Universidade do Minho

Braga, Portugal

TRABALHO PRÁTICO - RELATÓRIO

Aplicação de um Centro de Treinos

Programação Orientada a Objetos

Departamento de Informática

Engenharia Informática 2023/24

Equipa de Trabalho:

A104365 - Fábio Magalhães

A100830 - Margarida Pimenta

A100896 - João Rodrigues

Índice

1. Introdução

No âmbito da UC de Programação Orientada aos Objetos, foi-nos proposta a realização de uma aplicação dedicada à gestão de atividades físicas e planos de treino personalizados.

Este sistema pretende facilitar o acompanhamento e a organização das rotinas de treino de diferentes tipos de utilizadores, desde atletas profissionais a praticantes ocasionais, abrangendo uma variedade de atividades como no nosso caso a corrida, o remo, os abdominais e levantamento de pesos.

Como principal objetivo, pretendemos alcançar uma plataforma intuitiva e efeciente, mas acima de tudo que permita aos utilizadores registar e monitorizar as suas atividades, bem como receber planos de treino personalizados com base nos seus objetivos.

Para além disso, o nosso sistema incluirá uma funcionalidade que permite simular o avanço no tempo, o que facilita a planificação e a execução de treinos futuros conforme estabelecido no plano de treino do utilizador. Esta capacidade de "salto temporal" é importante em termos de teste do sistema, pois simula um cenário de uso real, o que garante que todos os planos programados sejam ativados e executados.

De um modo geral, este projeto visa então desenvolver uma aplicação acessível para a gestão de atividades físicas e planos de treino, adequada e aconselhada a todo o tipo de utilizadores.

2. Diagrama de Classes

Abaixo apresentamos o diagrama de classes do nosso projeto, que ilustra a estrutura de classes e a relação entre elas.

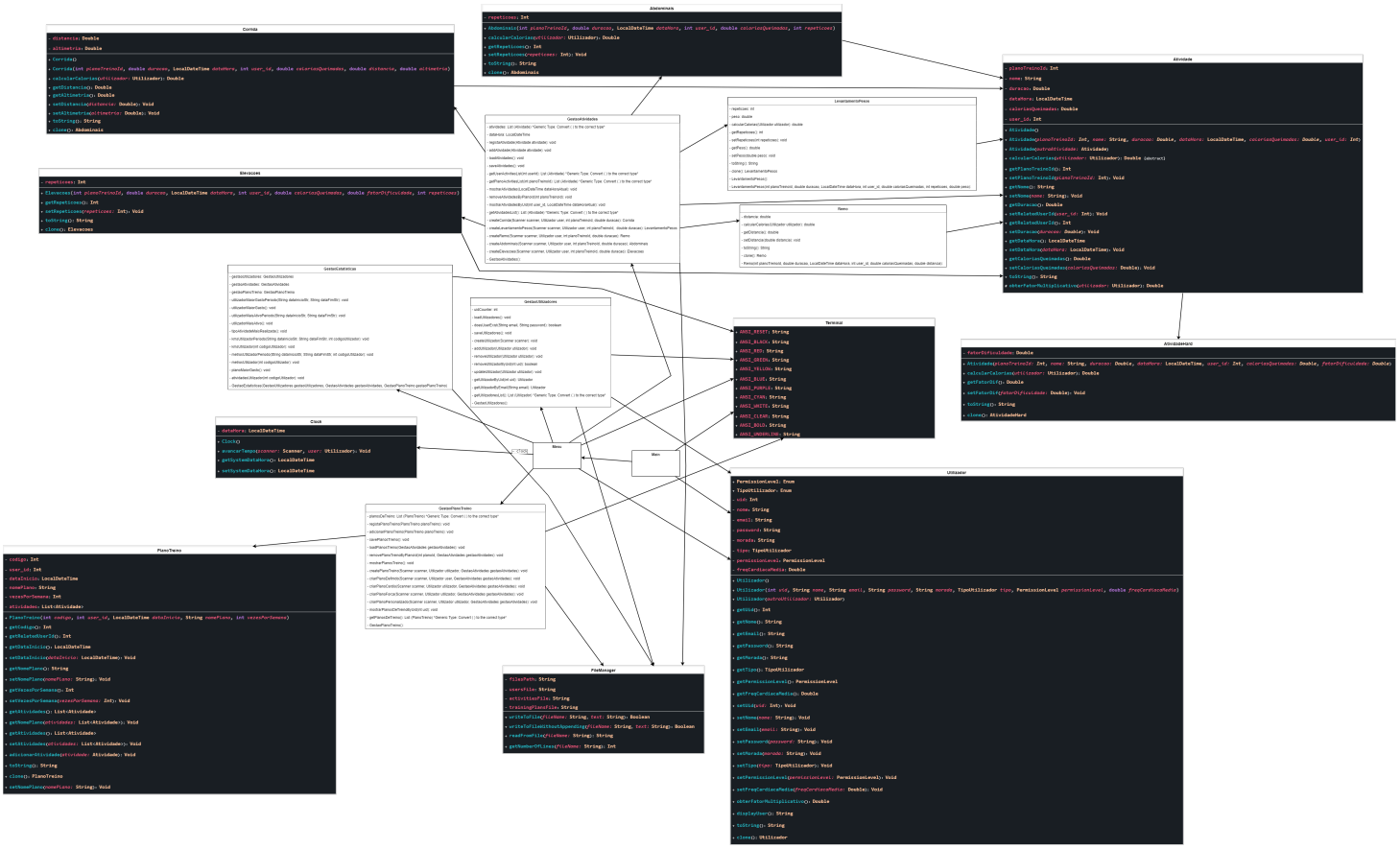

Figura 1: Diagrama de Classes

No caso de imagem não ser visível, é possivel aceder ao ficheiro 'diagrama.png' na pasta 'images'. Ou pelo link: https://github.com/POO-UM/GrupoTP-25/blob/main/Relatorio/images/diagrama.png

3. Utilizadores

No contexto desta aplicação, os Utilizadores do sistema são o foco do desenvolvimento deste produto.

Dado isto, e de modo a personalizar o sua utilização e atividade, é importante obter o máximo de informação e dados dos mesmos.

Começando pelos dados mais básicos mas, no entanto, cruciais, temos as informções que promovem a sua identificação e, consequentemente, garente uma maior personlização.

Destacamos abaixo os atributos criados que auxiliam neste processo.

```
private int uid;
private String nome;
private String email;
private String password;
private String morada;
public TipoUtilizador tipo;
private PermissionLevel permissionLevel;
private double freqCardiacaMedia;
```

Figura 2: Atributos Classe Utilizador

Como podemos verificar, temos atributos como o ID (identificação única), o nome, a morada, email e a frequência cardíaca, por exemplo.

Para além disso temos também o tipo de utilizador, que consiste em três possibilidades, sendo elas o Praticante Ocasional, Amador e Profissional, estando assim definidas no nosso programa:

```
public enum TipoUtilizador {
    PRATICANTE_OCASIONAL,
    AMADOR,
    PROFISSIONAL
}
```

Figura 3: Tipo de Utilizador

Como podemos verificar, temos também um nível de permissão, que está associado ao facto de o utilizador ser um cliente ou um administrados. Tal vai ser explicado e detalhado mais adiante no **Manual de Utilização**.

4. Atividades

No desenvolvimento deste programa, decidimos escolher quatro atividades, seguindo estas parâmetros associados a distância e altimetria, como é o caso da corrida, apenas distância, como no remo, e repetições para os abdominais e para o levantamento de pesos.

Para além de tratarmos das suas respetivas características, efetuamos também os seus respetivos cálculos e registos.

De seguida, iremos individualmente explicar as respetivas características e métodos adotados para cada das atividades escolhidas.

4.1. Classe Atividade

Através da aplicação dos conceitos e métodos abordados e estudados no seguimento da UC de Programação Orientada aos Objetos, decidimos criar a classe Atividade, sendo esta a classe principal relativa às atividades, ou seja, a super classe.

Nesta classe **definimos os atributos que são comuns a todas as atividades**, tal como evidenciamos na imagem abaixo representada.

```
public Atividade() {
    this.planoTreinoId = 0;
    this.nome = "";
    this.duracao = 0.0;
    this.user_id = 0;
    this.caloriasQueimadas = 0.0;
    this.dataHora = LocalDateTime.now();
}
```

Figura 4: Atributos Comuns às Atividades

Atributos como o nome, a duração, as calorias queimadas e a data e hora, são parâmetros para uma leitura mais simples e complexa. Já os parâmetros do ID do plano de treino e do user são utilizados para garantir uma **maior organização** e **utilização mais personalizada e individual a cada utilizador**.

Os atributos aqui definidos, serão **herdados por todas as suas subclasses**, ou seja, pelas classes das atividades.

4.1.1. Atividade *Hard*

Para além dos atributos comuns de uma atividade, a atividade *Hard* tem ainda um atributo que define a intensidade/dificuldade da atividade, que é um valor multiplicativo utilizado para calcular as calorias queimadas.

Como exemplo temos a atividade **Elevações**, que é uma atividade de força que requer um valor de repetições e o nivel de dificuldade (requerimento da classe de Atividade *Hard*).

Nesta atividade, o cálculo das calorias queimadas, ou pode ser feita dependendo da atividade, ou é feito com base na fórmula:

```
@Override
public double calcularCalorias(Utilizador utilizador) {
    double fatorMultiplicativo = utilizador.obterFatorMultiplicativo();

    return fatorDificuldade * 30 * getDuracao() * fatorMultiplicativo;
}
```

Figura 5: Cálculo de Calorias da Classe Atividade *Hard*

4.2. Subclasses Corrida e Remo

Iniciando a descrição das nossas subclasses, de modo a não tornar os tópicos repetitivos, consolidamos num só, subclasses que apesar de independentes, partilham algum atributo em comum. O mesmo irá ser aplicado ao tópico seguinte.

Começando por explorar a subclasse Corrida, nesta subclasse, definimos as características relativas e específicas a esta, sendo elas a distância e a altimetria que o utilizador deseja ou necessita para a sua atividade.

Para além disso, desenvolvemos um método de cáculo de calorias tendo em contas os atributos desta atividade, tal como evidenciamos na imagem abaixo descrita.

```
@Override
public double calcularCalorias(Utilizador utilizador) {
    double fatorMultiplicativo = utilizador.obterFatorMultiplicativo();
    return (distancia + altimetria) * 50 * getDuracao() * fatorMultiplicativo;
}
```

Figura 6: Cálculo de Calorias da Classe Corrida

Quanto à subclasse Remo, nesta classe, contrariamente à descrita acima, para além dos atributos herdados, apenas é definida a distância pelo o utilizador. No entanto, se este assim preferir, a mesma é calculada a partir de um Plano de Treino personalizado, questão que será abordada e detalhada mais adiante.

Deste modo, o cálculo de calorias para o remo é definido da seguinte forma:

```java
@Override
public double calcularCalorias(Utilizador utilizador) {
    double fatorMultiplicativo = utilizador.obterFatorMultiplicativo();

    return distancia * 30 * getDuracao() * fatorMultiplicativo;
}
```

Figura 7: Cálculo de Calorias da Classe Remo

Tal como podemos verificar, em ambas as atividades, para efetuar o **cálculo do gasto calórico de um determinado utilizador** recorremos aos dados introduzidos pelo cliente da aplicação e tendo em conta o seu estatuto, multiplicamos pelo fator multiplicativo, que é tanto maior, quanto maior o estatuto do utilizador.

Para além disto, usamos os métodos habituais, os **getters** e **setters**, o método **toString** e o **clone**, lecionados a abordados.

4.3. Subclasses Abdominais e Levantamento de Pesos

Quanto à classe Abdominais, sendo esta uma atividade de força que apenas requer o valor de repetições a efetuar, o nosso cálculo de calorias reside unicamente nesse valor relativo a esta atividade.

```java
@Override
public double calcularCalorias(Utilizador utilizador) {
    double fatorMultiplicativo = utilizador.obterFatorMultiplicativo();
    return repeticoes * 0.1 * getDuracao() * fatorMultiplicativo;
}
```

Figura 8: Cálculo de Calorias da Classe Abdominais

Na classe Levantamento de Pesos, comparativamente aos abdominais, a unica diferença é que nesta subclasse ainda existe a variável peso.

Assim sendo, abaixo destacamos como efetuamos o cálculo de calorias respetivo à atividade.

```java
@Override
public double calcularCalorias(Utilizador utilizador) {
    double fatorMultiplicativo = utilizador.obterFatorMultiplicativo();

    return (repeticoes * peso) * 0.2 * getDuracao() * fatorMultiplicativo;
}
```

Figura 9: Cálculo de Calorias da Classe Levantamento de Pesos

Identicamente ao ponto anterior, para efetuar o **cálculo do gasto de calorias de uma determinada atividade** recorremos aos dados introduzidos ou indicados pelo cliente da aplicação e tendo em conta o seu estatuto, multiplicamos pelo fator multiplicativo, que é tanto maior, quanto maior o estatuto do utilizador.

Para além disso, usamos igualmente os métodos habituais, os **getters** e **setters**, o método **toString** e o **clone**, lecionados a abordados em aula.

5. Plano Treino

A classe PlanoTreino permite a criação de planos de treino personalizados para os utilizadores. Estes planos são gerados com base numa lista de atividades que o utilizador deverá realizar com certa frequência (número de vezes por semana).

Destacamos abaixo os atributos criados que auxiliam neste processo.

```
public class PlanoTreino {
    private int codigo;
    private int user_id;
    private LocalDateTime dataInicio;
    private String nomePlano;
    private int vezesPorSemana;
    private List<Atividade> atividades;
```

Figura 10: Atributos Classe PlanoTreino

Para além disso, usamos igualmente os métodos habituais, os **getters** e **setters**, o método **toString** e o **clone**, lecionados a abordados em aula.

6. Gestão Utilizadores

A classe GestaoUtilizadores é responsável pela gestão de utilizadores no sistema, abrangendo uma série de funcionalidades essenciais para o controlo e manutenção de dados dos mesmos. Inicialmente, a classe configura um contador de identificadores únicos para os utilizadores. Esse contador é fundamental para atribuir um novo identificador sempre que um novo utilizador é adicionado ao sistema.

O método de carregamento dos utilizadores lê dados do users.csv e preenche um Map onde cada utilizador é indexado pelo seu identificador único. Esse processo envolve a análise de cada linha do arquivo, a separação dos campos e a criação de um novo utilizador com base nesses dados.

Outra funcionalidade é a gravação dos dados dos utilizadores de volta no csv. Essa operação converte toda a coleção de utilizadores numa string formatada e escreve-a no csv.

Além disso, a classe permite a criação de novos utilizadores através de uma interação direta que recebe informações pelo terminal. Antes de adicionar um utilizador, verifica se já não existe um com o mesmo email e password. Se confirmada a inexistência, o utilizador é então adicionado ao mapa com um novo identificador único.

7. Gestão Atividades

A classe GestaoAtividades trata da gestão das atividades no sistema.

À semelhança da classe GestaoUtilizadores, tem a capacidadde de ler dados de um arquivo csv, e preenche uma lista de atividades com base nesses dados. Tem também a funcionalidade de, posteriornmente, gravar os dados das atividades de volta no csv.

Para além disso, a classe permite a criação de novas atividades através de uma interação direta que recebe dados pelo terminal.

Outra funcionalidade importante é a capacidade de exibir as atividades de um determinado utilizador, que são filtradas com base no identificador do mesmo.

8. Gestão Plano de Treino

A classe GestaoPlanoTreino é responsável pela gestão dos planos de treino no sistema. Esta funciona de forma semelhante às GestaoAtividades, lendo e gravando dados de um arquivo csv, e permitindo a visualização dos planos de treino de um determinado utilizador.

A principal diferença é que esta classe permite a criação de planos de treino personalizados, que são gerados com base nas atividades que o utilizador deseja realizar. Permite também a criação de planos de treino predefinidos, como o "PlanoCardio" e o "PlanoForça".

9. Gestão Estatísticas

A classe GestaoEstatisticas é responsável pela gestão das estatísticas no sistema. Funciona através da utilização das classes GestaoUtilizadores, GestaoAtividades e GestaoPlanoTreino, que são utilizadas para obter os dados necessários para a geração de estatísticas.

Esta classe permite a visualização de estatísticas gerais, como o utilizador que mais calorias queimou, o utilizador mais ativo e o tipo de atividade mais realizada.

Permite também a visualização de estatísticas individuais, como as atividades realizadas por um determinado utilizador, o número de kilómetros percorridos num determinado período de tempo ou no total das suas atividades realizadas.

10. Manual de Utilização

Quanto ao manual de utilização, apresentamos funcionalidades como o **Registo** e o **Login**, tal como podemos confirmar na imagem abaixo.

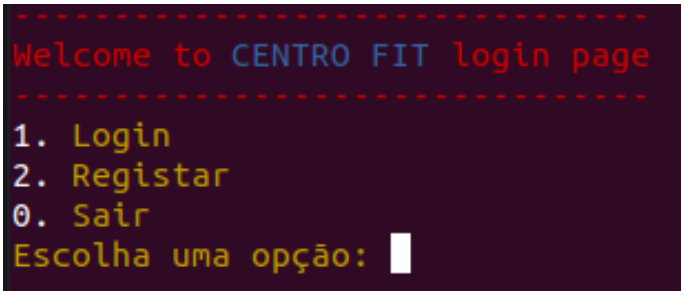

Figura 11: Página de Registo e Login

Após um utilizador efetuar o seu registo, e posteriormente o seu login, este terá acesso a uma página de funcionalidades onde terá opções como as que demonstramos de seguida.

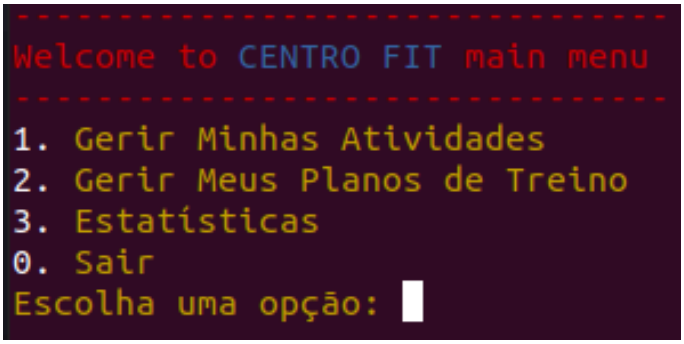

Figura 12: Funcionalidades Gerais do Programa

Tal como podemos ver, ponto a ponto, começando pela gestão das atividades do utilizador, neste é possível que o utilizador veja as suas atividades, adicione uma nova atividade à sua lista e avançar o tempo de modo a verificar a conclusão das atividades e os seus resultados obtidos.

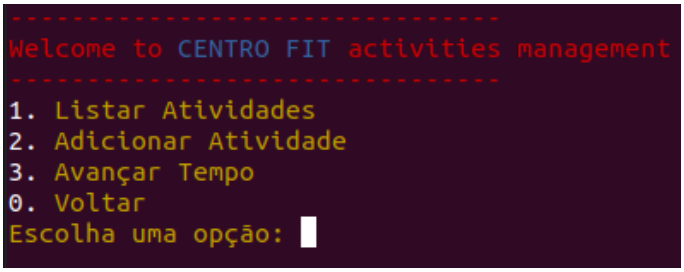

Figura 13: Gestão das Atividades

Quanto à gestão dos planos de treino, nesse ponto o utilizador pode ver a lista dos seus planos, pode criar e remover planos de treino e, para além disso, pode criar um plano definido ou personalizado. A diferença entre estes, reside no facto de o definido ser com base em cardio ou força, enquanto que o personalizado, tem em base objetivos do utilizador.

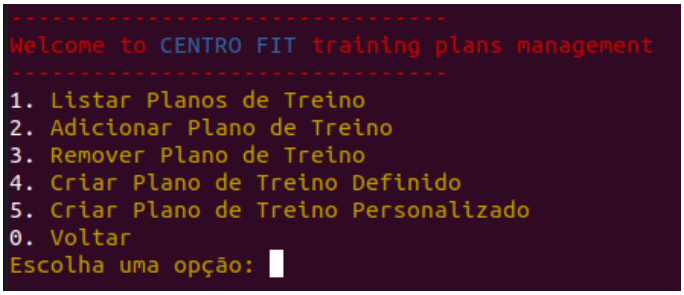

Figura 14: Gestão dos Planos de Treino

Por fim, nas estatísticas, tal como podemos verificar, somos capazes de extrair os seguintes dados apresentados abaixo.

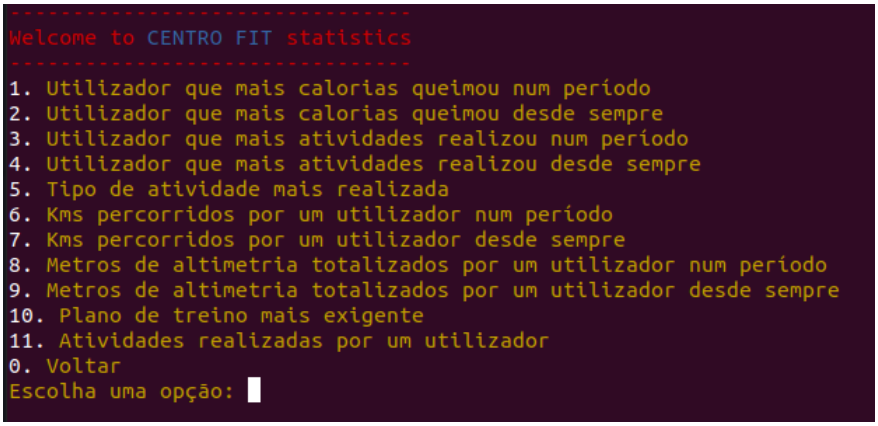

Figura 15: EStatísticas

11. Conclusão

Achamos que cumprimos maioritarimente os objetivos propostos, tendo obtido um resultado final bastante satisfatório. A nossa aplicação segue os princípios da Programação Orientada aos Objetos, com uma estrutura de classes bem definida e organizada, que permite uma fácil manutenção e expansão do código. Cumpre também os requisitos propostos, permitindo a gestão de utilizadores, atividades e planos de treino, bem como a visualização de estatísticas.