Trabalho da Disciplina de Fundamentos de Banco de Dados

Frederico Hansel dos Santos Gassen

João Vitor Forgearini Beltrame

Centro de Tecnologia – Prédio 7 – Universidade Federal de Santa Maria (UFSM) - Av. Roraima nº 1000 – Bairro Camobi – Santa Maria – RS - Brasil

fhgassen@inf.ufsm.br

jvbeltrame@inf.ufsm.br

1. Descrição do trabalho

Uma Universidade deseja construir um sistema para facilitar e agilizar o controle de acesso de veículos aos seus cinco estacionamentos para prover mais segurança e comodidade para os seus usuários. O sistema deverá permitir que se cadastre todos os tipos de usuários (alunos, professores e funcionários), que receberão um cartão com um código de barra para sua identificação. Cada usuário poderá solicitar o cadastramento de vários veículos com os quais utiliza os estacionamentos da universidade. Ao chegar a qualquer portão de acesso à universidade, o vigilante irá informar a placa do veículo e o usuário deverá passar o cartão magnético em um leitor de código de barras, e com isso, o sistema irá identificar se o veículo está relacionado com a identificação do usuário. Ao sair, o usuário simplesmente passará o seu cartão em outra leitora de código de barras. O visitante (usuário não cadastrado) deverá pegar um cartão especial com os vigilantes. Através desses procedimentos, o sistema poderá fornecer dados de ocupação de cada estacionamento, além de permitir a consulta de quais os veículos estão, ou estiveram, dentro da universidade em um determinado dia e horário.

O trabalho começou com a instalação da IDE NetBeans, juntamente com o pacote JDBC para a manipulação do banco de dados MySQL na linguagem Java. Logo após foi criado o banco de dados "estacionamento_rotativo". Eis o script usado para a criação do banco:

```
CREATE TABLE `cargo` (
  `id_cargo` int(11) NOT NULL AUTO_INCREMENT,
  `nome_cargo` varchar(50) NOT NULL,
  PRIMARY KEY (`id_cargo`)
) AUTO_INCREMENT=0;

INSERT INTO `cargo` (`id_cargo`, `nome_cargo`) VALUES
        (1, 'Aluno'),
        (2, 'Professor'),
```

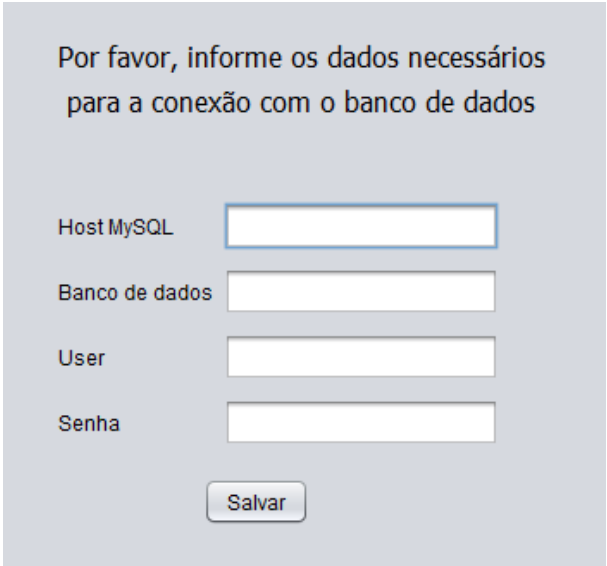
```
        (3, 'Funcionário'),
        (4, 'Visitante');

CREATE TABLE `estacionamento` (
  `id_estacionamento` int(11) NOT NULL AUTO_INCREMENT,
  `nome_estacionamento` varchar(50) NOT NULL,
  `vagas_estacionamento` int(11) NOT NULL,
  `vagas_ocupadas_estacionamento` int(11) NOT NULL,
  PRIMARY KEY (`id_estacionamento`)
) AUTO_INCREMENT=1;

CREATE TABLE `usuario` (
  `id_usuario` int(11) NOT NULL AUTO_INCREMENT,
  `id_cargo` int(11) NOT NULL DEFAULT '0',
  `nome_usuario` varchar(50) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id_usuario`),
  KEY `fk_cargo_usuario` (`id_cargo`),
  CONSTRAINT `fk_cargo_usuario` FOREIGN KEY (`id_cargo`) REFERENCES
`cargo` (`id_cargo`) ON DELETE NO ACTION ON UPDATE NO ACTION
) AUTO_INCREMENT=1;

CREATE TABLE `veiculo` (
  `placa_veiculo` varchar(8) NOT NULL,
  `id_usuario` int(11) DEFAULT NULL,
  PRIMARY KEY (`placa_veiculo`),
  KEY `fk_veiculo_usuario` (`id_usuario`),
  CONSTRAINT `fk_veiculo_usuario` FOREIGN KEY (`id_usuario`)
REFERENCES `usuario` (`id_usuario`) ON DELETE NO ACTION ON UPDATE
NO ACTION
);

CREATE TABLE `estacionada` (
  `placa_veiculo` varchar(8) NOT NULL DEFAULT '0',
  `id_estacionamento` int(11) NOT NULL DEFAULT '0',
```

```
 `datahora_entrada_estacionada` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
 `datahora_saida_estacionada` timestamp NULL DEFAULT NULL,
 PRIMARY KEY (`placa_veiculo`,`datahora_entrada_estacionada`),
 KEY `fk_estacionamento_estacionada` (`id_estacionamento`),
 CONSTRAINT `fk_estacionamento_estacionada` FOREIGN KEY (`id_estacionamento`) REFERENCES `estacionamento` (`id_estacionamento`) ON DELETE NO ACTION ON UPDATE NO ACTION,
 CONSTRAINT `fk_veiculo_estacionada` FOREIGN KEY (`placa_veiculo`) REFERENCES `veiculo` (`placa_veiculo`) ON DELETE NO ACTION ON UPDATE NO ACTION
);

INSERT INTO `estacionamento` (`id_estacionamento`, `nome_estacionamento`, `vagas_estacionamento`, `vagas_ocupadas_estacionamento`) VALUES
	(1, 'Estacionamento 1', 100, 0),
	(2, 'Estacionamento 2', 200, 0),
	(3, 'Estacionamento 3', 300, 0),
	(4, 'Estacionamento 4', 400, 0),
	(5, 'Estacionamento 5', 500, 0);

INSERT INTO `usuario` (`id_usuario`, `id_cargo`, `nome_usuario`) VALUES
	(3, 4, 'Visitante');

CREATE TABLE `vagas_disponiveis` (
	`id_estacionamento` INT(11) NOT NULL,
	`nome_estacionamento` VARCHAR(50) NOT NULL,
	`vagas_disponiveis` BIGINT(12) NOT NULL
);

INSERT INTO `veiculo` (`placa_veiculo`, `id_usuario`) VALUES
	('XXX-0000', 3);

DELIMITER //
CREATE TRIGGER `entrada_carro` AFTER INSERT ON `estacionada` FOR EACH ROW BEGIN
```

```
UPDATE estacionamento SET vagas_ocupadas_estacionamento = vagas_ocupadas_estacionamento + 1 WHERE id_estacionamento = NEW.id_estacionamento;
END//

CREATE TRIGGER `saida_carro` AFTER UPDATE ON `estacionada` FOR EACH ROW BEGIN
    UPDATE estacionamento SET vagas_ocupadas_estacionamento = vagas_ocupadas_estacionamento - 1 WHERE id_estacionamento = NEW.id_estacionamento;
END//
DELIMITER ;

CREATE VIEW `mostra_vagas_disponiveis` AS select `estacionamento`.`id_estacionamento` AS `id_estacionamento`,`estacionamento`.`nome_estacionamento` AS `nome_estacionamento`,(`estacionamento`.`vagas_estacionamento` - `estacionamento`.`vagas_ocupadas_estacionamento`) AS `vagas_disponiveis` from `estacionamento`;
```

E, então, começamos a implementação do programa em Java:

Na primeira execução do programa, uma janela exigindo os dados para a conexão com o banco de dados é aberta:



Ao salvar, os dados são criptografados e gravados. Sempre que o programa iniciar, irá buscar esses dados, descriptografá-los e realizar a conexão com o banco de

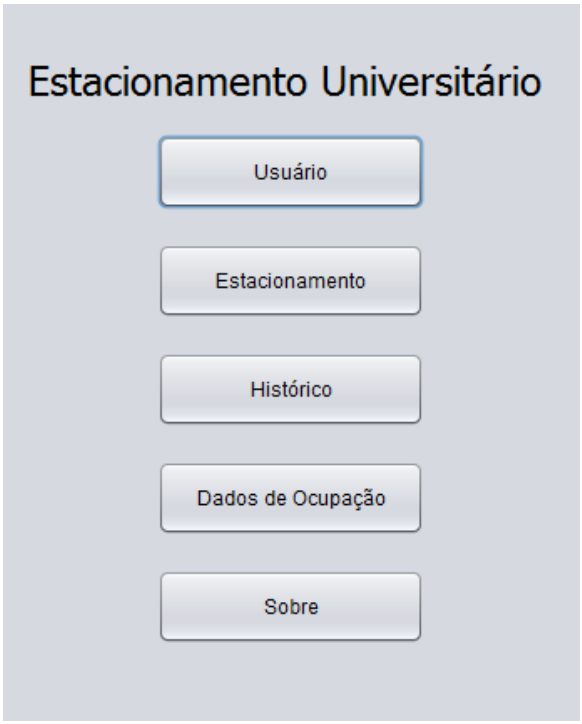
dados através do método abaixo, que está presente na classe “Conexao”, dentro do pacote “utilidade”:

```
public Conexao() throws SQLException{
    String dados[]= abreDados();
    String host = "/" + dados[0];
    String banco = "/" + dados[1];
    String login = dados[2];
    String senha = dados[3];
    String url = "jdbc:mysql:/"+host+banco;

    conexao = DriverManager.getConnection(url, login, senha);
}
```

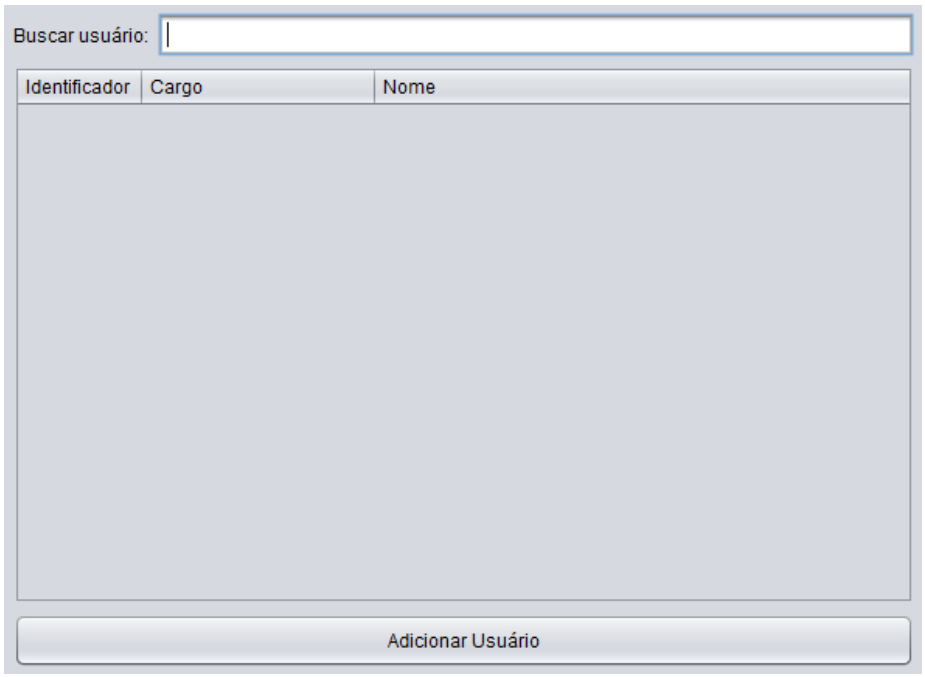
Ainda na classe “Conexao”, existe a opção para inserir e retirar dados no banco de dados ou realizar a pesquisa no mesmo, realizadas pelos seguintes métodos:

```
public ResultSet select(String query) throws SQLException {
    st=conexao.createStatement();
    st.executeQuery(query);
    rs=st.getResultSet();
    return rs;
}

public void insert (String query) throws SQLException {
    st=conexao.createStatement();
    st.executeUpdate(query);
}
```

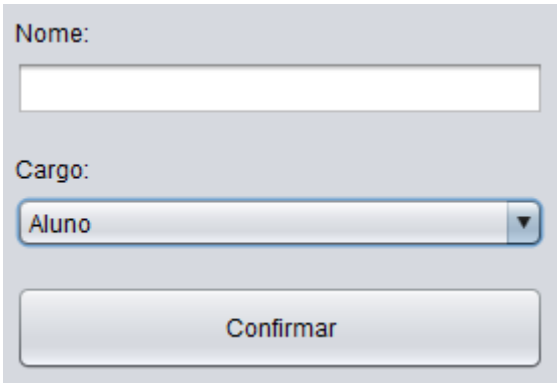
Na janela inicial do programa está presente o MENU, que oferece ao administrador a escolha entre gerir os usuários, administrar entrada e saída dos estacionamentos, histórico de cada usuário estacionado, os dados de ocupação dos estacionamentos e ler sobre o trabalho (na opção “Sobre”).



Na tela USUÁRIO, há a opção de pesquisar o usuário pelo nome. Há também uma tabela que lista todos os usuários cadastrados com o seu relativo Identificador e cargo. Abaixo disso existe a opção de adicionar um novo usuário.



Caso selecione “Adicionar Usuário”, abrirá a janela abaixo, exigindo o nome e o cargo do novo usuário.



Ao selecionar “Confirmar”, a conexão com o banco de dados é realizada, executando um Insert:

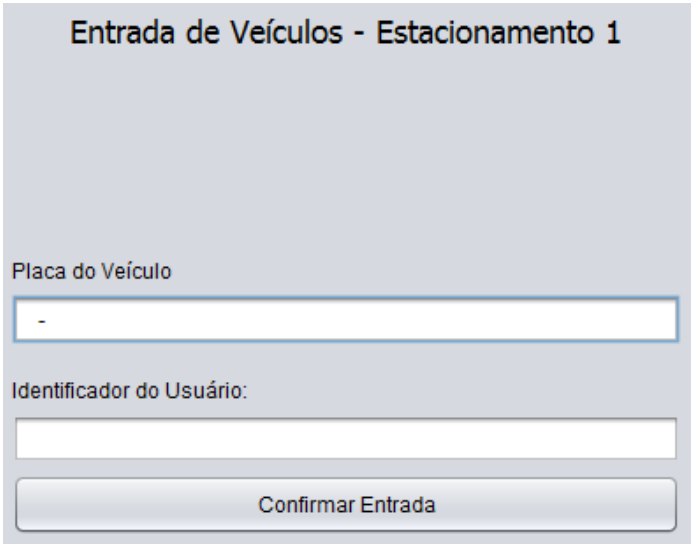
Caso o usuário não exista, o seguinte comando é realizado:

```
con.insert("INSERT INTO usuario (id_cargo, nome_usuario) VALUES ("+
        cargo+", '"+jTFNome.getText()+"')");
```

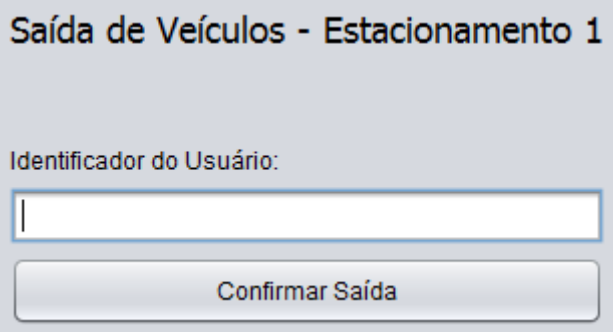
Caso o usuário já exista, é realizado um update com o novo cargo do mesmo:

```
con.insert("UPDATE usuario SET id_cargo="+cargo+", nome_usuario='"+
      jTFNome.getText()+"' WHERE id_usuario="+this.usuario.getId_usuario());
```

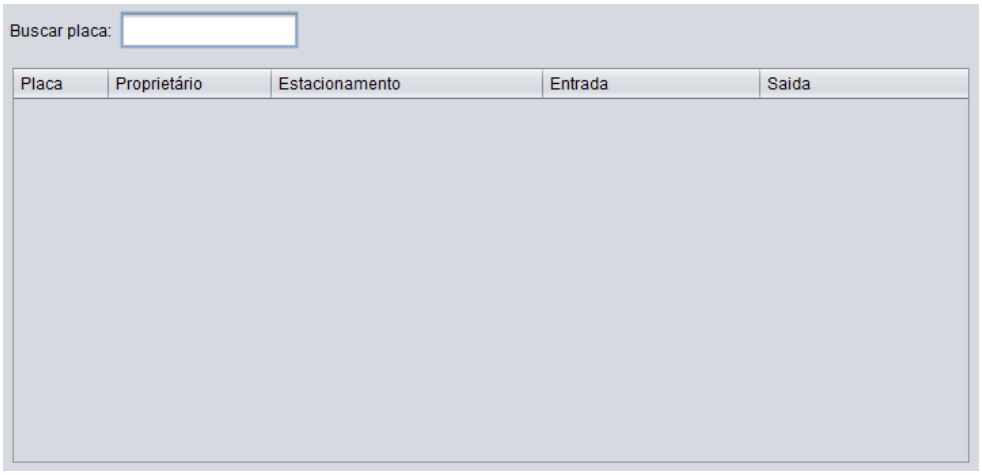
Na tela de MENU, se o usuário do programa selecionar a opção “Estacionamento”, é perguntado para o mesmo se ele quer administrar a entrada ou a saída de um dos cinco estacionamentos. Se escolher “Entrada”, a janela abaixo é exibida, perguntando a placa do veículo e o número identificador do usuário dono do veículo com a placa inserida para registrar a entrada do usuário no estacionamento.



Caso escolha a opção “Saída”, a tela abaixo é exibida. Nessa tela só é necessário inserir o identificador do usuário para registrar que o mesmo saiu do estacionamento.



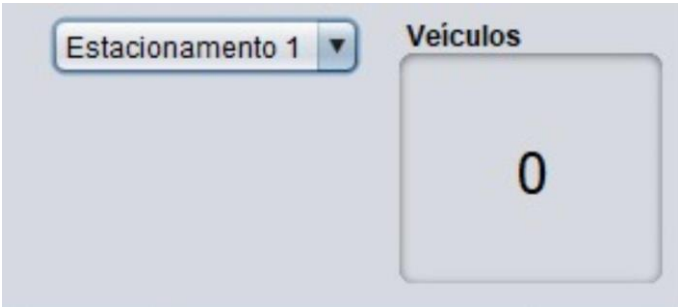
A entrada e saída de veículos no estacionamento podem ser visualizadas na tela de HISTÓRICO (imagem abaixo), que pode ser selecionada na tela de MENU. Ao acessar o histórico, é possível pesquisar o histórico pela placa do veículo no campo de pesquisa, ou visualizar todos os veículos estacionados, ordenados por Id do usuário, na tabela de histórico, contendo a placa do veículo, o proprietário do veículo, o estacionamento que estava estacionado e a data e hora da entrada e da saída do usuário no estacionamento.



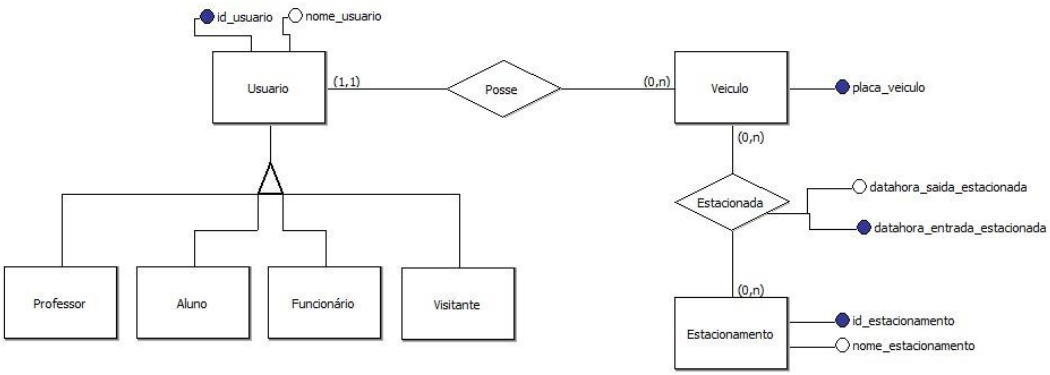
A pesquisa dos dados para serem exibidos na tabela de histórico é realizada da seguinte maneira:

```
try {
    ResultSet rs = con.select("SELECT v.placa_veiculo, u.nome_usuario,"
            + " es.nome_estacionamento, e.datahora_entrada_estacionada,"
            + " e.datahora_saida_estacionada FROM estacionada e "
            + "JOIN estacionamento es ON es.id_estacionamento=e.id_estacionamento "
            + "JOIN veiculo v ON v.placa_veiculo=e.placa_veiculo JOIN usuario u ON "
            + "u.id_usuario=v.id_usuario");
    while (rs.next()) {
        estacionadas.add(new Estacionada(rs.getString("v.placa_veiculo"),
                rs.getString("u.nome_usuario"),
                rs.getString("es.nome_estacionamento"),
                rs.getString("e.datahora_entrada_estacionada"),
                rs.getString("e.datahora_saida_estacionada")));
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(this, "SQLException: " + ex,
            "Estacionamento Universitário", JOptionPane.ERROR_MESSAGE);
}
```

Na tela de MENU, ao selecionar a opção “Dados de Ocupação”, a janela abaixo é exibida, nela pode-se conferir quantas vagas estão ocupadas em cada um dos cinco estacionamentos.



2. Diagrama de Classe



3. Funções implementadas

3.1. Auto incremento

Foi implementado auto incremento em todas as chaves primárias cujo nome é id no banco de dados para que a cada inserção o próprio banco de dados incremente o id anterior e guarde o novo registro

3.2. Trigger

Foram implementados dois triggers: um quando um carro entra no estacionamento e outro quando o carro sai do estacionamento. Eles incrementam ou decrementam a coluna vagas_ocupadas_estacionamento na tabela estacionamento para saber quantos carros há estacionados em cada estacionamento.

Trigger de entrada:

```
CREATE TRIGGER `entrada_carro` AFTER INSERT ON `estacionada` FOR EACH ROW
BEGIN
```

UPDATE estacionamento SET vagas_ocupadas_estacionamento = vagas_ocupadas_estacionamento + 1 WHERE id_estacionamento = NEW.id_estacionamento;

END

Trigger de saída:

CREATE TRIGGER `saida_carro` AFTER UPDATE ON `estacionada` FOR EACH ROW

BEGIN

UPDATE estacionamento SET vagas_ocupadas_estacionamento = vagas_ocupadas_estacionamento - 1 WHERE id_estacionamento = NEW.id_estacionamento;

END

3.3. View

Foi implementada uma view que mostra quantas vagas disponíveis há em cada estacionamento:

CREATE VIEW vagas_disponiveis AS SELECT id_estacionamento, nome_estacionamento, (vagas_estacionamento-vagas_ocupadas_estacionamento) AS vagas_disponiveis FROM estacionamento

4. Facilidades e dificuldades encontradas na realização do trabalho

Eu, Frederico, vim de um curso Técnico em Informática onde o estágio obrigatório requerido foi realizado implementando um sistema exatamente com a mesma ideia deste trabalho. Logo, ele foi mais trivial de ser realizado. Além disso, deixo minha opinião positiva para com a realização do mesmo nos semestres seguintes, visto que ele aplica na prática os conceitos vistos em sala de aula.

Já para mim, João, a realização desse trabalho foi mais desafiadora, visto que é o meu primeiro contato com o manuseio de um Banco de Dados, o que deixou o trabalho um tanto quanto interessante, pois foi possível aprender muito sobre como administrar e manipular um Banco de Dados, tanto através da pesquisa e das aulas, quanto com a ajuda do meu colega de trabalho, que já possui uma certa experiência nessa área.