



Programação Distribuída (CP406TIN1)

Guilherme de Oliveira Chaguri 190356

Guilherme Koji Yamada 173271

Higor da Silva Lins 190218

Matheus Jacob Bendel 190299

Documentação Sistema de Turismo

**SOROCABA, SP
2023**

1 INTRODUÇÃO

Com o avanço da tecnologia e a popularização da internet, o desenvolvimento de aplicações web se tornou cada vez mais comum. Duas das tecnologias mais utilizadas nesse contexto são os webservices e webapps.

Os webservices são uma solução para a integração entre diferentes sistemas. Eles permitem que diferentes aplicações possam trocar informações de forma padronizada e segura. Essa troca de informações é realizada através de protocolos como o HTTP, que é amplamente utilizado na internet. Dessa forma, um sistema pode utilizar os serviços de outro sistema sem precisar conhecer detalhes de sua implementação. Por exemplo, um sistema de vendas online pode utilizar um webservice de uma transportadora para calcular o valor do frete de um produto sem precisar conhecer os detalhes de como a transportadora realiza esse cálculo (W3SCHOOLS, 2023).

Já os webapps são aplicativos que são acessados e executados através de um navegador web. Eles utilizam tecnologias como HTML, CSS e JavaScript para criar interfaces interativas e dinâmicas, permitindo aos usuários interagirem com o aplicativo através do navegador. Os webapps podem ser acessados em diferentes dispositivos, como desktops, notebooks, smartphones e tablets, sem a necessidade de instalação de softwares adicionais. Essa característica torna os webapps uma opção interessante para empresas que desejam disponibilizar seus serviços em diferentes plataformas (GINIGE, 2022).

Tanto os webservices quanto os webapps são amplamente utilizados no desenvolvimento de aplicações web modernas. Eles permitem a criação de sistemas mais robustos e escaláveis, além de facilitar a integração entre diferentes sistemas. No entanto, é importante destacar que essas tecnologias apresentam desafios em relação à segurança e desempenho, que devem ser considerados durante o processo de desenvolvimento.

Em resumo, os webservices e webapps são tecnologias fundamentais no desenvolvimento de aplicações web modernas. Eles permitem a integração entre diferentes sistemas e a criação de interfaces interativas e dinâmicas que podem ser acessadas em diferentes dispositivos. No entanto, é importante que os desenvolvedores estejam atentos aos desafios relacionados à segurança e desempenho dessas tecnologias (GINIGE, 2022; W3SCHOOLS, 2023).

2 APRESENTAÇÃO

Nesse projeto iremos desenvolver um sistema de turismo que integra várias APIs externas, como uma de hotel, evento e de passagem aérea.

O principal objetivo do WebApp é o gerenciamento de pacotes de turismo. Esses pacotes podem ser configurados de diferentes maneiras, com combinações variadas de produtos disponibilizados por APIs externas. Em outras palavras, um pacote basicamente armazena referências para cada vínculo de produto com as outras APIs. Essa abordagem permite que os usuários tenham acesso a uma ampla variedade de produtos e serviços turísticos em um único pacote conveniente.

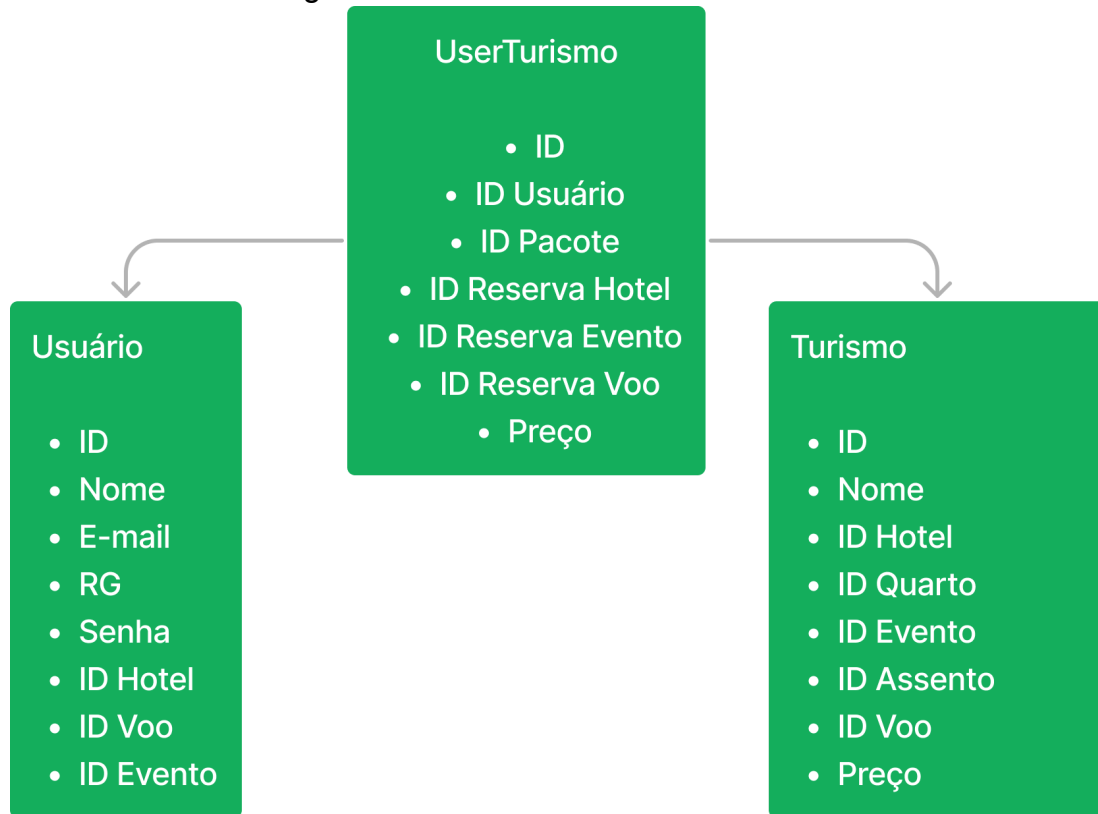
Existem várias ações que os usuários podem realizar no WebApp de Turismo. Eles podem pesquisar quartos por pacotes disponíveis com base em suas preferências. Após a pesquisa, os usuários podem visualizar informações detalhadas sobre cada pacote, incluindo preço, comodidades, dados do evento e dados do voo.

Uma vez que tenham encontrado um pacote adequado, os usuários têm a opção de adquirir aquele pacote para as datas desejadas. Além disso, eles podem gerenciar seus pacotes existentes, visualizar, modificar ou cancelar as mesmas.

A figura 1 representa as principais entidades envolvidas no gerenciamento do sistema de turismo. Cada entidade de turismo possui IDs externos que são utilizados para vincular informações. Quando um usuário adquire um pacote, um conjunto de rotas é acionado para agendar todas as atividades necessárias no fluxo do turismo. Os retornos dessas rotas são armazenados como referência para ações futuras.

Além disso, as outras entidades fazem referência ao usuário (User) e aos pacotes adquiridos por esse usuário (UserTurismo). Essa estrutura permite um controle eficiente das informações relacionadas aos usuários e aos pacotes de turismo, garantindo uma experiência personalizada e facilitando a administração do sistema.

Figura 1 - Estrutura base das entidades



Fonte: elaborado pelo autor

Para a implementação desse sistema, será utilizada uma aplicação NestJS, sendo um framework em Node.js. Como banco de dados, será utilizado o SQLite, que é uma opção adequada para esse nível de implementação. O uso do NestJS e do SQLite permitirá que o sistema seja desenvolvido de maneira eficiente, garantindo que todas as funcionalidades sejam implementadas com sucesso.

3 DESENVOLVIMENTO

Neste capítulo, serão apresentadas informações importantes sobre o processo de desenvolvimento da plataforma Web e da API.

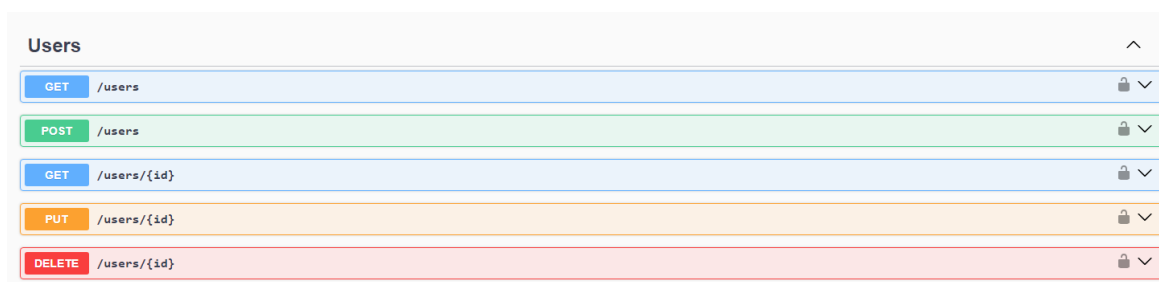
3.1 API

A API, como mencionado anteriormente, foi desenvolvida utilizando NestJs e SQLite. Seu principal objetivo é centralizar as operações do sistema de turismo, tornando-se a principal responsável por integrar as APIs de hotel, voo e eventos. O objetivo é garantir que todos os eventos estejam sincronizados com as expectativas do usuário.

Ao utilizar o NestJs e o SQLite, a API oferece uma estrutura robusta e eficiente para gerenciar as operações do sistema de turismo. Ela facilita a interação com as APIs externas de hotel, voo e eventos, permitindo que os dados sejam consolidados em um único local. Dessa forma, é possível garantir que todas as informações e eventos estejam atualizados e alinhados com as preferências do usuário, proporcionando uma experiência turística mais satisfatória.

Os principais módulos de rotas podem ser vistos na figura 2, figura 3 e figura 4;

Figura 2 - Módulo de Usuários



Users		^
GET	/users	🔒 ▼
POST	/users	🔒 ▼
GET	/users/{id}	🔒 ▼
PUT	/users/{id}	🔒 ▼
DELETE	/users/{id}	🔒 ▼

Fonte: elaborado pelo autor

Figura 3 - Módulo de Turismo

Turismo			^
GET	/turismo		🔒 ▼
POST	/turismo		🔒 ▼
GET	/turismo/{id}		🔒 ▼
PUT	/turismo/{id}		🔒 ▼
DELETE	/turismo/{id}		🔒 ▼

Fonte: elaborado pelo autor

Figura 4 - Módulos de User Turismo

User Turismo			^
GET	/user-turismo		🔒 ▼
POST	/user-turismo		🔒 ▼
GET	/user-turismo/{id}		🔒 ▼
PUT	/user-turismo/{id}		🔒 ▼
DELETE	/user-turismo/{id}		🔒 ▼

Fonte: elaborado pelo autor

Para isso as demandas de adquirir pacote foram criadas, rota de criação, consulta e cancelamento de pacote:

3.1.1 Rota de criação

A rota de criação é responsável por realizar a criação de todos os agendamentos em cada uma das APIs externas. Quando um usuário seleciona um pacote previamente configurado, a API estabelece uma conexão entre esse pacote e o usuário. Além disso, a API realiza chamadas para cada uma das APIs externas, permitindo que os agendamentos sejam feitos de acordo com as especificações do pacote.

Os retornos obtidos das APIs externas são armazenados para uso em operações futuras. Dessa forma, as informações dos agendamentos ficam registradas e podem ser acessadas posteriormente, se necessário. Essa abordagem garante a consistência e a integridade dos dados, permitindo que as operações no sistema de turismo sejam realizadas de forma eficiente e precisa.

3.1.2 Rota de consulta

A rota de consulta é responsável por fornecer informações sobre um pacote específico. Ao receber o ID do pacote, a API realiza uma chamada ao banco de dados para obter as informações necessárias. Com base nessas informações, é possível acessar os dados das APIs externas relacionadas ao pacote.

3.1.3 Rota de Cancelamento

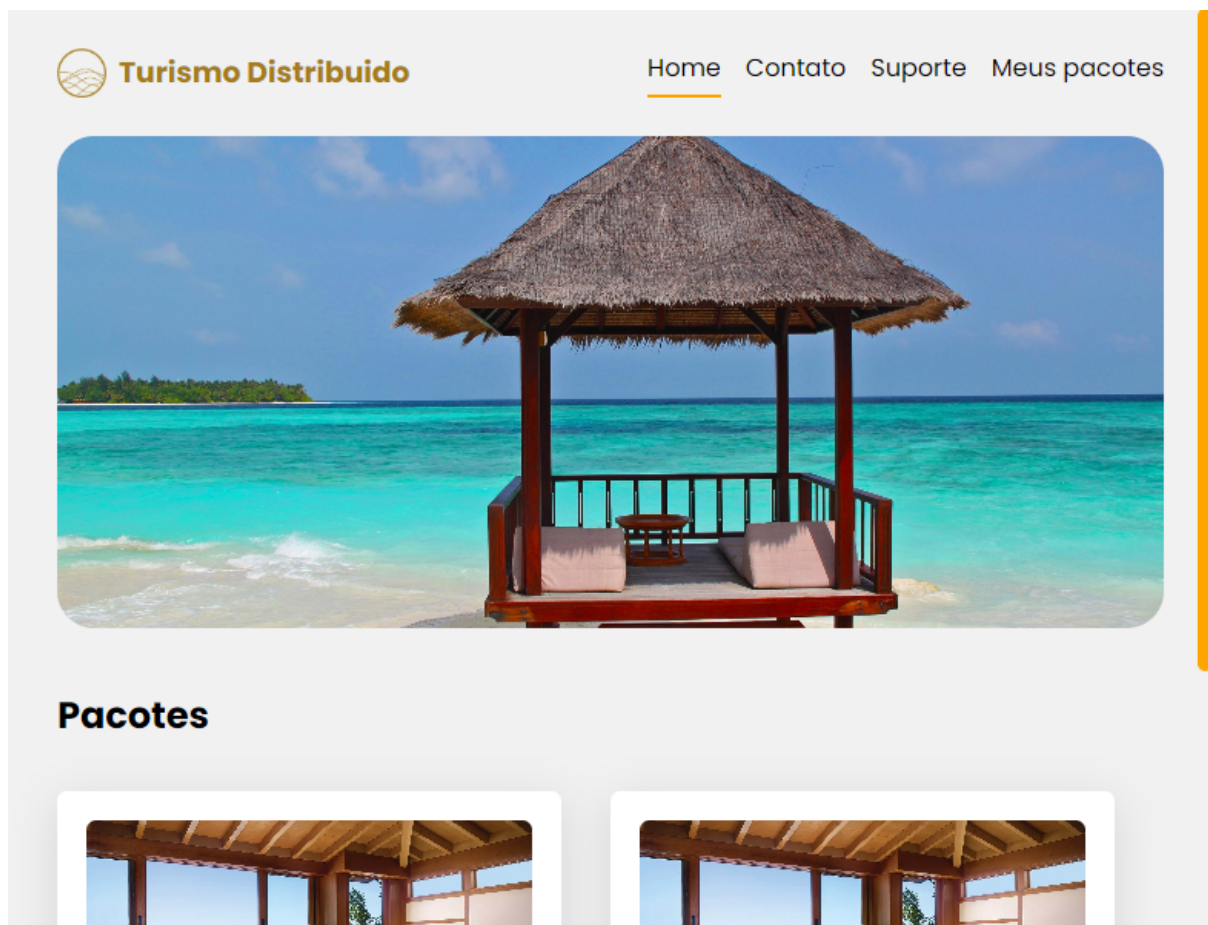
A rota de cancelamento funciona de forma que, ao receber o ID do pacote do usuário, desativa o pacote correspondente. Além disso, a API realiza chamadas às APIs externas utilizando os IDs relacionados, permitindo o cancelamento do pacote em si e de cada agendamento realizado nas outras plataformas.

Essa rota é essencial para fornecer aos usuários a capacidade de cancelar um pacote de turismo de forma conveniente. Ao realizar as chamadas às APIs externas, a API garante que todos os agendamentos sejam devidamente cancelados, evitando inconvenientes e assegurando que todas as partes envolvidas sejam notificadas sobre o cancelamento.

3.2 FRONT-END

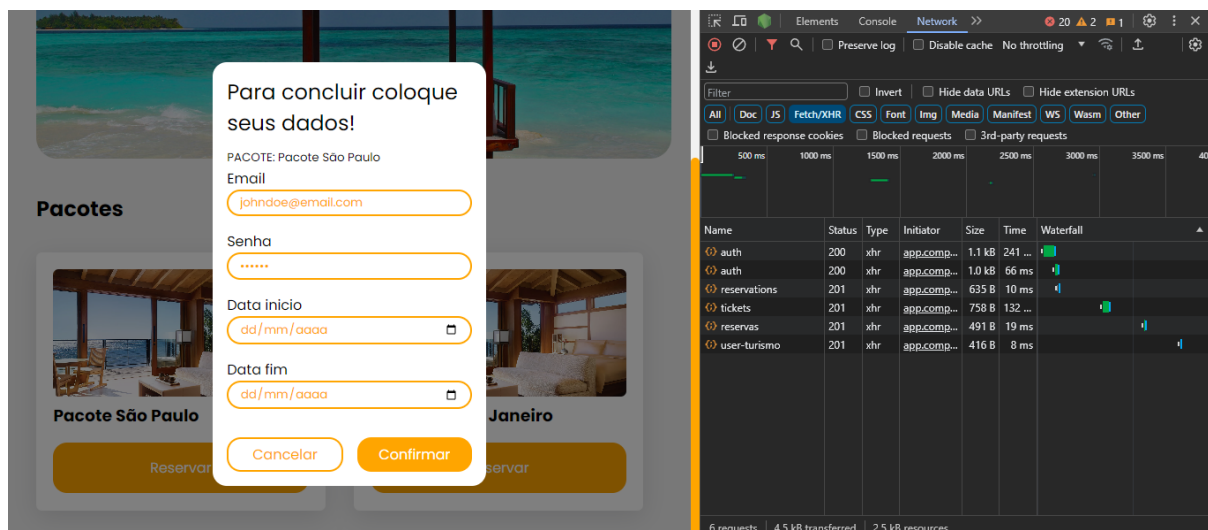
Abaixo temos a demonstração do resultado obtido no desenvolvimento do front-end;

Figura 5 - Tela Inicial



Fonte: elaborado pelo autor

Figura 6 - Adquirindo um pacote



Fonte: elaborado pelo autor

3.3 SERVIÇOS EXTERNOS

Para consumir cada serviço de plataforma externa, foi adotada uma estratégia específica para cada um. Inicialmente, foi necessário configurar um banco de dados para cada aplicação. Algumas delas exigiram o cadastro de dados fictícios, pois não possuíam dados salvos ou scripts disponíveis para esse propósito. Apenas a API de Hotel desenvolvida já possuía esses dados.

Em seguida, no processo de integração, as três APIs foram executadas localmente e configuradas para acessar corretamente os respectivos endpoints locais. Com base na documentação fornecida por cada API, foram geradas as validações necessárias para o processo de autenticação. Algumas APIs não exigiam autenticação, permitindo o consumo livre das rotas.

Já nas APIs que possuíam autenticação, como a API de Hotel, os dados de acesso foram salvos para gerar um token de autenticação e enviá-lo junto com as chamadas às outras rotas. Essa abordagem permitiu centralizar o processo de autenticação e garantir a segurança dos dados. Para as demais APIs, apenas foi necessário consumir e salvar as referências necessárias na API centralizada de Turismo.

Dessa forma, o processo de integração com os serviços externos foi realizado de maneira eficiente e segura, permitindo a utilização dos recursos oferecidos por cada plataforma e a consolidação dos dados relevantes na API centralizada de Turismo.

3.3 ACID

Durante o desenvolvimento do projeto, houve um foco em aplicar os conceitos de ACID (Atomicidade, Consistência, Isolamento e Durabilidade) na medida do possível. No entanto, surgiram desafios significativos na implementação desses conceitos, uma vez que exigiria que todas as APIs envolvidas estivessem comprometidas com esses princípios. Infelizmente, nem todas as APIs aderiram a esses conceitos, tornando especialmente difícil a implementação da atomicidade.

Para abordar essas limitações, foram estabelecidas algumas regras para cada um dos conceitos ACID. No entanto, apenas uma parte dessas regras pôde ser desenvolvida devido às restrições impostas pelas APIs externas. Por exemplo, a

atomicidade exigiria rotas que permitissem um rollback ou tratamento de erros mais adequado, o que não era viável de ser implementado.

Apesar dessas dificuldades, foi feito o máximo para garantir a consistência e a durabilidade dos dados nas operações realizadas no sistema de turismo.

Primeiramente, a ideia de atomicidade poderia ter sido aplicada para garantir que todas as operações relacionadas aos serviços do pacote fossem tratadas como uma única unidade lógica. Isso significa que, se uma operação falhar em qualquer um dos serviços, todas as alterações realizadas serão desfeitas, mantendo os dados em um estado consistente.

Em seguida, a consistência foi assegurada por meio de regras e restrições definidas no sistema. Isso garante que apenas transações válidas e consistentes sejam permitidas, evitando estados inconsistentes nos dados.

O isolamento para garantir que as transações ocorram de forma independente e não interfiram umas nas outras. Isso evita conflitos e problemas de concorrência entre as operações realizadas nos serviços do pacote.

Por fim, a durabilidade é garantida por meio do uso de técnicas de persistência de dados, garantindo que as alterações realizadas nas operações sejam permanentemente armazenadas e recuperáveis, mesmo em caso de falhas no sistema. Por se tratar de uma aplicação de demonstração, não é possível garantir a persistência dos dados sem um banco de dados com dados instalado em algum sistema de nuvem. Embora o desenvolvimento tenha sido realizado levando em consideração essas práticas, somente testes reais poderiam confirmar a efetividade dessa abordagem.

REFERÊNCIAS

GINIGE, A.; MURUGESAN, S. **Web engineering: an introduction**. IEEE MultiMedia, v. 8, n. 1, p. 14-18, Jan.-March 2001. DOI: 10.1109/93.923949.

W3SCHOOLS. **Web Services Tutorial**. Disponível em:
https://www.w3schools.com/xml/xml_services.asp. Acesso em: 5 nov. 2023.