Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Reserva de Bicicletas

Autores:	436378	Berto Afonso Pita Vieira Taborda Barata
	49504	Jorge Filipe de Medeiros Palácios da Silva

Relatório para a Unidade Curricular de Programação da Licenciatura em Engenharia Informática e de Computadores

Professor: Engenheira Matilde P. M. Pato

06 – 01 – 2023

<< Esta página foi intencionalmente deixada em branco >>

Resumo

No domínio da Unidade Curricular de Introdução a Sistemas de Informação, fomos encarregues de estabelecer uma ligação ao Sistema de Gerenciamento de Base de Dados, SGBD, uma interface entre vários usuários e a base de dados, utilizando Java Database Connectivity, JDBC, uma API baseada em X/Open SQL CLI que permite o acesso, em java, a base de dados relacionais e a execução em comandos SQL.

Para tal, foi utilizado o texto com os requisitos do sistema, apresentados na primeira fase do projeto, e o modelo de dados que foi implementado na segunda fase. A partir daí foi necessário desenvolver funções tais como a opção para inserir uma nova bicicleta elétrica ou clássica, ou colocar uma bicicleta em manutenção.

Abstract

Within the scope of the curricular unit Introduction of Information Systems, we carried out establishing a connection to the Data Base Management System, DBMS, an interface between users and the database, using Java Database Connectivity, an API based on X/Open SQL CLI that allows access, in java, the relational database and the execution in SQL commands.

To carry out this step, the text with the system requirements, presented in the first phase of the project, and the data model in the second phase were used. From then on, it was necessary to develop functions such as the option to insert a new classic bike or electric bike or put a bike on maintenance.

Índice

Listagens

1. Introdução

Pretende-se desenvolver um sistema de informação de reservas de bicicletas para uma empresa de turismo “GoCycle”. A empresa possui várias lojas. Para tal, é necessário desenvolver uma aplicação Java que permita realizar as seguintes operações: Opção para inserir uma nova bicicleta; Opção para colocar uma bicicleta em manutenção baseado no id, não podendo assim ser reservada; Opçã para calcular a média em termos de autonomia, de velocidade e de bateria do dispositivo para cada bicicleta elétrica dentro de uma marca; Opção para apresentar todas as lojas que tenham efectuado mais de 5 reservas, até à presente data, e enumere-as por ordem decrescente do número de reservas; Opção que apresente a lista de clientes com mais reservas efetuadas num espaço de tempo; Opção que apresente o conjunto de dispositivos de bicicletas cujo estado encontra-se em manutenção.

2. Modelo de dados conceptual

Foi usado o modelo ER utilizado na segunda fase do trabalho. Sendo algumas das funções da aplicação também algumas implementadas nessa fase, adaptando-as agora para linguagem JDBC.

3. Restrições

Para as restrições de integridade apenas implementamos duas que foram dadas inicialmente pelo enunciado. A primeira restrição é que uma bicicleta que se encontre em manutenção não pode ser reservada. A segunda restrição é que o sistema de mudanças pode ser alterado dinamicamente

```
public class Restriction {

    // Restrição: uma bicicleta que se encontra em manutenção não pode ser reservada
    no usages
    public static boolean verifyRestriction1(Bike bike) {
        return !bike.getEstado().equalsIgnoreCase( anotherString: "em manutenção");
    }

    // Restrição: o sistema de mudanças pode ser alterado dinamicamente
    no usages
    public static boolean verifyRestriction2() { return true; // Sem restrição para alteração do sistema de mudanças }
}
```

Listagem 1- Implementação das restrições

4. App

RT é a classe que possui as restrições todas. A primeira restrição é o limite de 20 veículos por proprietário, em que quando excede o limite, a tabela é atualizada deixando os top 20 veículos do proprietário.

```
public void rt1() {
    try {
        String jdbcURL = "jdbc:postgresql://10.62.73.73:5432/";
        String username = "mp25";
        String password = "mp25";
        Connection conn = DriverManager.getConnection(jdbcURL, username, password);
        Statement sta = conn.createStatement();
        ResultSet rs = sta.executeQuery(queries.t1);

        while (rs.next()) {
            if (rs.getInt( columnLabel: "n_veiculos") > 20) {
                Statement stm = conn.createStatement();
                stm.executeUpdate(update.delete_veiculos(rs.getInt( columnLabel: "idpessoa")));
            }
        }

        rs.close();
        sta.close();
        conn.close();

    } catch (SQLException E) {
        E.printStackTrace();
    }
}
```

Listagem 1 Implementação da primeira restrição

```
public void rt2() {
    try {
        String jdbcURL = "jdbc:postgresql://10.62.73.73:5432/";
        String username = "mp25";
        String password = "mp25";
        Connection conn = DriverManager.getConnection(jdbcURL, username, password);
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(queries.t2);

        while (rs.next()) {
            Statement stm = conn.createStatement();
            stm.executeUpdate(update.delete_condutor(rs.getInt( columnLabel: "idpessoa")));
        }
        rs.close();
        st.close();
        conn.close();

    } catch (SQLException exception) {
        exception.printStackTrace();
    }
}
```

A segunda restrição é eliminar os condutores com os mesmos id's de proprietários, garantindo assim que não existe condutores proprietários. Se existir algum condutor com o id do proprietário, este é eliminado.

Listagem 2 Implementação da segunda restrição

A terceira restrição é garantir que o número da carta de condução e a matrícula de um veículo cumprem o determinado formato para as tabelas. Esta restrição foi préviamente feita na fase anterior.

A quarta restrição é a verificação da existência da tabela veiculo_old e, caso não exista, é criada uma. Verifica também se há carros com mais de 5 anos, havendo estes são adicionados à nova tabela veiculo_old como todo os seus parâmetros com o número de viagens e os kilómetros percorridos e elemina os carros com mais de 5 anos da tabela veiculo.

5. Model

O ficheiro Model é onde se encontram as funções todas que são usadas na aplicação que se deseja desenvolver.

A função addClassicBike e addElectricBike adicionam uma bicicleta elétrica ou clássica, respetiva ao nome da função. Assim como todas as outras funções utiliza uma instrução SQL preparada para evitar injeção de SQL. São usados como parâmetros dados da bicicleta presentes nas respetivas classes ClassicBike e ElectricBike para preencher os valores na instrução SQL.

A função updateBikeState serve para atualizar o estado de uma bicicleta com base no ID. O ID é usado para identificar a bicicleta a ser atualizada. Realiza a conexão com o banco de dados, executa a atualização e imprime o resultado.

A função calculateAverageMetricforElectricBikes é utilizada para calcular a média de autonomia, velocidade máxima e bateria das bicicletas elétricas de uma determinada marca. A marca é passada como parâmetro para a query. Utiliza como parâmetro selectedBrand A marca das bicicletas elétricas para as quais calcular as médias.

A função getClientsWithReservations serve para obter os clientes com mais reservas no ano ou espaço temporal especificado. Utiliza uma instrução SQL preparada para evitar injeção de SQL. Recebe o ano ou intervalo temporal como parâmetro, executa a consulta no banco de dados e imprime os resultados.

A função getGPSDevice é um método estático para obter os dispositivos GPS de bicicletas em manutenção. Utiliza uma instrução SQL preparada para evitar injeção de SQL. Executa a consulta no banco de dados e imprime os resultados.

A função getStores serve para obter as lojas com mais de um número mínimo de reservas. Utiliza uma instrução SQL preparada para evitar injeção de SQL. Executa a consulta no banco de dados e imprime os resultados. Recebe como parâmetro o número mínimo de reservas para filtrar as lojas.

Por último a função getManagers serve para obter informações dos gerentes que efetuaram reservas utilizando também uma instrução SQL preparada para evitar injeção de SQL.

6. App

O ficheiro App.java constroi a app chamando as funções presentes no Model.java. Dentro da class App, tem uma enum class Option que enumera as opções que o utilizador têm ao iniciar a app e tem um função App que escolhendo uma opção dentro da enum class, é executada a função destinada a esse propósito. Tem também uma função DisplayMenu utilizada para apresentar as opções ao utilizador.

```
private enum Option
{
    // DO NOT CHANGE ANYTHING!
    1 usage
    Unknown,
    1 usage
    Exit,
    1 usage
    novelBike,
    1 usage
    updateBikeState,
    1 usage
    calculateAverageMetricsForElectricBikes,
    1 usage
    getClientsWithReservations,
    1 usage
    getGPSDevice,
    1 usage
    getStores,
    1 usage
    getManagers,
}
```

Listagem 3 Enumeração das opções do utilizador

```
private App()
{
    // DO NOT CHANGE ANYTHING!
    __dbMethods = new HashMap<Option,DbWorker>();
    __dbMethods.put(Option.novelBike, () -> App.this.novelBike());
    __dbMethods.put(Option.updateBikeState, () -> App.this.updateBikeState());
    __dbMethods.put(Option.calculateAverageMetricsForElectricBikes, () -> App.this.calculateAverageMetricsForElectricBikes());
    __dbMethods.put(Option.getClientsWithReservations, () -> App.this.getClientsWithReservations());
    __dbMethods.put(Option.getGPSDevice, new DbWorker() {public void doWork() {App.this.getGPSDevice();}});
    __dbMethods.put(Option.getStores, new DbWorker() {public void doWork() {App.this.getStores();}});
    __dbMethods.put(Option.getManagers, new DbWorker() {public void doWork() {App.this.getManagers();}});

}
```

Listagem 4 Função App

Listagem 5 Função utilizada para a apresentação das opções ao utilizador

```
private Option DisplayMenu()
{
    Option option = Option.Unknown;
    try
    {
        // DO NOT CHANGE ANYTHING!
        System.out.println("Bicycle reservation");
        System.out.println();
        System.out.println("1. Exit");
        System.out.println("2. Novel bikes");
        System.out.println("3. Update bike states");
        System.out.println("4. Average metrics for electric bikes");
        System.out.println("5. List of clients with reservations");//2j
        System.out.println("6. List of devices");//2e
        System.out.println("7. List of stores");//3f
        System.out.println("8. List of managers and made reservations");//3h
        System.out.print(">");
        Scanner s = new Scanner(System.in);
        int result = s.nextInt();
        option = Option.values()[result];
    }
    catch(RuntimeException ex)
    {
        //nothing to do.
    }
    return option;
```

Para o uso da aplicação é necessário usar alguns comandos por cada opção:

1- Peso, modelo, raio, mudança e caso seja elétrica, autonomia e velocidade;
2- Id da bicicleta
3- Marca da bicicleta elétrica.

7. Conclusão

O principal problema foi a conexão com o postgres e instalação das bibliotecas. O trabalho também não foi possivél a testagem devido a problemas de conexão com a base de dados visto terem ports diferentes, sendo que a base de dados tava com o port 5432 e na aplicação não foi possivel usar esse port usado o port 5433.