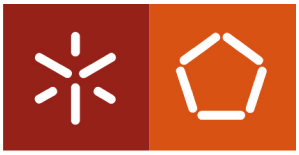

Universidade do Minho

LEI — Licenciatura de Engenharia Informática

UC8204P1 — Programação Orientada a Objectos

FitnessUM

Grupo 14

Rui Camposinhos - a72625, Rui Oliveira - a67661, André Santos - a61778



Braga, 1 de junho de 2014

Conteúdo

1 Introdução **1**
- 1.1 Objectivos . . . 1
- 1.2 Organização do Relatório . . . 2

2 Arquitectura e Descrição da Aplicação **3**
- 2.1 Packages . . . 3
- 2.2 Colections Manager . . . 3
- 2.3 Diagramas de Classes . . . 4
- 2.4 Funcionalidades da Aplicação . . . 4

3 Estruturas de Dados Principais **5**
- 3.1 User class . . . 5
- 3.2 Activity class . . . 6
- 3.3 Record interface . . . 7
- 3.4 Event class . . . 7

4 Cálculo do Consumo de Calorias por Actividade **8**
- 4.1 Cálculo Baseado na Frequência Cardíaca . . . 8
- 4.2 Cálculo Baseado no Tipo de Actividade . . . 10

5 Cálculo da Forma dos Utilizadores **11**

6 Estatísticas dos Utilizadores **12**

7 Simulação de Eventos **14**

8 Conclusões **15**

A Demo da aplicação **17**

Capítulo 1

Introdução

O presente projecto enquadra-se na unidade curricular de Programação Orientada a Objectos do curso de Licenciatura em Engenharia Informática da Universidade do Minho. O projecto pretende implementar uma aplicação, designada *FitnessUM*, para registar e simular actividades desportivas. A aplicação foi desenvolvida na linguagem *java* e pretende simular um ambiente de rede social.

1.1 Objectivos

O presente projeto tem como objectivo principal consolidar e colocar em práctica os conceitos fundamentais transmitidos durante a UC de Programação Orientada a Objectos.

De acordo com o enunciado [1], os principais objectivos definidos para a aplicação *FitnessUM* são os seguintes:

Requisitos básicos:

- Aceder à aplicação utilizando as credenciais (email e password);
- Visualizar a informação das actividades, do próprio e dos amigos, com possibilidade de aceder aos seus detalhes;
- Registar a informação de uma actividade;
- Consultar, por ordem cronológica, e remover actividades;
- Aceder às estatísticas de distância, tempo e calorias gastas;

Requisitos de valorização:

- Definir tipos de recordes e registar os recordes pessoais de cada utilizador;
- Definir eventos e proceder à sua simulação a partir dos administratores da aplicação.

1.2 Organização do Relatório

(...)

Capítulo 2

Arquitectura e Descrição da Aplicação

2.1 Packages

(...) Model–view–controller (MVC) figura 2.1. (...) (...)

De forma a manter uma estrutura de pastas partilhada entre os autores e um controlo de versões eficaz, foi utilizada a ferramenta open source *GIT* (http://git-scm.com/), com repositório privado no bitbucket (https://bitbucket.org/ruiOliveiras94/fitnessum-poo).

falar também do Manager, Dataset, ...

2.2 Colections Manager

(...) @COMENT: OLIVEIRA - work your magic :) (...)

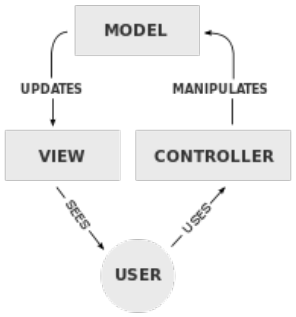

Figura 2.1: Diagrama com a relação típica dos componentes do MVC (ref.:http://en.wikipedia.org/wiki/Model-view-controller).

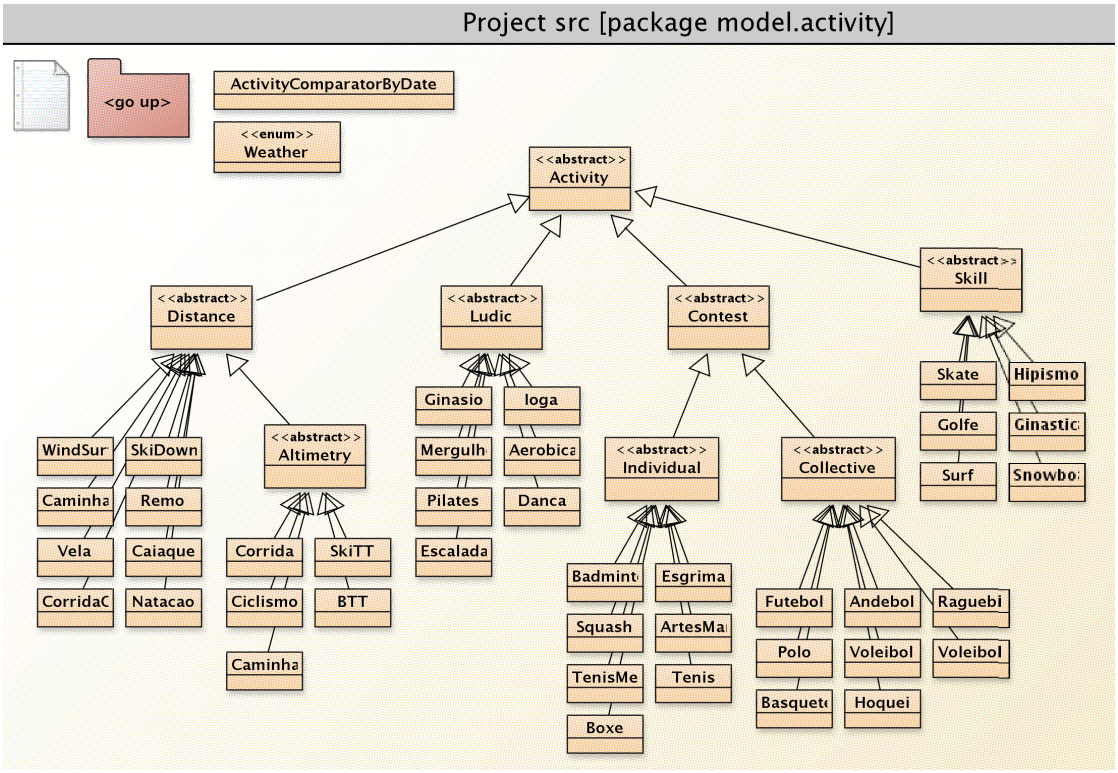

Figura 2.2: Diagrama com dependências da super classe *Activity*.

2.3 Diagramas de Classes

(...) A figura (...) apresenta o grafo de dependências dos vários ficheiros de código, obtido através do *BlueJ*.

COLOCAR TABELA EXCEL DAS ACTIVIDADES?

2.4 Funcionalidades da Aplicação

(...) Lista de features da aplicação. (...)

Capítulo 3

Estruturas de Dados Principais

3.1 User class

As variáveis de instância da classe `User` são apresentadas no excerto de código 3.1. Para as permissões foi usada uma enumeração correspondente aos diferentes tipos de utilizador. Estas permissões são usadas para controlar as funcionalidades a que o utilizador tem acesso na aplicação. Para os recordes foi usado um `Map` de uma classe correspondente a uma modalidade para o conjunto de recordes dessa modalidade. Um recorde é constituído pelo seu tipo, dado por um inteiro e por uma "Actividade modelo", que representa o recorde. Para os amigos e lista de amigos foi utilizado o `Manager`, que é instanciado numa `HashSet` de `String`. Para o conjunto de actividades realizadas, foi também utilizado o Manager instanciado numa árvore (TreeSet), ordenado por meio de um `Comparator`, que compara as actividades por ordem crescente da sua data de realização.

Código 3.1: Variáveis de instância da classe User (src/model/user).

```
1  private String email;
2  private String nome;
3  private String password;
4  private Genero genero;
5  private int altura; /*cm*/
6  private int peso; /*kg*/
7  private GregorianCalendar dataNascimento;
8  private Activity desportoFavorito;
9  private Permissoes permissoes;
10 private Map<Class<?>, HashMap<Integer, Activity>> recordes; /*ver documentacao*/
11 private int fcr; /*frequencia cardiaca em repouso - para calculo das calorias*/
12 private Manager<String> amigos; /*emails amigos:HashSet com chaves para rede social*/
13 private Manager<String> convitesAmigos; /*emails de amigos: convites - HashSet*/
14 private Manager<Activity> actividadesUser; /*Actividades do User:TreeSet -Comp.ByDate*/
```

3.2 Activity class

Na superclasse abstracta `Activity` foram incorporadas todas as variáveis de instância e métodos gerais das actividades, designadamente a data, duração, meteorologia e frequência cardíaca média durante a actividade, caso seja fornecida pelo utilizador. Para as condições meteorológicas foi usada uma enumeração para cobrir as situações mais frequentes em termos de clima. Num nível hierárquico abaixo da superclasse surgem as subclasses `Distance`, `Altimetry`, `Contest`, `Skill` e `Ludic`.

A subclasse `Distance` especializa a superclasse com duas variáveis de instância relativas a actividades que envolvam distâncias, a saber a distância percorrida e a velocidade atingida no percurso. A subclasse `Altimetry` descende da subclasse `Distance`, acrescentando atributos às actividades que além de terem distância têm associada altura, nomeadamente a distância que foi subida e descida, bem como as altitudes máxima e mínima atingidas. A subclasse `Contest` descende da superclasse e representa as actividades em que esteja envolvido algum tipo de disputa/jogo (de equipa ou individual). As suas variáveis de instância representam os pontos obtidos pela equipa adversária e pela própria equipa. A subclasse `Skill` representa as actividades em que a performance é avaliada em termos de uma pontuação, através de uma exibição individual. As variáveis de especialização são a pontuação final obtida e o máximo de pontuação obtida para um determinado movimento. A subclasse `Ludic` representa actividades mais simples e com um carácter mais lúdico. Esta classe não apresenta nenhum especialização relativamente à superclasse. Para além dos métodos abstractos impostos pela superclasse, são também definidos nesta classe os tipos de recordes específicos. Todas as variáveis da superclasse e subclasses referidas são apresentadas no excerto de código 3.2.

Para além dos atributos apresentados, a hierarquia de classes definida implica a herança de alguns métodos abstractos fundamentais para a aplicação. São exemplo disso os método `getIntensidade()` e o `getMET()`. O primeiro devolve o número de minutos diários recomendados para a actividade, para que o utilizador atinga uma forma ideal. O segundo, serve de base ao cálculo das calorias dessa actividade, devolvendo uma constante específica de cada actividade (ver secção 4.2).

Código 3.2: Variáveis de instância da superclasse Activity e subclasses respectivas (src/model/activity).

```
1  /***Super Classe***/
2  private GregorianCalendar mDate;
3  private long mDuration; /*activity duration [ms]*/
4  private Weather mWeather;
5  private int mHearthRate;/*heart rate [1/min] - for calorie burn calculation*/
6
7  /**Especializacoes Classe Distance**/
8  private int mDistance;
9  private int mMaxSpeed;
10
11 /**Especializacoes Classe Contest**/
12 private int mPointRival;
13 private int mPointTeam;
14 private Contest.Result mResult;
15
16 /**Especializacoes Classe Skill**/
17 private int mPoints;
18 private int mMaxTrick;
```

3.3 Record interface

(...) @COMENT: OLIVEIRA - work your magic :) (...)

3.4 Event class

As variáveis de instância da superclasse `Event` são apresentadas no excerto de código 3.3. Um evento pode ser caracterizado pelo seu nome, actividade associada, tipo de recorde associado, data de início e término, um valor correspondente a um pré-requisito necessário que um utilizador possa aderir ao evento, o número máximo de utilizadores que o evento suporta e o número de utilizadores actualmente inscritos. Os eventos são também classificados em dois tipos: de distância, ou do tipo torneio (jogos). As subclasses que traduzem essa classificação são a subclasse `EventDistance` e `EventContest`.

Código 3.3: Variáveis de instância da superclasse Event e subclasses respectivas (src/model/event).

```
1  /***Super Classe***/
2  private String mName;
3  private Activity mActivity;/*ATENTION: event weather inside activity*/
4  private int mRecordType;
5  private GregorianCalendar mEventDate;
6  private GregorianCalendar mEndDate;/*deadline for joining*/
7  private long mPreRequisite;
8  private int mMaxNumUsers;
9  private ManagerSet<String> mUsers; /*HashSet ok User keys(emails)*/
10
11 /**Especializacoes Classe EventDistance**/
12 private int mEventDistance;/*An event is about a Activity with its record type, like:
       Run 10000m*/
13
14 /**Especializacoes Classe Contest**/
15 private int mNum;/*Legs Between Users (1 Leg = 2 games)*/
16 private int mPointsVic;
17 private int mPointsDraw;
18 private int mPointsLoss;
```

Capítulo 4

Cálculo do Consumo de Calorias por Actividade

O procedimento de cálculo utilizado para estimar a quantidade de calorias dispendidas por actividade seguiu uma filosofia semelhante à utilizada na rede social *Endomondo* (www.endomondo.com). Na figura 4.1 apresenta-se um diagrama explicativo do mesmo, onde se evidenciam duas vias: uma baseada na frequência cardíaca e outra no tipo de actividade.

Os dados básicos necessários para o cálculo das calorias dispendidas estão relacionados com o individuo e são:

- Género;
- Características físicas — peso, altura;
- Idade;
- Frequência cardíaca em repouso (no caso da via de cálculo baseada na frequência cardíaca).

Os dados complementares são:

- Duração da actividade;
- O tipo de actividade (no caso da via de cálculo baseada no tipo de actividade);
- Frequência cardíaca média durante a actividade (no caso da via de cálculo baseada na frequência cardíaca).

4.1 Cálculo Baseado na Frequência Cardíaca

As formulações expostas na presente secção tiveram por base a aplicação web "Heart Rate Based Calorie Burn Calculator", da página http://www.shapesense.com/fitness-exercise/calculators/. Por uma questão de completitude apresentam-se todas as referências originais. No cálculo pela via da frequência cardíaca um dos parâmetros fundamentais é o $VO2_{max}$, que é a capacidade máxima do corpo de um indivíduo em transportar e fazer uso de oxigênio durante um exercício físico incremental. Este parâmetro pode ser estimado de acordo com [5], com base

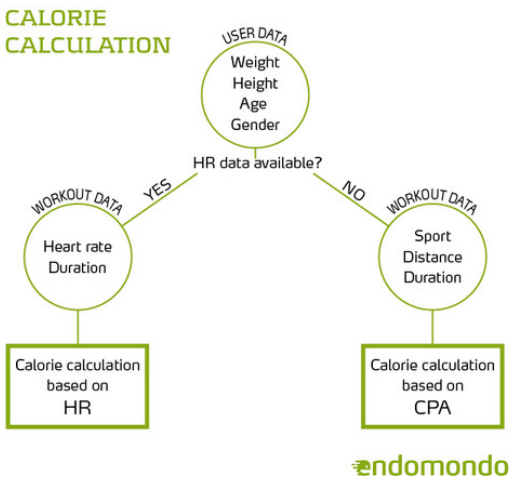

Figura 4.1: Diagrama com o procedimento de cálculo adoptado para a estimativa do consumo de calorias por actividade (ref. [3]).

na frequência cardíaca máxima (MHR) e em repouso (RHR) — equação 4.2. A frequência cardíaca máxima pode ser estimada com base na idade de acordo com [6] — equação 4.1. As calorias brutas dispendidas (GCB) foram estimadas de acordo com [7] — equação 4.3. Por fim, para se determinar o número de calorias efectivas (NCB) calculou-se a taxa metabólica basal (BMR) e a taxa de actividade ($RMRCB$) A estimativa das calorias efectivas foi realizada de acordo com [4] — equações 4.4, 4.5 e 4.6.

$$MHR = 208 - 0.7 \times Idade \quad [1/\text{min}] \tag{4.1}$$

$$VO2_{max} = 15.3 \times \frac{MHR}{RMR} \quad [\text{ml}/(\text{kg.min})] \tag{4.2}$$

Homens: (4.3)

$$GCB = \frac{-55.0969 + 0.6309 \times HR + 0.1988 \times W + 0.2017 \times A}{4.184} \times 60 \times T \quad [\text{kcal}]$$

Mulheres:

$$GCB = \frac{-20.4022 + 0.4472 \times HR + 0.1263 \times W + 0.074 \times A}{4.184} \times 60 \times T \quad [\text{kcal}]$$

onde:

HR = frequência cardíaca $[1/\text{min}]$
W = Peso [kg]
A = Idade [anos]
T = Duração exercício [h]

$$NCB = GCB - RMRCB \quad [\text{kcal}] \tag{4.4}$$

Tabela 4.1: Valores típicos do índice MET (ref. http://www.my-calorie-counter.com/mets_calculation.asp)

METS	Activity
1	sitting quietly and watching television
2	walking, less than 2.0 mph, level ground, strolling, very slow
3	loading /unloading a car
4	bicycling, $<$ 10 mph, leisure, to work or for pleasure
(...)	(...)
11	running, 6.7 mph
12	fire fighter, general

$$RMRCB = \frac{BMR \times 1.1}{24} \times T \quad \text{[kcal]} \tag{4.5}$$

Homens: (4.6)

$$BMR = (13.75 \times W) + (5 \times H) - (6.76 \times A) + 66 \quad \text{[kcal/24h]}$$

Mulheres:

$$BMR = (9.56 \times W) + (1.85 \times H) - 4.68 \times A) + 655 \quad \text{[kcal/24h]}$$

onde:

H = altura [cm]

restantes referências na eq. 4.3

4.2 Cálculo Baseado no Tipo de Actividade

Na ausência de dados relativos à frequência cardíaca do individuo, o cálculo do número de calorias dispendidas (CB) é efectuado com base no índice MET ("The Metabolic Equivalent of Task"). Este índice é uma medida fisiológica do custo energético de um dado exercício físico. Com base nesta abordagem mais simplificada, as calorias dispendidas podem ser calculadas de acordo com a equação 4.7.

$$CB = MET \times W \times T \quad \text{[kcal]} \tag{4.7}$$

Os índices MET foram definidos de acordo com o "Compendium of physical activities" [2], da autoria do "Healthy Lifestyles Research Center", "School of Nutrition and Health Promotion" da Arizona State University https://sites.google.com/site/compendiumofphysicalactivities/. Na tabela 4.1 apresentam-se alguns valores típicos dos índices MET. De forma a adaptar os valores dos índices MET a cada índividuo (peso, altura e idade), foram aplicadas correcções de acordo com [4] — valor corrigido $CMET$ (eq. 4.8), para mais detalhes ver https://sites.google.com/site/compendiumofphysicalactivities/corrected-mets.

$$CMET = MET \times \frac{3.5 \text{ ml/kg/min}}{RMRCB[\text{ ml/kg/min}]} \tag{4.8}$$

Capítulo 5

Cálculo da Forma dos Utilizadores

A forma é um valor de 0 a 1. (Pode ser mudado alterando as variáveis `MAX_FORMA` e `MIN_FORMA`).

O cálculo da forma assume um número de dias nos quais todas as actividades feitas nesse intervalo têm influência para a forma (variável `DIAS_RELEVANTES`). Se `DIAS_RELEVANTES=24`, significa eu só as actividades nos últimos 24 dias têm influência para a forma. Isto serve para simular o facto de só as actividades mais recentes deverem ter influência na forma actual e quanto mais recentes as actividades, maior a influência na forma.

A cada dia é atribuído um "peso", o quanto esse dia vai valer para o cálculo final da forma. No entanto, não é feita uma distribuição equitativa dos pesos pelos dias. Dias mais distantes no tempo têm menos influência e portanto, menos peso. Essa "menos influência" é dada por uma taxa (variável TAXA) que representa o decréscimo na forma resultante de ficar 1 dia sem fazer nenhuma actividade. Se `TAXA=0.05`, significa que cada dia que se ande para trás, tem -5% de influência no cálculo da forma. A contribuição para a forma de cada dia é calculado multiplicando o peso desse dia pelo quociente entre o número de minutos que o utilizador fez de uma actividade e o número de minutos recomendado para essa actividade (número de minutos recomendado para 1 actividade = `Intensidade`). Somadas todas as contribuições de cada dia, tem-se a forma final.

Capítulo 6

Estatísticas dos Utilizadores

(...) @COMENT: ANDRÉ - PEQUENA DESCRIÇÃO DOS MÉTODOS DE ESTATISTICA (metodo auxiliar actividadesEntre, metodologia geral com data de inicio e fim, pontos só para simulação) (...)

Código 6.1: Métodos para cálculo de estatísticas (src/core/UserStats).

```
1  static public int getDistanceStats(User usr, GregorianCalendar startDate,
       GregorianCalendar endDate){
2      int totalDist = 0;
3      for(Activity act : usr.actividadesEntre(startDate, endDate))
4          if(act instanceof Distance) totalDist += ((Distance) act).getDistance();
5      return totalDist;
6  }
7
8  static public int getCaloriesStats(User usr, GregorianCalendar startDate,
       GregorianCalendar endDate){
9      int totalCal = 0;
10     for(Activity act : usr.actividadesEntre(startDate, endDate))
11         totalCal += act.calculateCalories(usr);
12     return totalCal;
13 }
14
15 /** @return time in hours **/
16 static public double getWorkoutStats(User usr, GregorianCalendar startDate,
       GregorianCalendar endDate){
17     long totalTime = 0;
18     for(Activity act : usr.actividadesEntre(startDate, endDate))
19         totalTime += act.getDuration();
20     return (double) totalTime/(1000*3600);
21 }
22
23 /** points only for simulation purpose **/
24 static public int getPtsFromLastYear(User usr, Class<? extends Activity> category){
25     int totalPontos = 0;
26     /*date now*/
27     GregorianCalendar endDate = new GregorianCalendar();
28     /*date one year before*/
29     GregorianCalendar startDate = new GregorianCalendar(endDate.get(Calendar.YEAR) - 1,
           endDate.get(Calendar.MONTH), endDate.get(Calendar.DAY_OF_MONTH));
30     for(Activity act: usr.actividadesEntre(startDate, endDate)){
31         if( act instanceof Skill) totalPontos += ((Skill) act).getPoints();
32         else
33         if(act instanceof Contest)
34             totalPontos += ((Contest) act).getPointTeam() - ((Contest) act).
                   getPointRival();
35     }
36     return totalPontos;
37 }
```

Capítulo 7

Simulação de Eventos

(...)

Capítulo 8

Conclusões

(...)

Bibliografia

[1] Ribeiro AN. Projeto prático de programação orientada aos objectos, lei e lcc. 2014.

[2] Ainsworth BE; Haskell WL; Herrmann SD; Meckes N; Bassett Jr DR; Tudor-Locke C; Greer JL; Vezina J; Whitt-Glover MC; Leon AS. 2011 compendium of physical activities: a second update of codes and met values. *Medicine and Science in Sports and Exercise*, 43(8):1575–1581, 2011.

[3] Endomondo. Calories - calculation method. http://gsfn.us/t/4183d. [Acedido em maio de 2014, publicado em maio de 2013].

[4] Harris J.A.; Benedict F.G. A biometric study of human basal metabolism. *Proceedings of the National Academy of Sciences USA*, 4(12):370–373, 1918.

[5] Uth N.; Sørensen H.; Overgaard K.; Pedersen PK. Estimation of vo2max from the ratio between hrmax and hrrest–the heart rate ratio method. *European Journal of Applied Physiology*, 91(1):111 – 5, 2004.

[6] Tanaka H.; Monahan K. D.; Seals D. R. Age-predicted maximal heart rate revisited. *Journal of the American College of Cardiology*, 37(1):153 – 156, 2001.

[7] Keytel L.R.; Goedecke J.H.; Noakes T.D.; Hiiloskorpi H.; Laukkanen R.; van der Merwe L.; Lambert E.V. Prediction of energy expenditure from heart rate monitoring during submaximal exercise. *Journal of Sports Sciences*, 23(3):289 – 97, 2005.

Apêndice A

Demo da aplicação