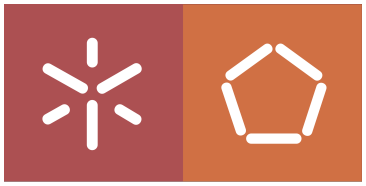

Universidade do Minho
Escola de Engenharia

Trabalho Prático

Laboratórios de Informática III

Outubro, 2022

1 Objetivos

- 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);
- 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.

2 Realização e avaliação do trabalho desenvolvido

O desenvolvimento deste projeto deve ser realizado colaborativamente com o auxílio de *Git* e *GitHub*. Os docentes serão membros integrantes da equipa de desenvolvimento de cada trabalho e irão acompanhar semanalmente a evolução dos projetos. A entrega do trabalho será realizada através desta plataforma e os elementos do grupo serão avaliados individualmente de acordo com a sua contribuição no repositório e na apresentação e discussão do mesmo. A estrutura do repositório deverá ser mantida, assim como as regras descritas ao longo do enunciado deverão ser seguidas de forma a que o processo de avaliação e de execução dos trabalhos possa ser uniforme entre os grupos e de forma tão automática quanto possível.

Tenham, em particular atenção, ao conjunto de instruções a seguir:

- O trabalho terá de ser desenvolvido por todos os elementos do grupo de trabalho e todos deverão registar as suas contribuições individuais no respetivo repositório *git*;
- O trabalho desenvolvido por cada grupo será avaliado em duas fases com base no conteúdo da pasta "trabalho-pratico" do repositório *GitHub*. A primeira fase de avaliação considerará o conteúdo do repositório no dia 12/11/2022 (23:59). A segunda fase considerará o conteúdo do repositório no dia 19/01/2023 (23:59).
- Cada fase será acompanhada de um relatório (máximo de 10 páginas de conteúdo, ou seja, sem capas ou anexos, em formato PDF) que deverá ser disponibilizado na raiz da pasta "trabalho-pratico" na mesma data da entrega da respetiva fase do trabalho. Os ficheiros cor-

respondentes terão os nomes “relatorio-fase1.pdf” e "relatorio-fase2.pdf", respetivamente. O conteúdo do relatório deverá centrar-se na resolução dos exercícios propostos, identificando as estratégias seguidas e eventuais limitações, bem como a medição de aspetos de desempenho, tais como o custo computacional e de armazenamento das estruturas de dados;

- O projeto terá de gerar o necessário ficheiro executável (com o nome "programa-principal") com base na preparação de um ficheiro *Makefile* (ambos na raiz da pasta “trabalho-pratico”) e por invocação do comando *make*. Da mesma forma, deverá limpar todos os ficheiros desnecessários ao projeto através da execução do comando *make clean*;

- O desenvolvimento da aplicação deverá ser feito com uso exclusivo da biblioteca padrão do C (i.e., *libc*). A única exceção é a possibilidade de recorrer à *glib2* para a manipulação das coleções de dados;

- O executável deverá assumir a existência de ficheiros de entrada com os nomes *users.csv*, *drivers.csv* e *rides.csv* numa pasta cujo caminho é passado como argumento na execução do programa;

- A aplicação deverá assumir dois modos de execução, diferenciados pelo número de argumentos recebidos ao executá-lo. Os modos são:

  - *Batch*: Neste modo, o programa é executado com dois argumentos, o primero é o caminho para a pasta onde estão os ficheiros de entrada. Já o segundo corresponde ao caminho para um ficheiro de texto que contém uma lista de comandos (*queries*) a serem executados. O resultado da execução de cada comando deverá ser escrito num ficheiro individual localizado na pasta "Resultados"da raiz da pasta "trabalho-pratico". O formato dos ficheiros de comandos e de resultados são descritos na Secção 4.

  - *Interativo*: Neste modo, o programa é executado sem argumentos. Nele, o grupo disponibilizará um menu interativo contendo toda a informação (instruções) necessária para a execução de cada comando (*query*). Aqui, cada comando é interpretado e executado individualmente, com o respetivo resultado apresentado no terminal. É importante observar que a fase inicial de interação do utilizador com o programa corresponda à introdução do caminho para a pasta onde estão os ficheiros de entrada.

- Em ambas as fases, o trabalho realizado por cada grupo será avaliado em sessões de discussão com a equipa docente que terão início na semana seguinte à entrega da componente em avaliação.

2.1 Fases e critérios de avaliação

Para uma melhor organização do desenvolvimento do trabalho e para potencializar um melhor resultado, cada fase da avaliação estará centrada num conjunto pré-definido de componentes, nomeadamente:

2.1.1 Fase 1

Esta fase do trabalho terá um peso de 50% na avaliação final da Unidade Curricular de Laboratórios de Informática III. Tendo em consideração a arquitetura da aplicação (descrita na Secção 3), na Fase 1, os trabalhos deverão compreender a estratégia e a implementação de componentes demonstráveis correspondentes a:

- *Parsing* dos dados de entrada;
- Modo de operação *batch*;
- 1/3 das queries descritas na Secção 4 funcionais (o grupo poderá escolher as queries demonstradas nesta fase);
- Catálogo de dados para os três ficheiros de entrada, i.e., *users.csv*, *drivers.csv* e *rides.csv*. Nesta fase, a solução deverá considerar que a dimensão dos ficheiros, em número de linhas, é 100.000, 10.000 e 1.000.000, respetivamente.

Além disso, será central também a apresentação e discussão sobre a estratégia definida pelo grupo para a modularização e encapsulamento da solução proposta (incluindo o código correspondente).

2.1.2 Fase 2

De forma semelhante, a segunda fase do trabalho terá um peso de 50% na avaliação final da Unidade Curricular de Laboratórios de Informática III. Ao final desta fase, é esperada a conclusão do trabalho proposto, assim como as considerações relativas aos aspetos de desempenho da solução desenvolvida. Mais especificamente, aspetos relevantes da avaliação incluirão:

- A totalidade das queries descritas na Secção 4;
- Modo de operação *interativo*;
  - Incluindo o menu de interação com o programa e um módulo de paginação para apresentação de resultados longos
- Evolução dos aspetos relacionados com a modularidade, encapsulamento e qualidade do código observados na sessão de apresentação da Fase 1;
- Análise e discussão sobre o desempenho da solução desenvolvida (ver Secção 5);
- Na fase 2, é preciso dimensionar a solução para ficheiros de entrada com uma ordem de grandeza superior;
- Algumas **validações simples sobre o formato dos ficheiros de dados** (Secção 6).

2.1.3 Critérios de avaliação

A avaliação do trabalho em ambas as fases estará concentrada, principalmente, nos seguintes critérios e pesos (indicativos)[1]:

[1]Os critérios e pesos serão ajustados aos requisitos de cada fase de avaliação.

- Compilação / Makefile (Peso = 5%)
- Estratégia de modularização e reutilização de código (Peso = 15%)
- Estratégia de encapsulamento e abstração (Peso = 15%)
- Adequação das estruturas de dados para cada coleção (Peso = 20%)
- Número de queries corretamente implementadas (Peso = 10%)
- Qualidade geral do código (Peso = 15%)
- Análise de desempenho (Peso = 10%)
- Relatório (Peso = 10%)

3 Arquitetura da aplicação

Com o objetivo de auxiliar o desenho da solução, a Figura 1 apresenta uma arquitetura exemplo da aplicação a desenvolver. Observe, com particular atenção, a divisão entre módulos de leitura de dados, interpretação de comandos e módulos de dados (catálogos).

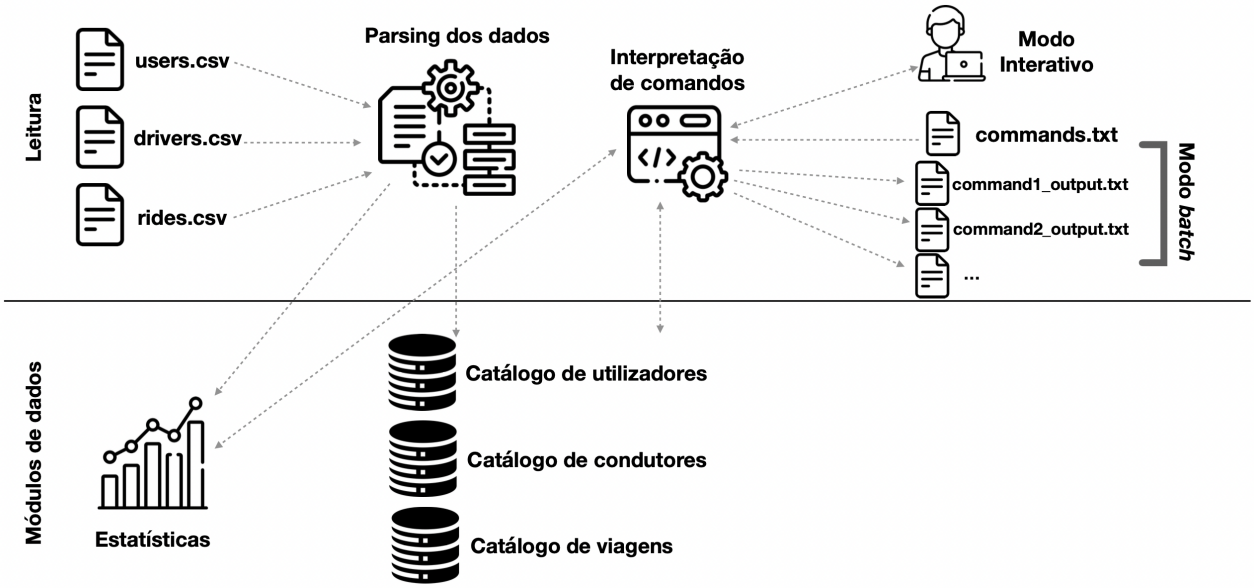

Figura 1: Arquitetura de referência para a aplicação a desenvolver

- Parsing dos dados: função ou parte do código no qual é realizada a leitura (e tratamento) dos dados dos 3 ficheiros de entrada;
- Interpretação dos comandos: parte do código responsável por ler o ficheiro de comandos, interpretar cada um e executar a respetiva query com os argumentos indicados (se existirem);
- Catálogo de Utilizadores (i.e., *users*): módulo de dados onde deverão ser guardados todos os utilizadores provenientes de registos válidos do ficheiro *users.csv*;

- Catálogo de Condutores (i.e., *drivers*): módulo de dados onde deverão ser guardados todos os condutores e respectiva informação contida no ficheiro *drivers.csv*;
- Catálogo de Viagens (i.e., *rides*): módulo de dados onde armazena convenientemente as viagens recolhidos do ficheiro *rides.csv*;
- Estatísticas: módulo que efetua relações entre os 3 principais modelos, proporcionando acesso mais rápido a dados e resultados pedidos nas queries do programa.

Para a estruturação adequada destes módulos de dados será crucial analisar as queries que a aplicação deverá implementar. É de realçar que não devem assumir que o programa irá trabalhar sempre sobre os mesmos ficheiros de dados. Ao conceber os módulos de dados seguindo a estratégia sugerida, obtém-se a primeira arquitetura de base para o projeto, sendo desde já de salientar que esta arquitetura (ou algo equivalente) irá ser construída por fases, módulo a módulo, testando cada fase e cada módulo até se ter a garantia da sua corretude. Só depois destes testes, deverão juntar todas as unidades do projeto na aplicação final.

4 Exercícios

De forma a convenientemente estruturar, gerir e expandir o projeto, pretende-se que na sua concepção tenham em conta os princípios de Modularidade e Encapsulamento abordados nas aulas. A aplicação destes conceitos é obrigatória e será objeto central de avaliação no momento da apreciação e discussão do trabalho.

Com vista a avaliar e validar o funcionamento e a eficiência do armazenamento e gestão da informação em memória, pretende-se que o trabalho prático dê resposta a um conjunto de interrogações (*queries*) sobre os dados. A interação do utilizador com o programa será única e exclusivamente através da sua invocação via linha de comandos seguindo os requisitos de cada modo de operação (i.e., *batch* ou *interativo*). No modo *batch*, as *queries* que o utilizador pretende executar serão especificadas como linhas de um ficheiro de texto, cujo nome é passado como argumento ao programa (ver Secção 2). O formato para a especificação de *queries* (dentro do ficheiro de texto) é o seguinte:

$$< query - id > [arg1...argN]$$

O ficheiro de *queries* deverá então especificar uma *query* por linha. Cada uma destas linhas corresponde a um comando. Exemplo do conteúdo de um ficheiro contendo comandos válidos:

```
1 000000745
4 Braga
6 Lisboa 01/01/2020 31/12/2020
1 AntManoel33
```

No exemplo acima, a primeira linha corresponde a um comando que invoca a *query 1* para consultar o resumo do perfil do condutor com *ID* "000000745". Note que a última linha também corresponde a um comando que invoca a *query 1*. Contudo, neste caso, a consulta refere-se ao perfil do utilizador com *ID* (i.e., *username*) "AntManoel33". A aplicação deverá ser capaz de distiguir a qual tipo de perfil o comando se refere. A segunda linha do ficheiro corresponde ao comando que invoca a

query 4, com o argumento "Braga". Já a terceira linha, corresponde ao comando que invoca a *query 6* com os respetivos argumentos.

Ainda no modo de operação *batch*, o resultado de cada comando deverá ser escrito num ficheiro de texto com nome que seguirá o formato *commandX_output.txt*[2] e que deverá ser armazenado na pasta "Resultados"da raiz da pasta "trabalho-pratico". Um exemplo de ficheiro de resultado para a primeira linha seria[3]:

command1_output.txt

Manuel Silva;M;34;4.5;122;1202.15

No modo de operação *interativo*, o grupo é responsável por definir o formato de entrada das *queries* e o *layout* dos respetivos resultados.

Descrição dos ficheiros de entrada

Considere agora os ficheiros disponibilizados na BB, cuja descrição é apresentada de seguida:

- **Utilizadores** (*users.csv*)
  $username$; $name$; $gender$; $birth\_date$; $account\_creation$; pay_method; $account\_status$
- **Condutores** (*drivers.csv*)
  id; $name$, $birth\_date$; $gender$; car_class; $license\_plate$; $city$; $account\_creation$; $account\_status$
- **Viagens** (*rides.csv*)
  id; $date$; $driver$; $user$; $city$; $distance$; $score\_user$; $score\_driver$; tip; $comment$

Considere ainda que:

- As datas devem seguir o formato $dd/mm/aaaa$
- O estado das contas (*account_status*) de utilizadores e de condutores podem assumir os valores active ou inactive;
- Cada classe de veículo (*car_class*) tem uma tarifa mínima acrescida ao custo por quilómetro de viagem, nomeadamente:
  - Basic: Tarifa mínima = 3.25€ + 0.62€/km
  - Green: Tarifa mínima = 4.00€ + 0.79€/km
  - Premium: Tarifa mínima = 5.20€ + 0.94€/km
- As gorjetas (i.e., *tips*) são valores em Euros e não consideram o valor da viagem[4];
- Os resultados representados por números decimais deverão ser **arredondados** a três casas

[2] Onde *X* é o número da linha do respetivo comando no ficheiro *commands.txt*
[3] Para *queries* com *inputs* inválidos, o ficheiro de *output* deverá ser criado, mas sem conteúdo.
[4] Os valores das viagens também são representados em Euros.

na parte decimal;[5]

- Os resultados das queries Q1, Q2, Q3, Q7, Q8 deverão levar em consideração apenas os utilizadores e condutores com contas ativas;
- No modo de operação *interativo*, caso os resultados excedam a capacidade de uma página, deverá existir um menu de navegação para consultar as diferentes páginas de resultados;
- A avaliação média dos utilizadores e condutores deverá levar em consideração os campos *score_user* e *score_driver*, respetivamente, e não deverá ser afetado pelo *status* da conta (o mesmo vale para o cálculo das médias de preços e viagens);
- Deverá ser considerara a data **09/10/2022** como referência para o cálculo de idades, devendo estar especificada no código através de um ou mais `#define`;
- Deverá ser usado o tipo de dados **`double`** (e não `float`) para a representação de valores decimais;
- Caso a *query* não retorne nenhum resultado (e.g., utilizador inexistente / está inativo na Q1, cidade sem viagens na Q4, ...), o ficheiro resultante no modo *batch* **deverá estar vazio** (não deverá ser colocado um *newline*).

Queries

Q1: Listar o resumo de um perfil registado no serviço através do seu identificador, representado por *<ID>*. Note que o identificador poderá corresponder a um utilizador (i.e., *username* no ficheiro *users.csv*) ou a um condutor (i.e., *id* no ficheiro *drivers.csv*). Em cada caso, o *output* será diferente, mais especificamente:

- Utilizador
  nome;genero;idade;avaliacao_media;numero_viagens;total_gasto
- Condutor
  nome;genero;idade;avaliacao_media;numero_viagens;total_auferido

Caso o utilizador/condutor não tenha nenhumas viagens associadas, considerar a *avaliacao_media*, *numero_viagens*, e *total_gasto* como sendo 0.000, 0, e 0.000, respetivamente.

Comando
1 <ID>
Output
nome;genero;idade;avaliacao_media;numero_viagens;total_gasto

Q2: Listar os N condutores com maior avaliação média. Em caso de empate, o resultado deverá ser ordenado de forma a que os condutores **com a viagem mais recente** surjam primeiro. Caso haja novo empate, deverá ser usado o ***id* do condutor** para desempatar (por ordem crescente). O

[5]Notar que um *score* médio de, e.g., 3.2 deverá ser apresentado como 3.200. Sugestão: o formato `%.3f` faz o arredondamento e coloca os zeros extra caso necessário.

parâmetro N é representado por *<N>*.

Comando
2 <N>
Output
id;nome;avaliacao_media
id;nome;avaliaca_omedia
...

Q3: Listar os N utilizadores com maior distância viajada. Em caso de empate, o resultado deverá ser ordenado de forma a que os utilizadores **com a viagem mais recente** surjam primeiro. Caso haja novo empate, deverá ser usado o ***username* do utilizador** para desempatar (por ordem crescente). O parâmetro N é representado por *<N>*.

Comando
3 <N>
Output
username;nome;distancia_total
username;nome;distancia_total
...

Q4: Preço médio das viagens (sem considerar gorjetas) numa determinada cidade, representada por *<city>*.

Comando
4 <city>
Output
preco_medio

Q5: Preço médio das viagens (sem considerar gorjetas) num dado intervalo de tempo, sendo esse intervalo representado por *<data A>* e *<data B>*.

Comando
5 <data A> <data B>
Output
preco_medio

Q6: Distância média percorrida, numa determinada cidade, representada por *<city>*, num dado intervalo de tempo, sendo esse intervalo representado por *<data A>* e *<data B>*.

Comando
6 <city> <data A> <data B>
Output
distancia_media

Q7: Top N condutores numa determinada cidade, representada por *<city>* (no ficheiro *rides.csv*), ordenado pela avaliação média do condutor. Em caso de empate, o resultado deverá ser ordenado através do *id* do condutor, de forma decrescente. A avaliação média de um condutor numa cidade é

referente **às suas viagens nessa cidade**, e não na cidade que está no seu perfil (ou seja, o mesmo condutor poderá ter médias diferentes dependendo da cidade).

Comando
7 <N> <city>
Output
id;nome;avaliacao_media
id;nome;avaliacao_media
...

Q8: Listar todas as viagens nas quais o utilizador e o condutor são do género passado como parâmetro, representado por *<gender>* e têm perfis **com X ou mais anos**, sendo X representado por *<X>*. O *output* deverá ser ordenado de forma que as contas mais antigas apareçam primeiro, mais especificamente, ordenar por conta mais antiga de condutor e, se necessário, pela conta do utilizador. Se persistirem empates, ordenar por *id* da viagem (em ordem crescente).

Comando
8 <gender> <X>
Output
id_condutor;nome_condutor;username_utilizador;nome_utilizador
id_condutor;nome_condutor;username_utilizador;nome_utilizador
...

Q9: Listar as viagens nas quais o passageiro deu gorjeta, no intervalo de tempo (data A, data B), sendo esse intervalo representado pelos parâmetros *<data A>* e *<data B>*, ordenadas por ordem de distância percorrida (em ordem decrescente). Em caso de empate, as viagens mais recentes deverão aparecer primeiro. Se persistirem empates, ordenar pelo *id* da viagem (em ordem decrescente).

Comando
9 <data A> <data B>
Output
id_viagem;data_viagem;distancia;cidade;valor_gorjeta
id_viagem;data_viagem;distancia;cidade;valor_gorjeta
...

5 Testes funcionais e de desempenho

Nesta componente do trabalho, pretende-se que sejam desenvolvidos testes que validem e avaliem o funcionamento do programa desenvolvido. Desta forma, deverão ser desenvolvidos testes que avaliem o funcionamento de cada *query* descrita na secção anterior. Para cada uma destas funcionalidades, deverá ser desenvolvido um teste que avalie se a *query* é executada em tempo útil (consideremos como tempo útil um tempo de execução inferior a *10 segundos*) e que o seu resultado é correto.

```
test_top_users_dist(){
    query3(10);  // write result to file
}
```

```
    main(){
        start_time = time_now();
        test_top_users_dist();
        time_passed = time_now() - start_time;
        compare(produced_file, expected_file);
    }
```

Listing 1: Pseudo-código de um módulo de testes que avalia e valida a query 3

```
    int main(int argc, const char* argv[]) {
     clock_t start, end;
     double cpu_time_used;
     start = clock();
     //execute intended work load (e.g., a query)
     end = clock();
     cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
     printf("CPU Time:%f\n", cpu_time_used );
     return 0;
}
```

Listing 2: Snippet de código C para medição de tempo

A *Makefile* do projeto deverá ser capaz de gerar um executável adicional de testes "programa-testes", capaz de invocar todos os testes e apresentar os seus resultados (a verificação de que cada uma das queries funciona conforme o especificado e respetivo tempo de execução). Este programa de testes deverá executar e validar a execução das funcionalidades do projeto, respeitando sempre a modularidade e encapsulamento dos componentes do sistema. O relatório da Fase 2 do projeto deverá conter uma tabela com os resultados obtidos para as diferentes máquinas dos elementos de grupo, bem como as respectivas especificações do ambiente de execução. Naturalmente, os resultados deverão ser acompanhados de uma discussão que vise justificar os resultados obtidos.

6 Validação dos ficheiros de dados

Para a segunda fase, deverão ainda considerar um conjunto de validações a executar sobre os dados recebidos (apenas referente aos *csvs* com a informação de utilizadores, condutores, e viagens; comandos são sempre válidos). Caso alguma entrada seja inválida, deverá ser ignorada. Apenas certos campos de certas linhas é que poderão ser inválidos. Contudo, os *csvs* em si continuarão a ter sempre o formato válido (i.e., mesmo número de colunas para todas as linhas). Caso, por exemplo, um condutor/utilizador seja inválido, é garantido que não existem viagens associadas a este (ou seja, não é necessário validar viagens de condutores/utilizadores inválidos, pois estas não existirão). Considerar sempre que o *user*/*driver* de cada viagem existe e é válido. Para reforçar, um erro numa linha de um ficheiro pode ser considerado como auto-contido a esse ficheiro, não sendo necessário efetuar a validação entre ficheiros.

De seguida, apresentam-se as validações que têm que considerar:

- Datas (*birth_date* do condutor e utilizador, *account_creation* do condutor e utilizador, *date* da viagem):
  - O formato deverá ser sempre do tipo *nn/nn/nnnn*, onde n é um número entre 0 e 9 (inclusivo). Exemplo de possíveis erros: `0910/2022`, `10/20222`, `09/10`, `09102022`, `2022/10/09`, `09|10|2022`, `09/10/abcd`, ...;
  - O mês deverá estar entre 1 e 12 (inclusivo) e o dia entre 1 e 31 (inclusivo). Ignorar a validação dos dias consoante o mês (e.g., datas como 31/02/2022 não surgirão). Exemplos de erros: `90/10/2022`, `09/15/2022`, ...
- *car_class* deverá ter o valor *premium*, *green*, ou *basic* (isto inclui, por exemplo, *basic*, *BASIC*, *Basic*, *BasiC*, ..., ou seja, combinações de maiúsculas e minúsculas dos valores válidos continuam a ser válidas);
- *account_status* deverá ter o valor *active* ou *inactive* (tal como em cima, combinações de maiúsculas e minúsculas dos valores válidos continuam a ser válidas);
- *distance*: valor inteiro maior que zero. Exemplo de erros: `-3`, `3.2`, `NaN`, `0`, ...;
- *score_user*: valor decimal (ou inteiro) maior ou igual a zero. Exemplo de erros: `3.m`, `asd`, `3,2` (notar que o separador decimal '`,`' é inválido), `-3.2`, ...;
- *score_driver*: valor decimal (ou inteiro) maior ou igual a zero;
- *tip*: valor decimal (ou inteiro) maior ou igual a zero;
- Os seguintes campos têm que ter tamanho superior a zero (i.e., linhas com estes campos vazios deverão ser descartadas):
  - Condutor: *id*, *name*, *gender*, *license_plate*, *city*;
  - Utilizador: *username*, *name*, *gender*, *pay_method*;
  - Viagem: *id*, *driver*, *user*, *city*.

Alterações desde a última versão

- Visto que o formato `%.3f` no printf efetua o arredondamento em vez de truncar o valor, por simplicidade, os resultados deverão ser arredondados em 3 casas (e não truncados em 3 casas);
- Para a uniformização dos resultados, deverão considerar a data 09/10/2022 como referência para o cálculo de idades (esta deverá estar definida através do uso de `#define`(s));
- Para uniformizar a representação interna e cálculos sobre valores decimais, esses valores deverão ser armazenados e manipulados usando o tipo de dados `double`;
- Clarificado que, para *queries* que não retornem resultado, o respetivo ficheiro de output não deverá conter nenhum conteúdo;
- Clarificado que na Q1, caso um condutor/utilizador não tenha viagens, a *avaliacao_media*, *numero_viagens*, e *total_gasto* deverão ser apresentados como 0, nomeadamente, 0.000, 0, e 0.000, respetivamente;
- Para simplificar, na Q2 e Q3 o desempate passa a ser por *score*, a viagem mais recente (apenas uma), e o identificador do condutor/utilizador (de forma ascendente);
- Clarificado que na Q7 a avaliação média deverá ser calculada com base na cidade da viagem e não na cidade do perfil do condutor;
- Clarificado que na Q8 deverão considerar contas com X ou mais anos, ou seja, contas com exatamente X anos também deverão ser apresentadas;
- Introduzido validação de dados (apenas para a segunda fase).