DOMAIN DRIVEN DESIGN USING JAVA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

-Gabriel Nakamura Ogata RM560671

- Guilherme Costeira Braganholo RM560628

- Julio Cesar Dias Vilella RM560494

PROJETO DE AUTOMAÇÃO DO CENTRO DE CONTROLE DE OPERAÇÕES (CCO)

São Paulo

2024

SUMÁRIO

Objetivo e Escopo do Projeto

O projeto tem como objetivo modernizar e automatizar as atividades do Centro de Controle de Operações (CCO) da CCR. Por meio da implementação de tecnologias como IoT, algoritmos de inteligência artificial e dashboards interativos, busca-se melhorar a eficiência operacional, aumentar a segurança e reduzir falhas humanas. O sistema permite monitoramento em tempo real da malha ferroviária, geração automática de relatórios e identificação preditiva de falhas. A aplicação permite operações básicas de cadastro, consulta e listagem desses domínios, estruturados em camadas de Model, DAO, Service e Controller

Modelo descritivo do negócio:

Automação do Centro de Controle de Operações (CCO) da CCR: O projeto de automação do Centro de Controle de Operações (CCO) da CCR tem como objetivo otimizar as operações ferroviárias, reduzir o trabalho manual e melhorar a eficiência e segurança do sistema. A CCR visa modernizar sua infraestrutura operacional por meio da implementação de tecnologias avançadas como inteligência artificial, algoritmos de otimização e dashboards interativos. Os principais objetivos que permitem a validação são: a geração automática de relatórios operacionais que irão ser gerados automaticamente para fornecer informações detalhadas sobre o desempenho operacional, que incluem: análise das manutenções realizadas, desempenho das operações e o tempo de inatividade. Isso tem como principal objetivo a melhora do controle operacional e fornecimento dados para melhorar a estratégia de gestão dos recursos. Outro objetivo é o monitoramento automatizado de trens e infraestruturas, que vai funcionar incluindo o monitoramento contínuo e automatizado de todas as linhas de trens e das estruturas ferroviárias em tempo real, na qual serão usados sensores de IOT e câmeras com o objetivo de reduzir a necessidade de monitoramentos manuais, e prever falhas antes delas acontecerem. E por fim a utilização de Inteligência Artificial para análise de dados em tempo real, que irá funcionar por meio de algoritmos de Inteligência Artificial que irão processar os dados coletados em tempo real e identificar falhas ou problemas na operação. Essa inteligência irá analisar o comportamento dos trens e identificar possíveis incidentes, o que irá prevenir possíveis lotações em horários de pico e melhorar a segurança dos passageiros.

Principais Funcionalidades + Métodos com Lógica

1.CADASTRAR USUÁRIO:

Descrição:

Este método cadastra um novo usuário no sistema, realizando validações de nome e email antes de persistir os dados.

```java
package service;

import ...

public class UsuarioClienteService {  3 usages
    private UsuarioClienteDAO dao = new UsuarioClienteDAO();  2 usages

    public void cadastrarUsuario(UsuarioCliente usuario) throws IllegalArgumentException {  1 usage
        if (usuario.getNome() == null || usuario.getNome().isEmpty()) {
            throw new IllegalArgumentException("Nome não pode ser vazio.");
        }

        if (usuario.getEmail() == null || !usuario.getEmail().contains("@")) {
            throw new IllegalArgumentException("Email inválido.");
        }

        dao.inserir(usuario);
    }
```

2. Listar Usuários

Descrição:

Este método retorna uma lista com todos os usuários cadastrados no sistema.

```java
    public List<UsuarioCliente> listarUsuarios() { return dao.listarTodos(); }
}
```

3.Cadastrar equipe

Descrição:

Este método cadastra uma nova equipe no sistema, validando se a severidade está preenchida corretamente.

```java
public void cadastrarEquipe(Equipe equipe) throws IllegalArgumentException {  1 usage
    if (equipe.getSeveridade() == null || equipe.getSeveridade().isEmpty()) {
        throw new IllegalArgumentException("Severidade não pode ser vazia.");
    }

    dao.inserir(equipe);
}
```

4.LISTAR EQUIPES

Descrição:
Este método retorna uma lista com todas as equipes cadastradas no sistema.

```java
    public List<Equipe> listarEquipes() { return dao.listarTodos(); }
}
```

5.**CONEXÃO COM BANCO ORACLE**

Descrição:
Este método estabelece a conexão com o banco de dados Oracle utilizando JDBC, com parâmetros de configuração padrão.

```java
    public static Connection conectar() throws SQLException {  4 usages
        return DriverManager.getConnection(URL, USUARIO, SENHA);
    }
}
```

4. Tabela de Endpoints (API Restful)

URI	Verbo HTTP	Status de Resposta
`/usuarios/cadastrar`	POST	201 Created
`/usuarios/{id}`	GET	200 OK / 404 Not Found
`/equipes/cadastrar`	POST	201 Created

`/equipes/{id}`	GET	200 OK / 404 Not Found

5.PROTÓTIPO- JORNADA DO USUÁRIO

A aplicação é executada via **console**.

Fluxo:

1. O usuário executa a aplicação
2. Escolhe a operação: cadastro, listagem etc.
3. Informa os dados no console
4. Recebe mensagem de sucesso ou erro

Exemplo:

```
Digite o nome do usuário: João
Usuário cadastrado com sucesso!
```

Modelo do Banco de Dados (MER)

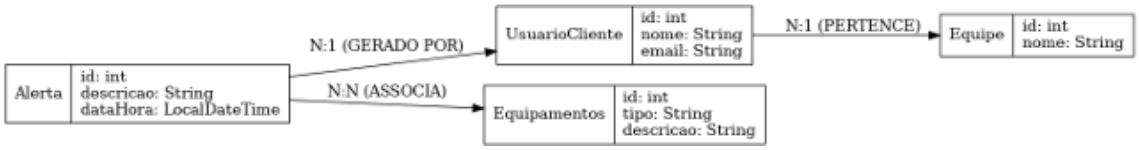
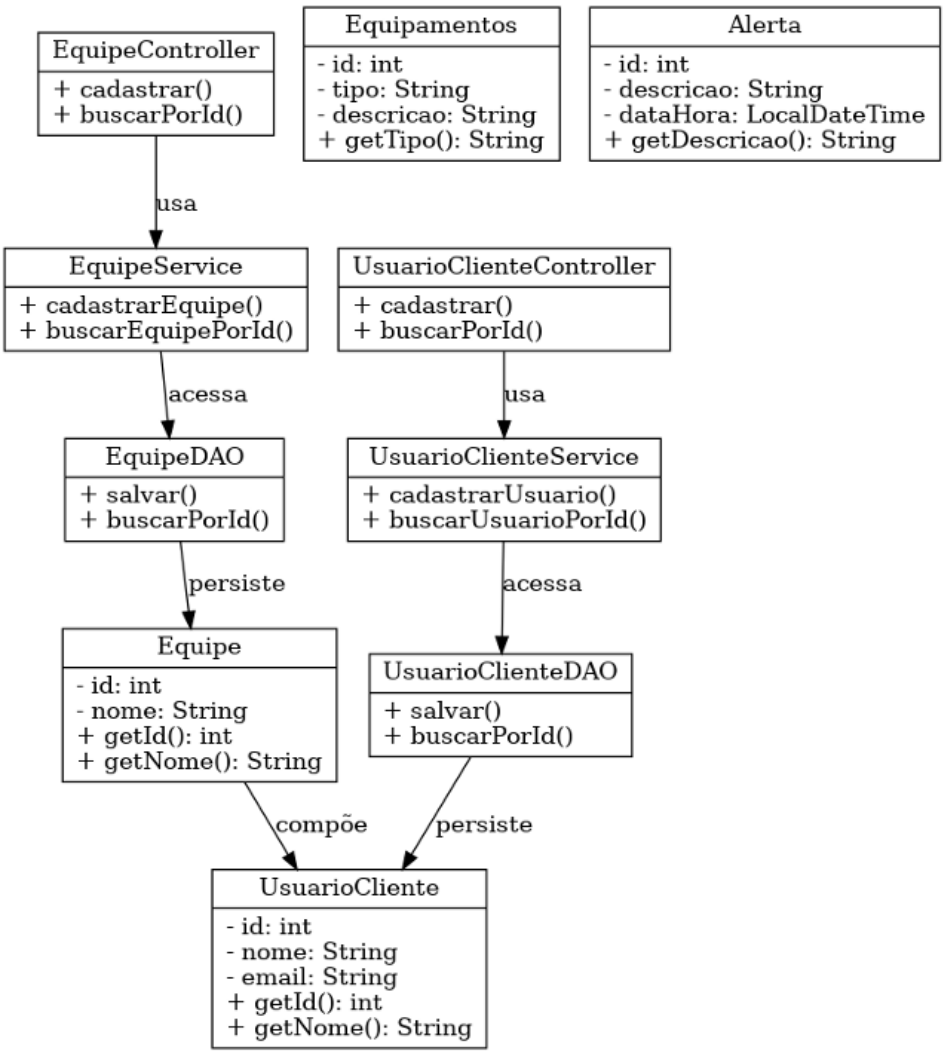


Diagrama de Classes (UML)



8. Procedimentos para Rodar a Aplicação

1. Abrir o projeto no **IntelliJ IDEA** ou **Eclipse**
2. Configurar a dependência do **Driver JDBC Oracle**
3. Alterar as credenciais no arquivo **ConexaoOracle.java**:
4. Executar a classe **Main.java**
5. Seguir as instruções no **console**