

# Especificação de Requisitos

---

*Universidade Federal de Santa Catarina*  
*Ciências da Computação*  
*Março de 2022*

## **Labyrinthics**

**Versão do documento: 2.0**  
**14/03/2022**

Eric Fernandes Evaristo (21250089)  
Naiara Sachetti (19100540)

## **Versões deste documento**

<b>Versão</b>	<b>Autor(es)</b>	<b>Data</b>	<b>Ação</b>
1.0	Eric Fernandes Evaristo; Naiara Sachetti	22/11/2021	Estabelecimento inicial de requisitos funcionais, não funcionais e regras de negócio.
2.0	Eric Fernandes Evaristo; Naiara Sachetti	14/03/2022	Mudanças nas regras.

## **Índice**

<b>1. INTRODUÇÃO</b>	<b>2</b>
<b>2. VISÃO GERAL</b>	<b>3</b>
<b>3. REQUISITOS DE SOFTWARE</b>	<b>3</b>
<b>3.1 Requisitos funcionais</b>	<b>3</b>
3.1.1 Referentes às regras do jogo	3
3.1.2 Referentes a outras funcionalidades do programa	4
<b>3.2 Requisitos não funcionais</b>	<b>4</b>
<b>4 REGRAS DE NEGÓCIO</b>	<b>5</b>

## 1. INTRODUÇÃO

O objetivo é o desenvolvimento de um jogo em que dois usuários possam competir entre si com a execução de decisões estratégicas com base nas regras, que podem ser encontradas no arquivo "Labyrinthics - Regras do jogo.pdf".

## 2. VISÃO GERAL

O programa será implementado em Python (versão 3.9.7) com uma arquitetura orientada a objetos. O jogo será baseado em turnos, apresentará uma interface bidimensional e será *stand-alone*.

## 3. REQUISITOS DE SOFTWARE

### 3.1 Requisitos funcionais

#### 3.1.1 Referentes às regras do jogo

**RF1 - Andar casas:** Para que os jogadores realizem suas jogadas, o programa deve permitir que estes andem casas no tabuleiro em seus respectivos turnos.

**RF2- Colocar blocos:** Para que os jogadores realizem suas jogadas, o programa deve permitir que estes coloquem blocos no tabuleiro em seus respectivos turnos.

**RF3 - Quebrar blocos:** Para que os jogadores realizem suas jogadas, o programa deve permitir que estes quebrem blocos dispostos no tabuleiro em seus respectivos turnos, observando RN1.

**RF4 - Jogadas:** A cada turno, o programa deve esperar que o jogador em turno efetue sua jogada, observando RN2 e RN3.

**RF5 - Ações indevidas:** O programa deve bloquear a ação caso ele tente realizar ações de jogada sobre uma casa proibida, observando RN4 e RN5.

**RF6 - Jogador trancado:** O programa deve checar, a cada ação no tabuleiro feita pelos jogadores, se algum deles está trancado, observando RN6. Caso verifique-se que um deles está, o oponente vence o jogo.

**RF7 - Jogador chega ao outro lado do tabuleiro:** Para a verificação da condição de vitória do jogo, o programa deve checar, a cada casa andada, se o jogador em turno chegou ao outro lado do tabuleiro. Caso verifique-se que um deles chegou, este ganha o jogo.

### ***3.1.2 Referentes a outras funcionalidades do programa***

**RF8 - Iniciar:** O jogo deve apresentar a opção de iniciar a partida no menu principal. A seleção desta opção prepara o jogo para uma nova partida, dispondo buracos aleatoriamente no tabuleiro e colocando ambos os jogadores em sua posição inicial, de acordo com RN7.

**RF9 - Sair:** O jogo deve apresentar a opção de sair no menu principal. Esta opção encerra o programa.

**RF10 - Pausar:** Durante uma partida o jogo deve apresentar a opção de pausa temporária do jogo, esta opção leva para um painel de “pausa”. No painel de pausa é possível continuar o jogo e retomar a partida.

**RF11 - Painel de fim de jogo:** Quando um jogador ganha o programa deve exibir o resultado e oferecer as opções de reinício da partida ou de retorno ao menu principal.

**RF12 - Reiniciar:** O programa deve exibir, no painel de pausa e no painel de fim de jogo, uma opção de reiniciar. Esta opção permitirá que a partida comece novamente, com um tabuleiro com uma nova disposição de buracos, sem blocos e jogadores na posição de início de jogo.

**RF13 - Retornar ao menu:** No painel de fim de jogo existirá uma opção de retornar ao menu que faz com que o jogo volte ao estado inicial no menu principal.

**RF14 - Encerrar partida:** No painel de pausa existirá uma opção de encerrar a partida que define que a partida acabou sem vencedores e leva para o painel de fim do jogo.

### 3.2 Requisitos não funcionais

**RNF1 - Em turnos:** O jogo deve ser em turnos.

**RNF2 - Quantidade de jogadores:** O jogo deve ter exatamente dois jogadores.

**RNF3 - *Stand-alone*:** O jogo deverá ser do tipo *stand-alone*.

**RNF4 - Linguagem de programação:** O jogo deve ser implementado em Python (versão 3.9.7).

**RNF5 - Interface gráfica:** O jogo deve ter uma interface gráfica única, compartilhada pelos usuários.

**RNF6 - Framework:** Será usado o framework de desenvolvimento de jogos em python *Pygame*.

### 4 REGRAS DE NEGÓCIO

**RN1** - Para que possa quebrar um bloco, o jogador deve esperar pelo menos 3 turnos seus desde a última quebra de bloco.

**RN2** - Uma jogada pode ter as seguintes composições: colocar um bloco no tabuleiro e andar uma casa; quebrar um bloco disposto no tabuleiro e andar uma casa.

**RN3** - Em uma jogada, o jogador deve primeiro colocar/quebrar um bloco no tabuleiro e somente depois andar uma casa.

**RN4** - Um jogador deve mover-se apenas uma casa por turno, sendo esta ortogonalmente adjacente à sua casa atual.

**RN5** - O jogador não pode mover-se para uma casa que possua um buraco, um bloco ou um outro jogador.

**RN6** - Um jogador é dito trancado se não consegue mover-se em seu turno.

**RN7** - A posição inicial dos jogadores é na metade vertical do tabuleiro, em bordas opostas.