**DIGITALKEY – SIMULAREA COMPORTĂRII SENZORULUI DIN MÂNERUL UȘII MAȘINII**

**Candidat: Paula-Kristine Senciuc**

**Coordonator științific: SL dr.ing. Daniela Stănescu**

Sesiunea: Iunie 2023

**REZUMAT**

În cadrul lucrării s-au realizat două aplicații mobile specifice pentru sistemul de operare Android: una destinată utilizării de către administrator, iar una destinată clienților aplicației. În plus față de partea software, s-a realizat și o partea hardware printr-o aplicație Arduino care simluează prezența unei mașini. Comunicarea dintre cele două aplicații mobile și aplicația Arduino se realizează printr-un protocol de comunicare bazat pe tehnologia Bluetooth. Scopul lucrării este acela de a înlocui cheia fizică a unei mașini prin folosirea uneia digitale cu funcționalitățiile de închidere și deschidere a mașinii prezente în aplicația mobile a clientului. Pentru a putea observa starea în care se află mașina: deschisă sau închisă, s-a utilizat un led conectat la aplicația Arduino. Prin faptul că led-ul este apris se înțelege că mașina este încuiată, iar prin faptul că led-ul este stins se înțelege că mașina este descuiată. Comunicarea dintre Android și Arduino se realizeză în ambele sensuri. Prin apăsarea unui buton cu funcționalitatea ori de închidere, ori de deschidere a mașinii, se trimite un mescaj către Arduino, iar led-ul va arăta starea solicitată. În direcția opusă, aplicația Arduino prezintă un senzor tactil, iar când acesta este apăsat va trimite un mesaj înapoi către Android care va arăta cererea printr-o imagine și prin schimbarea functionalității butonului de deschidere/închidere.

Pentru a exdine folosirea aplicației, iar utilizarea ei să poată fi realizată pentru orice mașină, s-a adăugat o bază de date gestionată de către un administrator. La solicitarea aplicației de către un client, administratorul, folosind aplicația mobile destinată, va creea un cont ce prevede email, parolă, marca mașinii și adresa MAC a modulului bluetooth a mașinii. În cadrul aplicației administratorului, acesta poate vedea datele tuturor clienților și poate adăuga noi adrese MAC, care reprezintă noi mașini, fiecărui utilizator. Astfel, daca un utilizator dorește să folosească aplicația pentru mai multe mașini, este posibil.

În concluzie, lucarea simulează atât o cheie inteligentă prin aplicația mobile, cât și prezența unei mașini prin aplicația Arduino având un modul bluetooth cu o adresa unică MAC, ceea ce duce la atingerea scopului de a realiza o comunicare între cele menționate și a face posibile funcționalitățile de închidere și deschidere a unei mașini.

**CUPRINS**

[1. INTRODUCERE 5](#_Toc135318975)

[1.1 MOTIVAȚIA ALEGERII TEMEI 5](#_Toc135318976)

[1.2 OBIECTIVELE GENERALE ALE LUCRĂRII 5](#_Toc135318977)

[2. ANALIZA STADIULUI ACTUAL ÎN DOMENIUL CHEILOR DIGITALE 6](#_Toc135318978)

[3. FUNDAMENTARE TEORETICĂ 7](#_Toc135318979)

[3.1. LIMBAJE ȘI MEDII DE PROGRAMARE 7](#_Toc135318980)

[3.1.1 ANDROID 7](#_Toc135318981)

[3.1.2. ANDROID STUDIO 10](#_Toc135318982)

[3.1.3. JAVA 10](#_Toc135318983)

[3.1.4. FIREBASE 10](#_Toc135318984)

[3.1.5. ARDUINO IDE ȘI LIMBAJUL C 10](#_Toc135318985)

[3.2. COMPONENTE HARDWARE 11](#_Toc135318986)

[3.2.1. ARDUINO MEGA 2560 11](#_Toc135318987)

[3.2.2. MODULUL BLUETOOTH HC-06 12](#_Toc135318988)

[3.2.3. SENZORUL CAPACITIV TTP223B 13](#_Toc135318989)

[4. METODA DE PROIECTARE/DEZVOLTARE 14](#_Toc135318990)

[4.1. ARHITECTURA GENERALĂ 14](#_Toc135318991)

[4.2. APLICAȚIA ARDUINO 14](#_Toc135318992)

[4.2.1. CONTROLUL SENZORULUI TACTIL 16](#_Toc135318993)

[4.2.2. CONEXIUNEA CU APLICAȚIA ANDROID 16](#_Toc135318994)

[4.2.3. CONTROLUL LED-ULUI 16](#_Toc135318995)

[4.3. APLICAȚIA ANDROID PENTRU ADMINISTRATOR 16](#_Toc135318996)

[4.3.1. MODULARIZAREA APLICAȚIEI PENTRU ADMINISTRATOR 16](#_Toc135318997)

[4.3.1.1. MODULUL DE AUTENTIFICARE 17](#_Toc135318998)

[4.3.1.2. MODULUL DE ÎNREGISTRARE A UNUI CLIENT 17](#_Toc135318999)

[4.3.1.3. MODULUL DE ADĂUGARE DE NOI MAȘINI PENTRU CLIENȚI 17](#_Toc135319000)

[4.4. APLICAȚIA ANDROID PENTRU CLIENT 18](#_Toc135319001)

[4.4.1. MODULARIZAREA APLICAȚIEI PENTRU CLIENT 18](#_Toc135319002)

[4.4.1.1. MODULUL DE AUTENTIFICARE 18](#_Toc135319003)

[4.4.1.2. MODULUL DE PORNIRE ȘI OPRIRE A BLUETOOTH-ULUI 18](#_Toc135319004)

[4.4.1.3. MODULUL DE CĂUTARE ȘI CONECTARE A MAȘINII 18](#_Toc135319005)

[4.4.1.4. MODULUL DE ÎNCHIDERE ȘI DESCHIDERE A MAȘINII 19](#_Toc135319006)

[4.4.1.5. MODULUL DE DECONECTARE A APLICAȚIEI DE LA MASINĂ 19](#_Toc135319007)

[4.5. RELAȚIA DINTRE APLICAȚIA ANDROID PENTRU CLIENT ȘI APLICAȚIA ARDUINO 20](#_Toc135319008)

[5. IMPLEMENTAREA APLICAȚIEI 21](#_Toc135319009)

[5.1. APLICAȚIA ANDROID 21](#_Toc135319010)

[5.1.1. APLICAȚIA ANDROID PENTRU ADMINISTRATOR 22](#_Toc135319011)

[5.1.1.1. SISTEMUL DE AUTENTIFICARE 22](#_Toc135319012)

[5.1.1.2. SISTEMUL DE ÎNREGISTRARE A UNUI NOU CLIENT 23](#_Toc135319013)

[5.1.1.3. SISTEMUL DE VIZUALIZARE AL CLIENȚILOR 26](#_Toc135319014)

[5.1.1.4. SISTEMUL DE ADĂUGARE DE NOI MAȘINI PENTRU CLIENȚI 28](#_Toc135319015)

[5.1.1.5. SISTEMUL DE DECONECTARE 30](#_Toc135319016)

[5.1.2. APLICAȚIA ANDROID PENTRU CLIENT 31](#_Toc135319017)

[5.1.2.1. SISTEMUL DE AUTENTIFICARE 31](#_Toc135319018)

[5.1.2.2. SISTEMUL DE PORNIRE ȘI OPRIRE BLUETOOTH 32](#_Toc135319019)

[5.1.2.3. SISTEMUL DE CĂUTARE ȘI AFIȘARE A MAȘINILOR 34](#_Toc135319020)

[5.1.2.4. SISTEMUL DE CONECTARE A MAȘINII 36](#_Toc135319021)

[5.1.2.5. SISTEMUL DE ÎNCHIDERE ȘI DESCHIDERE A MAȘINII 39](#_Toc135319022)

[5.1.2.6. SISTEMUL DE VIZUALIZARE A STĂRII MAȘINII 40](#_Toc135319023)

[5.1.2.7. SISTEMUL DE DECONECTARE A MAȘINII 41](#_Toc135319024)

[5.1.2.8. SISTEMUL DE DECONECTARE A CLIENTULUI 42](#_Toc135319025)

[5.2. APLICAȚIA ARDUINO 42](#_Toc135319026)

[6. UTILIZARE ȘI TESTARE APLICAȚIE 42](#_Toc135319027)

[6.1. MANUAL PENTRU UTILIZARE 42](#_Toc135319028)

[6.2. REZULTATE ÎN URMA TESTĂRII 42](#_Toc135319029)

[7. CONCLUZII ȘI DIRECȚII VIITOARE DE DEZVOLTARE 42](#_Toc135319030)

[7.1. CONCLUZII 42](#_Toc135319031)

[7.2. DIRECȚII DE CONTINUARE A DEZVOLTĂRII 42](#_Toc135319032)

[8. BIBLIOGRAFIE 43](#_Toc135319033)

# INTRODUCERE

## MOTIVAȚIA ALEGERII TEMEI

Lumea de astăzi se află într-o dezvoltare continuă și extrem de rapidă, fiecare domeniu având parte de inovații din punct de vedere tehnologic, chiar și de multiple ori pe an. Astfel, oamenii nu doar se aliniază cu noua tehnologie pe care sunt nevoiți să o folosească la locurile de muncă, ci majoritatea sunt tot mai atrași de folosirea acesteia și în viata de zi cu zi. În anul 2023 tot mai mulți pasionați de tehnologie își doresc conectarea dispozitivelor pe care le dețin, de la computere, la wearable devices, la electrocasnice, la televizoarele smart, până chiar și la mașină pentru a folosi toate acestea într-un „ecositem” interconectat, smartphone-ul personal fiind mecanismul de control. Datorită companiei în care activez, Continental Automotive, unde contribui la dezvoltarea aplicațiilor în domeniul automotive, am avut o mare vizibilitate asupra dezvoltării tehnologice a unui produs pe care oamenii îl folosesc în general în viața de zi cu zi, mașina. La fel ca și smartphone-urile și mașinile au tot mai multe funcții, tot mai inovative de la un model la altul pentru a crea clientului o cât mai bună experiență cu produsul. Astfel, sub îndrumarea mentorului meu din cadrul companiei, am decis să dezvolt o aplicație care există în acest moment doar pentru unele mașini, dar cel mai probabil va fi prezentă pe majoritatea mașiniilor în viitor – folosirea telefonului pentru a gestiona sistemul de închidere și deschidere al mașinii.

## OBIECTIVELE GENERALE ALE LUCRĂRII

# ANALIZA STADIULUI ACTUAL ÎN DOMENIUL CHEILOR DIGITALE

# FUNDAMENTARE TEORETICĂ

## 3.1. LIMBAJE ȘI MEDII DE PROGRAMARE

### 3.1.1 ANDROID

Android reprezintă un sistem de operare destinat în primul rând pentru dispozitive cu ecran tactil, cum ar fi telefoanele și tabletele.Acesta este dezvoltat în prezent de Google și se bazează pe un nucleu Linux [[1]](#one). Prin intermediul acestui nucleu care se află la baza stivei software, sistemul de operare Android interacționează cu părțile hardware ale unui dispozitiv [2].

O aplicație Android conține mai multe componente, iar cele de bază folosite pentru implementarea aplicației sunt Activity și Broadcast Receiver.

**Activity** este o clasă care reprezintă pentru utilizator un ecran cu care acesta poate intra în contact [3]. În figura 3.1 este caracterizată interacțuinea activităților, asftfel la pasul întâi activitatea unu este în prim plan, iar utilizatorul poate începe folosirea ei. La pasul al doilea se pornește a doua activitate, iar prima va fi pusă în stivă. La pasul al treilea începe cea de a treia activitate, iar cele două precedente sunt puse în stivă. În momentul în care se navighează înapoi de la activitatea trei la activitatea doi, activitatea trei va fi distrusă și activitatea doi va fi reluată.

Graphical user interface

Description automatically generated

Figura 3.1 – Interacțiunea activităților unei aplicații Android [4]

Fiecare activitate are propriul ei ciclu de viață care descrie evenimentele apărute în activitatea respectivă, iar fiecare eveniment este caracterizat de o stare [5].

|  |  |
| --- | --- |
| Eventiment | Stare |
| OnCreate() | se apelează în momentul în care activitatea este creată |
| OnStart() | se apelează în momentul în care activitatea este vizibilă utilizatorului aplicației |
| OnResume() | se apelează în momentul în care utilizatorul începe să interacționeze cu activitatea |
| OnPause() | se apelează în momentul în care activitatea care se desfășoară în momentul curent este pusă pe pauză și reluată activitarea precedentă |
| OnStop() | se apelează în momentul în care activitatea nu mai este vizibilă utilizatorului aplicației |
| OnDestroy() | se apelează înainte ca activitatea să fie distrusă de către sistem, fie manual, fie de sistem pentru a economisi memorie |
| OnRestart() | se apelează după ce activitatea a fost oprită și repornită |

Tabel 3.1 – Evenimentele împreună cu stările caracteristice prezente în ciclul de viață [5]

Diagram

Description automatically generated

Figura 3.2 – Ciclul de viață al unei activități al aplicației Android [5]

**Broadcast Receiver** este o componentă care descrie funcționalitatea aplicației

Android de a răspunde la mesaje provenite intern sau de la alte aplicații [6]. Există două tipuri pe care le poate avea componenta:

* Static Broadcast Receivers

Care se pot folosi nu doar când aplicația este pornită, ci și când este oprită, iar declararea acestora este prezentă în partea de fișier manifest.

* Dynamic Broadcast Receivers

Care se pot folosi doar în momentul în care aplicația este minimizată sau este în rulare de către utilizatorul acesteia [7].

În vederea oricărei aplicații Android se regăsește fișierul xml numit Manifest, creat automat la compilarea programului, care deține informații specifice și necesare fiecărei aplicații în parte. Acesta prezintă structura componentelor care se află în cadrul aplicației: fiecare Activity, Broadcast Receiver, Service și Provider, în funcție de existența implementării subclaselor ale componentelor menționate anterior. În plus, pentru fiecare componentă se precizează numele clasei Java, și modul de lansare a acestora [8].

Pentru a putea fi posibilă dezvoltarea aplicației, s-au folosit următoarele permisiuni aflate în interiorul fișierului Manifest.xml:

* BLUETOOTH\_CONNECT – folosit în cadrul aplicației pentru a putea opri și

porni bluetooth-ul telefonului mobil [9]

* BLUETOOTH\_ADMIN – folosit în cadrul aplicației pentru a putea căuta

dispozitive bluetooth în apropiere și pentru a se putea contecta la acestea [9]

* BLUETOOTH – folosit pentru a putea utiliza bluetooth-ul telefonului [9]
* BLUETOOTH\_SCAN – folosit pentru a putea dispune de funcționalitatea de a

căuta dispozitive în apropiere ce prezintă bluetooth-ul activat [9]

* ACCESS\_FINE\_LOCATION, ACCESS\_COARSE\_LOCATION and

*ACCESS\_BACKGROUND\_LOCATION* – folosite pentru a putea utiliza funcționalitatea de a căuta dispozitive bluetooth în apropiere [10]

* INTERNET și ACCESS\_NETWORK\_STATE – folosite pentru a putea utiliza

internetul pentru a avea acces la firebase

### 3.1.2. ANDROID STUDIO

În vederea dezvoltării aplicației Android s-a utilizat ca mediu de dezolvare (IDE – Integrated Development Environment) Android Studio, fiind cel oficial creat de Google. Acesta a fost conceput specific pentru Android, astfel prezintă un SDK – Software Development Kit gândit pentru a ajuta dezolvatorul pe tot parcursul aplicației. Acesta contribuie dezvoltării cu următoarele:

* Android SDK Build Tools – intervine în momentul procesului de compilare

convertind codul scris de către developer în cod mașină pentru a putea fi rulat pe un dispozitiv Android

* Android Virtual Device Manager – intervine în momentul testării aplicației,

oferind posibilitatea dezvoltatorului de a crea dispozitive virtuale Android, emulatoare, pentru a putea observa comportamentul aplicației în funcție de dispozitiv

* Layout Editor – intervine în momentul creării interfeței grafice cu utilizatorul cu

funcționalități de drag-and-drop pentru a ușura munca făcând să nu mai fie nevoie de scris cod manual [11].

### 3.1.3. JAVA

Pentru implementarea codului aplicației s-a ales folosirea limbajului de programare Java. Avantajul folosirii acestui limbaj de programare reprezintă posibilitatea rulării codului Java pe orice dispozitiv care suportă JVM – Java Virtual Machine putând fii portat cu ușurință după compilare, codul java devine bytecode, cod mașină, iar bytecode-ul este interpretat pentru a putea fi folosit pe mai multe platforme.

### 3.1.4. FIREBASE

Firebase reprezintă o colecție de servicii backend, dezvoltată de Google, care salvează și actualizează date pe cloud. Pentru o extindere a dezvoltării aplicației în care fiecare client poate avea propria adresă MAC a mașinii cu posibilitatea de adăugare a mai multor adrese MAC, s-a ales salvarea și actualizarea datelor într-o bază de date Firebase Realtime Database, astfel aplicația poate fi folosită de către mai mulți utilizatori. Prin intermediul acesteia data se salvează ca JSON [12].

Un alt beneficiu pe care Firebase îl aduce în dezvoltarea aplicației este observat prin intermediul serviciului: Firebase Authentication care oferă servicii de backend și este ușor de folosit având metode predefinite pentru crearea contului utilizatorului în care doar trebuie trimise ca parametrii emailul și parola, in cazul aplicației în cauză [12].

### 3.1.5. ARDUINO IDE ȘI LIMBAJUL C

Partea hardware a aplicației s-a implementat cu ajutorul mediului de dezvoltare Arduino IDE în care structura programului diferă față de programarea în limbajul C clasică, iar după procesul de compilare, cu ajutorul portului USB, programul va fi încarcat pe placa Arduino Mega 2560 [13].

Strunctura acestui program este împărțită în două secțiuni:

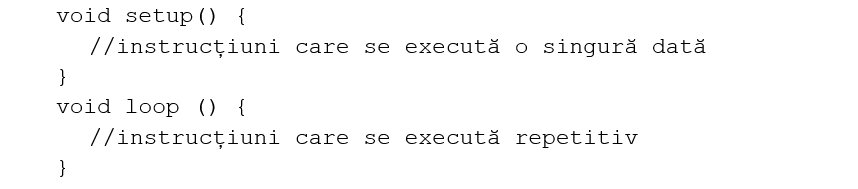


Figura 3.3 – Structura unui Program în Arduino IDE [13]

Secțiunea setup este executată o singură dată în momentul alimentării plăcii Arduino sau după ce butonul de Reset al plăcii este acționat. În această secțiune se inițializează variabilele utilizate în program, modul de lucru al terminalelor și se activează utilizarea bibliotecilor [13].

În cadrul secțiunii de loop, codul programului va fi executat ciclic atâta timp cât placa Arduino este contectată la o sursă de alimentare. Această secțiune descrie aplicația propriu-zisă [13].

Având în vedere specificațiile structurii unui program C în Arduino IDE se pot observa diferențele față de un program clasic C în care funcția clasică main s-a înlocuit cu cele două funcții: setup și loop

.

## 3.2. COMPONENTE HARDWARE

### 3.2.1. ARDUINO MEGA 2560

Sistemul hardware existent în aplicație prezintă ca și „creier al mașinii” placa de dezoltare Arduino Mega 2560 bazată pe microcontrolerul Atmega2560 care poate fi programată prin intermediul Arduino IDE [14].

Specificații tehnice:

Microcontroler: Atmega2560

Tensiunea de lucru: 5V

Tensiunea de intrare recomandată: 7-12V

Tensiuea de intrare (limită): 6-20V

Pini de intrare/ieșire digitali: 54 dintre care 15 pot fi folosiți ca PWM output

Pini de intrare analogici: 16

Curent per pin de intrare/ieșire: 20mA

Curent pentru 3.3V pin: 50mA

Memoria Flash: 256KB dintre care 8KB folosită pentru bootloader

SRAM: 8KB

EEPROM: 4KB

Clock Speed: 16 MHz

Led integrat: pinul 13 [14]

A close-up of a circuit board

Description automatically generated with medium confidence

Figura 3.4 – Arduino Mega 2560 [15]

### 3.2.2. MODULUL BLUETOOTH HC-06

Modulul Bluetooth HC-06 este folosit pentru comunicări cu transport de date fără fir pe o distanță relativ mică [16].

În contextul dezvoltării aplicației s-au folosit următorii pini:

|  |  |  |
| --- | --- | --- |
| Pinul | Numele | Funcționalitatea |
| 2 | Vcc | Are nevoie de tensiune de alimentare +5V pentru a alimenta modulul bluetooth |
| 3 | Gnd | Se conectează la masă |
| 4 | TXD | Datele se transmit serial de la modulul bluetooth prin intermediul acestul pin |
| 5 | RXD | Datele se primesc serial de la modulul bluetooth prin intermediul acestul pin |

Tabelul 3.2 – Descrierea Pinilor Modulului Bluetooth HC-06 [16]

Rolul acestui modul bluetooth în cadrul proiectului este de a realiza comunicarea dintre aplicația Android și aplicația Arduino, fiind o comunicație realizată pe bluetooth. Acest modul prezintă o adresă unică MAC care descrie unicitatea unei mașini, fiecare mașină are propria ei adresă MAC, iar prin intermediul acestei adrese MAC se poate realiza conexiunea între cele două aplicații, software și hardware.

Specificații tehnice:

Protocolul Bluetooth: Protocolul standard V2.0

Tensiunea de operare: 3.3 – 6V

Banda de frecvență: 2.40Ghz – 2.48 Ghz

Diagram, schematic

Description automatically generated

Figura 3.4 – Modulul Bluetooth HC-06 [17]

### 3.2.3. SENZORUL CAPACITIV TTP223B

Senzorul Capacitiv TTP223B funcționează cu o tensiune de alimentare cuprinsă intre 2,5V și 5V. Acesta are un timp de răspuns rapid de 60 ms, iar daca nu a mai fost atins de 12 s, va intra in modul de low power care va face ca timpul de răspuns să fie unul mai încet, mai exact de 220 ms [18].

În ceea ce privește utilizarea acestuia în cadrul aplicației dezvoltate, scopul este de a simula existența senzorului din mănerul ușii unei mașini, având și caracteristicile de deschidere și închidere observate prin intermediul stării unui led incorporat în placa Arduino și programat în Arduino IDE.

A picture containing text, electronics

Description automatically generated

Figura 3.5 - Senzor Capacitiv TTP223B [18]

# METODA DE PROIECTARE/DEZVOLTARE

## 4.1. ARHITECTURA GENERALĂ

Arhitectura generală este compusă din două aplicații Android care comunică cu două servicii: Firebase Authentication și Realtime Database Firebase. Aplicația Android destinată administratorului se ocupă cu gestionarea clienților, iar aplicația Android destinată utilizatorilor preia datele de logare, iar o data cu acestea câmpul specific adreselor MAC pentru a facilita comunicarea cu mașina.

În cadrul arhitecturii generale ale lucrării s-a adăugat și placa Arduino Mega 2560 care simulează existența unei mașini și comunica prin intermediul Bluetooth cu aplicația Android a clientului.

A picture containing text, electronics, screenshot, mobile phone

Description automatically generated

Figura 4.1 - Arhitectura generală

## 4.2. APLICAȚIA ARDUINO

Aplicația Arduino a fost construită în jurul ideii de a reprezenta o mașină împreună cu posibilitatea acesteia de a comunica cu telefonul pentru a facilita funcționalitățile de închidere și deschidere și verificare a acestora. Comunicarea se realizează prin intermediul bluetooth-ului cu trimiterea mesajelor: GET\_STATUS, LOCK și UNLOCK de la Android la Arduino și trimiterea mesajelor: STATUS\_LOCK și STATUS\_UNLOCK de la Arduino la Android. Închiderea și deschiderea mașinii poate fi făcută în două moduri, de la telefon din aplicația Android și prin atingerea seznorului mânerului mașinii care actualizează funcționalitățile din aplicația Android.

A picture containing text, screenshot, diagram, font

Description automatically generated

Figura 4.2 - Schema logică de funcționare a aplicației Arduino

### 4.2.1. CONTROLUL SENZORULUI TACTIL

Senzorul tactil este introdus în sistemul hardware al aplicației pentru a simula comportamentul unui senzor ce aparține mânerului mașinii. Funcționalitatea acestuia este descrisă cu ajutorul unui led ce indică starea mașinii. În momenul apăsării senzorului tactil, dacă ledul este aprins, ledul se va stinge și se va trimite către aplicația Android un mesaj ce indică noua stare, ledul stins, care reprezintă mașina în starea deschisă. Dacă ledul este stins, iar senzorul se apasă, starea acestuia se va schimba în aprins și aplicația mobile va fi înștiințată.

### 4.2.2. CONEXIUNEA CU APLICAȚIA ANDROID

Comunicarea între cele două aplicații prezente: aplicația Android și aplicația Arduino se realizează prin protocolul bluetooth al modulului HC06. Datele se trimit și se primesc serial, bit cu bit, mesajele transmise fiind pe 8 biți.

### 4.2.3. CONTROLUL LED-ULUI

Led-ul reprezintă mecanismul vizual care arată starea pe care o are mașina în funcție de comanda cerută, de închidere sau de deschidere. Acesta este controlat atât din aplicația Android destinată clientului, cât și din aplicația Arduino.

## 4.3. APLICAȚIA ANDROID PENTRU ADMINISTRATOR

### 4.3.1. MODULARIZAREA APLICAȚIEI PENTRU ADMINISTRATOR

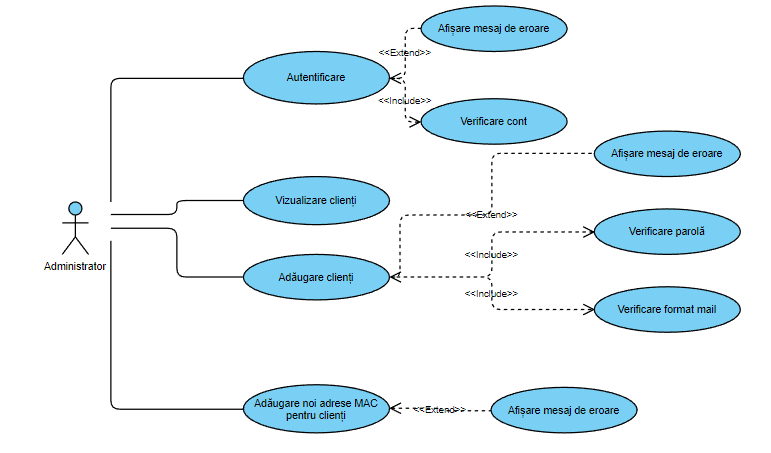


Figura 4.3 - Diagrama use case pentru aplicația administratorului

#### 4.3.1.1. MODULUL DE AUTENTIFICARE

Modulul de autentificare conține două câmpuri: adresa de mail și parola. În cadrul acestui proiect s-a ales să nu existe funcționalitatea de înregistrare pentru administrator. În cazul administratorului, el primește un cont destinat creat din baza de date, iar autentificarea se va realiza cu datele primite.

#### 4.3.1.2. MODULUL DE ÎNREGISTRARE A UNUI CLIENT

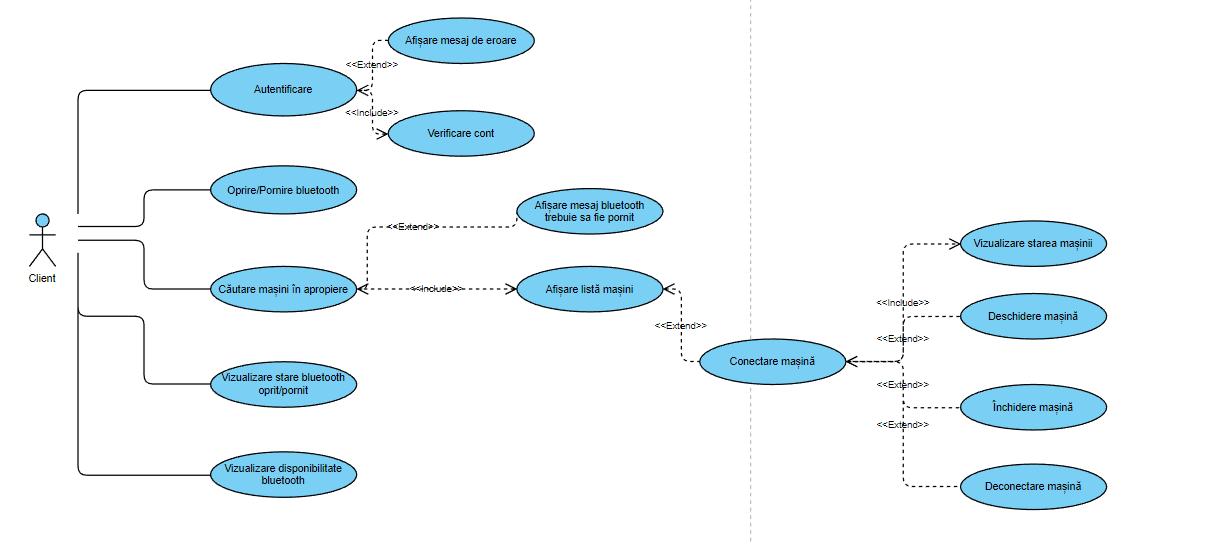
Administratorul are responsabilitatea ca la cererea creării unui cont pentru un nou client, acesta să adauge în baza de date datele noului client: adresa de mail, parola și câmpul destinat adreselor MAC (fiecare adresă este destinată pentru o mașină). În cazul în care parola introdusă în momentul creării continul conține mai putin de șapte caractere va apărea un mesaj de eroare, iar dacă adresa de mail aleasă pentru crearea contului nu are un format valid undei adrese de mail va fi afișat un mesaj corespunzător. La apariția celor două erori menționate anterior, înregistrarea unui nou client nu va fi efectuată.

#### 4.3.1.3. MODULUL DE ADĂUGARE DE NOI MAȘINI PENTRU CLIENȚI

Scopul funcționalității de a adăuga noi mașini pentru clienți vine din nevoia de a putea gestiona mai multe mașini folosind același cont de pe o singură aplicație din telefonul personal. Astfel un client care deține mai multe mașini, le poate controla pe toate folosind un sigur cont, fără a fi nevoit să se înregistreze cu credențiale diferite de fiecare dată. Administratorul va introduce atât adresa MAC necesară conectării la Arduino, cât și modelul mașinii.

## 4.4. APLICAȚIA ANDROID PENTRU CLIENT

### 4.4.1. MODULARIZAREA APLICAȚIEI PENTRU CLIENT

Figura 4.4 – Diagrama use case pentru aplicația clientului

#### 4.4.1.1. MODULUL DE AUTENTIFICARE

Autentificarea în cazul clientului se realizează prin introducerea adresei de mail și parolei în momentul pornirii aplicației. La fel ca la administrator, nu există un formular de înregistrare destinat clientului, deoarece în cazul solicitării folosirii aplicației clientul va primi un cont destinat creat de către administrator în care se regăsește adresa MAC unică mașinii necesară clientului de a se conecta la mașină.

#### 4.4.1.2. MODULUL DE PORNIRE ȘI OPRIRE A BLUETOOTH-ULUI

Clientul are la dispoziție în cadrul aplicației Android posibilitatea de a opri și porni bluetooth-ul telefonului. Aceste funcționalități sunt obersevate prin intermediul unei poze descriptive, dar și prin afișarea unui text corespunzător. La pornirea aplicației în activitatea principală există un mesaj sugestiv pentru client care îi transmite dacă telefonul suportă sau nu bluetooth.

#### 4.4.1.3. MODULUL DE CĂUTARE ȘI CONECTARE A MAȘINII

Comunicarea dintre cele două aplicații: cea Android destinată clientului și cea Arduino începe în momentul căutării dispozitivelor Bluetooth din apropiere care reprezintă mașinile clientului. Clientul se poate conecta în funcție de dorințele sale la mașina aleasă.

Lista destinată mașinilor clientului presupune căutarea prin dispozitivele bluetooth din apropiere și selectând adresele MAC. Aceste adrese MAC găsite se verifică să coincidă cu cele din câmpul adreselor MAC din baza de date, asftel clientul se poate conecta doar la mașinile sale.

#### 4.4.1.4. MODULUL DE ÎNCHIDERE ȘI DESCHIDERE A MAȘINII

După ce conectarea a fost făcută cu succes, clientul vede starea mașinii sale prin o imagine descriptivă cu o mașină împreună cu un lacăt care poate fi deschis sau închis. Dacă lacătul este deschis, clientul întelege că mașina sa este descuiată, iar dacă lacătul este închis înțelege că mașina este încuiată. Clientul poate comanda mașina prin intermediul a unui buton care are starea opusă față de starea în care se află mașina. În cazul în care mașina este descuiată, clientul o poate încuia cu ajutorul butonului, iar o dată cu acționarea butonului, acesta își va schimba starea.

În momentul apăsării butonului ce prezintă starea de închidere se trimite un mesaj pe 8 biți către Arduino care va aprinde ledul, iar la rândul său, Arduino va trimite un mesaj tot pe 8 biți înapoi către Android pentru a-l anunța să schimbe starea butonului în deschidere și imaginea reprezentativă acestei stări. Funcționalitatea de deschidere a mașinii este caracterizată la fel ca cea de închidere. Android trimite un mesaj pe 8 biți către Arduino, Arduino va stinge ledul și va trimite un mesaj pe 8 biți înapoi către Android pentru a schimba starea butonului și cea a imaginii.

#### 4.4.1.5. MODULUL DE DECONECTARE A APLICAȚIEI DE LA MASINĂ

Clientul are la dispoziție și posibilitatea de a deconecta aplicația Android de la mașină în cazul în care acesta nu mai dorește să aibe la dispoziție funcționalitățile de închidere și de deschidere a mașinii.

## 4.5. RELAȚIA DINTRE APLICAȚIA ANDROID PENTRU CLIENT ȘI APLICAȚIA ARDUINO

A close-up of a circuit board

Description automatically generated with medium confidence

Figura 4.5 – Relația dintre Android și Arduino

Comunicarea dintre aplicația Android și aplicația Arduino constă în cinci mesaje pe 8 biți și se realizează prin bluetooth. Aplicația Android trimite către Arduino mesajele: GET\_STATUS, LOCK și UNLOCK. Prin GET\_STATUS acesta cere informații despre statusul mașinii și le primește de la aplicația Arduino prin STATUS\_LOCK și STATUS\_UNLOCK. Prin LOCK și UNLOCK se trimite către Arduino comanda de a aprinde sau stinge led-ul, apoi aplicația Android primeste mesajele de STATUS\_LOCK și STATUS\_UNLOCK pentru a schimba starea butonului și a imaginii descriptive. Cele două mesaje de status sunt primite de aplicația clientului și în momentul în care atingerea senzorului tactil declanșeaza activarea sau dezactivarea ledului, fapt care schimbă starea mașinii. Mesajul de GET\_STATUS se trimite în momentul conectării mașinii și când se trimit mesajele de LOCK și UNLOCK.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NUME | SURSĂ | DESTINAȚIE | COMANDĂ | PAYLOAD |
| GET\_STATUS | 01 | 10 | 01 | 00 |
| LOCK | 01 | 10 | 10 | 00 |
| UNLOCK | 01 | 10 | 11 | 00 |
| STATUS\_LOCK | 10 | 01 | 00 | 10 |
| STATUS\_UNLOCK | 10 | 01 | 00 | 11 |

Tabel 4.1 – Structura mesajelor dintre Android și Arduino

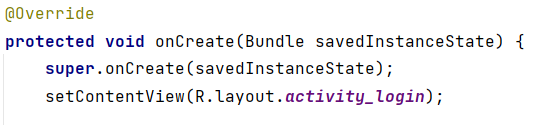
Mesajele GET\_STATUS, LOCK și UNLOCK au ca sursă 01 – Android, destinație 10 – Arduino și diferă prin intermediul celor doi biți ai comenzii.

Mesajele STATUS\_LOCK și STATUS\_UNLOCK au ca sursă 10 – Arduino, destinație 01 – Android și payload-ul oferă informația pentru a știi la care dintre cele două se face referire.

# IMPLEMENTAREA APLICAȚIEI

## 5.1. APLICAȚIA ANDROID

În cadrul ambelor aplicații Android dezvoltate s-a construit interfața grafică cu utilizatorul prin intermediul unui fișier .xml. Fiecare fișier .xml se află în strânsă legătură cu o activitate, iar conexiunea dintre acestea se găsește în interiorul metodei onCreate, apelată la crearea activității, prin metoda setContentView care încarcă activitatea cu elementele aflate în interiorul fișierului .xml.



Secvența de cod 5.1 – Încărcarea elementelor din interfața grafică



Secvența de cod 5.2 – Exemplu fișier .xml

Pentru a implementa interfața grafică s-au folosit mai multe elemente cum ar fi: ImageView, TextView, Button, ListView, RecyclerView grupate în interiorul unui părinte LinearLayout, astfel sunt inserate în ordine cu o orientare verficală.

Fiecare element prezintă un atribut id obligatoriu precizat pentru a putea fi folosit în cod după încărcarea fișierului .xml la crearea activității. Astfel, prin intermediul metodei findViewById se poate accesa referința către elementul dorit.



Secvența de cod 5.3 – Accesarea elementelor grafice în interiorul unei activități

### 5.1.1. APLICAȚIA ANDROID PENTRU ADMINISTRATOR

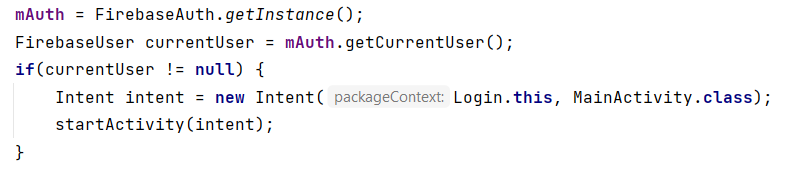
#### 5.1.1.1. SISTEMUL DE AUTENTIFICARE

În cadrul aplicației pentru administrator, funcționalitatea de înregistrare nu este necesara, deoarece acesta va primi o adresă de mail și o parolă pentru a se putea autentifica. Componentele de tip EditText vor fi transformate în variabile de tip String pentru a putea fi trimise ca argumente pentru metoda predefinită singInWithEmailAndPassword oferită de Firebase Authentication. Verificarea celor două variabile s-a făcut înainte de metoda predefinită, astfel dacă datele nu sunt goale și există în Firebase Authentication, autetificarea se va face cu succes.



Secvența de cod 5.4 – Autentificarea administratorului

În cazul în care administratorul nu a folosit funcționalitatea de a ieși din cont, la crearea activității de autentificare se verifică dacă există un utilizator al aplicației autentificat (în cazul de față administratorul). Dacă există, el nu va mai trebui să se autentifice și va fi redirecționat prin intermediul unui intent pe activitatea principală.



Secvența de cod 5.5 – Verificare dacă administratorul este autentificat

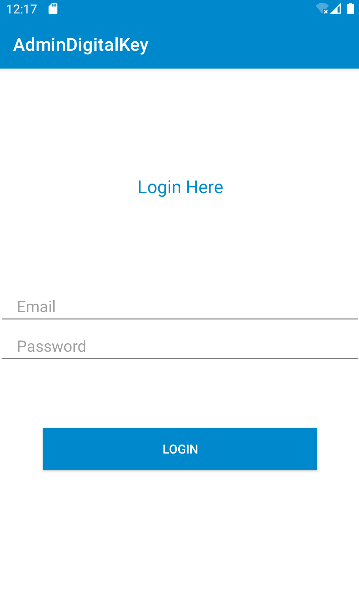


Figura 5.1 – Ecranul de autentificare pentru administratorul aplicației

#### 5.1.1.2. SISTEMUL DE ÎNREGISTRARE A UNUI NOU CLIENT

Deoarece clientul nu are posibilitatea de a intra în aplicație și a-și crea propriul cont, responsabilitatea este a administratorului de a face acest lucru pentru client, adăugând datele corecte.

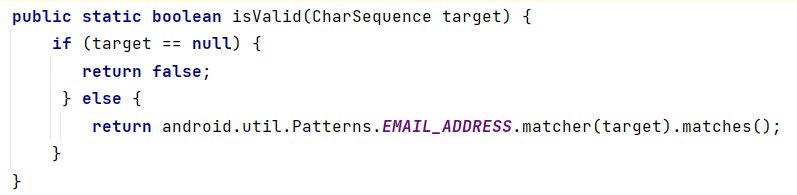
În vederea realizării acestei funcționalități înainte de a se apela metoda predefinită din Firebase Authentication s-au luat în considerare următoarele validări ale datelor unui client pentru a facilita responsabilitatea administratorului de a introduce datele corect: adresa de mail să aibe o structura corespunzătoare, parola să aibe minim șase caractere și niciun câmp să nu fie lăsat gol.

În cadrul metodei de autentificare: createUserWithEmailAndPasswrod, textul introdus în câmpurile destinate atât pentru adresa de mail cât și pentru parolă se salvează în variable de tip String și se trimite ca parametrii metodei. În cazul în care s-a creat cu succes un cont pentru autentificare, administratorul folosește id-ul unic al fiecărui client pentru a salva pe lânga adresa de mail și parolă, adresa MAC a mașinii și modelul mașinii respective într-o bază de date Realtime Database Firebase. Acest lucru este necesar deoarece clientul se va folosi de adresa MAC din baza de date pentru a se putea conecta la mașină.



Secvența de cod 5.6 – Înregistrarea unui nou client

Validarea pentru adresa de mail s-a realizat cu ajutorul unei expresii regulate predefinite.



Secvența de cod 5.7 – Validarea adresei de mail a unui client

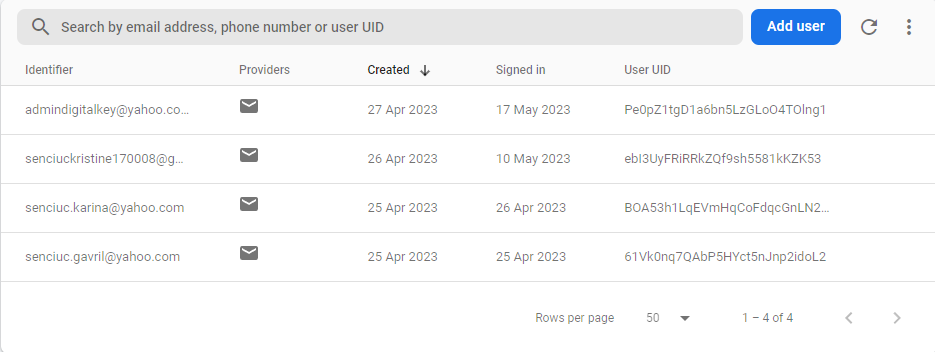


Figura 5.2 – Firebase Authentication pentru administrator și pentru clienți

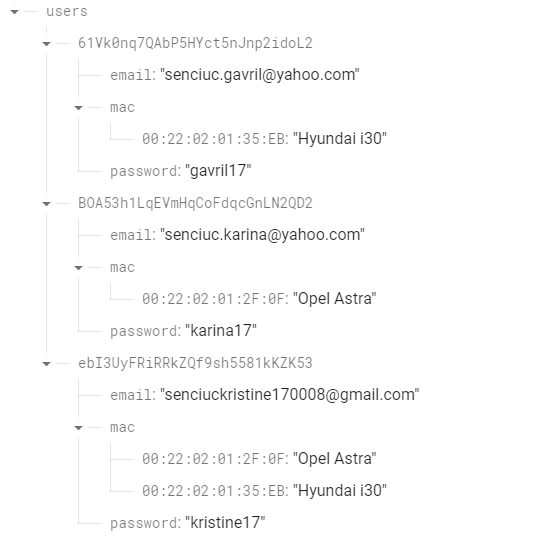


Figura 5.3 – Structura bazei de date pentru clienții adăugați de administrator

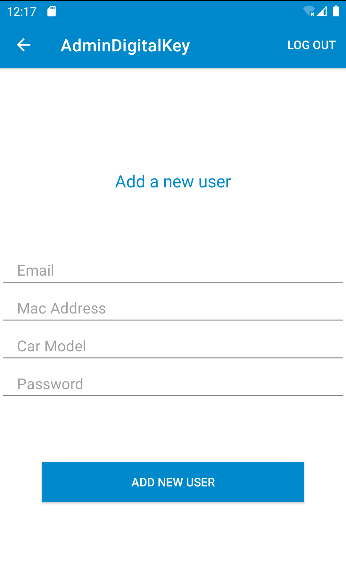


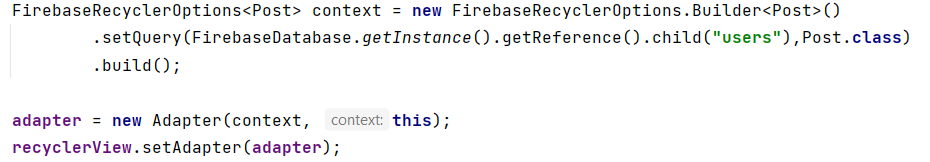
Figura 5.4 – Ecranul de înregistrare a unui nou client

#### 5.1.1.3. SISTEMUL DE VIZUALIZARE AL CLIENȚILOR

Sistemul de vizualizare al clienților destinat administratorului a fost implementat pentru o mai bună gestionare, astfel administraotul îi poate vedea dar și actualiza în funcție de cerințe lista adreselor MAC. S-a ales folosirea elementului grafic RecyclerView pentru a putea vizualiza lista clienților care conține adresa de mail și parolă și un buton care afișează lista adreselor MAC împreună cu posibilitatea adăugării unor noi adrese.

În cadrul clasei Post s-au definit variabilele folosite și în baza de date: adresa de mail și parola, și s-au creat metodele de returnare ale acestora.

Firebase pune la dispoziție FirebaseRecyclerOptions care face legătura dintre elementele bazei de date și cele din clasa Post. Fiecare element din lista aflat în RecyclerView se numește ViewHolder și este implementat în interiorul clasei Adapter care extinde FirebaseRecyclerAdapter și care face legătura dintre date și RecyclerView [19].



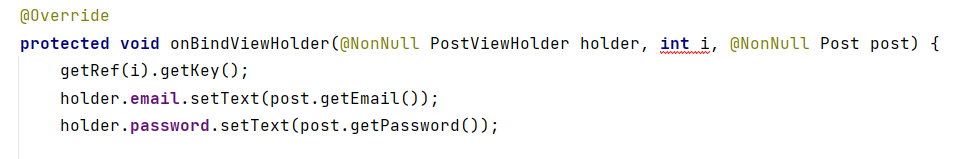
Secvența de cod 5.8 – Legătura dintre baza de date și clasa Post

Metoda onCreateViewHolder se apelează de fiecare dată când apare un element nou în listă pe care îl crează și îl inițializează [19].



Secvența de cod 5.9 – clasa ce reprezintă ViewHolder și crearea acestuia

Pentru a putea completa câmpurile din fiecare ViewHolder cu elementele dorinte RecyclerView apelează metoda onBindViewHolder care atribuie elementelor din interiorul ViewHolder-ului elementele luate din baza de date [19].



Secvența de cod 5.10 – metoda care completează datele ViewHolder-ului

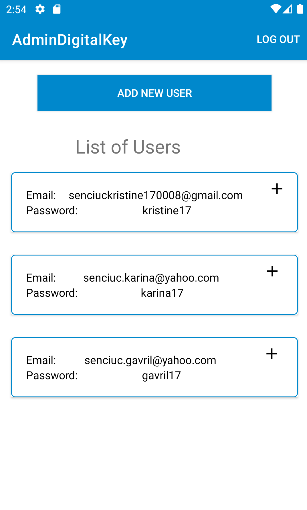
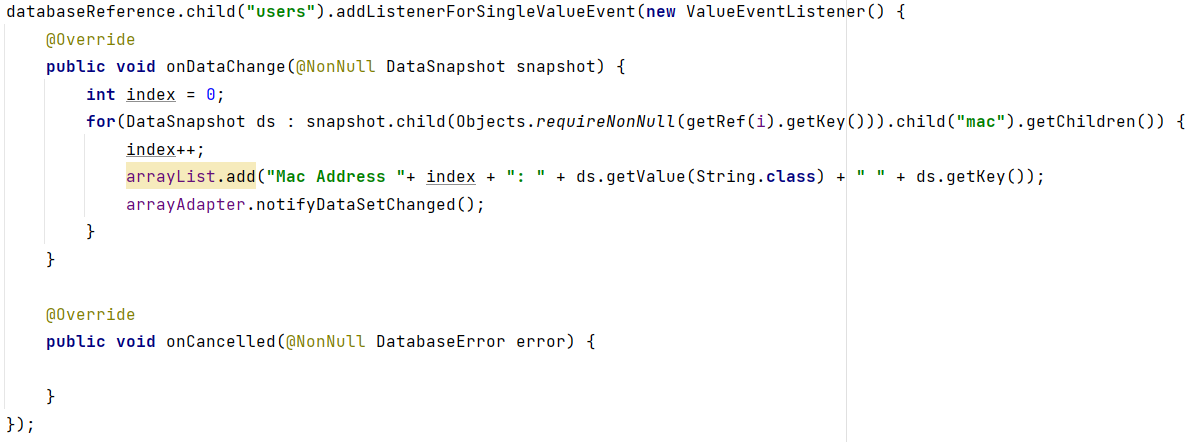


Figura 5.5 – Ecranul pentru vizualizarea clienților

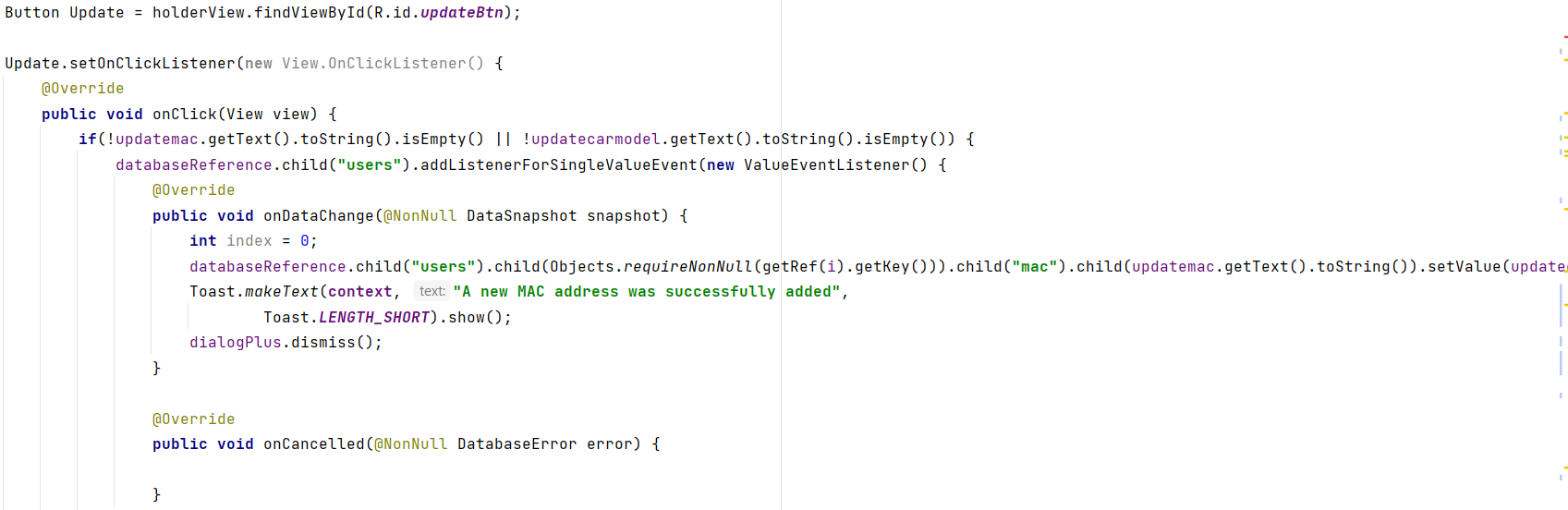
#### 5.1.1.4. SISTEMUL DE ADĂUGARE DE NOI MAȘINI PENTRU CLIENȚI

În interiorul fiecărui ViewHolder din RecyclerView se află un element de tip ImageView care în momentul apăsării acestuia se deschide o nouă fereastră unde se pot vizualiza adresele mac ale clientului dorit, salvate într-o listă, și se pot adăuga adrese noi.



Secvența de cod – 5.11 crearea listei adreselor MAC

Pentru a adăuga noi adrese MAC se completează două câmpuri: adresa MAC și modelul mașinii care nu pot fi lăsate goale. După acționarea butonui de actualizare se salvează noile date introduse în baza de date prin intermediul referinței databaseReference care indică prin metoda child() locația dorită și se revine la lista clienților.



Secvența de cod 5.12 – Adăugarea unei noi adrese MAC

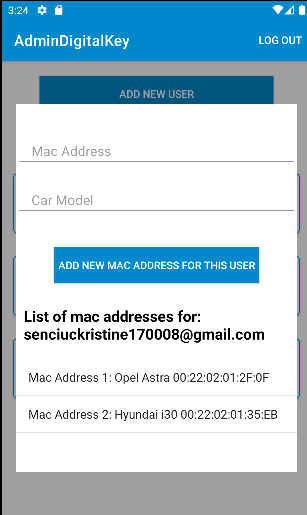
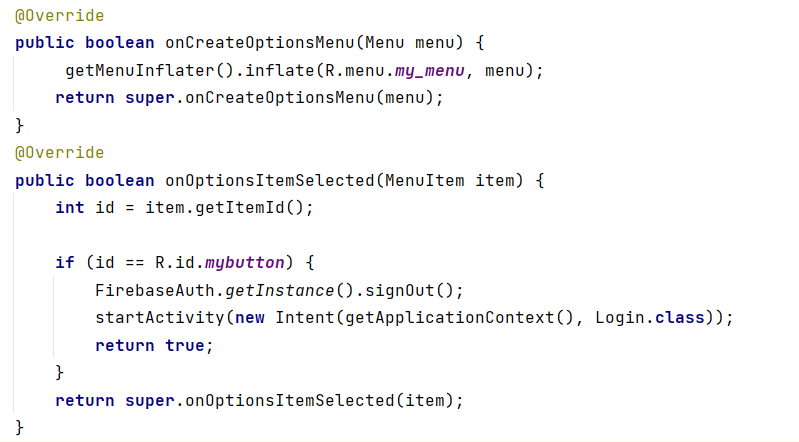


Figura 5.6 – Ecranul pentru vizualizarea adreselor MAC și adăugarea unei noi adrese MAC

#### 5.1.1.5. SISTEMUL DE DECONECTARE

Ca administratorul să poată facilita și de funcționarea de deconectare de la aplicație, aceasta s-a implementat prin suprascrierea a două metode: onCreateOptionsMenu care crează și initializează Meniul cu elementul de tip item care are ca și text: Log Out, și metoda onOptionsItemSelected care dacă gasește id-ul item-ulul creat va apela metoda de deconectare oferită de Firebase și va porni activitatea de Autentificare.

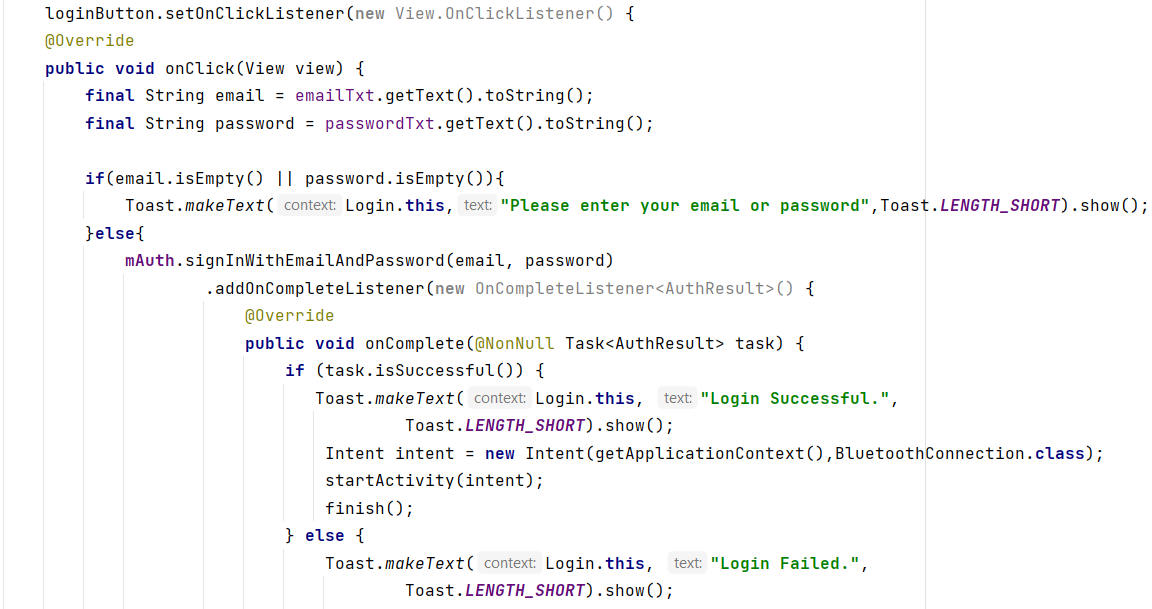


Secvența de cod 5.13 – Deconectarea administratorului

### 5.1.2. APLICAȚIA ANDROID PENTRU CLIENT

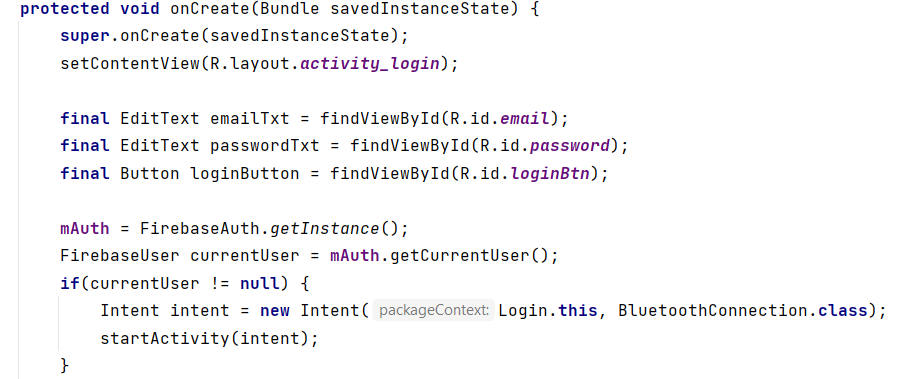
#### 5.1.2.1. SISTEMUL DE AUTENTIFICARE

În cazul clientului nu există posibilitatea de a se înregistra singur în aplicație, astfel clientul se poate autentifica după ce administratorul a creat un cont cu datele specifice. Înainte de autentificare se verifică ca datele introduse să fie completate. Dacă verificarea a fost realizată cu succes se utilizează o metodă din API-ul Firebase signInWithEmailAndPassword care are ca parametrii adresa de mail și parola introduse. În cazul în care există cu cont cu datele specificate în Firebase Authentication se va afișa un mesaj de succes, iar prin intermediul unui intent se va trece de la activitatea de autentificare la activitatea principală. În cazul în care nu există se va afișa un mesaj corespunzător.



Secvența de cod 5.12 – Autentificarea clientului

La crearea activității de autentificare se verifică prin intermediul metodei getCurrentUser care se apelează pe o instanță a clasei FirebaseAuth dacă există un client deja autentificat sau nu. Dacă există, acesta va fi redirecționat pe activitatea principală, iar în caz contrar rămâne pe activitatea de autentificare.



Secvența de cod 5.14 – Verificare client conectat

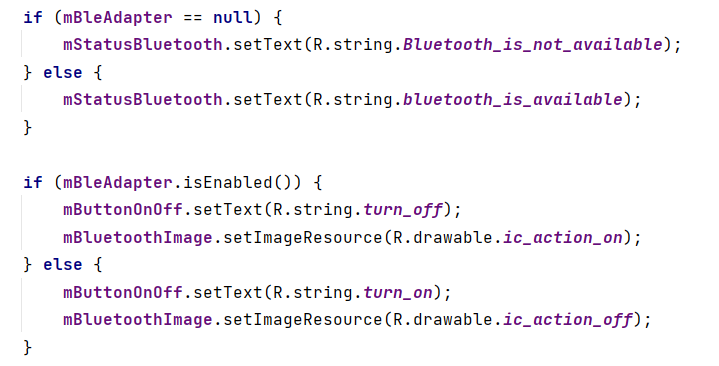
A screenshot of a login screen

Description automatically generated with medium confidence

Figura 5.7 – Ecranul de autentificare al clientului

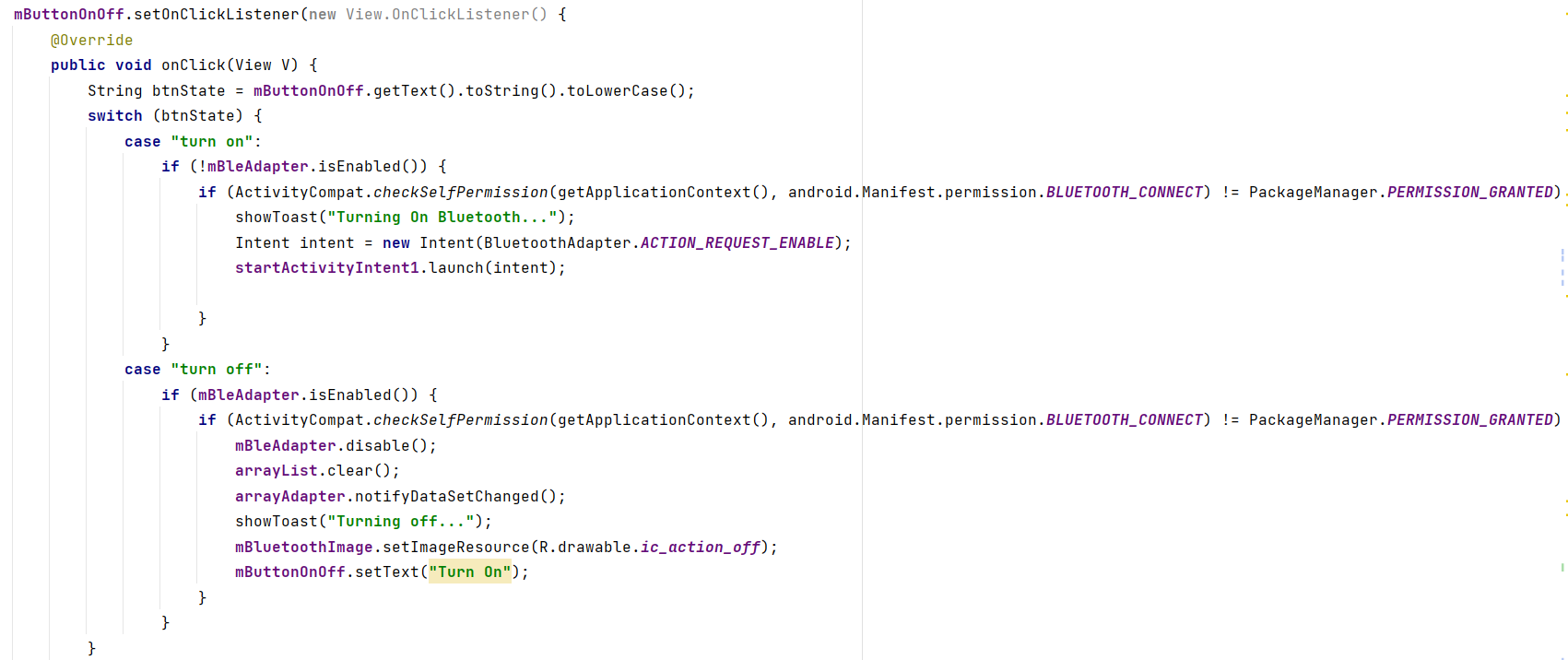
#### 5.1.2.2. SISTEMUL DE PORNIRE ȘI OPRIRE BLUETOOTH

La crearea activității principale se folosește mBleAdapter = BluetoothAdapter.getDefaultAdapter() care reprezintă un obiect al clasei BluetoothAdapter și este folosit pentru a obține bluetooth-ul dispozitivului curent. Prin verificarea acestuia cu null se dorește ca utilizatorul aplicației să fie informat dacă dispozitivul de pe care a pornit aplicația suportă sau nu bluetooth. Următoarele verificări au fost folosite pentru a arăta utilizatorului o imagine și un text descriptiv în cazul în care bluetooth-ul este pornit sau oprit.



Secvența de cod 5.15 – Starea bluetooth-ului la crearea activității

Există un singur buton ce prezintă două funcționalități: de pornire și se oprire a bluetoothului. În funcție de textul afișsat pe buton se alege una dintre cele două funcționalități. Dacă se apasă butonul cu textul: turn on înseamnă ca starea bluetooth-ului dispozitivului este de oprire, iar pentru a îl porni se trimite un intent pentru a solicita clientului aplicației să acorde sau nu permisiunea. Dacă permisiunea este primită, bluetoothul va fi pornit, iar butonul iși va schimba textul în: turn off. Pentru funcționalitatea de oprire nu se solicită clientului permisiunea. În momentul în care este apăsat butonul se apelează pe obiectul mBleAdapter metoda disable, iar lista mașinilor se golește deoarece nu este posibilă conectarea la alt dispozitiv bluetooth.



Secvența de cod 5.16 – Butonul de oprire sau pornire a bluetooth-ului

A screenshot of a bluetooth

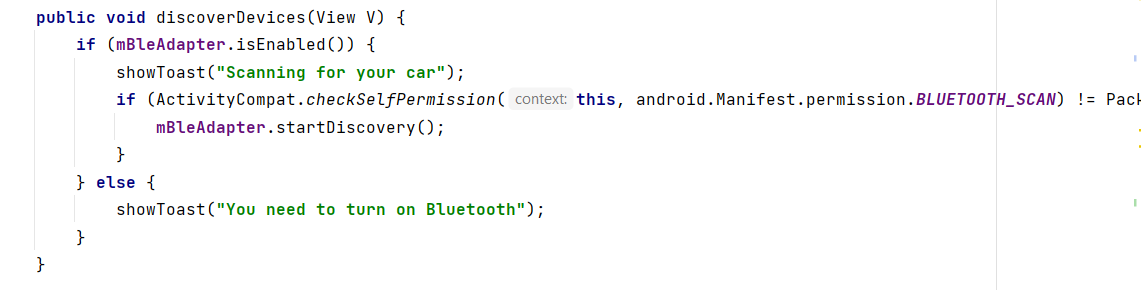
Description automatically generated with low confidenceA picture containing text, screenshot, font, design

Description automatically generated

Figura 5.8 – Ecranul principal al aplicației clientului

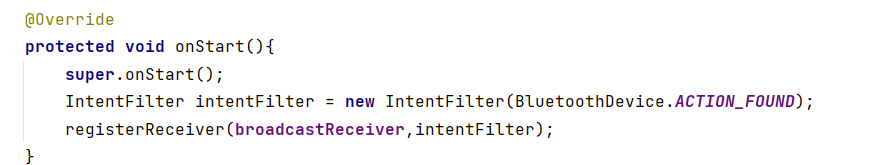
#### 5.1.2.3. SISTEMUL DE CĂUTARE ȘI AFIȘARE A MAȘINILOR

Căutarea mașinilor începe o dată cu acționarea butonului de căutare în care se verifică ca bluetooth-ul dispozitibului să fie activat și se apelează metoda care caută dispozitive bluetooth în apropiere, în caz contrar se afișează un mesaj corespunzător.



Secvența de cod 5.17 – Căutarea mașinilor

În interiorul metodei onStart se înregistrează un BroadcastReceiver pentru a putea primi și trata evenimente specificate de IntentFileter: când s-a găsit un dispozitiv bluetooth în apropiere.



Secvența de cod 5.18 – Înregistrarea BroadcastReceiver

În cadrul metodei onReceive este implementată logica pentru găsirea unui dispozivit bluetooth în apropiere. Acest dispozitiv este preluat din intent prin cheia BluetoothDevice.EXTRA\_DEVICE. Se verifică că dacă adresa MAC a dispozitivului găsit coincide cu o adresă MAC a utilizatorului, această adresă impreună cu modelul mașinii vor fi salvate într-o listă. De fiecare dată când un nou dispozitiv va fi descoperit se va folosi aceeași logică.



Secvența de cod 5.19 – Găsirea dispozitivelor din apropiere

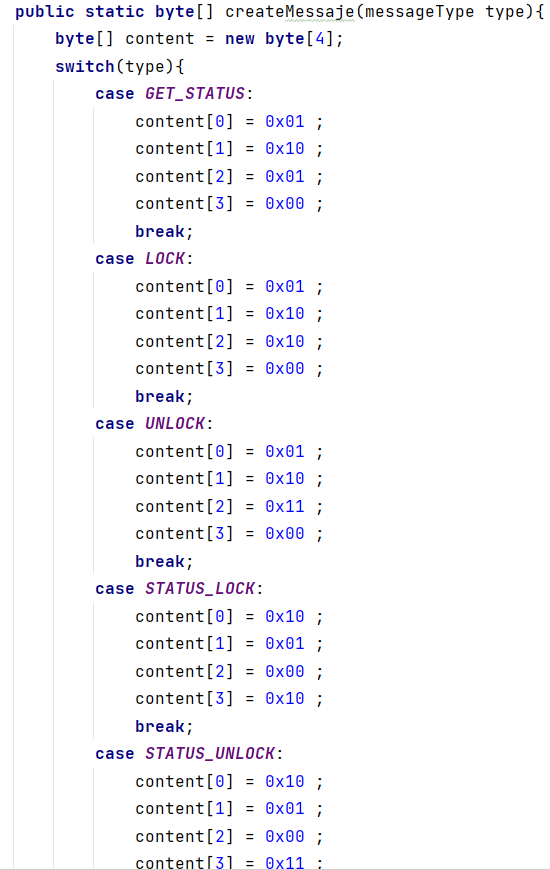
Clientul se poate conecta la oricare dispozitiv apărut în listă, astfel în momentul apăsării elementului din listă se va trimite către activitatea de conectare prin intermediul metodei putExtra apelată pe intent adresa MAC și va începe activitatea de conectare.



Secvența de cod 5.20 – Începerea conectării

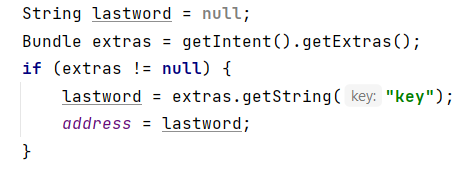
#### 5.1.2.4. SISTEMUL DE CONECTARE A MAȘINII

În interiorul clasei destinate mesajelor s-au definit cele cinci mesaje de tipul byte array. Acestea vor fi folosite în comunicarea dintre telefon și mașină.



Secvența de cod 5.21- Structura mesajelor

Adresa MAC a mașinii la care se va conecta telefonul este preluată din activitatea precedentă din lista mașinilor disponibile conectării.



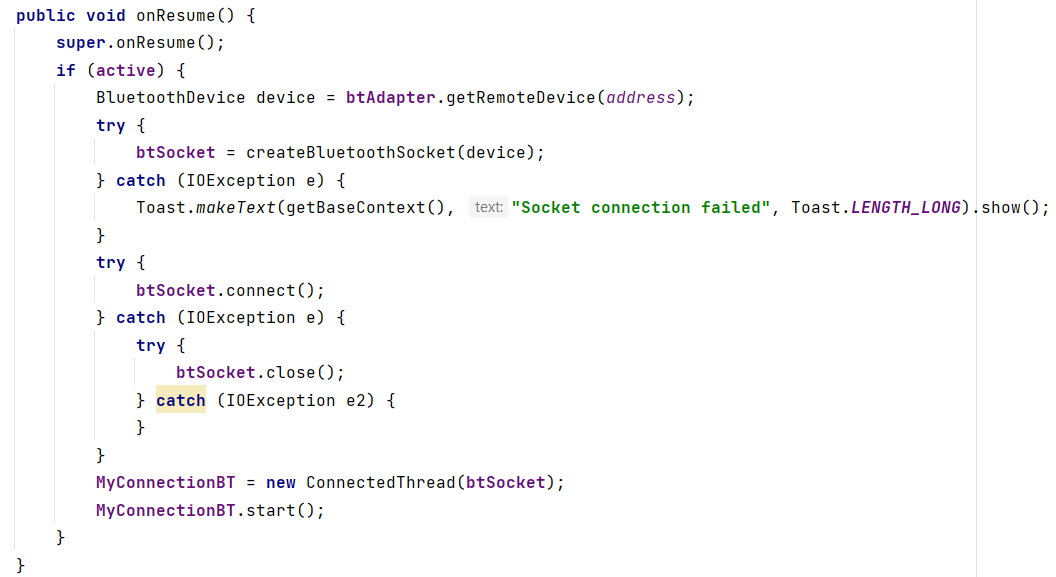
Secvența de cod 5.22 – Preluarea adresei MAC

Conectarea la mașină începe în momentul acționării butonului de conectare. Acesta devine invizibil, iar butonul de control al deschiderii/închiderii, butonul de deconectare și imaginea descriptivă a stării mașinii devin vizibile. Prin setarea variabilei active pe adevărat și a apelării metodei onResume se realizează conectarea. După acestea se trimite prim metoda de write către Arduino (mașină) mesajul de GET\_STATUS ca mai apoi aplicația Android să primească un mesaj referitor la starea mașinii și să afișeze butoanele și imaginea corespunzătoare următoarei acțiuni posibile. Pentru a putea trimite mesajul acesta a trebuit să treacă printr-o transformare de la un byte array la String.



Secvența de cod 5.23 – Butonul de conectare

În cadrul funcției ce se apelează în momentul interacțiunii clientului cu aceasta se apelează metoda createBluetoothSocket(device) care crează un obiect de tipul BluetoothSocket ca mai apoi prin metoda connect să stabilească o conexiune cu dispozitivul a cărei adrese a fost preluată din activitatea precedentă. După stabilirea conexiunii, se ponește firul de execuție care va realiza comunicarea dintre cele două dispozitive bluetooth.



Secvența de cod 5.24 – Crearea conexiuni cu mașina

Pentru a gestiona comunicarea s-a implementat metoda run care va rula atâta timp cât există o conexiune bluetooth între cele două dispozitive. Prin metoda read() se citesc datele primite și se salvează intr-un buffer,iar în momentul în care un caracter citit este \n indică faptul că s-a primit un mesaj complet și poate fi procesat. Se crează un obiect String cu datele primite și se trimite mai departe printr-un handler bluetoothIn. Metoda de write() face posibilă scrierea către Arduino.



Secvența de cod 5.25 – Comunicarea după stabilirea conexiunii

A screenshot of a bluetooth device

Description automatically generated with low confidenceA black and white image of a car

Description automatically generated with low confidenceA screenshot of a phone

Description automatically generated with medium confidence

Figura 5.9 – Ecranele de la selectarea mașinii până la conectarea acesteia

#### 5.1.2.5. SISTEMUL DE ÎNCHIDERE ȘI DESCHIDERE A MAȘINII

Prin intermediul butonului de închidere/deschidere a mașinii se realizează această funcționalitate. Starea butonului este dată de primirea mesajului de status de la Arduino. Dacă mesajul este de STATUS\_LOCK pe buton va fi afișat: unlock the car, iar acționarea acestuia va trimite un mesaj prin metoda write() către Arduino de UNLOCK. Dacă mesajul este de STATUS\_UNLOCK pe buton va fi afișat: lock the car și prin folosirea sa se va trimite prin metoda write() mesajul de LOCK către Arduino. La final în ambele cazuri ale stării butonului, se va trimite un mesaj prin metoda write() de GET\_STATUS, deoarece acționarea butonului schimbă starea mașinii, iar aplicația Android trebuie înștiințată pentru a schimba și starea butonului și imaginea descriptivă a mașinii.

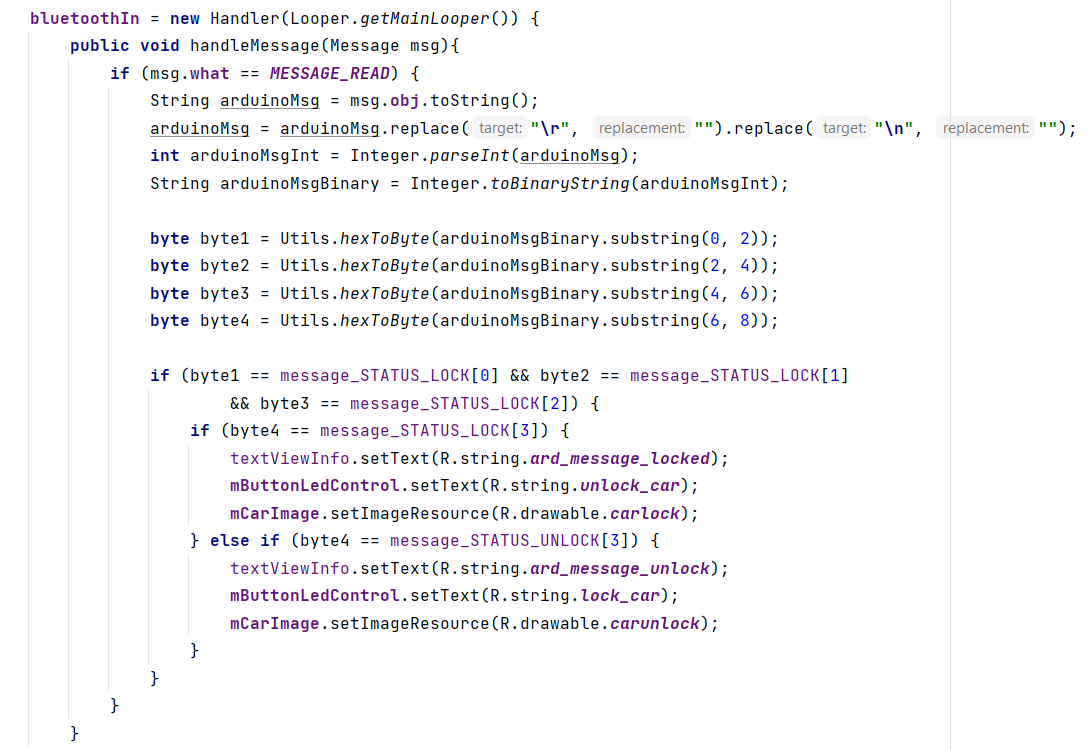
A picture containing text, screenshot

Description automatically generated

Secvența de cod 5.26 – Butonul de deschidere/închidere a mașinii

#### 5.1.2.6. SISTEMUL DE VIZUALIZARE A STĂRII MAȘINII

Comunicarea dintre cele două dispositive Bluetooth este bidirecțională. Astfel în momentul primirii mesajului de status de la Arduino starea mașinii se schimbă. Mesajul primit de la Arduino fiind de tip String a trebuit modificat ca Android sa îl ințeleagă și să verifice dacă corespunde cu mesajul de tip byte array. Există două cazuri: a primit mesajul de STATUS\_UNLOCK modificând textul, starea butonului în: lock the car și imaginea reprezentând o mașină descuiată, sau a primit mesajul de STATUS\_LOCK și va modifica textul, starea în care se află butonul cu: unlock the car și imaginea din mașină descuiată în mașină încuiată.



Secvența de cod 5.27 – Interpretarea mesajelor primite de Android

A screenshot of a phone

Description automatically generated with medium confidenceA screenshot of a phone

Description automatically generated with medium confidence

Figura 5.10 – Ecrane representative funcționalității de închidere/deschidere

#### 5.1.2.7. SISTEMUL DE DECONECTARE A MAȘINII

Există două posibilități ca clientul să oprească conexiunea dintre telefon și mașină: în momentul acționării butonului de deconectare sau în momentul în care părăsește activitatea.

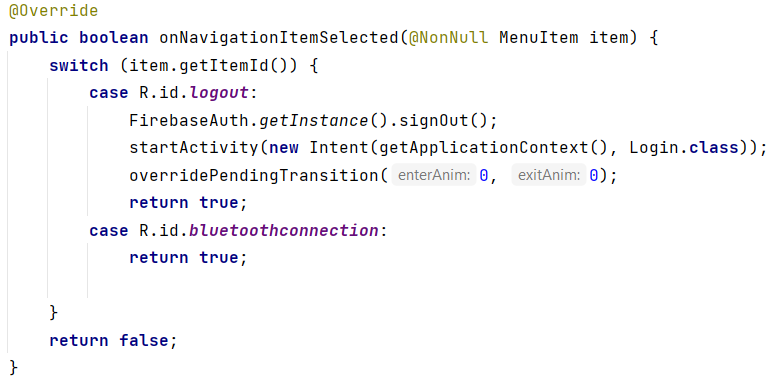
A picture containing text, screenshot

Description automatically generated

Secvența de cod 5.28 – Oprirea conexiunii

#### 5.1.2.8. SISTEMUL DE DECONECTARE A CLIENTULUI

Deconectarea clientului se poate realiza prin acționarea elementului destinat din cadrul meniului din subsolul aplicației, astfel se apelează metoda oferită de Firebase și prin intermediul unu intent se pornește activitatea de autentificare.



Secvența de cod 5.29 – Deconectarea clientului

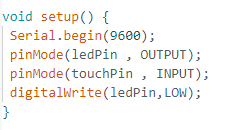
## 5.2. APLICAȚIA ARDUINO

În vederea implementării aplicației Arduino s-a început cu definirea pinilor utilizați pentru senzorul tactil și pentru led, cu inițializarea variabilelor folosite și cu definirea mesajelor ce facilitează comunicarea între Arduino și Android.



Secvența de cod 5.30 – Inițializarea variabilelor și definirea mesajelor

În cadrul funcției de setup() al programului care se apelează o sigură dată la încărcarea codului pe placă s-a apelat Serial.begin(9600) pentru a facilita comunicarea serială prin bluetooth cu un dispozitiv extern. În interiroul acesteia s-a specificat și modul de funcționare pentru led și pentru senzotul tactil, ledul este configurat ca pin de ieșire, iar senzorul tactil ca pin de intrare, apoi s-a scris valoarea digitală LOW pentru led.



Secvența de cod 5.31 – Funcția setup

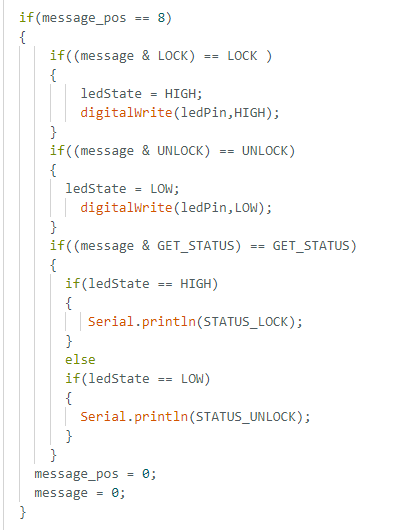
În cadrul funcției loop() care se execută atâta timp cât placa este conectată la o sursă s-a implementat funcționalitatea senzorului tactil care poate fi acționat doar dacă a trecut mai mult de o secundă de la ultima acționare. Dacă senzorul a fost atins, se citește starea acestuia prin digitalRead, iar timpul dintre acționări este mai mare de o secundă, se verifică starea led-ului. Dacă starea este setată pe HIGH în momentul atingerii senzorului, acesta setează starea pe LOW și trimite prin intermediul comunicării seriale Serial.println() mesajul de STATUS\_UNLOCK către aplicația Android. Altfel, dacă starea este setată pe LOW, se va salva noua stare de HIGH și se va trimite tot prin Serial.println() mesajul de STATUS\_LOCK către Android. Având noua stare a ledului, aceasta va fi transmisă către led prin digitalWrite().



Secvența de cod 5.32 – Funcționalitatea senzorului tactil

În interiorul buclei se verifică dacă există date pentru citire. Aceste date prin comunicarea serială se transmit bit cu bit și se salvează într-o variabilă. În variabila bitRead se reține data citită sub formă de int. Se pot citi maxim 8 biți deoarece mesajele de comunicare dintre Android și Arduino sunt pe 8 biți. Cu fiecare citire se formează mesajul.

În momentul în care mesajul este format din 8 caractere, el este complet și poate fi interpretat. Dacă s-a citit un mesaj de tip LOCK, se salvează starea ledului într-o variabilă și se scrie valoarea digitală HIGH pentru a aprinde ledul. Dacă s-a primit un mesaj de tipul UNLOCK, se salvează starea ledului ca fiind LOW și se stinge ledul prin digitalWrite(). Cu fiecare schimbare a stării Android trimite un mesaj de GET\_STATUS pentru a primi un răspuns din partea Arduino. Astfel, daca s-a citit GET\_STATUS există două posibilități: starea ledului este de HIGH și Arduino va trimite înapoi mesajul de STATUS\_LOCK sau starea ledului este de LOW și Arduino va scrie către Android mesajul de STATUS\_UNLOCK. După interpretarea unui mesaj complet primit se resetează conținutul mesajului și capacitatea sa pentru a putea citi în continuare alte mesaje.



Secvența de cod 5.33 – Primirea mesajelor de la Android

# UTILIZAREA ȘI TESTAREA APLICAȚIEI

## 6.1. MANUAL PENTRU UTILIZARE

## 6.2. REZULTATE ÎN URMA TESTĂRII

# CONCLUZII ȘI DIRECȚII VIITOARE DE DEZVOLTARE

## 7.1. CONCLUZII

## 7.2. DIRECȚII DE CONTINUARE A DEZVOLTĂRII

# 8. BIBLIOGRAFIE

[1] Aliferi, C., *Android Programming Cookbook,* 2016.

[2] Singh R., *An Ovewview of Android Operating System and Its Security Features,* 2014.

[3] \*\*\* <https://data-flair.training/blogs/android-application-components/> accesare 11 mai 2023)

[4] \*\*\* <https://developer.android.com/guide/components/activities/tasks-and-back-stack accesare 11 mai 2023>)

[5] \*\*\* <https://www.geeksforgeeks.org/lifecycle-in-android-architecture-components/> accesare 11 mai 2023)

[6] Walls, C., *Open Source, Embedded Linux, and Android,* Embedded Software, 2012.

[7] \*\*\* <https://www.geeksforgeeks.org/broadcast-receiver-in-android-with-example/> accesare 11 mai 2023)

[8] \*\*\* [https://emteria.com/learn/android-manifest-file accesare 11 mai 2023](https://emteria.com/learn/android-manifest-file  accesare 11 mai 2023))

[9] \*\*\* <https://developer.android.com/reference/android/Manifest.permission>

accesare 11 mai 2023)

[10] \*\*\* [https://developer.android.com/reference/android/bluetooth/BluetoothAdapter accesare mai 2023](https://developer.android.com/reference/android/bluetooth/BluetoothAdapter%20accesare%20mai%202023))

[11] \*\*\* <https://www.pangea.ai/dev-mobile-app-resources/the-pros-and-cons-of-android-studio-and-app-tools/> accesare 11 mai 2023)

[12] \*\*\* <http://firebase.google.com/> accesare 12 mai 2023)

[13] M. Popa, „Curs Sisteme Încorporate”, Universitatea Politehnica Timișoara, 2022.

[14] \*\*\* [https://docs.rs-online.com/6560/A700000007742264.pdf accesare 11 mai 2023](https://docs.rs-online.com/6560/A700000007742264.pdf%20%20accesare%2011%20mai%202023))

[15] \*\*\* [https://www.amazon.de/Arduino-Mega-2560-R3-Microcontroller/dp/B0046AMGW0 accesare](https://www.amazon.de/Arduino-Mega-2560-R3-Microcontroller/dp/B0046AMGW0 accesare 11 mai 2023)  accesare 11 mai 2023)

[16] \*\*\* https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet accesare 11 mai 2023)

[17]\*\*\*<https://classes.engineering.wustl.edu/ese205/core/index.php?title=Bluetooth_Module_%28HC-06%29_%2B_Arduino> accesare 11 mai 2023)

[18] \*\*\* https://robotica.md/senzor-capacitiv-ttp223b accesare 11 mai 2023)

[19] \*\*\* [Create dynamic lists with RecyclerView  |  Android Developers](https://developer.android.com/develop/ui/views/layout/recyclerview#:~:text=onCreateViewHolder()%20%3A%20RecyclerView%20calls%20this,been%20bound%20to%20specific%20data.) accesare 17 mai 2023)