Uma imagem com captura de ecrã, Tipo de letra, Gráficos, símbolo

Descrição gerada automaticamente

Universidade do Minho

Licenciatura em Ciências da Computação

Trabalho Prático Programação Orientada aos Objetos

Grupo xx

Trabalho realizado por:

João Pedro da Silva Barbosa (A100054)

Diogo Coelho da Silva (A100092)

Pedro Miguel Ramôa Oliveira (A97686)

Uma imagem com pessoa, Cara humana, Queixo, pescoço

Descrição gerada automaticamenteUma imagem com Cara humana, pessoa, sorrir, Testa

Descrição gerada automaticamenteUma imagem com sorrir, pessoa, vestuário, Cara humana

Descrição gerada automaticamente

**Conteúdo**

[**1.Introdução** 3](#_Toc166281933)

[**2.Arquitetura da aplicação** 4](#_Toc166281934)

[**2.1Classe Activity** 4](#_Toc166281935)

[**2.2Classe Distance** 4](#_Toc166281936)

[**2.3Classe DistanceAltitude** 5](#_Toc166281937)

[**2.4Classe WeightLifting** 5](#_Toc166281938)

[**2.5 Classe Body-Weight** 5](#_Toc166281939)

[3.Implementação da Aplicação 6](#_Toc166281940)

[4.Conclusão 7](#_Toc166281941)

[Anexo1.Diagrama Classes 8](#_Toc166281942)

# **1.Introdução**

No âmbito deste trabalho prático, foi proposta a implementação de uma aplicação que faça a gestão de atividades físicas. Com esta aplicação, deve ser possível a cada utilizador registar a realização de atividades e obter um resumo das estatísticas das realizações das mesmas.

Este projeto foi também pensado para respeitar as ideias deste paradigma da programação, a programação orientada aos objetos. Com base nisto, tivemos de respeitar os seus ideais que incluem a abstração, encapsulamento e herança de classes, entre outros conceitos base de programação orientada a objetos.

Este relatório tem como objetivo apresentar a nossa linha de pensamentos para a conceção e desenvolvimento de funcionalidades da aplicação proposta. Para isso vamos fazer uma pequena explicação da arquitetura de classes utilizada, descrevendo brevemente as mesmas e os seus métodos.

# **2.Arquitetura da aplicação**

(METER FOTO DO DIAGRAMA DE CLASSES)

## **2.1Classe Activity**

Esta classe desempenha o papel de uma estrutura base para representar uma variedade de atividades físicas. Nela, são encapsuladas informações comuns a todas as atividades, como identificação, tipo, data e duração e um identificador de dificuldade da atividade (isHard). Esta classe serve como superclasse para as classes abstratas "Distance", "DistanceAltitude", "Weightlifting" e "Bodyweight", partilhando varíaveis de instância e implementação de métodos com a mesma assinatura. Esta abordagem promove uma estrutura modular e organizada para o código, facilitando a manutenção e expansão da aplicação. Ao utilizar herança e polimorfismo, as outras classes podem estender a partir desta classe para representar diferentes tipos de atividades, mantendo a estrutura, aumentando assim a consistente e a coesia dentro da aplicação. Isto contribui para os princípios do POO.

**2.2Classe Distance**

A classe mencionada serve como uma estrutura base e fundamental para a classe “Running”, sendo a sua superclasse. Todos os exercícios relacionados com o tipo “Distance” são representados por esta classe. Ambas as classes compartilham variáveis de instância, como identificação, tipo, data e duração, o que facilita a herança e cria um nível mais elevado de abstração. No entanto, para a classe "Running", são adicionados os parâmetros de ritmo (pace) e passos (steps), que são específicos para essa atividade. Essa abordagem foi escolhida pelo grupo por proporcionar uma maior facilidade e flexibilidade para a expansão da aplicação, permitindo a adição de novos exercícios deste tipo. Além disso, na classe "Distance", foi implementado o cálculo de calorias específico para atividades de corrida, levando em consideração fatores como a idade do usuário. Esse cálculo é realizado através da função calories(User user), que multiplica o MET (Metabolic Equivalent of Task) específico para a corrida, a duração da atividade, o peso do usuário e o fator de calorias do usuário. Essa integração do cálculo de calorias na classe "Distance" reflete a preocupação do grupo em criar uma estrutura coesa e abrangente para representar diversas atividades físicas, mantendo os princípios da Programação Orientada a Objetos (POO) de reutilização de código, modularidade e hierarquia de classes para promover uma arquitetura de código robusta e escalável.

**2.3 Classe DistanceAltitude**

Em semelhança com a anterior, as três próximas classes foram implementadas com o mesmo objetivo da primeira, satisfazer os requisitos do paradigma de Programação Orientada aos Objetos (POO). A classe mencionada serve como uma estrutura base e fundamental para a classe “MountainBike”, sendo a sua superclasse. Todos os exercícios relacionados com o tipo “DistanceAltitude” são representados pela classe. Assim como nas classes anteriores, ambas compartilham variáveis de instância comuns, como identificação, tipo, data e duração. No entanto, para a classe "MountainBike", é adicionado o parâmetro de ritmo (pace), que é específico para esse tipo de atividade. Além disso, o cálculo de calorias é diferente para o "MountainBike", levando em consideração fatores específicos, como a idade do usuário. Este cálculo é realizado através da função calories(User user) que multiplica o MET (Metabolic Equivalent of Task) específico para o MountainBike, a duração da atividade, o peso do usuário, e o fator de calorias do usuário. Essa abordagem foi escolhida pelo grupo por proporcionar uma maior facilidade e flexibilidade para a expansão da aplicação, permitindo a adição de novos exercícios deste tipo, mantendo a consistência e a coesão do código. Esses princípios refletem os requisitos do paradigma de Programação Orientada a Objetos (POO), enfatizando a reutilização de código, modularidade e hierarquia de classes para uma arquitetura de código mais robusta e escalável.

## 2.4 Classe WeightLifting

A classe mencionada serve como uma estrutura base e fundamental para a classe “BenchPress”, sendo a sua superclasse. Todos os exercícios relacionados com o tipo “WeightLifting” são representados por esta classe. Ambas as classes compartilham variáveis de instância, como identificação, tipo, data e duração, o que facilita a herança e cria um nível mais elevado de abstração. Além disso, para a classe "BenchPress", é adicionado o parâmetro de inclinação (inclination), que representa a inclinação do banco durante o exercício. Essa abordagem foi escolhida pelo grupo por proporcionar uma maior facilidade e flexibilidade para a expansão da aplicação, permitindo a adição de novos exercícios deste tipo. Também foi implementado o cálculo de calorias específico para a atividade de "BenchPress". Esse cálculo é realizado através da função calories(User user), que leva em consideração o fator de calorias do usuário. Essa integração do cálculo de calorias na classe "BenchPress" reflete a preocupação do grupo em criar uma estrutura coesa e abrangente para representar diversos exercícios de levantamento de peso, mantendo os princípios da Programação Orientada a Objetos (POO) de reutilização de código, modularidade e hierarquia de classes para promover uma arquitetura de código robusta e escalável.

## 2.5 Classe BodyWeight

A classe mencionada serve como uma estrutura base e fundamental para a classe “Squat”, sendo a sua superclasse. Todos os exercícios relacionados com o tipo “BodyWeight” são representados por esta classe. Ambas as classes compartilham variáveis de instância, como identificação, tipo, data, duração, repetições (reps) e séries (sets), o que facilita a herança e cria um nível mais elevado de abstração. Além disso, para a classe "Squat", é adicionado o parâmetro RPE (Rate of Perceived Exertion), que representa a percepção subjetiva do esforço durante o exercício. Essa abordagem foi escolhida pelo grupo por proporcionar uma maior facilidade e flexibilidade para a expansão da aplicação, permitindo a adição de novos exercícios deste tipo. Também foi implementado o cálculo de calorias específico para a atividade de "Squat". Esse cálculo é realizado através da função calories(User user), que leva em consideração o fator de calorias do usuário. Essa integração do cálculo de calorias na classe "Squat" reflete a preocupação do grupo em criar uma estrutura coesa e abrangente para representar diversos exercícios de peso corporal, mantendo os princípios da Programação Orientada a Objetos (POO) de reutilização de código, modularidade e hierarquia de classes para promover uma arquitetura de código robusta e escalável.

## 2.6 Classe Utilizador

Esta classe abstrata representa todos os utilizadores que interagem com o programa. Esta classe foi escolhida para superclasse de “Professional”, “Amateur” e “Occasional” devido a possuírem varíaveis de instancia em comum. Estas último servem para representar um tipo de utilizador em específico. Cada tipo de utilizador conta com métodos de cálculos diferentes para refletir na sua condição.

Da forma que a classe está implementada é relativamente fácil adicionar outro tipo de utilizador à aplicação.

## 2.7 Classe Fitness

A classe “Fitness” é responsável por fazer a gestão de dados da aplicação relacionados com as atividades dos utilizadores. Esta classe foi criada para encapsular os dados relativos às atividades físicas dos utilizadores, incluindo as variáveis de instância e métodos.

## 2.8 Classe Menu

A classe “Menu” é responsável por fornecer uma interface interativa com o utilizador.

## 2.9 Classe WorkoutPlan

-TODO

# 3.Funcionalidades

-TODO

# 4.Explicação da arquitetura e algumas decisões

# 5. Conclusão

Em suma, a conceção e desenvolvimento desta aplicação de gestão de atividades físicas foi uma jornada empolgante e desafiadora, na qual aplicamos os princípios fundamentais da Programação Orientada a Objetos (POO) de forma eficaz. Desde a fase inicial de planejamento até a implementação das classes e funcionalidades, o nosso foco principal foi garantir uma arquitetura sólida, coesa e flexível, alinhada com os ideais da POO.

Ao adotarmos uma abordagem baseada em herança e encapsulamento, conseguimos criar uma hierarquia de classes bem estruturada, permitindo uma fácil extensibilidade e manutenção do código. A utilização de classes abstratas como Activity e User proporcionou uma organização eficiente dos dados e comportamentos comuns, enquanto as subclasses especializadas permitiram uma representação mais precisa e específica das diferentes atividades e tipos de usuários.

Além disso, a implementação de métodos de validação e verificação de dados contribuiu para a robustez e confiabilidade da aplicação, garantindo que apenas dados válidos fossem aceitos e processados.

Destacamos também o sucesso em implementar uma interface de usuário interativa através da classe Menu, proporcionando uma experiência intuitiva e amigável para os utilizadores interagirem com a aplicação.

Em conclusão, estamos satisfeitos com o resultado final deste projeto, que demonstra o nosso compromisso em aplicar os princípios e práticas da POO de forma eficaz e aprimorar as nossas habilidades de desenvolvimento de software. Esta aplicação não só atende aos requisitos propostos, mas também reflete o nosso comprometimento em criar soluções robustas e de alta qualidade utilizando os conceitos da POO.

# 6.Bibliografia