



FACULDADE DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE DE COIMBRA

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Base de Dados

2º Relatório do Projeto

NETFLOX

Bernardo Alexandre dos Santos Costa Leite, 2017268958

André Alexandre Ribeiro Araújo, 2018276072

– 12 de dezembro de 2020 –

# 1 Introdução

A empresa *NETFLOX* pretende um sistema que lhe permite gerir o seu negócio online, que consiste essencialmente em alugar artigos de vários tipos, nomeadamente, series, filmes e documentários para visualização online e gerir o seu inventário de artigos disponíveis para alugar. Assim, é necessário construir uma aplicação de bases de dados que permita que clientes possam, por exemplo, pesquisar e alugar artigos disponíveis, e que permita que os administradores, que são manualmente adicionados à base de dados, possam desempenhar tarefas típicas de gestão do sistema (e.g., adicionar um novo artigo, ou visualizar estatísticas).

Assim, no âmbito da cadeira de Base de Dados foi-nos proposto criar e idealizar uma base de dados que tenha de certa forma a capacidade de lidar com tudo o que foi referido anteriormente.

# 2 Aplicação

Optamos por dividir a aplicação *NETFLOX* em 3 ficheiros, *main.py*, *funcoes.py* e *menu.py*.

## *menu.py*

No ficheiro *menu.py*, 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 em menu.

## *funcoes.py*

No ficheiro *funcoes.py*, temos ao dispor todas as funções que permitem ter acesso ao servidor da base de dados de forma a possibilitar inserir, obter ou atualizar dados.

## *main.py*

Relativamente ao ficheiro *main.py* apenas é utilizado como um “executável” onde damos import do ficheiro *menu.py* que

depois chama a função dada pelo nome de *menu.menu\_inicial* que leva o utilizador para o respetivo menu de login.

### 3 Distribuição final das tarefas

As tarefas deste projeto foram distribuídas de forma equitativa de modo que os 2 membros do grupo tivessem a mesma carga de trabalho por isso, o aluno André Araújo ficou encarregue da parte do admin e por sua vez o aluno Bernardo Leite ficou encarregue da parte do cliente.

### 4 Diagrama Entidade-Relacionamento

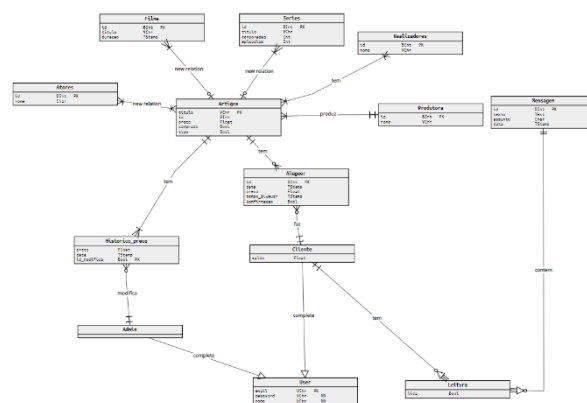


Figura 1: Diagrama Entidade-Relacionamento, meta 1

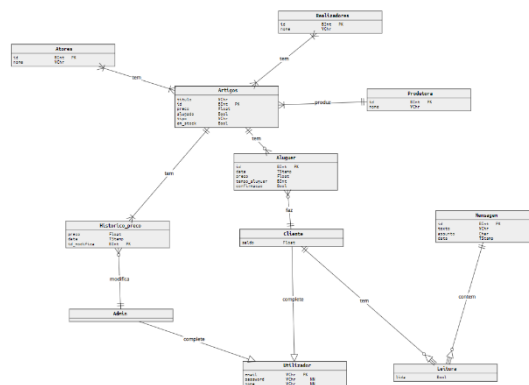


Figura 2: Diagrama Entidade-Relacionamento, meta 2

## Entidades

### User

User divide-se em *cliente* e *admin*. Assim, no diagrama Entidade Relacionamento, *User* é uma super-entidade e tem uma relação de herança mutuamente exclusiva com o cliente e o administrador (ou é uma ou é outra, mas não os dois ao mesmo tempo). O email é primary key, pois o email é único.

### Cliente

A entidade Cliente tem duas relações: uma relação (0:n) com a entidade *aluguer* 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 omissão será atribuído um saldo inicial no valor de 20€.

### Mensagem

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*.

### Leitura

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.

### Adim

A entidade *admin* não precisa de nenhum parâmetro pois herda todos da super-entidade *user*. Tem uma relação com a entidade *historico\_preco* para que na entidade Histórico

preço fique registado o email do administrador que fez a alteração de preço.

### Artigo

A entidade *artigo* tem varias relações: relação “tem” (0:n) com a entidade *aluguer*, para que a entidade *aluguer* tenha a primary key da entidade *artigo*; relação “produz” (1:1) com a entidade *produtora*, tem obrigatoriedade porque para um artigo existir tem de ter uma produtora, um artigo só pode ter uma produtora; tem 2 relações iguais com as entidades *filme* e *serie* pois o artigo só pode ter um *filme* ou uma *serie*; tem também outras 2 relações iguais com as entidades *atores* e *realizadores* pois um artigo pode ter um ou mais atores e um ou mais realizadores; a relação (1:n) com a entidade *Historico preco*, com esta relação fazemos com que seja necessário inserir o preço inicial do artigo no histórico preço.

### Historico\_preco

A primary key desta entidade é um id (id modifica). Esta entidade foi criada para registar todas as alterações de preço que um determinado artigo sofre. Tem apenas uma relação (1:1) com *artigo*, ou seja, o histórico dos preços é único de cada artigo, não há históricos “partilhados”.

### Aluguer

Esta entidade foi criada com a função de registar todos os alugueres realizados pelos clientes. Esta entidade tem duas relações: uma relação (1:1) com a entidade *artigo* pois um aluguer é apenas constituído por um só artigo; uma relação (1:1) com a entidade *cliente* pois um aluguer pertence apenas a um cliente.

## 5 Diagrama Físico

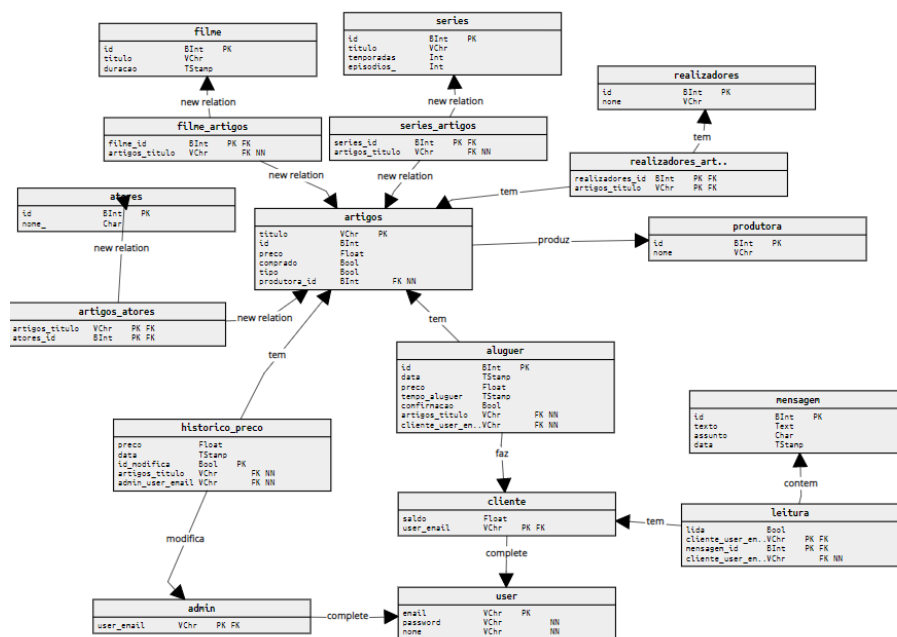


Figura 3:Diagrama Físico, meta 1

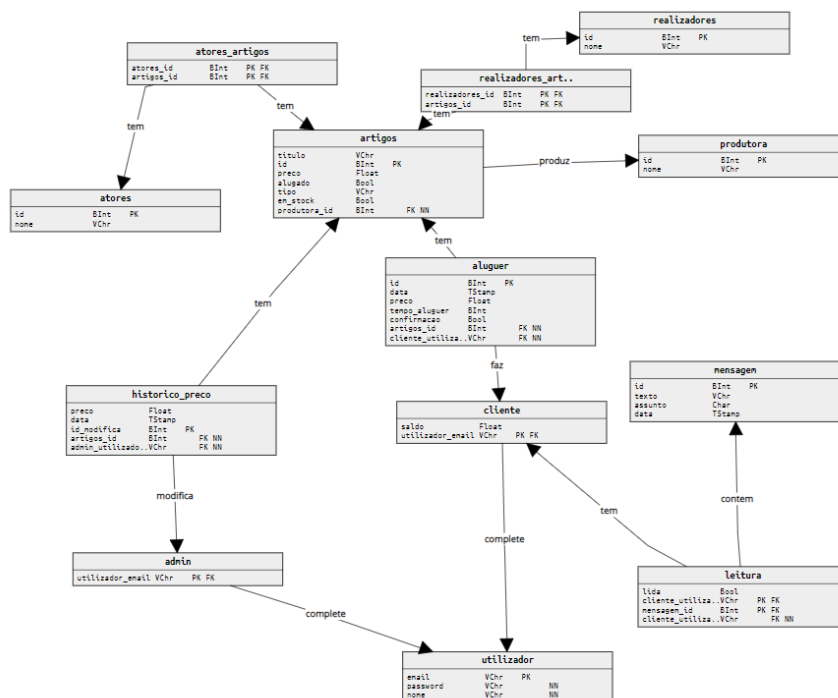


Figura 4:Diagrama Físico, meta 2

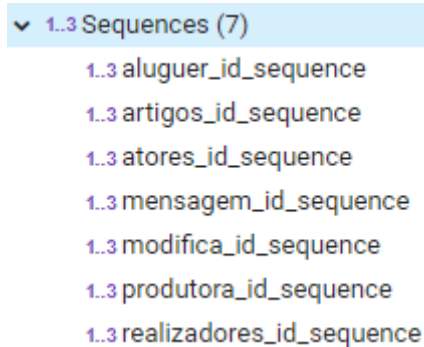
## 6 Triggers e Sequencias

Para inserir os dados na tabela utilizador e na tabela cliente, quando um novo cliente se regista, foi criado um trigger. Para tal acontecer criamos a função trigger cliente insert() e associamos a função ao trigger insere cliente e associamos o trigger à tabela utilizador.

```
CREATE OR REPLACE FUNCTION cliente_insert( )
  RETURNS trigger AS
  $$
  BEGIN
      INSERT INTO cliente ( saldo , utilizador_email )
      VALUES(20,NEW. email) ;
      RETURN NEW;
  END;
  $$
LANGUAGE ' plpgsql ' ;
```

```
CREATE TRIGGER insere_cliente
  AFTER INSERT
  ON utilizador
  FOR EACH ROW
  EXECUTE PROCEDURE cliente_insert( ) ;
```

Criamos também várias sequencias (Fig. 4) todas com o mesmo propósito: incrementar o campo id nas tabelas que tem um campo id, por exemplo, aluguer, realizadores, atores, etc... Para cada campo id criamos uma sequência do tipo:



```
▼ 1.3 Sequences (7)
1.3 aluguer_id_sequence
1.3 artigos_id_sequence
1.3 atores_id_sequence
1.3 mensagem_id_sequence
1.3 modifica_id_sequence
1.3 produtora_id_sequence
1.3 realizadores_id_sequence
```

Figura 5:sequencias

## 7 Extras

No ficheiro *menu.py* damos import há biblioteca *getpass*.  
Decidimos usar a biblioteca *getpass* porque pensamos que quando o utilizador inseria a sua password não devia aparecer os caracteres que o utilizador digitava.  
Para isso usamos:

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

## 8 Screenshots

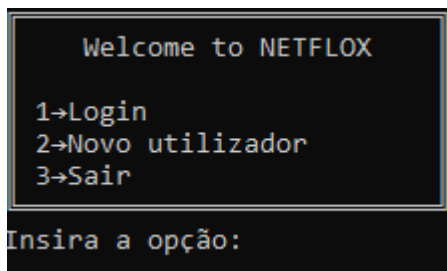


Figura 6: menu inicial

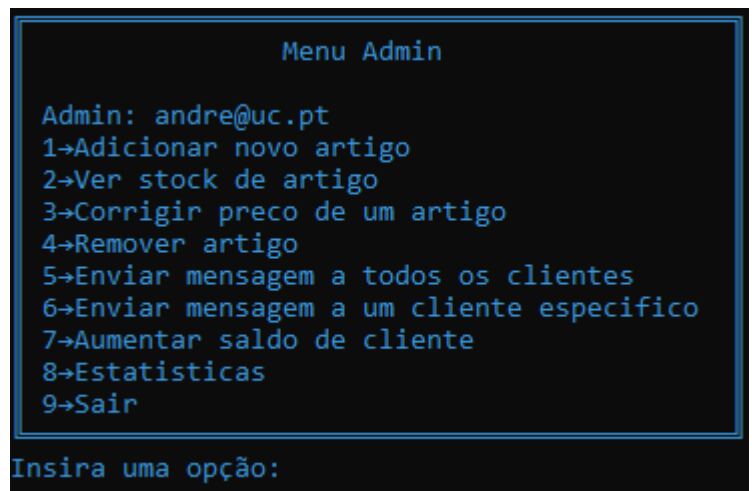


Figura 7: menu admin

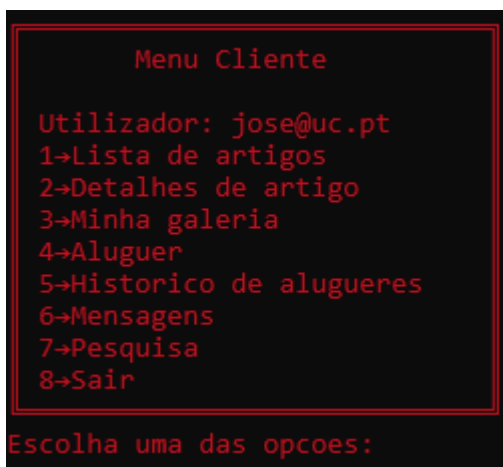


Figura 8: menu cliente

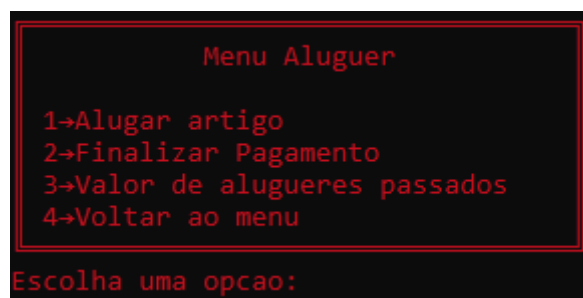


Figura 9: menu aluguer



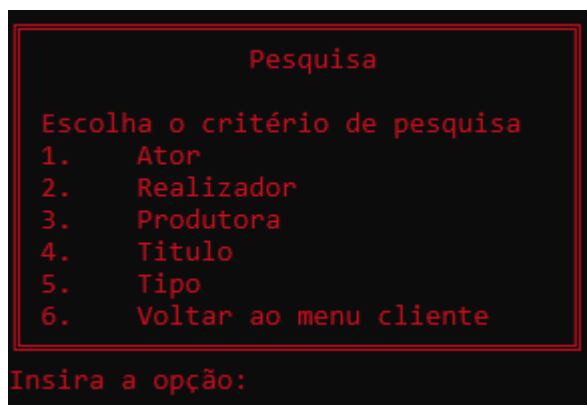


Figura 11: menu pesquisa

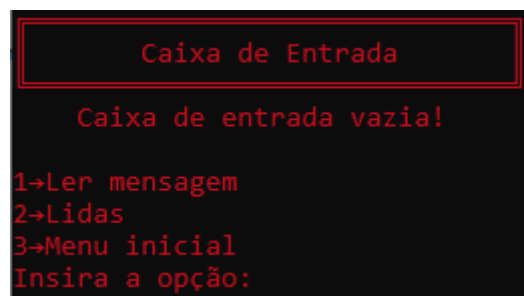


Figura 10: menu mensagem

## 9 Tempo gasto na disciplina

O total de tempo despendido na disciplina ao longo do semestre foi de:

Durante as aulas: Bernardo Leite –  $\pm 3h$ /semana

André Araújo –  $\pm 5h$ /semana

Fora de aulas: Bernardo Leite –  $\pm 2h$ /semana

André Araújo –  $\pm 2h$ /semana