

ETAPA 4

Trabalho Prático - Engenharia de Software - GuiaDê

Nomes: João Pedro Licks Corso, Maurício Ramos Araujo Martins, Arthur Lopes Sauer, Juliana Rodrigues de Vargas

1. Introdução

O GuiaDê é um sistema desenvolvido para facilitar o compartilhamento de informações e avaliações entre usuários, especialmente no que diz respeito a profissionais e serviços em áreas de interesse específicas. O sistema oferece uma plataforma intuitiva e segura, onde os usuários podem interagir, trocar opiniões, e tomar decisões baseadas em feedbacks confiáveis, criando uma comunidade engajada e informada.

1.1 Problema a ser solucionado

A dificuldade enfrentada pelos usuários em encontrar informações confiáveis e avaliações detalhadas sobre profissionais e serviços em suas áreas de interesse.

1.2 Escopo do Produto

- **Funcionalidades principais:**
 - Autenticação de Usuários;
 - Criação de Guias para a avaliação de profissionais;
 - Visualização de avaliações em comunidades;
 - Publicação de opiniões em comunidades;
- **Funcionalidades secundárias:**
 - Recurso de pesquisa;
- **Fora do Escopo (Funcionalidades descartadas):**
 - Ferramentas para moderação da comunidade;

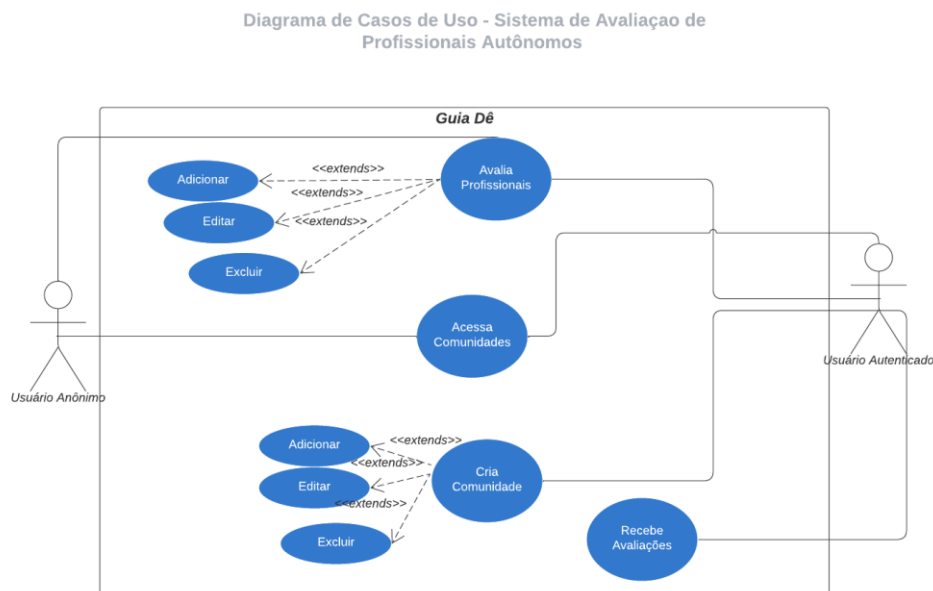
1.3 Públicos-Alvo

- **Usuários Comuns:** Indivíduos interessados em encontrar informações e avaliações sobre profissionais e serviços em suas áreas de interesse. Eles podem ser clientes em potencial, procurando recomendações ou avaliações antes de contratar um profissional ou serviço específico.
- **Profissionais Avaliados:** Profissionais ou prestadores de serviços que são avaliados e revisados pelos usuários. Eles podem usar o sistema para monitorar suas avaliações, bem como para gerenciar sua reputação online.
- **Moderadores/Proprietários do Espaço:** Indivíduos responsáveis por criar comunidades dentro do sistema. Eles podem ser profissionais ou especialistas em

determinados campos que desejam fornecer um espaço para discussão e avaliação em sua área de atuação, como também usuários comuns buscando opiniões

2. Casos de Uso e Requisitos

Os casos de uso do sistema GuiaDê foram elaborados para detalhar as principais interações entre os usuários e o sistema. Eles descrevem, de forma clara e estruturada, como os usuários podem utilizar as funcionalidades oferecidas pelo GuiaDê para atingir seus objetivos, seja pesquisando guias, criando novos conteúdos, ou deixando opiniões sobre serviços e profissionais. Esses casos de uso servem como uma base fundamental para o desenvolvimento e a validação do sistema, garantindo que todas as necessidades dos usuários sejam devidamente atendidas.



- **Legenda:** Identificação dos casos de uso implementados no protótipo e no MVP.

2.1 Caso de Uso 1: Pesquisar e Ler Recomendações de um Guia

Nome: Pesquisar Guia e Ler Recomendações

Atores: Usuário Comum (logado ou não)

Objetivo: Permitir que o usuário pesquise por guias na barra de pesquisa, escolha um guia de seu interesse e leia as recomendações associadas.

- **Pré-condições:**
 - O sistema deve estar acessível ao usuário.
 - O guia pesquisado deve estar disponível na plataforma.

- **Pós-condições:**
O usuário visualiza as recomendações associadas ao guia escolhido.
- **Fluxo Principal:**
 1. O usuário acessa a página inicial do sistema.
 2. O usuário insere uma palavra-chave na barra de pesquisa.
 3. O sistema exibe uma lista de guias relacionados à palavra-chave inserida.
 4. O usuário seleciona um guia da lista.
 5. O sistema exibe as recomendações e avaliações associadas ao guia escolhido.
 6. O usuário lê as recomendações.
- **Fluxos Alternativos:**
 - 3a. Se a palavra-chave não encontrar correspondências:
O sistema exibe mostra uma mensagem que não foram encontrados guias relacionados.
O usuário pode refinar a pesquisa ou inserir uma nova palavra-chave.

2.2 Caso de Uso 2: Criar um Guia e Disponibilizá-lo para Pesquisa

Nome: Criar Guia Disponível para Pesquisa

Atores: Usuário Logado com Google

Objetivo: Permitir que um usuário logado com sua conta Google crie um guia que ficará disponível para ser pesquisado por outros usuários.

- **Pré-condições:**
O usuário deve estar autenticado no sistema via Google.
O usuário deve ter permissão para criar guias.
- **Pós-condições:**
O guia criado pelo usuário estará disponível para pesquisa e visualização na plataforma.
- **Fluxo Principal:**
 1. O usuário loga na aplicação utilizando sua conta Google.
 2. O sistema autentica o usuário e exibe a interface principal.
 3. O usuário seleciona a opção de criar um novo guia, disponível na tela abaixo da barra de pesquisa.
 4. O usuário insere as informações necessárias para a criação do guia (ex.: título, descrição, seções).

5. O usuário confirma a criação do guia.
6. O sistema salva o guia e o disponibiliza para a pesquisa.

- **Fluxos Alternativos:**

- 3a. Se o usuário não tiver permissão para criar guias:
 - O usuário não encontra a opção de criar guia.
 - O usuário faz login e encontra a opção.

2.3 Caso de Uso 3: Deixar Opinião em um Guia como Usuário Anônimo

Nome: Deixar Opinião em um Guia como Usuário Anônimo

Atores: Usuário Anônimo

Objetivo: Permitir que um usuário anônimo selecione um guia disponível na página inicial, acesse-o e deixe sua opinião.

- **Pré-condições:**

O guia escolhido deve permitir a inserção de opiniões por usuários anônimos.
O sistema deve estar acessível ao usuário.

- **Pós-condições:**

A opinião do usuário anônimo é salva e fica disponível para outros usuários no guia.

- **Fluxo Principal:**

1. O usuário acessa a página inicial da aplicação.
2. O sistema exibe uma lista de guias disponíveis na home.
3. O usuário anônimo seleciona um guia de interesse.
4. O sistema exibe as recomendações e a opção de deixar uma opinião.
5. O usuário anônimo insere sua opinião no campo correspondente.
6. O usuário confirma o envio da opinião.
7. O sistema salva a opinião e a disponibiliza para outros usuários.

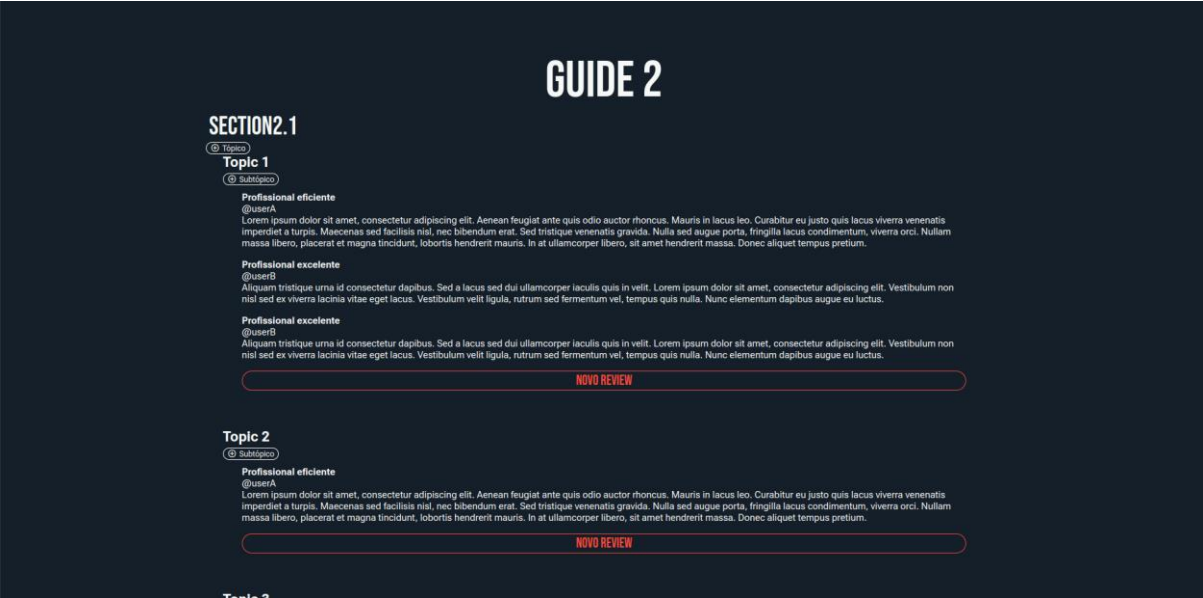
- **Fluxos Alternativos:**

- 4a. Se o guia não permitir opiniões de usuários anônimos:
 - O sistema exibe uma mensagem informando que apenas usuários logados podem deixar opiniões.
 - O usuário pode optar por logar no sistema ou visualizar apenas as opiniões existentes.

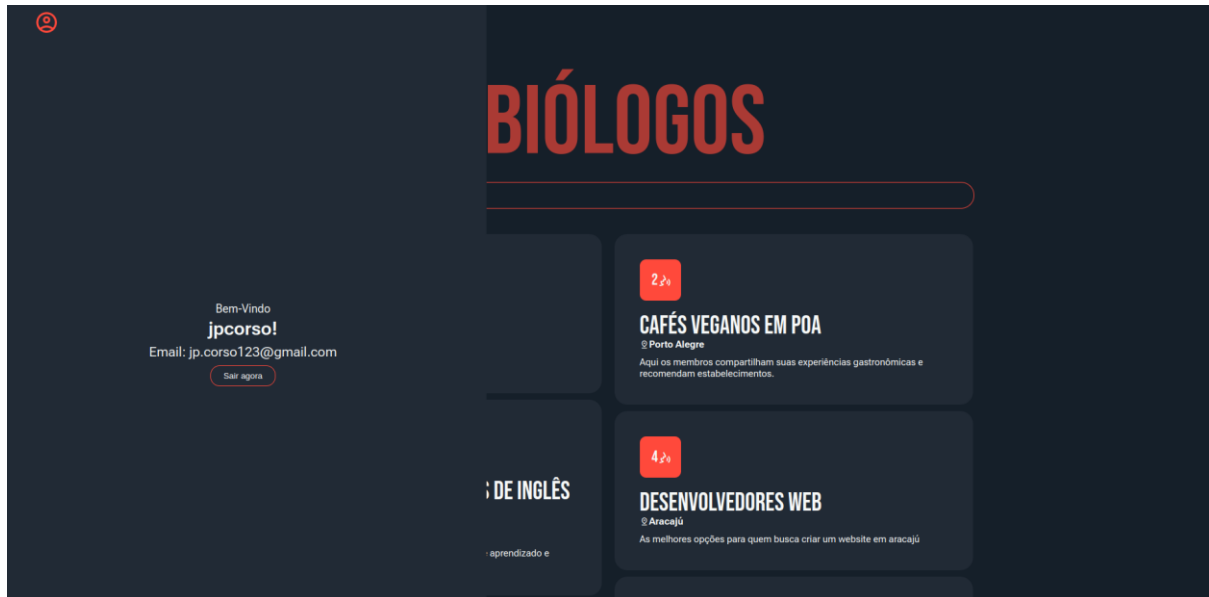
2.4 Casos de Uso Priorizados

Para a aplicação, foram priorizados e implementados os casos de uso dos diagramas de uso 1 e 2.

2.4.1 Implementação do Caso de Uso 1



2.4.2 Implementação do Caso de Uso 2




2.5 Requisitos da Especificação

Funcionais	Não Funcionais
RF1 – A aplicação deve permitir o cadastro dos dados de identificação dos Usuários.	RNF1 – Autenticação deve ser segura, para evitar vazamento de dados do usuário
RF2 - A aplicação deve permitir que os usuários criem e gerenciem comunidades de profissionais.	RNF2 – Se adaptar a diferentes dispositivos e tamanhos de tela, incluindo desktops, laptops, tablets e smartphones
RF3 - A aplicação deve permitir a consulta de comunidades e <i>profissionais cadastrados na plataforma</i> .	RNF3 – Esteja em conformidade com a LGPD
RF4 – A aplicação deve permitir que os usuários visualizem as avaliações realizadas por outros usuários nas comunidades em que tiver acesso.	RNF4 - Sistema em conformidade com padrões de acessibilidade
RF5 - A aplicação deve permitir que os usuários criem entradas para compartilhar avaliações, opiniões ou análises sobre profissionais e serviços em comunidades específicas.	RNF5 - Sistema com boa performance de carregamento, com bom desempenho e responsivo
RF6 -A aplicação deve permitir que os usuários criem grupos de avaliações e deem acesso para pessoas selecionadas	
RF7 - A aplicação deve possuir ferramentas que permitam aos usuários moderadores a criação, gerenciamento e configuração de comunidades, configurando visibilidade, permissões e outros aspectos relevantes.	

3. Negócio


3.1 Benchmarking de Soluções Similares

1. Reddit

	
Características da aplicação:	Comparação com nosso serviço:
Comunidades específicas e moderadas, mas sem foco na avaliação de serviços e profissionais.	O serviço será voltado para um público específico.

Os usuários podem criar postagens em texto, links, imagens ou vídeos, e outros usuários podem comentar sobre essas postagens, criando discussões	As discussões serão o foco dos espaços de compartilhamento de informações. O principal conteúdo será diretamente as opiniões dos usuários, facilitando o acesso à essa informação.
Os usuários podem criar contas sem a necessidade de divulgar informações pessoais, o que permite a participação anônima.	Assim como no Reddit, será possível publicar de forma anônima ou pública, a depender das configurações selecionadas pelo criador do espaço.
Usuários podem votar positivamente (upvote) ou negativamente (downvote) nas postagens e comentários. Esse sistema de votação ajuda a destacar o conteúdo mais popular e relevante.	As avaliações serão ordenadas em ordem cronológica e isso possibilitará acompanhar a evolução do profissional ou serviço. Entretanto, poderá haver sinalização das avaliações mais bem recebidas pelos usuários do espaço.

2. GetNinjas

	
Características da aplicação:	Comparação com nosso serviço:
Foco na avaliação e recomendação de profissionais	Além da avaliação de profissionais, o foco também se dividirá em avaliação de serviços.
Profissionais de diferentes áreas podem se cadastrar, criar um perfil detalhado e oferecer seus serviços.	A abordagem da aplicação prioriza a opinião escrita dos usuários e o compartilhamento de informações entre uma comunidade, ao invés de um perfil detalhado dos profissionais.
Clientes podem descrever suas necessidades e receber orçamentos de vários profissionais cadastrados na plataforma.	A plataforma prioriza conectar clientes que podem avaliar com clientes interessados em um serviço, e não clientes e profissionais diretamente.
Os clientes podem usar filtros para encontrar exatamente o tipo de serviço e profissional que estão procurando.	

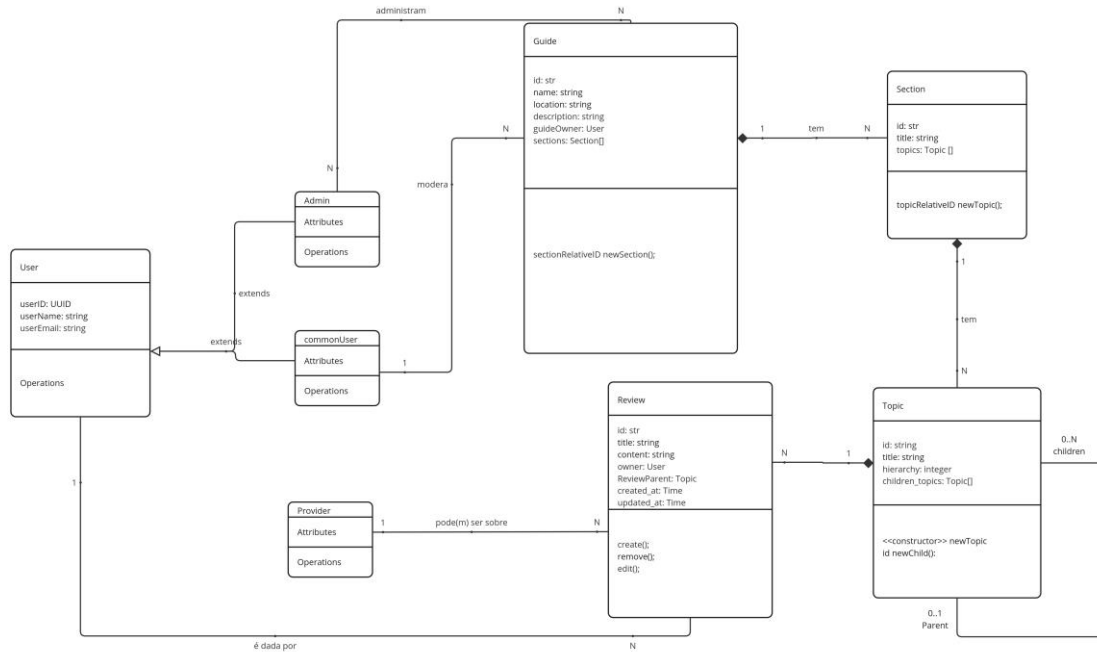
3. Triider



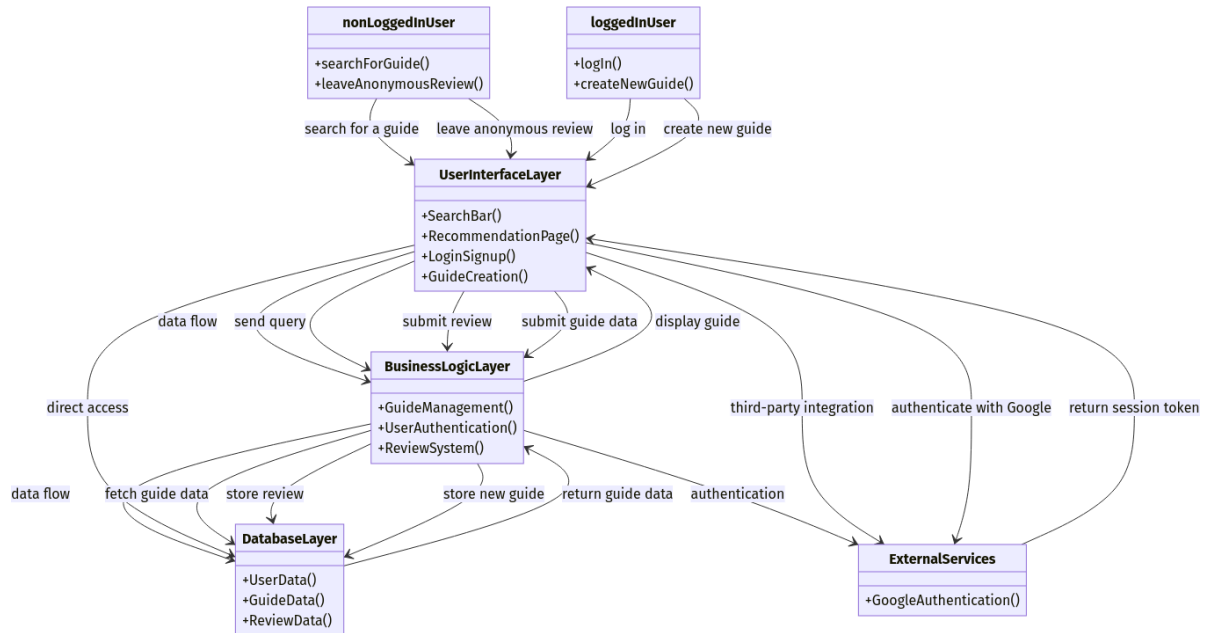
Características da aplicação:	Comparação com nosso serviço:
Especializado em serviços como reformas, manutenção, instalações elétricas, hidráulicas, entre outros.	Aberto para diversos tipos de serviços e também profissionais.
Permite que os clientes solicitem e agendem serviços de maneira prática e rápida.	Não se propõe a possibilitar agendamentos, mas o número do profissional poderá ficar disponível para os usuários caso o profissional seja mencionado em uma avaliação e tenha o selo pago.
Clientes podem avaliar os serviços realizados, ajudando outros usuários a escolherem os melhores profissionais, mas a avaliação não é o foco do serviço. .	Avaliação é o foco do serviço.

4. Modelagem

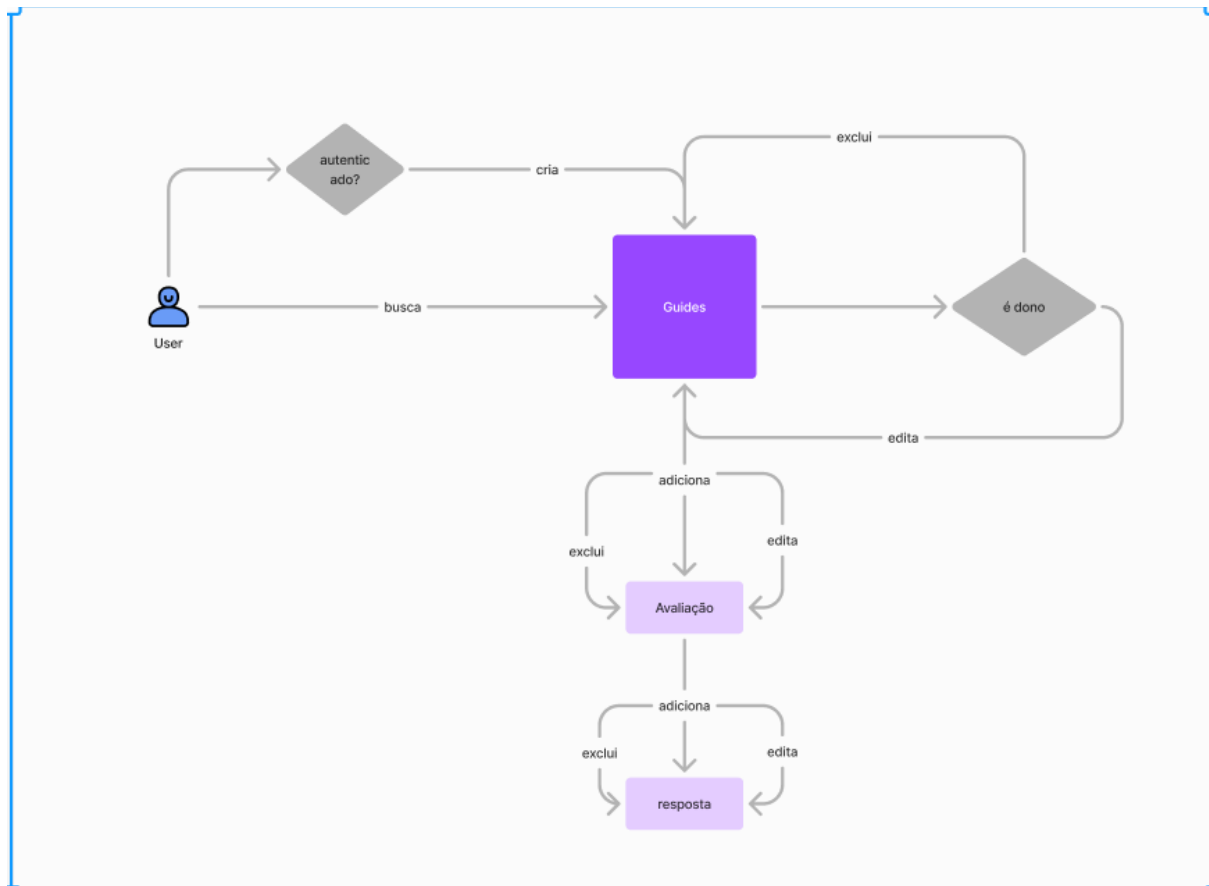
4.1 Diagrama de Classes



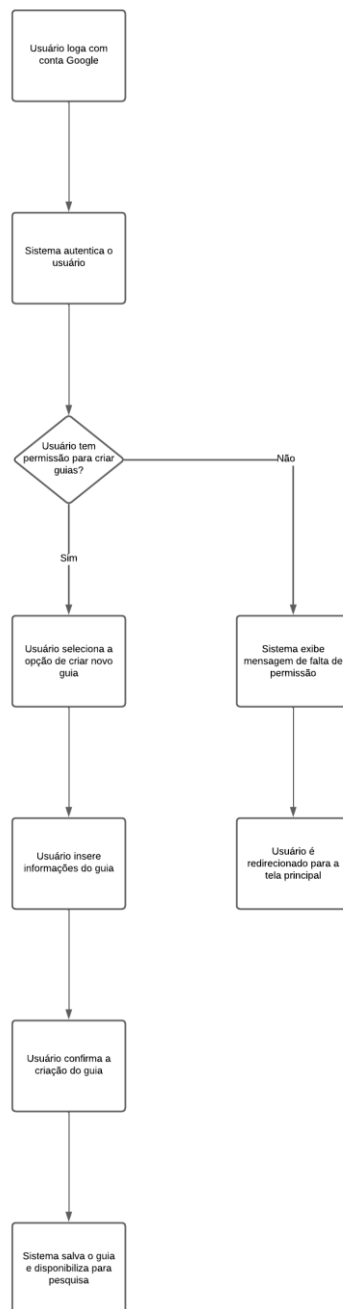
4.2 Diagrama de Pacotes



4.3 Diagrama de Sequência



4.4 Diagrama de Atividades



5. Tecnologias e Arquitetura

5.1 Tecnologias Selecionadas

Frontend

React

React é utilizado para construir a interface do usuário de forma dinâmica e interativa.

- **Motivação:** React é uma das bibliotecas de JavaScript mais populares para a criação de interfaces de usuário. Ela permite a construção de aplicativos web altamente interativos e escaláveis, com uma arquitetura baseada em componentes reutilizáveis.
- **Prós:**
 - **Componentização:** Permite a criação de componentes reutilizáveis, facilitando a manutenção e expansão do sistema.
 - **Grande Ecossistema:** Possui uma vasta quantidade de bibliotecas e ferramentas disponíveis, como React Router para roteamento e Redux para gerenciamento de estado.
 - **Desempenho:** Utiliza o Virtual DOM, melhorando o desempenho ao minimizar atualizações reais no DOM.
 - **Popularidade:** Grande comunidade de desenvolvedores e suporte de grandes empresas, facilitando encontrar soluções e bibliotecas.
- **Contras:**
 - **Curva de Aprendizado:** Pode ser desafiador para iniciantes, especialmente ao lidar com conceitos como hooks, estado global e gerenciamento de efeitos colaterais.
 - **Manutenção do Estado:** Para aplicativos grandes, o gerenciamento de estado pode se tornar complexo sem ferramentas adicionais como Redux ou Context API.

Backend

FastAPI

O backend foi desenvolvido em Python, utilizando o FastAPI para gerenciar as requisições HTTP recebidas do frontend.

- **Motivação:** FastAPI é um framework moderno, rápido e eficiente para construir APIs com Python. Ele é baseado em padrões como OpenAPI e JSON Schema, facilitando a criação de APIs bem documentadas e seguras.
- **Prós:**
 - **Desempenho:** Extremamente rápido e comparável a frameworks como Node.js e Go em termos de performance, graças ao uso de Python assíncrono (async/await).
 - **Simplicidade e Flexibilidade:** Combina a simplicidade do Python com a performance necessária para construir sistemas escaláveis.
 - **Popularidade Crescente:** Tem sido amplamente adotado e está ganhando popularidade, com boa documentação e suporte.
- **Contras:**
 - **Dependência de Async:** O uso intensivo de async/await pode adicionar complexidade ao código para desenvolvedores que não estão familiarizados com programação assíncrona.

- **Menos Maturidade:** Apesar de estar crescendo rapidamente, ainda pode não ter a maturidade de frameworks mais antigos como Django ou Flask em termos de bibliotecas e extensões prontas.

Bibliotecas Adicionais

- **pytest:** Utilizado para criar e rodar testes automatizados, garantindo a qualidade do código e facilitando a detecção de bugs.
- **uuid4:** Gera identificadores únicos universalmente (UUIDs), muito úteis para criar IDs únicos para objetos ou registros.
- **datetime:** Fornece classes para manipulação de datas e horas, essencial para qualquer aplicação que lida com tempo ou agendamentos.
- **BaseModel:** Utilizado para criar classes de modelos de dados que garantem a validação de dados de entrada e saída em APIs, facilitando a conversão de dados entre diferentes tipos.
- **abstractmethod:** Módulo que permite criar classes base abstratas, essenciais para definir interfaces e garantir que subclasses implementem métodos específicos.

A. Banco de Dados

MongoDB (Banco de Dados)

Armazena os dados do nosso sistema em documentos flexíveis, permitindo a rápida inserção e consulta de dados sem a rigidez de outros bancos de dados mais clássicos.

- **Motivação:** MongoDB é um banco de dados NoSQL orientado a documentos que armazena dados em formato JSON-like. Ele é ideal para aplicativos que lidam com dados semi-estruturados e que precisam de flexibilidade no esquema de dados.

- **Prós**

Flexibilidade de Esquema: Permite armazenar documentos com estrutura flexível, ideal para cenários onde os dados podem variar entre documentos.

Escalabilidade: Foi projetado para escalar horizontalmente, facilitando o crescimento do banco conforme o volume de dados aumenta.

JSON-like Storage: A facilidade de armazenamento em formato JSON-like facilita a integração com APIs RESTful e outras ferramentas modernas.

- **Contras**

Consistência Eventual: Em sistemas distribuídos, MongoDB pode operar em consistência eventual, o que pode não ser adequado para aplicações que exigem forte consistência transacional.

Curva de Aprendizado para Indexação: O gerenciamento de índices e otimização de consultas pode ser mais complexo do que em bancos de dados relacionais.

B. Serviços Externos

Google Authentication Service

O serviço de autenticação do Google é integrado no nosso sistema para fornecer um método seguro e fácil de login para os usuários.

- **Motivação:** O serviço de autenticação do Google permite que os usuários façam login usando suas contas do Google, simplificando o processo de autenticação e evitando a necessidade de gerenciar credenciais de login diretamente.

- **Prós**

Segurança: Os usuários podem confiar no sistema de autenticação do Google, que oferece uma camada extra de segurança, incluindo autenticação multifator.

Facilidade de Implementação: Simplifica o gerenciamento de usuários, já que o Google lida com a autenticação, verificação de identidade e recuperação de senhas.

Popularidade: Muitos usuários já possuem contas do Google, o que torna o processo de login rápido e fácil para eles.

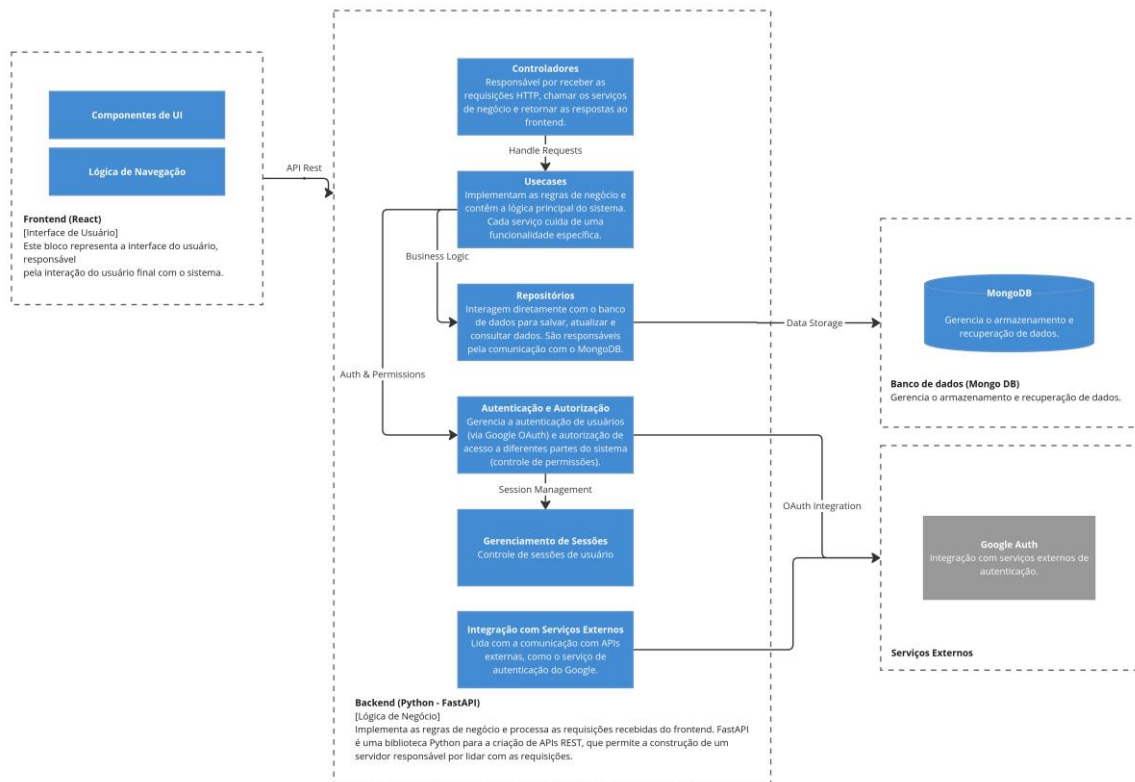
- **Contras**

Dependência Externa: Você depende de um serviço externo para a autenticação, o que significa que qualquer problema no serviço do Google pode afetar o acesso ao seu sistema.

Privacidade: Alguns usuários podem não querer vincular sua conta do Google a outros serviços devido a preocupações com privacidade.

Custo: Dependendo do volume de usuários, o uso do Google Authentication pode ter custos associados.

5.2 Arquitetura do Sistema



- Esquema ilustrativo mostrando como as tecnologias (frontend, backend, banco de dados) estão integradas.

6. Implementação

6.1 Casos de Uso Implementados

- Descrição dos dois casos de uso implementados (excluindo cadastro/login).
- Explicação de como os casos de uso foram implementados no código.

6.2 Repositório GitHub

- <https://github.com/ju-vargas/TP-engenharia-de-software>
- **Instruções de Uso**
 - **1. Configuração do Servidor FastAPI**
 - **Instalar as Bibliotecas Necessárias**
 - Navegue até a pasta python-server.
 - Crie um ambiente virtual em Python executando o comando:
python -m venv venv
 - **Ative o ambiente virtual:**
 - No Windows: .\venv\Scripts\activate
 - No Linux: source ./venv/bin/activate

- Instale as bibliotecas necessárias com o comando: `pip install -r requirements.txt`
- **Iniciar o Servidor FastAPI**
 - Execute o script `start.sh` para iniciar o servidor: `sh start.sh`
- **2. Configuração do Front-End**
 - **Instalar as Dependências**
 - Navegue até a pasta `guia-de`.
 - Instale as dependências do front-end executando: `npm install`
 - **Iniciar o Front-End**
 - Para iniciar o front-end, execute: `npm start`
- **Acessar a Aplicação**
 - Abra a URL gerada no terminal após a inicialização do front-end para acessar a aplicação.

7. Conclusão

Etapa 1 e 2- Especificação dos requisitos:

Necessidades dos Usuários: Identificamos as demandas principais dos usuários;

Requisitos Funcionais e Não Funcionais: Definimos as funcionalidades essenciais do sistema.

Diagrama de classes e casos de uso.

Usuários do Sistema: Categorizamos os tipos de usuários.

Modelo de Negócio: Análise do negócio e aplicações semelhantes.

Protótipo do sistema

Etapa 3:

Definição das tecnologias: Motivação para escolha das tecnologias.

Diagrama de arquitetura.

Considerações Finais:

Todas as etapas de desenvolvimento deste projeto foram enriquecedoras, pois aprofundamos nosso conhecimento em diversas tecnologias e em práticas de engenharia de software.

Conhecimentos adquiridos:

Importância dos requisitos: Percebemos que a definição clara dos requisitos e boa documentação facilita muito para a parte de implementação.

Integração de Tecnologias: A importância de analisar bem as tecnologias utilizadas para se adequar as especificações.

Desafios da arquitetura: Compreendemos os desafios de implementar uma arquitetura em camadas, especialmente em termos de modularidade e separação de responsabilidades.