

# Projeto Jogo Skips

## Especificação de Requisitos de Software

Versão do jogo: 1.0

27/09/2022

Changelog:

Versão	Data	Autores	Ação
1.0	27/09/2022	Lucas Feliciano Luiz Muraro Vitor Chaves	Primeira especificação de requisitos
2.0	12/12/2022	Lucas Feliciano Luiz Muraro Vitor Chaves	Ultima especificação de requisitos

## **Conteúdo:**

<b>1. Introdução</b>	<b>3</b>
1.1 Objetivo	3
1.2 Resumo do jogo	3
<b>2. Regras detalhadas</b>	<b>4</b>
2.1 Regras	4
<b>3. Visão Geral</b>	<b>5</b>
2.1 Arquitetura do programa	5
2.2 Premissas de desenvolvimento	5
<b>3.Requisitos de Software</b>	<b>6</b>
3.1 Requisitos Funcionais:	6
3.2 Requisitos Não Funcionais	7

# 1. Introdução

## 1.1 Objetivo

Desenvolvimento de um software para a disciplina de Análise de Projetos e Sistemas, no semestre de 2022.2, que suporte o jogo [Skips](#), na modalidade de jogo online (distribuído), jogando dois jogadores, sendo os dois adversários.

## 1.2 Resumo do jogo

Um jogo para dois jogadores, que disputam quem consegue tirar todas as suas peças do jogo. O tabuleiro do jogo tem 6 casas, e cada jogador recebe 3 peças cada.

Cada uma das peças tem uma marcação (as mesmas marcações do tabuleiro)

Exemplo de peça

**
----

Exemplo de tabuleiro:

***	**	*	*	**	***
-----	----	---	---	----	-----

Para ganhar o jogo, o usuário tem que fazer suas peças percorrerem de uma lado pro outro do tabuleiro, no entanto, existem algumas regras, como por exemplo:

1. Uma peça não pode ficar na casa com a mesma marcação
2. Duas peças não podem ocupar o mesmo espaço
3. Uma peça pula a casa que já estiver ocupada
4. Se o usuário fica impossibilitado de movimentar uma peça, ele perde o jogo

## 2. Regras detalhadas

### 2.1. Regras

1. O tabuleiro inicia vazio
2. Cada jogador tem 3 peças
3. As jogadas são alternadas entre os dois jogadores
4. Na sua vez, o jogador é obrigado a mover uma de suas peças
5. O objetivo de cada jogador é retirar suas três peças do jogos depois delas terem percorrido todo tabuleiro, ou então travas as peças do adversário, impedindo elas de se moverem
6. Uma peça deve sempre andar a quantidade exata correspondente ao seu valor (1, 2 ou 3)
7. Uma peça nunca pode parar numa casa do seu valor
8. Para tirar uma peça do tabuleiro, é necessário andar com o valor exato correspondente, não podendo ultrapassar esse valor (peça fica travada)

## 2. Visão Geral

### 2.1 Arquitetura do programa

Cliente-servidor distribuído.

### 2.2 Premissas de desenvolvimento

- O programa deve ser implementado em Python;
- O programa deve usar DOG como suporte para execução distribuída;
- Além do código, deve ser produzida especificação de projeto baseada em UML, segunda versão.

## 3.Requisitos de Software

### 3.1 Requisitos Funcionais:

**Requisito funcional 1 – Iniciar programa:** quando executado o programa, deve ser apresentado um tabuleiro com 6 casas e as peças do jogo em seu estado inicial (3 peças de cada lado do tabuleiro). Além disso, deve solicitar o nome do jogador, juntamente com o logotipo do jogo. Deve solicitar conexão com DOG Server (utilizando os recursos de DOG). O resultado da tentativa de conexão deve ser informado ao usuário. Apenas em caso de conexão bem sucedida as demais funcionalidades estarão habilitadas. No caso de conexão mal sucedida, a única alternativa deve ser encerrar o programa;

**Requisito funcional 2 – Iniciar jogo:** assim que executado, o programa deve apresentar pro usuário a opção de iniciar a partida. Ao clicar nesse botão, será enviado uma solicitação de inicio ao DOG. O DOG retornará pro usuário a identificação e a ordem dos jogadores, caso tenha dado certa a conexão. Caso tenha ocorrido um erro, será mostrada uma mensagem ao usuário. A interface deve ser atualizada com as informações recebidas, o usuário com a ordem 1 deve ser habilitado para fazer sua jogada e uma mensagem “Selecione sua peça” deve ser mostrada à ele.

**Requisito funcional 3 – Zerar programa:** o programa deve permitir, ao ter a partida finalizada (seja ela encerrada após um jogo, ou inesperadamente), a opção de zerar o programa, ou seja, para seu estado inicial.

**Requisito funcional 4 – Selecionar peças:** O programa deve permitir a um jogador habilitado (quando for sua vez de jogar) selecionar uma peça presente em sua área de peças. Ao selecionar a peça, deve ser feita a verificação se a seleção da peça é válida (verificar nas regras do jogo uma jogada válida). Se a peça selecionada for inválida, deve notificar o usuário. Se a peça for válida, deve enviar a jogada para o

DOG (consequentemente, para o outro usuário - ver o RF 6): A peça selecionada. Além de atualizar a interface de ambos.

**Requisito funcional 5 – Receber determinação de início:** o programa deve poder receber uma notificação de início de partida, originada em Dog Server, em função de solicitação de início de partida por parte de outro jogador conectado ao servidor. O procedimento a partir do recebimento da notificação de início é o mesmo descrito no 'Requisito funcional 2 – Iniciar jogo', isto é, a interface do programa deve ser atualizada com as informações recebidas e caso o jogador local seja quem inicia a partida, a interface deve estar habilitada para seu procedimento de lance.

**Requisito funcional 6 – Receber jogada:** o programa deve poder receber uma jogada do adversário, enviada por Dog Server, quando for a vez do adversário do jogador local. A jogada recebida deve ser um lance regular e conter a informação da peça selecionada pelo adversário. O programa deve remover a peça da posição de origem e colocá-la no destino (uma das 6 casas do tabuleiro, ou fora dele, caso a peça tenha alcançado seu destino). Após isso, deve avaliar o encerramento de partida. No caso de encerramento de partida, deve ser notificado o nome do jogador vencedor; no caso de não encerramento, deve ser habilitado o jogador local, para que possa proceder a seu lance;

**Requisito funcional 7 – Receber notificação de abandono:** o programa deve poder receber uma notificação de abandono de partida por parte do adversário remoto, enviada por Dog Server. Neste caso, a partida deve ser considerada encerrada e o abandono notificado na interface.

## 3.2 Requisitos Não Funcionais

**Requisito não funcional 1 – Tecnologia de interface gráfica para usuário:** A interface gráfica deve ser baseada em TKinter;

**Requisito não funcional 2 – Suporte para a especificação de projeto:** a especificação de projeto deve ser produzida com a ferramenta Visual Paradigm;

**Requisito não funcional 3 – Interface do programa:** A interface do programa será produzida conforme o esboço da imagem abaixo:

