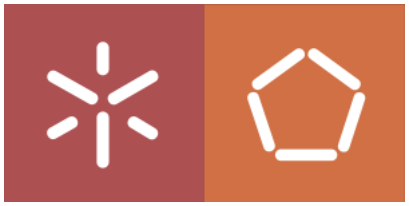


**Universidade do Minho**
Escola de Engenharia
Licenciatura em Engenharia Informática

# Unidade Curricular de Programação Orientada a Objetos

Ano Letivo de 2023/2024

# *Activity Planner*

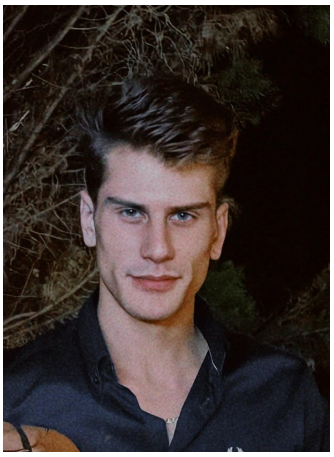
Flávia Alexandra Silva Araújo (A96587)
Miguel Torres Carvalho (A95485)

11 de maio de 2024

POO

**Equipa de Trabalho:**

**Flávia Alexandra Silva Araújo (A96587)**



**Miguel Torres Carvalho (A95485)**

# Resumo

No âmbito da Unidade Curricular Programação Orientada a Objetos, foi-nos proposto o desenvolvimento de uma aplicação de gestão de atividades físicas, à qual desginamos de *Activity Planner*.

A aplicação desenvolvida permite a gestão de utilizadores, atividades, planos de treino, simulação de atividades e visualização de estatísticas. A aplicação foi desenvolvida em *Java*, utilizando o paradigma de programação orientada a objetos aprendido nas aulas.

Neste relatório, é apresentada a arquitetura de classes da aplicação, bem como as funcionalidades implementadas nesta e a forma como as mesmas foram desenvolvidas.

# Índice

# Índice de Figuras

# 1 Arquitetura de Classes

## 1.1 Diagrama de Classes Simplificado

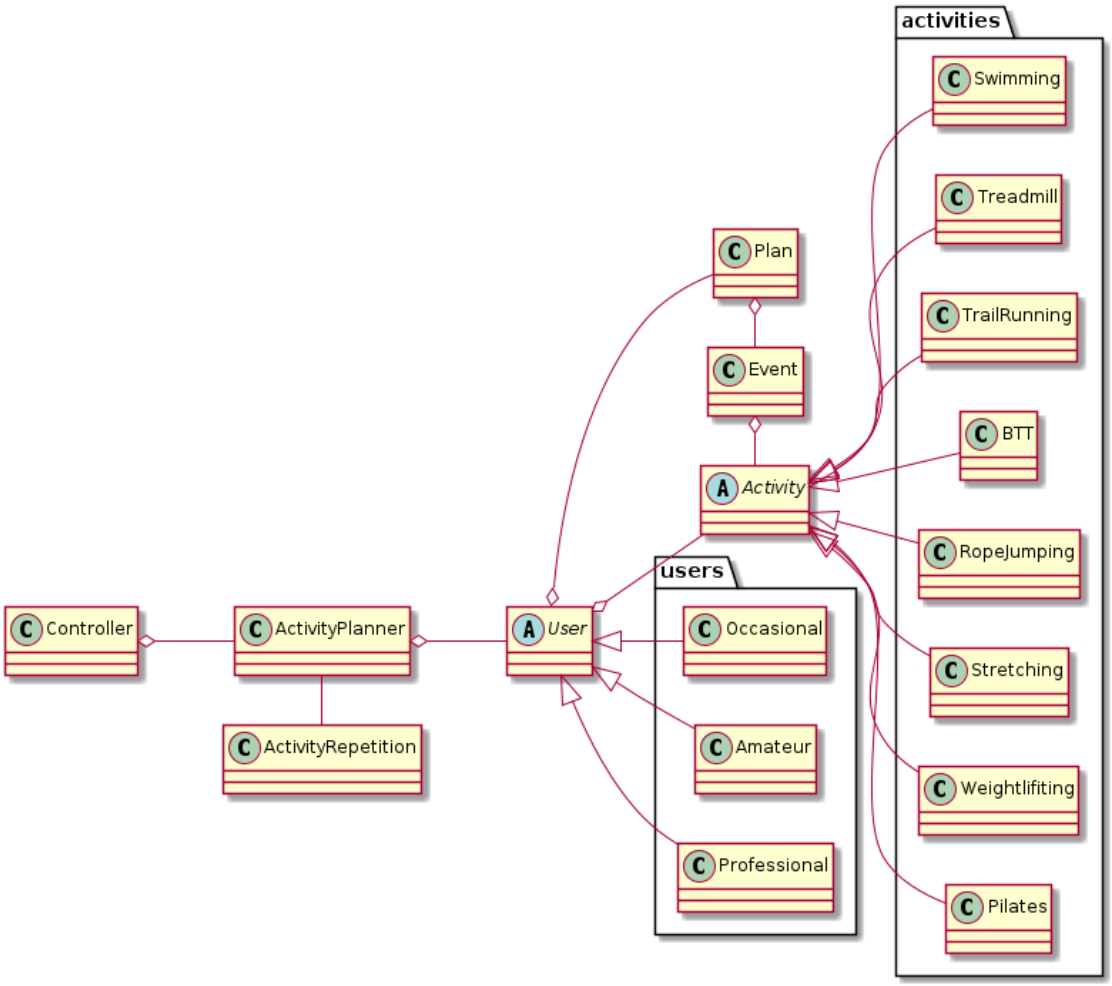



Figura 1.1: Diagrama de Classes Simplificado

## 1.2 Diagrama de Classes

Para uma mehor visiualização da arquitetura de classes da aplicação, apresentada na seguinte figura *1.2*, consulte o diagrama de classes em formato *pdf* em diagram.pdf.

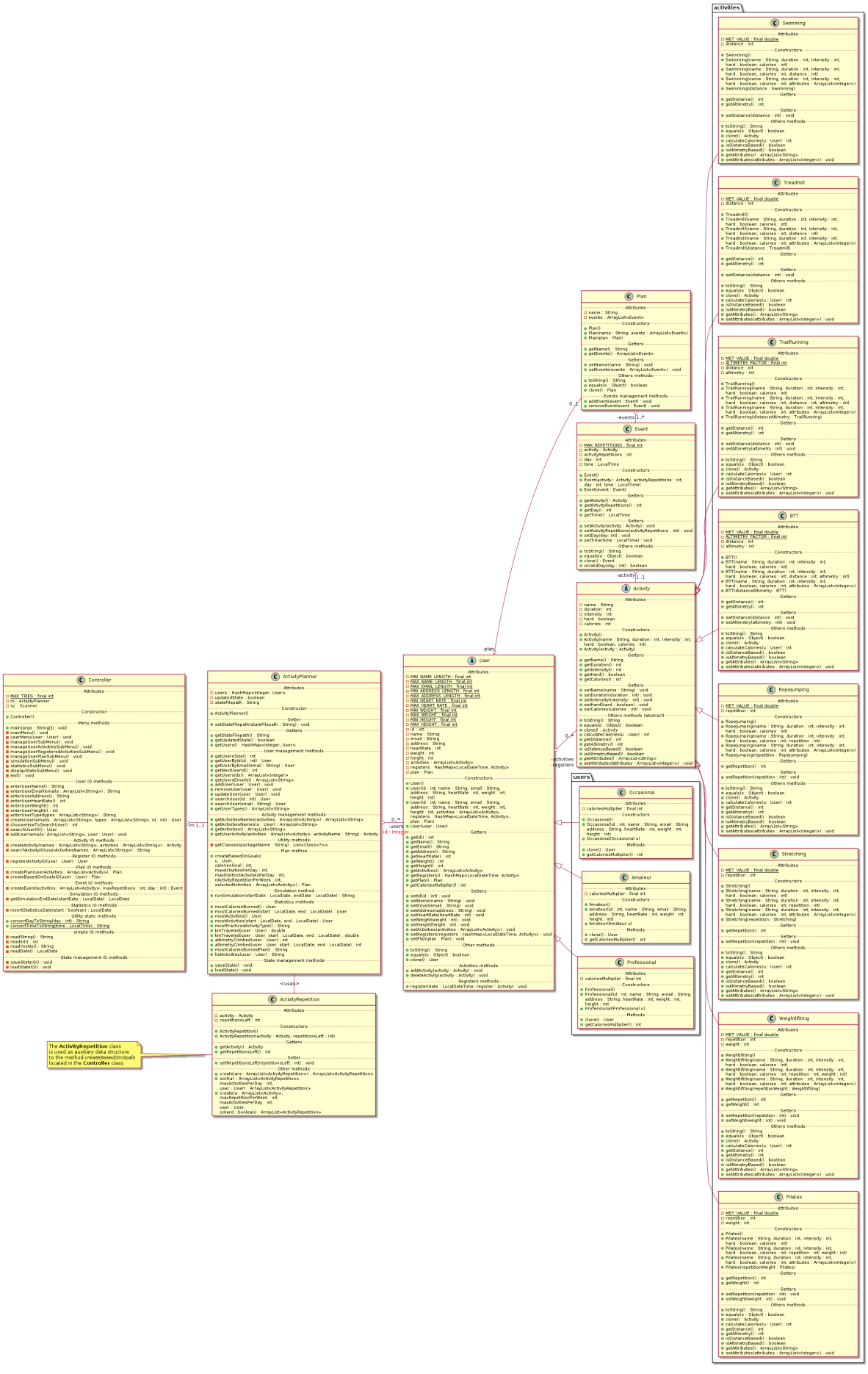



Figura 1.2: Diagrama de Classes

## 1.3 Diagrama de Exceções

Foram utilizadas as seguintes exceções em certas componentes da aplicação:

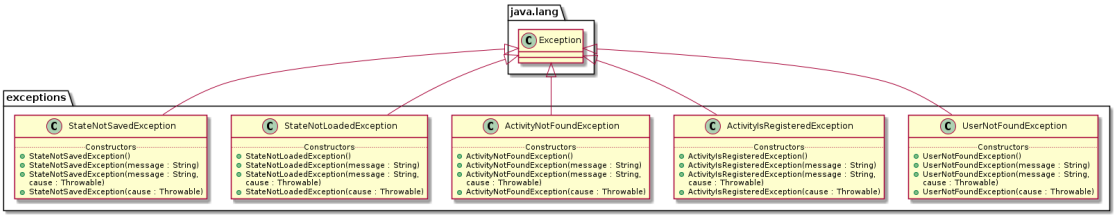



Figura 1.3: Diagrama de Exceções

Consulte a secção *Package exceptions* para mais detalhes sobre as exceções utilizadas.

## 1.4 Classe *ActivityPlanner*

A classe *ActivityPlanner* funciona como o modelo e *facade* da aplicação, sendo responsável por guardar e carregar o estado da aplicação, bem como por fornecer métodos para a manipulação e acesso deste estado.

Por conseguinte, a *ActivityPlanner* é constituída pelos seguintes atributos:

- ***users : HashMap<Integer, User>*** - Lista de utilizadores carregados;
- ***updatedState : boolean*** - Indica se o estado da aplicação foi alterado;
- ***defaultStateFilepath : String*** - Caminho para o ficheiro binário que contém o estado da aplicação.

Providencia métodos para a gestão de utilizadores, acesso ao nome de atividades, simulação de atividades e cálculo de estatísticas, assim como para a salvaguarda do estado da aplicação e métodos que retornam a lista das subclasses de *User* e *Activity*, utilizados para a listagem de utilizadores e atividades oferecidas pela aplicação, ao criar um novo utilizador ou atividade iterativamente, na classe *Controller*.

## 1.5 Classe *Controller*

A classe *Controller* é a classe responsável pela execução do programa - faz o *parsing* dos argumentos da linha de comandos, carrega e salva o estado da aplicação através da classe *ActivityPlanner*, e executa o menu principal ou da perspetiva de um utilizador, dependendo dos argumentos passados.

É nesta classe que ocorrem as interações com o utilizador, através de menus interativos, e a execução das funcionalidades da aplicação.

Foram desenvolvidos os seguintes menus interativos, com os respetivos submenus e opções:

- Menu principal da aplicação:
  - Adicionar, editar, remover e visualizar utilizadores;
  - Adicionar, remover e visualizar atividades de interesse;
  - Registar e visualizar atividades completas;
  - Adicionar, remover e visualizar plano de treino;
  - Simular atividades;
  - Visualizar estatísticas;
  - Carregar e guardar o estado da aplicação;
  - Sair da aplicação.
- Menu com a perspetiva de um utilizador:
  - Visualizar e editar o perfil do utilizador;
  - Adicionar, remover e visualizar atividades de interesse;
  - Registar e visualizar atividades completas;
  - Adicionar, remover e visualizar plano de treino;
  - Visualizar estatísticas;
  - Carregar e guardar o estado da aplicação;
  - Sair da aplicação.

A classe *Controller* funciona como uma *interface* entre o utilizador e a aplicação, sendo responsável por chamar os métodos da classe *ActivityPlanner* e por apresentar os resultados.

No sistema MVC esta classe representa o *controller* assim como o *view*, uma vez que é responsável por apresentar os menus interativos, fazendo as respetivas execuções através dos métodos da classe *ActivityPlanner*.

## 1.6 Classe *User*

A superclasse *User* representa um utilizador da aplicação, sendo constituída pelos seguintes atributos:

- ***id : int*** - Identificador único do utilizador;
- ***name : String*** - Nome completo do utilizador;
- ***email : String*** - Endereço de correio eletrónico do utilizador, sendo este único em relação a todos os utilizadores da aplicação;
- ***address : String*** - Morada do utilizador;
- ***heartRate : int*** - Frequência cardíaca em repouso do utilizador, em batimentos por minuto;
- ***weight : int*** - Peso do utilizador, em kilogramas;
- ***height : int*** - Altura do utilizador, em centímetros;
- ***activities : ArrayList<Activity>*** - Conjunto de atividades de interesse de um utilizador, utlizada para a simplificação de um registo da atividade praticada ou da criação de um plano de treino;
- ***registers : HashMap<LocalDateTime, Activity>*** - Conjunto de actividades praticadas por um utilizador, com a respetiva data e hora de prática, atividade praticada e consumo calórico;
- ***plan : Plan*** - Plano de treino semanal de um utilizador.

Os atributos *weight*, *height* e *heartRate* são utilizados para o cálculo do consumo calórico de uma atividade, sendo estes parâmetros mutáveis, uma vez que podem ser alterados ao longo do tempo.

Para além dos vários construtores, dos métodos tradicionais de acesso e modificação dos atributos (*getters* e *setters*), métodos para a cópia profunda dos objetos desta classe, conversão para *String* e igualdade entre objetos desta classe, foram implementados métodos para a adição/remoção de uma atividade de interesse, bem como métodos para o registo de uma atividade por parte de um utlizador.

A classe *User* foi desenvolvida de forma a permitir a extensão desta, através da implementação de novos tipos de utilizadores. Assim, foi definida uma hierarquia de classes, onde a classe *User* é a superclasse e as suas subclasses são *Occasional*, *Amateur* e *Professional*.

A aplicação foi desenvolvida de forma a permitir a adição de novos tipos de utilizadores, através da criação de uma nova subclasse de *User* com as características específicas deste tipo de utilizador, sem que fosse necessário alterar a classe *User* e outras componentes da aplicação, garantindo, deste modo, a modularidade, manutenção e extensabilidade da superclasse *User* e das suas subclasses.

## 1.7 Classe *Activity*

A superclasse *Activity* representa uma atividade física, sendo constituída pelos seguintes atributos:

- ***name : String*** - Nome da atividade;
- ***duration : int*** - Duração da atividade, em minutos;
- ***intensity : int*** - Intensidade da atividade (um valor de 1 a 100 para representar um valor percentual);
- ***hard : boolean*** - Indica, através de um valor booleano, se a atividade é considerada *hard* pelo utilizador;
- ***calories : int*** - Número de calorias queimadas durante a atividade. O registo deste atributo deve-se à mutação dos parâmetros do utilizador para o cálculo do consumo calórico, desde a frequência cardíaca, o peso, bem como a altura deste. Assim, este atributo foi utilizado de forma a garantir a salvaguarda do valor de calorias queimadas num registo de uma atividade.

A classe *Activity*, assim como a classe *User*, foi desenvolvida de forma a permitir a extensão desta, através da implementação de novos tipos de atividades, sem que fosse necessário alterar a classe *Activity* e outras componentes da aplicação.

As subclasses de *Activity* contêm atributos específicos que caracterizam a atividade em questão, como por exemple, a distância percorrida, o número de repetições, o peso levantado, entre outros.

Foram implementadas as seguintes subclasses de *Activity*:

- ***Swimming*** - Atividade de natação, compostas pelos atributos *distance : int*;
- ***Treadmill*** - Atividade de corrida em passadeira, compostas pelos atributos *distance : int*;
- ***TrailRunning*** - Atividade de corrida em trilhos, compostas pelos atributos *distance : int* e *altimetry : int*;
- ***BTT*** - Atividade de BTT, compostas pelos atributos *distance : int* e *altimetry : int*;
- ***RopeJumping*** - Atividade de saltar à corda, compostas pelos atributos *repetitions : int*;
- ***Stretching*** - Atividade de alongamentos, compostas pelos atributos *repetitions : int*;
- ***Weightlifting*** - Atividade de levantamento de pesos, compostas pelos atributos *repe-*

*titions : int* e *weight : int*;

- ***Pilates*** - Atividade de pilates, compostas pelos atributos *repetitions : int* e *weight : int*;

Para cada uma destas subclasses é definido o atributo MET (*Metabolic Equivalent of Task*) da atividade, que é utilizado para o cálculo do consumo calórico da atividade, este também definido como um método abstrato na superclasse *Activity*.

Nas subclasses de *Activity* foram implementados métodos para obter e definir os atributos específicos da atividade, estes permitem a criação destas atividades de forma iterativa, através da classe *Controller*.

A classe *Activity* foi desenvolvida tirando proveito do conceito de polimorfismo, permitindo a criação de atividades de forma genérica, através da superclasse *Activity*, e a sua manipulação de forma específica, através dos atributos únicos das respetivas subclasses.

## 1.8 Classe *Event*

A classe *Event* representa um evento correspondente a um plano de treino, sendo constituída pelos seguintes atributos:

- ***activity : Activity*** - Atividade praticada no evento;
- ***activityRepetitions : int*** - Número de vezes que a atividade será praticada;
- ***day : int*** - Dia da semana do evento, onde 1 corresponde a domingo e 7 a sábado;
- ***time : LocalTime*** - Hora do evento.

## 1.9 Classe *Plan*

A classe *Plan* representa um plano de treino, por definição, semanal, que um utilizador pode criar, vizualizar e remover.

Esta é composta pelos seguintes atributos:

- ***name : String*** - Nome do plano de treino;
- ***events : ArrayList<Event>*** - Lista de eventos que compõem o plano de treino.

Os planos de treino têm uma duração semanal, sendo compostos por eventos, que representam as atividades a serem praticadas em cada dia da semana.

As funcionalidades proporcionadas por esta classe, bem como a classe *Event*, serão detalhadas no subcapítulo *Gestão de Plano de Treino*.

## 1.10 Classe *ActivityRepetition*

A classe *ActivityRepetition* é utlizada como um objeto auxiliar para a criação de um plano de treino baseado nos objetivos de um utilizador.

O processo de geração de planos de treino baseados em objetivos será aprofundado no subcapítulo *Gestão de Plano de Treino*, a qual será detalhada a forma como esta classe é utilizada.

## 1.11 Package *exceptions*

Foi também criado um *package exceptions*, onde foram definidas as seguintes exceções:

- ***ActivityIsRegisteredException*** - Exceção lançada quando uma atividade com o mesmo nome já foi registada;
- ***ActivityNotFoundException*** - Exceção lançada quando uma atividade não é encontrada;
- ***UserNotFoundException*** - Exceção lançada quando um utilizador não é encontrado;
- ***StateNotSavedException*** - Exceção lançada quando o estado da aplicação não pode ser guardado;
- ***StateNotLoadedException*** - Exceção lançada quando o estado da aplicação não pode ser carregado;

Estas exceções foram criadas de forma a garantir a robustez da aplicação, permitindo a deteção de situações de erro e a sua correta gestão.

# 2 Descrição de Funcionalidades da Aplicação

```
$ java src.Controller
Welcome to Activity Planner!

[Main menu] Please select an option:
(1) Manage users (create, delete, edit, view)
(2) Manage user activities (create, delete, view)
(3) Manage user registered activities (register, view)
(4) Manage user plan (create, delete, view)
(5) Start a simulation
(6) Statistics menu
(7) Save program state
(8) Load program state
(9) Exit
Option:
```

Figura 2.1: Menu Principal da Aplicação

```
$ java src.Controller --load data/state.ser -u -i 1
Loading state from data/state.ser
State loaded successfully, 5 users loaded.
Welcome to Activity Planner, Miguel Carvalho!

[User menu] Please select an option:
( 1) View your profile
( 2) Edit your profile
( 3) Create an activity
( 4) Delete an activity
( 5) View an activity
( 6) View all activities
( 7) Register activity
( 8) View registered activities
( 9) Create your plan
(10) Create plan based on your goals
(11) Delete your plan
(12) View your plan
(13) Statistics menu
(14) Save program state
(15) Load program state
(16) Exit
Option:
```

Figura 2.2: Menu da Perspetiva de um Utilizador

## 2.1 Gestão de Utilizadores

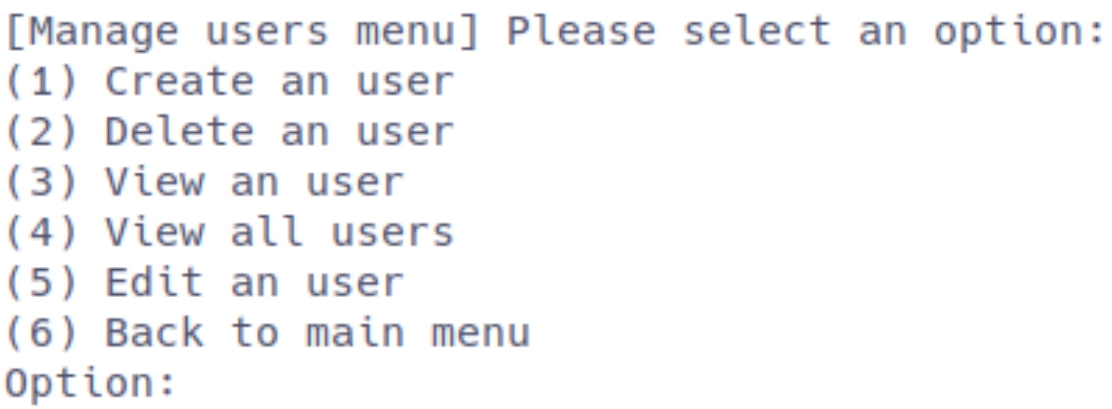



Figura 2.3: Submenu de Gestão de Utilizadores

### 2.1.1 Criar Utilizador

A criação de um utilizador é feita de forma iterativa, em um método do *Controller*, onde são questionados os dados do utilizador a ser criado, como o nome, o email, a morada, a frequência cardíaca em repouso, o peso, a altura e o tipo de utilizador.

De seguida é utilizado o construtor parametrizado da classe *User* para criar o novo utilizador, e este é adicionado à lista de utilizadores da aplicação, através do método *addUser* da classe *ActivityPlanner*.

```
[Manage users menu] Please select an option:
(1) Create an user
(2) Delete an user
(3) View an user
(4) View all users
(5) Edit an user
(6) Back to main menu
Option: 1
Enter user full name: Jorge Teste
Enter the user email: jorge@example.com
Enter the user address: Rua do Jorge
Enter the user resting heart rate (in BPM): 96
Enter the user weight (in kg): 80
Enter the user height (in cm): 185
Possible user types:
  1 - Professional
  2 - Occasional
  3 - Amateur
Enter the user type: 3
User created successfully. (ID: 1)
```

Figura 2.4: Criação de Utilizador

### 2.1.2 Remover Utilizador

É possível remover um utilizador da aplicação, através do método *removeUser*, esta usa um método auxiliar para procurar pelo utilizador a ser removido, tanto pelo seu id como pelo seu email, e remove o utilizador da lista de utilizadores da aplicação através do método *removeUser* do *facade* da aplicação, a classe *ActivityPlanner*.

```
Chose how to search for an user:
(1) By ID
(2) By email
Option: 2
Enter the user email: jorge@example.com
Selected user: Jorge Teste
User deleted successfully.
```

Figura 2.5: Remoção de Utilizador

### 2.1.3 Visualizar Utilizador

Em primeiro lugar, é feita a seleção do utilizador a ser visualizado, através do seu id ou email, similar ao processo de remoção de um utilizador.

De seguida é apresentado o perfil do utilizador, com todos os seus dados.

```
Chose how to search for an user:
(1) By ID
(2) By email
Option: 1
Enter the user ID: 1
Selected user: Jorge Teste
User {
  id: 1,
  name: Jorge Teste,
  email: jorge@example.com,
  address: Rua do Jorge,
  heartRate: 96 BPM,
  weight: 80 kg,
  height: 185 cm,
  type: Amateur,
  caloriesMultiplier: 80,
  activities: [],
  registers: [],
  no training plan
}
```

Figura 2.6: Visualização de Utilizador

### 2.1.4 Visualizar Utilizadores

Nesta opção são apresentados todos os utilizadores da aplicação, atrvés da iteração sobre a lista de utilizadores da aplicação, e a apresentação dos dados de cada utilizador.

```
[Manage users menu] Please select an option:
(1) Create an user
(2) Delete an user
(3) View an user
(4) View all users
(5) Edit an user
(6) Back to main menu
Option: 4
User {
  id: 1,
  name: Miguel Carvalho,
  email: miguel@example.com,
  address: Rua das Oliveiras, 13,
  heartRate: 80 BPM,
  weight: 60 kg,
  height: 180 cm,
  type: Occasional,
  caloriesMultiplier: 70,
  activities: [],
  registers: [],
  no training plan
}
User {
  id: 2,
  name: Flávia Araújo,
  email: flavia@example.com,
  address: Rua das Gatas, 707,
  heartRate: 70 BPM,
  weight: 60 kg,
  height: 170 cm,
  type: Professional,
  caloriesMultiplier: 90,
  activities: [
    Pilates,
    Corrida na Passadeira
  ],
  registers: [],
  no training plan
}
```

Figura 2.7: Visualização de Utilizadores

### 2.1.5 Editar Utilizador

É permitido pela aplicação a edição do perfil de um utilizador, onde são questionados os dados a serem alterados, e estes são alterados através dos métodos *set* da classe *User*, e de seguida o respectivo utilizador é atualizado na lista de utilizadores da aplicação, através do método *updateUser* da classe *ActivityPlanner*.

```
Chose how to search for an user:
(1) By ID
(2) By email
Option: 1
Enter the user ID: 1
Selected user: Jorge Teste

Chose what to edit:
(1) Name
(2) Email
(3) Address
(4) Heart rate
(5) Weight
(6) Height
(7) Go back
Option: 1
Enter user full name: Jorge Teste da Silva
Name updated successfully.

Chose what to edit:
(1) Name
(2) Email
(3) Address
(4) Heart rate
(5) Weight
(6) Height
(7) Go back
Option: 5
Enter the user weight (in kg): 78
Weight updated successfully.
```

Figura 2.8: Edição de Utilizador

## 2.2 Gestão de Atividades

Antes da aplicação prosseguir com a apresentação do submenu de gestão de atividades de intersse, é feita a seleção do utilizador a quem se as pretende gerir.

```
[Manage user activities menu] Please select an option:
(1) Create an activity
(2) Delete an activity
(3) View an activity
(4) View all activities
(5) Back to main menu
Option:
```

Figura 2.9: Submenu de Gestão de Atividades

### 2.2.1 Adicionar Atividade

A adição de uma atividade de interesse é feita de forma iterativa, em um método do *Controller*, onde são questionados os dados da atividade a ser criada dependendo do tipo de atividade, e esta é adicionada à lista de atividades de interesse do utilizador, e este é atualizado na lista de utilizadores da aplicação.

```
[Manage user activities menu] Please select an option:
(1) Create an activity
(2) Delete an activity
(3) View an activity
(4) View all activities
(5) Back to main menu
Option: 1
Possible activities:
  1 - Pilates
  2 - BTT
  3 - TrailRunning
  4 - Weightlifting
  5 - Treadmill
  6 - Swimming
  7 - Stretching
  8 - RopeJumping
Enter the activity: 4
Enter the name of the activity: Levantamento de Pesos - Pernas
Enter the duration of the activity in minutes: 90
Enter the intensity of the activity (1-100): 75
Is the activity hard? [y/n]: n
Enter the repetition: 48
Enter the weight: 100
Activity created successfully.
```

Figura 2.10: Adição de Atividade

### 2.2.2 Remover Atividade

Nesta opção são apresentadas todas os nomes das atividades de interesse de um utilizador, de seguida o utilizador seleciona a atividade que pretende remover, esta é removida da lista de atividades de interesse do utilizador, e o respetivo utilizador é atualizado na lista de utilizadores da aplicação.

```
[Manage user activities menu] Please select an option:
(1) Create an activity
(2) Delete an activity
(3) View an activity
(4) View all activities
(5) Back to main menu
Option: 2
User activities:
  -> Levantamento de Pesos - Pernas
Enter the name of the activity: Levantamento de Pesos - Pernas
Activity deleted successfully.
```

Figura 2.11: Remoção de Atividade

### 2.2.3 Visualizar Atividade

Nesta opção são apresentadas todos os nomes das atividades de interesse de um utilizador, de seguida o utilizador seleciona a atividade que pretende visualizar, e são apresentados os seus detalhes.

```
[Manage user activities menu] Please select an option:
(1) Create an activity
(2) Delete an activity
(3) View an activity
(4) View all activities
(5) Back to main menu
Option: 3
User activities:
  -> Levantamento de Pesos - Pernas
Enter the name of the activity: Levantamento de Pesos - Pernas
Activity {
  Name: Levantamento de Pesos - Pernas,
  Duration: 90 minutes,
  Intensity: 75,
  Repetition: 48 times
  Weight: 100 kg
  Hard: false,
}
```

Figura 2.12: Visualização de Atividade

### 2.2.4 Visualizar Atividades

As atividades de interesse de um utilizador são apresentadas de forma iterativa, através da iteração sobre a lista de atividades de interesse do utilizador, e a apresentação dos dados de cada atividade.

```
[Manage user activities menu] Please select an option:
(1) Create an activity
(2) Delete an activity
(3) View an activity
(4) View all activities
(5) Back to main menu
Option: 4
Activity {
  Name: Flexões,
  Duration: 120 minutes,
  Intensity: 70,
  Repetition: 100 times
  Hard: false,
}
Activity {
  Name: Levantamento de Peso,
  Duration: 80 minutes,
  Intensity: 80,
  Repetition: 30 times
  Weight: 80 kg
  Hard: false,
}
Activity {
  Name: Corrida no Monte,
  Duration: 60 minutes,
  Intensity: 100,
  Distance: 4000 meters,
  Altimetry: 600 meters
  Hard: true,
}
```

Figura 2.13: Visualização de Atividades

## 2.3 Registo e Visualização de Atividades Completas

```
Chose how to search for an user:
(1) By ID
(2) By email
Option: 2
Enter the user email: flavia@example.com
Selected user: Flávia Araújo

[Manage user registered activities menu] Please select an option:
(1) Register an activity
(2) View registered activities
(3) Back to main menu
Option:
```

Figura 2.14: Submenu de Registo de Atividades

### 2.3.1 Registar Atividade

Em primeiro lugar, é feita a seleção do utilizador que praticou a atividade, de seguida é dada opção de resgitar a atividade de interesse do utilizador ou de criar uma nova atividade de interesse, caso esta não exista.

Após a seleção da atividade, é questionada a data e hora da prática da atividade, havendo a possibilidade de escolher a data e hora atuais.

Com os dados recolhidos, é criado um novo registo de atividade, com a data e hora da prática, a atividade praticada, e o consumo calórico da atividade, e este é adicionado ao registo de atividades do utilizador, e este é atualizado na lista de utilizadores da aplicação.

```
[Manage user registered activities menu] Please select an option:
(1) Register an activity
(2) View registered activities
(3) Back to main menu
Option: 1
Choose an option:
(1) Register an existing activity
(2) Register a new activity
Option: 1
User activities:
  -> Pilates
  -> Corrida na Passadeira
Enter the name of the activity: Pilates

Registering activity: Pilates
(1) Enter date manually
(2) Use current date
Option: 1
Enter date (yyyy-mm-dd): 2024-05-12
Enter time (hh:mm): 14:00
Date must be less than current date.
Enter date (yyyy-mm-dd): 2024-05-09
Enter time (hh:mm): 14:00
Activity registered successfully:
Pilates on 2024-05-09 14:00
376 calories burned.
```

Figura 2.15: Registo de Atividade

### 2.3.2 Visualizar Registos de Atividades

Nesta opção são apresentados todos os registos de atividades de um utilizador, através da iteração sobre o registo de atividades do utilizador, e a apresentação dos dados de cada registo.

```
[Manage user registered activities menu] Please select an option:
(1) Register an activity
(2) View registered activities
(3) Back to main menu
Option: 2
2024-05-09 14:00: {
Activity {
  Name: Pilates,
  Duration: 45 minutes,
  Intensity: 65,
  Repetition: 20 times
  Weight: 5 kg
  Hard: false,
  Calories: 376,
}
}
```

Figura 2.16: Visualização de Registos de Atividades

## 2.4 Gestão de Plano de Treino

```
[Main menu] Please select an option:
(1) Manage users (create, delete, edit, view)
(2) Manage user activities (create, delete, view)
(3) Manage user registered activities (register, view)
(4) Manage user plan (create, delete, view)
(5) Start a simulation
(6) Statistics menu
(7) Save program state
(8) Load program state
(9) Exit
Option: 4

Chose how to search for an user:
(1) By ID
(2) By email
Option: 1
Enter the user ID: 5
Selected user: Maria Magia

[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option:
```

Figura 2.17: Submenu de Gestão de Plano de Treino

### 2.4.1 Adicionar Plano de Treino Interativamente

A aplicação verifica se o utlizador já tinha um plano de treino, e caso tenha, é questionado se este pretende substituir o plano atual.

Inicialmente é perguntado ao utilizador o nome do plano de treino a ser criado, de seguida para cada dia da semana é questionado o número de atividades a serem praticadas, e são selecionadas as atividades a serem praticadas, a hora e o número de repetições destas.

A aplicação garante que o utlizador escolheu pelo menos uma atividade em toda a semana.

Por fim é apresentada uma mensagem de sucesso, é criado um novo plano de treino, com o nome e os respetivos eventos, com o dia da semana e a hora da prática da atividade, e este é adicionado ao plano de treino do utilizador, e este é atualizado na lista de utilizadores da aplicação.

```
[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option: 1
Enter the name of the plan: Treino para Maratona
How many activities do you want on Sunday? (0-3): 0
How many activities do you want on Monday? (0-3): 1
Event number 1
User activities:
  -> Maratona
  -> Natação
  -> Corrida na Passadeira
  -> Bicicleta Elétrica
Enter the name of the activity: Bicicleta Elétrica
Enter the time of the event (HH:mm): 08:00
Event scheduled successfully.
How many activities do you want on Tuesday? (0-3): 0
```

Figura 2.18: Adição de Plano de Treino Interativamente - Parte 1

```
How many activities do you want on Wednesday? (0-3): 2
Event number 1
User activities:
  -> Maratona
  -> Natação
  -> Corrida na Passadeira
  -> Bicicleta Elétrica
Enter the name of the activity: Natação
Enter the number of times you want to repeat the activity (1-2): 1
Enter the time of the event (HH:mm): 08:00
Event scheduled successfully.
Event number 2
User activities:
  -> Maratona
  -> Natação
  -> Corrida na Passadeira
  -> Bicicleta Elétrica
Enter the name of the activity: Corrida na Passadeira
Enter the time of the event (HH:mm): 10:00
Event scheduled successfully.
How many activities do you want on Thursday? (0-3): 0
How many activities do you want on Friday? (0-3): 3
Event number 1
User activities:
  -> Maratona
  -> Natação
  -> Corrida na Passadeira
  -> Bicicleta Elétrica
Enter the name of the activity: Corrida na Passadeira
Enter the number of times you want to repeat the activity (1-3): 1
Enter the time of the event (HH:mm): 10:00
Event scheduled successfully.
```

Figura 2.19: Adição de Plano de Treino Interativamente - Parte 2

```
Event number 2
User activities:
  -> Maratona
  -> Natação
  -> Corrida na Passadeira
  -> Bicicleta Elétrica
Enter the name of the activity: Bicicleta Elétrica
Enter the number of times you want to repeat the activity (1-2): 2
Enter the time of the event (HH:mm): 16:35
Event scheduled successfully.
How many activities do you want on Saturday? (0-3): 0
Plan created successfully.
```

Figura 2.20: Adição de Plano de Treino Interativamente - Parte 3

### 2.4.2 Adicionar Plano de Treino Baseado em Objetivos

Como na opção anterior, a aplicação verifica se o utlizador já tinha um plano de treino, e caso tenha, é questionado se este pretende substituir o plano atual.

De seguida é questionado as seguintes restrições à criação do plano de treino:

- *Calorias mínimas* - Calorias mínimas a serem queimadas no plano de treino;
- *Número máximo de atividades por dia* - Número máximo de atividades a serem praticadas por dia, máximo três;
- *Número máximo de atividades distintas por dia* - Número máximo de atividades distintas a serem praticadas por dia, máximo três;
- *Número máximo de repetições por atividade* - Número máximo de repetições a serem praticadas por atividade por semana;
- *Ativadades de interesse* - Atividades de interesse selecionadas pelo utilizador, que podem ser praticadas no plano de treino;

Para além destas restrições está implícito que, se o utilizador seleciona uma atividade do tipo *hard*, estas não podem ser praticadas no mesmo dia, nem em dias consecutivos.

Para a criação do plano de treino baseado em objetivos, é utilizado um algoritmo de maximização de calorias, para tal é utlizada a classe *ActivityRepetition* como um objeto auxiliar, que é instanciada com cada atividade selecionada e o número de repetições de uma atividade por semana.

Estes objetos são guardados numa lista, que depois é ordenada de forma descrescente, em relação ao número de calorias queimadas multiplicadas pelo número de repetições possíveis por dia, ou seja o valor mínimo entre o número de repetições por semana restantes da ativadade em questão e o número de repetições por dia restantes de atividades em geral. Se uma atividade é do tipo *hard*, o valor máximo de repetições por dia é um automaticamente.

De seguida, para cada dia da semana, é selecionada a atividade (com o seu respetivo número de repetições restantes) com o maior valor de calorias, e se possível - não ultrapassando o número máximo de atividades por dia nem o número máximo de atividades distintas por dia -, são adicionadas mais atividades para esse dia, até que esta condição não seja satisfeita passando para o próximo dia.

É também registado se um dia têm uma atividade do tipo *hard*, para que não seja praticada uma atividade deste tipo no dia seguinte (nem no próprio dia) sendo o número máximo de atividades do tipo *hard* por semana limitado a três (exemplo: domingo, terça-feira, quinta-feira).

Em caso de empante entre uma atividade do tipo *hard* e uma ou mais atividades normais, é

sempre selecionada a atividade do tipo *hard*, com intuito de maximizar o número de calorias queimadas, visto que estas atividades de tipo hard são restritas a três por semana é necessário, nestes casos, colocar-las em primeiro lugar, caso contrário pode haver a possibilidade de não serem praticadas no resto da semana.

No fim do algoritmo, este verifica se o número de calorias queimadas é superior ao número de calorias mínimas desejadas, e se tal acontecer o plano é criado, adicionado e apresentado ao utlizador, caso contrário é apresentada uma mensagem que afirma a impossibilidade de criar um plano de treino com as restrições impostas.

```
[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option: 2
Enter caloric goal per week: 20000
Enter max activities per day: 4
Invalid number of activities. Please enter a number between 1 and 3.
Enter max activities per day: 2
Enter max distinct activities per day: 1
Enter number of repetitions of activities per week: 3
Enter number of your activities to select (1-4): 4
Plan based on goals
Activity: Maratona
Repetitions: 1
Day: Sunday
Time: 08:00
Expected calories: 6157

Activity: Bicicleta Elétrica
Repetitions: 2
Day: Monday
Time: 08:00
Expected calories: 1552

Activity: Maratona
Repetitions: 1
Day: Tuesday
Time: 08:00
Expected calories: 6157
```

Figura 2.21: Adição de Plano de Treino Baseado em Objetivos

```
Activity: Bicicleta Elétrica
Repetitions: 1
Day: Wednesday
Time: 08:00
Expected calories: 776

Activity: Maratona
Repetitions: 1
Day: Thursday
Time: 08:00
Expected calories: 6157

Activity: Corrida na Passadeira
Repetitions: 2
Day: Friday
Time: 08:00
Expected calories: 918

Activity: Corrida na Passadeira
Repetitions: 1
Day: Saturday
Time: 08:00
Expected calories: 459

Total activities: 9 / 12
Calories: 22176 / 20000
Plan generated successfully.
```

Figura 2.22: Adição de Plano de Treino Baseado em Objetivos

### 2.4.3 Remover Plano de Treino

Nesta opção é removido o plano de treino de um utilizador se tal existir, deseguida o utlizador é atualizado na lista de utilizadores da aplicação.

A remoção de um plano é feita através do método *user.setPlan(null)*, para tal atualizados os outros métodos de forma a garantir que não hajam erros na aplicação, como por exemplo, a visualização de um plano de treino, ou o clone de um plano de treino não existente.

```
[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option: 3
Plan deleted successfully.

[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option: 3
The selected user has no plan.
```

Figura 2.23: Remoção de Plano de Treino

### 2.4.4 Visualizar Plano de Treino

Nesta opção são apresentados todos os eventos de um plano de treino de um utilizador, através da iteração sobre a lista de eventos do plano de treino do utilizador, e a apresentação dos dados de cada evento.

```
[Manage user plan menu] Please select an option:
(1) Create plan interactively
(2) Create plan based on user goals
(3) Delete plan
(4) View plan
(5) Back to main menu
Option: 4
Plan: Treino para Maratona
Events:
Activity: Bicicleta Elétrica
Repetitions: 1
Day: Monday
Time: 08:00
Activity: Natação
Repetitions: 1
Day: Wednesday
Time: 08:00
Activity: Corrida na Passadeira
Repetitions: 1
Day: Wednesday
Time: 10:00
Activity: Corrida na Passadeira
Repetitions: 1
Day: Friday
Time: 10:00
Activity: Bicicleta Elétrica
Repetitions: 2
Day: Friday
Time: 16:35
```

Figura 2.24: Visualização de Plano de Treino

## 2.5 Simulação

TODO

## 2.6 Estatísticas

TODO

## 2.7 Salvaguarda do Estado da Aplicação

Para garantir que o estado da aplicação é preservado entre execuções, esta permite guardar e carregar o estado atual através de um ficheiro binário. As opcões de guardar e carregar o estado do programa estão disponíveis no menu principal da aplicação. Adicionalmente o ficheiro binário pode ser carregado diretamente no início da execução do programa através da passagem da localização deste na linha de comandos. Este ficheiro, por definição, é guardado na diretoria *data* e tem o nome *state.ser*, havendo a opção de carregar diferentes estados através da funcionalidade da linha de comandos supramencionada.

Para a implementação desta funcionalidade foram definidos dois métodos na classe *Main*:

- *saveState* - Método que guarda o estado atual da aplicação num ficheiro binário, passado como argumento. Este método deteta se alguma mudança foi feita no estado do programa antes de a guardar, de forma evitar salvar o mesmo estado.
- *loadState* - Método que carrega o estado da aplicação a partir de um ficheiro binário, passado como argumento.

Como os objetos da classe *User* contêm referências para todos os objetos relevantes de serem guardados/carregados - lista de Atividades, conjunto de registos de atividades, Plano de treino semanal com os respetivos Eventos - foi necessário garantir que estes e a própria classe referente ao Utilizador implementassem a interface *Serializable*, de forma a que fossem possíveis de ser guardados, e futuramente carregados, num ficheiro binário.

A aplicação também dispõe de uma capacidade inteligente de detetar mudanças no seu estado, através do atributo booleano *updatedState* na classe *Main*, o que permitiu a implementação das seguintes funcionalidades:

- Notificar o utilizador de que o estado atual não foi guardado, caso este tente sair da aplicação, dando a opção de o guardar, caso o utilizador o deseje fazer.
- Notificar o utilizador que, ao carregar um novo estado, o estado atual será perdido, se houver alterações, dando a opção de retornar atrás se este não quiser perder o estado atual.

O valor do atributo *updatedState* é incializado a *false* no ínicio da execução do programa e é alterado para *false* sempre que o estado da aplicação é guardado, no método *saveState*, ou carregado, no método *loadState*, referidos anteriormente.

Este valor booleano é alterado para *true* sempre que o estado da aplicação é alterado, seja através da adição, edição ou remoção de um utilizador, de uma atualização de uma atividade de um utilizador, do registo de uma nova atividade ou da criação/remoção de um plano de treino para um utlizador em específico.

## 2.8 Argumentos de Linha de Comandos

Através do argumento *–help* passado aquando à execução da aplicação, é possível visualizar a lista de argumentos disponíveis para a execução da mesma.

```
Usage: java src.Controller [--load <file>] [--user <--id|--email> <id|email>]
Options:
  -h --help: show this help message
  -l --load: load a program state from file
  -u --user:
select an user by id or email to view his perspective
  -e --email: select an user by email (use with --user option after --load option)
  -i --id: select an user by ID (use with --user option after --load option)
```

Figura 2.25: Argumentos de Linha de Comandos

Como é possível visualizar na figura, uma das possibilidades é a de carregar um estado da aplicação através do argumento *-l* ou *–load*, onde é necessário passar o caminho do ficheiro binário a ser carregado (exemplo: *java src.Controller –load data/state.ser*).

Outra possibilidade disponível é a de carregar a aplicação sobre a perspetiva de um utilizador, através do argumento *-u* ou *–user*. Neste caso, além de passar os argumentos para carregar o estado da aplicação e sobre a perspetiva de um utilizador, é também necessário passar um argumento que identifique o utilizador escolhido, podendo este ser o seu *email* (*-e* ou *–email*, seguido do *email* do utilizador) ou o seu id (*-i* ou *–id*, seguido do seu id) (exemplo: *java src.Controller -l data/state.ser -u -i 1*).

# 3 Conclusão

Em suma, a aplicação *Activity Planner* foi desenvolvida com o intuito de permitir a gestão de atividades físicas, através da gestão de utilizadores, atividades, planos de treino, simulação de atividades e visualização de estatísticas.

Foram implementadas todas as funcionalidades propostas no enunciado, bem como outras funcionalidades adicionais, como a salvaguarda do estado da aplicação e a capacidade de carregar diferentes estados da aplicação através da linha de comandos, usufruindo das metodologias de programação orientada a objetos aprendidas nas aulas - desde a separação da aplicação numa derivação do MVC, hierarquia e polimorfismo de classes, uma arquitetura modular e extensível, com a utilização de exceções para a deteção e gestão de erros, assim como o encapsulamento de dados.