

# **Documentação do Sistema de Busca de Acomodações**

**Dev:** Diego Raylan Silva Araujo

Visão Geral	3
Tecnologias Utilizadas:	3
Arquitetura do Projeto	3
1. API REST (Back-end):	3
2. REACT (Front-end):	4
Parte 1: API REST com Python (FastAPI)	4
Endpoints Disponíveis	4
Parte 2: Front-end em React	<b>6</b>
Dependências e Estrutura de Arquivos	6
Como Rodar o Projeto	7
Pré-requisitos	7
Execução da API (Back-end)	7
Execução do Front-end	7

## Visão Geral

Este projeto tem como objetivo facilitar a busca de acomodações disponíveis para locação de temporada. A solução é composta por dois componentes principais:

1. **API REST em Python** – Responsável por fornecer dados fictícios de acomodações e expor endpoints para consulta e filtragem.
2. **Front-end em React** – Interface web que consome os dados da API, exibe as acomodações, permite filtrar por cidade e favoritar itens (armazenando os favoritos no *localStorage*).

## Tecnologias Utilizadas:

- **FastAPI** - Framework para desenvolvimento da API.
- **Pydantic** - Validação de dados.
- **JSON** - Base de dados para as acomodações.
- **CORS** - Configurado para permitir chamadas do frontend.

## Arquitetura do Projeto

A solução é dividida em duas partes independentes, facilitando a manutenção e a escalabilidade:

1. API REST (Back-end):
  - **Tecnologia:** Python com FastAPI.
  - **Endpoints:**
    - **GET /acomodacoes** – Retorna uma lista de acomodações.
    - **GET /acomodacoes/{id}** – Retorna detalhes de uma acomodação específica.
    - **GET /acomodacoes?cidade=nome\_da\_cidade** – Retorna acomodações filtradas pela cidade.
  - **Dados:** Informações fictícias de acomodações armazenadas diretamente em um dicionário.

## 2. REACT (Front-end):

- **Tecnologia:** React.
- **Funcionalidades:**
  - Consumo da API para exibir acomodações (imagem, nome, preço e localização).
  - Campo de busca para filtrar acomodações por cidade.
  - Marcação de acomodações como favoritas, com armazenamento no *localStorage*.
  - (Opcional) Página de detalhes para cada acomodação e design responsivo.

## 3. Docker:

Para os componente possui seu próprio *Dockerfile* se necessário, facilitando a criação de imagens e a execução em contêineres.

## Parte 1: API REST com Python (FastAPI)

### Endpoints Disponíveis

#### 1. GET /acomodacoes

Retorna uma lista com todas as acomodações cadastradas.

Exemplo:

```
[
  {
    "id": 1,
    "nome": "Pousada Mar Azul",
    "cidade": "floripa",
    "preco": 250,
    "imagem": "/static/images/pousada-mar-azul.jpg",
    "descricao": "Uma pousada aconchegante perto da praia."
  },
  {
    "id": 2,
    "nome": "Flat Vista Mar",
    "cidade": "floripa",
    "preco": 320,
    "imagem": "/static/images/flat-vista-mar.jpg",
    "descricao": "Apartamento com vista panorâmica do mar."
  },
  {
    "id": 3,
    "nome": "Cabana Serra Verde",
    "cidade": "gramado",
    "preco": 400,
    "imagem": "/static/images/cabana-serra-verde.jpg",
    "descricao": "Cabana rústica no meio da serra com lareira."
  }
]
```

## 2. GET /acomodacoes/{id}

Retorna os detalhes de uma acomodação específica (identificada pelo parâmetro `id`).

Exemplo:

```
[
  {
    "id": 3,
    "nome": "Cabana Serra Verde",
    "cidade": "gramado",
    "preco": 400,
    "imagem": "/static/images/cabana-serra-verde.jpg",
    "descricao": "Cabana rústica no meio da serra com lareira."
  }
]
```

## 3. GET /acomodacoes?cidade=nome\_da\_cidade

Permite filtrar as acomodações por cidade.

Exemplo de uso: **GET /acomodacoes?cidade=Florianópolis**

cidade

string

(query)

Execute

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/acomodacoes?cidade=floripa' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/acomodacoes?cidade=floripa
```

Server response

Code	Details
200	<div><div>Response body</div><div><pre>[   {     "id": 1,     "nome": "Pousada Mar Azul",     "cidade": "floripa",     "preco": 250,     "imagem": "/static/images/pousada-mar-azul.jpg",     "descricao": "Uma pousada aconchegante perto da praia."   },   {     "id": 2,     "nome": "Flat Vista Mar",     "cidade": "floripa",     "preco": 320,     "imagem": "/static/images/flat-vista-mar.jpg",     "descricao": "Apartamento com vista panorâmica do mar."   } ]</pre></div></div>

## Parte 2: Front-end em React

### Funcionalidades do Front-end

- **Exibição das acomodações:**  
Listagem com imagem, nome, preço, cidade e descrição.
- **Filtro por cidade:**  
Campo de busca que permite filtrar as acomodações com base na cidade.
- **Favoritar acomodações:**  
Botão para favoritar/desfavoritar, com armazenamento no *localStorage*.
- **(Opcional) Página de detalhes:**  
Exibição de informações mais completas sobre a acomodação (pode ser implementado com React Router).

### Dependências e Estrutura de Arquivos

#### Estrutura da API:

```
Api-locacao/  
├── backend/  
│   ├── main.py           # Arquivo principal da API (FastAPI)  
│   ├── data.json         # Base de dados das acomodações  
│   ├── static/           # Pasta para arquivos estáticos  
│   ├── __pycache__/      # Cache do Python  
│   └── venv/             # Ambiente virtual (bibliotecas do projeto)  
├── frontend/  
│   ├── node_modules/     # Dependências do projeto gerenciadas pelo npm/yarn  
│   ├── public/           # Arquivos públicos, incluindo imagens e ícones  
│   ├── src/              # Código-fonte do frontend (React)  
│   │   ├── assets/       # Recursos visuais  
│   │   ├── AccommodationDetails.jsx # Página de detalhes da acomodação  
│   │   ├── App.css       # Estilos globais  
│   │   ├── App.jsx       # Componente principal  
│   │   ├── Home.jsx      # Página inicial  
│   │   ├── index.css     # Estilos globais adicionais  
│   │   ├── main.jsx      # Ponto de entrada do React  
│   │   └── styles.css    # Outros estilos  
│   ├── package.json      # Configuração do projeto e dependências  
│   └── vite.config.js     # Configuração do Vite  
└── star-all              # Arquivo extra dentro da pasta Api-locacao
```

## Dockerfile da API

```
# Dockerfile para a API de Acomodações

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY main.py .

EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

## Como Rodar o Projeto

### Pré-requisitos

- **Docker:** Certifique-se de que o Docker esteja instalado na sua máquina.

**No Windows:** basta clicar start-all dentro da Pasta “Api-locacao/start-all”

### Execução da API (Back-end)

Abra o terminal e execute:

1. **Navegue até o diretório da API:** `cd Api-locacao\backend`
2. **Construa a imagem Docker:** `docker build -t api-acomodacoes.`
3. **Inicie um container mapeando a porta 8000:** `docker run -d -p 8000:8000 api-acomodacoes`
4. **API estará disponível em:** <http://localhost:8000>

### Execução do API (Front-end)

1. **Navegue até o diretório do front-end:** `cd Api-locacao\backend`
2. **Construa a imagem Docker:** `docker build -t frontend-acomodacoes.`
3. **Inicie o container:** `docker run -d -p 3000:3000 frontend-acomodacoes`

4. **Inicie o container do Front-end:** `docker run -d -p 3000:3000 frontend-acomodacoes`
5. **abrir o programa:** `npm run dev`

Ele deve abrir no link: <http://localhost:5175/>