Problema 1: Jogo de Cartas Multiplayer

Cronograma

Encontro	Data	Atividade
1	14/08	Apresentação do Problema e Tutorial
2	19/08	Tutorial
3	21/08	Desenvolvimento
4	26/08	Tutorial
5	28/08	Desenvolvimento
6	02/08	Tutorial
7	04/09	Desenvolvimento
8	09/09	Desenvolvimento
9	11/09	Entrega do Barema e Elaboração do Relatório
10	16/09	Entrega e Apresentação do Produto
11	18/09	Apresentação do Produto

Contexto

No mercado de desenvolvimento de jogos é possível encontrar oportunidades em um nicho crescente ao focar no desenvolvimento de experiências online inovadoras. A ascensão dos jogos multiplayer, impulsionada pela demanda por conectividade através de interações sociais digitais, têm transformado essa indústria. Grandes empresas da área demonstram o vasto potencial do setor, focando no lançamento de jogos baseados em servidores centralizados, onde a experiência dos jogadores é gerenciada e sincronizada em tempo real. Este modelo permite não apenas a persistência de dados e a gestão de estados complexos, mas também a implementação de sistemas robustos e a entrega de atualizações contínuas, elementos esses que são quase sempre cruciais para a longevidade e sucesso de um título. A comunicação eficiente e de baixa latência entre clientes e servidores deve ser a espinha dorsal desse tipo de sistema, envolvendo o uso de tecnologias de comunicação que, quando bem implementadas, desempenham papel fundamental na garantia de uma experiência de jogo fluida e responsiva.

Problema

Você e seus amigos do curso de Engenharia de Computação da UEFS acabaram de fundar uma startup focada no desenvolvimento de jogos *indie*. O foco da sua equipe é entrar no mercado de jogos online com experiências divertidas e confiáveis. O primeiro grande desafio da startup é desenvolver um novo jogo de cartas online multiplayer, focado em duelos táticos e na coleção de cartas, onde os jogadores devem interagir em um ambiente compartilhado. Este jogo será baseado em um servidor centralizado que gerenciará parte da lógica, o estado dos jogadores e a comunicação entre eles.

A comunicação entre os jogadores e o servidor central deve ser implementada de forma bidirecional e em tempo real, o que é essencial para a dinâmica do jogo. O sistema também deve permitir: (i) a conexão de múltiplos jogadores simultaneamente; (ii) visualizar o atraso da comunicação dos jogadores conectados ao servidor; (iii) permitir aos jogadores se enfrentarem em partidas 1v1. Sobre as partidas, o sistema implementado deve ter o cuidado de garantir que dois jogadores sejam pareados para um duelo único, sem que um jogador seja pareado com múltiplos oponentes simultaneamente.

Além dessas funcionalidades, a aquisição de novas cartas através da abertura de pacotes deve ser uma mecânica central, introduzindo elementos de sorte e disputa pela obtenção de cartas raras em pacotes únicos. Assim, a implementação do sistema deve tratar os pacotes de cartas como um "estoque" global e garantir que, quando múltiplos jogadores tentarem abrir pacotes ao mesmo tempo, a distribuição de cartas seja justa e que apenas um jogador a receba, evitando duplicações ou perdas de cartas.

Por fim, para garantir a confiabilidade do sistema, é crucial realizar testes. Essa avaliação deve verificar requisitos e submeter o sistema a testes automáticos de estresse para assegurar a justiça em situações de concorrência. Além disso, é preciso medir o desempenho do servidor com múltiplos jogadores simultâneos para identificar e corrigir falhas, otimizando o sistema para entregar uma experiência de jogo estável, justa e fluida.

Restrições

1. Os componentes do jogo devem ser desenvolvidos e testados em contêineres Docker, permitindo a execução de múltiplas instâncias no laboratório;
2. Por questões comerciais, não deve ser utilizado nenhum framework para a comunicação entre os componentes do jogo. A comunicação deve ser implementada exclusivamente por meio do uso da biblioteca nativa de sockets disponível no sistema para permitir a comunicação sobre a arquitetura da Internet.

Nossas Regras

- Os alunos devem implementar o projeto de forma individual;
- O prazo final para entrega e apresentação do trabalho será **16/09/2025**;
- O código-fonte deve ser entregue devidamente comentado por meio da plataforma GitHub, com README explicando como executar o jogo junto e os scripts de testes;
- O aluno deve entregar, junto com o produto, um relatório de no máximo 8 páginas, seguindo o formato padrão da SBC (Sociedade Brasileira de Computação), contendo conceitos e justificativas para a solução adotada;
- O sistema do jogo deve ser desenvolvido, testado e apresentado no Laboratório de Redes e Sistemas Distribuídos (LARSID), onde as apresentações seguirão uma agenda sequencial definida antes da apresentação;
- Cada aluno terá 25 minutos para apresentar o sistema do jogo em funcionamento e responder às questões técnicas sobre a implementação.

Observações

- Trabalhos entregues fora do prazo serão penalizados com 20% do valor da nota + 5% por dia de atraso, dentro da mesma semana da entrega final;
- Trabalhos copiados de qualquer fonte e trabalhos idênticos terão nota ZERO;
- As informações sobre o problema podem ser alteradas no decorrer das sessões.

Avaliação

A nota final será a composição das seguintes três notas:

1. Desempenho tutorial - 30%
2. Relatório do produto (em PDF) - 20%
3. Produto no GitHub (com README) - 50%