Database Technologies (TBD)
2021/2022

Object Relational Assignment
Teaching Service

Henrique Reis Sendim Rodrigues (UP201606462)
João Carlos Machado Rocha Pires (UP201806079)
Mariana Catarina Pereira Soares (UP201605775)

Contents

1. Problem description

Our problem consists on a database with information regarding the distribution of teaching service in a faculty.

There are courses (table XUCS), described by a code (codigo), a designation (designacao), an acronym (sigla_uc) and a program (curso). Courses have occurrences in several years. Each occurrence is recorded by a row in the table XOCORRENCIAS, with information on the course code (codigo), academic year (ano_letivo), period of classes (periodo, that may be A-annual, 1S- first semester, 1T- first trimester, etc.), number of enrolled students (inscritos), students with distributed assessment (com_frequencia), number of approved (aprovados), course goals (objetivos) and content (conteudo), and department in charge (departamento).

Each occurrence may have one or more class types (T-theoretic, P-practical, L-laboratory, TP-theoretic/practical, OT- tutorial guidance). Each class type for an occurrence is recorded on table XTIPOSAULA with the number of similar classes (turnos), the number of week hours for each class (horas_turno), and in some cases the number of weekly classes (n_aulas).

The table XDSD records the teaching service distribution, in each semester, for each professor. More specifically, it records, for each class type of an occurrence, how many weekly hours are assigned to that professor. If a professor is teaching, in a single class, more than one course at the same time, for example from different programs, the weight of that course, in the perspective of the professor, may be less than 1 and recorded in attribute fator. Otherwise, the attribute fator will be 1. From the program perspective, the attribute fator is ignored. The attribute ordem enables listing the set of professors of a specific course occurrence in a specific order.

The professors are recorded in the table XDOCENTES with a number (nr), a name (nome), an acronym (sigla), a category code (categoria), a given name (proprio), a family name (apelido), and a status (estado: A-ativo, NA-não ativo, R-reformado).

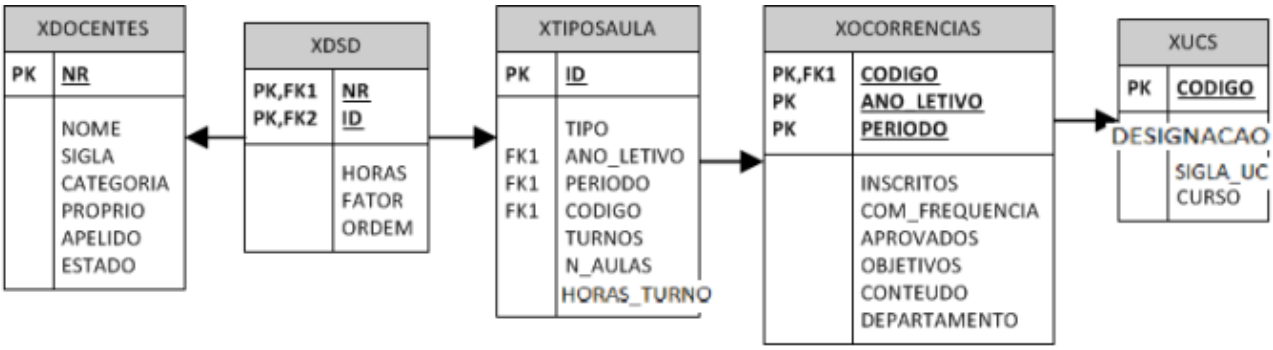

Figure 1.1: Relational model for the case Teaching Service.

2. Object-Relational Data Model

The first of this work was to design an object-relational data model for the situation described in the previous chapter, exploiting the SQL3 extensions.

We started with the schematic draw of our object-relational data model, as we present in the following picture:

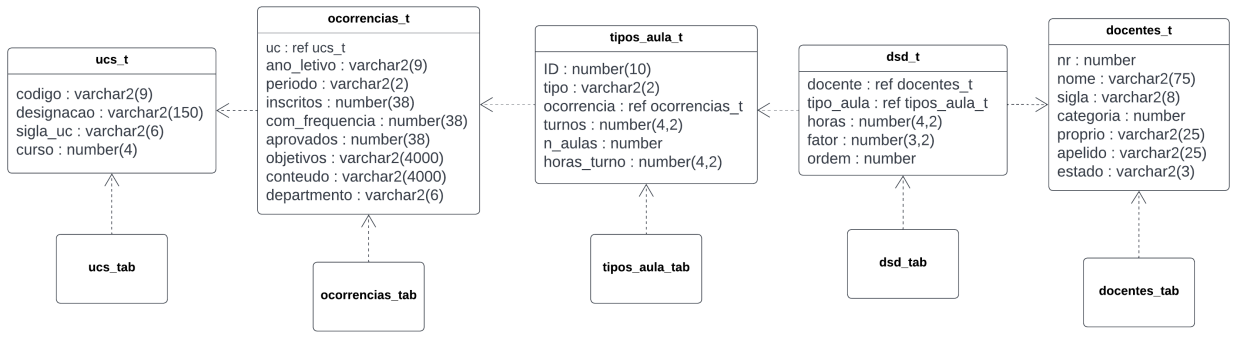

Figure 2.1: Object-Relational model for the case Teaching Service.

As we can see above, the schema of our object-relational data model is similar to the relational data model schema. The major differences come from the use of tables built of types previously defined that contain references to other defined types. Each attribute of each type was created with the same data type presented in the relational model.

The following code snippet represents the actual DDL and the implementation in the Oracle SQL Developer:

```
1  create type ucs_t as object(
2      codigo varchar2(9),
3      designacao varchar2(150),
4      sigla_uc varchar2(6),
5      curso number(4)
6  );
7
8  create table ucs_tab of ucs_t;
9
10 create type ocorrencias_t as object(
11     uc REF ucs_t,
12     ano_letivo varchar2(9),
13     periodo varchar2(2),
14     inscritos number(38),
15     com_frequencia number(38),
16     aprovados number(38),
```

```
17      objetivos varchar2(4000),
18      conteudo varchar2(4000),
19      departmento varchar2(6)
20  );
21
22  create table ocorrencias_tab of ocorrencias_t;
23
24  create type tipos_aula_t as object(
25      ID number(10),
26      tipo varchar2(2),
27      ocorrencia REF ocorrencias_t,
28      turnos number(4,2),
29      n_aulas number,
30      horas_turno number(4,2)
31  );
32
33  create table tipos_aula_tab of tipos_aula_t;
34
35  create type docentes_t as object(
36      nr number,
37      nome varchar2(75),
38      sigla varchar2(8),
39      categoria number,
40      proprio varchar2(25),
41      apelido varchar2(25),
42      estado varchar2(3)
43  );
44
45  create table docentes_tab of docentes_t;
46
47  create type dsd_t as object(
48      docente REF docentes_t,
49      tipo_aula REF tipos_aula_t,
50      horas number(4,2),
51      fator number(3,2),
52      ordem number
53  );
54
55  create table dsd_tab of dsd_t;
56
57  /* Add primary keys */
58
59  alter table ucs_tab add primary key (codigo);
60  alter table tipos_aula_tab add primary key (ID);
61  alter table docentes_tab add primary key (nr);
```

The last 3 instructions of the above code represent the creation of 3 primary keys on the tables

ucs_tab, tipos_aula_tab and docentes_tab.

3. Populate the Object-Relational Data Model

Next, we had to populate the object relational model with the data in the relational database. For that, for each newly created table, we inserted the corresponding data from the same table in the relational database. Note for the fact that, in some of the tables, we needed to insert the references to previously defined types instead of the foreign keys used in the relational model.

```
1 insert into ucs_tab (codigo, designacao, sigla_uc, curso)
2 select codigo, designacao, sigla_uc, curso from xucs;
3
4 insert into ocorrencias_tab (uc, ano_letivo, periodo, inscritos, com_frequencia,
      aprovados, objetivos, conteudo, departmento)
5 select (select REF(u) from ucs_tab u where u.codigo = o.codigo), o.ano_letivo, o.periodo
      , o.inscritos, o.com_frequencia, o.aprovados, o.objetivos, o.conteudo, o.departamento
       from zocorrencias o;
6
7 insert into tipos_aula_tab (id, tipo, ocorrencia, turnos, n_aulas, horas_turno)
8 select ta.id, ta.tipo, (select ref(o) from ocorrencias_tab o join ucs_tab u on ref(u) =
      o.uc where u.codigo = ta.codigo and o.ano_letivo = ta.ano_letivo and o.periodo = ta.
      periodo), ta.turnos, ta.n_aulas, ta.horas_turno from xtiposaula ta;
9
10 insert into docentes_tab (nr, nome, sigla, categoria, proprio, apelido, estado)
11 select nr, nome, sigla, categoria, proprio, apelido, estado from xdocentes;
12
13 insert into dsd_tab (docente, tipo_aula, horas, fator, ordem)
14 select (select ref(doc) from docentes_tab doc where d.nr = doc.nr), (select ref(ta) from
       tipos_aula_tab ta where ta.id = d.id), d.horas, d.fator, d.ordem from xdsd d;
```

4. Useful Methods

The last step before moving on to the queries was the creation of useful methods that would then be used on the queries to the database.

The first one, *horasfator*, is a method that returns the result of the multiplication of 'horas' and 'fator', both attributes of the dsd_table, giving us the total number of weekly hours, considering already the factor, for each 'docente'.

The *getclasshours* returns another multiplication, this time of 'turnos' times 'horas_turno', which gives us the total number of hours for a specific type of class.

The *getcurso* returns the 'curso' associated to a specific class type and the *getanoletivo* does the same but returns the 'ano_letivo'.

```
1 /* Useful for queries c) and f) */
2
```

```
3 alter type dsd_t add member function horasfator return number cascade;
4 alter type dsd_t add member function getProfessor return varchar2 cascade;
5
6 create or replace type body dsd_t as
7     member function horasfator return number is
8     begin
9     return horas * fator;
10    end horasfator;
11
12    member function getProfessor return varchar2 is
13    nome varchar2(75);
14    begin
15    select d.nome into nome from docentes_tab d where ref(d) = self.docente;
16    return nome;
17    end getProfessor;
18 end;
19
20 /* Useful for query f) */
21
22 alter type ocorrencias_t add member function baduc return varchar2 cascade;
23
24 create or replace type body ocorrencias_t as
25    member function baduc return varchar2 is
26    begin
27    IF (com_frequencia <= 0.10*inscritos) then
28        return 'TRUE';
29    ELSE
30        return 'FALSE';
31    END IF;
32    end baduc;
33 end;
34
35 /* Useful for queries a), b), c) and e) */
36
37 alter type tipos_aula_t add member function getclasshours return number cascade;
38 alter type tipos_aula_t add member function getcurso return number cascade;
39 alter type tipos_aula_t add member function getanoletivo return varchar2 cascade;
40
41 create or replace type body tipos_aula_t as
42    member function getclasshours return number is
43    begin
44    return turnos * horas_turno;
45    end getclasshours;
46    member function getcurso return number is
47    curso number;
48    begin
49    select u.curso into curso from ucs_tab u join ocorrencias_tab o on ref(u) = o.uc
      where self.ocorrencia = ref(o);
50    return curso;
51    end getcurso;
52    member function getanoletivo return varchar2 is
53    anoletivo varchar2(9);
54    begin
55    select o.ano_letivo into anoletivo from ocorrencias_tab o where self.ocorrencia =
```

```
      ref(o);
56     return anoletivo;
57     end getanoletivo;
58 end;
```

The following diagram shows the methods created per each type:

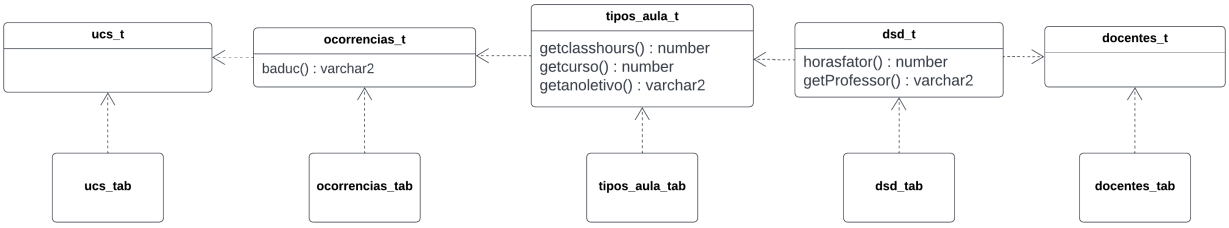

Figure 4.1: Methods per type

5. Queries

In this chapter, in each section, we present the written query and the SQL query that answers it, as well as the result and a brief explanation of the SQL query.

5.1 Query a)

How many class hours of each type did the program 233 got in year 2004/2005?

This SQL query uses 3 of the previously defined methods. It gets the type and the total number of class hours from 'tipos_aula_tab', restricting the results for the academic year of '2004/2005' and for the 'curso' 233, grouping the results by type.

```
1 SELECT ta.tipo, sum(ta.getclasshours()) as "CLASS_HOURS"
2 FROM tipos_aula_tab ta
3 WHERE ta.getanoletivo() = '2004/2005' and ta.getcurso() = 233
4 GROUP BY ta.tipo;
```

The result of the above query is the following:

	TIPO	CLASS_HOURS
1	P	581,5
2	TP	697,5
3	T	308

Figure 5.1: Question 4.a) - Answer.

5.2 Query b)

Which courses (show the code, total class hours required, total classes assigned) have a difference between total class hours required and the service actually assigned in year 2003/2004?

The SQL query to answer this question is divided into three parts.

The first one creates a view of a table with the code of UCs and the total class hours required for the academic year of '2003/2004'.

Then, a second view lists the code of UCs and the total hours assigned for the academic year of '2003/2004'.

The last query gets the code, the total class hours required, and the total hours assigned from the previous views where 'codigo' is the same and the total class hours required are different from the total hours assigned.

```
1  CREATE OR REPLACE VIEW hours_required as
2  SELECT t.ocorrencia.uc.codigo as code, sum(t.getclasshours()) as
       total_class_hours_required
3  FROM tipos_aula_tab t
4  WHERE t.getanoletivo() = '2003/2004'
5  GROUP BY t.ocorrencia.uc.codigo;
6
7  CREATE OR REPLACE VIEW hours_assigned as
8  SELECT d.tipo_aula.ocorrencia.uc.codigo as code, sum(d.horas) as total_assigned_hours
9  FROM dsd_tab d
10 WHERE d.tipo_aula.getanoletivo() = '2003/2004'
11 GROUP BY d.tipo_aula.ocorrencia.uc.codigo;
12
13 SELECT hours_required.code, total_class_hours_required, total_assigned_hours
14 FROM hours_required, hours_assigned
15 WHERE hours_required.code = hours_assigned.code and total_class_hours_required <>
       total_assigned_hours;
```

The result of the above query is the following:

	CODE	TOTAL_CLASS_HOURS_REQUIRED	TOTAL_ASSIGNED_HOURS
1	EEC5020	1	0
2	MGI1204	4,5	2,5
3	EM631	10	9
4	MRSC1104	1	2,3
5	EEC4277	4	1
6	EIC5202	24	22
7	MFAMF1101	2	1,99
8	MEB100	3	2
9	EMG2202	7	6
10	EC4104	20	22
11	EQ502	12	0
12	MGI1210	3	1,5
13	EEC5040	11	0
14	EMG4103	2	5
15	EMG2001	6	3
16	EI1206	4	1
17	EQ101	28	8
18	MGI1208	2	1,5
19	MEEC1054	3	1
20	EIC5102	7	4
21	MGI1209	2	1,5
22	MMI1204	3	13
23	GEI205	4	5
24	MGI1207	2	1,5
25	EEC5060	11	0
26	EM337	30	31,5
27	EMM527	40	0
28	EM114	39	25
29	GEI304	5	4,5
30	MRSC1201	5	3,2
31	MGI1108	2	1,5
32	EEC5021	16	17
33	EIC4101	24	14
34	GEI210	4	2
35	EMG1001	12	6
36	EC5183	6	3
37	EMG3102	6,5	5
38	MEM156	4	3

Figure 5.2: Question 4.b) - Answer.

5.3 Query c)

Who is the professor with more class hours for each type of class, in the academic year 2003/2004? Show the number and name of the professor, the type of class and the total of class hours times the factor.

The SQL query to answer this question is divided into three parts.

The first one creates a view of a table with the number of professors, the class types, and the total number of hours times the factor for the academic year of '2003/2004'.

Then, a second view lists the maximum number of hours per class type from the view created previously.

The last query gets the number and name of the professor, the class type, and the total number of hours (already considering the factors). For that, it was made a join between the first created view and the table 'docentes_tab' having the constraint of the number of hours being the maximum (since we only want the professor with the most number of hours per class type).

```
1  CREATE OR REPLACE VIEW sum_horas_tipo as
2  SELECT sum(d.horasfator()) as total_horas, d.docente.nr as nr, d.tipo_aula.tipo as
       tipoaula
3  FROM dsd_tab d
4  WHERE d.tipo_aula.getanoletivo() = '2003/2004'
5  GROUP BY d.docente.nr, d.tipo_aula.tipo;
6
7  CREATE OR REPLACE VIEW max_horas_tipo as
8  SELECT tipoaula, max(total_horas) as max_horas
9  FROM sum_horas_tipo
10 GROUP BY tipoaula;
11
12 SELECT sum_horas_tipo.nr, nome, sum_horas_tipo.tipoaula, sum_horas_tipo.total_horas as
       classhours_factor
13 FROM sum_horas_tipo join docentes_tab on docentes_tab.nr = sum_horas_tipo.nr
14 WHERE sum_horas_tipo.total_horas = (select max_horas from max_horas_tipo where
       max_horas_tipo.tipoaula = sum_horas_tipo.tipoaula);
```

The result of the above query is the following:

	NR	NOME	TIPOAULA	CLASSHOURS_FACTOR
1	208187	António Almerindo Pinheiro Vieira	P	30
2	249564	Cecília do Carmo Ferreira da Silva	TP	26
3	210006	João Carlos Pascoal de Faria	OT	3,5
4	207638	Fernando Francisco Machado Veloso Gomes	T	30,67

Figure 5.3: Question 4.c) - Answer.

5.4 Query d)

Which is the average number of hours by professor by year in each category, in the years between 2001/2002 and 2004/2005?

For this question there were two possible interpretations we identified:

- A. The average hours a Professor worked per week during each year and his category.
- B. The average hours Professors worked per week in each category during each year.

The interpretation B. was the one considered, so we moved on with that one in what respects to the analysis of the answer. However, before proceeding with the explanation of that option, the following two SQL queries represent the ones that would retrieve the results for the interpretations A and B.

```
1 /*A*/
2 SELECT dsd.docente.nome as "NOME", dsd.docente.categoria as "CATEGORIA", avg(dsd.horas)
      as "MEDIA HORAS POR SEMANA", dsd.tipo_aula.ocorrencia.ano_letivo as "ANO LETIVO"
3 FROM dsd_tab dsd
4 WHERE dsd.tipo_aula.ocorrencia.ano_letivo >= '2001/2002' AND dsd.tipo_aula.ocorrencia.
      ano_letivo <= '2004/2005'
5 GROUP BY dsd.docente.categoria, dsd.tipo_aula.ocorrencia.ano_letivo, dsd.docente.nome;
6
7 /*B*/
8 SELECT dsd.docente.categoria as "CATEGORIA", avg(dsd.horas) as "MEDIA HORAS POR SEMANA",
       dsd.tipo_aula.ocorrencia.ano_letivo as "ANO LETIVO"
9 FROM dsd_tab dsd
10 WHERE dsd.tipo_aula.ocorrencia.ano_letivo >= '2001/2002' AND dsd.tipo_aula.ocorrencia.
      ano_letivo <= '2004/2005'
11 GROUP BY dsd.docente.categoria, dsd.tipo_aula.ocorrencia.ano_letivo;
```

For this query, there was no need for joins as the structure used for the Database allows all the information required to be accessed through cursors. So what was done was a group by with the category of each Professor and the year with a filter to only show the years that were specified in the question.

Because the answer for this query was too large (88 rows), in the picture there is only a subset of the rows returned (first 24 rows).

	CATEGORIA	MEDIA HORAS POR SEMANA	ANO LETIVO
1	117	3,339837398373983739837398373983739837398374	2003/2004
2	110	2,56491228070175438596491228070175438596	2001/2002
3	519	3,85	2001/2002
4	111	2,6875	2001/2002
5	144	1,766	2001/2002
6	107	2,176454849498327759197324414715719063555	2002/2003
7	19999	2,22381578947368421052631578947368421053	2002/2003
8	11005	4,333333333333333333333333333333333333333	2003/2004
9	122	4,5	2003/2004
10	374	3	2003/2004
11	103	1,785714285714285714285714285714285714285	2002/2003
12	11007	4	2002/2003
13	10801	0,6	2002/2003
14	110	2,308374792703150912106135986733001658375	2004/2005
15	112	1,5	2001/2002
16	144	1,851111111111111111111111111111111111111	2002/2003
17	10119	2	2003/2004
18	519	3,035714285714285714285714285714285714285	2004/2005
19	19997	2	2004/2005
20	11007	2	2004/2005
21	144	1,5975	2003/2004
22	122	4,5	2001/2002
23	374	8	2004/2005
24	116	3,006252446183953033268101761252446183953	2003/2004

Figure 5.4: Question 4.d) - Answer, first 24 results.

5.5 Query e)

Which is the total hours per week, on each semester, that an hypothetical student enrolled in every course of a single curricular year from each program would get.

For this query, we do three group by's (by year, period and course) to visualize the total hours per week, using the method created before ('getclasshours').

```
1 SELECT sum(t.getclasshours()) as "SUM OF WEEKLY HOURS", t.ocorrencia.ano_letivo as "ANO
    LETIVO", t.ocorrencia.periodo as "PERIODO", t.ocorrencia.uc.curso as "CURSO"
2 FROM tipos_aula_tab t
3 GROUP BY t.ocorrencia.ano_letivo, t.ocorrencia.periodo, t.ocorrencia.uc.curso;
```

As with the question before, the rows returned were too large for one picture (847 rows), so below is a subset of the answer (first 25 rows).

	SUM OF WEEKLY HOURS	ANO LETIVO	PERIODO	CURSO
1	557	1998/1999	1S	255
2	13,5	1997/1998	1S	2005
3	6	1997/1998	2S	275
4	301	1998/1999	2S	304
5	312	1998/1999	1S	331
6	12	1998/1999	1T	2025
7	(null)	1999/2000	2S	700
8	12	2000/2001	1T	2020
9	22	1999/2000	3T	100
10	125,5	1998/1999	1S	433
11	(null)	1998/1999	12	1100
12	3	1998/1999	1S	2010
13	12	1998/1999	3T	2020
14	80	1999/2000	2S	318
15	52	1999/2000	1S	318
16	187	1999/2000	1S	275
17	(null)	1998/1999	12	100
18	18	1993/1994	1S	331
19	3	1995/1996	2S	275
20	51	1999/2000	2S	400
21	13	2003/2004	1T	2075
22	122	2000/2001	1S	315
23	12	2003/2004	3T	2075
24	137	2003/2004	2S	649
25	13,5	1999/2000	2S	2005

Figure 5.5: Question 4.e) - Answer, first 25 results.

5.6 Query f)

The last query was up to us to define one that illustrated the use of OR extensions. We've came up with the following query:

Which Professors gave lessons in a curricular unit where the number of students with frequency is no more than 10% of the number of students enrolled?

This query uses two created methods to get the name of the Professors that gave lessons in a curricular unit classified as bad UC, i.e. where the relation between the number of enrolled students and the number of students with frequency is *0,10*inscritos >= com_frequencia*. The query result was ordered by the Professors names.

The big advantage of this query being done using an object-relational data model is that we can use the created methods ('baduc' and 'getProfessors') to check if a certain UC is bad and get the

corresponding Professor.

```
1 select distinct d.getProfessor() as Professor
2 from dsd_tab d
3 where d.tipo_aula.ocorrencia.baduc() = 'TRUE'
4 order by d.getProfessor();
```

The result of the above query is the following:

	PROFESSOR
1	António Acácio Couto Jorge Lima
2	António Manuel Antunes Fiúza
3	Belmira de Almeida Ferreira Neto
4	Carlos Alberto Silva Ribeiro
5	Eduardo Filipe Júdice Nunes de Vilhena Crespo
6	Fernando Francisco Machado Veloso Gomes
7	Filomena Maria da Conceição Viana
8	Henrique Manuel Cunha Martins dos Santos
9	João Gonçalves de Oliveira Neves
10	Laura Maria Melo Ribeiro
11	Luís Filipe Malheiros de Freitas Ferreira
12	Manuel Afonso Magalhães da Fonseca Almeida
13	Manuel Fernando Gonçalves Vieira
14	Paulo Tenreiro dos Santos Monteiro
15	Rui Alfredo da Rocha Boaventura
16	Vitor Manuel Branco Martins Augusto

Figure 5.6: Question 4.f) - Answer.