



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

PROIECT

la disciplina

BAZE DE DATE

Parc Auto



Student:

Pop Raul-George

Grupa: 30225

An academic

2021-2022

Curprins

1.Introducere	3
2.Analiza cerințelor utilizatorilor	4
2.1 Cerințe și constrângeri	4
2.2 Organizarea structurală a cerințelor utilizator	4
2.3 Determinarea și caracterizarea profilurilor de utilizator	5
3. Modelul de date și descrierea acestuia	6
3.1 Entități și atributele lor	6
3.2 Normalizarea datelor.....	7
3.3 Diagrama EER/UML pentru modelul de date complet	8
3.4 Proceduri, trigger, rapoarte și view-uri	8
4.Detalii de implementare	11
4.1 Descrierea funcțională a modulelor – Structura claselor Java.....	11
4.2 Manual de utilizare/instalare.....	12
4.3 Elemente de securizare a aplicației	15
5. Concluzii, limitări și dezvoltări ulterioare.....	16

1.Introducere

Proiectul presupune dezvoltarea unei aplicații care lucrează cu baze de date pentru evidența unui parc auto. Scopul aplicației este simplificarea operațiilor cu baza de date prin oferirea unei interfețe grafice prin care angajații sau vizitatorii parcului o pot utiliza. Aplicația oferă sprijin atât pentru interogarea bazei de date cât și pentru manipularea acesteia. Cu ajutorul aplicației se pot vizualiza totalitatea vehiculelor din parcul auto, numărul de loturi pe care parcul le deține, evenimentele de tipul vânzare, cumpărare, închiriere, parcare din cadrul fiecărui lot cât și inserarea sau ștergerea de noi vehicule, angajați sau evenimente din cadrul bazei de date.

Există mai multe tipuri de utilizatori: administratori care pot să prelucreze orice date din cadrul bazei, utilizatori care pot să introducă date noi și să vizualizeze datele deja existente și utilizatori anonimi care au doar posibilitatea de a vizualiza datele din cadrul bazei de date.

Pentru crearea acestei baze de date au fost luate în considerare mai multe scopuri:

- Eficientizarea operațiilor de bază care se pot realiza în cadrul unui parc auto. Accesul la date este considerabil mai rapid în comparație cu varianta clasică de stocare a datelor în format fizic în dosare și rapoarte de hârtie;
- Concurența operațiilor, adică posibilitatea de a completa simultan mai multe rapoarte, actualizarea bazei de date realizându-se în timp real;
- Prin faptul că accesul la baza de date este limitat de gradul fiecărui utilizator, datele existente nu rămân în siguranță fără a fi modificate de oricine are acces la baza de date;
- Interfața realizată este ușor de folosit de orice persoană care deține informații minime informatice, reducând astfel timpul de așteptare la vizitarea parcului auto.

Pentru dezvoltarea acestei aplicații au fost folosite:

- MySQL Workbench 6.2 – pentru crearea bazei de date, popularea inițială, realizarea diagramei UML și dezvoltarea de vederi, proceduri și triggeri specifice bazei de date;
- Eclipse – mediu de dezvoltare Java.

2. Analiza cerințelor utilizatorilor

2.1 Cerințe și constrângeri

Aplicația gestionează acțiunile dintr-o unitate care utilizează baza de date. Cerințele constau în:

- Există 3 tipuri de utilizatori: administratori, utilizatori normali și vizitatori;
- Administratorul are puterea de vizualiza, introduce, actualiza sau șterge orice tupla de date din cadrul bazei de date în timp ce un utilizator normal are posibilitatea de a vizualiza și introduce date iar vizitatorii de a vizualiza datele din cadrul bazei de date;
- Un utilizator, indiferent de rangul său, este unic identificat prin numele de utilizator și parola, care vor fi folosite pentru accesarea bazei de date. Pe lângă aceste informații, la crearea unui cont nou se memorează și numele complet al utilizatorului cât și o adresă de email. Adresa de email asociată fiecărui cont este unică, fiind permisă crearea unui singur cont per adresă de email;
- În cadrul aplicației există un set de evenimente specifice, menționate într-un tabel, pe care un utilizator le poate realiza asupra unui vehicul;
- Sistemul poate fi împărțit în mai multe module, fiecare modul având posibilitatea de gestionare a unei componente din cadrul parcului auto. Astfel se pot evidenția:
 - Modul pentru gestionarea angajaților, posturilor acestora cât și alte date
 - Modul pentru gestionarea șoferilor angajați în cadrul parcului auto, rutele pe care le parcurg în mod normal cât și alimentările pe care aceștia le realizează;
 - Modul pentru gestionarea activităților operaționale, precum adăugarea unui noi vehicul în parcul auto, vânzarea unui vehicul, parcare sau indisponibilitatea mașinilor angajate în curse regulate alături de șoferii înregistrați în baza de date cât și vehiculele aflate în reparații sau indisponibile din orice alte motive.

2.2 Organizarea structurală a cerințelor utilizator

Baza de date trebuie să stocheze următoarele informații:

- Loturile parcului auto;
- Vehiculele depozitate în fiecare lot;
- Utilizatorii;

- Șoferii;
- Angajații;
- Alimentările realizate pentru fiecare vehicul;
- Evenimentele care s-au întâmplat în parcul auto;
- Rutele posibile pe care le pot realiza șoferii;
- Brandurile de mașini depozitate în parc;
- Statusurile pe care le pot avea vehiculele;
- Locațiile la care se află loturile parcului;
- Tipurile de evenimente care se pot întâmpla în parc.

Mai mult, trebuie să permită și următoarele operații:

- Alimentarea unei mașini;
- Afișarea consumului mediu pentru fiecare șofer;
- Înregistrarea unui eveniment;
- Înregistrarea unei noi rute;
- Afișarea unui raport detaliat despre vehicule;
- Afișarea tuturor vehiculele aparținând unui brand;
- Afișarea tuturor evenimentelor la care a luat parte un angajat;
- Mărirea salariului unui angajat dacă acesta are vechimea mai mare de 2 ani;
- Ștergerea din baza de date a tuturor angajaților care au participat la mai puțin de 3 evenimente;

Aplicația Java trebuie să permită preluarea informațiilor din baza de date, căutarea și afișarea informațiilor cât și conectarea utilizatorilor înregistrați.

2.3 Determinarea și caracterizarea profilurilor de utilizator

În cadrul aplicației se evidențiază 3 tipuri de utilizatori:

1. Administratori

- Autentificare în aplicație;
- Vizualizarea datelor existente;
- Introducerea de date noi;

- Actualizarea datelor existente;
 - Ștergerea datelor existente.
2. Utilizatori
- Autentificare în aplicație;
 - Vizualizarea datelor existente;
 - Introducerea de date noi.
3. Vizitatori
- Vizualizarea datelor existente.

3. Modelul de date și descrierea acestuia

3.1 Entități și atributele lor

Alimentare oferă informații despre fiecare alimentare a unui vehicul. Atribute: ID (*"id"*), numărul de înmatriculare al vehiculului (*"nr"*), cantitatea de combustibil introdusă (*"fuel"*), data la care a avut loc alimentarea (*"fueling_date"*), numărul de kilometri realizați de la ultima alimentare (km), ID-ul șoferului care a realizat alimentarea (*"driverid"*).

Brand oferă informații despre fiecare brand care are vehicule depozitate în parcul auto. Atribute: ID (*"id"*), numele brandului (*"nname"*), capacitatea de combustibil pe care o poate stoca un vehicul aparținând brandului respectiv (*"fuel_capacity"*).

Driver oferă informații cu privire la șoferii care au vehicule înregistrate în parcul auto. Atribute: ID (*"id"*), numele șoferului (*"nname"*), prenumele șoferului (*"surname"*), vârsta șoferului (*"age"*), salariului șoferului (*"salary"*), adresa de reședință a șoferului (*"adress"*), ID-ul rutei pe care o urmează în mod regulat șoferul (*"routeid"*).

Employee oferă informații cu privire la angajații parcului auto. Atribute: ID (*"id"*), numele angajatului (*"nname"*), prenumele angajatului (*"surname"*), vârsta angajatului (*"age"*), salariului angajatului (*"salary"*), adresa de reședință a șoferului (*"adress"*), funcția pe care angajatul o are în cadrul parcului auto (*"emp_role"*), data angajării (*"emp_date"*), ID-ul lotului în care angajatul lucrează (*"lotid"*).

Event oferă informații cu privire la tipurile de evenimente care pot fi înregistrate în parcul auto. Atribute: ID (*"id"*), detalii despre eveniment (*"details"*).

Location oferă informații cu privire la locațiile la care se găsesc loturi aparținând parcului. Atribute: ID ("id"), orașul ("city"), strada ("street"), numărul ("nb").

Lot oferă informații cu privire la loturile aparținând parcului. Atribute: ID ("id"), dimensiunea maximă a lotului ("size"), ID-ul locației lotului ("locationid"), dimensiunea curentă a lotului ("current_size").

Lot_event stochează legătura între evenimentele care s-au întâmplat și loturile în care acestea s-au întâmplat. Atribute: ID ("id"), ID-ul lotului ("lotid"), ID-ul evenimentului ("eventid"), ID-ul angajatului care a luat parte la eveniment ("emp_id"), data la care s-a întâmplat evenimentul ("ev_date").

Route oferă informații despre rutele pe care le urmează șoferii în mod normal. Atribute: ID ("id"), orașul din care începe cursa ("departure"), orașul în care se termină cursa ("arrival"), distanța întregii curse ("distance").

Vehicle oferă informații despre vehiculele depozitate în parc. Atribute: ID ("id"), numărul de înmatriculare ("nb"), ID-ul brandului vehiculului ("brandid"), kilometrajul mașinii ("km"), ID-ul statusului vehiculului ("statusid"), ID-ul lotului în care se află vehiculul ("lotid"), ID-ul șoferului vehiculului ("driverid"), data ultimei alimentări ("fuel_date"), cantitatea de combustibil introdus la ultima alimentare ("fuel"), anul fabricației vehiculului ("out_year").

Vehicle_status oferă informații cu privire la statusurile pe care un vehicul le poate avea. Atribute: ID ("id"), titlul statusului ("ttype"), detalii despre status ("details").

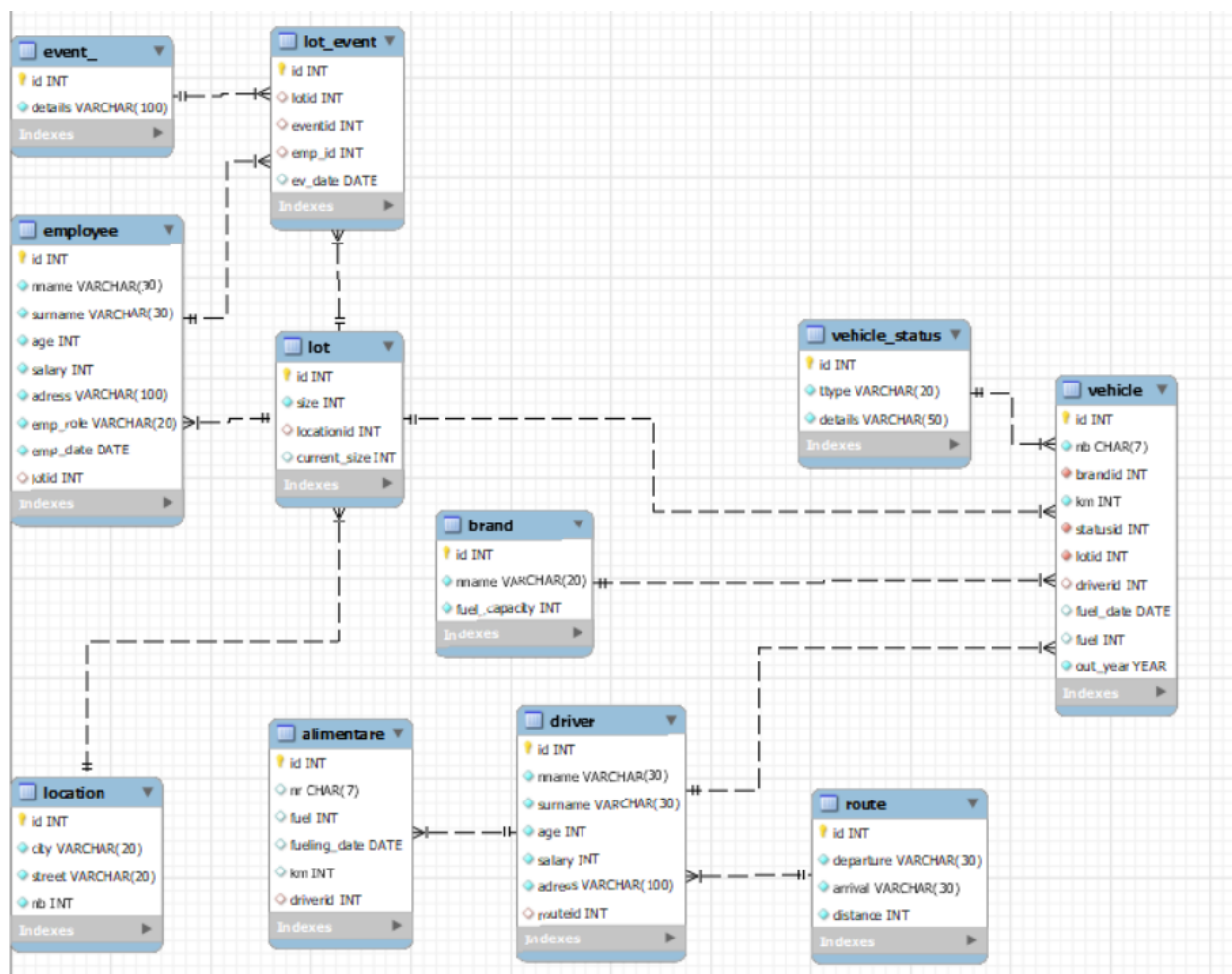
Pentru a transforma baza de date în relațional, a fost necesară introducerea unui tabel suplimentar ("lot_event") pentru a despărți relația de *many-to-many* dintre tabelele *event_* și *lot*, în 2 relații de tip *one-to-many*. Cheia primară a acestui tabel suplimentar este o cheie primară compusă și este alcătuită din *event_.id* și *lot.id*, adică cheile primare din tabele din relația *many-to-many*.

3.2 Normalizarea datelor

Definiția formei normale Boyce-Codd (FNBC) este: Fie R o schemă de relație și F mulțimea de dependențe funcționale asociată. Se spune că R este în forma normală Boyce-Codd dacă și numai dacă oricare ar fi o dependență netrivială $X \rightarrow Y$ din F, atunci X este super-cheie

pentru R. Baza de date construita, respectă aceasta definiție. Atributele fiecărui tabel nu depind 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. În fiecare tabel avem doar o cheie și toate dependențele au în partea stângă o super-cheie (cheia primară a tabelului). De exemplu, pentru tabela *vehicle* avem cheia primară *id* și toate dependențele au în stânga această super-cheie.

3.3 Diagrama EER/UML pentru modelul de date complet



3.4 Proceduri, trigger, rapoarte și view-uri

Proceduri:

1. alimentare_vehicul(numar CHAR(7), data_alimentarii DATE, km INT, sofer INT) – Adaugă o alimentare a unui vehicul și actualizează tabela vehicle;
2. consum_mediu(IN sofer INT, OUT consum_mediu DOUBLE) – Returnează consumul mediu de combustibil al unui șofer;
3. inregistrare_eveniment(tip INT, ev_date DATE, employee INT, numar CHAR(7)) – Adaugă un nou eveniment din parcul auto și actualizează tabelele aferent influențate;
4. inregistrare_ruta(numar CHAR(7), sofer INT, data_sosire DATE) – Adaugă date în tabela alimentare și tabelele influențate de aceasta, în funcție de ruta pe care o urmează în mod normal șoferul;
5. all_brand_cars(brand VARCHAR(20), stat VARCHAR(20)) – Afișează toate vehiculele de la o anumită firmă producătoare și cu un anumit status;
6. show_events(employee INT) – Afișează toate evenimentele la care a participat un anumit angajat;
7. marire_salariu(emp INT) – Modifică tabela employee prin mărirea salariului unui anumit angajat dacă acesta are vechimea mai mare de 2 ani și a luat parte la mai mult de 5 evenimente;
8. delete_employees() – Șterge din tabela employee, toți angajații care au luat parte la mai puțin de 3 evenimente;

Triggere:

1. alimentare_ins – Deoarece la introducerea unui vehicul în parc, acesta conține o cantitate de combustibil, putem considera adăugarea unui nou vehicul ca fiind o nouă alimentare. Triggerul adaugă datele corespunzătoare tabelii alimentare la introducerea unui nou vehicul;
2. add_vehicul – Acest trigger verifică la fiecare adăugare în tabela vehicle dacă lotul în care este introdus vehiculul nu a ajuns la dimensiunea sa maximă, iar dacă nu, va mări dimensiunea curentă cu 1.

Rapoarte:

1. raport_vehicule() – Oferă informații cu privire la vehiculele depozitate în parc, cuprinzând numărul de înmatriculare, anul fabricației, și consumul mediu al tuturor mașinilor din parc;

2. raport_detaliat_vehicule() – Oferă informații cu privire la vehiculele depozitate în parc, cuprinzând numărul, anul fabricației, data alimentării, șoferul și consumul, ordonat după șofer, consum, număr, în sens crescător și data alimentării în sens descrescător.

View-uri:

1. vehicule(Numar_inmatriculare, Kilometraj, Combustibil, Data_ultimei_alimentari) – Arată vehiculele cu un kilometraj mai mare de 500;
2. angajati(Nume, Prenume, Varsta, Salariu) – Arată angajații cu o vârstă mai mare de 40 de ani;
3. alimentari(Numar_inmatriculare, Data_alimentarii, Kilometri_parcursi, Sofer) – Arată alimentările realizate în ultimul an.

Interogări în algebra relațională:







1. $\sigma_{\text{out_year} = 2007 \text{ and brandid} = 6 \text{ or driverid} = 4}(\text{vehicle})$ – Interogarea selectează toate vehiculele care au anul fabricației 2007 și brandul 6 sau vehiculele al căror șofer este cel cu ID-ul 4;
2. $\Pi_{\text{id, emp_role}}(\text{employee})$ – Interogarea selectează și proiectează coloane numite id și emp_role din tabela employee;
3. $\text{driver} \bowtie_{\text{driver.salary} > \text{employee.salary}}(\text{employee})$ – Interogarea selectează toate tuplele din joinul dintre tabelele driver și employee care îndeplinesc condiția ($\text{driver.salary} > \text{employee.salary}$);
4. $\sigma_{\text{age} = 29}(\text{driver} \times \text{employee})$ – Interogarea arată toate tuplele din tabelele driver și employee care au valoarea coloanei age = 29;
5. $\Pi_{\text{id}}(\text{employee}) - \Pi_{\text{emp_id}}(\text{lot_event})$ – Interogarea arată idul angajaților care nu au luat parte la niciun eveniment.

4. Detalii de implementare




























4.1 Descrierea funcțională a modulelor – Structura claselor Java

Clasele sunt organizate în 3 pachete:

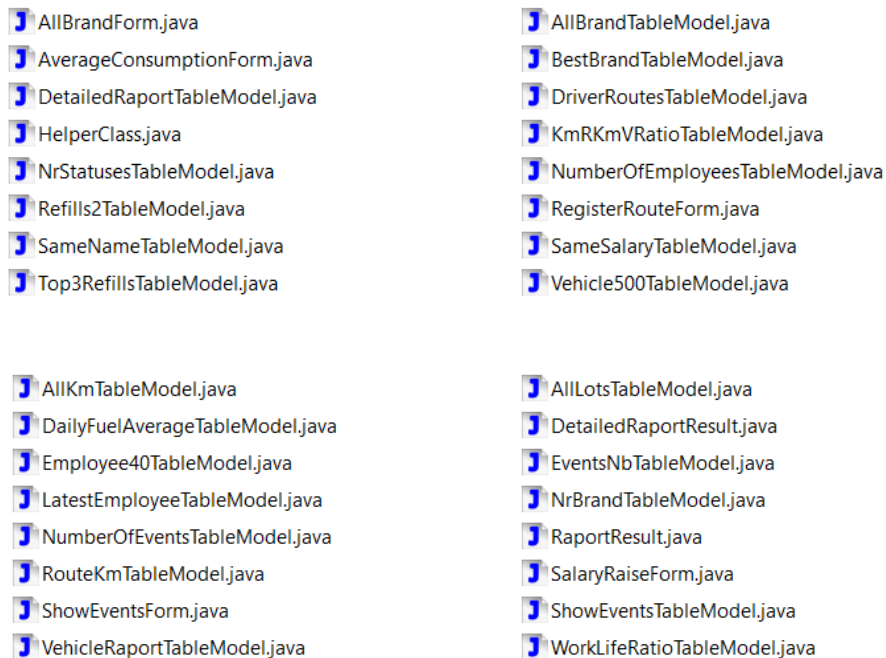
1. Main: Cuprinde totalitatea claselor responsabile cu interfața inițială a aplicației și realizarea conexiunii între Java și SGBD.

 Admin.java
 AppManager.java
 BDlink.java
 MainView.java
 Registration.java
 Ul.java

2. Tables: Cuprinde totalitatea claselor responsabile cu stocarea și echivalarea datelor și a tabelor din SQL dar și orice interfețe necesare pentru modificarea acestor tabele de bază.

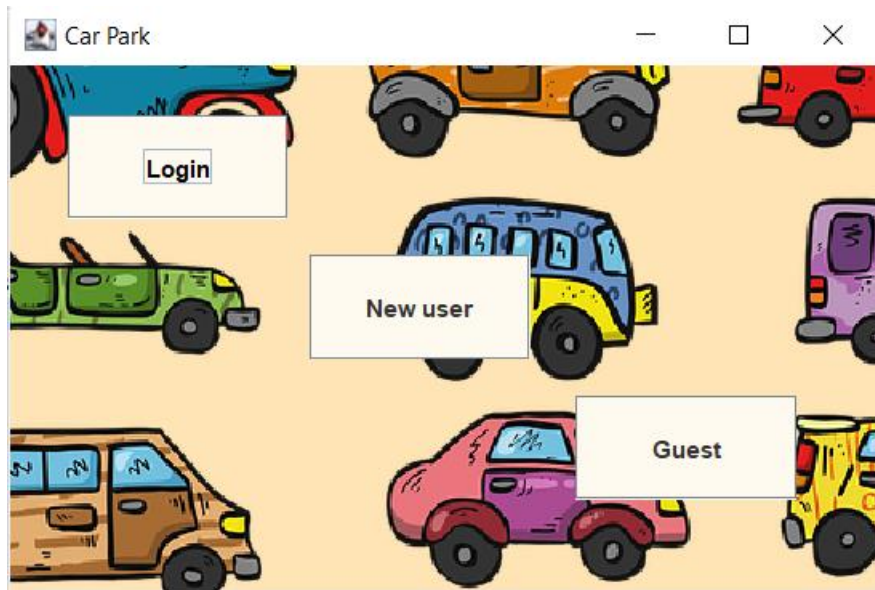
 Alimentare.java	 LocationTableModel.java
 AlimentareTableModel.java	 Lot.java
 Brand.java	 Lot_event.java
 BrandTableModel.java	 LotEventTableModel.java
 Driver.java	 LotTableModel.java
 DriverAdd.java	 RefillAdd.java
 DriverTableModel.java	 Route.java
 Employee.java	 RouteTableModel.java
 EmployeeAdd.java	 Vehicle.java
 EmployeeTableModel.java	 Vehicle_status.java
 Event_java	 VehicleAdd.java
 EventAdd.java	 VehicleStatusTableModel.java
 EventTableModel.java	 VehicleTableModel.java
 Location.java	

3. Other: Cuprinde totalitatea claselor responsabile cu realizarea interogărilor, procedurilor și rapoartelor din SQL, cuprinzând inclusiv clasele necesare interfeței Java.



4.2 Manual de utilizare/instalare

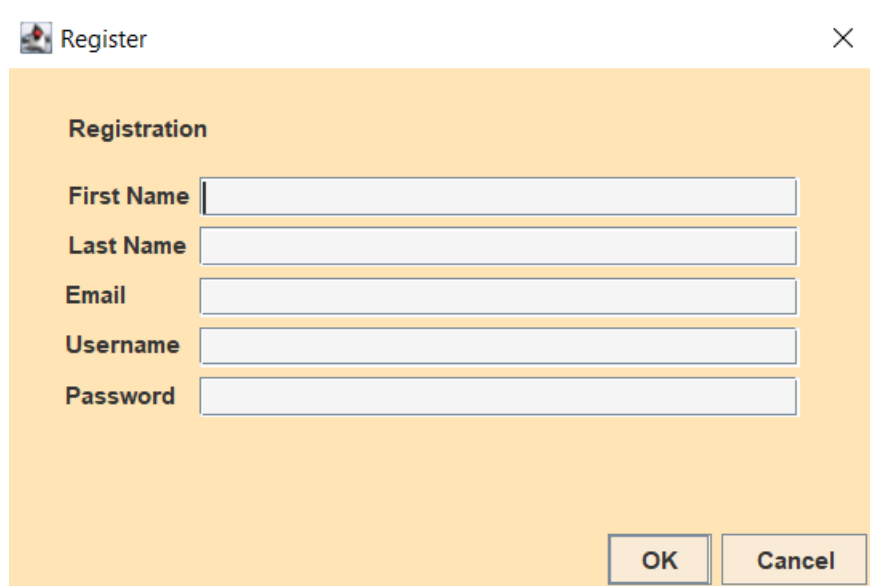
Pentru început, asigurați-vă că dețineți un server MySQL și pe acesta aveți implementată toată baza de date descrisă mai sus (tabele și popularea acestora, proceduri, triggeri și view-uri). După ce v-ați asigurat că totul a rulat cu succes, deschideți fișierul „AppManager.java” din pachetul „Main”. Prin aplicarea comenzii „Run & Build” a fișierului anterior menționat, aplicația se va deschide. Odată cu aceasta, se va deschide o pagină inițială unde aveți posibilitatea de a alege între a vă conecta la un cont deja existent, de a crea un nou cont sau de a vizualiza baza de date ca vizitator.



Prin apăsarea butonului Login, se va deschide o nouă fereastră în care vor trebui introduse username-ul și parola aferente unui cont deja existent. După introducerea datelor, prin apăsarea butonului "OK", se va realiza conectarea la baza de date.

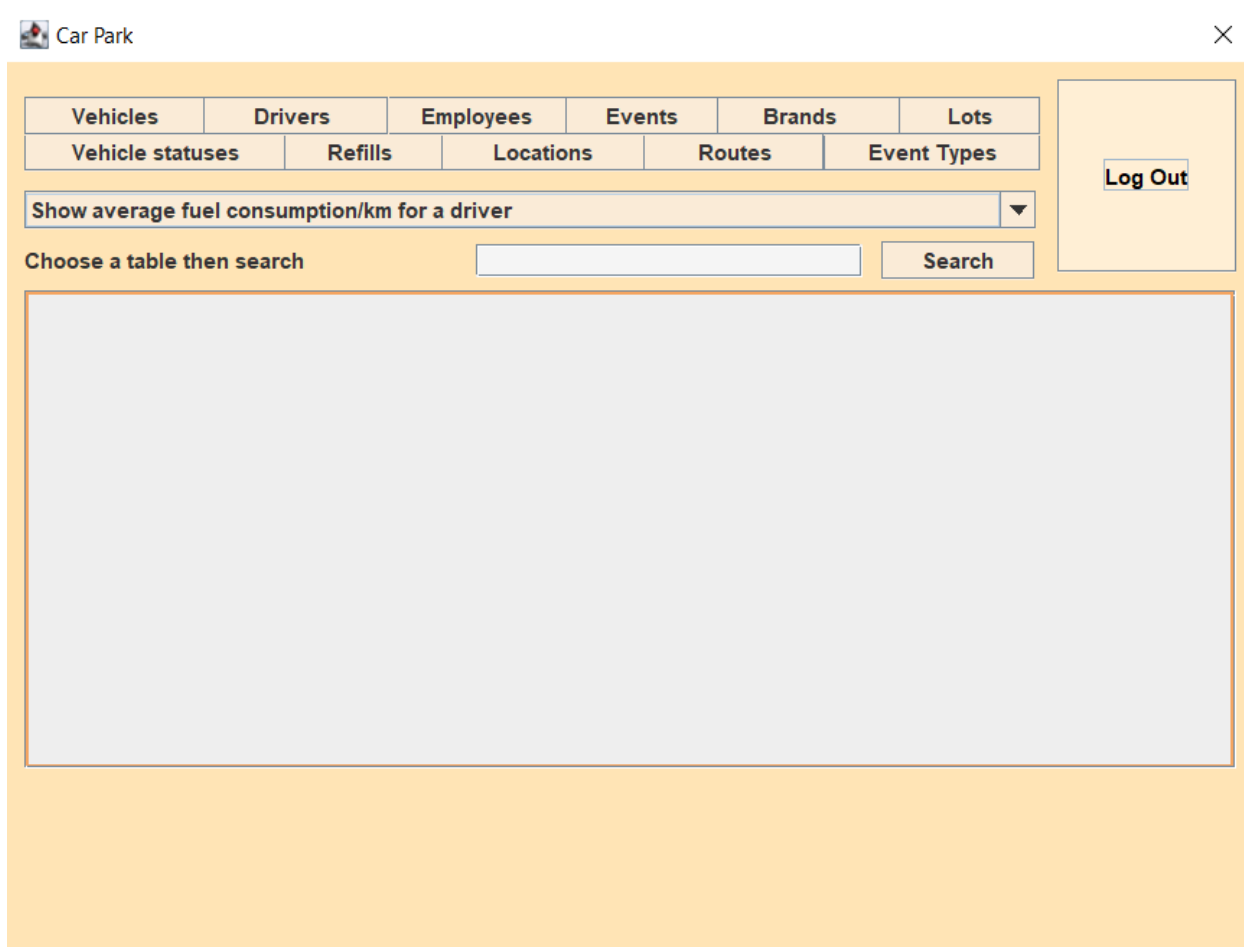
The image shows a "Login" dialog box with a light orange background. At the top, it says "Introduce your creditantials". Below this, there are two labels: "Username" and "Password". The "Username" label is next to a text input field containing the word "User". The "Password" label is next to a password input field showing six dots. At the bottom right of the dialog box, there are two buttons: "OK" and "Cancel".

Prin apăsarea butonului New user, se va deschide o nouă fereastră în care vor trebui introduse datele corespunzătoare unui nou cont. După introducerea datelor, prin apăsarea butonului "OK", veți fi notificat dacă crearea contului s-a realizat cu succes sau a apărut o eroare.

A screenshot of a 'Register' dialog box. The title bar shows a small icon and the text 'Register' with a close button (X) on the right. The dialog has a light orange background. Inside, the word 'Registration' is centered at the top. Below it are five labels with corresponding text input fields: 'First Name', 'Last Name', 'Email', 'Username', and 'Password'. At the bottom right, there are two buttons: 'OK' and 'Cancel'.

Prin apăsarea butonului Guest, veți avea acces direct la baza de date cu titlul de vizitator.

În funcție de tipul de cont cu care vă conectați, va depinde modul în care vizualizați baza de date și funcționalitățile de care beneficiați. Mai jos este prezentată o imagine cu interfața bazei de date prin conectarea la server ca vizitator.



Pe pagina principală a interfeței există un set de butoane care îndeplinesc individual câte o funcție diferită, rezultatul fiind afișat în tabelul din mijlocul ferestrei. Fiecare funcție va deschide o nouă fereastră, dacă este necesar, iar dacă nu, aceasta va realiza funcția menționată și va notifica realizarea cu succes a acesteia sau apariția unei erori. După terminarea folosirii aplicației, tot ce trebuie să faceți este să apăsați butonul de Log out și să închideți meniul principal.

4.3 Elemente de securizare a aplicației

Aplicația prezintă un nivel de securitate în momentul conectării la baza de date dar și în momentul logării propriu zise. În funcție de modul de conectare ales (admin, user sau guest), sunt disponibile următoarele operații:

- User: poate accesa baza de date pentru a vedea datele și pentru a introduce date.

- Administrator: poate accesa, modifica în orice fel toate datele si de asemenea șterge si insera date.
- Vizitator: poate vizualiza datele, fără a insera sau modifica.
- De asemenea datele sunt securizate prin faptul ca nimeni nu poate vedea parola niciunui utilizator, la momentul introducerii acesteia, fiind înlocuită cu caracterul ●

5. Concluzii, limitări și dezvoltări ulterioare

Aplicația prezentată oferă gestionarea în mod eficient a sistemului informatic al unui parc auto în cadrul căruia sunt oferite diferite funcționalități pentru utilizatorii acestuia. Aplicația oferă servicii de diferite tipuri în funcție de tipul de utilizator care se loghează. Aceste servicii sunt facilitate de interfața grafică care face navigarea printre opțiuni foarte ușoară și asigură corectitudinea datelor.

O limitare o constituie faptul că, conexiunea la baza de date este una locală, accesul pe platforma se poate face doar dacă dispozitivul de pe care se face conectarea are creată în prealabil baza de date.

În ceea ce privește dezvoltarea ulterioară, putem menționa: îmbunătățiri pe partea de interfață grafică, criptarea parolelor pentru a îmbunătăți securizarea datelor, adăugarea unui sistem de cerere de închiriere pe aceasta platforma unde utilizatorii pot depune cereri de închiriere a unui vehicul prezent în parc, fără a mai fi nevoiți să interacționeze cu o persoană fizică. De asemenea, aceasta platformă ar mai putea fi îmbunătățită prin adăugarea unui spațiu de sugestii/plângeri, unde utilizatorii pot încărca opinii/idei cu privire la parcul auto.