ISPGAYA
instituto superior politécnico

Licenciatura em Engenharia Informática
Linguagens e Teoria da Computação
2ºAno

Pedro Miguel Fevereiro da Silva - 2024102255
Jadir Amador Neves de Resende - 2723

Sistema de Gestão de Eventos Académicos

Relatório de projeto orientado pelo Professor José Arnaud e apresentada à Escola Superior de Ciência e Tecnologia

dezembro de 2025

Índice

Índice de imagens

Introdução

O presente relatório descreve o desenvolvimento e implementação de um sistema de eventos, realizado como trabalho prático no âmbito da unidade curricular Linguagens e Teoria da Computação (LTC), integrada na licenciatura em engenharia informática.

O projeto teve como principal finalidade a implementação de um sistema de gestão de eventos académicos, destinado a apoiar instituições de ensino na organização e acompanhamento das suas atividades. Este trabalho integrou a consolidação de conhecimentos teóricos e práticos relacionados com estruturas de dados, validação, ordem de grandeza e análise de algoritmos. Paralelamente, permitiu aplicar os conhecimentos adquiridos ao longo do semestre, nomeadamente no âmbito da algoritmia, programação em Java e gestão de estruturas de dados complexas.

Em conclusão, este projeto permitiu aplicar de forma prática os conhecimentos adquiridos na unidade curricular, resultando num sistema funcional e adequado à gestão de eventos académicos. O trabalho desenvolvido contribuiu para reforçar competências essenciais na programação e na estruturação de soluções em Java. Nos pontos seguintes serão apresentadas as etapas do desenvolvimento e as principais decisões tomadas ao longo do processo.

Objetivos

Na nossa opinião, o principal objetivo deste projeto foi desenvolver um software de pequena a média dimensão, aplicando os conhecimentos adquiridos nas aulas e reconhecendo a importância da pesquisa e investigação autónoma para ultrapassar as dificuldades encontradas ao longo do processo.

No decorrer do trabalho, foi possível alcançar os seguintes objetivos:

- Compreender e consolidar conhecimentos teóricos e práticos;
- Desenvolver competências na resolução de problemas complexos;
- Aplicar estruturas de dados e algoritmos adequados ao contexto do sistema;
- Implementar um sistema integrado de gestão de eventos;
- Garantir a integridade e validação dos dados tratados;
- Criar relatórios analíticos que apoiem a gestão e tomada de decisão.

Descrição do trabalho

O Sistema de Gestão de Eventos Académicos foi desenvolvido com o objetivo de disponibilizar uma solução capaz de criar, supervisionar e analisar eventos, participantes, recursos e sessões no contexto académico. O sistema permite operações completas de criar, ler, atualizar e apagar registos, garantindo simultaneamente a integridade dos dados e a manutenção de um histórico consistente.

A solução implementada cumpre os requisitos definidos no enunciado, contemplando a gestão de Eventos, Participantes, Recursos e Sessões, cada um com um conjunto de atributos específicos e com relacionamentos estruturados de forma clara.

Arquitetura do Sistema

O software foi estruturado em quatro camadas de forma a simplificar, a manutenção, os testes e um possível crescimento do software.

- **Camada de Modelos de Domínio (classes)**

  São as classes Java, contem as definições das entidades principais do sistema, construtores e métodos de acesso. Nesta camada incluímos também os diferentes estados, tipos e categorias das entidades.

- **Camada de Acesso a dados (‘modules.connection’)**

  Esta camada é responsável pela execução das operações Criar, Ler, Editar e eliminar. Utiliza ‘SQLite’ como sistema de gestão de bases de dados relacional, proporcionando armazenamento e consultas estruturadas através de SQL.

- **Camada de Apresentação (‘modules.windows’ e ‘modules.ui’)**

  Desenvolve a interface gráfica utilizando ‘JavaFX’, oferece ao utilizador final janelas intuitivas de navegação. A navegação é centralizada na classe ‘Window’ que gere a janela principal.

- **Camadas Auxiliares**

  Fornecem as funcionalidades complementares e componentes visuais reutilizáveis.

Entidades e Atributos

Entidade Evento

A entidade Evento representa as atividades académicas a organizar. Implementa os atributos obrigatórios:

- **ID do Evento:** Identificador único e sequencial, gerado automaticamente pelo sistema.
- **Nome:** Designação descritiva do evento.
- **Local**: Localização física onde ocorrerá o evento.
- **Data de Início e Data de Término:** Delimitação temporal do evento.
- **Estado:** Representado por enumeração, com os valores "Planeado", "Em Progresso" ou "Concluído".
- **Ativo:** Campo booleano que permite a inativação lógica, preservando o histórico de dados.
- **Sessões e Recursos Associados:** Referências às entidades relacionadas.

Entidade Participante

A entidade Participante regista os intervenientes nos eventos académicos:

- **ID do Participante:** Identificador único e sequencial.
- **Nome:** Identificação do participante.
- **Tipo:** Categorização através de enumeração, distinguindo entre "Palestrante", "Moderador" e "Participante Geral".
- **Contacto:** Informações de comunicação (telefone e email), com validação de formato de email.
- **Data de Inscrição:** Registo temporal da inscrição do participante.
- **Ativo:** Campo para inativação lógica.

Entidade Recurso

A entidade Recurso descreve os meios materiais e técnicos necessários para realização dos eventos:

- **ID do Recurso:** Identificador único e sequencial.
- **Nome:** Designação do recurso.
- **Categoria:** Enumeração que classifica o recurso (ex.: "Equipamento", "Espaço", "Suporte Técnico").
- **Quantidade Disponível:** Número de unidades em stock.
- **Custo Unitário**: Valor monetário por unidade.
- **Ativo:** Campo para inativação lógica.

Entidade Sessão

A entidade Sessão representa as divisões temáticas ou cronológicas dentro de um evento:

- **ID da Sessão:** Identificador único e sequencial.
- **ID do Evento:** Referência à entidade Evento (chave estrangeira).
- **Descrição:** Conteúdo temático ou objetivos da sessão.
- **Data de Início e Data de Término Prevista:** Agendamento temporal.
- **Estado:** Enumeração com valores "Planeada", "Em Progresso" ou "Concluída".
- **ID do Moderador:** Referência ao Participante responsável pela moderação.
- **Ativo:** Campo para inativação lógica.

Metodologias Aplicadas

O desenvolvimento deste sistema foi concebido por 5 etapas, incrementando a dificuldade à medida que avançávamos:

1º - Análise e Desenho

Análise detalhada do enunciado, identificação das entidades, seus atributos e relacionamentos. Desenhamos o diagrama de relações entre entidades garantindo a integridade

referencial.

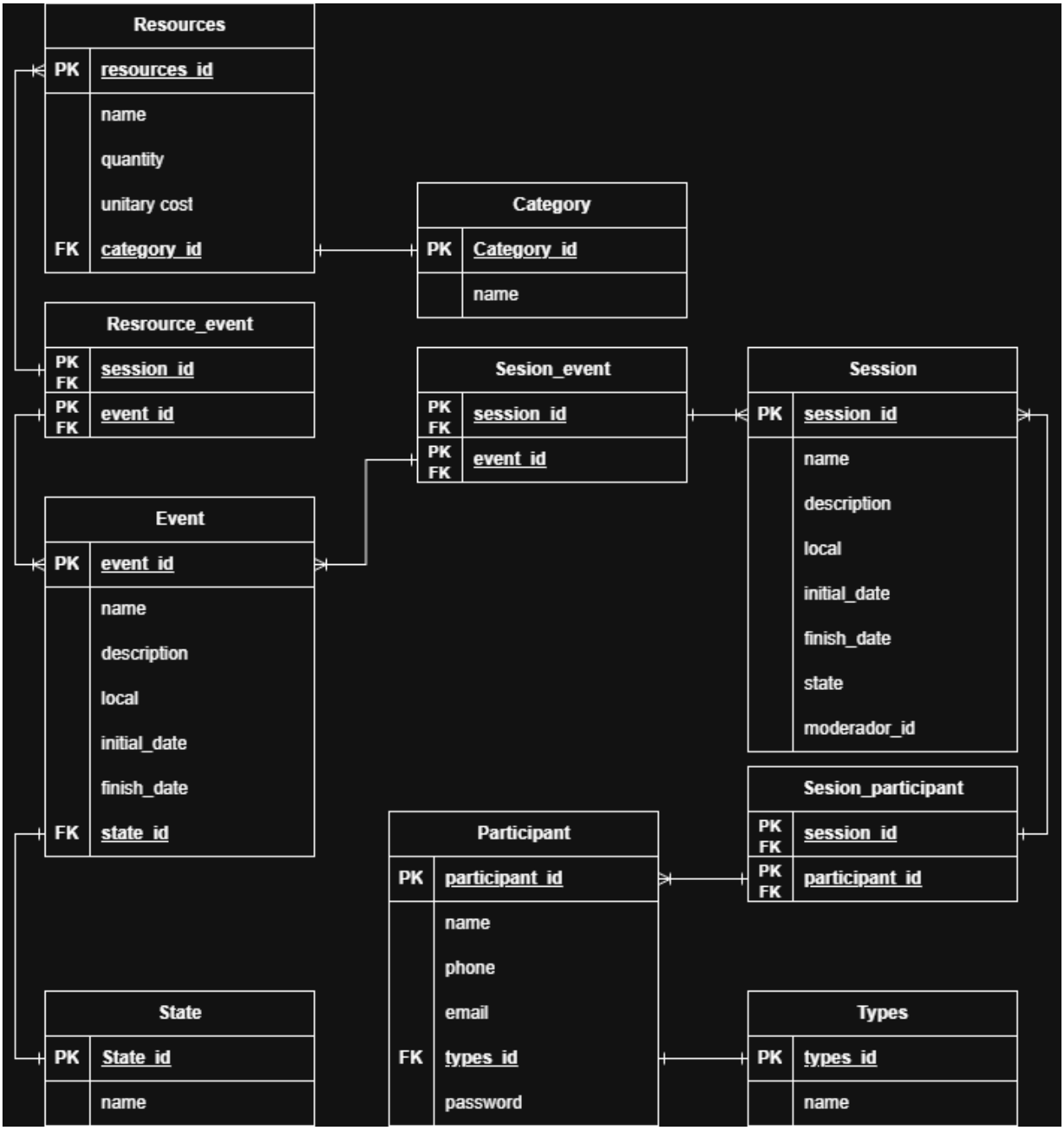

Figura 1 - Diagrama de relações entre entidades

2º - Implementação da Camada de Modelos

Criaram-se as classes Java correspondentes a cada entidade, foram usados métodos getters e setters, assim como métodos auxiliares para validação e formatação.

```java
public class Participant {  Pedro Fevereiro +1
    private final String id;  3 usages
    private String name;  5 usages
    private String email;  4 usages
    private String phone;  4 usages
    private Types type;  3 usages
    private String password;  2 usages
    private String gender;  4 usages
    private String taxNumber;  4 usages
    private Date birthdate;  4 usages
    private String photo;  4 usages

    public Participant(ResultSet rs) throws SQLException {  Pedro Fevereiro +1
        this.id = rs.getString( columnLabel: "participant_id");
        this.name = rs.getString( columnLabel: "name");
        this.email = rs.getString( columnLabel: "email");
        this.phone = rs.getString( columnLabel: "phone");
        this.password = rs.getString( columnLabel: "password");
        this.gender = rs.getString( columnLabel: "gender");
        this.taxNumber = rs.getString( columnLabel: "tax_number");
        this.birthdate = parseBirthdate(rs.getString( columnLabel: "birthdate"));
        this.photo = rs.getString( columnLabel: "photo");

        this.type = new Types(
                rs.getInt( columnLabel: "types_id"),
                rs.getString( columnLabel: "types_name")
        );
    }

    // Construtor simples para criar um utilizador em memoria (ex.: sessao admin direta).
    public Participant(String id, String name, String email, String phone, Types type) {  Pedro Fevereiro
        this.id = id;
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.type = type;
        this.password = null;
        this.gender = null;
        this.taxNumber = null;
        this.birthdate = null;
        this.photo = null;
    }

    // Getters
    public String getId() { return id; }  fevereiro2
    public String getName() { return name; }  fevereiro2
    public String getEmail() { return email; }  fevereiro2
    public String getPhone() { return phone; }  1 usage  fevereiro2
    public String getGender() { return gender; }  no usages  Pedro Fevereiro
    public String getTaxNumber() { return taxNumber; }  no usages  Pedro Fevereiro
    public Date getBirthdate() { return birthdate; }  no usages  Pedro Fevereiro
    public String getPhoto() { return photo; }  no usages  Pedro Fevereiro
    public Types getType() { return type; }  18 usages  Pedro Fevereiro

    // Setters para AccountScreens atualizar:
    public void setName(String name) { this.name = name; }  Pedro Fevereiro
    public void setEmail(String email) { this.email = email; }  Pedro Fevereiro
    public void setPhone(String phone) { this.phone = phone; }  no usages  Pedro Fevereiro
    public void setGender(String gender) { this.gender = gender; }  1 usage  Pedro Fevereiro
    public void setTaxNumber(String tax) { this.taxNumber = tax; }  1 usage  Pedro Fevereiro
    public void setBirthdate(Date birthdate) { this.birthdate = birthdate; }  no usages  Pedro Fevereiro
    public void setPhoto(String photo) { this.photo = photo; }  no usages  Pedro Fevereiro

    private Date parseBirthdate(String value) {  1 usage  Pedro Fevereiro
        if (value == null || value.isBlank()) return null;
        try {
            long epoch = Long.parseLong(value.trim());
            return new Date(epoch);
        } catch (NumberFormatException ignored) {
            try {
                LocalDate d = LocalDate.parse(value.trim());
                return Date.valueOf(d);
            } catch (DateTimeParseException ex) {
                return null;
            }
        }
    }
```

Figura 2 - Exemplo da classe participante

3º - Desenvolvimento da Interface Gráfica

Implementámos a camada de apresentação, onde usámos Java FX para criar o ambiente gráfico para o utilizador conseguir navegar e gerir cada entidade.

4º - Implementação de Funcionalidades Avançadas

Neste ponto virámos o foco aos relatórios que vão permitir uma análise de toda a gestão de eventos, recursos e participantes.

5º - Testes e Refinamento

Realizaram-se testes funcionais em cada módulo e validação de dados.

Funcionalidades Implementadas

Gestão de Eventos

- Criação de eventos com ID sequencial, atribuição de nome, local e datas de início/término.
- Definição de estado do evento (planeado, em progresso, concluído).
- Associação de sessões e recursos a cada evento.
- Edição de informações de eventos.
- Inativação lógica com preservação de dados.

Gestão de Sessões

- Criação de sessões associadas a eventos específicos.
- Definição de descrição, datas de início/término previstas e estado.
- Associação de um moderador (participante) a cada sessão.
- Edição e inativação de sessões.
- Listagem de sessões por evento.

Gestão de Participantes

- Criação de novos registos de participantes com validação de dados obrigatórios.
- Listagem de participantes com filtros por tipo (palestrante, moderador, participante geral).
- Edição de informações de participantes existentes.
- Inativação lógica de participantes, preservando histórico.
- Validação de formato de email e contacto telefónico.

Gestão de Recursos

- Registo de recursos com categoria (equipamento, espaço, suporte técnico, etc.).
- Definição de quantidade disponível e custo unitário.
- Edição de informações e quantidades.
- Inativação lógica com histórico preservado.

Relatórios Administrativos

Relatórios de Eventos

- Contagem do número total de eventos ativos.
- Distribuição de eventos por estado (planeado, em progresso, concluído).
- Análise de desvios de cronograma, identificando eventos atrasados em relação às datas previstas.

Relatórios de Sessões

- Contagem do número total de sessões por evento.
- Análise do tempo médio para conclusão de sessões.
- Identificação de atrasos, mostrando sessões que excedem a data de término prevista.
- Listagem de eventos a decorrer nos próximos 15 dias (calendário).

Relatórios de Participantes

- Estatísticas sobre o número total de participantes.
- Distribuição de participantes por tipo (palestrante, moderador, participante geral).
- Análise de participação, identificando os participantes mais envolvidos com base no número de sessões concluídas ou palestras realizadas.

Relatórios de Recursos

- Listagem do total de recursos disponíveis em stock.
- Distribuição de recursos por categoria.
- Análise do uso de recursos por evento, identificando recursos mais requisitados.
- Identificação de recursos que necessitam reposição (quantidade abaixo de limiar configurável).

Desafios e Soluções

Integração de 'JavaFX'

Desafio: Integrar o' JavaFX', atualização dinâmica das operações realizadas e complexidade de navegação entre janelas.
Solução: Tentamos uma pesquisa para implementar, uma arquitetura de controladores centralizada com a classe 'Window' para gerir a janela principal, cada janela comunica unicamente com a sua classe. Recorremos a padrões 'callback' que fazem recarregar os dados sempre que são criados, eliminados e editados assim garantimos que a informação é atualizada.

Validação de Dados e Tratamento de Erros

Desafio: Garantir que todos os dados de entrada do utilizador sejam validados, evitando dados inválidos, inconsistentes ou mal formatados.
Solução: Implementámos camadas de validação, a camada interface gráfica realiza validações locais, formato do e-mail, valores numéricos e comprimento de strings. Na camada de modelos implementámos validações como a data de término ser posterior à data de início e a data de início ser posterior a data de marcação do evento.

Relatórios Complexos e Análise de Dados

Desafio: Implementar relatórios que unam dados de múltiplas tabelas, calculem métricas e apresentam resultados de forma clara, para os mesmos serem estruturados.
Solução: Desenvolveram-se classes especializadas de relatório que ligam a lógica de consulta e cálculo. Utilizaram-se consultas SQL para relacionar dados entre tabelas. Os resultados são organizados em estruturas de dados apropriadas (listas, mapas e matrizes) e apresentados em tabelas ou gráficos simples na interface, facilitando compreensão visual.

Reflexão sobre o Processo de Desenvolvimento

O desenvolvimento deste projeto prático proporcionou uma consolidação profunda dos conhecimentos teóricos adquiridos nas aulas de Linguagens e Teoria da Computação. A aplicação de conceitos como algoritmos, estruturas de dados (vetores e matrizes), programação orientada a objetos e manipulação de coleções revelou-se essencial para a resolução eficiente de problemas práticos.

A separação de responsabilidades por meio da arquitetura em camadas evidenciou a importância de um desenho sólido desde as fases iniciais do desenvolvimento.

Um código bem organizado facilita significativamente a identificação e correção de erros, a implementação de novas funcionalidades e a manutenção futura do sistema. Ficou particularmente evidente a relevância da validação de dados e do tratamento robusto de erros, especialmente em cenários com intensa interação com o utilizador final e persistência em bases de dados relacionais. A ausência de validação adequada pode conduzir rapidamente a inconsistências, corrupção de dados e degradação da experiência do utilizador.

Propostas de Melhoria

O Sistema cumpre os requisitos principais, mas é possível sempre realizar melhorias e permitir a própria evolução, segue algumas propostas:

- **Interface Gráfica**

  Melhorar a interface gráfica, paginação em listagem para melhorar a performance e usabilidade. Implementar gráficos que permitam uma leitura rápida.

- **Relatórios**

  Adicionar a funcionalidade de exportação de relatórios em vários formatos.

- **Notificação**

  Criar um sistema de reminders sobre eventos próximos, melhorando a comunicação e coordenação.

Conclusão

O presente trabalho prático constituiu um exercício integrador abrangente dos conhecimentos adquiridos na unidade curricular de Linguagens e Teoria da Computação. O desenvolvimento do Sistema de Gestão de Eventos Académicos exigiu a aplicação prática de algoritmos, estruturas de dados, programação orientada a objetos, gestão de bases de dados e criação de interfaces gráficas.

As soluções implementadas evidenciam uma compreensão sólida dos princípios da unidade curricular, incluindo arquitetura em camadas, separação de responsabilidades e validação rigorosa de dados. Os desafios enfrentados ao longo do desenvolvimento proporcionaram uma aprendizagem significativa sobre a resolução de problemas complexos e a importância de um planeamento cuidadoso desde as fases iniciais.

Considera-se que os objetivos propostos foram obtidos, o trabalho apresentado oferece uma base consistente para futuras extensões e melhorias.

O conhecimento e competências adquiridas são um alicerce fundamental para disciplinas posteriores e para a prática profissional.

Referências

- **Oracle Corporation. (2024).** ***The Java Language Specification***
  https://docs.oracle.com/javase/specs/

- **SQLite Consortium. (2024).** ***SQLite Documentation***
  https://www.sqlite.org/docs.html

- **Oracle Corporation. (2024).** ***JavaFX Documentation***
  https://openjfx.io/javadoc/latest/