Documento de Entrega: Trabalho Prático – Engenharia de Software II (MDA)

1. Introdução e Descrição do Sistema

1.1 Objetivo

O objetivo deste trabalho é aplicar os conceitos de **Model Driven Architecture (MDA)** ao projeto legado **Suporte TI CRUD**. O projeto consiste em um sistema de gerenciamento de suporte técnico desenvolvido em Java, permitindo o controle de usuários e tickets de atendimento.

1.2 Descrição do Sistema

O **Suporte TI CRUD** é uma aplicação Desktop destinada ao gerenciamento de demandas de TI. Ele permite que usuários sejam cadastrados e abram chamados relatando problemas ou solicitações.

Principais Funcionalidades:

- **Autenticação:** Login de usuários via e-mail e senha (com hash SHA-256).
- **Gestão de Usuários:** Cadastro, edição, listagem e exclusão de usuários.
- **Gestão de Tickets:** Criação de tickets com título, descrição, status (Aberto, Em Andamento, Fechado) e solução.
- **Persistência:** Os dados são armazenados em banco de dados relacional através de conexões JDBC manuais e padrão DAO.

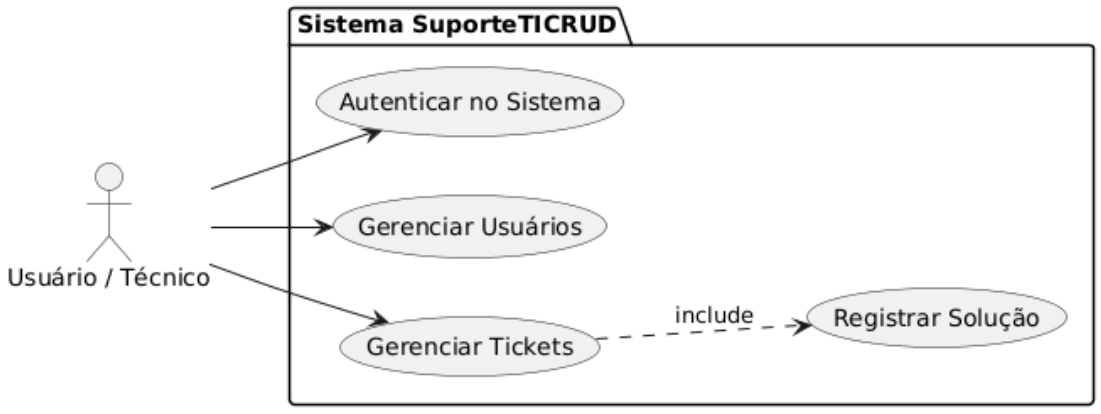
---

2. Modelagem MDA

2.1 CIM (Computation Independent Model)

Modelo independente de computação, focado nos requisitos e no ambiente do negócio.

Diagrama de Casos de Uso:

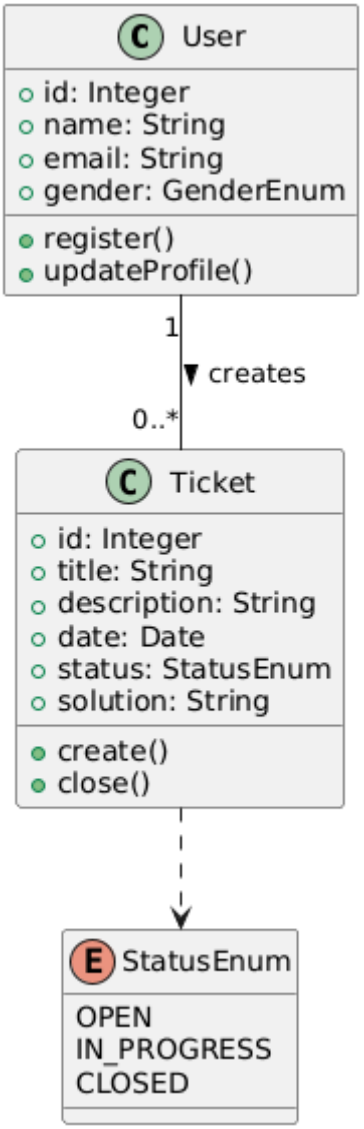


Descrição: O ator principal interage com o sistema para acessar sua conta, manter o cadastro de outros usuários e gerenciar o ciclo de vida dos tickets de suporte.

2.2 PIM (Platform Independent Model)

Modelo independente de plataforma, focado na estrutura lógica e regras de negócio, sem detalhes de linguagem (Java) ou banco (SQL).

Diagrama de Classes Lógico:

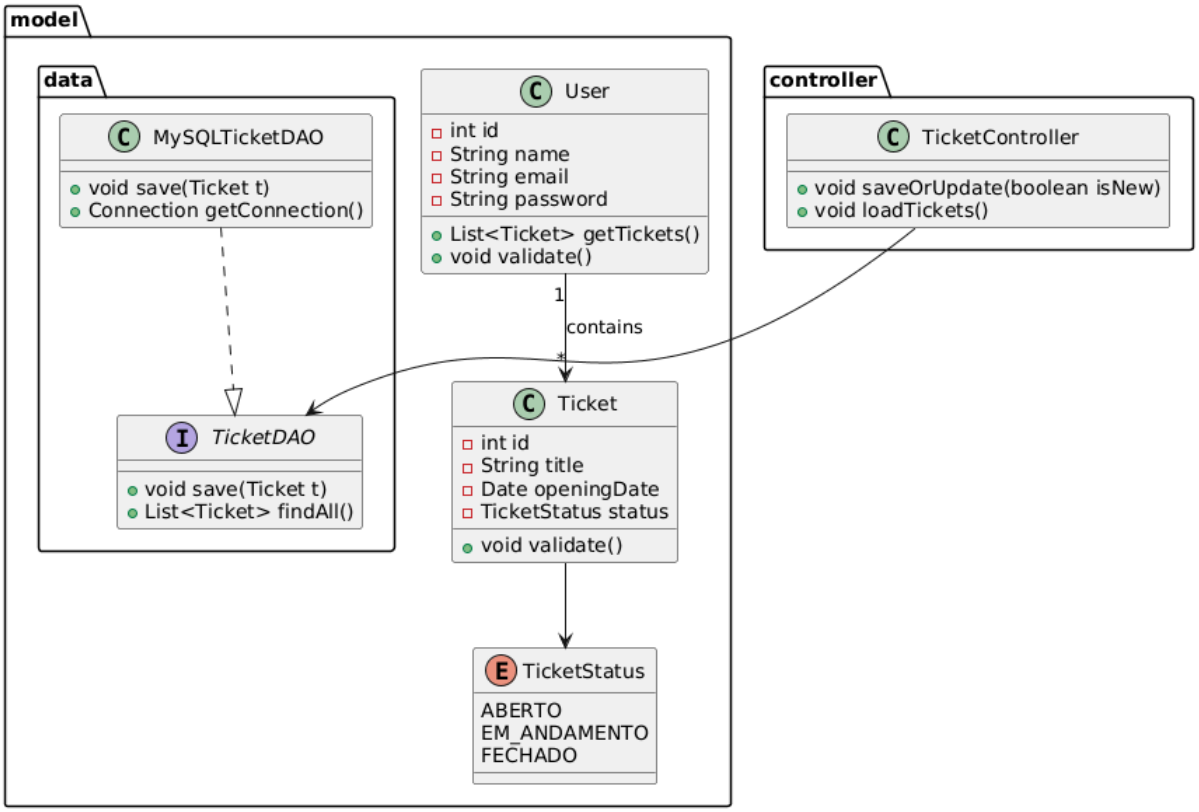


Descrição: O PIM define as entidades User e Ticket e seus relacionamentos lógicos. Tipos genéricos são usados, e não há menção a PreparedStatement, JFrame ou bibliotecas específicas.

2.3 PSM (Platform Specific Model)

Modelo específico de plataforma, detalhando a implementação na tecnologia Java, JDBC e Swing.

Diagrama de Classes de Implementação (Notação PlantUML):



Descrição: O PSM reflete a estrutura exata do código Java, incluindo tipos específicos do Java, padrões de projeto e visibilidade de atributos.

---

3. Geração e Mapeamento de Código

Classe Selecionada: Ticket

Modelo PSM (UML):

- **Classe:** Ticket
- **Atributos:** id (int), title (String), status (TicketStatus), user (User).
- **Métodos:** Getters, Setters e validate().

Código Gerado / Existente:

Java

```
package model;


import java.util.Date;


public class Ticket {


    private int id;
    private String title;
    private String description;
    private Date openingDate;
    private TicketStatus status;
    private String solution;
    private User user;


    public Ticket(int id) {
        this.id = id;
    }


    public void validate() {
        if (title == null || title.isBlank()) {
            throw new IllegalArgumentException("O Título do ticket não pode ser vazio.");
        }
        // ... outras validações
    }
}
```

[Fonte: src/model/Ticket.java]

Ajustes Manuais Necessários:

Ao gerar código automaticamente a partir de ferramentas CASE (como StarUML), normalmente obtemos apenas a "casca" da classe (assinaturas). Os seguintes ajustes manuais são necessários após a geração:

1. **Lógica de Negócio:** O conteúdo do método validate() (os ifs e throw) não pode ser gerado apenas pelo diagrama de classes, devendo ser codificado manualmente.
2. **Persistência (SQL):** No caso das classes DAO (MySQLTicketDAO), a ferramenta gera os métodos save ou delete, mas o código SQL interno (INSERT INTO tickets...) e o tratamento de SQLException devem ser implementados pelo desenvolvedor.

---

4. Análise Crítica

Quais vantagens percebeu na abordagem orientada por modelos?

A principal vantagem é a abstração e comunicação. O modelo CIM permite discutir requisitos com stakeholders sem se preocupar com código. O PIM ajuda a estruturar o banco de dados e as relações antes de programar. Além disso, a documentação se mantém mais coerente com o sistema, facilitando a manutenção e a entrada de novos desenvolvedores no projeto.

Quais limitações encontrou ao gerar código automaticamente?

Ferramentas de MDA geralmente geram bem a estrutura estática, mas falham na lógica comportamental complexa. Detalhes como queries SQL específicas, validações de regras de negócio complexas e lógica de interface gráfica dificilmente são gerados de forma completa, exigindo intervenção manual significativa.

Em que tipos de projeto o uso da MDA seria mais vantajoso?

A MDA é mais vantajosa em sistemas complexos e de longa duração, onde a independência de tecnologia é crucial. Se o projeto precisar migrar de Java para C#, ou de MySQL para MongoDB, o modelo PIM permanece o mesmo, sendo necessário apenas regenerar o PSM e o código. Para projetos pequenos como este CRUD, o overhead de manter os modelos sincronizados pode ser custoso em comparação à codificação direta.