UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - CAMPUS FLORESTAL

SISTEMAS DISTRIBUÍDOS E PARALELOS

Trabalho Prático

Modelagem do Sistema

Aymê Faustino dos Santos - 4704

Pedro Augusto Martins Pereira - 4692

Juan Pablo Andrade Avelar - 4229

Florestal - MG

2025

Sumário

1. Modelo Físico

Nosso sistema distribuído será um marketplace voltado para a comunidade bruxa, com funcionalidades que exigem escalabilidade, confiabilidade e disponibilidade para atender um grande número de usuários simultaneamente. Optamos por adotar o modelo de Internet-Scale, ele é ideal pois permite que o sistema opere em escala global, distribuindo recursos e serviços em diferentes localidades. Isso é necessário para que os usuários possam acessá-lo em qualquer lugar, seja para cadastrar uma loja, realizar uma compra ou acompanhar pedidos. Além disso, a escolha desse modelo é justificada pelo perfil dinâmico do *BecoDiagonal*, que envolve interações constantes entre compradores e vendedores, atualizações frequentes de produtos, avaliações e transações financeiras.

2. Modelo de Arquitetura

2.1. Paradigma de Comunicação

Na etapa inicial do desenvolvimento do sistema, as entidades arquitetônicas serão modeladas como processos independentes, que se comunicam diretamente entre si. Essa abordagem permite uma troca de informações mais simples e eficiente, favorecendo o compartilhamento de recursos. Posteriormente, na segunda etapa, será introduzido uma abstração por meio de middleware, permitindo que as entidades sejam modeladas como objetos . Nesse novo cenário, utilizaremos o paradigma de invocação remota (RPC – Remote Procedure Call), que abstrai a complexidade da comunicação entre processos e contribui para a modularidade e escalabilidade do sistema.

2.2. Modelo de Arquitetura Básico

O modelo de arquitetura básico adotado no sistema *BecoDiagonal* será o modelo Cliente-Servidor (C/S). Nesse modelo, os processos assumem os papéis de cliente ou servidores, sendo que os processos clientes interagem com processos servidores, para acessar e manipular os recursos compartilhados que os servidores gerenciam. No contexto do nosso marketplace, essa divisão pode ser vista de forma prática: os usuários interagem com o sistema por meio de navegadores (cliente), enviando requisições de negócio para um servidor de aplicação, responsável por processar as ações, aplicar as regras de negócio e repassar comandos ao servidor de dados, que gerencia o armazenamento das informações persistentes, como cadastros de lojas, produtos e transações.

2.3. Funções e Responsabilidades das Entidades do Modelo

O nosso sistema será composto por 3 entidades: Cliente, Servidor de Aplicação e Servidor de Dados. A seguir estão listadas as responsabilidades de cada um deles:

2.3.1 Cliente

As responsabilidades de uma entidade Cliente são de interagir com usuário final por meio de uma interface, realizar operações como criação de conta, login, edição de perfil, visualização e criação de lojas, publicação e edição de produtos, compra e avaliação de itens. Além disso, o cliente é responsável por enviar requisições ao Servidor de Aplicação e lidar com as respostas recebidas, atualizando a interface do usuário de acordo com as informações retornadas.

2.3.2 Servidor de Aplicação

Executa a lógica do negócio. Processa regras do sistema como cadastro de loja, criação de anúncio, controle de login e pagamentos. Gerência autenticação, sessão, lógica de compra e controle de estado dos produtos, comunicando com o servidor de dados. Intermedia a comunicação entre cliente e servidor de dados, garantindo que as operações sigam as regras definidas para o funcionamento correto do sistema.

2.3.3 Servidor de Dados

Esta entidade tem como principal responsabilidade gerenciar o acesso ao banco de dados, garantir a comunicação eficiente com os demais componentes do sistema e garantir a persistência das informações. É encarregada de armazenar e recuperar informações persistentes, como dados de contas de usuários, produtos, transações, histórico de compras e detalhes das lojas cadastradas.

2.4. Camadas Físicas e Camadas Lógica

Nosso sistema será estruturado com camadas físicas e lógicas.
Na primeira parte do desenvolvimento, focaremos em duas camadas lógicas principais: a camada de aplicações e serviços, e a camada de plataforma. Essa decisão se baseia por que neste momento trabalharemos somente com processos do sistema. Já na segunda parte será incluída uma nova camada lógica: a camada de middleware.

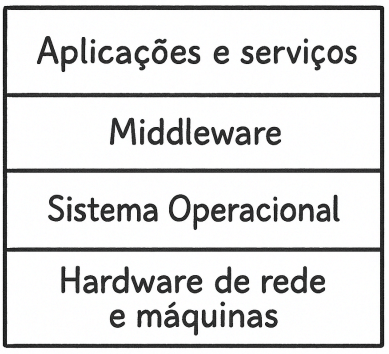

Figura 1 - Diagrama Representativo das Camadas Lógicas

Durante as duas etapas, adotaremos o modelo de 3 camadas físicas: aplicação, apresentação e dados. Essa abordagem foi escolhida por proporcionar uma organização mais clara do sistema, o que contribui para facilitar a manutenção e permitir maior escalabilidade ao longo do tempo.

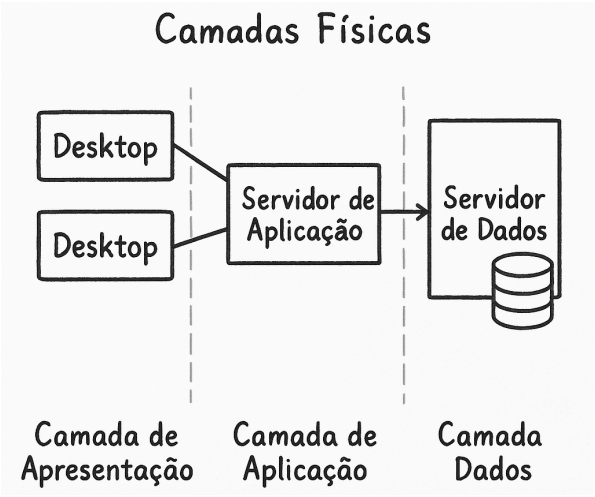

Figura 2 - Diagrama Representativo das Camadas Físicas

3. Modelos Fundamentais

3.1. Modelo de Interação

Para o nosso sistema de marketplace, a latência elevada pode comprometer significativamente a experiência do usuário, afetando etapas críticas como o carregamento de produtos e a finalização de compras. A taxa de transmissão também é essencial, pois influencia diretamente a exibição de imagens dos produtos e o carregamento eficiente das listagens, que são funções centrais nesse tipo de plataforma. Embora o jitter tenha um impacto relativamente menor se comparado à latência e à taxa de transmissão, variações bruscas neste parâmetro devem ser evitadas, uma vez que podem comprometer a consistência da navegação e causar sensações de instabilidade na interface. Embora o sistema seja projetado para mitigar esses efeitos, nem sempre será possível evitar falhas ou lentidão, especialmente em conexões instáveis ou quando o sistema estiver sobrecarregado. Nesses casos, o sistema poderá exibir mensagens informativas para orientar o usuário, mesmo que o problema não possa ser resolvido imediatamente.

3.1.1 Modelo Síncrono ou Assíncrono

O sistema utilizará um modelo assíncrono. O modelo assíncrono permite que o cliente envie uma requisição e continue funcionando normalmente enquanto aguarda uma resposta do servidor. Esse modelo é mais adequado para aplicações que lidam com muitos usuários simultaneamente e que nem sempre exigem respostas imediatas para continuar funcionando. Em um marketplace, por exemplo, o usuário pode navegar por diferentes páginas, adicionar produtos ao carrinho ou editar o perfil, mesmo que uma solicitação anterior ainda esteja sendo processada. Isso torna o sistema mais resiliente a falhas e flutuações na rede, o que é ideal para o *BecoDiagonal*. uma vez que não há garantias estritas de tempo de execução nem de entrega das mensagens.

3.2. Modelo de Falhas

O modelo de falhas que adotamos será o de falhas por omissão, que ocorrem quando processos ou canais de comunicação deixam de realizar as ações esperadas. No contexto do sistema, esse tipo de falha pode acontecer, por exemplo, quando o servidor de aplicação não responde a uma requisição de compra devido à sobrecarga ou falha momentânea. O cliente pode perceber esse tipo de falha caso não receba uma resposta do servidor dentro de um intervalo de tempo previamente definido. Para contornar a situação, ele pode realizar novas tentativas de envio da solicitação. Caso as tentativas falhem, o sistema notificará o usuário sobre a indisponibilidade do serviço e recomendará que ele tente novamente mais tarde. Dessa forma, as falhas por omissão serão identificadas, mas não resolvidas diretamente apenas ocultadas do usuário por meio de mensagens informativas.

3.3. Modelo de Segurança

Proteção aos processos: É essencial garantir que apenas usuários devidamente autenticados possam realizar ações sensíveis, como editar informações de perfil, criar um anúncio de produtos ou alterar status de uma loja. Sem essa proteção, o sistema ficaria vulnerável a ataques internos e externos, como ações de usuários mal-intencionados tentando modificar ou acessar dados alheios. Para mitigar esse risco, será implementado um sistema de autenticação segura, além de mecanismos de autorização que garantem que cada ação só possa ser executada por usuário com permissões válidas.

Proteção ao canal de comunicação: Para a proteção dos canais de comunicação será utilizado a criptografia de dados sensíveis como senhas, informações pessoais e de pagamento. Isso

garante a confidencialidade e integridade das mensagens trocadas, impedindo que dados sejam interceptados ou modificados durante o tráfego entre cliente e servidor.