iesc.emfunitbv.emf**Universitatea *Transilvania* din Brașov**

**Facultatea de Inginerie Electrică și Știința Calculatoarelor**

**Departamentul Automatică şi Tehnologia Informaţiei**

**PROIECT JAVA**

**TopClassDetailing**

Dănilă Mihai Bogdan

*Gr. 4LF801 ROBO II*



Munteanu Anca-Elisabeta

*Gr. 4LF801 ROBO II*



Țîmpu Francisc

*Gr. 4LF802 ROBO II*



**BRAȘOV 2022**

CUPRINS

1. **Aplicație**
   1. **Tema de proiectare**
   2. **Schema generală a aplicației**
   3. **Descrierea interfeței modulelor si funcțiilor**
   4. **Stadiul actual**
   5. **Proiectarea propriu-zisă**
2. **Software**
   1. **Descrierea algoritmilor clasici utilizaţi**
   2. **Funcțiile utilizate**
   3. **Descrierea interfeței**
3. **Baza de date**
   1. **Definiție**
   2. **Despre baza de date**
   3. **Avantaje/Dezavantaje**
4. **Concluzii**
   1. **Concluzii generale**
   2. **Elemente de originalitate**
   3. **Direcții viitoare de dezvoltare**
   4. **Impresii**
5. **Bibliografie**

**Capitolul I**

**Aplicație**

I.1.TEMA DE PROIECTARE

Titlul temei: Aplicație web pentru detailing auto

Clientul: Top Class Detaling

Persoana de contact: Vasile Cojenelu

Date de contact:

Email: [cojeneluv@gmail.com](mailto:cojeneluv@gmail.com)

Tel.: 0770780696

Clientul își dorește o aplicație web care să prezinte serviciile oferite, alături de o galerie foto care să expună rezultatele finale (pozele în sistem before/ after) alături de prețurile aferente serviciilor. Aplicația conține un calendar care să faciliteze programarea clienților în vederea constatării la autoturism. Aplicația trebuie să conțină datele de contact ale clientului și locația, cu coordonatele gps din google maps.

I.2.SCHEMA GENERALĂ A APLICAȚIEI

Fig.1.Schema bloc a aplicației

În figura de mai sus este exemplificată schema aplicației, ordinea și legătura dintre fiecare pagină a acesteia. Mai jos este legenda fiecărei litere, corespondența paginilor:

A:Animație intro

B:Informații și Login

C:Informații

D:Pagină de autentificare sau înregistrare

E:Pagină de autentificare

F:Pagină de înregistrare

G:Meniu Principal

H:Informații

I:Servicii

J:Locație GPS

K:Coordonate GPS Maps

I.3.DESCRIEREA INTERFEȚEI, MODULELOR ȘI FUNCȚIILOR

Deschiderea aplicației este realizată cu o animație a logoului firmei clientului și prima pagină, unde pot fi accesate informații despre aceasta sau se poate merge mai departe înspre înregistrare/autentificare. Realizarea unui cont necesar autentificării poate fi creat foarte ușor apăsând butonul de înregistrare și completând informațiile cerute în noua pagină deschisă. După autentificarea realizată cu succes se ajunge în meniul principal. Aici clientul are o serie de butoane cu diferite opțiuni: meniul de informații, serviciile oferite de firmă și locația acesteia, cu opțiunea de GPS Maps.

I.4.STADIUL ACTUAL

Aplicația are următoarele pagini funcționale în stadiul actual: înregistrarea și autentificarea în aplicație, meniul principal de unde se poate naviga în informațiile despre firmă, locația cu Gps, ce oferă indicațiile de navigare și serviciile oferite de firmă.

I.5.PROIECTAREA PROPRIU-ZISA

În proiectarea aplicației, fiecare persoană a avut următoarele contribuții:

Dănilă Mihai Bogdan a realizat animația logoului, editarea videoclipului pentru layout-ul de informații, layout-ul pentru informații și login, baza de date pentru cont, tab-urile de autentificare și înregistrare, funcționalitatea butoanelor și remedierea erorilor.

Munteanu Anca-Elisabeta a realizat design-ul și schema aplicației, layout-urile de servicii și locația GPS.

Țîmpu Francisc a realizat layout-urile pentru meniul principal, informațiile legate de firmă, autentificarea și înregistrarea în aplicație.

**Capitolul II**

**Software**

II.1.DESCRIEREA ALGORITMILOR CLASICI UTILIZAȚI

Algoritmul lui Dijkstra are mai multe variante, dar una dintre cele mai frecvent utilizate este cea folosită pentru găsirea celor mai scurte căi de la vârful sursă la toate celelalte vârfuri din graf.

De exemplu, avem o anumită zonă pe harta geografică și luați această hartă geografică drept graf. Acum locațiile de pe hartă sunt verticelele noastre din algoritm, iar marginile vor fi drumurile dintre locații. Deci, greutățile marchiilor vor reprezenta distanța dintre aceste două locații. Prin această metodă aflăm calea cea mai scurtă între cele două locații.

În mod obișnuit află cea mai scurtă cale între două puncte pe o hartă și este folosit în multe aplicații, cum ar fi Google Maps. Deși acesta folosește algoritmi mai complecși și mai eficienți, algoritmul Dijkstra este baza.

II.2.FUNCȚIILE UTILIZATE

Pentru o utilizare fluentă a aplicației am folosit următoarele funcții:

“ImageView” este funcția de încărcare imagini, casete text și background pentru layout-uri, timer-ul este funcția folosită pentru afișarea videoclipului de început, “Button” este funcția pentru încărcarea unui buton cu ajutorul căruia am putut face legăturile între layout-urile aplicației, “EditText” este funcția ce ne permite editarea unui text și prin intermediul căruia utilizatorul își introduce datele contului propriu, “googleMap” este funcția cu ajutorul căreia am introdus locația GPS a firmei, iar pentru funcția folosită la GPS am generat de la google o cheie API.

Timer timer;  
 Animation topAnim;  
 ImageView image;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 setContentView(R.layout.*activity\_main*);  
  
 topAnim = AnimationUtils.*loadAnimation*(this, R.anim.*top\_animation*);  
 image = findViewById(R.id.*imagine\_logo*);  
 image.setAnimation(topAnim);  
  
 timer = new Timer();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 Intent intent = new Intent(MainActivity.this, Logo\_si\_Info.class);  
 startActivity(intent);  
 finish();  
 }  
 }, 3000);  
  
  
}}

}}Fig.2.Funcții pentru animație

public class DBHelper extends SQLiteOpenHelper {  
  
 public static final String *DBNAME*="Login.db";  
  
 public DBHelper(Context context) {  
 super(context, "Login.db", null, 1);  
 }  
  
  
 @Override  
 public void onCreate(SQLiteDatabase MyDB) {  
 MyDB.execSQL("create Table users(username TEXT primary key, password TEXT)");  
  
  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase MyDB, int oldVersion, int newVersion) {  
 MyDB.execSQL("drop Table if exists users");  
  
 }  
  
 public Boolean insertData(String username, String password)  
 {  
 SQLiteDatabase MyDB=this.getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put("username",username);  
 contentValues.put("password",password);  
 long result = MyDB.insert("users",null,contentValues);  
 if(result==-1) return false;  
 else return true;  
 }  
  
 public Boolean checkusername(String username){  
 SQLiteDatabase MyDB=this.getWritableDatabase();  
 Cursor cursor = MyDB.rawQuery("Select \* from users where username = ?", new String[]{username});  
 if(cursor.getCount()>0)  
 return true;  
 else  
 return false;  
 }  
 public Boolean checkusernamepassword(String username, String password){  
 SQLiteDatabase MyDB = this.getWritableDatabase();  
 Cursor cursor = MyDB.rawQuery("Select \* from users where username = ? and password = ?", new String[]{username,password});  
 if(cursor.getCount()>0)  
 return true;  
 else  
 return false;  
 }  
}

Fig.3.Funcții pentru baza de date

public class Autentificare extends AppCompatActivity {  
 EditText username,password;  
 Button btnlogin,inapoi;  
 DBHelper DB;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_autentificare*);  
 username = (EditText) findViewById(R.id.*username1*);  
 password = (EditText) findViewById(R.id.*password1*);  
 btnlogin = (Button) findViewById(R.id.*btnsignin1*);  
 inapoi = (Button) findViewById(R.id.*InapoiInMeniu*) ;  
 DB = new DBHelper(this);  
 btnlogin.setOnClickListener(v->{  
 String user = username.getText().toString();  
 String pass = password.getText().toString();  
  
 if (user.equals("")||pass.equals(""))  
 Toast.*makeText*(Autentificare.this, "Completati toate campurile", Toast.*LENGTH\_SHORT*).show();  
 else{  
 Boolean checkuserpass = DB.checkusernamepassword(user, pass);  
 if(checkuserpass==true){  
 Toast.*makeText*(Autentificare.this, "Autentificare cu succes", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(this ,Meniul\_Principal.class);  
 startActivity(intent);  
 }else{  
 Toast.*makeText*(Autentificare.this, "Numele si Parola invalide", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 inapoi.setOnClickListener(v->{  
 Intent intent = new Intent(this ,Meniu\_Autentificare\_Inregistrare.class);  
 startActivity(intent);  
 });  
 }  
}

Fig.4.Funcții pentru autentificare

public class inregistrare extends AppCompatActivity {  
  
 EditText username, password, repassword;  
 Button singup, singin;  
 DBHelper DB;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_inregistrare*);  
  
 username = (EditText) findViewById(R.id.*username*);  
 password = (EditText) findViewById(R.id.*password*);  
 repassword = (EditText) findViewById(R.id.*repassword*);  
 singup = (Button) findViewById(R.id.*btnsignup*);  
 singin = (Button) findViewById(R.id.*btnsignin*);  
 DB = new DBHelper(this);  
  
 singup.setOnClickListener(v-> {  
 String user = username.getText().toString();  
 String pass = password.getText().toString();  
 String repass = repassword.getText().toString();  
 if(user.equals("")||pass.equals("")||repass.equals(""))  
 Toast.*makeText*(inregistrare.this, "Completati toate campurile", Toast.*LENGTH\_SHORT*).show();  
 else{  
 if(pass.equals(repass)){  
 Boolean checkuser = DB.checkusername(user);  
 if(checkuser==false){  
 Boolean insert = DB.insertData(user, pass);  
 if(insert==true){  
 Toast.*makeText*(inregistrare.this, "Inregistrare cu succes", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(this , Meniul\_Principal.class);  
 startActivity(intent);  
 }else{  
 Toast.*makeText*(inregistrare.this, "Inregistrare esuata", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 else{  
 Toast.*makeText*(inregistrare.this, "Numele exista deja! Va rugam Autentificativa", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 else{  
 Toast.*makeText*(inregistrare.this, "Parola nu se potriveste", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
  
  
  
 singin.setOnClickListener(v->{  
 Intent intent = new Intent(this , Autentificare.class);  
 startActivity(intent);  
 });  
 }}

Fig.5.Funcții pentru înregistrare

public class DateDeContact extends AppCompatActivity {  
  
 Button btninapoi;  
 VideoView video2;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_date\_de\_contact*);  
  
 video2 = (VideoView) findViewById(R.id.*video2*);  
 btninapoi = (Button) findViewById(R.id.*inpaoiinmeniulprincipal*);  
  
 Button buttonInapoipentruMeniulPrincipal = findViewById(R.id.*inpaoiinmeniulprincipal*);  
 buttonInapoipentruMeniulPrincipal.setOnClickListener(v -> {  
 Intent intent = new Intent(this,Meniul\_Principal.class);  
 startActivity(intent);  
 });  
  
 String path = "android.resource://com.example.topclassdetailing/"+R.raw.*video\_prezentare\_topclassdetailing*;  
 Uri u = Uri.*parse*(path);  
 video2.setVideoURI(u);  
 video2.start();  
  
 video2.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {  
 @Override  
 public void onPrepared(MediaPlayer mp) {  
 mp.setLooping(true);  
 }  
 });  
 }  
  
 @Override  
 protected void onResume(){  
 video2.resume();  
 super.onResume();  
 }  
  
 @Override  
 protected void onPause(){  
 video2.suspend();  
 super.onPause();  
 }  
  
 @Override  
 protected void onDestroy(){  
 video2.stopPlayback();  
 super.onDestroy();  
 }  
}

Fig.6.Funcții pentru videoclip

public class GPS\_pentru\_TopClassDetailing extends FragmentActivity implements OnMapReadyCallback {  
  
 private GoogleMap mMap;  
 private ActivityGpsPentruTopClassDetailingBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 binding = ActivityGpsPentruTopClassDetailingBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.*map*);  
 mapFragment.getMapAsync(this);  
 }  
  
 */\*\*  
 \* Manipulates the map once available.  
 \* This callback is triggered when the map is ready to be used.  
 \* This is where we can add markers or lines, add listeners or move the camera. In this case,  
 \* we just add a marker near Sydney, Australia.  
 \* If Google Play services is not installed on the device, the user will be prompted to install  
 \* it inside the SupportMapFragment. This method will only be triggered once the user has  
 \* installed Google Play services and returned to the app.  
 \*/* @Override  
 public void onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
  
 // Add a marker in Sydney and move the camera  
 LatLng sydney = new LatLng(45.670201403357, 25.62380669139235);  
 mMap.addMarker(new MarkerOptions().position(sydney).title("TopClassDetailing"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(sydney));  
 }  
}

Fig.7.Funcții pentru locație GPS

public class Taburi extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_taburi*);  
  
 TabLayout tabLayout= findViewById(R.id.*baraTab*);  
 TabItem tabDet= findViewById(R.id.*detailing*);  
 TabItem tabFar= findViewById(R.id.*faruri*);  
 TabItem tabCar= findViewById(R.id.*caroserie*);  
 TabItem tabMot= findViewById(R.id.*motor*);  
 TabItem tabCer= findViewById(R.id.*ceramica*);  
 ViewPager viewPager= findViewById(R.id.*viewpager*);  
  
 PagerAdapter pagerAdapter= new PagerAdapter(getSupportFragmentManager(), tabLayout.getTabCount()  
 );  
  
 viewPager.setAdapter(pagerAdapter);  
  
 tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
 @Override  
 public void onTabSelected(TabLayout.Tab tab) {  
 viewPager.setCurrentItem(tab.getPosition());  
 }  
  
 @Override  
 public void onTabUnselected(TabLayout.Tab tab) {  
  
 }  
  
 @Override  
 public void onTabReselected(TabLayout.Tab tab) {  
  
 }  
 });  
 }  
}

Fig.8.Funcții pentru tab-uri

public class PagerAdapter extends FragmentPagerAdapter {  
 //@NonNull  
  
 private int numOfTabs;  
 public PagerAdapter(FragmentManager fm, int numOfTabs){  
 super(fm);  
 this.numOfTabs= numOfTabs;  
  
 }  
 @Override  
 public Fragment getItem(int position) {  
  
 switch (position){  
  
 case 0:  
 return new Detailing();  
 case 1:  
 return new Faruri();  
 case 2:  
 return new Caroserie();  
 case 3:  
 return new Motor();  
 case 4:  
 return new Ceramica();  
 default:  
 return null;  
 }  
  
 }  
  
 @Override  
 public int getCount() {  
 return numOfTabs;  
 }  
}

Fig.9.Funcții pentru PageAdapter

II.3.DESCRIEREA INTERFETEI

Interfața aplicației are drept fundal o poză făcută în Photoshop. Culoarea butoanelor a fost aleasă în funcție de culoarea logoului firmei. Animația de deschidere a aplicației este cu logoul firmei. Are și un videoclip de prezentare a oamenilor din firmă lucrând. Are locație GPS utilizată de client pentru a vedea unde se află exact firma. Sunt prezentate slide-uri cu poze pentru a observa munca firmei și stilul său.

**Capitolul III**

**Baza de date**

III.1.DEFINIȚIE

SQLite este o bază de date cu sursă deschisă și o mică bibliotecă C care implementează un motor de baze de date SQL încapsulat, oferind posibilitatea de a-l introduce în diverse sisteme și necesită zero-configurare. SQLite suportă doar tipuri de date Text, Integer și Real, asa că orice dată introdusă trebuie convertită într-unul din cele trei tipuri. SQLite este instalat în fiecare dispozitiv Android, nefiind necesară o procedură de instalare sau de administrare a bazei de date. Doar instrucțiunile SQL trebuie definite pentru crearea și actualizarea bazei de date, iar apoi aceasta este gestionată automat de platforma Android.

III.2.DESPRE BAZA DE DATE

Aceasta bază de date poate fi folosită cu uşurinţă pentru aplicațiile de pe dispozitivele mobile. Codul sursă pentru SQLite este de domeniu public. Nu există nici o pretenție de drepturi de autor pe nici o parte din codul sursă. Acest lucru înseamnă că oricine poate să folosească codul sursă SQLite. SQLite este optimizat pentru dimensiune, având o dimensiune redusă. Întreaga librărie cu toate opțiunile activate are dimensiunea mai mică de 225KiO. O bază de date întreagă este stocată într-un singur fișier de pe HDD. Fișierele bază de date pot fi liber partajate între maşini cu ordine diferită de octeți. Suportă baze de date de până la 2 terraocteți (2^41 octeți) în mărime. Deoarece o bază de date SQLite necesită puțină (sau deloc) administrare, este o alegere bună pentru dispozitive sau servicii ce necesită să meargă neadministrate sau fără sprijin uman. SQLite este potrivit pentru a fi folosit în telefoane mobile, PDA-uri, etc.

III.3.AVANTAJE ȘI DEZAVANTAJE

Există avantaje și dezavantaje în a lucra fără server. Marele avantaj este acela că nu există un proces server de instalat, setat, configurat, inițializat, administrat și reparat. Acesta este unul din motivele pentru care SQLite este un motor de baze de date "zero-configurare". Programele care folosesc SQLite nu necesită suport administarativ pentru a pregăti motorul de baze de date înainte că acesta să fie rulat.

Pe de altă parte, un motor de baze de date care folosește un server poate oferi o protecție îmbunătățită în aplicații. Deoarece un server este un proces unic persistent, el este capabil să controleze accesul la baza de date cu o mai mare precizie, permițând o blocare rafinată și concurență îmbunătățită.

Majoritatea bazelor de date SQL sunt bazate pe modelul client/server. Din cele care sunt fără server, SQLite este singura care este știută, de autor, că permite mai multor aplicații să acceseze aceeași bază de date în același timp.

**Capitolul IV**

**CONCLUZII**

IV.1. CONCLUZII GENERALE

Aplicația noastră vine cu beneficii atât pentru firmă, cât și pentru clienții acesteia. Printre ele se numară accesul la informațiile generale și cele destinate clienților (locația, datele de contact, serviciile oferite și prețurile acestora) și ocazia de a folosi GPS-ul pentru a ajunge la destinația dorită de oriunde te-ai afla.

Stadiul actual și natura aplicației nu ar permite o audiență globală, în schimb este accesibilă oricărui client. Butoanele sunt sugestive, ușor de citit și fiecare pagină are câte unul pentru întoarcerea la pagina antecedentă.

Programarea clienților este în lucru, dar le va permite înscrierea pe o pagină tip calendar. Firma o să dețină un cont de admin, cu acces la baza de date, programările făcute și posibilitatea de a le accepta sau nu. Clientul va primi o notificare informativă in legatură cu înscrierea sa.

IV.2 ELEMENTE DE ORIGINALITATE

Elementele de originalitate în această aplicație sunt reprezentate de design. Culorile sunt plăcute, bazate pe logo și suportabile în lumina scăzută. Fundalul negru cu butoanele aurii creează și un efect de “lux”.

IV.3. DIRECȚII VIITOARE DE DEZVOLTARE

Această aplicație are un potențial de a crește pentru clienții săi în următoarele puncte: o secțiune pentru anunțurile firmei, una pentru comentariile utilizatorilor, autentificarea pe baza amprentei și opțiunea de personalizare a contului.

IV.4. IMPRESII

A fost un proiect interesant, plin de provocări. A trebuit să ne organizam și să comunicăm eficient, în timp ce încercăm să folosim o aplicație complet străină. În același timp ne bucurăm că am făcut parte din acest proiect prin care am învățat ce înseamnă munca în echipă și, totodată, suntem siguri că această experiență ne va ajuta pe viitor în domeniile în care vom aplica.

**Capitolul V**

**BIBLIOGRAFIE**

1. <https://www.youtube.com/watch?v=IxhIa3eZxz8>
2. <https://www.youtube.com/watch?v=8Lq3HyBCuAA&t=82s&ab_channel=CodinginFlow>
3. <https://www.youtube.com/watch?v=hJPk50p7xwA&ab_channel=Stevdza-San>
4. <https://www.youtube.com/watch?v=YCFPClPjDIQ&ab_channel=AndroidCoding>
5. <https://www.youtube.com/watch?v=lSW9Rz0q4Eo&ab_channel=SimplifiedCoding>
6. <https://www.youtube.com/watch?v=0rdG9iYRDI4&t=249s&ab_channel=AaviskarInfotech>
7. <https://www.youtube.com/watch?v=O3HrkvFiGcI&ab_channel=ARSLTech>
8. <https://developer.android.com/training/basics/firstapp>
9. <https://developer.android.com/training/location>
10. <https://developer.android.com/training/location/retrieve-current>
11. <https://www.youtube.com/watch?v=ziJ6-AT3ymg>