

Projeto No Thanks!

Especificação de Requisitos de Software

REQ_PQ_jogoNoThanks_26Mar2023

Versão 2.1

18/06/2023

Versão	Autores	Data	Ação
1.0	Eduardo de Moraes, Lucas Pagotto Coutinho de Oliveira e Matheus Antonio de Souza	26/03/2023	Esboço inicial dos requisitos
2.0	Eduardo de Moraes, Lucas Pagotto Coutinho de Oliveira e Matheus Antonio de Souza	27/04/2023	Refinando requisitos funcionais com o modelo
2.1	Eduardo de Moraes, e Matheus Antonio de Souza	18/06/2023	Alterado RF de recebimento de determinação de partida e iniciar partida

Conteúdo:

1. Introdução	2
1.1 Objetivos do desenvolvimento	2
1.2 Referências	2
2. Visão geral	2
2.1 Arquitetura do programa	2
2.2 Premissas de desenvolvimento	2
3. Requisitos de Software	2
3.1 Requisitos funcionais	2
3.2 Requisitos não funcionais	4

1. Introdução

1.1 Objetivos do desenvolvimento

Desenvolvimento de um programa que suporte a disputa de partidas de No Thanks! na modalidade usuários contra usuário, online.

1.2 Referências

O manual do jogo pode ser encontrado [aqui](#);

Para ser mais prático, pode-se assistir um [vídeo tutorial](#).

2. Visão geral

2.1 Arquitetura do programa

Programa orientado a objetos, cliente-servidor distribuído.

2.2 Premissas de desenvolvimento

O programa deve:

- Ser implementado em Python;
- Ser executado em qualquer plataforma que possua o interpretador Python;
- Usar o framework DOG na execução distribuída;
- Usar a plataforma Visual Paradigm no desenvolvimento da documentação UML.

3. Requisitos de Software

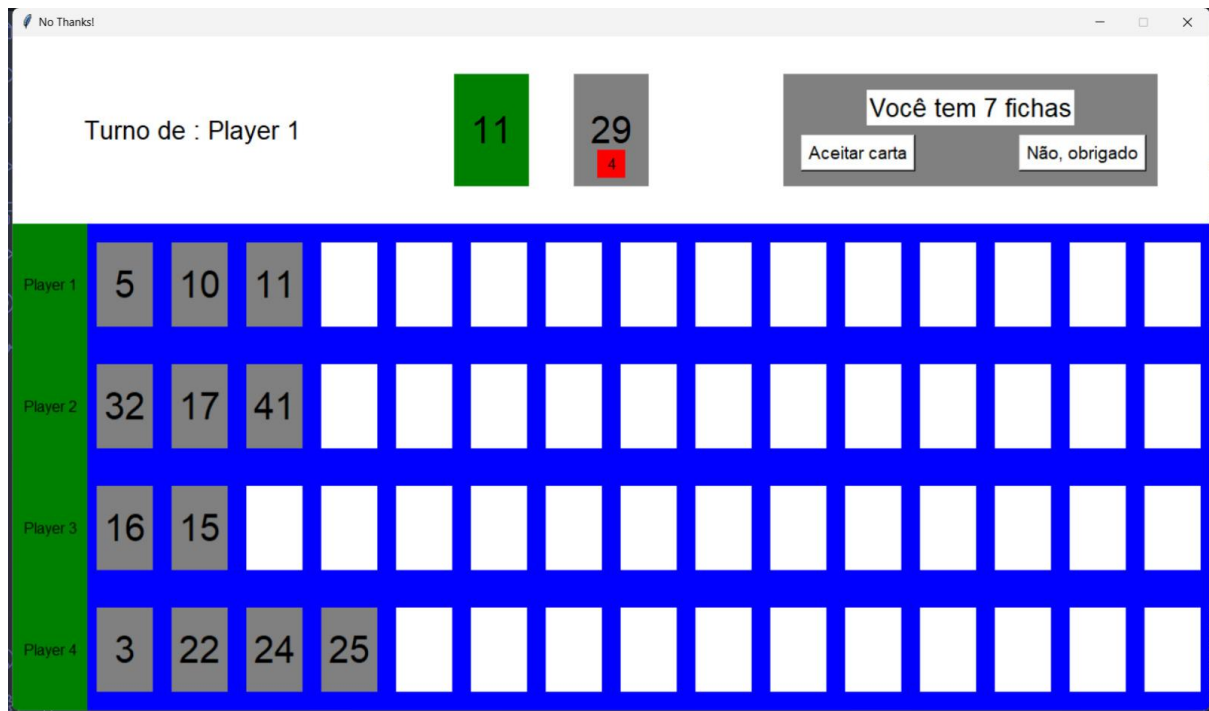
3.1 Requisitos funcionais

- Iniciar partida
 - Após a conexão de todos os 4 jogadores, um pede início da partida e então deverá ser criada a instância da partida junto ao DOG e aberta a tela de partida, em seguida as fichas são distribuídas, as cartas retiradas do baralho e uma carta é virada, como especifica as regras, e enviada para cada outro jogador para sincronização.
- Recebimento de determinação de início de partida
 - A interface DOG deve enviar aos jogadores em conexão que não iniciaram partida os valores iniciais definidos pelo DOG para sincronização do jogo.
- Receber Jogada
 - Com o jogador que não possui a vez, a interface DOG deve intermediar e receber a escolha do jogador da vez pela rede, reproduzindo-as no mesmo momento que acontecem e na tela de quem as recebeu.
- 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.

- Iniciar programa
 - O programa deve pedir o nome do jogador e apresentar a opção “iniciar jogo” na interface gráfica do jogo, onde o jogador deve ser colocado em uma tela de aguardo para apertar o botão de início de partida. A partida iniciará caso tenha êxito, caso aconteça uma falha, a aplicação deve fechar após apresentar o erro ao usuário.
- Recusar carta
 - O jogador que tiver ficha disponível pode adicionar uma ficha em cima da carta e passar a vez, deixando para o próximo jogador a escolha entre aceitar e recusar a carta.
- Aceitar carta
 - O jogador ao escolher aceitar a carta da mesa, deve pegá-la e junta-la a sua coleção de cartas, deixando a mostra para os outros jogadores, enquanto guarda as fichas que a carta carregava consigo. Vira nova carta para o próximo jogador. Se não há mais cartas no baralho após aceitar, entrar no estado de fim de jogo.

3.2 Requisitos não funcionais

- Versão do Python
 - O programa será desenvolvido na linguagem Python versão **3.10**
- IDE (ambiente de desenvolvimento integrado)
 - A interface de desenvolvimento escolhida para programar o jogo será o **Pycharm** e o **VSCode**.
- Bibliotecas Python
 - Será usado a biblioteca gráfica **tKinter**, por ser proprietário do Python.
- Interface gráfica
 - A interface gráfica do programa será produzida com base no esboço da imagem abaixo.



- Plataforma de execução
 - Deve ser possível executar em qualquer sistema operacional que comporte o interpretador do Python.
- Características de qualidade do Software
 - É esperado que o software seja capaz de garantir que não haja duplicidade de ações:
 - escolher mais de uma vez a ação de passar ou puxar a carta, quando o mesmo deve ser desativado;
 - usar mais de uma vez a ficha por rodada.
 - Evitar que o jogador possa escolher ou puxar uma carta durante o momento errado, como quando se espera o término do outro jogador;
 - O software deve manter sigilo da quantidade de fichas que cada jogador tem.