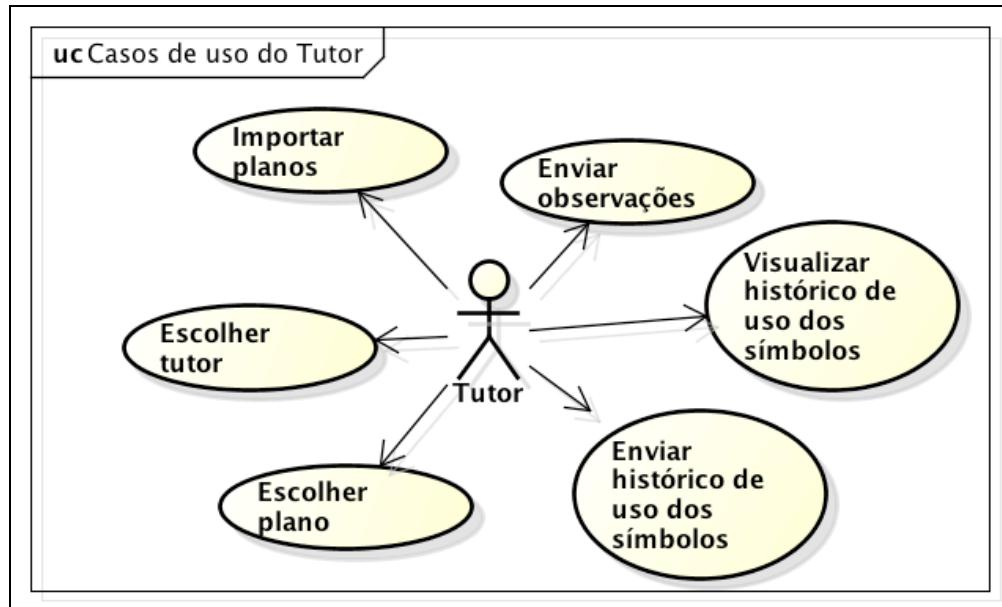


Figura 9 – Diagrama de casos de uso do ator tutor

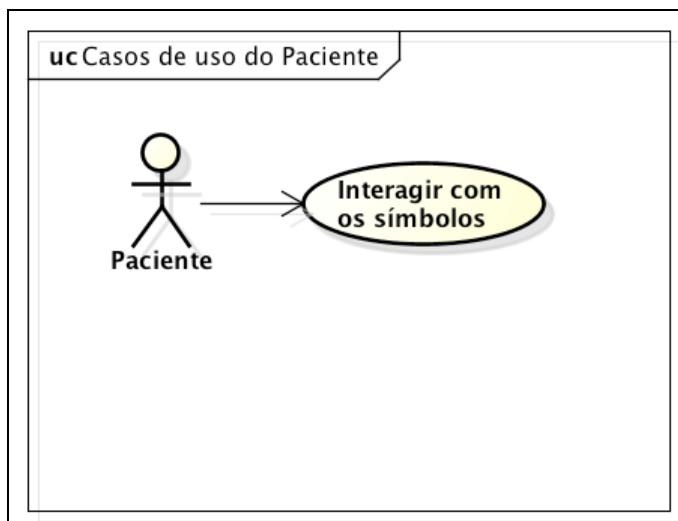


Figura 10 – Diagrama de caso de uso do ator paciente

Os diagramas de casos de uso foram desenvolvidos observando os padrões da UML. Cada caso de uso foi detalhado em cenários e vinculado a, pelo menos, um requisito funcional. Esta vinculação tem como objetivo facilitar a identificação do propósito do caso de uso e justificar sua existência.

A seguir serão apresentados os respectivos casos de uso. As descrições detalhadas destes casos de uso podem ser visualizadas no apêndice A.

### 3.2.1.1 Criar paciente

Este caso de uso permite que o fonoaudiólogo realize a criação de um paciente. O

aplicativo permite que o fonoaudiólogo informe dados do paciente, como: nome, data de nascimento, patologia, e-mail e a data em que o paciente iniciou o uso do aplicativo. Além disto, o aplicativo permite que o fonoaudiólogo associe uma foto para o paciente, esta foto pode ser capturada através da câmera do iPad ou escolhida através da galeria de fotos do iPad.

### 3.2.1.2 Criar tutor

Este caso de uso permite que o fonoaudiólogo realize a criação de um tutor. O aplicativo permite que o fonoaudiólogo informe o nome do tutor, seu local de atuação e seu e-mail. Além disto, o aplicativo permite que o fonoaudiólogo associe uma foto para o tutor, esta foto pode ser capturada através da câmera do iPad ou escolhida através da galeria de fotos do iPad.

### 3.2.1.3 Criar categorias de símbolos

Este caso de uso permite que o fonoaudiólogo crie categorias para os símbolos. A tela de criação de categorias é acessível a partir da tela de biblioteca de símbolos. Para criar uma categoria o aplicativo permite que o fonoaudiólogo informe o nome da categoria e uma cor que representa esta categoria. O fonoaudiólogo seleciona a cor da categoria a partir de uma lista de 16 cores pré-definidas.

### 3.2.1.4 Criar símbolos

Este caso de uso permite que o fonoaudiólogo crie os símbolos para o paciente. A tela de criação de símbolos é acessível a partir da tela de detalhes do paciente ou da tela de biblioteca de símbolos. Para criar um símbolo o fonoaudiólogo é obrigado a informar o nome do símbolo, o significado deste símbolo, sua categoria e uma imagem que o representa. O fonoaudiólogo também tem a opção de gravar uma mensagem de áudio que representa o significado do símbolo. Os símbolos que são criados a partir da tela de detalhes do paciente são os símbolos personalizados para o paciente, ou seja, são os símbolos que são exclusivos para ele e que servem apenas para atender as suas necessidades específicas. O símbolos que

são criados a partir da tela de biblioteca de símbolos são os símbolos gerais, que servem para atender as necessidades genéricas de todos os pacientes que são atendidos pelo fonoaudiólogo.

### 3.2.1.5 Criar planos

Este caso de uso permite que o fonoaudiólogo crie os planos que serão enviados para o paciente. A tela de criação de planos é acessível a partir da tela que lista os tutores. O fonoaudiólogo seleciona um tutor na lista e a tela de criação de planos é apresentada com a foto do tutor selecionado, seu nome e a data atual em que o plano começou a ser criado. Na tela de criação de planos, o fonoaudiólogo é obrigado a informar o nome do plano, seu tipo e uma descrição. O plano pode ser de dois tipos: prancha ou símbolos grandes.

### 3.2.1.6 Enviar planos

Este caso de uso permite que o fonoaudiólogo envie os planos criados via e-mail. Para enviar um plano é necessário que o fonoaudiólogo entre na tela de criação de planos e pressione o botão **Enviar**. Quando o botão **Enviar** é pressionado, o aplicativo apresenta a tela padrão de envio de e-mails do iOS com dois arquivos anexados. O primeiro arquivo anexado é o arquivo com os símbolos do plano compactados em um único arquivo no formato **.psyb**. O segundo arquivo é a representação dos símbolos do plano em um arquivo no formato **.pdf**.

### 3.2.1.7 Criar histórico de observações

Este caso de uso permite que o fonoaudiólogo escreva observações sobre o paciente. A tela de observações é acessível a partir da tela de detalhes do paciente. O fonoaudiólogo tem a opção de escrever observações durante todo o período de tratamento clínico do paciente.

### 3.2.1.8 Enviar histórico de observações

Este caso de uso permite que o fonoaudiólogo envie o histórico de observações do paciente via e-mail. Para enviar o histórico de observações o fonoaudiólogo deve entrar na tela de observações e pressionar o botão **Enviar**. Assim que o botão **Enviar** é pressionado, o aplicativo gera um arquivo **.pdf** com todo o histórico do paciente e apresenta a tela padrão de envio de e-mails do iOS com o arquivo **.pdf** anexado.

### 3.2.1.9 Importar planos

Este caso de uso permite que o tutor do paciente importe para o aplicativo os planos enviados por e-mail pelo fonoaudiólogo. Esta importação é feita pelo tutor através do aplicativo padrão de e-mail do iOS.

### 3.2.1.10 Escolher tutor

Este caso de uso permite que o tutor que está auxiliando o paciente escolha o tutor de qual deseja visualizar os planos criados. Após importar o plano enviado pelo fonoaudiólogo, o aplicativo apresenta em sua tela inicial uma lista com todos os tutores de todos os planos que já foram importados. Através desta lista, o tutor que está auxiliando o paciente deve escolher um tutor de qual deseja visualizar os seus planos.

### 3.2.1.11 Escolher plano

Este caso de uso permite que o tutor do paciente escolha o plano que deseja seguir. Após escolher um tutor na tela inicial do aplicativo, o sistema apresenta todos os planos do tutor que foi selecionado. O tutor, deve então, selecionar um plano para que o paciente possa interagir com os seus símbolos.

### 3.2.1.12 Interagir com os símbolos

Este caso de uso permite que o paciente interaja com os símbolos do plano selecionado pelo seu tutor. Se o plano selecionado for do tipo prancha ou símbolos grandes, o aplicativo apresenta uma tela com todos os símbolos daquele plano. Se o paciente pressionar um símbolo e este símbolo tiver uma mensagem de áudio associada, o aplicativo reproduz esta mensagem de áudio no volume padrão do iOS.

### 3.2.1.13 Enviar observações sobre o paciente para o fonoaudiólogo

Este caso de uso permite que o tutor do paciente envie observações sobre o paciente para o fonoaudiólogo. Para enviar observações sobre o paciente, o tutor deve selecionar seu nome na lista de tutores na tela inicial do aplicativo e após isto, na tela que o aplicativo apresenta, pressionar o botão `Enviar observações`. Após pressionar este botão, o tutor deve escrever a observação e pressionar o botão `Enviar`. Assim que o tutor pressionar o botão `Enviar`, o aplicativo apresenta ao tutor a tela padrão de envio de e-mails do iOS com um arquivo no formato `.pobs` anexado. O arquivo `.pobs` contém as observações escritas em um formato que possibilita que o fonoaudiólogo importe estas observações para o aplicativo quando receber elas via e-mail.

### 3.2.1.14 Visualizar histórico de uso dos símbolos

Este caso de uso permite que o tutor do paciente visualize o histórico de uso dos símbolos do plano. Para visualizar o histórico de uso dos símbolos, o tutor deve selecionar seu nome na lista de tutores na tela inicial do aplicativo e após isto, na tela que o aplicativo apresenta, pressionar o botão `Visualizar histórico`. O aplicativo irá apresentar uma tela com uma tabela informando os símbolos utilizados e as suas datas de utilização.

### 3.2.1.15 Enviar histórico de uso dos símbolos para o fonoaudiólogo

Este caso de uso permite que o tutor do paciente envie o histórico de uso dos símbolos para o fonoaudiólogo. Para enviar o histórico de uso dos símbolos, é necessário que o tutor esteja visualizando o histórico de uso dos símbolos e pressione o botão `Enviar`. Quando o tutor pressionar este botão, o aplicativo apresenta ao tutor a tela padrão de envio de e-mails do iOS com um arquivo no formato `.phist` anexado. O arquivo `.phist` contém o histórico de uso dos símbolos em um formato que possibilita que o fonoaudiólogo importe este histórico para o aplicativo quando receber ele via e-mail.

### 3.2.1.16 Importar arquivo com histórico de uso dos símbolos enviados pelo tutor

Este caso de uso permite que o fonoaudiólogo importe para o aplicativo o histórico de uso dos símbolos enviados por e-mail pelo tutor do paciente. Esta importação é feita pelo fonoaudiólogo através do aplicativo padrão de e-mail do iOS.

### 3.2.1.17 Visualizar histórico de uso dos símbolos enviados pelo tutor

Este caso de uso permite que o fonoaudiólogo visualize o histórico de uso dos símbolos do paciente enviados pelo tutor. Após importar o arquivo com o histórico de uso dos símbolos, o fonoaudiólogo deve entrar na tela de detalhes do paciente e pressionar o botão `Visualizar histórico`.

### 3.2.1.18 Importar arquivo com observações do paciente enviados pelo tutor

Este caso de uso permite que o fonoaudiólogo importe para o aplicativo as observações do paciente enviadas por e-mail pelo tutor do paciente. Esta importação é feita pelo fonoaudiólogo através do aplicativo padrão de e-mail do iOS.

### 3.2.1.19 Visualizar observações do paciente enviadas pelo tutor

Este caso de uso permite que o fonoaudiólogo visualize as observações do paciente enviadas pelo tutor. Após importar o arquivo com as observações do paciente, o fonoaudiólogo deve entrar na tela de detalhes do paciente e pressionar o campo de texto. O aplicativo irá abrir a tela de observações do paciente e irá demonstrar junto com as observações já criadas pelo fonoaudiólogo as observações que foram enviadas pelo tutor e importadas no aplicativo.

### 3.2.2 Diagrama de pacotes

Nesta seção são descritas as classes e as estruturas desenvolvidas que constituem o aplicativo. Para facilitar o entendimento de como as classes estão organizadas, na Figura 11 são ilustrados os pacotes e suas dependências, bem como as classes que os compõem.

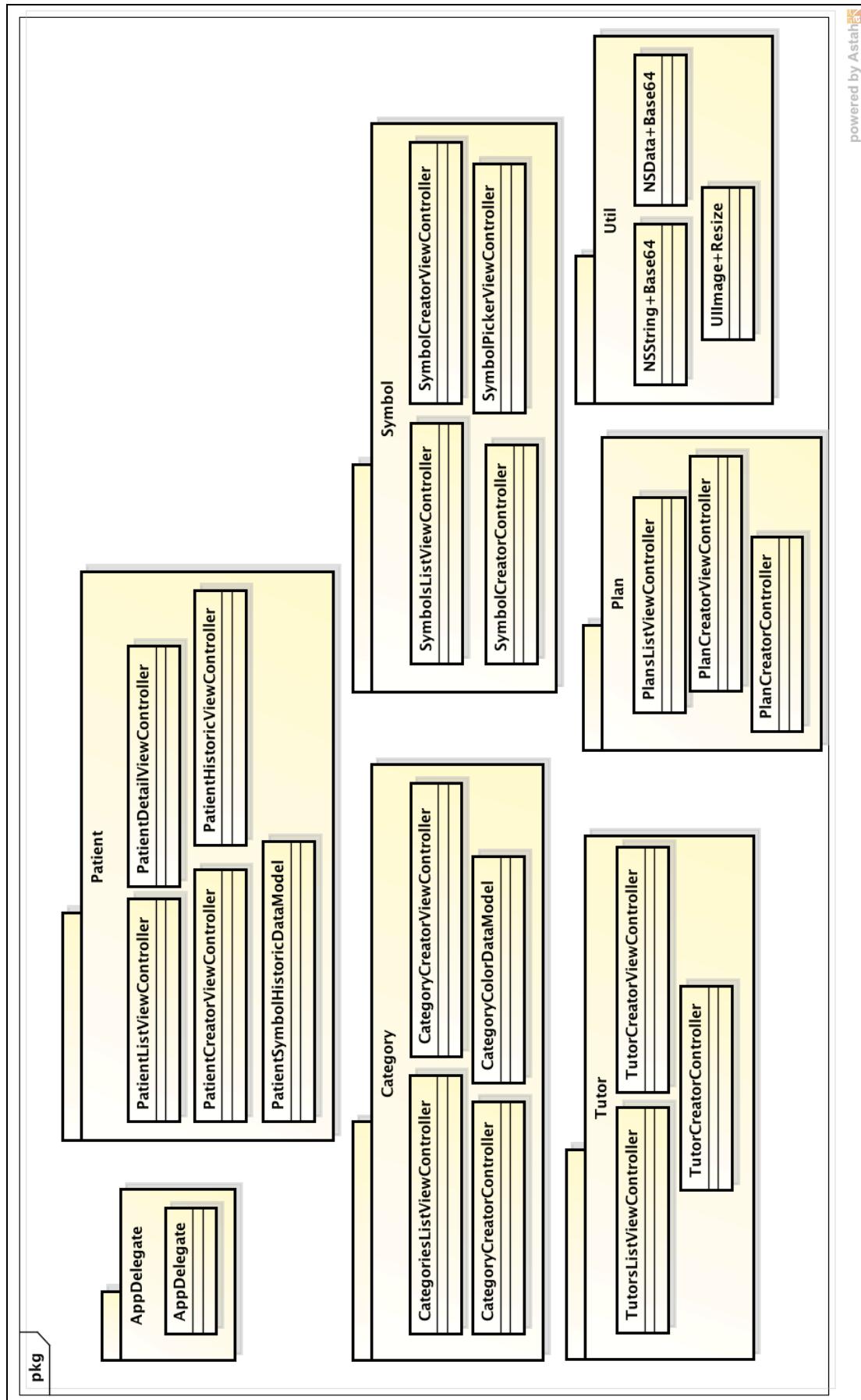


Figura 11 – Diagrama de pacotes

### 3.2.2.1 Pacote Patient

As classes do pacote `Patient`, ilustradas na Figura 12, são responsáveis por lidar com todas as informações relativas aos pacientes criados pelo fonoaudiólogo. As classes que pertencem a este pacote tem como funções criar, manter e mostrar ao usuário do aplicativo informações referentes ao paciente.

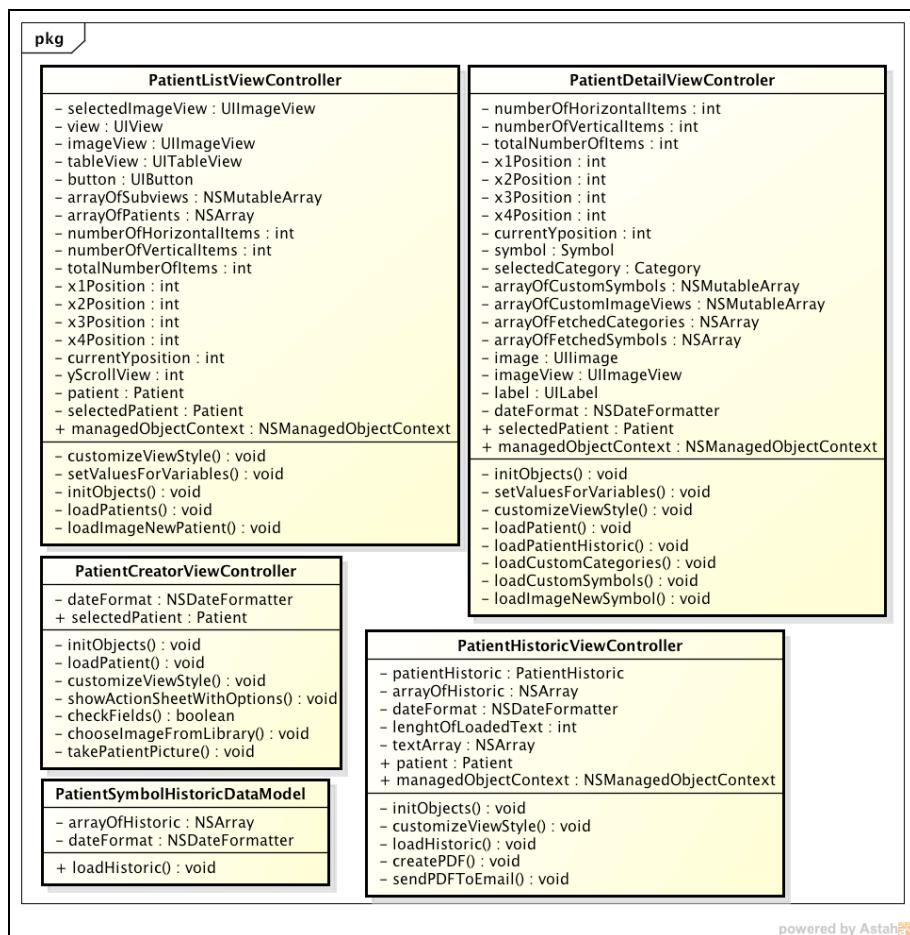


Figura 12 – Pacote Patient

A classe `PatientListViewController` herda da classe `UIViewController` e tem como funções listar ao fonoaudiólogo todos os pacientes criados por ele e também permitir que o fonoaudiólogo chame a tela para criar um novo paciente. Para listar todos os pacientes criados, a classe `PatientListViewController` utiliza o método `loadPatients`. Este método faz uma *request* a entidade `Patient` no *Core Data* para recuperar todos os pacientes já criados e mostrá-los na tela. Para chamar a tela de criação de pacientes, a classe `PatientListViewController` utiliza o método `presentModalViewController`: da sua superclasse `UIViewController`. Este método recebe como parâmetro a classe da tela de criação de pacientes e mostra a tela de forma modal para o usuário.

A classe `PatientDetailViewControler` também herda da classe `UIViewController` e tem como funções apresentar ao fonoaudiólogo todas as informações relacionadas ao paciente selecionado na tela que lista os pacientes. Esta classe permite que o fonoaudiólogo veja em uma mesma tela os dados pessoais do paciente, um resumo do histórico de observações criados para este paciente e todos os símbolos personalizados que o paciente possui. Para mostrar os dados pessoais do paciente, a classe `PatientDetailViewControler` utiliza o objeto `selectedPatient` que foi passado como parâmetro no momento da sua instanciação. O objeto `selectedPatient` é uma instância da classe `Patient` e possui como atributos os dados pessoais do paciente, como o nome, a data de nascimento, entre outros. Para mostrar o histórico de observações do paciente, a classe `PatientDetailViewControler` utiliza o método `loadPatientHistoric`. Este método faz uma *request* a entidade `PatientHistoric` do *Core Data* para recuperar as observações criadas pelo fonoaudiólogo e mostrá-las na tela de forma resumida. Para mostrar todos os símbolos personalizados do paciente, a classe `PatientDetailViewControler` utiliza o método `loadCustomSymbols`. Este método faz uma *request* a entidade `Symbol` do *Core Data* para recuperar os símbolos criados pelo fonoaudiólogo e mostrá-los na tela de acordo com a sua categoria.

A classe `PatientCreatorViewController` herda da classe `UIViewController` e tem como função permitir que o fonoaudiólogo crie ou edite um paciente. Para criar um novo paciente, a classe `PatientCreatorViewController` instancia um novo objeto da classe `Patient` com os dados informados na tela pelo fonoaudiólogo. Para fazer a edição de um paciente, a classe `PatientCreatorViewController` recebe como parâmetro uma instância da classe `Patient` e altera seus atributos de acordo os dados digitados pelo fonoaudiólogo na tela de criação de pacientes.

A classe `PatientHistoricViewController` também herda da classe `UIViewController` e tem como função permitir que o fonoaudiólogo visualize e crie observações para o paciente. Para permitir a visualização das observações, a classe `PatientHistoricViewController` utiliza o método `loadHistoric`. Este método faz uma *request* a entidade `PatientHistoric` do *Core Data* para recuperar as observações criadas pelo fonoaudiólogo e mostrá-las na tela. Para permitir a criação de novas observações, a classe `PatientHistoricViewController` instancia um novo objeto da classe `PatientHistoric` com os dados da observação informados na tela pelo fonoaudiólogo.

A classe `PatientSymbolHistoricDataModel` herda da classe `UITableViewController` e tem como função carregar a lista com o histórico de uso dos

símbolos do paciente. Para carregar a lista com o histórico de uso dos símbolos do paciente, a classe `PatientSymbolHistoricDataModel` utiliza o método `loadHistoric`. Este método faz uma *request* a entidade `PatientSymbolHistoric` do *Core Data* para recuperar as informações e mostrá-las na tela.

### 3.2.2.2 Pacote Category

As classes do pacote `Category`, ilustradas na Figura 13, são responsáveis por lidar com todas as informações relativas as categorias criadas pelo fonoaudiólogo. As classes que pertencem a este pacote tem como funções criar, manter e mostrar ao usuário do aplicativo informações referentes à categoria.

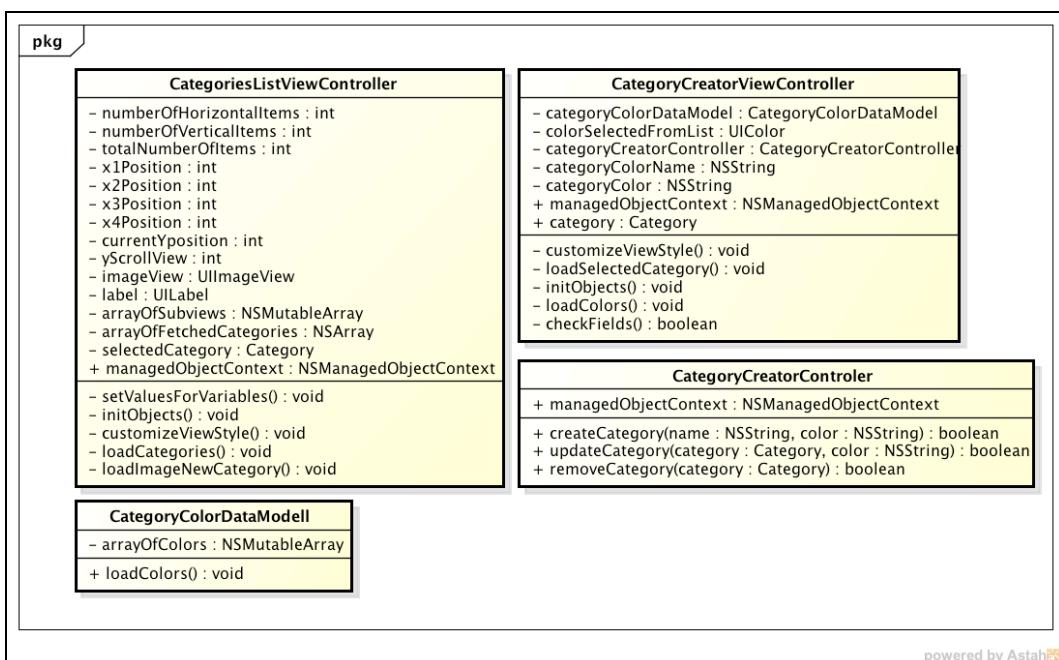


Figura 13 – Pacote Category

A classe `CategoriesListViewController` herda da classe `UIViewController` e tem como funções listar ao fonoaudiólogo todas as categorias criadas por ele e também permitir que o fonoaudiólogo chame a tela para criar uma nova categoria. Para listar todas as categorias criadas, a classe `CategoriesListViewController` utiliza o método `loadCategories`. Este método faz uma *request* a entidade `Category` no *Core Data* para recuperar todos as categorias já criadas e mostrá-las na tela. Para chamar a tela de criação de categorias, a classe `CategoriesListViewController` utiliza o método `presentModalViewController:` da sua superclasse `UIViewController`. Este método

recebe como parâmetro a classe da tela de criação de categorias e mostra a tela de forma modal para o usuário.

A classe `CategoryCreatorViewController` também herda da classe `UIViewController` e tem como função apresentar ao fonoaudiólogo a tela de criação e edição de categorias. Esta classe é utilizada somente como controladora dos componentes visuais que são apresentados ao usuário. Todas as rotinas de criação e edição estão escritas na classe `CategoryCreatorController`. Esta classe possui métodos que a classe `CategoryCreatorViewController` utiliza para criar e editar uma categoria. Para criar uma nova categoria, a classe `CategoryCreatorController` instancia um novo objeto da classe `Category` com os dados informados pelo fonoaudiólogo na tela controlada pela classe `CategoryCreatorViewController`. Para editar uma categoria, a classe `CategoryCreatorController` recebe como parâmetro uma instância da classe `Category` e altera seus atributos de acordo os dados digitados pelo fonoaudiólogo na tela de criação de categorias.

A classe `CategoryColorDataModel` herda da classe `UITableViewController` e tem como função carregar a lista das dezesseis cores disponíveis para a criação de uma nova categoria. Para carregar a lista de cores disponíveis, a classe `CategoryColorDataModel` utiliza o método `loadColors`. Este método instancia dezesseis objetos da classe `MyColor` que possuem como atributos dados de uma cor, como o nome e seu código *Red Green Blue* (RGB).

### 3.2.2.3 Pacote `Symbol`

As classes do pacote `Symbol`, ilustradas na Figura 14, são responsáveis por lidar com todas as informações relativas aos símbolos criados pelo fonoaudiólogo. As classes que pertencem a este pacote tem como funções criar, manter e mostrar ao usuário do aplicativo informações referentes aos símbolos.

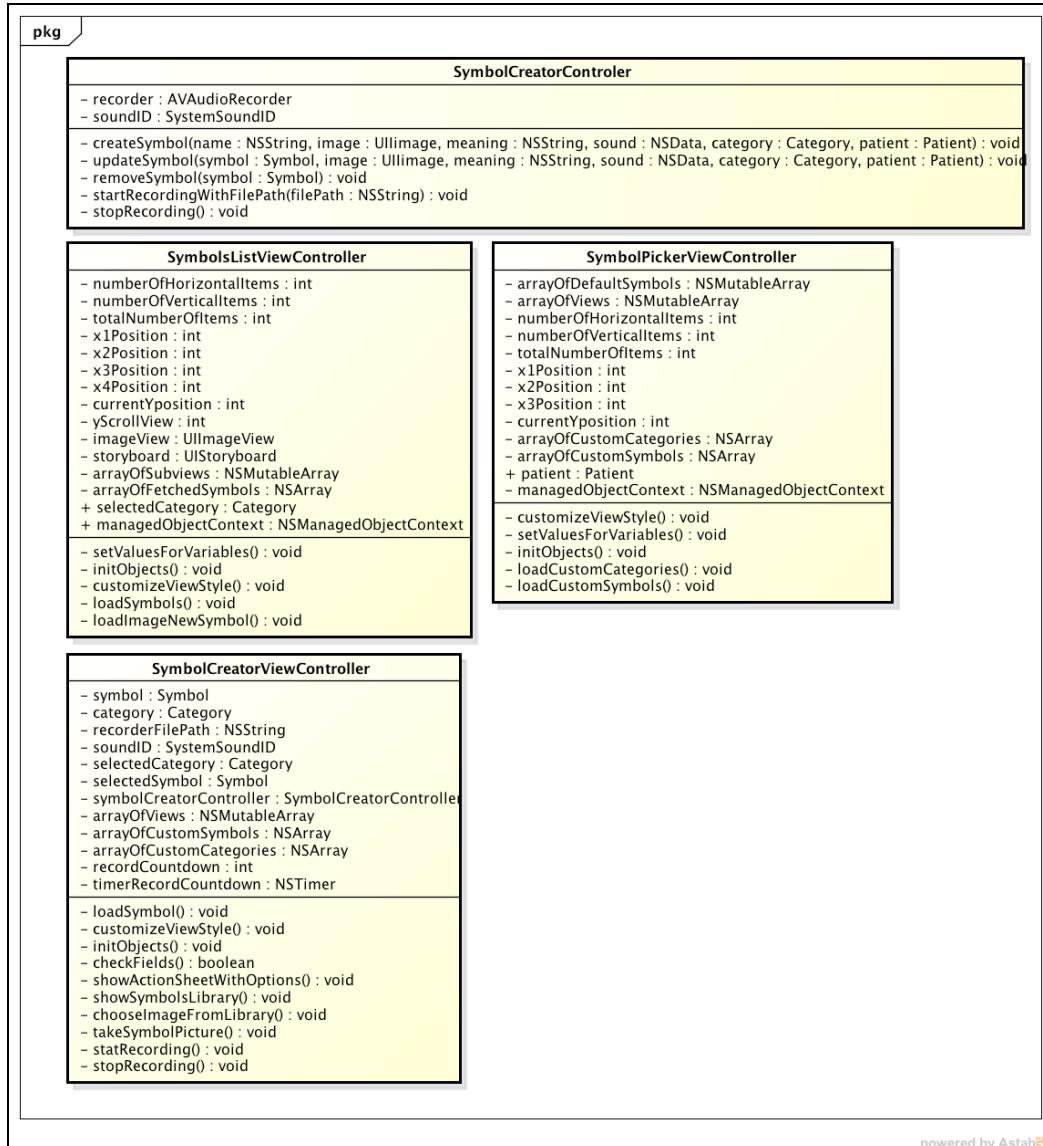


Figura 14 – Pacote Symbol

A classe `SymbolsListViewController` herda da classe `UIViewController` e tem como funções listar ao fonoaudiólogo todos os símbolos criados por ele e também permitir que o fonoaudiólogo chame a tela para criar um novo símbolo. Para listar todos os símbolos criados, a classe `SymbolsListViewController` utiliza o método `loadSymbols`. Este método faz uma *request* a entidade `Symbol` no *Core Data* para recuperar todos os símbolos já criados e mostrá-los na tela. Para chamar a tela de criação de símbolos, a classe `SymbolsListViewController` utiliza o método `presentModalViewController`: da sua superclasse `UIViewController`. Este método recebe como parâmetro a classe da tela de criação de símbolos e mostra a tela de forma modal para o usuário.

A classe `SymbolCreatorViewController` também herda da classe `UIViewController` e tem como função apresentar ao fonoaudiólogo a tela de criação e edição de símbolos. Esta classe é utilizada somente como controladora dos componentes

visuais que são apresentados ao usuário. Todas as rotinas de criação e edição estão escritas na classe `SymbolCreatorController`. Esta classe possui métodos que a classe `SymbolCreatorViewController` utiliza para criar e editar um símbolo. Para criar um novo símbolo, a classe `SymbolCreatorViewController` instancia um novo objeto da classe `Symbol` com os dados informados pelo fonoaudiólogo na tela controlada pela classe `SymbolCreatorViewController`. Para editar um símbolo, a classe `SymbolCreatorViewController` recebe como parâmetro uma instância da classe `Symbol` e altera seus atributos de acordo os dados digitados pelo fonoaudiólogo na tela de criação de símbolos.

A classe `SymbolPickerController` herda da classe `UIViewController` e tem como função apresentar ao fonoaudiólogo a tela de escolha de símbolos, utilizada principalmente no momento de criação das pranchas ou símbolos grandes de um plano. Esta classe inicialmente carrega todas as categorias de símbolos, e de acordo com a categoria selecionada, carrega todos os seus símbolos. Para carregar todas as categorias de símbolos, a classe `SymbolPickerController` utiliza o método `loadCustomCategories`. Este método faz uma *request* a entidade `Category` no *Core Data* para recuperar todas as categorias já criadas e mostrá-las na tela. Quando o fonoaudiólogo seleciona uma categoria, o método `loadCustomSymbols`: é invocado recebendo como parâmetro a categoria que foi selecionada. Este método faz uma *request* a entidade `Symbol` no *Core Data* para recuperar todos os símbolos da categoria selecionada e mostrá-los na tela.

### 3.2.2.4 Pacote Tutor

As classes do pacote `Tutor`, ilustradas na Figura 15, são responsáveis por lidar com todas as informações relativas aos tutores criados pelo fonoaudiólogo. As classes que pertencem a este pacote tem como funções criar, manter e mostrar ao usuário do aplicativo informações referentes aos tutores.

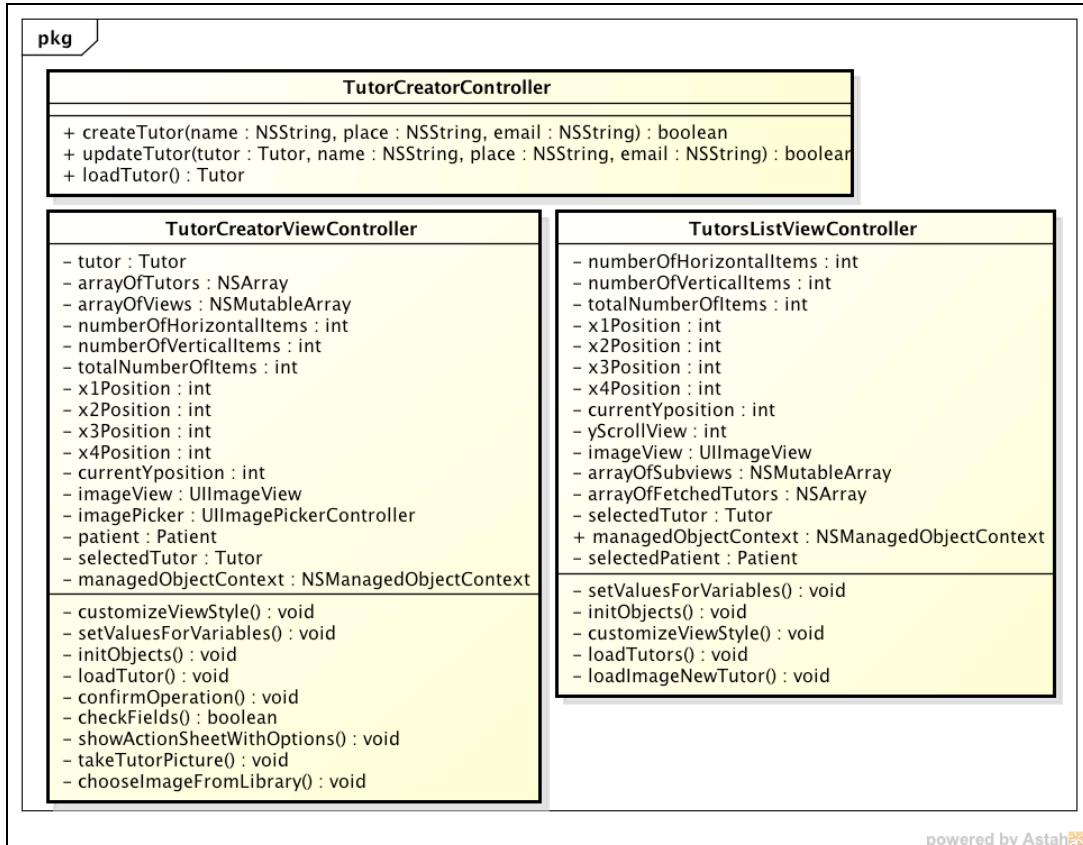


Figura 15 – Pacote Tutor

A classe `TutorsListViewController` herda da classe `UIViewController` e tem como funções listar ao fonoaudiólogo todas os tutores criados por ele e também permitir que o fonoaudiólogo chame a tela para criar um novo tutor. Para listar todos os tutores criados, a classe `TutorsListViewController` utiliza o método `loadTutors()`. Este método faz uma *request* a entidade `Tutor` no *Core Data* para recuperar todos os tutores já criados e mostrá-los na tela. Para chamar a tela de criação de tutores, a classe `TutorsListViewController` utiliza o método `presentModalViewControllerAnimated:` da sua superclasse `UIViewController`. Este método recebe como parâmetro a classe da tela de criação de tutores e mostra a tela de forma modal para o usuário.

A classe `TutorCreatorViewController` também herda da classe `UIViewController` e tem como função apresentar ao fonoaudiólogo a tela de criação e edição de tutores. Esta classe é utilizada somente como controladora dos componentes visuais que são apresentados ao usuário. Todas as rotinas de criação e edição estão escritas na classe `TutorCreatorController`. Esta classe possui métodos que a classe `TutorCreatorViewController` utiliza para criar e editar um tutor. Para criar um novo tutor, a classe `TutorCreatorViewController` instancia um novo objeto da classe `Tutor` com os dados informados pelo fonoaudiólogo na tela controlada pela classe

`TutorCreatorViewController`. Para editar um tutor, a classe `TutorCreatorController` recebe como parâmetro uma instância da classe `Tutor` e altera seus atributos de acordo os dados digitados pelo fonoaudiólogo na tela de criação de tutores.

### 3.2.2.5 Pacote Plan

As classes do pacote `Plan`, ilustradas na Figura 16, são responsáveis por lidar com todas as informações relativas aos planos criados pelo fonoaudiólogo. As classes que pertencem a este pacote tem como funções criar, manter e mostrar ao usuário do aplicativo informações referentes aos planos.

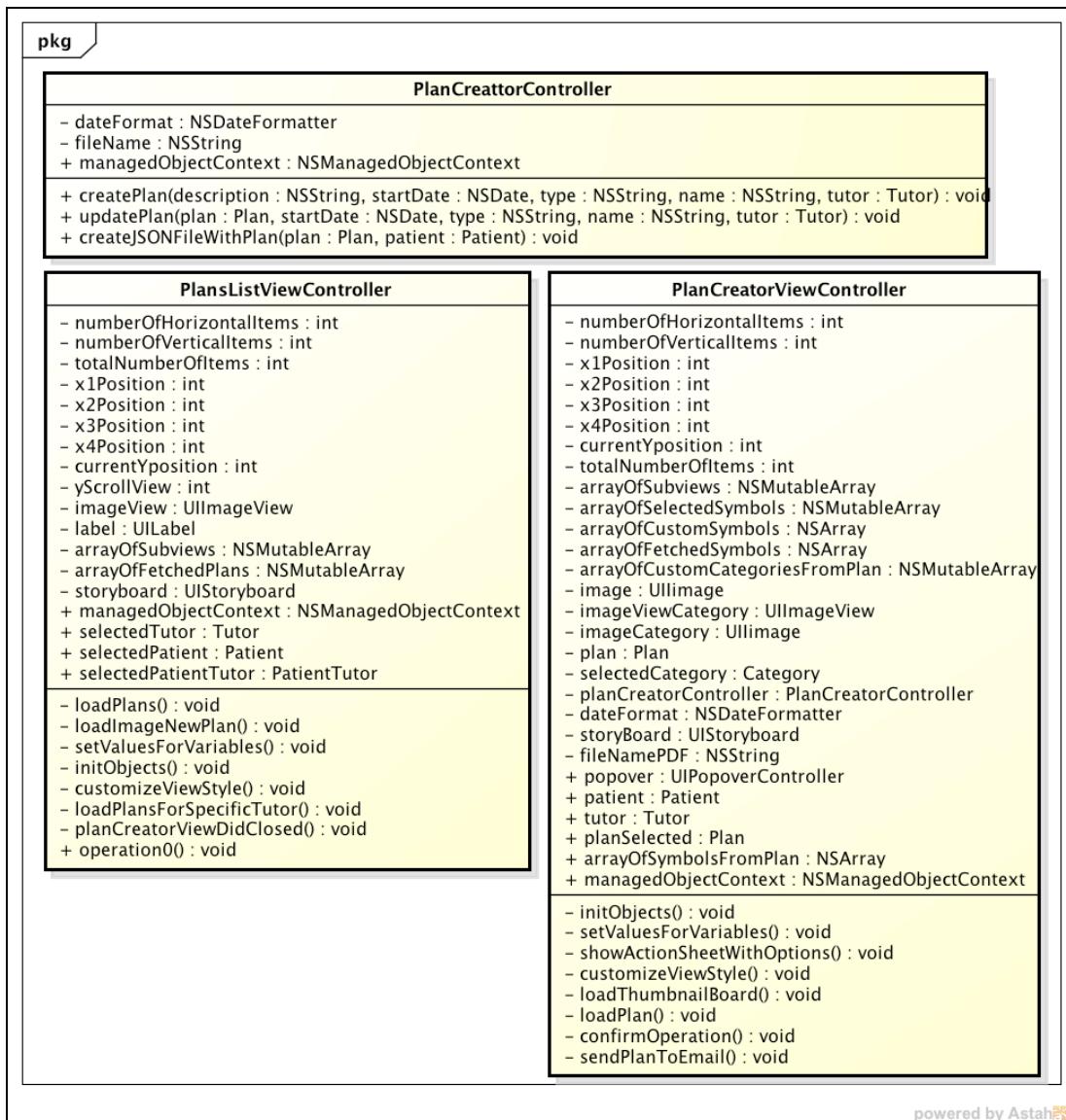


Figura 16 – Pacote Plan

A classe `PlansListViewController` herda da classe `UIViewController` e tem como funções listar ao fonoaudiólogo todos os planos criados por ele e também permitir que o fonoaudiólogo chame a tela para criar um novo plano. Para listar todos os planos criados, a classe `PlansListViewController` utiliza o método `loadPlans`. Este método faz uma *request* a entidade `Plan` no *Core Data* para recuperar todos os planos já criados e mostrá-los na tela. Para chamar a tela de criação de planos, a classe `PlansListViewController` utiliza o método `presentModalViewControllerAnimated:` da sua superclasse `UIViewController`. Este método recebe como parâmetro a classe da tela de criação de planos e mostra a tela de forma modal para o usuário.

A classe `PlanCreatorViewController` também herda da classe `UIViewController` e tem como função apresentar ao fonoaudiólogo a tela de criação e edição de planos. Esta classe é utilizada somente como controladora dos componentes visuais que são apresentados ao usuário. Todas as rotinas de criação e edição estão escritas na classe `PlanCreatorController`. Esta classe possui métodos que a classe `PlanCreatorViewController` utiliza para criar e editar um plano. Para criar um novo plano, a classe `PlanCreatorController` instancia um novo objeto da classe `Plan` com os dados informados pelo fonoaudiólogo na tela controlada pela classe `PlanCreatorViewController`. Para editar um plano, a classe `PlanCreatorController` recebe como parâmetro uma instância da classe `Plan` e altera seus atributos de acordo os dados digitados pelo fonoaudiólogo na tela de criação de planos.

### 3.2.2.6 Pacote Util

As classes do pacote `Util`, ilustradas na Figura 17, são as classes que fornecem as funções utilitárias do aplicativo.

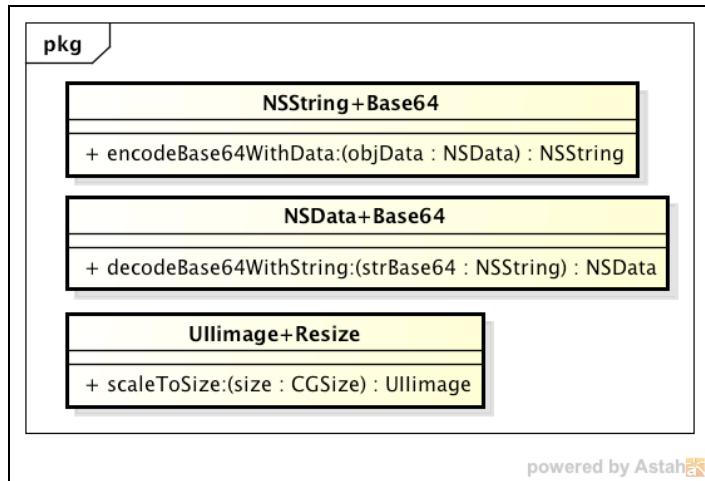


Figura 17 – Pacote Util

A classe `NSString+Base64` acrescenta a função `encodeBase64WithData` à objetos do tipo `NSString`. Esta função permite codificar objetos binários que são do tipo `NSData` em um formato adequado para a transmissão via e-mail. Esta função é utilizada no aplicativo para realizar a transmissão dos símbolos de um plano, que são compostos principalmente por um arquivo de imagem e um arquivo de áudio.

A classe `NSData+Base64` acrescenta a função `decodeBase64WithString` à objetos do tipo `NSData`. Esta função permite decodificar objetos do tipo `NSString` para um formato binário. Esta função é utilizada no aplicativo para realizar a decodificação dos símbolos de um plano que são recebidos via e-mail.

A classe `UIImage+Resize` acrescenta a função `scaleToSize` à objetos do tipo `UIImage`. Esta função permite reduzir o tamanho de um objeto `UIImage` mantendo sua proporção.

### 3.2.2.7 Pacote AppDelegate

O pacote `AppDelegate`, ilustrado na Figura 18, contém apenas uma classe, sendo esta instanciada no momento em que o aplicativo inicia. A principal tarefa desta classe é fornecer a janela inicial na qual será desenhado o conteúdo do aplicativo e gerenciar as transições de estado do mesmo.

Esta classe também é responsável pela gerência do banco de dados do *Core Data*. Esta gerência é feita através dos métodos `saveContext` e das variáveis `managedObjectContext`, `managedObjectModel` e `persistentStoreCoordinator` que são padrões para todo tipo de aplicativo que utiliza o framework *Core Data*.

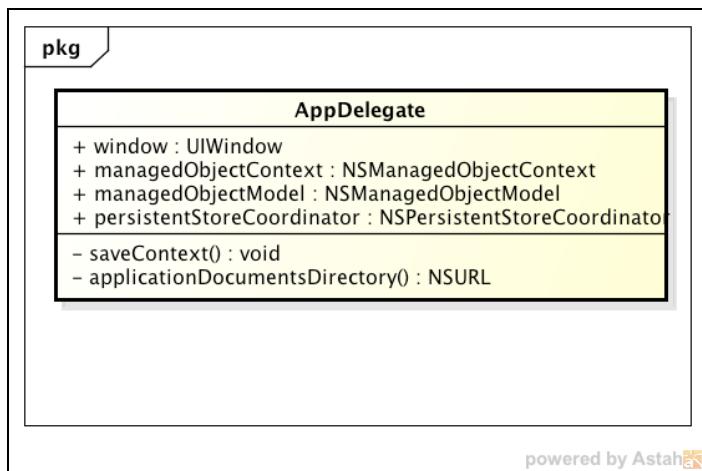


Figura 18 – Pacote AppDelegate

### 3.2.2.8 Pacote Core Data Classes

As classes do pacote `Core Data Classes` são as classes que representam o modelo das entidades no *Core Data*. Estas classes são geradas automaticamente pelo Xcode no momento em que uma nova entidade no *Core Data* é criada e possuem apenas os métodos que fornecem acesso aos atributos destas entidades.

### 3.2.3 Diagramas de sequência

Os diagramas de sequência tem como finalidade demonstrar a troca de mensagens entre as classes criadas, facilitando a implementação dos casos de uso. Nesta seção são apresentados os diagramas de sequência para os casos de uso `Criar símbolos` (UC04) e `Criar planos` (UC05).

#### 3.2.3.1 Diagrama de sequência Criar símbolos

O caso de uso `Criar símbolos` tem como requisito que o fonoaudiólogo já tenha criado uma categoria. Desta forma, quando o fonoaudiólogo entra na tela para criar um novo símbolo, o campo `Categoria` já vem preenchido com a categoria que foi selecionada para

iniciar a criação do símbolo.

Para criar um símbolo é necessário que o fonoaudiólogo informe o nome do símbolo e o seu significado. Após informar esses dois campos o fonoaudiólogo pressiona uma imagem padrão, para escolher uma imagem para ser associada ao símbolo. Ao pressionar esta imagem, o evento `touchesBegan:withEvent:` é invocado pelo sistema operacional. Este evento tem a função de capturar qualquer toque feito na tela do dispositivo. Quando este evento é invocado, o aplicativo apresenta ao usuário três opções para associar uma imagem ao símbolo. Se a opção de capturar uma nova foto for escolhida, o aplicativo apresenta a tela padrão de captura de fotos do iOS ao usuário. Assim que o fonoaudiólogo capturar uma foto a imagem padrão é substituída pela imagem capturada pela câmera do dispositivo. Após informar os dois campos e associar uma imagem ao símbolo, o fonoaudiólogo tem a opção de gravar uma mensagem de áudio para ser associada ao símbolo. Para realizar a gravação do áudio, é utilizado a classe `SymbolCreatorController`, que tem a função de gravar o áudio utilizando os métodos do framework `AVFoundation`. Após a gravação do áudio, o aplicativo apresenta mensagem ao fonoaudiólogo informando que o áudio foi gravado com sucesso. Sendo assim, resta ao fonoaudiólogo confirmar a gravação do símbolo pressionando o botão `OK` na tela de criação de símbolos. Assim que o botão é pressionado, o método `createSymbol:withName:andImage:andMeaning:andSound:andCategory:andPatient` da classe `SymbolCreatorController` é invocado com todos os parâmetros necessários para a gravação do símbolo na base de dados.

A Figura 19 apresenta o diagrama de sequência `Criar símbolos`.

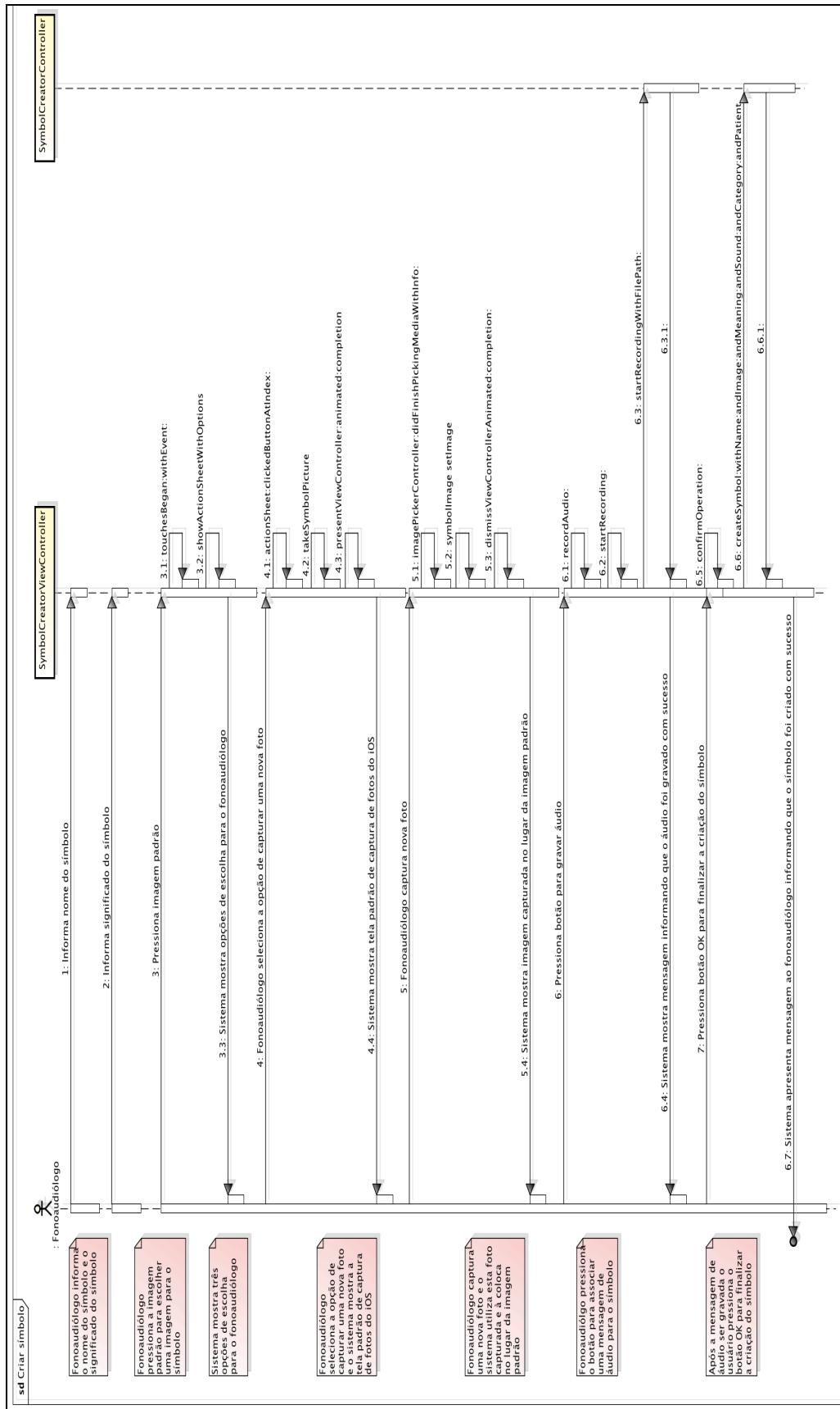


Figura 19 – Diagrama de sequência Criar símbolos

### 3.2.3.2 Diagrama de sequência Criar planos

O caso de uso `Criar planos` tem como requisito que o fonoaudiólogo já tenha criado um tutor e pelo menos um símbolo. Desta forma, quando o fonoaudiólogo entra na tela para criar um novo plano, o campo nome do tutor já vem preenchido com o tutor que foi selecionado para iniciar a criação do plano.

Para criar um plano, inicialmente é necessário que o fonoaudiólogo informe o nome do plano. Após informar o nome do plano, o fonoaudiólogo pressiona o campo tipo do plano para escolher o tipo do plano. Ao pressionar este campo, o evento `textFieldDidBeginEditing:` é invocado pelo sistema operacional. Este evento tem a função de capturar qualquer toque feito em um campo de texto. Quando este evento é invocado, o aplicativo apresenta ao usuário duas opções para criar um plano. Se a opção Símbolos Grandes for escolhida, o método `presentModalViewController:animated:` é chamado. Este método tem a função de apresentar ao fonoaudiólogo a tela utilizada para escolher o símbolo que fará parte deste plano. Após exibição desta nova tela, o fonoaudiólogo pressiona a imagem padrão para escolher o símbolo que fará parte deste plano, assim que a imagem padrão é pressionada o aplicativo apresenta ao usuário um *popover* com todos os símbolos disponíveis para escolha. O fonoaudiólogo faz então a escolha do símbolo e o aplicativo substitui a imagem padrão pelo símbolo selecionado. Após isso, o fonoaudiólogo pressiona o botão `Concluir` e o aplicativo inicia a geração do arquivo `.pdf` deste plano através do método `generatePDF`. Depois de gerar o `.pdf`, o método `dismissModalViewControllerAnimated:` é invocado e o aplicativo volta a apresentar a tela de criação de planos. Nesta tela o fonoaudiólogo preenche o campo descrição do plano, que é o último campo necessário para criar o plano e pressiona o botão `OK`. Assim que o botão é pressionado, o método `createPlan:withDescription:andStartDate:andType:andName:andTutor:andBoardPath:` da classe `PlanCreatorController` é invocado com todos os parâmetros necessários para a gravação do plano na base de dados. Após a gravação do plano, o aplicativo pergunta ao fonoaudiólogo se ele deseja enviar o plano para o e-mail do paciente. Caso a resposta for afirmativa, o aplicativo mostra a tela padrão de envio de e-mails do iOS com o PDF do plano anexado. Quando o usuário pressionar o botão `Enviar` na tela de envio de e-mails o aplicativo volta a exibir a tela de criação de planos e emite um aviso sonoro ao usuário indicando que o plano foi enviado com sucesso.

A Figura 20 apresenta o diagrama de sequência `Criar planos`.

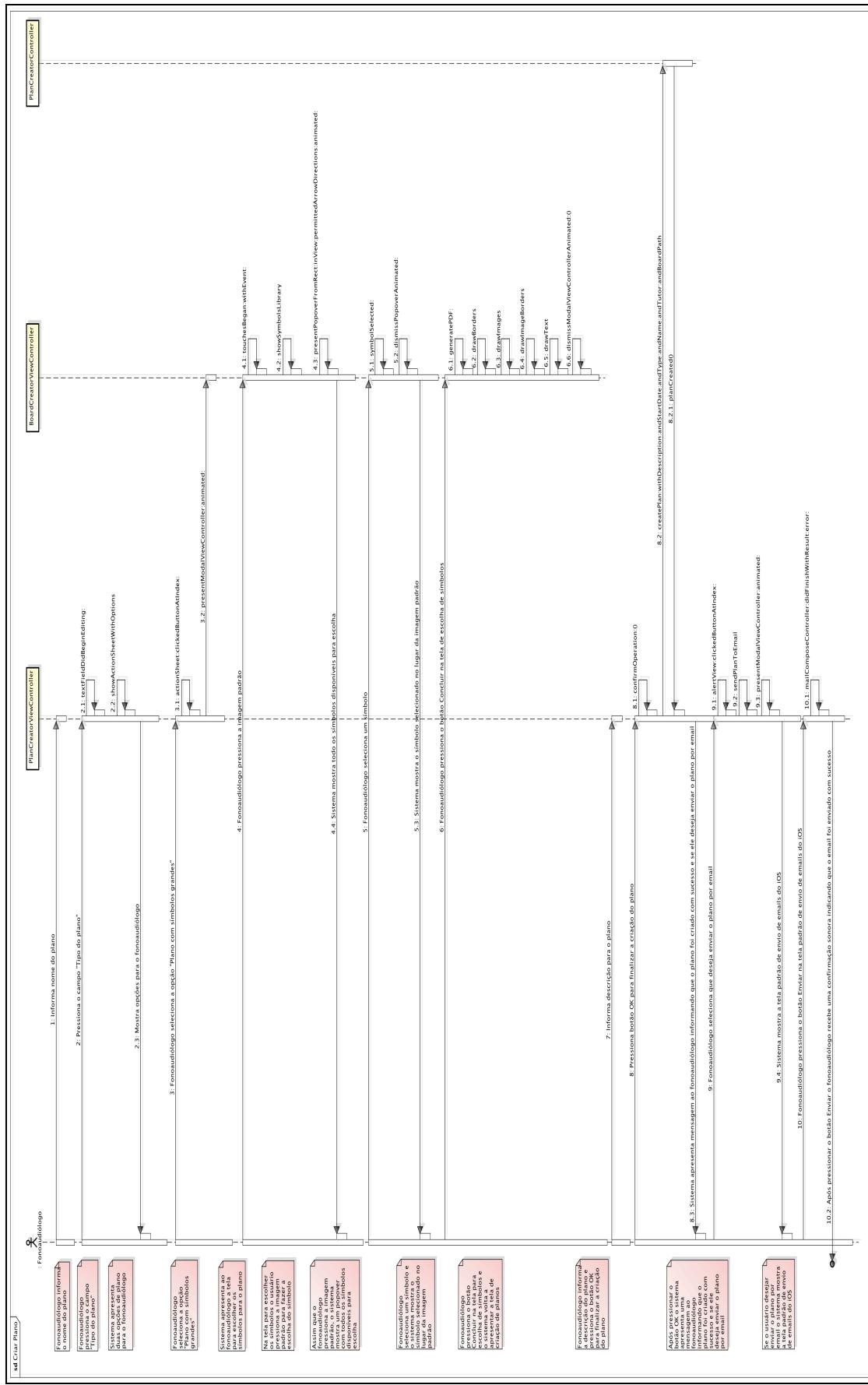


Figura 20 – Diagrama de sequência Criar planos

### 3.2.4 Diagramas de atividades

O diagrama de atividades pode ser visualizado na Figura 21 e demonstra em uma visão de alto nível a utilização do aplicativo pelos três atores envolvidos no processo.

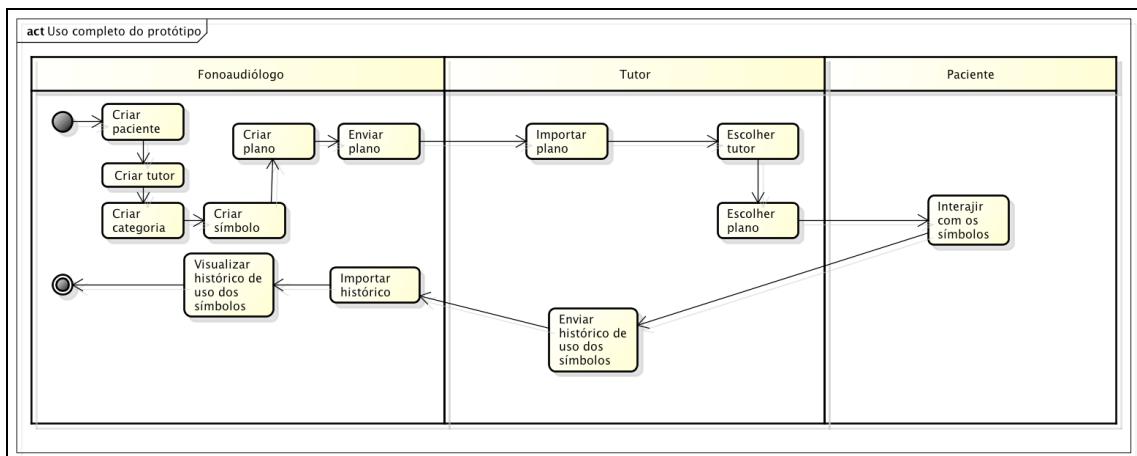


Figura 21 – Diagrama de atividades do aplicativo

Conforme pode ser visualizado na Figura 21, as atividades no aplicativo iniciam-se com o uso do aplicativo pelo fonoaudiólogo. O fonoaudiólogo precisa executar uma série de etapas para que o plano seja feito de forma correta e enviado ao paciente para uso posterior. A primeira etapa a ser cumprida pelo fonoaudiólogo é a criação de um paciente. Após criar um paciente o fonoaudiólogo deve criar um tutor e associar este tutor ao paciente. Depois de criar um tutor, o fonoaudiólogo inicia o processo de criação um símbolo que fará parte do plano. Para criar um símbolo é necessário que antes, o fonoaudiólogo crie uma categoria da qual o símbolo fará parte. Após criar a categoria e o símbolo, o fonoaudiólogo inicia a criação de um plano e faz o envio deste plano para o dispositivo do paciente.

Quando o paciente receber este plano em seu dispositivo, é papel do tutor auxilia-lo na importação do plano para o aplicativo. Após a importação do plano, cabe também ao tutor auxiliar o paciente na escolha do tutor, já que um paciente pode ter diversos tutores e também escolher qual plano do tutor selecionado o paciente deseja utilizar, já que é permitido também que um tutor tenha diversos planos diferentes.

Após a escolha do plano, é papel do paciente interagir com o plano através dos símbolos que foram escolhidos pelo fonoaudiólogo nas primeiras etapas de uso do aplicativo. Conforme o paciente interage com os símbolos, um histórico do uso destes símbolos é registrado. Este histórico pode ser visualizado pelo tutor do paciente, e se o tutor desejar, é possível enviar este histórico por e-mail para que o fonoaudiólogo faça a importação deste histórico e visualize-o.

### 3.2.5 Diagrama de classes das entidades do aplicativo

Para realizar a persistência das informações criadas no aplicativo de forma consistente e organizada, foi utilizado o *framework Core Data*.

Este *framework*, criado e mantido pela Apple implementa um mapeamento de objeto relacional, que fornece as funcionalidades básicas de gerenciamento de dados. Ou seja, o *framework Core Data* tem como objetivo atuar como uma camada em cima de um banco de dados. Esta camada tem como principal função abstrair os comandos SQL a um paradigma de orientação a objetos.

O *framework Core Data* também permite que o desenvolvedor abstraia as regras de gerência e modelagem de dados.

A abstração da modelagem de dados é obtida através da ferramenta de modelagem integrada ao Xcode. Esta ferramenta permite que o desenvolvedor descreva os dados que compõem o sistema por meio de um modelo de entidade e relacionamentos. Após descrever todos os dados através do modelo, o Xcode gera as classes e arquivos necessários para a gerência e persistência dos dados. Esta camada de persistência gerada pelo *Core Data* abstrai também do desenvolvedor os detalhes de implementação, isto possibilita que o desenvolvedor na aplicação interaja somente com os objetos do banco de dados, sem a necessidade de executar comandos SQL. Portanto, ao invés do desenvolvedor fazer *queries* para excluir um registro da base de dados ou *updates* para alterar um registro, é possível com o *Core Data* que ele apague um objeto que representa um registro ou altere uma propriedade desse objeto para realizar a operação de *update*.

As classes resultantes da modelagem das entidades do aplicativo podem ser visualizadas nas figuras 22 e 23. A Figura 22 apresenta as classes das entidades relativas as informações que o fonoaudiólogo cria no sistema e a Figura 23 apresenta as classes das entidades relativas as informações que o tutor e o paciente mantém no sistema após a importação dos planos. O detalhamento de cada entidade pode ser visualizado no apêndice B.

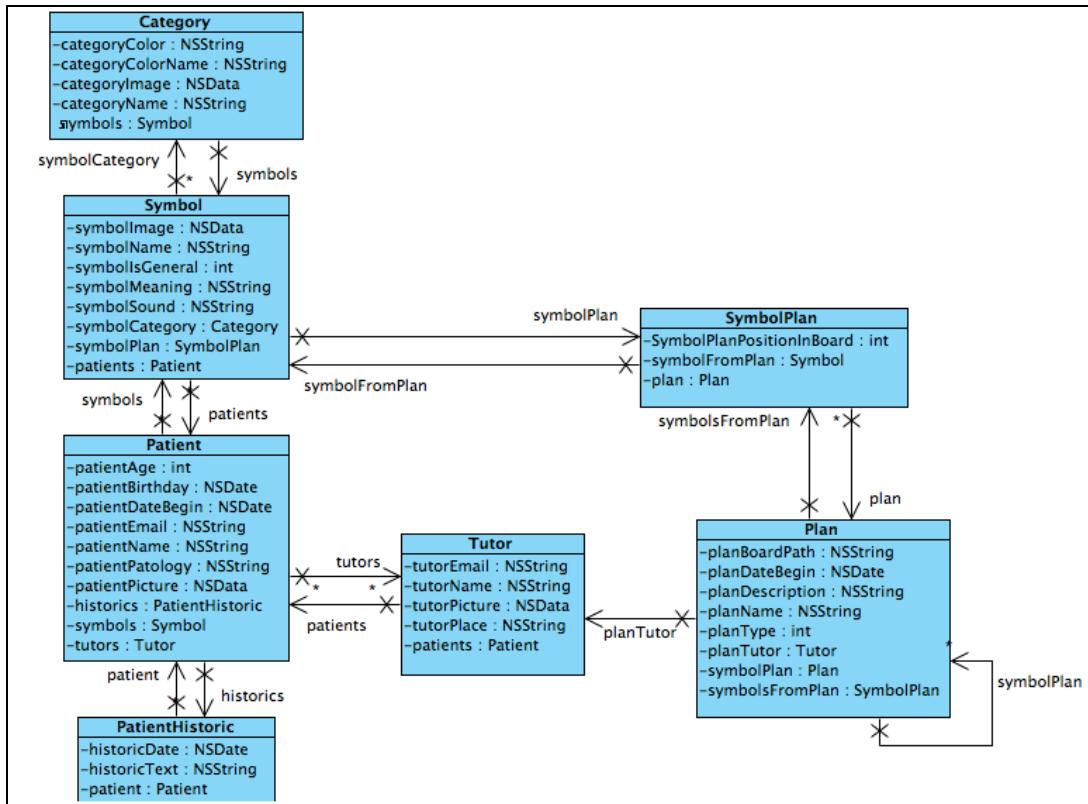


Figura 22 – Diagrama de classes das entidades relativas ao fonoaudiólogo

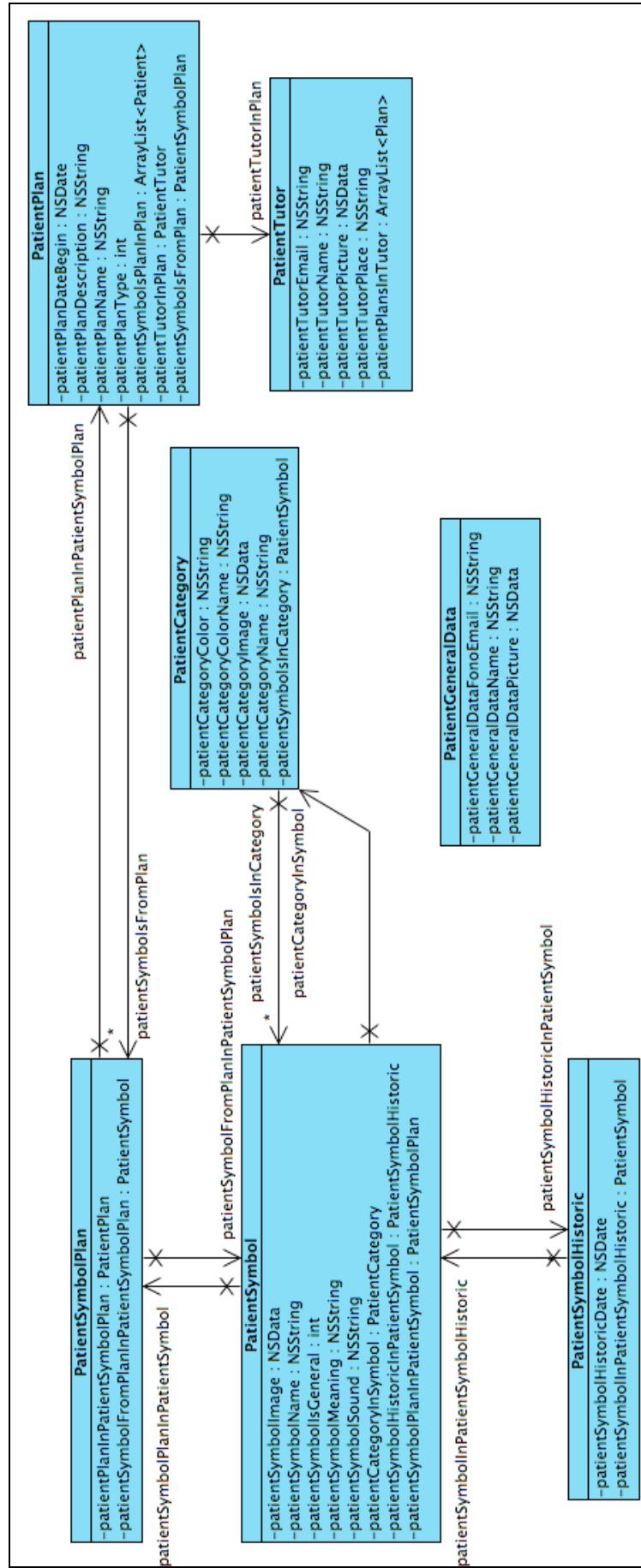


Figura 23 – Diagrama de classes das entidades relativas ao tutor e ao paciente

### 3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas para a implementação do aplicativo e a operacionalidade desta implementação.

#### 3.3.1 Técnicas e ferramentas utilizadas

Para realizar a implementação do aplicativo foi utilizada a linguagem de programação Objective-C em conjunto com o iOS SDK 6. O ambiente de desenvolvimento escolhido foi a versão 4.5 do Xcode. Para realizar os testes do aplicativo foi utilizado o iPhone Simulator e três modelos diferentes do dispositivo iPad 2 rodando a versão 6.0 do sistema operacional iOS.

A documentação completa da implementação, gerada a partir da ferramenta Doxygen, está disponível no repositório do trabalho (FABENI, 2012) e é hospedado pelo Dropbox.

#### 3.3.2 Criação do arquivo JSON com os dados do plano

Uma das principais funcionalidades do aplicativo é o envio dos planos elaborados pelo fonoaudiólogo para os pacientes via e-mail. Para realizar o envio dos planos é necessário que as informações criadas pelo fonoaudiólogo sejam compactadas em um único arquivo e também que estas informações estejam bem organizadas para que na importação dos planos pelo tutor nenhuma informação seja perdida.

Antes de realizar o envio das informações, o aplicativo armazena os dados que serão enviados em um único arquivo no formato JavaScript Object Notation (JSON). O formato JSON foi escolhido devido a sua simplicidade e ao seu formato leve que é apropriado para o intercâmbio de informações. Além destas características, o formato JSON possui compatibilidade nativa com o iOS que possui bibliotecas capazes de fazer o *parse* de qualquer tipo de informação neste formato.

O método `createJSONFileWithPlan:andPatient:` da classe `PlanCreatorController` é responsável por efetuar a recuperação de informações da base de dados e criar o arquivo JSON para ser enviado via e-mail.

Este método recebe como parâmetro o plano que deve ser enviado por e-mail e o paciente que deve receber este plano. Inicialmente, é possível visualizar que na linha 91, o método armazena os dados do plano e do paciente passados como parâmetro em uma instância da classe `NSDictionary`. Com esta instância, o método de classe `dataWithJSONObject:options:` da classe `NSData` é invocado para fazer o *parse* do `NSDictionary` em um arquivo JSON. O retorno do método `dataWithJSONObject:options:` é então armazenado em uma instância do `NSMutableArray` de nome `arrayOfjsonData`, conforme pode ser visualizado na linha 107. Estas etapas podem ser visualizadas no Quadro 2.

```

89. NSMutableArray *arrayOfjsonData = [[NSMutableArray
                                         alloc] initWithCapacity:0];

90. NSError *error;

91. NSDictionary *infoPlan = [NSDictionary dictionaryWithObjectsAndKeys:
92.                               [plan planDescription], @"plan.description",
93.                               [dateFormat stringFromDate:[plan
94.                               planDateBegin]], @"plan.datebegin",
95.                               [plan planType], @"plan.type", [plan planName],
96.                               @"plan.name",
97.                               [[plan planTutor] tutorName], @"tutor.name",
98.                               [[plan planTutor] tutorEmail], @"tutor.email",
99.                               [[plan planTutor] tutorPlace], @"tutor.place",
100.                              [[plan planTutor] tutorPicture], @"tutor.picture",
101.                             [patient patientName], @"patient.name",
102.                             @"fono@gmail.com", @"fono.email",
103.                             [patient patientPicture]],
104.                             @"patient.picture", nil];

106. NSData* jsonDataPlan = [NSJSONSerialization dataWithJSONObject:infoPlan
                           options:NSJSONWritingPrettyPrinted error:nil];

107. [arrayOfjsonData addObject:jsonDataPlan];

```

Quadro 2 – Primeira etapa do método de criação do arquivo JSON

Após formatar os dados do plano e do paciente em formato JSON, o aplicativo recupera da base de dados as informações referentes aos símbolos que pertencem ao plano passado como parâmetro. Para isso, é feito uma *request* ao *Core Data*. Esta *request* pode ser visualizada na linha 113, tem como destino a entidade `SymbolPlan` e recebe como parâmetro o plano que foi passado como parâmetro no método. Se a *request* for bem sucedida, ou seja, existirem símbolos associados ao plano, uma iteração sobre todos os símbolos encontrados é realizada. Esta iteração, que ocorre entre as linhas 115 e 118, tem como objetivo armazenar os dados em outra instância da classe `NSDictionary` para formatação dos dados encontrados em formato JSON através de nova chamada ao método `dataWithJSONObject:options:` da classe `NSData`. O resultado do método `dataWithJSONObject:options:` é então armazenado na instância do `NSMutableArray` já utilizado anteriormente. A segunda etapa

do método descrito acima pode ser visualizada no Quadro 3.

```

108. NSArray *arraySymbolsFromPlan;
109. NSFetchedResultsController *fetchRequest = [[NSFetchedResultsController alloc] init];
110. NSEntityDescription *entity = [NSEntityDescription
111.                                         entityForName:@"SymbolPlan"
112.                                         inManagedObjectContext:managedObjectContext];
113. [fetchRequest setPredicate:predicate];
114. arraySymbolsFromPlan = [managedObjectContext
115.                           executeFetchRequest:fetchRequest error:&error];
116. for (int i = 0; i < [arraySymbolsFromPlan count]; i++) {
117.     NSString *planSymbolName = [[[arraySymbolsFromPlan
118.                                 objectAtIndex:i] symbolFromPlan] symbolName];
119.     NSString *planSymbolMeaning = [[[arraySymbolsFromPlan
120.                                 objectAtIndex:i] symbolFromPlan] symbolMeaning];
121.     NSString *planSymbolSound = [[[arraySymbolsFromPlan
122.                                 objectAtIndex:i] symbolFromPlan] symbolSound];
123.     NSDictionary *infoSymbolCategory = [NSDictionary
124.                                         dictionaryWithObjectsAndKeys:
125.                                             planSymbolName, @"symbol.name",
126.                                             planSymbolMeaning, @"symbol.meaning",
127.                                             planSymbolImage, @"symbol.image",
128.                                             planSymbolSoundData, @"symbol.sound",
129.                                             nil];
130.     NSData* jsonDataSymbolCategory = [NSJSONSerialization
131.                                         dataWithJSONObject:infoSymbolCategory options:NSJSONWritingPrettyPrinted
132.                                         error:&error];
133.     [arrayOfjsonData addObject:jsonDataSymbolCategory];
}

```

Quadro 3 – Segunda etapa do método de criação do arquivo JSON

A ultima etapa do método `createJSONFileWithPlan:andPatient:` é percorrer o `NSMutableArray` que contém os dados formatados em JSON e para cada elemento desse `array` salvar o conteúdo dele em um único arquivo no formato `.txt`. Este arquivo `.txt` é necessário para que depois de terminada a execução do método `createJSONFileWithPlan:andPatient:` o conteúdo do `.txt` seja enviado via e-mail em um arquivo no formato `.psyb`. A terceira e ultima etapa do método pode ser visualizada entre as linhas 122 e 125 do Quadro 4.

```

122. for (int i = 0; i < [arrayOfjsonData count]; i++) {
123.     NSData *jsonData = [arrayOfjsonData
124.                           objectAtIndex:i];
125.     NSString *contents = [[NSString
126.                               alloc] initWithData:jsonData
127.                               encoding:NSUTF8StringEncoding];
128.     [textToWrite appendString:[NSString
129.                               stringWithFormat:@"%@%@", contents]];
130.     [textToWrite writeToFile:fileName atomically:NO
131.                           encoding:NSUTF8StringEncoding error:&error];
}

```

Quadro 4 – Terceira etapa do método de criação do arquivo JSON

Um trecho da estrutura do arquivo `.psyb` em formato JSON pode ser visualizada na

Figura 24. Os outros arquivos gerados e enviados via e-mail pelo aplicativo, no formato .phist e .pobs, também obedecem a mesma formatação, mudando apenas os nomes dos campos.

```
"symbol.sound": "NULL",
"category.name": "Verbos",
"category.colordesc": "Red",
"symbol.meaning": "Andar",
"category.color": "0 1 0 1",
"symbol.name": "Andar",
"category.image": "VBORw0KGaoAAAANSUhEUqAAAAEAAAABCAYAAAAIfcsJAAAADUIEQVQIHVNq+MvwHwAEAQHVMH0Y7gAAAABJRUSErkJagg==",
"symbol.position": "1"
```

Figura 24 – Estrutura do arquivo .psyb

### 3.3.3 Gravação de áudio

O método `startRecordingWithFilePath:` da classe `SymbolCreatorController` é responsável por realizar a gravação do áudio que será associado ao símbolo no momento de sua criação. Este método recebe como parâmetro o caminho aonde o áudio deve ser gravado e utiliza métodos das classes integrantes do *framework* AVFoundation.

A primeira etapa para realizar a gravação do áudio é obter uma instância da classe `AVAudioSession` através do método de classe `sharedInstance`. Os métodos da classe `AVAudioSession` não são utilizados diretamente na gravação. Esta classe é instanciada apenas para definir o contexto da sessão de áudio que será gravada. A primeira etapa descrita acima pode ser visualizada no Quadro 5 entre as linhas 141 e 144.

```
141. AVAudioSession *audioSession = [AVAudioSession sharedInstance];
142. NSError *error;
143. [audioSession setCategory:AVAudioSessionCategoryPlayAndRecord error:&error];
144. [audioSession setActive:YES error:&error];
```

Quadro 5 – Primeira etapa do método de gravação de áudio

Após obter a instância da classe `AVAudioSession`, é necessário definir os parâmetros necessários para a gravação, esses parâmetros são passados através de uma instância da classe `NSTMutableDictionary`, uma estrutura de dados composta por chaves e valores. Cada chave do `NSTMutableDictionary` contém como valor um parâmetro que será passado à instância da classe `AVAudioRecorder`.

O primeiro parâmetro passado para o `NSTMutableDictionary` é o `AVFormatIDKey`, esse parâmetro define o formato do áudio que será gravado. O formato de áudio escolhido foi o `IMA4` devido a sua alta taxa de compressão e também a sua compatibilidade nativa com o iOS, o que facilita a sua decodificação pela CPU do dispositivo.

O segundo parâmetro passado para o `NSMutableDictionary` é o `AVSampleRateKey`, esse parâmetro define a frequência de amostragem do áudio a ser gravado. O valor da frequência foi fixado em 12 kHz, que se demonstrou satisfatória para o propósito do trabalho. O iOS suporta frequências de amostragem de até 32 kHz, porém isso iria aumentar o tamanho final do arquivo de áudio.

O terceiro e último parâmetro passado para o `NSMutableDictionary` é o `AVNumberOfChannelsKey`, esse parâmetro define o número de canais que será usado para gravar o áudio. O valor do número de canais foi fixado em 1, pois o iPad possui apenas um microfone para captura de áudio. A terceira etapa descrita acima pode ser visualizada entre as linhas 145 e 148 no Quadro 6.

```
145. NSMutableDictionary *recordSettings = [[NSMutableDictionary alloc] init];
146. [recordSettings setValue:[NSNumber numberWithInt:kAudioFormatAppleIMA4]
147.           forKey:AVFormatIDKey];
148. [recordSettings setValue:[NSNumber numberWithFloat: 12000]
149.           forKey:AVSampleRateKey];
150. [recordSettings setValue:[NSNumber numberWithInt: 1]
151.           forKey:AVNumberOfChannelsKey];
```

Quadro 6 – Segunda etapa do método de gravação de áudio

Após a passagem dos parâmetros, uma instância da classe `AVAudioRecorder` é criada com os parâmetros definidos anteriormente no `NSMutableDictionary` e com o caminho aonde o áudio deve ser gravado. Por fim, o método `recordForDuration` da classe `AVAudioRecorder` é invocado recebendo como parâmetro a duração máxima da gravação do áudio, este método tem como finalidade efetuar a gravação utilizando todos os parâmetros de configuração passados anteriormente. A última etapa do método pode ser visualizada abaixo entre as linhas 149 e 153 do Quadro 7.

```
149. recorder = [[AVAudioRecorder alloc] initWithURL:url settings:recordSettings
150.           error:&error];
151. [recorder setDelegate:self];
152. [recorder prepareToRecord];
153. [recorder setMeteringEnabled:YES];
154. [recorder recordForDuration:(NSTimeInterval)5];
```

Quadro 7 – Terceira etapa do método de gravação de áudio

### 3.3.4 iCloud

Todos os dados do aplicativo estão armazenados no *Core Data*. Desta forma, para realizar o backup de todas as informações criadas e mantidas pelo fonoaudiólogo, tutor e paciente é necessário que haja uma integração entre as informações mantidas no *Core Data* e o iCloud. Com esta integração, toda informação criada ou alterada na base de dados é

sincronizada com o iCloud, desde que haja conexão com a internet.

Para integrar o aplicativo com o iCloud é necessário realizar algumas configurações no portal do desenvolvedor da Apple e no projeto do aplicativo. Estas configurações podem ser visualizadas no apêndice C.

Após realizar as devidas configurações no projeto, é necessário habilitar o *Core Data* para persistir as informações da base de dados no iCloud. Para isso, o primeiro passo é verificar se o iCloud está habilitado no dispositivo através do método, `URLForUbiquityContainerIdentifier` da classe `NSFileManager`. Este método retorna uma URL do iCloud caso ele esteja ativo para o dispositivo. Se a URL retornada for válida, o próximo passo é definir os parâmetros necessários para a integração entre o iCloud e o *Core Data*, esses parâmetros são passados através de uma instância da classe `NSMutableDictionary`, uma estrutura de dados composta por chaves e valores. Cada chave do `NSMutableDictionary` contém como valor um parâmetro que será passado à instância da classe `NSPersistentStoreCoordinator`.

O primeiro parâmetro passado para o `NSMutableDictionary` é o valor booleano YES para a chave `NSMigratePersistentStoresAutomaticallyOption`. O valor passado para esta chave indica que o *Core Data* deve migrar automaticamente os dados da base de dados caso ocorra alguma mudança em sua estrutura. Isto é necessário para que quando ocorra alguma mudança estrutural nas entidades do *Core Data*, não seja necessário realizar uma migração manual dos dados que já estavam contidos na base.

O segundo parâmetro passado para o `NSMutableDictionary` é o App ID do aplicativo para a chave `NSPersistentStoreUbiquitousContentNameKey`. O valor passado para esta chave é utilizado para identificar qual o nome da base de dados do *Core Data* que deve ser compartilhada entre os diferentes dispositivos que estão utilizando o aplicativo. Desta forma, as transações feitas na base de dados do *Core Data* são sincronizadas de forma consistente caso mais de um dispositivo esteja utilizando o aplicativo ao mesmo tempo.

O terceiro e último parâmetro passado para o `NSMutableDictionary` é o caminho do arquivo do log de transações do *Core Data*. Esta caminho é passado para a chave `NSPersistentStoreUbiquitousContentURLKey`. O log de transações é gerado pelo *Core Data* e armazena cada mudança que foi comitada na base de dados. Desta forma, somente as mudanças no arquivo de log de transações são enviadas para a nuvem. Isto é necessário para a otimização da integração com o iCloud, já que somente as mudanças ocorridas serão persistidas no iCloud ao invés de toda a base de dados.

Após a passagem destes parâmetros, que ocorre entre as linhas 225 e 229, o método `addPersistentStoreWithType:configuration:URL:options:error` da instância da classe `NSPersistentStoreCoordinator` é invocado. Este método tem como função criar a base de dados do *Core Data* com os parâmetros passados caso ela ainda não tenha sido criada anteriormente.

O algoritmo de conexão ao iCloud descrito acima pode ser visualizado entre as linhas 202 e 229 do Quadro 8.

```

202     if ([iCloud] {
203         NSURL *iCloudLogsPath = [NSURL fileURLWithPath:[[[iCloud path]stringByAppendingPathComponent:iCloudLogsDirectoryName];
204
205         if ([fileManager fileExistsAtPath:[[[iCloud path]stringByAppendingPathComponent:iCloudDataDirectoryName]] == NO) {
206             NSError *fileSystemError;
207             [fileManager createDirectoryAtPath:[[[iCloud path] stringByAppendingPathComponent:iCloudDataDirectoryName]
208                                         withIntermediateDirectories:YES
209                                         attributes:nil
210                                         error:&fileSystemError];
211         }
212
213         NSString *iCloudData = [[[iCloud
214             path]stringByAppendingPathComponent:iCloudDataDirectoryName]stringByAppendingPathComponent:dataFileName];
215
216         NSMutableDictionary *options = [NSMutableDictionary dictionary];
217         [options setObject:[NSNumber numberWithBool:YES] forKey:NSMigratePersistentStoresAutomaticallyOption];
218         [options setObject:[NSNumber numberWithBool:YES] forKey:NSInferMappingModelAutomaticallyOption];
219         [options setObject:iCloudEnabledAppID           forKey:NPSPersistentStoreUbiquitousContentNameKey];
220         [options setObject:iCloudLogsPath               forKey:NPSPersistentStoreUbiquitousContentURLKey];
221
222         [psc lock];
223
224         [psc addPersistentStoreWithType:NSSQLiteStoreType
225                                     configuration:nil
226                                     URL:[NSURL fileURLWithPath:iCloudData]
227                                     options:options
228                                     error:nil];
229
230     [psc unlock];

```

Quadro 8 – Conexão com o iCloud

### 3.3.5 Operacionalidade da implementação

As principais funcionalidades do aplicativo estão relacionadas à criação de um plano de atividades pelo fonoaudiólogo e ao uso deste plano pelo paciente. Porém, para chegar na funcionalidade de uso do plano pelo paciente, é necessário que o fonoaudiólogo cumpra certas etapas, como criar um paciente, criar um tutor, criar uma categoria de símbolos, criar um símbolo para esta categoria e finalmente criar o plano de atividades que será enviado ao paciente para utilização. Esta seção apresenta de forma resumida a operacionalidade de cada uma destas etapas.

#### 3.3.5.1 Criação de pacientes

A tela inicial do aplicativo, na visão do fonoaudiólogo é apresentada com uma lista com todos os pacientes já criados e uma imagem que quando pressionada, apresenta a tela de

criação de pacientes. Esta tela contém os campos para serem preenchidos com os dados do paciente. Após preencher todos os campos, o fonoaudiólogo poderá salvar as informações pressionando o botão **OK**.

A Figura 25a apresenta a tela com a listagem de todos os pacientes e a Figura 25b apresenta a tela de criação de pacientes.

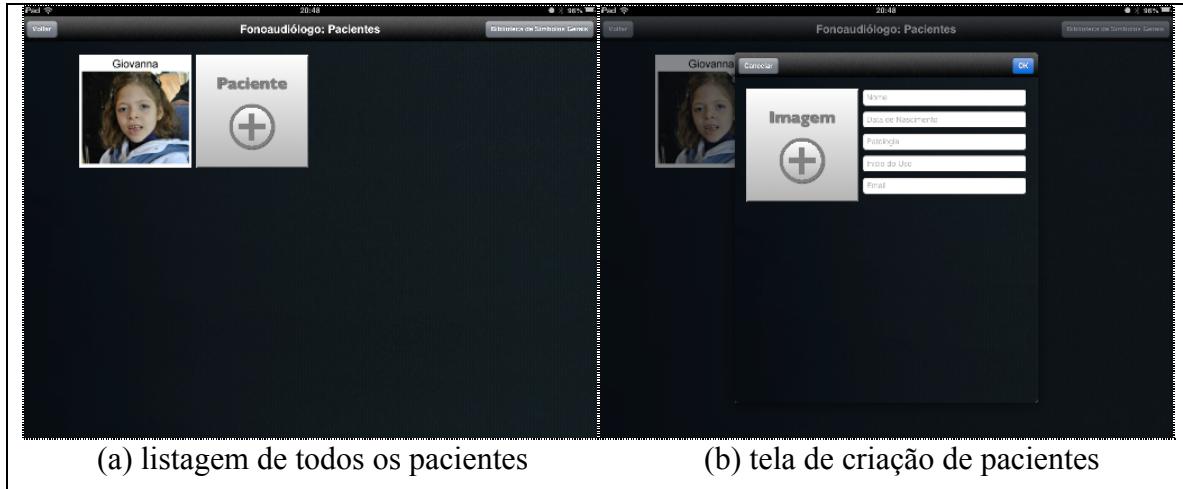


Figura 25 – Criação de pacientes

### 3.3.5.2 Criação de tutores

Após criar um paciente, o fonoaudiólogo tem a opção de visualizar os detalhes deste paciente em uma tela que é acessível a partir da tela que lista todos os pacientes já criados. Para criar um tutor, é necessário que o fonoaudiólogo entre na tela de detalhes do paciente e pressione o botão **Tutores**. Ao pressionar este botão, o aplicativo apresenta uma tela com uma lista com todos os tutores já criados e uma imagem que quando pressionada, apresenta a tela de criação de tutores. Esta tela contém os campos para serem preenchidos com os dados do tutor. Após preencher todos os campos, o fonoaudiólogo poderá salvar as informações pressionando o botão **OK**.

A Figura 26a apresenta a tela com a listagem de todos os tutores e a Figura 26b apresenta a tela de criação de tutores.

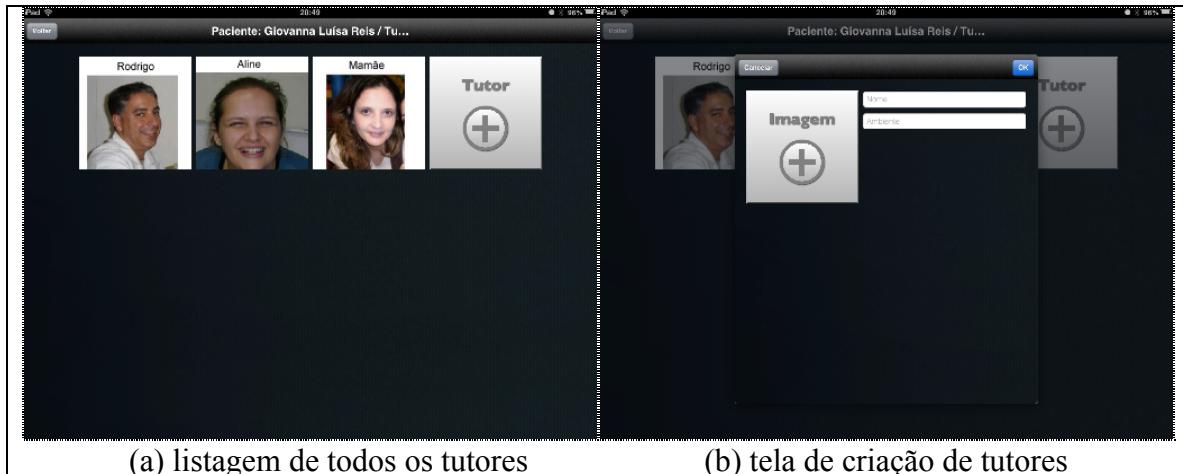


Figura 26 – Criação de tutores

### 3.3.5.3 Criação de categorias de símbolos

Para criar uma categoria de símbolos para o paciente, é necessário que o fonoaudiólogo entre na tela de biblioteca de símbolos e pressione a imagem que indica a criação de uma nova categoria. Ao pressionar esta imagem, o aplicativo apresenta a tela de criação de categorias. Esta tela contém um campo para informar o nome da categoria e uma lista com as 16 cores disponíveis para a categoria. Após preencher o nome e selecionar uma cor, o fonoaudiólogo poderá salvar as informações pressionando o botão **OK**.

A Figura 27a apresenta a tela de biblioteca de símbolos com a listagem de todas as categorias já criadas e a Figura 27b apresenta a tela de criação de categorias.

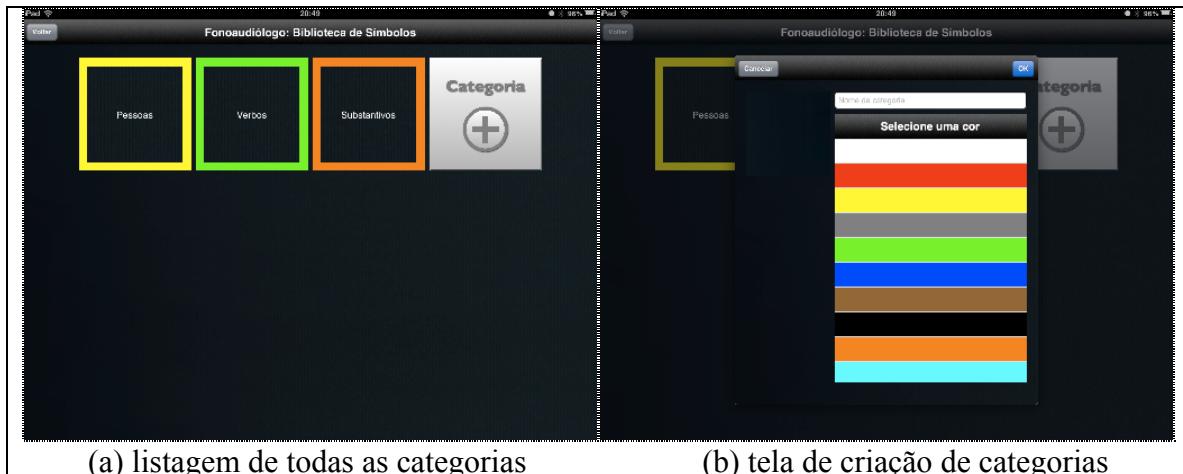


Figura 27 – Criação de categorias de símbolos

### 3.3.5.4 Criação de símbolos

Para criar um símbolo para o paciente o fonoaudiólogo tem duas opções. A primeira opção é criar um símbolo personalizado, que é exclusivo para determinado paciente. Para criar um símbolo personalizado é necessário que o fonoaudiólogo entre na tela de detalhes do paciente e no repositório de categorias pressione a categoria desejada. Após pressionar a categoria desejada, na qual o símbolo fará parte, o aplicativo apresenta ao fonoaudiólogo todos os símbolos já criados para esta categoria. Junto com todos os símbolos já criados para esta categoria, o aplicativo mostra também uma imagem com uma indicação para a criação de um novo símbolo. Ao pressionar esta imagem, o aplicativo apresenta ao usuário a tela de criação de símbolos. Esta tela contém os campos para serem preenchidos com os dados do símbolo. Após preencher todos os campos, o fonoaudiólogo poderá salvar as informações pressionando o botão **OK**.

Para criar um símbolo geral, ou seja, um símbolo que não é exclusivo para um determinado paciente, o fonoaudiólogo deve entrar na biblioteca de símbolos e pressionar a categoria desejada, na qual o símbolo fará parte. Após pressionar a categoria, o aplicativo irá apresentar ao fonoaudiólogo todos os símbolos já criados para esta categoria, junto com uma imagem com uma indicação para a criação de um novo símbolo. Ao pressionar esta imagem, o procedimento para a criação de um símbolo geral é o mesmo que o procedimento para a criação de um símbolo personalizado, a única diferença é que este símbolo geral irá aparecer para todos os pacientes.

A Figura 28a apresenta a tela com a listagem de todos os símbolos e a Figura 28b apresenta a tela de criação de símbolos.

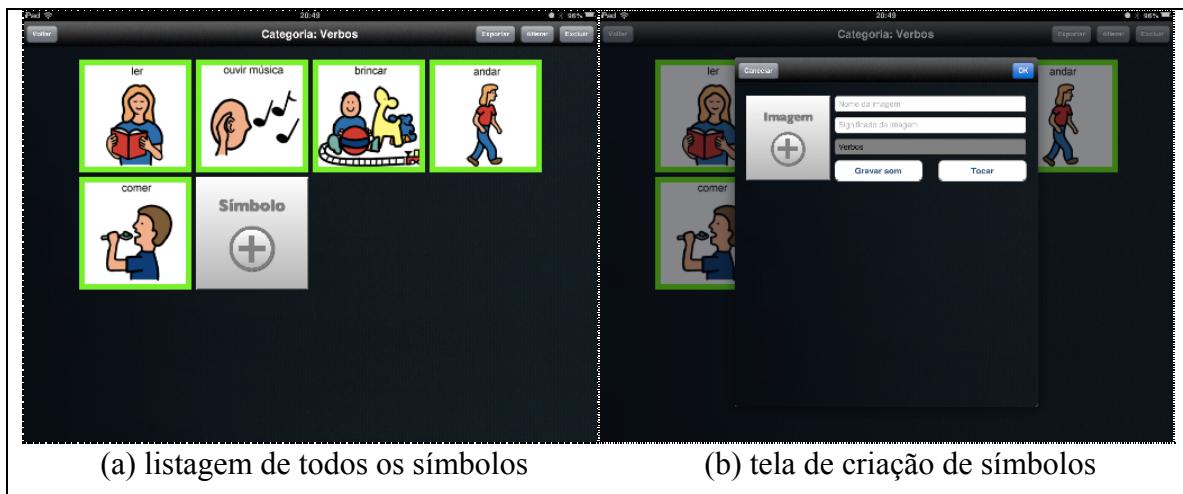


Figura 28 – Criação de símbolos

### 3.3.5.5 Criação de planos

Para criar um plano para o paciente, é necessário que o fonoaudiólogo entre na tela de detalhes do paciente e pressione o botão **Tutores**. Ao pressionar este botão, o aplicativo apresenta uma lista com todos os tutores já criados. O fonoaudiólogo, deve então selecionar um tutor e o aplicativo apresentará uma outra tela contendo todos os planos já associados para o tutor selecionado e uma imagem com indicação para a criação de um novo plano. Ao pressionar esta imagem, o aplicativo apresenta ao usuário a tela de criação de planos. Esta tela contém os campos para serem preenchidos com os dados do plano. Um plano pode ser de dois tipos, Símbolos Grandes ou Prancha. Quando o fonoaudiólogo seleciona qualquer um dos dois tipos, o sistema apresenta uma outra tela que dependendo do tipo de plano escolhido, apresentará uma ou nove imagens a serem preenchidas com os símbolos já criados para este paciente. Após preencher as imagens com os símbolos para o plano, o fonoaudiólogo é direcionado novamente a tela de criação de planos para finalizar a criação do mesmo. Assim que o fonoaudiólogo preenche todos os campos restantes, ele poderá enviar o plano criado por e-mail para o paciente.

A Figura 29a apresenta a tela com a listagem de todos os planos, a Figura 29b apresenta a tela de criação de planos, a Figura 30 apresenta a tela para selecionar os símbolos que farão parte do plano e a Figura 31 apresenta a tela de envio de e-mails com o plano anexado.



Figura 29 – Criação de planos



Figura 30 – Tela para selecionar os símbolos do plano

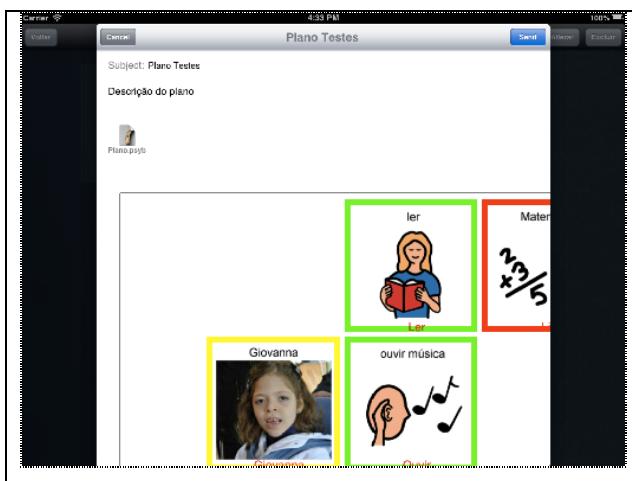


Figura 31 – Tela de envio de e-mails com o plano anexado

### 3.3.5.6 Importação e utilização de um plano

Dependendo das condições motoras do paciente, para utilizar um plano talvez seja necessário o auxílio de um tutor. O tutor tem a função de importar o plano criado pelo fonoaudiólogo e fazer a escolha do plano correto caso haja mais de um plano já importado no dispositivo do paciente.

Para importar um plano para o dispositivo do paciente, é necessário que o tutor entre no aplicativo de e-mail que já vem instalado com o dispositivo iOS e pressione o arquivo .psyb anexado no e-mail do enviado pelo fonoaudiólogo. Após manter pressionando este arquivo, o iOS fornece opções para abrir este arquivo com os aplicativos instalados no dispositivo que suportam a extensão .psyb. Desta forma, quando o tutor escolher a opção para abrir o arquivo anexado com o aplicativo Tagarela, o aplicativo é inicializado e

automaticamente importa os dados contidos no arquivo para a base de dados do aplicativo.

Feita a importação do plano, é necessário que o tutor que está auxiliando o paciente escolha de qual tutor deseja visualizar os planos, já que um paciente pode possuir diversos tutores associados a ele. A escolha do tutor é feita a partir da tela inicial do aplicativo, aonde é apresentado uma lista com todos os tutores daquele paciente. Após escolher um tutor, o aplicativo apresenta ao tutor que está auxiliando o paciente uma listagem de todos os planos que estão associados ao tutor selecionado. Quando o tutor seleciona um plano desta lista, o aplicativo apresenta a ele todos os símbolos que foram escolhidos pelo fonoaudiólogo para fazerem parte daquele plano. Por fim, o paciente agora é livre para utilizar o plano interagindo com os símbolos que lá estão.

A Figura 32 apresenta o e-mail enviado pelo fonoaudiólogo com as opções para abrir o arquivo .psyb, a Figura 33a apresenta a tela com a lista de tutores importados, a Figura 33b apresenta a tela com os planos associados ao tutor escolhido e a Figura 34 apresenta a tela com os símbolos que compõem um plano escolhido.



Figura 32 – E-mail enviado pelo fonoaudiólogo com as opções para importar o plano

(a) listagem dos tutores importados	(b) listagem dos planos associados ao tutor

Figura 33 – Lista dos tutores importados e lista dos planos associados ao tutor escolhido



Figura 34 – Símbolos que compõem o plano escolhido

### 3.4 RESULTADOS E DISCUSSÃO

Este trabalho apresenta a utilização dos recursos multimídia da plataforma iOS para a implementação de um aplicativo de comunicação alternativa.

O objetivo inicial deste trabalho era disponibilizar um aplicativo de comunicação alternativa, desenvolvido para a plataforma iOS, capaz de atender as necessidades específicas de cada paciente através da troca de mensagens entre os usuários envolvidos. Esta troca de mensagens, inicialmente seria feita através do iCloud, porém, após uma pesquisa, identificou-se que o iCloud apresentava restrições quanto a troca de mensagens entre diferentes usuários.

Optou-se então, em realizar esta troca de mensagens através de e-mails que são enviados por dentro do aplicativo. A utilização do iCloud ficou restrita ao *backup* das informações geradas pelo aplicativo, como por exemplo, os símbolos criados pelo fonoaudiólogo.

Para um melhor entendimento do resultado final da implementação do aplicativo, é apresentado o cenário do sistema na Figura 35.



Figura 35 – Cenário do sistema

Como ponto inicial deste trabalho foi utilizado o roteiro de implementação de um recurso de tecnologia assistiva elaborado pelo *Center on Disabilities* da *California State University* de Northridge (SCHIRMER et al., 2007).

Esta seção foi separada em três subseções. A seção 3.4.1 descreve como o roteiro de implementação de um recurso de tecnologia assistiva foi utilizado para elaborar os requisitos do aplicativo. A seção 3.4.2 descreve os resultados obtidos através da análise de desempenho do aplicativo. A seção 3.4.3 compara o aplicativo desenvolvido com os trabalhos correlatos.

### 3.4.1 Análise do roteiro de implementação de um recurso de tecnologia assistiva para elaboração dos requisitos do aplicativo

Em conjunto com o fonoaudiólogo Rodrigo França, especialista na área de comunicação alternativa e com Dalton Solano dos Reis, tutor de um paciente, o roteiro de implementação de um recurso de tecnologia assistiva foi utilizado para ajudar a identificar os principais requisitos do aplicativo.

De acordo com o roteiro, o primeiro passo é identificar as necessidades do futuro usuário do aplicativo. Estas necessidades foram identificadas após uma entrevista com o especialista Rodrigo França e o tutor Dalton Solano dos Reis. Nesta entrevista foi definido os principais requisitos do aplicativo e as metas que deveriam ser atingidas através da implementação destes requisitos.

Os requisitos foram definidos através da análise das necessidades do fonoaudiólogo em conjunto com a avaliação do que os trabalhos correlatos já ofereciam e o funcionamento das

ferramentas de comunicação alternativa manuais previamente utilizadas pelo fonoaudiólogo e pelo tutor.

A conclusão obtida através da análise das ferramentas de comunicação alternativa manuais é que elas, apesar de serem muito utilizadas no mundo todo, sofrem com a falta de recursos que somente a tecnologia pode proporcionar.

O principal problema identificado no processo manual, é que a medida que o paciente começa a utilizar mais planos, o processo de gerenciar estes planos e os símbolos contidos neles se torna cada vez mais complexo. Este processo pode ser visualizado no vídeo (REIS, 2012) que apresenta alguns dos planos e símbolos utilizados pela tutora Ana Lucia Anacleto Reis, mãe da paciente Giovanna.

Outro problema identificado no processo manual, é que constantemente havia a perda do histórico do paciente com a troca de um tutor, isto é prejudicial pois força a relembrar os planos e atividades já realizados pelo tutor anterior. A troca de tutores de um paciente não é algo incomum, pois pelo lado pedagógico, é algo recomendado periodicamente para evitar laços afetivos entre o paciente e o tutor (FRANÇA, 2012a). No apêndice D pode-se visualizar um plano de atividades realizado pelo processo manual.

A conclusão obtida através da análise dos trabalhos correlatos é que grande parte deles, apesar de resolverem os problemas das ferramentas manuais, não forneciam uma maneira eficiente de realizar a comunicação entre o fonoaudiólogo e os tutores do paciente. Esta comunicação é importante no processo da comunicação alternativa pois permite que haja uma troca de experiências a respeito da produtividade do paciente em relação aos planos criados pelo fonoaudiólogo (FRANÇA, 2012a).

Após a implementação do aplicativo baseado nos requisitos definidos, uma nova entrevista foi realizada. Esta entrevista foi realizada para demonstrar o uso do aplicativo e para definir a primeira fase de testes do aplicativo em um ambiente real, aonde o fonoaudiólogo e o paciente estejam envolvidos na utilização do aplicativo (FRANÇA, 2012b). Nesta entrevista foi definido que a primeira fase de testes seria feita através da utilização pelo paciente das pranchas de comunicação elaboradas pelo fonoaudiólogo dentro do aplicativo.

O vídeo (FURB TV, 2012) demonstra os testes feitos pelo paciente em uma das sessões com o fonoaudiólogo. Neste vídeo é possível observar o paciente interagindo com os símbolos de um plano de atividades elaborados pelo fonoaudiólogo e pelo tutor. Analisando o vídeo, pode-se notar que em comparação com a versão manual, o aplicativo desenvolvido pareceu despertar mais interesse do paciente. Este interesse, provavelmente ocorreu devido a possibilidade da interação com o toque e o fato do paciente ter o retorno do áudio que é

reproduzido a cada símbolo tocado. Outro aspecto observado, é que o aplicativo facilita a montagem dos planos de atividade pelo fonoaudiólogo. No apêndice E pode ser visualizado um relatório com o histórico de observações escritas pelo fonoaudiólogo enquanto fazia suas sessões com o paciente.

### 3.4.2 Desempenho do *Core Data*

Entre as opções de análise de desempenho disponibilizadas pelo Instruments, foi escolhida realizar a análise da performance das duas principais operações do *Core Data*. A análise de alocação de memória e processamento foi considerada para este trabalho, porém durante os testes, observou-se que as principais operações do aplicativo não faziam uso intensivo de memória e processamento, motivo que justificaria uma análise mais detalhada.

As operações *save* e *fetch* do *Core Data* são utilizadas para inserir um registro e recuperar um registro da base dados, respectivamente. Para avaliar a performance foram analisadas as seguintes situações:

- a) inserção sem relacionamentos entre tabelas;
- b) inserção com relacionamento entre tabelas;
- c) busca sem relacionamentos entre tabelas;
- d) busca com relacionamentos entre tabelas.

Os testes foram realizados com o iOS Simulator em um MacBook (processador Intel Core 2 Duo 2.26 GHz, 2GB DDR3) através da ferramenta Instruments.

Na Figura 36 pode-se visualizar o tempo gasto na operação de *save*.

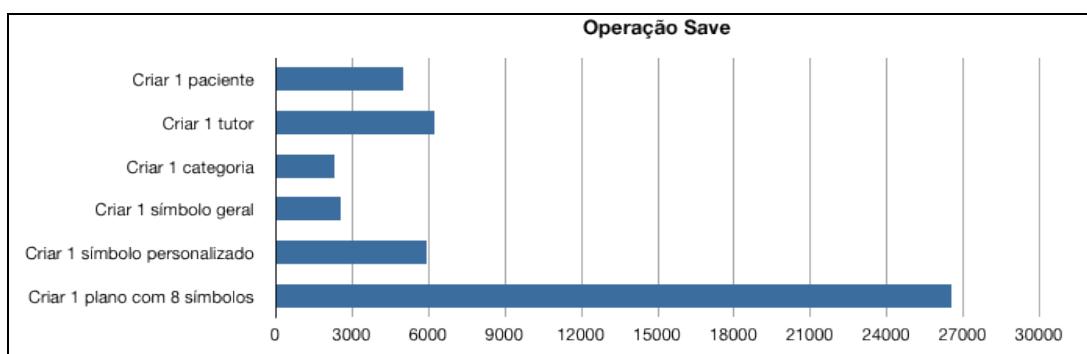


Figura 36 – Tempo de inserção dos dados em microsegundos

Com a análise da Figura 36, é possível concluir que o tempo de inserção em uma entidade em que há um relacionamento é ligeiramente maior que o tempo de inserção em uma entidade em que não há. Isto pode ser observado analisando o tempo de inserção de um

símbolo geral e de um símbolo personalizado. A inserção de um símbolo personalizado leva mais que o dobro de tempo que a inserção de um símbolo geral. Isto acontece pois na inserção de um símbolo geral, não ocorre nenhuma associação entre entidades, ou seja, o símbolo não é diretamente associado a um paciente em específico. Por outro lado, na inserção de um símbolo personalizado, além de salvar os atributos próprios do símbolo (imagem, nome, áudio, entre outros) é necessário que haja a associação deste símbolo a um paciente em específico.

Na Figura 37 pode-se visualizar o tempo gasto na operação de *fetch*.

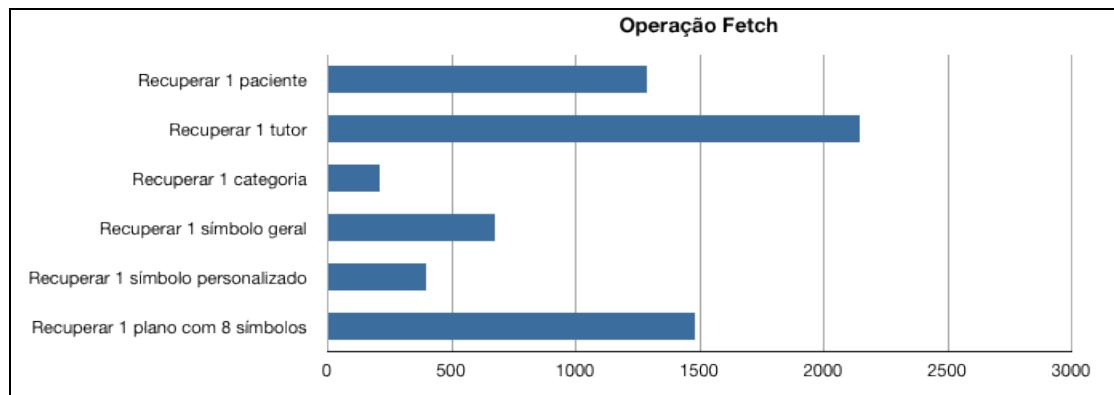


Figura 37 – Tempo de recuperação dos dados em microssegundos

Com a análise da Figura 37, é possível concluir que quanto mais relacionamentos uma entidade possui, maior é o seu tempo de recuperação. Isto pode ser observado analisando o tempo de recuperação de um tutor, que leva mais de 600 microssegundos para ser recuperado do que a segunda entidade que mais demora. No exemplo da Figura 36, um tutor possui relacionamento com um plano e este plano possui relacionamento com oito símbolos.

Considerando as análises feitas nesta seção, pode-se concluir que a quantidade de relacionamentos de uma entidade influencia de forma considerável a inserção e a recuperação de dados utilizando o *framework Core Data*. No entanto, no aplicativo desenvolvido não há grande preocupação com este aspecto, pois foi observado que as operações rotineiras acontecem em questão de microssegundos, tempo imperceptível para o usuário.

### 3.4.3 Comparativo entre o trabalho desenvolvido e os trabalhos correlatos

Nesta seção são apresentadas e discutidas as principais características deste trabalho em comparação com os trabalhos correlatos, as quais estão ilustradas no Quadro 9. O Quadro 9 é uma reformulação do quadro apresentado na seção de trabalhos correlatos. Este quadro

inclui características exclusivas do aplicativo desenvolvido para serem comparadas e discutidas com as funcionalidades dos trabalhos previamente estudados.

<b>Características / Trabalhos Correlatos</b>	<b>Tagarela</b>	<b>Projeto AMPLISOFT</b>	<b>Alexicom AAC</b>
criação de símbolos personalizados	x	x	x
associação de áudio aos símbolos	x	x	x
troca de mensagens entre as pessoas envolvidas	x		
criação de plano de atividades	x		
possibilidade de impressão das pranchas	x	x	
histórico de observações do paciente	x		
histórico de uso dos símbolos	x		
criação de símbolos via plataforma web			x

Quadro 9 – Comparação do trabalho desenvolvido com os trabalhos correlatos

O Quadro 9 ilustra as principais características do trabalho desenvolvido em comparação com os trabalhos correlatos, tendo como principais diferenciais a criação de plano de atividades e a troca de mensagens entre as pessoas envolvidas no processo de comunicação.

Duas funcionalidades exclusivas do Tagarela também podem ser observadas no Quadro 9, as funcionalidades de histórico de observações do paciente e histórico de uso dos símbolos são importantes, pois de acordo com França (2012a) facilitam o acompanhamento da evolução clínica do paciente durante o período de tratamento.

Em comparação com os trabalhos correlatos, neste trabalho não foi implementado uma plataforma web para a criação e sincronização de novos símbolos com o dispositivo móvel do paciente. Esta plataforma se mostra importante pois permite que o fonoaudiólogo crie e envie símbolos para os pacientes mesmo sem ter um iPad disponível. Atualmente o aplicativo desenvolvido permite a sincronização de novos símbolos, porém esta sincronização é feita através de um arquivo anexado em um e-mail. Esta sincronização, ao contrário do que é feito no trabalho correlato, não permite que os usuários controlem quais informações devem ser adicionadas, atualizadas ou removidas, portanto não é muito flexível neste aspecto.

## 4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento do Tagarela, um aplicativo para comunicação alternativa para iOS. O aplicativo desenvolvido fornece uma plataforma para que as pessoas envolvidas no processo de comunicação do paciente interajam entre si para que haja uma evolução na capacidade de comunicação do paciente, através de planos de atividades elaborados pelo fonoaudiólogo em conjunto com o tutor do paciente.

Através dos resultados obtidos com o uso do aplicativo em sessões de terapia, provou-se que o aplicativo desenvolvido pode trazer benefícios reais às pessoas envolvidas no processo de comunicação do paciente. Estes benefícios, a longo prazo, podem significar melhorias significativas no processo de comunicação do paciente, possibilitando que ele amplie sua participação na sociedade.

A principal limitação do aplicativo desenvolvido é a forma como as mensagens entre as pessoas envolvidas (fonoaudiólogo, tutor e paciente) são trocadas. A troca de mensagens via e-mail, apesar de eficiente, não é muito prática, pois envolve que o usuário tenha que sair do aplicativo para fazer a importação do conteúdo recebido e voltar ao aplicativo novamente para visualizar este conteúdo. Outra grande limitação do aplicativo, é que, atualmente não é possível controlar o que deve ser sincronizado no conteúdo recebido nesta troca de mensagens.

Durante o desenvolvimento deste trabalho, foi possível observar que as ferramentas de desenvolvimento fornecidas pela Apple, como o iOS Simulator e o Instruments, foram importantíssimas para a conclusão deste trabalho. O iOS Simulator foi essencial na fase inicial de desenvolvimento do aplicativo, pois permitiu que as funcionalidades desenvolvidas fossem rapidamente testadas antes de serem executadas no iPad, que não estava o tempo todo acessível. Já a ferramenta Instruments foi de grande importância na fase final de desenvolvimento. Com esta ferramenta foi possível analisar o desempenho do aplicativo e identificar eventuais pontos de melhoria.

Por fim, espera-se que o trabalho desenvolvido esteja em constante evolução, com novos recursos sendo adicionados de acordo com a demanda dos usuários, possibilitando atender os diversos tipos de necessidades existentes.

#### 4.1 EXTENSÕES

Sugerem-se as seguintes extensões para trabalhos futuros:

- a) implementar um servidor para realizar a troca de mensagens entre as pessoas envolvidas;
- b) permitir que o paciente faça a movimentação dos símbolos nas pranchas;
- c) permitir que o fonoaudiólogo crie imagens para os símbolos dentro do aplicativo, através da junção de duas ou mais imagens já existentes;
- d) permitir que o fonoaudiólogo crie pranchas dinâmicas, com número variável de símbolos;
- e) permitir que o fonoaudiólogo crie os símbolos de um plano em uma plataforma web;
- f) analisar a possibilidade de implementar outras formas de interação com os símbolos;
- g) analisar a possibilidade de incluir vídeos;
- h) analisar o formato de imagem mais adequado para a criação dos símbolos;
- i) adaptar o aplicativo para os dispositivos iPhone e iPod Touch;
- j) adaptar todas as imagens do aplicativo para o iPad Retina;
- k) efetuar testes com outros pacientes e fonoaudiólogos;
- l) disponibilizar o aplicativo na App Store.

## REFERÊNCIAS BIBLIOGRÁFICAS

ALEXICOM AAC. **How it works.** [S.l.], 2011. Disponível em:  
 <<http://www.alexicomaac.com/dnn/Default.aspx?tabid=67>>. Acesso em: 29 mar. 2012.

APPLE INC. **iOS technology overview.** [S.l.], 2011. Disponível em:  
 <<https://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>>. Acesso em: 27 mar. 2012.

\_\_\_\_\_. **Alexicom AAC.** [S.l.], 2012a. Disponível em:  
 <<http://itunes.apple.com/us/app/alexicom-aac/id395122088?mt=8>>. Acesso em: 14 abr. 2012.

\_\_\_\_\_. **iCloud for developers.** [S.l.], 2012b. Disponível em:  
 <<https://developer.apple.com/icloud/>>. Acesso em: 17 abr. 2012.

BLISSYMBOLICS. **Blissymbolics communication international.** [S.l.], 2012. Disponível em: <<http://www.blissymbolics.org>>. Acesso em: 27 out. 2012.

BRAATZ, Jennifer P. **Paulo Freire na fisioterapia? Ben-10 com voz ativa!** 2012. 118 f.  
 Dissertação (Mestrado em Educação) - Programa de Pós-graduação em Educação, Centro de Ciências da Educação, Fundação Universidade Regional de Blumenau, Blumenau. Disponível em: <[http://www.bc.furb.br/docs/DS/2012/351455\\_1\\_1.PDF](http://www.bc.furb.br/docs/DS/2012/351455_1_1.PDF)>. Acesso em: 19 dez. 2012.

BRASIL. **Constituição.** [S.l.], 1988. Disponível em:  
 <[http://www.planalto.gov.br/ccivil\\_03/Constituicao/Constituicao.htm](http://www.planalto.gov.br/ccivil_03/Constituicao/Constituicao.htm)>. Acesso em: 11 abr. 2012.

CRESER COMUNICANDO. **Sistema PIC (Pictogram Ideogram Communication).** [S.l.], 2011. Disponível em: <<http://crescercomunicando.blogspot.com.br/2011/05/sistema-pic-pictogram-ideogram.html>>. Acesso em: 27 out. 2012.

FABENI, Alan F. C. **Documentação Tagarela.** Blumenau, 2012. Disponível em:  
 <<https://www.dropbox.com/sh/l1ix3mo2el1gdtf/ksI4cNQHhs>>. Acesso em: 19 dez. 2012.

FRANÇA, Rodrigo. **Entrevista sobre elicitação de requisitos.** Entrevistadores: Alan Filipe Cardozo Fabeni e Dalton Solano dos Reis. Blumenau. 2012a. Entrevista feita através de conversação – não publicada.

\_\_\_\_\_. **Entrevista para demonstração do aplicativo e definição da fase de testes.** Entrevistadores: Alan Filipe Cardozo Fabeni e Dalton Solano dos Reis. Blumenau. 2012b. Entrevista feita através de conversação – não publicada.

FURB TV. **Demonstração da utilização do aplicativo pelo paciente e fonoaudiólogo.** Disponível em: <<http://www.youtube.com/watch?v=DrxAfNJZGy8>>. Acesso em: 13 nov. 2012.

NOHAMA, Percy et al. **AMPLISOFT**. Curitiba, 2010. Disponível em: <<http://www.ler.pucpr.br/amplisoft/>>. Acesso em: 24 mar. 2012.

PELOSI, Miryam. **Tecnologia assistiva**. Rio de Janeiro, 2011. Disponível em: <<http://www.comunicacaoalternativa.com.br/tecnologia-assistiva>>. Acesso em: 24 mar. 2012.

REIS, Dalton S. **Demonstração do processo manual de comunicação alternativa**. Disponível em: <[http://www.youtube.com/watch?v=jo3QL\\_Z5AAQ](http://www.youtube.com/watch?v=jo3QL_Z5AAQ)>. Acesso em: 11 nov. 2012.

SARTORETTO, Mara L.; BERSCH, Rita. **O que é tecnologia assistiva?** Porto Alegre, 2007. Disponível em: <<http://www.assistiva.com.br/tassistiva.html>>. Acesso em: 24 mar. 2012.

SCHIRMER, Carolina R. et al. **Atendimento educacional especializado**. [S.I.], 2007. Disponível em: <[http://portal.mec.gov.br/seesp/arquivos/pdf/aee\\_df.pdf](http://portal.mec.gov.br/seesp/arquivos/pdf/aee_df.pdf)>. Acesso em: 13 abr. 2012.

ZDZIARSKI, Jonathan. **iPhone SDK: application development**. Sebastopol: O'Reilly, 2009.

## **APÊNDICE A – Descrição detalhada do casos de uso**

A descrição detalhada do UC01 pode ser visualizada no Quadro 10. A descrição detalhada do UC02 pode ser visualizada no Quadro 11. A descrição detalhada do UC03 pode ser visualizada no Quadro 12. A descrição detalhada do UC04 pode ser visualizada no Quadro 13. A descrição detalhada do UC05 pode ser visualizada no Quadro 14. A descrição detalhada do UC06 pode ser visualizada no Quadro 15. A descrição detalhada do UC07 pode ser visualizada no Quadro 16. A descrição detalhada do UC08 pode ser visualizada no Quadro 17. A descrição detalhada do UC09 pode ser visualizada no Quadro 18. A descrição detalhada do UC10 pode ser visualizada no Quadro 19. A descrição detalhada do UC11 pode ser visualizada no Quadro 20. A descrição detalhada do UC12 pode ser visualizada no Quadro 21. A descrição detalhada do UC13 pode ser visualizada no Quadro 22. A descrição detalhada do UC14 pode ser visualizada no Quadro 23. A descrição detalhada do UC15 pode ser visualizada no Quadro 24. A descrição detalhada do UC16 pode ser visualizada no Quadro 25. A descrição detalhada do UC17 pode ser visualizada no Quadro 26. A descrição detalhada do UC18 pode ser visualizada no Quadro 27. A descrição detalhada do UC19 pode ser visualizada no Quadro 28.

UC01 – Criar paciente	
<b>Requisitos atendidos</b>	RF01
<b>Pré-condições</b>	Nenhuma.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo pressiona a imagem padrão que indica a criação de um novo paciente na tela que lista os pacientes.</p> <p>2. O aplicativo apresenta uma tela com campos para informar os dados do paciente, como: nome, data de nascimento, patologia, e-mail e a data em que o paciente começou a usar o aplicativo. O aplicativo também apresenta na mesma tela uma imagem padrão, que quando o fonoaudiólogo pressiona-a é apresentado a ele duas opções para escolher uma foto para o paciente.</p> <p>3. O fonoaudiólogo preenche os campos e pressiona a imagem padrão para escolher uma foto para o paciente.</p> <p>4. O aplicativo apresenta duas opções para o fonoaudiólogo: escolher uma foto da galeria de fotos ou tirar uma nova foto.</p> <p>5. O fonoaudiólogo seleciona a opção para escolher uma foto da galeria de fotos do iPad.</p> <p>6. O aplicativo apresenta em um <i>popover</i> todas as imagens contidas na galeria de fotos do iPad.</p> <p>7. O fonoaudiólogo seleciona uma foto da galeria.</p> <p>8. O aplicativo substitui a imagem padrão pela foto que foi selecionada pelo fonoaudiólogo.</p> <p>9. O fonoaudiólogo pressiona o botão OK para salvar as informações.</p> <p>10. O aplicativo salva as informações e apresenta novamente a tela com a listagem de todos os pacientes já criados.</p>
<b>Cenário alternativo 01</b>	<p>No passo 5, caso o fonoaudiólogo escolha a opção de tirar uma nova foto:</p> <p>5.1. O aplicativo apresenta a tela padrão de captura de fotos do iOS.</p> <p>5.2. O fonoaudiólogo tira uma foto.</p> <p>5.3. O aplicativo substitui a imagem padrão pela foto que foi capturada pelo fonoaudiólogo.</p> <p>5.4. Retorna ao cenário principal.</p>
<b>Exceção 01</b>	<p>No passo 10, caso ocorra um erro ao salvar as informações do paciente:</p> <p>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações do paciente.</p> <p>2. Retorna ao passo 2 do cenário principal.</p>
<b>Pós-condições</b>	Informações do paciente salvas ou nenhuma informação salva.

Quadro 10 – Caso de uso Criar paciente

UC02 – Criar tutor	
Requisitos atendidos	RF02
<b>Pré-condições</b>	Para o fonoaudiólogo criar um tutor é necessário que o fonoaudiólogo já tenha criado um paciente.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo pressiona o botão <i>Tutores</i> na tela de detalhes do paciente.</p> <p>2. O aplicativo apresenta uma tela com todos os tutores já criados para o paciente.</p> <p>3. O fonoaudiólogo pressiona a imagem padrão que indica a criação de um novo tutor na tela que lista os tutores.</p> <p>4. O aplicativo apresenta uma tela com campos para informar os dados do tutor, como: nome, ambiente e o e-mail do tutor. O aplicativo também apresenta na mesma tela uma imagem padrão, que quando o fonoaudiólogo do aplicativo pressiona-a é apresentado a ele duas opções para escolher a foto do tutor.</p> <p>5. O fonoaudiólogo preenche os campos e pressiona a imagem padrão para escolher uma foto para o tutor.</p> <p>6. O aplicativo apresenta duas opções para o fonoaudiólogo: escolher uma foto da galeria de fotos ou tirar uma nova foto.</p> <p>7. O fonoaudiólogo seleciona a opção para escolher uma foto da galeria de fotos do iPad.</p> <p>8. O aplicativo apresenta em um <i>popover</i> as imagens contidas na galeria de fotos do iPad.</p> <p>9. O fonoaudiólogo seleciona uma foto da galeria.</p> <p>10. O aplicativo substitui a imagem padrão pela foto que foi selecionada pelo fonoaudiólogo.</p> <p>11. O fonoaudiólogo pressiona o botão <i>OK</i> para salvar as informações.</p> <p>12. O aplicativo salva as informações e apresenta novamente a tela com a listagem de todos os tutores já criados.</p>
<b>Cenário alternativo 01</b>	<p>No passo 7, caso o fonoaudiólogo escolha a opção de tirar uma nova foto:</p> <p>7.1. O aplicativo apresenta a tela padrão de captura de fotos do iOS.</p> <p>7.2. O fonoaudiólogo tira uma foto.</p> <p>7.3. O aplicativo substitui a imagem padrão pela foto que foi capturada pelo fonoaudiólogo.</p> <p>7.4. Retorna ao cenário principal.</p>
<b>Exceção 01</b>	<p>No passo 12, caso ocorra um erro ao salvar as informações do tutor:</p> <p>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações do tutor.</p> <p>2. Retorna ao passo 2 do cenário principal.</p>
<b>Pós-condições</b>	Informações do tutor salvas ou nenhuma informação salva.

Quadro 11 – Caso de uso Criar tutor

<b>UC03 – Criar categorias de símbolos</b>	
<b>Requisitos atendidos</b>	RF03
<b>Pré-condições</b>	Nenhuma.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo pressiona a imagem padrão que indica a criação de uma nova categoria na tela de biblioteca de símbolos.</p> <p>2. O aplicativo apresenta a tela de criação de categorias com um campo para informar o nome da categoria e uma lista de cores para que o fonoaudiólogo defina a cor da categoria.</p> <p>3. O fonoaudiólogo preenche o campo com o nome da categoria.</p> <p>4. O fonoaudiólogo seleciona uma cor da lista de cores.</p> <p>5. O aplicativo cria uma imagem preenchida com a cor selecionada pelo fonoaudiólogo e mostra-a ao fonoaudiólogo.</p> <p>6. O fonoaudiólogo pressiona o botão OK para salvar as informações.</p> <p>7. O aplicativo salva as informações e apresenta a tela da biblioteca de símbolos novamente.</p>
<b>Exceção 01</b>	<p>No passo 7, caso ocorra um erro ao salvar as informações:</p> <p>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações da categoria.</p> <p>2. Retorna ao passo 2 do cenário principal.</p>
<b>Pós-condições</b>	Informações da categoria salvas ou nenhuma informação salva.

Quadro 12 – Caso de uso Criar categoria de símbolos

UC04 – Criar símbolos	
Requisitos atendidos	RF04
<b>Pré-condições</b>	Para o fonoaudiólogo criar um símbolo é necessário que o fonoaudiólogo já tenha criado uma categoria.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo seleciona uma categoria na listagem de categorias na tela de detalhes do paciente.</p> <p>2. O aplicativo mostra todos os símbolos já criados para a categoria selecionada.</p> <p>3. O fonoaudiólogo pressiona a imagem padrão que indica a criação de um novo símbolo.</p> <p>4. O aplicativo apresenta uma tela com campos para informar o nome do símbolo e o significado do símbolo. O aplicativo também apresenta 2 botões. O primeiro botão é utilizado para o fonoaudiólogo gravar uma mensagem de áudio representando o significado do símbolo. O segundo botão é utilizado para o fonoaudiólogo reproduzir a mensagem que foi gravada, afim de certificar-se de que ela foi gravada corretamente. Na mesma tela o aplicativo também apresenta uma imagem padrão, que quando o fonoaudiólogo pressiona-a é apresentado a ele três opções para escolher uma foto para ser associada ao símbolo.</p> <p>5. O fonoaudiólogo preenche os campos e pressiona botão para gravar uma mensagem de áudio.</p> <p>6. O aplicativo começa uma contagem regressiva de 5 segundos representando o tempo limite de duração para a mensagem de áudio.</p> <p>7. Após o tempo esgotar-se o aplicativo mostra uma mensagem ao fonoaudiólogo informando que a mensagem foi gravada com sucesso.</p> <p>8. Se o fonoaudiólogo pressionar o botão para reproduzir a mensagem o aplicativo reproduz a mensagem gravada.</p> <p>9. O fonoaudiólogo pressiona a imagem padrão para escolher uma imagem para associar ao símbolo.</p> <p>10. O aplicativo apresenta três opções para o fonoaudiólogo: escolher uma imagem da galeria de fotos, tirar uma nova foto ou escolher uma imagem da biblioteca de imagens do aplicativo.</p> <p>11. O fonoaudiólogo seleciona a opção para escolher uma foto da galeria de fotos do iPad.</p> <p>12. O aplicativo apresenta em um <i>popover</i> as imagens contidas na galeria de fotos do iPad.</p> <p>13. O fonoaudiólogo seleciona uma foto da galeria.</p> <p>14. O aplicativo substitui a imagem padrão pela imagem que foi selecionada pelo fonoaudiólogo.</p> <p>15. O fonoaudiólogo pressiona o botão OK para salvar as informações.</p> <p>16. O aplicativo salva as informações e apresenta a tela com os detalhes do paciente.</p>
<b>Cenário alternativo 01</b>	<p>No passo 1, caso o fonoaudiólogo escolha criar um símbolo geral:</p> <p>1.1. O fonoaudiólogo entra na tela de biblioteca de símbolos.</p> <p>1.2. O fonoaudiólogo seleciona uma categoria para qual o símbolo deve pertencer.</p> <p>1.3. O aplicativo mostra uma listagem de todos os símbolos pertencentes a categoria selecionada.</p> <p>1.4. O fonoaudiólogo pressiona a imagem padrão que indica a criação de um novo símbolo.</p> <p>1.5. Retorna ao cenário principal.</p>
<b>Cenário alternativo 02</b>	<p>No passo 11, caso o fonoaudiólogo selecione a opção de tirar uma nova foto:</p> <p>11.1. O aplicativo apresenta a tela padrão de captura de fotos do iOS.</p> <p>11.2. O fonoaudiólogo tira uma foto.</p> <p>11.3. O aplicativo substitui a imagem padrão pela imagem que foi capturada pelo fonoaudiólogo.</p> <p>11.4. Retorna ao cenário principal.</p>
<b>Cenário alternativo 03</b>	No passo 11, caso o fonoaudiólogo selecione a opção de escolher uma imagem da biblioteca de imagens do aplicativo:

	<p>11.1. O aplicativo apresenta um <i>popover</i> com imagens de todas as categorias da biblioteca de imagens do aplicativo.</p> <p>11.2. O fonoaudiólogo seleciona uma categoria.</p> <p>11.3. O aplicativo apresenta no <i>popover</i> todas as imagens que fazem parte daquela categoria.</p> <p>11.4. O fonoaudiólogo seleciona uma imagem.</p> <p>11.5. O aplicativo substitui a imagem padrão pela imagem que foi selecionada pelo fonoaudiólogo.</p> <p>11.6. Retorna ao cenário principal.</p>
<b>Exceção 01</b>	<p>No passo 16, caso ocorra um erro ao salvar as informações:</p> <ol style="list-style-type: none"> <li>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações do símbolo.</li> <li>2. Retorna ao passo 2 do cenário principal.</li> </ol>
<b>Pós-condições</b>	Informações do símbolo salvas ou nenhuma informação salva.

Quadro 13 – Caso de uso Criar simbolos

UC05 – Criar planos	
Requisitos atendidos	RF05
<b>Pré-condições</b>	Para o fonoaudiólogo criar um plano é necessário que o fonoaudiólogo já tenha criado um tutor e pelo menos um símbolo.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo pressiona o botão <i>Tutores</i> na tela de detalhes do paciente.</p> <p>2. O aplicativo apresenta uma tela com a listagem de todos os tutores já criados para o paciente.</p> <p>3. O fonoaudiólogo seleciona um tutor nesta listagem.</p> <p>4. O aplicativo apresenta uma tela com a listagem de todos os planos já criados para o tutor selecionado.</p> <p>5. O fonoaudiólogo pressiona a imagem padrão que indica a criação de um novo plano.</p> <p>6. O aplicativo apresenta uma tela com a imagem do tutor selecionado, um campo preenchido com o nome do tutor, um campo preenchido com a data atual e campos em branco para o fonoaudiólogo informar o nome do plano e escrever a descrição do plano. O aplicativo também apresenta nesta tela um campo, que quando o fonoaudiólogo seleciona-o é apresentado duas opções para criação de planos.</p> <p>7. O fonoaudiólogo preenche o campo com o nome do plano.</p> <p>8. O fonoaudiólogo seleciona o campo tipo.</p> <p>9. O aplicativo apresenta duas opções para o fonoaudiólogo: Prancha e Símbolos Grandes.</p> <p>10. O fonoaudiólogo escolhe a opção Prancha.</p> <p>11. O aplicativo apresenta uma nova tela, com nove imagens padrões.</p> <p>12. O fonoaudiólogo pressiona uma das imagens padrões.</p> <p>13. O aplicativo apresenta um <i>popover</i> com imagens de todas as categorias criadas para o paciente.</p> <p>14. O fonoaudiólogo seleciona uma categoria.</p> <p>15. O aplicativo apresenta no <i>popover</i> todos os símbolos que fazem parte daquela categoria.</p> <p>16. O fonoaudiólogo seleciona um símbolo.</p> <p>17. O aplicativo substitui a imagem padrão pelo símbolo selecionado pelo fonoaudiólogo. O símbolo que é mostrado contém uma borda com a cor de sua categoria.</p> <p>18. O fonoaudiólogo pressiona o botão Concluir.</p> <p>19. O aplicativo volta a apresentar a tela de criação de planos.</p> <p>20. O fonoaudiólogo pressiona o botão OK na tela de criação de planos para salvar as informações.</p> <p>21. O aplicativo salva as informações e executa o UC06.</p>
<b>Cenário alternativo 01</b>	<p>No passo 10, caso o fonoaudiólogo escolha a opção Símbolos Grandes:</p> <p>10.1. O aplicativo apresenta uma nova tela, com apenas uma imagem padrão.</p> <p>10.2. O fonoaudiólogo pressiona esta imagem.</p> <p>10.3. O aplicativo apresenta um <i>popover</i> com imagens de todas as categorias já criadas para o paciente.</p> <p>10.4. O fonoaudiólogo seleciona uma categoria.</p> <p>10.5. O aplicativo apresenta no <i>popover</i> todos os símbolos que fazem parte daquela categoria.</p> <p>10.6. O fonoaudiólogo seleciona um símbolo.</p> <p>10.7. O aplicativo mostra o símbolo selecionado pelo fonoaudiólogo no local aonde estava a imagem padrão. O símbolo que é mostrado contém uma borda com a cor de sua categoria.</p> <p>10.8. O fonoaudiólogo pressiona o botão Concluir.</p> <p>10.9. Retorna ao cenário principal.</p>
<b>Exceção 01</b>	<p>No passo 21, caso ocorra um erro ao salvar as informações:</p> <p>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações do plano.</p> <p>2. Retorna ao passo 2 do cenário principal.</p>
<b>Pós-condições</b>	Informações do plano salvas ou nenhuma informação salva.

Quadro 14 – Caso de uso Criar planos

<b>UC06 – Enviar planos</b>	
<b>Requisitos atendidos</b>	RF06
<b>Pré-condições</b>	Para o fonoaudiólogo enviar um plano é necessário que ele já tenha selecionado um plano na tela que lista os planos ou que ele tenha pressionado o botão OK no momento de criação de um plano.
<b>Cenário principal</b>	<p>1. O aplicativo apresenta a tela padrão de envio de e-mails do iOS com um arquivo no formato .psyb anexado, com o endereço de destinatário preenchido com o e-mail do paciente, com o assunto do e-mail preenchido com o nome do plano e o corpo da mensagem de e-mail preenchido com a descrição do plano. Caso o plano seja do tipo Pranchas ou Símbolos grandes, no e-mail também é anexado um arquivo .pdf com a representação do plano criado.</p> <p>2. O fonoaudiólogo pressiona o botão Enviar.</p> <p>3. O aplicativo envia o e-mail para o paciente.</p>
<b>Pós-condições</b>	Arquivos enviados ou nenhum arquivo enviado.

Quadro 15 – Caso de uso Enviar planos

<b>UC07 – Criar histórico de observações</b>	
<b>Requisitos atendidos</b>	RF07
<b>Pré-condições</b>	Para o fonoaudiólogo criar um histórico é necessário que o fonoaudiólogo já tenha criado um paciente.
<b>Cenário principal</b>	<p>1. O fonoaudiólogo pressiona o campo de texto na tela de detalhes do paciente.</p> <p>2. O aplicativo apresenta uma tela com um campo de texto preenchido com as observações e as suas datas que já foram criadas para o paciente.</p> <p>3. O fonoaudiólogo escreve as observações.</p> <p>4. O fonoaudiólogo pressiona o botão OK.</p> <p>5. O aplicativo salva o texto escrito e associa o texto escrito à data em que ele foi criado.</p>
<b>Exceção 01</b>	No passo 5, caso ocorra um erro ao salvar o histórico: <p>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao salvar as informações do histórico.</p> <p>2. Retorna ao passo 2 do cenário principal.</p>
<b>Pós-condições</b>	Informações do histórico salvas ou nenhuma informação salva.

Quadro 16 – Caso de uso Criar histórico de observações

<b>UC08 – Enviar histórico de observações</b>	
<b>Requisitos atendidos</b>	RF08 e RF09
<b>Pré-condições</b>	Para o fonoaudiólogo enviar um histórico é necessário que o fonoaudiólogo já tenha criado um histórico.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. O fonoaudiólogo pressiona o campo de texto na tela de detalhes do paciente.</li> <li>2. O aplicativo apresenta uma tela com um campo de texto preenchido com as observações e as suas datas que já foram criadas para o paciente.</li> <li>3. O fonoaudiólogo pressiona o botão Enviar.</li> <li>4. O aplicativo gera o arquivo .pdf.</li> <li>5. O aplicativo apresenta ao fonoaudiólogo a tela padrão de envio de e-mails do iOS com um arquivo no formato .pdf anexado, com o assunto do e-mail preenchido com o texto “Histórico do paciente (nome do paciente)” e o corpo da mensagem de e-mail preenchido com o conteúdo do arquivo PDF.</li> <li>6. O fonoaudiólogo pressiona o botão Enviar.</li> <li>7. O aplicativo envia o e-mail para o fonoaudiólogo.</li> </ol>
<b>Exceção 01</b>	<p>No passo 4, caso ocorra um erro ao gerar o arquivo .pdf:</p> <ol style="list-style-type: none"> <li>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao gerar o .pdf.</li> <li>2. Retorna ao passo 2 do cenário principal.</li> </ol>
<b>Pós-condições</b>	Arquivo .pdf enviado ou nenhum arquivo enviado.

Quadro 17 – Caso de uso Enviar histórico de observações

<b>UC09 – Importar planos</b>	
<b>Requisitos atendidos</b>	RF14
<b>Pré-condições</b>	Para o tutor importar um plano é necessário que ele já tenha recebido um plano do fonoaudiólogo via e-mail.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. No aplicativo de e-mail padrão do iOS, o tutor do paciente faz a importação do arquivo com os dados do plano.</li> <li>2. O aplicativo apresenta uma mensagem ao tutor que o plano foi importado com sucesso.</li> </ol>
<b>Exceção 01</b>	<p>No passo 2, caso ocorra um erro ao importar os dados do arquivo:</p> <ol style="list-style-type: none"> <li>1. O aplicativo apresenta uma mensagem ao tutor contendo o erro que ocorreu ao importar os dados do arquivo.</li> <li>2. Retorna ao passo 1 do cenário principal.</li> </ol>
<b>Pós-condições</b>	Plano importado com sucesso ou plano não importado.

Quadro 18 – Caso de uso Importar planos

<b>UC10 – Escolher tutor</b>	
<b>Requisitos atendidos</b>	RF15
<b>Pré-condições</b>	Para o tutor que está auxiliando o paciente escolher um tutor é necessário que algum plano já tenha sido importado.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. O aplicativo apresenta em sua tela inicial uma lista com todos os tutores dos planos já importados.</li> <li>2. O tutor que está auxiliando o paciente seleciona um tutor.</li> <li>3. O aplicativo apresenta na mesma tela uma lista com todos os planos do tutor selecionado.</li> </ol>
<b>Pós-condições</b>	Lista com os planos do tutor selecionado apresentada na tela ou nenhuma lista apresentada.

Quadro 19 – Caso de uso Escolher tutor

<b>UC11 – Escolher plano</b>	
<b>Requisitos atendidos</b>	RF16
<b>Pré-condições</b>	Para o tutor que está auxiliando o paciente escolher um plano, é necessário que algum tutor já tenha sido selecionado por ele no UC10.
<b>Cenário principal</b>	1. O aplicativo apresenta em sua tela inicial uma lista com todos planos do tutor selecionado. 2. O tutor que está auxiliando o paciente seleciona um plano. 3. O aplicativo apresenta uma tela com os símbolos do plano selecionado.
<b>Pós-condições</b>	Símbolos do plano apresentados ou nenhum símbolo apresentado.

Quadro 20 – Caso de uso Escolher plano

<b>UC12 – Interagir com os símbolos</b>	
<b>Requisitos atendidos</b>	RF20
<b>Pré-condições</b>	Para o paciente interagir com os símbolos é necessário que o tutor que está auxiliando-o já tenha selecionado um plano no UC11.
<b>Cenário principal</b>	1. O paciente pressiona um símbolo na tela com os símbolos do plano. 2. Se o símbolo tiver uma mensagem de áudio associada, o aplicativo reproduz o áudio associado ao símbolo no volume padrão do sistema operacional. 3. O aplicativo salva a data de utilização do símbolo pressionado no histórico de uso dos símbolos.
<b>Pós-condições</b>	Nenhuma.

Quadro 21 – Caso de uso Interagir com os símbolos

<b>UC13 – Enviar observações sobre o paciente para o fonoaudiólogo</b>	
<b>Requisitos atendidos</b>	RF17
<b>Pré-condições</b>	Para o tutor enviar observações sobre o paciente para o fonoaudiólogo é necessário que o paciente já tenha pelo menos um plano importado.
<b>Cenário principal</b>	1. Na tela inicial do aplicativo o tutor seleciona seu nome na lista de tutores apresentados. 2. O aplicativo apresenta uma tela com todos os planos associados ao tutor selecionado e dois botões. Um botão para enviar observações e outro para visualizar o histórico de uso dos símbolos. 3. O tutor pressiona o botão Enviar Observações. 4. O aplicativo apresenta uma tela com um campo de texto e dois botões, um para sair da tela e outro para enviar as observações. 5. O tutor escreve as observações. 6. O tutor pressiona o botão Enviar. 7. O aplicativo apresenta ao tutor a tela padrão de envio de e-mails do iOS com um arquivo no formato .pobs anexado, com o endereço de destinatário preenchido com o e-mail do fonoaudiólogo, com o assunto do e-mail preenchido o texto “Observações do paciente (nome do paciente)” e o corpo da mensagem de e-mail preenchido com as observações escritas. 8. O tutor pressiona o botão Enviar. 9. O aplicativo envia o e-mail para o fonoaudiólogo.
<b>Pós-condições</b>	Observações enviadas ou nenhuma informação enviada.

Quadro 22 – Caso de uso Enviar observações sobre o paciente para o fonoaudiólogo

<b>UC14 – Visualizar histórico de uso dos símbolos</b>	
<b>Requisitos atendidos</b>	RF18
<b>Pré-condições</b>	Para o tutor visualizar o histórico de uso dos símbolos para o fonoaudiólogo é necessário que o paciente já tenha interagido com pelo menos um símbolo do plano.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. Na tela inicial do aplicativo o tutor seleciona seu nome na lista de tutores apresentados.</li> <li>2. O aplicativo apresenta uma tela com todos os planos associados ao tutor selecionado e dois botões. Um botão para enviar observações e outro para visualizar o histórico de uso dos símbolos.</li> <li>3. O tutor pressiona o botão <i>Visualizar Histórico</i>.</li> <li>4. O aplicativo apresenta uma tela com uma tabela mostrando todos os símbolos utilizados pelo paciente e a sua data de utilização e dois botões, um botão para sair da tela e um para botão enviar o histórico.</li> </ol>
<b>Pós-condições</b>	Nenhuma.

Quadro 23 – Caso de uso Visualizar histórico de uso dos símbolos

<b>UC15 – Enviar histórico de uso dos símbolos para o fonoaudiólogo</b>	
<b>Requisitos atendidos</b>	RF19
<b>Pré-condições</b>	Para o tutor enviar o histórico de uso dos símbolos é necessário que ele já tenha executado o UC14.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. O tutor pressiona o botão <i>Enviar</i>.</li> <li>2. O aplicativo apresenta ao tutor a tela padrão de envio de e-mails do iOS com um arquivo no formato .phist anexado, com o endereço de destinatário preenchido com o e-mail do fonoaudiólogo, com o assunto do e-mail preenchido com o texto “Histórico de uso dos símbolos do paciente (nome do paciente)”.</li> <li>3. O tutor pressiona o botão <i>Enviar</i> na tela de envio de e-mails.</li> <li>4. O aplicativo envia o e-mail para o fonoaudiólogo.</li> </ol>
<b>Pós-condições</b>	Histórico de uso dos símbolos enviados ou nenhuma informação enviada.

Quadro 24 – Caso de uso Enviar histórico de uso dos símbolos para o fonoaudiólogo

<b>UC16 – Importar arquivo com histórico de uso dos símbolos enviados pelo tutor</b>	
<b>Requisitos atendidos</b>	RF10
<b>Pré-condições</b>	Para o fonoaudiólogo importar o histórico de uso dos símbolos é necessário que ele já tenha recebido o histórico via e-mail.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. No aplicativo de e-mails padrão do iOS, o fonoaudiólogo faz a importação do arquivo com os dados do histórico de uso dos símbolos.</li> <li>2. O aplicativo apresenta uma mensagem ao fonoaudiólogo informando que o histórico foi importado com sucesso.</li> </ol>
<b>Exceção 01</b>	No passo 2, caso ocorra um erro ao importar os dados do arquivo: <ol style="list-style-type: none"> <li>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao importar os dados do arquivo.</li> <li>2. Retorna ao passo 1 do cenário principal.</li> </ol>
<b>Pós-condições</b>	Histórico importado com sucesso ou histórico não importado.

Quadro 25 – Caso de uso Importar arquivo com histórico de uso dos símbolos enviados pelo tutor

<b>UC17 – Visualizar histórico de uso dos símbolos enviados pelo tutor</b>	
<b>Requisitos atendidos</b>	RF11
<b>Pré-condições</b>	Para que o fonoaudiólogo visualize o histórico de uso dos símbolos enviados pelo tutor é necessário que ele já tenha feita a importação de pelo menos um arquivo de histórico enviado pelo tutor.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. O fonoaudiólogo entra na tela de detalhes do paciente.</li> <li>2. Na tela de detalhes do paciente o fonoaudiólogo pressiona o botão <b>Visualizar histórico</b>.</li> <li>3. O aplicativo apresenta uma tela com uma tabela mostrando todos os símbolos utilizados pelo paciente e a sua data de utilização.</li> </ol>
<b>Pós-condições</b>	Nenhuma.

Quadro 26 – Caso de uso Visualizar histórico de uso dos símbolos enviados pelo tutor

<b>UC18 – Importar arquivo com observações do paciente enviadas pelo tutor</b>	
<b>Requisitos atendidos</b>	RF12
<b>Pré-condições</b>	Para o fonoaudiólogo importar as observações do paciente enviadas pelo tutor é necessário que ele já tenha recebido as observações via e-mail.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. No aplicativo de e-mail padrão do iOS, o fonoaudiólogo faz a importação do arquivo com as observações enviadas pelo tutor.</li> <li>2. O aplicativo apresenta uma mensagem ao fonoaudiólogo informando que as observações foram importadas com sucesso.</li> </ol>
<b>Exceção 01</b>	<p>No passo 2, caso ocorra um erro ao importar os dados do arquivo:</p> <ol style="list-style-type: none"> <li>1. O aplicativo apresenta uma mensagem ao fonoaudiólogo contendo o erro que ocorreu ao importar os dados do arquivo.</li> <li>2. Retorna ao passo 1 do cenário principal.</li> </ol>
<b>Pós-condições</b>	Observações importadas com sucesso ou observações não importadas.

Quadro 27 – Caso de uso Importar arquivo com observações do paciente enviadas pelo tutor

<b>UC19 – Visualizar observações do paciente enviadas pelo tutor</b>	
<b>Requisitos atendidos</b>	RF13
<b>Pré-condições</b>	Para que o fonoaudiólogo visualize as observações do paciente enviadas pelo tutor é necessário que ele já tenha feito a importação de pelo menos um arquivo de observações enviado pelo tutor.
<b>Cenário principal</b>	<ol style="list-style-type: none"> <li>1. O fonoaudiólogo entra na tela de detalhes do paciente.</li> <li>2. O fonoaudiólogo pressiona o campo de texto na tela de detalhes do paciente.</li> <li>3. O aplicativo apresenta uma tela com um campo de texto preenchido com as observações e as suas datas. As observações importadas pelo fonoaudiólogo são mostradas em conjunto com as observações já escritas anteriormente pelo fonoaudiólogo.</li> </ol>
<b>Pós-condições</b>	Nenhuma.

Quadro 28 – Caso de uso Visualizar observações do paciente enviadas pelo tutor

## **APÊNDICE B – Descrição das entidades do aplicativo**

O aplicativo desenvolvido possui dois grupos distintos de entidades. O primeiro grupo é relativo às entidades que armazenam as informações criadas pelo fonoaudiólogo. O segundo grupo de entidades é relativo às entidades que armazenam as informações que foram criadas durante a importação de um plano.

A entidade `Patient` é responsável por armazenar as informações pessoais de cada paciente criado pelo fonoaudiólogo. Esta entidade possui relacionamento com as entidades `Tutor`, `Symbol` e `PatientHistoric`.

A entidade `Category` é responsável por armazenar as informações das categorias de símbolos criadas pelo fonoaudiólogo. Esta entidade possui relacionamento apenas com a entidade `Symbol`.

A entidade `Symbol` é responsável por armazenar as informações dos símbolos criados pelo fonoaudiólogo. Esta entidade possui relacionamento com as entidades `Patient`, `Category` e `SymbolPlan`.

A entidade `Tutor` é responsável por armazenar as informações dos tutores criados pelo fonoaudiólogo. Esta entidade possui relacionamento com as entidades `Patient` e `Plan`.

A entidade `Plan` é responsável por armazenar as informações dos planos criados pelo fonoaudiólogo. Esta entidade possui relacionamento com as entidades `Tutor` e `SymbolPlan`.

A entidade `SymbolPlan` é responsável por armazenar os símbolos de determinado plano. Esta entidade possui relacionamento com a entidade `symbol` e com a entidade `Plan`.

A entidade `PatientHistoric` é responsável por armazenar o histórico de observações do paciente. Esta entidade possui relacionamento apenas com a entidade `Patient`.

A entidade `PatientCategory` é responsável por armazenar as informações das categorias de símbolos que foram criadas durante a importação de um plano. Esta entidade possui relacionamento apenas com a entidade `PatientSymbol`.

A entidade `PatientTutor` é responsável por armazenar as informações dos tutores que foram criadas durante a importação de um plano. Esta entidade possui relacionamento apenas com a entidade `PatientPlan`.

A entidade `PatientSymbol` é responsável por armazenar as informações dos símbolos que foram criados durante a importação de um plano. Esta entidade possui relacionamento apenas com a entidade `PatientCategory`.

A entidade `PatientSymbolHistoric` é responsável por armazenar as informações do

histórico de uso dos símbolos do paciente. Esta entidade possui relacionamento apenas com a entidade `PatientSymbol`.

A entidade `PatientGeneralData` é responsável por armazenar as informações pessoais do paciente que está utilizando a aplicação. Esta entidade não possui relacionamento com nenhuma outra entidade.

A entidade `PatientPlan` é responsável por armazenar as informações dos planos que foram criados durante a importação de um plano que foi enviado pelo fonoaudiólogo. Esta entidade possui relacionamento com as entidades `PatientTutor` e `PatientSymbolPlan`.

A entidade `PatientSymbolPlan` é responsável por armazenar as informações dos símbolos de um determinado plano. Esta entidade possui relacionamento apenas com a `PatientPlan`.

## APÊNDICE C – Configuração do iCloud

Para configurar o iCloud no projeto é necessário entrar no *iOS Provisioning Portal*, criar um novo App ID e habilitá-lo para o iCloud. Essa configuração pode ser visualizada na Figura 38 e na Figura 39.

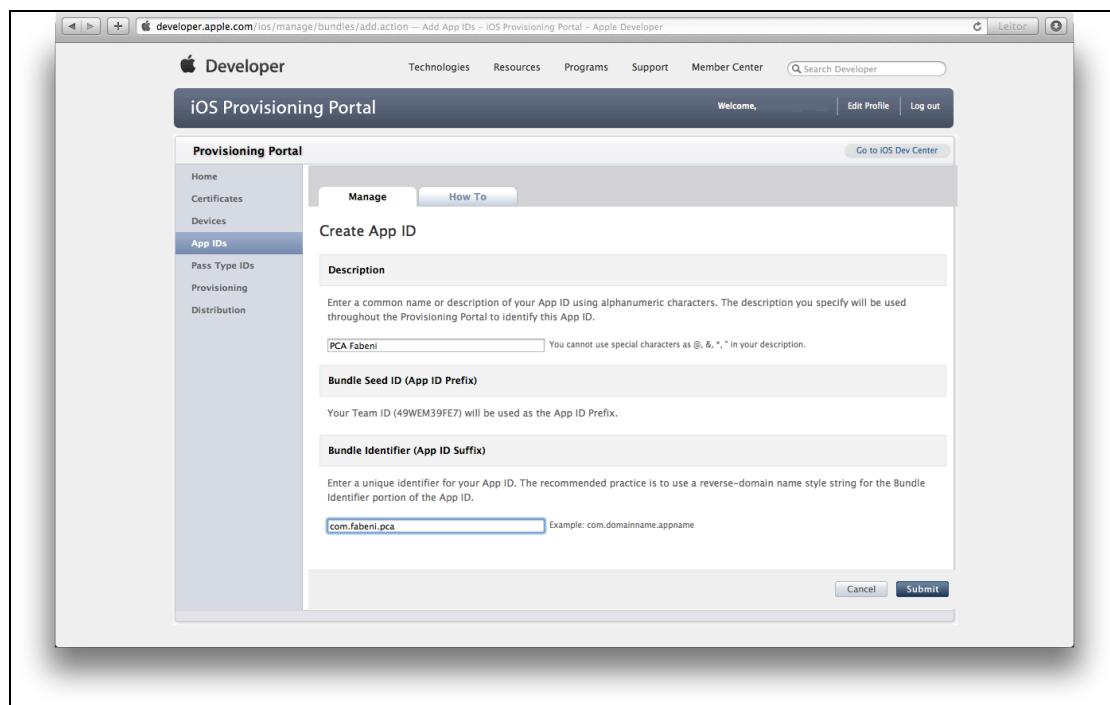


Figura 38 – Criação de um novo App ID

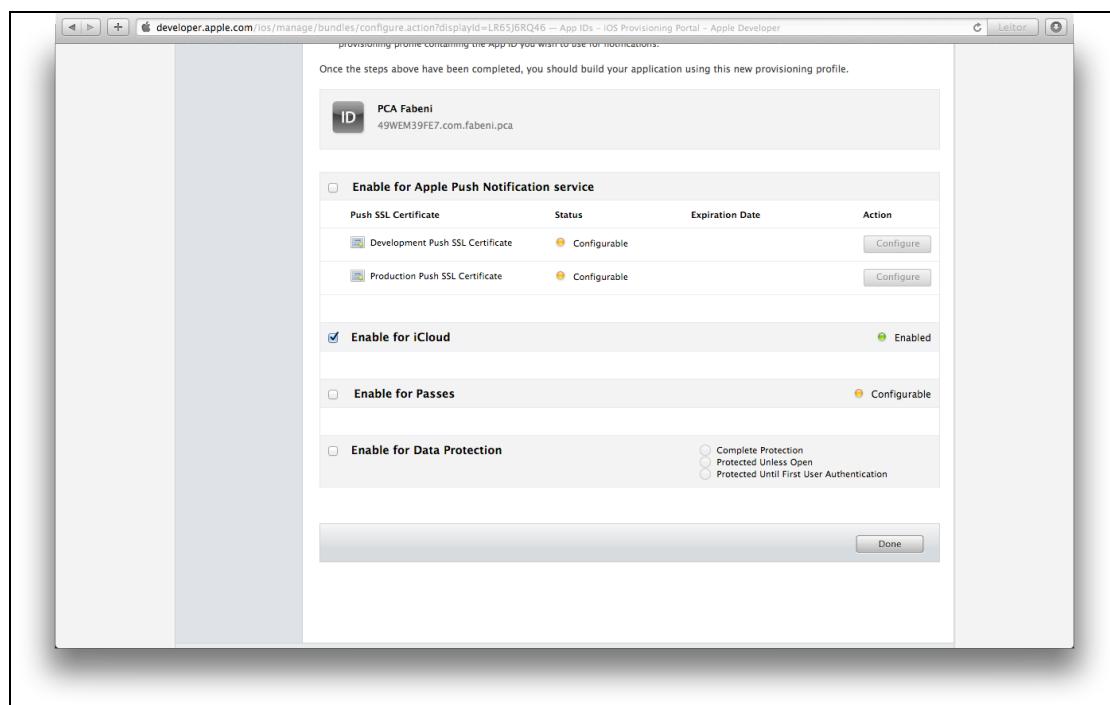


Figura 39 – Habilitação do App ID para o iCloud

Após criar e habilitar o App ID para o iCloud é necessário entrar no sumário do projeto no Xcode e configurar os *Entitlements*. Os *Entitlements* de um projeto são capacidades ou permissões de seguranças especiais específicas do aplicativo. A configuração dos *Entitlements* pode ser visualizada na Figura 40.

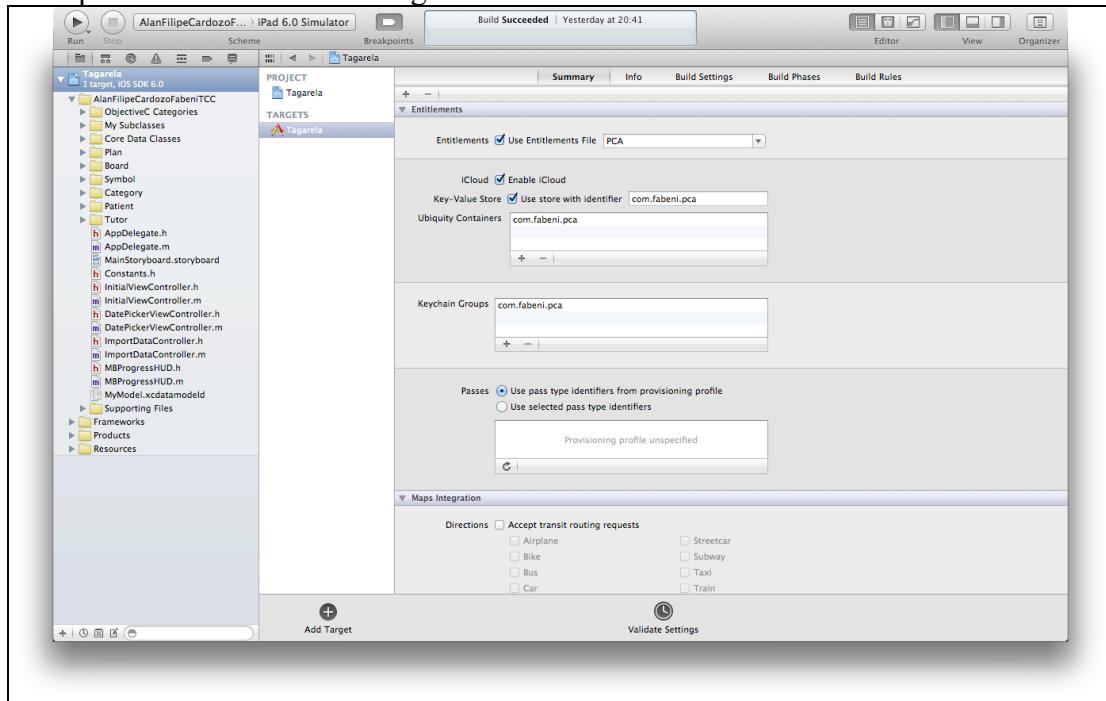


Figura 40 – Configuração dos *Entitlements*

## APÊNDICE D – Plano de atividades manual

Na figura 41 é possível visualizar um plano de atividades feito de forma manual.

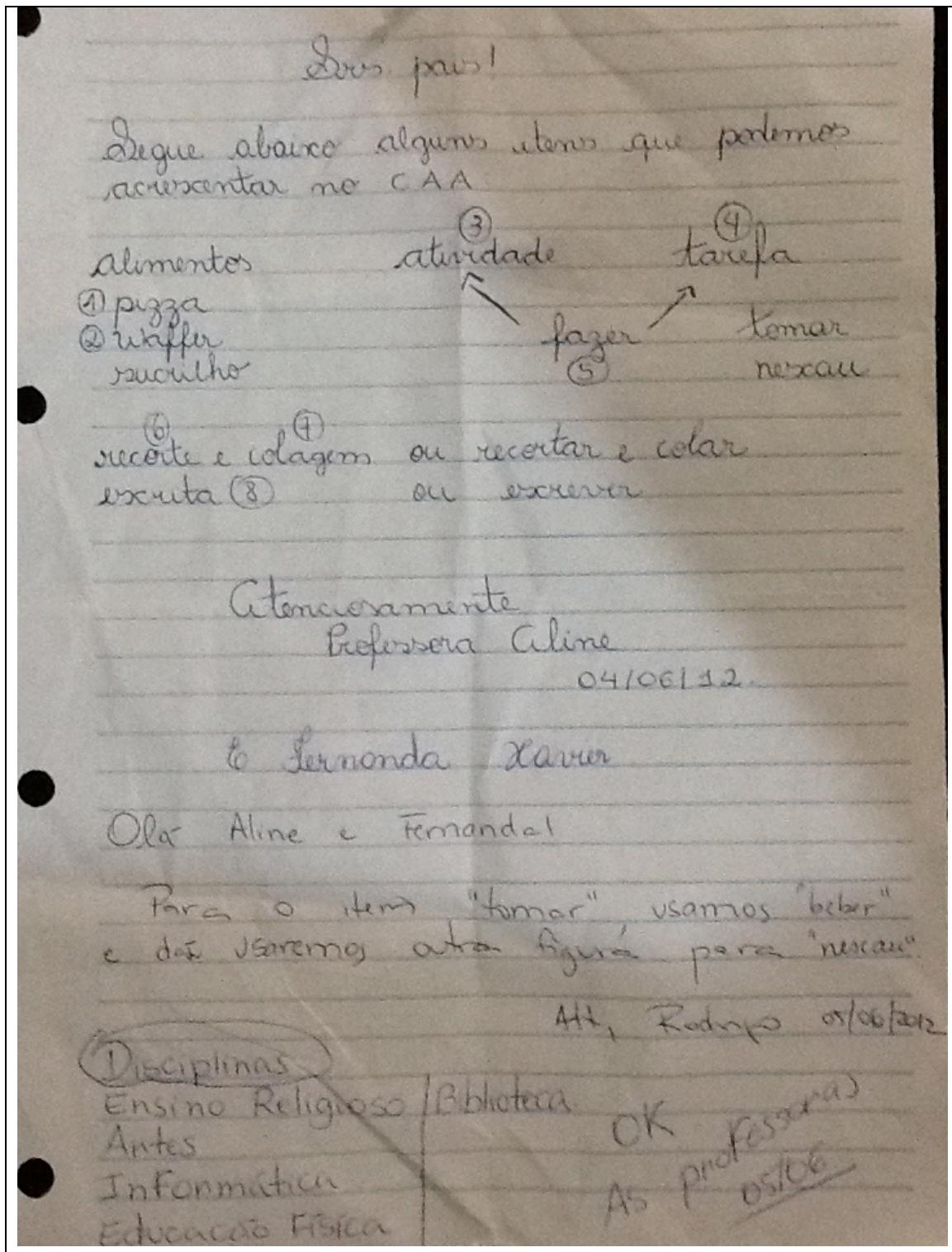


Figura 41 – Plano de atividades manual

## **APÊNDICE E – Histórico de observações da paciente Giovanna com o fonoaudiólogo Rodrigo França**

O Quadro 29 apresenta o histórico de observações da paciente Giovanna com o fonoaudiólogo Rodrigo França em suas sessões.

### **05/11/2012 - 09:15**

Giovanna atendeu a solicitação gerada pelo PCA prancha de ações, andando até ao livro, ao suco e ao fantoche. Precisou de ajuda, pois os amigos que estão no quadro de fotos lhe chamam a atenção. Na sessão do dia 30/10/2012 selecionou os campos dos PCAs.

### **06/11/2012 - 11:45**

Giovanna olhou para as figuras de PCA e logo em seguida para os objetos reais de livro, cócorico e fantoche. Indicou com a mão que queria o fantoche, pegando-o. Aproveitei e mostrei a ela que ela poderia comunicar Gio+brincar+fantoche.

### **12/11/2012 - 09:16**

Hoje a Gio estava dispersa e não selecionou os PCAs. Fazia movimentos repetitivos de bater com a mão na cabeça ou palmas.

### **13/11/2012 - 11:40**

Giovanna ficou muito interessada pela câmera, mas respondeu com varredura de olhar as figuras do PCA. Logo em seguida passamos a sessão como de costume e a Gio selecionou sozinha Gio+andar+livros.

Quadro 29 – Histórico de observações da paciente Giovanna com o fonoaudiólogo Rodrigo França