Universidade Federal de Viçosa
Campus Florestal
Ciência da Computação

Trabalho Prático 3

Tópicos especiais

Alunos: Gustavo Graf de Sousa
Samuel José Jardim da Silva
Gerferson Rodrigues Coelho
Valdiney Soares de Araujo
Professor: Daniel Mendes Barbosa

Outubro
2017

Universidade Federal de Viçosa
Instituto de Ciências Exatas e Tecnológicas

Relatório

Aluno: Gustavo Graf de Sousa
Matrícula: 1283
Aluno: Samuel José Jardim da Silva
Matrícula: 1309
Aluno: Gerferson Rodrigues Coelho
Matrícula: 1763
Aluno: Valdiney Soares de Araújo
Matrícula: 1789
Professor: Daniel Mendes Barbosa

Outubro
2017

Conteúdo

1 Introdução **1**

2 Desenvolvimento **2**

2.1 Conceitos básicos 5

3 Conclusão **8**

1 Introdução

Os bancos de dados são utilizados para armazenar diversos tipos de informações, desde dados sobre uma conta de e-mail até dados importantes da Receita Federal.

Uma das principais dificuldaes enfrentadas na manuteção e controle do banco de dados e a segurança pois possui a mesmas dificuldades que a segurança da informação enfrenta, que é garantir a integridade, a disponibilidade e a confidencialidade. Um Sistema gerenciador de banco de dados deve fornecer mecanismos que auxiliem nesta tarefa e trabalhar com algum sistemma de credencial ou permissões de acesso. O nosso trabalho consiste na elaboração de uma aplicação para realização de controle de usuarios fornecendo acesso seguro para usuarios e liberando acesso aos dados do banco especificando o usuario e sua permisão de acesso. Na nossa aplicação cada usuário tem um domínio de segurança, um conjunto de propriedades que determinam coisas como ações (privilégios e papeis , ações) disponíveis para o usuário; Acesso ainformação, busca, modificar etc.

2 Desenvolvimento

Para a realização deste trabalho diversas ferramentas com o intuito de facilitar o trabalho foram utilizadas para o desenvolvimento do mesmo. Foram usadas as seguintes ferramentas: *Astah*, *MySql Workchcbench*, *GitHub*, *NetBeans*. O *Astah* foi usado para a criação do diagrama de classes que ajudou a modelar o problema e consequentemente a entendermos melhor o mesmo. O *MySql Workbench* foi usado para modelar e criar o banco de dados, já o *GitHub* foi usado para o versionamento do projeto. E por último o *NetBeans* foi utilizado para implementarmos o projeto na linguagem *JAVA*.

A primeira parte do projeto consistiu na modelagem do diagrama de classes e do banco de dados. No diagrama de classes foi utilizado o padrão de projeto *DAO - Data Access Object* já pensando na conexão com o banco de dados. O diagrama de classes pode ser visto na figura abaixo

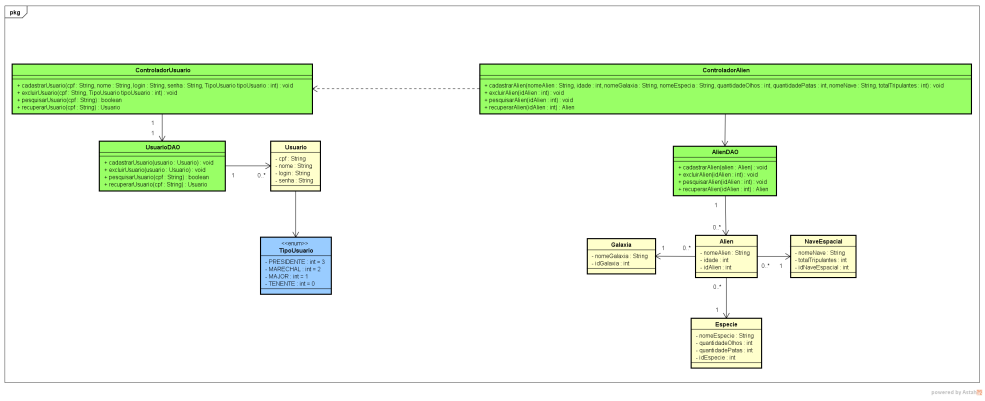

Figura 1: Diagrama de Classes

Para fazer a conexeão do banco de dados com o *NetBeans* foi necessário realizar o *download* do *mysql-connector* e adicioná-lo ao projeto. Na imagem a seguir, mostra-se como adicionar o *mysql-connector*. Após isso, foi criado uma classe chamada *Connection Factory* que têm como responsabilidade fazer a conexão em si com o banco de dados. Para isto se concretizar, foi criado um método chamado *abreConexao* que retorna um objeto do tipo *Connection*. Neste método foi usado também a classe *DriverManager* e seu método *getConnection* e passando três parâmetros para ele, que são: local em que o banco de dados foi armazenado, o usuário e a senha. Este processo pode ser visto na figura abaixo

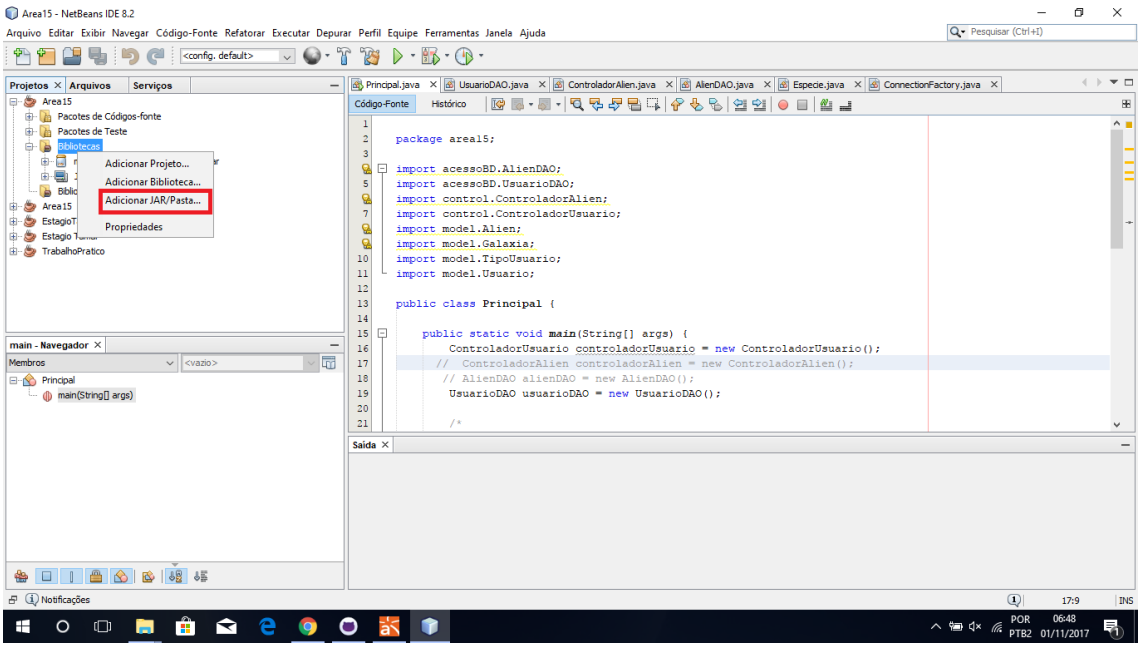

Figura 2: Adicionando mysql-connector - Passo 1

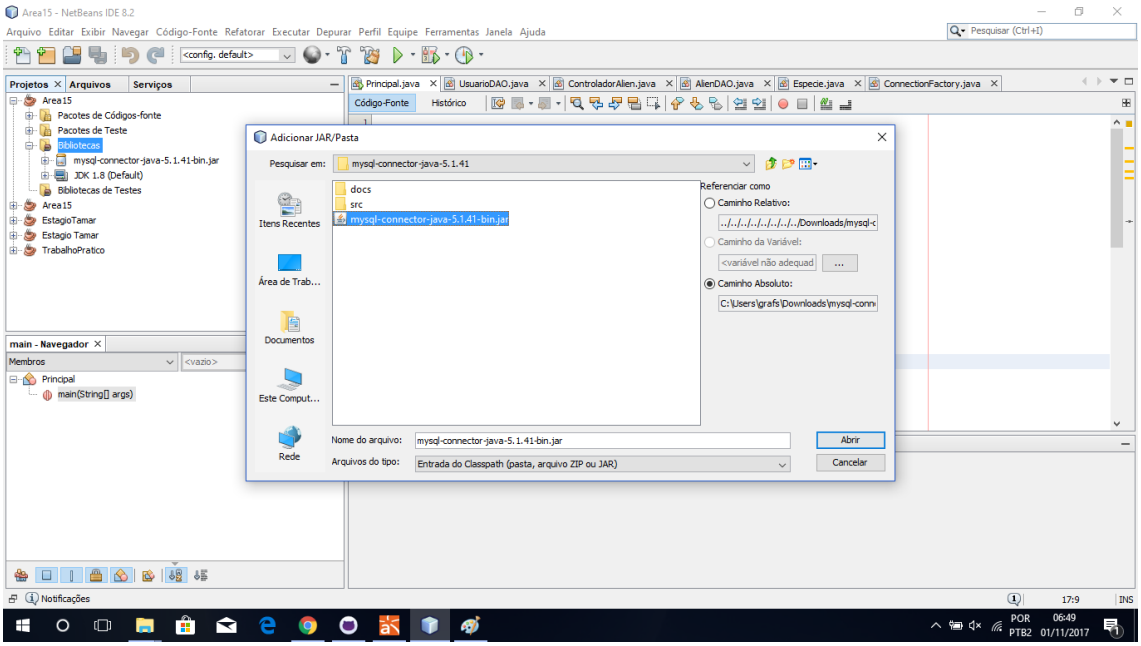

Figura 3: Configurando - Passo 2

```java

package acessoBD;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionFactory {

        public Connection abreConexao(){
        try{
            //Colocar no segundo parâmetro o usuário do BD, normalmente é root
            //Colocar no terceiro parâmetro a senha do banco
            return DriverManager.getConnection("jdbc:mysql://localhost/Area15","" , "");
        } catch( SQLException e){
            throw new RuntimeException();
        }
    }
}

```

Figura 4: ConnectionFactory - Passo 3

A classe *UsuarioDAO* tem um atributo do tipo *Connection* usar o *ConnectionFactory* para fazer a conexão com o banco de dados. Além disto, ela implementa o método *cadastrarUsuario* recebendo um parâmetro *Usuario*. Este método faz um *insert* na tabela de usuarios utilizando um *PreparedStatment* para aumentar a segurança da aplicação evitando ataques do tipo *mysql injection*. Esta classe juntamente com seu método pode-ser visto na figura abaixo.

```java
public class UsuarioDAO {
    Connection connection = new ConnectionFactory().abreConexao();

    public void cadastrarUsuario( Usuario usuario ){
        String sql = "insert into usuario(cpf, nome, login, senha, TipoUsuario) values" +
                "(?,?,?,?,?)";

        try {
            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setString(1, usuario.getCpf());
            stmt.setString(2, usuario.getNome());
            stmt.setString(3, usuario.getLogin());
            stmt.setString(4, usuario.getSenha());
            stmt.setInt(5, usuario.getTipoUsuario().getPatente());

            stmt.execute();
            //stmt.close();
            //connection.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
```

Figura 5: Classe UsuarioDAO

2.1 Conceitos básicos

Para a modelagem do banco de dados criou-se quatro relações, planeta, alien, nave e usuario, conforme a figura a baixo:

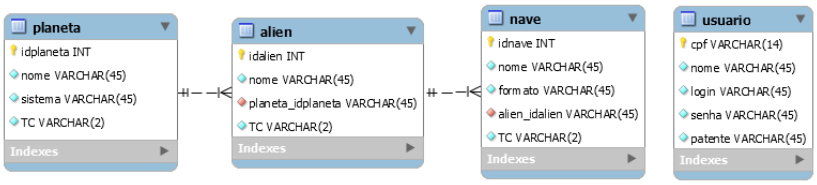

Figura 6: Banco de dados.

Foi pensado que cada usuário possui uma patente do exército, e que cada grau na hierarquia garante acesso a mais dados no banco de dados. Por exemplo, um usuário com patente de soldado somente pode consultar dados de outros usuários com mesma patente. Abaixo pode-se conferir os códigos *sql* para a criação das relações:

```sql
-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`planeta`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`planeta` (
  `idplaneta` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `sistema` VARCHAR(45) NOT NULL,
  `TC` VARCHAR(2) NOT NULL,
  PRIMARY KEY (`idplaneta`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`alien`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`alien` (
  `idalien` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `planeta_idplaneta` VARCHAR(45) NOT NULL,
  `TC` VARCHAR(2) NOT NULL,
  PRIMARY KEY (`idalien`),
  INDEX `fk_alien_planeta1_idx` (`planeta_idplaneta` ASC),
  CONSTRAINT `fk_alien_planeta1`
    FOREIGN KEY (`planeta_idplaneta`)
    REFERENCES `mydb`.`planeta` (`idplaneta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 7: Trecho de código de criação do banco de dados.

```
37  -- -----------------------------------------------------
38  -- Table `mydb`.`nave`
39  -- -----------------------------------------------------
40  CREATE TABLE IF NOT EXISTS `mydb`.`nave` (
41    `idnave` INT NOT NULL AUTO_INCREMENT,
42    `nome` VARCHAR(45) NOT NULL,
43    `formato` VARCHAR(45) NOT NULL,
44    `alien_idalien` VARCHAR(45) NOT NULL,
45    `TC` VARCHAR(2) NOT NULL,
46    PRIMARY KEY (`idnave`),
47    INDEX `fk_nave_alien_idx` (`alien_idalien` ASC),
48    CONSTRAINT `fk_nave_alien`
49      FOREIGN KEY (`alien_idalien`)
50      REFERENCES `mydb`.`alien` (`idalien`)
51      ON DELETE NO ACTION
52      ON UPDATE NO ACTION)
53  ENGINE = InnoDB;
54
55
56  -- -----------------------------------------------------
57  -- Table `mydb`.`usuario`
58  -- -----------------------------------------------------
59  CREATE TABLE IF NOT EXISTS `mydb`.`usuario` (
60    `cpf` VARCHAR(14) NOT NULL,
61    `nome` VARCHAR(45) NOT NULL,
62    `login` VARCHAR(45) NOT NULL,
63    `senha` VARCHAR(45) NOT NULL,
64    `patente` VARCHAR(45) NOT NULL,
65    PRIMARY KEY (`cpf`))
66  ENGINE = InnoDB;
67
```

Figura 8: Trecho de código de criação do banco de dados.

3 Conclusão

Podemos concluir que com a realização deste trabalho prático foi possível compreender melhor os conceitos da disciplina vistos em sala de aula. Fomos capazes de visualizar a importância dos níveis de segurança e como pode ser complicado o desenvolvimento de um sistema seguro e eficiente. O grupo enfrentou dificuldades na vinculação do banco de dados com a aplicação, já que necessitou de um estudo mais aprofundado destes conceitos para então dar início a implementação. Inicialmente foi pensado em um diagrama de classes que comportasse a nossa idéia, mas que na prática se tornou inviável por conter muita informação, então foi pensado em simplificar o modelo para melhor elaboração. A sincronização com o banco de dados também encontramos dificuldade, pois cada comando efetuado possui sua peculiaridade, o qual tomou bastante tempo de ser resolvidas.