FATEC COTIA

Desenvolvimento de Software Multiplataforma

LÁZARO SANTOS

LEANDRO CARDOSO

CELSO SEBASTIÃO

JHONATHAM PEDROSO

IES011 - Projeto Integrador I - Atividade 2

Sistema Inteligente de Recomendação para Bares e Restaurantes

COTIA – SP

NOVEMBRO/ 2024

Lista De Figuras

SUMÁRIO

1. FERRAMENTAS, LINGUAGENS E SOFTWARES

A escolha das tecnologias para este projeto foi baseada nas necessidades específicas de cada parte do sistema, buscando garantir desempenho, escalabilidade e uma boa experiência para o usuário final. Abaixo, listamos as ferramentas, linguagens e softwares adotados, incluindo a justificativa para cada escolha.

1.1. Linguagem de Back-End

Escolha: C#
Justificativa: Optamos pelo C# devido à integração com o .NET, que oferece um ambiente robusto para o desenvolvimento de aplicações escaláveis e seguras. O .NET proporciona diversas bibliotecas e suporte de alta performance para integração de APIs e manipulação de dados, essenciais para nosso sistema de recomendações e gerenciamento de pedidos.

1.2. Framework de Back-End

Escolha: ASP.NET Core
Justificativa: ASP.NET Core foi escolhido pelo suporte à criação de APIs RESTful e pela compatibilidade com containers, facilitando a escalabilidade do sistema. Este framework é amplamente utilizado em sistemas de grande porte e possui recursos para garantir segurança e eficiência, necessários para um sistema com requisitos de acesso frequente e personalização.

1.3. Framework de Front-End

Escolha: React.js
Justificativa: React.js foi escolhido para desenvolver interfaces dinâmicas e interativas, essenciais para uma experiência de navegação fluida. O framework conta com uma comunidade ativa e uma vasta biblioteca de componentes, o que facilita tanto o desenvolvimento quanto a manutenção do front-end.

1.4. Sistema Gerenciador de Banco de Dados

Escolha: Microsoft SQL Server (SQL) e MongoDB (NoSQL)

Justificativa: Optamos por uma abordagem híbrida com um banco de dados relacional (SQL Server) e um banco não-relacional (MongoDB), pois cada um atende a requisitos distintos do sistema:

- Microsoft SQL Server é adequado para dados que exigem consistência e integridade, como informações de usuários, pedidos e transações financeiras. Sua estrutura relacional facilita a criação de consultas complexas e garante a integridade dos dados.

- MongoDB é ideal para dados que mudam com frequência ou exigem flexibilidade, como registros de recomendações e preferências dos usuários. O modelo NoSQL facilita consultas rápidas e escaláveis, além de permitir o armazenamento de dados semi-estruturados que se adaptam melhor às atualizações frequentes e acessos em tempo real.

Essa combinação proporciona um desempenho otimizado e permite que cada tipo de dado seja armazenado e acessado de forma eficiente.

1.5. Versionador de Código

Escolha: Git (GitHub)
Justificativa: Utilizaremos o GitHub para controle de versão, dado seu suporte a trabalho colaborativo e integração contínua. O GitHub permite o rastreamento de mudanças no código, garantindo um histórico completo e facilitando o trabalho em equipe.

1.6. Ferramenta de Diagramação UML

Escolha: LucidChart
Justificativa: LucidChart foi escolhido por ser uma ferramenta gratuita e simples de usar, permitindo a criação de diagramas UML para visualizar a arquitetura do sistema e os fluxos de dados.

1.7. Outras Ferramentas

Jira: Utilizado para o gerenciamento de tarefas e organização do fluxo de trabalho com a metodologia Scrum.

Figma: Ferramenta de design e prototipagem, garantindo um layout intuitivo e atraente para o usuário.

Essas escolhas visam um desenvolvimento eficiente e colaborativo, ao mesmo tempo que garantem a escalabilidade e a estabilidade do sistema.

2. FORMA DE CONSTRUÇÃO: MICROSERVIÇOS

Para um sistema de recomendação e gestão para bares e restaurantes, é crucial escolher uma arquitetura de software que suporte escalabilidade, flexibilidade e manutenção simplificada, considerando os requisitos de personalização e alto volume de interações com o usuário.

Optamos pela arquitetura de microserviços, pois ela se alinha aos requisitos do sistema ao permitir que diferentes partes do aplicativo sejam desenvolvidas, implantadas e escaladas de forma independente. Para um sistema com diversas funcionalidades – como recomendação personalizada, controle de estoque, gestão de pedidos e perfis de usuários – a arquitetura de microserviços facilita a modularização de cada componente, promovendo uma construção mais flexível e robusta.

Os microserviços também trazem vantagens específicas para este projeto:

1. Escalabilidade: Cada microserviço pode ser escalado separadamente conforme a demanda, essencial para uma aplicação que espera um grande volume de consultas e interações em tempo real. Por exemplo, o módulo de recomendação pode ser escalado independentemente do módulo de gestão de pedidos, otimizando o uso de recursos.

2. Manutenção e Atualizações: Cada serviço pode ser atualizado sem interferir no funcionamento dos demais, reduzindo o tempo de inatividade e permitindo que novas funcionalidades sejam implementadas gradualmente.

3. Flexibilidade na Escolha de Tecnologias: A arquitetura de microserviços permite a escolha de tecnologias distintas para cada serviço, conforme suas necessidades. Por exemplo, podemos usar o Microsoft SQL Server para microserviços que precisam de dados transacionais e consistentes, enquanto o MongoDB pode servir aos microserviços de recomendação, que exigem um banco de dados flexível e de leitura rápida.

2.1. Adequação das Ferramentas à Arquitetura de Microserviços

As ferramentas escolhidas são altamente compatíveis com essa abordagem:

- ASP.NET Core: Ideal para criação de APIs RESTful, facilita a construção de microserviços que podem se comunicar com eficiência e segurança.

- C#: Integrado ao .NET, permite uma programação robusta e segura para os microserviços que necessitam de alta performance, como o gerenciamento de pedidos e controle de estoque.

- React.js: Permite a criação de uma interface de usuário dinâmica e interativa que se comunica facilmente com diferentes microserviços, mantendo uma experiência coesa para o usuário.

- SQL e NoSQL (Microsoft SQL Server e MongoDB): A combinação de bancos de dados relacionais e não relacionais permite que cada microserviço utilize a melhor solução para suas necessidades, garantindo a integridade de dados em transações e a flexibilidade de dados de consulta rápida.

- Git (GitHub): Ferramenta essencial para versionamento e controle de código, especialmente em um ambiente de microserviços onde cada serviço pode evoluir de maneira independente.

2.2. Para concluir este tópico

A escolha pela arquitetura de microserviços, juntamente com as ferramentas selecionadas, garante que o sistema possa crescer de maneira ágil e escalável, mantendo a qualidade de serviço, mesmo com o aumento de usuários e interações.

3.0 RESOLUÇÃO DOS DESAFIOS TÉCNICOS

Este projeto enfrenta desafios técnicos na implementação de um sistema de recomendação de restaurantes, abordando aspectos como volumetria de acessos, personalização de recomendações e controle de superlotação de estabelecimentos. Abaixo, detalhamos as soluções propostas para cada desafio:

3.1 Métodos de Recomendação

Para gerar recomendações precisas, o sistema utilizará algoritmos que combinam:

- Filtragem Colaborativa: Identifica preferências semelhantes entre usuários e itens.

- Filtragem Baseada em Conteúdo: Baseada nas características dos restaurantes, como tipo de culinária.

- Recomendações Contextuais: Considera o contexto atual do usuário (localização, horário, preferências).

- Aprendizado de Máquina: Modelos preditivos para personalizar sugestões a novos usuários.

Solução Proposta:

Implementar um sistema híbrido que utilize a filtragem colaborativa e baseada em conteúdo, aprimorado com machine learning para personalizar recomendações de acordo com o contexto e preferências do usuário.

3.2. Integração com Google Places API

A integração com a Google Places API proporcionará dados adicionais para enriquecer as recomendações, permitindo:

- Busca de locais e detalhes sobre restaurantes, facilitando a localização e filtragem.
- Geocodificação para identificar estabelecimentos próximos.
- Autocompletar e avaliações de clientes para otimizar a experiência.

Solução Proposta:

Integrar a API para fornecer informações detalhadas e em tempo real sobre restaurantes, enriquecendo a experiência do usuário com sugestões mais precisas.

3.3. Monetização: Destaque Pago para Restaurantes

Para permitir a monetização, será oferecido um modelo de destaque pago, no qual:

- Restaurantes podem pagar por cliques (PPC) ou optar por uma assinatura mensal.
- Opções de leilão garantem posição de destaque em momentos de alta demanda.

Solução Proposta:

Um modelo híbrido de PPC e assinatura mensal, permitindo a utilização de leilões para restaurantes que buscam maior visibilidade.

3.4. Métricas de Recomendação para Restaurantes

Os restaurantes precisam de métricas para ajustar suas estratégias de visibilidade, como:

- Relatórios periódicos e um dashboard em tempo real para monitoramento de recomendações, cliques e impressões.

- Notificações para atualizar sobre o desempenho das campanhas.

Solução Proposta: Desenvolver um sistema de relatórios e dashboards em tempo real para fornecer feedback contínuo e apoiar ajustes de estratégia dos restaurantes.

3.5 Controle de Superlotação

Para evitar superlotação, será implementado um sistema que monitora a capacidade dos estabelecimentos:

- Informações sobre a capacidade e ocupação serão coletadas em tempo real, e o número de recomendações ajustado automaticamente.
- Segmentação de recomendações por horário e perfil para distribuir a demanda.

Solução Proposta:

Algoritmos que monitoram a capacidade dos estabelecimentos, ajustando as recomendações para evitar sobrecarga e garantir uma distribuição equilibrada.

4.0 DESENHO DA ARQUITETURA DE COMPONENTES VISÃO GERAL

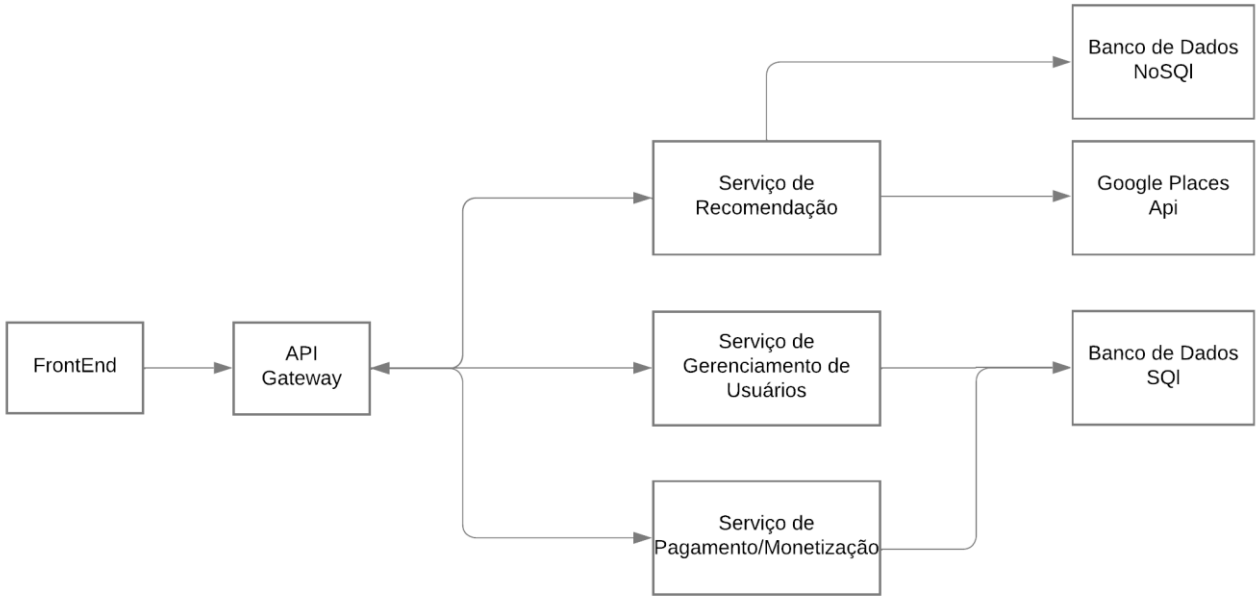

Figura 1 – Desenho da Arquitetura de Componentes

4.1. Fluxo Geral:

1. O Frontend recebe as interações do usuário (por exemplo, fazer um pedido ou buscar um restaurante) e envia para o API Gateway.
2. O API Gateway direciona a solicitação para o microserviço apropriado:
    a. Serviço de Recomendação para gerar recomendações de restaurantes.
    b. Serviço de Gerenciamento de Usuários para verificar o histórico e dados do usuário.
    c. Serviço de Pagamento/Monetização para processar o pagamento ou verificar transações.
3. O Serviço de Recomendação acessa o Google Places API para obter informações sobre os restaurantes e armazena as preferências dos usuários no Banco de Dados NoSQL.
4. O Serviço de Gerenciamento de Usuários consulta e armazena informações no Banco de Dados SQL, como dados de login e histórico de pedidos.
5. O Serviço de Pagamento/Monetização também usa o Banco de Dados SQL para registrar transações de pagamento, e envia notificações relacionadas ao pagamento via Serviço de Mensageria.

5. MODELAGEM DO BANCO DE DADOS (MODELO ENTIDADE RELACIONAMENTO)

5.1. DER

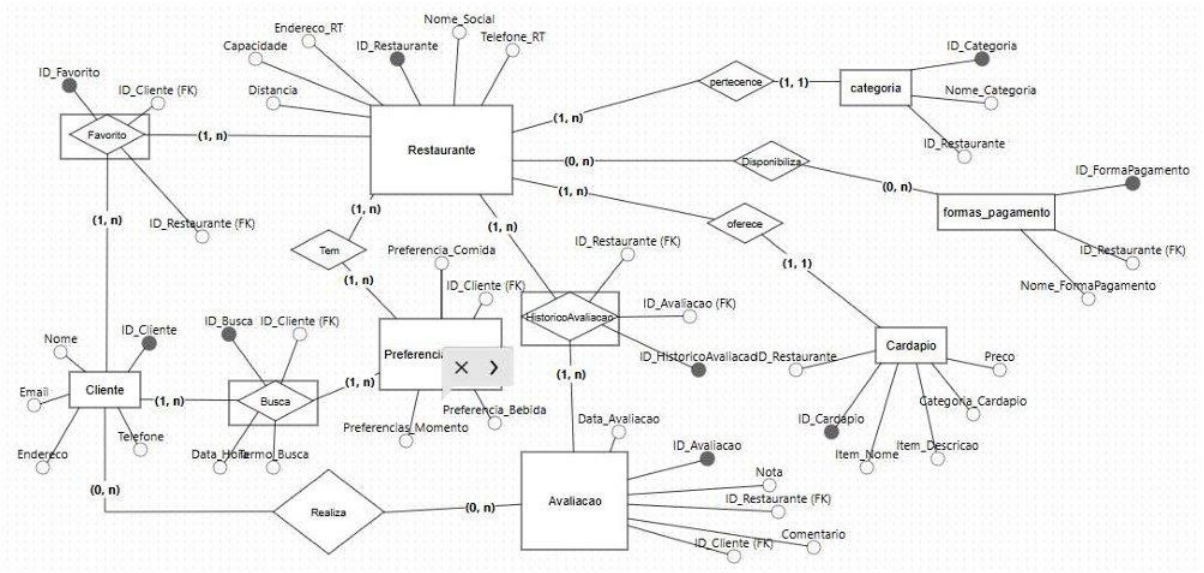

Figura 2 – Modelo DER

5.2. MER

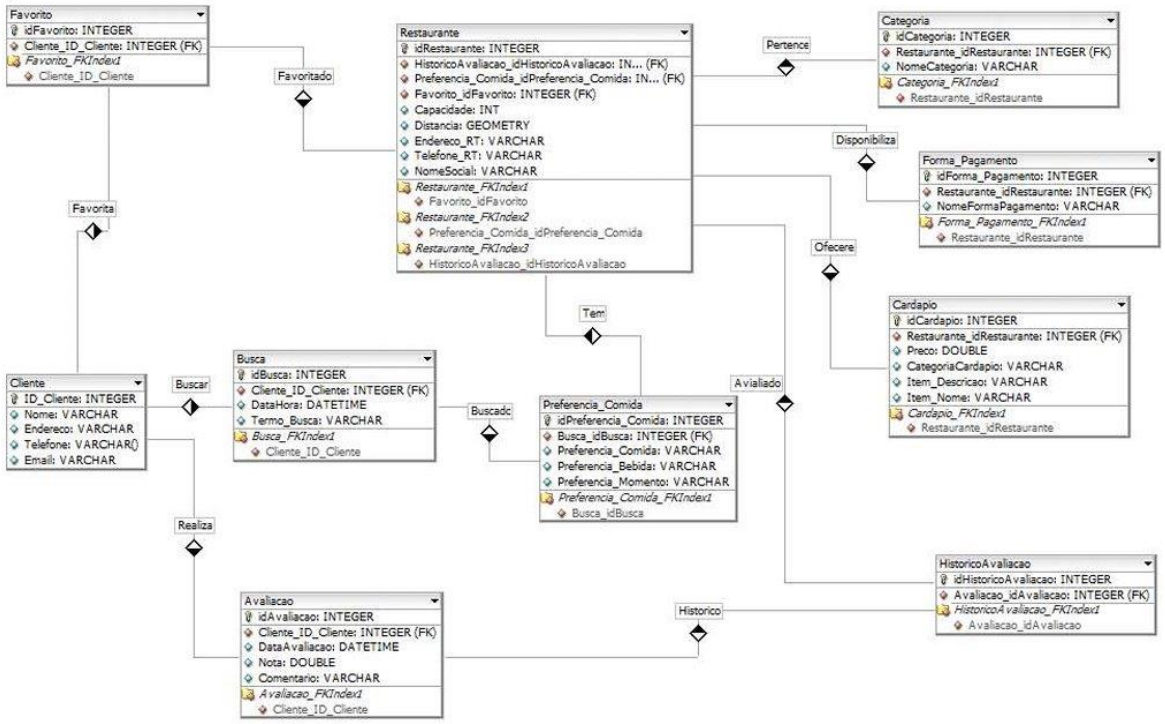

Figura 3 – Modelo MER