

Inchiriere filme

Proiect la disciplina
Baze de Date

Studenti:
Salavastu Andrei-Ionut 1311A
Teslaru Victor 1311A

Cuprins

Capitolul 1. Introducere	1
Capitolul 2. Instalare aplicații necesare	2
2.1. Click versiunea 6.7	2
2.2. Flask versiunea 0.12.2	2
2.2.1. Django	3
2.2.2. Werkzeug	3
2.2.3. Jinja2	3
2.3. cx_Oracle	4
Capitolul 3. Functionalitatea aplicației	5
3.1. Functia Select	6
3.2. Functia Insert	7
3.3. Functia Update	8
3.4. Functia Delete	9

Capitolul 1. Introducere

Aplicația noastră este creată cu scopul de a tine gestiunea bazei de date a unui serviciu de închirieri de filme. Baza noastră de date cuprinde un număr de 8 tabele:

- Clienti
- Contracte_Inchirieri
- Detalii_Film
- Particularitati
- Film
- Film_Particularitati

Prin această aplicație am încercat să implementăm vizual funcționalitatea principalelor funcții de interogare a unei baze de date. Astfel putem modifica/ accesa o bază de date cu ușurință datorită bibliotecilor din python menționate mai sus.

Principalele funcții folosite sunt:

- Select
- Insert
- Update
- Delete

Capitolul 2. Instalare aplicații necesare

Se instaleaza python versiunea 3.7 care poate fi descarcata de pe site -ul oficial <https://www.python.org/downloads/>.

Aplicația are nevoie de următoarele biblioteci:

2.1. Click versiunea 6.7

Click este un pachet Python folosit pentru a crea interfețe în linia de comandă, într-un mod ușor de înțeles, eficient și ușor de utilizat, cu cât mai puțin cod posibil.

Există mai multe astfel de pachete Python. Însă de ce folosim Click?

Sunt câteva motive pentru care se folosește Click-ul:

- este ușor de compilat și nu are restricții
- este complet imbricat
- generează automat o pagină help

Comanda de instalare:

```
pip install click == 6.7
```

Un exemplu simplu de folosire a bibliotecii click:

```
import click

@click.command()
@click.option('--count', default=1, help='Number of greetings.')
@click.option('--name', prompt='Your name',
              help='The person to greet.')
def hello(count, name):
    """Simple program that greets NAME for a total of COUNT times."""
    for x in range(count):
        click.echo('Hello %s!' % name)

if __name__ == '__main__':
    hello()
```

2.2. Flask versiunea 0.12.2

Flask este un API (Application Programming Interface) Python ce permite construirea de aplicații web. A fost dezvoltat de Armin Ronacher (un programator austriac și un speaker cunoscut la conferințele în domeniul software).

Acest framework este mult mai ușor de înțeles decât framework-ul Django's și mult mai ușor de învățat deoarece are mai puțin cod de implementat în cazul unei aplicații web.

Comanda instalare Flask

```
pip install Flask
```

Un exemplu de folosire a Flask-ului:

```
from flask import Flask
app = Flask(__name__) # Flask constructor
# A decorator used to tells the application
# which URL is associated function
@app.route('/')
def hello():
    return 'HELLO'
if __name__ == '__main__':
    app.run()
```

2.2.1. Django

Django este un framework web bazat pe Python, care vă permite să creați rapid aplicații web fără toate problemele de instalare sau de dependență pe care le veți găsi în mod normal.

Când construiți un site web, aveți întotdeauna nevoie de un set similar de componente: o modalitate de a gestiona autentificarea utilizatorilor (înscrierea, conectarea, deconectarea), un panou de gestionare pentru site-ul dvs. web, formulare, o modalitate de a încărca fișiere etc. Django vă oferă componente gata de utilizat.

De ce Django?

1. Este foarte ușor să schimbați baza de date în cadrul Django.
2. Acesta a construit în interfața de administrare care face ușor de a lucra cu ea.
3. Django este un cadru complet funcțional care nu necesită altceva.
4. Are mii de pachete suplimentare disponibile.
5. Este foarte scalabil.

2.2.2. Werkzeug

Werkzeug este o bibliotecă utilitară pentru limbajul de programare Python , cu alte cuvinte un set de instrumente pentru aplicații WSGI (Web Server Gateway Interface) și este licențiat sub licență BSD . Werkzeug poate realiza obiecte software pentru funcții de solicitare, răspuns și funcții utilitare.

2.2.3. Jinja2

Jinja2 este o librărie pentru Python creata pentru a fi flexibila, rapidă și sigură. Poți instala ultimele versiuni de Jinja2 folosind easy_install sau pip.

```
easy_install Jinja2
pip install Jinja2
```

Caracteristici:

- Conține server de dezvoltare și depanator;
- Suport integrat pentru testarea unităților;
- Utilizează Jinja2 templating;
- Suport pentru cookie-urile securizate;
- Compatibilitatea Google App Engine.

```
from flask import Flask
app = Flask ( __name__ )

@app.route ( "/" )
def hello ():
    return "Hello World!"

if __name__ == "__main__" :
    app . run ()
```

2.3. *cx_Oracle*

cx_Oracle este un modul de extensie Python care permite accesul la Oracle Database.

Conformă cu specificația API 2.0 a bazei de date Python, cu un număr considerabil de adăugiri și câteva excluderi.

Pentru a utiliza cx_Oracle 7 cu Python și Oracle Database aveți nevoie de:

- Python 2.7 sau 3.5 și mai mult. Versiunile mai vechi ale cx_Oracle pot funcționa cu versiuni mai vechi ale Python.
- Bibliotecile client Oracle. Acestea pot fi de la clientul Oracle Instant Client gratuit sau de la cele incluse în Oracle Database dacă Python se află pe aceeași mașină ca și baza de date. Bibliotecile clienților Oracle 19, 18, 12 și 11.2 sunt acceptate pe Linux, Windows și MacOS. Utilizatorii au raportat, de asemenea, succesul cu alte platforme.
- O bază de date Oracle. Standardul de interoperabilitate Oracle standard pentru client-server permite cx_Oracle să se conecteze la bazele de date mai vechi și mai noi.

Instalare

```
python - m pip install cx_Oracle - upgrade
```


Capitolul 3. Funcționalitatea aplicației

Prin aceasta aplicație am încercat să implementăm vizual funcționalitatea principalelor funcții de interogare a unei baze de date. Astfel putem modifica/ accesa o bază de date cu ușurință datorită bibliotecilor din python menționate mai sus.

Principalele funcții folosite sunt:

3.1. *Funcția Select*

Fiecare din următoarele declarații sunt valide:

```
SELECT * FROM CLIENTI;
```

```
SELECT  
*  
FROM  
CLIENTI  
;
```

```
SELECT *  
FROM CLIENTI;
```

Este comanda cea mai utilizată. Este folosită pentru obținerea datelor din bazele de date. Exemplu de utilizare a funcției select pentru afișarea datelor dintr-o tabelă în python.

```
@app.route('/')  
@app.route('/clienti')  
def clt():  
    clienti = []  
  
    cur = con.cursor()  
    cur.execute('select * from clienti')  
    for result in cur:  
        client = {}  
        client['id_client'] = result[0]  
        client['serie_act_identitate'] = result[1]  
        client['tip_act'] = result[2]  
        client['nume'] = result[3]  
        client['prenume'] = result[4]  
        client['email'] = result[5]  
        client['nr_telefon'] = result[6]  
        client['data_nasterii'] = datetime.strptime(str(result[7]),  
            '%Y-%m-%d %H:%M:%S').strftime('%d.%m.%y')  
  
        clienti.append(client)  
    cur.close()  
    return render_template('clienti.html', clienti=clienti)
```

Rezultatul acestui cod este următorul:

id_client	serie_act_identitate	tip_act	nume	prenume	email	nr_telefon	data_nasterii	Editare/Stergere
3	XS	CI	Luchian	Alexandru	alexandru.luchian@student.tuiasi.ro	0756727899	16.02.01	<div>Editaza client</div> <div>Sterge client</div>
2	IS	CI	Teslaru	Victor	victor.teslaru@student.tuiasi.ro	0746540091	23.06.01	<div>Editaza client</div> <div>Sterge client</div>
11	VS	Pasaport	Blanaru	Gabriela	blanaru@student.tuiasi.ro	0744780022	12.01.16	<div>Editaza client</div> <div>Sterge client</div>
1	VS	CI	Salavastu	Andrei	andrei-ionut.salavastu@student.tuiasi.ro	0761609550	14.01.01	<div>Editaza client</div> <div>Sterge client</div>
4	LM	Pasaport	Paduraru	George	george.paduraru@student.tuiasi.ro	0751919164	23.10.01	<div>Editaza client</div> <div>Sterge client</div>

3.2. *Funcția Insert*

Adaugare o noua inregistrare.

Sintaxa este :

```
INSERT INTO tabela [ ( coloana [ , coloana . . . ] ) ]
VALUES ( valoare [ , valoare . . . ] );
```

Exemplu de cod pentru introducerea unor date noi în tabela:

```
@app.route('/addClient', methods=['GET', 'POST'])
def add_clt():
    error = None
    if request.method == 'POST':

        cur = con.cursor()
        values = []

        values.append("'" + request.form['id_client'] + "'")
        values.append("'" + request.form['serie_act_identitate'] + "'")
        values.append("'" + request.form['tip_act'] + "'")
        values.append("'" + request.form['nume'] + "'")
        values.append("'" + request.form['prenume'] + "'")
        values.append("'" + request.form['email'] + "'")
        values.append("'" + request.form['nr_telefon'] + "'")
        values.append(
            "'" +
            datetime.strptime(str(request.form['data_nasterii']), '%Y-%m-%d').strftime('%d-%b-%y')
            + "'"
        )

        fields = ['id_client', 'serie_act_identitate', 'tip_act', 'nume',
                  'prenume', 'email', 'nr_telefon',
                  'data_nasterii']

        query = 'INSERT INTO %s (%s) VALUES (%s)' % ('clienti', ', '.join(fields),
```

```

', '.join(values))

    cur.execute(query)
    cur.execute('commit')
    return redirect('/clienti')
else:
    cont_in = []
    cur = con.cursor()
    cur.execute('select id_client from clienti')
    for result in cur:
        cont_in.append(result[0])
    cur.close()

    return render_template('addClient.html')

```

Pagina pentru insert este următoarea:

3.3. *Funcția Update*

Modifica datele existente cu alte date valide.

Exemplu de cod:

```

@app.route('/editClient', methods=['POST'])
def edit_clt():
    emp = 0

    cur = con.cursor()

    id_client = "" + request.form['id_client'] + ""
    serie_act_identitate = "" + request.form['serie_act_identitate'] + ""
    tip_act = "" + request.form['tip_act'] + ""
    nume = "" + request.form['nume'] + ""
    prenume = "" + request.form['prenume'] + ""
    cur.execute('select id_client from clienti where nume=' + nume)
    for result in cur:

```

```

        emp = result[0]
        cur.close()
        email = "'" + request.form['email'] + "'"
        nr_telefon = "'" + request.form['nr_telefon'] + "'"
        data_nasterii = "'" + datetime.strptime(str(request.form['data_nasterii']),
'%Y-%m-%d').strftime('%d-%b-%y') + "'"

        cur = con.cursor()
        query = "UPDATE clienti SET id_client=%s, serie_act_identitate=%s, tip_act=%s,
nume=%s, prenume=%s, email=%s, nr_telefon=%s, data_nasterii=%s where id_client=%s" % (
            id_client, serie_act_identitate, tip_act, nume, prenume, email,
nr_telefon, data_nasterii, emp)
        cur.execute(query)
        cur.execute('commit')

        return redirect('/clienti')

```

La efectuarea unui update, datele existente sunt citite din tabela și completate automat în campurile necesare pentru a putea modifica doar una sau mai multe campuri fără a fi nevoie de a se completa campurile nemodificate.

Pagina update arata astfel:

Functia de completare automata a campurilor este următoarea:

```

@app.route('/getClient', methods=['POST'])
def get_clt():
    emp = request.form['id_client']
    cur = con.cursor()
    cur.execute('select * from clienti where id_client=' + emp)

    emps = cur.fetchone()
    id_client = emps[0]
    serie_act_identitate = emps[1]
    tip_act = emps[2]
    nume = emps[3]
    prenume = emps[4]
    email = emps[5]
    nr_telefon = emps[6]

```

```

        data_nasterii = datetime.strptime(str(emps[7]),
'%Y-%m-%d %H:%M:%S').strftime('%d.%m.%Y')
        cur.close()

    return render_template('editClient.html', id_client=id_client,
serie_act_identitate=serie_act_identitate,
tip_act=tip_act, nume=nume, prenume=prenume,
email=email, nr_telefon=nr_telefon,
data_nasterii=data_nasterii)

```

3.4. *Funcția Delete*

Sterge o înregistrare din tabela.

Exemplu de cod utilizat în aplicația noastră:

```

@app.route('/delDetalii', methods=['POST'])
def del_detalii():
    emp = request.form['id_film']
    cur = con.cursor()
    cur.execute('delete from detalii_film where id_film=' + emp)
    cur.execute('commit')
    return redirect('/detalii_film')

```