Universidade Federal de Viçosa - *Campus* Florestal
Bacharelado em Ciência da Computação
CCF 492 - Tópicos Especiais II
Prof. Daniel Mendes Barbosa

Trabalho Prático 03

Aplicação com Banco de Dados com controle de acesso obrigatório

Samuel Jhonata S. Tavares 2282
Wandella Maia de Oliveira 2292

Florestal - MG
2018

Sumário

1 Introdução

Este trabalho tem o objetivo de implementar uma aplicação que fará a simulação de um banco de dados respeitando as regras de controle de acesso obrigatório e classes de segurança. No controle de acesso obrigatório, utilizou-se as propriedades de segurança simples e estrela. A propriedade simples não autoriza o sujeito ler um objeto que possua classe maior que ele, e a propriedade estrela não autoriza o sujeito escrever um objeto se sua classe for menor que a do objeto.

Esta aplicação permite que o usuário faça controle de usuários (*CREATE USER, SET PASSWORD, SET CLASS e DROP USER*) e consulte o banco *processo*, utilizando os comandos básicos de operação: *SELECT*, *INSERT*, *DELETE* e *DELETE*.

Além disso, é possível visualizar a execução dos comandos no modo *ANALÍTICO*, mostrando os comandos reais executados e as classes de segurança, ou *USUAL*, sendo transparente o controle de acesso ao usuário.

2 Desenvolvimento

2.1 Considerações Gerais

Neste trabalho, utilizou-se a linguagem de programação *Java* para implementar a aplicação e um banco de dados *MySQL*.

Para a criação do banco de dados, utilizou-se o *MySQL Workbench*, sendo criado o modelo ER e gerado o *script* de criação *SQL*, para gerenciamento, o *phpMyAdmin*, facilitando a manipulação dos comandos e testes.

Para gerar os diagramas de classe, utilizou-se a engenharia reversa do *Astha UML*, versão *Student* e, para o fluxograma, *Word* e *Paint*.

2.2 Contexto e metadados

2.2.1 Contexto

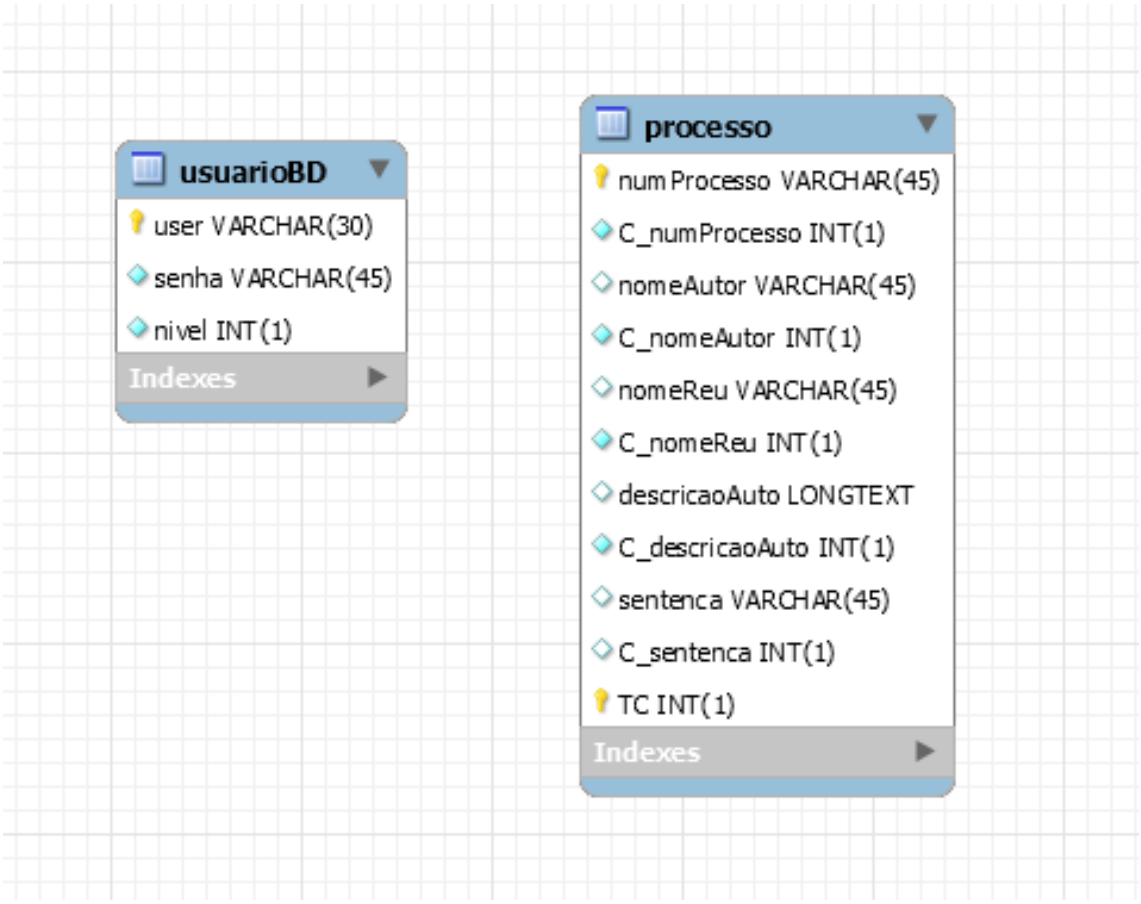
O contexto deste trabalhado é de área jurídica, onde possibilita a inserção e manipulação de processos, processos jurídicos, compostos por número do processo, nomes dos réus e autores, a descrição dos autos e a sentença aplicada.

Este contexto possibilitou que toda a teoria pedida na descrição do trabalho pudesse ser aplicada. Sendo assim, na vida real, muitos tem o direito de ver o número do processo, autores, réus, descrição dos autos e sentença, porém pode acontecer de algumas informações serem bloqueadas a certos níveis de usuários.

2.2.2 Metadados

A Figura 1 abaixo, apresenta o modelo criado, com os metadados que são utilizados neste trabalho. Em suma, foram criadas duas tabelas: *usuárioBD*, para gerenciamento de usuários do sistema, e *processo*, para a operação da aplicação.

Figura 1 – Modelo Entidade Relacional



A tabela *usuarioBD* apresenta os atributos: *user* (usuário), *senha* e *nivel* (nível de segurança do usuário).

A tabela *processo* apresenta os atributos: *numProcesso* (Número do processo), que é a chave aparente, *nomeAutor* (Nome do Autor do processo), *nomeReu* (Nome do Réu no processo), *descricaoAuto* (Descrição do auto do processo) e *sentenca* (Conclusão do juiz)

Além desses atributos, também existem os atributos de controle de acesso para cada um deles: *C_numProcesso*, *C_nomeAutor*, *C_nomeReu*, *C_descricaoAuto* e *C_sentenca*. Além desses, existe o atributo de segurança da tupla, *TC*, que também faz parte da chave.

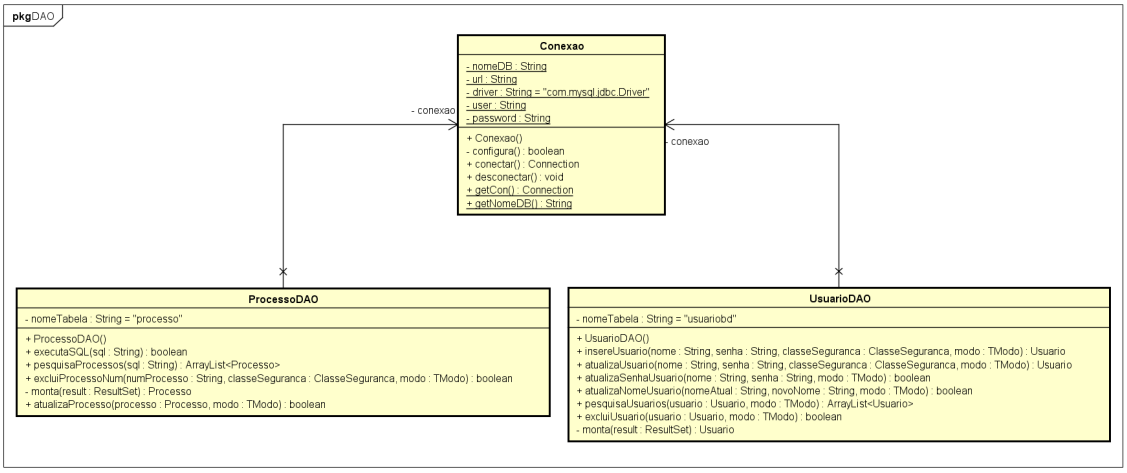
2.3 Modelagem

Na aplicação, além de ela ser Orientada a Objetos, foi utilizado o modelo **MVC**, com uma camada de persistência aplicando o padrão *DAO* (*Data Access Object*) para separar o código de acesso ao banco de dados (DEVMEDIA, 2014).

2.3.1 Persistência

Na Figura 2, apresenta-se o diagrama de classes da persistência dos dados. A mesma é composta pela classe *Conexao*, que é responsável por criar a conexão com o banco de dados, conectar e desconectar, e as classes *ProcessoDAO* e *UsuarioDAO*, responsáveis por manipular no banco as informações sobre Processo e Usuário, que possuem métodos como exclusão, atualização, inserção, entre outros, que auxiliam a comunicação com o banco.

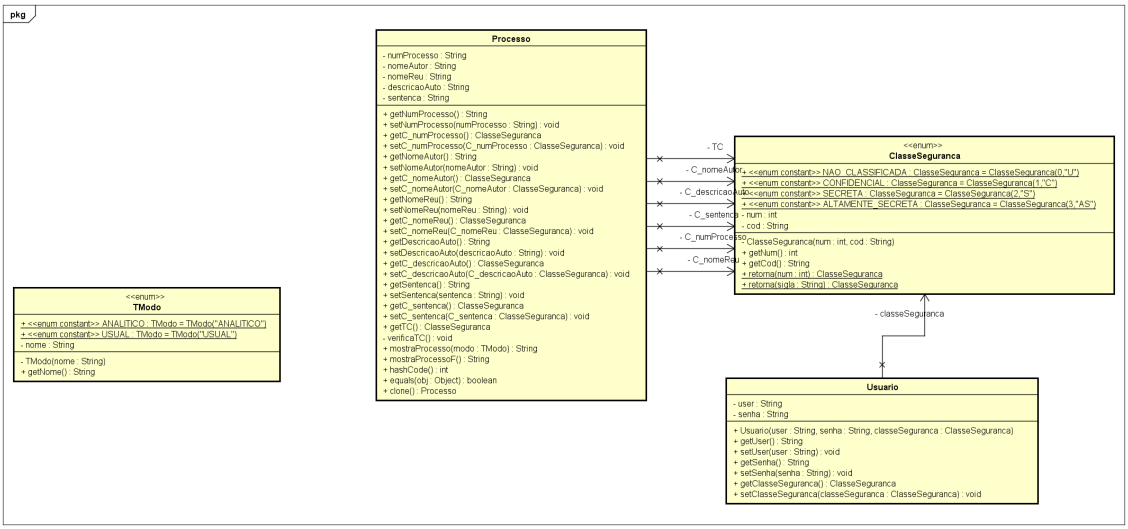
Figura 2 – Diagrama de classes *DAO*



2.3.2 Modelo

Na figura 3, ilustra-se as classes de entidade *Processo* e *Usuario* e duas classes de enumeração, para facilitar a manipulação dos dados: *ClasseSeguranca*, responsável por mapear as classes de segurança dos atributos e do usuário, e *TModo*, para identificar o modo de execução (Analítico ou Usual) .

Figura 3 – Diagrama de Classes Modelo



Vale ressaltar que, as classes de segurança foram numeradas de 0 a 3, onde 0 é Não Classificada (U), 1 é Confidencial(C), 2 é Secreta(S) e 3 é Altamente Secreta(AS). Tal configuração pode ser observada na classe *ClasseSeguranca*. Os modos representados na classe *TModo* exibem a execução da aplicação em dois modos: *Analítico* (exibe as classes de segurança da tupla e os comandos reais executados no banco) e *Usual* (exibe o controle de acesso de forma transparente ao usuário).

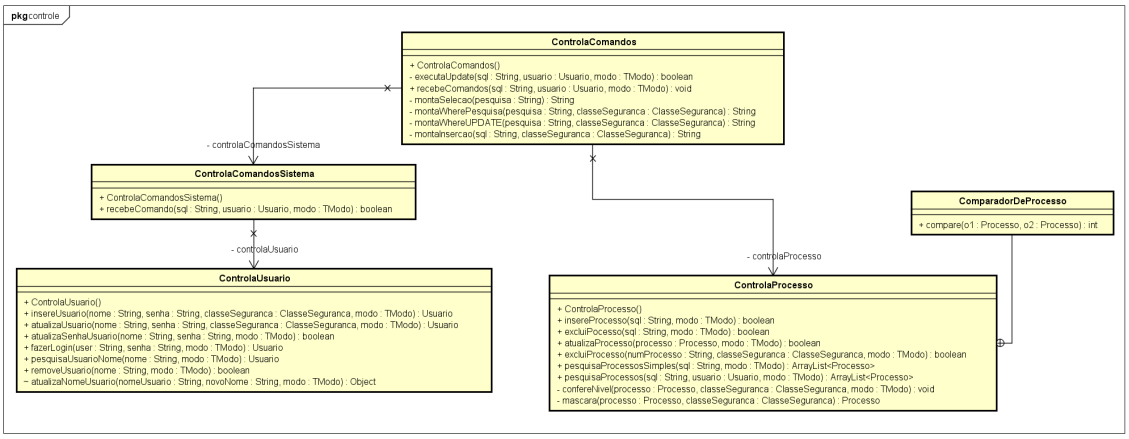
2.3.3 Visão

A visão do sistema é implementada na classe **ModoTexto**, que apresenta a interface para que o usuário possa executar os comandos desejados, de forma bem simples.

2.3.4 Controle

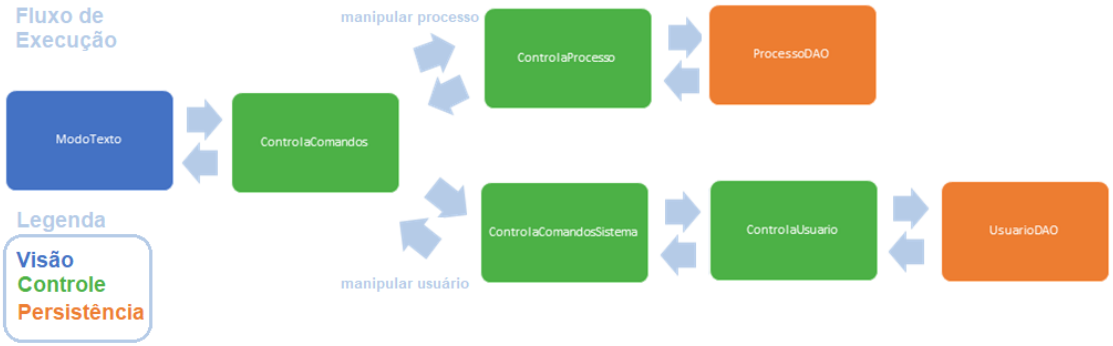
A Figura 4, apresenta as controladoras do programa, onde foi necessário utilizar 5 classes que pudessem, inclusive, facilitar toda manipulação para realizar consultas ao banco, assim como envio e recebimento de respostas ao banco, manipulando os comandos recebidos do usuário para que pudessem ser executados da forma correta no banco real.

Figura 4 – Diagrama de Classes de Controle



Em suma, o fluxo de execução do programa funciona conforme é mostrado na Figura 5, que começa ao usuário utilizar o **Modo texto** (interface do sistema), entrando comandos a serem executados, que por sua vez passa a informação para **ControlaComandos**, responsável por identificar qual comando deve ser executado e fazer seu pré-processamento e, dependendo do comando, seguirá o processo de ida da informação, passando pelo controle, até chegar no banco, retornando as informações ao **ModoTexto**. Quando recebe um comando de manipulação da tabela processo, as informações passam por **ControlaProcesso** e chegam a **ProcessoDAO**, já quando recebe um comando de controle de usuários, passam por **ControlaComandosSistema**, que termina o pré-processamento do comando, passa por **ControlaUsuario**, que, por fim, chega em **UsuarioDAO**:

Figura 5 – Fluxo de Execução



2.4 Implementação

Nas figuras a seguir, serão mostrados algumas partes do código que foi implementado, buscando sempre explicar o funcionamento em alto nível, complementado os modelos apresentados anteriormente.

A figura 6, ilustra a classe de *Conexão*, utilizada para conectar a aplicação ao banco de dados. A implementação foi utilizada de implementações já utilizadas anteriormente, com base em uma implementação retirada da *web* (DEVMEDIA, 2010).

Figura 6 – Conexão com banco de dados

```
public class Conexao {

    private static java.sql.Connection con = null;
    private static String nomeDB;
    private static String url;
    private static String driver = "com.mysql.jdbc.Driver";
    private static String user;
    private static String password;

    public Conexao() {
        conectar();
    }
```

A Figura 7, ilustra a classe *Usuário*. Para acesso, é necessário que o usuário informe o *user* e senha. Para a elaboração dos comandos, foi utilizados comandos já existentes do *MySQL* (BóSON TREINAMENTOS, 2016) com a adaptação da classe do usuário. Quando um usuário deseja criar outro, ele só pode criar com classe de segurança menor ou igual a sua.

Figura 7 – Usuário

```
public class Usuario {

    private String user;
    private String senha;
    private ClasseSeguranca classeSeguranca;
```

A Figura 8, apresenta a classe *enum ClasseSeguranca*, que é utilizada para definir as classes de segurança de algum atributo ou usuário.

Figura 8 – Classe de segurança

```
public enum ClasseSeguranca {
    NAO_CLASSIFICADA(0, "U"),
    CONFIDENCIAL(1, "C"),
    SECRETA(2, "S"),
    ALTAMENTE_SECRETA(3, "AS");

    private int num;
    private String cod;

    private ClasseSeguranca(int num, String cod) {
        this.num = num;
        this.cod = cod;
    }
```

A Figura 9, apresenta a classe *Processo*. Quando um processo é inserido, ele recebe, em cada um dos atributos de classe de segurança, a classe de segurança do usuário que está inserindo. Quando um processo é atualizado, todas as tuplas que existem poli instanciadas devem ser atualizadas e, caso o atributo a ser modificado seja de uma classe menor ou maior que a do usuário e não exista uma tupla com esse nível, é criado uma nova instância. Quando um processo é deletado, todas as instâncias com TC menores ou iguais à classe do usuário são excluídas e, caso haja tuplas com classe maior, os atributos de classe menor são atualizados para um nivel maior que do usuário que fez a exclusão. Assim, as propriedades simples e estrela são sempre respeitadas.

Figura 9 – Processo

```
public class Processo {

    private String numProcesso;
    private ClasseSeguranca C_numProcesso;
    private String nomeAutor;
    private ClasseSeguranca C_nomeAutor;
    private String nomeReu;
    private ClasseSeguranca C_nomeReu;
    private String descricaoAuto;
    private ClasseSeguranca C_descricaoAuto;
    private String sentenca;
    private ClasseSeguranca C_sentenca;
    private ClasseSeguranca TC;
```

2.5 Execução

A seguir, serão apresentam as execuções que foram realizadas depois que o programa foi implementado.

Para aumentar a usabilidade do programa, inseriu-se um comando de ajuda (***help***), com as informações sobre os comandos disponíveis no programa, além da sintaxe aceita para realizar consultas no banco e na manipulação de usuários, bem como sair do sistema ou mudar o modo de execução, mostrado na Figura 10.

Figura 10 – *Help*

```
* * * * Sistema Gerenciador de Banco de Dados Locked - SGBD-L * * * *
Usuario: visitante
Senha:
Bem vindo, visitante

->help

 * * * * * * * * * * * * * * * * * * * * * * * * * * Ajuda * * * * * * * * * * * * * * * * * * * * * *  * * * * * * *
 * - Comandos do sistema:                                                                                          *
 *  help - ajuda                                                                                                   *
 *  exit - sair                                                                                                    *
 *  modo <ANALITICO,USUAL> - mostrar ou nao as classes de seguranca                                                *
 *  reconnect - trocar de usuario                                                                                  *
 *                                                                                                                 *
 * - Para controle de usuario:                                                                                     *
 *  CREATE USER '<nome_usuario>' IDENTIFIED BY '<senha>' CLASS <sigla_classe_seguranca> - insere usuario           *
 *  -Classes disponiveis: NAO CLASSIFICADO (U); CONFIDENCIAL (C); SECRETO (S); ALTAMENTE SECRETO (AS)              *
 *  DROP USER '<nome_usuario>' - exclui usuario                                                                    *
 *  SET PASSWORD FOR '<nome_usuario>' = '<senha>' - modifica senha do usuario                                      *
 *  SET CLASS FOR '<nome_usuario>' = <sigla_classe_seguranca> - modifica classe do usuario                         *
 *  RENAME USER '<nome_usuario>' = '<novo_nome>' - renomear usuario                                                *
 *                                                                                                                 *
 * - Comandos SQL válidos:                                                                                         *
 *  SELECT <*,numProcesso,nomeReu,nomeAutor,descricaoAuto,sentenca> FROM processo                                  *
 *      <WHERE  <atributo> <operador_logico> '<valor>' <, <atributo> <operador_logico> '<valor>'> >                *
 *      <ORDER BY <nomeAtributo> <ordem>>                                                                          *
 *  INSERT INTO processo VALUES('<numProcesso>','<nomeReu>','<nomeAutor>','<descricaoAuto>','<sentenca>')          *
 *  DELETE FROM processo                                                                                           *
 *      <WHERE  <atributo> <operador_logico> '<valor>' <, <atributo> <operador_logico> '<valor>'> >                *
 *  UPDATE processo SET <atributo> <operador_logico> '<valor>' <, <atributo> <operador_logico> '<valor>'>          *
 *      <WHERE  <atributo> <operador_logico> '<valor>' <, <atributo> <operador_logico> '<valor>'> >                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

->
```

2.5.1 Manipulando Usuários

Para exemplificação, foram criados quatro usuários com diferentes privilégios. São eles:

1. Usuário nível 0 (Visitante): Usuário com o nível mais baixo, Não Classificado.
2. Usuário nível 1 (Cidadão): Usuário com o nível Confidencial.
3. Usuário nível 2 (Funcionário): Usuário com o nível Secreto.
4. Usuário nível 3(Juiz): Usuário com o nível Altamente Secreto.

Na Figura 11, primeiramente tentou-se criar o usuário 'Samuel', e como o usuário logado não tinha privilégios suficiente, logo o sistema retornou uma mensagem de erro.

Sendo assim, conectou-se um usuário de privilégio mais alto, no caso, o 'Juiz', que por sua vez possui classe de privilégio 3. Entretanto, o mesmo adicionou o usuário 'Samuel' e em seguida, inseriu senha e classe de privilégio 0, e depois excluiu o mesmo.

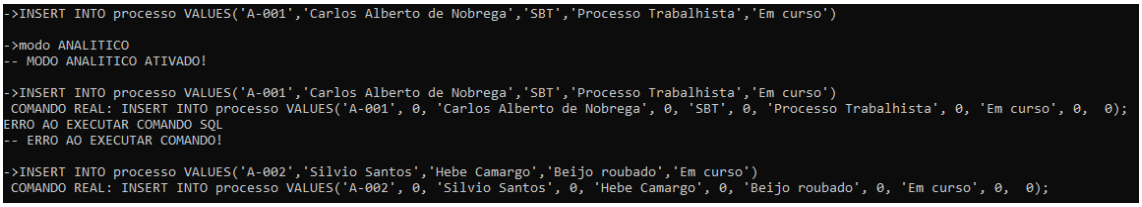
Figura 11 – Manipulação de usuários

```
->CREATE USER 'samuel' IDENTIFIED BY '123' CLASS AS
ERRO AO INSERIR USUARIO - USUARIO ATUAL NAO TEM PRIVILEGIOS SUFICIENTES
-- ERRO AO EXECUTAR COMANDO!

->reconnect
Usuario: juiz
Senha:
 COMANDO REAL: SELECT * FROM usuariobd WHERE user = 'juiz' AND senha = ''
Bem vindo, juiz

->CREATE USER 'samuel' IDENTIFIED BY '123' CLASS AS
 COMANDO REAL: INSERT INTO `tp03`.`usuariobd` (`user`, `senha`, `nivel`) VALUES ('samuel', '123', '3');

->SET PASSWORD FOR 'samuel' = '123456'
 COMANDO REAL: SELECT * FROM usuariobd WHERE user = 'samuel'
 COMANDO REAL: UPDATE usuariobd SET `senha` = '123456' WHERE `usuariobd`.`user` = 'samuel';

->SET CLASS FOR 'samuel' = U
 COMANDO REAL: SELECT * FROM usuariobd WHERE user = 'samuel'
 COMANDO REAL: UPDATE `tp03`.`usuariobd` SET `senha` = '123456', `nivel` = '0' WHERE `usuariobd`.`user` = 'samuel';

->DROP USER 'samuel'
 COMANDO REAL: SELECT * FROM usuariobd WHERE user = 'samuel'
 COMANDO REAL: DELETE FROM usuariobd WHERE `user` = 'samuel'
```

2.5.2 Manipulando Processos

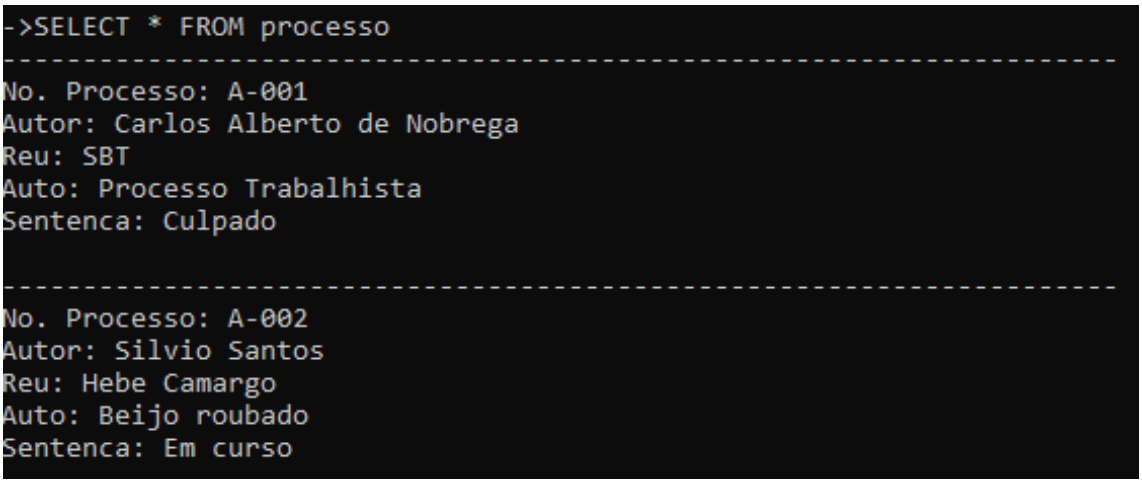
A Figura 12 abaixo, ilustra como realizar uma inserção na aplicação. Note que, a primeira linha apresenta uma inserção de sucesso, e na linha seguinte, tentou-se inserir a mesma tupla e apareceu uma mensagem de erro. Isso ocorreu porque já havia uma tupla presente no banco com a mesma chave. Contudo, na penúltima linha, inseriu-se outra tupla com sucesso.

Figura 12 – Execução de comando *INSERT*



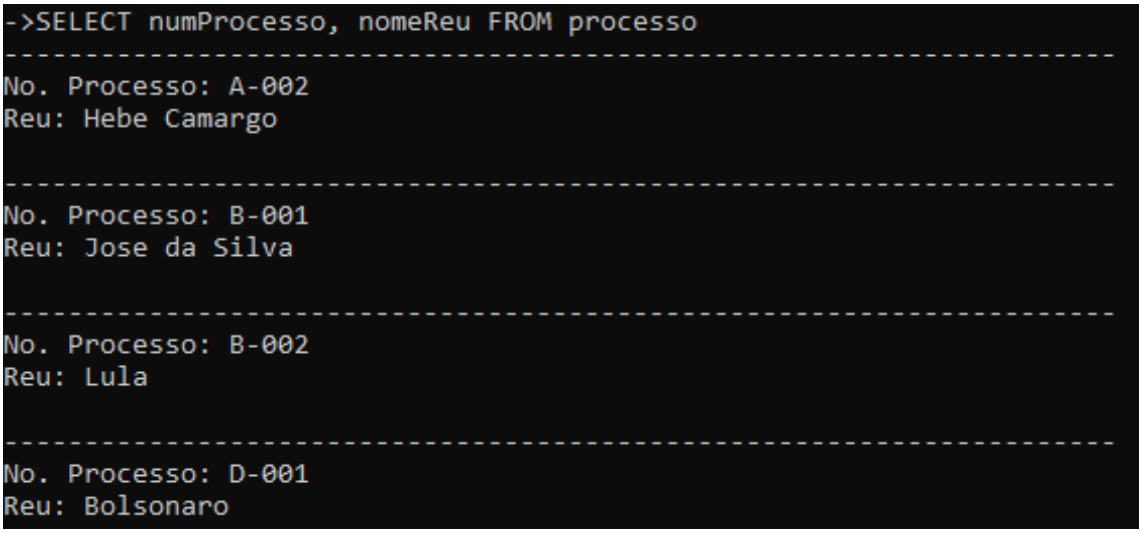
Nas duas figuras abaixo, apresenta-se dois tipos de consulta *SELECT*. Na figura 13, é realizado uma consulta simples, onde todos os processos são recuperados com seus respectivos atributos.

Figura 13 – Execução de comando *SELECT* no modo USUAL



Na figura 14, recupera-se atributos específicos de um processo. Neste exemplo, é recuperado o número do processo e nome do réu.

Figura 14 – Execução de comando *SELECT* de forma restrita, no modo USUAL



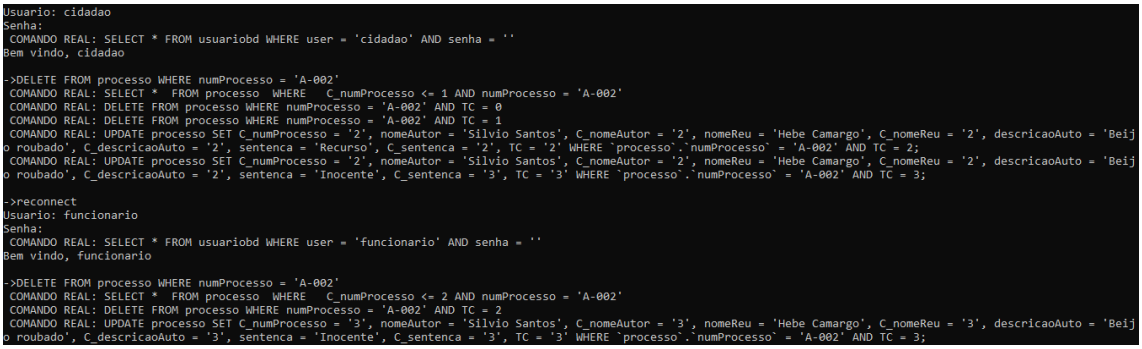
Para ilustrar o exemplo do comando *UPDATE*, primeiramente realizou-se o comando *SELECT* para exibir os processos e seus respectivos atributos, e em seguida realizar o comando de modificação. Note que o exemplo está em modo analítico e que há apenas um processo de número **A-001** e que todas classes são de nível 0(U). Nas linhas seguintes, um usuário de classe nível 1, modifica o atributo de sentença, porém sua classe é superior ao do objeto(processo presente). Sendo assim, o sistema cria uma nova tupla com classe superior ao do objeto, onde o único atributo modifica é o de sentença, que possui agora o mesmo nível que o usuário.

Figura 15 – Execução de comando *UPDATE*

```
->reconnect
Usuario: cidadao
Senha:
 COMANDO REAL: SELECT * FROM usuariobd WHERE user = 'cidadao' AND senha = ''
Bem vindo, cidadao

->SELECT * FROM processo
 COMANDO REAL: SELECT * FROM processo
------------------------------------------------------------------------
No. Processo: A-001(U)
Autor: Carlos Alberto de Nobrega(U)
Reu: SBT(U)
Auto: Processo Trabalhista(U)
Sentenca: Em curso(U)
TC: NAO_CLASSIFICADA
------------------------------------------------------------------------
No. Processo: A-002(U)
Autor: Silvio Santos(U)
Reu: Hebe Camargo(U)
Auto: Beijo roubado(U)
Sentenca: Em curso(U)
TC: NAO_CLASSIFICADA

->UPDATE processo SET sentenca='Culpado' WHERE numProcesso = 'A-001'
 COMANDO REAL: SELECT * FROM processo WHERE  numProcesso = 'A-001' AND TC <= 1 ORDER BY numProcesso ASC, TC DESC
 COMANDO REAL: INSERT INTO processo VALUES ('A-001', 0, 'Carlos Alberto de Nobrega', 0, 'SBT', 0, 'Processo Trabalhista', 0, 'Culpado', 1,1)

->SELECT * FROM processo WHERE numProcesso = 'A-001'
 COMANDO REAL: SELECT * FROM processo  WHERE   C_numProcesso <= 1 AND numProcesso = 'A-001'
------------------------------------------------------------------------
No. Processo: A-001(U)
Autor: Carlos Alberto de Nobrega(U)
Reu: SBT(U)
Auto: Processo Trabalhista(U)
Sentenca: Em curso(U)
TC: NAO_CLASSIFICADA
------------------------------------------------------------------------
No. Processo: A-001(U)
Autor: Carlos Alberto de Nobrega(U)
Reu: SBT(U)
Auto: Processo Trabalhista(U)
Sentenca: Culpado(C)
TC: CONFIDENCIAL
```

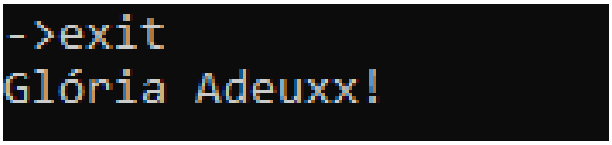
A figura 16, exibe exemplos(está no modo analítico) de como realizar o comando *delete*. No exemplo apresentado, existiam 4 tuplas com diferentes classes de segurança. O comando *Delete* implementado, tem o efeito cascata ao excluir, com base nas propriedades de segurança. Logo, ele exclui do seu nível de segurança para baixo e atualiza as tuplas de nível maior para que usuário com seu nível ou menor não tenham mais acesso (e pareça não existir mais a tupla). No exemplo abaixo, o usuário cidadão faz exclusão de um processo, note que ao executar o comando, são excluídos 2 processos de diferentes classe de segurança. No caso do exemplo, a tupla 1 e 0.

Figura 16 – Execução de comandos *DELETE*



Na figura 17, ilustra o comando *exit*, onde é realizado o encerramento do programa.

Figura 17 – Execução de comando *exit*



3 Conclusão

Com esse trabalho foi possível aprender, na prática, como trabalhar com controle de acesso obrigatório. O mesmo contribuiu para o trabalho da lógica, capacidade de manipulação e tomada de decisões. Além disso, deu-se a oportunidade de conhecer melhor alguns comandos SQL utilizados na manipulação de usuários.

Referências

BóSON TREINAMENTOS. *Curso de MySQL – Gerenciamento de Usuários do sistema - Criar, Consultar, Renomear e Excluir*. 2016. Disponível em: <http://www.bosontreinamentos.com.br/mysql/curso-de-mysql-gerenciamento-de-usuarios-do-sistema-criar-consultar-renomear-e-excluir/>. Acesso em: 15 out. 2018.

DEVMEDIA. *Criando uma conexão Java + MySQL Server*. 2010. Disponível em: <https://www.devmedia.com.br/criando-uma-conexao-java-mysql-server/16753>. Acesso em: 15 out. 2018.

DEVMEDIA. *DAO Pattern: Persistência de Dados utilizando o padrão DAO*. 2014. Disponível em: <https://www.devmedia.com.br/dao-pattern-persistencia-de-dados-utilizando-o-padrao-dao/30999>. Acesso em: 15 out. 2018.