**Projeto RAW**

**Documento de Arquitetura de Software**

**Disciplina Projeto de Software**

**Versão<1.0>**

Objetivo deste Documento

Este documento tem como objetivo ser a documentação parcial da arquitetura do sistema, e deve conter, entre outros, o escopo, requisitos, padrões, visões, e todos os diagramas que as acompanham.

Entrega 1, da Disciplina Projeto de Software, Professora Vânia de Oliveira Neves,

Integrantes

Gabriel Ripper Agostinho Cordeiro

Gustavo Lauria Gonçalves da Silva

Henrique Martine Dias

Marcelo Valentino Maia Leite

**Sumário**

Sumário.................................................................................................................................................. 2

1. Descrição do escopo do sistema .......................................................................................................3

2. Descrição dos requisitos arquiteturais, objetivos e restrições da arquitetura …….............................8

3. Definição dos padrões arquiteturais adotados……………………………………...............................11

4. Atributos de qualidade, descrições e soluções…………………………………………………………..12

5. Documentação das visões arquiteturais...........................................................................................14

5.1 Diagramas de casos de usos, modelo conceitual e diagramas de sequência do sistema.............15

5.2 Diagrama com a visão geral do sistema……………………………….……………………………….19

6. Diagramas de estado ou atividades,.................................................................................................20

7. Diagramas de classe detalhado…………………………………………………………………………...21

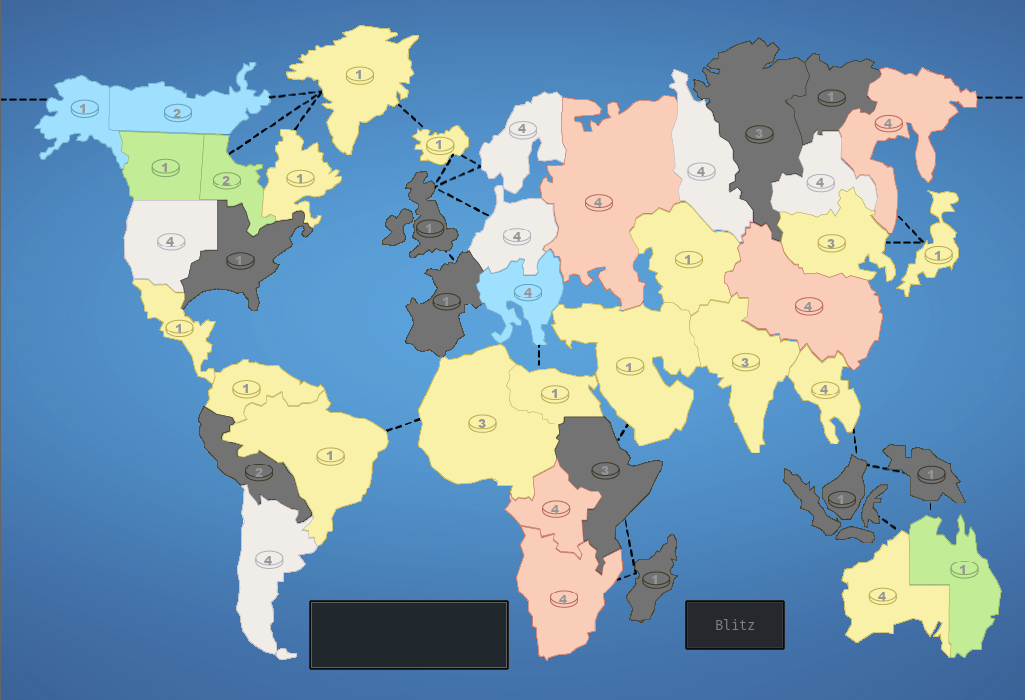
1. **Descrição do Escopo do Sistema**

**Introdução**

RAW é um jogo baseado no original de tabuleiro War, com elementos da sua versão estrangeira, o Risk. Os jogadores podem jogar tanto partidas single-player contra inteligências artificiais, quanto partidas multiplayer contra outros jogadores online.

**O Jogo**

RAW é um jogo de estratégia de tabuleiro, jogado sobre um mapa dividido em territórios. Cada jogador começa com a mesma quantidade de territórios, que são aleatoriamente distribuídos entre eles quando começa a partida. O objetivo principal do RAW é conquistar territórios, eliminando os exércitos de outros jogadores para dominar o mapa. Na nossa implementação, **jogam no máximo 6 jogadores,** cada um controlando uma cor**.**



O jogo acaba quando:

1. Um jogador conquistar todos os territórios e ser o último sobrevivente da partida;
2. Quando só houver um jogador, por motivos de desistência dos outros.

**Início**

Cada jogador começa com uma certa quantidade de exércitos por território. No início de cada turno, ele recebe uma quantidade adicional de exércitos para alocar em seus territórios como desejar, determinada por quantos territórios estão em sua posse, adicionado dos bônus das regiões que controla em sua totalidade. Essa é a **fase de distribuição de exércitos.**

**Ataque**

Distribuídos os exércitos, o jogo entra na **fase de ataque:** Durante essa fase, o jogador tem a opção de atacar territórios vizinhos a algum de seus territórios controlados por outros jogadores.

Antes de iniciar um ataque, o jogador deve selecionar o território a partir do qual deseja iniciar o ataque e o território que deseja atacar. O território de origem deve conter pelo menos dois exércitos para lançar um ataque, enquanto o território alvo deve ser adjacente ao território de origem. O atacante escolhe quantos exércitos deseja enviar para o território alvo. Ele pode escolher enviar o máximo possível, que é o número total de exércitos no território de origem menos um, ou uma quantidade menor se desejar.

**Rolagem de Dados**

O ataque é resolvido através do lançamento de dados. O atacante lança um dado para cada exército envolvido no ataque, com um máximo de três dados, e o defensor faz o mesmo.

Após os lançamentos de dados, cada jogador ordena seus dados em ordem decrescente.

Os resultados são emparelhados, com o maior dado do atacante sendo comparado ao maior dado do defensor, o segundo maior dado do atacante sendo comparado ao segundo maior dado do defensor, e o mesmo para o terceiro.



Em cada par, se o resultado do dado do atacante for maior, o defensor perde um exército. Se o resultado do dado do defensor for igual ou maior, o atacante perde um exército. Este processo é repetido até que todas as unidades de um dos lados tenham sido eliminadas ou até que o atacante desista de atacar.

Se o atacante conseguir eliminar todos os exércitos do defensor, ele conquista o território e deve mover pelo menos um exército do território de origem para o território recém-conquistado.

Se o atacante não conseguir conquistar o território alvo, os exércitos restantes no território de origem permanecem lá, e o ataque é encerrado.

**Ataque “Blitz”**

O ataque blitz é um ataque onde você ataca sucessivas vezes, com o máximo de exércitos possível, até conquistar o território alvo ou perder todos os exércitos. Esses ataques ocorrem automaticamente: Uma vez que é decidido, não há como voltar atrás, todos os ataques possíveis serão feitos. Esse ataque é pensado para os casos em que a vitória do atacante é muito provável, a fim de acelerar o jogo.

Em nossa versão, essa rolagem de dados acontece de forma automática, sem requerer ação do jogador atacado. Em seu turno, ou no caso de estar conectado à partida, poderá ver o resultado dos dados após o fim da ação de ataque.

**Fim da Fase de Ataque**

O jogador pode realizar quantos ataques desejar (e puder) durante a sua fase de ataque. Depois disso, o jogador opta por passar para a fase final, a **fase de reforço.**

**Reforço**

Nessa fase, o jogador pode reforçar um de seus territórios com exércitos de outro território seu, **desde que esses estejam ligados por outros territórios de sua posse.** O jogador tem total controle sobre a quantidade que deseja realocar, mantendo a regra de que cada território deve ter no mínimo um exército. Feito o reforço, a vez é passada para o próximo jogador, que irá passar pelas mesmas fases.

**Trocas**

Se um jogador realizou pelo menos um ataque bem sucedido na sua vez, ele recebe uma **carta de território.** Essas cartas contêm nelas um símbolo: Quadrado, círculo ou triângulo. Existem cartas coringas, que podem contar como **qualquer símbolo.**

Se o jogador juntar três cartas com símbolos diferentes, ou três cartas com símbolos iguais, ele tem direito a trocar essas cartas, em sua fase de distribuição, por exércitos adicionais. O número de exércitos por troca **aumenta com cada troca feita na partida.** A progressão é:

2, 4, 6, 8, 10, 12, 15, 20, 25...

O jogador não é obrigado a realizar a troca, mas pode trocar se desejar desde que esteja em sua vez e na fase de distribuição de exércitos.

**Sobre a Implementação**

No momento, já há uma implementação do jogo com toda a lógica necessária para jogar uma partida. No entanto, o jogo carece de:

* Uma **UI mais bem desenvolvida**, o que deixa a experiência de usuário confusa;
* **Qualquer opção de configuração**, e.g. mudar a dificuldade, escolher a cor com a qual vai jogar;
* Um **menu principal**;
* **Multiplayer local** e a lógica necessária para tal;
* **Multiplayer online** e a estrutura necessária para isso;
* **Opção de desistência de uma partida** por parte do jogador.
* **Música e efeitos sonoros;**

O projeto da disciplina visa o planejamento da construção dessas features, a fim de melhorar o jogo e adicionar a possibilidade de um multiplayer local e online.

**Modo Singleplayer**

Neste modo, os jogadores podem enfrentar oponentes controlados por IA, que agem de acordo com uma heurística para simular o comportamento de um jogador humano. No entanto, essa heurística, apesar de perfeitamente funcional, é bastante simples. Cogitamos uma melhoria na IA, no entanto, **não analisamos ainda se isso é factível**.

Como já foi colocado, o jogo já tem o modo singleplayer funcional. Além do que já está feito, pretendemos construir:

* Deixar o jogador escolher a cor com a qual vai jogar (e.g. preto, branco, azul…);
* Afinar a velocidade com que as IAs jogam (no momento, todos os movimentos da IA ocorrem muito rapidamente);
* Destacar pela UI de maneira mais clara de quem é o turno;
* Melhorias nos elementos de UI que já estão feitos;
* Dar ao jogador a opção de desistir de uma partida, se ele não quiser mais jogar;
* Sinalizar pela UI que o jogador venceu e deixar ele voltar para o menu principal;

As melhorias de UI e UX no modo singleplayer serão reaproveitadas para as outras modalidades, tanto para o multiplayer offline como o online.

**Modo Multiplayer Offline**

Similar ao modo singleplayer, com a diferença que são dois ou mais jogando na mesma máquina. Ou seja, o jogo vai receber as ações de dois jogadores a cada rodada, cada um em sua vez. Se um dos jogadores humanos desistir da partida, **uma IA assume a posição deixada por ele.** Isso ocorre até que haja somente um jogador, ponto a partir do qual o jogo irá se comportar como se estivesse no modo singleplayer.

**Modo Multiplayer Online**

O modo online permite que os jogadores joguem com outros jogadores em outros computadores, com o mínimo de dois jogadores por partida.

War segue a definição de um jogo sequencial, logo um jogador deve esperar seus oponentes terminarem seus turnos para poder jogar. O jogador, então, não precisa estar disponível todo o tempo durante a partida, e sim apenas quando é seu turno. Para minimizar o tempo que um jogador deve estar comprometido, permitiremos ao jogador a capacidade de jogar diversas partidas simultaneamente e/ou estar fora do jogo, realizando algum afazer pessoal, e receber notificações de quando deve retornar a partida e atuar. É opcional, então, permanecer dentro da partida. No caso de permanência, este sistema de notificação será o responsável por avisar ao cliente que o mesmo deve solicitar ao servidor novas informações sobre o estado do jogo.

Isso é possível devido a uma arquitetura de comunicação direta através de um **cliente-servidor** como base, e se utilizando do padrão **Publisher/Subscriber** para enviar e receber notificações sobre os estados das partidas.

**Menu Principal**

O menu principal é a primeira tela que aparece para os jogadores. A partir dela, o jogador terá acesso a todas as funcionalidades do jogo:

* Iniciar uma partida singleplayer, só com oponentes controlados por IA;
* Iniciar uma partida multiplayer local, com até 6 jogadores na mesma máquina;
* Iniciar uma partida multiplayer online:
  + Após escolher essa opção, o jogador poderá criar a sua própria sala onde um jogo irá ser jogado com até mais cinco jogadores, ou entrar em uma sala de outro jogador **onde a partida ainda não tenha sido iniciada;**
  + Toda sala onde a partida não tenha sido iniciada será listada na interface do cliente;
  + O jogador que cria uma sala é o **dono.** Ele espera pelo menos mais um jogador entrar e, assim que achar o número de jogadores suficiente, **começa a partida**. A partir desse momento, a sala se torna invisível para outros jogadores que queiram iniciar uma partida ou se juntar a uma.
* Acessar configurações, como o volume;
* Sair do jogo.

**2. Descrição dos requisitos arquiteturais, objetivos e restrições da arquitetura**

| **Requisitos** | **Descrições** |
| --- | --- |
| **Escalabilidade** | -A arquitetura deve ser projetada para dimensionamento horizontal, permitindo a adição fácil de mais servidores e publishers para lidar com aumentos de carga.  -A adição de novos servidores e publishers deve ser automaticamente |
| **Disponibilidade** | -O jogo deve ser altamente disponível, minimizando o tempo de inatividade planejado para atualizações e manutenção.  -Deve haver redundância em componentes críticos para garantir que uma falha de hardware ou software não resulte em uma interrupção significativa do serviço.  -Deve ser implementado um sistema de monitoramento proativo que identifique e resolva problemas antes que afetem a disponibilidade do serviço. |
| **Comunicação e Sincronização** | -As comunicações entre os clientes e o servidor devem ser eficientes e confiáveis, garantindo a sincronização precisa do estado do jogo entre todos os jogadores.  -O servidor deve ser capaz de processar comandos dos jogadores em tempo real e garantir que todas as ações sejam refletidas consistentemente para todos os participantes.  -As comunicações entre cliente e servidor devem ser otimizadas para minimizar o tráfego de rede.  -Deve ser implementado um sistema de rollback para lidar com discrepâncias de estado entre os clientes.  -Deve ser implementado um sistema de compressão de dados para reduzir a largura de banda necessária para a transmissão de informações entre o cliente e o servidor. |
| **Segurança** | -Todos os dados sensíveis dos jogadores devem ser armazenados de forma segura e em conformidade com regulamentações de privacidade.  -O sistema deve implementar medidas robustas de segurança para proteger contra trapaças, hacking e acessos não autorizados.  -Todas as comunicações devem ser criptografadas para proteger os dados dos jogadores.  -Deve ser implementado um sistema de autenticação de dois fatores para garantir a segurança das contas dos jogadores. |
| **Arquitetura de Rede** | -A arquitetura de rede deve ser otimizada para minimizar a latência e maximizar a estabilidade da conexão para jogadores em diferentes regiões geográficas. |
| **Manutenção e Atualizações** | -Deve ser possível realizar atualizações do jogo de forma incremental para minimizar o tamanho dos patches.  -Deve ser implementado um sistema de rollbacks para reverter atualizações problemáticas sem impactar a experiência do jogador.  -Deve ser possível aplicar patches e correções de bugs de forma transparente e eficiente.  -Deve ser implementado um sistema de roteamento de tráfego que redirecione os jogadores para servidores alternativos durante as atualizações programadas para minimizar o tempo de inatividade. |
| **Gerenciamento de Dados e Persistência** | -Deve ser realizado o backup regular dos dados do jogador para evitar perda de progresso.  -Os dados do jogo, como o estado do mundo virtual, devem ser persistentes .  -Deve ser implementado um sistema de armazenamento em banco de dados para lidar com volumes de dados de forma rápida e confiável. |
| **Experiência do Usuário** | -Deve ser implementado um sistema de chat seguro e moderado para garantir um ambiente de jogo saudável. |
| **Monitoramento e Análise** | -Deve ser implementado um sistema de monitoramento em tempo real para identificar gargalos de desempenho e problemas de infraestrutura.  -Deve ser realizado o registro e análise de métricas de jogo para entender o comportamento dos jogadores e melhorar a experiência. |

**RESTRIÇÕES**

1. A plataforma de desenvolvimento utilizada deve ser compatível com as tecnologias e frameworks específicos necessários para suportar a jogabilidade online.
2. A linguagem de programação escolhida deve ser aquela mais adequada para desenvolver a lógica de jogo e interações multiplayer em tempo real.
3. O servidor de aplicação deve ser configurado de acordo com as exigências de desempenho e segurança do jogo, conforme definido pelas especificações técnicas.
4. O sistema operacional do servidor deve ser compatível com os requisitos de software e segurança, e preferencialmente ser de código aberto para minimizar custos de licenciamento.
5. Os servidores que hospedam o jogo devem ser configurados em um ambiente clusterizado para garantir alta disponibilidade e escalabilidade.
6. O jogo deve ser projetado e desenvolvido com recursos de acessibilidade para garantir que os jogadores com deficiências visuais possam participar plenamente da experiência.
7. O sistema de autenticação e autorização deve ser implementado de acordo com os padrões de segurança.
8. O jogo deve ser compatível com uma variedade de navegadores web, incluindo, mas não se limitando à, Google Chrome, Mozilla Firefox, Microsoft Edge e Safari, garantindo uma ampla acessibilidade para os jogadores.

**OBJETIVOS**

1. **Criar uma experiência de jogo viciante e envolvente:** O jogo deve ser divertido e desafiador o suficiente para manter os jogadores entretidos e voltando para mais.
2. **Fornecer um fluxo constante de novos conteúdos:** Novos conteúdos, como mapas, modos de jogo e eventos, devem ser adicionados regularmente para manter os jogadores interessados.
3. **Monetizar o jogo de forma eficaz:** O jogo deve ter um modelo de monetização que seja sustentável a longo prazo. Isso pode ser feito por meio de assinaturas, microtransações ou outros meios.
4. **Construir uma comunidade forte:** Uma comunidade forte de jogadores pode ajudar a promover o jogo e mantê-lo vivo.
5. **Garantir que o jogo seja justo para todos os jogadores:** Todos os jogadores devem ter as mesmas chances de vencer, independentemente de seu nível de habilidade ou status de pagamento.

**3. Definição dos padrões arquiteturais adotados**

**Cliente**

* Um cliente por jogador, responsável pela renderização do jogo e da interface e por enviar os comandos (i.e. reforços, ataques, movimentações) que ele deseja realizar no jogo para o **servidor,** no caso de uma partida multiplayer**;**
* O clientetambém faz uma validação das ações do jogador antes de enviar para o servidor, para evitar o envio de mensagens que representem movimentos ilegais;
* Responsável por calcular e gerenciar as ações das IAs no modo singleplayer, evitando dependência de internet.
* Recebe notificações do **Sistema de Notificação** avisando-o das ações dos outros jogadores.

**Servidor**

* Responsável por processar os comandos enviados pelos jogadores via cliente e retornar para todos os membros da partida os resultados do comando do jogador que está em sua vez;
* Gerencia todas as partidas que estão ocorrendo no momento. Para cada partida, é criada uma **sala,** onde é temporariamente guardado o estado atual da partida (ou seja, como o tabuleiro está, de quem é a vez atualmente, e se há algum vencedor) e cor escolhida dos jogadores humanos daquela partida que **ainda estão jogando**;
* Após o fim do jogo (por desistência de todos os jogadores humanos ou até um jogador vencer), a **sala é destruída**. Caso só um jogador fique na partida, o jogo continua normalmente.
* Uma sala também é destruída **em caso de vitória de um jogador**;

**Sistema de Notificações**

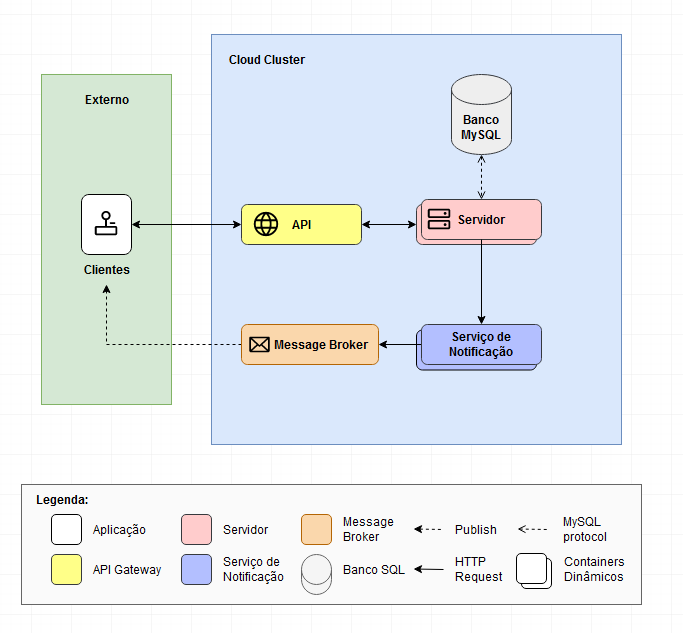
* Devido a possível duração de um jogo de War ser na casa das horas, é inviável pensar que os jogadores vão ter disponibilidade de estarem presentes e disposição de jogar durante horas sem pausa.
* Assim, no momento que o jogador realiza seu turno, se entra em um modo de “Espera”, em que deve aguardar o oponente realizar sua jogada. Nesse modo, o jogador está liberado para sair do jogo ou começar uma nova partida.
* Quando seu oponente jogar, ou um período de tempo máximo de espera for excedido, o jogador receberá uma notificação no menu principal do jogo, informando-o que deve atuar novamente. Esta mensagem contém apenas os dados necessários para que o jogador possa consultar o servidor e buscar o estado do jogo.

**4. Atributos de qualidade, descrição e solução**

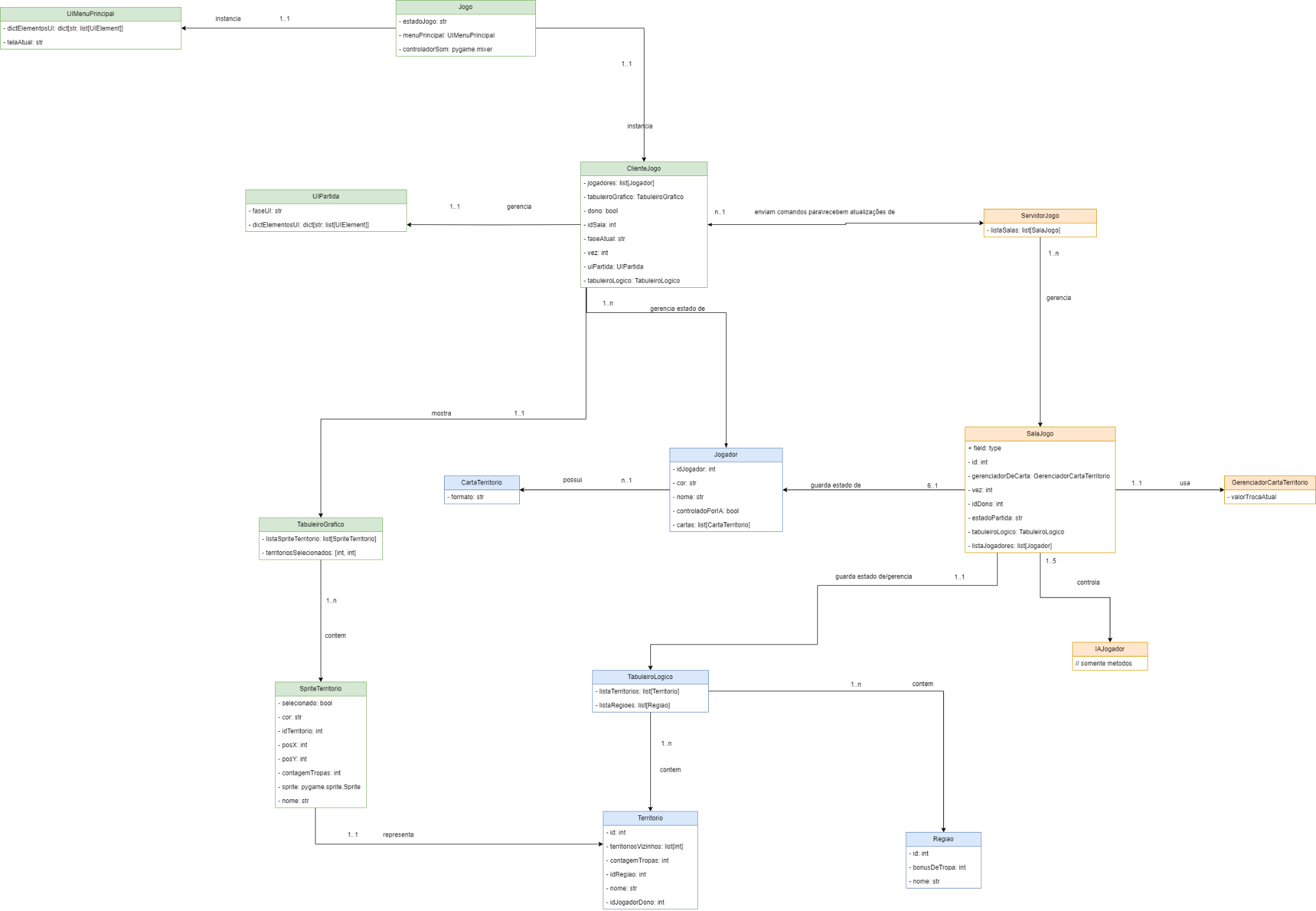
| **Atributos** | **Descrições** | **Soluções** |
| --- | --- | --- |
| **Escalabilidade e Desempenho** | -Dada a necessidade de dimensionamento horizontal para lidar com aumentos de carga e a exigência de alta disponibilidade, a arquitetura deve ser projetada para garantir um desempenho eficiente e escalável. Isso inclui a capacidade de adicionar facilmente novos servidores e publishers, minimizar o tempo de inatividade planejado e garantir uma comunicação eficiente e confiável entre os clientes e o servidor. | -A arquitetura foi projetada com uma abordagem de dimensionamento horizontal, permitindo a adição fácil de mais servidores e publishers para lidar com aumentos de carga. Isso permite que o sistema se ajuste dinamicamente à demanda dos jogadores, garantindo um desempenho eficiente e escalável, mesmo durante picos de tráfego intenso. |
| **Disponibilidade e Tolerância a Falhas** | -A disponibilidade contínua do serviço é crucial para garantir uma experiência de jogo ininterrupta. Portanto, a arquitetura deve ser projetada com redundância em componentes críticos e implementar um sistema de monitoramento proativo para identificar e resolver problemas antes que afetem a disponibilidade do serviço. | -A arquitetura incorpora redundância em componentes críticos, garantindo que uma falha de hardware ou software não resulte em uma interrupção significativa do serviço. Além disso, foi implementado um sistema de monitoramento proativo que identifica e resolve problemas antes que afetem a disponibilidade do serviço, garantindo uma alta disponibilidade do jogo. |
| **Segurança** | **-**Devido à natureza sensível dos dados dos jogadores e à necessidade de proteger contra trapaças e acessos não autorizados, a segurança é um atributo de qualidade fundamental. A arquitetura deve garantir que todos os dados sensíveis sejam armazenados de forma segura, implementar medidas robustas de segurança, como criptografia e autenticação de dois fatores, e garantir que todas as comunicações sejam criptografadas. | -A arquitetura prioriza a segurança dos dados dos jogadores, armazenando-os de forma segura e em conformidade com regulamentações de privacidade. Medidas robustas de segurança, como criptografia e autenticação de dois fatores, são implementadas para proteger contra trapaças, hacking e acessos não autorizados, garantindo a integridade e confidencialidade dos dados dos jogadores. |
| **Portabilidade** | -Para garantir uma ampla acessibilidade para os jogadores, especialmente aqueles com deficiências visuais, a arquitetura deve ser compatível com uma variedade de navegadores web e dispositivos. Isso garante que o jogo possa ser acessado e jogado sem problemas em diferentes plataformas, aumentando assim sua acessibilidade e alcance. | -A arquitetura é compatível com uma variedade de navegadores web e dispositivos, garantindo uma ampla acessibilidade para os jogadores. Isso significa que o jogo pode ser acessado e jogado sem problemas em diferentes plataformas, aumentando assim sua acessibilidade e alcance entre os jogadores. |

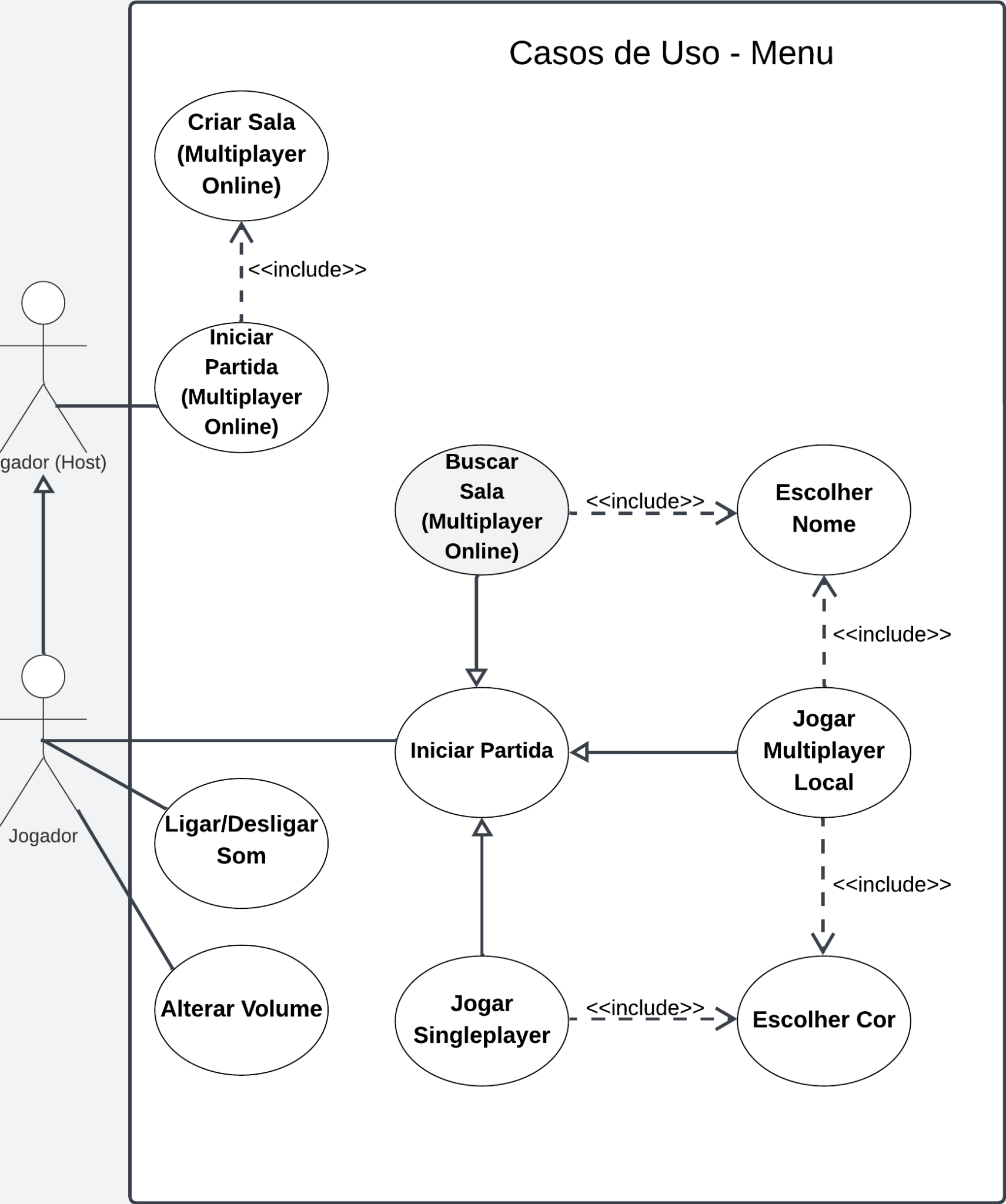
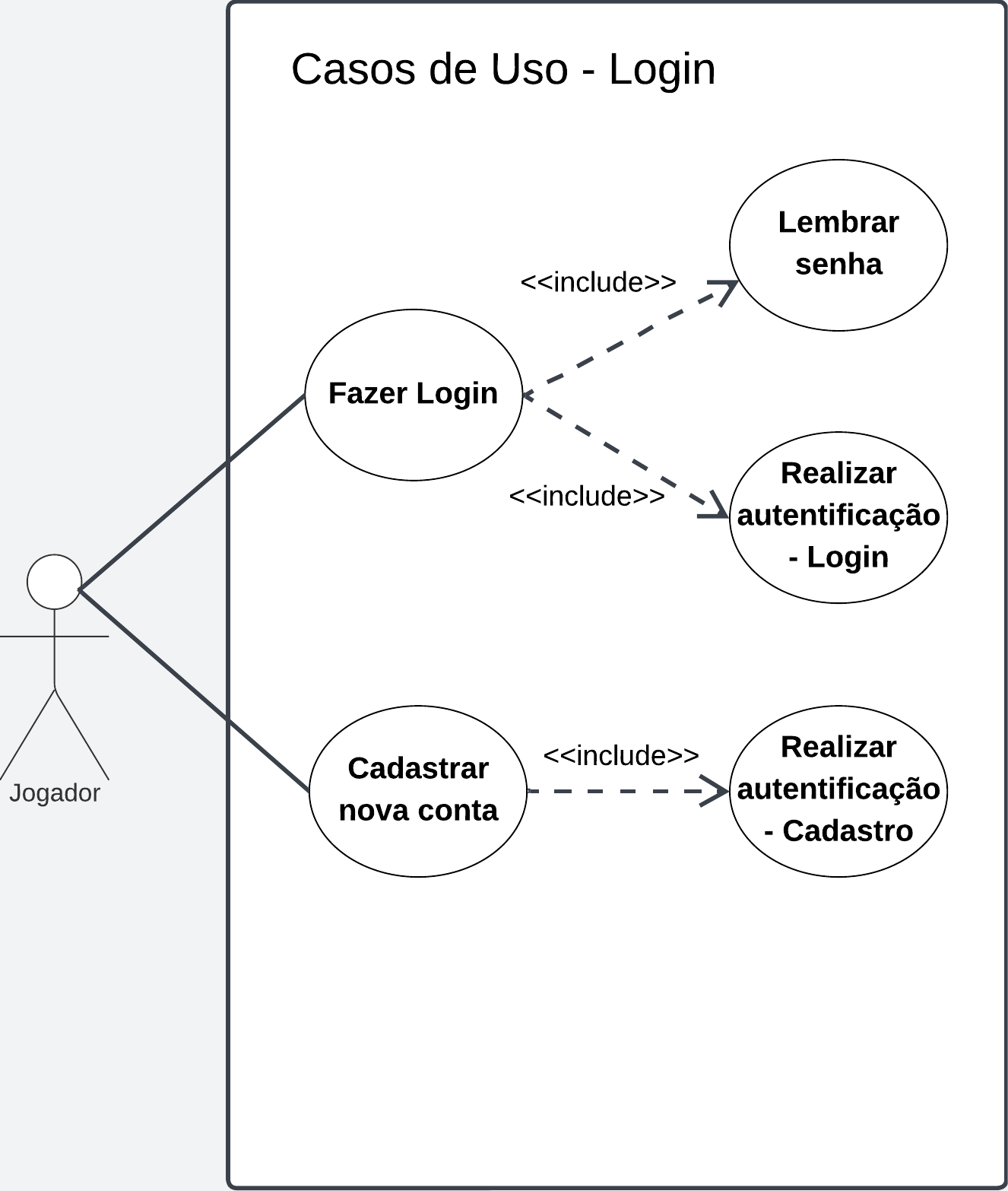
**5. Documentação das visões arquiteturais**

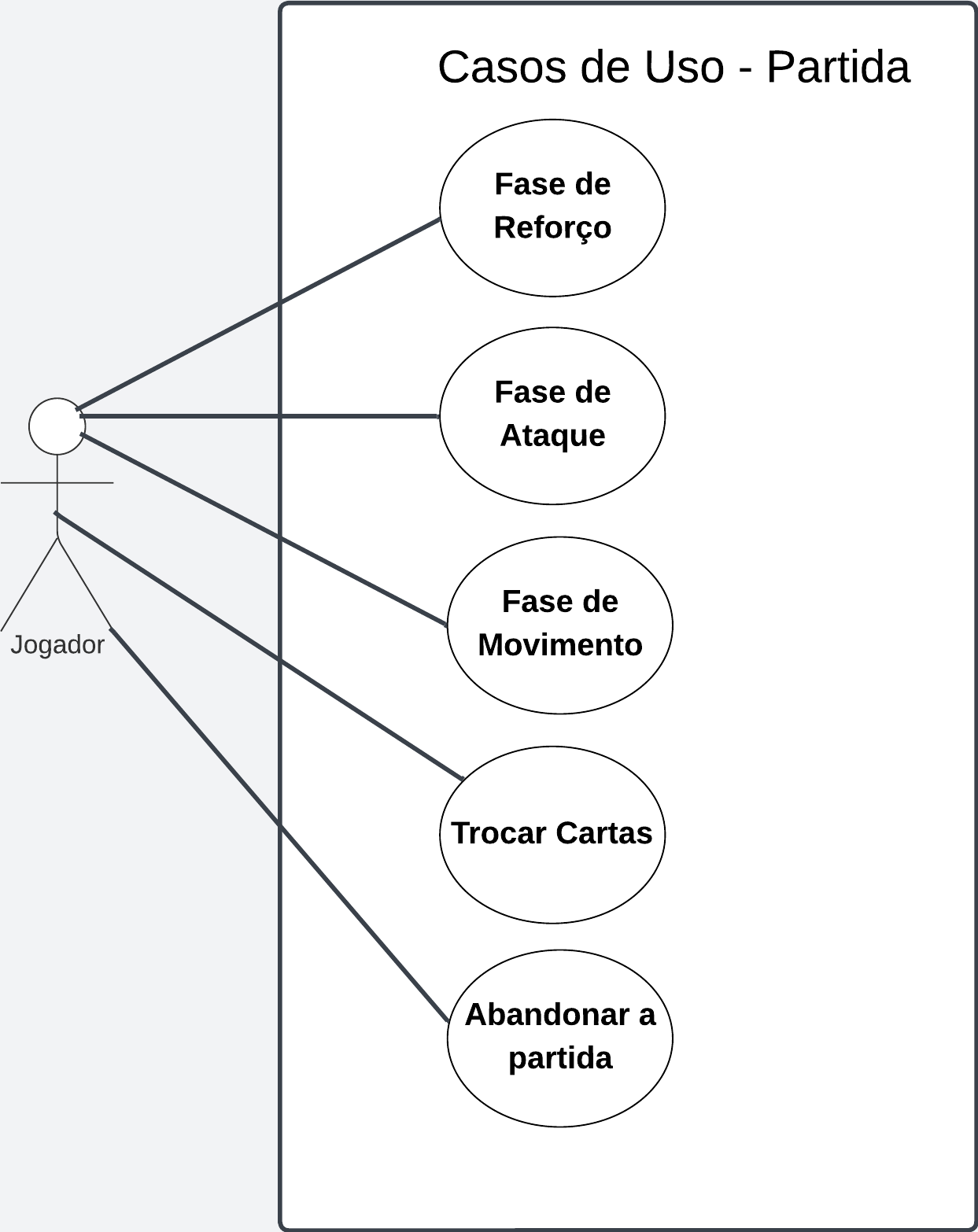
**VISÕES ARQUITETURAIS**

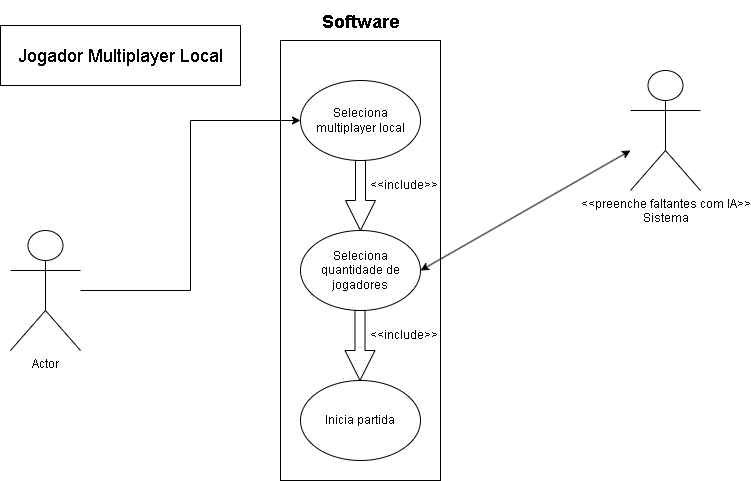
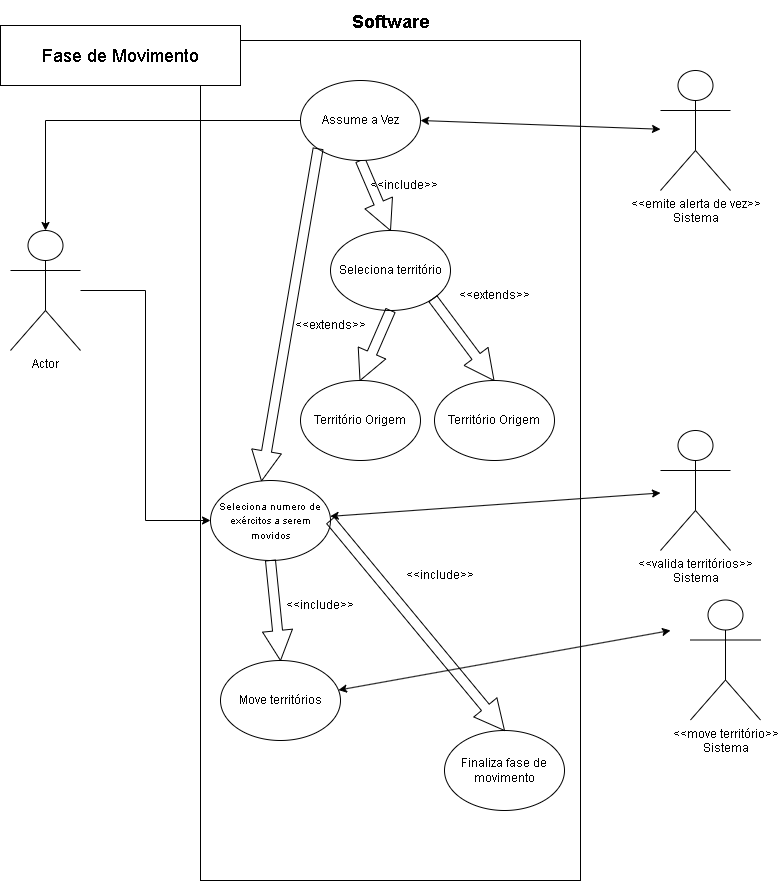
**Alto-nível:**

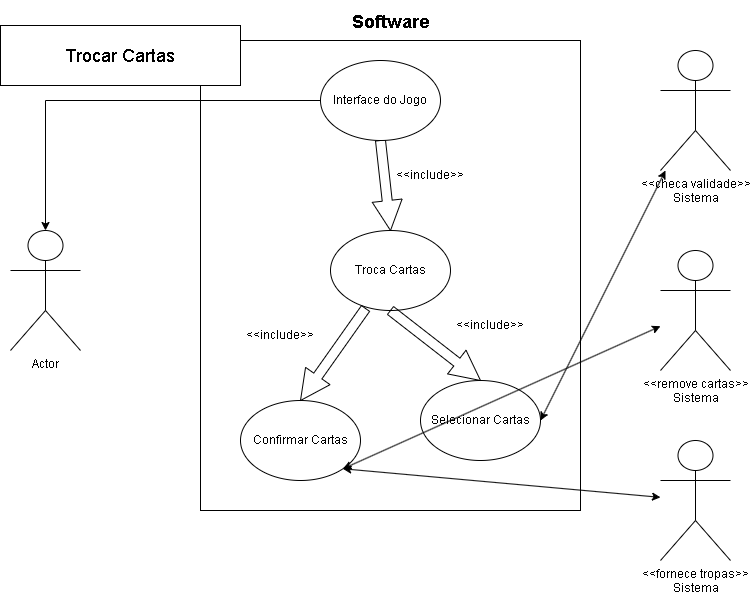
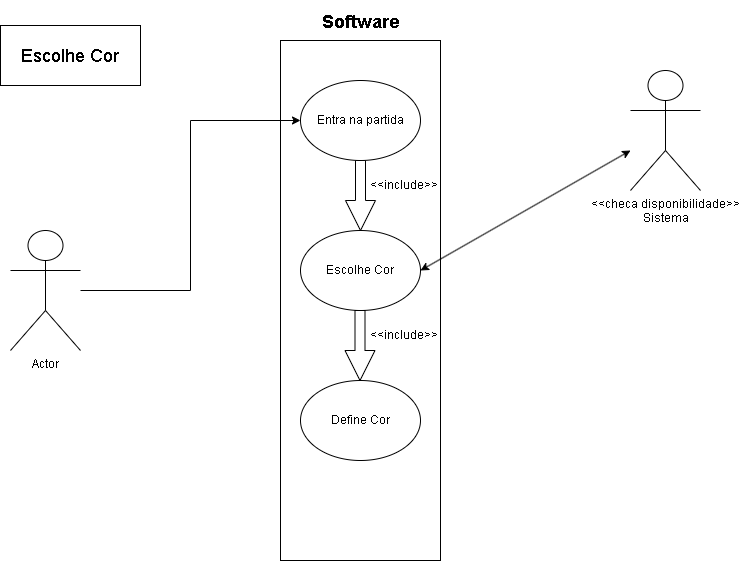
**5.1 Diagramas de casos de usos, modelo conceitual e diagramas de sequência do sistema**

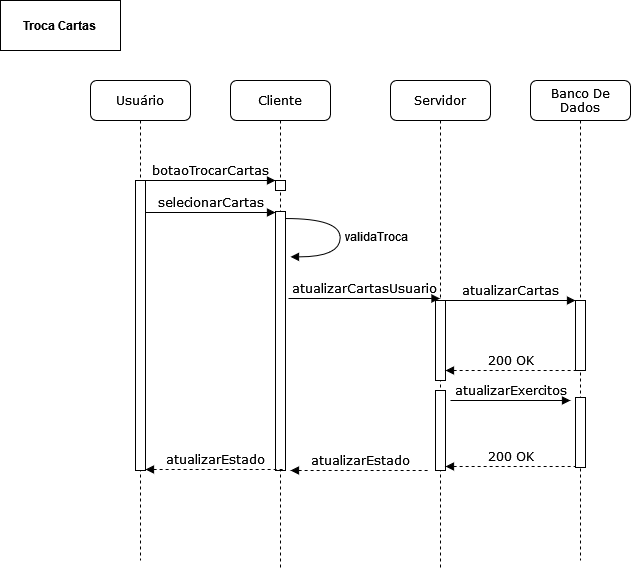
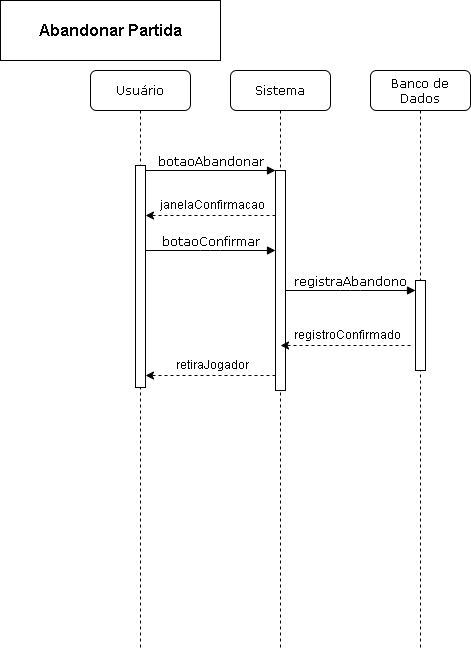
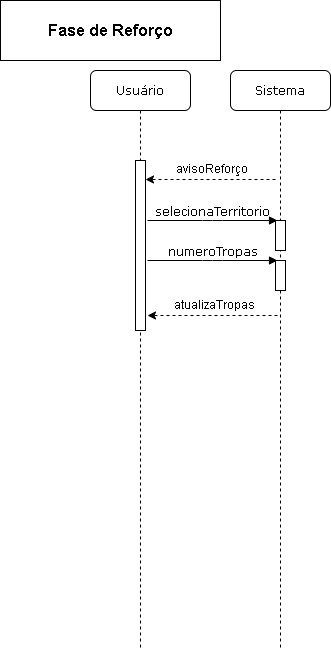
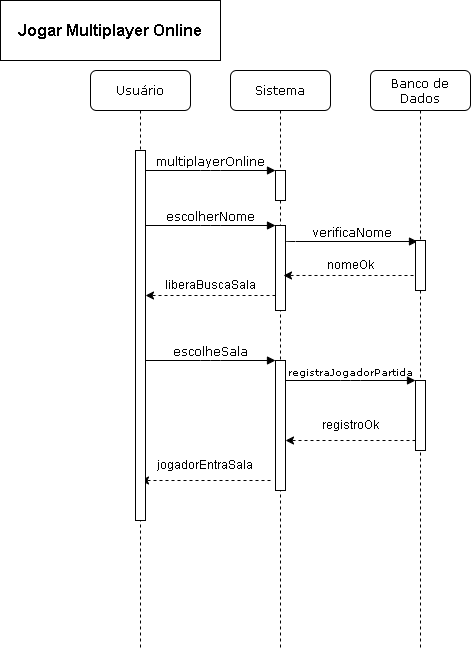
****

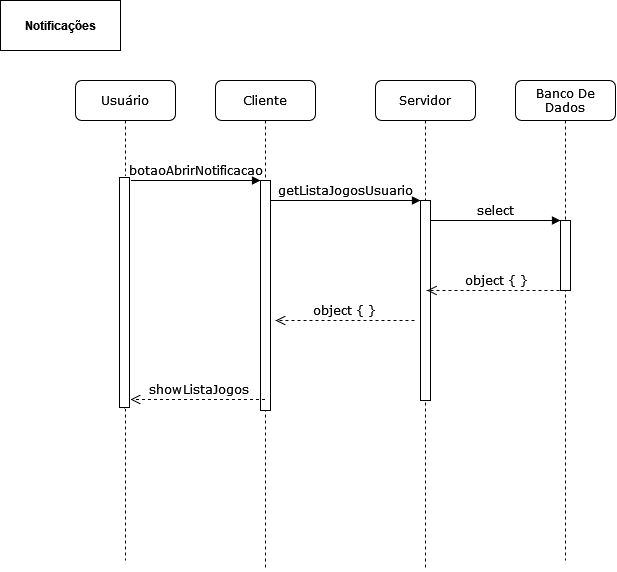


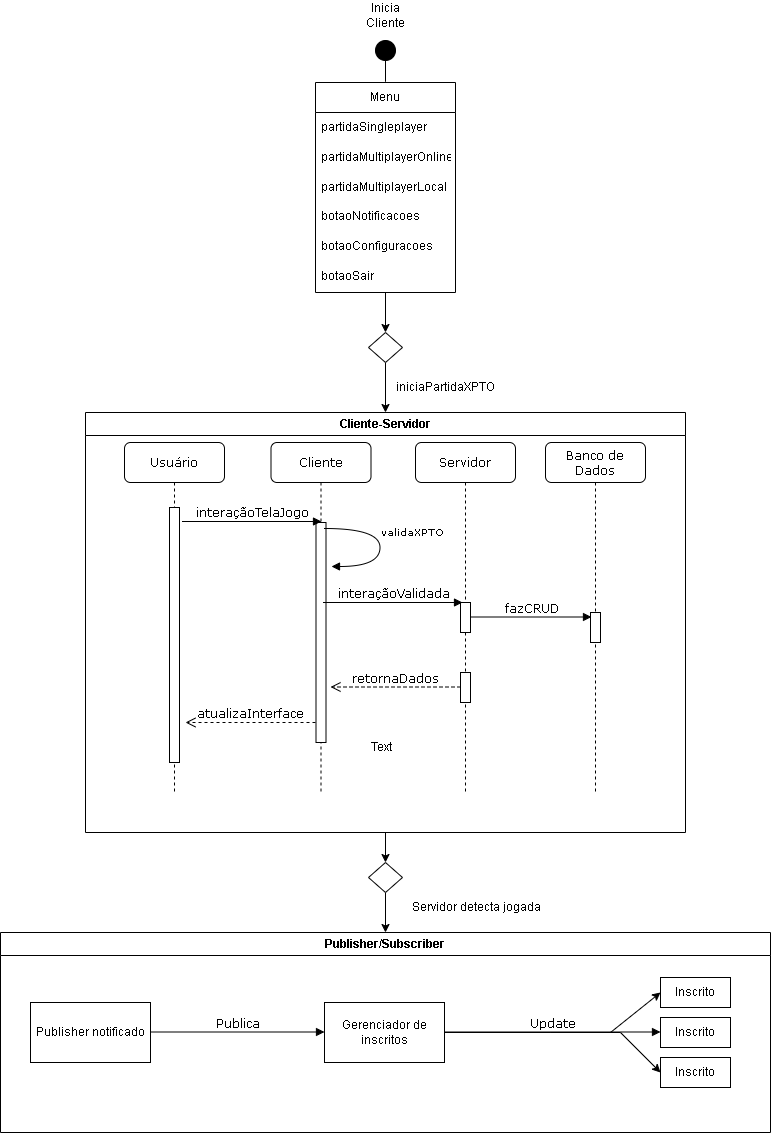




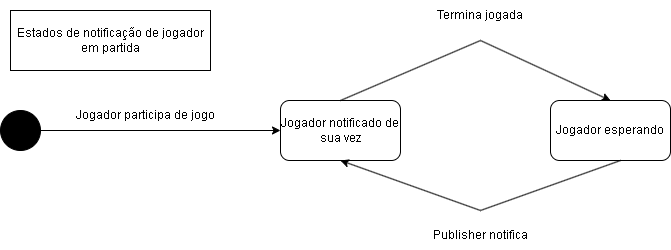


****

****

**5.2 Diagrama com a visão geral do sistema**

**6. Diagramas de estado ou atividades**

****

**7. Diagramas de classe detalhado**

**Link para o arquivo draw.io para melhor visualização** (inclui tanto o detalhado quanto o conceitual)**:** [modeloConceitual.drawio](https://drive.google.com/file/d/13dN1T95KDauRg6q99PAemzyEUE8VKPZ4/view?usp=sharing)

