Projeto: Sistema de Curadoria e Compartilhamento de Recursos

Autor: Gerado automaticamente - pacote para A3 (entrega)

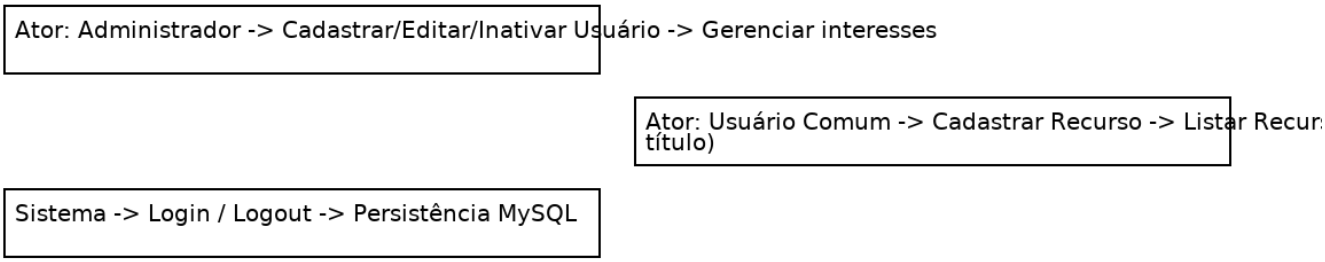
Conteúdo: requisitos, diagramas UML, SQL, código-fonte (resumo).

Requisitos Funcionais (selecionados):
- RF01: Login (email + senha).
- RF02: Controle de Acesso: apenas usuários autenticados.
- RF03: Perfis: ADMIN, COMUM.
- RF04: Admin: cadastrar/editar/inativar usuários, gerir interesses (até 2).
- RF05: Usuário Comum: cadastrar recursos (título, autor, categoria).
- RF06: Listagem de recursos ordenada por título.
- RF07: Menu principal com navegação.
- RF08: Logout.

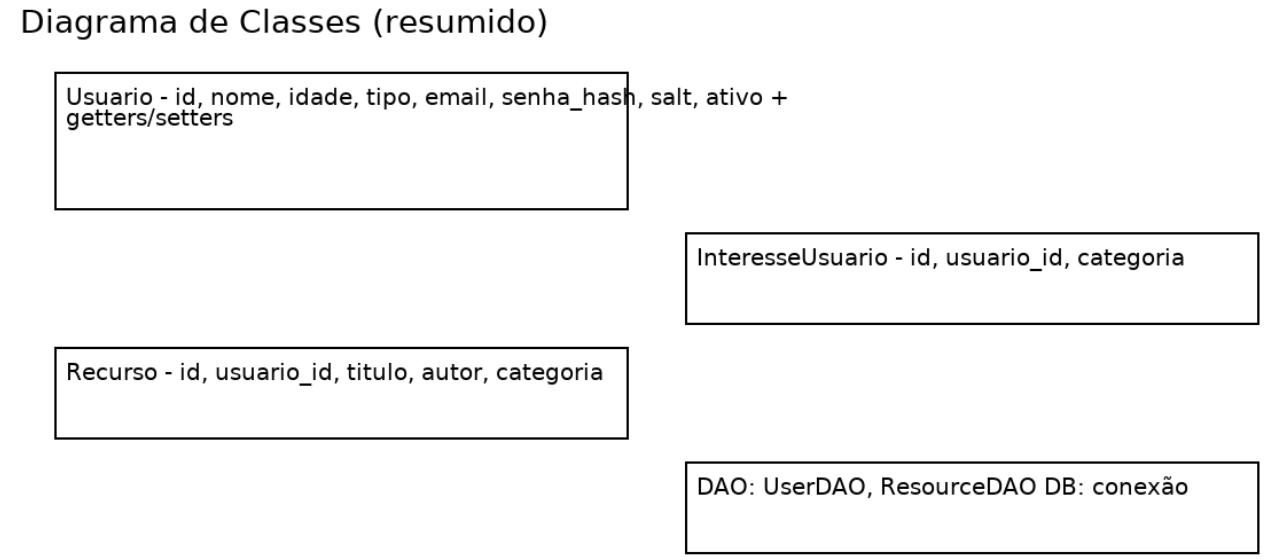
Requisitos Não Funcionais:
- Interface: Java Swing (javax.swing).
- BD: MySQL.
- Senhas: armazenadas com hash (SHA-256 + salt neste projeto, recomendado bcrypt/Argon2).
- Categorias fixas: IA, CIBER, PRIVACIDADE.

Use Case

Diagrama de Casos de Uso (resumido)



Class Diagram



Sequence Login

Diagrama de Sequência: Login (resumido)

Usuário -> Tela Login: inserir email/senha

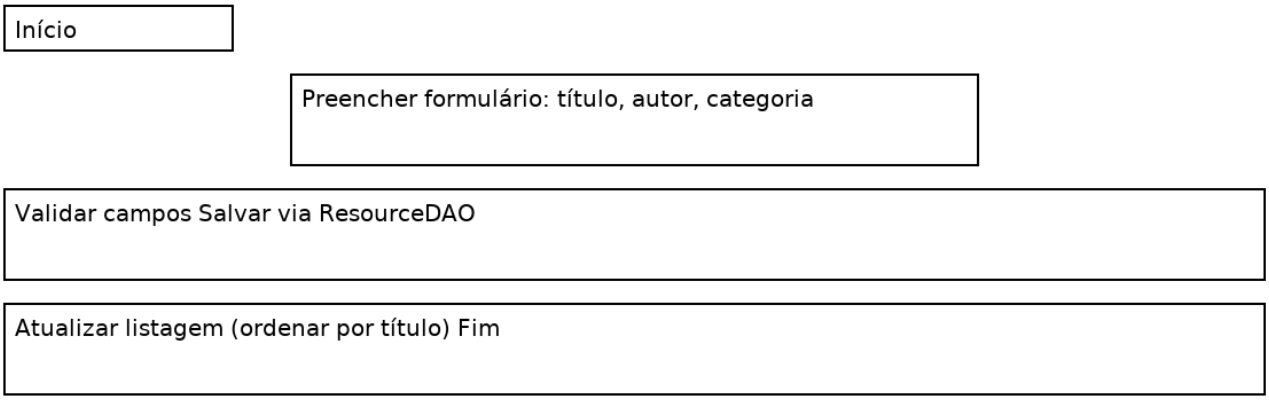
LoginFrame -> UserDAO.login(email, pass)

UserDAO -> DB: SELECT usuario WHERE email=?

DB -> UserDAO: dados (hash + salt) UserDAO verifica hash Resposta: usuário ou falha

Activity - Recurso

Diagrama de Atividade: Cadastrar Recurso



SQL: schema.sql (criação do banco e seed admin)

```
-- Schema for curadoria project
CREATE DATABASE IF NOT EXISTS curadoria DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE curadoria;
CREATE TABLE IF NOT EXISTS usuario (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  idade INT,
  tipo ENUM('ADMIN','COMUM') NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  senha_hash VARCHAR(255) NOT NULL,
  salt VARCHAR(64) NOT NULL,
  ativo BOOLEAN DEFAULT TRUE
);
CREATE TABLE IF NOT EXISTS interesse_usuario (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT NOT NULL,
  categoria ENUM('IA','CIBER','PRIVACIDADE') NOT NULL,
  FOREIGN KEY (usuario_id) REFERENCES usuario(id) ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS recurso (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT NOT NULL,
  titulo VARCHAR(200) NOT NULL,
  autor VARCHAR(100),
  categoria ENUM('IA','CIBER','PRIVACIDADE') NOT NULL,
  FOREIGN KEY (usuario_id) REFERENCES usuario(id) ON DELETE CASCADE
);
-- Seed admin user (email: admin@local, senha: admin123)
-- Uses SHA256(salt + password). Salt and hash below generated and inserted.
INSERT INTO usuario (nome, idade, tipo, email, senha_hash, salt, ativo)
VALUES ('Administrador', 30, 'ADMIN', 'admin@local', '9f02b427a22d3427ce52533f689aac3b8870bb9fe13a69911d979c2e
7e94b0cb', 'df39195294b1586305e3f10a5c66b434', TRUE)
ON DUPLICATE KEY UPDATE nome=VALUES(nome);
```

Código: trechos selecionados

```
// Main.java
package app;
import javax.swing.SwingUtilities;
import gui.LoginFrame;
public class Main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new LoginFrame();
        });
    }
}
// UserDAO.java (login snippet)
package dao;
import model.User;
import db.DB;
import java.sql.*;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.*;
public class UserDAO {
    public static User login(String email, String password) {
        try (Connection c = DB.getConnection()) {
            String sql = "SELECT id,nome,idade,tipo,email,senha_hash,salt,ativo FROM usuario WHERE email=?";
            PreparedStatement ps = c.prepareStatement(sql);
            ps.setString(1,email);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                String hash = rs.getString("senha_hash");
                String salt = rs.getString("salt");
                String candidate = sha256(salt + password);
                if (hash.equals(candidate)) {
                    User u = new User();
                    u.setId(rs.getInt("id"));
                    u.setNome(rs.getString("nome"));
                    u.setIdade(rs.getInt("idade"));
                    u.setTipo(rs.getString("tipo"));
                    u.setEmail(rs.getString("email"));
                    u.setAtivo(rs.getBoolean("ativo"));
                    return u;
                }
            }
        } catch(Exception ex){ ex.printStackTrace(); }
        return null;
    }
    public static List<User> findAll() {
        List<User> list = new ArrayList<>();
        try (Connection c = DB.getConnection()) {
            PreparedStatement ps = c.prepareStatement("SELECT id,nome,idade,tipo,email,ativo FROM usuario");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                User u = new User();
                u.setId(rs.getInt("id"));
                u.setNome(rs.getString("nome"));
                u.setIdade(rs.getInt("idade"));
                u.setTipo(rs.getString("tipo"));
                u.setEmail(rs.getString("email"));
                u.setAtivo(rs.getBoolean("ativo"));
                list.add(u);
            }
        } catch(Exception ex){ ex.printStackTrace(); }
        return list;
    }
    public static User findById(int id) {
        try (Connection c = DB.getConnection()) {
            PreparedStatement ps = c.prepareStatement("SELECT id,nome,idade,tipo,email,ativo FROM usuario WHER
E id=?");
            ps.setInt(1,id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                User u = new User();
                u.setId(rs.getInt("id"));
                u.setNome(rs.getString("nome"));
                u.setIdade(rs.getInt("idade"));
                u.setTipo(rs.getString("tipo"));
```

```
                u.setEmail(rs.getString("email"));
                u.setAtivo(rs.getBoolean("ativo"));
                return u;
            }
        } catch(Exception ex){ ex.printStackTrace(); }
        return null;
    }
    public static void saveOrUpdate(User u, String cat1, String cat2, String cat3) {
        try (Connection c = DB.getConnection()) {
            if (u.getId()==0) {
                // create
                String salt = genSalt();
                String pwd = u.getSenhaPlain()!=null?u.getSenhaPlain():"usuario123";
                String hash = sha256(salt + pwd);
                PreparedStatement ps = c.prepareStatement("INSERT INTO usuario (nome,idade,tipo,email,se
sh,salt,ativo) VALUES (?,?,?,?,?,?,?)", Statement.RETURN_GENERATED_KEYS);
                ps.setString(1, u.getNome()); ps.setInt(2, u.getIdade()); ps.setString(3, u.getTipo()); ps.set
String(4, u.getEmail());
                ps.setString(5, hash); ps.setString(6, salt); ps.setBoolean(7, u.isAtivo());
                ps.executeUpdate();
                ResultSet keys = ps.getGeneratedKeys();
                int userId = 0;
                if (keys.next()) userId = keys.getInt(1);
                // insert interests (up to two based on non-null params)
                PreparedStatement pi = c.prepareStatement("INSERT INTO interesse_usuario (usuario_id,cate
) VALUES (?,?)");
```