

UNICOL - Aplicação de gestão de inventário

Leonardo Toledo

August 2016

1 Tools and configurations:

1.1 Tools used

- **Database:** MySQL
- **IDE:** IntelliJ IDEA (project with *Maven* to help with the necessary plugins)
- **OS:** Linux

1.2 Configurations:

Database configurations:

The **links** that i followed were:

- **To install:** <http://www.cyberciti.biz/faq/linux-completely-reinstall-mysql-server/>
- **To remove:** <http://www.cyberciti.biz/faq/uninstall-mysql-ubuntu-linux-command/>

1.2.1 To install and create the database and an user:

To **install** the database, i used the command line that are below:

- `sudo apt-get install mysql-client mysql-server mysql-common`

After this, we need to create a database and an user. To do this, it is necessary entry as admin (root). Normally, it is just run the command line below:

- `mysql -u root -p` (ENTER and the password is null if you do not put any password during the installation, just ENTER again)

I created with these names:

- **Nome:** UNICOL
- **User:** user1
- **Password:** password1

To create the database i used the command line below:

- `CREATE DATABASE UNICOL;`

To create a new user and give to him all permissions, i used the command lines below:

- `CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';`
- `GRANT ALL PRIVILEGES ON UNICOL . * TO 'user1'@'localhost';`
- `FLUSH PRIVILEGES;`

To **start** the database directly i use this command line:

- `mysql -u user1 -p UNICOL` (ENTER and after insert the password)

1.2.2 To create all tables:

To create all tables that are explained on the next section, on the entity relationship diagram, i used the below SQL:

Table Status:

```
drop table Status;
```

```
create table Status ( status_id int not null auto_increment, name varchar(45) not null, primary key (status_id) );
```

Table Location:

```
drop table Location;
```

```
create table Location ( location_id int not null auto_increment, name varchar(45) not null, department varchar(45) not null, room varchar(45) not null, actually_used bool not null primary key (location_id) );
```

Table Date:

```
drop table Date;
```

```
create table Date ( date_id int not null auto_increment, year int not null, month int not null, day int not null, primary key (date_id) );
```

Table Category:

```
drop table Category;
```

```
create table Category ( category_id int not null auto_increment, name varchar(45) not null, actually_used bool not null primary key (category_id) );
```

Table Family:

```
drop table Family;
```

```
create table Family ( family_id int not null auto_increment, name varchar(100) not null, actually_used bool not null primary key (family_id) );
```

Table Equipments:

```
drop table Equipments;
```

```
create table Equipments ( equipments_id int not null auto_increment, id_location int references Location (location_id) on delete cascade on update cascade, id_family int references Family (family_id) on delete cascade on update cascade, id_category int references Category (category_id) on delete cascade on update cascade, id_date int references Date (date_id) on delete cascade on update cascade, id_status int references Status(status_id) on delete cascade on update cascade, code varchar(45) not null, observations varchar(255) not null, primary key (equipments_id) );
```

Table Historic:

```
drop table Historic;
```

```
create table Historic ( historic_id int not null auto_increment, id_location int references Location (location_id) on delete cascade on update cascade, id_family int references Family (family_id) on delete cascade on update cascade, id_category int references Category (category_id) on delete cascade on update cascade, id_date int references Date (date_id) on delete cascade on update cascade, id_status int references Status(status_id) on delete cascade on update cascade, code varchar(45) not null, observations varchar(255) not null, primary key (historic_id) );
```

2 Entity Relationship Diagram (Database):

I drew the database in to order to achieve that you can create or delete new locations, families, categories and dates. The status it will be insert a priori. The table *Equipments* will contain all equipments and the table *Historic* will contain all records saved about each equipment.

In the table *Location* the parameter *name* is to insert the location's name, for example "Zona industrial Praia da Vitória", the parameter *department* is to the department's name, for example "Informática" and the parameter *room* is to insert the room's number (or name).

The parameter *actually_used* is a boolean and it is used to verify if a specific location, or family, or category still exist, because the user can delete a specific location, or family, or category, but we can to save an historic so it is necessary save this records's ID. This field is 1 is created and 0 when it is deleted.

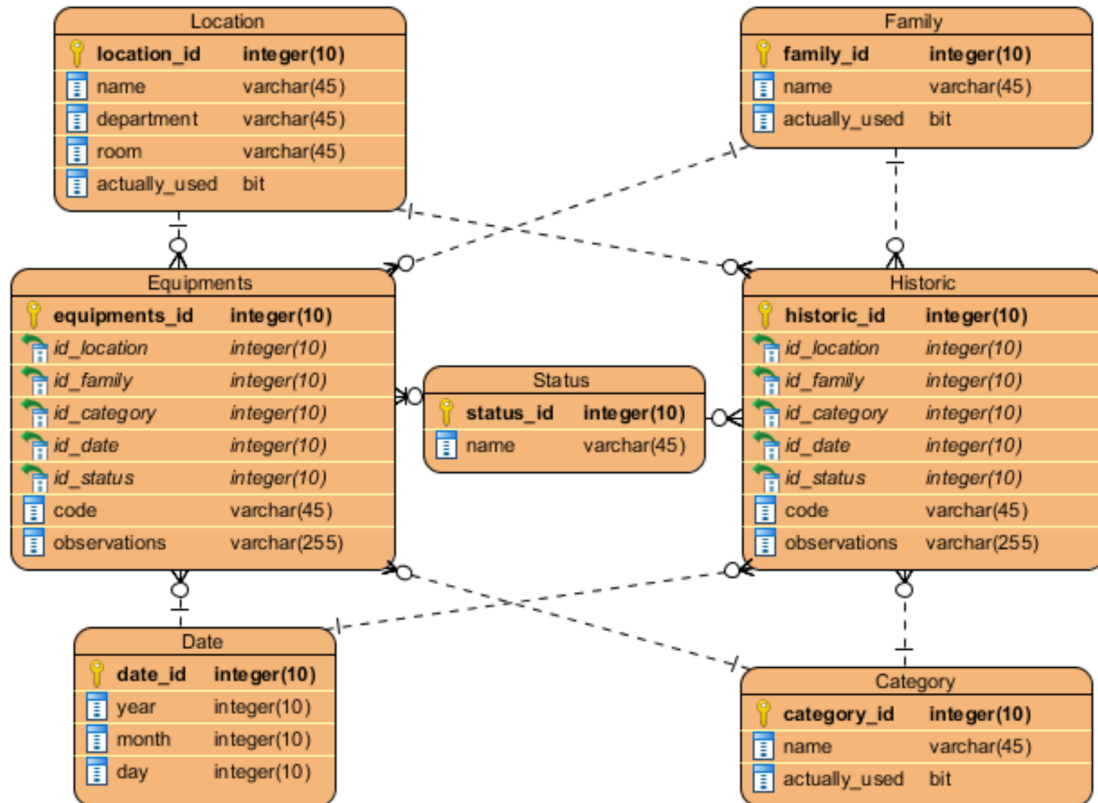


Figure 1: Entity Relations Diagram - Database

3 Code and methods:

The project is organised by 2 packages called *Main_Graphics* and *Model*. Inside the package *Main_Graphics* exist 2 java files, one is the *Main.java* where contain the main method and it is here that is called the UI and the initializaton of the database and the other file is the *Controller.java* where it is intended to handle user interactions with the UI. Inside the package *Model* exist more 2 java files, one is the *DatabaseConnection* that does the connection to the database and the other file is the *ModelFunctions* where i think that it has already created all necessary methods to the model to exchange the information between the user and the database, but the mockups that i drew are in the next section and you can verify. The methods that are created in the packge *Model* are all well commented in order to understand quickly the objective of each one of them.

To do the **user interface** i thought do this using the *JavaFX*, but it can be changed quickly and at the same time it is the part of the project that need to be finished. I opted to use the JavaFX because i was reading some papers and they tell that JavaFX is better than Swing(Java) and i think that it is the better option to this project.

4 Mockups:

The next images are the mockups that i drew to the user interface as i thought and i did almost every necessary methods to interact with the databse to do the selects, inserts, updates and deletes.

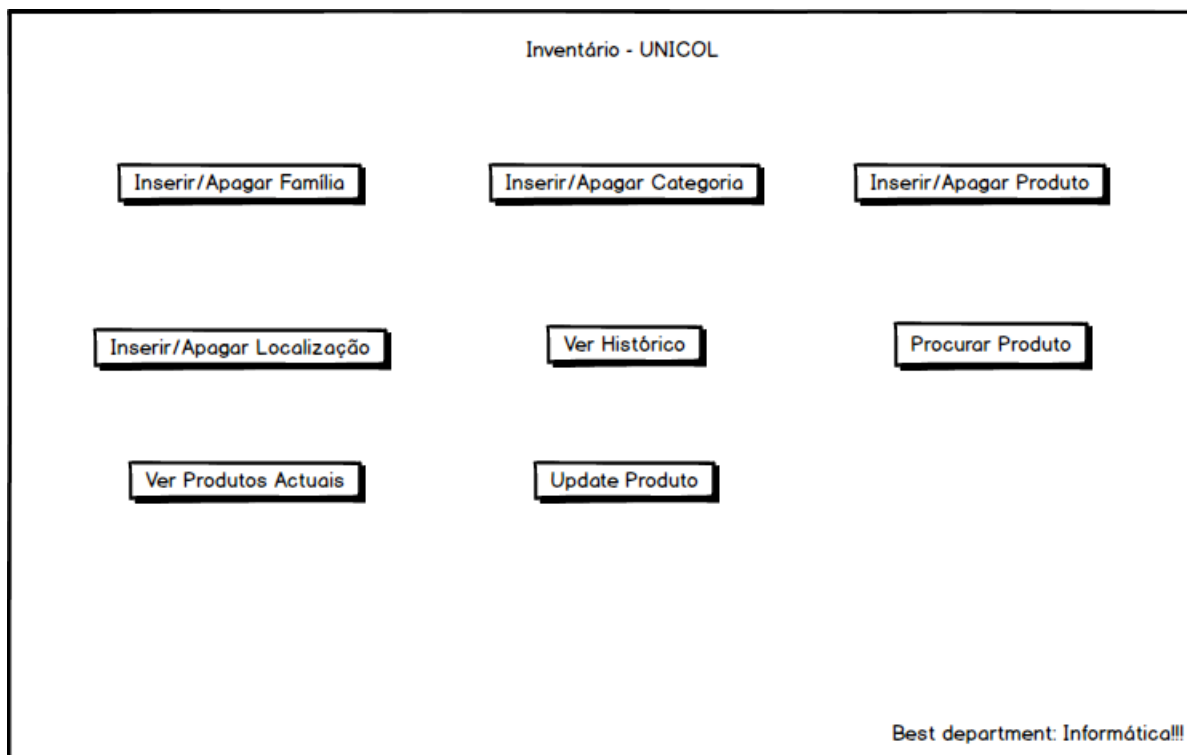


Figure 2: Main panel

Inventário - UNICOL

Inserir uma categoria:

Nome:

Remover uma categoria:

Informações de erro ou sucesso:

Exemplo1: Erro: Categoria ja existente! Verifique e tente novamente!

Exemplo2: Sucesso: Categoria inserida com sucesso!

Exemplo3: Sucesso: Categoria removida com sucesso!

Figure 3: Insert or delete a category

Inventário - UNICOL

Inserir uma família:

Nome:

Remover uma família:

Informações de erro ou sucesso:

Exemplo1: Erro: Família ja existente! Verifique e tente novamente!

Exemplo2: Sucesso: Família inserida com sucesso!

Exemplo3: Sucesso: Família removida com sucesso!

Figure 4: Insert or delete a family

Inventário - UNICOL

Inserir nova localização (necessita preencher todos os campos): Remover uma localização (deve preencher todo):

Nome: Nome

Departamento: Departamento

Sala: Sala

Informações de erro ou sucesso:

Exemplo1: Erro: Localização já existente! Verifique e tente novament

Exemplo2: Sucesso: Localização inserida com sucesso!

Exemplo3: Sucesso: Localização removida com sucesso!

Figure 5: Insert or delete a location

Inventário - UNICOL

Inserir ou remover um produto (necessita preencher todos os campos):

Família Categoria Estado Código:

Dia Mês Ano Observações:

Observações

Localização Departamento Sala

Informações de erro ou sucesso:

Exemplo1: Erro: Produto já existente! Verifique o codigo e tente novamente!

Exemplo2: Sucesso: Produto inserido com sucesso!

Exemplo3: Sucesso: Produto removido com sucesso!

Figure 6: Insert or remove a product

Inventário - UNICOL

Procurar Produto (preencher só o campo 'Código' ou pelo menos os 2 campos 'Família' e 'Categoria'):

Código:

Informação do produto selecionado:

Código	Família	Categoria	Estado	Dia	Mês	Ano	Localização	Departamento	Sala	Observações
PC001	PC	Laptop	ON	16	08	2016	Central	Informatica	escritorio	Sem office

Figure 7: Search a product

Inventário - UNICOL

Update da informação do produto (pode mudar só os campos que desejar, consoante as regras descritas):

Código:

Necessita preencher os 3 campos (dia, mês, ano) para mudar:

Necessita preencher os 3 campos (localização, departamento, sala) para mudar:

Observações:

Informações de erro ou sucesso:

Exemplo1: Erro: Update não realizado! Tente novamente!

Exemplo2: Sucesso: Update realizado!

Figure 8: Update the informations about a specific product

Inventário - UNICOL

Ver o histórico por família, categoria ou estado:

Ver o histórico de um produto específico:

Código:

Ver o histórico por localização, departamento ou sala:

Ver todo o histórico no geral:

Informação do produto selecionado:

Código	Família	Categoria	Estado	Dia	Mês	Ano	Localização	Departamento	Sala	Observações
PC001	PC	Laptop	ON	16	08	2016	Central	Informatica	I escritorio	Sem office

Figure 9: Show historic

Inventário - UNICOL

Ver produtos actuais (Pode procurar só por 'Família' ou 'Categoria' ou 'Estado' ou pelo sítio onde está o produto):

Para procurar só pelo sítio, tem que preencher os 3 campos ('Localização', 'Departamento' e 'Sala'):

Ver todos os produtos actuais no geral:

Informação do produto selecionado:

Código	Família	Categoria	Estado	Dia	Mês	Ano	Localização	Departamento	Sala	Observações
PC001	PC	Laptop	ON	16	08	2016	Central	Informatica	I escritorio	Sem office

Figure 10: Show some products