

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Transporte de passageiros

49412 : Vasco Fonseca Amado da Costa (A49412@alunos.isel.pt)

49418 : Roberto Carlos Petrisoru (A49418@alunos.isel.pt)

49506 : Pedro David Marques Malafaia (A49506@alunos.isel.pt)

Relatório para a Unidade Curricular de Introdução aos Sistemas de Informação
da Licenciatura em Engenharia Informática e de Computadores

Professora : Professora Matilde Pós-de-Mina Pato

Resumo

O JDBC é uma API baseada em X/Open SQL CLI que permite o acesso (em código Java), a base de dados relacionais e a execução em comandos SQL.

Os dados fornecidos pelo utilizador em comandos de DML podem levar a vulnerabilidades de software. Um invasor pode enviar uma instrução SQL válida que altera a lógica da consulta inicial. Podendo, este visualizar/modificar/remover dados confidenciais de outros utilizadores ou mesmo ter acesso não autorizado do sistema, por isso temos sempre que garantir que os dados não sejam perdidos, mal gerenciados ou mesmo cheguem tarde a terceiros.

O problema analisado foi a implementação de uma aplicação que realizava determinadas operações à base de dados anteriormente desenvolvida numa outra fase deste trabalho prático. Para o resolvermos, esta aplicação foi desenvolvida em java tendo como base a API JDBC, referida anteriormente.

Decidimos adotar este tipo de abordagem visto que nos proporcionaria uma melhor resolução do trabalho.

Os resultados obtidos foram úteis quer do ponto de vista educacional, uma vez que completaram a nossa metodologia.

Palavras chave: JDBC, Java, query, resultSet, preparedStatement, connection, base de dados,...

Abstract

JDBC is an X/Open SQL CLI-based API that allows access (in Java code) to relational databases and execution of SQL commands.

User data provided in DML commands can lead to software vulnerabilities. An attacker can send a valid SQL statement that changes the logic of the initial query. This can allow the attacker to view/modify/remove confidential data from other users or even gain unauthorized access to the system, so it is always necessary to ensure that data is not lost, poorly managed, or even late to third parties.

The problem analyzed was the implementation of an application that performed certain operations on the database developed in an earlier phase of this practical work. To solve this problem, this application was developed in Java using the JDBC API mentioned earlier.

We decided to adopt this approach because it would provide us with a better solution to the work.

The results obtained were useful both from an educational point of view, as they completed our methodology.

Keywords: JDBC, Java, query, resultSet, prepared Statement, connection, database,...

Índice

Lista de Figuras	ix
1 Introdução	1
1.1 Classes do Java.sql	1
1.1.1 Driver Manager	1
1.1.2 Connection	1
1.1.3 Statement	2
1.1.4 PreparedStatement	2
1.1.5 CallableStatement	2
1.1.6 ResultSet	2
1.1.7 SQLException	2
1.2 Divisão do Relatório	2
2 Metodologia	3
2.1 main.java	3
2.2 app.java	3
2.3 model.java	4
3 Resultados	5
3.1 Menu principal	5
3.2 Inserir condutor	5

3.3	Colocar um veículo fora de serviço	6
3.4	Calculo de horas, quilómetros e custo total de um veículo	7
3.5	Mostrar clientes com mais viagens em um ano	7
3.6	Mostrar clientes sem viagens	8
3.7	Mostrar total de viagens dado um NIF	8
3.8	Mostrar informação de um condutor dado um ano	8
4	Conclusões	11
5	Referências	13
A	Anexos	i

Lista de Figuras

3.1	Apresentação do menu principal da aplicação	5
3.2	Inserir um novo condutor	6
3.3	Resultado obtido da imagem 3.2 na aplicação DBeaver	6
3.4	Colocar veículo fora de serviço	6
3.5	Resultado obtido da imagem 3.4 na aplicação DBeaver	6
3.6	Informações do veículo apartir de uma matrícula	7
3.7	Apresentação dos clientes com mais viagens em um dado ano	7
3.8	Apresentação dos clientes sem viagens	8
3.9	Apresentação do total de viagens dado um NIF e um ano	8
3.10	Apresentação da informação do condutor com mais custos dado um ano	9



Introdução

“Uma empresa do sector dos transportes pretende criar um sistema informática para a gestão dos seus condutores, clientes e viagens. A empresa funciona à base de serviços, desse modo nenhum veículo e condutor pertence à empresa, permite sim quem pretenda prestar um serviço adicione à plataforma os veículos e os condutores associados.”

O objetivo deste trabalho prático será implementar uma aplicação em Java que realiza determinadas operações à base de dados anteriormente desenvolvida numa outra fase deste trabalho prático. Mas antes de avançar, vamos dar uma breve introdução teórica sobre o trabalho.

1.1 Classes do Java.sql

1.1.1 Driver Manager

Responsável por carregar os drivers JDBC drivers em memória. Também disponibiliza métodos para criar ligações para fontes de dados.

1.1.2 Connection

Interface que disponibiliza os métodos necessários para realizar uma ligação à fonte de dados, permitindo criar um conjunto de objectos necessários à execução de comandos.

1.1.3 Statement

Interface que permite a execução de comandos SQL estáticos.

1.1.4 PreparedStatement

Interface que permite a manipulação de comandos SQL pré compilados.

1.1.5 CallableStatement

Interface que representa um procedimento armazenado.

1.1.6 ResultSet

Interface que representa o resultado gerado pelo SGBD em resposta a um comando **SELECT**.

1.1.7 SQLException

Classe que permite o acesso aos erros gerados durante a execução de um comando SQL.

1.2 Divisão do Relatório

Este relatório encontra-se dividido em 6 capítulos.

Este capítulo faz uma breve introdução teórica sobre os conteúdos programáticos usados na realização deste trabalho.

O capítulo 2, **Metodologia**, demonstra a nossa abordagem ao trabalho, descrevendo as nossas ideias e maneiras ao resolvermos o problema.

O capítulo 3, **Resultados**, descreve, passo a redundância, os resultados obtidos após concluída a nossa abordagem ao problema.

O capítulo 4, **Conclusões**, retrata os pontos finais do nosso trabalho.

O capítulo 5, **Referências**, especifica o local de onde foram buscadas as informações usadas ao longo do relatório.

O capítulo A, Anexos, especifica os ficheiros, como por exemplo, imagens colocadas em anexo por não se encaixarem no sítio original por serem, por exemplo, muito grandes.

2

Metodologia

Nesta fase do relatório iremos falar sobre a abordagem tomada para o desenvolvimento desta aplicação em Java. Decidimos criar três módulos de modo a dividir o problema em diferentes secções, obtendo assim estruturação do mesmo, e facilitando o seu desenvolvimento e implementação. Esses 3 módulos são: *main.java*, *app.java* e *model.java*.

2.1 main.java

Este módulo é o ponto de começo da execução do programa. Neste caso, este módulo *printa* na consola antes de iniciar o programa os respetivos autores e, logo de seguida, faz uma chamada ao módulo *app.java*, iniciando assim a execução do programa.

2.2 app.java

Este módulo é responsável por toda a escrita e leitura de valores na consola, permitindo assim uma interação simples e intuitiva do utilizador com o programa. Este módulo apresenta um menu de operações numeradas de 1 a 8. Após o utilizador inserir o número da operação que quer realizar, este módulo faz a ligação entre número inserido pelo utilizador com a sua respetiva função do *model.java*. Esta ligação é feita por um *HashMap*. Para cada operação que necessite que o utilizador insira mais dados, este

módulo possui funções que são responsáveis por indicar o formato correto que deve ser inserido, bem como verificar se os dados inseridos correspondem realmente com o formato pedido. Caso isso não se verifique, estas funções certificam-se de informar o utilizador que o formato está incorreto. Isto repete-se até que o utilizador insira um formato válido. Este módulo possui também duas funções auxiliares: `clearConsole` e `printResults`. Na função `printResults`, de modo a printar os resultados de uma forma visualmente apelativa, utilizamos uma utility class retirada do [github](#). O link para esse github encontra-se nas referências do relatório.

2.3 model.java

Este módulo, que é chamado em `app.java`, é responsável pela interação do programa com a base de dados. Aqui existem funções para cada uma das operações que, a partir da execução de queries, com valores passados por parametro (provenientes do que o utilizador insere na consola), permitem adicionar, apagar e/ou obter informação da base de dados. Nas operações em que o programa insere informação na base de dados, é este módulo que verifica se os dados introduzidos pelo utilizador passam nas restrições impostas. Para além disso, também são realizados determinados cálculos com informações proveniente da base de dados, como por exemplo, o cálculo de quilómetros a partir da longitude e latitude do início e do fim da viagem. É importante referir que, em cada função, a primeira coisa a fazer é estabelecer a conexão com a base de dados e no final, é necessário fechar essa conexão. No entanto, em funções que retornam um *ResultSet*, ou seja, funções cuja finalidade é apresentar na consola uma tabela com os dados pedidos, a conexão é somente iniciada e não encerrada. O mesmo deve-se ao facto da classe utilizada para apresentar as tabelas na consola, necessitar que a ligação se mantenha aberta. Assim, a conexão só é encerrada após os resultados terem sido *printados* na consola.



Resultados

3.1 Menu principal

A imagem 3.1 apresenta o menu das possíveis opções quando a aplicação é executada.

Figura 3.1: Apresentação do menu principal da aplicação

```
ISI TP3

1. Exit
2. Insert new driver
3. Put a vehicle out of service
4. Calculate total hours, kilometers and total cost of a vehicle
5. Print the list of client(s) (ID, name and NIF) with the most trips in a given year
6. Print the list of drivers with no trips done
7. Prints the number of trips done by the cars of a given proprietary (NIF) in a given year
8. Given a year, prints information about the driver which had the greatest accumulated cost
>
```

3.2 Inserir condutor

A imagem 3.2 apresenta um exemplo da inserção de um novo condutor, sendo todos os formatos dos dados verificados (data de nascimento, formato da carta de condução, etc.). Na imagem 3.3 podemos observar o sucesso da ação executada, na base de dados, pela aplicação DBeaver.

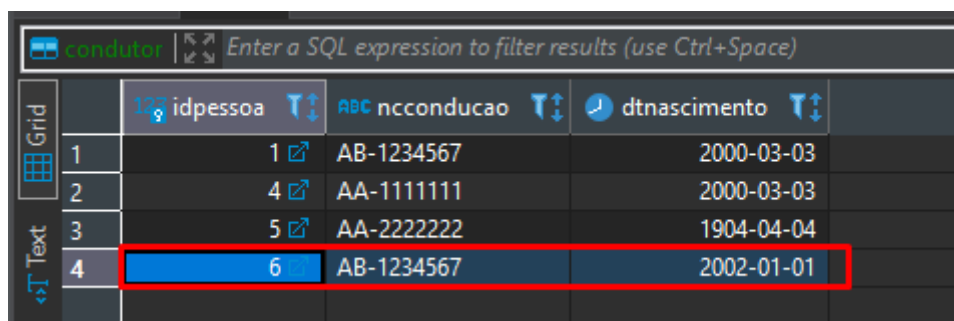
Figura 3.2: Inserir um novo condutor

```

Insert driver
idpessoa: 6
ncconducao (CC-DDDDDD): aa-aaa
Invalid format, try CC-DDDDDD
ncconducao (CC-DDDDDD): AB-1234567
birthday (YYYY-MM-DD): 123
Invalid format, try YYYY-MM-DD
birthday (YYYY-MM-DD): 2002-01-01
Done, press enter to continue.

```

Figura 3.3: Resultado obtido da imagem 3.2 na aplicação DBeaver



	idpessoa	ncconducao	dtnascimento
1	1	AB-1234567	2000-03-03
2	4	AA-1111111	2000-03-03
3	5	AA-2222222	1904-04-04
4	6	AB-1234567	2002-01-01

3.3 Colocar um veículo fora de serviço

A imagem 3.4 apresenta um exemplo da colocação de um veículo fora de serviço a partir de uma matrícula. Na imagem 3.5 podemos observar o sucesso da ação executada, na base de dados, pela aplicação DBeaver.

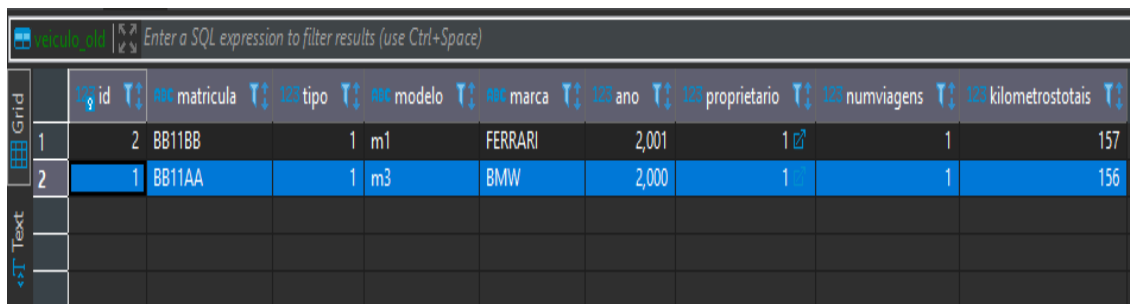
Figura 3.4: Colocar veículo fora de serviço

```

matricula (CCDDCC ou CCDDCC): asdf
Invalid format, try CCDDCC or CCDDCC
matricula (CCDDCC ou CCDDCC): BB11AA
Done, press enter to continue.

```

Figura 3.5: Resultado obtido da imagem 3.4 na aplicação DBeaver



	id	matricula	tipo	modelo	marca	ano	proprietario	numviagens	kilometrostotais
1	2	BB11BB	1	m1	FERRARI	2,001	1	1	157
2	1	BB11AA	1	m3	BMW	2,000	1	1	156

3.4 Calculo de horas, quilómetros e custo total de um veículo

A imagem 3.6 apresenta um exemplo do cálculo do total de horas, quilómetros e preço total de um veículo a partir de uma matrícula. A aplicação mostra os veículos existentes ao utilizador antes de pedir a matrícula.

Figura 3.6: Informações do veículo a partir de uma matrícula

```
Printing 4 rows from table veiculo
+-----+-----+-----+-----+-----+-----+-----+
| id | matricula | tipo | modelo | marca | ano | proprietario |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | BB11AA | 1 | m3 | BMW | 2000 | 1 |
+-----+-----+-----+-----+-----+-----+-----+
| 2 | BB11BB | 1 | m1 | FERRARI | 2001 | 1 |
+-----+-----+-----+-----+-----+-----+-----+
| 3 | BB12AA | 1 | m2 | FORD | 2000 | 2 |
+-----+-----+-----+-----+-----+-----+-----+
| 4 | BB11BB | 1 | m4 | RangeRover | 2001 | 3 |
+-----+-----+-----+-----+-----+-----+-----+

matricula (CCDDCC ou CCDDCC): BB11AA
Info of Vehicle BB11AA
Total Hours: 2 Total Distance: 156 Total Cost: 7.0
Done, press enter to continue.
█
```

3.5 Mostrar clientes com mais viagens em um ano

A imagem 3.7 apresenta um exemplo da apresentação dos clientes com mais viagens em um dado ano, introduzido pelo utilizador.

Figura 3.7: Apresentação dos clientes com mais viagens em um dado ano

```
year: 2004
List of drivers with most trips done:
Printing 2 rows from tables viagem, pessoa,
+-----+-----+-----+-----+-----+
| id | nproprio | apelido | nif | totalviagens |
+-----+-----+-----+-----+-----+
| 1 | Antonio | Silva | 11122233344 | 1 |
+-----+-----+-----+-----+-----+
| 4 | Afonso | Mateus | 12345678999 | 1 |
+-----+-----+-----+-----+-----+

Done, press enter to continue.
█
```

3.6 Mostrar clientes sem viagens

A imagem 3.8 apresenta um exemplo da apresentação dos clientes sem viagens.

Figura 3.8: Apresentação dos clientes sem viagens

```
List of drivers with no trips done:
Printing 1 row from table pessoa
+-----+-----+-----+-----+
| id | nproprio | apelido | nif |
+-----+-----+-----+-----+
| 5 | Vasco | Constantinople | 22222222222 |
+-----+-----+-----+-----+

Done, press enter to continue.
█
```

3.7 Mostrar total de viagens dado um NIF

A imagem 3.9 apresenta um exemplo da apresentação do total de viagens dado um NIF de um proprietário em um dado ano.

Figura 3.9: Apresentação do total de viagens dado um NIF e um ano

```
nif: 11122233344
year: 2004
Printing 2 rows from tables veiculo,
+-----+-----+
| id | numviagens |
+-----+-----+
| 1 | 1 |
+-----+-----+
| 2 | 1 |
+-----+-----+

Done, press enter to continue.
█
```

3.8 Mostrar informação de um condutor dado um ano

A imagem 3.10 apresenta um exemplo da apresentação da informação de um condutor dado um ano (condutor com mais custos).

Figura 3.10: Apresentação da informação do condutor com mais custos dado um ano

```
year: 2004
Printing 1 row from tables pessoa,
+-----+-----+-----+-----+-----+
| nproprio | apelido | noident | morada | totalvalfinal |
+-----+-----+-----+-----+-----+
| Afonso   | Mateus  | 11111   | Rua da Alface | (numeric) |
+-----+-----+-----+-----+-----+

Done, press enter to continue.
█
```

4

Conclusões

Ao longo do trabalho, foram aplicados os conhecimentos dados em aula, tais como a class Statement, ResultSet e muitos outros já referidos na Introdução, 1.

Na aplicação destes conhecimentos, foram obtidos os resultados esperados: a aplicação em java que suporta as operações pedidas e explicadas em 2, que a nosso ver representam fielmente o problema original referenciado também na Introdução, 1.

Não obtivemos quaisquer problemas enquanto realizavamos este trabalho, no entanto surgiram sim, dúvidas, sendo uma delas como iríamos apresentar os resultados das instruções, o qual foi resolvido usando a classe DBTablePrinter.

Em relação a trabalhos futuros, visto que não obtivemos quaisquer problemas, mas sim certas dúvidas ocasionais posteriormente escalrecidas, consideramos continuar com o nosso atual método de trabalho.



Referências

JDBCTM API Tutorial and Reference (3rd ed)

M. Fisher, J. Ellis, J. Bruce

Prentice Hall (2003);

SR 221: JDBCTM 4.0 API Specification

<https://www.jcp.org/en/home/index>

Sun Microsystem;

JDBC Recipes: A Problem-Solution Approach M. Parsian

Apress (2005)

ISBN-10: 1590595203;

Material de Apoio de SI1: JDBC

N. Datia

ISEL(2016)

Utility Class

<https://github.com/htorun/dbtableprinter>



Anexos

