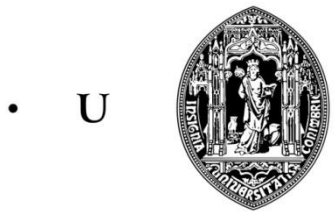


UNIVERSIDADE DE COIMBRA

FACULDADE DE CIÊNCIAS E TECNOLOGIAS

MIEEC – MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

-- Base de Dados--

Projeto (2ª Parte):

"Songs"

Docente: Pedro Nuno San-Bento Furtado

Realizado por:

Manuel Filipe Ribeiro Restolho Mateus **Nº:2012169738**
Miguel Soares Maranha Tiago **Nº:2012138309**

1. Introdução

Nesta segunda parte do projeto da unidade curricular de Base de Dados, pretende-se implementar a base de dados normalizada obtida na primeira parte do trabalho.

Optamos por utilizar como ferramentas de trabalho o software Phyton onde foi feito todo o código, juntamente com o software XAMPP para podermos ligar o código com o a base de dados.

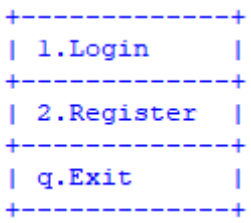
Para podermos ver e editar a base de dados usamos o site http://localhost/phpmyadmin.

Esta base de dados tem como objetivo principal conseguir-se organizar e gerir informação sobre músicas.

2. Descrição da Aplicação- Manual do Utilizador

A aplicação desenvolvida ajuda a organizar e gerir a informação de várias músicas e por exemplo, ajudar um utilizador que esteja à procura de uma música específica e saber em que album está disponível.

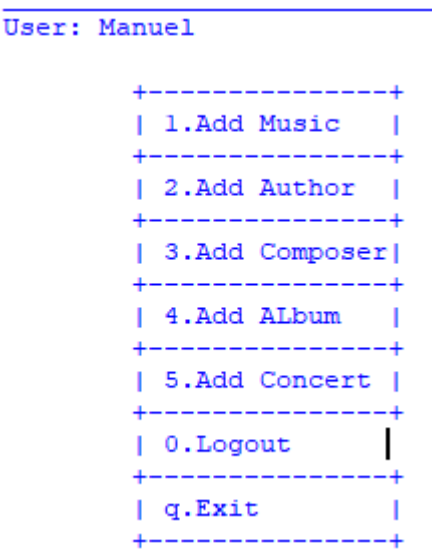
Inicialmente a consola pede um login e caso o utilizador não esteja registado pede o seu registo, chamando a função(int_main).



No caso do utilizador não estar registado, selecionando a opção “2”, o interface pede o nome, o username e uma password encriptada*.

Caso já esteja registado é feito o login (opção “1”), com um username e password.

De seguida é nos apresentado o segundo menu, chamado através da função (editor_menu):



Como demonstrado na imagem ao lado, apenas um utilizador registado e declarado com editor pode adicionar músicas, autores, compositores, álbuns, concertos.

Optamos por ter também uma opção de logout caso queiramos entrar com outro utilizador.

Em caso de logout, volta para o menu da imagem anterior.

Caso o utilizador não seja declarado com editor é lhe apresentado o seguinte menu (user_ menu):

Neste caso, qualquer utilizador desde que esteja registado pode procurar músicas, autores, compositores, álbuns, concertos, fazer uma critica, criar uma playlist ou ver playlist.

Tal como no caso anterior também temos uma opção de logout caso queiramos entrar com outro utilizador.

Em caso de logout, volta para o menu inicial.

3. Descrição da Aplicação- Manual de Instalação

open mysql and import the result from ONDA.com for <bd6_5estrelas.json>`

Install python3

pip3 install pymysql

4. Diagramas ER

Optamos por fazer algumas alterações às tabelas através das linhas de código:

- ALTER TABLE `musica` CHANGE `tempo_da_musica` `tempo_da_musica` TIME NULL DEFAULT NULL;
- ALTER TABLE `membros` CHANGE `mebro_id` `membro_id` BIGINT(20) NOT NULL AUTO_INCREMENT;
- ALTER TABLE album CHANGE ano ano YEAR NULL DEFAULT NULL;
- ALTER TABLE `album_review` ADD `album_album_id` BIGINT(20) NOT NULL AFTER `data`, ADD INDEX `album_review_fk1` (`album_album_id`) USING BTREE;

- ALTER TABLE `album_review` ADD FOREIGN KEY (`album_album_id`) REFERENCES `album`(`album_id`) ON DELETE RESTRICT ON UPDATE CASCADE;
- ALTER TABLE `album_review` ADD `utilizador_user_id` BIGINT(20) NOT NULL AFTER `album_album_id`, ADD INDEX `album_review_fk2` (`utilizador_user_id`) USING BTREE;
- ALTER TABLE `album_review` ADD FOREIGN KEY (`utilizador_user_id`) REFERENCES `utilizador`(`user_id`) ON DELETE RESTRICT ON UPDATE CASCADE;

5. Algumas Funções

a. Get_user

```
#|----  Get User From DB ----| compare with program login
def get_user(user,password,type):
    hashed_pwd=hashlib.sha512(password.encode('utf-8')).hexdigest()
    value = False
    db = None
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = "SELECT username, e_pwd FROM utilizador"
        cursor.execute(sql)
        row = cursor.fetchone()
        while row is not None:
            #Type L (Login) User and PWD
            if type=="l":
                if(user==row[0] and hashed_pwd==row[1].decode('utf-8')):
                    value = True
            #Type R (Register) User only Validaion
            if type =="r":
                if(user==row[0]):
                    value = True
            row = cursor.fetchone()
        cursor.close()
```

Dado o username, password e tipo faz uma comparação quando tipo=l indica que é um login e compara os parâmetros dados com os da base de dados. Quando é do tipo=r apenas verifica se o utilizador já está registado, devolvendo o valor (“true”).

b. User_lvl

```
#|----  Get Usr_lvl ----| admin or editor
def user_lvl(user):
    db = None
    db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
    cursor = db.cursor()
    sql="SELECT user_level FROM utilizador WHERE username = '%s'"%user
    cursor.execute(sql)
    result = cursor.fetchone()
    return(result[0])
```

Vai a base de dados buscar o nível de edição á tabela utilizador onde o username é igual ao parâmetro passado(user). Esta retorna o valor que está na base de dados que é editor ou autor.

c. **Check_user**

```
#|----  Check User ----| returns user_id(int)
def check_user(user):
    db = None
    db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
    cursor = db.cursor()
    sql = "SELECT user_id FROM utilizador WHERE username='%s'"%user
    cursor.execute(sql)
    result = cursor.fetchone()
    if(result!=None):
        return(result[0])
    else:
        return(False)
```

Pesquisa na tabela utilizador o user_id= username dado por parâmetro, se for diferente de NULL, devolve o user_id caso contrário devolve ("false").

d. Insert_autor

```
#|---- Insert Author ----|
def insert_autor(a_nome,a_historia):
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = "INSERT INTO autor(nome,historia)VALUES ('%s','%s')"%(a_nome,a_historia)
        cursor.execute(sql)
        db.commit()
        value = True
    except pymysql.err.OperationalError:
        print("db error, continue")
        value = False
    finally:
        if db is not None:
            db.close()
    return value
```

Insere na tabela autor os valores, nome de autor e história.

e. List_autor

```
#|---- List Author ----| Displays All Authors / Returns Array with a.id
def list_autor():
    cnt=0
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = "SELECT nome,autor_id FROM autor ORDER BY nome"
        #Create Music_ID Array Map
        id_map=[]
        cursor.execute(sql)
        row = cursor.fetchone()
        while row is not None:
            cnt += 1
            print(" %d - %s"%(cnt,row[0]))
            #Append Each Result to Array
            id_map.append(row[1])
            row = cursor.fetchone()
        cursor.close()

    except pymysql.err.OperationalError:
        print("db error, continue")
    finally:
        if db is not None:
            db.close()
    return id_map
```

Vai à tabela autor, buscar o nome e o utilizador_id e faz display a todos os resultados ordenados por nome.

f. lookup_autor

```
#|---- Lookup Author ----|
def lookup_autor(nome_autor):
    cnt=0
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = """SELECT a.nome,m.nome,a.autor_id
                    FROM autor a, musica m
                    WHERE a.autor_id = m.autor_autor_id
                    AND a.nome='%s'
                """%(nome_autor)
        print("|ID|    Autor    |   Nome\n")
        #Create Music_ID Array Map
        id_map=[]
        cursor.execute(sql)
        row = cursor.fetchone()
        while row is not None:
            cnt += 1
            print(" %d  %s '%s'"%(cnt,row[0],row[1]))
            #Append Each Result to Array
            id_map.append(row[2])
            row = cursor.fetchone()
        cursor.close()

    except pymysql.err.OperationalError:
        print("db error, continue")
    finally:
        if db is not None:
            db.close()
    return id_map
```

Seleciona o nome do autor, o nome da música e o autor_id da tabela musica e autor com autor_id=autor_autor_id que vêm da tabela música.

g. Update_music_album

```
#|---- Update Music Album ----| given music id,autor_id Join Music to album id
def update_music_album(music_id,autor_id,album_id):
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = "UPDATE musica SET album_album_id ='%s' WHERE musica_id='%s' \
                AND autor_autor_id='%s' "%(album_id,music_id,autor_id)
        cursor.execute(sql)
        db.commit()
        value = True
    except pymysql.err.OperationalError:
        print("db error, continue")
        value = False
    finally:
        if db is not None:
            db.close()
    return value
```

Vai á tabela música inserir um album, onde a musica_id e o autor_id são passados por parâmetros. Isto é adiciona músicas ao album.

h. Delete_playlist

```
def delete_playlist(pl_id,uid):
    try:
        db = pymysql.connect(host=db_ip,user=db_user,password=db_pwd,db=db_name,port=db_port)
        cursor = db.cursor()
        sql = "DELETE FROM playlist WHERE playlist_id='%s' and utilizador_user_id='%s'"%(pl_id,uid)
        cursor.execute(sql)
        db.commit()
        value = True
    except pymysql.err.OperationalError:
        print("db error, continue")
        value = False
    finally:
        if db is not None:
            db.close()
    return value
```

Vai à tabela playlist procurar o user_id e a playlist_id dados por parâmetros e apaga os dados respectivos.

6. Conclusão

Neste trabalho teve-se a oportunidade de aprender vários conceitos e aplicá-los, relacionados com *PHP*, *Phyton*, *MySQL*.

A falta tempo fez com que a aplicação não tivesse muitas funcionalidades como poderiam ter sido implementados, mas o balanço continua a ser positivo, visto que se adquiriu muitos conhecimentos relacionados com a unidade curricular.