

---

# **ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE**

**para**

**JOGO DE TABULEIRO “ALAPO”**

**Versão 1.0**

**Preparado por:**

- 1. Pedro Santi Binotto (20200634)**
- 2. Gabriel Lemos da Silva (18200628)**
- 3. Petterson Pereira da Rosa (20105097)**

**Professor: Ricardo Pereira e Silva**

**26 de setembro de 2024**

# Sumário

<b>1</b>	<b>Introdução</b>	<b>4</b>
	Propósito . . . . .	4
	Regras do jogo . . . . .	4
	Sobre a movimentação . . . . .	5
	Configuração inicial do tabuleiro . . . . .	6
	Progressão da partida . . . . .	7
<b>2</b>	<b>Visão Geral</b>	<b>8</b>
	Histórico de Versões . . . . .	8
	Arquitetura do programa . . . . .	8
	Premissas de desenvolvimento . . . . .	8
<b>3</b>	<b>Requisitos de Software</b>	<b>9</b>
	Requisitos Funcionais . . . . .	9
	RF1: Iniciar programa . . . . .	9
	RF2: Iniciar jogo . . . . .	9
	RF3: Restaurar estado inicial . . . . .	9
	RF4: Selecionar peça . . . . .	10
	RF5: Selecionar destino . . . . .	10
	RF6: Receber determinação de início . . . . .	11
	RF7: Receber jogada . . . . .	11
	RF8: Receber notificação de abandono . . . . .	11
	Requisitos Não-Funcionais . . . . .	12
	RNF1: Tecnologia de interface gráfica para usuário . . . . .	12
	RNF2: Suporte para a especificação de projeto . . . . .	12
	RNF3: Interface do programa . . . . .	12



# 1 Introdução

## Propósito

Desenvolvimento de um programa distribuído que possibilite a realização de disputas do jogo “Alapo”[?] na modalidade *usuário vs. usuário*.

## Regras do jogo

O jogo deve ser jogado em um tabuleiro quadrado de 36 (6 × 6) posições; cada jogador deve iniciar uma partida com 12 peças para si, para um total de 24 peças no tabuleiro no momento que a partida é iniciada.

As peças do jogo são de seis (6) tipos diferentes, e cada jogador deve receber duas peças de cada tipo, sendo que as peças devem ser identificadas por cor, para que se possa fazer a distinção entre quais peças no tabuleiro pertencem a cada jogador:

### Peça Triangular Grande:

Sinalizada neste documento pelo símbolo “▲”, o movimento dessa peça é oblíquo, ou seja, pode movimentar-se diagonalmente no tabuleiro, e pode fazê-lo por quantas posições o jogador desejar; ***duas peças para cada jogador***;

### Peça Quadrada Grande:

Sinalizada neste documento pelo símbolo “■”, o movimento dessa peça é ortogonal, ou seja, pode movimentar-se horizontalmente como verticalmente no tabuleiro, e pode fazê-lo por quantas posições o jogador desejar; ***duas peças para cada jogador***;

### Peça Circular Grande:

Sinalizada neste documento pelo símbolo “●”; o movimento dessa peça abrange manobras diagonais assim como ortogonais, podendo ser movida em todas as direções, por quantas posições o jogador desejar; ***duas peças para cada jogador***;

### Peça Triangular Pequena:

Sinalizada neste documento pelo símbolo “▲”; o movimento dessa peça é oblíquo, ou seja, pode movimentar-se diagonalmente no tabuleiro, porém apenas uma por posição por vez; ***duas peças para cada jogador***;

### Peça Quadrada Pequena:

Sinalizada neste documento pelo símbolo “■”; o movimento dessa peça é ortogonal, ou seja, pode movimentar-se horizontalmente como verticalmente no tabuleiro, porém apenas uma por posição por vez; ***duas peças para cada jogador***;

**Peça Circular Pequena:**

Sinalizada neste documento pelo símbolo “●”; o movimento dessa peça abrange manobras diagonais assim como ortogonais, podendo ser movida em todas as direções, porém apenas uma por posição por vez; ***duas peças para cada jogador***;

**Sobre a movimentação**

- A movimentação do jogo **não permite** que peças “pulem” por cima umas das outras, ou seja, as peças só podem transitar por um trecho que está inobstruído do início ao fim;
- O jogo é jogado em turnos; cada jogador deve realizar **apenas um movimento por turno**;
- A *captura* é realizada ao passar uma peça para a posição onde uma peça do oponente está ocupando (análogo à forma que essa mecânica se apresenta no *xadrez* ou na *dama*);

## Configuração inicial do tabuleiro

Para começar uma partida as peças devem ser posicionadas, em extremidades do tabuleiro, por cada jogador, na seguinte disposição:

- As peças **maiores** (▲, ■, ●) devem ser posicionadas ao longo da **primeira linha** relativa ao jogador (linha “1” ou linha “6”, dependendo da perspectiva);
- As peças **menores** (▲, ■, ●) devem ser posicionadas ao longo da **segunda linha** relativa ao jogador (linha “2” ou linha “5”, dependendo da perspectiva);
- As peças **quadradas** (■, ■) devem ser posicionadas nas posições pertencentes às colunas mais externas do tabuleiro (colunas “A” e “F”);
- As peças **triangulares** (▲, ▲) devem ser posicionadas nas posições imediatamente internas às peças **quadradas** (colunas “B” e “E”);
- As peças **circulares** (●, ●) devem ser posicionadas nas posições pertencentes às colunas mais internas do tabuleiro (colunas “C” e “D”);

De maneira que o tabuleiro apresente a seguinte configuração, assim que todas as peças estejam devidamente posicionadas:

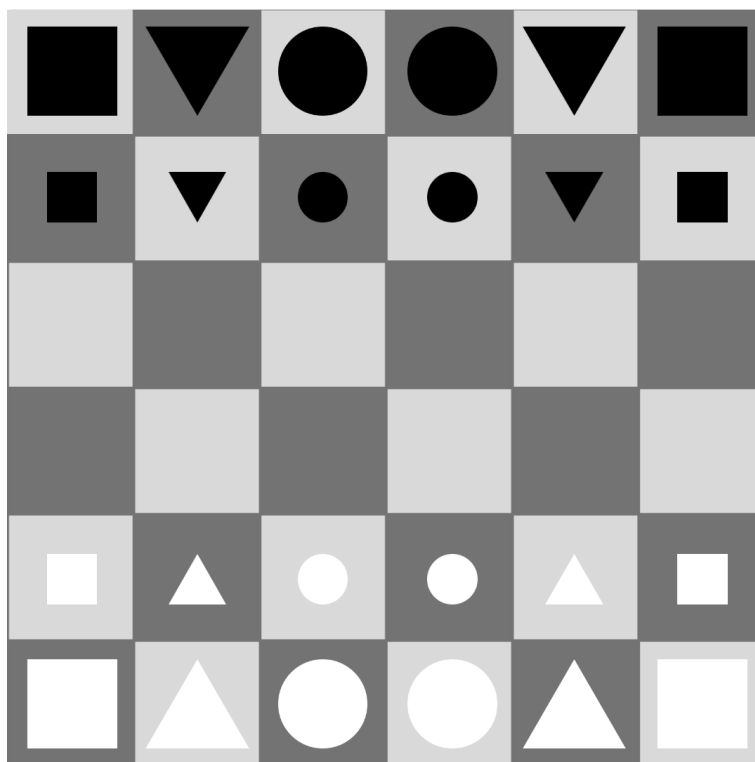


Figura 1.1: Configuração inicial do tabuleiro

## Progressão da partida

Assim que as peças estiverem posicionadas, um dos jogadores (qual jogador iniciará a primeira jogada pode ser escolhido aleatoriamente) escolhe uma das suas peças e faz o primeiro movimento. Os jogadores tomam turnos realizando lances (movimentos) e capturando as peças um do outro, com o objetivo alcançar a extremidade oposta do tabuleiro com uma de suas peças, sem ser capturado.

- Uma **vitória** é configurada quando um jogador alcança o extremo oposto do tabuleiro *sem ter sua posição imediatamente contestada*, isto é, quando a posição ocupada pela peça não está a alcance de captura de nenhuma peça do oponente;
- Um **empate** é configurado quando a condição de vitória é alcançada por ambos os jogadores *no intervalo de um turno*;
- Caso um jogador alcance o extremo oposto do tabuleiro e a posição ocupada pela peça seja contestável pelo oponente, **a contestação por parte do oponente é obrigatória** e deve ser realizada no turno imediatamente após a ocupação da posição;

## 2 Visão Geral

### Histórico de Versões

Versão	Autor(es)	Data	Registro de alterações
v1.0	BINOTTO, P. S.; SILVA, G. S.; ROSA, P. P	16/09/2024	Definição de regras de domínio; Especificação de requisitos

### Arquitetura do programa

A arquitetura do programa é baseada em *cliente-servidor distribuído*.

### 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

### Requisitos Funcionais

#### RF1: Iniciar programa

Ao iniciar, o programa deve mostrar a interface do **tabuleiro em sua configuração inicial**, e solicitar que o usuário informe o seu nome, que será usado para identificar o jogador. Após o usuário informar seu nome e solicitar [RF2] **Iniciar jogo**, o programa deverá requisitar uma conexão com o servidor;

- Caso a conexão seja bem sucedida, apresentar uma mensagem de sucesso ao usuário e liberar demais funcionalidades do jogo;
- Caso contrário, informar o erro ao usuário e apresentar as opções:
  - Tentar novamente;
  - Fechar o programa;

#### RF2: Iniciar jogo

Na interface inicial apresentada ao [RF1] **Iniciar programa**, está inclusa a ação “**Iniciar jogo**”, liberada após o jogador informar seu nome. Para iniciar a partida, o programa enviará uma requisição ao servidor, que caso bem-sucedida, mostrará qual jogador realizará a primeira jogada, assim como suas respectivas identificações.

A interface deverá ser atualizada com as informações recebidas; caso o jogador local seja quem inicia a partida, a interface deve estar habilitada para seu procedimento de lance [RF4] **Selecionar peça**. Esta funcionalidade só deve estar habilitada se o programa estiver em seu estado inicial, isto é, sem partida em andamento e com o tabuleiro em seu estado inicial.

#### RF3: Restaurar estado inicial

O programa deve apresentar a opção de menu “**Restaurar estado inicial**” para reverter o programa à sua configuração inicial, isto é, sem partida em andamento e com o **tabuleiro em seu estado inicial**. Esta funcionalidade só deve estar habilitada se o programa estiver com uma partida finalizada.

#### RF4: Selecionar peça

O programa deve permitir a um jogador, quando está em seu turno, selecionar uma de suas peças no tabuleiro para jogar.

- Caso o oponente tenha realizado um lance que configura uma *vitória* no final do tabuleiro, e esta for *contestável* (ver *Regras do jogo*), então o jogador deve, obrigatoriamente, selecionar uma das peças que contestará a vitória do oponente;
- Caso contrário, o jogador pode selecionar qualquer uma de suas peças que possam realizar movimentos imediatamente;

As peças passíveis de seleção devem ser destacadas na interface. A peça selecionada deve, também, visualmente destacada da interface do programa, e após a seleção a interface deve destacar visualmente as posições para qual a peça pode ser movida; caso uma ou mais posições alcançáveis configure uma *captura* de uma peça do oponente, esta posição deve distinguir-se visualmente das demais alternativas, sinalizando que representa uma *captura*, e não apenas um movimento.

#### RF5: Selecionar destino

O programa deve permitir que um jogador, após realizar [RF4] **Selecionar peça**, selecione a posição de destino desta mesma peça, de acordo com o tipo de movimentos dessa peça, da configuração atual do tabuleiro, e do atual estado da partida (ver *Regras do jogo*);

- Caso exista uma peça do oponente em uma *posição que exige ser contestada*, esta posição deverá ser a única passível de seleção, e deverá ser visualmente realçada para sinalizar que pode ser selecionada;
- Caso contrário, todas as possíveis posições alcançáveis pela peça em sua posição atual devem ser destacadas e habilitadas para seleção;

Ao realizar a seleção de uma posição válida para completar o movimento, a instância do programa deve enviar ao servidor uma mensagem contendo informações sobre a jogada:

**Peça selecionada:** Um identificador que especifique qual peça está sendo movimentada no lance;

**Posição de origem:** Posição de origem do movimento;

**Posição de destino:** Posição de destino do movimento;

**Estado da partida:** Estado da partida, determinado pela última jogada; pode ser um valor entre:

**EM ANDAMENTO:** Quando o movimento realizada não categoriza, imediatamente, uma vitória para o jogador que realizou a jogada;

**FINALIZADA:** Quando o movimento categoriza, precisamente, uma *vitória* para o jogador que realiza a jogada (ver *Progressão da partida*);

Caso um movimento categorize uma *vitória* e finalize a partida, apresentar ao usuário a opção de [RF3] **Restaurar estado inicial**.

#### **RF6: Receber determinação de início**

O programa deve receber e processar uma notificação de início de partida, originada no servidor, 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 [RF2] **Iniciar jogo**, isto é, a interface do programa deve ser atualizada com as informações recebidas. Caso o jogador local seja quem inicia a partida, a interface deve estar habilitada para seu procedimento de lance, especificado em [RF4] **Selecionar peça** e [RF5] **Selecionar destino**.

#### **RF7: Receber jogada**

O programa deve poder receber e processar uma jogada do adversário, enviada pelo servidor, quando for o turno do oponente jogar. A jogada recebida deve ser um lance regular e conter as informações especificadas para o envio de jogada no [RF5] **Selecionar destino**. O programa deve remover a peça de origem definida e colocá-la no destino, e atender os critérios a seguir:

- O programa deve processar uma *captura*, quando este for o cenário caracterizado pela jogada recebida, removendo a peça capturada do tabuleiro;
- O programa deve processar uma *vitória* do oponente, quando este for o cenário caracterizado pela jogada recebida, finalizando a partida e apresentando [RF3] **Restaurar estado inicial**, da mesma forma como acontece em [RF5] **Selecionar destino**;
- O programa deve processar uma situação de *contestação obrigatória*, quando este for o cenário caracterizado pela jogada recebida, obrigando o jogador local a contestar o último lance do oponente, como especificado em [RF4] **Selecionar peça** e [RF5] **Selecionar destino**;

Caso a jogada não caracterize nenhum dos casos especiais listados acima, o programa segue para o turno do jogador local, com [RF4] **Selecionar peça**.

#### **RF8: Receber notificação de abandono**

O programa deverá poder receber e processar um sinal de abandono de partida por parte do oponente remoto, enviada pelo servidor; neste caso, a partida deve ser considerada encerrada e o abandono notificado na interface, apresentando ao usuário a opção de [RF3] **Restaurar estado inicial**.

## Requisitos Não-Funcionais

### RNF1: Tecnologia de interface gráfica para usuário

A interface gráfica deve ser baseada em **Tkinter**.

### RNF2: Suporte para a especificação de projeto

A especificação e documentação do projeto deve ser produzida com a ferramenta **Visual Paradigm**.

### RNF3: Interface do programa

A interface do programa será produzida conforme o esboço da imagem abaixo:

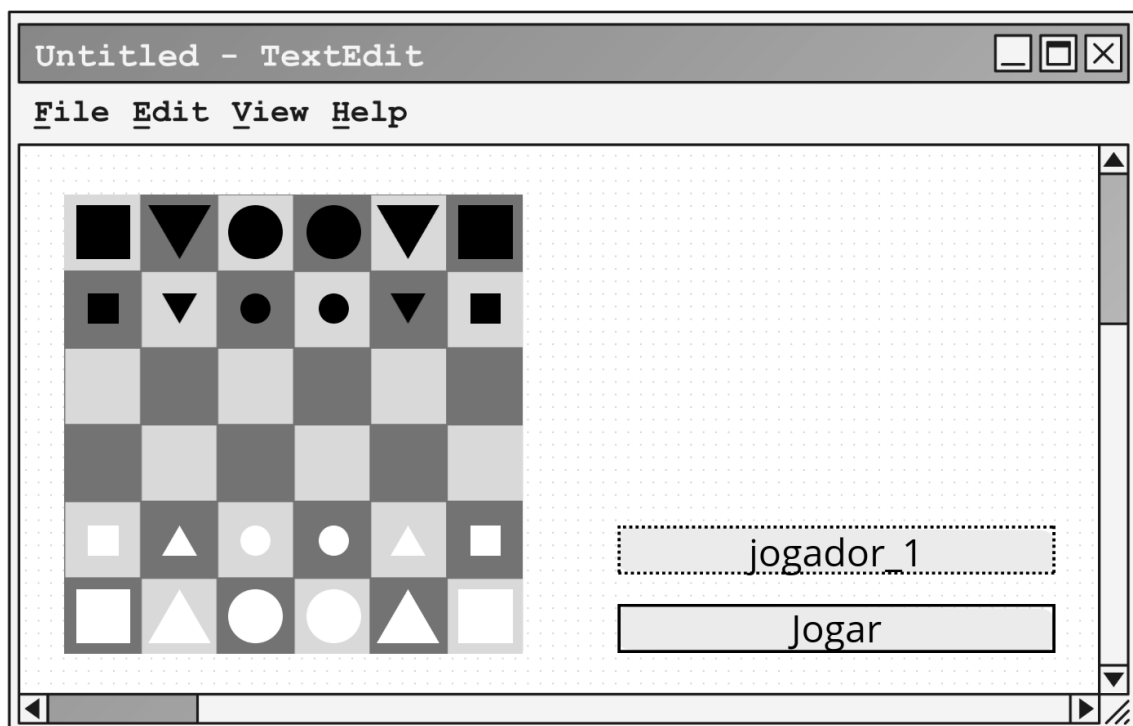


Figura 3.1: *Mockup* de interface gráfica do programa