Sistema
Primordial Soup

MAC-5853: Desenvolvimento de Sistemas de Computação

Banca examinadora: Flávio Soares Corrêa da Silva
Alfredo Goldman
Marco Aurélio Gerosa

Aluna: Maysa Malfiza Garcia de Macedo
Orientador: Marcel Parolin Jackowski

17-08-2009

Sumário

Lista de Figuras

Resumo

Este documento apresenta a documentação referente à primeira fase da avaliação exigida pela disciplina MAC-5853: Desenvolvimento de Sistemas de Computação do Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP). O projeto consiste no jogo *Primordial Soup* [Doris & Frank, 1997], o qual simula o comportamento de amebas nos primórdios da vida na Terra.

Este projeto consiste de dois sistemas e neste documento eles são modelados por meio de diagramas e descrições de casos de uso, diagramas de classes, modelo de dados, um diagrama de componentes e um diagrama de instalação.

Capítulo 1

Introdução

A modelagem deste projeto envolve a implementação de dois sistemas: o cliente e o servidor. O jogo *Primordial Soup* foi projetado para ser realizado por quatro jogadores acessando o sistema cliente em máquinas diferentes e todos eles acessando o sistema servidor por meio de uma rede.

Trata-se de um jogo de tabuleiro que simula o comportamento de amebas nos primórdios da vida na Terra. Em um breve resumo, o jogo pode ser descrito da seguinte forma: O jogo é dividido em fases e rodadas. A cada rodada os jogadores passam por 6 fases: movimento e alimentação, defeitos genéticos, novos genes, divisão de células, morte e evolução. Cada jogador pode gerenciar até sete amebas no tabuleiro, pode comprar cartas gene, receber pontos biológicos e pontos na estrada de pontuação. Cada ameba pode comer nutrientes, deslizar conforme a bússola, pode movimentar-se aleatoriamente a partir da rolagem de um dado, pode matar outra ameba e também se defender e ficar com fome. Ao adquirir uma nova carta gene, o jogador dá às suas amebas uma nova habilidade e por consequência, as regras para essas amebas são ligeiramente modificadas. Um espaço do tabuleiro pode armazenar várias amebas e vários nutrientes. Uma partida deve conter quatro jogadores e por sua vez um jogador pode jogar várias partidas. Cada partida tem até onze cartas ambiente (bússola), e a cada rodada essa carta ambiente é trocada. Uma ameba morre se contiver muitos pontos prejudiciais. Esses pontos são adquiridos quando se passa fome ou quando é atacada por outra ameba. Para uma descrição mais detalhada sobre as regras, veja as seções de descrição de casos de uso nos capítulos 3 e 4.

Cada jogador deve utilizar um sistema cliente local e cada uma de suas requisições na interface gráfica, tais como movimentações de tabuleiro, requisições de histórico, requisições de interrupção ou reinício de uma partida e requisições dos maiores vencedores devem ser resolvidas pelo sistema servidor. A linguagem de programação Java foi escolhida para este projeto e o MySQL foi escolhido como o sistema de banco de dados a ser utilizado. Para a documentação deste projeto foram utilizados alguns diagramas UML, cujos conceitos foram consultados em Fowler [2005]; Larman [2002] e Bezerra [2007]. Além disso, precisaremos implementar alguns padrões de projetos [Freeman *et al.*, 2009].

Este relatório está organizado da seguinte maneira: no Capítulo 2 mostramos a arquitetura criada para o desenvolvimento do sistema de computação. No Capítulo 3 mostramos alguns diagramas UML que esclarecem como o sistema cliente vai ser desenvolvido e no Capítulo 4 mostramos como o sistema servidor foi documentado, bem como, seu modelo de banco de dados. Finalmente, no Capítulo 5 discutimos sobre os resultados dessa primeira fase.

Para obter informações de como executar o sistema Primordial Soup veja o arquivo README.txt presente na raiz do diretório do projeto.

Capítulo 2

Arquitetura do sistema

2.1 Arquitetura Cliente-Servidor

Há vários meios de se resolver o problema da comunicação cliente-servidor. A princípio foi cogitado utilizar Socket e enviar arquivos XML entre os sistemas e sempre fazer um *parse* para resgatar as informações, mas seria bem trabalhoso, haja vista que neste projeto as informações trafegadas diferenciam muito dependendo sempre da etapa do jogo. A melhor solução seria trafegar diretamente objetos pela rede e uma forma de fazer isso é utilizar a tecnologia RMI (*Remote Method Invocation*) do Java, que permite invocar métodos de um sistema em uma máquina remota. Com certeza existem outros meios mais sofisticados de fazer isso, mas para este projeto o RMI funciona muito bem. Para que o RMI funcione, é necessário que tanto o cliente quanto o servidor possuam as mesmas assinaturas de método, ou seja, que possuam a mesma classe de interface e os mesmos objetos de transferência. Esses objetos que trafegarão entre o cliente e o servidor devem ser serializáveis, ou seja, devem se permitir o empacotamento e o desempacotamento durante o transporte pela rede. Neste projeto, esse tipo de objeto é identificado com as letras *TO* no final do nome de sua classe. Baseado nessa tecnologia, foram elaborados os diagramas de componentes e de instalação, os quais esclarecem a arquitetura deste projeto de sistema cliente-servidor, mostrados nas Figuras 2.2 e 2.1.

Segundo Fowler [2005], o diagrama de instalação mostra o *layout* físico de um sistema, revelando quais partes do software são executados em quais partes do hardware. Neste projeto, precisamos estabelecer como o cliente vai se comunicar com o servidor e qual componente dentro do servidor receberá objetos de transferência e se um dos sistemas conecta ou não banco de dados.

Fowler [2005] também diz que podemos usar o diagrama de componentes se quisermos dividir nosso sistema em componentes e mostrar seus relacionamentos por intermédio de interfaces e camadas. O componente *Stub*, que é gerado pelo RMI, mostrado na figura 2.2, representa a classe que auxilia o cliente a acessar um método remoto que por sua vez está implementado no componente

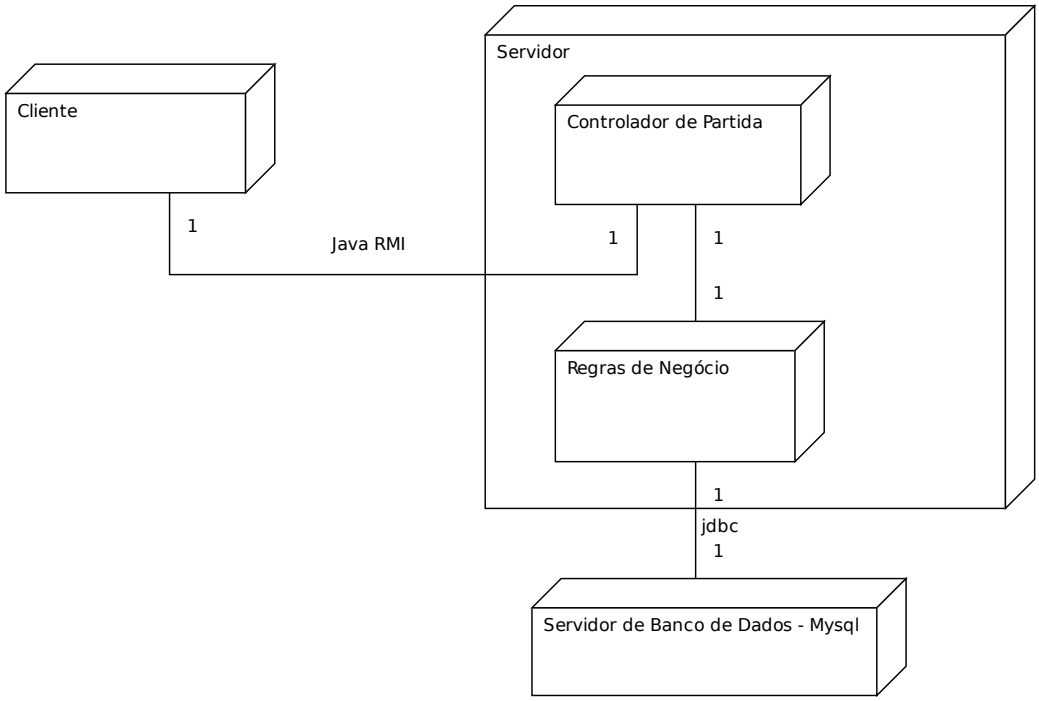

Figura 2.1: Diagrama de instalação.

ControladorPartidaServidor e este por sua vez pode acessar métodos dos objetos de negócio, ou seja, que contém as regras do sistema. Já a criação de dois componentes: um para conexão a um banco de dados e outro para implementar os métodos de acesso aos dados, facilita a mudança do sistema de banco de dados.

As seções 3.2 e 4.2 mostram como essa tecnologia funciona classe a classe.

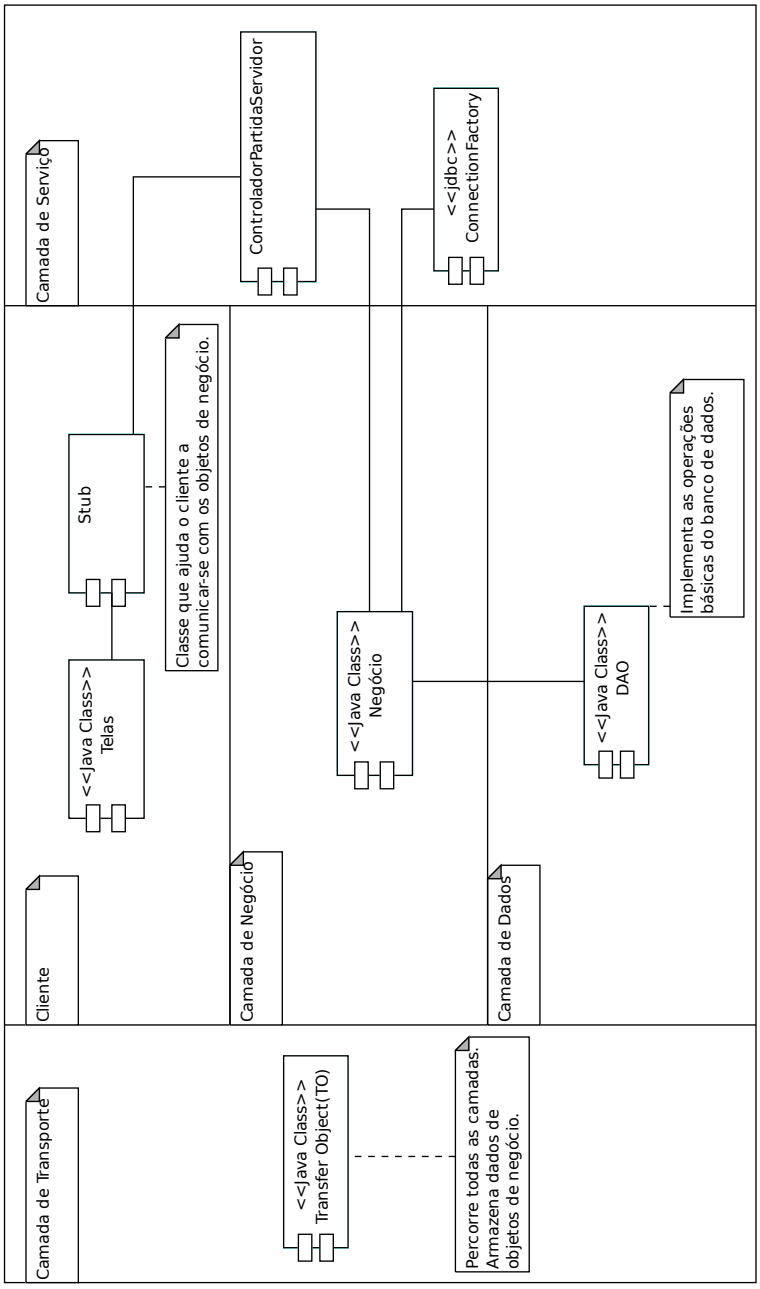

Figura 2.2: Diagrama de componentes.

Capítulo 3

Sistema cliente

Neste projeto, o sistema cliente é magro, ou seja, apenas recebe requisições feitas pelo jogador através da interface gráfica e envia as requisições para serem tratadas pelo sistema servidor. Na Seção 3.1 mostramos o diagrama de caso de uso do sistema e sua respectiva descrição, já na Seção 3.2 mostramos os diagramas de classes que fornecem uma noção mais técnica do sistema.

3.1 Descrição de casos de uso

A Figura 3.1 mostra o diagrama de caso de uso aplicável ao sistema cliente, cuja descrição detalhada está contida nas seções a seguir. Esta descrição cobre toda a regra do jogo e mostra primeiramente uma sequência típica de eventos e logo após, se for cabível, eventos alternativos.

3.1.1 Caso de uso: solicitar jogar

Ator: Jogador.

Finalidade: Adiciona jogador na próxima partida ou devolve as partidas abertas interrompidas por ele.

Visão geral: Um jogador loga-se a fim de participar da próxima partida ou de uma partida aberta que foi interrompida por ele. Caso ainda não esteja cadastrado, o cadastramento deve ser feito.

Tipo: Primário.

Sequência Típica de Eventos:

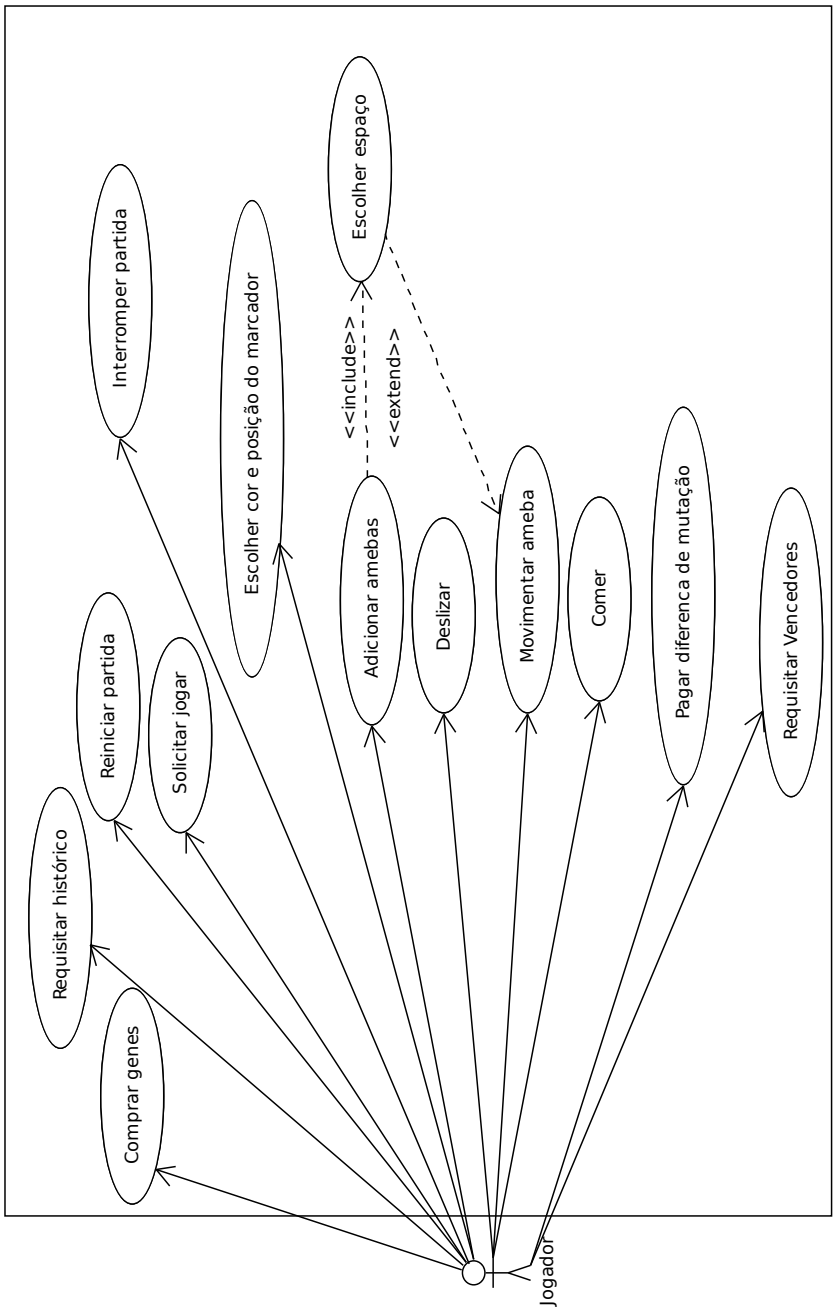

Figura 3.1: Diagrama de casos de uso para o sistema cliente.

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um jogador deseja participar de uma partida.	
2- O jogador insere seu nome e senha, além de informar se deseja reiniciar uma partida aberta e se é novo jogador ou não.	3- Registra o jogador. Para uma nova partida: Caso haja 3 jogadores esperando, então inicializa a partida. Caso contrário, envia uma mensagem de espera. Para reiniciar uma partida: Mostra uma lista de partidas abertas que foram interrompidas pelo jogador.
	4- Se o tempo de espera passar do limite, o sistema envia uma mensagem: "Tempo de espera expirou".
5- Rola 2 dados	6- Calcula por meio dos resultados dos dados, do primeiro ao quarto lugar.

Alternativa do item 3: Caso o jogador não seja cadastrado e o jogador não tenha indicado um novo cadastro, mostra uma mensagem de erro.

Alternativa do item 3: Caso o jogador rebeba uma lista de partidas abertas, inicia-se o caso de uso *Reiniciar Partida*.

3.1.2 Caso de uso: interromper partida

Ator: Jogador.

Finalidade: Paralisar a partida corrente.

Visão geral: Um dos quatro jogadores resolve paralisar a partida. Os outros jogadores ficam impedidos de continuar a partida.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um dos quatro jogadores resolve paralisar a partida por um tempo indeterminado.	

Continua na próxima página.

Continuação.

Ação do ator	Resposta do sistema
2- O jogador informa a paralisação por meio da interface do sistema.	3- Paralisa a partida e impede qualquer outro jogador de continuar.
	4-permite que o jogador que solicitou a paralisação possa reiniciá-la num período de tempo.

3.1.3 Caso de uso: reiniciar partida

Ator: Jogador.

Finalidade: Reiniciar uma partida interrompida.

Visão geral: O jogador que interrompeu a partida solicita a reinicialização da mesma partida do mesmo ponto que foi paralisada.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o jogador que interrompeu uma partida resolve reiniciá-la.	
2- O jogador informa a reinicialização da partida por meio da interface.	3- Devolve o controle da partida aos jogadores exatamente do ponto que foi paralisada.

3.1.4 Caso de uso: escolher cor e posição de marcador

Ator: Jogador.

Finalidade: Registrar a cor e a posição do marcador do jogador.

Visão geral: O jogador escolhe a cor que deseja usar na partida, bem como a posição do marcador, caso seja o último a escolher, fica com a cor que resta.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando recebe a permissão para escolher a cor de sua ameba e a posição inicial de seu marcador.	2- Mostra as cores disponíveis.
3- O Jogador escolhe uma cor.	4- Registra cor.
5- O jogador escolhe a posição disponível entre 1 e 4.	6- Registra a posição escolhida.

3.1.5 Caso de uso: escolher espaço

Ator: Jogador.

Finalidade: O jogador escolhe em qual espaço sua ameba ficará.

Visão geral: O jogador deseja colocar uma ameba no tabuleiro ou movê-la e escolhe um espaço para esta.

Tipo: Secundário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o jogador deseja pôr uma nova ameba no tabuleiro ou mover a ameba por meio de uma direção livre.	2- Verifica quais são as posições possíveis para colocar a ameba. E mostra todos os espaços habilitados.
3- O jogador escolhe qual espaço deseja.	4- Registra a nova posição da ameba.

3.1.6 Caso de uso: deslizar ameba

Ator: Jogador.

Finalidade: Levar a ameba para a direção indicada na bússola.

Visão geral: O jogador aceita que sua ameba seja levada para a direção especificada na bússola.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o jogador aceita que sua ameba seja levada para a direção especificada na bússola.	2- Registra a nova posição da ameba de acordo com a direção da bússola.

3.1.7 Caso de uso: movimentar ameba

Ator: Jogador.

Finalidade: Movimentar ameba de acordo com a rolagem de um dado.

Visão geral: O jogador deseja que a ameba se movimente não coordenadamente.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o jogador deseja que sua ameba se movimente não coordenadamente.	2- Rola um dado e registra a nova posição da ameba de acordo com os dados da bússola. Caso o dado apresente o valor 6 então o jogador poderá escolher um espaço. Ver caso de uso *Escolher Espaço*.

Alternativa item 2: Caso o jogador tenha a carta velocidade:

2a- **Sistema:** Movimenta a ameba duas vezes, uma rolagem de dado para cada vez. Caso o dado apresente o valor 6 então o jogador poderá escolher um espaço. Ver caso de uso *Escolher Espaço*.

Alternativa item 2: Caso o jogador tenha a carta movimento I:

2b- **Sistema:** Mostra a rolagem de 2 dados ao invés de 1.

3b- **Ator:** Jogador escolhe um dos dados.

4b- **Sistema:** Executa 2 ou 2a.

Alternativa item 2: Caso o jogador tenha a carta movimento II:

2c- **Sistema:** Não rola dados. Permite que o jogador escolha um espaço. Ver caso de uso *Escolher Espaço*.

3.1.8 Caso de uso: comer

Ator: Jogador.

Finalidade: Alimentar a ameba.

Visão geral: A ameba come três cubos de nutrientes de cores diferentes da sua.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando a ameba do jogador necessita alimentar-se.	2- Mostra os cubos de nutrientes disponíveis antes da alimentação. Libera dois cubos de nutrientes da mesma cor da ameba.

Alternativa item 2: Caso o jogador não tenha nutrientes o suficiente uma das ações a seguir pode acontecer:

2a- **Ator:** O jogador possui a carta luta pela sobrevivência.

3a- **Sistema:** Mostra quais amebas podem ser removidas do tabuleiro.

4a- **Ator:** Jogador escolhe a ameba a ser removida.

5a- **Sistema:** Remove ameba. Libera um cubo de nutriente de cada cor.

Alternativa item 5a: Caso o jogador da ameba a ser atacada tenha a carta defesa, fuga ou carapaça então:

- Carta defesa:

5b- **Sistema:** Rola um dado até não haver empate. Se o atacante vencer, retira a ameba do tabuleiro e não substitui por comida. Se o defensor vencer o atacante recebe um ponto prejudicial.

- Carta fuga:

5c- **Sistema:** Locomove a ameba se ela possuir outras cartas como: Movimento I/II, velocidade, aerodinâmica, tentáculo.

- Carta carapaça:

5d- **Sistema:** Não acontece nada.

3.1.9 Caso de uso: pagar diferença de mutação

Ator: Jogador.

Finalidade: O jogador paga a diferença entre seus pontos de mutação e a espessura da camada de ozônio.

Visão geral: Quando a carta de ambiente muda, é feito um somatório de todos os pontos de mutação do jogador. Se este valor ultrapassar o valor de espessura da camada de ozônio, o jogador precisa pagar esta diferença.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando a carta de ambiente muda e o somatório de todos os pontos de mutação do jogador ultrapassa a espessura da camada de ozônio.	
2- O jogador informa quantos BP's quer pagar e quantas cartas de gene quer devolver.	3- Verifica se o valor pago é suficiente e registra o pagamento.

3.1.10 Caso de uso: comprar genes

Ator: Jogador.

Finalidade: O jogador compra cartas de genes e modifica o comportamento de suas amebas.

Visão geral: O jogador decide comprar cartas de gene de acordo com o valor de seu saldo de BP's e modifica assim o comportamento de todas as suas amebas.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o jogador decide comprar cartas de genes.	
2- O jogador informa que quer comprar cartas de genes.	3- Mostra todas as cartas disponíveis.
4- O jogador escolhe a(s) carta(s).	5- Registra a compra e disponibiliza a(s) carta(s) ao jogador.

3.1.11 Caso de uso: adicionar amebas

Ator: Jogador.

Finalidade: O jogador adiciona amebas ao tabuleiro.

Visão geral: O jogador adiciona amebas ao tabuleiro e escolhe o espaço.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando é a vez do jogador adicionar amebas ao tabuleiro, seja na inicialização ou na fase de divisão de amebas.	
2- O jogador informa que deseja adicionar amebas.	3- Mostra todas as amebas disponíveis.
4- Na etapa de inicialização o jogador pode escolher apenas uma ameba por vez. O jogador escolhe a(s) ameba(s).	5- Para cada ameba o jogador escolhe um espaço: Ver caso de uso *Escolher Espaço*.

3.1.12 Caso de uso: requisitar histórico

Ator: Jogador.

Finalidade: Emitir histórico de partidas finalizadas.

Visão geral: Um jogador solicita o histórico de uma partida específica no qual ele tenha participado.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1-Este caso de uso começa quando um jogador deseja ver o histórico de uma partida anterior.	2- Lista todas as partidas anteriores deste jogador.
3- O jogador escolhe a partida.	4- Mostra histórico da partida.

3.1.13 Caso de uso: requisitar vencedores

Ator: Jogador

Finalidade: Listar os dez vencedores com maior número de partidas vencidas.

Visão geral: Um jogador deseja saber quais foram os dez jogadores que mais ganharam partidas.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1-Este caso de uso começa quando um jogador deseja ver quais jogadores mais ganharam partidas.	2- Lista os dez maiores ganhadores.

3.2 Diagrama de Classes

A Figura 3.3 mostra o diagrama de classes do sistema cliente que representa as telas do sistema, que herdam classes de interface gráfica, sendo que, a interface *ControladorRemote* e a classe *ControladorPartidaServidor* (classe de implementação) também presentes neste diagrama, são utilizadas tanto pelo sistema cliente quanto pelo servidor e servem como porta de comunicação entre esses dois sistemas. Observando-se o diagrama de classes do cliente, é possível notar referências a classes com o sulfixo *TO* no final. Elas nada mais são que classes de transferência, ou seja, suas instâncias são os objetos de transferência serializáveis que trafegam entre os dois sistemas. O diagrama dessas classes é representado pela Figura 3.2. É importante mencionar que elas também devem estar presentes em ambos os sistemas.

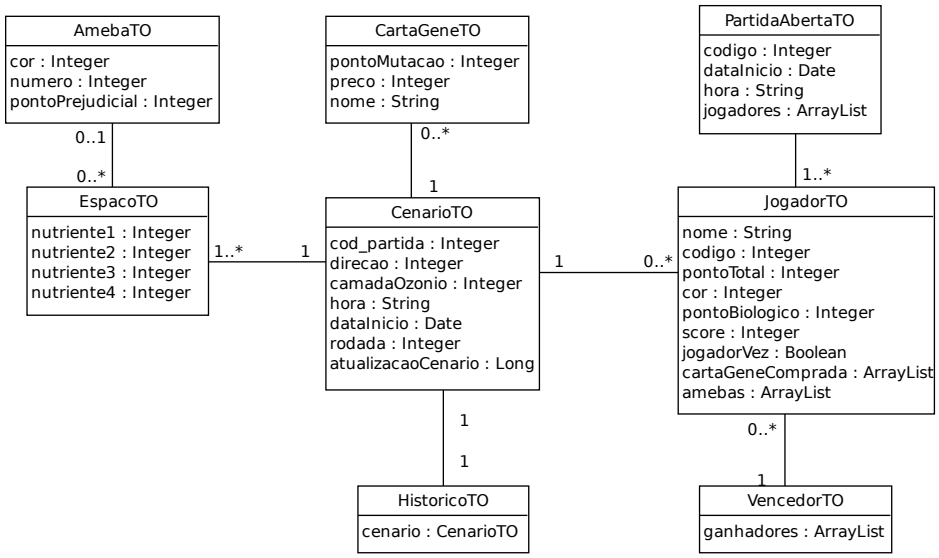

Figura 3.2: Diagrama de classes de transferência presentes tanto no cliente quanto no servidor. As classes de transferência são identificadas neste projeto pelo sulfixo *TO*.

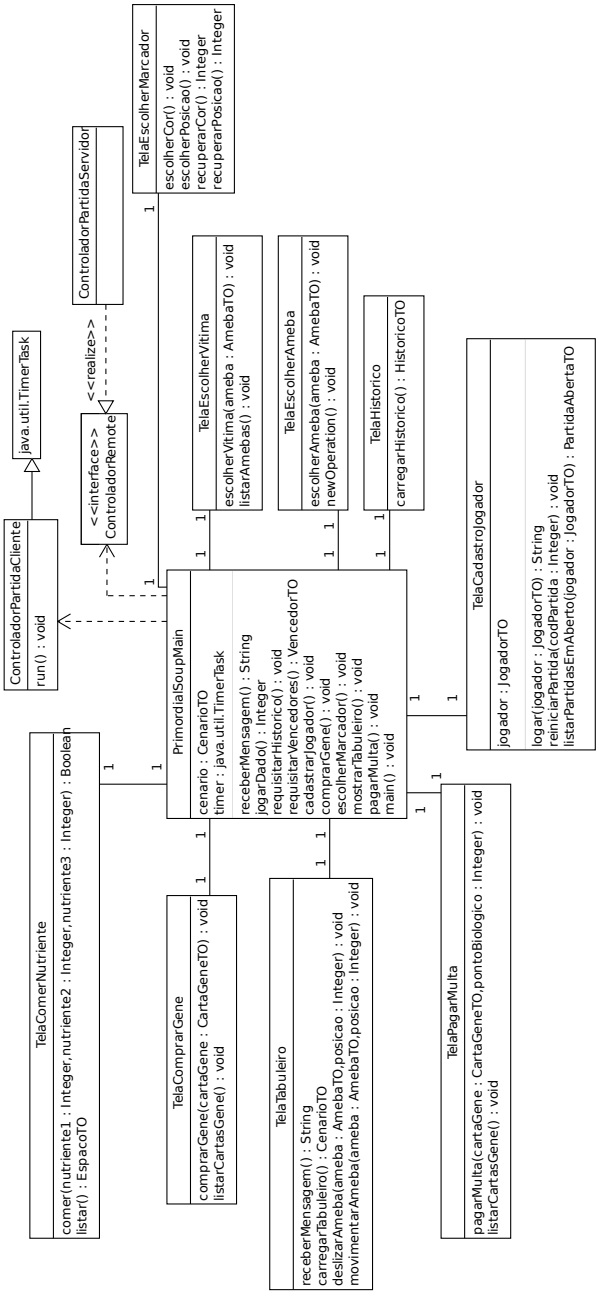

Figura 3.3: Diagrama de classes do sistema cliente.

Capítulo 4

Sistema servidor

O sistema servidor executa continuamente a espera de requisições dos sistemas cliente e processa todas as regras de negócio a partir das informações fornecidas pelos mesmos. Apenas suas classes de negócio utilizam o banco de dados para consultas e inserções de dados, tais como históricos e jogadores. A Seção 4.1 mostra o diagrama de caso de uso do sistema, bem como sua descrição detalhada. A Seção 4.2 mostra os diagramas de classes desse sistema e a Seção 4.3 mostra seu modelo de dados conceitual e físico.

4.1 Descrição de casos de uso

A Figura 4.1 mostra o diagrama de caso de uso aplicável ao sistema servidor, cuja descrição detalhada está contida nas seções a seguir.

4.1.1 Caso de uso: solicitar jogo

Ator: Cliente.

Finalidade: Cadastrar jogador para a próxima partida.

Visão geral: Um jogador cadastra seu nome a fim de participar da próxima partida.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1-Este caso de uso começa quando chega uma requisição do cliente para iniciar uma partida.	

Continua na próxima página.

Continuação.

Ação do ator	Resposta do sistema
2- O cliente envia o nome do jogador, senha e uma opção que pode ser reinício de uma partida e se trata de um novo jogador ou não.	3- Registra o jogador. Para nova partida: Caso haja 3 jogadores esperando então inicializa a partida. Ver caso de uso *Iniciar Partida*. Caso contrário, envia uma mensagem de espera. Para o reinício de uma partida: envia ao cliente uma lista de partidas abertas por ele.
	4- Se o tempo de espera passar do limite, o sistema envia uma mensagem: "Tempo de espera expirou".

Alternativa item 2: Caso o jogador não esteja cadastrado, mas o cliente indica que é um novo jogador então cadastra novo jogador no banco de dados com os dados enviados pelo cliente. Caso o cliente não tenha indicado que é um novo jogador então envia uma mensagem de erro.

4.1.2 Caso de uso: iniciar partida

Ator: Cliente.

Finalidade: Iniciar uma partida com 4 jogadores.

Visão geral: Quatro jogadores já estão disponíveis para jogar.

Tipo: Secundário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1-Este caso de uso começa quando quatro jogadores já estão prontos para jogar.	2- Cadastra início de jogo.
	3-Distribui 2 cubos de nutrientes de cada cor por espaço. E deposita 4 BP's pra cada jogador.

4.1.3 Caso de uso: solicitar interrupção

Ator: Cliente.

Finalidade: Cliente solicita paralisar partida.

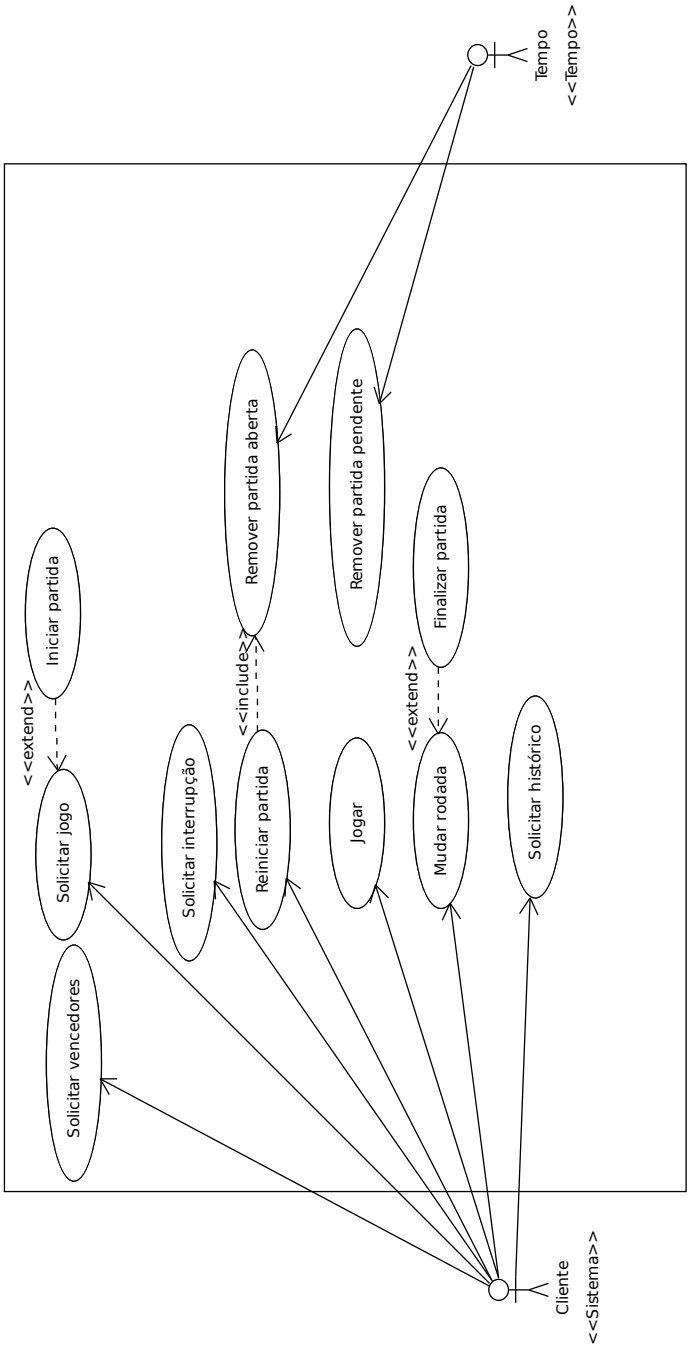

Figura 4.1: Diagrama de casos de uso para o sistema servidor.

Visão geral: Um dos clientes paralisa a partida e impede que os outros jogadores continuem a jogar.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1-Este caso de uso começa quando um cliente solicita a paralisação de uma partida.	2- Paralisa a partida para todos os clientes.
	3- Insere a partida na lista de partidas em aberto e seu horário de interrupção.
	4- Habilita que o cliente possa reiniciar a partida.

4.1.4 Caso de uso: reiniciar partida

Ator: Cliente.

Finalidade: Reinicia partida que foi paralisada.

Visão geral: O cliente resolve reiniciar uma partida que foi paralisada anteriormente e todos os outros clientes ficam habilitados a continuar.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um cliente reinicia uma partida.	2- Procura a partida em partidas em aberto.
	3- Carrega partida.
	4- Remove a partida da lista de partidas em aberto. Ver caso de uso *Remover Partida Aberta*.
	5- Inicia a partida do ponto de parada e habilita os outros clientes a continuar o jogo.

4.1.5 Caso de uso: remover partida aberta

Atores: Tempo, cliente

Finalidade: Remove partida da lista de partidas em aberto.

Visão geral: Quando um cliente solicita a reinicialização de uma partida ou quando seu tempo de espera expira, é necessário removê-la de uma lista de partidas em aberto.

Tipo: Primário, secundário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o cliente reinicializa uma partida ou quando seu tempo de espera para ser inicializada expira.	2- Remove a partida da lista de partidas em aberto.

4.1.6 Caso de uso: jogar

Ator: Cliente.

Finalidade: Receber as ações de jogo do cliente e validar.

Visão geral: O cliente envia uma execução de fase e recebe uma resposta de ação ou de recusa.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando o cliente envia uma determinada execução do jogo. Tipos de execução: 1A- Escolher espaço. 1B- Escolher marcador. 1C- Deslizar. 1D- Movimentar ameba. 1E- Comer. 1F- Pagar diferença de mutação. 1G- Comprar genes. 1H- Adicionar amebas.	
2a- Se a execução for: 1A.	3a- Se for fase 1: envia todos os espaços vazios como opção. Se for fase 4: envia todos os espaços vizinhos verticais ou horizontais as amebas de mesma cor já presentes no tabuleiro.

Continua na próxima página.

Continuação.

Ação do ator	Resposta do sistema
4a- Cliente envia sua escolha.	5a- Registra posição.
2b- Se a execução for: 1B	3b- Envia os espaços vazios na estrada de pontuação entre 1 e 4 e as cores disponíveis.
4b- Cliente envia sua escolha.	5b- Registra posição e cor.
2c- Se a execução for: 1C	3c- Consulta direção da bússola e registra nova posição da ameba. Caso o jogador da ameba possua carta tentáculo então pengunta ao cliente quais nutrientes deseja levar consigo (total de 3).
4c- Cliente escolhe os nutrientes, se houver.	5c- Muda os nutrientes de posição.
2d- Se a execução for: 1D	3d- Debita um BP do cliente, rola 1 dado e mostra o resultado.
	4d- Registra nova posição. Caso o jogador da ameba possua carta tentáculo então pengunta ao cliente quais nutrientes deseja levar consigo (total de 3).
5d- Cliente escolhe os nutrientes, se houver.	6d- Muda os nutrientes de posição.
2e- Se a execução for: 1E	3e- Consulta os cubos contidos no espaço e mostra ao cliente se ele possui comida o suficiente ou não, ou seja, 3 cubos de 3 cores diferentes da pertencente a ele.
4e- Cliente confirma estar ciente.	5e- Se ameba possui comida o suficiente então: registra a alimentação e substitui a comida por dois cubos de nutriente da mesma cor da ameba. Senão, se ameba não possui comida suficiente então: a ameba recebe um DP.
2f- Se a execução for: 1F	3f- Mostra a quantia devida.
4f- Envia o valor pago com BP's e/ou cartas gene.	5f- Verifica se a quantia paga está correta e registra o pagamento.
2g- Se a execução for: 1G	3g- Mostra as cartas que ele pode comprar.
4g- Escolhe as cartas que deseja comprar.	5g- Efetua a compra e debita o valor da(s) carta(s).

Continua na próxima página.

Continuação.

Ação do ator	Resposta do sistema
2h- Se a execução for: 1H	3h- Credita 10 BP's para o cliente e envia consulta de amebas disponíveis.
4h- Escolhe a(s) ameba(s).	Para cada ameba debita 6 BP's do cliente e mostra os espaços disponíveis. Ver opção 1A.

Alternativa 3a: Caso o jogador tenha a carta esporos então: envia todos os espaços que não contenham amebas de mesma cor.

Alternativa 3c: Caso o jogador tenha a carta abraço na fase 1:

3ca- **Sistema:** Envia ao cliente a opção de não *deslizar*.

3cb- **Ator:** Cliente envia resposta do jogador.

3cc- **Sistema:** Executa opção enviada pelo cliente.

Alternativa 3c: Se no mesmo espaço houver uma outra ameba com o gene abraço então:

3ca- **Sistema:** Envia ao segundo cliente a opção de uma de suas amebas se agarrarem a ameba que se movimentou.

3cb- **Ator:** Segundo cliente envia sua resposta.

3cc- **Sistema:** Caso a resposta seja positiva, desliza as duas amebas para a nova posição.

Alternativa 3d: Caso o jogador tenha a carta aerodinâmica então não cobra 1 BP pelo movimento.

Alternativa 3d: Se no mesmo espaço houver uma outra ameba com o gene abraço então:

3ca- **Sistema:** Envia ao segundo cliente a opção de uma de suas amebas se agarrarem a ameba que se movimentou.

3cb- **Ator:** Segundo cliente envia sua resposta.

3cc- **Sistema:** Caso a resposta seja positiva debita 1 BP do jogador do primeiro cliente, rola 1 dado e mostra o resultado. Registra a nova posição para as duas amebas.

Alternativa 3d: Caso o jogador tenha a carta velocidade então:

3da- **Sistema:** Movimenta a ameba duas vezes, uma rolagem de dado para cada vez. Caso o dado apresente o valor 6 então o jogador poderá escolher um espaço. Ver opção 1A

Alternativa item 3d e 3da: Caso o jogador tenha a carta movimento I então:

3db- **Sistema:** Rola 2 dados ao invés de 1.

4db- **Ator:** Cliente envia qual dos dados o jogador escolheu.

5db- **Sistema:** Executa 3d ou 3da.

Alternativa item 3d e 3da: Caso o jogador tenha a carta movimento II então:

3dc- **Sistema:** Não se usam os dados. Ver opção 1A.

Alternativa item 3e: Caso o jogador tenha a carta substituição, a alimentação de sua ameba poderá ser: 2 cubos de 2 cores diferentes ou 3 cubos de 1 cor e 1 cubo de outra.

Alternativa item 3e: Caso o jogador possua a carta frugalidade a alimentação da ameba poderá ser com um cubo a menos.

Alternativa item 3e: Caso o jogador possua a carta parasitismo a alimentação da ameba poderá ser com um cubo a menos e o jogador da ameba escolhida para ser prejudicada perde 1 BP.

Alternativa item 5e: Caso a ameba não tenha nutrientes o suficiente uma das ações a seguir pode acontecer:

1. O jogador possui a carta luta pela sobrevivência.

5ea- **Sistema:** Verifica e envia quais amebas podem ser removidas do tabuleiro.

6ea- **Ator:** O cliente envia qual ameba o jogador escolheu para ser removida.

7ea- **Sistema:** Remove ameba. Libera um cubo de nutriente de cada cor.

Alternativa item 5ea: Caso o jogador da ameba a ser atacada tenha a carta defesa, fuga ou carapaça então:

Carta defesa:

5eb- **Sistema** Rola um dado para cada um dos jogadores até não haver empate. Se o atacante vencer, retira a ameba do tabuleiro e não substitui por comida. Se o defensor vencer o atacante recebe um ponto prejudicial. Debita 1 BP do defensor. Caso um dois jogadores tenha a carta persistência então pode ter a chance de atacar ou defender mais uma vez.

Carta fuga:

5ec- **Sistema:** Registra nova posição da ameba se ela possuir outras cartas como: Movimento I/II, velocidade, aerodinâmica, tentáculo. E debita 1 BP do defensor caso ele não tenha uma carta de aerodinâmica.

Carta carapaça:

5ed- **Sistema:** Não acontece nada.

Alternativa 3e: Caso o jogador tenha a carta proteção de raios então:

3ea- **Sistema:** Conta -2 na soma dos pontos de mutação ou conta 4 pontos para compensar a diferença de pontos. Verifica então se a quantia é suficiente e registra o pagamento.

4.1.7 Caso de uso: mudar rodada

Ator: Cliente.

Finalidade: Mudar para a próxima rodada de uma partida.

Visão geral: Quando todos os clientes já passaram pela fase 4 que é de divisão de amebas, é necessário verificar quais amebas vão morrer, qual a sua pontuação.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um cliente indica que finalizou a fase 4 da partida (em ordem decrescente).	2- Verifica se alguma ameba possui dois ou mais DP's. Caso contenha, então a ameba é retirada do tabuleiro e dois cubos de nutrientes de cada cor são depositados no espaço onde a ameba estava.
	3- Caso um dos jogadores possua a carta Agressividade, disponibiliza a oportunidade de matar uma outra ameba. Envia uma lista de possíveis amebas mortas.
4- Escolhe a ameba para morrer.	Registra a morte da ameba e acrescenta 2 cubos de comida de cada cor ao espaço onde a ameba estava. Debita 1 BP do jogador agressivo.
	5- Calcula a pontuação por meio da tabela de evolução e grava. Sendo que a carta de proteção contra raios não conta na pontuação.

Continua na próxima página.

Continuação.

Ação do ator	Resposta do sistema
	6- Caso a pontuação do jogador do cliente chegue à zona de finalização ou o número de cartas ambiente chegue ao fim então registra a finalização da partida.
	5- Se o cliente for o quarto e último a indicar mudança de rodada e, a finalização da partida foi feita então executa caso de uso Finalizar Partida, caso contrário, inicia-se a próxima rodada.

Alternativa 2: Caso o jogador possua carta longevidade a ameba só poderá ser retirada do tabuleiro se possuir 3 DP's.

4.1.8 Caso de uso: finalizar partida

Ator: Cliente.

Finalidade: Finaliza a partida em andamento.

Visão geral: Ao final de uma rodada, um dos clientes possui pontuação na zona de finalização ou as cartas de ambiente acabaram. Assim a partida é finalizada.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando é identificado que um dos clientes possui pontuação correspondente a finalização da partida.	2- Paralisa a partida e consulta a pontuação de todos os clientes.
	3- Identifica o vencedor e caso ele tenha um dos 10 maiores números de partidas vencidas é adicionado na lista dos vencedores (Hall of fame).
	4- Grava toda as ações da partida em um histórico.
	5- Envia uma mensagem com o vencedor da partida e seus valores de pontuação.

4.1.9 Caso de uso: remover partida pendente

Ator: Tempo.

Finalidade: Remover partidas que não começaram efetivamente.

Visão geral: Ao passar um determinado tempo limite, uma partida que não tenha conseguido quatro jogadores para iniciar é removida.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando expira o tempo limite para iniciar uma partida.	2- Resgata os clientes que estavam esperando o início da partida , exclui a partida pendente e envia uma notificação de expiração do tempo limite.

4.1.10 Caso de uso: solicitar histórico

Ator: Cliente.

Finalidade: Emitir relatório sobre informações básicas de uma partida finalizada.

Visão geral: O cliente envia informação sobre uma partida e recebe um relatório básico sobre uma partida.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um cliente deseja obter um histório sobre uma partida.	2- Envia uma lista de partidas finalizadas .
3- Escolhe uma partida	4- Consulta informações básicas sobre a partida e emite o histórico.

4.1.11 Caso de uso: solicitar vencedores

Ator: Cliente.

Finalidade: Informar os dez maiores vencedores.

Visão geral: O cliente envia requisição sobre vencedores e recebe uma lista dos maiores vencedores.

Tipo: Primário.

Sequência Típica de Eventos:

Ação do ator	Resposta do sistema
1- Este caso de uso começa quando um cliente deseja obter uma lista de maiores vencedores.	2- Consulta os vencedores de cada partida e envia uma lista dos dez maiores vencedores.

4.2 Diagrama de Classes

Da mesma forma que o sistema cliente, o servidor utiliza a interface *Controlador-Remote* e sua respectiva classe de implementação *ControladorPartidaServidor*, que neste caso, consegue acessar diretamente objetos reais e referencia objetos de transferência serializáveis, cujas classes estão representadas na Figura 3.2. A figura Figura 4.2 mostra a estrutura geral de classes do servidor. As classes *Gerente** referenciam tanto objetos de transferência serializáveis quanto objetos de negócio, presentes no pacote *primordialsoupservidor*, cujas classes são representadas pelo diagrama da Figura 4.4. Os objetos de negócio acessam por sua vez, objetos de conexão com o banco de dados, cujas classes estão dentro do pacote *bancodados* e seu diagrama é mostrado na Figura 4.3.

4.3 Modelo de Dados

Neste projeto, somente o sistema servidor acessa banco de dados. A princípio o banco seria utilizado para armazenar histórico, cadastrar jogadores e talvez armazenar o último cenário do jogo, caso o tempo de expiração para partidas em aberto seja longo demais. Pensando nisso, foi projetado um modelo que suporta vários tipos de abordagem e futuras consultas um pouco mais robustas. A Figura 4.5 mostra o modelo de dados conceitual e a Figura 4.6 mostra seu respectivo modelo de dados físico. Sobre o modelo conceitual, podemos listar algumas frases que ajudarão no entendimento do diagrama, caso seja necessário:

- O jogador que joga uma tal partida pode comprar várias cartas gene.
- O jogador que joga uma tal partida pode executar uma ação.
- O jogador que joga uma tal partida pode controlar várias amebas.
- Um jogador pode vencer várias partidas.
- Uma partida pode ter várias cartas ambiente.

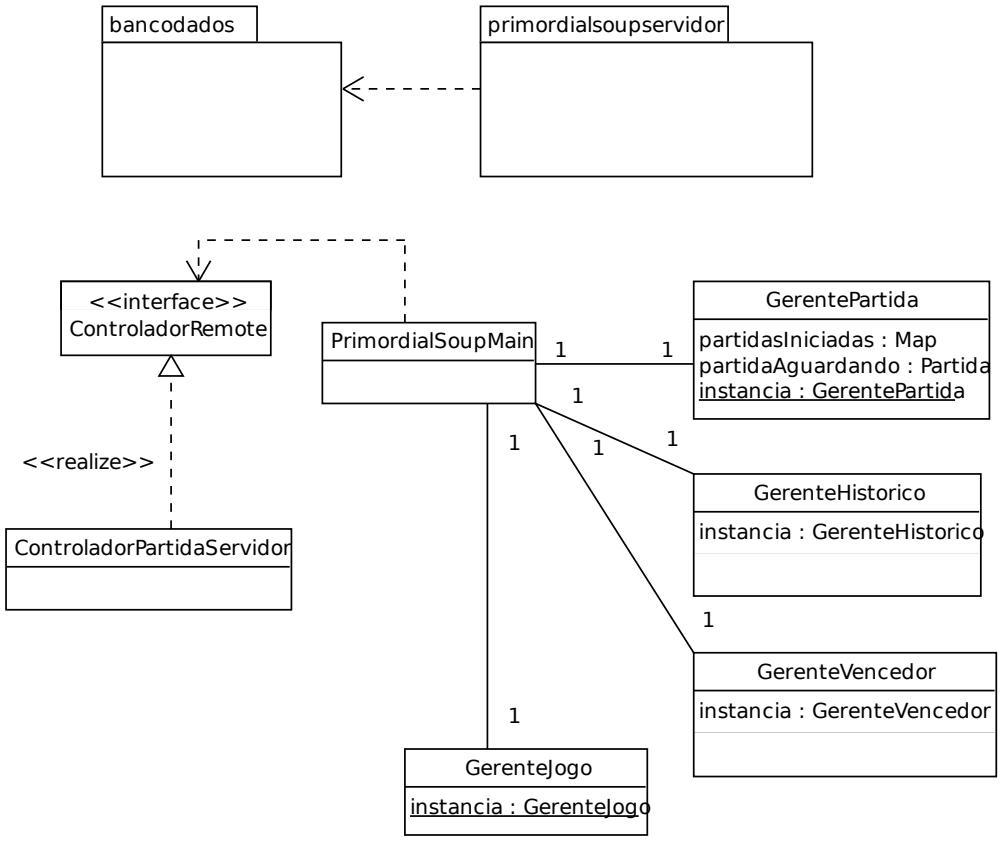

Figura 4.2: Diagrama de classes do servidor.

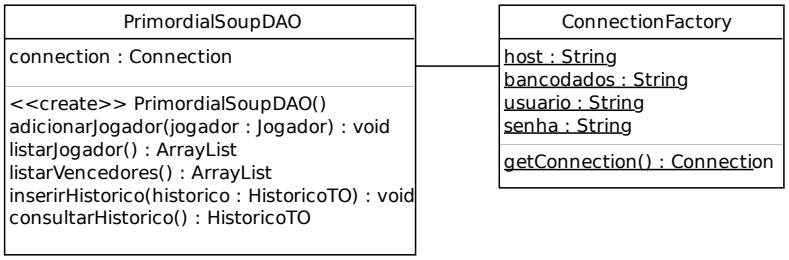

Figura 4.3: Diagrama de classes do pacote *bancodados* que representa as classes de conexão com um banco de dados.

- Um espaço no tabuleiro pode hospedar várias amebas.

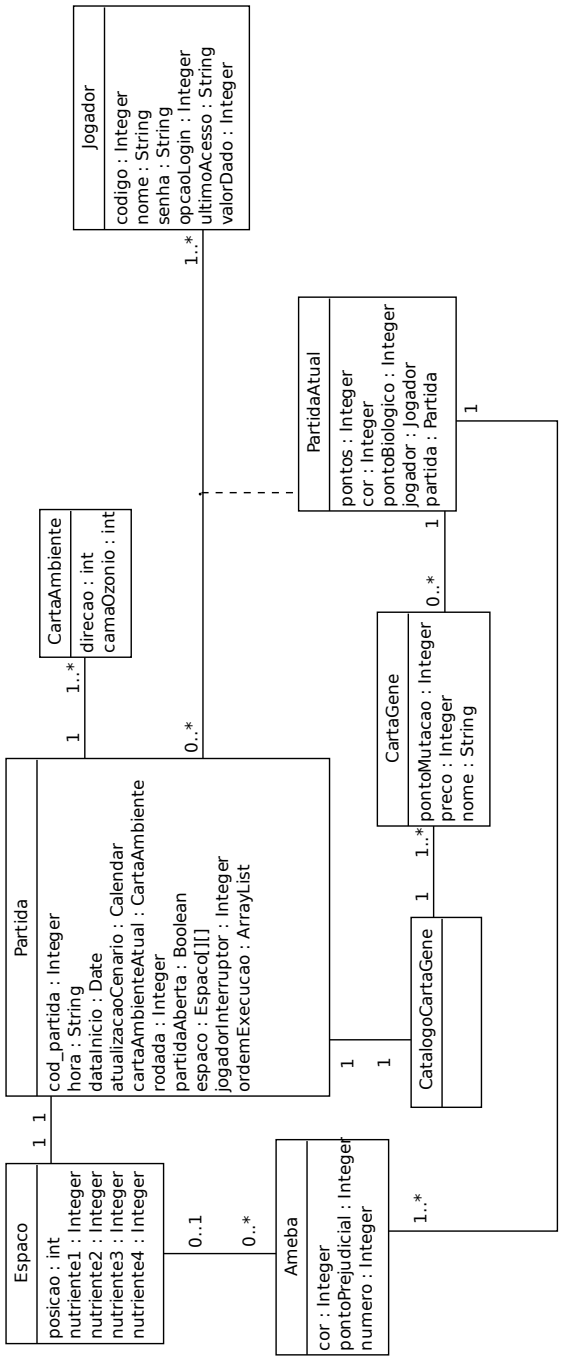

Figura 4.4: Diagrama de classes do pacote *primordialsoupservidor* que presenta as classes de negócio.

Outras considerações:

- A entidade *Ação* deve armazenar ações do jogo como: deslizar, mover, comer, matar, defender-se, parasitar, etc.
- Os atributos dado1..4 referem-se aos dados que indicam: esquerda, direita, superior e inferior da bússola.

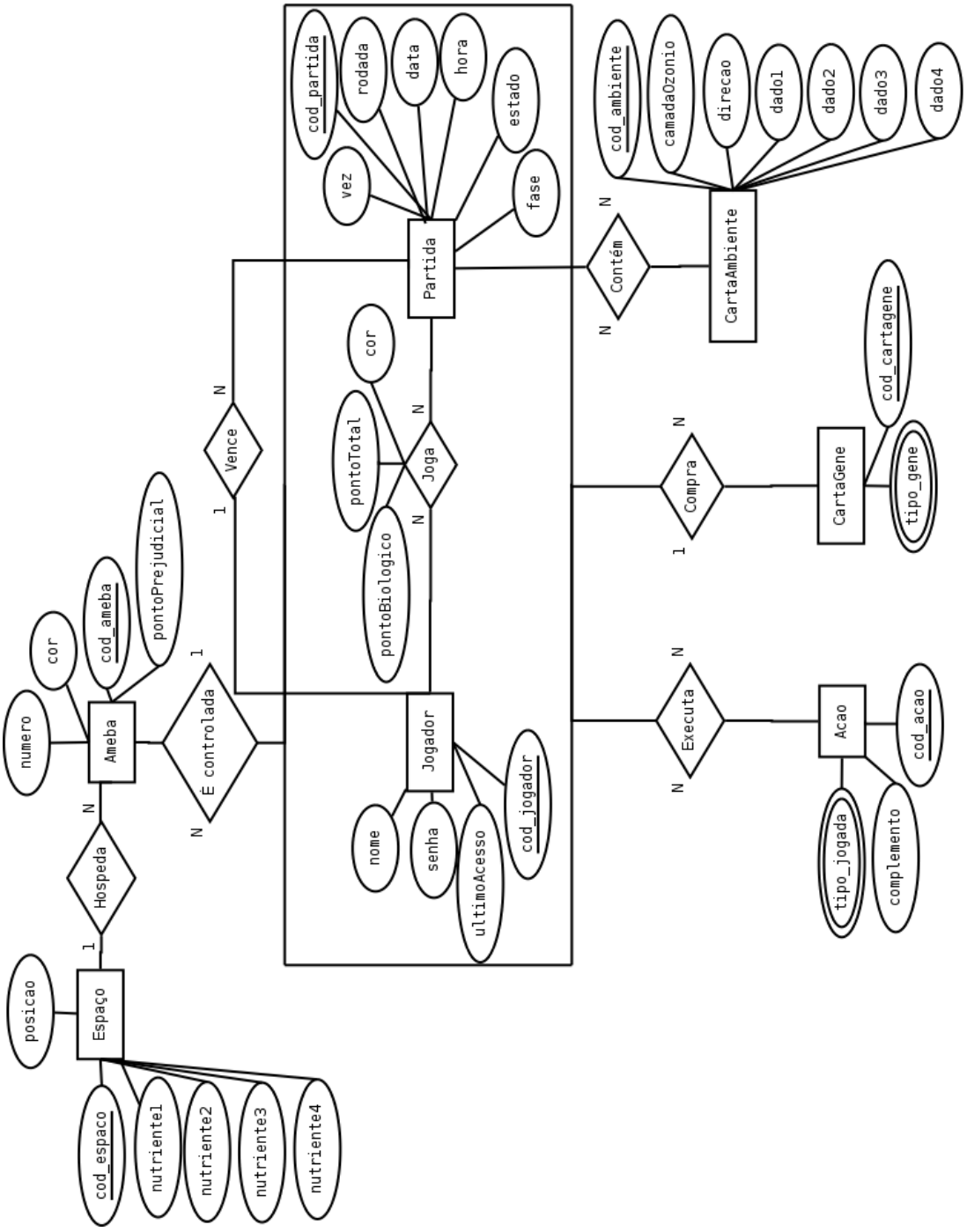

Figura 4.5: Modelo Entidade-Relacionamento Conceitual.

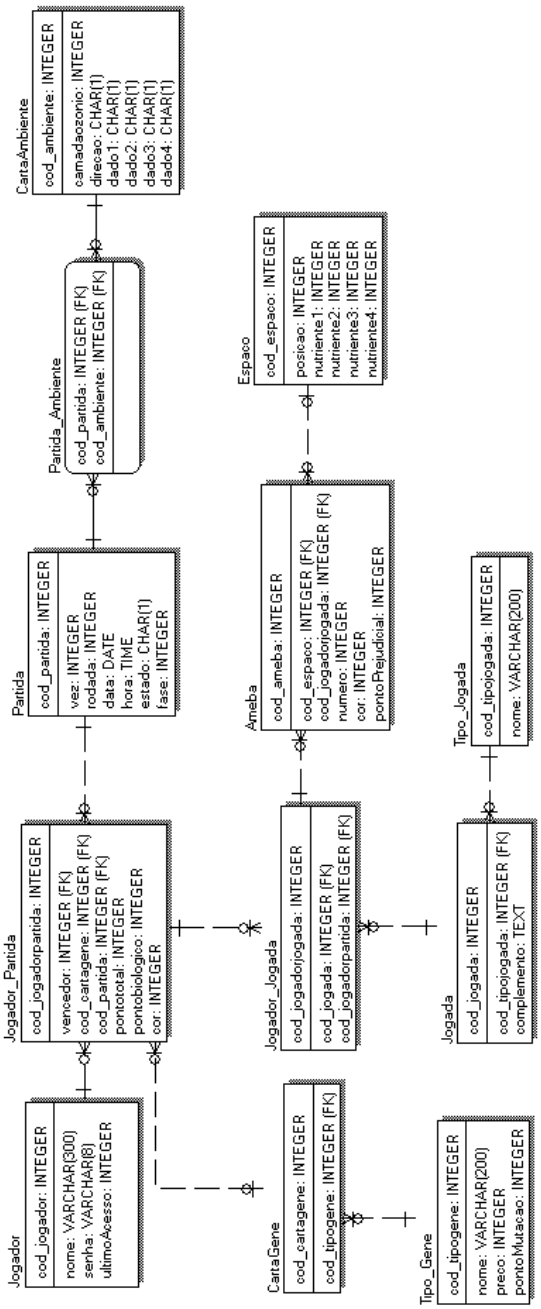

Figura 4.6: Modelo de Dados Físico.

Capítulo 5

Conclusão

O projeto ainda em andamento foi modelado neste documento através dos diagramas UML: diagrama de casos de uso, diagrama de classes, diagrama de componentes e diagrama de instalação, e dos modelos entidade-relacionamento conceitual e físico. Como trata-se da primeira fase do projeto, esta documentação é passível de pequenas modificações. São bem-vindas sugestões e críticas para melhorar o projeto e obter sucesso durante a implementação. A tecnologia RMI proposta aqui como uso para resolver a comunicação cliente-servidor promete ser uma solução sem muitas complicações e o modelo de banco de dados está projetado para suprir tanto a demanda de gravação de históricos quanto a demanda de gravação de partidas em aberto.

Referências Bibliográficas

Bezerra, E. (2007). *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier.

Doris & Frank (1997). Primordial soup. `http://www.zmangames.com/boardgames/primordial_soup.htm`. Acessado em julho de 2009.

Fowler, M. (2005). *UML Essencial: um breve guia para a linguagem padrão de modelagem de objetos*. bookman.

Freeman, E., Freeman, E., Sierra, K. & Bates, B. (2009). *Use a Cabeça: Padrões de Projetos*. Alta Books.

Larman, C. (2002). *Utilizando UML e Padrões*. bookman.