MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
CATEDRA CALCULATOARE

Proiect de semestru

la disciplina

Baze de date

AEROPORTUL MANAGEMENT

Studenti:Moldovan Liana
Senila Constantin

An academic: 2020-2021

Grupa:20223

CUPRINS

1.Introducere

Domeniul bazelor de date ocupa astazi un loc foarte important in cadrul informaticii, indiferent data suntem un utilizator obisnuit sau un specialist IT.

Informația trebuie să fie disponibilă în timp util, să fie corectă, necontracdictorie, neredundantă și să aibă o forma adecvata necesităților factorului de decizie. Aceste cerințe sunt realizate prin existența unui volum imens de date care trebuie culese, memorate, organizate, regăsite si prelucrate in mod corespunzător pentru utilizarea lor ca surse informaționale.

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

In prezent aeroportul din Cluj-Napoca este al doilea cel mai cautat la nivel national, fiind cel mai mare din Transilvania si deservind o populatie de aproape 5 milioane de locuitori.

Astfel am decis sa cream o baza de date care sa poata intruni toate datele aeroportului, fara pierderi si sa poata deservi un mare flux de zboruri si memora datele tuturor calatorilor.

De asemenea, pentru a facilita accesul intr-un mod interactiv la aceasta baza de date am implementat o interfata usor accesibila utilizatorului prin care poate cumpara direct un bilet de la aeroport,printr-un simplu click, nu prin intermediul intermediarilor(agentii de turism, site-uri de rezervare a biletelor) care ofera aceleasi servicii dar la un pret mult mai ridicat , sa poata inchiria o masina si sa afle toate informatiile utile despre zboruri, aeroport. De asemenea, exista o sectiune de mesaje prin care utilizatorul sau potentialul client poate discuta cu admin-ul.

De exemplu, orice utilizator al site-ului poate vizualiza pagina de start, orarul curselor pe toata saptamana, poate afla care sunt cele mai in voga destinatii la ora actuala,poate intra in contact cu reprezentatii aeroportului si beneficia de diverse servcii precum inchirierea de masini, rezervarea si cumpararea unui bilet de calatorie. Pentru aceste doua servicii este necesar ca utilizatorul sa isi introduca datele (numele, prenumele, varsta, adresa_mail si numarul de telefon).

De asemenea, reprezentatii aeroportului pot folosi aceeasi platforma pentru a efectua operatiile necesare managementului si functionarii adecvate a aeroportului. Admin-ul isi foloseste credentialele pentru a se putea loga.

Scopul nostru este de a oferi direct pe site-ul aeroportului clientilor informatiile necesare si optiunile de inchiere masina/cumparare bilet intr-un mod user-friendly, administatorului- acces la toate optiunele/operatiile de adaugare/stergere/update, cat si 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.

Această aplicație este dinamică deoarece în dependență de tip de utilizator pot fi accesate total alte meniuri și opțiuni. 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

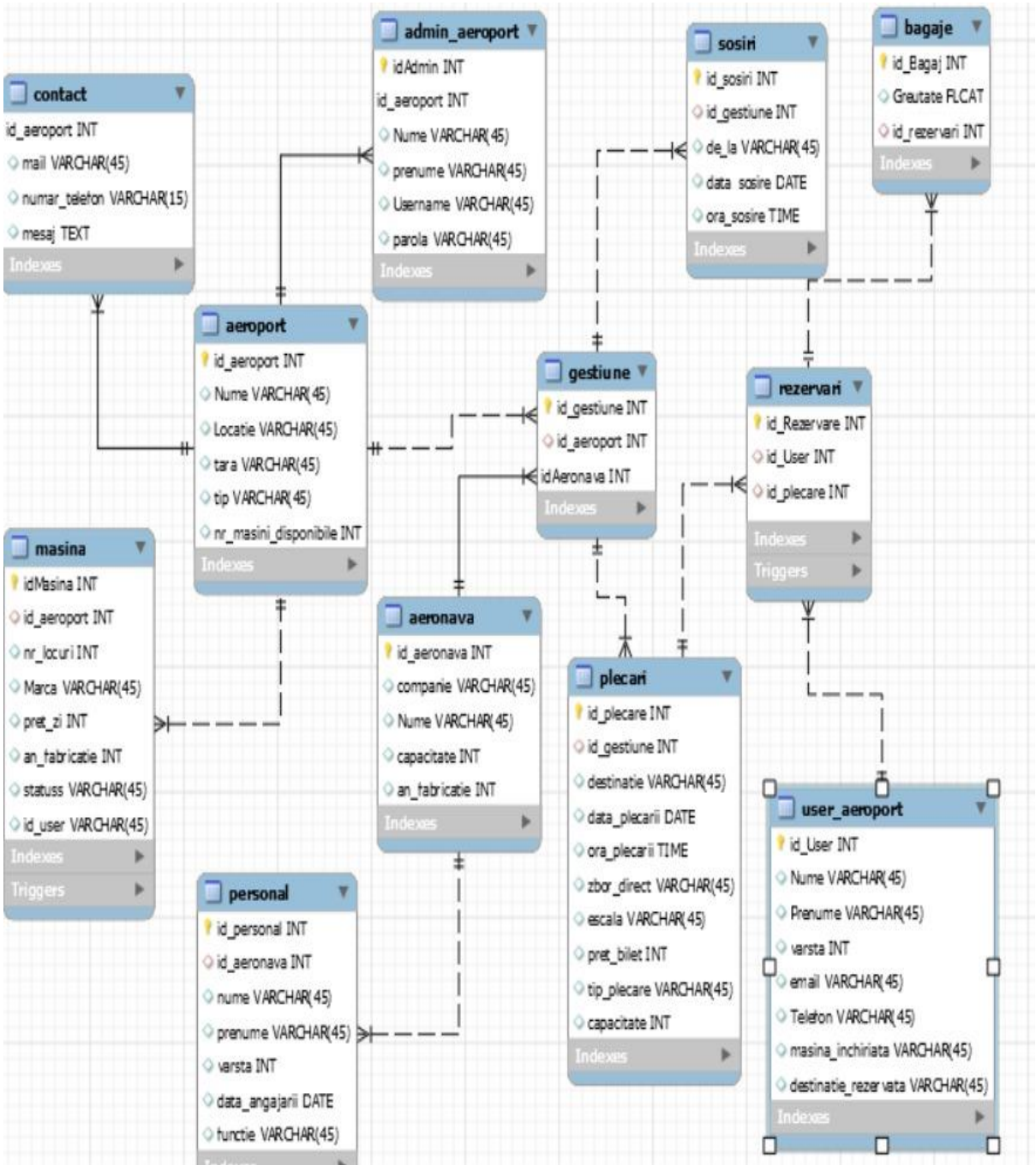
și diverselorinstrucțiuni MySQL. Fiecare tabel are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date.

Caracteristica de profiluri de utilizatori:

- *Anonymous*-utilizatorii care vizualizeaza site-ul nu sunt cunoscuti pana in momentul in care doresc sa cumpere un bilet, sa inchirieze o masina sau sa trimita un mesaj admin-ului.
- *Administrator*-un administrator la optiunea Edit are un set de operatii disponibile: adaugare aeronava, adaugare plecari, adaugare personal, Adaugare masina respectiv stergere/editare pentru fiecare dintre acestea. Pe pagina principala sunt vizibile mesajele primite de la utilizatori si de asemenea isi poate schimba parola si poate actualiza datele de contact ale aeroportului. Optiunea “Pagina user” permite utilizatorului sa vada ce vede user-ul, adica sa observe rapid modificarile pe care le-a facut.

3.Modelul de date

3.1 Diagrama EER



3.2 Entitati si atributele lor

Tabelele:

- **Aeroport**-informatii despre aeroport. Are legatura cu tabelele aeronava(deoarece un aeroport are mai multe aeronave care il tranziteaza), gestiune(gestiunea o folosim pentru a atribui plecarilor si sosirilor cate o aeronava, pentru ca evita anomaliile), masina (un aeroport->mai multe masini), contact (un aeroport poate avea mai multe contacte), admin_aeroport(un aeroport poate avea unul sau mai multi admini);
- **Contact** – informatiile aeroportului care vor fi afisate la sectiunea de contact in interfata si de asemenea se retine mesajul pe care userul il transmite adminului. Legatura cu aeroportul prin cheia straina id_aeroport;

- **Admin_Aeroport**- informatiile adminului si datele pentru logare; Legatura cu aeroportul prin cheia straina id_aeroport;

- **Masina-** informatii legate de masina si numele persoanei care a inchiriat-o. Legatura cu aeroportul prin cheia straina id_aeroport;

- **Aeronava-** informatii despre aeronava. Legatura cu tabela gestiune prin cheia straina id_aeroport.

- **Personal-** informatii despre angajatii aeroportului si despre personalul aeronavelor(piloti si stewardese). Legatura cu tabela aeronava prin cheia straina id_aeronava.

- **Gestiune-** aceasta tabela este utila pentru folosirea unui aeronave pentru mai multe zboruri(de sosire sau plecare). Astfel evitam anomaliile si putem accesa mai usor informatiile din tabele. Legatura cu tabelele plecari si sosiri.

- **Sosiri-**informatii despre avioanele care sosesc in aeroport.

- **Plecari-** informatii despre plecarile din aeroport.

- **Rezervari-**informatii despre bilet: unde zboara user-ul si care este destinatia.

- **Bagaje-** informatii despre bagaj(daca exista). Legatura cu tabela rezervari prin cheia straina id_rezervare.

- **User_aeroport-**retine informatiile clientului(user-ului) si informatiile despre serviciile pe care le-a accesat (cumparare bilet/inchiriere masina).

3.3 Interogari

```
#1.Selecteaza informatii despre masina

Select  idMasina, Marca, pret_zi from masina;

#2.Selecteaza toate informatiile user-ului

Select * from user_aeroport;



#3.Afisati personalul cu vechime mai mare de 2 ani

select personal.nume as Nume, personal.prenume as Prenume,personal.data_angajarii from personal where Year(Current_date)-Year(personal.data_angajarii)>=2 order by personal.data_angajarii;

#4.Afiseaza mesajele trimise de utilizatori adminului.

Select mesaj from contact where id_aeroport=1;

#5. Afiseaza infromatiile unei plecari.

Select plecari.id_plecare, plecari.destinatie, plecari.data_plecarii, plecari.ora_plecarii, plecari.pret_bilet, plecari.capacitate

 from plecari;
```

#6. Afisarea pasagerilor fara bagaj de calatorie;

```
select user_aeroport.nume as Nume,
user_aeroport.prenume as Prenume from user_aeroport
join rezervari on user_aeroport.id_user=rezervari.id_user
join bagaje on rezervari.id_rezervare=bagaje.id_rezervari
and bagaje.greutate is null;
```

#7. Afiseaza infromatiile unei sosiri.

```
Select id_sosiri, de_la,data_sosire, ora_sosire from Sosiri;
```

#8. Afisarea masinilor neinchiriate.

```
Select idMasina, Marca,pret_zi from Masina where
statuss='neinchiriat';
```

#9. Afisarea companiilor care efectueaza zboruri interne

```
select aeronava.companie from aeronava
 join gestiune on
aeronava.id_aeronava=gestiune.idaeronava
 join plecari on plecari.id_gestiune=gestiune.id_gestiune and
plecari.tip_plecare='intern';
```

#10. Afisarea destinatie cu cele mai multe zboruri(plecari)

```
select destinatie, count(destinatie) from plecari group by
plecari.destinatie having count(plecari.destinatie)>1;
```

#11. Afisarea tuturor informatiilor legate de un angajat.

```
Select * from personal;
```

#12. Afisarea personalului cu atributiile fiecaruia.

```
Select personal.nume as Nume,
personal.prenume,personal.functie from personal;
```

#13. Afisarea pilotilor si a zborurilor pe care le piloteaza.

```
Select personal.nume as Nume, personal.prenume as
Prenume, aeronava.Nume as nume_aeronava from personal
join aeronava on
aeronava.id_aeronava=personal.id_aeronava and
personal.functie='pilot';
```

#14. Afisarea masinilor dupa ordonate dupa pretul inchirierii.

```
Select Inchiriaza_masina.marca from Inchiriaza_masina
where Inchiriaza_masina.statuss='neinchiriat' order by
Inchiriaza_masina.pret_zi;
```

#15.Afisarea medie preturilor biletelor de calatorie.

```
Select avg(plecari.pret_bilet) as pret_mediu from plecari;
#Afisarea persoanlul zborului cu destinatia Lisabona;
select personal.nume from personal join aeronava on
aeronava.id_aeronava=personal.id_aeronava join gestiune
on gestiune.idaeronava=aeronava.id_aeronava join plecari
on plecari.id_gestiune=gestiune.id_gestiune and
plecari.destinatie='Lisabona';
```

Interogari in algebra relationala;

1

```
SELECT *
```

FROM personal

WHERE Rol = ‚pilot’;

σ Rol=’pilot’(personal)

2

SELECT *

FROM aeronave

WHERE capacitate = 100;

σ capacitate=100 (aeronave)

3

SELECT *

FROM personal p

WHERE p.functie = ‚stewardesa’

σ Rol=’stewardesa’(staff)

4

select *from contact;

π*σ(contact)

3.4 Triggere

#trigger care la rezervarea unui bilet scade capacitatea aeronave care efectueaza acel zbor cu 1.

```
drop trigger if exists ActualizarePlecari;
delimiter //
create trigger ActualizarePlecari after insert on rezervari
for each row
begin
update plecari set capacitate=capacitate-1 where
new.id_plecare=plecari.id_plecare;
end //
delimiter ;
```

```
# trigger care la adaugarea unei noi masini disponibile creste numarul de masini disponibile de la aeroport cu 1.
drop trigger if exists Nrmasini;
delimiter //
create trigger Nrmasini after insert on masina
for each row
begin
update aeroport set
nr_masini_disponibile=nr_masini_disponibile+1;
end
//
delimiter ;
```

```
# trigger care dupa inchirierea unei masini scade numarul de masini disponibile de la aeroport cu 1.
drop trigger if exists ActualizareNrMasini;
delimiter //
```

```
create trigger ActualizareNrMasini after update on Masina
for each row
begin
update Aeroport set
nr_masini_disponibile=nr_masini_disponibile-1;
end //
delimiter ;
```

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.

```
#Procedura care insereaza un user;
drop procedure if exists insert_user;
delimiter //
create procedure insert_user(Nume varchar(45), Prenume
varchar(45), varsta int,email varchar(45), telefon varchar(45))
begin
insert into user_aeroport(Nume,Prenume, varsta,email,
telefon) values (Nume, Prenume, varsta,email, telefon);
end //
delimiter ;

#Procedura care actualizeaza statusul unei masini in inchiriat
si insereaza numele persoanei care a inchiriat masina;
drop procedure if exists update_masini;
```

```
delimiter //
create procedure update_masini(iddMasina int)
begin
select max(id_user) into @idP from user_aeroport;
select concat(nume,' ',prenume) into @numeP from
user_aeroport where id_user=@idP;
update Masina
set statuss='inchiriat',
id_user=@numeP where idMasina=iddMasina;
end //
 delimiter ;

 #Procedura care actualizeaza tabela user prin adaugarea
masinii pe care acesta a inchiriat-o.
drop procedure if exists update_user;
delimiter //
create procedure update_user(denumire varchar(45))
begin
select max(id_user) into @idU from user_aeroport;
update user_aeroport set masina_inchiriata=denumire where
id_user=@idU;
end //
delimiter ;

#Procedura de stergere a unui user.
drop procedure if exists deleteUser;
delimiter //
create procedure deleteUser()
begin
    Delete from user_aeroport where
user_aeroport.destinatie_rezervata is null and
user_aeroport.masina_inchiriata is null;
end //
```

```
delimiter ;

#Procedura care actualizeaza tabela user prin adaugarea destinatiei pe care a rezervat-o.
drop procedure if exists update_user_bilet;
delimiter //
create procedure update_user_bilet(denumire varchar(45))
begin
    select max(id_user) into @idU from user_aeroport;
    update user_aeroport set destinatie_rezervata=denumire
where id_user=@idU;
end //
delimiter ;



#Procedura care creeaza o noua rezervare a unei calatorii in urma introducerii datelor user-ului.
drop procedure if exists insert_rezervari;
delimiter //
create procedure insert_rezervari(idPlecare varchar(20))
begin
    set @idUser=(select max(user_aeroport.id_User) from
user_aeroport);
    insert into Rezervari(id_User,id_plecare) values
(@idUser,idPlecare);

end //
delimiter ;
#Procedura care schimba parola admin-ului.
drop procedure if exists schimbare_parola;
delimiter //
```

```
create procedure schimbare_parola(parolaveche
varchar(45), poarolaNoua varchar(45))
begin
    set @pV=(select admin_aeroport.parola from
admin_aeroport);
    if @pV = parolaveche then
            update admin_aeroport set parola=poarolaNoua;
    end if;

end//
delimiter ;
```

#Procedura care editeaza numarul de telefon si adresa de mail ale aeroportului (cele vizibile in interfata la sectiunea contact);

```
drop procedure if exists editContact;
delimiter //
create procedure editContact (email varchar(45), telefon
varchar(45))
begin
    if email is not null and email!="" then
            update contact set mail=email;
    end if;
    if telefon is not null and telefon!=""  then
            update contact set numar_telefon=telefon;
    end if;
end//
delimiter ;
```

```
#Procedura care adauga o noua masina.
drop procedure if exists insert_masina;
delimiter //
create procedure insert_masina(locuri varchar(45),MarcaM
varchar(45), pret varchar(45),anFabricatie varchar(45))
begin
    insert into masina(id_aeroport,
nr_locuri,Marca,pret_zi,an_fabricatie, statuss) values (1,
locuri, MarcaM, pret, anFabricatie, "neinchiriat");
end //
delimiter ;

#Procedura care sterge o masina dupa id_ul acesteia;
drop procedure if exists deleteMasina;
delimiter //
create procedure deleteMasina(id int)
begin
    delete from masina where idMasina=id;
end //
delimiter ;

#Procedura care actualizeaza pretul unei masini disponibile;
drop procedure if exists updateMasina;
delimiter //
create procedure updateMasina(id varchar(45), pretNou
varchar(45))
begin
    update masina set pret_zi=pretNou where id=idMasina;
end //
delimiter ;

#Procedura care insereaza o noua aeronava;
```

```
drop procedure if exists insertAeronava;
delimiter //
create procedure insertAeronava(comp varchar(45), num
varchar(45), cap varchar(45), an varchar(45))
begin
    insert into aeronava (companie, Nume, capacitate,
an_fabricatie) values (comp,num,cap,an);
end //
delimiter ;

#Procedura care sterge o aeronava dupa un anumit id;
drop procedure if exists deleteAeronava;
delimiter //
create procedure deleteAeronava(id varchar(45))
begin
    update gestiune set idAeronava=1 where
gestiune.idAeronava=id;
    update personal set id_aeronava=1 where
personal.id_aeronava=id;
     delete from aeronava where aeronava.id_aeronava=id;
end //
delimiter ;

#Procedura care sterge o plecare, stergand toate legaturile
cu celalalte tabele;
drop procedure if exists stergerePlecare;
delimiter //
create procedure stergerePlecare(id varchar(45))
begin

          -- set @i=(select gestiune.id_gestiune from gestiune,
plecari where plecari.id_plecare=id and
gestiune.id_gestiune=plecari.id_gestiune);
```

```
--          delete from gestiune where gestiune.id_gestiune=@i;

    delete from bagaje  where bagaje.id_Bagaj=id;
    delete from rezervari where rezervari.id_plecare=id;
    delete from plecari where plecari.id_plecare=id;

end //
delimiter ;

#procedura care adauga o noua gestiune( o gestiune are o
aeronava pe care ulterior o atribuim unui zbor
(plecare/sosire));
drop procedure if exists InsertGestiune;
delimiter //
create procedure InsertGestiune( aeronava int)
begin
insert into gestiune(id_aeroport, idAeronava ) values(1,
aeronava);
end //
 delimiter ;


 #Procedura care adauga o noua plecare si ii atribuie o
gestiune existenta(o aeronava).
drop procedure if exists InsertPlecari;
delimiter //
create procedure InsertPlecari(destinatie varchar(45),
data_plecarii date, ora_plecarii time, zbor_direct varchar(45),
escala varchar(45), pret_bilet int , tip_plecare varchar(45))
begin
    select max(id_gestiune) into @idG from gestiune;
    insert into Plecari(id_gestiune, destinatie, data_plecarii,
ora_plecarii, zbor_direct, escala, pret_bilet, tip_plecare)
```

```
    values (@idG, destinatie, data_plecarii,ora_plecarii,
zbor_direct, escala, pret_bilet, tip_plecare);

     select max(id_plecare) into @id from plecari;

    select a.capacitate into @c
    from aeronava a
    join gestiune g on a.id_aeronava=g.idAeronava
    join plecari p on g.id_gestiune=p.id_gestiune where
p.id_plecare=@id;

    update plecari set capacitate=@c where id_plecare=@id;

end //
 delimiter ;
 #Procedura care actualizeaza pretul unei plecari.
drop procedure if exists UpdatePlecari;
delimiter //
create procedure UpdatePlecari(id int, pret int)
begin
update plecari set pret_bilet=pret where id_plecare=id;

end //
 delimiter ;

 #Procedura care adauga un nou angajat avand o anumita
functie;
 drop procedure if exists InsertPersonal;
delimiter //
create procedure InsertPersonal(id_aeronava int,nume
varchar(45),prenume varchar(45),varsta int,data_angajarii
date,functie varchar(45))
begin
```

```
insert into Personal(id_aeronava ,nume,prenume,varsta
,data_angajarii ,functie) values (id_aeronava
,nume,prenume,varsta ,data_angajarii ,functie);
end //
delimiter ;

 #Procedura care sterge un anumit angajat.
drop procedure if exists DeletePersonal;
delimiter //
create procedure DeletePersonal( id int)
begin
    delete from Personal where id_personal=id;
end //
delimiter ;

#Procedura care adauga un bagaj unei rezervari(practic unui
user)
drop procedure if exists insertBagaj;
delimiter //
create procedure insertBagaj(greu varchar(45))
begin
    set @id=(select max(id_Rezervare) from rezervari);
    insert into bagaje (Greutate,id_rezervari) values
(greu,@id);
end //
delimiter ;

#Procedura care angajatilor care presteaza servicii
aeriene(pilot/stewardesa) li se atribuie o alta aeronava.
drop procedure if exists UpdatePersonal;
delimiter //
create procedure UpdatePersonal(id int, aeronava int)
begin
```

```
update Personal set id_aeronava=aeronava where
id_personal=id;
end //
delimiter ;
```

3.6 View-uri

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.

```
#Vedere care afiseaza cele mai cautate destinatii
drop view DestinatiiCautate;
create view DestinatiiCautate as
select destinatie,count(destinatie) as 'numar'
from plecari group by destinatie order by count(destinatie)
desc;

#Vedere care afiseaza numarul de zboruri zilnice
 drop view AfisareZile;
 create view AfisareZile as
 select distinct data_plecarii, count(data_plecarii) as
Numar_Zboruri from plecari group by data_plecarii order by
data_plecarii asc;

 #Vedere care afiseaza plecarile de noapte.
 drop view Afisareore;
 create view Afisareore as
```

```
select destinatie as Destinatia, ora_plecarii as
Ora_Plecarii, data_plecarii as Data_Plecarii from plecari
where ora_plecarii  between '00:00' and '06:00';
```

```
#Vedere care afiseaza zborurile care aterizeaza in
intervalul orar 7 dimineata-11 seara.
drop view AfisareSosiriZi;
 create view AfisareSosiriZi as
 select de_la as De_la, ora_sosire as Ora_Sosire,
data_sosire as Data_Sosirii from sosiri where ora_sosire
between '07:00' and '20:00';

#Vedere care afiseaza zborurile care aterizeaza noaptea
drop view AfisareSosiriNoapte;
create view AfisareSosiriNoapte as
 select de_la as De_la, ora_sosire as Ora_Sosire,
data_sosire as Data_Sosire from sosiri where ora_sosire
between '20:00' and '07:00';
```

3.7Cod sql

```
Drop database if exists ProiectAeroport;
CREATE SCHEMA IF NOT EXISTS ProiectAeroport;
USE ProiectAeroport;

 create table if not exists Gestiune(
id_gestiune int auto_increment,
id_aeroport int,
idAeronava int,
```

```
    constraint pk_Gestiune primary key nonclustered(id_gestiune,idAeronava)
    );

    Create table Aeroport(
    id_aeroport int auto_increment,
    Nume Varchar(45),
    Locatie varchar(45),
    tara varchar(45),
    tip varchar(45),
    nr_masini_disponibile int,
    constraint PK_Aeroport primary key NONCLUSTERED(id_aeroport)
    );
    alter table Gestiune add
    constraint fk_Gestiune_Aeroport foreign key(id_aeroport) references Aeroport(id_aeroport);


    create table if not exists Admin_Aeroport(
    idAdmin int auto_increment,
    id_aeroport int,
    Nume varchar(45),
    prenume varchar(45),
    Username varchar(45),
    parola varchar(45),
    constraint pk_Admin_Aeroport primary key nonclustered(idAdmin, id_aeroport)
    );

     alter table Admin_Aeroport add
     constraint fk_admin_aeroport_aeroport1_idx foreign key(id_aeroport) references Aeroport(id_aeroport);
```

```
Create table Plecari(
id_plecare int auto_increment,
id_gestiune int,
destinatie varchar(45),
data_plecarii date,
ora_plecarii time,
zbor_direct varchar(45),
escala varchar(45),
pret_bilet int,
tip_plecare varchar(45),
capacitate int,
constraint pk_Plecari primary key NONCLUSTERED(id_plecare)
);
  create table Aeronava(
id_aeronava int primary key auto_increment,
companie varchar(45),
Nume varchar(45),
capacitate int,
an_fabricatie int
);
create table Personal(
id_personal int auto_increment,
id_aeronava int,
nume varchar(45),
prenume varchar(45),
varsta int,
data_angajarii date,
functie varchar(45),
constraint pf_Personal primary key nonclustered(id_personal)
```

```
);

alter table Personal add(
-- constraint fk_Aeroport_Personal foreign key(id_aeroport) references Aeroport(id_aeroport),
constraint fk_Personal_Aeronava foreign key(id_aeronava) references Aeronava(id_aeronava));


alter table Plecari add(
constraint FK_Gestiune_Plecari foreign key(id_gestiune) references Gestiune(id_gestiune));



alter table Gestiune add
constraint fk_Aeronava_Gestiune foreign key(idAeronava) references Aeronava(id_Aeronava);

create table if not exists User_Aeroport(
id_User int primary key auto_increment ,
Nume varchar(45),
Prenume varchar(45),
varsta int,
email varchar(45),
Telefon varchar(45),
masina_inchiriata varchar(45),
destinatie_rezervata varchar(45)
);
create table if not exists Rezervari(
id_Rezervare int auto_increment,
id_User int,
id_plecare int,
```

```
    constraint PK_Rezervari primary key NONCLUSTERED(id_rezervare)
    );
    create table if not exists Bagaje(
    id_Bagaj int auto_increment,
    Greutate float,
    id_rezervari int,
    constraint pk_Bagaje primary key nonclustered(id_Bagaj)
    );

    alter table Bagaje add(
    constraint fk_bagaje_rezervari foreign key(id_rezervari) references rezervari(id_rezervare));

    alter table Rezervari add(
    constraint FK_Rezervari_User_Aeroport foreign key(id_User) references User_aeroport(id_User),
    constraint Fk_Plecari_Rezervari foreign key(id_plecare) references Plecari(id_plecare));

    drop table if exists Sosiri;
    create table Sosiri(
    id_sosiri int primary key auto_increment,
    id_gestiune int,
    de_la varchar(45),
    data_sosire date,
    ora_sosire time);


    alter table Sosiri add
    constraint fk_Gestiune_Sosiri foreign key(id_gestiune) references Gestiune(id_gestiune);
```

```
create table if not exists Contact(
id_aeroport int,
mail varchar(45),
numar_telefon varchar(15),
mesaj text,
constraint pk_Contact primary key nonclustered(id_aeroport)

);
create table if not exists Masina(
idMasina int primary key auto_increment,
id_aeroport int,
nr_locuri int,
Marca varchar(45),
pret_zi int,
an_fabricatie int,
statuss varchar(45),
id_user varchar(45)
);
alter table Masina add(
constraint fk_Aeroport_Masina foreign key(id_aeroport) references Aeroport(id_aeroport) );
alter table Contact add
constraint fk_Aeroport_Contact foreign key(id_aeroport) references Aeroport(id_aeroport);



-- Inserarile
insert into aeroport values
(1, 'Avram Iancu', 'Cluj-Napoca', 'Romania', 'international', 3);
```

```
insert into contact values
(1, 'aeroportinternational@yahoo.com', '0364457',null);

insert into admin_aeroport values
 (1,1, 'Admin', 'Aeroport', 'admin', 'avramiancu');

 insert into masina values
 (1,1,5, 'Audi', 100, 2013, 'neinchiriat', null),
 (2,1,7, 'Ford', 60,2008,'neinchiriat', null),
 (3,1,5, 'BMW', 150, 2015,'neinchiriat', null);

 insert into aeronava values
 (1,'','',0,0),
(2, 'British Airlines','Boeing 707', 120, 2008),
(3, 'Tarom', 'Airbus 10', 80, 2005),
(4, 'Turkish Airlines','Antonotov', 300, 2020),
(5, 'Wizzair','Boeing 707', 240, 2018),
(6, 'Wizzair', 'Boeing 707', 140, 2019),
(7, 'Qatar', 'Airbus 100', 30, 2020),
(8,'ceva','altceva',50,2019);
insert into personal values
 (1, 2, 'Maria', 'Enescu', 20, '2020-06-06', 'stewardesa'),
(2, 3, 'Lucian', 'Echim', 34, '2016-10-06', 'pilot'),
(3, 7, 'Marian', 'Enescu', 25, '2018-06-29', 'steward'),
(4, 5, 'Andreea', 'Bucur', 45, '2008-03-18', 'pilot'),
(5, null, 'Antonia', 'Andreica', 55, '2008-03-28', 'casierita'),
(6, null, 'Constantin', 'Senila', 30, '2019-07-12', 'politist'),
(7, 2, 'Erik', 'Popescu', 24, '2020-07-10', 'pilot');

 insert into gestiune values
 (1,1,7),
 (2,1,2),
 (3,1,3),
```

```
(4,1,4),
(5,1,5),
(6,1,6),
(7,1,7),
(8,1,2),
(9,1,3),
(10,1,4),
(11,1,5),
(12,1,6);
```

```
insert into plecari values
(1, 2, 'Lisabona', '2020-12-30', '13:20','nu', 'Bucuresti',1000, 'intern',120),
(2, 3, 'Madrid', '2020-12-14', '13:45', 'da', null,2000, 'extern',80),
(3, 5, 'Roma', '2020-12-14', '15:40', 'da', null, 1400,'extern',300),
(4, 4, 'Tokyo', '2020-12-14', '18:00', 'nu', 'Paris',2890, 'extern',240),
(5, 1, 'Bucuresti', '2020-12-14', '20:30', 'da', null,400, 'intern',140),
(6, 6, 'Roma', '2020-12-15', '02:30', 'da', null, 2000, 'extern',140);
```

```
insert into sosiri values
(1, 7,'Frankfurt',  '2020-12-14', '04:30'),
(2, 8,'Dublin',  '2020-12-14', '07:45'),
(3,9,'Bucuresti',  '2020-12-14', '08:55'),
(4, 10,'Roma', '2020-12-14', '14:55'),
(5, 12,'Sofia', '2020-12-15', '00:30');
```

```
insert into user_aeroport values
(1, 'Popa', 'Mihaela',45,'popamihaela@gmail.com', '0745985830',null,'da'),
(2, 'Miron', 'Alina', 6,null,null,null,'da'),
(3, 'Mateescu', 'Bogdan', 30,'mateescubogdan@gmail.com', '0745985876',null,'da');

insert into rezervari values
(1, 1, 1),
(2, 2, 2);

insert into bagaje values
(1, 30, 1),
(2,  40, 2);
```

3.8 Forma 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.

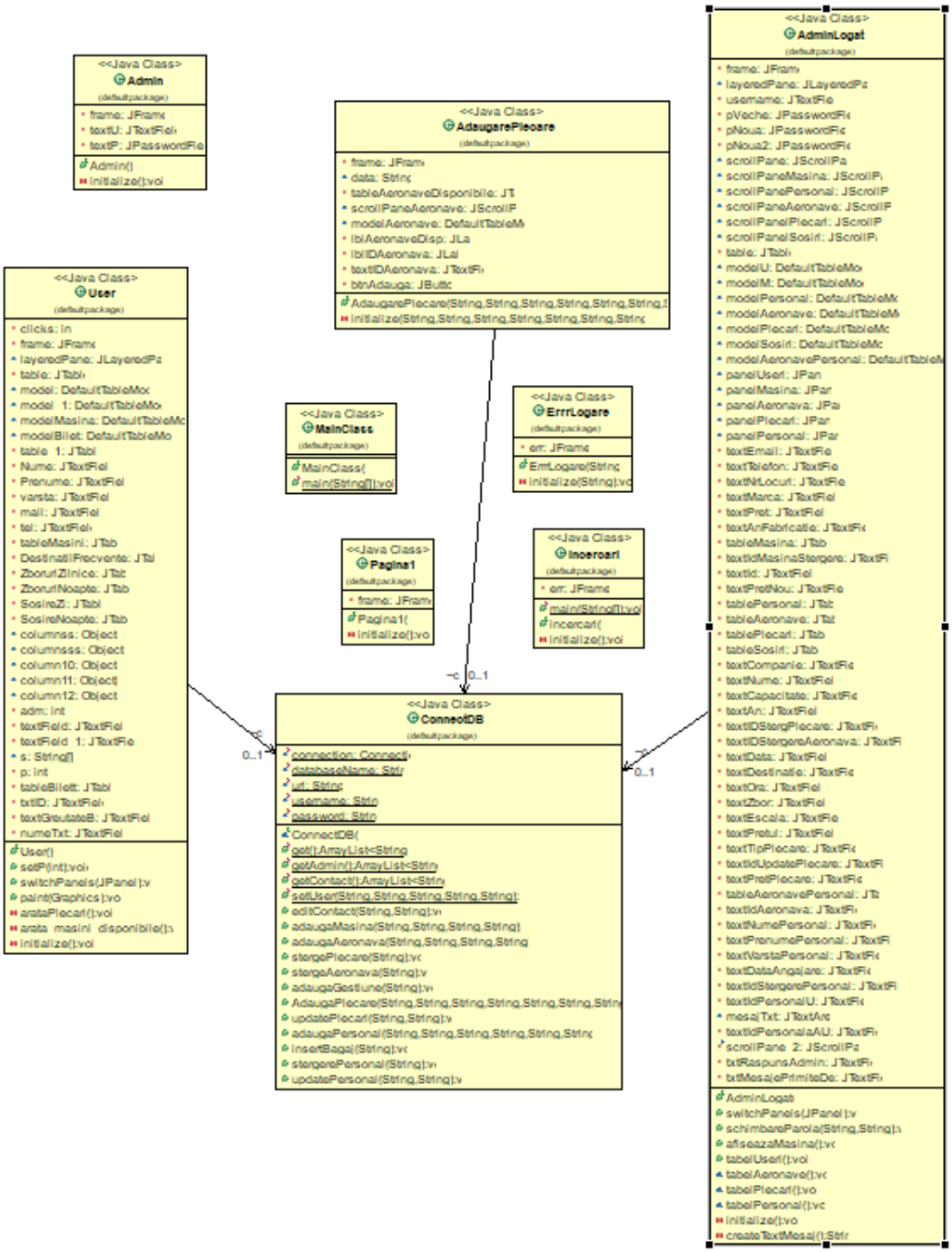
Formele normale reprezintă criterii de ghidare a proiectantului bazei de date în ceea ce privește alegerea

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 respecta forma normala 3 deoarece “relatia depinde de cheie si toate atributele non-cheie sunt (trebuie sa fie) mutual independente.

4.Detalii de implementare

4.1 Structura claselor in Java



4.2 Manual de utilizare

Aceasta aplicatie poate fi utilizata de doua tipuri de utilizatori:

1.User

Acesta intra pe pagina dedicata apasand butonul din stanga user, acest buton o sa trimita in pagina principala a aplicatie.

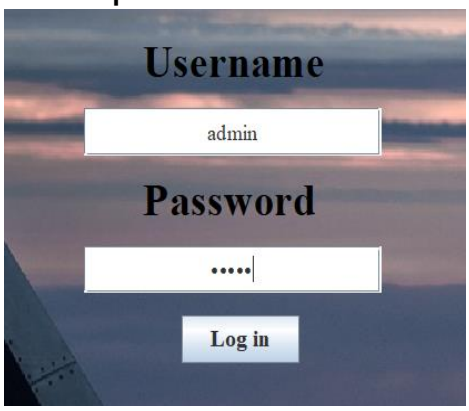
Pagina principala aeste prezentata in imaginea de mai jos:

Pentru a vedea detalii despre sosiri sau plecari utilizatorul trebuie sa foloseasca butoanele dedicate (Sosiri respectiv

Plecari). Butonul *Inchiriaza* respectiv *Rezerva* ii cere datele utilizatorului apoi este trimis sa isi treaca datele personaale pt a putea inchiria o masina sau rezerva un zbor. Pe butonul *Contact* poate sa gaseasca date despre aceasta companie aeriana sau chiar poate trimite un mesaj.

2. Admin

Admin-ul are acces la toate datele pe care le poate vedea userul si se ocupa de administrarea companie in aplicatie. Pentru a ajunge la pagina dedicata admin-ului este nevoie sa se apese butonul *Admin* apo este redirectionat intr o pagina unde o sa trebuiasac sa isi introduca username si parola altfel nu poate avea acces.



Dupa ce atat username-ul cat si parola au fost introduse corect admin-ul are acces la toate datele din aplicatie. Pe pagina principala isi poate schimba parola sau poate raspunde la mesajele primite de la useri.

Admin-ul are un meniu un meniu din care poate sa aaleaga sa mearga in pagina user-ului sau sa adauge si sa stearga zboruri, masini, aeronave sau personal. Deasemena poate sa modifice datele de contact sau sa vada userii inregistrati si scopul lor, iar daca nu au niciun scop pot fi stersi.

4.3 Elemente de securitate ale aplicatiei

Elementele de securitate sunt facute pentru admin, acesta se poate conecta in pagina dedicata folosind un username si o parola valida. Daca admin-ul nu introduce atat usename-ul cat si parola valida acesta nu se poate conecta. Partea la care are acces doar admin-ul este facuta in java, unde daca username-ul si parola sunt corecte este directionat catre pagina dedicata in care poate efectua modificarile necesare.

5.Concluzii

Consideram ca cerintele acestui proiect au fot indeplinite cu succes. Am reusit sa creem de la zero o baza de date complexa pt a putea stoca toate datele despre aeroport cat mai eficient si mai simplu. Interfata creata ajuta la interactionarea mai usoara cu tot ce pune la dispozitie acest proiect.

Aplicatia dezvoltata ofera utilizatorilor informatiile necesare pt a putea rezerva un zbor spre o locatie anume sau pentru a-si rezerva o masina din lista de masini disponibile pt o perioada de timp. Utilizatorul poate sa vada si date despre aeroport sau poate chiar sa trimita un mesaj companiei. Aceasta aplicatie este prietenoasa si cu adminul deoarece acesta se poate conecta simplu folosind username-ul si parola valide. Admin-ul poate sa adauge sau sa stearga sau chiar sa updateze anumite informatii referitoare la functionarea acestui aeroport. De asemenea, adminul pooate sa- si modifice parola poate sa vada toti userii si are un buton prin care ii poate sterge pe cei care s-au inscris fara sa isi inchirieze sau rezerve nimic.

In ideea dezvoltarii, aceasta aplicatie ar mai putea fi dezvoltata pe partea de efectuare a platii sau inchirierea a masinilor pe un numar precizat de zile, lucru care momentan nu e implementat. Ar mai putea fi implementat un sistem prin care varstnicii ar putea beneficia de reduceri la bilete.