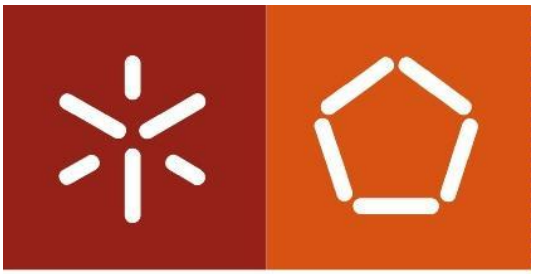

Universidade do Minho

Licenciatura em Engenharia Informática

Programação Orientada aos Objetos

Trabalho Prático - Grupo 34

Tiago Matos Guedes (A97369)

Diogo Afonso Costa Gonçalves (a101919)

Bruno Gonçalo Costa Campos (A98639)

11 de maio de 2024

Índice

Introdução

No âmbito da Unidade Curricular de Programação Orientada aos Objetos, foi-nos atribuído um projeto prático que envolve o desenvolvimento de uma plataforma de gestão de atividades físicas e saúde. Esta plataforma tem como objetivo principal proporcionar aos utilizadores uma experiência interativa onde possam monitorar e gerir suas atividades físicas, bem como acompanhar seu progresso em direção a metas de condicionamento físico e bem-estar.

Dado que este projeto deve aderir aos princípios da programação orientada a objetos, é crucial estabelecer uma hierarquia de classes bem estruturada que promova a escalabilidade e a manutenção do sistema. Isso é especialmente importante, pois novos recursos e funcionalidades podem ser adicionados no futuro sem comprometer a estrutura existente.

Neste relatório, apresentaremos e justificaremos a hierarquia de classes adotada para o projeto de plataforma de fitness, bem como detalhes arquiteturais que consideramos fundamentais para o desenvolvimento sustentável da plataforma.

Classes

Fitness App

O programa como um todo ao qual decidimos denominar por Fitness App.

Atividades

```
private double tempo;
private boolean isHard;
```

Atividade é uma das classes do nosso sistema, ao qual se refere a características que todo o tipo de atividades contém, existindo vários tipos de atividades;

Corrida em Pista

Uma das subclasses da classe Atividade que consiste numa das atividades que apenas engloba a distância como variável de instância.

```
private double distancia;
```

Natação

Uma das subclasses da classe Atividade que consiste numa das atividades que apenas engloba a distância como variável de instância, sendo neste a diferença para a corrida em pista que menos distância tem o mesmo gasto calórico.

```
private double distancia;
```

Corrida no Monte

Uma das subclasses da classe Atividade que consiste numa das atividades que engloba tanto a distância como a altimetria da atividade.

```
private double distancia;
private double altimetria;
```

Bicicleta no Monte

Uma das subclasses da classe Atividade que consiste numa das atividades que engloba tanto a distância como a altimetria da atividade, sendo neste a diferença para a corrida no monte que tem menor gasto calórico.

```
private double distancia;
private double altimetria;
```

Flexões

Uma das subclasses da classe Atividade que consiste numa das atividades que apenas engloba as repetições como variável de instância.

```
private int repeticoes;
```

Abdominais

Uma das subclasses da classe Atividade que consiste numa das atividades que apenas engloba as repetições como variável de instância.

```
private int repeticoes;
```

Supino

Uma das subclasses da classe Atividade que consiste numa das atividades que engloba as repetições e o peso como variáveis de instância.

```
private int repeticoes;
private double peso;
```

Prensa

Uma das subclasses da classe Atividade que consiste numa das atividades que engloba as repetições e o peso como variáveis de instância, mas como sendo um exercício de pernas exige menos gasto calórico.

```
private int repeticoes;
private double peso;
```

Utilizadores

A classe Utilizador possui todas as variáveis de instância que nós achámos intrínsecas um utilizador desta esta app conter:

```
private int código;
private String Nome;
private String Morada;
private String Email;
private double frequenciaCardiacaMedia;
private HashMap<LocalDate, List<AtividadecomCaloriasGastas>> atividades;
private double caloriasGastas;
```

Utilizadores Profissionais

Uma das subclasses da classe Utilizador que define os utilizadores mais experientes, que por exemplo, contêm o maior fator multiplicativo e por isso tem um gasto calórico superior aos restantes.

Utilizadores Amadores

Uma das subclasses da classe Utilizador que se refere ao patamar intermédio de um utilizador, isto é, os utilizadores que têm um fator multiplicativo médio no gasto calórico.

Utilizadores Praticantes Ocasionais

Uma das subclasses da classe Utilizador, ao qual se refere aos utilizadores cujo têm um fator multiplicativo mais baixo, devido a serem menos regulares que os anteriores.

Planos de Treino

```
private Utilizador utilizador;
private LocalDate data;
private Map<Atividade,Integer> atividades;
```

Planos de Treino é uma das classes da nossa app que tem a funcionalidade de agrupar um certo número de atividades que um respetivo utilizador tem planeado fazer numa certa data.

Estatísticas

Estatísticas foi um módulo criado na nossa app com a funcionalidade de conseguir saber algumas particularidades acerca do nosso "save", por exemplo:

- Qual foi o utilizador que gastou mais calorias.
- Qual foi o utilizador que mais atividades realizou.
- Qual o tipo de actividade mais realizada
- Quantos kms é que um utilizdor realizou num período ou desde sempre
- Quantos metros de altimetria é que um utilizar totalizou num período ou desde sempre
- Qual o plano de treino mais exigente em função do dispêndio de calorias proposto
- Listar as actividades de um utilizador

Recordes

```
private Atividade atividade;
private Utilizador utilizador;
private LocalDate data;
private String especificacao;
private double valor;
```

A classe recordes é responsável pelos recordes de cada modalidade representada na app, podendo representar ou o maior número de calorias queimadas, ou a maior distância nas atividades de distância, ou até mesmo o maior peso levantado nas atividades de peso, etc...

Controller

Recebe uma mensagem do Reader e a partir daí faz uma interpretação de qual a funcionalidade requerida pelo utilizador, de seguida essa mesma mensagem é reenviada, juntamente com uma flag a indicar a funcionalidade, para o respetivo controlador.

ControllerRegistos

Ao receber a mensagem e flag enviadas pelo Controller, sabe de que forma deve notificar o Gestor sem que a mensagem tenha de voltar a ser interpretada, como se trata do controlador de registos, é natural que todas as suas notificações estejam relacionadas com a alteração das estruturas de dados presentes no Gestor.

ControllerEstatisticas

Tal como o controller anterior recebe uma mensagem e flag do controller principal e notifica o Gestor relativamente a questões estatísticas.

Gestor

É dentro desta secção que estão incluídas as estruturas de dados que suportam por completo o correto funcionamento do sistema, como tal é aqui que a classe *Gestor* opera conforme as notificações que recebe dos controladores.

Uma vez que o *Gestor* tem de salvaguardar os registos de objetos como utilizadores, atividades, planos de treino e recordes, este deve possuir estruturas de dados que permitam o acesso direto a um determinado objeto. Contudo, não é correto implementar um *Gestor* que possua coleções de objetos tão distintos

Foram criadas então classes auxiliares para se poder lidar com cada coleçao individualmente:

- GestorUtilizadores
- GestorAtividades
- GestorPlanosdeTreino
- GestorRecordes

Menu

Esta é a classe que possibilita a comunicação entre o utilizador e a app, tentando ser o mais concisa possível e de fácil interpretação para o utilizador em causa.

Funcionalidades

- Inserir Utilizador ⇒ Adiciona um novo utilizador ao sistema.
- Inserir Atividade ⇒ Permite adicionar uma nova atividade ao sistema.
- Inserir Plano de Treino ⇒ Permite a criação de um novo plano de treino por um utilizador.
- Remover Utilizador ⇒ Remove um utilizador existente do sistema.
- Remover Atividade ⇒ Remove uma atividade existente do sistema.
- Remover Plano de Treino ⇒ Remove um plano de treino existente do sistema.
- Listar Utilizadores ⇒ Mostra uma lista de todos os utilizadores registrados no sistema.
- Listar Atividades ⇒ Exibe uma lista de todas as atividades disponíveis no sistema.
- Listar Planos de Treino ⇒ Apresenta uma lista de todos os planos de treino registrados no sistema.
- Registar Atividade ⇒ Permite que um utilizador registe uma atividade específica no sistema.
- Obter Recordes ⇒ Permite aos utilizadores visualizarem os recordes de atividades no sistema.
- Avançar no tempo ⇒ Permite que o administrador do sistema avance a data atual.
- Estatísticas ⇒ Fornece estatísticas diversas sobre o uso do sistema.

Diagrama de Classes

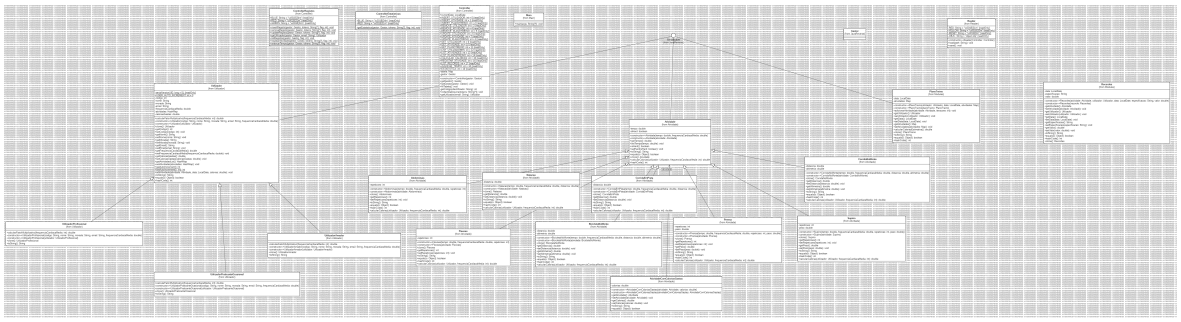

Figure 4.1: Diagrama de Classes

Conclusão

Para finalizar, podemos afirmar que, apesar de poder não estar alguma funcionalidade 100 por cento implementada ou exatamente como foi pedida, temos um projeto funcional e, a nosso parecer, bem organizado. Demos o nosso melhor com o nosso conhecimento para colocar o código o mais legível possível e preservando ao máximo a noção de hierarquia e encapsulamento. Foi, no geral, um projeto que contribuiu bastante para o nosso crescimento no âmbito da programação de objetos.

Adicionalmente, gostaríamos de mencionar algumas dificuldades encontradas durante o desenvolvimento do projeto. Em particular, tivemos dificuldade em compreender completamente o que era solicitado em certos pontos do enunciado, como na categorização de atividades como "hard", a definição precisa do conceito de tempo de uma atividade - se era algo fixo ou preenchido posteriormente pelo utilizador - e até mesmo a própria definição do que constitui uma atividade dentro do contexto da aplicação. Apesar desses desafios, conseguimos contornar essas dificuldades e produzir uma solução que atende aos requisitos principais do projeto.