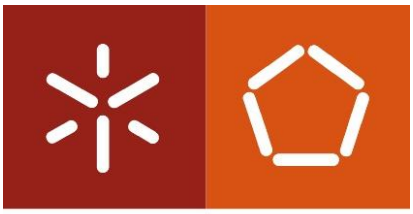


Universidade do Minho

Licenciatura em Engenharia Informática

Laboratórios de Informática - Projeto Prático

Grupo 82

João Ferreira (A79495)

Ano Letivo 2022/2023

Conteúdo

Capítulo 1

Resumo

Neste relatório encontra-se descrita a resolução do Enunciado do Projeto de 2022/2023 para a UC de Laboratórios de Informática, e descritas as estruturas e estratégias usadas por este grupo de forma a cumprir as exigências propostas.

1.1 Introdução

O foco deste trabalho é consolidação de conhecimentos essenciais da linguagem C e de Engenharia de Software,nomeadamente, modularidade e encapsulamento, estruturas dinâmicas de dados, validação funcional e medição de desempenho (computacional, consumo de memória, etc), bem como,a consolidação do uso de ferramentas essenciais ao desenvolvimento de projetos em C, nomeadamente, compilação, linkagem, definição de objetivos de projeto com base nas suas dependências e depuração de erros, e de gestão de repositórios colaborativos.

Capítulo 2

Análise do Problema

2.1 Primeira abordagem

De uma forma simplista e tendo como contexto os ficheiros "users.csv", "drivers.csv"e "rides.csv"fornecidos, o primeiro contacto com estes dados foi uma importante fonte de informação a que se deu a devida análise de forma a criar as primeiras estruturas compatíveis com os dados fornecidos e também com vista á resolução das queries propostas em que os conteúdos teóricos, também fornecidos pela equipa docente, fossem aplicados de modo a cumprir os princípios de encapsulamento e modularidade. Da análise dos dados, recorrendo á plataforma Knime, foram retiradas as seguintes informações:

- Dados users.csv - 5.1
- Dados drivers.csv - 5.2
- Dados rides.csv - 5.3

2.2 Estruturação

Durante o processo de abordagem ao problema, vários padrões foram retirados da análise dos ficheiros *.csv fornecidos. Foram também encontrados dados que seriam redundantes para o problema, mas que no entanto, não foram removidos das bases de dados por decisão de uma não perda de informação e também por avaliação ao sistema a desenvolver (de modo a ter o sistema sobrecarregado com informação redundante, e ainda assim ter um sistema que fosse lidar com o excesso de informação em tempo útil, ainda que não pertinente para a solução.)

- Estruturas
- Informações redundantes
-

Capítulo 3

Resolução

3.1 Implementação

- Leitura de input
- Estruturação da formatação da Linguagem
- Criação da gramática
- Aplicação da gramática gerada
- Criação de resultado

3.2 Leitura de input

A leitura do input é feita linha a linha e desta forma, não foi gerada uma implementação que resolvesse declarações muilti-linha.

3.3 Estruturação da formatação da Linguagem

Foram alterados alguns aspetos como por exemplo as funções pertencentes ao lexer no formato Ply-simple seriam iniciadas pelo char '.' ou que "%%"é uma forma de "desligar"o compilador e utilizar o código em txt como código devidamente formatado como PLY Lang.

3.4 Criação da gramática

A partir dos padrões retirados da abordagem foi gerada a gramática.

Capítulo 4

Conclusão

A realização deste projeto permitiu consolidar a matéria lecionada não só sobre PLY, mas também sobre Python. Possibilitou, especificamente, a prática e consolidação de: criação de lexer's, o desenvolvimento de uma gramática para um problema especifico, desenvolvimento de um parser para a geração de resultados e também trabalhar com Python.

Capítulo 5

Anexos

Table "default" - Rows: 100000 | Spec - Columns: 7 | Properties | Flow Variables

Columns: 7	Column Type	Co...	C...	...	Shape Handler	Filter Handler	Low...	Upper Bo...	Value 0	Value 1	Value 2
username	String	0					?	?	?	?	?
name	String	1					?	?	?	?	?
gender	String	2					?	?	M	F	?
birth_date	String	3					?	?	?	?	?
account_creation	String	4					?	?	?	?	?
pay_method	String	5					?	?	cash	credit_card	debit_card
account_status	String	6					?	?	active	inactive	?

Figura 5.1: Análise a dados users.csv

Table "default" - Rows: 10000 | Spec - Columns: 9 | Properties | Flow Variables

Columns: 9	Column Type	Column In...	Color Han...	Size Hand...	Shape Ha...	Filter Han...	Lower Bo...	Upper Bo...	Value 0	Value 1	Value 2	Value 3	Value 4
id	Number (integer)	0					1	10000	?	?	?	?	?
name	String	1					?	?	?	?	?	?	?
birth_day	String	2					?	?	?	?	?	?	?
gender	String	3					?	?	F	M	?	?	?
car_class	String	4					?	?	green	premium	basic	?	?
license_plate	String	5					?	?	?	?	?	?	?
city	String	6					?	?	Setúbal	Braga	Lisboa	Porto	Faro
account_creation	String	7					?	?	?	?	?	?	?
account_status	String	8					?	?	active	inactive	?	?	?

Figura 5.2: Análise a dados drivers.csv

Table "default" - Rows: 1000000 | Spec - Columns: 10 | Properties | Flow Variables

Columns: 10	Column Type	Co...	...	...	...	...	Lower Bo...	Upper Bo...	Value 0	Value 1	Value 2	Value 3	Value 4
id	Number (integer)	0					1	1,000,000	?	?	?	?	?
date	String	1					?	?	?	?	?	?	?
driver	Number (integer)	2					1	10,000	?	?	?	?	?
user	String	3					?	?	?	?	?	?	?
city	String	4					?	?	Setúbal	Faro	Porto	Braga	Lisboa
distance	Number (integer)	5					1	14	?	?	?	?	?
score_user	Number (integer)	6					1	5	?	?	?	?	?
score_driver	Number (integer)	7					1	5	?	?	?	?	?
tip	Number (double)	8					0.5	5	?	?	?	?	?
comment	String	9					?	?	?	?	?	?	?

Figura 5.3: Análise a dados rides.csv

```
typedef struct data_user{

    char *username;
    char *name;
    enum gender gender;
    struct tm birth_date;
    struct tm account_creation;
    enum pay_method pay_method;
    enum account_status account_status;
    int idade;
    int num_viagens;
    int distancia_viajada;
    int total_avaliacao;
    double total_gasto;
    struct tm data_ultima_ride_user;

}*DATA_USER;
```

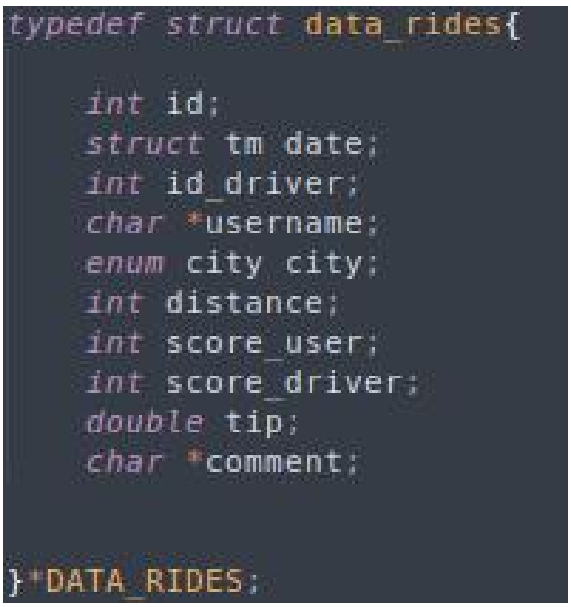
Figura 5.4: Struct Users

```
typedef struct data_driver{

    int id;
    char *name;
    struct tm birth_day;
    enum gender gender;
    enum car_calss car_class;
    char *license_plate;
    enum city city;
    struct tm account_creation;
    enum account_status account_status;
    int age;
    int num_viagens;
    int avaliacao_total;
    double total_auferido_driver;
    struct tm data_ultima_ride_driver;


}*DATA_DRIVER;
```

Figura 5.5: Struct Drivers



```
typedef struct data_rides{

    int id;
    struct tm date;
    int id_driver;
    char *username;
    enum city city;
    int distance;
    int score_user;
    int score_driver;
    double tip;
    char *comment;


}*DATA_RIDES;
```

Figura 5.6: Struct Rides