Tagarela – Módulo Jogo de Letras e Números

Aluno: André Felipe Ferreira

Orientador: Dalton Solano dos Reis

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Requisitos
- Especificação
- Implementação
- Resultados
- Conclusões e sugestões

Introdução

- Tecnologia assistiva
  - Inclusão social
  - Auxilio na educação e desenvolvimento
- Dispositivos móveis
  - Evolução
  - Tagarela
  - Jogo de Letras e Números

Objetivos

- Estender o aplicativo Tagarela adicionando o módulo Jogo de Letras e Números
- Objetivos específicos
  - Migra o Jogo de Letras e Números (REETZ, 2013)
  - Criar a miniatura para as pranchas do aplicativo Tagarela
  - Utilizar o *framework* Ionic para a construção da interface

Fundamentação Teórica

- Aprendizagem da escrita
- Tagarela
- Jogo de Letras e Números
- Cordova e Ionic

Aprendizagem da escrita

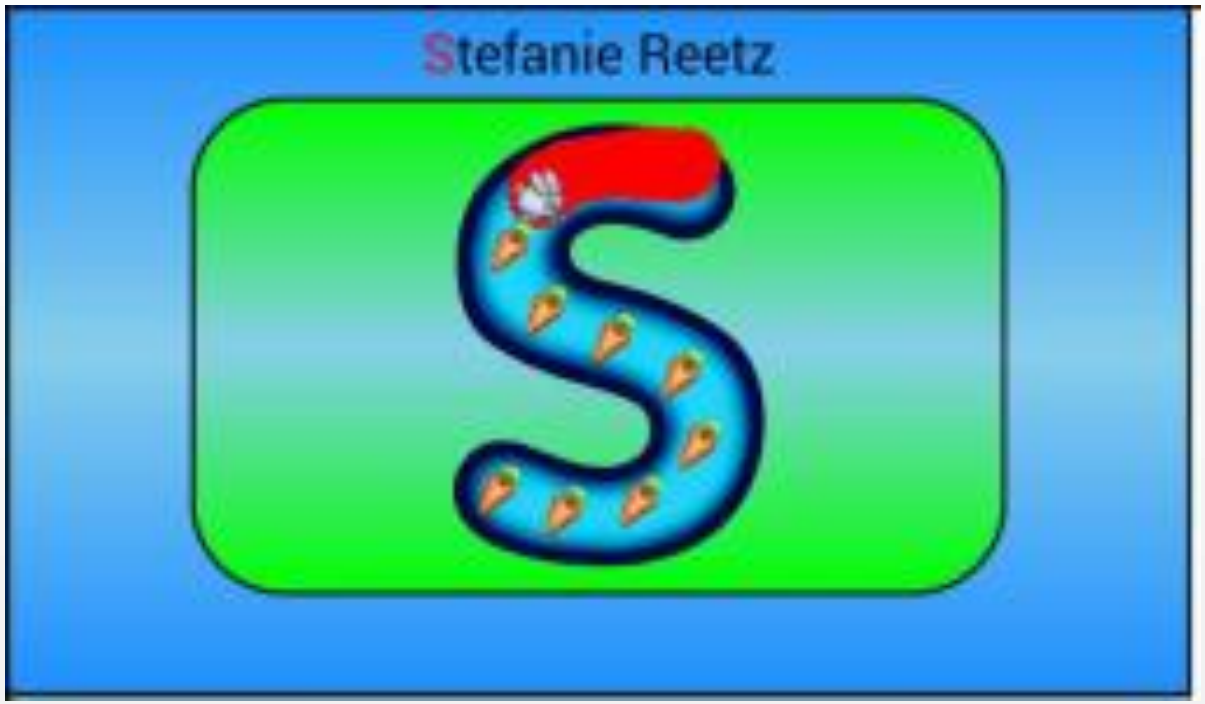
- Automatização da linguagem com os erros cometidos
- A leitura é ligada a escrita
- Jogos e brincadeiras proporcionam aprendizagem e interação social.
  - Instrumentos pedagógicos
  - Estimula a criatividade

Tagarela

- Ferramenta de Comunicação Alternativa
  - 2012: Alan Fabeni (iOS)
  - 2014: Darlan Diego de Marco (Android)
- André Felipe Wippel (2015), PhoneGap
- Pranchas e planos

Jogo de Letras e Números

- 2013: Wagner Jean Reetz (Android)



Cordova e Ionic

- HTML/Javascript
- Multiplataforma
- Plugins
- Canvas
- Ionic

Trabalhos Correlatos

Participar

Trabalhos Correlatos

Livox

Trabalhos Correlatos

Scala

Trabalhos Correlatos

Características	Participar (2014)	Livox (2015)	Scala (2015)	Trabalho Desenvolvido
Perfil de usuário	Não	Sim	Não	Sim
Plano de atividade	Não	Não	Sim	Sim
Histórico	Não	Sim	Sim	Sim
Gratuito	Sim	Não	Sim	Sim
Plataformas	Windows Linux	Android	Android Web	Android iOS Web

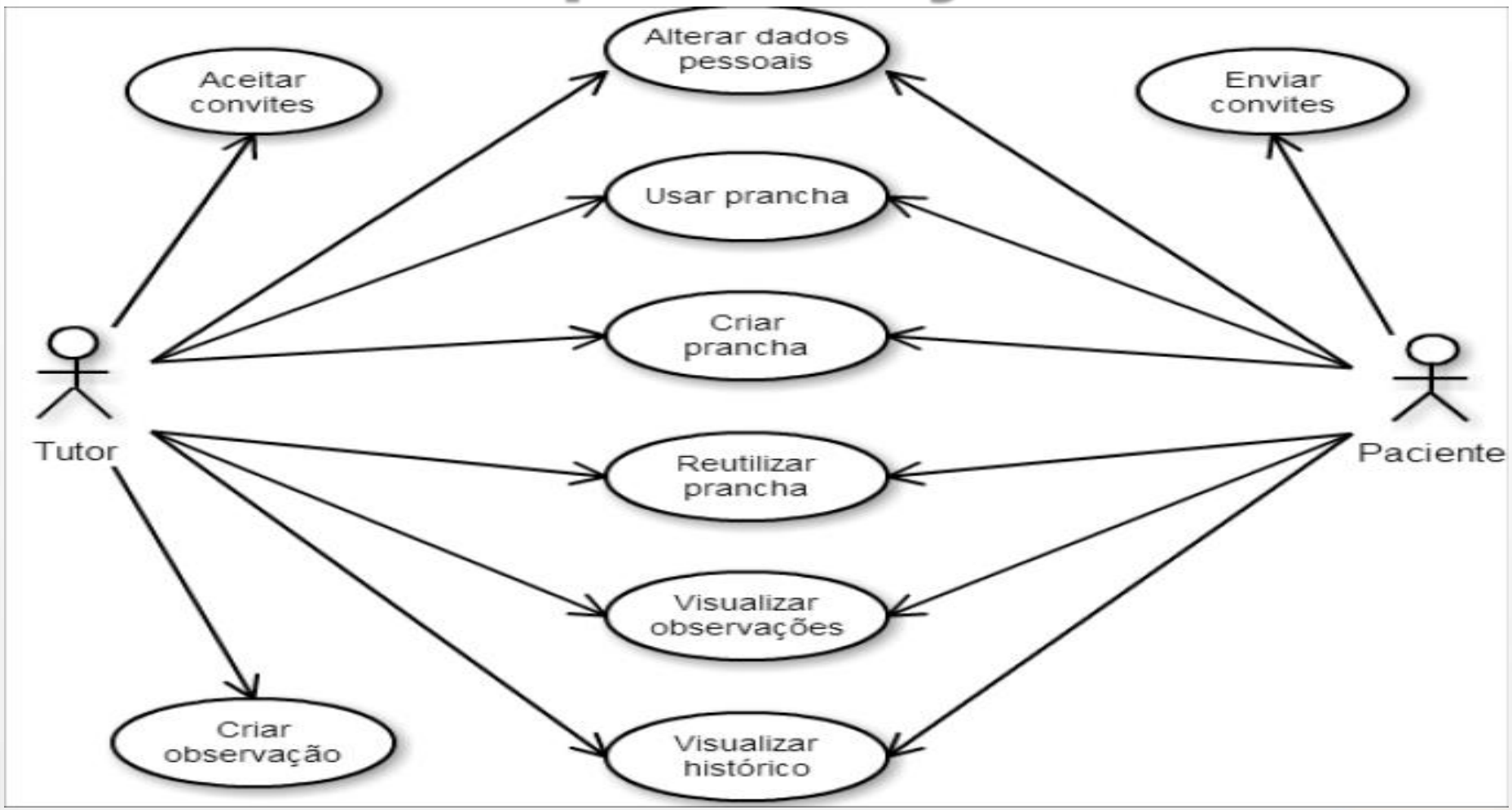
Requisitos funcionais

- o sistema deve manter um histórico de cada paciente;
- utilizar recursos áudio e imagem para aumentar a imersão do usuário no jogo;
- o sistema deve permitir trabalhar com todas as letras e números da língua portuguesa;
- o sistema deve permitir a interação do usuário;
- o sistema deve controlar a colisão no preenchimento dos símbolos.

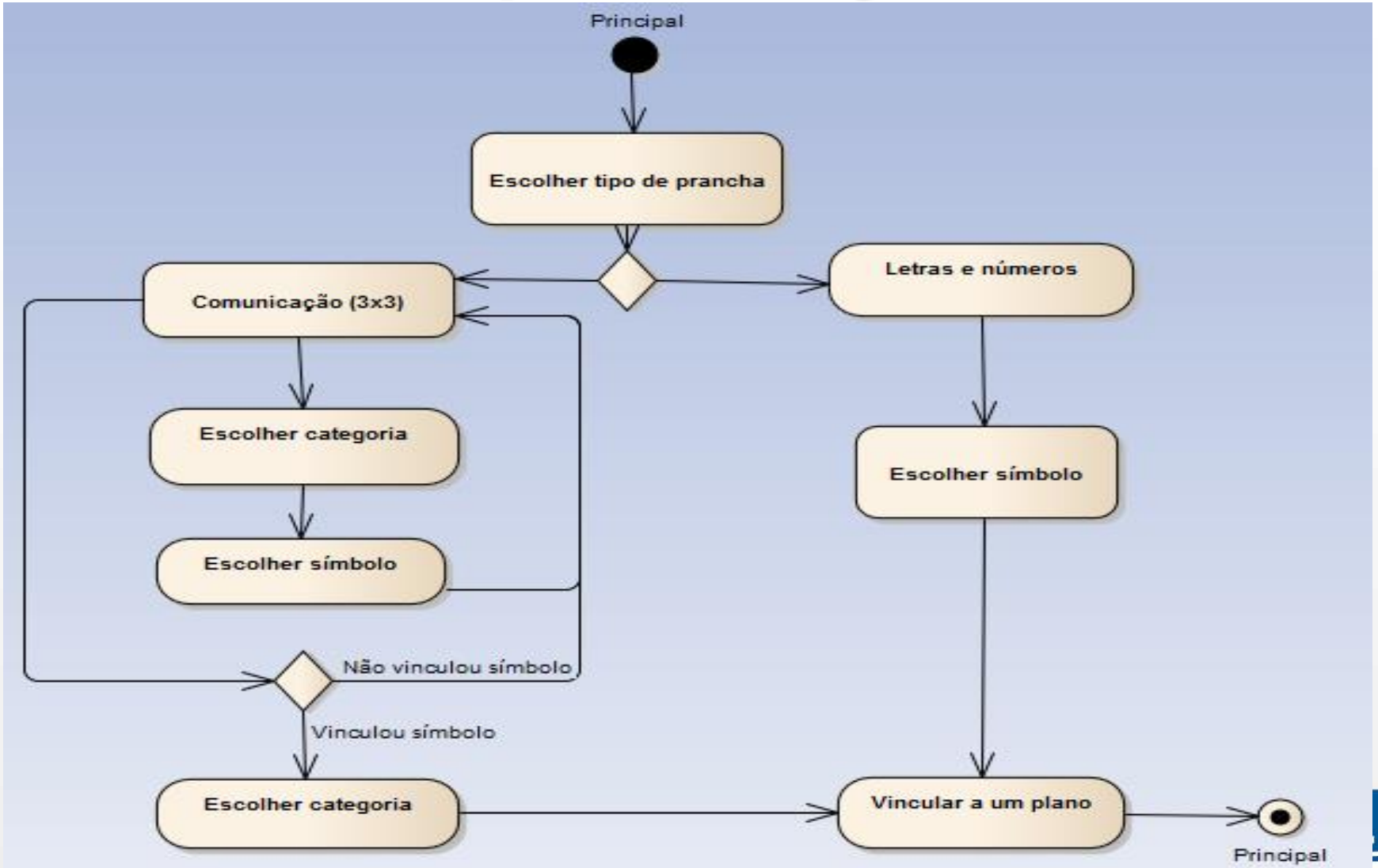
Requisitos não funcionais

- ser implementado utilizando tecnologia web, como HTML, PHP e Javascript;
- ser implementado utilizando o ambiente PhoneGap;
- apresentar uma interface que utilize o conceito de pranchas, seguindo o padrão já adotado pelo aplicativo Tagarela;
- utilizar o *framework* Ionic;
- ser um módulo integrado do Tagarela;
- utilizar os planos presentes no Tagarela.

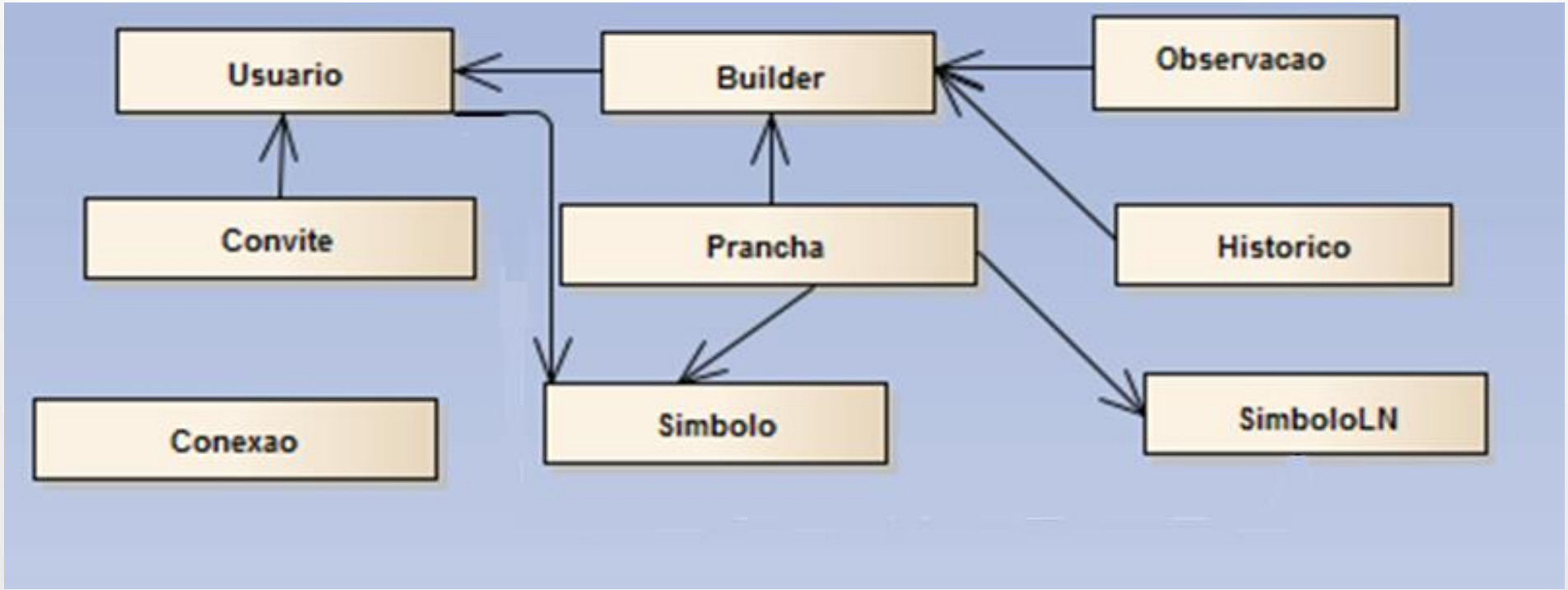
Especificação

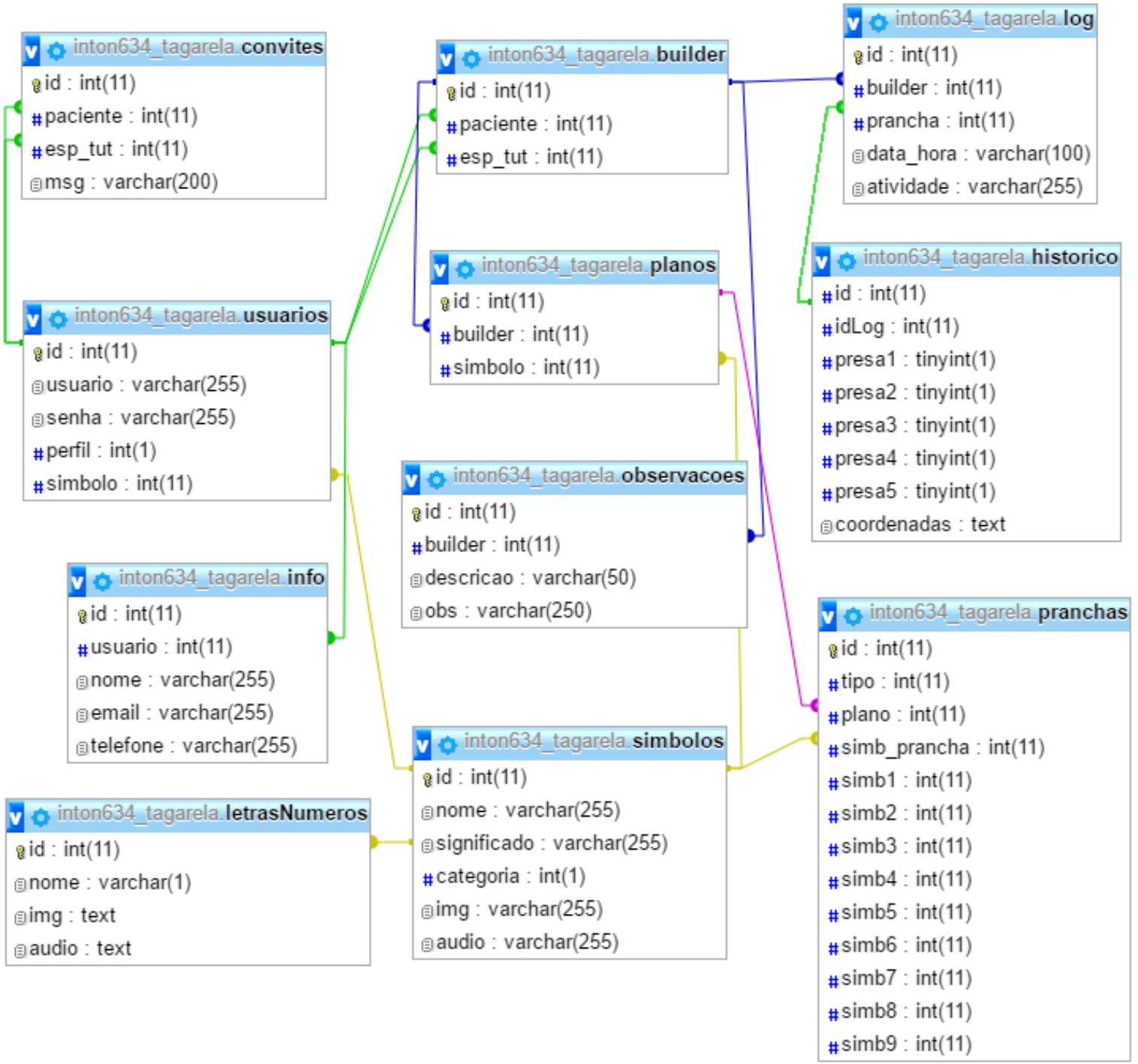


Especificação



Especificação



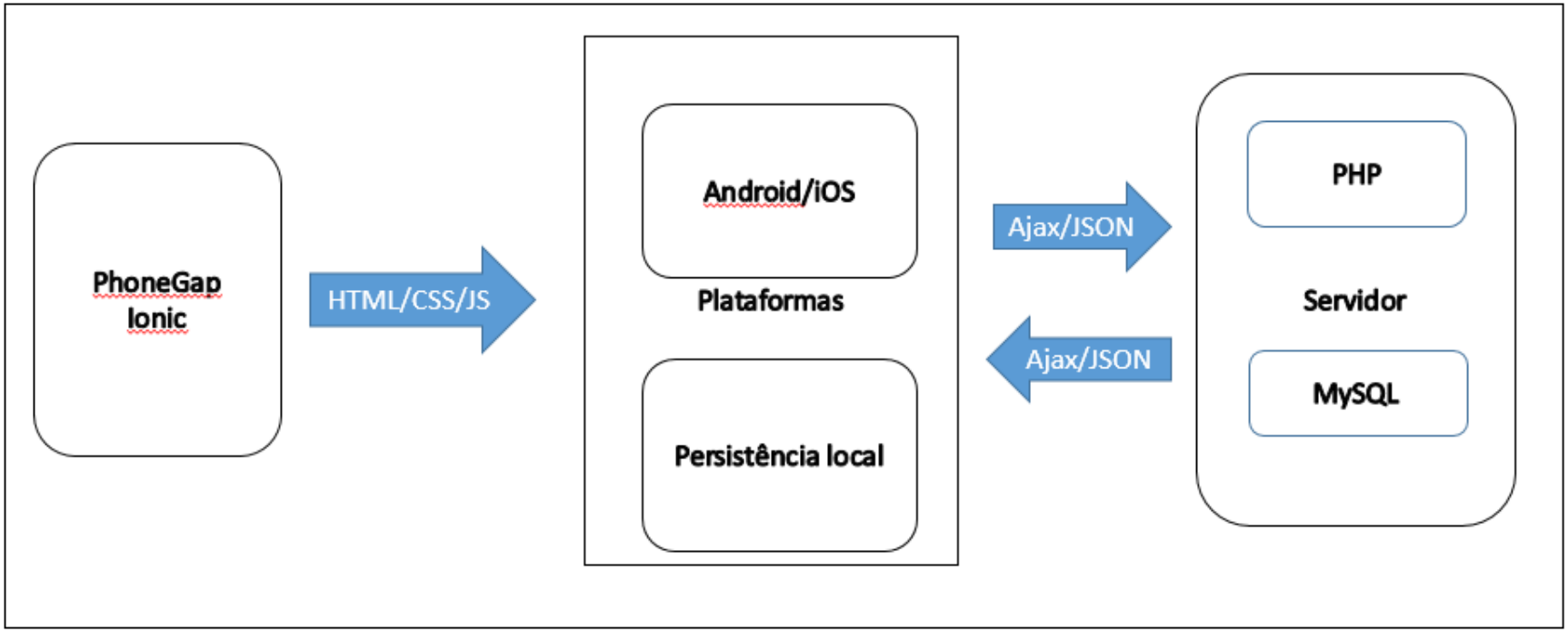

inton634_tagarela.convites
id : int(11)
paciente : int(11)
esp_tut : int(11)
msg : varchar(200)
inton634_tagarela.builder
id : int(11)
paciente : int(11)
esp_tut : int(11)
inton634_tagarela.log
id : int(11)
builder : int(11)
prancha : int(11)
data_hora : varchar(100)
atividade : varchar(255)
inton634_tagarela.planos
id : int(11)
builder : int(11)
simbolo : int(11)
inton634_tagarela.historico
id : int(11)
idLog : int(11)
presa1 : tinyint(1)
presa2 : tinyint(1)
presa3 : tinyint(1)
presa4 : tinyint(1)
presa5 : tinyint(1)
coordenadas : text
inton634_tagarela.usuarios
id : int(11)
usuario : varchar(255)
senha : varchar(255)
perfil : int(1)
simbolo : int(11)
inton634_tagarela.observacoes
id : int(11)
builder : int(11)
descricao : varchar(50)
obs : varchar(250)
inton634_tagarela.info
id : int(11)
usuario : int(11)
nome : varchar(255)
email : varchar(255)
telefone : varchar(255)
inton634_tagarela.pranchas
id : int(11)
tipo : int(11)
plano : int(11)
simb_prancha : int(11)
simb1 : int(11)
simb2 : int(11)
simb3 : int(11)
simb4 : int(11)
simb5 : int(11)
simb6 : int(11)
simb7 : int(11)
simb8 : int(11)
simb9 : int(11)
inton634_tagarela.simbolos
id : int(11)
nome : varchar(255)
significado : varchar(255)
categoria : int(1)
img : varchar(255)
audio : varchar(255)
inton634_tagarela.letrasNumeros
id : int(11)
nome : varchar(1)
img : text
audio : text

Implementação

- Ferramentas utilizadas:
  - Brackets 1.7
  - Easy PHP 14.1
  - Pronegap/Ionic
- Tecnologias:
  - PHP
  - HTML 5/Canvas
  - Javascript
  - CSS 3
  - MySQL
- Bibliotecas:
  - jQuery 1.7.2

Implementação

- Arquitetura:



Implementação

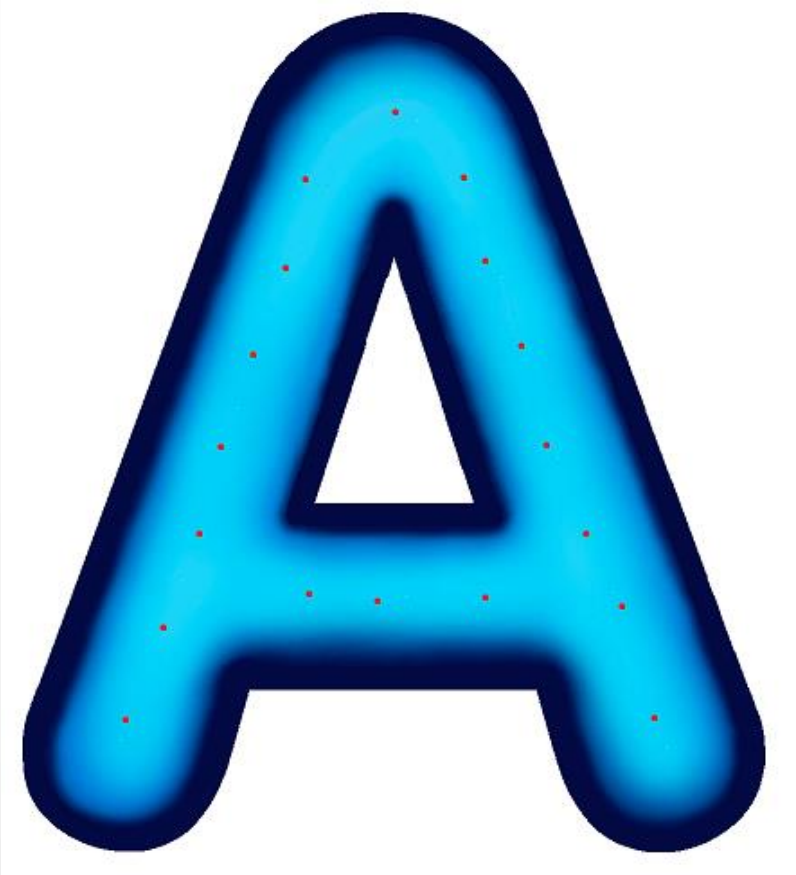
- Persistência dos dados:
  - Local: WebSQL
    - Comunicação: AJAX
    - Formato: JSON
    - Offline
  - Web: MySQL

Implementação

- Pranchas:
  - Propriedades do pixel (RGBA)
  - Alpha:
    - 0-5: presa e predador
    - 253: posição das presas
    - Controle de colisão

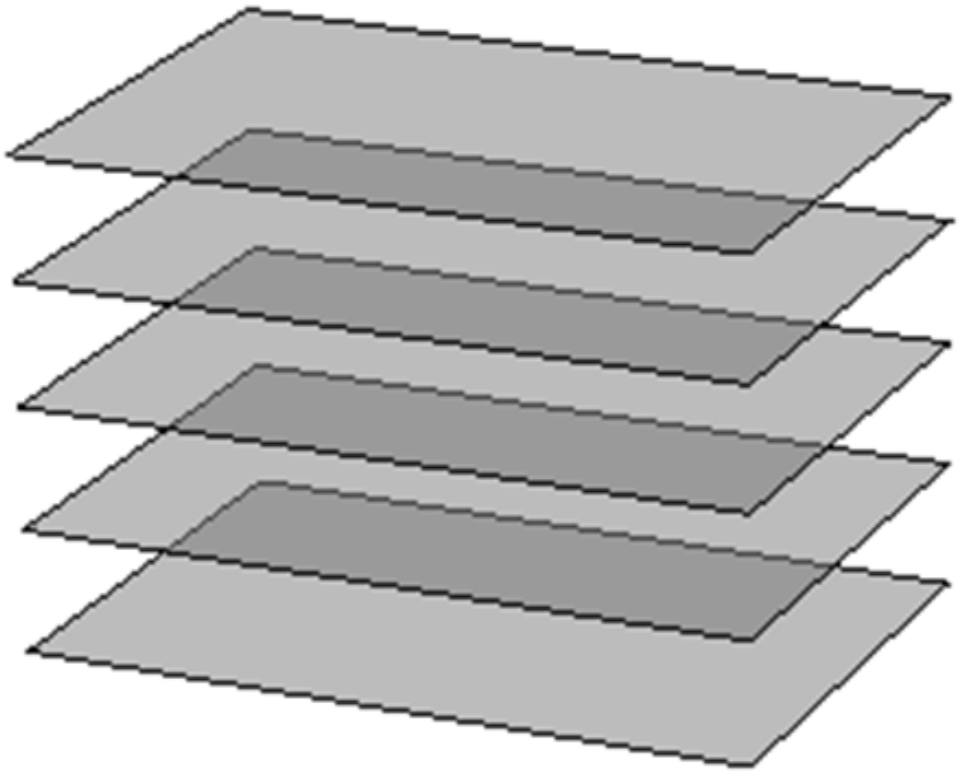
Implementação

- Pranchas:



Implementação

- Camadas da aplicação:



5) Tratamento do toque

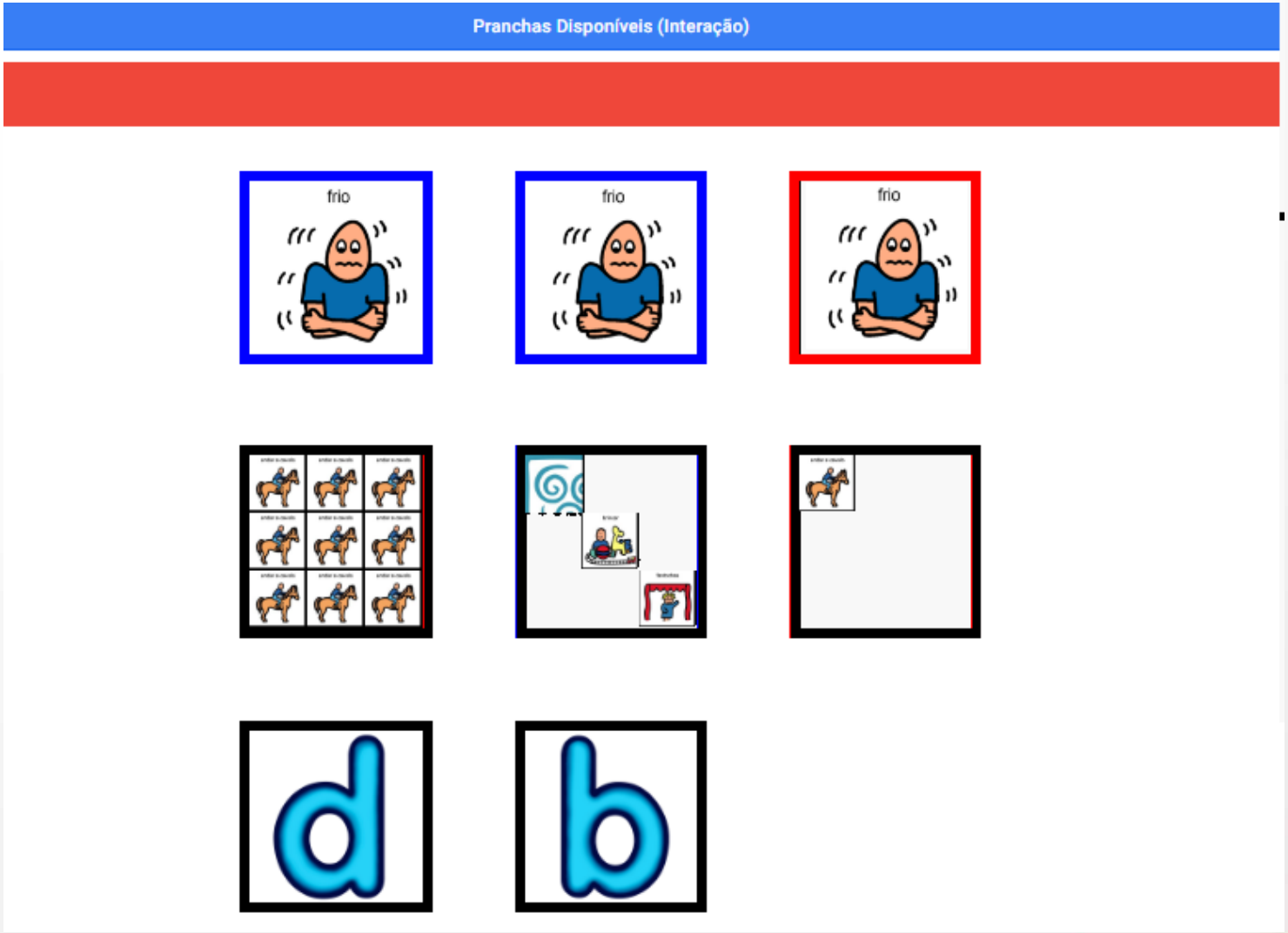
4) Símbolos (presas e predador)

3) Canvas do traçado

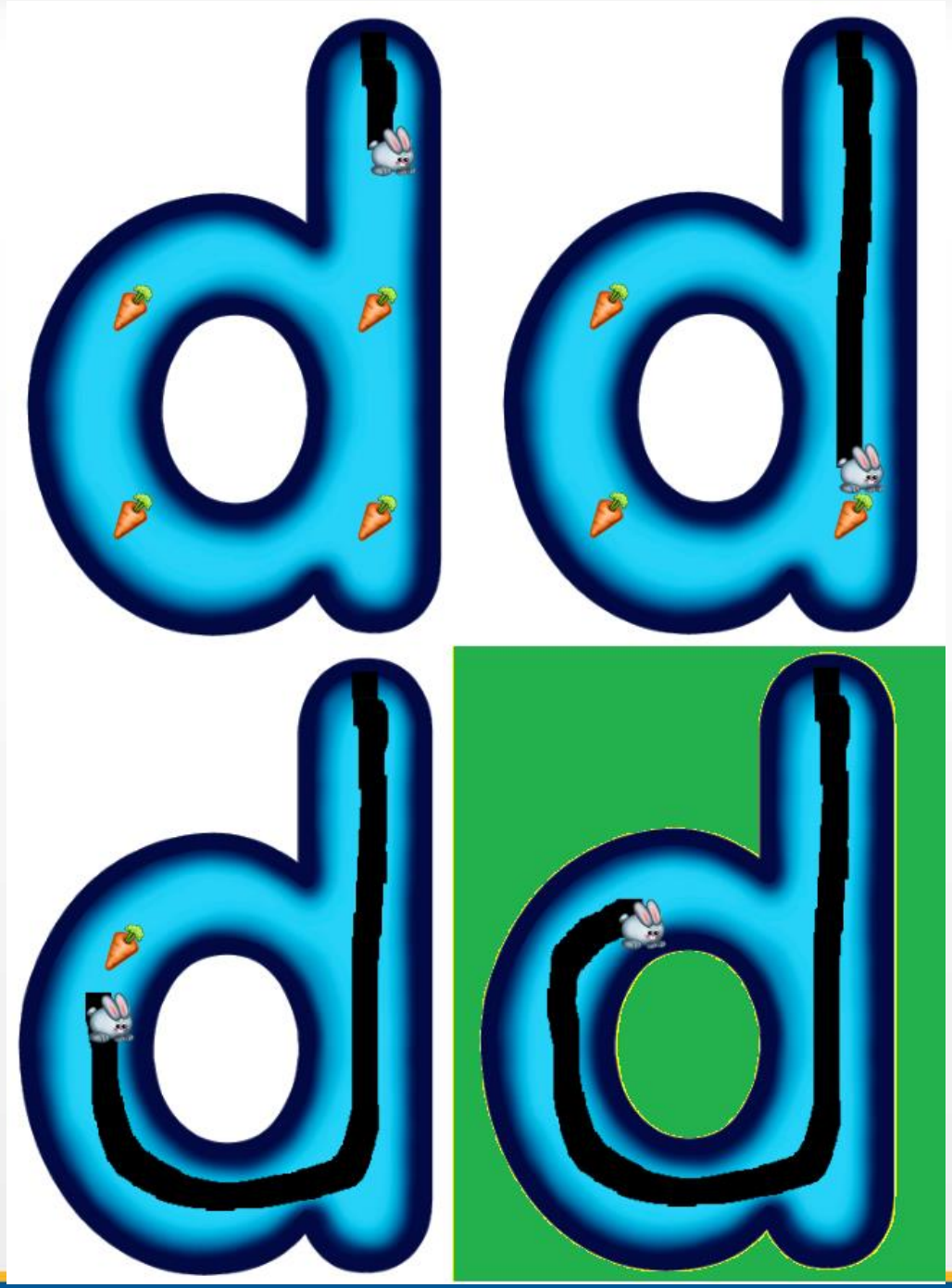
2) Canvas da imagem

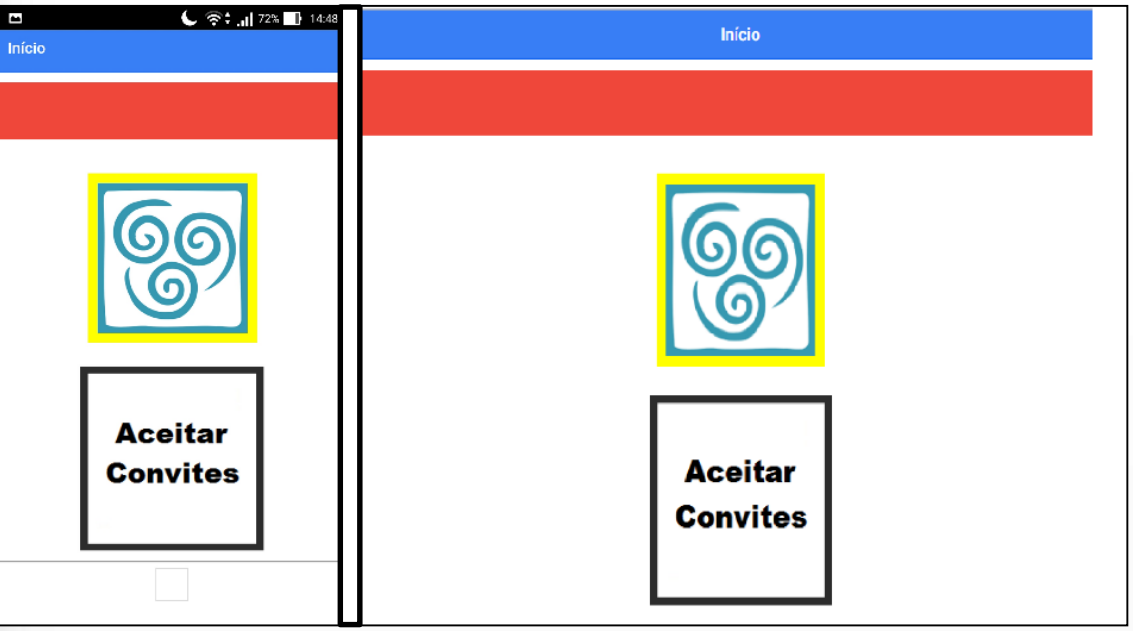
1) Imagem original

Utilização das pranchas



Login

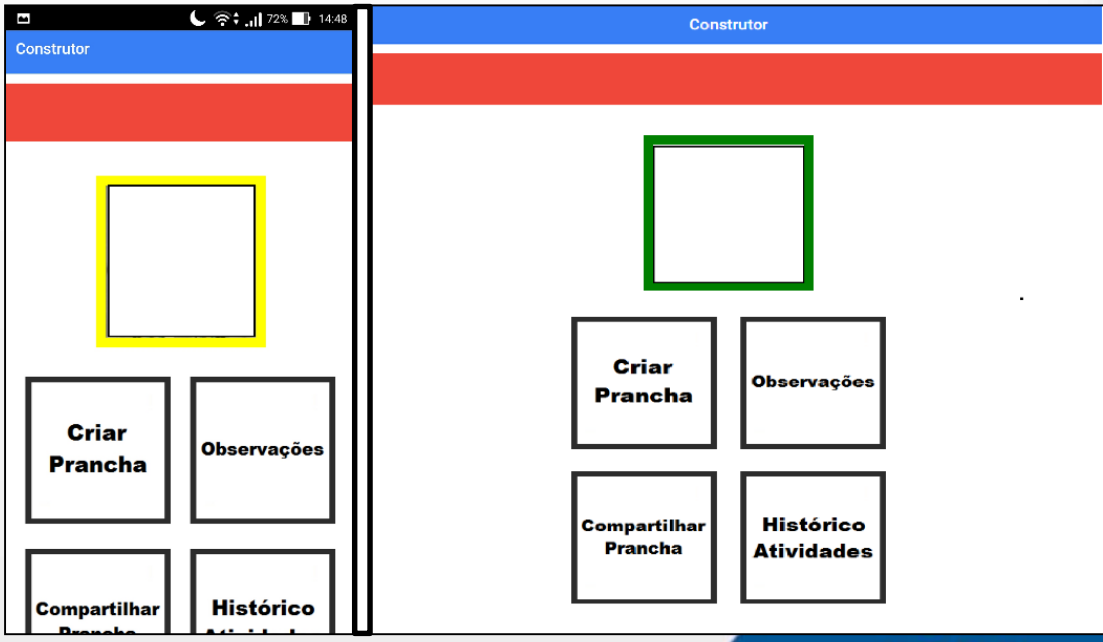




Google Chrome 54.0

ASUS ZenPhone 2
com Android versão 5.0

Comparação de interface



Resultados e Discussões

- Teste de usabilidade
- Resultados obtidos
  - Persistência dos dados
  - Navegação entre as pranchas
  - Desempenho

Conclusões e Sugestões

- Migração
- Interface das plataformas
- Controle do jogo pelas imagens
- Ferramentas adequadas
- Sincronização dos dados com o servidor
- Tratamento dos eventos
- Tratamento das imagens

Extensões

- Persistência de dados integrada entre as plataformas
- Melhorar navegação com Ionic e Angular
- Estender aplicação para Windows Phone
- Melhoria da interface com símbolos
- Criação de símbolos personalizados
- Desempenho (utilização de *plugin*)

Apresentação prática

Implementação

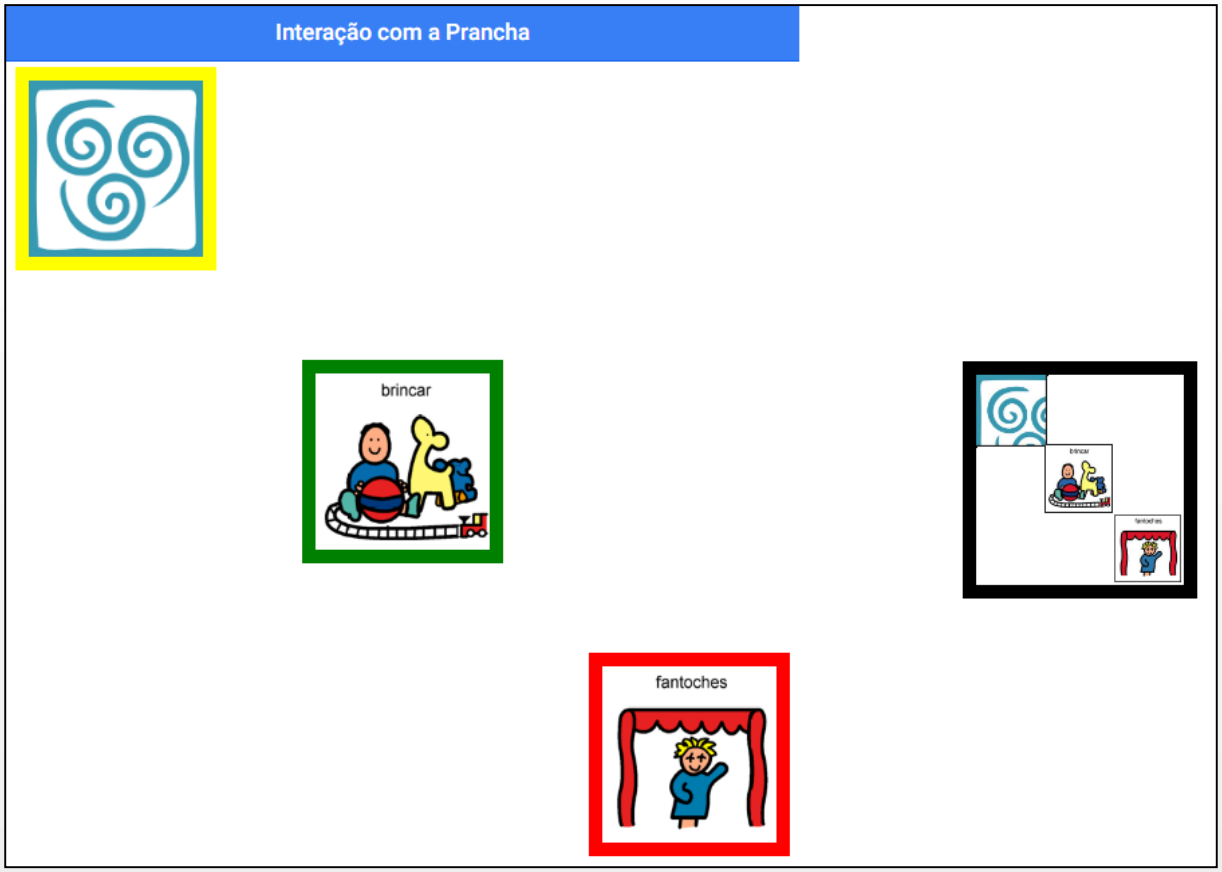
- Desenho do traçado

```
106                 mouseX = e.pageX - this.offsetLeft;
107                 mouseY = e.pageY - this.offsetTop;
108
109                 // find all points between
110                 var x1 = mouseX,
111                     x2 = lastX,
112                     y1 = mouseY,
113                     y2 = lastY;
```

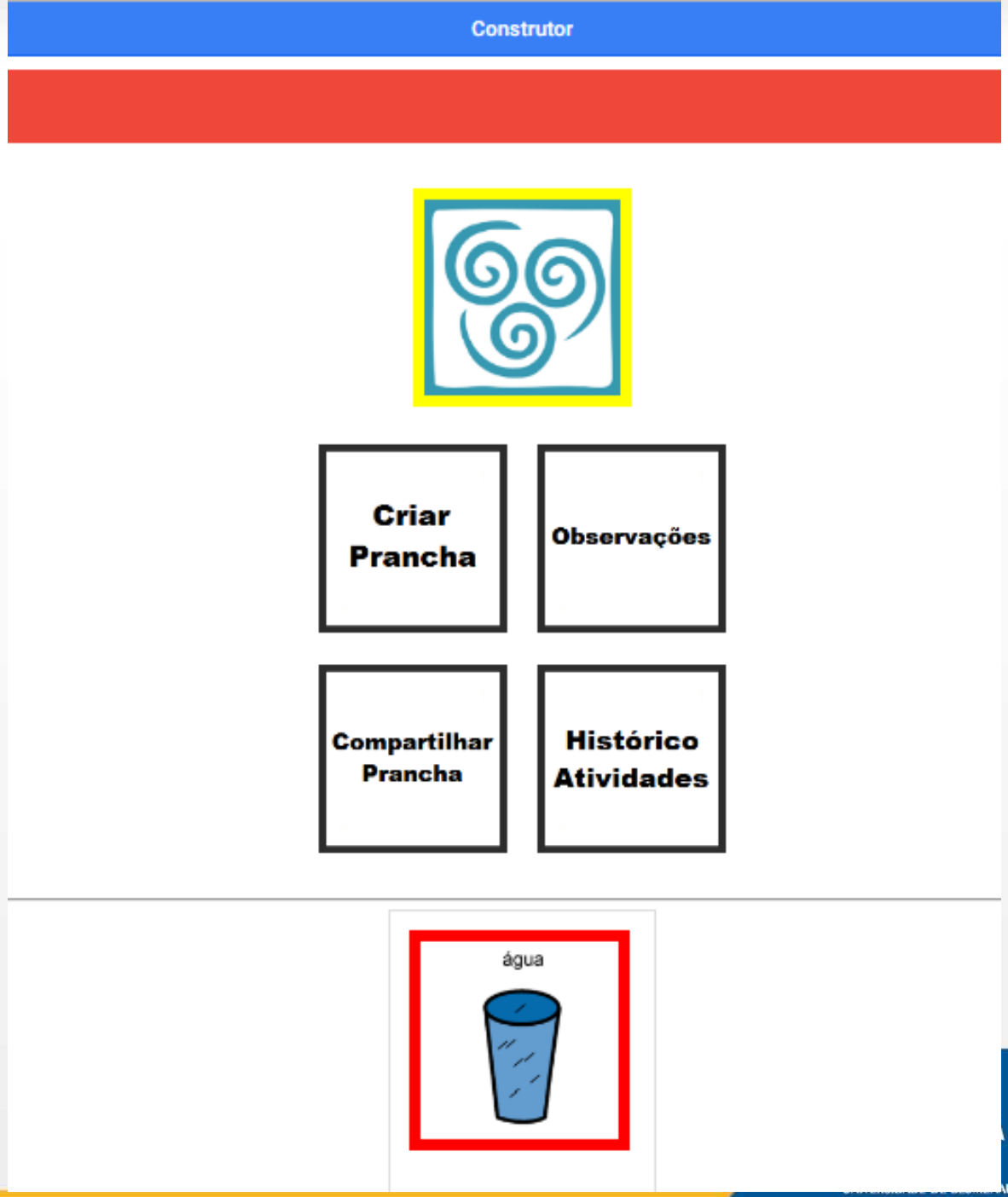
```
147                 lineThickness = 20 - Math.sqrt((x2 - x1) *(x2-x1) + (y2 - y1) * (y2-y1))/10;
148                 if(lineThickness < 1){
149                     lineThickness = 1;
150                 }
151
152                 for (var x = x1; x < x2; x++) {
153                     if (steep) {
154                         ctx.fillRect(y, x, lineThickness , lineThickness );
155                     } else {
156                         ctx.fillRect(x, y, lineThickness , lineThickness );
157                     }
158
159                     error += de;
160                     if (error >= 0.5) {
161                         y += yStep;
162                         error -= 1.0;
163                     }
164                 }
165                 lastX = mouseX;
166                 lastY = mouseY;
```

Implementação

- Miniaturas:



Criação das pranchas



Criação das pranchas