

# Relatório de Base de Dados - Meta II

João Pedro Castilho 2017263424 e Gonçalo Arsénio 2017246034

Dezembro de 2019

## Contents

<b>1</b>	<b>Introdução e Objectivos</b>	<b>3</b>
<b>2</b>	<b>Organização geral da aplicação</b>	<b>3</b>
2.1	main.py . . . . .	3
2.2	menu.py . . . . .	3
2.3	funcoes.py . . . . .	3
<b>3</b>	<b>Distribuição final das tarefas</b>	<b>3</b>
<b>4</b>	<b>Diagrama Entidade-Relacionamento</b>	<b>4</b>
4.1	Alterações . . . . .	6
<b>5</b>	<b>Diagrama Físico</b>	<b>6</b>
<b>6</b>	<b>Tempo gasto na disciplina</b>	<b>7</b>
6.1	João Pedro Castilho . . . . .	7
6.2	Gonçalo Arsénio . . . . .	7
<b>7</b>	<b>Triggers e Sequências</b>	<b>7</b>
<b>8</b>	<b>Extras</b>	<b>8</b>
<b>9</b>	<b>Screenshots</b>	<b>9</b>

# 1 Introdução e Objectivos

A empresa *Vinyl Records Lda.* pretende criar um sistema que lhe permita gerir o seu negocio online. O objectivo principal é projectar uma aplicação de base de dados que permita aos clientes, por exemplo, comprar e pesquisar discos, e que os administradores possam desempenhar tarefas de gestão da loja, por exemplo, adicionar um álbum ou visualizar estatísticas da loja.

## 2 Organização geral da aplicação

Decidimos dividir a aplicação em 3 ficheiros, `main.py`, `funcoes.py` e `menu.py`.

### 2.1 `main.py`

Este ficheiro funciona apenas como um género de executável. Apenas damos import do ficheiro `menu.py` e chamamos a função `menu.menu_inicial` que leva o utilizador para o menu de login.

### 2.2 `menu.py`

Neste ficheiro, temos toda a parte que o cliente ao usar a aplicação vê no seu terminal. Cada funcionalidade da aplicação tem um menu e, é a partir das funções presentes neste ficheiro que o cliente ou o admin escolhe o que é que quer fazer, por isso, foi uma das nossas prioridades fazer com que o "design" dos menus fosse de fácil compreensão e *user-friendly*.

As principais funções deste ficheiro recebem todas um parâmetro `user` para que não percamos o email de quem está a usar a aplicação à medida que vai avançando de menu para menu.

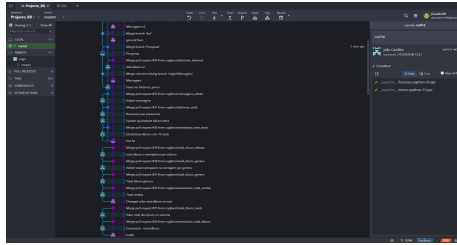
### 2.3 `funcoes.py`

Neste ficheiro, temos todas as funções que servem para aceder ao servidor da base de dados para obter, inserir ou actualizar dados.

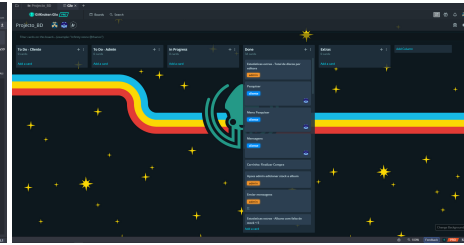
## 3 Distribuição final das tarefas

Dividimos as tarefas em dois grupos: funcionalidades do cliente e funcionalidades do admin. Tendo a parte do cliente ficado com o Gonçalo Arsénio e a parte do admin com o João Pedro Castilho.

Para facilitar o trabalho em equipa usamos o *git* para version control, em conjunto com o *GitHub* para os branch remotos e o software *GitKraken* (Fig. 1a) para nos ajudar com os comandos *git*, visto que nunca tínhamos trabalhado com *git*. Dentro do software *GitKraken* usamos também as *GloBoards* Fig. 1b para uma divisão mais fácil do trabalho e para conseguirmos acompanhar o trabalho um do outro mais facilmente.

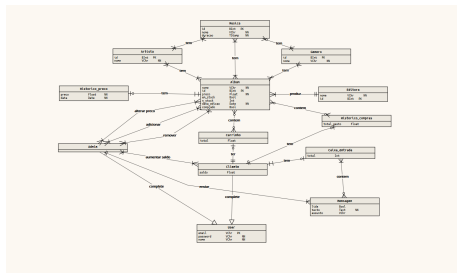


(a) Git Kraken

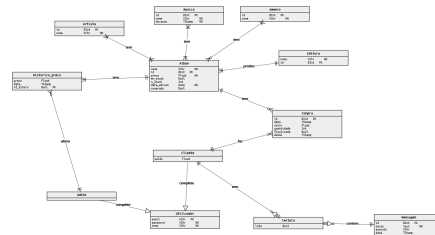


(b) Glo Boards

## 4 Diagrama Entidade-Relacionamento



(a) Diagrama ER meta 1



(b) Diagrama ER final

### User (email, password, nome)

User divide-se em Cliente e Admin. Assim, no diagrama ER, Utilizador é uma super-entidade e tem uma relação de herança mutuamente exclusiva com Cliente e Admin (ou é uma ou é outra, mas não os dois ao mesmo tempo).

O email é *primary key*, pois o email é único.

### Cliente (saldo)

A entidade *Cliente* tem duas relações: uma relação (0:n) com a entidade *Compra* pois o cliente pode não ter nenhuma compra ou então varias compras; uma relação (0:n) com a entidade fraca *leitura* pois o cliente é capaz de fazer varias leituras das mensagens.

Por *default* o valor inicial do saldo vai ser 20.

### Admin

A entidade *admin* não precisa de nenhum parâmetro pois herda todos da super entidade *Utilizador*. Tem uma relação com a entidade *Historico\_preco* para que na entidade *Historico\_preco* fique registado o email do admin que fez a alteração de preço.

### **Album (id, nome, preco, em\_stock, n\_stock,data\_edicao,comprado)**

A *primary key* desta entidade é o id, pois todos os outros campos podem se repetir.

A entidade *album* tem varias relações: relação "tem" (0:n) com a entidade *compra*, para que a entidade *compra* tenha a primary key da entidade *album*; relação "produz" (1:1) com a entidade *Editora*, tem obrigatoriedade porque para um álbum existir tem de ter uma editora, um álbum só pode ter uma editora; tem 3 relações iguais com as entidades *artista*, *genero* e *musica*, um álbum pode ter um ou mais géneros musicais, uma ou mais musicas e um ou mais artistas; a relação (1:n) com a entidade *Historico\_preco*, com esta relação fazemos com que seja necessário inserir o preço inicial do álbum no histórico preço.

### **Editora(nome,id)**

Entidade criada pois um álbum tem uma editora. Tem apenas uma relação "produz" com Album(1:n), pois uma editora pode ter produzido um ou mais albuns.

### **Genero(id, nome)**

A entidade para o género musical tem uma relação: relação "tem" (0:n) com Album, um género pode não ter de zero a vários albuns.

### **Musica(id, nome)**

Esta entidade foi criada, pois um álbum contém músicas, a *primary key* é o id, pois pode haver músicas com o mesmo nome. Esta entidade tem uma relação (1:n) com a entidade *Album*, pois uma música tem de pertencer a pelo menos um álbum.

### **Historico\_preco(preco, data, id\_altera)**

A primary key desta entidade é um id (id\_altera). Esta entidade foi criada para registar todas as alterações de preço que um álbum sofre. Tem apenas uma relação (1:1) com Album, ou seja, o histórico dos preços é único de cada álbum, não há históricos "partilhados".

### **Compra(id, data, valor,quantidade,finalizada,data2)**

Esta entidade foi criada para registar as compras dos clientes. Tem como função ser um carrinho, constituído pelas compras não finalizadas do cliente, e função de histórico de compras constituído pelas compras finalizadas do cliente. Esta entidade tem duas relações: uma relação (1:1) com a entidade Album pois uma compra é apenas constituída por um álbum; uma relação (1:1) com a entidade Cliente pois uma compra pertence apenas a um cliente.

### leitura(lida)

Esta entidade fraca tem como função registar se o cliente já leu uma dada mensagem. Tem duas relações: uma relação (1:1) com o cliente pois uma "leitura" apenas corresponde a um cliente; uma relação (1:1) com a entidade mensagem pois a uma leitura apenas corresponde a uma mensagem.

### Mensagem(id,texto,assunto,data)

Esta entidade tem como função registar as mensagens enviadas pelos administradores. Tem apenas uma relação (0:n) com entidade *leitura* pois a mesma mensagem vai aparecer varias vezes na entidade *leitura*

## 4.1 Alterações

As principais alterações feitas foram na redução de relações desnecessárias, nas parte das compras/carrinho e nas mensagens.

Apagamos as entidades *Carrinho* e *Histórico\_compras* e criamos uma entidade *Compra* em que o histórico de compras vai ser constituído por compras que se encontram finalizadas, ou seja, quando o campo *finalizada* estiver a *True*.

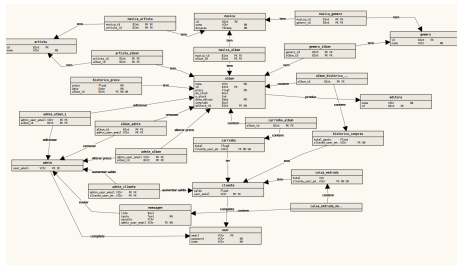
Substituímos a entidade *Caixa\_entrada* pela entidade fraca *leitura*, em que a leitura é de um cliente mas um cliente pode ler zero ou n mensagens, que serve para fazer a verificação se o cliente já leu ou não a mensagem, através do campo *lida* que vai ficar a *True* quando o cliente ler a mensagem.

Adicionamos à entidade *Mensagem* um campo *id* como *Primary Key*, pois um dos nosso erros era ter várias tabelas sem *Primary Key* e um campo *data*, que vai ser a data em que o cliente recebeu a mensagem, e tem a mesma relação que a entidade *Cliente* tem com *leitura*.

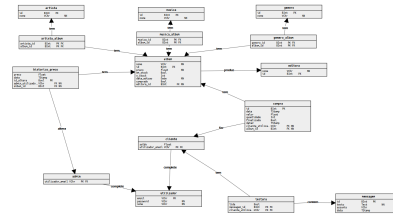
Na entidade *Historico\_preco* adicionamos uma *Primary Key* e uma relação *altera* com a entidade *Admin*, em que um administrador pode alterar o preço de zero ou n albuns, e alteramos a sua relação com a entidade *Album*.

## 5 Diagrama Físico

Como se pode verificar depois das alterações feitas ao diagrama ER o diagrama físico ficou muito mais simples e conciso.



(a) Diagrama Físico meta 1



(b) Diagrama Físico final

## 6 Tempo gasto na disciplina

### 6.1 João Pedro Castilho

O tempo gasto por semana na disciplina foi mais ou menos 5 horas (T+TL). Extra aula para a realização desta aplicação foi mais ou menos uma semana.

### 6.2 Gonçalo Arsénio

O tempo gasto por semana na disciplina foi mais ou menos 3 horas (T+TL). Extra aula para a realização desta aplicação foi mais ou menos uma semana.

## 7 Triggers e Sequências

Criamos um trigger para, quando um cliente novo se regista, para além de inserir os seus dados na tabela utilizador, inserimos também na tabela cliente. Para isso criamos a função trigger *cliente\_insert()* e associamos a função ao trigger *insere\_cliente* e associamos o trigger a tabela utilizador.

```
CREATE OR REPLACE FUNCTION cliente_insert()
  RETURNS trigger AS
  $$
  BEGIN
    INSERT INTO cliente(saldo, utilizador_email)
      VALUES(DEFAULT,NEW.email);

    RETURN NEW;
  END;
  $$
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER insere_cliente
  AFTER INSERT
```

```

ON utilizador
FOR EACH ROW
EXECUTE PROCEDURE cliente_insert ();

```

Criamos também varias sequências (Fig. 4) todas com o mesmo propósito: incrementar o campo *id* nas tabelas que têm um campo *id*, por exemplo, compra, género, artista, etc... Para cada campo *id* criamos uma sequência do tipo:

```

CREATE SEQUENCE album_id_sequence
INCREMENT 1
START 1;

```

### ▼ 1.3 Sequences (8)

```

1.3 album_id_sequence
1.3 altera_id_sequence
1.3 artista_id_sequence
1.3 compra_id_sequence
1.3 editora_id_sequence
1.3 genero_id_sequence
1.3 mensagem_id_sequence
1.3 musica_id_sequence

```

Figure 4: Sequências

## 8 Extras

No ficheiro *menu.py* damos import a duas bibliotecas *getpass* e *datetime*. Decidimos usar a biblioteca *getpass* porque achámos que quando o utilizador escrevia a palavra passe não devia aparecer os caracteres que o utilizador digitava. Para isso usamos:

```
passwd_input = getpass.getpass('Password:')
```

A utilização da biblioteca *datetime* surgiu da necessidade de precisarmos de validar as datas que o utilizador introduzia, por exemplo na data de edição. Para isso criamos a função *validate(date\_text)*:

```

def validate(date_text):
    try:

```



```

        if date_text != datetime.strptime(date_text,"%Y-%m-%d").strftime('%Y-%m-%d'):
            raise ValueError
        return True
    except ValueError:
        return False

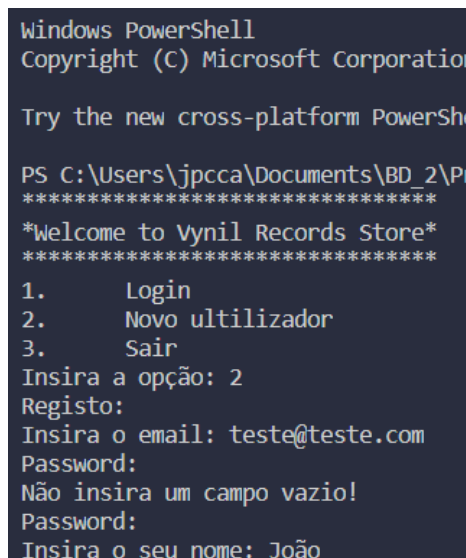
```

Usamos esta função para depois validarmos a data que o utilizador insere.

```
if validate(album_data.edicao) == False:
```

## 9 Screenshots

Algumas screenshots da aplicação:



```

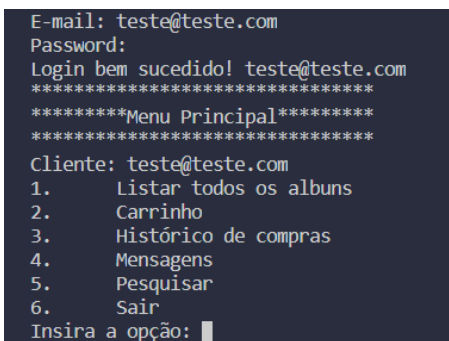
Windows PowerShell
Copyright (C) Microsoft Corporation

Try the new cross-platform PowerShell
https://aka.ms/pscore6

PS C:\Users\jpcca\Documents\BD_2\Projeto>
*****
*Welcome to Vinyl Records Store*
*****
1.      Login
2.      Novo utilizador
3.      Sair
Insira a opção: 2
Registo:
Insira o email: teste@teste.com
Password:
Não insira um campo vazio!
Password:
Insira o seu nome: João

```

(a) Registo de um cliente



```

E-mail: teste@teste.com
Password:
Login bem sucedido! teste@teste.com
*****
*****Menu Principal*****
*****
Cliente: teste@teste.com
1.      Listar todos os albuns
2.      Carrinho
3.      Histórico de compras
4.      Mensagens
5.      Pesquisar
6.      Sair
Insira a opção: 

```

(b) Menu cliente

```

ID:      25      Nome:  kfkkgkfdkg
ID:      23      Nome:  L0000L
ID:      24      Nome:  L00000L

1.      Ver detalhes de um album
2.      Sair
Insira a opção: █

```

(a) Listagem de álbuns

```

ID:      25      Nome:  kfkkgkfdkg
ID:      23      Nome:  L0000L
ID:      24      Nome:  L00000L

1.      Ver detalhes de um album
2.      Sair
Insira a opção: 1
Insira o ID do album: 25
ID: 25
Nome: kfkkgkfdkg
Artista 1 : Chico
Artista 2 : Drake
Musica 1 : musica45
Gênero 1 : Pop
Editora: FR
Data de Edição: 1111-11-11
Preço: 14.0 €
Exemplares em Stock: 19
Enter para continuar█

```

(b) Detalhes de um álbum

```

ID da compra: 15
ID do album: 25
Nome do album: kfkkgkfdkg
Quantidade: 1
Valor: 14.0
Data em que foi adicionado ao carrinho: 2019-12-10 16:46:37.295858
Data em que foi comprado: 2019-12-10 16:48:33.321874
-----//-----
ID da compra: 16
ID do album: 23
Nome do album: L0000L
Quantidade: 1
Valor: 10.0
Data em que foi adicionado ao carrinho: 2019-12-10 16:50:09.173406
Data em que foi comprado: 2019-12-10 16:50:24.813763
-----//-----
ID da compra: 17
ID do album: 24
Nome do album: L00000L
Quantidade: 1
Valor: 10.0
Data em que foi adicionado ao carrinho: 2019-12-10 16:50:27.955666
Data em que foi comprado: 2019-12-10 16:50:34.435018
-----//-----
Gastou 20.0 € no genero: RaP
Gastou 10.0 € no genero: Rock
Gastou 14.0 € no genero: Pop
Enter para continuar

```

(a) Histórico de compras

```

*****Menu Principal*****
*****
Cliente: teste@teste.com
1.  Listar todos os albuns
2.  Carrinho
3.  Histórico de compras
4.  Mensagens
5.  Pesquisar
6.  Sair
Insira a opção: 4

*****Caixa de Entrada*****
*****
ID: 13 Assunto: asddfg Data: 2019-12-10 17:08:02.006136
-----//-----
1.  Ler mensagem
2.  Lidas
3.  Menu inicial
Insira a opção: 1
Insira o id da mensagem que quer ler: 13
Assunto: asddfg

mensagem de teste
Enter para continuar█

```

(b) Mensagens

```

Admin: admin@admin.com bem vindo!
Admin: admin@admin.com
1. Adicionar álbum.
2. Visualizar álbuns em stock e quantidades.
3. Enviar mensagens.
4. Aumentar saldo de um cliente.
5. Estatística da loja.
6. Sair.
Insira uma opção: 1
Nome: Frkl
Preço 12
Qual vai ser o stock inicial: 10
Insira no formato yyyy-mm-dd
Data de edição: 1111-11-11
Editora: QQQ
Quantos artistas tem o album: 1
Artista: FRR
Quantos generos tem o album: 1
Genero: KPop
Quantas musicas tem o album: 1
Musica: plpl
Admin: admin@admin.com
1. Adicionar álbum.
2. Visualizar álbuns em stock e quantidades.
3. Enviar mensagens.
4. Aumentar saldo de um cliente.
5. Estatística da loja.
6. Sair.
Insira uma opção: █

```

(a) Adicionar um álbum

```

*****
*****Estatisticas*****
*****
1. Total clientes
2. Total de discos
3. Valor total dos discos em stock
4. Valor total das vendas
5. Total de discos por genero
6. Total de discos por editora
7. Albuns com falta de stock(<5)
8. Sair
Insira a opção: 5
pop : 0 Exemplares: 0
Brasileirada : 0 Exemplares: 0
RaP : 2 Exemplares: 17
Rock : 1 Exemplares: 12
Pop : 1 Exemplares: 16
Fado : 1 Exemplares: 10
KPop : 1 Exemplares: 10
*****
*****Estatisticas*****
*****
1. Total clientes
2. Total de discos
3. Valor total dos discos em stock
4. Valor total das vendas
5. Total de discos por genero
6. Total de discos por editora
7. Albuns com falta de stock(<5)
8. Sair
Insira a opção: █

```

(b) Numero de exemplares por género musical