ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Kauê Vinicius Samartino da Silva - 559317 João dos Santos Cardoso de Jesus - 560400 Davi Praxedes Santos Silva – 560719

Train Sentinel Tracker

Challenge CCR – Domain Driven Design

São Paulo 2025

Sumário

1. Descrição 2
2. Descrição técnica 3
3. Tabela dos Endpoints 9
4. Prototipo. 17
5. MER 20
6. Procedimentos para rodar API. 21

# Descrição:

A CCR, uma das maiores operadoras de transporte público, enfrenta desafios relacionados à segurança e à operação eficiente de sua frota de trens. Com o grande fluxo diário de passageiros, especialmente nas linhas 8 e 9 em São Paulo, atividades como o comércio ambulante se tornam comuns. Cerca de 600 mil pessoas utilizam essas linhas todos os dias, o que atrai vendedores ambulantes que oferecem produtos sem controle de procedência ou emissão de nota fiscal, o que não está de acordo com o Código de Defesa do Consumidor. Além de representar um risco para os usuários, isso prejudica os estabelecimentos legais nas estações, que perdem espaço para os vendedores informais.

Com base na necessidade de promover um ambiente mais seguro e eficiente, identificamos a oportunidade de implementar um sistema baseado em inteligência artificial (IA) que automatiza a detecção dessas ocorrências. Esse sistema busca a otimização do tempo de resposta, além de permitir uma atuação mais estratégica da equipe de segurança.

A proposta visa a criação de um sistema automatizado que detecte e registre ocorrências em tempo real, comunicando-as diretamente ao Centro de Controle Operacional (CCO). Com essa solução, buscamos:

* Aumentar a segurança dos passageiros através da rápida identificação e resposta a eventos anômalos.
* Otimizar a gestão de recursos humanos, com uma alocação mais eficiente de agentes de segurança.
* Identificar padrões de recorrência, como horários e linhas com maior incidência de ocorrências, permitindo um melhor planejamento estratégico de segurança.

A empresa de transporte público de São Paulo, ao integrar sistemas de monitoramento automatizado, registrou uma queda de 15% nas vendas não autorizadas e 25% nos casos de agressões no primeiro ano de implementação. A rápida resposta aos eventos críticos e a coleta de dados permitiram um melhor planejamento estratégico, com aumento da presença de segurança em horários de pico.

# Descrição Técnica do Sistema Sentinel

## Visão Geral

O Sentinel é um sistema robusto de gerenciamento e monitoramento de

operações ferroviárias, desenvolvido em Java utilizando o framework Quarkus. O sistema implementa uma arquitetura moderna baseada em microserviços, com foco em escalabilidade, manutenibilidade e performance.

## Arquitetura

O projeto segue uma arquitetura em camadas bem definidas:

## Camada de Modelo (Model)

* Entidades JPA que representam o domínio do negócio
* Relacionamentos complexos entre entidades
* Suporte a chaves primárias compostas
* Validações e regras de negócio embutidas

# Camada de Serviço (Service)

* Implementação da lógica de negócio
* Validações complexas
* Regras de negócio específicas
* Integração entre diferentes entidades

# Camada de Recursos (Resource)

* Endpoints RESTful
* Tratamento de requisições HTTP
* Validação de entrada
* Respostas HTTP padronizadas

# Funcionalidades Principais

## Gestão de Ocorrências

* + Registro detalhado de incidentes
  + Classificação por níveis de severidade
  + Rastreamento temporal
  + Associação com trens e estações
  + Gestão de evidências

## Gestão de Infraestrutura

* + Controle de trens e vagões
  + Gestão de estações e linhas
  + Monitoramento de câmeras
  + Controle de manutenção
  + Gestão de CCO (Centro de Controle Operacional)

## Gestão de Pessoas

* + Cadastro de funcionários
  + Gestão de terceirizados
  + Controle de cargos
  + Registro de efetivos
  + Gestão de usuários

## Monitoramento e Relatórios

* + Acompanhamento em tempo real
  + Geração de relatórios
  + Histórico de ocorrências
  + Análise de infrações
  + Rastreamento de manutenções

# Aspectos Técnicos

## Tecnologias Principais

* + Java 17
  + Quarkus Framework
  + Hibernate ORM com Panache
  + REST API
  + Oracle Database
  + Docker

## Padrões de Projeto

* + Injeção de Dependência
  + Transações Gerenciadas
  + REST Resources
  + Service Layer
  + Repository Pattern

# Persistência

* + Mapeamento objeto-relacional (ORM)
  + Relacionamentos JPA
  + Chaves primárias compostas
  + Transações gerenciadas
  + Queries otimizadas

## API REST

* + Endpoints CRUD
  + Validação de dados
  + Respostas HTTP padronizadas
  + Suporte a JSON
  + Tratamento de erros

## Segurança e Validação

* + Validação de dados de entrada
  + Tratamento de exceções
  + Respostas HTTP apropriadas
  + Controle de status
  + Níveis de severidade

## Escalabilidade

* + Arquitetura baseada em containers
  + Suporte a cloud
  + Microserviços
  + Alta disponibilidade
  + Performance otimizada

# Manutenibilidade

* + Código modular
  + Separação de responsabilidades
  + Documentação via anotações
  + Testabilidade
  + Padrões de código consistentes

# Estrutura do Projeto

O projeto é organizado em pacotes específicos:

* + model: Entidades JPA
  + resource: Endpoints REST
  + service: Lógica de negócio
  + config: Configurações do sistema

-util: Utilitários e helpers

Esta estrutura permite uma clara separação de responsabilidades e facilita a manutenção e evolução do sistema.

# Tabela dos Endpoints

|  |  |
| --- | --- |
| **URL** | **Description** |
| /ocorrencia | GET (consumes: application/json)  (produces: application/json) (List all occurrences) |
| /ocorrencia | POST (consumes: application/json) (produces: application/json) (Create  new occurrence) |
| /ocorrencia/{id} | GET (consumes: application/json) (produces: application/json) (Get  occurrence by ID) |
| /ocorrencia/{id} | PUT (consumes: application/json)  (produces: application/json) (Update occurrence by ID) |
| /ocorrencia/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete occurrence by ID) |
| /ocorrencia/trem/{numeroTrem} | GET (consumes: application/json) (produces: application/json) (Get  occurrences by train and period) |
| /funcionario | GET (consumes: application/json) (produces: application/json) (List all  employees) |
| /funcionario | POST (consumes: application/json)  (produces: application/json) (Create new employee) |
| /funcionario/{id} | GET (consumes: application/json)  (produces: application/json) (Get employee by ID) |
| /funcionario/{id} | PUT (consumes: application/json)  (produces: application/json) (Update employee by ID) |
| /funcionario/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  employee by ID) |
| /trem | GET (consumes: application/json)  (produces: application/json) (List all trains) |
| /trem | POST (consumes: application/json) (produces: application/json) (Create  new train) |
| /trem/{id} | GET (consumes: application/json) (produces: application/json) (Get train  by ID) |

|  |  |
| --- | --- |
| /trem/{id} | PUT (consumes: application/json)  (produces: application/json) (Update train by ID) |
| /trem/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  train by ID) |
| /estacao | GET (consumes: application/json) (produces: application/json) (List all  stations) |
| /estacao | POST (consumes: application/json) (produces: application/json) (Create  new station) |
| /estacao/{id} | GET (consumes: application/json)  (produces: application/json) (Get station by ID) |
| /estacao/{id} | PUT (consumes: application/json)  (produces: application/json) (Update station by ID) |
| /estacao/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  station by ID) |
| /camera | GET (consumes: application/json) (produces: application/json) (List all  cameras) |
| /camera | POST (consumes: application/json)  (produces: application/json) (Create new camera) |
| /camera/{id} | GET (consumes: application/json) (produces: application/json) (Get  camera by ID) |
| /camera/{id} | PUT (consumes: application/json) (produces: application/json) (Update  camera by ID) |
| /camera/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete camera by ID) |
| /vagao | GET (consumes: application/json)  (produces: application/json) (List all wagons) |
| /vagao | POST (consumes: application/json)  (produces: application/json) (Create new wagon) |
| /vagao/{id} | GET (consumes: application/json) (produces: application/json) (Get  wagon by ID) |

|  |  |
| --- | --- |
| /vagao/{id} | PUT (consumes: application/json)  (produces: application/json) (Update wagon by ID) |
| /vagao/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  wagon by ID) |
| /manutencao | GET (consumes: application/json) (produces: application/json) (List all  maintenances) |
| /manutencao | POST (consumes: application/json) (produces: application/json) (Create  new maintenance) |
| /manutencao/{id} | GET (consumes: application/json)  (produces: application/json) (Get maintenance by ID) |
| /manutencao/{id} | PUT (consumes: application/json)  (produces: application/json) (Update maintenance by ID) |
| /manutencao/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  maintenance by ID) |
| /infrator | GET (consumes: application/json) (produces: application/json) (List all  offenders) |
| /infrator | POST (consumes: application/json)  (produces: application/json) (Create new offender) |
| /infrator/{id} | GET (consumes: application/json) (produces: application/json) (Get  offender by ID) |
| /infrator/{id} | PUT (consumes: application/json) (produces: application/json) (Update  offender by ID) |
| /infrator/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete offender by ID) |
| /relatorio | GET (consumes: application/json)  (produces: application/json) (List all reports) |
| /relatorio | POST (consumes: application/json)  (produces: application/json) (Create new report) |
| /relatorio/{id} | GET (consumes: application/json) (produces: application/json) (Get  report by ID) |

|  |  |
| --- | --- |
| /relatorio/{id} | PUT (consumes: application/json)  (produces: application/json) (Update report by ID) |
| /relatorio/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  report by ID) |
| /cco | GET (consumes: application/json) (produces: application/json) (List all  CCOs) |
| /cco | POST (consumes: application/json) (produces: application/json) (Create  new CCO) |
| /cco/{id} | GET (consumes: application/json)  (produces: application/json) (Get CCO by ID) |
| /cco/{id} | PUT (consumes: application/json)  (produces: application/json) (Update CCO by ID) |
| /cco/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  CCO by ID) |
| /cargo | GET (consumes: application/json) (produces: application/json) (List all  roles) |
| /cargo | POST (consumes: application/json)  (produces: application/json) (Create new role) |
| /cargo/{id} | GET (consumes: application/json) (produces: application/json) (Get role  by ID) |
| /cargo/{id} | PUT (consumes: application/json) (produces: application/json) (Update  role by ID) |
| /cargo/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete role by ID) |
| /tercerizado | GET (consumes: application/json)  (produces: application/json) (List all outsourced) |
| /tercerizado | POST (consumes: application/json)  (produces: application/json) (Create new outsourced) |
| /tercerizado/{id} | GET (consumes: application/json) (produces: application/json) (Get  outsourced by ID) |

|  |  |
| --- | --- |
| /tercerizado/{id} | PUT (consumes: application/json)  (produces: application/json) (Update outsourced by ID) |
| /tercerizado/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  outsourced by ID) |
| /usuario | GET (consumes: application/json) (produces: application/json) (List all  users) |
| /usuario | POST (consumes: application/json) (produces: application/json) (Create  new user) |
| /usuario/{id} | GET (consumes: application/json)  (produces: application/json) (Get user by ID) |
| /usuario/{id} | PUT (consumes: application/json)  (produces: application/json) (Update user by ID) |
| /usuario/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  user by ID) |
| /evidencia | GET (consumes: application/json) (produces: application/json) (List all  evidence) |
| /evidencia | POST (consumes: application/json)  (produces: application/json) (Create new evidence) |
| /evidencia/{id} | GET (consumes: application/json) (produces: application/json) (Get  evidence by ID) |
| /evidencia/{id} | PUT (consumes: application/json) (produces: application/json) (Update  evidence by ID) |
| /evidencia/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete evidence by ID) |
| /linha | GET (consumes: application/json)  (produces: application/json) (List all lines) |
| /linha | POST (consumes: application/json)  (produces: application/json) (Create new line) |
| /linha/{id} | GET (consumes: application/json) (produces: application/json) (Get line  by ID) |

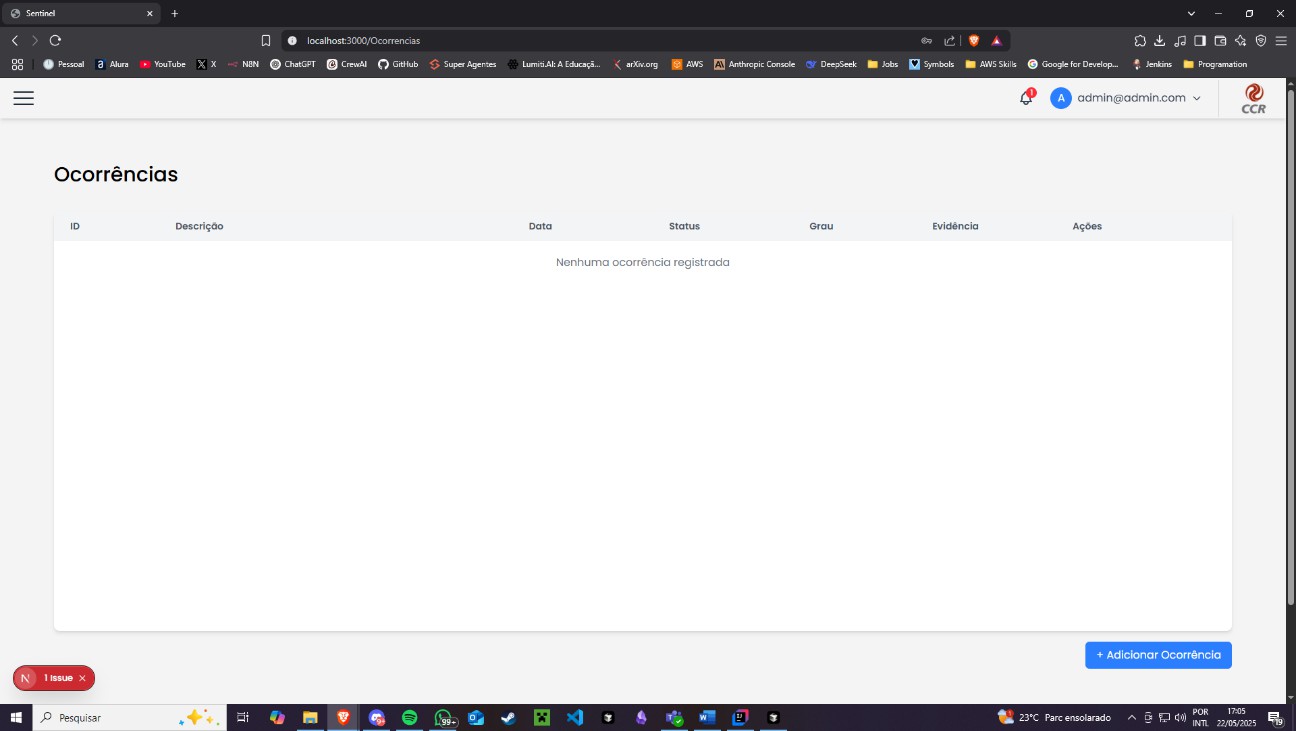
|  |  |
| --- | --- |
| /linha/{id} | PUT (consumes: application/json)  (produces: application/json) (Update line by ID) |
| /linha/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  line by ID) |
| /estacao-linha | GET (consumes: application/json) (produces: application/json) (List all  station-line relationships) |
| /estacao-linha | POST (consumes: application/json) (produces: application/json) (Create  new station-line relationship) |
| /estacao-linha/{id} | GET (consumes: application/json)  (produces: application/json) (Get station-line relationship by ID) |
| /estacao-linha/{id} | PUT (consumes: application/json)  (produces: application/json) (Update station-line relationship by ID) |
| /estacao-linha/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  station-line relationship by ID) |
| /funcionario-ocorrencia | GET (consumes: application/json) (produces: application/json) (List all  employee-occurrence relationships) |
| /funcionario-ocorrencia | POST (consumes: application/json) (produces: application/json) (Create new employee-occurrence  relationship) |
| /funcionario-ocorrencia/{id} | GET (consumes: application/json) (produces: application/json) (Get  employee-occurrence relationship by  ID) |
| /funcionario-ocorrencia/{id} | PUT (consumes: application/json) (produces: application/json) (Update employee-occurrence relationship by  ID) |
| /funcionario-ocorrencia/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete employee-occurrence relationship by  ID) |
| /usuario-ocorrencia | GET (consumes: application/json) (produces: application/json) (List all  user-occurrence relationships) |
| /usuario-ocorrencia | POST (consumes: application/json)  (produces: application/json) (Create new user-occurrence relationship) |

|  |  |
| --- | --- |
| /usuario-  ocorrencia/{usuarioId}/{ocorrenciaId} | GET (consumes: application/json)  (produces: application/json) (Get user- occurrence relationship by IDs) |
| /usuario-  ocorrencia/{usuarioId}/{ocorrenciaId} | PUT (consumes: application/json) (produces: application/json) (Update  user-occurrence relationship by IDs) |
| /usuario-  ocorrencia/{usuarioId}/{ocorrenciaId} | DELETE (consumes: application/json) (produces: application/json) (Delete  user-occurrence relationship by IDs) |
| /camera-ocorrencia | GET (consumes: application/json) (produces: application/json) (List all  camera-occurrence relationships) |
| /camera-ocorrencia | POST (consumes: application/json)  (produces: application/json) (Create new camera-occurrence relationship) |
| /camera-ocorrencia/{id} | GET (consumes: application/json)  (produces: application/json) (Get camera-occurrence relationship by ID) |
| /camera-ocorrencia/{id} | PUT (consumes: application/json) (produces: application/json) (Update  camera-occurrence relationship by ID) |
| /camera-ocorrencia/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  camera-occurrence relationship by ID) |
| /efetivo | GET (consumes: application/json)  (produces: application/json) (List all personnel) |
| /efetivo | POST (consumes: application/json) (produces: application/json) (Create  new personnel record) |
| /efetivo/{id} | GET (consumes: application/json) (produces: application/json) (Get  personnel record by ID) |
| /efetivo/{id} | PUT (consumes: application/json)  (produces: application/json) (Update personnel record by ID) |
| /efetivo/{id} | DELETE (consumes: application/json)  (produces: application/json) (Delete personnel record by ID) |
| /vagao-carro | GET (consumes: application/json)  (produces: application/json) (List all wagon-car relationships) |
| /vagao-carro | POST (consumes: application/json) (produces: application/json) (Create  new wagon-car relationship) |

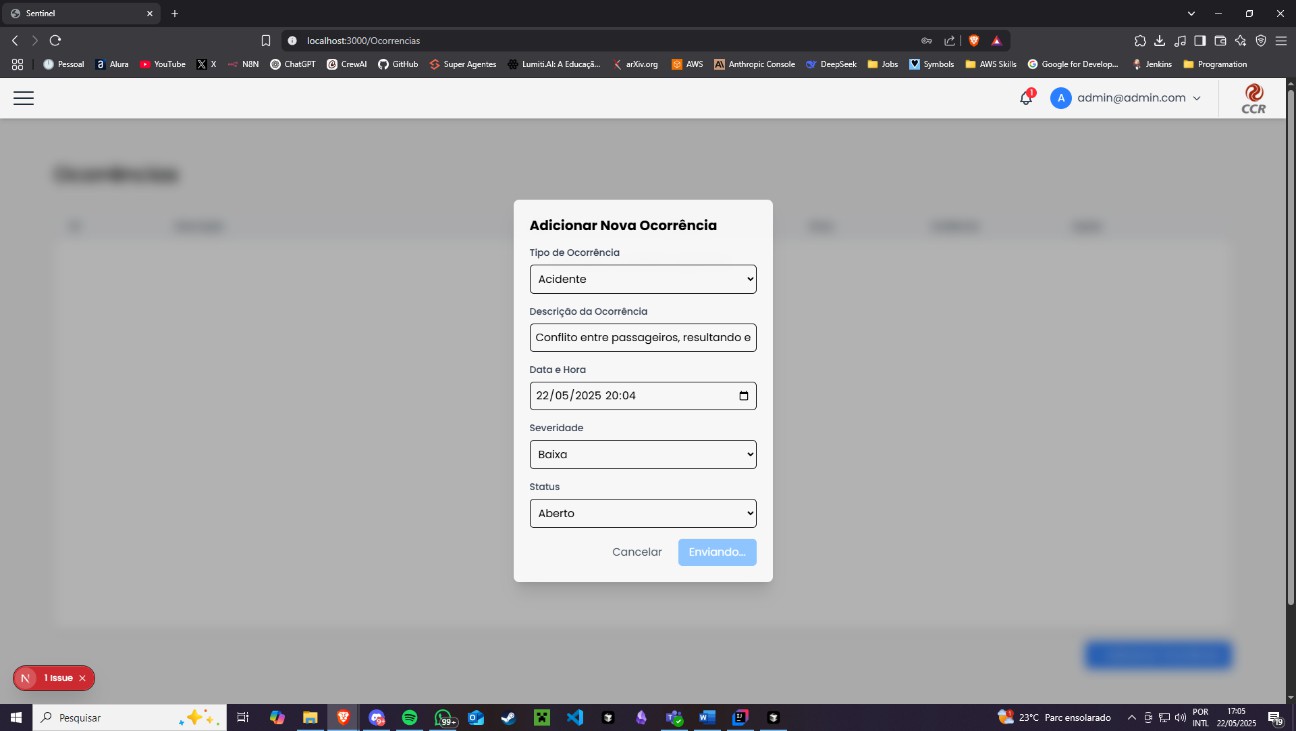
|  |  |
| --- | --- |
| /vagao-carro/{id} | GET (consumes: application/json)  (produces: application/json) (Get wagon-car relationship by ID) |
| /vagao-carro/{id} | PUT (consumes: application/json) (produces: application/json) (Update  wagon-car relationship by ID) |
| /vagao-carro/{id} | DELETE (consumes: application/json) (produces: application/json) (Delete  wagon-car relationship by ID) |

**Protótipo**

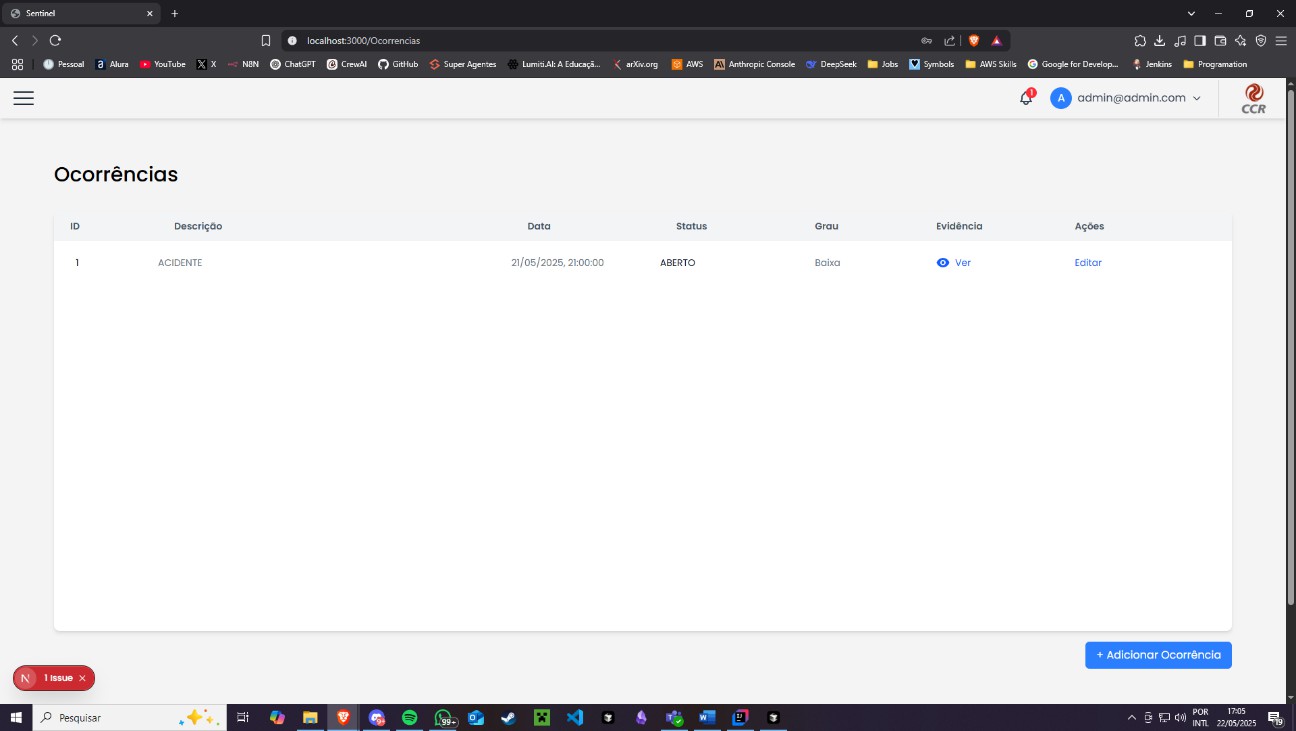
**Usuário abre tela de ocorrências**

****

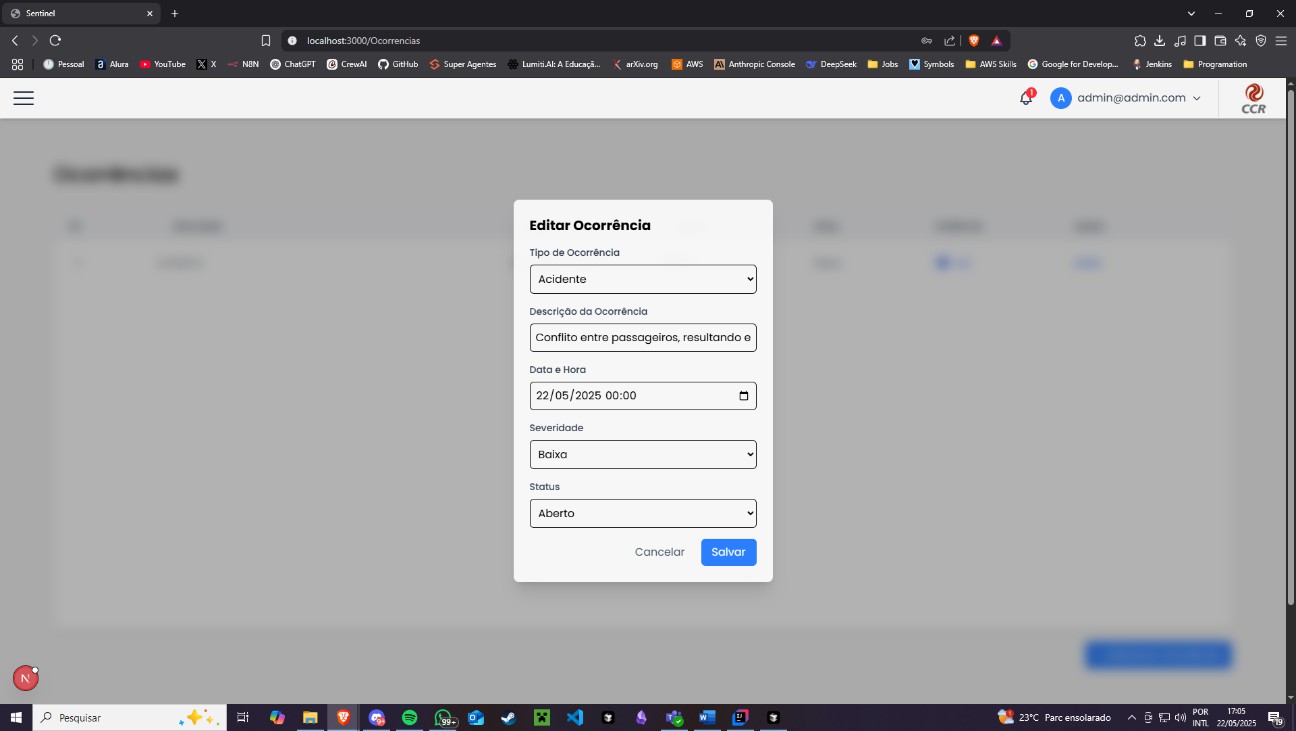
**Usuário decide adicionar nova ocorrência e preenche o formulário com as informações da ocorrência**

****

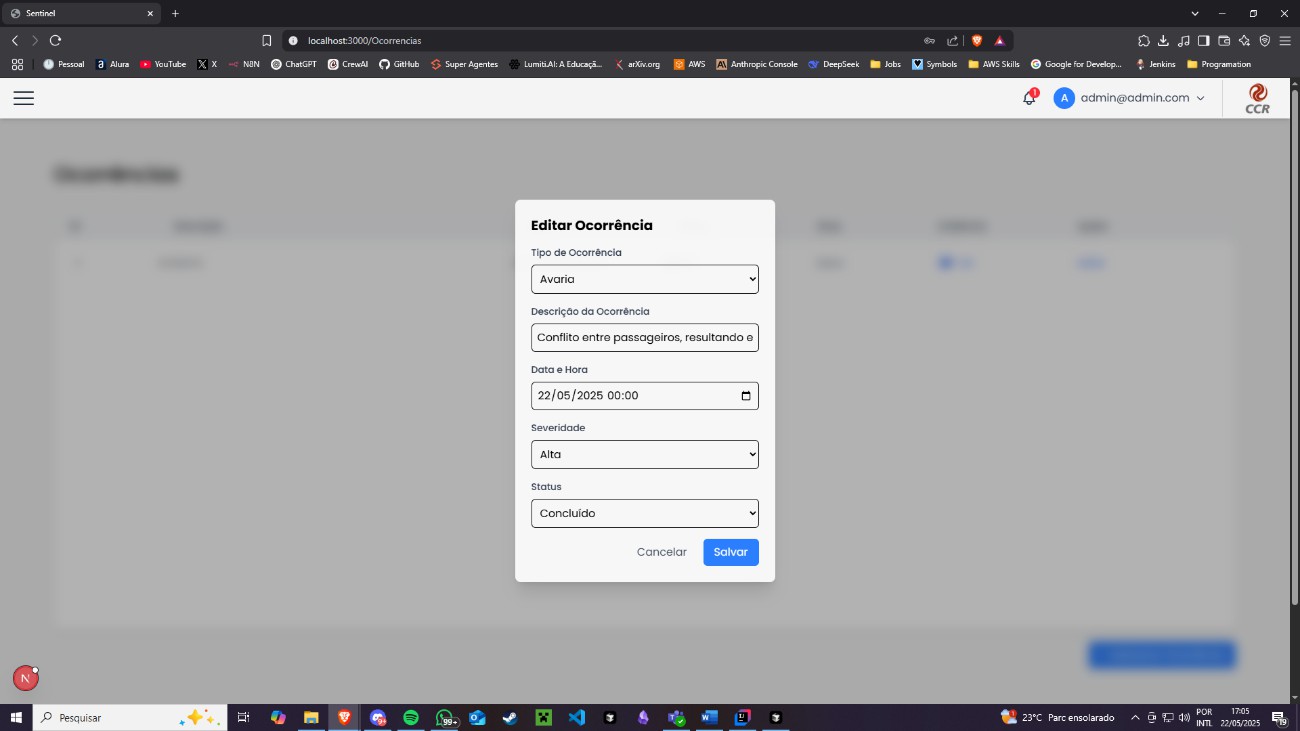
**Usuário consegue visualizar a ocorrência adicionada na tabela de ocorrências**

****

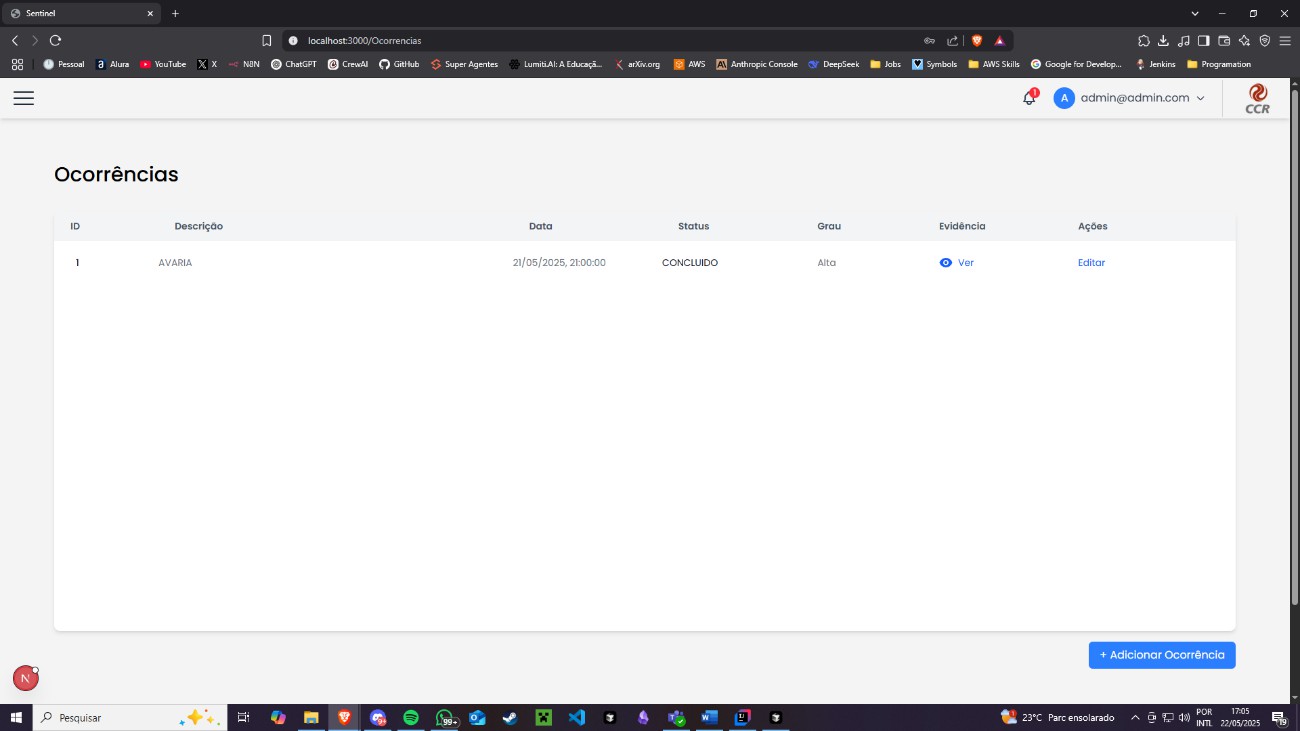
**Usuário decide editar informações adicionadas na tabela**

****

**Usuário altera informações e aperta em salvar**

****

**Usuário consegue visualizar os dados atualizados da ocorrência. Dados também são atualizados no banco de dados.**

****

# MER – Modelo Entidade Relacionamento

**Procedimentos para rodar a aplicação**

**Para rodar o projeto localmente, certifique-se de ter os seguintes pré- requisitos instalados:**

* JDK 17
* Maven 3.9+
* Acesso a uma instância do Oracle Database configurada. As credenciais e URL de conexão devem ser configuradas no arquivo

src/main/resources/application.properties.

**Passos para executar a API localmente:**

1. Clone o repositório do projeto em https://github.com/KadajoFiap/Java- Sentinel-API.
2. Navegue até o diretório raiz do projeto no terminal.
3. Execute o seguinte comando Maven para compilar e iniciar a aplicação no modo de desenvolvimento (Quarkus Dev Mode):

**mvn quarkus:dev**

O Dev Mode permite recarregamento a quente (hot reloading) das mudanças no código.

1. A API estará disponível em [http://localhost:8080.](http://localhost:8080/) O Dev UI, com a lista de endpoints e outras ferramentas, pode ser acessado em

[http://localhost:8080/q/dev-ui.](http://localhost:8080/q/dev-ui)

Build e Execução com Docker

Você pode construir e executar a aplicação em um container Docker.

1. Certifique-se de ter o Docker instalado e rodando.
2. Navegue até o diretório raiz do projeto no terminal.
3. Construa a imagem Docker utilizando o Dockerfile na raiz do projeto:

**docker build -t sentinel-api .**

(Este comando utiliza o Dockerfile que configura a build do fast-jar)

1. Execute o container Docker, mapeando a porta 8080: docker run -i --rm -p 8080:8080 sentinel-api

A API estará disponível em [http://localhost:8080](http://localhost:8080/) no seu host Docker. Endpoints da API - Ocorrências

A API expõe endpoints específicos para gerenciar ocorrências. A URL base para acesso é [http://localhost:8080](http://localhost:8080/) (ou a URL de deploy no Render).

**Lista dos endpoints de ocorrências:**

* URL: /ocorrencia
  + Método: GET
  + Descrição: Lista todas as ocorrências
  + Detalhes: Produz application/json
* URL: /ocorrencia
  + Método: POST
  + Descrição: Cria ocorrência
  + Detalhes: Consome e produz application/json

**Json/content para requisição no postman:**

{

"dataInicio": "YYYY-MM-DDTHH:MM:SS", "dataFim": "YYYY-MM-DDTHH:MM:SS",

"tipoOcorrencia": "TIPO\_DA\_OCORRENCIA",

"descricaoOcorrencia": "Descrição detalhada da ocorrência", "severidadeOcorrencia": "BAIXA" // ou "MEDIA", "ALTA", "CRITICA"

// "statusOcorrencia": "FECHADO" // Opcional, padrão é "ABERTO"

}

* URL: /ocorrencia/{id}
  + Método: GET
  + Descrição: Busca ocorrência por ID
  + Detalhes: Produz application/json
* URL: /ocorrencia/{id}
  + Método: PUT
  + Descrição: Atualiza ocorrência por ID
  + Detalhes: Consome e produz application/json

**Json/content para requisição no postman:**

{

"dataInicio": "YYYY-MM-DDTHH:MM:SS", "dataFim": "YYYY-MM-DDTHH:MM:SS",

"tipoOcorrencia": "TIPO\_DA\_OCORRENCIA\_ATUALIZADO", "descricaoOcorrencia": "Nova descrição da ocorrência",

"severidadeOcorrencia": "ALTA", // ou "BAIXA", "MEDIA", "CRITICA"

"statusOcorrencia": "EM\_ANDAMENTO" // Exemplo de atualização de status

}

* URL: /ocorrencia/{id}
  + Método: DELETE
  + Descrição: Remove ocorrência por ID
  + Detalhes: Consome application/json