Universidade do Minho

Universidade Do Minho

Relatório

Programação Orientada aos Objetos



Grupo 60

- Pedro Miguel Araújo Gomes, A104540
- João Nuno Pereira Machado, A104084
- Pedro Manuel Dias Teixeira, A106998

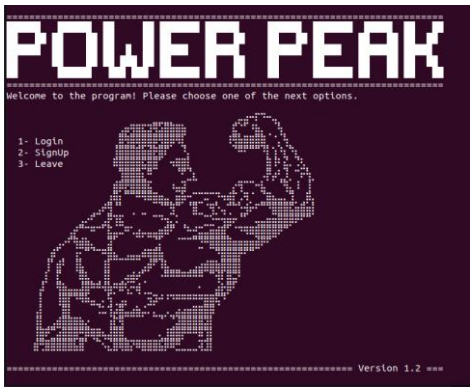
1 - Introdução

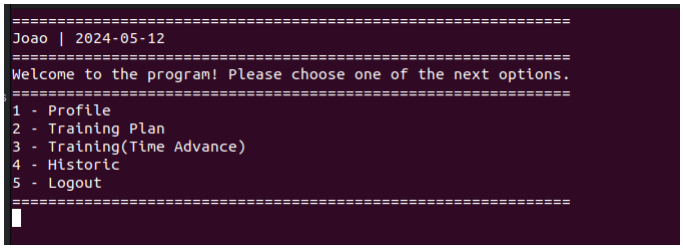
Este relatório tem como intuito abordar o trabalho prático da UC Programação Orientada aos Objetos.

Mais à frente falaremos das decisões que foram tomadas quanto às funcionalidades a serem implementadas pelo grupo. Este projeto envolve o desenvolvimento de uma aplicação para gerir atividades e planos de treino de praticantes de atividades físicas, ***Power Peak***.

2 - Interface

A nossa interface apresenta ao utilizador menus onde o mesmo é apresentado diferentes tipos de opções a seguir através do input de um *int* no terminal.



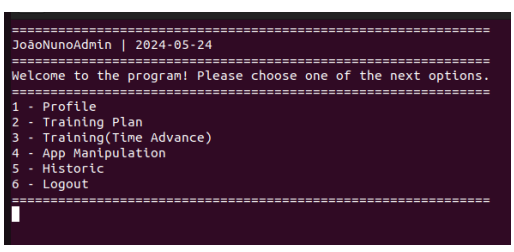


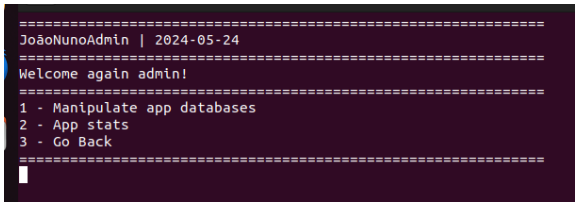
2.1 – Conceito de Admin

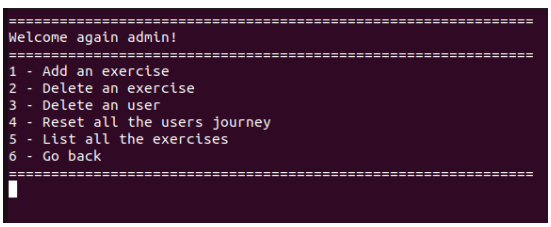
Devido a certas funcionalidades, foi necessário criar uma forma de separar utilizadores de administradores, sendo assim, decidimos que a melhor forma seria criar contas dedicadas a administradores(“GomesAdmin” | ”PedroAdmin” | ”JoãoNunoAdmin”).

Esta implementação foi necessária, pois existem ações relativas às informações do programa (*database* de exercícios, adicionar ou remover exercícios, remover utilizadores e lista de utilizadores e exercícios) que só podem ser acessadas e modificadas pelos *admin*.

O que diferencia o utilizador regular de um *admin* é a aparição da aba “*App Manipulation*” no menu principal.







3 – Arquitetura

A arquitetura do programa que escolhemos foi a seguinte:

3.1 – Packages

Com intuito de organizar melhor as classes do projeto foram criadas os seguintes *packages:*

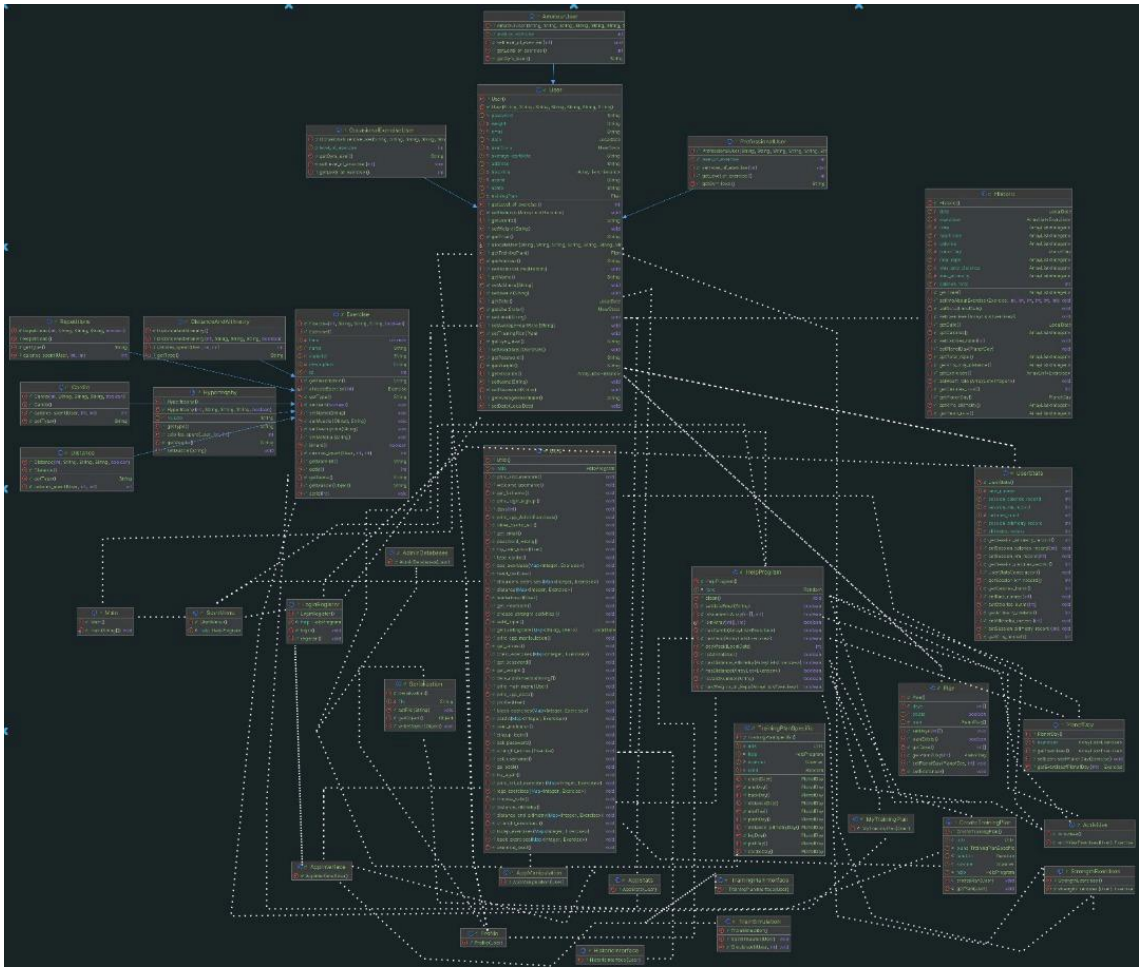
- Entities: aqui guardamos todas as entidades que foram desenvolvidas durante o projeto

- Help: repositório onde são armazenadas todas as classes que possuem métodos que ajudam a melhorar o programa

- Menus: todos as interfaces “gráficas” que são apresentados durante o programa são reunidas aqui

- Training: todas as classes relacionadas a treino, sejam elas criação de planos ou de treinos singulares.

3.2 – Diagrama de classe



3.3 – Entidades

3.3.1 – Utilizadores

A aplicação dá liberdade ao utilizador aquando do seu registro de escolher o seu nível de treino. Assim, a mesma possui três diferentes tipos de utilizadores:

- profissionais;

- amadores;

- praticantes ocasionais;

Estes mesmos são instâncias de uma classe abstrata ***User***.

```java
public abstract class User implements Serializable{

    private String userId;
    private String name;
    private String password;
    private String address;
    private String email;
    private String averageHeartRate;
    private String weight;
    private LocalDate date;
    private Plan trainingPlan;
    private UserStats userStats;
    private ArrayList<Historic> historics;
```

```java
public int level_of_exercise;

public ProfessionalUser(String userId, String password, String name, String address, String email, String averageHeartRate, String weight) {
    super(userId, password, name, address, email, averageHeartRate, weight);
    this.level_of_exercise = 2;
}
```

Estes tipos de utilizador são diferenciados pelo ***level_of_exercise***(0 – Praticante Ocasional | 1 – Amador | 2 – Profissional) que vai incidir diretamente no cálculo de calorias despendidas a treinar.

3.3.2 – Atividades

Com vista a facilitar alterações no cálculo de calorias, foi criada uma classe abstrata ***Exercise*** seguida dos seus diferentes tipos: ***Cardio***, ***Distance***, ***DistanceAndAltimetry***, ***Hypertrophy*** e ***Repetitions***.

```java
public abstract class Exercise implements Serializable {

    private int id;
    private String name;
    private String material;
    private String description;
    private boolean hard;
```

```java
public class Cardio extends Exercise{

    public Cardio() {
    }

    public Cardio(int id, String name, String material, String description, boolean hard) {
        super(id, name, material, description, hard);
    }
```

```java
@Override
public int calories_spent(User user, int durationMinutes, int heart_rate) {
    double baseCaloriesPerMinute = 20.0;
    double levelMultiplier = 1.0 + (user.getLevel_of_exercise() - 1) * 0.2;
    double heartRateMultiplier = (heart_rate - 70) / 100.0 + 1;
    double weight_multiplier = 1 + (double) (Integer.parseInt(user.getWeight()) - 70) /100;

    double caloriesBurnedPerMinute = baseCaloriesPerMinute * levelMultiplier * heartRateMultiplier * weight_multiplier;

    double totalCaloriesBurned = caloriesBurnedPerMinute * durationMinutes;

    return (int) Math.round(totalCaloriesBurned);
}
```

3.3.3 – Planos de Treino

A nossa abordagem quanto à implementação de planos de treino no projeto consistiu em criar uma classe ***Plan*** que reúne uma lista de planos diários, associada à classe ***Daily Plan***, e uma lista de dias que posteriormente serão associados aos planos diários.

```
public class Plan implements Serializable {

    private boolean exists;
    private PlanofDay[] plan;
    private int[] days;
```

```
public class PlanofDay implements Serializable {
    private ArrayList<Exercise> exercises;
```

3.3.4 – Estatísticas e Histórico

Uma das funcionalidades pedidas na resolução do projeto é a implementação de um histórico dos treinos do utilizador e as suas estatísticas bem como a de outros utilizadores. Posto isto foram criadas duas classes ***Historic*** e ***UserStats***, respetivamente.

```
public class Historic implements Serializable {

    private LocalDate date;
    private ArrayList<Exercise> exercises;
    private PlanofDay planofDay;
    private ArrayList<Integer> total_reps;
    private ArrayList<Integer> time;
    private ArrayList<Integer> kms_only_distance;
    private ArrayList<Integer> kms_altimetry;
    private ArrayList<Integer> calories;
    private int calories_total;
    private ArrayList<Integer> heart_rate;
```

```
public class UserStats implements Serializable {

    private int calories_burnt;
    private int kms_runned;
    private int altimetry_meters;
    private int session_km_record;
    private int session_altimetry_record;
    private int session_calories_record;
```

4 – Funcionalidades

4.1 – Gestão de Entidades

É possível a quem utiliza a aplicação criar a sua conta, planos de treino e registar as suas atividades, bem como é possível apagar a sua conta. *Admins* podem também retirar qualquer utilizar, assim como adicionar e remover exercícios da "base de dados".

4.2 – Persistência do estado do programa

Ao desenvolver a aplicação, e face à necessidade de armazenar utilizadores, planos de treino e exercícios, recorremos à interface ***Serializable*** do Java para serializar os mesmos (conversão para sequência de bytes) facilitando o armazenamento das informações em ficheiro.

4.3 – Criação de planos de treino de acordo com objetivos

A nossa aplicação permite uma variedade de formas do utilizador criar o seu plano de treino assim como requisitar um ao sistema.

Na criação do plano de treino por parte do cliente é apresentado a este uma biblioteca de exercícios que este pode percorrer e consequentemente adicionar as atividades ao seu plano de treino.

Na geração do plano de treino por parte da aplicação, o utilizador pode escolher entre três opções de acordo com os seus objetivos, sendo elas ganhar músculo, perder peso, ou ele pedir certos tipos de treino para um certo dia. Os planos de treino gerados são sempre diferentes (*randomizados)* em termos de exercícios podendo o cliente facilmente pedir outro caso não goste do que obteve.

4.4 – Avanço do tempo

No nosso projeto é possível avançar o tempo de duas formas:

- Plano de Treino : onde o *user* pode avançar o tempo que assim desejar mediante a posse de um plano de treino e neste avanço de tempo é simulado tudo e apresentado no histórico e nas estatísticas.

- Treino singular : onde o utilizador regista as suas atividades do dia e por fim a data avança para o dia seguinte.

É importante também dizer que ao realizar um avanço de tempo este é feito para todos os utilizadores que estejam inscrito na aplicação bem como a simulação dos seus treinos e atualização dos respetivos históricos e estatísticas.

4.5 – Noção de Atividade “Hard”

No projeto é proposta a implementação de atividades difíceis. Esta foi implementada com sucesso e pode ser aplicada a qualquer atividade que esteja no programa.

4.6 – Estatísticas do programa

É possível ao utilizador ver as suas estatísticas relativas aos seus treinos, bem como o *leaderbord* da aplicação.

5 – Conclusão

A realização deste projeto permitiu-nos perceber melhor os conceitos da Programação Orientada a Objetos, tendo sido uma experiência enriquecedora.

Durante o desenvolvimento do projeto houveram dificuldades, como por exemplo a serialização e o avanço do tempo para todos os utilizadores bem como a simulação dos seus treinos. Porém, estas adversidades permitiram-nos melhorar.