



**CENTRO UNIVERSITÁRIO FACENS
ENGENHARIA DA COMPUTAÇÃO**

ANTÔNIO LUÍS CANNO DE ARAÚJO - 190929 - CP406TIN3
EDUARDO DE OLIVEIRA ALMEIDA - 190963 - CP406TIN3
JHONATAN JOSÉ GOMES DO AMARAL - 190932 - CP406TIN2
LUCAS DE ABREU FURTADO GARCIA - 190581 - CP406TIN3
DANIEL DE LIMA FERREIRA - 200010 - CP406TIN1
ENRICO VENTURINI COSTA - 210901 - CP406TIN1

Programação Distribuída

**SOROCABA
2023**

Sumário

1	Introdução	3
2	Descrição do projeto	3
2.1	Objeto do hotel	3
2.2	Objeto do endereço	5
2.3	Objeto do aluguel de quarto	6
2.4	Comunicação entre os objetos	7
3	Desenvolvimento	7
3.1	Contrato	8
3.1.1	Hotel	8
3.1.2	Checkin e Checkout	9
3.2	IDL	9
3.2.1	Busca unitária de hotéis	10
3.2.2	Listagem de hotéis	11
3.2.3	Adição de hotéis	12
3.2.4	Atualização de hotéis	12
3.2.5	Remoção de hotéis	13
3.2.6	Checkin	14
3.2.7	Checkout	15
4	Notas de implantação e consumo	15

Lista de Figuras

1	Interface do hotel.	4
2	Interface do endereço.	5
3	Interface do aluguel de quarto.	6
4	Comunicação entre os objetos principais da aplicação.	7
5	Retorno da busca de um hotel.	10
6	Retorno da busca de um hotel.	11
7	Retorno da busca de um hotel.	12
8	Retorno da busca de um hotel.	13
9	Payload de envio para o checkin em um hotel.	14
10	Retorno do checkin em um hotel.	14
11	Payload de envio para o checkin em um hotel.	15

1 Introdução

Webservice e Webapp são dois conceitos fundamentais no desenvolvimento de aplicações web. Um webservice é uma solução que permite a comunicação entre diferentes sistemas através da internet, utilizando protocolos padrão como HTTP, XML e JSON (W3C, 2004). Ele fornece uma interface que permite que os clientes acessem e interajam com os serviços disponibilizados pelo servidor. Já um webapp, ou aplicativo web, é uma aplicação que roda em um navegador web e pode ser acessada pelos usuários através da internet. Ele utiliza tecnologias como HTML, CSS e JavaScript para criar interfaces interativas e dinâmicas. Tanto os webservices quanto os webapps desempenham um papel crucial na construção de soluções web modernas, permitindo a troca de informações e a criação de experiências interativas na web (PRESSMAN, 2010).

2 Descrição do projeto

O grupo decidiu desenvolver uma API que irá permitir a listagem de hotéis com base em diferentes filtros, ou seja, o modelo proposto: WebApp Hotéis. Os quartos disponíveis podem ser de três tipos: com uma cama simples para uma pessoa, com uma cama dupla para duas pessoas ou com duas camas simples também para duas pessoas. Além disso, os hotéis serão classificados em três categorias: baratos, econômicos e luxuosos. Essa API proporcionará aos usuários a possibilidade de encontrar opções de hospedagem que atendam às suas necessidades e preferências, oferecendo uma experiência personalizada e completa.

2.1 Objeto do hotel

A interface presente na figura 2, define o tipo "Hotel" em TypeScript.

Figura 1: Interface do hotel.

```
export type Hotel = {  
  id: string;  
  name: string;  
  address: Address;  
  starsQuantity: number;  
  description: string;  
  hasBreakfast: boolean;  
  hasLunch: boolean;  
  hasDinner: boolean;  
  category: HotelCategory;  
  parkingLotsQuantity?: number;  
  roomCategories: RoomCategory[];  
  rentedRooms: RoomRent[];  
};
```

Fonte: criado pelo autor

A interface descreve as propriedades que um objeto do tipo Hotel pode ter. Aqui está uma explicação de cada uma das propriedades:

- id: Uma string que representa o identificador único do hotel;
- name: Uma string que representa o nome do hotel;
- address: Um objeto do tipo "Address" que contém informações sobre o endereço do hotel;
- starsQuantity: Um número que representa a quantidade de estrelas do hotel, indicando seu nível de qualidade;
- description: Uma string que contém uma descrição do hotel;
- hasBreakfast: Um valor booleano que indica se o hotel oferece café da manhã;
- hasLunch: Um valor booleano que indica se o hotel oferece almoço;
- hasDinner: Um valor booleano que indica se o hotel oferece jantar;
- category: Um valor do tipo "HotelCategory" que representa a categoria do hotel;
- parkingLotsQuantity: Um número opcional que representa a quantidade de vagas de estacionamento disponíveis no hotel;
- roomCategories: Um array de objetos do tipo "RoomCategory" que descreve as categorias de quartos disponíveis no hotel;

- rentedRooms: Um array de objetos do tipo "RoomRent" que demonstra os quartos que já foram alugados no hotel, sejam ativos ou inativos;

Essa interface define a estrutura de um objeto do tipo Hotel, indicando quais propriedades são necessárias e quais são opcionais. Isso ajuda a garantir que os dados sejam consistentes e facilita a manipulação desses objetos em um programa TypeScript.

2.2 Objeto do endereço

A interface presente na figura 2, define o tipo "Endereço" em TypeScript, que é um objeto que está em toda instância de Hotel.

Figura 2: Interface do endereço.

```
export type Hotel = {  
  id: string;  
  name: string;  
  address: Address;  
  starsQuantity: number;  
  description: string;  
  hasBreakfast: boolean;  
  hasLunch: boolean;  
  hasDinner: boolean;  
  category: HotelCategory;  
  parkingLotsQuantity?: number;  
  roomCategories: RoomCategory[];  
  rentedRooms: RoomRent[];  
};
```

Fonte: criado pelo autor

Segue a explicação de cada uma das propriedades:

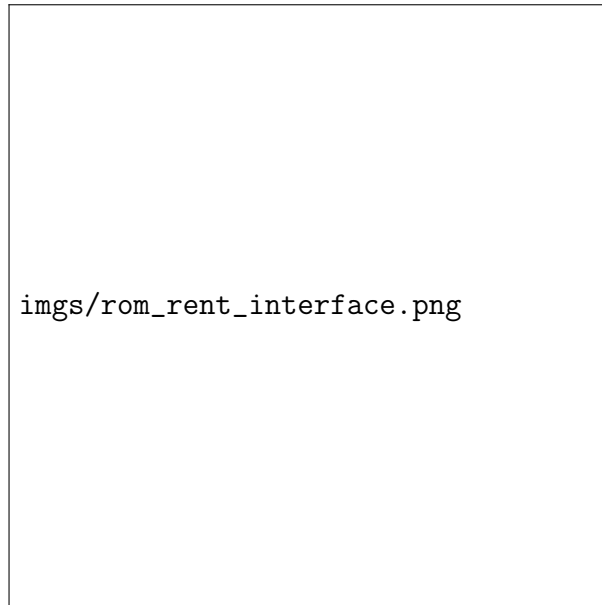
- city: Uma string que representa o nome da cidade do endereço;
- street: Uma string que representa o nome da rua do endereço;
- zipNumber: Uma string que representa o número do CEP do endereço;
- number: Um número opcional que representa o número da residência no endereço;
- neighborhood: Uma string que representa o bairro do endereço;
- country: Uma string que representa o país do endereço;
- state: Uma string que representa o estado do endereço;

- complement: Uma string opcional que representa informações adicionais sobre o endereço;

2.3 Objeto do aluguel de quarto

A interface presente na figura 3, define o tipo "Aluguel de quarto" em TypeScript, que é um objeto utilizado dentro da interface Hotel.

Figura 3: Interface do aluguel de quarto.



Fonte: criado pelo autor

Segue a explicação de cada uma das propriedades:

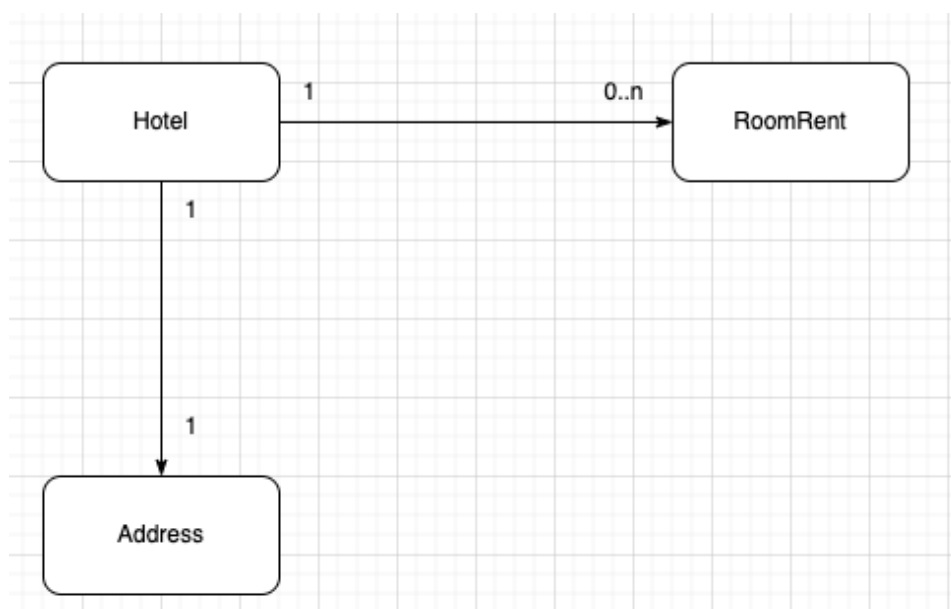
- id: Uma string que representa o identificador único do aluguel do quarto;
- hotelId: Uma string que representa o identificador único do hotel;
- name: Uma string que representa o nome do cliente que alugou o quarto;
- cpf: Uma string que representa o cpf do cliente que alugou o quarto;
- starsQuantity: Um número que representa a quantidade de estrelas do hotel, indicando seu nível de qualidade;
- category: Uma string do tipo "RoomCategory" que descreve as categorias de quartos disponíveis no hotel, e especificamente a categoria do quarto alugado;
- checkinTime: Uma data que representa o momento em que o cliente que alugou o quarto;

- checkoutTime: Uma data que representa o momento em que o cliente que deixou o quarto;

2.4 Comunicação entre os objetos

De acordo com as considerações previamente abordadas, constata-se que o objeto hotel é intrinsecamente vinculado a um endereço, e, reciprocamente, o endereço está diretamente associado a um hotel. No que tange ao aluguel de quartos, é possível observar que um determinado hotel pode apresentar uma variação que pode oscilar entre a ausência de aluguéis e a potencialidade de ter um número infinito deles. Essa relação entre o hotel, o endereço e o aluguel de quartos evidencia a interdependência existente entre esses elementos, ressaltando assim a relevância de se considerar tais aspectos para uma compreensão adequada do objeto de estudo em questão, valide a partir da figura 4.

Figura 4: Comunicação entre os objetos principais da aplicação.



Fonte: criado pelo autor

3 Desenvolvimento

É essencial o entendimento dos contratos e da IDL na criação de uma API de listagem de hotéis, utilizando a ferramenta Swagger. Os contratos estabelecerão as regras, responsabilidades e interfaces necessárias para garantir a comunicação efetiva entre os diferentes componentes envolvidos. Através da IDL, a estrutura será definida e os detalhes da interface da API, especificando as operações disponíveis, os parâmetros requeridos, os tipos de dados envolvidos e os resultados esperados. É de grande importância ressaltar que utilizou-se o Swagger para documentar e visualizar de forma clara e intuitiva a API, faci-

litando a integração e o entendimento por parte dos desenvolvedores. Assim, a utilização dessas tecnologias de contrato e IDL, em conjunto com o Swagger, será fundamental para garantir a consistência e a interoperabilidade da API de listagem de hotéis.

3.1 Contrato

Pode-se dividir os contratos em duas seções: crud do hotel, e checkin e checkout de um cliente.

3.1.1 Hotel

Para adicionar um novo hotel na aplicação, é preciso seguir algumas regras:

- O nome do estabelecimento deve possuir mais de três letras e o endereço deve ser fornecido corretamente, sendo opcional apenas o envio do número e complemento;
- A quantidade de estrelas deve ser um número entre 0 e 5;
- Embora a quantidade de vagas no estacionamento não seja um requisito obrigatório, quando fornecida, deve ser maior ou igual a zero;
- As categorias dos quartos devem ser apresentadas em uma lista contendo no máximo três campos, podendo ser apenas "1_single_bed", "2_single_bed" ou "1_couple_bed";
- A categoria do hotel deve ser selecionada entre as opções "barato", "econômico" ou "luxuoso".

Já para atualizar o hotel, as regras são as mesmas, tendo apenas uma a mais:

- É preciso passar o identificador do hotel que será editado, caso contrário não será possível atualizar nenhuma instância.

Para obter as informações de apenas um hotel, é preciso passar apenas o identificador do hotel já encontrado, contudo para listar mais de uma entidade, é possível fazer alguns tipos de filtros:

- O nome do estabelecimento pode ser passado, não é necessário passar o nome completo, apenas uma parte é o suficiente. Caso não seja passado, não será aplicado nenhuma lógica no retorno de hotéis;
- As categorias dos quartos devem ser apresentadas em uma lista contendo no máximo três campos, podendo ser apenas "1_single_bed", "2_single_bed" ou "1_couple_bed". Caso não seja passado, não será aplicado nenhuma lógica no retorno de hotéis;

- A categoria do hotel deve ser selecionada entre as opções "barato", "econômico" ou "luxuoso". Caso não seja passado, não será aplicado nenhuma lógica no retorno de hotéis.

Por fim, para deletar um hotel, é necessário passar apenas o identificador para a requisição, mas caso o identificador não seja válido ele retornará a mensagem da mesma maneira.

3.1.2 Checkin e Checkout

Para realizar o checkin em um quarto de hotel, ou seja alugar um dos quartos, é preciso:

- O identificador do hotel em que se está fazendo o checkin, além de ser obrigatório, ele precisa existir entre os hotéis mapeados no software, gerando um erro ao passar um que não exista na lista;
- O cpf do cliente que está fazendo o checkin, obrigatoriamente;
- O nome do cliente que está fazendo o checkin, obrigatoriamente;
- O momento em que o cliente fez o checkin no hotel, no formato ISOString, obrigatoriamente, e não deve ser no futuro, ou seja, tem que ser menor que o horário da requisição à API;
- A categoria do quarto do quarto alugado deve se apresentado contendo um dos três campos: "1_single_bed", "2_single_bed" ou "1_couple_bed". É importante ressaltar que caso o hotel não tenha essa categoria disponível, será retornado um erro.

Já para o checkout, que seria sair do hotel alugado e devolver as chaves, é preciso:

- O identificador do hotel em que se está fazendo o checkin, além de ser obrigatório, ele precisa existir entre os hotéis mapeados no software, gerando um erro ao passar um que não exista na lista;
- O momento em que o cliente fez o checkout no hotel, no formato ISOString, obrigatoriamente, e não deve ser no futuro, ou seja, tem que ser menor que o horário da requisição à API.

3.2 IDL

Nesse capítulo, será abordado a IDL para cada requisição na API de hotéis, sendo dividido em: busca unitária de hotéis, listagem de hotéis, adição de hotéis, atualização de hotéis, remoção de hotéis, checkin e checkout.

É importante ressaltar, que os pontos descritos estão disponíveis, também, no swagger da aplicação.

3.2.1 Busca unitária de hotéis

Para realizar a requisição, é preciso chamar a requisição com o método GET, no endpoint: /hotels/id. É importante ressaltar que id deve ser substituído pelo identificador do hotel.

O retorno da API segue o formato conforme a figura 5. Esse retorno segue o seguinte padrão: todo conteúdo que o cliente realmente buscou está dentro do atributo data, já a mensagem que dirá o status da requisição está no atributo message, caso seja uma mensagem positiva. Em cenários de erro, um objeto error é criado, e dentro dele terá a mensagem e, caso esteja acordado, um código que facilitará o dê para entre o cliente e o servidor.

Figura 5: Retorno da busca de um hotel.

```
{
  "data": {
    "id": "string",
    "name": "string",
    "address": {
      "city": "string",
      "street": "string",
      "zipNumber": "string",
      "number": 0,
      "neighborhood": "string",
      "country": "string",
      "state": "string",
      "complement": "string"
    },
    "starsQuantity": 0,
    "description": "string",
    "hasBreakfast": true,
    "hasLunch": true,
    "hasDinner": true,
    "category": "string",
    "parkingLotsQuantity": 0,
    "roomCategories": [
      "string"
    ]
  },
  "message": "string",
  "error": {
    "code": "string",
    "message": "string"
  }
}
```

Fonte: criado pelo autor

3.2.2 Listagem de hotéis

Para realizar a requisição, é preciso chamar a requisição com o método GET, no endpoint: /hotels. Além disso, é possível passar três tipos de query params: name, que filtra pelo nome do hotel; category, que filtra pela categoria do hotel; roomCategories, que filtra pela tipo de quarto do hotel;

O retorno da API segue o formato conforme a figura 6. Esse retorno segue o seguinte padrão: todo conteúdo que o cliente realmente buscou está dentro do atributo data, e é uma lista de hotéis, já a mensagem que dirá o status da requisição está no atributo message, caso seja uma mensagem positiva. Em cenários de erro, um objeto error é criado, e dentro dele terá a mensagem e, caso esteja acordado, um código que facilitará o dê para entre o cliente e o servidor.

Figura 6: Retorno da busca de um hotel.

```
{
  "data": [
    {
      "id": "string",
      "name": "string",
      "address": {
        "city": "string",
        "street": "string",
        "zipNumber": "string",
        "number": 0,
        "neighborhood": "string",
        "country": "string",
        "state": "string",
        "complement": "string"
      },
      "starsQuantity": 0,
      "description": "string",
      "hasBreakfast": true,
      "hasLunch": true,
      "hasDinner": true,
      "category": "string",
      "parkingLotsQuantity": 0,
      "roomCategories": [
        "string"
      ]
    }
  ],
  "message": "string",
  "error": {
    "code": "string",
    "message": "string"
  }
}
```

Fonte: criado pelo autor

3.2.3 Adição de hotéis

Para realizar a requisição, é preciso chamar a requisição com o método POST, no endpoint: /hotels. Além disso, é preciso enviar no body da requisição o json conforme a figura 7.

Figura 7: Retorno da busca de um hotel.

```
{
  "name": "string",
  "address": {
    "city": "string",
    "street": "string",
    "zipNumber": "string",
    "number": 0,
    "neighborhood": "string",
    "country": "string",
    "state": "string",
    "complement": "string"
  },
  "starsQuantity": 0,
  "description": "string",
  "hasBreakfast": true,
  "hasLunch": true,
  "hasDinner": true,
  "category": "string",
  "parkingLotsQuantity": 0,
  "roomCategories": [
    "string"
  ]
}
```

Fonte: criado pelo autor

O retorno da API segue o mesmo formato conforme a figura 5, além dos padrões propostos na requisição de GET.

3.2.4 Atualização de hotéis

Para realizar a requisição, é preciso chamar a requisição com o método PATCH, no endpoint: /hotels/id. É importante ressaltar que id deve ser substituído pelo identificador do hotel. Além disso, é preciso enviar no body da requisição o json conforme a figura 8.

Figura 8: Retorno da busca de um hotel.

```
{
  "id": "string",
  "name": "string",
  "address": {
    "city": "string",
    "street": "string",
    "zipNumber": "string",
    "number": 0,
    "neighborhood": "string",
    "country": "string",
    "state": "string",
    "complement": "string"
  },
  "starsQuantity": 0,
  "description": "string",
  "hasBreakfast": true,
  "hasLunch": true,
  "hasDinner": true,
  "category": "string",
  "parkingLotsQuantity": 0,
  "roomCategories": [
    "string"
  ]
}
```

Fonte: criado pelo autor

O retorno da API segue o mesmo formato conforme a figura 5, além dos padrões propostos na requisição de GET.

3.2.5 Remoção de hotéis

Para realizar a requisição, é preciso chamar a requisição com o método DELETE, no endpoint: /hotels/id. É importante ressaltar que id deve ser substituído pelo identificador do hotel.

O retorno da API segue o mesmo formato conforme a figura 5, além dos padrões propostos na requisição de GET

3.2.6 Checkin

Para realizar a requisição, é preciso chamar a requisição com o método POST, no endpoint: /hotels/checkin. Já como corpo da requisição, deve-se enviar conforme a figura 9.

Figura 9: Payload de envio para o checkin em um hotel.

```
{
  "hotelId": "string",
  "category": "string",
  "cpf": "string",
  "name": "string",
  "checkinTime": "2023-11-07T21:59:23.419Z"
}
```

Fonte: criado pelo autor

O retorno da API segue conforme a figura 10, além dos padrões já propostos nos outros endpoints.

Figura 10: Retorno do checkin em um hotel.

```
{
  "data": {
    "id": "string",
    "hotelId": "string",
    "cpf": "string",
    "name": "string",
    "category": "string",
    "checkinTime": "2023-11-07T21:59:23.421Z",
    "checkoutTime": "2023-11-07T21:59:23.421Z"
  },
  "message": "string",
  "error": {
    "code": "string",
    "message": "string"
  }
}
```

Fonte: criado pelo autor

3.2.7 Checkout

Para realizar a requisição, é preciso chamar a requisição com o método POST, no endpoint: `/hotels/checkout`. Já como corpo da requisição, deve-se enviar conforme a figura 11.

Figura 11: Payload de envio para o checkin em um hotel.

```
{  
  "checkoutTime": "2023-11-07T22:00:16.167Z",  
  "id": "string",  
  "hotelId": "string"  
}
```

Fonte: criado pelo autor

O retorno da API segue conforme a figura 10, além dos padrões já propostos nos outros endpoints.

4 Notas de implantação e consumo

De acordo com as normas da ABNT (Associação Brasileira de Normas Técnicas), não há diretrizes específicas para a redação de um conteúdo explicativo sobre o consumo de uma aplicação. No entanto, posso fornecer uma explicação sobre os passos necessários para consumir a aplicação mencionada no repositório fornecido, utilizando uma estrutura de parágrafos.

Para consumir a aplicação, primeiramente é necessário clonar o repositório no seguinte domínio: <https://github.com/DuduAlmeida/facens.sistema.operacionais.embarcados>. Essa clonagem pode ser realizada utilizando o Git e o comando `git clone https://github.com/DuduAlmeida/facens.sistema.operacionais.embarcados`. Ao clonar o repositório, uma cópia local do projeto será obtida em seu computador.

Uma vez que o repositório foi clonado, é preciso navegar até a pasta do projeto no terminal. Para isso, abra um terminal ou prompt de comando e utilize o comando `cd af_jardini/webapp/impl/` para acessar a pasta correta onde o projeto foi clonado. Essa etapa é importante para garantir que você esteja no diretório adequado para executar os comandos subsequentes.

Após navegar até a pasta correta, é necessário executar o comando `yarn && yarn dev` no terminal. Esse comando iniciará o servidor local e permitirá que a aplicação seja executada no ambiente de desenvolvimento. Certifique-se de ter o gerenciador de pacotes Yarn instalado em seu sistema para poder executar esse comando corretamente.

Uma vez que o servidor local esteja em execução, será possível acessar a aplicação através do domínio `http://localhost:4321/api` em seu navegador. Ao acessar esse domínio, você poderá verificar as idls (Interface Description Languages) da aplicação. Essas idls fornecem informações sobre a interface da aplicação, como os métodos disponíveis, os parâmetros esperados e as respostas esperadas.

Contudo, para fazer as requisições de utilização, basta chamar o domínio `http://localhost:4321` como base, e adicionar os endpoints mapeados na IDL.

Para facilitar os testes, o grupo preparou uma collection no postman, que ao instalar o aplicativo é possível clicar em importar e colocar o json da collection feita pelo grupo. A collection encontra-se ao seguir o caminho: "af__jardini/webapp/idl/Hotel Api.postman_collection.json"

Referências

PRESSMAN, Roger S. **Engenharia de software**. [S.l.]: McGraw Hill, 2010.

W3C. **Web Services Architecture**. 2004. Disponível em:

<https://www.w3.org/TR/ws-arch/>.