Projeto *Exploding Kittens*

**Especificação de Requisitos de Software**

Versão 1.1.2

30/11/2018

|  |  |  |  |
| --- | --- | --- | --- |
| **Histórico de Versões** | | | |
| **Versão** | **Autor(es)** | **Data** | **Ação** |
| 1.0 | Gustavo Ghedin Ditzel  Mateus Nunes Cechetto  Vinícius Monteiro Dias | 29/08/2018 | Especificação inicial de requisitos. |
| 1.1 | Gustavo Ghedin Ditzel  Mateus Nunes Cechetto  Vinícius Monteiro Dias | 14/09/2018 | Refinamento dos requisitos funcionais |
| 1.1.1 | Gustavo Ghedin Ditzel  Mateus Nunes Cechetto  Vinícius Monteiro Dias | 24/09/2018 | Refinamento do requisito funcional 4 e renumeração dos requisitos funcionais. |
| 1.1.2 | Gustavo Ghedin Ditzel  Mateus Nunes Cechetto  Vinícius Monteiro Dias | 30/11/18 | Refinamento dos requisitos funcionais |

**Conteúdo:**

[**Introdução**](#_gjdgxs) **2**

[Objetivos](#_30j0zll) 2

[Abreviaturas e Definições](#_1fob9te) 2

[Referências](#_3znysh7) 2

[Regras do jogo (adaptadas)](#_2et92p0) 2

[**Visão Geral**](#_tyjcwt) **3**

[Arquitetura da aplicação](#_3dy6vkm) 3

[Premissas de desenvolvimento](#_1t3h5sf) 3

[**Requisitos da aplicação**](#_4d34og8) **3**

[Requisitos funcionais](#_2s8eyo1) 3

[Requisitos não-funcionais](#_17dp8vu) 4

[**Protótipo da interface gráfica do programa**](#_3rdcrjn) **4**

# Introdução

## Objetivos

O projeto aqui especificado tem por objetivo o desenvolvimento de um jogo eletrônico baseado no jogo de cartas *Exploding Kittens*, adaptado para propiciar a disputa de partidas apenas entre dois jogadores.

## Abreviaturas e Definições

A seguir, as definições das cartas utilizadas no jogo:

* *Exploding Kitten*: carta explosiva que provoca a morte do jogador (fim de partida), caso o mesmo não possua uma carta *Defuse*;
* *Defuse*: carta que permite desarmar a explosão da carta *Exploding Kitten*;
* *Nope*: carta que pode ser jogada imediatamente (mesmo que não seja a vez do jogador) para anular a carta jogada pelo adversário;
* *See the Future*: carta que permite ver as 3 primeiras cartas do baralho;
* *Change the future*: carta que permite ver as 3 primeiras cartas do baralho e alterar suas posições, entre si, se desejado;
* *Skip*: carta que permite encerrar o turno sem comprar uma carta, passando a vez para o adversário;
* *Favor:* carta que obriga o adversário a ceder uma carta aleatória de sua mão;
* *Kitten*: carta sem ação específica, representa um gato de uma determinada tribo; só é útil se utilizada em par com outra carta igual (ou seja, outro gato da mesma tribo), ao permitir pegar uma carta aleatória da mão do adversário.

## Referências

As regras do jogo original (*First, Original and NSFW*), bem como vídeos explicativos e exemplos de jogo, podem ser encontradas em <https://explodingkittens.com/how-to-play/>.

## Regras do jogo (adaptadas)

Em relação às regras originais, adaptamos a quantidade de cartas para 2 jogadores, assim como removemos as cartas *Attack* e *Shuffle*,e adicionamos o tipo *Change the future*. Combo (combinação) de 2 cartas será apenas válido para as cartas *Kitten*. Os combos especiais de 3 cartas do mesmo tipo e 5 cartas diferentes também não estarão presentes.

# Visão Geral

## Arquitetura da aplicação

Programa orientado a objetos e distribuído com base no modelo cliente-servidor.

## Premissas de desenvolvimento

* Programa deve possuir uma interface gráfica;
* Linguagem de programação adotada para desenvolvimento deve ser Java;
* A modelagem da especificação deve ser baseada em UML 2;
* O programa deve obrigatoriamente executar distribuído, com o suporte para aplicações distribuídas *NetGamesNRT*.

# Requisitos da aplicação

## Requisitos funcionais

1. Conectar-se ao servidor: o programa deve ter uma opção de menu para que o jogador possa se conectar ao servidor de uma rede de jogadores e disputar partidas;
2. Iniciar nova partida: o programa deve ter uma opção de menu para que o jogador possa iniciar uma nova partida do jogo, se já estiver conectado a um servidor;
3. Desconectar-se do servidor: o programa deve ter uma opção de menu para que o jogador se desconecte do servidor, se estiver conectado a um;
4. Receber solicitação de início: recebe uma solicitação do servidor, que sorteia o jogador que começará o jogo. Este instanciará a mesa, consequentemente o baralho, e distribuirá 7 cartas para cada jogador, para que a partida possa efetivamente ser iniciada. O outro jogador receberá a mesa pronta para o início da partida.

Se houver uma partida em andamento, o programa deve possibilitar ao jogador:

1. Jogar Par: proceder uma jogada de par de cartas *Kitten* da mesma tribo (*combo*); ao jogar o par, será saqueada uma carta aleatória do adversário;
2. Jogar Carta *Nope*: proceder uma jogada da carta *Nope*, mesmo que não seja sua vez de jogar;
3. Jogar Carta *Defuse*: caso o jogador pesque a carta *Exploding Kitten* e possua uma carta *Defuse*, a carta *Defuse* é utilizada e a carta *Exploding Kitten* volta ao baralho, na posição escolhida pelo jogador;
4. Receber jogada: o programa deve receber a jogada do adversário, através do servidor do jogo, e informar ao jogador, perguntando se deseja ou não jogar carta *Nope*, caso a possua;
5. Comprar carta: além de incluir uma nova carta na mão, comprar carta do baralho encerra o turno em andamento; caso o jogador tenha jogado uma carta *Skip*, ao efetuar a compra o turno será passado sem incluir carta em sua mão;
6. Jogar Carta: proceder uma jogada se for sua vez de jogar;

Se uma carta for jogada, o programa deve:

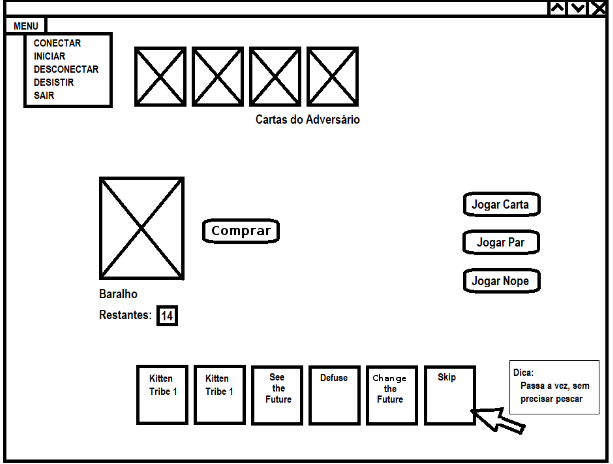
1. Ver topo: Ao jogar uma carta de *See the Future* ou C*hange the Future*, o programa deve mostrar as 3 cartas do topo do baralho;
2. Mudar posição: Ao jogar uma carta de *Change the Future,* o programa deve indicar as 3 cartas do topo do baralho e ordená-las da forma que jogador escolher;
3. Roubar uma carta do adversário: Ao jogar uma carta *Favor,* o programa deve retirar uma carta da mão do adversário e adicioná-la a mão do jogador.

## Requisitos não-funcionais

1. O desenvolvimento deve ser efetuado com o paradigma de orientação a objetos, na linguagem Java;
2. A modelagem do jogo deve ser efetuada com a linguagem UML 2;
3. A ferramenta utilizada para modelagem deve ser o Visual Paradigm;
4. A aplicação deve ter uma interface gráfica, com estados compartilhados pelos usuários que estiverem jogando, desenvolvida com a biblioteca gráfica *Swing*;
5. A conexão entre os jogadores deve ser realizada com a plataforma *NetGamesNRT*;
6. O jogo deve ser disponibilizado aos usuários por meio de um arquivo executável no formato JAR e notas de lançamento (*release notes*).

# Protótipo da interface gráfica do programa

A figura a seguir apresentam um esboço do que se pretende ter em termos de interface gráfica ao fim do processo de produção do jogo. A versão final pode diferir.

****