

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

**Projeto: Especificação de Requisitos de Software**

**Toribash**

**Versão 3.0**  
12/12/2020

**Florianópolis  
Dezembro de 2020**

## ESPECIFICAÇÕES DE REQUISITOS DE SOFTWARE

Versão	Autores	Data	Ação
1.0	Marcelo Muller Lucas Espindola	15/09/2020	Análise inicial dos requisitos do software
1.1	Marcelo Muller Lucas Espíndola	18/10/2020	Adequação ao feedback recebido sobre a 1.0
2.0	Marcelo Muller Lucas Espindola	25/10/2020	Pequena alteração no funcionamento da mecânica Nirvana
2.1	Lucas Espindola	06/12/2020	Refinamento do dano dos ataques
3.0	Lucas Espindola Marcelo Muller	12/12/2020	Refinamento das funcionalidades, tendo em vista a implementação final

<b>Introdução</b>	<b>4</b>
1.1 Objetivos	4
1.2 O Jogo	4
1.2.1 Regras e funcionamento	5
<b>Visão Geral</b>	<b>7</b>
2.1 Arquitetura do programa	7
2.2 Premissas de desenvolvimento	7
<b>Requisitos de Software</b>	<b>8</b>
3.1 Requisitos funcionais	8
3.2 Requisitos não-funcionais	9
<b>Interface gráfica</b>	<b>9</b>

# **1. Introdução**

## **1.1 Objetivos**

Desenvolver um programa utilizando a linguagem Java, baseando-se no jogo Toribash, consistente de partidas entre dois jogadores por meio de conexão por servidor.

## **1.2 O Jogo**

Toribash é um jogo inovador que simula movimentos de luta criados pelo jogador, levando em consideração os limites do corpo humano. O objetivo é nocautear o adversário realizando uma série de movimentos e desviando ou bloqueando os movimentos do inimigo. Vamos produzir uma versão simplificada que se encaixe nos moldes e requisitos da disciplina como homenagem a um jogo tão amado pela comunidade da Internet.

### 1.2.1 Regras e funcionamento

- Cada jogador possui um personagem com uma barra de vida (que começa em 100) e uma série de movimentos.
- A cada turno, o lutador terá a opção de:
  - Atacar o inimigo rapidamente - 5 de dano;
  - Atacar o inimigo com força - 10 de dano;
  - Se defender - cancela um ataque fraco ou reduz o dano de um forte para 5;
  - Realizar um Counter - inverte o ataque forte, causando dano extra ao inimigo (20 de dano);
  - Concentrar *chakra*;
- Depois de os dois jogadores escolherem o movimento, o resultado é mostrado de acordo (a ordem em que os jogadores selecionam seu movimento é irrelevante, dado que ambos são processados simultaneamente):
  - Ataque Rápido vs. Ataque Rápido: os dois tomam dano;
  - Ataque Rápido vs. Defesa: defesa feita com sucesso, ambos evitam dano;
  - Ataque Rápido vs. Concentrar: o jogador que se concentrou toma dano;
  - Defesa vs. Defesa: nada acontece;
  - Defesa vs. Concentrar: nada acontece;
- Counter e Ataque Forte são apostas em que o jogador tenta prever o que o outro vai fazer;

- Counter: movimento de defesa junto com contra-ataque:
  - Counter vs. Ataque Rápido: o jogador previu o ataque errado. Ele falha em se preparar a tempo e recebe o dano do ataque (5 de dano).
  - Counter vs. Defesa/Counter/Concentrar: nada acontece;
  - Counter vs. Ataque Forte: o lutador consegue defender e retribuir o ataque, causando dano extra (20 de dano).
  
- Ataque Forte: movimento de ataque mais lento, mas que ignora defesa:
  - Ataque Forte vs. Defesa: o jogador previu que o outro defenderia. Ele acerta o ataque forte, mas tem o dano reduzido pela defesa (5 de dano);
  - Ataque Forte vs. Ataque Rápido: o ataque rápido atinge antes, cancelando o Ataque Forte e causando 5 de dano;
  - Ataque Forte vs. Ataque Forte: os dois tomam o dano do Ataque Forte (10 de dano);
  - Ataque Forte vs Counter: o jogador tem seu movimento revertido, tomando dano extra (20 de dano)
  - Ataque Forte vs Concentrar: o lutador atinge o outro, causando 10 de dano no jogador que se concentrou.
  
- Concentrar Chakra deixa o jogador parado por um turno para abrir um ponto de *chakra* no corpo. São três, ao todo.
  - Ao abrir todos os três pontos de *chakra*, o jogador entra no modo Fúria;
    - Nos próximos três turnos, o jogador causará 2x o dano em todos seus movimentos;
  
- O vencedor será aquele que nocautear seu oponente, ou seja, reduzir a vida do inimigo a zero primeiro.

## **2. Visão Geral**

### **2.1 Arquitetura do programa**

Aplicação desenvolvida utilizando a linguagem Java, orientada a objetos, no padrão MVC de projeto e com suporte do NetGames para o aspecto distribuído.

### **2.2 Premissas de desenvolvimento**

A aplicação deve:

- Utilizar inteiramente a linguagem Java, sendo executável em qualquer ambiente que disponha da JVM (*Java Virtual Machine*);
- Ter suporte a rede para partidas online, utilizando a ferramenta NetGames;
- Apresentar uma interface gráfica aos usuários para facilitar seu uso.

## 3. Requisitos de Software

### 3.1 Requisitos funcionais

#### 1. Conectar

O jogador deve ter em seu menu a opção de conectar ao NetGames.

#### 2. Desconectar

O jogador deve ter em seu menu a opção de se desconectar do NetGames.

#### 3. Iniciar partida

Após a conexão efetuada, o jogador deve ter em sua tela inicial a opção de iniciar a partida, conectando-se a outro jogador por meio do NetGames.

#### 4. Realizar jogada

Durante a partida, o jogo deve permitir aos jogadores realizarem um dos cinco movimentos disponíveis.

#### 5. Mostrar o resultado de uma jogada

O jogo deve mostrar o resultado após um combate e/ou se o jogador aumentou seu status/atingiu um dos estados possíveis, isto é, se algum jogador causou ou recebeu dano, a quantidade de dano e/ou se entrou em modo Fúria/Nirvana ou liberou um ponto de *chakra*.

#### 6. Exibir informações sobre a partida e o oponente

O jogo deve exibir, em sua interface, o estado atual da luta, assim como dos lutadores, durante toda a partida, para que os jogadores saibam o que está acontecendo.



## **3.2 Requisitos não-funcionais**

### **1. Interface gráfica**

Com exceção das telas iniciais de conexão, o jogo deve conter interfaces compartilhadas entre os dois jogadores.

### **2. Projeto**

O jogo deve ser desenvolvido na linguagem Java, seguindo o paradigma de orientação a objetos, o padrão MVC (*Model View e Controller*).

### **3. Modelagem**

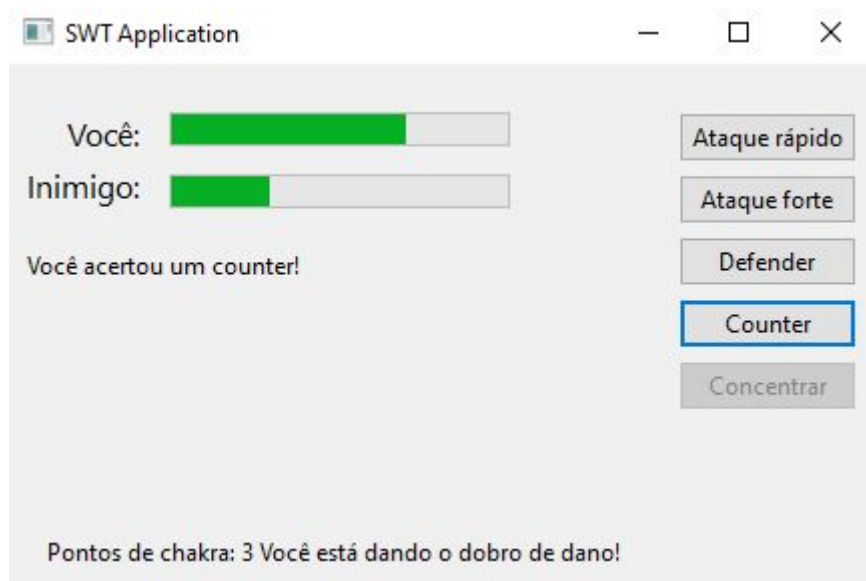
A modelagem do jogo deve ser feita em UML2, utilizando a ferramenta Visual Paradigm.

### **4. Tecnologia de interface gráfica**

A Interface gráfica com o usuário é baseada em Java Swing.

# Interface gráfica

## Primeira versão:



## Versão da entrega final:

