

Proiect BD: Sistem de Gestionare a Site-ului Companiei Aeriene Tarom

Studenti:

Parvana Vlad-Stefan

Vartolomei Mihai-Sebatian

Cadru didactic coordonator:

Tizu Matei-Victor

Descrierea proiectului

Proiectul de gestionare a site-ului pentru compania aeriană Tarom reprezintă o soluție comprehensivă și eficientă pentru facilitarea accesului la informații despre zboruri și achiziționarea de bilete online. Cu un accent puternic pe utilizabilitate și funcționalitate, acest sistem oferă utilizatorilor o experiență intuitivă și plăcută în procesul de călătorie. Proiectul de gestionare a site-ului Tarom se axează pe aducerea unei soluții moderne și eficiente pentru călătoriile aeriene. Cu o interfață atrăgătoare, acest sistem redefineste experiența călătoriilor aeriene și facilitează procesul de planificare și achiziționare a билетelor pentru călătoriile cu Tarom.

Tehnologii folosite

FRONT-END:

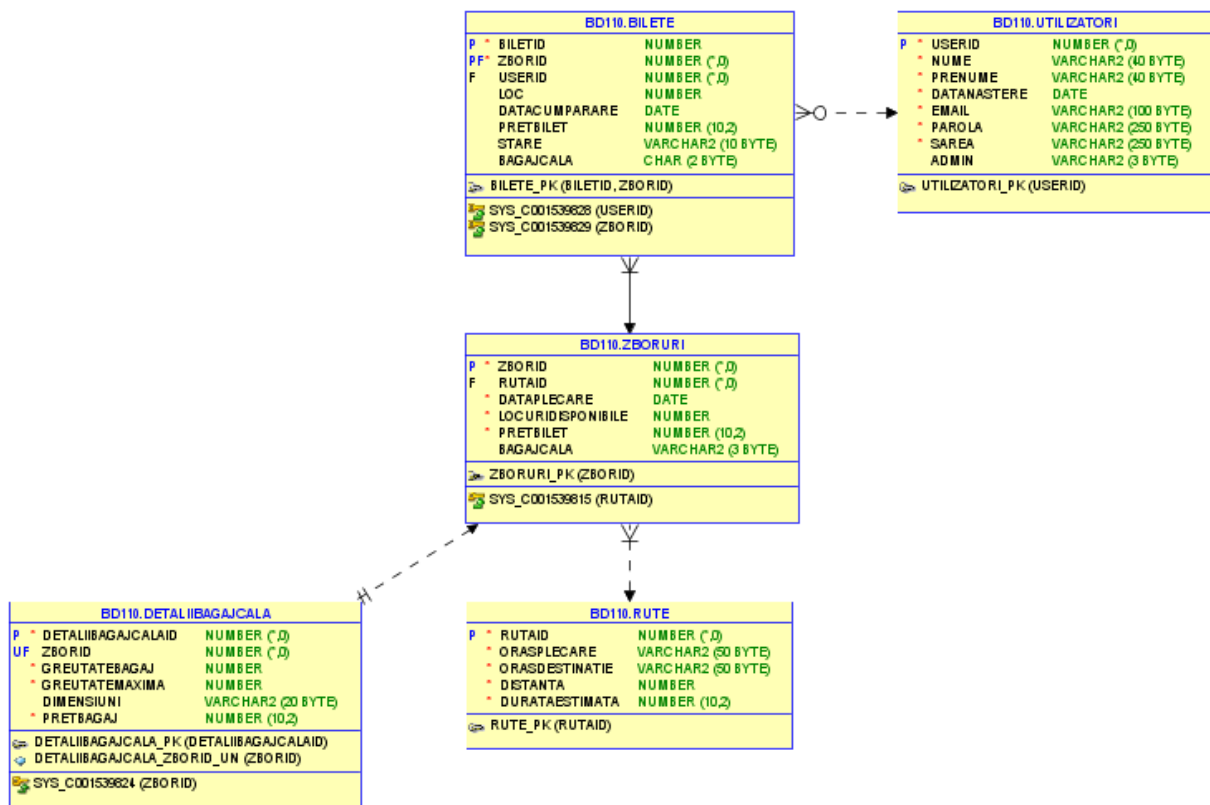
- **HTML:** este un limbaj standardizat utilizat pentru a crea structura de bază a paginilor web. Acesta constă într-un set de elemente sau etichete, fiecare având un rol specific în organizarea și afișarea conținutului pe pagină.
- **Bootstrap4:** este un framework front-end care simplifică dezvoltarea paginilor web, permițând dezvoltatorilor să creeze rapid și eficient interfețe responsive și estetice.

Bootstrap furnizează o colecție extinsă de componente, stiluri predefinite și funcționalități JavaScript, facilitând astfel procesul de construire a paginilor web moderne.

BACK-END:

- FLASK: este un framework web open-source care se concentrează pe simplitate și extensibilitate. Flask oferă un set de instrumente și facilități esențiale pentru construirea aplicațiilor web eficiente și ușor de întreținut.
- CLICK: este un pachet Python folosit pentru a crea interfețe în linia de comanda, într-un mod ușor de înțeles, eficient și ușor de utilizat, cu cât mai puțin cod posibil.
- FLASK LOGIN: este o extensie pentru framework-ul Flask în Python, care simplifică gestionarea autentificării utilizatorilor într-o aplicație web. Acesta oferă funcționalități pentru gestionarea sesiunilor, verificarea autentificării și manipularea utilizatorilor autentificați.
- 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.*
- JINJA2: este un motor de șabloane pentru limbajul de programare Python și este frecvent folosit în aplicațiile web bazate pe framework-ul Flask. A fost dezvoltat pentru a permite integrarea eficientă a șabloanelor în aplicațiile Python, permițând în același timp o sintaxă ușor de înțeles și de utilizat.
- CX ORACLE: este un modul Python care furnizează o interfață pentru accesul la bazele de date Oracle. Este un driver oficial Oracle pentru Python și permite dezvoltatorilor să conecteze aplicațiile Python la baze de date Oracle, să execute interogări SQL și să manipuleze datele într-un mod eficient.
- BCRYPT: este o bibliotecă de hashing de parole proiectată pentru a oferi securitate împotriva atacurilor de tip "brute force" și "rainbow table" asupra parolelor. Această bibliotecă este frecvent utilizată în dezvoltarea aplicațiilor back-end pentru a asigura stocarea securizată a parolelor utilizatorilor.

Structura si inter-relationarea tabelelor



Fiecare tabel are o cheie primară și coloane care depind direct de cheia primară (**UserID** pentru **Utilizatori**, **RutaID** pentru **Rute**, etc.). Fiecare tabel servește unui scop specific și nu există redundanță semnificativă în datele stocate. Cheile străine sunt utilizate pentru a stabili relațiile între tabele. De exemplu, **Zboruri** are o cheie străină către **Rute** prin intermediul coloanei **RutaID**, și **DetaliiBagajCala** și **Bilete** au chei străine către **Zboruri**. Constrângerile, cum ar fi **CHECK**, sunt utilizate pentru a asigura integritatea datelor și pentru a stabili reguli asupra valorilor admise. **DataNastere** și **DataPlecare** sunt stocate ca **DATE**, ceea ce este adecvat pentru informații legate de dată. **DataCumparare** are un default setat pe **CURRENT_DATE**, ceea ce este o opțiune bună pentru a captura data actuală la momentul inserării.

- **Prima Formă Normală (1NF):**

O tabel este în 1NF dacă fiecare valoare din fiecare coloană este atomică (nu poate fi divizată în componente mai mici).

- **A Doua Formă Normală (2NF):**

O tabel este în 2NF dacă este în 1NF și dacă fiecare coloană care nu face parte din cheia primară este complet dependentă de cheia primară.

Exemplu: În tabelul Zboruri, DataPlecare, LocuriDisponibile, PretBilet, și BagajCala sunt complet dependente de cheia primară ZborID.

- **A Treia Formă Normală (3NF):**

O tabelă este în 3NF dacă este în 2NF și dacă nu există dependențe tranzitive între coloane.

Exemplu: În tabelul Utilizatori, coloana Admin este dependentă de cheia primară UserID, nu de altă coloană din tabel.

Constrangeri

1. Constrângerile pe Secvențe:

- **CREATE SEQUENCE utilizatori_seq:** Această secvență este creată pentru a furniza o valoare unică pentru coloana **UserID** din tabela **Utilizatori**. Secvențele sunt adesea utilizate pentru a asigura identificatori unici și monotoni.
- **CREATE SEQUENCE rute_seq:** Similar cu **utilizatori_seq**, această secvență furnizează valori unice pentru coloana **RutaID** din tabela **Rute**.
- **CREATE SEQUENCE zboruri_seq:** Asigură valori unice pentru coloana **ZborID** din tabela **Zboruri**.
- **CREATE SEQUENCE bagajcala_seq:** Furnizează valori unice pentru coloana **DetaliiBagajCalaID** din tabela **DetaliiBagajCala**.

2. Constrângerile în Tabela Utilizatori:

- **PRIMARY KEY (UserID):** Definind o cheie primară pe coloana **UserID**, se asigură că fiecare utilizator are un identificator unic.
- **CHECK (admin IN ('DA', 'NU')):** O constrângere de verificare pentru a asigura că coloana **Admin** poate avea doar valorile 'DA' sau 'NU'.

3. Constrângerile în Tabela Rute:

- **PRIMARY KEY (RutaID):** Asigură că fiecare rută are un identificator unic.
- **CHECK (Distanța > 0):** Asigură că distanța este o valoare pozitivă.
- **CHECK (DurataEstimată > 0):** Asigură că durata estimată este o valoare pozitivă.

4. Constrângerile în Tabela Zboruri:

- **PRIMARY KEY (ZborID):** Asigură că fiecare zbor are un identificator unic.
- **CHECK (LocuriDisponibile >= 0):** Asigură că numărul de locuri disponibile este non-negativ.
- **CHECK (PretBilet > 0):** Asigură că prețul biletului este o valoare pozitivă.
- **CHECK (BagajCala IN ('DA', 'NU')):** O constrângere de verificare pentru a asigura că coloana **BagajCala** poate avea doar valorile 'DA' sau 'NU'.

- **FOREIGN KEY (RutaID) REFERENCES Rute(RutaID):** Creează o cheie străină care asociază fiecare zbor cu o rută.

5. Constrângerile în Tabela DetaliiBagajCala:

- **PRIMARY KEY (DetaliiBagajCalaID):** Asigură că fiecare înregistrare de detalii despre bagajul de cală are un identificator unic.
- **UNIQUE (ZborID):** Asigură că există o legătură unică între **DetaliiBagajCala** și **Zboruri** prin intermediul coloanei **ZborID**.
- **CHECK (GreutateBagaj > 0):** Asigură că greutatea bagajului este o valoare pozitivă.
- **CHECK (GreutateMaxima >= 0):** Asigură că greutatea maximă a bagajului este non-negativă.
- **CHECK (PretBagaj > 0):** Asigură că prețul bagajului este o valoare pozitivă.
- **FOREIGN KEY (ZborID) REFERENCES Zboruri(ZborID):** Creează o cheie străină pentru a asocia fiecare detaliu despre bagaj cu un zbor specific.

6. Constrângerile în Tabela Bilete:

- **PRIMARY KEY (BiletID, ZborID):** Asigură că fiecare bilet are un identificator unic în cadrul unui zbor.
- **DEFAULT CURRENT_DATE:** O valoare implicită pentru coloana **DataCumparare**, care va fi setată la data curentă.
- **CHECK (Stare IN ('Valabil', 'Cumparat')):** O constrângere de verificare pentru a asigura că coloana **Stare** poate avea doar valorile 'Valabil' sau 'Cumparat'.
- **CHECK (BagajCala IN ('DA', 'NU')):** O constrângere de verificare pentru a asigura că coloana **BagajCala** poate avea doar valorile 'DA' sau 'NU'.
- **FOREIGN KEY (UserID) REFERENCES Utilizatori(UserID) ON DELETE SET NULL:** Creează o cheie străină pentru a asocia fiecare bilet cu un utilizator specific (cu opțiunea de a seta la **NULL** în cazul ștergerii utilizatorului).
- **FOREIGN KEY (ZborID) REFERENCES Zboruri(ZborID):** Creează o cheie străină pentru a asocia fiecare bilet cu un zbor specific.

Configurarea conexiunii

- Se folosește **cx_Oracle.makedsn** pentru a construi string-ul de conexiune pe baza detaliilor serverului Oracle (adresa, portul, și numele serviciului).
- Se utilizează **cx_Oracle.connect** pentru a stabili conexiunea la baza de date folosind un nume de utilizator și o parolă.
- Se configurează cheia secretă pentru a asigura securitatea sesiunii.
- **@login_required** este un decorator oferit de Flask-Login pentru a asigura că doar utilizatorii autentificați au acces la această rută.
- Se creează un cursor Oracle pentru a executa interogări și a interacționa cu baza de date.


- Sunt efectuate diverse interogări și operații de inserare sau actualizare a datelor în baza de date.
- La sfârșitul scriptului, conexiunea la baza de date este închisă folosind **conn.close()**.
- Se utilizează blocuri **try-except** pentru tratarea excepțiilor și gestionarea tranzacțiilor în mod corespunzător.

Interfața aplicației

1.Functia Select:

SELECT * FROM ZBORURI ORDER BY ZBORID;

```
@app.route('/zboruri_admin')
@login_required
def zboruri_admin():
    if current_user.is_admin:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Zboruri order by ZBORID")
        zbor_data = cursor.fetchall()
        cursor.close()
        return render_template('zboruri_admin.html', zbor_data=zbor_data)
    else:
        return redirect(url_for('index'))
```

Zboruri Ruta  Utilizatori Statistici Iesire					
Adauga zbor					
ID	Ruta	Data plecare	Locuri disponibile	Pret bilet	Bagaj de cala permis
49	9	2024-01-06 00:00:00	0	1.0	NU
50	9	2024-01-07 00:00:00	0	1.0	NU
51	3	2024-01-07 00:00:00	12	2.0	DA
52	12	2024-01-07 00:00:00	27	40.0	DA

© Parvana Vlad-Stefan & Vartolomei Mihai-Sebastian
Proiect BD 2023-2034

2.Functia Delete:

DELETE FROM RUTE WHERE RutaID = ...;

```

@app.route('/display_ruta/<int:ruta_id>', methods=['GET', 'POST'])
@login_required
def display_ruta(ruta_id):
    if current_user.is_admin:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM RUTE WHERE RutaID = :ruta_id", ruta_id=ruta_id)
        ruta_data = cursor.fetchone()

        cursor.execute("SELECT COUNT(*) FROM ZBORURI WHERE RutaID = :ruta_id", ruta_id=ruta_id)
        numar_zboruri = cursor.fetchone()[0]

        if request.method == 'POST':
            if numar_zboruri == 0:
                cursor.execute("DELETE FROM RUTE WHERE RutaID = :ruta_id", ruta_id=ruta_id)
                conn.commit()
                cursor.close()
                return render_template('mesaj.html', title="Succes",
                                      message="Ruta a fost stearsa.", tipActiune="Acasa",
                                      redirectLink=url_for('rute'))
            else:
                cursor.close()
                return render_template('mesaj.html', title="Eroare",
                                      message="Nu poți șterge ruta pentru că este asociată cu zboruri existente.",
                                      tipActiune="Acasa", redirectLink=url_for('rute'))

        cursor.close()
        if ruta_data:
            return render_template('display_ruta.html', ruta_data=ruta_data)
        else:
            return render_template('mesaj.html', title="Eroare",
                                  message="Ruta nu există.", tipActiune="Acasa",
                                  redirectLink=url_for('rute'))
    return redirect(url_for('index'))

```

Ruta 12

Oras plecare	Oras destinatie
Iasi	Cluj
Distana (km)	Durata estimata (ore)
352	1.5
<div>Sterge</div>	

Succes

Ruta a fost stearsa.

Acasa

Eroare

Nu poți șterge ruta pentru că este asociată cu zboruri existente.


Acasa

3.Functia Update:

UPDATE Utilizatori SET NUME= ..., PRENUME = ..., EMAIL = ... WHERE USERID =

$$\dots,$$

```
@app.route('/modificare_informatii', methods=['GET', 'POST'])
@login_required
def modificare_informatii():
    if not current_user.is_admin:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Utilizatori WHERE USERID = :user_id", user_id=current_user.id)
        user_data = cursor.fetchone()
        if request.method == 'POST':
            nume= request.form['nume']
            prenume = request.form['prenume']
            email = request.form['email']
            if not contains_digits(nume) and not contains_digits(prenume):
                cursor.execute("SELECT * FROM Utilizatori WHERE EMAIL=: email AND USERID <>
:user_id",email=email, user_id=current_user.id)
                if cursor.fetchone() is None:
                    cursor.execute("UPDATE Utilizatori SET NUME= :nume, PRENUME = :prenume, EMAIL =
:email WHERE USERID =: user_id",nume=nume,prenume=prenume,email=email,user_id=current_user.id)
                    conn.commit()
                    cursor.close()
                    return redirect(url_for('contul_meu'))
            else:
                return render_template('mesaj.html', title="Eroare",
                    message="Exista deja un cont pe acest email.",
                    tipActiune="Acasa",
                    redirectLink=url_for('contul_meu'))
        else:
            return render_template('mesaj.html', title="Eroare",
                message="Numele si prenumele nu pot contine cifre.",
                tipActiune="Acasa",
                redirectLink=url_for('contul_meu'))
```

Zboruri Biletele mele  Contul meu Iesire

Utilizator 61

Nume

Prenume

Data de nastere

Email

Salveaza

© Parvana Vlad-Stefan & Vartolomei Mihai-Sebastian
Proiect BD 2023-2034

4. Functia Insert:

INSERT INTO RUTE (ORASPLECARE, ORASDESTINATIE, DISTANTA, DURATAESTIMATA) VALUES (... , ... , ... , ...);

```
@app.route('/new_ruta', methods=['GET', 'POST'])
@login_required
def new_ruta():
    if current_user.is_admin:
        if request.method == 'POST':
            orasPlecare = request.form['OrasPlecare']
            orasDestinatie = request.form['OrasDestinatie']
            distanta = request.form['Distanta']
            durataEstimata = request.form['DurataEstimata']
            if contains_digits(orasPlecare) or contains_digits(orasDestinatie):
                return render_template('mesaj.html', title="Eroare",
                                       message="Orasele nu pot contine cifre in denumire.", tipActiune="Acasa",
                                       redirectLink=url_for('rute'))
            if not contains_letters(durataEstimata):
                return render_template('mesaj.html', title="Eroare",
                                       message="Durata estimata trebuie sa fie numar", tipActiune="Acasa",
                                       redirectLink=url_for('rute'))

            cursor = conn.cursor()
            cursor.execute("SELECT * FROM RUTE WHERE ORASDESTINATIE = :orasDestinatie and
ORASPLECARE = :orasPlecare", orasPlecare=orasPlecare, orasDestinatie=orasDestinatie)
            if cursor.fetchone() is None:
                try:
```

```

cursor.execute("""
    INSERT INTO RUTE (ORASPLECARE, ORASDESTINATIE, DISTANTA, DURATAESTIMATA)
    VALUES (:1, :2, :3,:4)
    """, (orasPlecare, orasDestinatie, distanta, durataEstimata))
conn.commit()
return render_template('mesaj.html', title="Success",
    message="Ruta a fost adaugata in baza de date.", tipActiune="Acasa",
    redirectLink=url_for('rute'))
except Exception :
    conn.rollback()
    return render_template('mesaj.html', title="Eroare",
        message="Campurile nu au fost completate corect.", tipActiune="Acasa",
        redirectLink=url_for('rute'))

else:
    return render_template('mesaj.html', title="Eroare",
        message="Ruta este deja in baza de date.", tipActiune="Acasa",
        redirectLink=url_for('rute'))
return render_template('new_ruta.html')
return redirect(url_for('index'))

```

Zboruri Rute  Utilizatori Statistici Iesire

Ruta noua

Oras plecare	Oras destinatie
<input type="text" value="Iasi"/>	<input type="text" value="Bucuresti"/>
Distana (km)	Durata estimata (ore)
<input type="text" value="400"/>	<input type="text" value="1.5"/>
<input type="button" value="Adauga"/>	

© Parvana Vlad-Stefan & Vartolomei Mihai-Sebastian
Proiect BD 2023-2034

Contributii:

- Parvana Vlad-Stefan: front-end ul si back-end ul aplicatii pe partea de administrator si inregistrare.
- Vartolomei Mihai-Sebastian: front-end ul si back-end ul aplicatiei pe partea de utilizator si autentificare.