

# Relatório - CheckIn

## 1) Rotas existentes — input / output (JSON modelo)

- **POST /usuarios/register**
  - **Input**

```
{ "nome": "string", "email": "string", "CNPJ": "string", "senha": "string" }
```
  - **Output**

```
{ "message": "Usuário criado com sucesso!", "usuario": { "id": 1, "nome": "string", "email": "string" } }
```
- **POST /usuarios/login**
  - **Input**

```
{ "email": "string", "senha": "string" }
```
  - **Output**

```
{ "access_token": "jwt", "user": { "id": X, "nome": "string", "email": "string" } }
```
- **GET /hospede/hospedes**
  - **Input:** query status
  - **Output**

```
[ { "id": 1, "nome": "", "cpf": "", "telefone": "", "email": "", "endereco": "" } ]
```
- **POST /hospede/**
  - **Input**

```
{ "nome": "", "cpf": "", "telefone": "", "email": "", "endereco": "" }
```
  - **Output:** Hóspede criado.
- **PUT /hospede/{hospede\_id}** — atualiza hóspede.
  - **Input**

```
{ "nome": "string", "email": "string", "senha": "string" }
```
  - **Output**

```
{ "message": "Usuário atualizado com sucesso!", "usuario": { "id": X, "nome": "string", "email": "string" } }
```
  -
- **DELETE /hospede/{hospede\_id}**
  - **Output:** { "message": "Hóspede removido com sucesso!" }
- **GET /hospede/{hospede\_id}** — retorna hóspede.

- **GET /quarto/quartos**

- **Input:** query status
- **Output**

```
[ { "id":1, "numero":101, "capacidade":2, "tipo":"Standard", "preco_diaria":120.5,  
"status":"disponível", "recursos":"string" } ]
```

- **POST /quartos/**

- **Input**

```
{ "número":101, "capacidade":2, "tipo":"Standard", "preco_diaria":120.5,  
"status":"disponível", "recursos":"string" }
```

- **Output:** Quarto Cadastrado com sucesso.

- **PUT /quartos/{quarto\_id}**

- **Input**

```
{ "número":101, "capacidade":2, "tipo":"Standard", "preco_diaria":120.5,  
"status":"disponível", "recursos":"string" }
```

- **Output:** Quarto atualizado com sucesso.

- **DELETE /quartos/{quarto\_id}**

- **Output:** { "message": "Quarto removido com sucesso!" }

- **GET /reserva/reservas**

- **Input:** ()

- **Output**

```
[ { "id":1, "hospede_id":1, "quarto_id":1, "data_entrada":"ISO8601",  
"data_saida":"ISO8601", "status":"", "observacoes":"" } ]
```

## 2) Dicionários de dados

### Usuario

```
{ "id": "int", "nome": "string", "email": "string", "senha": "string", "CNPJ": "string" }
```

### Hospede

```
{ "id": "int", "nome": "string", "cpf": "string", "telefone": "string", "email": "string", "endereco":  
"string" }
```

## **Quarto**

```
{ "id": "int", "numero": "int", "capacidade": "int", "tipo": "string", "preco_diaria": "float", "status": "string", "recursos": "string"}
```

## **Reserva**

```
{ "id": "int", "hospede_id": "int", "quarto_id": "int", "data_entrada": "datetime", "data_saida": "datetime"}
```

## **3) Prazo estimado**

- **Correções críticas (typos, auth e não expor a senha):** 15 dias
- **Implementar CRUD de Reservas + Check-in/Check-out + testes:** 15 dias

**Tempo previsto: 15 dias**

## **4) Fluxo de Login / Logout**

- **Registro:** POST /usuarios/register → senha é hashada (bcrypt) e usuário salvo.
- **Login:** POST /usuarios/login → valida credenciais, gera JWT (access\_token) com sub=email e expiração (30 min). Resposta contém o token. Frontend envia Authorization: Bearer <token> nas requisições.
- **Proteção:** rotas usam Depends(oauth2\_scheme) e get\_current\_user que decodifica JWT e retorna usuário; rotas com user = Depends(get\_current\_user) exigem token válido.
- **Logout: não implementado no servidor** (JWT sem blacklist). Logout atual é cliente removendo token localmente; para logout server-side é necessário implementar blacklist/armazenamento dos tokens inválidos ou usar refresh tokens + revogação.

## **5) Equipe**

- Davi Guedes Noberto;
- Estela Áurea da Nóbrega Calixto;
- João Marcos de Azevedo Dantas;
- Raun Allyson de Araújo Felix.

## 7) Link GitHub

- [https://github.com/joao-marcos-azvd/Gestao\\_Hoteis](https://github.com/joao-marcos-azvd/Gestao_Hoteis)