

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

TAGARELA: INTEGRAÇÃO E MELHORIAS NO
APLICATIVO DE REDE DE COMUNICAÇÃO
ALTERNATIVA

ANDRÉ FILIPE WIPPEL

BLUMENAU
2015

2015/1-02

ANDRÉ FILIPE WIPPEL

**TAGARELA: INTEGRAÇÃO E MELHORIAS NO
APLICATIVO DE REDE DE COMUNICAÇÃO
ALTERNATIVA**

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

**BLUMENAU
2015**

2015/1-02

**TAGARELA: INTEGRAÇÃO E MELHORIAS NO
APLICATIVO DE REDE DE COMUNICAÇÃO
ALTERNATIVA**

Por

ANDRÉ FILIPE WIPPEL

Trabalho de Conclusão de Curso 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, M. Sc. – Orientador, FURB

Membro: _____
Prof. Nome do professor, Titulação – FURB

Membro: _____
Prof. Nome do professor, Titulação – FURB

Dedico este trabalho à família e a todos os meus amigos, especialmente meus pais e meu irmão que estiveram ao meu lado me apoiando sempre que necessário.

AGRADECIMENTOS @@DEIXAR PARA O FINAL

A meus pais e meu irmão pelo apoio e incentivo a mim dispostos.

Aos meus tios Amarildo e Tereza Doege que também me apoiaram muito ao longo destes anos.

Ao meu orientador Dalton Solano dos Reis pela confiança, dedicação e apoio a mim dispostos ao longo do desenvolvimento deste trabalho.

A todos os meus amigos e colegas de turma que trilharam este caminho comigo e me apoiaram constantemente.

A todos os professores da FURB que me concederam direta ou indiretamente conhecimento e capacidade para que eu pudesse completar este percurso de minha vida.

Só se pode alcançar um grande êxito quando
nos mantemos fiéis a nós mesmos.

Friedrich Nietzsche

RESUMO @@DEIXAR PARA O FINAL

O resumo é uma apresentação concisa dos pontos relevantes de um texto. Informa suficientemente ao leitor, para que este possa decidir sobre a conveniência da leitura do texto inteiro. Deve conter OBRIGATORIAMENTE o **OBJETIVO, METODOLOGIA, RESULTADOS e CONCLUSÕES**. O resumo deve conter de 150 a 500 palavras e deve ser composto de uma sequência corrente de frases concisas e não de uma enumeração de tópicos. O resumo deve ser escrito em um único texto corrido (sem parágrafos). Deve-se usar a terceira pessoa do singular e verbo na voz ativa (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2003).

Palavras-chave: Ciência da computação. Monografia. Resumo. Formato.

[Palavras-chave são separadas por ponto, com a primeira letra maiúscula. Caso uma palavra-chave seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]

ABSTRACT @@DEIXAR PARA O FINAL

Abstract é o resumo traduzido para o inglês. *Abstract* vem em uma nova folha, logo após o resumo. Escrever com letra normal (sem itálico).

Key-words: Computer science. Monograph. Abstract. Format.

[*Key-words* são separadas por ponto, com a primeira letra maiúscula. Caso uma *key-word* seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]

LISTA DE FIGURAS

Figura 1– Utilização da prancha de comunicação no aplicativo	18
Figura 2– <i>Plugins</i> suportados pelo PhoneGap.....	19
Figura 3– Tela principal do Sono Flex	20
Figura 4– Visualização do perfil e símbolos no HelpTalk	21
Figura 5– Casos de uso do aplicativo	24
Figura 6– Diagrama de atividades principal do Tagarela.....	30
Figura 7– Diagrama de criação de uma prancha de comunicação	31
Figura 8– Diagrama de classes da aplicação	32
Figura 9– MER do banco de dados	33
Figura 10– Arquitetura da aplicação	34
Figura 11– Comandos do PhoneGap CLI	35
Figura 12– Tela inicial e criação de usuário.....	43
Figura 13– Envio e aceitação de convites	43
Figura 14– Tela principal do <i>builder</i>	44
Figura 15– Escolha dos símbolos para a prancha.....	45
Figura 16– Reutilização de uma prancha	46
Figura 17– Nova observação	47
Figura 18– Histórico de atividades.....	48
Figura 19– Utilização de uma prancha	49

LISTA DE QUADROS

Quadro 1 – Características dos trabalhos correlatos e o proposto	22
Quadro 2 – Caso de uso UC01	25
Quadro 3 – Caso de uso UC02	25
Quadro 4 – Caso de uso UC03	26
Quadro 5 – Caso de uso UC04	26
Quadro 6 – Caso de uso UC05	27
Quadro 7 – Caso de uso UC06	28
Quadro 8 – Caso de uso UC07	28
Quadro 9 – Caso de uso UC08	29
Quadro 10 – Caso de uso UC09	29
Quadro 11 – Método <i>reutilizarPrancha</i>	37
Quadro 12 – Método <i>criarPrancha</i>	39
Quadro 13 – <i>Script</i> de gravação da prancha na base de dados	40
Quadro 14 – Métodos <i>usarPrancha e gravarLog</i>	41

LISTA DE TABELAS @@ATUALIZAR

Tabela 1 – Trabalhos finais realizados no Curso de Ciência da Computação**Error! Bookmark not defined.**

[Só deixar mais de uma lista por página se as mesmas couberem por completo. O título das listas fica centralizado em negrito.]

LISTA DE ABREVIATURAS E SIGLAS @@ATUALIZAR

[Deve conter as abreviaturas e siglas utilizadas mais de uma vez ao longo do texto em ordem alfabética. A seguir estão dois exemplos de forma de apresentação.]

ABNT – Associação Brasileira de Normas Técnicas

API – *Application Programming Interface*

SUMÁRIO

1 INTRODUÇÃO.....	14
1.1 OBJETIVOS DO TRABALHO	15
1.2 ESTRUTURA.....	15
2 FUNDAMENTAÇÃO TEÓRICA	16
2.1 COMUNICAÇÃO ALTERNATIVA E PRANCHAS DE COMUNICAÇÃO	16
2.2 TAGARELA.....	16
2.3 PHONEGAP E MATERIALIZE	18
2.4 TRABALHOS CORRELATOS	19
2.4.1 Sono Flex	20
2.4.2 HelpTalk.....	20
2.4.3 Comparação entre os trabalhos correlatos e o proposto.....	21
3 DESENVOLVIMENTO	23
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	23
3.2 ESPECIFICAÇÃO	23
3.2.1 Casos de uso.....	24
3.2.1.1 Aceitar convites	24
3.2.1.2 Enviar convites	25
3.2.1.3 Alterar dados pessoais	25
3.2.1.4 Usar prancha	26
3.2.1.5 Criar prancha	26
3.2.1.6 Reutilizar prancha.....	27
3.2.1.7 Visualizar observações	28
3.2.1.8 Criar observação	28
3.2.1.9 Visualizar históricos	29
3.2.2 Diagrama de atividades	29
3.2.3 Diagrama de classes	31
3.2.4 Diagrama MER	33
3.3 IMPLEMENTAÇÃO	33
3.3.1 Técnicas e ferramentas utilizadas.....	34
3.3.1.1 Arquitetura proposta	34
3.3.1.2 Utilização do PhoneGap	35

3.3.1.3 Códigos da criação e reutilização de pranchas	36
3.3.1.4 Interação com as pranchas criadas	40
3.3.2 Operacionalidade da implementação	42
3.3.2.1 Criação de usuários	42
3.3.2.2 Vínculo de usuários	43
3.3.2.3 Criação de prancha de comunicação.....	44
3.3.2.4 Compartilhamento de pranchas	45
3.3.2.5 Criação de observações.....	46
3.3.2.6 Visualização dos históricos.....	47
3.3.2.7 Utilização das pranchas	48
3.4 RESULTADOS E DISCUSSÕES @@ FAZER.....	49
4 CONCLUSÕES @@ FAZER.....	50
4.1 EXTENSÕES @@ FAZER	50
REFERÊNCIAS @@ IR ATUALIZANDO A MEDIDA QUE SÃO REFERENCIADOS	
NA MONOGRAFIA	51
APÊNDICE A – Relação dos formatos das apresentações dos trabalhos @@ Ir	
atualizando a medida que são referenciados na monografia.....	52
ANEXO A – Representação gráfica de contagem de citações de autores por semestre nos	
trabalhos de conclusões realizados no Curso de Ciência da Computação @@ Ir	
atualizando a medida que são referenciados na monografia.....	53

1 INTRODUÇÃO

A comunicação é um mecanismo presente entre os humanos desde a época em que viviam em cavernas e sabe-se hoje em dia que existem diversas formas de se comunicar, dentre elas a comunicação oral é sem dúvida a mais utilizada (FOTON, 2008). Porém, muitas pessoas, devido a fatores físicos ou até mesmo psicológicos, não conseguem se expressar através da fala.

Desde quando crianças a fala já é algo muito importante, pois é ela que humaniza a criança, que a coloca como ser humano na sociedade atual, e que nomeia e define tudo que está ao seu redor (TEIXEIRA, 2013). Portanto, crianças que se mostram incapazes de se comunicar têm grandes chances de se sentirem excluídas socialmente.

Atualmente já existem diversas ferramentas e aplicações comerciais que servem como um auxílio para estas pessoas com deficiência ou dificuldade na fala, definidas como ferramentas de CA (Comunicação Alternativa). As ferramentas CAs, na sua maioria, utilizam o conceito de Pranchas de Comunicação, que são um conjunto de símbolos no qual o usuário utiliza para se comunicar com maior clareza e facilidade. Porém, nem todas estas ferramentas são de fácil uso, ou mesmo que sejam, muitas vezes se faz necessário o acompanhamento de um responsável para o auxílio na utilização dependendo do estado clínico do usuário.

Desta forma, é possível identificar que existe um grande avanço ainda a ser feito em relação as ferramentas de CA. Foi visto que o aplicativo Tagarela (TAGARELA, 2014), desenvolvido em um projeto acadêmico pela Universidade Regional de Blumenau, possui várias funcionalidades que auxiliam na comunicação de pessoas com limitações fonoarticulares, mas procura principalmente propiciar um ambiente que permita armazenar e compartilhar as experiências relacionadas ao desenvolvimento do usuário. Estas experiências são de grande valia para os profissionais que auxiliam o usuário, seja o tutor, ou mesmo o fonoaudiólogo (especialista).

Diante do exposto, esta proposta propõe o desenvolvimento de uma série de melhorias no projeto Tagarela, onde será fornecida uma interface mais acessível para os usuários, além de fornecer o acesso as funcionalidades de maneira distinta para cada tipo de usuário presente na aplicação (Especialista, Tutor e Paciente). Todas as funcionalidades desenvolvidas, inclusive as já existentes, deverão ser disponibilizadas em todas as versões atuais do aplicativo. Para isto será utilizado o framework de desenvolvimento móvel PhoneGap.

Assim como outras ferramentas de CA, o aplicativo Tagarela fornece uma alternativa e auxilia na insuficiência da fala de pessoas com alguma dificuldade ou deficiência. No entanto,

é de suma importância que aplicações deste tipo possuam uma interface acessível e que sejam simples de usar, para que os usuários se motivem e gradativamente possam adquirir a autonomia na sua utilização.

O trabalho proposto se mostra relevante no âmbito social, pois visa aprimorar o aplicativo Tagarela para atender um maior grupo de usuários, independente da plataforma a ser utilizada (Android, iOS e Web). Além disso, se mostra relevante também na área tecnológica, pois será estudado e utilizado um único framework de desenvolvimento para integrar as três versões disponíveis do aplicativo, Android, iOS e Web. Isto facilitará muito o desenvolvimento de futuras melhorias e continuações no projeto Tagarela, pois um dos problemas atuais do projeto é o grande esforço necessário para manter e evoluir as três versões da aplicação de forma separada.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é aprimorar o “Tagarela: Aplicativo de Comunicação Alternativa na Plataforma Android” (MARCO, 2014), integrando o desenvolvimento entre as versões Android, iOS e Web.

Os objetivos específicos do trabalho são:

- a) disponibilizar o acesso ao aplicativo através de perfis distintos (Especialista, Tutor e Paciente);
- b) disponibilizar uma interface mais acessível utilizando o conceito de pranchas de comunicação;
- c) integrar o desenvolvimento das versões do Tagarela em um único framework.

1.2 ESTRUTURA

Este trabalho está estruturado em quatro capítulos, sendo que no primeiro é apresentada a introdução ao tema, bem como os objetivos e a estrutura deste trabalho.

O segundo capítulo aborda a fundamentação teórica necessária para o melhor entendimento deste trabalho.

O capítulo três contempla as etapas de desenvolvimento do aplicativo, onde são apresentados os requisitos, os diagramas para melhor entendimento e a implementação, demonstrando alguns quadros com código fonte.

Por fim, o capítulo quatro trata das conclusões obtidas do presente trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 trata sobre Comunicação Alternativa e o conceito de Pranchas de Comunicação. A seção 2.2 descreve o projeto acadêmico Tagarela. A seção 2.3 apresenta a plataforma de desenvolvimento móvel PhoneGap em conjunto com o *framework* Materialize. Por fim, na seção 2.4, são apresentados os trabalhos correlatos.

2.1 COMUNICAÇÃO ALTERNATIVA E PRANCHAS DE COMUNICAÇÃO

Comunicação Alternativa (CA) é a área da tecnologia assistiva que se destina a ampliação das habilidades de comunicação de pessoas sem fala ou sem escrita funcional. Cada indivíduo fará uso dos recursos de comunicação com características distintas, que por sua vez devem atender suas necessidades específicas (SARTORETTO; BERSCH, 2012).

Atualmente os sistemas de CA são usados na área clínica, para compensar temporária ou permanentemente as dificuldades de indivíduos com desordens de expressão. Antes de fazer uso funcional de qualquer ferramenta assistiva, os profissionais e especialistas da área devem tomar como preocupação a diversidade de aspectos envolvidos em cada usuário, para então indicar um recurso que otimize o uso dos sistemas de CA (GONÇALVES, 2008).

Dentro da CA um recurso, é o objeto ou ferramenta que será utilizado para transmitir as mensagens. Um dos mais utilizados atualmente são as pranchas de comunicação, que são aglomerados de símbolos, letras, sílabas, palavras, frases ou números. Cada prancha deve ser personalizada com seus símbolos de acordo com as possibilidades cognitivas, visuais e motoras de seu usuário (PELOSI, 2011).

Cada conjunto de símbolos está ligado diretamente a um sistema de símbolos gráficos, que são uma coleção de imagens que apresentam características comuns entre si e foram criadas para atender a diferentes necessidades de comunicação dos usuários. Existem diferentes sistemas simbólicos, sendo que o mais utilizado em todo o mundo é o Picture Communication Symbols (PCS), que possui como característica os desenhos simples e de fácil reconhecimento (SARTORETTO; BERSCH, 2012).

Utilizando todos estes conceitos em conjunto pode-se fornecer uma ferramenta muito eficaz para pessoas com dificuldades na fala, fazendo com que elas quebrem barreiras sociais e tenham uma melhor qualidade de vida.

2.2 TAGARELA

O Tagarela é um projeto da Universidade Regional de Blumenau que tem por objetivo desenvolver uma plataforma que auxilie no tratamento de usuários com necessidades

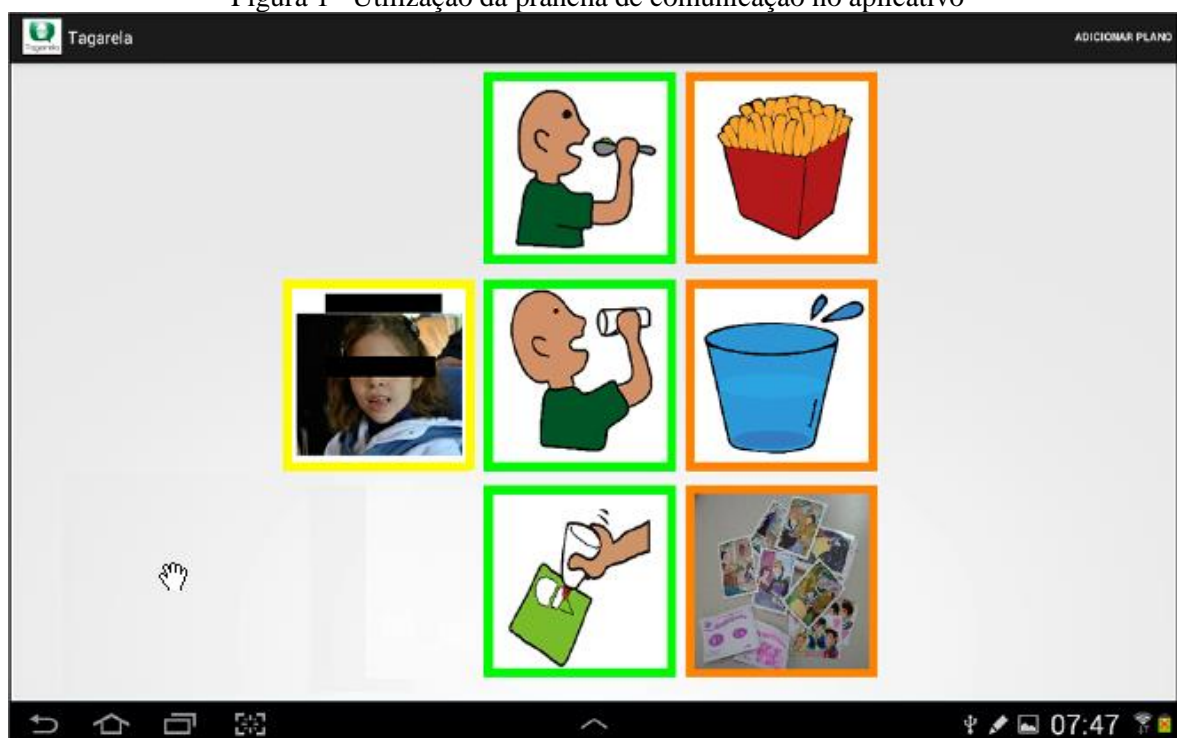
especiais, ou alguma limitação fonoarticulosa, fornecendo uma forma de Comunicação Alternativa. Além disso, o projeto busca facilitar a troca de experiências entre os indivíduos no decorrer do tratamento, que são denominados pelos perfis “Especialista”, “Tutor” e “Paciente” (TAGARELA, 2014).

Fabeni (2012) desenvolveu o primeiro produto deste projeto, um aplicativo para a plataforma iOS. Um dos principais objetivos deste aplicativo é disponibilizar um ambiente no qual o especialista, representado por um fonoaudiólogo, possa trocar experiências com o usuário em conjunto com seu tutor, geralmente um profissional da área da pedagogia, montar um plano de atividade para estimular a capacidade de comunicação do usuário (TAGARELA, 2014). Posteriormente este aplicativo foi desenvolvido também para a plataforma Android, através do trabalho acadêmico “Tagarela: Aplicativo de Comunicação Alternativa na Plataforma Android”, desenvolvido por Darlan Diego de Marco (MARCO, 2014).

O foco principal do trabalho desenvolvido por Marco (2014) é tornar a experiência do usuário interativa através da utilização de pranchas de comunicação, além de possibilitar a sincronização das informações entre dispositivos e permitir o uso da aplicação sem conexão com a internet. Porém, algumas funcionalidades não foram tratadas, como permitir a utilização de perfis para cada tipo de usuário (Especialista, Tutor e Paciente), possibilitar a reprodução de áudio encadeada nas pranchas de comunicação e também permitir que o usuário movimente os símbolos nas pranchas (MARCO, 2014).

Uma das principais atividades do aplicativo está relacionada à criação de um plano, pois é por meio desta atividade que o especialista e o tutor irão interagir e buscar resultados com seus usuários. Um plano contém as atividades do usuário e os símbolos que ele irá interagir, por meio de uma prancha de comunicação conforme a Figura 1 (MARCO, 2014).

Figura 1– Utilização da prancha de comunicação no aplicativo



Fonte: Marco (2014).

Está em construção também uma aplicação Web que deve possuir as mesmas funcionalidades das versões atuais do Tagarela disponíveis para os dispositivos móveis com Android e iOS.

2.3 PHONEGAP E MATERIALIZE

PhoneGap é um *framework* de código desenvolvido pela Adobe Systems, tem licença de uso gratuita e permite a criação de aplicações móveis utilizando APIs (Application Programming Interfaces) *web* padronizadas.

Em Outubro de 2011 o PhoneGap foi doado a Apache Software Foundation (ASF), porém permanece sob licença gratuita, fazendo parte agora do projeto Apache Cordova (ADOBE, 2014).

Existem diversas aplicações criadas através do PhoneGap, que já estão sendo distribuídas comercialmente (ADOBE, 2014). O framework PhoneGap permite o desenvolvimento de aplicativos para múltiplas plataformas de maneira simples, apenas escrevendo o código uma única vez, utilizando HTML, CSS e JavaScript. E, após isso, é possível realizar o deploy para as plataformas móveis desejadas (ADOBE, 2014).

O PhoneGap possui diversos *plugins* que são instalados na aplicação para auxiliar no desenvolvimento. A Figura 2, mostra quais são os *plugins* suportados em relação a cada plataforma móvel, no desenvolvimento de um aplicativo utilizando o PhoneGap. Onde a

primeira coluna mostra o *plugin* e as demais colunas mostram um comparativo entre as plataformas.

Figura 2– *Plugins* suportados pelo PhoneGap

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	✓	✓	✓	X
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

Fonte: Adobe (2014).

Para a construção do estilo das telas no PhoneGap é utilizada a linguagem CSS, o que permite também a utilização de *frameworks* CSS, que surgem para facilitar este desenvolvimento de estilos. O Materialize é um *framework* CSS, baseado no Material Design da Google, que atende perfeitamente a necessidade de agilizar e facilitar este processo de desenvolvimento. Além de proporcionar um *design* limpo e intuitivo, ele possui também componentes padronizados, o que torna a experiência do usuário unificada (MATERIALIZE, 2015).

2.4 TRABALHOS CORRELATOS

Foram selecionados dois trabalhos correlatos, ambos são aplicações de comunicação alternativa que utilizam o conceito de pranchas de comunicação. O item 2.4.1 descreve a ferramenta Tobii Sono Flex e o item 2.4.2 descreve a aplicação HelpTalk, ambas ferramentas comerciais disponíveis na plataforma Android. Por fim, no item 2.4.3, é demonstrada uma tabela comparativa em relação aos trabalhos correlatos e o trabalho proposto.

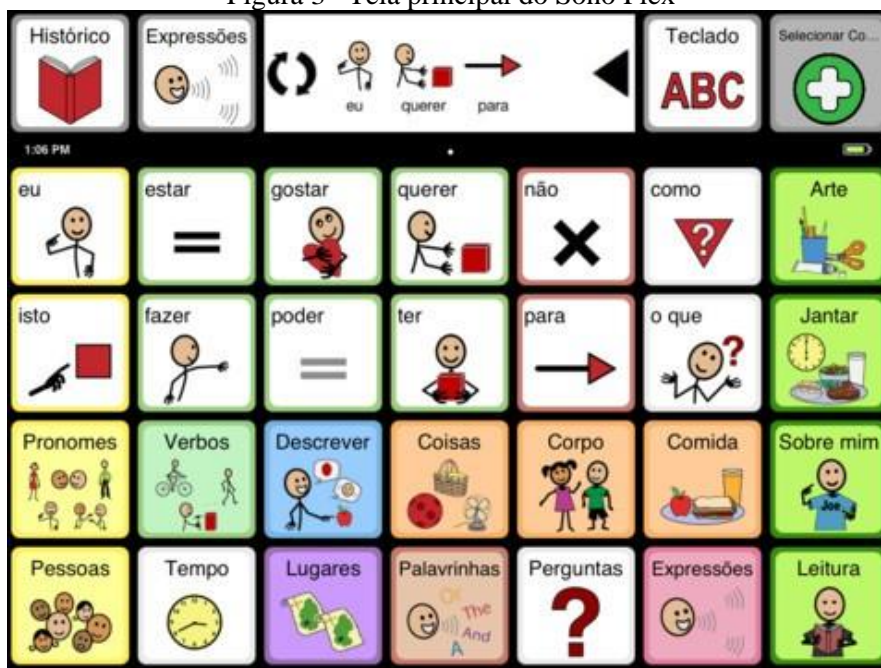
2.4.1 Sono Flex

Desenvolvido pela Comércio de Materiais Esportivos e Educativos Civiam Ltda EPP, é um aplicativo de comunicação assistiva e alternativa que oferece o recurso da linguagem aos usuários sem capacidade verbal ou alfabetização (CIVIAM, 2014). O Sono Flex é disponibilizado nas plataformas Android e iOS.

O Sono Flex oferece uma grande flexibilidade no que diz respeito a personalização do vocabulário, atendendo necessidades individuais e situacionais. Além de permitir a utilização da câmera e álbum de fotos do dispositivo para a criação de novos símbolos (CIVIAM, 2014).

A principal característica do aplicativo é sua fácil utilização, pois faz uso de uma interface baseada em símbolos, como visto na Figura 3, tornando a experiência do usuário muito mais agradável. Isto também permite que usuários sem alfabetização utilizem a aplicação sem maiores dificuldades.

Figura 3– Tela principal do Sono Flex



Fonte: Civiam (2014).

O aplicativo ainda permite ao usuário customizar esta disposição de símbolos na tela principal, para ficar de acordo com sua necessidade.

2.4.2 HelpTalk

O HelpTalk, assim como outros aplicativos de Comunicação Alternativa, é voltado para pessoas incapazes de estabelecer comunicação fluente oral ou até mesmo escrita. Por meio de um conjunto de ações (símbolos) pré-definidas qualquer pessoa pode se comunicar,

inclusive crianças e analfabetos, uma vez que são utilizados ícones juntamente com o texto (HELPTALK, 2014).

Uma das particularidades da aplicação é a criação de perfis, que irão conter os símbolos mais apropriados para determinada situação ou determinado tipo de usuário. Estes perfis podem ser mantidos de maneira privada ou pública, sendo que nesta última qualquer usuário poderá clonar o perfil e adequá-lo as suas necessidades. De maneira a facilitar o uso do aplicativo, os usuários podem também carregar seus perfis para o dispositivo e utilizá-los de modo offline.

A Figura 4 mostra a tela principal do aplicativo que contém as ações disponíveis para um determinado perfil criado.

Figura 4– Visualização do perfil e símbolos no HelpTalk



Fonte: HelpTalk (2014).

2.4.3 Comparação entre os trabalhos correlatos e o proposto

O Quadro 1 apresenta de forma comparativa algumas características em relação aos trabalhos correlatos e o trabalho apresentado nesta proposta.

Quadro 1 – Características dos trabalhos correlatos e o proposto

Características	Sono Flex (2014)	HelpTalk (2014)	Tabalho Proposto
tratamento diferenciado para perfis de usuário	não	sim	sim
utiliza o conceito de pranchas de comunicação	sim	sim	sim
plataformas suportadas	Android e iOS	Android	Android, iOS e Web
reprodução de áudio encadeada	sim	sim	não
permite o uso do aplicativo no modo <i>offline</i>	não	sim	sim
interface acessível para usuários com mobilidade reduzida	sim	sim	não

Em relação ao trabalho proposto, pode-se perceber através do Quadro 1 que os trabalhos correlatos possuem melhor apresentação das telas do aplicativo, além de possuírem a possibilidade da reprodução de áudio encadeada. Porém, o trabalho proposto se mostra mais interessante em relação a diversidade de plataformas que suporta, permitindo que o usuário tenha uma maior liberdade na escolha da plataforma na qual irá utilizar a aplicação.

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as etapas de desenvolvimento do aplicativo. A primeira seção apresenta os requisitos funcionais e não-funcionais. Em seguida, a segunda seção contém a especificação do aplicativo, utilizando diagramas da *Unified Modeling Language* (UML). A terceira seção detalha a implementação do aplicativo, apresentando os principais trechos de código e exemplos de uso das telas. Por fim, a quarta seção aborda os resultados obtidos deste trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O aplicativo descrito nesta proposta deverá:

- a) permitir a criação de usuários com o perfil de “Tutor” ou “Paciente” (RF);
- b) permitir o vínculo entre usuários com o perfil de “Tutor” e usuários com o perfil de “Paciente” (RF);
- c) permitir aos usuários a inserção e alteração de suas informações pessoais (RF);
- d) permitir a criação e reutilização de pranchas de comunicação (RF);
- e) permitir a interação do usuário com suas pranchas de comunicação (RF);
- f) permitir que o usuário com o perfil de “Tutor” possa criar e visualizar observações referentes a seus pacientes (RF);
- g) permitir que o usuário possa visualizar seu histórico de uso das pranchas de comunicação (RF);
- h) apresentar uma interface acessível, utilizando o conceito de símbolos de uma prancha nas telas da aplicação (Requisito Não-Funcional - RNF);
- i) permitir uma fácil integração de novas funcionalidades entre as versões Android, iOS e Web (RNF);
- j) implementar todas as funções já existentes nas versões anteriores utilizando um único framework (RNF);
- k) ser implementado utilizando o framework PhoneGap (RNF).

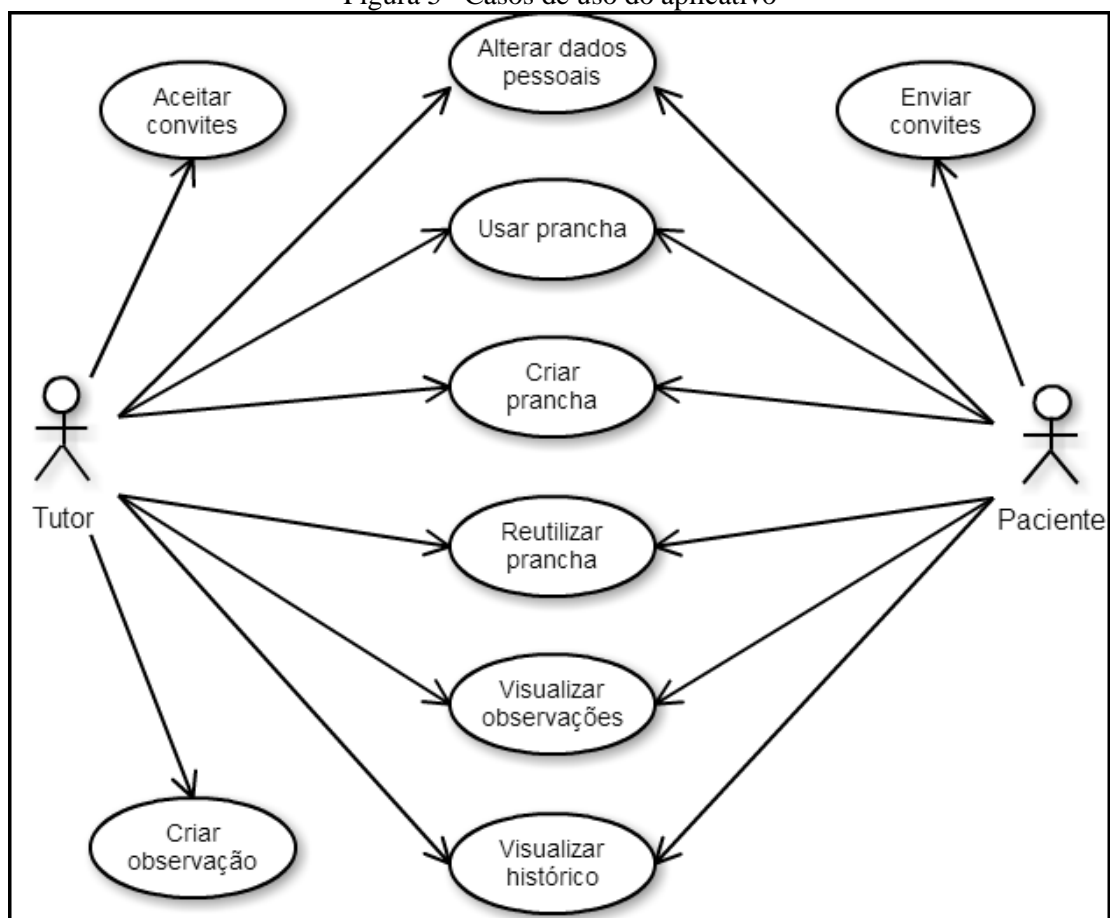
3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando modelagem de diagrama de casos de uso, diagrama de atividades, modelo de entidade e relacionamento (MER) e diagrama de classes, todos da UML.

3.2.1 Casos de uso

Nesta seção são apresentados os casos de uso que descrevem as funcionalidades do aplicativo após os usuários estarem cadastrados. No aplicativo existem dois perfis de usuário, o tutor e o paciente. O tutor tem como principal responsabilidade criar e gerenciar os planos e as pranchas de comunicação que serão utilizadas pelos pacientes. O paciente, por sua vez, tem a responsabilidade de interagir com estes planos e pranchas, criados por seus tutores. Desta forma, foram identificados dois atores dentro do diagrama de casos de uso, conforme a Figura 5.

Figura 5– Casos de uso do aplicativo



Os casos de uso acima foram criados a partir dos requisitos funcionais, visando organizá-los em operações que possam ser executadas pelos usuários. A seguir são detalhados os respectivos casos de uso.

3.2.1.1 Aceitar convites

Este caso de uso descreve como o *Tutor* aceita os convites enviados por outros usuários. Detalhes são mostrados no Quadro 2.

Quadro 2 – Caso de uso UC01

UC01 – Aceitar convites	
Requisitos atendidos	RF2
Pré-condições	1. O usuário deve ter o perfil <i>Tutor</i> . 2. Algum usuário com o perfil <i>Paciente</i> precisa ter enviado um convite para este usuário (UC02).
Cenário principal	1. São apresentados para o usuário todos os <i>Pacientes</i> que enviaram convites para ele. 2. O usuário seleciona de qual <i>Paciente</i> deseja aceitar o convite e visualiza sua mensagem. 3. O usuário toca no botão <i>Confirmar</i> para aceitar o convite.
Exceção	Se o usuário não possuir nenhum convite é apresentada a mensagem “Não há convites pendentes”.
Pós-condição	O <i>Paciente</i> ficará vinculado ao usuário que aceitou o convite, permitindo então realizar outras operações no aplicativo.

3.2.1.2 Enviar convites

Este caso de uso descreve como o *Paciente* envia convites para outros usuários que possuam o perfil *Tutor*. Detalhes são mostrados no Quadro 3.

Quadro 3 – Caso de uso UC02

UC02 – Enviar convites	
Requisitos atendidos	RF2
Pré-condições	1. O usuário precisa ter perfil de <i>Paciente</i> .
Cenário principal	1. O <i>Paciente</i> escolhe para qual usuário deseja enviar o convite, podendo também escrever uma mensagem para ele. 2. O usuário toca no botão <i>Confirmar</i> para enviar o convite.
Exceção	Não existindo nenhum usuário com perfil <i>Tutor</i> cadastrado, o convite não será criado.
Pós-condição	O <i>Tutor</i> escolhido pelo <i>Paciente</i> receberá o convite junto com a mensagem digitada.

3.2.1.3 Alterar dados pessoais

Este caso de uso descreve como o usuário pode alterar seus dados pessoais. Detalhes são mostrados no Quadro 4.

Quadro 4 – Caso de uso UC03

UC03 – Alterar dados pessoais	
Requisitos atendidos	RF3
Pré-condições	Nenhuma.
Cenário principal	<ol style="list-style-type: none"> 1. O usuário toca em sua foto de perfil. 2. Após visualizar suas informações pessoais, ele seleciona o botão <i>Alterar</i>. 3. Será mostrada uma tela onde o usuário poderá alterar seu nome, telefone, local, função e símbolo.
Exceção	Não há.
Pós-condição	O usuário terá seus novos dados pessoais gravados.

3.2.1.4 Usar prancha

Este caso de uso descreve como o usuário interage com suas pranchas de comunicação. Detalhes são mostrados no Quadro 5.

Quadro 5 – Caso de uso UC04

UC04 – Usar prancha	
Requisitos atendidos	RF5
Pré-condições	1. O usuário deve possuir ao menos uma prancha de comunicação.
Cenário principal	<ol style="list-style-type: none"> 2. O usuário seleciona o plano no qual a prancha desejada está vinculada. 3. Seleciona a prancha. 4. Serão mostrados os símbolos da prancha selecionada, no formato 3x3, nesta tela o usuário poderá interagir com os símbolos.
Exceção	Não há.
Pós-condição	Será gravado no histórico de atividades que o usuário interagiu com a prancha selecionada.

3.2.1.5 Criar prancha

Este caso de uso descreve como o usuário cria uma nova prancha de comunicação. Detalhes são mostrados no Quadro 6.

Quadro 6 – Caso de uso UC05

UC05 – Criar prancha	
Requisitos atendidos	RF4
Pré-condições	Nenhuma.
Cenário principal	<ol style="list-style-type: none"> 1. O usuário toca no símbolo <i>Criar Prancha</i>. 2. É apresentada a ele uma prancha 3x3, sem símbolos. O usuário então insere os símbolos que desejar tocando no símbolo <i>Adicionar</i>. 3. Após adicionar os símbolos desejados, pressiona o botão <i>Confirmar</i>. 4. Escolhe um símbolo para a prancha. 5. Escolhe um plano já existente ou cria um novo plano, para vincular a prancha. 6. Se desejar criar um novo plano, deve selecionar um símbolo para este plano, assim como foi feito para a prancha no passo 4. Retorna então para o passo 5.
Exceção	Caso o usuário não selecionou nenhum símbolo para a prancha, no passo 3 o botão <i>Confirmar</i> não ficará habilitado, impossibilitando o usuário de criar a prancha.
Pós-condição	Será criada a nova prancha e ela ficará vinculada ao plano selecionado no passo 5.

3.2.1.6 Reutilizar prancha

Este caso de uso descreve como o usuário pode reutilizar uma prancha de qualquer usuário. Detalhes são mostrados no Quadro 7.

Quadro 7 – Caso de uso UC06

UC06 – Reutilizar prancha	
Requisitos atendidos	RF4
Pré-condições	1. Deve existir ao menos uma prancha criada na base.
Cenário principal	<ol style="list-style-type: none"> 1. O usuário toca no símbolo <i>Compartilhar Prancha</i>. 2. O usuário deve escolher reutilizar uma prancha sua ou de outro usuário. 3. Conforme o filtro acima, são apresentados todos os planos do próprio usuário ou dos outros usuários da base. O usuário seleciona um plano. 4. São apresentadas todas as pranchas do plano selecionado. O usuário seleciona uma prancha. 5. É apresentada a ele a prancha selecionada, sem os símbolos da categoria <i>Pessoa</i>. O usuário então altera os símbolos que desejar. 6. Após obter os símbolos desejados, pressiona o botão <i>Confirmar</i>. 7. Escolhe um símbolo para a prancha. 8. Escolhe um plano já existente ou cria um novo plano, para vincular a prancha. 9. Se desejar criar um novo plano, deve selecionar um símbolo para este plano, assim como foi feito para a prancha no passo 7. Retorna então para o passo 8.
Exceção	Caso a prancha não possuir nenhum símbolo, no passo 6 o botão <i>Confirmar</i> não ficará habilitado, impossibilitando o usuário de criar a nova prancha.
Pós-condição	Será criada a nova prancha e ela ficará vinculada ao plano selecionado no passo 8.

3.2.1.7 Visualizar observações

Este caso de uso descreve como o usuário visualiza suas observações. Detalhes são mostrados no Quadro 8.

Quadro 8 – Caso de uso UC07

UC07 – Visualizar observações	
Requisitos atendidos	RF6
Pré-condições	1. Deve existir ao menos uma observação criada para o usuário.
Cenário principal	<ol style="list-style-type: none"> 1. O usuário toca no símbolo <i>Observações</i>. 2. Seleciona uma observação através da sua descrição. 3. O texto da observação selecionada será mostrado logo abaixo da descrição.
Exceção	Não há.
Pós-condição	A observação é visualizada pelo usuário.

3.2.1.8 Criar observação

Este caso de uso descreve como o *Tutor* cria uma nova observação. Detalhes são mostrados no Quadro 9.

Quadro 9 – Caso de uso UC08

UC08 – Criar observação	
Requisitos atendidos	RF6
Pré-condições	1. O usuário deve possuir o perfil <i>Tutor</i> .
Cenário principal	1. O usuário toca no símbolo <i>Observações</i> . 2. Toca no símbolo <i>Adicionar Obs</i> . 3. Digita uma descrição para a nova observação, um texto e toca no botão <i>Confirmar</i> .
Exceção	Não há.
Pós-condição	A nova observação é criada.

3.2.1.9 Visualizar históricos

Este caso de uso descreve como o usuário visualiza seu histórico de uso de pranchas. Detalhes são mostrados no Quadro 10.

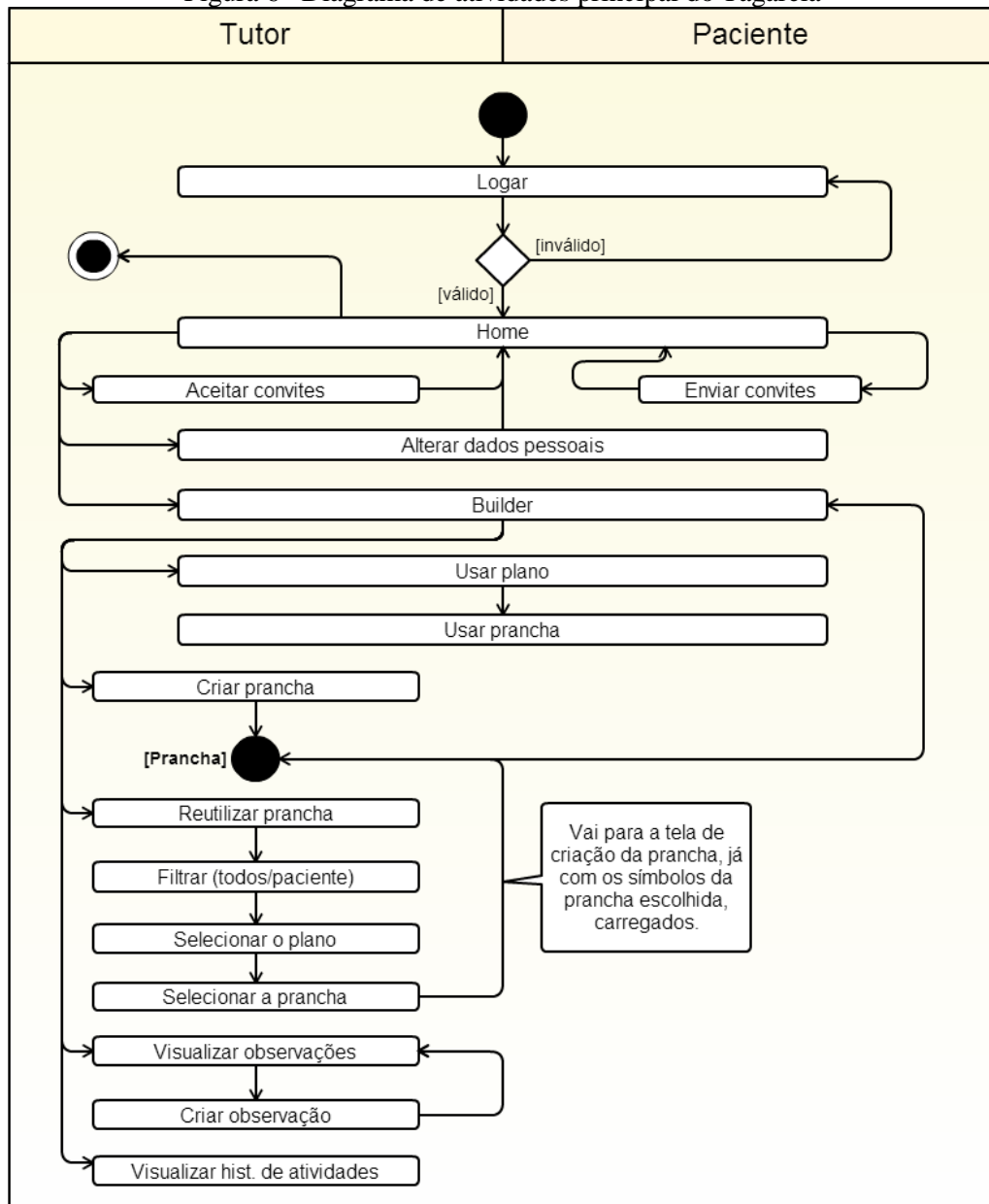
Quadro 10 – Caso de uso UC09

UC09 – Visualizar históricos	
Requisitos atendidos	RF7
Pré-condições	Nenhuma.
Cenário principal	1. O usuário toca no símbolo <i>Histórico Atividades</i> . 2. Será apresentada uma tela com todos os históricos de uso das pranchas de comunicação. No formato “[data - hora] - prancha utilizada”.
Exceção	Não há.
Pós-condição	O histórico das pranchas utilizadas é visualizado pelo usuário.

3.2.2 Diagrama de atividades

O diagrama de atividades demonstra de forma geral a utilização de todos os recursos do Tagarela pelos dois atores envolvidos. Este diagrama divide-se em duas partes, a primeira contém as atividades principais e a segunda parte a criação de uma prancha, que é uma funcionalidade importante do Tagarela.

Figura 6– Diagrama de atividades principal do Tagarela

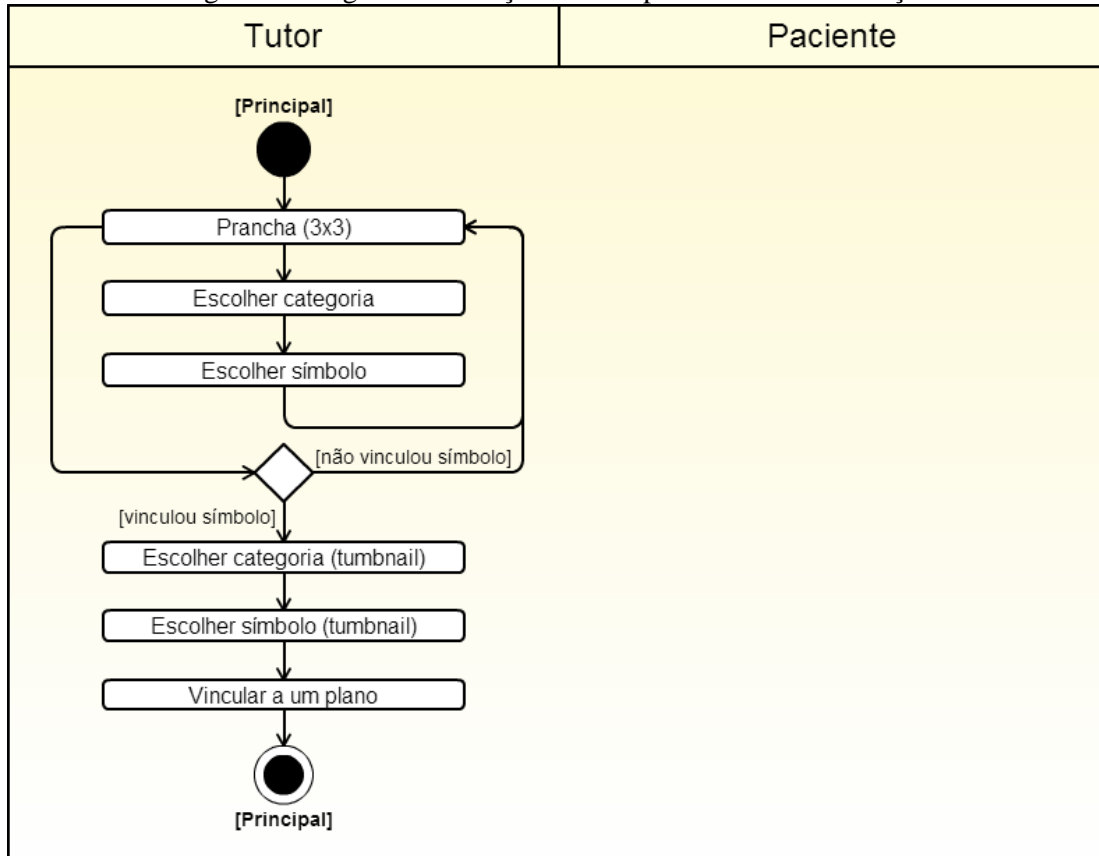


Conforme pode ser visto na Figura 6, as atividades no aplicativo iniciam-se com o login do usuário, caracterizado como tutor ou paciente. Caso o usuário seja um paciente, na tela principal (*home*) ele poderá enviar convites para outros usuários com perfil de tutor. Um tutor, por sua vez, poderá aceitar os convites que lhe foram enviados. Ambos os perfis podem alterar suas informações pessoais ainda nesta tela. Quando um convite é aceito, é criado um vínculo entre o especialista e o paciente. Este vínculo é chamado de construtor ou *builder*, dentro da aplicação, e irá aparecer na tela principal do usuário logado.

Após o usuário escolher o seu *builder* desejado, ele poderá utilizar as pranchas já criadas para ele, ou realizar as seguintes operações: visualizar e criar observações, visualizar o

histórico de atividades e criar ou reutilizar uma prancha. A atividade de criação de uma prancha de comunicação é demonstrada através da Figura 7.

Figura 7– Diagrama de criação de uma prancha de comunicação



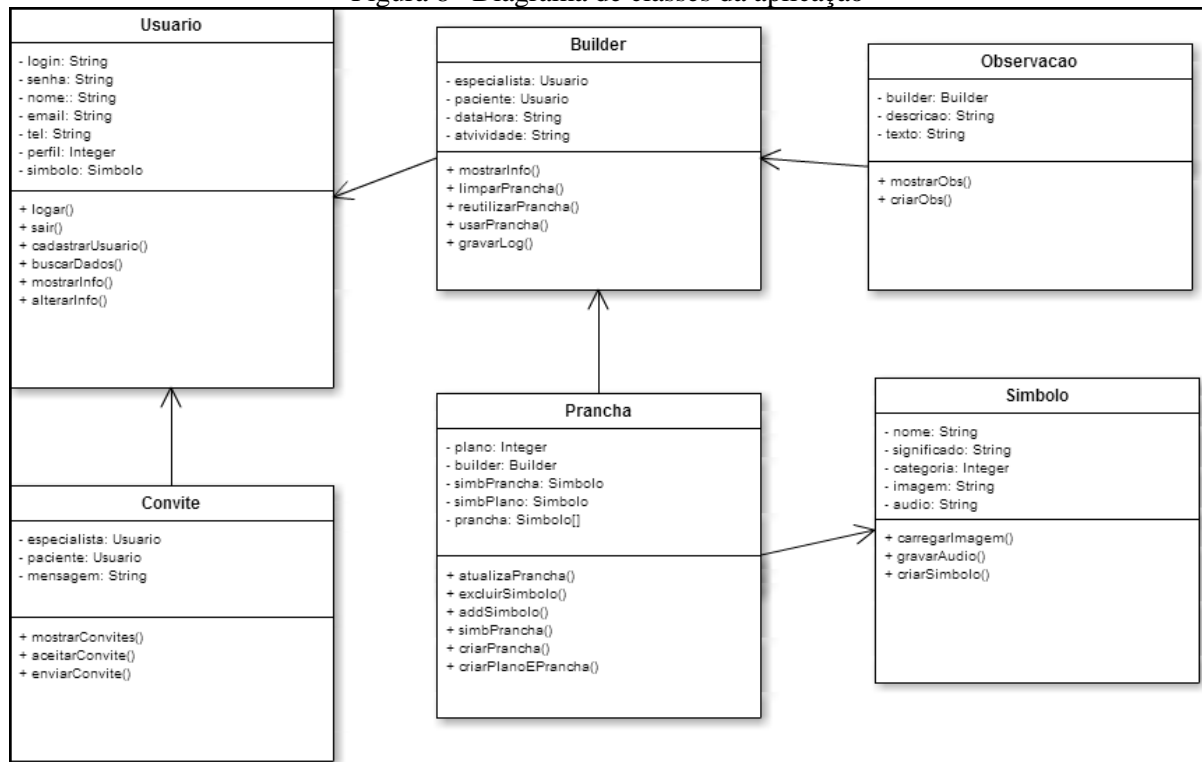
Ao criar uma prancha, o usuário deverá primeiramente escolher os símbolos desejados, que estarão separados entre quatro categorias de cores distintas, pessoas (amarelo), verbos (verde), substantivos (vermelho) e descritivos (azul). Caso optou-se pela reutilização de uma prancha já existente, os símbolos dela virão previamente carregados na nova prancha, com exceção daqueles que possuírem a categoria pessoas.

O usuário poderá então alterar os símbolos conforme desejar e confirmar a criação da prancha, escolhendo logo em seguida o plano no qual esta nova prancha ficará vinculada. Neste momento também poderá ser criado um novo plano para vincular a prancha.

3.2.3 Diagrama de classes

A Figura 8 apresenta o diagrama de classes que compõem o sistema e seus relacionamentos. Cada arquivo JavaScript da aplicação foi definido através de uma classe, separados por funcionalidades ou assuntos, com o intuito de organizar a estrutura da aplicação e atender os requisitos especificados de maneira mais clara. Logo após será exposto uma explicação mais detalhada de cada classe.

Figura 8– Diagrama de classes da aplicação



No diagrama a primeira classe definida é a classe *Usuario*, que tem como responsabilidade o gerenciamento dos usuários da aplicação, possuindo métodos para acessar e sair do sistema, bem como criar um novo usuário escolhendo entre o perfil de especialista e paciente, informação que será armazenada no atributo *perfil*. Além disso, através do método *alterarInfo* o usuário consegue modificar seus dados pessoais, como nome, email, telefone, função, etc.

A classe *Usuario* possui um relacionamento com a classe *Convite*, que através dos métodos *enviarConvites* e *aceitarConvites* cria um vínculo entre dois usuários com perfis distintos. Este vínculo será tratado pela classe *Builder*, que irá verificar qual o perfil do usuário logado na aplicação e definirá quem é o especialista e quem é o paciente, armazenando esta informação nos atributos *especialista* e *paciente*. O método *usarPrancha* permite ao usuário interagir com uma de suas pranchas de comunicação, sendo que cada vez que é chamado cria um registro no histórico de atividades através do método *gravarLog*, informando que o usuário utilizou determinada prancha.

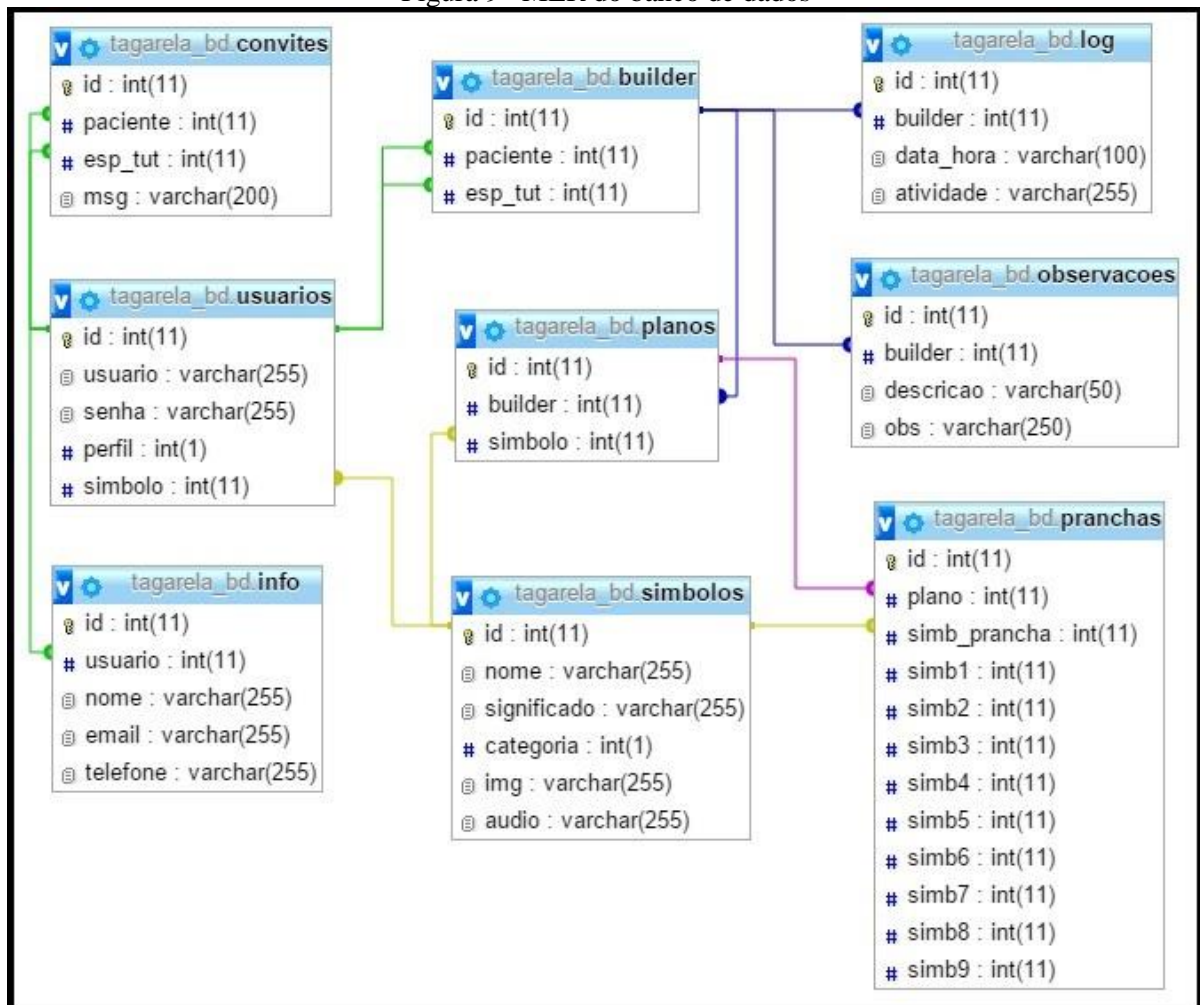
A classe *Builder* pode ser considerada a classe central da aplicação, pois além de ser responsável por tratar a interação dos usuários com suas pranchas, é através dela que são chamadas as demais classes responsáveis por efetuar as principais funcionalidades do sistema. Como a classe *Observação*, que possui os métodos para visualizar e criar observações. E também a classe *Prancha*, que fica responsável por criar e compartilhar pranchas de

comunicação, através dos métodos *criarPrancha* e *criarPlanoEPrancha*. Como se sabe, uma prancha deve possuir um ou mais símbolos, este relacionamento é determinado em conjunto com a classe *Simbolo*, que fará o gerenciamento dos símbolos cadastrados na aplicação.

3.2.4 Diagrama MER

As informações do sistema são persistidas em banco de dados, tanto no servidor quanto localmente na versão de dispositivos móveis, sendo que a estrutura das tabelas é a mesma nas duas versões. A Figura 9 apresenta o MER do banco de dados do aplicativo.

Figura 9– MER do banco de dados



3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

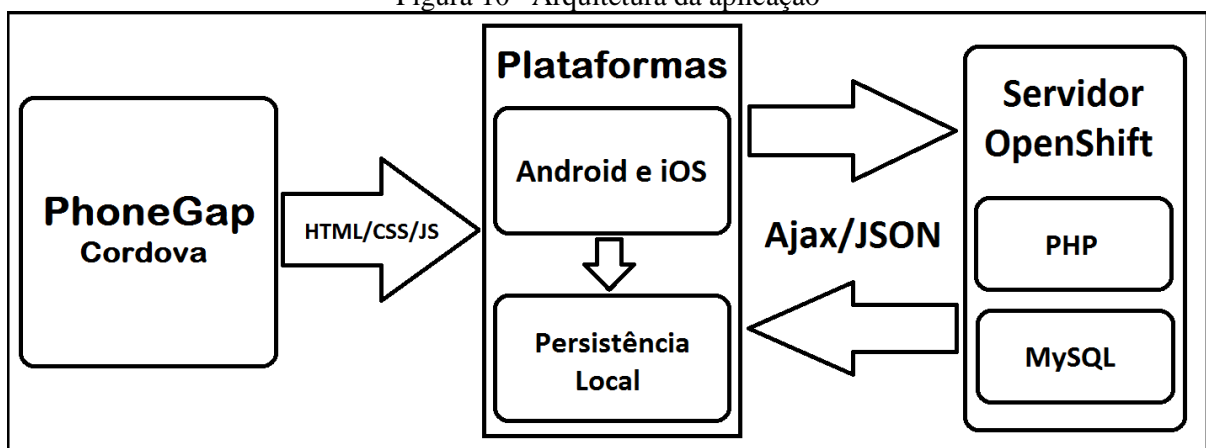
O desenvolvimento da aplicação foi realizado utilizando o *framework* PhoneGap. Desta forma, pôde-se desenvolver a mesma aplicação nas três plataformas desejadas (*web*, Android e iOS), utilizando o mesmo código fonte fazendo uso apenas de recursos de desenvolvimento *web*. O ambiente de desenvolvimento escolhido para a criação do trabalho foi a ferramenta Microsoft Expression Web 4, no qual foram utilizadas as linguagens HTML5, CSS 3.0, PHP 5.4 e JavaScript. Para o desenvolvimento no JavaScript foi usada a biblioteca jQuery v2.1.1 e para o CSS optou-se por fazer uso do Materialize, um *framework* que fornece um padrão de estilos com botões e campos *flat*, visando acelerar o desenvolvimento e também proporcionar ao usuário uma experiência unificada. Como ambiente de testes para a versão *web*, foram utilizados os navegadores Chrome versão 46.0.2490.80 m, Firefox versão 42.0 e Internet Explorer versão 11.0.9600.18053.

@@Especificar também em quais dispositivos foi testada a versão mobile da aplicação

3.3.1.1 Arquitetura proposta

Nesta seção será descrita a arquitetura proposta para implementar o trabalho. A aplicação foi concebida sob uma arquitetura cliente/servidor, considerando também o *framework* PhoneGap, conforme mostra a Figura 10.

Figura 10– Arquitetura da aplicação



Utilizando o *framework* PhoneGap é realizada a conversão e o *deploy* da aplicação *web* construída utilizando HTML, CSS e Javascript, para as plataformas Android e iOS. Além da persistência local para estas duas plataformas, as informações são gravadas em uma base de dados MySQL no servidor, onde estão também os *scripts php*, responsáveis por interagir com o código JavaScript, utilizando a metodologia Ajax para resgatar estas informações. Foi escolhido utilizar o servidor OpenShift, devido a sua robustez e facilidade de uso.

3.3.1.2 Utilização do PhoneGap

O PhoneGap é uma ferramenta de desenvolvimento híbrido simples e vantajosa, porém é necessário efetuar algumas configurações no seu ambiente de trabalho antes de começar a utilizá-lo. Primeiramente deve ser decidido com quais plataformas deseja-se trabalhar, pois apesar das aplicações serem desenvolvidas utilizando recursos *web*, é necessário obter os *SDKs* (*Software Development Kit*) específicos para cada plataforma. O Tagarela é disponibilizado nas plataformas Android e iOS, portanto para efetuar o *deploy* nas duas versões foi utilizado o sistema operacional Mac OS X, com o Android SDK instalado e o aplicativo Xcode em conjunto com a ferramenta Xcode Command Line Tools.

Após isso foi instalada a ferramenta *npm* (*NodeJS package manager*), que é responsável por fazer o *download* e a instalação do PhoneGap. Para isso, é executado o comando `sudo npm install -g phonegap`, dentro do aplicativo *Terminal*, desta forma será efetuada a instalação do PhoneGap CLI (*Command Line Tools*) em seu ambiente de trabalho. A partir deste momento o PhoneGap está pronto para uso, todas as operações são realizadas através de comandos via *Terminal*, conforme mostra a Figura 11.

Figura 11– Comandos do PhoneGap CLI

```
1  $ phonegap create tagarela com.tagarela Tagarela
2
3  $ phonegap platform add android
4  $ phonegap platform add ios
5
6  $ phonegap plugin add org.apache.cordova.device
7  $ phonegap plugin add org.apache.cordova.camera
8  $ phonegap plugin add org.apache.cordova.media-capture
9  $ phonegap plugin add org.apache.cordova.media
10 $ phonegap plugin add org.apache.cordova.file
11
12 $ phonegap build android
13 $ phonegap build ios
14
15 $ phonegap emulate android
16 $ phonegap emulate ios
17 $ phonegap run android
18 $ phonegap run ios
```

Primeiramente executou-se o comando `phonegap create` passando três parâmetros, *tagarela*, *com.tagarela* e *Tagarela*. Este comando cria um novo projeto no diretório corrente, dentro da pasta *tagarela*, com identificador em estilo domínio reverso *com.tagarela*, de um aplicativo chamado *Tagarela*. Após isso, são copiados os arquivos da aplicação *web* criada para esta nova pasta. O aplicativo não pode ser executado ou simulado nos dispositivos desejados até que sejam adicionadas as plataformas. Isto é realizado através do comando

phonegap platform add, onde se deve passar por parâmetro as plataformas desejadas, neste caso Android e iOS.

Para executar o aplicativo existem dois passos necessários, que são mostrados da linha 12 até a linha 18. O primeiro comando, *phonegap build*, compila a aplicação para a plataforma desejada, gerando os arquivos específicos de cada plataforma dentro do subdiretório *platforms* do projeto. Após executar este comando, pode-se então rodar o aplicativo, tanto em um simulador quanto em um dispositivo móvel conectado ao computador, respectivamente através dos comandos *phonegap emulate* e *phonegap run*.

Entre as linhas 6 e 10 são adicionados vários *plug-ins* que serão utilizados na aplicação. Os *plugins camera*, *media-capture* e *media*, por exemplo, são necessários para acessar a câmera do dispositivo e também reproduzir e gravar áudio. Já o *plugin file* é responsável por ler, escrever e navegar no sistema de arquivos do dispositivo.

3.3.1.3 Códigos da criação e reutilização de pranchas

Nesta seção será detalhada a implementação das duas principais funcionalidades do Tagarela, que são a criação e a reutilização de uma prancha de comunicação. Estas duas funcionalidades compartilham os mesmos métodos, a diferença é que quando é escolhido reutilizar uma prancha, primeiramente deve ser efetuada a seleção da prancha que servirá como modelo para esta nova prancha. O Quadro 11 mostra o método *reutilizarPrancha* da classe *Builder*, implementado em JavaScript, que é responsável por realizar este tratamento.

Quadro 11 – Método *reutilizarPrancha*

```

138 $(".img-prancha").click( function reutilizarPrancha() {
139     var alt = $(this).attr("alt");
140     prancha = Number(alt);
141
142     // Busca os simbolos da prancha selecionada
143     var dados = { "prancha" : prancha};
144     $.ajax({
145         type      : "post",
146         url       : "http://tagarela-afwippel.rhcloud.com/scripts/
147                     buscar-prancha.php",
148         data      : dados,
149         dataType  : "json",
150         success   : function(ret) {
151             $("body").removeClass("loading");
152             if (ret.erro) {
153                 alert(ret.msg);
154             }
155             else {
156                 localStorage.simb1 = "../img/"+ret.simbolosImg[0];
157                 localStorage.simb2 = "../img/"+ret.simbolosImg[1];
158                 localStorage.simb3 = "../img/"+ret.simbolosImg[2];
159                 localStorage.simb4 = "../img/"+ret.simbolosImg[3];
160                 localStorage.simb5 = "../img/"+ret.simbolosImg[4];
161                 localStorage.simb6 = "../img/"+ret.simbolosImg[5];
162                 localStorage.simb7 = "../img/"+ret.simbolosImg[6];
163                 localStorage.simb8 = "../img/"+ret.simbolosImg[7];
164                 localStorage.simb9 = "../img/"+ret.simbolosImg[8];
165                 localStorage.idSimb1 = ret.simbolosId[0];
166                 localStorage.idSimb2 = ret.simbolosId[1];
167                 localStorage.idSimb3 = ret.simbolosId[2];
168                 localStorage.idSimb4 = ret.simbolosId[3];
169                 localStorage.idSimb5 = ret.simbolosId[4];
170                 localStorage.idSimb6 = ret.simbolosId[5];
171                 localStorage.idSimb7 = ret.simbolosId[6];
172                 localStorage.idSimb8 = ret.simbolosId[7];
173                 localStorage.idSimb9 = ret.simbolosId[8];
174                 localStorage.idSimbPrancha = 0;
175                 localStorage.idPlanoPrancha = 0;
176
177                 location.href = "../prancha/prancha.html";
178             }
179         },
180         error      : function(ret) {
181             $("body").removeClass("loading");
182             alert("Erro no servidor (TIMEOUT)!");
183         },
184         beforeSend: function() {
185             $("body").addClass("loading");
186         },
187         complete: function() {
188             $("body").removeClass("loading");
189         }
190     });
191 });

```

No código pode-se observar que o método *reutilizarPrancha* está vinculado ao evento de clique na imagem da prancha, significando que este método será chamado quando o usuário selecionar a prancha desejada. A partir da linha 144 é então realizada a busca dos

símbolos desta prancha no banco de dados. Foi escolhido utilizar a metodologia *Ajax* (*Asynchronous Javascript and XML*) para realizar o acesso ao banco de dados da aplicação, por ser uma metodologia simples e eficaz que atende bem a este tipo de requisição. A *URL* passada por parâmetro condiz com o script PHP que realizará este acesso ao banco, este script está localizado dentro do servidor OpenShift. Após a requisição retornar com sucesso, as variáveis que armazenam os símbolos da nova prancha em memória são inicializadas com os símbolos da prancha modelo e a página HTML que permite o usuário alterar estes símbolos é chamada, seguindo o processo normal de criação de uma prancha.

Após o usuário selecionar todos os símbolos desejados para sua nova prancha, bem como o plano no qual ela será vinculada, o método *criarPrancha* da classe *Prancha* é chamado. O Quadro 12 apresenta a implementação deste método e posteriormente uma explicação de cada trecho de código.

Quadro 12 – Método *criarPrancha*

```

17  $(".img-plano").click( function criarPrancha() {
18      var alt = $(this).attr("alt");
19      plano = Number(alt);
20
21      // Grava nova prancha na base com os dados que foram selecionados
22      var dados = {
23          "plano" : plano,
24          "simbPrancha" : localStorage.idSimbPrancha,
25          "simb1" : localStorage.idSimb1,
26          "simb2" : localStorage.idSimb2,
27          "simb3" : localStorage.idSimb3,
28          "simb4" : localStorage.idSimb4,
29          "simb5" : localStorage.idSimb5,
30          "simb6" : localStorage.idSimb6,
31          "simb7" : localStorage.idSimb7,
32          "simb8" : localStorage.idSimb8,
33          "simb9" : localStorage.idSimb9,
34      };
35      $.ajax({
36          type      : "post",
37          url       : "http://tagarela-afwippel.rhcloud.com/scripts/
38                      gravar-prancha.php",
39          data      : dados,
40          dataType  : "json",
41          success   : function(ret) {
42              $(".body").removeClass("loading");
43              if (ret.erro) {
44                  alert(ret.msg);
45              }
46              else {
47                  location.href = "../builder.html";
48              }
49          },
50          error      : function(ret) {
51              $(".body").removeClass("loading");
52              alert("Erro no servidor (TIMEOUT)!");
53          },
54          beforeSend: function() {
55              $(".body").addClass("loading");
56          },
57          complete: function() {
58              $(".body").removeClass("loading");
59          }
60      });
61  });

```

Na linha 22 é criada uma variável chamada *dados* que armazenará as informações que serão enviadas ao *script gravar-prancha.php*, no formato *Array JSON (JavaScript Object Notation)*. Estas informações são variáveis que dizem respeito aos símbolos e ao plano que o próprio usuário selecionou a medida que realizava o processo de criação da prancha, nas telas anteriores. O *script* que realiza a gravação da prancha no banco de dados é demonstrado no Quadro 13, onde é feita uma operação *INSERT* na tabela *pranchas* com as informações que foram enviadas pela requisição AJAX, no método *criarPrancha*.

Quadro 13 – Script de gravação da prancha na base de dados

```

1  <?php
2      // IN
3      $plano = $_POST["plano"];
4      $simbPrancha = $_POST["simbPrancha"];
5      $simb1 = $_POST["simb1"];
6      $simb2 = $_POST["simb2"];
7      $simb3 = $_POST["simb3"];
8      $simb4 = $_POST["simb4"];
9      $simb5 = $_POST["simb5"];
10     $simb6 = $_POST["simb6"];
11     $simb7 = $_POST["simb7"];
12     $simb8 = $_POST["simb8"];
13     $simb9 = $_POST["simb9"];
14     // OUT
15     $erro = false;
16     $msg = "";
17
18     $con = mysqli_connect("127.6.181.2","adminHh2zviV",
19                           "q7DGYPAINIsG","tagarela_bd","3306");
20     if (!$con) {
21         $erro = true;
22         $msg = "Não foi possível conectar no banco de dados! Erro:"
23             .mysqli_connect_error();
24     }
25     else {
26         $query = "INSERT INTO pranchas (plano, simb_prancha, simb1,
27             simb2, simb3, simb4, simb5, simb6, simb7, simb8,
28             simb9) " . "VALUES ($plano, '$simbPrancha',
29             '$simb1', '$simb2', '$simb3', '$simb4', '$simb5',
30             '$simb6', '$simb7', '$simb8', '$simb9')";
31         mysqli_query($con,$query);
32     }
33     mysqli_close($con);
34
35     $ret = array();
36     $ret["erro"] = $erro;
37     $ret["msg"] = $msg;
38
39     sleep(1);
40     echo json_encode($ret);
41 ?>

```

3.3.1.4 Interação com as pranchas criadas

A utilização das pranchas pelos usuários é também implementada dentro da classe *Builder*, no método *usarPrancha*. No Quadro 14 é apresentado um trecho de código com a implementação deste método e também do método *gravarLog*.

Quadro 14 – Métodos *usarPrancha* e *gravarLog*

```

52 function usarPrancha() {
53     var audioElement = document.createElement("audio");
54     $.get();
55
56     // Verifica quem é o tutor e quem é o paciente
57     var tutor = localStorage.idUser;
58     var paciente = localStorage.idBuilder;
59     if (localStorage.perfil == 3) {
60         tutor = localStorage.idBuilder;
61         paciente = localStorage.idUser;
62     }
63     // Data-hora atual e atividade
64     var hoje = new Date();
65     var dia = hoje.getDate();
66     var mes = hoje.getMonth()+1;
67     var ano = hoje.getFullYear();
68     var hora = hoje.getHours();
69     var min = hoje.getMinutes();
70     var seg = hoje.getSeconds();
71     var data = dia+'/'+mes+'/'+ano;
72     var hora = hora+':'+min+':'+seg;
73     var dataHora = data+ " - " +hora;
74     var atv = "Usou a prancha "+localStorage.idPrancha;
75
76     // Busca os simbolos da prancha e grava o log de uso
77     var dados = {
78         "idBuilder" : paciente,
79         "espTut" : tutor,
80         "dataHora" : dataHora,
81         "atv" : atv,
82         "idPrancha" : localStorage.idPrancha
83     };
84     $.ajax({
85         type      : "post",
86         url       : "http://tagarela-afwippel.rhcloud.com/scripts/
87                     usar-prancha.php",
88         data      : dados,
89         dataType  : "json",
90         success   : function gravarLog(ret) {
91             $("body").removeClass("loading");
92             if (ret.erro) {
93                 alert(ret.msg);
94             }
95             else {
96                 for (var i = 0; i < 9; i++) {
97                     $(".simbolos").append("<li style='display:inline-
98                                         block;'>"
99
100                     +"<img src='img/'+ret.simbolosImg[i]+' title='"
101                     +ret.simbolosAudio[i]+' alt='"+ret.simbolosId[i]
102                     +' class='img-simbolo' style='margin:20px'
103                     height='150' width='150'/>"
104                     +"</li>");
105                 }
106             },
107             error   : function(ret) {
108                 $("body").removeClass("loading");
109                 alert("Erro no servidor (TIMEOUT)!");
110             },
111             beforeSend: function() {

```

```

112     $("body").addClass("loading");
113 },
114     complete: function() {
115         $("body").removeClass("loading");
116
117         $(".img-simbolo").click(function() {
118             var src = "audio/"+$(this).attr("title");
119             audioElement.setAttribute("src",src);
120             audioElement.play();
121         });
122     }
123 });
});

```

Primeiramente na linha 53 é criado um elemento de áudio, chamado *audioElement*, para reproduzir o som vinculado ao símbolo quando o usuário pressioná-lo, operação que é realizada entre as linhas 117 e 121. Para que esta funcionalidade seja também executada nos dispositivos móveis, é necessária a instalação do *plugin media*, apresentado no capítulo 3.3.1.2, pois sem ele o PhoneGap não consegue efetuar corretamente a conversão deste elemento JavaScript para as plataformas desejadas.

Ao carregar os símbolos da prancha e apresentá-los para o usuário no formato 3x3, é realizada a gravação de um registro no histórico de atividades, informando que o usuário utilizou esta prancha de comunicação. Isto é realizado através do método *gravarLog*, na linha 90. Este registro é gravado com a data atual do sistema e a prancha que o usuário utilizou, informações que são montadas entre as linhas 64 e 74.

3.3.2 Operacionalidade da implementação

No Tagarela, a principal atividade do usuário é a interação com suas pranchas de comunicação. Porém, antes disso é necessário citar algumas etapas necessárias para que esta funcionalidade seja realizada, como a criação de usuários, o envio e a aceitação de convites e a criação de uma prancha de comunicação. Além destas, ainda existem outras funcionalidades secundárias, como a reutilização de uma prancha já existente, a criação de observações e a visualização do histórico de atividades do paciente. Esta seção apresenta de forma resumida a operacionalidade de cada uma das etapas supracitadas.

3.3.2.1 Criação de usuários

A tela inicial da aplicação possibilita o usuário realizar o *login* no Tagarela ou criar um novo usuário, conforme Figura 12a. Ao pressionar o símbolo *Adicionar Usuário*, será apresentada a tela de criação de usuários, onde deverá ser informado o usuário, a senha e o perfil desejado para o novo usuário, que poderá ser tutor ou paciente.

Figura 12– Tela inicial e criação de usuário

(a) Tela de login

(b) Criação de usuário

3.3.2.2 Vínculo de usuários

Após criar os tutores e pacientes deve-se criar um vínculo entre eles, para concretizar o papel destes dois perfis dentro da aplicação. Um tutor, por exemplo, não consegue exercer seu papel se não estiver vinculado a um paciente, assim como um paciente também precisa estar vinculado a pelo menos um tutor. Este vínculo dentro da aplicação forma um *builder*. Sendo assim, todos os pacientes devem possuir um *builder* com o perfil tutor e todos os tutores devem possuir um *builder* com o perfil paciente, para que possam utilizar as funcionalidades da aplicação. A criação destes vínculos é feita através do envio e aceitação de convites, onde os pacientes podem apenas enviar convites para tutores e os tutores podem apenas aceitar os convites que lhe foram enviados.

A Figura 13 abaixo mostra a tela de envio de convites.

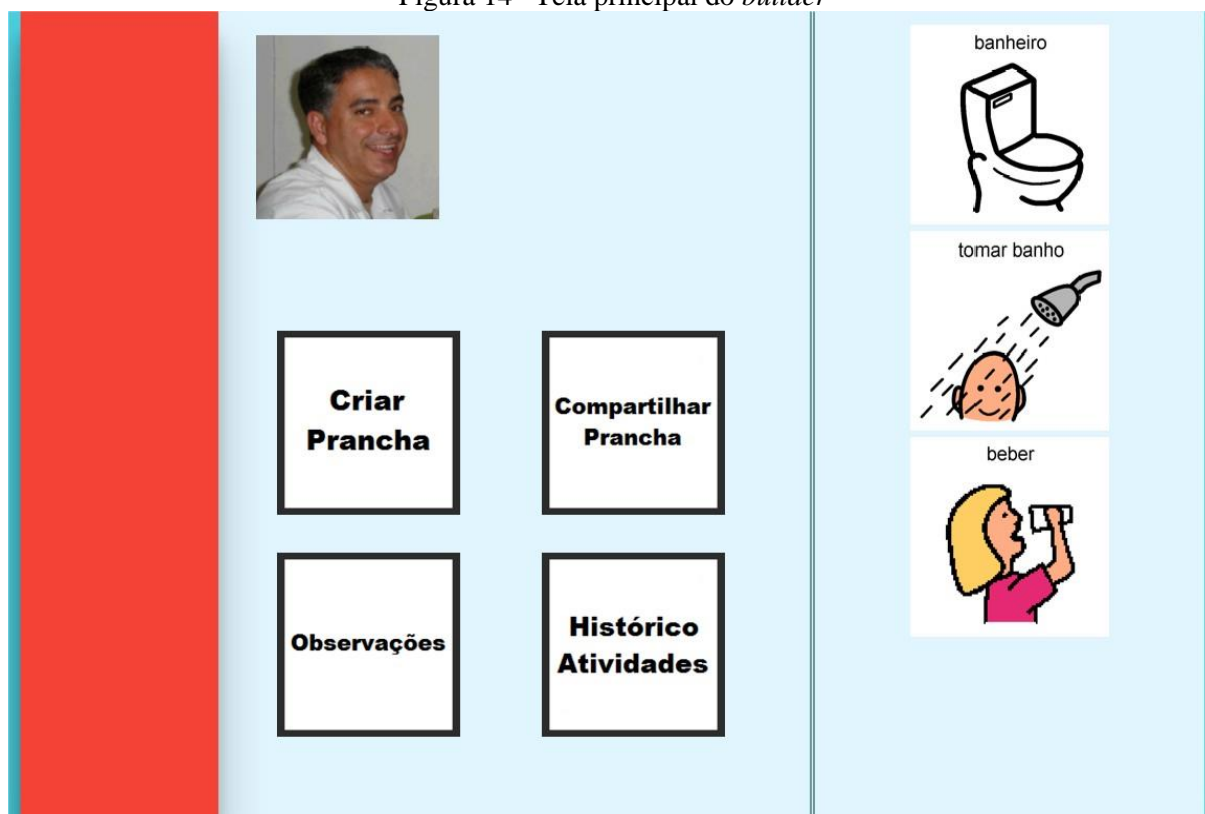
Figura 13– Envio e aceitação de convites

Ao criar um paciente, já é criado automaticamente um *builder* para ele. Foi criado este facilitador para os usuários que não necessitam estar vinculados a algum tutor e também não queiram passar por este processo de vínculo de usuários para conseguirem utilizar todas as funcionalidades da aplicação.

3.3.2.3 Criação de prancha de comunicação

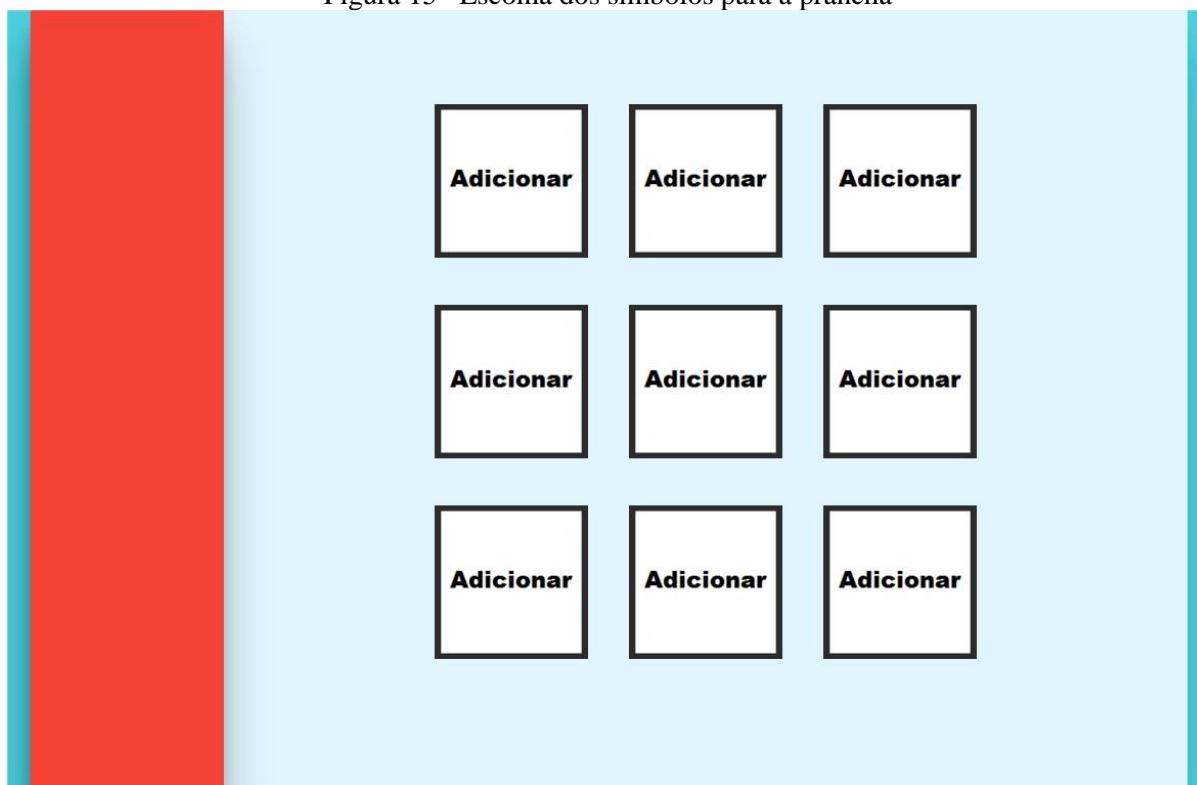
O processo de criação de uma prancha de comunicação, assim como as demais funcionalidades da aplicação, é realizado a partir da tela do *builder*, conforme Figura 14. Nesta tela é possível também acessar, além da criação de pranchas, todas as funcionalidades que serão demonstradas nos próximos capítulos. Além disso, no lado direito são listados todos os planos deste *builder*.

Figura 14– Tela principal do *builder*



Ao pressionar o símbolo *Criar Prancha*, será mostrado ao usuário uma prancha 3x3 vazia, conforme a Figura 15. Ele então deve selecionar dentro de uma categoria, quais símbolos farão parte desta nova prancha pressionando o símbolo *Adicionar*.

Figura 15– Escolha dos símbolos para a prancha



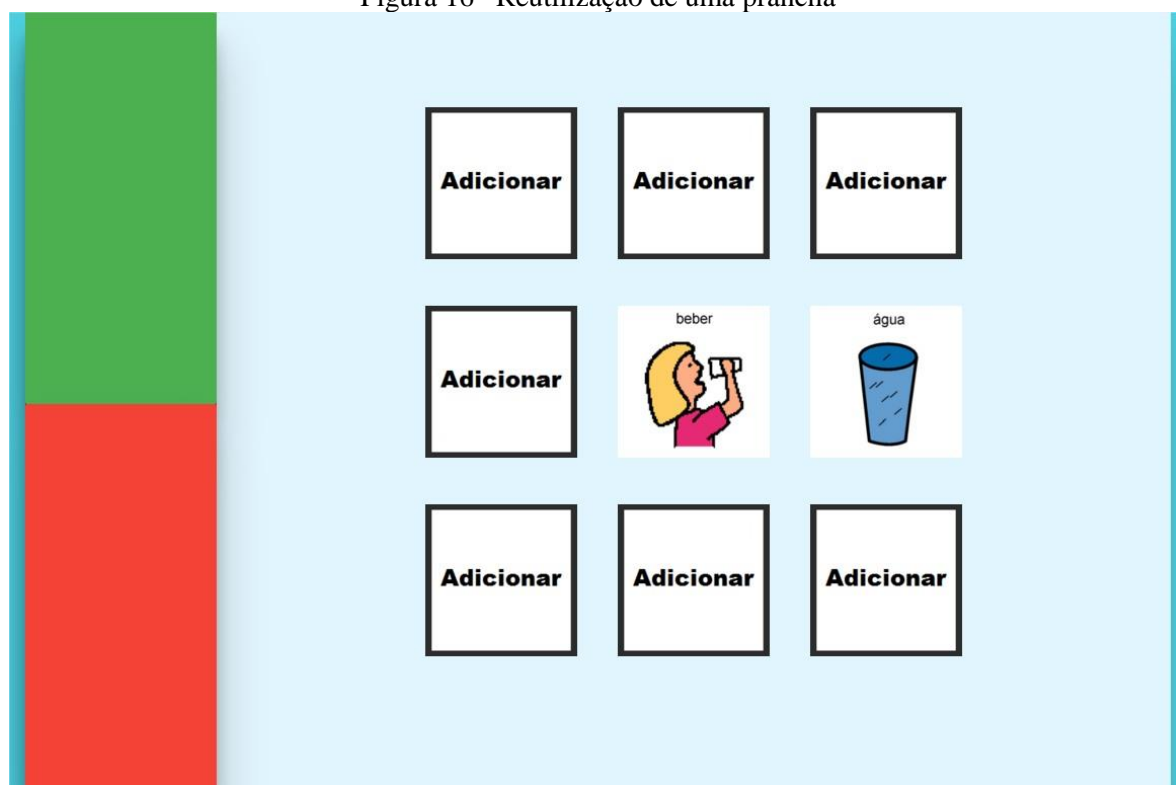
Após escolher os símbolos desejados o usuário deverá vincular a prancha a um plano existente, ou então criar um plano para incluir esta nova prancha de comunicação. A aplicação então retornará para a tela principal do *builder*, onde a nova prancha já estará disponível para ser utilizada.

3.3.2.4 Compartilhamento de pranchas

Além de criar uma prancha, há também a possibilidade de ser reutilizada uma prancha já existente. Este processo é similar a criação de uma prancha, tendo como única diferença a escolha da prancha que será utilizada como modelo. Neste caso, na tela onde seria mostrada uma prancha 3x3 vazia, conforme Figura 15, serão previamente carregados os símbolos da prancha escolhida como modelo.

A seleção da prancha modelo pode ser feita de duas formas, filtrando apenas as pranchas do próprio paciente selecionado, ou todas as pranchas dos demais pacientes cadastrados. Caso for escolhida esta segunda opção, não serão carregados os símbolos da categoria *Pessoa*, pois estes são considerados confidenciais para cada paciente. A Figura 16 mostra uma prancha, já criada para outro paciente, sendo reutilizada.

Figura 16– Reutilização de uma prancha

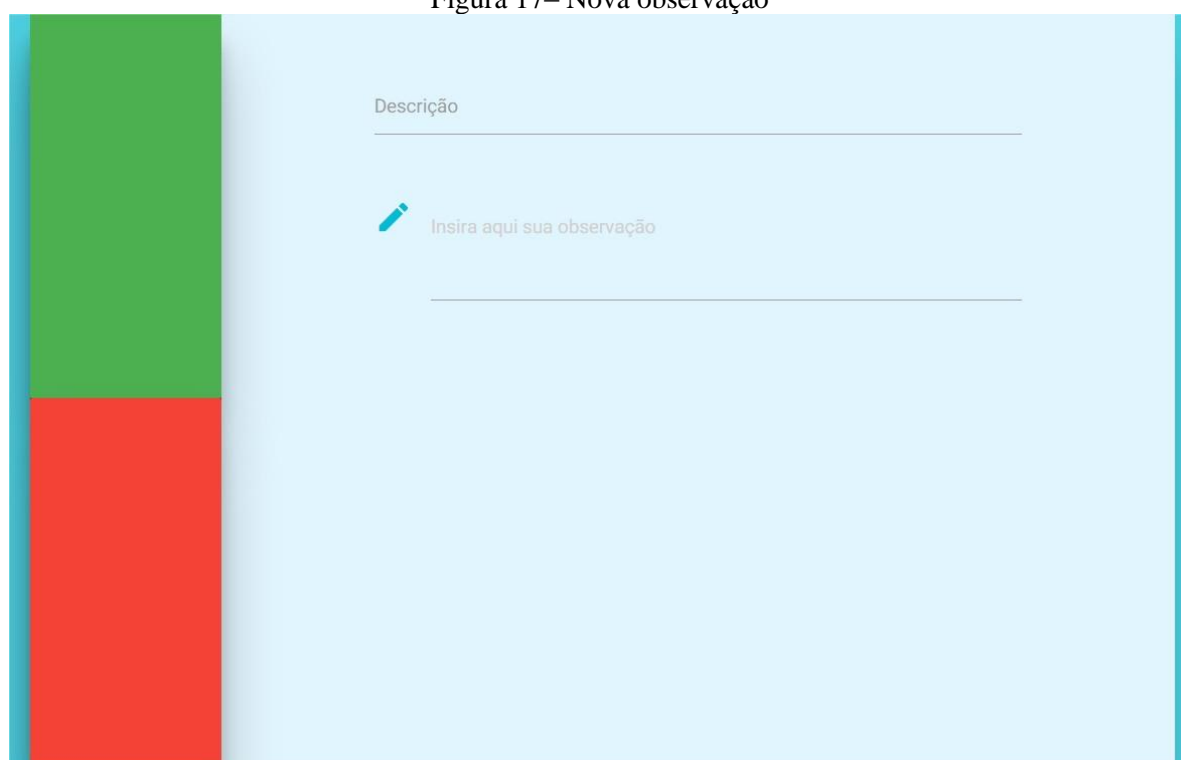


Após a escolha da prancha modelo o usuário poderá alterar os símbolos pré-carregados, ou apenas confirmar a criação da prancha e selecionar um plano para vinculá-la. Segue o mesmo processo de criação de uma prancha.

3.3.2.5 Criação de observações

A criação de observações pode ser realizada apenas por usuários com o perfil tutor, pois são direcionadas especificamente aos pacientes. Esta função também é acessada pela tela principal do *builder*, pressionando o símbolo *Observações*. Será apresentada uma tela onde poderão ser visualizadas as observações já cadastradas, ou através do símbolo *Adicionar Obs.* será possível criar uma nova observação. Conforme mostra a Figura 17, a observação possui apenas duas informações, uma descrição e o texto da observação. Ela ficará registrada para o paciente que foi selecionado como *builder*.

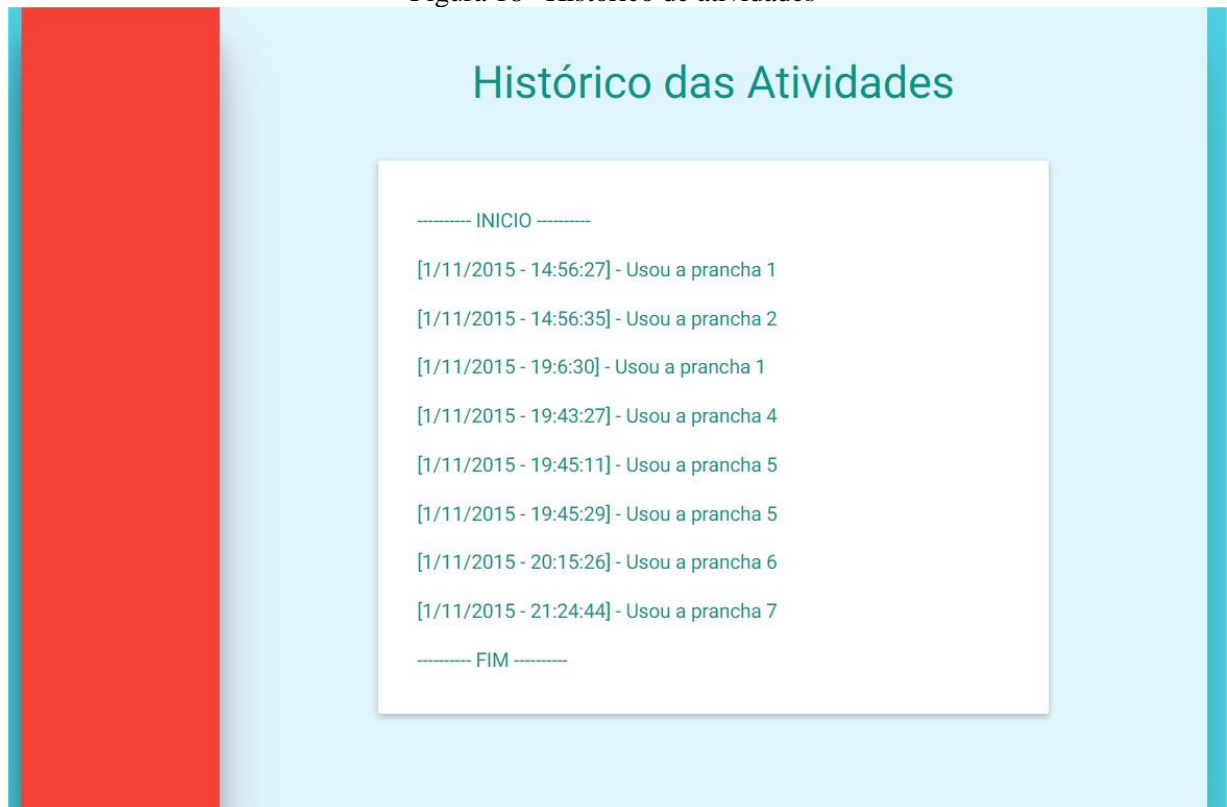
Figura 17– Nova observação

The image shows a user interface for adding a new observation. On the left, there is a vertical sidebar with a green rectangular button at the top and a red rectangular button below it. The main area has a light blue background. At the top of this area, the word "Descrição" is followed by a horizontal line. Below this, there is a blue pencil icon and the text "Insira aqui sua observação", which is also followed by a horizontal line.

3.3.2.6 Visualização dos históricos

Toda vez que um paciente utiliza uma prancha de comunicação, é gravada esta interação no seu histórico de atividades. Posteriormente, apenas seus tutores poderão consultar este histórico para fins didáticos. Esta consulta é realizada através do símbolo *Histórico Atividades*, na tela principal do *builder*, conforme a Figura 18.

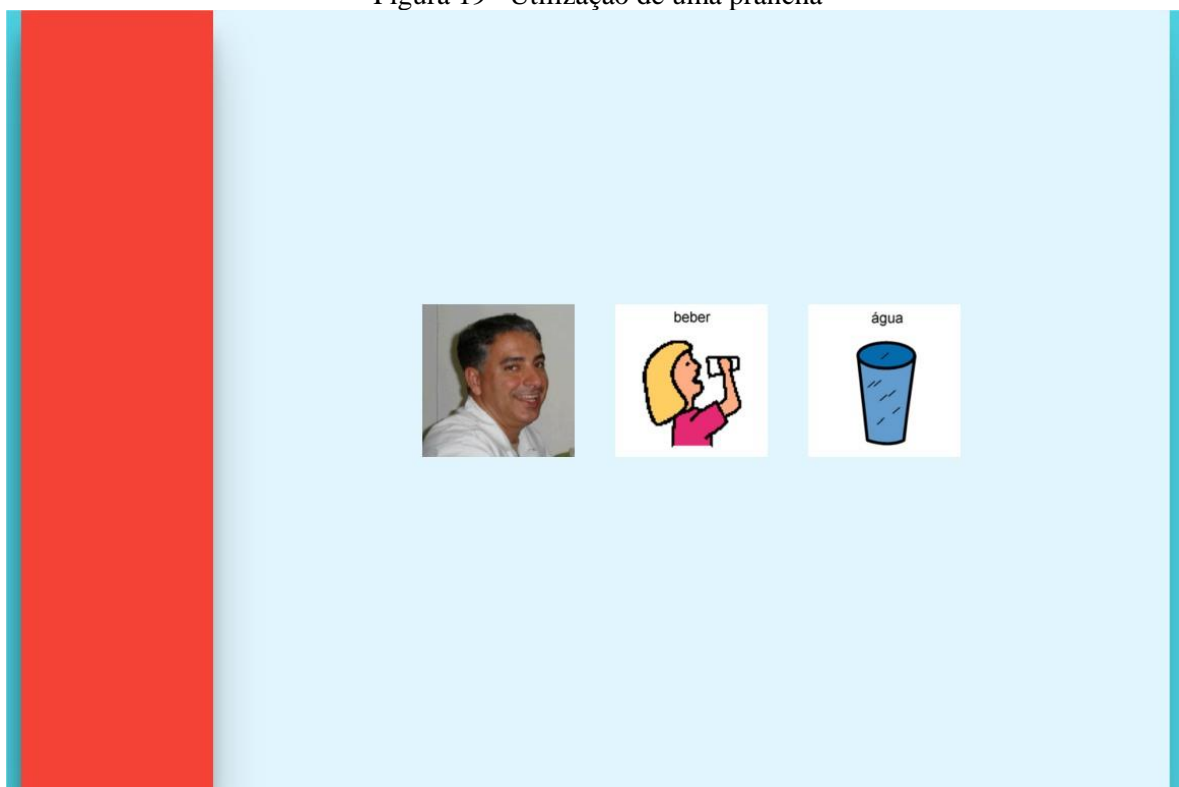
Figura 18– Histórico de atividades



3.3.2.7 Utilização das pranchas

A principal atividade do Tagarela, a interação com as pranchas de comunicação, é realizada a partir da tela principal do *builder*. Primeiramente deve-se selecionar um plano de atividades, objeto que agrupa pranchas de comunicação, para então selecionar a prancha desejada. Será apresentada uma tela para o usuário, com os símbolos da prancha 3x3, conforme a Figura 19.

Figura 19– Utilização de uma prancha



Ao pressionar um símbolo, o áudio vinculado a ele será reproduzido. As posições da prancha que não possuírem símbolos ficarão em branco e não reproduzirão nenhum áudio ao serem pressionadas.

3.4 RESULTADOS E DISCUSSÕES @@ FAZER

[Apresentar os resultados obtidos e confrontar com os trabalhos correlatos apresentados na fundamentação teórica. Apresentar, preferencialmente em forma de gráficos ou tabelas, os testes e avaliações realizadas, fazendo comentários sobre os mesmos.]

4 CONCLUSÕES @@ FAZER

[As conclusões devem refletir os principais resultados alcançados, realizando uma avaliação em relação aos objetivos previamente formulados. Deve-se deixar claro se os objetivos foram atendidos, se as ferramentas utilizadas foram adequadas e quais as principais contribuições do trabalho para o seu grupo de usuários ou para o desenvolvimento científico/tecnológico.]

[Deve-se também incluir aqui as principais vantagens do seu trabalho e limitações.]

4.1 EXTENSÕES @@ FAZER

[Sugestões para trabalhos futuros.]

REFERÊNCIAS @@ IR ATUALIZANDO A MEDIDA QUE SÃO REFERENCIADOS NA MONOGRAFIA

- ADOBE. PhoneGap. San Jose, 2014. Disponível em: <<http://phonegap.com>>. Acesso em: 12 set. 2014.
- CIVIAM. Sono Flex - Comunicação Alternativa para Ipad e Tablet Android. São Paulo, 2014. Disponível em: <<http://www.civiam.com.br/civiam/index.php/necessidadesespeciais/sono-flex-comunicacao-alternativa-para-ipad-tablet-android.html#>>. Acesso em: 06 set. 2014.
- FABENI, Alan Filipe C. Tagarela: Aplicativo para Comunicação Alternativa no iOS. 2012. 107 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- FOTON. Evolução da Comunicação Humana e dos Meios de Comunicação. Campinas, 2008. Disponível em: <http://www.foton.com.br/empresa.php?id=informacoes&sid=dados_administrativos>. Acesso em: 12 set. 2014.
- GONÇALVES, Maria de J. Comunicação alternativa na fonoaudiologia: uma área em expansão. São Paulo, 2008. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1516-18462008000300002>. Acesso em: 12 set. 2014.
- HELPTALK. HelpTalk. Guimarães, 2014. Disponível em <<http://www.helptalk.mobi/pt>>. Acesso em: 11 set. 2014.
- MARCO, Darlan D. de. Tagarela: Aplicativo de Comunicação Alternativa na Plataforma Android. 2014. 94 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- MATERIALIZE. Sobre o Materialize. Pittsburgh, 2015. Disponível em: <<http://materializecss.com/about.html>>. Acesso em: 09 nov. 2015.
- PELOSI, Miryam. Recursos na CAA. Rio de Janeiro, 2011. Disponível em: <<http://www.comunicacaoalternativa.com.br/recursos-na-cao>>. Acesso em: 12 set. 2014.
- SARTORETTO, Mara L.; BERSCH, Rita. O que é comunicação alternativa? Porto Alegre, 2007. Disponível em: <<http://www.assistiva.com.br/ca.html>>. Acesso em: 12 set. 2014.
- TAGARELA. Tagarela: rede social de comunicação alternativa. Blumenau, 2014. Disponível em <<http://gcg.inf.furb.br/tagarela>>. Acesso em: 11 set. 2014.
- TEIXEIRA, Milene. A importância da fala. Primavera do Leste, 2013. Disponível em: <<http://www.jornalodiario.com.br/TNX/conteudo.php?sid=204&cid=30649>>. Acesso em: 12 set. 2014.

APÊNDICE A – Relação dos formatos das apresentações dos trabalhos @@ Ir atualizando a medida que são referenciados na monografia

[Elemento opcional. **Apêndices são textos elaborados pelo autor** a fim de complementar sua argumentação. Os apêndices são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deverá haver no mínimo uma referência no texto anterior para cada apêndice.]

[Colocar sempre um preâmbulo no apêndice. Não colocar tabelas e ou ilustrações sem identificação no apêndice. Caso existirem, identifique-as através da legenda, seguindo a numeração normal do volume final (para as legendas). Caso existirem tabelas e ou ilustrações, sempre referenciá-las antes.]

ANEXO A – Representação gráfica de contagem de citações de autores por semestre nos trabalhos de conclusões realizados no Curso de Ciência da Computação @@ Ir atualizando a medida que são referenciados na monografia

[Elemento opcional. **Anexos são documentos não elaborados pelo autor**, que servem de fundamentação, comprovação ou ilustração, como mapas, leis, estatutos, entre outros. Os anexos são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deverá haver no mínimo uma referência no texto anterior para cada anexo.]

[Colocar sempre um preâmbulo no anexo. Não colocar tabelas e ou ilustrações sem identificação no anexo. Caso existirem, identifique-as através da legenda, seguindo a numeração normal do volume final (para as legendas). Caso existirem tabelas e ou ilustrações, sempre referenciá-las antes.]