Universidade da Beira Interior

Departamento de Informática

Licenciatura em Engenharia Informática

UC: Programação Orientada a Objetos

Relatório do Trabalho Prático – Grupo 17

A MINHA DIETA

André • Carlos • Henrique • João

Covilhã, 29 de dezembro de 2025

Agradecimentos

A equipa agradece ao corpo docente da Unidade Curricular de *Programação Orientada a Objetos* pela orientação científica e pelo acompanhamento contínuo ao longo do desenvolvimento do projeto. Agradece-se igualmente o contributo de colegas através de *feedback* técnico e validação informal de funcionalidades.

Resumo

O presente relatório descreve o desenvolvimento do projeto *A Minha Dieta*, uma aplicação *desktop* em Java (JavaFX) destinada ao registo e acompanhamento de alimentação, hidratação e atividade física, integrando cálculo automático de metas e indicadores. A solução foi concebida segundo o padrão **Model–View–Controller (MVC)** (Model–View–Controller), promovendo separação de responsabilidades, manutenibilidade e extensibilidade. Entre as funcionalidades implementadas destacam-se: gestão de perfis, cálculo de **Índice de Massa Corporal (IMC)** e **Taxa Metabólica Basal (TMB)**, diário alimentar com registos de macronutrientes, monitorização de água com progresso face à meta diária, registo de exercício, histórico consultável e exportação de relatórios em **Portable Document Format (PDF)**. O relatório apresenta decisões de desenho e exemplos representativos de implementação (Java, FXML, CSS, persistência e gráficos), bem como uma estratégia de validação por cenários de utilização.

Palavras-chave: Programação Orientada a Objetos (POO), Java, JavaFX, MVC, Persistência, Gráficos, PDF.

Abstract

This report presents *A Minha Dieta*, a Java (JavaFX) desktop application designed to support diet tracking, hydration monitoring, and exercise logging, including automatic computation of personalized goals and indicators. The solution follows the **MVC** architecture, focusing on maintainability and extensibility. Key features include profile management, **IMC** (BMI) and **TMB** (BMR) computation, daily macronutrient logging, hydration progress tracking, exercise monitoring, searchable history, and **PDF** export. We also provide representative code excerpts (Java, FXML, CSS, persistence, and charts) and a validation strategy based on usage scenarios.

Keywords: OOP, Java, JavaFX, MVC, Persistence, Charts, PDF.

Lista de Acrónimos

POO Programação Orientada a Objetos

MVC Model–View–Controller

UI Interface do Utilizador

FXML Ficheiro declarativo de interface JavaFX

CSS Cascading Style Sheets

TMB Taxa Metabólica Basal

IMC Índice de Massa Corporal

PDF Portable Document Format

Conteúdo

Listings

1 Introdução

A aplicação *A Minha Dieta* foi desenvolvida no âmbito da Unidade Curricular de POO com o objetivo de consolidar conceitos fundamentais de modelação orientada a objetos e boas práticas de desenho de software. O domínio do problema envolve registo de hábitos (alimentação, água, exercício e peso), cálculo de metas/indicadores e visualização de progresso através de um *dashboard* interativo.

1.1 Objetivos

- Modelar o domínio com classes coesas e responsabilidades bem definidas;
- Implementar uma interface JavaFX consistente e intuitiva;
- Calcular metas e indicadores (IMC, TMB, metas de água e macronutrientes);
- Agregar registos diários e históricos, suportando visualização por gráficos;
- Persistir o estado da aplicação e exportar relatórios em PDF.

2 Requisitos e Execução

2.1 Requisitos

- Java (JDK 17 ou superior);
- Maven 3.6+;
- Dependências JavaFX geridas via Maven.

2.2 Execução (Maven)

```
git clone https://github.com/MooniePT/AminhaDieta.git
cd AminhaDieta
mvn clean compile
mvn javafx:run
```

Listing 1: Compilar e executar o projeto (exemplo)

3 Arquitetura do Sistema (MVC)

3.1 Visão geral

A solução adota o padrão MVC, separando o domínio (*Model*) da interface (*View*) e do controlo de fluxo (*Controller*). Esta decisão reduz acoplamento, melhora testabilidade do domínio e facilita manutenção incremental.

```
src/main/java/
  app/
    model/        (UserProfile, AppState, MealEntry, WaterEntry,
   ExerciseEntry, ...)
    controller/   (DashboardController, MealsController,
   HydrationController, ...)
    service/      (DataStore, PdfExporter, helpers)
src/main/resources/
  fxml/           (dashboard.fxml, meals.fxml, ...)
  css/            (theme.css)
```

Listing 2: Organização típica por pacotes (representativa)

4 Modelação do Domínio e Regras de Negócio

4.1 Entidades centrais

O domínio é centrado no perfil do utilizador, que agrega registos e fornece operações de cálculo e agregação (metas, totais diários e histórico).

4.2 Encapsulamento, invariantes e validação

A consistência é assegurada por validações (entradas inválidas são rejeitadas) e por encapsulamento do estado. Sempre que são expostas coleções internas, devolvem-se cópias defensivas, evitando fuga de representação.

```
public UserProfile(String name, double weightKg, int heightCm, int age)
   {
    if (name == null || name.isBlank()) throw new
   IllegalArgumentException("Nome inv lido.");
    if (weightKg <= 0) throw new IllegalArgumentException("Peso
   inv lido.");
    if (heightCm <= 0) throw new IllegalArgumentException("Altura
   inv lida.");
```

```
5       if (age <= 0) throw new IllegalArgumentException("Idade inv lida.")
        ;
6       this.name = name.trim();
7       this.weightKg = weightKg;
8       this.heightCm = heightCm;
9       this.age = age;
10  }
```

Listing 3: Construtor com validação (representativo)

```
1   public List<MealEntry> getMeals() {
2       return new ArrayList<>(meals);
3   }
```

Listing 4: Cópias defensivas (representativo)

5 Dashboard e Visualização de Dados

5.1 Indicadores exibidos no Dashboard

O *dashboard* apresenta indicadores de acompanhamento diário:

- **Calorias do dia**: total consumido vs meta diária (barra de progresso);
- **Água do dia**: volume ingerido vs meta diária (barra de progresso);
- **IMC**: valor atual e classificação qualitativa;
- **Macronutrientes**: proteína, hidratos e gordura (consumo vs objetivo);
- **Gráficos**: *PieChart* (consumo diário), *BarChart* (atividade física) e gráfico de **peso** por data.

5.2 IMC e classificação

O IMC é calculado por:

$$IMC = \frac{peso(kg)}{altura(m)^2}$$

A Tabela 1 resume uma classificação típica utilizada em contexto académico.

Intervalo (IMC)	Classificação
< 18.5	Baixo peso
$[18.5, 24.9]$	Saudável
$[25.0, 29.9]$	Excesso de peso
≥ 30.0	Obesidade

Tabela 1: Classificação típica do IMC.

5.3 FXML do Dashboard (estrutura dos gráficos)

O excerto seguinte ilustra a declaração dos três gráficos principais e alguns nós associados (IDs usados pelo *Controller*).

```
<AnchorPane xmlns="http://javafx.com/javafx"
            xmlns:fx="http://javafx.com/fxml"
            fx:controller="app.controller.DashboardController">

  <PieChart fx:id="dailyPie" layoutX="40" layoutY="430" prefWidth="480"
   prefHeight="240"/>

  <BarChart fx:id="activityBar" layoutX="560" layoutY="430" prefWidth="
   520" prefHeight="240">
    <xAxis>
      <CategoryAxis label="Dia"/>
    </xAxis>
    <yAxis>
      <NumberAxis label="Kcal"/>
    </yAxis>
  </BarChart>

  <ScatterChart fx:id="weightChart" layoutX="1100" layoutY="430"
   prefWidth="360" prefHeight="240">
    <xAxis>
      <CategoryAxis label="Data"/>
    </xAxis>
    <yAxis>
      <NumberAxis label="Kg"/>
    </yAxis>
  </ScatterChart>

</AnchorPane>
```

Listing 5: dashboard.fxml (excerto representativo): PieChart, BarChart e Peso

5.4 Controller: povoamento dos gráficos (lógica central)

O *Controller* agrega dados do *Model* e atualiza os gráficos. O ciclo típico é: evento do utilizador → atualização do modelo → `refresh()` do dashboard.

```
public class DashboardController {

    @FXML private PieChart dailyPie;
    @FXML private BarChart<String, Number> activityBar;
    @FXML private ScatterChart<String, Number> weightChart;

    private AppState state;

    public void setState(AppState state) {
        this.state = state;
        refresh();
    }

    private void refresh() {
        UserProfile p = state.getActiveProfile();
        LocalDate today = LocalDate.now();

        updateDailyPie(p, today);
        updateActivityBar(p, today);
        updateWeightChart(p);
    }

    private void updateDailyPie(UserProfile p, LocalDate day) {
        MacroTotals t = p.totalsFor(day);

        dailyPie.getData().setAll(
            new PieChart.Data(" gua  (" + p.totalWaterTodayMl() + "ml)",
    p.totalWaterTodayMl()),
            new PieChart.Data("Prote na (" + (int)t.getProteinG() + "g)
   ", t.getProteinG()),
            new PieChart.Data("Hidratos (" + (int)t.getCarbsG() + "g)",
   t.getCarbsG()),
            new PieChart.Data("Gordura (" + (int)t.getFatG() + "g)", t.
   getFatG())
        );
    }

    private void updateActivityBar(UserProfile p, LocalDate today) {
        XYChart.Series<String, Number> s = new XYChart.Series<>();
        s.setName("Kcal queimadas");

        for (int i = 6; i >= 0; i--) {
```

```
            LocalDate d = today.minusDays(i);
            int burned = p.totalBurnedCaloriesOn(d);
            s.getData().add(new XYChart.Data<>(d.getDayOfMonth() + "/" +
    d.getMonthValue(), burned));
        }

        activityBar.getData().setAll(s);
    }

    private void updateWeightChart(UserProfile p) {
        XYChart.Series<String, Number> s = new XYChart.Series<>();
        s.setName("Peso");

        for (WeightEntry w : p.getWeightHistory()) {
            String label = w.getDate().getDayOfMonth() + "/" + w.getDate
   ().getMonthValue();
            s.getData().add(new XYChart.Data<>(label, w.getWeightKg()));
        }

        weightChart.getData().setAll(s);
    }
}
```

Listing 6: DashboardController: refresh e update dos gráficos (representativo)

5.5 Agregações no Model (exemplos)

Para suportar os gráficos, o *Model* disponibiliza métodos de agregação diária.

```
public MacroTotals totalsFor(LocalDate day) {
    int kcal = 0;
    double p = 0, c = 0, f = 0;

    for (MealEntry m : meals) {
        if (m.getWhen().toLocalDate().equals(day)) {
            kcal += m.getCalories();
            p += m.getProteinG();
            c += m.getCarbsG();
            f += m.getFatG();
        }
    }
    return new MacroTotals(kcal, p, c, f);
}
```

Listing 7: Agregação de macros por dia (representativo)

```
public int totalBurnedCaloriesOn(LocalDate day) {
```

```
    int total = 0;
    for (ExerciseEntry e : exercises) {
        if (e.getWhen().toLocalDate().equals(day)) {
            total += e.getCaloriesBurned();
        }
    }
    return total;
}
```

Listing 8: Total de calorias queimadas por dia (representativo)

6 Persistência e Exportação PDF

6.1 Persistência por serialização

A persistência é assegurada por serialização do estado global da aplicação, permitindo restaurar perfis e histórico entre execuções.

```
public final class DataStore {
    private static final String FILE = "data/appstate.dat";

    public static void save(AppState state) throws IOException {
        Files.createDirectories(Paths.get("data"));
        try (ObjectOutputStream out =
             new ObjectOutputStream(new FileOutputStream(FILE))) {
            out.writeObject(state);
        }
    }

    public static AppState load() throws IOException,
   ClassNotFoundException {
        File f = new File(FILE);
        if (!f.exists()) return new AppState();
        try (ObjectInputStream in =
             new ObjectInputStream(new FileInputStream(FILE))) {
            return (AppState) in.readObject();
        }
    }
}
```

Listing 9: Guardar e carregar estado (representativo)

6.2 Exportação PDF (OpenPDF/LibrePDF)

```java
public final class PdfExporter {
    public static void exportDaily(UserProfile p, LocalDate day, String
    outFile) throws Exception {
        Document doc = new Document();
        PdfWriter.getInstance(doc, new FileOutputStream(outFile));
        doc.open();

        Font title = new Font(Font.HELVETICA, 16, Font.BOLD);
        doc.add(new Paragraph("Relat rio Di rio     A Minha Dieta",
    title));
        doc.add(new Paragraph("Utilizador: " + p.getName()));
        doc.add(new Paragraph("Data: " + day));
        doc.add(new Paragraph(" "));

        PdfPTable table = new PdfPTable(4);
        table.addCell("Descri  o");
        table.addCell("Kcal");
        table.addCell("Prot (g)");
        table.addCell("HC (g)");

        for (MealEntry m : p.mealsOn(day)) {
            table.addCell(m.getDescription());
            table.addCell(String.valueOf(m.getCalories()));
            table.addCell(String.valueOf(m.getProteinG()));
            table.addCell(String.valueOf(m.getCarbsG()));
        }
        doc.add(table);
        doc.close();
    }
}
```

Listing 10: Exportação PDF (excerto representativo)

7 Testes e Validação

A validação foi conduzida por cenários de uso (testes manuais) e verificação de invariantes no modelo. A independência do *Model* relativamente à UI sugere, como extensão natural, testes unitários ao nível de cálculos e agregações.

CT	Descrição e resultado esperado
CT01	Criar perfil válido → metas calculadas e persistência após reinício.
CT02	Registar refeição → totais e macros diários atualizados + PieChart coerente.
CT03	Registar água → barra de progresso e pie slice de água atualizados.
CT04	Registar exercício → BarChart semanal atualizado.
CT05	Registar peso → gráfico de peso atualizado e histórico consistente.
CT06	Exportar relatório → PDF gerado com dados do dia selecionado.

Tabela 2: Casos de teste representativos (testes manuais).

8 Conclusão e Trabalho Futuro

8.1 Conclusão

O projeto *A Minha Dieta* permitiu consolidar conceitos fundamentais de POO num cenário realista, integrando arquitetura MVC, modelação de domínio, validação, persistência e visualização de dados através de gráficos no *dashboard*. A separação entre domínio e interface promove manutenibilidade e suporta evolução incremental do sistema.

8.2 Trabalho futuro

- Persistência baseada em SQLite para maior robustez e consultas avançadas;
- Integração com APIs externas para enriquecer a base de alimentos;
- Testes unitários automatizados no *Model* e testes de UI;
- Relatórios analíticos (tendências semanais/mensais) com indicadores comparativos.

9 Bibliografia e Referências

Referências

[1] Java Documentation. https://docs.oracle.com/en/java/

[2] OpenJFX (JavaFX). https://openjfx.io/

[3] OpenPDF (LibrePDF). https://github.com/LibrePDF/OpenPDF

[4] Basal Metabolic Rate (contexto TMB/BMR). https://en.wikipedia.org/wiki/Basal_metabolic_rate

A Anexo A — Dashboard (opcional)

Se pretender incluir uma captura do *dashboard* no relatório, adicione ao Overleaf um ficheiro chamado `dashboard.png`.

Imagem não encontrada: `dashboard.png`
(coloca este ficheiro no Overleaf ou atualiza o nome)

Figura 1: Captura do dashboard da aplicação (opcional).