MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

PROIECT

la disciplina
Introducere în Baze de Date

Airport Management

Danciu Maria Alexandra

Miron Andreea Gentiana

Grupa 30222,CTI

An academic :2019 – 2020

PROIECT de SEMESTRU
Catedra de Calculatoare
Coordonator: s.l. ing. Cosmina IVAN

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Cuprins

1. Introducere

În ultimii ani, dezvoltarea sistemelor de baze de date reprezintă unul dintre cele mai importante aspecte în domeniul tehnologiei informaţiei, având un impact decisiv asupra modului de organizare şi funcţionare a numeroaselor instituţii şi servicii. Acestea sunt companiile de comunicaţie, intreprinderile de comerţ, serviciile bancare, serviciile de transport, asigurările, universităţile etc. Ele sunt dependente de funcţionarea corectă şi neîntreruptă a sistemelor de baze de date.

O bază de date este o colecţie de date centralizate, creată şi menţinută computerizat, în scopul prelucrării datelor în contextul unui set de aplicaţii. Prelucrarea datelor se referă la operaţiile de introducere, ştergere, actualizare şi interogare a datelor.

Sistemele de baze de date sunt o componentă importantă a vieţii de zi cu zi în societatea modernă. Zilnic, majoritatea persoanelor desfăşoară activităţi care implică interacţiunea cu o bază de date: depunerea sau extragerea unei sume de bani din bancă, rezervarea biletelor de tren sau de avion, căutarea unei cărţi într-o bibliotecă computerizată, gestiunea angajaţilor dintr-o firmă, cumpărarea unor produse etc.

Unul dintre domeniile în care cele menționate mai sus au o importanță majoră este gestionarea unui aeroport.

Astfel am decis să creăm o bază de date complexă ce va întruni toate datele aeroportului international Avram Iancu Cluj-Napoca pentru ca acesta să poată opera pe 64 de zboruri prin intermediul a 15 Companii aeriene ce deservesc 29 de desinații mondiale și să ofere absolut toate opțiunile de management posibile.

De asemenea, pentru a facilita accesul într-un mod interactiv la această bază de date am implementat o interfață ce întrunește o gamă largă de opțiuni / operații în dependență de actor. Deoarece planificarea unei călătorii devine din ce în ce mai

complicată cu fiecare an datorită numărului intermediarilor (agențiile de turism, alte site-uri de rezervare a biletelor etc.) ce oferă opțiunile disponibile conform cerințelor clientului, însă prețurile acestor opțiuni diferă enorm de adevăratul preț al unui zbor. Astfel am decis să încadrăm chiar pe site-ul aeroportului un sistem de rezervare online a biletelor printr-un simplu click, un sistem fără plăți ascunse și care funcționează garantat.

Scopul nostru este de a oferi direct pe site-ul aeroportului clienților informațiile necesare și opțiunea de rezervare a unui bilet într-un mediu user-friendly, operatorilor –acces la opțiunile de configurare specifice, administratorilor – acces la toate opțiunile/ operațiile de adăugarea / modificarea/ reinnoirea /ștergerea datelor, cât și la cele de editare a site-ului.

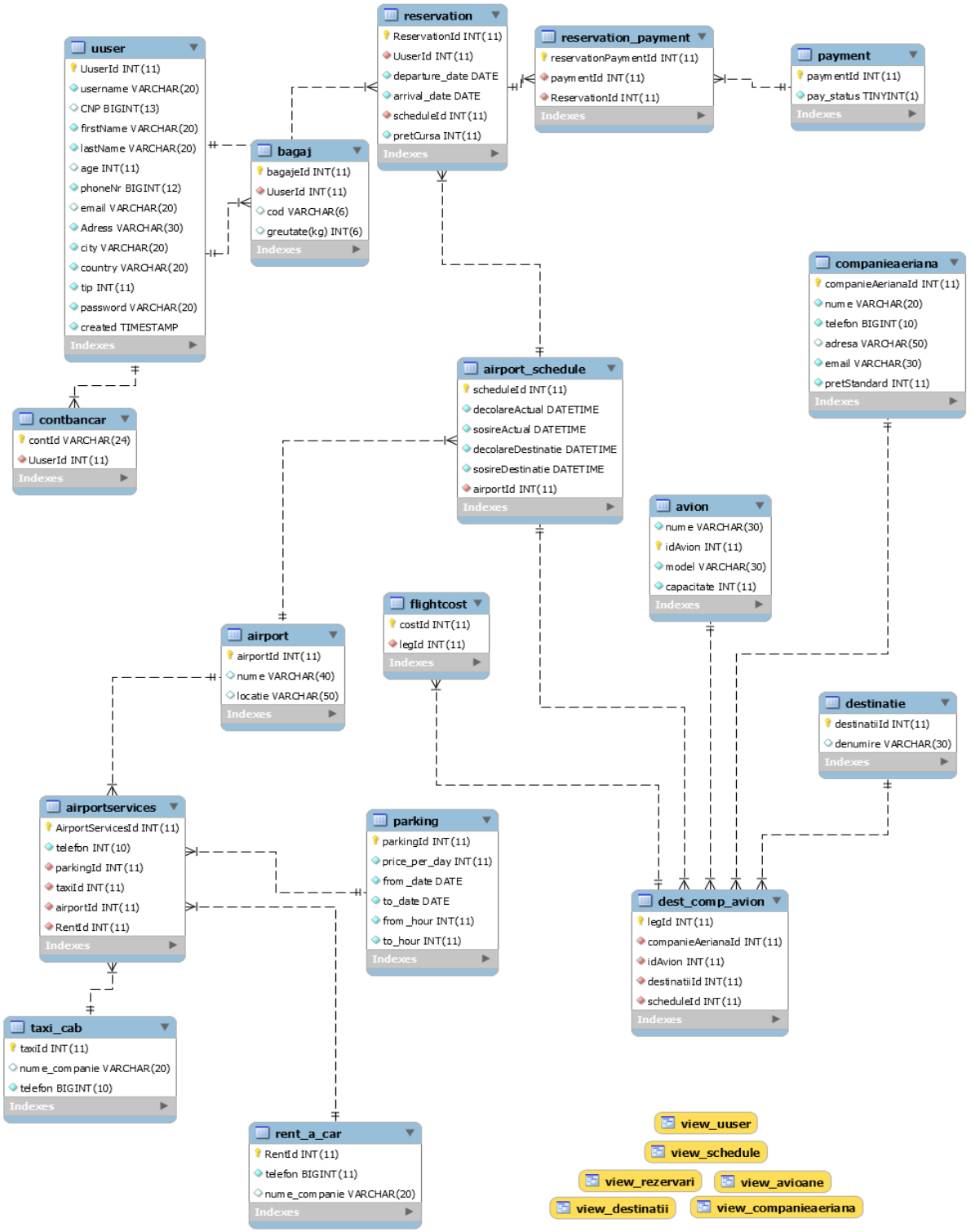
2. Specificațiile de proiect

Implementarea acestui proiect constă în crearea unei baze de date complexe care va servi ca o sursă dinamică de informații (datele bazei de date pot fi reînnoite/ modificate/ șterse și completate cu informație nouă direct prin intermediul interfeței); dezvoltarea unei aplicații care să ofere toate opțiunile de vizualizare, accesare a informației din baza de date, toate operațiile de administrare și management necesare în cadrul unui aeroport.

În cadrul bazei de date , am creat tabelele astfel încât, să existe o corespondeță logică între acestea și să nu existe cicluri, legătura dintre tabele putând fi efectuată prin intermediul cheilor străine și diverselor instrucțiuni MySQL. Fiecare tabel are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date.

3. Modelul de date

3.1 Diagrama EER



3.2 Entități și descrierea lor

1.Destinatii

Fiecare înregistrare este identificată cu un ID unic și are ca atribut denumirea destinației. Tabela *Destinatie* are cheia primară **destinatiiId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS  Destinatie(
        destinatiiId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        denumire  VARCHAR(30)
);
```

Modelul de inserare a datelor in tabela Destinatie:

```
INSERT INTO  `Destinatie`(`destinatiiId`,`denumire`) VALUES
(1,'Antalya'),(2,'Atena'),(3,'Barcelona'),(4,'Bologna'),
(5,'Bucuresti'),(6,'Dubai'),(7,'Dublin'),(4,'Frankfurt');
```

2.Companii Aeriene

Fiecare înregistrare nu are un ID unic și are următoarele atribute: numele companiei, telefon,adresa,e-mailsi un pret standard . Tabela *companieAeriana* are cheia primară **companieAerianaId** și o cheie straină **idDestinatii**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS companieAeriana(
        companieAerianaId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        nume VARCHAR(20) NOT NULL,
        telefon BIGINIT(10) NOT NULL,
        adresa VARCHAR(50) ,
        email VARCHAR(30) NOT NULL,
        pretStandard INT NOT NULL
);
```

Modelul de inserare a datelor in tabela companieAeriana:

```
INSERT INTO`companieAeriana`(`companieAerianaId`,`nume`,`telefon`,`adresa`,`email`,
pretStandard`) `VALUES
```

```
(1, 'Tarom ',0264432669,  'Piata Mihai Viteazu, nr. 11', 'agcluj@tarom.ro', 125), (2,
'Lufthansa ', 0216550719 ,  Aleea Alexandru 9A, Bucuresti, 011821', 'www.lufthansa.com',
150);
```

3.Avion

Fiecare înregistrare este identificată cu un ID unic și are urmatoarele atribute: numele aeronavei, modelul si capacitatea acestuia. Tabelul *Avion* are cheia primară **idAvion**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS Avion (
        idAvion INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        nume VARCHAR(30) NOT NULL,
        model VARCHAR(15) NOT NULL,
        capacitate INT NOT NULL
);
```

Modelul de inserare a datelor in tabela *Avion*:

```
INSERT INTO `Avion`( `idAvion`,`nume`, `model`,`capacitate`) VALUES
(1,'Airbus', 'A 319' , 132),
(2,'Airbus', 'A 318' , 117),
(3,'Airbus', 'A 320-200' , 150),
(4,'ATR', 42-500' , 48);
```

4.Parcare

Tabela *parking* conține un ID unic și are următoarele atribute: pretul pe o zi,data inregistrarii,data eliberarii ,ora de la care se inregistreaza masina in parcare ,ora la care trebuie sa paraseasca zona.Are cheia primară **parkingId**

Modelul de inserare a datelor in tabela *parking*:

```
INSERT INTO  `parking`(`price_per_day`,`from_date `,`to_date`,`from_hour`,`to_hour`)
VALUES
(20,' 2019-08-12',' 2019-08-19',14, 20), (20,'2019-08-13,'2019-08-23',21, 1);
```

5.Orar

Tabel *Airport_Schedule* conține un ID unic și are următoarele atribute: data decolarii din aeroportul referinta, sosirea la destinatie, decolarea din destinatie si sosirea in aeroportul referinta. Tabela *Airport_Schedule* are cheia primară **scheduleId** .

```
CREATE TABLE IF NOT EXISTS Airport_Schedule (
        scheduleId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        decolareActual  DATETIME NOT NULL,
        sosireActual  DATETIME NOT NULL,
        decolareDestinatie  DATETIME NOT NULL,
        sosireDestinatie DATETIME NOT NULL,
        airportId INT NOT NULL
);
```

Modelul de inserare a datelor in tabela *Airport_Schedule* :

```
INSERT INTO `Airport_Schedule`( `scheduleId`,`decolareActual`, `sosireActual`,
`decolareDestinatie`,`sosireDestinatie`,`airportId`) VALUES
(1,  ' 2020-08-12 14:30:00,' 2020-08-12 16:30:00', ,' 2020-08-19 17:20:00,' 2020-08-19
19:20:00', 1),
(2,  ' 2020-08-13 20:00:00,' 2020-08-13 21:50:00', ' 2020-08-22 22:50:00', ' 2020-08-22
00:40:00',  1);
```

6.Rent a car

Tabela este identificată cu un ID unic și are următoarele atribute:telefonul si numele companiei. Are cheia primară **RentID**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS Rent_a_car(
        RentId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        telefon BIGINT(11) NOT NULL,
        nume_companie VARCHAR(20)
 );
```

Modelul de inserare a datelor in tabela *Rent_a_car* :

```
INSERT INTO `Rent_a_car`(`Rentid`,`telefon`,`nume_companie`) VALUES
(1, 0747288200, 'PHP'),
(2, 0264274046, 'SIXT'),
(1, 0264450711, 'Rodna');
```

7.Taxi

Tabela este identificată cu un ID unic și are următoarele atribute: numele companiei de taxi și telefonul. Are cheia primară **taxiId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS taxi_cab (
    taxiId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nume_companie VARCHAR(20) ,
    telefon BIGINT(10) NOT NULL
    );
```

Modelul de inserare a datelor in tabela *taxi_cab*:

```
INSERT INTO `taxi_cab`(`taxiId`, `nume_companie`,`telefon`) VALUES
(1, 'Diesel', 0744646663),
(2, 'Pritax', 0264942),
(3, 'Nova', 0264949);
```

8.User

Tabela *Uuser* conține toate informațiile despre utilizatorii înregistrați pe site. Entitatea este identificată cu un ID unic și are următoarele atribute: username,parola utilizatorului, nume, prenume, CNP,varsta, adresa,oras,tara, telefon, email și tip. Atributul tip definește tipul de utilizator și poate lua valorile: 0, 1 sau 2; **0 -Admin**;**1-Angajat**,**2-Client** .Se verifica conditia varsteisa fie mai mare sau egala decat optsprezece ani .Are cheia primară **UuserId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS Uuser(
    UuserId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
```

```
username VARCHAR(20) NOT NULL,
CNP BIGINT(13) NULL,
firstName VARCHAR(20) NOT NULL,
lastName VARCHAR(20) NOT NULL,
age INT,
CHECK (AGE>=18),
phoneNr  BIGINT(12) NOT NULL,
email VARCHAR(20) ,
Adress VARCHAR(30) NOT NULL,
city VARCHAR(20) NOT NULL,
country VARCHAR(20) NOT NULL,
tip INT NOT NULL,
CHECK(TIP IN(0,1,2)) ,
password VARCHAR(20) NOT NULL,
CHECK(CHAR_LENGTH(password)>=8) ,
created TIMESTAMP NOT NULL DEFAULT CURRENT_ TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP

);
```

Modelul de inserare a datelor in tabela *Uuser*:

```
INSERT INTO `Uuser` (`username`,`CNP`, `firstName`,  `lastName`, `age`, `phoneNr`,
`email`, `Adress`, `city`, `country`,`tip`,`password`, `created`)VALUES (' alinar1', '
2890703290034', ' Roman', 'Alina', '30, ' 0743567234', ' alliinaa@yahoo.com, ' Piatra-Neamt',
' Piatra-Neamt',  ' Romania',2, 'PPPPP888', '2018-05-25 06:17:14' );
```

9.Bagaje

Tabela este identificată cu un ID unic și are următoarele atribute: userID care deține bagajul respectiv codul bagajului si greutatea. Are cheia primară **bagajeId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS bagaj (
bagajeId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
UuserId INT NOT NULL,
Cod VARCHAR(6) NULL,
`greutate(kg)` INT(6) NULL

);
```

Modelul de inserare a datelor in tabela *bagaj*:

```
INSERT INTO `bagaj`( `UuserId`,`cod`, `greutate(kg)`) VALUES
(1, '302fv1',’5’), (2, '2456gh',’10’);
```

10.Rezervari

Tabela este identificată cu un ID unic și are următoarele atribute: userul, data decolării ,data aterizarii ,orarul din care se poate planifica si pretul cursei. Are cheia primară **ReservationId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS Reservation (
        ReservationId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        UuserId  INT NOT NULL,
        departure_date  DATE NOT NULL,
        arrival_date  DATE NOT NULL,
        scheduleId INT NOT NULL,
        pretCursa INT NOT NULL
);
```

Modelul de inserare a datelor in tabela *Reservation*:

```
INSERT INTO`Reservation`(`UuserId`,`departure_date`,`arrival_date`,`scheduleId`,
`pretCursa`) VALUES (1, '2019-08-12',’ 2019-08-12’, 1, 0);
```

11. Aeroport

Tabela este identificată cu un ID unic și are următoarele atribute:denumirea aeroportului referinta si locatia. Are cheia primară **AirportId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
INSERT INTO`Airport`(`airportId`,`nume`,`locatie`) VALUES (1,’Aeroportul International
Avram Iancu',’ Traian Vuia,nr 149,400397 Cluj-Napoca,Romania’);
```

12.Cont bancar

Tabel *contbancar* conține un ID unic și are următoarele atribute:user id. Are cheia primară **contId** .

```
CREATE TABLE IF NOT EXISTS contbancar(
```

```
    contId VARCHAR(24) PRIMARY KEY,
    UuserId INT NOT NULL
);
```

Modelul de inserare a datelor in tabela *contbancar*:

```
INSERT INTO `contbancar`( `contId`,`UuserId``) VALUES
('RO02INGB0001008214498950',1),
('RO28BRDE450SV21000004500',2),
('RO30BTRLRONCRT0V05650502',3);
```

13.Payment

Tabela este identificată cu un ID unic și are următoarele atribute:statusul platii. Are cheia primară **paymentId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS payment (
    paymentId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    pay_status BOOLEAN NOT NULL
);
```

Modelul de inserare a datelor in tabela *payment*:

```
INSERT INTO`payment`( `paymentId`,`pay_status`)
VALUES (1,true),(2,true),(3,true),(4,false);
```

14.Airport Services

Tabela este identificată cu un ID unic și are următoarele atribute:telefon,parkingId,taxiId,airportId,RentId. Are cheia primară **AirportServicesId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS AirportServices (
    AirportServicesId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    telefon INT(10) NOT NULL,
    parkingId INT NOT NULL,
    taxiId INT NOT NULL,
    airportId INT NOT NULL,
    RentId INT NOT NULL
);
```

15.Flight cost

Tabela este identificată cu un ID unic și are următoarele atribute:legId. Are cheia primară **costId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS FlightCost(
        costId  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        legId INT NOT NULL
);
```

Modelul de inserare a datelor in tabela *FlightCost*:

```
INSERT INTO`flightcost`
VALUES (1,1),(2,2),(3,3),(4,4);
```

16.Dest_comp_avion

Tabela este identificată cu un ID unic și are următoarele atribute:id-ul companiei aeriene,id avion,id-ul destinatiilor si un id catre orar. Are cheia primară **legId**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS Dest_comp_Avion(
        legId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        companieAerianaId  INT NOT NULL,
        idAvion INT NOT NULL,
       destinatiiId INT NOT NULL,
       scheduleId INT NOT NULL


);
```

Modelul de inserare a datelor in tabela *Dest_comp_Avion*:

```
INSERT INTO`Dest_comp_Avion` VALUES (1,1,1,1,1),(2,1,2,2,2),(3,1,3,3,3),(4,1,4,4,1),
(5,2,5,5,2);
```

17.Reservation_payment

Tabela este identificată cu un ID unic și are următoarele atribute:id-ul platii,id-ul rezervarii. Are cheia primară **reservationPayment**. Instrucțiunea de creare a tabelului este prezentata mai jos:

```
CREATE TABLE IF NOT EXISTS reservation_payment(
        reservationPaymentId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        paymentId  INT NOT NULL,
        idAvion INT NOT NULL,
        ReservationId INT NOT NULL


);
```

Modelul de inserare a datelor in tabela *reservation_payment*:

```
INSERT INTO`reservation_payment` VALUES (1,1,2),(2,2,4),(3,3,6);
```

3.3.1 Interogări

```
# 1. Afisati informatiilor despre inchiriere taxi

SELECT nume_companie,telefon
 FROM taxi_cab;


#2. Afisarea  informatiilor inchirierii unei masini in functie de numele companiei ordonata alfabetic

SELECT  nume_companie,telefon
 FROM rent_a_car
 ORDER BY nume_companie;


#3. Afisarea destinatiilor care incep cu litera B
SELECT denumire
 FROM Destinatie
WHERE denumire LIKE 'B%';


#4.Afisarea informatiilor despre aeroport
SELECT nume,locatie
FROM airport;
```

#Afisarea companiilor aeriene in ordine alfabetica

```
SELECT nume,telefon,adresa
FROM companieAeriana
GROUP BY nume;
```

#6.

```
SELECT dca.legId,d.denumire,c.nume,av.nume,av.model,s.decolareActual,s.sosireDestinatie
,s.decolareDestinatie,s.sosireActual
FROM airport_schedule s , dest_comp_avion dca,destinatie d,avion av,companieAeriana c
WHERE (s.scheduleId=dca.scheduleId)AND
  (dca.destinatiiId=d.destinatiiId)AND
  (av.idAvion=dca.idAvion) AND(c.companieAerianaId=dca.companieAerianaId)
ORDER BY dca.legId;
```

#7. Afișarea aeronavelor cu o capacitate mai mare de 100 de locuri

```
SELECT av.nume,av.model,av.capacitate
FROM avion av
GROUP BY av.idAvion
HAVING (av.capacitate >= 100);
```

#8. Afișarea user de tip admin in ordinea alfabetica

```
SELECT u.firstName,u.lastName,u.email,u.username
FROM Uuser u
WHERE u.tip= 1
ORDER  BY  firstName;
```

3.3 Triggere

Trigerrele sunt o clasă specială de proceduri stocate, asociate unei tabele, definite pentru a fi lansate în execuție automat la inițierea unei operații de tip UPDATE, INSERT sau DELETE asupra tabelei în cauză.Trigerrele sunt un instrument puternic pentru implementarea a ceea ce în aplicațiile de baze de date poartă numele de business rules.

#1

```
DELIMITER //
drop trigger if exists delete_companie;
CREATE TRIGGER delete_companie BEFORE DELETE ON companieAeriana
FOR EACH ROW BEGIN
  DELETE FROM dest_comp_avion WHERE
companieAerianaId=OLD.companieAerianaId;

END //
DELIMITER ;

DELETE FROM companieAeriana WHERE wordCount(nume) = 2 ;
Select * from dest_comp_avion;
```

3.5 Proceduri

O procedură stocată este o mulţime ordonată de instrucţiuni SQL stocată permanent pe server şi compilată la utilizare. Procedurile stocate reprezintă o modalitate de a crea rutine şi proceduri care să fie rulate pe server de către procesele serverului.

#1 Procedura pentru adaugare user

```
DROP PROCEDURE IF EXISTS addUser;
DELIMITER //
CREATE PROCEDURE  addUser (_username varchar(20),_firstName
VARCHAR(20),_lastName VARCHAR(20), _age int,_CNP bigint(13), _Adress
VARCHAR(30),
                    _phoneNr  bigint(12),_Email VARCHAR(20),_city
VARCHAR(20),_country varchar(20) ,_password varchar(20),_tip INT)
```

```
        BEGIN

        INSERT INTO
Uuser(username,CNP,firstName,lastName,age,phoneNr,email,Adress,city,country,tip,passwo
rd)
        VALUES
(_username,_CNP,_firstName,_lastName,_age,_phoneNr,_Email,_Adress,_city,_country,_tip
,_password);

        END //
        DELIMITER ;
#2 Procedura pentru calculare pret parcare

drop procedure if exists parkingPrice;
 DELIMITER //
 create procedure parkingPrice(price_per_day int,_from_date date ,_to_date date,_from_hour
int ,_to_hour int,price_per_hour int)
 begin
    SET @price_per_day = 20, @from_date = NULL, @to_date = NULL, @from_hour =
NULL,@to_hour=NULL,@price_per_hour=3;
                set @total_price:=0;
       #select @from_date:=from_date from parking where _from_date=from_date;
       #select @to_date:=to_date from parking where _to_date=to_date;
       #select @from_hour:=from_hour from parking where _from_hour=from_hour;
       #select @to_hour:=to_hour from parking where _to_hour=to_hour;

       set @date_diff=datediff(_to_date,_from_date);
       set @hour_diff=_to_hour-_from_hour;

      select SUM(@price_per_day*@date_diff+@price_per_hour*@hour_diff) as sum;

 end//
DELIMITER ;


#3 Procedura pentru rezervare

drop procedure if exists makeReservation;
DELIMITER //
create procedure makeReservation(_from_date date,_to_date date,_pretCursa int,lastName
varchar(20),firstName varchar(20),CNP varchar(20))
begin
```

```
insert into  Reservation values
(findUuser(firstName,lastName,CNP),_from_date,_to_date,findSchedule(_from_date,_to_dat
e),_pretCursa);
end //
DELIMITER;
```

#4 Procedura pentru calculare pret bilet in functie de data si companie

```
drop procedure if exists flightCostPrice;
 DELIMITER //
 create procedure flightCostPrice(_from_date date,_to_date date,_companie
varchar(20),_lastName varchar(20),_firstName varchar(20),_CNP varchar(20))
 begin
SET  @from_date = NULL, @to_date=NULL, @actual_date=curdate()
,@pretCompanie=NULL ,@pretCursa=NULL;
select @from_date:=date(decolareActual) from Airport_Schedule where
_from_date=Date(decolareActual);
select @to_date:=date(sosireDestinatie) from Airport_Schedule where
_to_date=date(sosireDestinatie);
select @pretCompanie:=pretStandard from companieAeriana where _companie=nume;
set @date_diff=datediff(@from_date,@actual_date);
   if @date_diff<10 then
           set @pretCursa=@pretCompanie*5.14;
    elseif @date_diff <20 then
     set @pretCursa=@pretCompanie*3.14;
          else
     set @pretCursa=@pretCompanie*1.14;
   end if ;
select @pretCursa ;
call makeReservation(@from_date,@to_date,@pretCursa,_lastName,_firstName,_CNP);
  end//
DELIMITER ;
```

#5 Procedura pentru setare plata

```
drop procedure if exists setPay;
DELIMITER //
create procedure setPay(_contId varchar(24),uuserId int )
begin
set @cont=NULL;
select @pay_status:=pay_status from Payment;
```

```
select @cont:=contId from Uuser u join contbancar c on c.UuserId=u.UuserId;
if @cont is not null then
        set @pay_status=true;
   update Payment set pay_status=@pay_status where
   (select * from Uuser u join contbancar c on u.UuserId=c.UuserId and u.UuserId=uuserId
and c.contId=_contId join Reservation r on r.UuserId=u.UuserId
   join Reservation_Payment rp on rp.ReservationId=r.ReservationId join Payment p on
p.paymentId=rp.paymentId
   );
   else set @pay_status=false;
   update Payment set pay_status=@pay_status where
   (select * from Uuser u join contbancar c on u.UuserId=c.UuserId and u.UuserId=uuserId
and c.contId=_contId join Reservation r on r.UuserId=u.UuserId
   join Reservation_Payment rp on rp.ReservationId=r.ReservationId join Payment p on
p.paymentId=rp.paymentId
   );
end if;
end //

DELIMITER ;
```

3.6 Vederi

O vedere este o tabelă virtuală al cărei conținut este definit printr-o interogare. La fel ca orice tabelă reală, o vedere constă dintr-un set de atribute și se materializează printr-un set de tuple.

```
# 1
CREATE VIEW View_avioane AS
SELECT av.nume,av.model,av.capacitate
FROM avion av
ORDER BY av.nume;


# 2
CREATE VIEW View_companieAeriana AS
SELECT c.companieAerianaId,c.nume,d.denumire
FROMcompanieAeriana c
JOIN Dest_comp_Avion dca
ON c.companieAerianaId=dca.companieAerianaId
JOIN Destinatie d ON
d.destinatiiId=dca.destinatiiId
ORDER BY c.nume;
```

```
# 3
CREATE VIEW View_Destinatii AS
SELECT d.destinatiiId,d.denumire
FROM destinatie d
ORDER BY d.denumire;

# 4
CREATE VIEW View_Uuser AS
SELECT u.username,u.firstName,u.lastName,u.tip
FROM Uuser u
ORDER BY u.tip,u.lastName;

#5
CREATE VIEW View_schedule AS
SELECT sch.scheduleId,sch.decolareActual,sch.sosireActual
FROM Airport_Schedule sch
ORDER BY sch.decolareActual,sch.sosireActual;

#6
CREATE VIEW View_Rezervari as
select r.ReservationId ,u.UuserId,u.firstName,u.lastName ,r.departure_date
,r.arrival_date,r.pretCursa from
Uuser u join Reservation r
on r.UuserId=u.UuserId
order by u.firstName;
```

3.7 Functii

```
# Functie pentru calcularea numarului de cuvinte

drop function if exists wordCount;
DELIMITER //
 CREATE FUNCTION wordCount(str TEXT)
          RETURNS INT
          DETERMINISTIC
          SQL SECURITY INVOKER
          NO SQL
      BEGIN
        DECLARE wordCnt, idx, maxIdx INT DEFAULT 0;
        DECLARE currChar, prevChar BOOL DEFAULT 0;
        SET maxIdx=char_length(str);
        WHILE idx < maxIdx DO
```

```
        SET currChar=SUBSTRING(str, idx, 1) RLIKE '[[:alnum:]]';
        IF NOT prevChar AND currChar THEN
           SET wordCnt=wordCnt+1;
        END IF;
        SET prevChar=currChar;
        SET idx=idx+1;
      END WHILE;
      RETURN wordCnt;
    END//
DELIMITER ;
```

#2

```
drop function if exists findUuser;
DELIMITER //
 CREATE FUNCTION findUuser(_firstName TEXT,_lastName TEXT,_CNP TEXT)
        RETURNS INT
        DETERMINISTIC
    BEGIN
      DECLARE id, idx, maxIdx INT DEFAULT 0;
      select max(UuserId) into maxIdx from Uuser;
      WHILE idx < maxIdx DO
      select UuserId into id from Uuser where firstName=_firstName and CNP=_CNP and
lastName=_lastName;
      if id>0 then
      set idx=maxIdx+1;
      else
         SET idx=idx+1;
              end if;
      END WHILE;
      RETURN id;
    END//
DELIMITER ;
```

#3

```
drop function if exists findSchedule;
DELIMITER //
 CREATE FUNCTION findSchedule(_from_date date,_to_date date)
        RETURNS INT
        DETERMINISTIC
    BEGIN
      DECLARE id, idx, maxIdx INT DEFAULT 0;
```

```
        select max(scheduleId) into maxIdx from Airport_schedule;
       WHILE idx < maxIdx DO
       select scheduleId into id from Airport_schedule where
date(decolare_actual)=_from_date and date(decolareDestinatie)=_to_date;
       if id>0 then
       set idx=maxIdx+1;
       else
          SET idx=idx+1;
                end if;
       END WHILE;
       RETURN id;
     END//
DELIMITER ;
```

3.8 Forma de normalizare

Normalizarea este descompunerea relațiilor astfel încât acestea să ajungă într-o formă relațională corectă, fără pierdere de informație și cu evitarea redundanței. Normalizarea este utilă pentru aplicații cu baze de date ce presupun operații frecvente de actualizare, ștergere, adăugare și mai puțin în sisteme ce presupun interogări complexe.

Ideea care stă la baza criteriilor de proiectare a unei baze de date relaționale este de dependență a datelor. Se referă la faptul că între atributele unei relații sau între atributele din relații diferite pot exista anumite conexiuni logice, care influentează proprietățile schemelor de relație în raport cu operațiile: adăugare, ștergere, actualizare.

Formele normale reprezintă criterii de ghidare a proiectantului bazei de date în ceea ce privește alegerea schemelor de relație, și se aplică în scopul evitării anomaliilor de ștergere, adăugare, actualizare dar și de inconsistența datelor atunci când aceste operații se realizează frecvent.

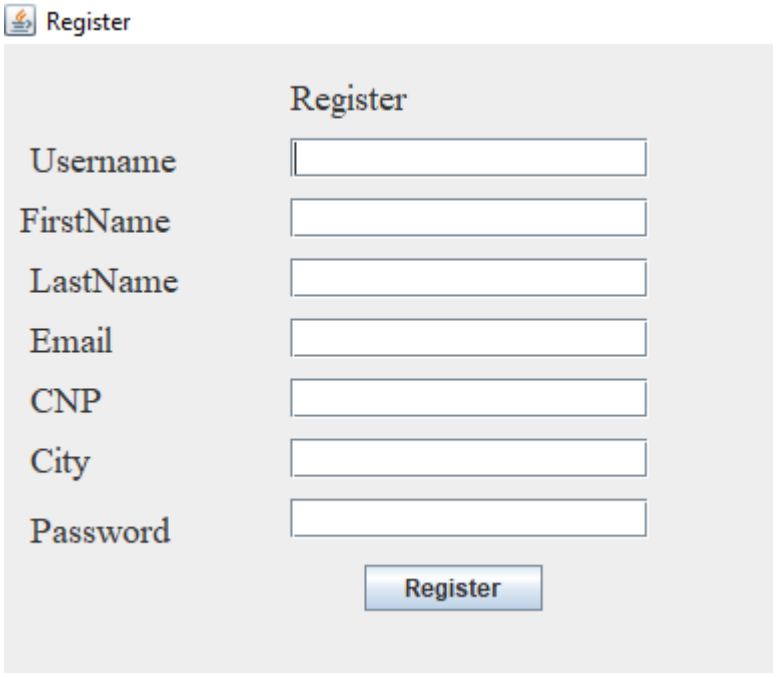
Baza de date respectă **NFBC (norma formală Boyce-Codd)**. Atributele fiecărui tabel nu depinde de alte atribute. Fiecare tabel are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date.

4. Detalii de implementare

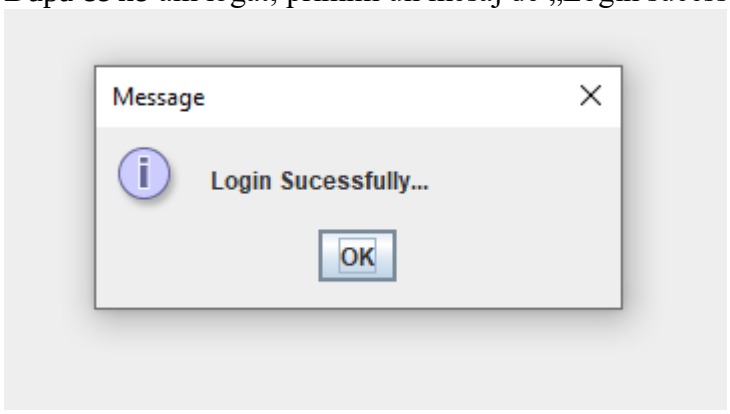
4.1 Java

Pentru a implementa aplicatia Java ne-am folosit de Java swing si Java window builder. Swing este un subset JFC (Java Foundation Classes) şi constă dintr-o serie de componente vizuale care extind (îmbunătăţesc) componentele AWT, şi furnizează noi facilităţi precum tabele şi arbori. Structura de clase din Swing este asemănătoare cu cea din AWT, în sensul că toate componentele interfeţei grafice sunt derivate dintr-un singur părinte numit JComponent (care este derivat din clasa AWT Container). Pachetul de clase Swing reprezintă soluţia furnizată de Sun pentru crearea unor interfeţe utilizator grafice complet portabile pe orice platformă. În Swing, toate numele claselor încep cu litera J, şi atunci când este posibil, numele este acelaşi cu cel al clasei AWT pe care o înlocuieşte. La fel ca la AWT, punctul de plecare pentru un program bazat pe Swing, este clasa JFrame sau clasa JApplet.
Aplicatia noastra are ca si punct de start un formular de login care are urmatoarele butoane: Login, ResetInfo, Register, Exit.

ResetInfo reseteaza informatiile adaugate in cele doua text-box-uri, butonul de Register ne redirectioneaza pe un nou JFrame unde putem sa ne inregistram ca si utilizator nou.



Dupa ce ne-am logat, primim un mesaj de „Login sucessfully”.



O parte din codul folosit pentru a realiza acest formular il gasiti mai jos :

```
package com.airline.backend1;

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

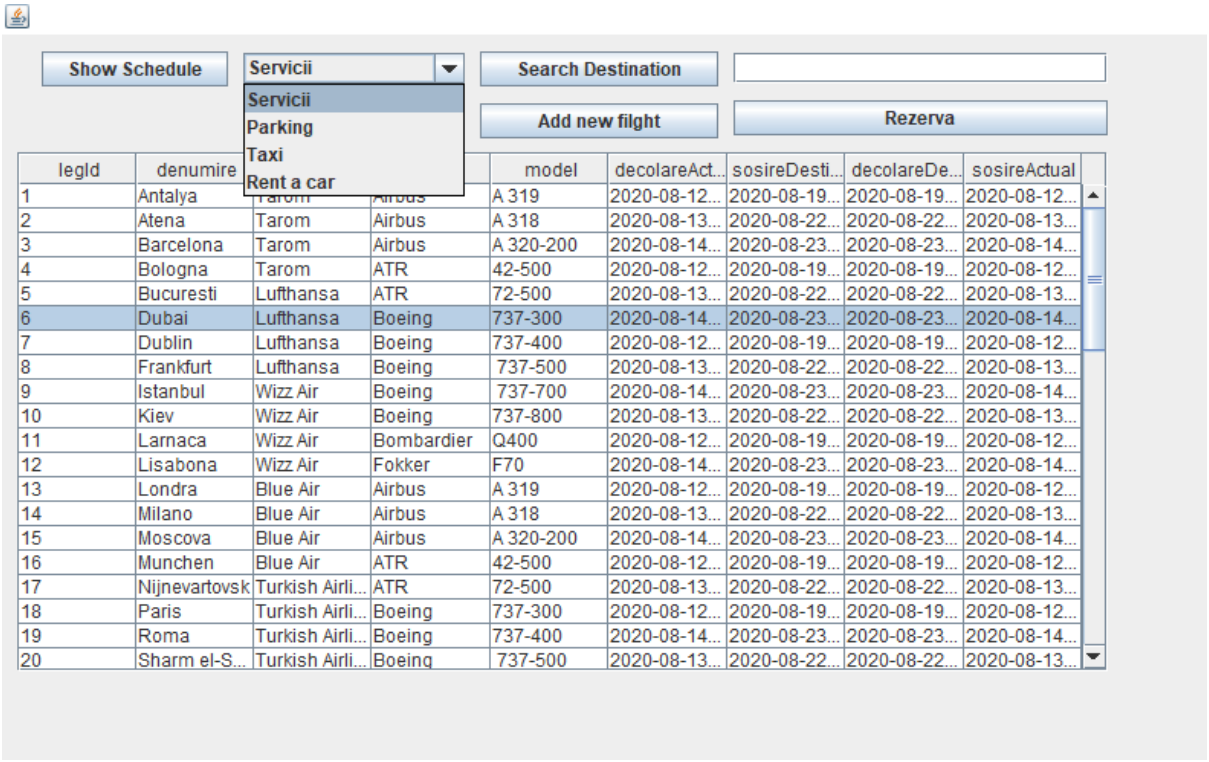
```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Login {

public JFrame Login;
private JTextField usernameField;
private JPasswordField passwordField;

public Login() {
                                initializeLogin();
    }

private void initializeLogin() {
                        Login = new JFrame("Login");
                    Login.setBounds(100, 100, 443, 225);
        Login.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                    Login.getContentPane().setLayout(null);
                    Login.setBackground(Color.BLUE);

                JLabel lblLogin = new JLabel("LOGIN");
            lblLogin.setFont(new Font("Serif", Font.PLAIN, 18));
                lblLogin.setBounds(194, 11, 84, 36);
                lblLogin.setBackground(Color.BLUE);
                Login.getContentPane().add(lblLogin);

            JLabel lblUsername = new JLabel("Username");
        lblUsername.setFont(new Font("Serif", Font.PLAIN, 18));
            lblUsername.setBounds(36, 55, 120, 30);
            lblUsername.setBackground(Color.BLUE);
            Login.getContentPane().add(lblUsername);

            JLabel lblPassword = new JLabel("Password");
        lblPassword.setFont(new Font("Serif", Font.PLAIN, 18));
            lblPassword.setBounds(36, 97, 95, 22);
            lblPassword.setBackground(Color.BLUE);
            Login.getContentPane().add(lblPassword);

                usernameField = new JTextField();
```

```
usernameField.setBounds(141, 58, 250, 30);
Login.getContentPane().add(usernameField);
Login.setBackground(Color.BLUE);
usernameField.setColumns(10);

passwordField = new JPasswordField();
passwordField.setBounds(141, 96, 250, 30);
Login.setBackground(Color.BLUE);
Login.getContentPane().add(passwordField);

JButton btnLogin = new JButton("Login");
btnLogin.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0)
{
try {

Class.forName("com.mysql.cj.jdbc.Driver");
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinereservation","root"
,"root");
Statement stmt =
con.createStatement();
String sql = "Select * from Uuser
where username='"+usernameField.getText() + "' and
password='"+passwordField.getText()+"';";
ResultSet rs
=stmt.executeQuery(sql);

if(rs.next()) {

JOptionPane.showMessageDialog(null,"Login Sucessfully...");
//ShowSchedule regFace
=new ShowSchedule();
//regFace.setVisible(true);

ShowSchedule.ShowSchedulerInit();
//regFace.dispose();
}
else{

JOptionPane.showMessageDialog(null,"Incorrect info...");
}
con.close();
}catch(Exception e) {
```

```
                                        }
                                    }
                                });
                        btnLogin.setBounds(25, 136, 89, 23);
                        Login.getContentPane().add(btnLogin);

                    JButton btnResetinfo = new JButton("ResetInfo");
                btnResetinfo.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent arg0)
        {
                                            usernameField.setText(null);
                                            passwordField.setText(null);
                                        }
                                    });
                        btnResetinfo.setBounds(125, 136, 89, 23);
                        Login.getContentPane().add(btnResetinfo);

                        JButton btnExit = new JButton("Exit");
                    btnExit.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent e) {
                                            System.exit(0);
                                        }
                                    });
                        btnExit.setBounds(317, 137, 89, 23);
                        Login.getContentPane().add(btnExit);

                    JButton btnRegister = new JButton("Register");
                btnRegister.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent arg0)
        {
                                            Register reg =new Register();
                                            reg.RegisterInit();
                                        }
                                    });
                        btnRegister.setBounds(224, 137, 84, 22);
                        Login.getContentPane().add(btnRegister);
            }


}
```

In final, se deschide pagina principala, de unde de pe butonul „Show Schedule” putem vizualiza orarul.Dupa ce selectam o cursa, vom apasa pe butonul „Rezerva”, de unde vom

introduce datele de rezervare si le vom salva.Pagina principala are de asemenea un scrolling panel de unde putem vizualiza serviciile oferite de aeroport.Pe butonul de Parking se va deschide o fereastra noua de unde vom putea face o rezervare pentru parcare, iar pe restul paginilor se vor afisa detaliile legate de Taxi sau inchirieri masini.De asemenea,se vor putea gasi mai usor cursele in functie de destinatia dorita.Utilizatorii logati ca si admini sau angajati,vor putea adauga noi useri sau curse noi de pe cele doua butoane corespunzatoare.



legId	denumire			model	decolareAct...	sosireDesti...	decolareDe...	sosireActual
1	Antalya			A 319	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
2	Atena	Tarom	Airbus	A 318	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
3	Barcelona	Tarom	Airbus	A 320-200	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
4	Bologna	Tarom	ATR	42-500	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
5	Bucuresti	Lufthansa	ATR	72-500	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
6	Dubai	Lufthansa	Boeing	737-300	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
7	Dublin	Lufthansa	Boeing	737-400	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
8	Frankfurt	Lufthansa	Boeing	737-500	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
9	Istanbul	Wizz Air	Boeing	737-700	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
10	Kiev	Wizz Air	Boeing	737-800	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
11	Larnaca	Wizz Air	Bombardier	Q400	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
12	Lisabona	Wizz Air	Fokker	F70	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
13	Londra	Blue Air	Airbus	A 319	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
14	Milano	Blue Air	Airbus	A 318	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
15	Moscova	Blue Air	Airbus	A 320-200	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
16	Munchen	Blue Air	ATR	42-500	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
17	Nijnevartovsk	Turkish Airli...	ATR	72-500	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...
18	Paris	Turkish Airli...	Boeing	737-300	2020-08-12...	2020-08-19...	2020-08-19...	2020-08-12...
19	Roma	Turkish Airli...	Boeing	737-400	2020-08-14...	2020-08-23...	2020-08-23...	2020-08-14...
20	Sharm el-S...	Turkish Airli...	Boeing	737-500	2020-08-13...	2020-08-22...	2020-08-22...	2020-08-13...

O parte din codul utilizat pentru a realiza aceasta fereasta este reprezentat mai jos:

```
package com.airline.backend1;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
```

```
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;

import net.proteanit.sql.DbUtils;

public class mainWindow {

private JFrame frame;
private JTable table;
private JTextField searchTextField;

public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() {
                                        public void run() {
                                                try {
                                                        mainWindow window = new
mainWindow();
                                                        window.frame.setVisible(true);
                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }
                                        }
                                    });
    }

public mainWindow() {
                                initialize();
    }

private void initialize() {
                                frame = new JFrame();
                        frame.setBounds(100, 100, 750, 467);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        frame.getContentPane().setLayout(null);

                    JScrollPane scrollPane = new JScrollPane();
                      scrollPane.setBounds(10, 77, 714, 340);
                      frame.getContentPane().add(scrollPane);

                                table = new JTable();
                            scrollPane.setViewportView(table);

    JButton showScheduleButton = new JButton("Show Schedule");
```

```
    showScheduleButton.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent arg0)
{
                                                try {

Class.forName("com.mysql.cj.jdbc.Driver");
                                                        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinereservation",
"root", "root");
                                                        @SuppressWarnings("unused")
                                                        Statement stmt =
con.createStatement();

                                                                String select  = "SELECT
dca.legId,d.denumire,c.nume,av.nume,av.model,s.decolareActual,s.sosireDestinatie,s.
decolareDestinatie,s.sosireActual\r\n" +
                                                                        "FROM
airport_schedule s , dest_comp_avion dca,destinatie d,avion av,companieAeriana c
where (s.scheduleId=dca.scheduleId) and\r\n" +
                                                                        "
(dca.destinatiiId=d.destinatiiId) and\r\n" +
                                                                        "
(av.idAvion=dca.idAvion) and  (c.companieAerianaId=dca.companieAerianaId)\r\n"
+
                                                                        "order by
dca.legId;" ;
                                                                //PreparedStatement pstmt
= con.prepareStatement(select);
                                                                ResultSet rs =
stmt.executeQuery(select);

table.setModel(DbUtils.resultSetToTableModel(rs));

                                                                con.close();

                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }



                                        }
                                });
                showScheduleButton.setBounds(26, 11, 122, 23);
```

```
frame.getContentPane().add(showScheduleButton);



JComboBox serviciiComboBox = new JComboBox();
serviciiComboBox.setModel(new DefaultComboBoxModel(new String[]
{"Servicii", "Parking", "Taxi", "Rent a car"}));
serviciiComboBox.setMaximumRowCount(4);
serviciiComboBox.setBounds(158, 12, 145, 20);
frame.getContentPane().add(serviciiComboBox);

searchTextField = new JTextField();
searchTextField.setBounds(479, 12, 245, 20);
frame.getContentPane().add(searchTextField);
searchTextField.setColumns(10);

JButton searchDestinationButton = new JButton("Search Destination");
searchDestinationButton.setBounds(313, 11, 156, 23);
frame.getContentPane().add(searchDestinationButton);

JButton addNewUserButton = new JButton("Add new user");
addNewUserButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0)
{
}
});
addNewUserButton.setBounds(158, 43, 145, 23);
frame.getContentPane().add(addNewUserButton);

JButton addNewFlightButton = new JButton("Add new filght");
addNewFlightButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
}
});
addNewFlightButton.setBounds(313, 45, 156, 23);
frame.getContentPane().add(addNewFlightButton);

JButton rezervaButton = new JButton("Rezerva");
rezervaButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
try {


Class.forName("com.mysql.cj.jdbc.Driver");
```

```
                                                    Connection con2 =
DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinereservation",
"root", "root");
                                                    Statement pstmt =
con2.createStatement();
                                                    String select2 ="select
firstName,lastName,CNP from Uuser";

                                                    ResultSet rs2 =
pstmt.executeQuery(select2);
                                                    String firstName =
rs2.getString("firstName");
                                                    String lastName =
rs2.getString("lastName");
                                                    String CNP =
rs2.getString("CNP") ;
                                                    con2.close();



Class.forName("com.mysql.cj.jdbc.Driver");
                                                    Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinereservation",
"root", "root");

                                                    String decolareActual =
table.getValueAt(table.getSelectedRow(), 5).toString();
                                                    String sosireDestiantie =
table.getValueAt(table.getSelectedRow(), 6).toString();
                                                    String numeCompanie =
table.getValueAt(table.getSelectedRow(), 2).toString();


System.out.println(decolareActual);

System.out.println(sosireDestiantie);

                                                    Statement stmt =
con.createStatement();
                                                    String select  = "call procedure ("
+ decolareActual + "," + sosireDestiantie + "," + numeCompanie + "," + lastName +
"," + firstName + "," + CNP +")";
```

```
                                                    ResultSet rs =
        stmt.executeQuery(select);

        table.setModel(DbUtils.resultSetToTableModel(rs));

                                                    con.close();

                                                    } catch (Exception p) {
                                                            p.printStackTrace();
                                                    }

                                                }
                                            });
                                rezervaButton.setBounds(479, 43, 245, 23);
                                frame.getContentPane().add(rezervaButton);
                        }
}
```

4.2 Elemente de securitate

Persoanele care au acces la baza de date sunt de 3 tipuri: 0-admin, 1-angajat si 2-client. Orice persoană care doreşte să se conecteze (log in) la o bază de date trebuie să deţină un cont (account, user) şi o parolă (password). Sistemul de gestiune verifică contul şi parola şi autentifică acel utilizator, dacă acestea sunt corecte. Programele de aplicaţii sunt considerate de asemenea utilizatori şi se conectează pe un anumit cont şi trebuie să furnizeze parola acestuia.

Prin protecţia şi securitatea datelor se înţelege totalitatea mijloacelor, metodelor şi a mecanismelor destinate prevenirii distrugerii, modificării sau folosirii neautorizate a informaţiei protejate. Referitor la protecţia şi securitatea datelor, în literatura de specialitate se definesc următoarele concepte de bază:

• Securitatea datelor – totalitatea măsurilor de protecţie împotriva distrugerii accidentale sau intenţionate, a modificării neautorizate sau a divulgării acestora
• Caracterul secret – este un concept ce se aplică la un individ sau organizaţie şi constă în dreptul acestora de a decide ce informaţii se pot folosi în comun şi în ce condiţii
• Confidenţialitatea – se aplică la date şi se referă la statutul acordat, acesta reprezentând nivelul sau gradul de protecţie ce trebuie acordat informaţiei respective
• Integritatea – se referă la restricţia ca sensul datelor să nu difere faţă de cel înscris pe documentul sursă, impunând totodată ca datele să nu fie alterate accidental sau voit.

Comanda GRANT este capabilă să aplice o mare varietate de privilegii, de la abilitatea de a CREA tabele și baze de date, de a citi sau de a scrie fișiere, chiar de a închide serverul, astfel un utilizator logat ca si admin va avea aceste privilegii.

5. Concluzii și dezvoltări ulterioare

În ideea dezvoltării ulterioare a proiectului, suntem de părere că ar fi benefică cearea unei pagini Web, ceea ce ar face mult mai usor modul de a interactiona cu procesul de cumparare si rezervare de bilete, cautarea curselor si folosirea serviciilor oferite de aeroport. Pagina va avea un proces de logare sau inregistrare pe site, o pagina principala, o pagina de unde se vor putea vizualiza informatii despre aeroportul nostru, o pagina care va contine orarul aeroportului, destinatiile acestuia, companiile care relationeaza cu aeroportul cat si avioanele detinute de catre acesta.Serviciile oferite de catre aeroport vor avea de asemenea o pagina de prezentare.In cadrul acestei pagini Web utilizatorul va putea sa isi faca rezervari, sa le plateasca online sau sa isi vizualizeze cursele.

O alta dezvoltare ulterioara va avea loc asupra aplicatiei in Java, care va fi mult mai complexa si mult mai usor de folosit .In cadrul acestei aplicatii utilizatorul va putea de asemenea sa se logheze si/sau inregistreze si va putea vizualiza cursele oferite de catre aeroport.Acesta va putea de asemenea sa faca rezervari mult mai usor sau sa acceseze serviciile oferite de aeroport.

6. Bibliografie

1. https://www.youtube.com/watch?v=cjxYjwqz39Y&list=PLr-LnFzODWU2IAGh-pIkLBdvZiHo8yJHo&index=15&t=0s
2. https://www.youtube.com/watch?v=-hpX9oEvoXc&list=PLr-LnFzODWU2IAGh-pIkLBdvZiHo8yJHo&index=15
3. https://eikhart.com/blog/mysql-word-count-function
4. https://stackoverflow.com/questions/17112852/get-the-new-record-primary-key-id-from-mysql-insert-query
5. https://www.w3schools.com/
6. https://codinginfinite.com/signup-login-page-php-mysql-database-source-code/
7. http://inf.ucv.ro/documents/tudori/laborator7_41.pdf