Gestão despesa pessoal

Engenharia Informática | 2025/2026

Laboratório de programação | Paula Morais

Gonçalo Coutinho 44193 | Francisco Jacinto 48917

UNIVERSIDADE PORTUCALENSE
DEPARTAMENTO
CIÊNCIA E TECNOLOGIA

ÍNDICE

.

Introdução

O presente relatório descreve o desenvolvimento de um Sistema de Gestão de Despesas Pessoais, desenvolvido no âmbito da unidade curricular, com o objetivo de aplicar conhecimentos de engenharia de software, arquitetura de sistemas, programação orientada a objetos e desenvolvimento de aplicações cliente-servidor. O sistema permite que cada utilizador registe e gira as suas despesas de forma individual e segura, possibilitando a criação de categorias personalizadas, o registo detalhado de despesas, a aplicação de filtros e a visualização de estatísticas financeiras relevantes, tais como totais mensais, totais por categoria e indicadores de resumo num Dashboard.

A solução foi implementada seguindo uma arquitetura cliente-servidor, composta por:

- Um backend RESTful desenvolvido em Spring Boot, responsável pela lógica de negócio, persistência de dados e segurança.
- Uma aplicação cliente desktop desenvolvida em JavaFX, responsável pela interação com o utilizador.

Esta separação permitiu uma melhor organização do código, maior escalabilidade e facilidade de manutenção do sistema.

1. Especificação dos requisitos do Sistema:

Os requisitos funcionais foram definidos com base em **User Stories**, garantindo que o sistema responde diretamente às necessidades do utilizador final.

1.1 Requisitos Funcionais (RF)

RF1 — Registo de Utilizadores
O sistema deve permitir que um utilizador crie uma conta fornecendo nome, email e password.
User Stories relacionadas:US01

RF2 — Autenticação de Utilizadores
O sistema deve permitir que utilizadores registados iniciem sessão introduzindo email e password.
User Stories relacionadas:US02

RF3 — Atualização de Perfil
O sistema deve permitir que o utilizador atualize dados do seu perfil, incluindo a password (mediante confirmação da password atual).
User Stories relacionadas:US03

RF4 — Criação de Categorias
O sistema deve permitir a criação de categorias personalizadas, garantindo que o nome é único para cada utilizador.
User Stories relacionadas:US04

RF5 — Edição de Categorias
O sistema deve permitir a edição do nome de categorias existentes.
User Stories relacionadas:US05

RF6 — Remoção de Categorias
O sistema deve permitir a remoção de categorias, desde que não tenham despesas associadas.
User Stories relacionadas:US06

RF7 — Registo de Despesas

O sistema deve permitir a criação de despesas contendo valor, data, categoria, descrição e método de pagamento.

User Stories relacionadas:US07

RF8 — Edição de Despesas

O sistema deve permitir a edição de despesas existentes, aplicando as mesmas validações da criação.

User Stories relacionadas:US08

RF9 — Remoção de Despesas

O sistema deve permitir a eliminação de despesas registadas pelo utilizador.

User Stories relacionadas:US09

RF10 — Listagem de Despesas

O sistema deve apresentar ao utilizador uma listagem de todas as suas despesas, ordenadas por data decrescente.

User Stories relacionadas:US10

RF11 — Filtro por Intervalo de Datas

O sistema deve permitir filtrar despesas por intervalo de datas introduzido pelo utilizador.

User Stories relacionadas:US11

RF12 — Filtro por Categoria

O sistema deve permitir filtrar despesas por categoria existente.

User Stories relacionadas:US12

RF13 — Filtro por Valores

O sistema deve permitir filtrar despesas por valor mínimo e/ou máximo definido pelo utilizador.

User Stories relacionadas:US13

RF14 — Cálculo do Total Mensal

O sistema deve calcular e apresentar o total de despesas de um mês selecionado.

User Stories relacionadas:US14

RF15 — Cálculo de Totais por Categoria

O sistema deve calcular e apresentar o total de despesas agregadas por categoria.

User Stories relacionadas:US15

RF16 — Dashboard de Estatísticas

O sistema deve apresentar indicadores (total período, média, máximo, top categoria) com base nas despesas do utilizador.

User Stories relacionadas:US16

RF17— Gestão de Despesas Recorrentes

O sistema deve permitir criar recorrências e gerar despesas automaticamente segundo a periodicidade.

User Stories relacionadas:US17

1.2 Requisitos Não Funcionais (RNF)

RNF1 — Segurança das Passwords
As passwords devem ser armazenadas de forma cifrada (hash seguro, como BCrypt), nunca em texto simples.

RNF2 — API REST com Spring Boot
O sistema deve disponibilizar um backend RESTful desenvolvido em Spring Boot.

RNF3 — Persistência com Hibernate/JPA
A comunicação com a base de dados deve ser implementada com Hibernate/JPA, recorrendo a mapeamento objeto-relacional.

RNF4 — Interface do Cliente em JavaFX
A aplicação cliente deve ser desenvolvida em JavaFX, garantindo uma interface gráfica desktop funcional.

RNF5 — Gestão de Dependências com Maven
O projeto deve utilizar Maven para gestão de dependências e compilação.

RNF6 — Usabilidade
A interface deve ser clara, intuitiva e apresentar mensagens de erro compreensíveis e informativas.

RNF7 — Desempenho
As operações principais (listar despesas, aplicar filtros e calcular totais) devem apresentar resposta fluida, sem atrasos percetíveis para o utilizador.

RNF8 — Fiabilidade
O sistema não deve falhar silenciosamente e deve apresentar mensagens de erro adequadas em caso de exceções ou problemas inesperados.

1.3 Regras de Negócio (RN)

RN1 — Propriedade dos dados

- Um utilizador só pode ver/editar/eliminar as suas próprias despesas e categorias.

RN2 — Valor da despesa

- O valor de uma despesa tem de ser maior que 0.

RN3 — Data válida

- A data de uma despesa não pode ser vazia.
- (Opcional) A data não deve ser no futuro.

RN4 — Categoria obrigatória

- Toda a despesa deve estar associada a uma categoria existente.

RN5 — Categorias únicas

- O nome de categoria deve ser único por utilizador (ex.: não pode haver 2 "Alimentação" na mesma conta).

RN6 — Eliminar categoria

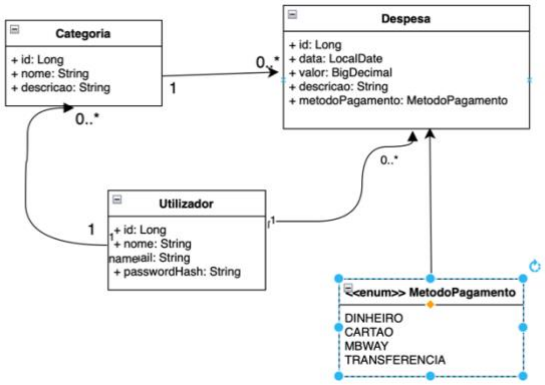
- Não é permitido eliminar uma categoria se existirem despesas associadas a essa categoria.
    - (Alternativa possível: ao eliminar, mover despesas para "Outros". Escolhe uma e mantém no relatório.)

RN7 — Método de pagamento

- O método de pagamento deve ser um dos valores aceites (ex.: DINHEIRO, CARTAO, MBWAY, TRANSFERENCIA, CRIPTOMOEDAS).

RN8 — Edição controlada

- Ao editar uma despesa, os campos valor/data/categoria/método/descrição devem continuar válidos (RN2, RN3, RN4, RN7).



1.4.1 Entidade Utilizador

A entidade **Utilizador** representa cada utilizador registado no sistema e é responsável pela posse das categorias e despesas.

Atributos:

- id : Long – identificador único do utilizador.
- nome : String – nome do utilizador.
- email : String – endereço de email único no sistema.
- passwordHash : String – password do utilizador armazenada de forma segura (hash).

Relações:

- Um utilizador pode estar associado a **zero ou mais categorias** (1 → 0..*).
- Um utilizador pode estar associado a **zero ou mais despesas** (1 → 0..*).

1.4.2 Entidade Categoria

A entidade **Categoria** permite ao utilizador organizar as suas despesas de forma personalizada.

Atributos:

- id : Long – identificador único da categoria.
- nome : String – nome da categoria.
- descricao : String – descrição opcional da categoria.

Relações:

- Cada categoria pertence a **um único utilizador** (1).

- Uma categoria pode estar associada a **zero ou mais despesas** (1 → 0..*).

Restrições:

- O nome da categoria deve ser **único por utilizador**.
- Uma categoria **não pode ser removida** se tiver despesas associadas.

1.4.3 Entidade Despesa

A entidade **Despesa** representa cada registo individual de gasto efetuado pelo utilizador.

Atributos:

- id : Long – identificador único da despesa.
- data : LocalDate – data da despesa.
- valor : BigDecimal – valor monetário da despesa.
- descricao : String – descrição da despesa.
- metodoPagamento : MetodoPagamento – método de pagamento utilizado.

Relações:

- Cada despesa pertence a **um único utilizador** (1).
- Cada despesa está associada a **uma única categoria** (1).

Restrições:

- O valor da despesa deve ser **superior a zero**.
- A data da despesa não pode ser futura.
- A categoria associada deve existir.

1.4.4 Enumeração MetodoPagamento

A enumeração **MetodoPagamento** define os métodos de pagamento suportados pelo sistema, garantindo consistência nos dados.

Valores possíveis:

- DINHEIRO
- CARTAO
- MBWAY
- TRANSFERENCIA

O modelo de dados apresentado assegura que:

- Cada utilizador apenas tem acesso às suas próprias categorias e despesas.
- A integridade referencial é mantida entre utilizadores, categorias e despesas.
- As regras de negócio definidas nos requisitos funcionais são respeitadas ao nível da estrutura de dados.
- O modelo é facilmente mapeável para uma base de dados relacional utilizando **Hibernate/JPA**.

Conclusão

O desenvolvimento deste projeto de **Gestão de Despesas Pessoais** permitiu consolidar a implementação de uma solução **cliente-servidor**, com um **backend REST em Spring Boot** (persistência com **Hibernate/JPA**) e um **cliente JavaFX**para interação com o utilizador. Ao longo do processo, foi possível transformar os requisitos do enunciado em funcionalidades concretas, garantindo uma ligação consistente entre modelo de dados, API e interface gráfica.

O que correu bem

- **Estruturação por camadas no backend** (Controller/Service/Repository), o que facilitou a manutenção, a leitura do código e a evolução do projeto.
- **Integração da API com o cliente JavaFX**, permitindo validar fluxos completos (ex.: registo/login, CRUD de categorias e despesas, filtros e estatísticas).
- **Aplicação de regras de negócio e validações** (ex.: unicidade, restrições de remoção, valores e datas válidas) e inclusão de tratamento de erros, reduzindo comportamentos inconsistentes e melhorando a experiência do utilizador.

O que não correu bem

- Gestão e alinhamento das validações entre cliente e servidor**: em alguns casos, a validação duplicada ou incompleta aumentou o esforço de manutenção e dificultou a uniformização das mensagens de erro.**
- Configuração e ambiente **(BD, propriedades e execução): ajustes de configuração exigiram testes adicionais e aumentaram o tempo de estabilização do sistema.**
- Garantia de qualidade contínua**: a ausência de testes automatizados (unitários/integração) tornou a deteção de regressões mais dependente de testes manuais, especialmente em funcionalidades mais sensíveis como filtros e recorrências.**

Sugestões de melhoria

- **Adicionar testes automatizados** (JUnit + testes de integração aos endpoints) para validar regras críticas e prevenir regressões.
- **Melhorar a configuração e segurança**
- **Evoluir a robustez e usabilidade**

Anexos

Anexo III do Regulamento Pedagógico

Declaração de Autoria

Para Trabalho de Grupo

Nomes: Gonçalo Coutinho e Francisco Jacinto, estudantes do curso de Licenciatura / Mestrado / Doutoramento Engenharia Informática, declaram que o Trabalho/ Projeto / Dissertação / Relatório apresentado para avaliação é da sua autoria e cumpre as normas de integridade académica.

Assim comprometem-se a:

- Esclarecer explicitamente se partes do trabalho foram já apresentadas para avaliação de outras UCs ou provas de grau na Universidade Portucalense ou noutras instituições;

- Identificar corretamente as fontes que utilizaram, de forma a que possam ser consultadas e atestada a autenticidade do trabalho que apresentam;

- Assumir, sob compromisso de honra, a responsabilidade da autoria integral do trabalho, não tendo contratado serviços de terceiros para a sua realização;

- Indicar a supervisão que receberam para elaboração do trabalho;

- Reconhecer como fraudulentas práticas que correspondem a formas de plágio, cópia servil, omissão ou citação deficiente de fontes, percebendo que tais práticas infringem direitos de autoria e são contrárias à integridade académica;

- Submeter, quando solicitado, à consideração do(s) docente(s), relatórios que tenham sido emitidos por equipamento especializado na deteção de plágio.

Data dd de mm de aaaa

Assinaturas Gonçalo Coutinho
Francisco Jacinto

UNIVERSIDADE
PORTUCALENSE

upt.pt