Documentação da API

Visão Geral

Esta API gerencia operações relacionadas a tickets e atendentes, utilizando Node.js, Express, Mongoose, e MongoDB. A aplicação está containerizada com Docker, facilitando o gerenciamento e a escalabilidade.

Arquitetura

- **Docker**: Hospeda contêineres da aplicação e do MongoDB.
- **MongoDB**: Banco de dados NoSQL, rodando dentro de um contêiner Docker.
- **Models**: Definem a estrutura dos dados usando Mongoose.
- **Repositories**: Camada de acesso ao banco de dados, interagindo com os Models.
- **Controllers**: Processam requisições HTTP, chamam os Repositories, e retornam respostas HTTP.
- **Routes**: Definem as rotas da API e associam aos Controllers.

Endpoints da API

Tickets

1. Listar Tickets

- **Endpoint:** GET /tickets
- **Descrição:** Retorna todos os tickets.
- **Resposta de Sucesso:**
    - **Status:** 200 OK
    - **Corpo:** Lista de tickets.

2. Criar Ticket

- **Endpoint:** POST /tickets
- **Descrição:** Cria um novo ticket.
- **Corpo da Requisição:**

Json {

"tituloTicket": "string",

"telefone": "string"

}

- **Respostas:**
  - **201 Created:** Ticket criado com sucesso.
  - **400 Bad Request:** Entradas inválidas.

3. Atualizar Ticket

- **Endpoint:** PUT /tickets/:id
- **Descrição:** Atualiza um ticket existente.
- **Parâmetros de URL:**
  - id (Identificador do ticket)
- **Corpo da Requisição:**

  Json {

  “idPessoa": "string",

  "tituloTicket": "string",

  "telefone": "string"

  }

- **Respostas:**
  - **204 No Content:** Atualizado com sucesso.
  - **400 Bad Request:** Entradas inválidas.
  - **404 Not Found:** Ticket não encontrado.

4. Deletar Ticket

- **Endpoint:** DELETE /tickets/:id
- **Descrição:** Remove um ticket existente.
- **Parâmetros de URL:**
  - id (Identificador do ticket)
- **Respostas:**

- **200 OK:** Deletado com sucesso.
- **404 Not Found:** Ticket não encontrado.

Atendentes

1. Listar Atendentes

- **Endpoint:** GET /atendentes
- **Descrição:** Retorna todos os atendentes.
- **Resposta de Sucesso:**
  - **Status:** 200 OK
  - **Corpo:** Lista de atendentes.

2. Criar Atendente

- **Endpoint:** POST /atendentes
- **Descrição:** Cria um novo atendente.
- **Corpo da Requisição:**

  Json {

  "idPessoa": "string",

  "idDepartamento": "string",

  "idSetor": "string"

  }
- **Respostas:**
  - **201 Created:** Atendente criado com sucesso.
  - **400 Bad Request:** Entradas inválidas.

3. Atualizar Atendente

- **Endpoint:** PUT /atendentes/:id
- **Descrição:** Atualiza um atendente existente.
- **Parâmetros de URL:**
  - id (Identificador do atendente)

- **Corpo da Requisição:**

Json {

"idPessoa": "string",

"idDepartamento": "string",

"idSetor": "string"

}

Respostas:

- **204 No Content:** Atualizado com sucesso.
- **400 Bad Request:** Entradas inválidas.
- **404 Not Found:** Atendente não encontrado.

4. Deletar Atendente

- **Endpoint:** DELETE /atendentes/:id
- **Descrição:** Remove um atendente existente.
- **Parâmetros de URL:**
  - id (Identificador do atendente)
- **Respostas:**
  - **200 OK:** Deletado com sucesso.
  - **404 Not Found:** Atendente não encontrado.

Executando a Aplicação

Pré-requisitos

- Docker instalado e funcionando.

Instruções

1. **Iniciar a aplicação:**
   - npm start
   - docker-compose up -d

2. (Assumindo que existe um docker-compose.yml configurado para a aplicação e o MongoDB).
3. **Acessar a API:**
    - A API estará disponível em http://localhost3000