



Programação Distribuída (CP406TIN1)

Guilherme de Oliveira Chaguri 190356

Guilherme Koji Yamada 173271

Higor da Silva Lins 190218

Matheus Jacob Bendel 190299

Documentação API de HOTEL

**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

O WebApp escolhido é o de Hotéis. Ele é uma plataforma que permite aos usuários encontrar e reservar quartos de hotel de acordo com suas preferências.

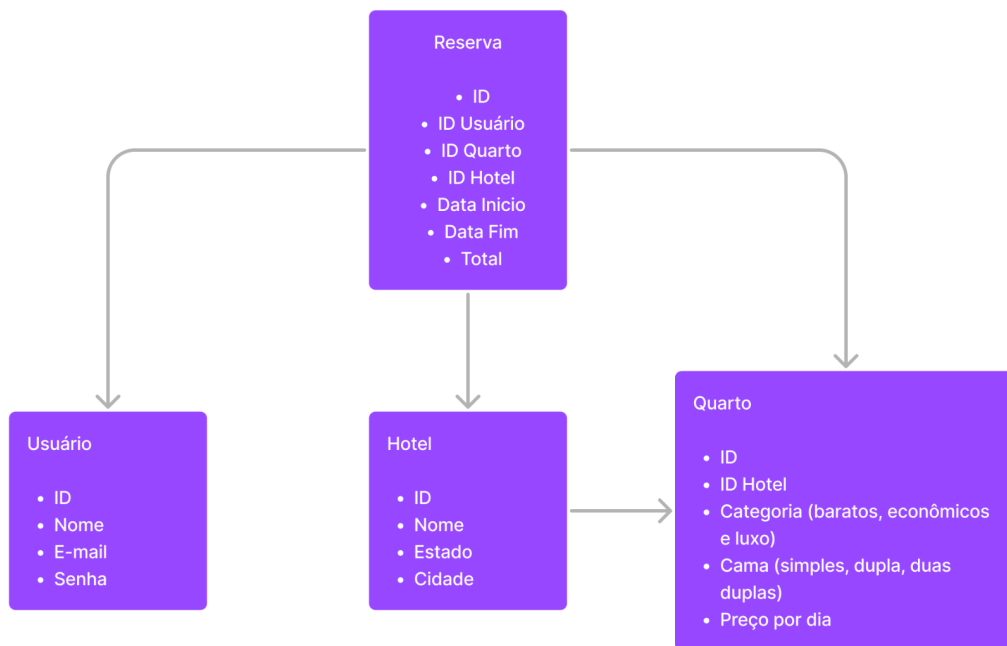
O principal objeto do WebApp são os quartos de hotel. Os quartos podem ter diferentes configurações, como 1 cama simples (para uma pessoa), 1 cama dupla (para duas pessoas) ou 2 camas simples (para duas pessoas). Além disso, os quartos podem ser categorizados como baratos, econômicos ou de luxo.

Existem várias ações que os usuários podem realizar no WebApp Hotéis. Eles podem pesquisar quartos disponíveis com base em suas preferências, como o número de camas e a categoria. Após a pesquisa, os usuários podem visualizar informações detalhadas sobre cada quarto, incluindo preço, comodidades.

Uma vez que tenham encontrado um quarto adequado, os usuários têm a opção de fazer uma reserva para as datas desejadas. Além disso, eles podem gerenciar suas reservas existentes, visualizando, modificando ou cancelando as mesmas.

A figura 1 representa as 4 entidades principais que gerenciam o sistema de hotel. Cada hotel tem vários quartos, e quando um usuário faz uma reserva, as informações relevantes são salvas, incluindo o ID do usuário, o ID do quarto, o ID do hotel, datas de check-in e check-out, preço e outras informações importantes. As entidades incluem hotéis, quartos, usuários e reservas, que trabalham em conjunto para garantir um funcionamento eficiente 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 contrato e a documentação da Interface Description Language (IDL) que descreve o serviço de API para hotéis. Serão abordados aspectos relevantes do contrato, como os requisitos do serviço, as responsabilidades das partes envolvidas e as garantias oferecidas. Além disso, a documentação IDL será detalhada, apresentando as especificações técnicas do serviço, como os métodos disponíveis, os parâmetros de entrada e saída e as possíveis respostas do servidor.

3.1 CONTRATO

3.1.1 Objetos

Abaixo a estrutura das entidades presentes no contrato do serviço de hotel, apresentado anteriormente.

Tabela 1 - Entidade de usuário

UserEntity	
id	number
name	string
email	string
password	string

Tabela 2 - Entidade de hotel

HotelEntity	
id	number
name	string
state	string
city	string

Tabela 3 - Entidade de quarto

RoomEntity	
id	number
hotelId	number
category	string
bedType	string
dailyPrice	number

Tabela 4 - Entidade de reserva

ReservationEntity	
Id	number
hotelId	number
roomId	number
userId	number
startDate	Date
endDate	Date
totalPrice	number

3.1.2 Ações

Tabela 5 - Listagem de quartos

LISTAR QUARTOS			
Metodo	GET	Endpoint	/rooms
Headers			
Propriedade		Tipo	
Authorization		string	
Body			
Apenas para metodos POST ou PUT			
Response			
Status		Body	
200 OK		{ "id": "string", "createdAt": "2023-11-06T22:13:03.815Z", "updatedAt": "2023-11-06T22:13:03.815Z", "category": "string", "bedType": "string", "dailyPrice": "string", "hotelId": "string" }	

Tabela 6 - Criar usuário

CRIAR USUÁRIO			
Metodo	POST	Endpoint	/users
Headers			
Propriedade		Tipo	
Content-Type		application/json	
Body			
<pre>{ "name": "string", "email": "string", "password": "string" }</pre>			
Response			
Status		Body	
200 OK		<pre>{ "id": "string", "createdAt": "2023-11-06T22:16:42.289Z", "updatedAt": "2023-11-06T22:16:42.289Z", "name": "string", "email": "string", "reservations": [] }</pre>	
400 BAD REQUEST		"O e-mail informado já possui um cadastro."	

Tabela 7 - Informações de um usuário

DADOS DE UM USUÁRIO			
Metodo	GET	Endpoint	/users/:id (id = indentificação do usuário a ser recuperado)
Headers			
Propriedade		Tipo	
Content-Type		application/json	
Body			
Apenas para metodos POST ou PUT			
Response			
Status		Body	
200 OK		{ "id": "string", "createdAt": "2023-11-06T22:16:42.289Z", "updatedAt": "2023-11-06T22:16:42.289Z", "name": "string", "email": "string", "reservations": [] }	
404 NOT FOUND		"O usuário com essa identificação não foi encontrado."	

Tabela 8 - Autenticação

AUTENTICAÇÃO			
Metodo	POST	Endpoint	/auth
Headers			
Propriedade		Tipo	
Content-Type		application/json	
Body			
{ "email": "string", "password": "string" }			
Response			
Status		Body	
200 OK		{ "token": "string", "expiresAt": "Date", "refreshToken": "string", "refreshExpiresAt": "Date" }	
401 UNAUTHORIZED		"Cadastro não encontrado."	
401 UNAUTHORIZED		"A senha ou o e-mail enviado estão incorretos."	
401 UNAUTHORIZED		"As informações para a autenticação não foram encontradas."	

Tabela 9 - Lista de reservas

LISTA RESERVAS DE UM USUÁRIO			
Metodo	GET	Endpoint	/reservation/user/:id (id = identificação do usuário para pegar as reservas)
Headers			
Propriedade		Tipo	
Authorization		string (Bearer token)	
Content-Type		application/json	
Body			
Apenas para métodos POST ou PUT			
Response			
Status		Body	
200 OK		<pre>[{ "id": "string", "createdAt": "2023-11-06T22:53:20.619Z", "updatedAt": "2023-11-06T22:53:20.619Z", "userId": "string", "roomId": "string", "hotelId": "string", "startDate": "2023-11-06T22:53:20.619Z", "endDate": "2023-11-06T22:53:20.619Z", "totalPrice": 0, "user": {}, "room": {}, "hotel": {} }]</pre>	
404 NOT FOUND		"O usuário com essa identificação não foi encontrado."	

Tabela 10 - Fazer reserva

FAZER RESERVA			
Metodo	POST	Endpoint	/reservation
Headers			
Propriedade		Tipo	
Authorization		string (Bearer token)	
Content-Type		application/json	
Body			
<pre>{ "userId": "string", "roomId": "string", "hotelId": "string", "startDate": "string", "endDate": "string", "totalPrice": 0 }</pre>			
Response			
Status	Body		
200 OK	<pre>{ "id": "string", "createdAt": "2023-11-06T22:53:20.619Z", "updatedAt": "2023-11-06T22:53:20.619Z", "userId": "string", "roomId": "string", "hotelId": "string", "startDate": "2023-11-06T22:53:20.619Z", "endDate": "2023-11-06T22:53:20.619Z", "totalPrice": 0, "user": {}, "room": {}, "hotel": {} }</pre>		
404 NOT FOUND	"O usuário com essa identificação não foi encontrado."		
404 NOT FOUND	"O hotel com essa identificação não foi encontrado."		
404 NOT FOUND	"O quarto com essa identificação não foi encontrado."		
400 BAD REQUEST	"Já existe reserva para esse dia, tente novamente."		

Tabela 11 - Cancelar reserva

CANCELAR RESERVA			
Metodo	DELETE	Endpoint	/reservation/:id (id = indentificação da reserva a ser cancelada)
Headers			
Propriedade		Tipo	
Authorization		string (Bearer token)	
Content-Type		application/json	
Body			
Apenas para metodos POST ou PUT			
Response			
Status		Body	
200 OK		{}	
404 NOT FOUND		"A reserva com essa identificação não foi encontrado."	

3.2 IDL

É possível acessar o IDL acessando o link disponível no GitHub: <https://github.com/Matheus-jacobb/api.nestjs.programacao-distribuida-ac2-grupo6/blob/main/webapp/idl/idl.json>, que também estará anexado a entrega.

4 NOTAS DE IMPLEMENTAÇÃO

A seguir, serão apresentadas informações sobre como utilizar os recursos disponibilizados por este serviço de hotéis, com foco na utilização de um ambiente Node.js com NestJS.

4.1 PUBLICANDO E RODANDO SERVIÇO

4.1.1 Rodando local

Para conseguir acessar o serviço será necessário seguir os passo descritos abaixo, como detalhes de instalação e execução:

Certifique-se de ter o Node.js 18 instalado em seu sistema. Você pode verificar a versão do Node.js digitando o comando `node -v` no terminal (caso não tenha instalado).

Instale o pacote `@nestjs/cli` globalmente em seu sistema executando o seguinte comando no terminal: **`npm install -g @nestjs/cli`**

Navegue até o diretório do projeto, execute o comando `npm run start` para iniciar o serviço NestJS.

Agora, o serviço NestJS estará em execução e você poderá acessá-lo por meio da URL fornecida no terminal, normalmente sendo `http://localhost:3000`.

Caso você não queira instalar todas as dependências manualmente, no projeto está disponível um arquivo Docker. Com o Docker instalado em seu sistema, siga estes passos:

Abra o terminal e navegue até o diretório do projeto NestJS.

Certifique-se de ter o arquivo `Dockerfile` presente no diretório.

Execute o comando `docker build -t nome-do-container` para criar a imagem Docker.

Após a criação da imagem, execute o comando `docker run -p 3000:3000 nome-do-container` para iniciar o contêiner Docker.

Agora, o serviço NestJS estará em execução e você poderá acessá-lo através da porta 3000 em seu navegador.

Lembre-se de substituir "nome-do-projeto" e "nome-do-container" pelos nomes desejados para o projeto e contêiner Docker, respectivamente.

4.1.2 Publicando serviço

Caso haja o interesse aqui está como fazer o deploy de uma aplicação NestJS usando as seguintes ferramentas:

Heroku:

- Certifique-se de ter uma conta no Heroku (<https://www.heroku.com/>) e o Heroku CLI instalado em seu sistema.
- Navegue até o diretório do projeto NestJS no terminal.
- Execute o comando `heroku login` para fazer login na sua conta do Heroku.
- Execute o comando `heroku create` para criar um novo aplicativo no Heroku.
- Em seguida, execute o comando `git push heroku main` para fazer o deploy do seu projeto para o Heroku.

- Após o processo de deploy ser concluído, você receberá a URL do seu aplicativo no Heroku.

Docker:

- Certifique-se de ter uma conta em um provedor de nuvem que suporte a execução de contêineres Docker, como o Amazon Web Services (AWS), Google Cloud Platform (GCP) ou Microsoft Azure.
- Instale o Docker em seu sistema local, caso ainda não tenha feito isso. Você pode seguir as instruções específicas do site oficial do Docker para a sua plataforma.
- Navegue até o diretório do projeto da sua aplicação no terminal.
- Certifique-se de ter um arquivo Dockerfile presente no diretório. O Dockerfile é responsável por definir as instruções para construir a imagem Docker da sua aplicação.
- Execute o comando `docker build -t nome-da-imagem .` para construir a imagem Docker da sua aplicação. Substitua "nome-da-imagem" pelo nome desejado para a imagem.
- Após a construção da imagem, faça login na sua conta do provedor de nuvem usando as credenciais apropriadas. Isso pode variar dependendo do provedor de nuvem escolhido.
- Execute o comando `docker tag nome-da-imagem nome-do-repositorio` para adicionar uma tag à imagem Docker, associando-a ao repositório na nuvem. Substitua "nome-do-repositorio" pelo nome do repositório fornecido pelo provedor de nuvem.
- Execute o comando `docker push nome-do-repositorio` para fazer o upload da imagem Docker para o repositório na nuvem.
- Após o upload ser concluído, você pode implantar a imagem Docker em um serviço de contêiner gerenciado pelo provedor de nuvem, como Amazon ECS, Google Kubernetes Engine (GKE) ou Azure Container Instances (ACI). Consulte a documentação específica do provedor de nuvem para obter instruções detalhadas sobre como implantar contêineres Docker na nuvem.

4.2 COMO CONSUMIR SERVIÇO

Para consumir o serviço, basta acessá-lo após a execução, batendo nos endpoints disponíveis em `http://localhost:3000`. Para cada endpoint, verifique os headers necessários conforme descrito na documentação do projeto. Todas as requisições usam o protocolo HTTPS.

Caso o serviço seja publicado na nuvem, a metodologia é a mesma, mas você deve apontar para o endpoint disponibilizado pelo provedor de nuvem utilizado. Certifique-se de seguir as instruções específicas do provedor de nuvem para obter o endpoint correto e quaisquer outras informações necessárias para consumir o serviço na nuvem.

4.3 COMO IDENTIFICAR USUÁRIO

Cada usuário criado na API é associado a um código UUID único no formato v4, no formato `3cfb0d8e-8dfc-4313-9138-1a5dcebae313`. Esse código UUID é o identificador principal para realizar ações como reservas e outras interações com o usuário.

Caso você esteja utilizando um serviço intermediário para interagir com essa API, é importante salvar esse ID para poder utilizar a maioria dos eventos e funcionalidades relacionadas ao usuário. O ID UUID fornecido pela API será necessário para realizar operações específicas e garantir a integridade dos dados do usuário. Ao utilizar a API, é importante armazenar e utilizar o ID do usuário adequadamente para obter o máximo de funcionalidade da API. Para gerenciar as informações de outros usuários, você pode criar um usuário admin e fazer as requisições necessárias em nome desse usuário.

Para essa implementação, os tokens de admin não expiram. Para fazer login como admin e utilizar a rota de autenticação descrita anteriormente, utilize o email "**admin@email.com**" e a senha "**123456**".

4.4 TESTANDO COM O FRONT

Abaixo demonstramos os teste realizados, implementando o front para consumir a API documentada, o código do front estará anexado a esse documento, para rodar assim como nestJs é necessário ter node 18 e tem angular instalado em sua máquina, para mais informações siga o seguinte tutorial: <https://www.devmedia.com.br/angular-cli-como-criar-e-executar-um-projeto-angular/38246>.

Figura 2 - Tela principal

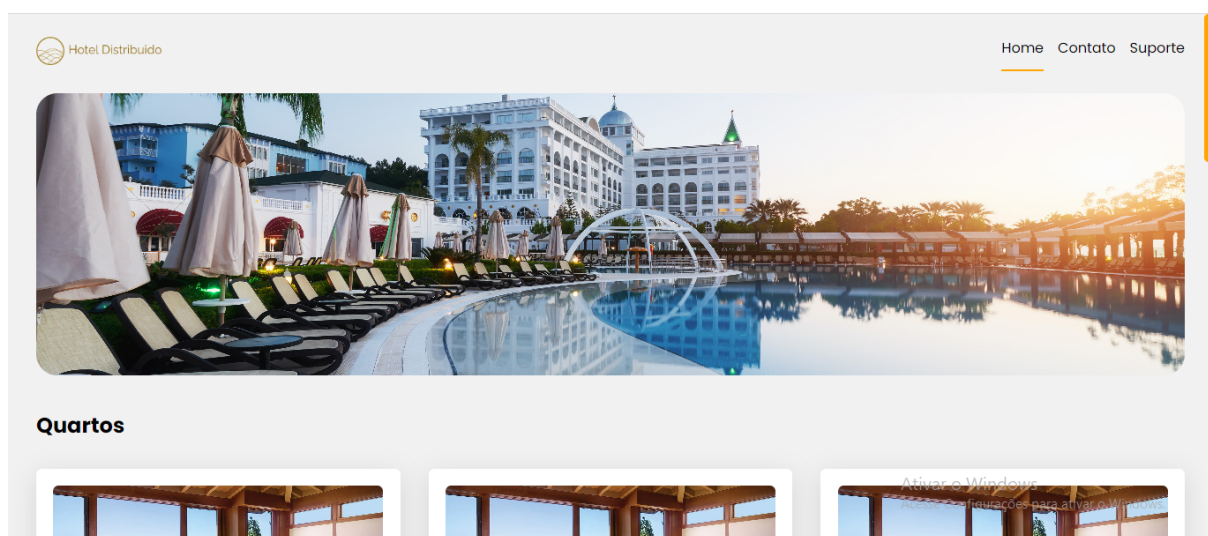
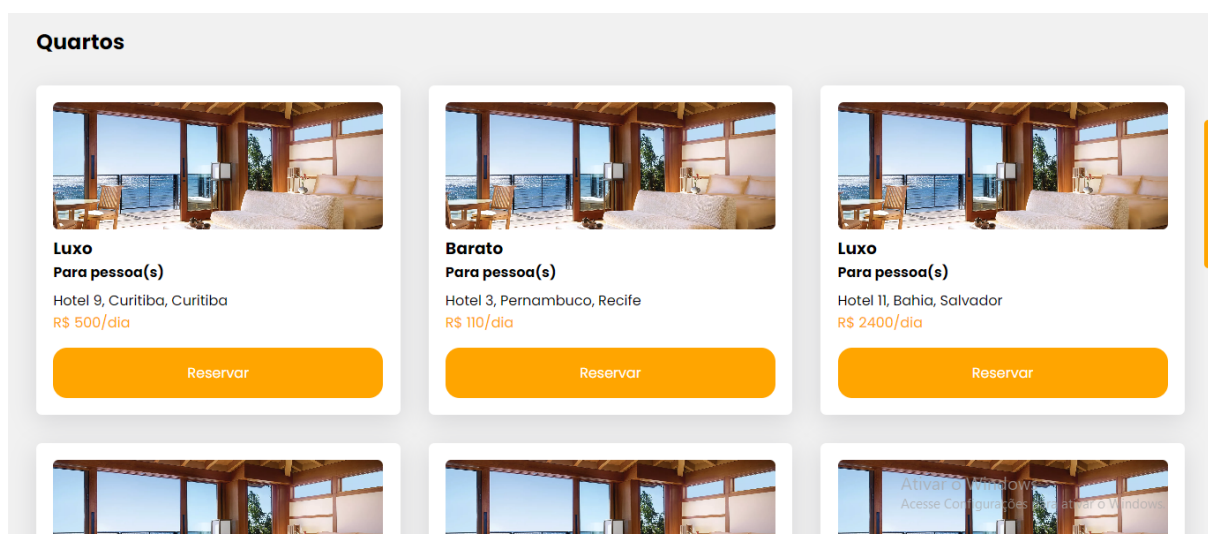
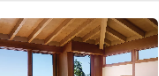


Figura 3 - Lista de hotéis



Quartos



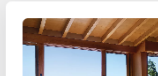
Luxo

Para pessoa(s)

Hotel 9, Curitiba, Curitiba

R\$ 500/dia

Reservar




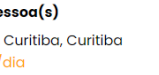
Barato


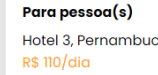
Para pessoa(s)

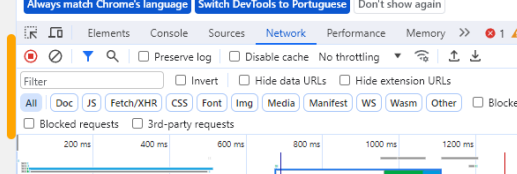
Hotel 3, Pernambuco, Recife


R\$ 110/dia

Reservar

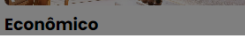





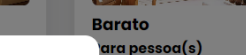


Econômico
Para pessoa(s)
Hotel 5, Rio de Janeiro, Rio de Janeiro
R\$ 90/dia

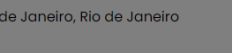
Reservar



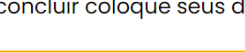
Econômico
Para pessoa(s)
Hotel 10, Curitiba, Curitiba
R\$ 1200/dia




Barato
Para pessoa(s)
Hotel 8, São Paulo, São Paulo
R\$ 120/dia



Luxo
Para pessoa(s)
Hotel 12, Bahia, Salvador
R\$ 1000/dia



Econômico
Para pessoa(s)
Hotel 10, Curitiba, Curitiba
R\$ 1200/dia



Barato
Para pessoa(s)
Hotel 8, São Paulo, São Paulo
R\$ 120/dia

Para concluir coloque seus dados!

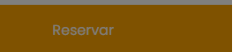
Email

Senha

Data início


Data fim

Cancelar Confirmar




Econômico
Para pessoa(s)
Hotel 5, Rio de Janeiro, Rio de Janeiro
R\$ 90/dia


Reservar



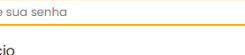
Econômico
Para pessoa(s)
Hotel 10, Curitiba, Curitiba
R\$ 1200/dia




Barato
Para pessoa(s)
Hotel 8, São Paulo, São Paulo
R\$ 120/dia



Luxo
Para pessoa(s)
Hotel 12, Bahia, Salvador
R\$ 1000/dia



Econômico
Para pessoa(s)
Hotel 10, Curitiba, Curitiba
R\$ 1200/dia



Barato
Para pessoa(s)
Hotel 8, São Paulo, São Paulo
R\$ 120/dia

Figura 6 - Validando reserva

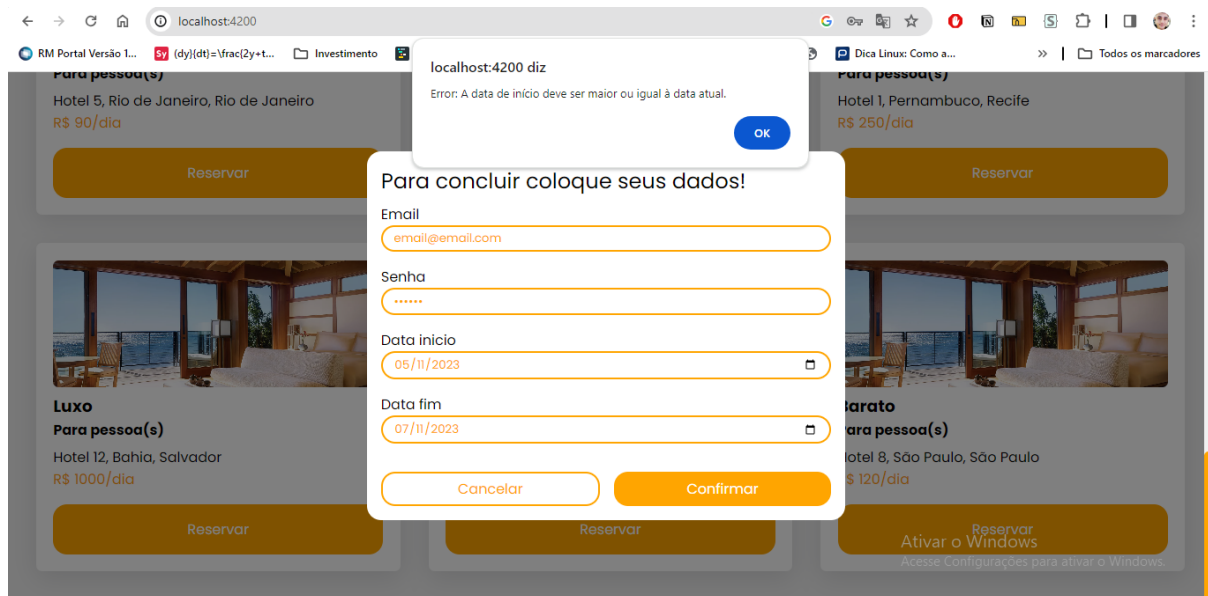


Figura 7 - Confirmar reserva

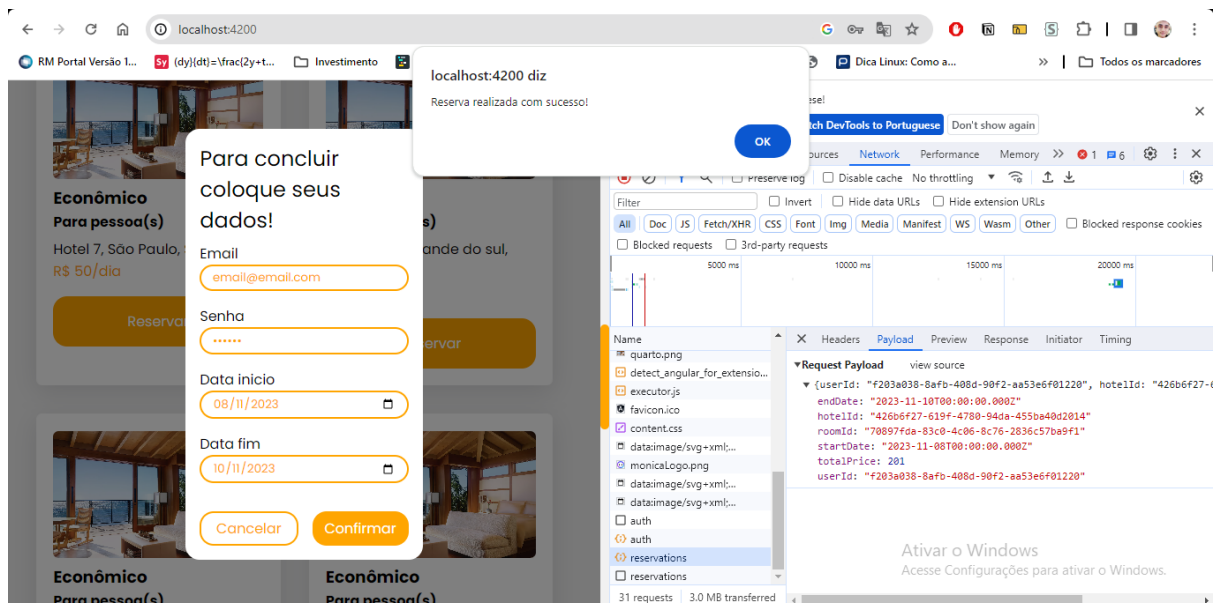
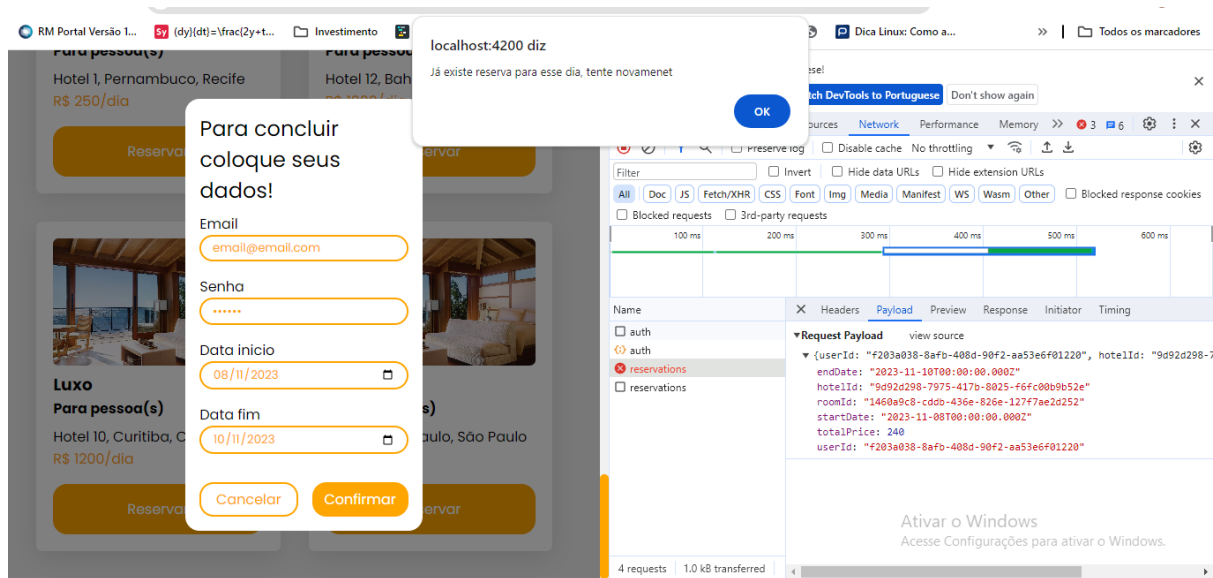


Figura 8 - Conflito de data



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.