UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: INFORMATICĂ

LUCRARE DE LICENȚĂ

COORDONATOR:
Asistent Drd. Ivașcu Todor

ABSOLVENT:
Băguț Andrei Alexandru

TIMIŞOARA
2021

UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: INFORMATICĂ

CHEIE INTELIGENTĂ PENTRU MAȘINĂ - SMART KEY

COORDONATOR:
Asistent Drd. Ivașcu Todor

ABSOLVENT:
Băguț Andrei Alexandru

TIMIŞOARA
2021

Abstract

Nowadays, most of car production companies have developed mobile applications with the purpose of facilitating the interaction between car and its owner. For example, Mercedes developed the Mercedes Me application and BMW launched the My BMW App to the market. Although, the most significant utilities of these applications are being able to locate the car and to park it directly from your phone application, there are some disadvantages too. These include lower availability, the lack of data about the number of kilometers at which it is recommended to change the engine oil and the braking pads. Besides these, in case of companies or owners of many cars from different producers, it shows an economical disadvantages because every single app must be bought. Smart Key is the solution to some of these problems. Its user can use it like a virtual car key to lock/unlock car's doors. In the same time, he can see the data from some sensors in real time. The application notifies the user when he should change the engine oil and braking pads and in case of any errors detected. Although there are a lot of similar applications on the market, Smart Key has much more advantages like larger availability, tracking more data and the economical advantage.

Keywords: Android application, automotive, intelligent system, car's parameters.

Rezumat

În prezent, majoritatea firmelor producătoare de mașini au dezvoltat aplicații mobile cu scopul de a facilita interacțiunea dintre proprietar și automobil. Spre exemplu, Mercedes a dezvoltat aplicația Mercedes Me, iar BMW a lansat pe piață My BMW App. Cele mai importante utilități ale acestora ar fi localizarea mașinii și parcarea acesteia direct din aplicație. Cu toate acestea, astfel de aplicații au o serie de dezavantaje, dintre care enumerăm: disponibilitatea redusă, lipsa datelor despre numărul de kilometri la care este recomandat să se efectueze următorul schimb de ulei și de plăcuțe de frână. Pe lângă acestea, în cazul firmelor sau al proprietarilor de mai multe mașini apare un dezavantaj financiar deoarece fiecare aplicație trebuie cumpărată. Aplicația Smart Key este soluția pentru o parte din problemele aplicațiilor concurente. Utilizatorul o poate folosi ca și o cheie virtuală pentru blocarea/deblocarea ușilor. Totodată, acesta poate să vadă în timp real datele de la anumiți senzori ai mașinii. De asemenea, aplicația sugerează când ar trebui efectuat schimbul de ulei și de plăcuțe de frână și trimite alerte în cazul detectării de erori. Deși există foarte multe aplicații similare cu Smart Key, aplicația dezvoltată are mai multe avantaje decât celelalte aplicații deja existente pe piață. Din aceste avantaje enumerăm: aplicabilitatea mărită, posibilitatea de urmărire a mai multor parametrii ai mașinii, avantajul economic dar și semnalarea imediată a problemelor mașinii.

Cuvinte cheie: aplicație android, automotive, sistem inteligent, parametrii mașinii

Cuprins

Listings

Listă de figuri

Listă de tabele

Capitolul 1

Introducere

1.1 Context

Răspândirea tot mai mare a smartphone-urilor a schimbat într-o bună măsură design-ul sistemelor inteligente. Dezvoltările tehnologice din ultimii ani au dus la crearea unor dispozitive mobile cu caracteristici tehnice, care în trecut au fost folosite doar în arhitecturile calculatoarelor sau a dispozitivelor similare [8]. Odată cu această evoluție apare nevoia de a se monitoriza și controla funcționalitatea anumitor sisteme de la distanță, cu ajutorul telefonului mobil. În prezent, aplicațiile care permit controlul unor sisteme de la distanță se bucură de un număr de utilizatori foarte mare. Printre aceste aplicații se pot număra cele care controlează mașinile industriale, cele care ajută la monitorizarea autovehiculelor sau cele care pot controla aparatele de uz casnic.

Majoritatea firmelor care produc mașini dispun de aplicații cu ajutorul cărora utilizatorii pot bloca/debloca ușa sau, de exemplu, să programeze mașina să fixeze o anumită temperatură pentru a crea un mediu plăcut.

Cu ajutorul acestor aplicații [9] [10], utilizatorul poate seta ca în mașină, la o anumită oră să fie o temperatură specifică. Acest lucru este extrem de util în zilele caniculare de vară sau în diminețile de iarnă când geamurile mașinii sunt înghețate. Datorită aplicației, nivelul de confort crește, iarna, utilizatorul având posibilitatea de a programa autovehiculul să se dezghețe la o anumită oră în mod automat. În cazul lipsei aplicației, proprietarul mașinii stă în frig o perioadă bună de timp pentru a curăța mașina.

Pe lângă creșterea nivelului de confort, aceste aplicații pot salva proprietarii de accidente. Acest lucru se realizează prin alertele trimise în momentul în care presiunea dintr-o roată a scăzut sub nivelul optim stabilit de utilizator sau când sistemul de răcire al mașinii nu mai funcționează și aceasta este pasibilă să ia foc.

Pe lângă aceste mari avantaje, aplicațiile descrise mai sus vin și cu anumite dezavantaje la fel de mari. Spre exemplu, în cadrul unei firme de transport în comun există autobuze care aparțin unor producători diferiți. Astfel, doar unele autobuze ar putea fi monitorizate (cele care se încadrează cerințelor aplicației), acest lucru realizându-se incomod deoarece nu există o aplicație care să poată monitoriza în același timp autovehicule de la producători diferiți.

1.2 Motivație și obiective

Un mare dezavantaj pe care îl au aceste aplicații este faptul că nu oferă date despre anumiți parametri (de exemplu numărul de km la care s-a efectuat schimbul de ulei de motor, respectiv numărul de km la care este sugerat următorul schimb). Un alt dezavantaj al acestor aplicații își face simțită prezența în cazul proprietarilor de mai multe mașini de la firme diferite. În cazul acestora, ei trebuie să aibă acces la mai multe aplicații de la fiecare firmă în parte. În același timp, reprezintă și un dezavantaj din punct de vedere financiar deoarece fiecare aplicație trebuie cumpărată.

Aplicațiile respective sunt creeate special pentru anumite modele de mașini ("BMW Digital Key Plus" [11] creată special pentru modelul electric "BMW iX") sau sunt create pentru mașini mai noi de o anumită perioadă ("MercedesME" [9] este special făcută pentru modelele Mercedes din anul 2016 sau mai noi). Din cauza aceasta, foarte mulți oameni pierd accesul la o aplicație de asistență deoarece nu își pot permite o astfel de mașină.

În România, în data de 31 decembrie 2019, statisticile comisiilor de la "Direcția Regim Permise de Conducere și Înmatriculare a Vehiculelor" arătau că sunt înscrise în circulație peste 8,5 milioane de autovechicule. Dintre acestea, numărul automobilelor care depășesc 10 ani de la prima înregistrare este de aproximativ 78,6% [12], așa cum se poate observa în Figura 1.1.

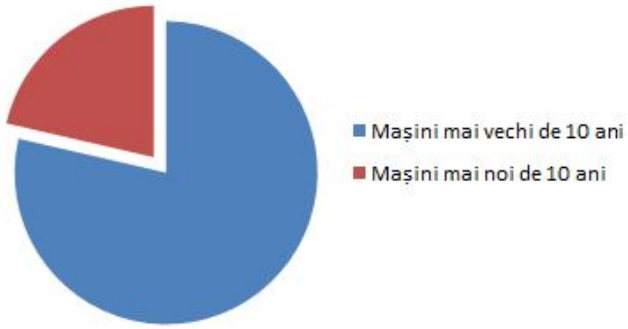


Figura 1.1: Statistică vechime mașini din România.

Având în vedere cele de mai sus, mi-am propus dezvoltarea unei aplicații mobile cu ajutorul căreia proprietarii de mașini își pot bloca/debloca ușile mașinii și pot urmări indicatori despre anumiți parametrii ai mașinii, precum temperatura lichidului de răcire, nivelul combustibilului din rezervor, presiunea cu care intră combustibilul în motor, etc. Pe lângă aceștia, mai este vizibil și numărul de kilometri la care s-a efectuat ultimul schimb de ulei și de plăcuțe de frână, respectiv numărul de kilometri la care ar trebui efectuat următorul. În momentul detectării de erori, aplicația anunță utilizatorul prin notificări. Aplicația se va conecta via Bluetooth la modulul Bluetooth instalat pe mașină. Pentru a putea folosi aplicația, utilizatorul va avea nevoie de un cont personal, unde își va adăuga fiecare mașină în parte. Datele despre mașini vor fi preluate din computerul de bord al mașinii și vor fi afișate pe ecran. Ca și metode de login, aplicația dispune de varianta clasică (introducerea manuală a datelor), dar și de o variată mai rapidă, prin ampretă, în cazul în care dispozitivul utilizatorului dispune

de senzori biometrici.

1.3 Structura lucrării

Lucrarea este structurată pe 5 capitole. Capitolul 2 al lucrării, "Starea de artă și tehnologii folosite", începe cu descrierea unor aplicații care au sisteme de control wireless (secțiunea 2.1). Urmează o scurtă prezentare a domeniului automotive în secțiunea 2.2; apoi, în secțiunea 2.3 sunt prezentate aplicațiile similare (Mercedes Me și My BMW app). Smart key afișează datele de la anumiți senzori ai mașinii care sunt prezentați în secțiunea 2.4. Urmează prezentarea computerului de bord OBD2 (secțiunea 2.5), cel care citește datele de la senzori și le trimite către aplicație. Pentru a vedea cum ajung datele la aplicație, sunt prezentate standardele de comunicare (secțiunea 2.6). La finalul acestui capitol sunt prezentate tehnologiile utilizate în crearea Smart Key, în secțiunea 2.7.

În capitolul 3, "Proiectarea și dezvoltarea aplicației", este prezentată aplicația (secțiunea 3.1) cu arhitectura (secțiunea 3.2) și implementarea (secțiunea 3.3) acesteia. În capitolul 4, în secțiunea 4.1 este prezentată, pe scurt, funcționalitatea aplicației. În secțiunea 4.2 este prezentat setul de date. Urmează prezentarea citirii datelor în timp real (secțiunea 4.3), felul în care aplicația detectează erorile (secțiunea 4.4) și felul în care înștiințează utilizatorul cu privire la schimbul de ulei (secțiunea 4.5) și cel de plăcuțe de frână (secțiunea 4.6). În ultima secțiune a acestui capitol (secțiunea 4.7) este prezentat cum proprietarul își poate bloca/debloca ușile mașinii prin intermediul aplicației. În capitolul 5 sunt prezentate concluziile (secțiunea 5.1), limitările aplicației (secțiunea 5.2) și direcțiile viitoare de dezvoltare ale aplicației (secțiunea 5.3).

Capitolul 2

Starea de artă și tehnologii folosite

2.1 Sisteme de control wireless

Tehnologia a ajuns în punctul în care fiecare dintre noi avem opțiunea de a utiliza conexiunea prin Bluetooth a telefonului mobil. În ziua de astăzi, majoritatea oamenilor iau telefonul cu ei indiferent unde ar merge. Datorită acestor fapte, numărul utilizatorilor de aplicații care facilitează efectuarea anumitor acțiuni este într-o continuă creștere.

"Phillips Hue" [13] este o aplicație mobilă care permite utilizatorului să controleze starea becurilor conectate via Bluetooth. Aplicația tratează problema autentificării. După ce s-a autentificat, pentru a adăuga becuri noi în aplicație, utilizatorul pornește detectarea de becuri cu senzori specifici. Odată adăugate becurile, utilizatorul poate regla intensitatea luminii, culoarea luminii și poate seta anumite programe specifice (stingerea/aprinderea anumitor becuri la o anumită oră).

Totuși, într-o hală utilizarea "Phillips Hue" nu este la fel de rentabilă, deoarece conexiunea Bluetooth între două dispozitive se poate realiza numai în cazul în care cele două sunt aflate la o distanță mai mică de 100 de metri. În plus, undele semnalului Bluetooth sunt blocate de pereții cu o structură solidă, ceea ce poate cauza probleme conexiunii dintre telefon și becuri într-o casă de dimensiuni mari.

Din categoria aplicațiilor care controlează anumite obiecte se pot număra și aplicațiile de tip "Remote control" [14] pentru Smart TV-uri. Majoritatea acestor aplicații folosesc conexiuni via Wi-Fi. Pentru a putea conecta o astfel de aplicație la un Smart TV, smartphone-ul sau tableta trebuie să fie conectate la aceeași rețea de Wi-Fi ca și Smart TV-ul. Partea de autentificare nu este tratată, însă în momentul împerecherii dispozitivelor, pe telefon sau tabletă trebuie introdus codulul de securitate care apare pe TV. Conexiunea dintre cele două rămâne salvată după închiderea aplicației sau a Smart TV-ului.

Aceste aplicații tind să înlocuiască telecomanda clasică. Acest lucru se realizează din mai multe motive. Unul dintre ele este că este mai ușor de folosit datorită interfeței mai prietenoase. Alt motiv este reprezentat de comoditate, deoarece omul nu trebuie să mai caute telecomanda, el știind tot timpul unde se află telefonul mobil. Totodată el nu mai trebuie să orienteze telefonul într-o poziție specifică pentru a utiliza televizorul (așa cum trebuia să facă atunci când utiliza telecomanda). Pe lângă acestea, un alt motiv pentru care telecomanda nu mai este la fel de folosită ca și în trecut este nepoluarea mediului încojurător. În trecut lumea nu recicla bateriile, fapt ce ducea

la ridicarea nivelului de poluare al Terrei. Neutilizarea telecomenzilor într-un număr la fel de mare, a dus implicit la scăderea numărului de baterii nereciclate, adică la scăderea gradului de poluare al Pământului.

2.2 Automotive

Industria auto este într-o continuă dezvoltare, dar în același timp este supusă la presiunnea de a inova în mod constant [15]. Acest lucru se datorează clienților deoarece ei vor ca prețul mașinilor să nu mai crească, însă, totodată își doresc ca mașina să fie din ce în ce mai bună, mai calitativă .

Presiunea inovării mai are la bază și alte cauze. Una dintre acestea este că trăim într-o lume în care marea majoritate a oamenilor dețin o mașină. Un studiu statistic [1] realizat în anul 2015, ne arată că, încă de acum 6 ani, în anumite țări raportul dintre numărul de mașini înregistrate în țara respectivă și numărul de locuitori este supraunitar (Finlanda: 1.07 și Andorra: 1.05). În Figura 2.1 se poate observa că în alte țări raportul este aproape de unitate (Italia: 0.84; Statele Unite ale Americii: 0.83).

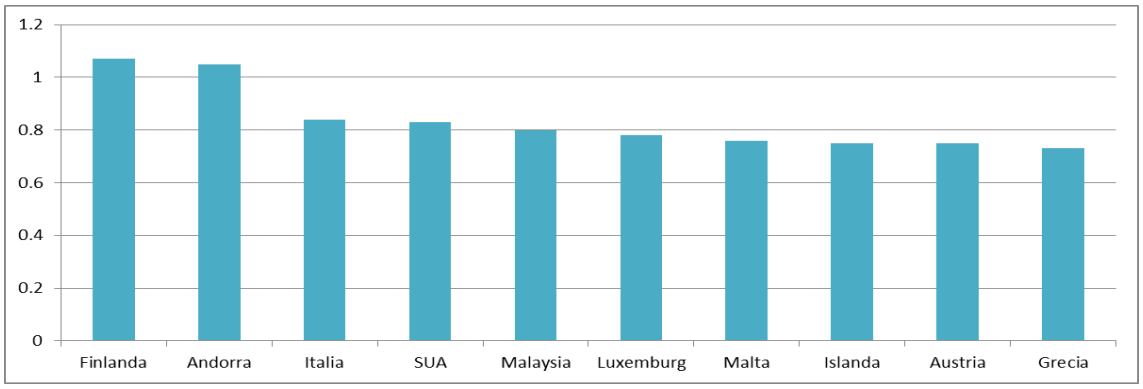


Figura 2.1: Raport între numărul total de mașini din țară și numărul de locuitori [1].

Luând în vedere acest studiu statistic, producătorii de mașini trebuie să inoveze pentru a-i putea convinge pe oameni să își schimbe mașinile. Din această cauză, majoritatea marilor producători de mașini au început să dezvolte și autovehicule care să fie în totalitate electrice. Astfel, prețul de întreținere al mașinii scade semnificativ, aproape de tot, deoarece nu mai trebuie plătite nici impozitul, nici reviziile anuale, respectiv schimburile de ulei și filtre etc. Spre exemplu, cei de la Mercedes au dezvoltat modelul EQC 400 4MATIC și cei de la BMW au dezvoltat modelul BMW iX. Pe lângă marii producători de mașini neelectrice, există și producători care scot pe piață doar modele de autovehicule în totalitate electrice, precum Tesla.

2.3 Abordări similare

Aplicația “Mercedes me” [9] este proiectată de către dezvoltatorii firmei Mercedes în scopul facilitării interacțiunilor pe care proprietarul le are cu mașina. Utilizarea

acesteia aduce de la sine anumite avantaje, cum ar fi verificarea volumului de combustibil rămas în rezervor indiferent de locația telefonului sau a mașinii. Proprietarul poate verifica cât combustibil mai are de oriunde, el nemaitrebuind să fie în mașină pentru acest lucru. Aplicația dispune de o funcție care calculează ruta cea mai eficientă în funcție de trafic sau ruta cea mai scurtă spre o destinație introdusă de conducător.

Mulți oameni care parchează mașina într-o parcare aglomerată pentru o perioadă mai mare de timp, când se întorc la aceasta, uită unde au lăsat-o. "Mercedes me" are o funcție care accesează locația mașinii și o transmite către utilizator, iar astfel "pierderea" mașinii prin parcări devine imposibilă. Tot legat de parcare, aplicația detectează locurile de parcare libere după ce mașina a trecut de ele. Pentru unii șoferi parcatul mașinii reprezintă o mare bătaie de cap. Funcția "Remote parking" ia stresul parcării mașinii de pe capul acestor șoferi, ea oferindu-le posibilitatea de a parca mașina din afara ei, prin intermediul telefonului. Totuși, pe cât de mare poate fi considerat acest avantaj, el vine la pachet cu un dezavantaj la fel de mare, deoarece în timpul execuției manevrei de parcare sistemul poate genera anumite erori, care pot duce la accidente grave.

O altă funcție a aplicației este cea de "Remote Car Lock" care permite utilizatorului să blocheze/deblocheze mașina indiferent de distanța dintre el și mașină. Acest lucru este văzut de multă lume ca pe un avantaj deoarece nu mai trebuie să fie la o anumită distanță de mașină pentru a putea debloca/bloca ușile acesteia. Totuși această funcție are dezavantajul că proprietarul mașinii poate uita mașina deblocată. Chiar dacă își poate bloca mașina de oriunde, în momentul în care își aduce aminte să verifice ușile mașinii s-ar putea sa fie deja prea târziu.

Pentru a putea folosi aplicația "My BMW app" [10], utilizatorul trebuie să își facă cont direct în aplicație. În comparație cu aplicația celor de la Mercedes, adăugarea de mașini noi in cont nu se realizează în mod automat, ci trebuie introdusă seria de șasiu a mașinii (VIN-ul). O altă deosebire între cele două aplicații este faptul că "My BMW app" nu are implementată funcția de "Car Parking Remote". Totuși cu ajutorul aplicației celor de la BMW conducătorul poate vedea modelul mașinii și tot ceea ce se află în jurul acesteia din orice perspectivă dorește. Bineînțeles, acest lucru este posibil doar la unele modele de BMW care dispun de camere de vedere în fiecare parte a mașinii.

"BMW Digital Key Plus" [11] este o aplicație realizată de BMW în colaborare cu Apple care este făcută special pentru modelul "BMW iX". Aplicația este disponibilă exclusiv pe IOS. Aplicația se bazează pe un sistem de unde radio pe distanță foarte scurtă care localizează utilizatorul și mașina cu o marjă de eroare foarte mică. Datorită acestui sistem din spatele aplicației, proprietarul își poate bloca/debloca ușile mașinii, respectiv porni mașina fără utilizarea telefonului.

2.4 Senzorii mașinii

Prin definiție, un senzor [16] este un modul al cărui scop este de a detecta anumite evenimente sau schimbări specifice și de a trimite informații celorlalte componente (în majoritatea cazurilor, informația este trimisă către procesor). Fie că este vorba despre un ecran de afișare sau despre un sistem complex, un senzor nu poate fi folosit de sine stătător .

Chiar dacă nu ne dăm seama de acest lucru, fiecare dintre noi avem tangență cu senzorii căci îi întâlnim în foarte multe situații, pornind de la deblocarea telefonului mobil asistată de un senzor de amprentă și până la aprinderea automată a instalațiilor de iluminat realizată prin prezența senzorilor de mișcare. De asemenea, întâlnim senzori și în situații particulare (nu la fel de comune), cum ar fi senzorii biochimici care ajută la măsurarea unor parametrii specifici sau senzorii de monitorizare a traficului care au capacitatea de a măsura viteza participanților la trafic sau de a detecta aglomerația.

Senzorii care fac parte din piesele folosite la construcția de mașini sunt senzori inteligenți. Aceștia sunt folosiți pentru a putea determina nivelul uleiului, temperatura, nivelul lichidului de răcire, etc. Dintre senzorii importanți care deservesc acestor funcții ale mașinii îi putem aminti pe următorii [2]: senzor de debit masic de aer, senzor pentru turația motorului, senzor de oxigen, senzor pentru lichidul de răcire, senzor de temperatură sau senzor de tensiune.

Senzorul de temperatură pentru răcire (Figura 2.2a) [17] este unul dintre cei mai importanți senzori în construcția de mașini. Acesta comunică cu calculatorul de bord (OBD2), transmițându-i anumite erori. Este folosit pentru a determina temperatura motorului și transmite calculatorului de bord când această temperatură nu este optimă. Senzorul de presiune barometrică (KPA) măsoară presiunea atmosferică în care rulează autovehiculul și transmite anumite semnale către alți senzori pentru a ajusta alți parametri, precum sincronizarea aprinderii combustibilului [2].

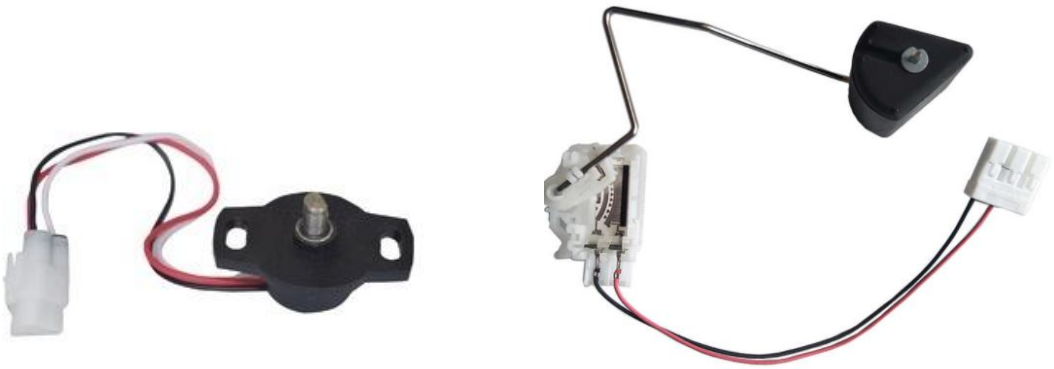
(a) (b)

Figura 2.2: Senzorul pentru temperatura lichidului de răcire (a); Senzorul pentru turația motorului (b) [2].

Mașina înștiințează conducătorul cu privire la nivelul combustibilului din rezervor. Acest senzor (Figura 2.3b) este foarte util deoarece conducătorul este foarte puțin probabil să rămână fără combustibil. Senzorul pentru turația motorului (Figura 2.2b) are ca scop înregistrarea turației în fiecare secundă. Acest lucru este util în momentul în care apare o eroare pe computerul de bord al mașinii deoarece turarea excesivă a motorului poate să fie cauza erorii.

Senzorul de presiune al galeriei de admisie (Figura 2.4b) transmite către computerul de bord presiunea cu care intră combustibilul în motor în funcție de cât de mult este apăsată pedala de accelarație. Acest lucru este determinat de senzorul pentru poziția clapetei de accelerație (Figura 2.3a) [2].

Senzorul pentru debitul masic de aer (Figura 2.4a) măsoară cantitatea de aer care intră în motor. Aerul care intră in motor se numește aer de admisie, iar temperatura acestuia este măsurată de un senzor specific [2].



(a) (b)

Figura 2.3: Senzorul pentru poziția pedalei de accelerație (a); Senzorul pentru nivelul combustibilului din rezervor (b) [2].

(a) (b)

Figura 2.4: MAF (a); MAP(b) [2].

2.5 OBD2

Majoritatea autovehiculelor din prezent dețin un sistem care îl atenționează pe conducător despre problemele mașinii, de la cele mai minore cum ar fi una dintre uși neînchise bine, până la probleme grave cum ar fi nemaifuncționarea pistoanelor în parametrii optimi. În acest caz se sugerează ca șoferul să nu mai circule cu mașina și să o ducă la cel mai apropiat service autorizat.

OBD2 [3] (Figura 2.5) este un sistem de autodiagnoză al mașinii. Un OBD acordă tehnicianului posibilitatea de a avea acces la un anumit subsistem de informații referitoare la reparațiile de care are nevoie mașina la momentul respectiv și la analiza performanței mașinii. Datorită sistemului OBD șoferul observă neregulile mașinii prin indicatorii luminoși din bord.

Sistemul OBD este foarte important deoarece cu ajutorul acestuia este posibilă menținerea mașinii la capacitate maximă. Totodată, sistemul joacă un rol important

[1]https://www.amazon.com/KOBRA-Wireless-OBD-Scanner-Connects/dp/B01C3HAHCS

Figura 2.5: OBD2 [1].

în sănătatea celui care conduce mașina, deoarece la cea mai mică problemă care apare este atenționat print-un semnal luminos în bord. Pe lângă avantajele menționate, sistemul OBD2 ajută și mecanicii deoarece aceștia trebuie doar să se conecteze la portul OBD pentru a afla problema mașinii. Datorită acestui fapt, ei economisesc timp prețios.

OBD2 oferă acces la informații și codul problemei găsite în urma diagnozei efectuate pentru: trenul propulsor al mașinii (motorul și sistemul de transmisie), sistemele de control al emisiilor, VIN (numărul de identificare al autovehiculului), CIN (numărul de identificare al calibrării), contorul de aprindere și contoarele sistemului de control al emisiilor.

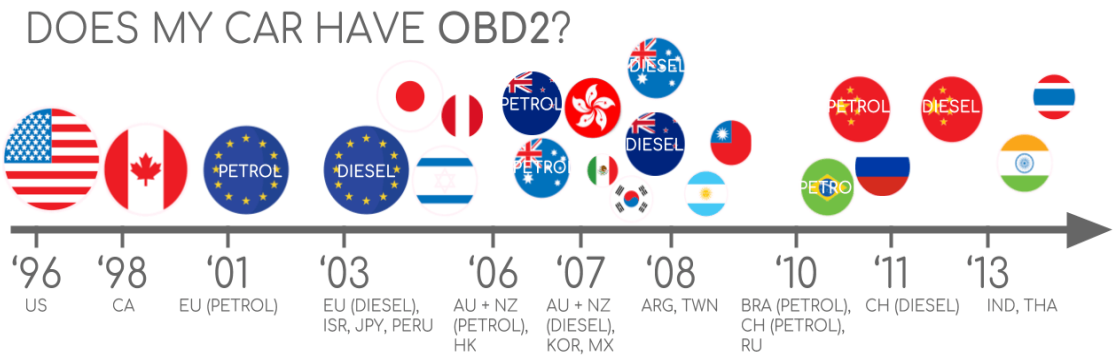


Figura 2.6: Apariția OBD2 în lume [3].

Așa cum se poate observa în Figura 2.6, sistemul OBD a fost introdus pentru prima dată în anul 1996 în Statele Unite ale Amercii, apoi 2 ani mai târziu este introdus și în Canada. În Europa, sistemul este introdus prima oară în anul 2001 pentru mașinile care funcționază cu benzină, iar mașinile care funcționează cu diesel au beneficiat de OBD 2 ani mai târziu, în anul 2001. În Australia și Noua Zeelandă sistemul este introdus pentru mașinile pe benzină în anul 2007 iar, un an mai târziu, pentru mașinile diesel. Cel mai târziu sistemul a fost introdus in India și Thailanda în anul 2013.

În Tabela 2.1 sunt prezentate câteva exemple de erori generate împreună cu o posbilă cauză pentru autovehiculele produse de către Audi [18]:

Tabela 2.1: Erori generate de către calculatoarele de bord ale mașinilor Audi.

Cod OBD	Descriere eroare	Cauza probabilă
P1192	Senzor de presiune de combustibil – tensiune de alimentare	Cablare, senzor de presiune a combustibilului
P1193	Senzor de presiune de combustibil – circuit deschis / scurtcircuit la pozitiv	Circuit deschis / scurtcircuit până la pozitiv
P1249	Sistemul de consum de combustibil – defecțiune a circuitului	Cablare, senzor de nivel al rezervorului de combustibil, ECM
P1298	Senzorul temperaturii lichidului de răcire a motorului (ECT) – semnal neclar	Cablare, senzor ECT
P1009	Senzor de masă a debitului de aer (MAF) 1/2 – semnal de detectare a încărcăturii neclar	Cablare, senzor MAF
P1552	Senzor de presiune barometrică (BARO) – deschis / scurtcircuit la masă	Cablare, senzor BARO
P1160	Senzorul pentru temperatura aerului de admisie (IAT) – scurtcircuit la masă	Cablaj scurt la masă, senzor IAT
P1396	Senzorul de viteză a motorului (RPM) lipsit de dinte	Rotor nesigur / deteriorat, senzor RPM
P1195	Supapa de control a presiunii combustibilului – circuit deschis / scurtcircuit la masă	Cablu deschis / scurtcircuit la masă, supapă de control a presiunii combustibilului

2.6 Standarde de comunicare

Trăim în era "comunicațiilor wireless", în care aproape nicio persoană nu mai folosește telefonul doar în scopuri tradiționale (de a efectua apeluri telefonice, de a trimite mesaje, etc.). Lumea utilizează telefonul mobil pentru a se ajuta de acesta în realizarea unor acțiuni într-un mod mai eficient din punct de vedere al timpului sau al modalității efective de realizare [19].

Tehnologiile wireless devin din ce în ce mai populare în lume. Sunt foarte apreciate de majoritatea persoanelor deoarece nu mai este nevoie de o mulțime de cabluri și legături pentru a putea conecta două dispozitive.

BT Media Access Control Scanner (BMS) este o tehnologie folosită la conexiunile wi-fi și bluetooth. Conceptul din spatele acesteia se bazează pe scanarea zonei pentru a descoperi Media Access Control Identifiers (MAC-ID) a dispozitivelor uzuale precum telefonul mobil, sistemul de navigație al mașinii sau sistemul de sunet al mașinii [4].

2.6.1 Bluetooth

Un sistem de conexiune Bluetooth este configurat pentru a putea modifica starea unui dispozitiv (LED, telefon mobil, etc) de la dispozitivul principal. Pe lângă aceste

două dispozitive, sistemul mai cuprinde și un controller care are scopul de a detecta ce dispozitive se conectează sau pierd conexiunea Bluetooth și de a ține cont de condițiile specifice în care trebuie să se desfășoare operațiile de modificare a stării unui dispozitiv.

Metoda de conexiune prin Bluetooth constă în două stări principale (standby și connection) și șapte substări. În momentul stării standby nu există nicio interacțiune între dispozitive; în starea connection utilizatorul poate transmite date între cele două dispozitive. Cele șapte substări sunt: inquiry, inquiry-scan, inquiry response, page, page scan, slave response, master response [4].

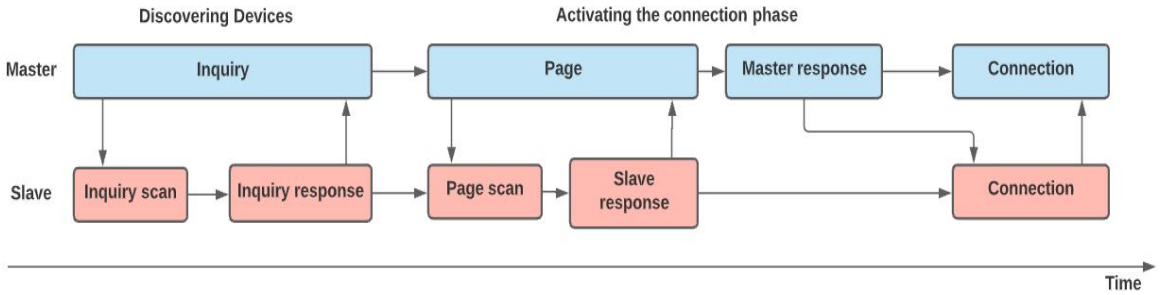


Figura 2.7: Realizarea conexiunii Bluetooth [4].

Așa cum se poate observa în figura de mai sus (Figura 2.7), pentru a se putea realiza conexiunea dintre dispozitive, dispozitivul părinte este în modul de scanare pentru a detecta dispozitivele disponibile pe o anumită rază. În cazul în care dispozitivul "slave" este în modul de scanare, atunci va scana semnalul primit de către dispozitivul părinte. Dispozitivul "slave" intră în modul răspuns pentru a trimite o atestare părintelui. Acesta detectează răspunsul primit și poate să intre în "Page Mode". Pentru a putea scana și a putea răspunde părintelui, dispozitivul "slave" trebuie să intre în modul "page-scan". Într-un final, părintele intră în modul "Master-response" pentru a putea trimite informații și a se putea realiza conexiunea finală între cele două dispozitive [4].

2.6.2 Wi-Fi

Wi-Fi Media Access Control Scanners (WMS) folosesc unde radio pentru transmiterea datelor către alt dispozitiv.

Interfețele Wi-Fi pot scana canale de comunicare în două moduri: activ și pasiv. În modul pasiv de scanare, dispozitivul primește mesajele Beacon la intervale de timp regulate, schimbând informațiile între două canale. În modul pasiv, dispozitivul nu răspunde niciunui mesaj Beacon. Așa cum se poate observa în figura de mai jos (Figura 2.8), modul activ presupune că dispozitivul caută posibilități de conexiune prin trimiterea de mesaje pe fiecare canal posibil așteptând un răspuns [4].

2.6.3 Bluetooth vs Wi-Fi

De cele mai multe ori lumea preferă conexiunea de tip Wi-Fi în detrimentul conexiunii Bluetooth deoarece este posibilă conectarea a două dispozitive care se află la o distanță mai mare. Din cauza acestui lucru, lumea nu mai ține cont și de costul

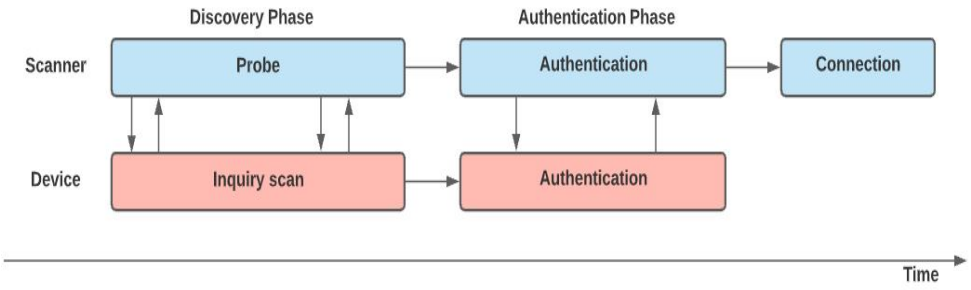


Figura 2.8: Trimiterea mesajelor prin Wi-Fi [4].

părții hardware care, în cazul sistemelor de conexiune Wi-Fi, este mult mai ridicat. Pentru a se putea realiza conexiunea Bluetooth este nevoie doar de un modul Bluetooth care permite conectarea dispozitivelor, pe când pentru realizarea conexiunii Wi-Fi sunt necesare adaptoare de rețea, routere și/sau puncte de acces wireless.

Aplicația "Smart Key" va folosi un sistem de conectare Bluetooh. Principalul motiv pentru care se întâmplă acest lucru este dat de costul mai ridicat al unui sistem Wi-Fi. Datorită caracteristicilor aplicației, utilizatorul se va folosi de informațiile oferite de către "Smart Key" în imediata apropiere a mașinii. Având în vedere acest lucru, nu este necesar un sistem de conectare care să ofere posibilitatea de a conecta două dispozitive aflate la distanță mai mare unul față de celălalt.

Pe lângă configurările făcute în fabrică, sistemul de conexiune Wi-Fi necesită și alte configurări făcute de utilizator atât la partea de hardware cât și la partea de software. Publicul țintă al aplicației "Smart Key" este reprezentat de populația posesoare de una sau mai multe mașini. Mulți dintre aceștia nu dispun de cunoștințele necesare pentru a realiza configurările necesare. Din această cauză, pentru a face cât mai ușoară utilizarea aplicației, se folosește conexiunea prin Bluetooth.

2.7 Tehnologii folosite

2.7.1 Limbajul Java

Limbajul Java [20] apare în toamna anului 1991, când firma "Sun Microsystems" finanțează proiectul "Green", condus de James Gosling. Proiectul "Green" era menit să adauge firma "Sun Microsystems" pe piața produselor electronice. În anul 1995, echipa "Green" finalizează caracteristicile limbajului Java.

Java este un limbaj compilat pentru că toate programele scrise în Java pot fi transformate într-un cod pe care mașina să îl poată rula mai ușor. Deoarece instrucțiunile unui program scris cu ajutorul limbajului Java sunt procesate linie cu linie, putem spune ca Java este un limbaj interpretat.

În momentul instalării limbajului, în mod implicit se va crea o mașină virtuală cu scopul de a scrie în instrucțiuni mașină (specifice pentru fiecare platformă în parte), instrucțiunile din fișierele de tip "byte-code" generate la compilarea programului; deci limbajul este independent de platformă.

Limbajul Java este orientat pe obiect deoarece întrunește toate caracteristicile programării orientate pe obiect: transmiterea parametrilor, moștenire, încapsulare,

obiecte, clase, biblioteci și modificatori de acces.

În comparație cu alte limbaje de programare, Java este un limbaj simplu. În Java nu există moștenire multiplă (C++) și nici pointeri (C/C++). Alocarea memoriei se face în momentul execuției. Spre deosebire de C++, unde dealocarea memoriei se face manual de către programator, în Java dealocarea se face automat cu ajutorul unui "Garbage collector".

Performanțele limbajului Java reies din rapiditatea cu care Java execută fișierul "byte-code" (aproximativ la fel de repede ca și execuția unui cod compilat) și din opțiunea utilizatorului de a programa mai multe fire de execuție. De exemplu, în timp ce memoria este dealocată de către "Garbage collector", mai poate exista un fir de execuție care să scrie într-un fișier datele rezultate în urma executării unui program.

Programele Java nu pot accesa memoria protejată (heap, stack) deoarece alocarea memoriei se face în mod automat în momentul execuției.

Cel mai important avantaj al folosirii Java este faptul că oferă o bună securitate, lucru foarte important pentru aplicațiile mobile. Android-ul lasă utilizatorul să folosească aplicații de tip "semi-trusted" care, fără o foarte bună securitate, pot afecta buna funcționalitate a telefonului. În momentul folosirii unor aplicații care rulează pe o mașină virtuală (ex: aplicațiile scrise în Java), este garantat faptul că nu va fi afectată funcționalitatea telefonului. Totuși, există posibilitatea ca și aceste aplicații să dăuneze telefonului în cazul în care la implementarea mașinii virtuale există un fir de execuție special pentru asta.

2.7.2 Android

Android [21] este un sistem de operare pentru dispozitive mobile cu touchscreen precum telefoanele mobile sau tablete. Este creat de Google. Apariția lui este anunțată în anul 2007, pentru ca în luna Septembrie a anului 2008 să apară primul dispozitiv mobil care folosea Android ca și sistem de operare.

Android-ul este o platformă open-source începând din anul 2008. Google oferă tot codul sursă al sistemului, iar dezvoltatorii pot să adauge diferite extensii (sub licența Apache), însă nu au voie să le facă disponibile comunității open-source [21]. Totodată faptul ca sistemul este de tip open-source facilitează publicarea aplicațiilor, deoarece nu mai este nevoie de anumite permisiuni, iar toate API-urile sunt publice, toată lumea având acces la ele. Faptul ca Android-ul este open-source [5] nu are avantaje numai pentru dezvoltatori, ci și pentru utilizatori, aceștia putând sa dețină controlul asupra sistemului și implicit, asupra a ceea ce se instalează pe dispozitiv .

Arhitectura Android [5] este formată pe mai multe straturi (Figura 2.9). Primul strat reprezintă kernelul pentru linux. Bibiliotecile din nucleu (Android Runtime) aduc funcționalitatea limbajului de programare JAVA. Aplicațiile Android rulează procese proprii pe mașina virtuală. Pe al doilea strat se află Bibiliotecile cu diferite funcționalități. De exemplu, biblioteca SGL este responsabilă pentru randarea imaginilor 2D, OpenGL—ES ajută la randarea imaginilor 3D sau SQLite, care are incorporată baza de date. Pe al trilea strat se află framework-ul aplicației, iar pe ultimul strat se află aplicația propriu-zisă.

Rezumând cele de mai sus, principalele avantaje ale folosirii Android ca și sistem de operare sunt: faptul că este un sistem de tip open-source, gratis; sistemul se poate dezvolta rapid și aplicațiile care suportă acest sistem de operare pot fi publicate de

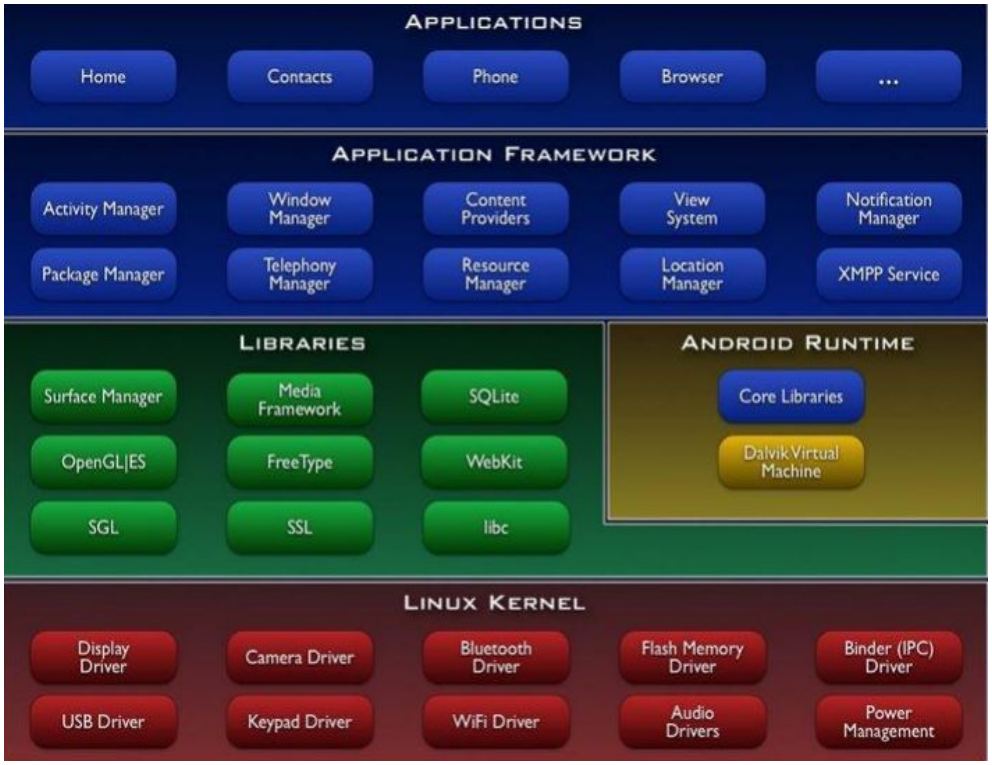


Figura 2.9: Arhitectura sistemului Android [5].

către oricine.

2.7.3 Android API

API-ul [6] (Application Programming Interface) este folosit de cele mai multe ori în cadrul aplicațiilor web. Acesta este format dintr-un set de instrucțiuni și diferite specificații care ajută programele să comunice între ele. API-ul este un intermediar dintre utilizatorul aplicației și server. Așa cum se poate observa și în Figura 2.10, utilizatorul trimite cererea, iar API-ul execută instrucțiunile pentru ca mai apoi să întoarcă datele primite de la server înapoi utilizatorului.

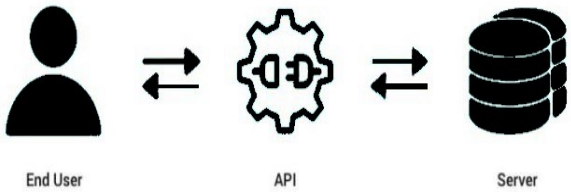


Figura 2.10: Schema API [6].

Android API-ul [22] este o colecție de module software care formează Android SDK. Altfel spus, Android API și Android SDK reprezintă același lucru deoarece codul scris de către programator interacționează cu software-ul intern al Androidului (respectă

definiția API-ului). Spre exemplu, pentru a găsi coordonatele exacte ale dispozitivului mobil, se folosește Maps API-ul. În cazul în care aplicația are un sistem de login pe bază de amprentă, acesta are implementat Biometric API. Prin intermediul acestuia, aplicația respectivă poate să se folosească de senzorii biometrici ai telefonului mobil.

2.7.4 Baze de date

O bază de date [23] reprezintă o colecție structurată de date, stocată pe un server în cele mai multe cazuri. Baza de date este controlată de un sistem de management al acesteia (DBMS). DBMS-ul suportă scheme și subscheme logice, acces la metode și grupare de date, un limbaj de manipulare a datelor (de exemplu SQL). Nu în ultimul rând, DBMS-ul are nevoie de un nivel de securitate .

În practică există două tipuri de sisteme de baze de date [23]: sistem de baze de date relaționare și sistem de baze de date nerelaționare. Una dintre principalele deosebiri dintre cele două tipuri de sisteme este că atunci când vorbim despre prima categorie putem vorbi și despre relațiile dintre tabelele acesteia: one-to-one (cheia primară din tabela copil este asociată cu cheia externă din tabela părinte), one-to-many (o coloană din tabela părinte este asociată mai multor coloane din tabela copil) și many-to-many (constă într-o tabelă pivot care are două chei externe asociate celor două tabele membre ale relației) .

2.7.5 Firebase

Firebase [24] este un instrument oferit de Google. Acesta are scopul de a construi, îmbunătăți și dezvolta aplicația. Datorită acestui instrument, programatorii nu mai trebuie să implementeze anumite servicii deoarece ele sunt oferite gata implementate de Firebase. Printe aceste servicii se numără autentificarea, baza de date, anumite configurații, mesaje de atenționare și lista poate continua. Firebase este stocat în Cloud .

Așa cum se poate observa în Figura 2.11 structura backend-ului unei aplicații [25] care utilizează Firebase este mult mai simplă. În cazul aplicațiilor tradiționale, programatorul trebuie să creeze și să configureze baza de date, să implementeze partea de server și să își creeze API-urile. În schimb, dacă se utilizează firebase, partea de backend este deja implementată. În codul pentru frontend sunt introduse doar endpoint-urile API-ului iar mai apoi backend-ul își face treaba.

Utilizarea Firebase pentru aplicații mobile aduce de la sine anumite avantaje față de restul aplicațiilor care nu utilizează acest instrument. În primul rând, viteza cu care răspunde serverul la cererile aplicației este mai mare. Acest lucru se întâmplă datorită faptului că Firebase funcționează pe serverele de la Google care sunt mult mai bine optimizate. Un alt avantaj al utilizării Firebase este baza de date în sine deoarece se utilizează o bază de date de tip NoSQL (nerelațională). Bazele de date NoSQL sunt mai eficiente în cazul în care se stochează un număr ridicat de date. Totodată, NoSQL asigură o performanță superioară și este mult mai ușor să fie scalate [26].

[2]https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

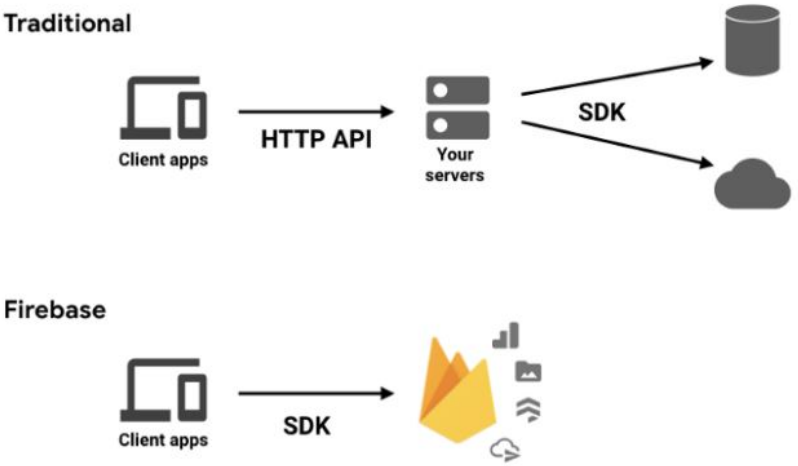


Figura 2.11: Structura aplicației care folosește Firebase vs. Structura aplicației care nu folosește Firebase [2].

Aplicația "Smart Key" va fi conectată la Firebase deoarece are deja implementată partea de autentificare. Un alt motiv pentru folosirea Firebase în detrimentul altor baze de date este că Firebase este stocat în Cloud și funcționează pe serverele de la Google, lucruri ce duc la inecesitatea utilizării unui server propriu.

2.7.6 Arduino

Arduino [27] este una dintre puținele firme care produc atât partea de hardware (microcontrolere) cât și partea de software necesară programării acestora. Plăcuțele Arduino citesc anumite date de intrare cum ar fi apăsarea unui buton de către utilizator, semnalul conexiunii bluetooth dintre senzor și utilizator sau chiar primirea unui mesaj pe rețelele de socializare. Aceste date de intrare sunt convertite în date de ieșire care pot fi aprinderea unui LED sau emiterea de semnal electric pe un anumit fir. Transformarea datelor de intrare în date de ieșire este facută prin instrucțiuni ale limbajului Arduino.

Orice persoană, nu numai pasionații de robotică, își poate crea roboți proprii sau propriile sisteme bazate pe senzori. Acest lucru este realizabil datorită documentației stufoase care conține inclusiv algoritmi pentru cele mai simple utilizări ale Arduino (de exemplu algoritmi pentru setarea intensității unui LED cu ajutorul potențiometrului sau pentru pâlpâirea LED-ului o dată la un număr de secunde). Pe lângă algoritimi, în documentație este specificat inclusiv cum ar trebui construită partea hardware a proiectului.

Un alt avantaj al folosirii Arduino este cel financiar. Microcontrolerele Arduino au cel mai bun raport preț/servicii oferite. Nu este necesară nicio sursă externă de curent deoarece, pentru încărcarea codului pe microcontroler, este necesară conexiunea prin USB cu PC/laptop, conexiune ce joacă și rol de sursă de curent pentru plăcuță.

Arduino UNO [7] este un model al microcontrolerelor Arduino. În Figura 2.12 este un model de Arduino UNO. Utilizatorul conectează Arduino-ul la PC/laptop prin

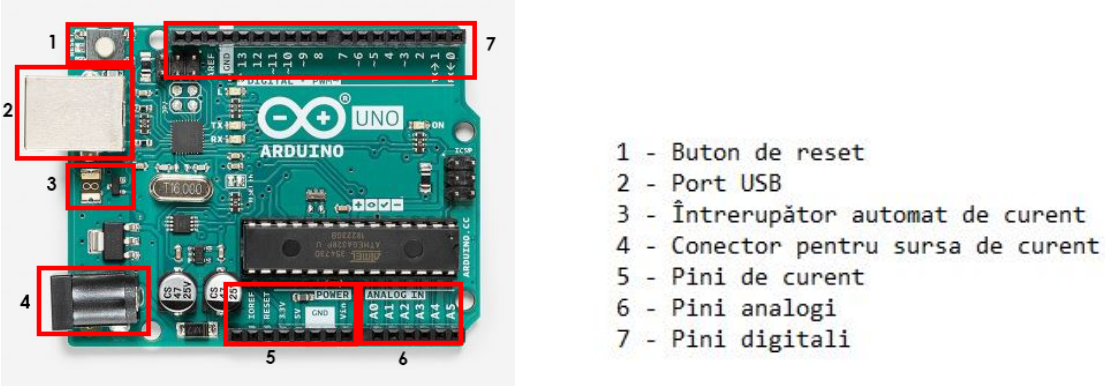


Figura 2.12: Arduino UNO [7].

portul USB (2). În cazul în care pe plăcuță este încărcat un anumit cod, utilizatorul poate conecta microcontrolerul la o sursă de curent (4) iar codul se va rula automat. Pentru a putea încărca alt cod pe plăcuță, utilizatorul se folosește de butonul de reset (1). Prin pinii de curent (5), Arduino-ul transmite curent către celelalte componente conectate (ex LED). Pinii digitali (7) sunt folosiți pentru a stabili pe ce pin este transmis semnalul electric. Pinii analogi (6) sunt folosiți pentru citirea anumitor valori. Pe lângă acestea, Arduino-ul mai dispune și de un întrerupător automat de curent (3).

Raportându-ne la alte microcontrolere, Arduino UNO are un procesor care are o viteză destul de mică (16 MHz). Pentru un sistem prin care utilizatorul modifică culoarea și intensitatea luminii unui LED RGB viteza mică nu va fi o problemă, însă pentru un robot care are un număr mare de task-uri de efectuat pe secundă viteza scăzută a procesorului va deveni o problemă semnificativă. Din punctul de vedere al memoriei, Arduino UNO este limitat la 32KB, însă din nou, acest lucru poate însemna un dezavantaj în funcție de proiect.

Capitolul 3

Proiectarea și dezvoltarea aplicației

3.1 Introducere

Scopul proiectului este de a dezvolta o aplicație care să poată fi utilizată ca o cheie virtuală pentru a bloca și debloca ușile mașinii la care este conectată. Un alt beneficiu al utilizării aplicației Smart Key este că în aceasta se pot vedea datele de la anumiți senzori ai mașinii. Aceste date se primesc prin intermediul microcontrolerului Arduino instalat pe mașina respectivă, care la rândul lui primește datele de la calculatorul de bord al mașinii.

Aplicația dezvoltată poate fi folosită numai de persoanele care au un cont personal. Ca metode de login, aceasta dispune de logarea clasică, în care utilizatorul trebuie să își introducă emailul și parola contului manual și de metoda de logare cu amprentă. A doua metodă este valabilă numai în cazul în care dispozitivul mobil pe care este instalată aplicația dispune de senzorii necesari și utilizatorul are setată opțiunea de deblocare a ecranului prin amprentă.

Totodată, aplicația înștiințează conducătorul în momentul în care calculatorul de bord al mașinii detectează o eroare. Înștiințarea se realizează prin afișarea unei notificări care este formată din numele erorii detectate (ex P1190) și o scurtă descriere a acesteia.

O altă utilitate importantă a aplicației ar fi că utilizatorul poate să vadă numărul de kilometri la care s-a efectuat ultimul schimb de plăcuțe de frână și de ulei de motor. Aplicația transmite în mod regulat memento-uri legate de numărul actual de kilometri și câți kilometri mai pot fi parcurși până în momentul în care este recomandată schimbarea uleiului de motor, respectiv a plăcuțelor de frână. Datorită acestora, este imposibil pentru un utilizator Smart Key să uite să schimbe uleiul sau plăcuțele.

Aplicația nu are numai avantaje pentru șoferi. În momentul în care un utilizator Smart Key ajunge cu mașina într-un atelier cu o anumită problemă, mecanicul poate vedea imediat din aplicația clientului care este eroarea ce a cauzat defecțiunea, economisind timp. Datorită faptului că aplicația este gratuită și că în ea se pot adăuga mașini de orice marcă indiferent de anul lor de fabricație (singura condiție este să lege microcontrolerul la calculatorul de bord al mașinii), apare un avantaj financiar simțit de propietarii de mai multe mașini de la mai mulți producători. Acesta este simțit de firmele de transport sau de curierat care au în posesie un număr foarte mare de mașini.

Așa cum se poate observa în Figura 3.1, aplicația mobilă urmează să comunice prin

Bluetooth cu microcontrolerul de Arduino instalat pe mașină. La rândul său, acesta este responsabil de controlul ușilor mașinii. Totodată, Arduino-ul primește datele de la calculatorul de bord al mașinii, care la rândul său le primește de la senzori. Pe lângă Arduino, aplicația mai comunică și cu bazele de date ale aplicației. Baza de date utilizatori stochează conturile, iar baza de date mașini stochează datele generale despre mașinile adăugate în conturi. În baza de date erori sunt stocate erorile generate de către computerul de bord al mașinii, iar în baza de date producător sunt stocate intervalele la care constructorul recomandă efectuarea schimbului de ulei și de plăcuțe de frână.

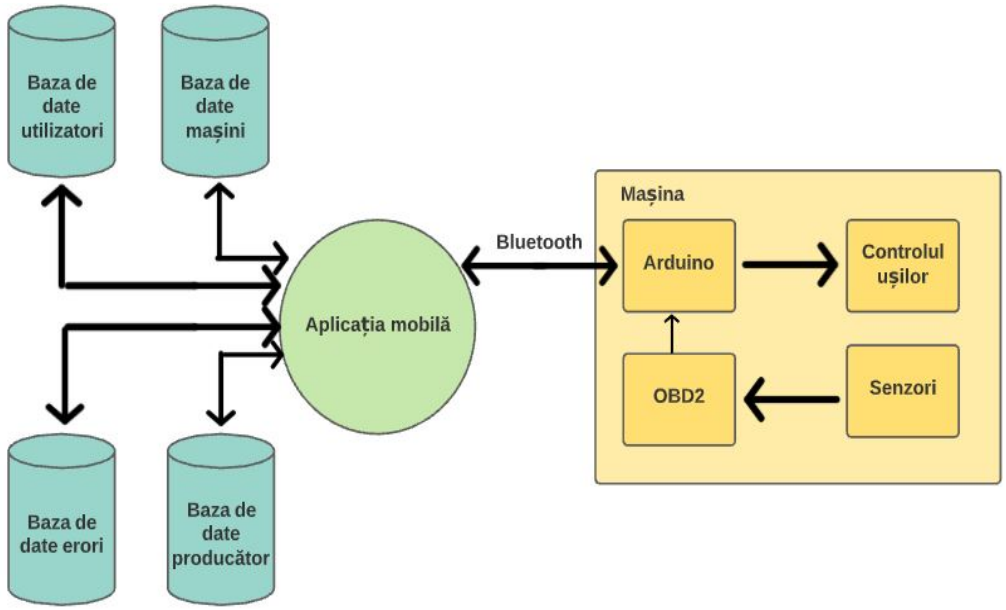


Figura 3.1: Fluxul aplicației.

3.2 Arhitectura aplicației

Arhitectura aplicației dezvoltate este bazată pe 3 nivele arhitecturale. Primul nivel al arhitecturii este "Presentation layer". Scopul acestui nivel este de a asigura conexiunea dintre utilizator și aplicație. Acesta afișează informațiile cerute de către utilizator și colectează datele pe care le primește de la el, pentru a le transmite mai departe către celelalte nivele arhitecturale ale aplicației. În cazul aplicației Smart Key, primul nivel arhitectural este realizat cu ajutorul XML (Extensible Markup Language).

"Business layer" este al doilea nivel arhitectural al aplicației dezvoltate. Acesta cuprinde toată logica aplicației. În acest nivel sunt procesate cererile trimise de către utilizator prin intermediul primului nivel arhitectural. Aceste cereri sunt procesate utilizând un set de reguli. În cazul aplicației Smart Key, nivelul de procesare al datelor este implementat cu ajutorul limbajul de programare Java. Comunicarea dintre acest nivel și ultimul nivel arhitectural este realizată prin intermediul API-urilor.

"Data layer" este ultimul nivel în arhitectura aplicației dezvoltate. Acesta mai este numit și nivelul bazei de date. Pe acest nivel se stochează și se efectuează management-ul rezultatelor proceselor efectuate de aplicație pe nivelul al doilea. Al treilea nivel

al aplicației dezvoltate nu poate comunica direct cu nivelul de prezentare. În cadrul aplicației Smart Key, acest nivel arhitectural este reprezentat de un sistem de baze de date nerelaționare numit Firebase.

În cadrul acestei arhitecturi, primul nivel nu poate comunica direct cu al treilea. Legătura dintre aceste două nivele se realizează prin intermediul celui de-al doilea nivel arhitectural al aplicației. Așa cum se observă în Figura 3.2 de mai jos, prin intermediul primului nivel arhitectural, utilizatorul trimite comenzi către cel de-al doilea. În momentul primirii cererii de la primul nivel, "Buisness layer"-ul se apucă de procesat. În timpul procesării există posibilitatea de a fi nevoie să se efectueze modificări asupra bazei de date. Având în vedere faptul că al doilea nivel nu are acces direct la baza de date a aplicației, acesta va trimite o cerere către nivelul al treilea din arhitectură. În al treilea nivel de arhitectură al aplicației, pe baza cererii primite se construiesc interogări specifice pentru a putea fi aplicate asupra bazei de date. După efectuarea interogării, "Data layer"-ul întoarce un răspuns către nivelul de prelucrare a datelor la cererea făcută. Acesta, la rândul lui, termină procesul și întoarce rezultatul către nivelul de prezentare al aplicației, care îi prezintă utilizatorului rezultatul cererii inițiale.

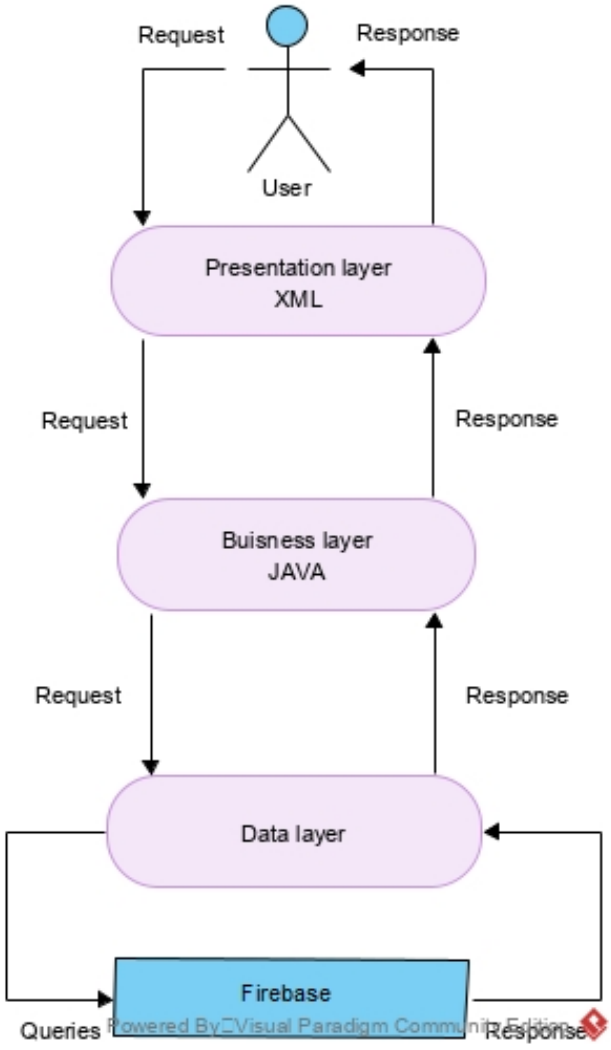


Figura 3.2: Arhitectura aplicației.

Smart Key are 3 subsisteme la bază. Primul astfel de subsistem este aplicația mobilă, al doilea este microcontrolerul Arduino instalat pe mașină împreună cu senzorii și cel de al treilea subsistem este Firebase-ul.

Subsistemul care constă în aplicația mobilă este gestionat într-o mare măsură de către primul nivel arhitectural Smart Key. Acest lucru se întâmplă datorită faptului că prin intermediul aplicației, utilizatorul trimite diferite cereri. Totodată, acest subsistem intră și în gestiunea celui de-al doilea nivel arhitectural deoarece în aplicația mobilă sunt procesate cererile.

Subsistemul alcătuit din microcontrolerul Arduino montat pe mașină și din senzorii acestuia intră în totalitate în gestiunea celui de-al doilea nivel arhitectural al Smart Key. Acest lucru se întâmplă deoarece cu ajutorul microcontrolerului sunt trimise datele de la computerul de bord al mașinii către aplicație (acesta preia datele de la fiecare senzor în parte). Totodată, el primește solicitările utilizatorului pentru blocare sau deblocare a ușilor mașinii prin intermediul aplicației mobile. Aceste cereri sunt procesate cu ajutorul microcontrolerului.

Ultimul subsistem este în totalitate gestionat de ultimul nivel din arhitectura aplicației ("Data layer"). Acest lucru se întâmplă deoarece "Data layer"-ul primește anumite solicitări din partea celui de-al doilea nivel din arhitectură pentru prelucrarea sau extragerea anumitor date din baza de date.

Cele 3 subsisteme ale aplicației nu funcționează independent. În sprijinul acestei afirmații luăm în considerare următorul scenariu: utilizatorul dorește să vadă toate erorile pe care calculatorul de bord al acesteia le-a detectat de când a adăugat mașina în aplicație. Acesta trimite această cerere prin intermediul aplicației. Cererea este procesată tot la nivelul aplicației, iar apoi este trimisă solicitarea către "Data layer". Acesta urmează să extragă din baza de date ceea ce este relevant pentru solicitarea primită și să trimită rezultatul înapoi aplicației, pentru ca aceasta să trimită un răspus utilizatorului. Subsistemul mașinii format din microcontrolerul Arduino, are și el un rol la fel de important în trimiterea răspunsului pentru utilizator, pentru că el trimite datele către aplicație unde sunt procesate și trimise pentru salvarea în baza de date; deci, fără acest subsistem, la solicitarea utilizatorului ar fi imposibil de găsit un răspuns, sau răspunsul trimis ar fi irelevant pentru utilizator.

3.3 Implementarea aplicației

În Figura 3.3 sunt prezentate acțiunile pe care utilizatorul le poate săvârși asupra sistemului aplicației. Prima interacțiune pe care utilizatorul o are cu sistemul este autentificarea, fie prin crearea unui nou cont, fie prin logarea pe un cont deja existent. După logare, acesta este redirecționat către meniul aplicației. În acest meniu, el poate să adauge o nouă mașină în aplicație sau să se conecteze la altă mașină. După conectarea la o mașină, utilizatorul poate să folosească aplicația ca o cheie virtuală pentru blocarea/deblocarea ușilor mașinii, poate să vizualizeze istoricul erorilor generate de către mașină și să actualizeze numărul de kilometri la care s-a efectuat ultimul schimb de ulei de motor și de plăcuțe de frână. Totodată, utilizatorul poate să verifice în timp real valorile transmise de anumiți senzori ai mașinii.

Subsistemul format din aplicația mobilă conține la rândul său mai multe module. Modulul responsabil de înregistrarea de noi utilizatori ai aplicației utilizate este ges-

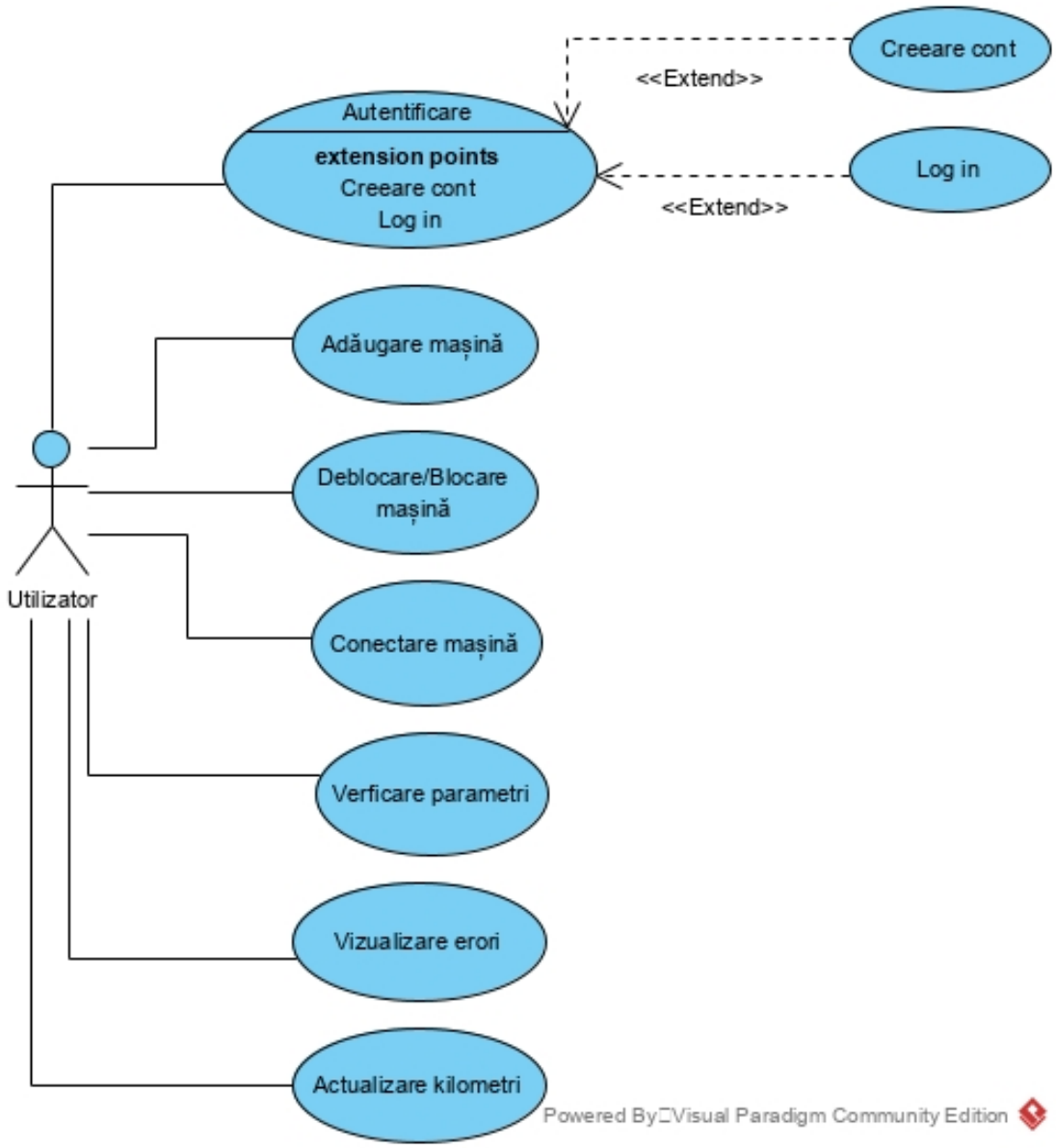


Figura 3.3: Diagrama cazurilor de utilizare.

tionat atât de subsistemul aplicației cât și de cel al bazei de date. Pentru a se putea înregistra, utilizatorul trebuie să completeze câmpurile cu datele cerute. După ce utilizatorul introduce datele necesare, urmează validarea acestora în baza restricțiilor impuse. Un exemplu de restricție este cea care se aplică asupra câmpului unde utilizatorul trebuie să își introducă adresa de email. Astfel, acel câmp permite doar utilizarea de șiruri de caractere care să respecte structura unui email valid. În cazul în care datele au trecut de validare urmează să fie salvate în baza de date sub forma din Figura 3.4 .

Logarea se poate realiza prin două metode: cea clasică și cea cu amprentă. Pentru logarea clasică, utilizatorul trebuie să își introducă datele. Aceste date sunt preluate de aplicație și sunt verificate cu cele din baza de date a conturilor pentru a vedea dacă există o potrivire între ele.

Cea de-a doua metodă de logare este cea cu amprentă. Aceasta este disponibilă

Figura 3.4: Stocarea contului în baza de date.

numai în anumite cazuri: dacă dispozitivul de pe care rulează aplicație are senzori biometrici, dacă acești senzori sunt disponibili și dacă telefonul are setată și metoda de deblocare a ecranului prin amprentă.

Listing 3.1: Determinarea valabilității logării cu amprentă

```
BiometricManager biometricManager = BiometricManager.from(this);
switch (biometricManager.canAuthenticate())
{
    case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
        login_btn.setVisibility(View.INVISIBLE);
        break;
    case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
        login_btn.setVisibility(View.INVISIBLE);
        break;
    case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
        login_btn.setVisibility(View.INVISIBLE);
        break;
    case BiometricManager.BIOMETRIC_SUCCESS:
        break;
}
```

Așa cum se poate observa în secvența de cod 3.1, sunt tratate patru cazuri. În primul rând, este tratat cazul în care dispozitivul mobil nu are senzori biometrici pentru realizarea logării cu amprentă. În cazul în care dispozitivul nu dispune de astfel de senzori, butonul de la care se efectuează logarea devine invizibil, deci imposibil de utilizat. Dacă se trece de acest caz, urmează să se verifice disponibilitatea acestui senzor. În cazul în care nu este disponibil, se întâmplă același lucru ca și la cazul anterior, metoda aceasta de logare devenind imposibilă. Dacă se trece și de acest caz, urmează să fie verificat dacă dispozitivul pe care rulează aplicația are setată și metoda de deblocare a ecranului prin amprentă. Numai dacă se trece și de această evaluare logarea cu amprentă va funcționa în aplicația Smart Key.

În momentul înregistrării unui noi cont în aplicație, aceasta reține ID-ul dispozitivului mobil de pe care a fost făcut contul. Acest ID este unic pentru fiecare dispozitiv în parte. Datorită acestui fapt, în momentul în care se efectuează logarea cu amprentă (dupa ce s-au efectuat verificările necesare) se caută în baza de date ID-ul dispoziti-

vului de pe care se transmite cererea, pentru a vedea ce cont a fost creat de pe acel dispozitiv și a efectua logarea cu contul respectiv.

Un alt modul din cadrul aplicației este cel în care se realizează conexiunea dintre dispozitivul utilizatorului și mașina de la care urmează să îi citească datele. În primul rând, aplicația detectează toate dispozitivele împerecheate prin Bluetooth cu telefonul utilizatorului, pentru ca mai apoi să le filtreze după nume și să îi arate utilizatorului doar mașinile adăugate în cont. După ce utilizatorul alege la ce mașină dorește să se conecteze, urmează ca aplicația să efectueze următoarea etapă.

Listing 3.2: Conexiunea cu Arduino

```
try
{
    if ( btSocket==null || !isBtConnected )
    {
        myBluetooth = BluetoothAdapter.getDefaultAdapter();
        BluetoothDevice dispositiv = myBluetooth.
            getRemoteDevice(address);
        btSocket = dispositiv.
            createInsecureRfcommSocketToServiceRecord(myUUID);
        BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
        btSocket.connect();

        adr = btSocket.getRemoteDevice().getAddress();
    }
} catch (IOException e) {
    ConnectSuccess = false;
    }
```

În secvența de cod 3.2 se poate observa următoarea etapă. La început se reține dispozitivul în funcție de ceea ce a ales utilizatorul la pasul precedent. Mai apoi, se creează un obiect de tip "btSocket" pentru a se putea inițializa și pentru a se putea gestiona conexiunea.

Adăugarea unor noi mașini în cont formează un alt modul al aplicației. Pentru a putea efectua acest lucru, utilizatorul trebuie doar să fie logat și să introducă corect datele necesare în pagina respectivă. După introducerea datelor, se realizează o serie de verificări asupra acestora. Acestea sunt realizate deoarece există câmpuri cu anumite restricții. Un exemplu de restricție este neacceptarea altor date de intrare în afara numerelor pentru câmpul "Car's fabrication year", unde este cerut anul de fabricație al mașinii. În cazul în care se trece de toate verificările, mașina este salvată în baza de date sub următoarea formă: Figura 3.5.

Un modul care implică subsistemul aplicației și cel format din microcontrolerul Arduino este cel care se ocupă de blocarea și deblocarea ușilor mașinii. În momentul în care utilizatorul apasă butonul destinat pentru blocare sau deblocare, aplicația va trimite un semnal către mașină (Secvența de cod 3.3). Semnalul trimis este "1" în cazul în care se dorește deblocarea ușilor (adică aprinderea LED-ului), iar în cazul în care utilizatorul dorește să blocheze ușile mașinii este trimis semnalul "0" (adică stingerea LED-ului).

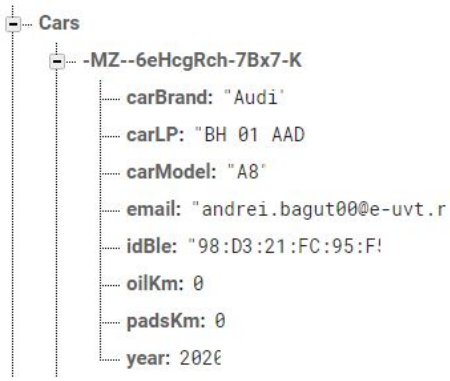


Figura 3.5: Mașina în baza de date.

Listing 3.3: Trimiterea semnalului către mașină.

```
private void sendSignal ( String number )
{
    if ( btSocket != null ) {
        try {
            btSocket.getOutputStream().write(number.toString().
                                                    getBytes());
        } catch (IOException e) {
            msg("Error");
        }
    }
}
```

Așa cum se poate observa în secvența de cod 3.4, mai întâi se definește care pin va fi cel care va transmite semnalul electric către LED (pinul 13 în cazul de față). După definire se verifcă dacă se primește vreun semnal de la aplicație. În caz afirmativ, se verifică dacă este 0 pentru a se putea închide LED-ul sau dacă este 1 pentru a se putea deschide LED-ul.

Listing 3.4: Cod Arduino.

```
#define ledPin 13
int state = 0;
void setup(){

    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
    Serial.begin(9600); // Default communication rate
}
void loop() {
  // Checks if the data is coming from the serial port
  if(Serial.available() > 0){
    state = Serial.read(); // Read the data from the serial port
  }
```

```
  if (state == '0') {
    digitalWrite(ledPin, LOW); // Turn LED OFF
     state = 0;
  }
  else if (state == '1') {
      digitalWrite(ledPin, HIGH);
      state = 0;
  }

}
```

Un alt modul care implică două subsisteme ale aplicației dezvoltate este cel care deservește pentru actualizarea numărului de kilometri la care s-au efectuat ultimele schimburi de ulei de motor și de plăcuțe de frână. Pentru a se realiza acest lucru, aplicația caută în baza de date mașina la care este conectat utilizatorul. După găsirea acesteia se actualizează câmpul "oilKm" (Figura 3.5) sau "padsKm" (Figura 3.5) cu numărul pe care utilizatorul îl introduce în aplicație.

Afișarea datelor de la senzori se realizează prin citirea setului de date(Secțiunea 4.2). Pentru a se putea citi datele la interval de o secundă am creat un nou fir de execuție care să ruleze în fundal, în același timp cu firul principal de execuție al aplicației. Acesta este realizat cu ajutorul clasei Runnable. Această clasă se instanțiază cu anumiți parametrii extrași din baza de date (numele proprietarului, marca, modelul și numărul de înmatriculare al mașinii, intervalul de kilometri la care se efectuează schimburile de ulei și de plăcuțe de frână, ultimul număr de kilometri la care s-a efectuat ultimul schimb de ulei și de plăcuțe de frână și emailul utilizatorului).

Clasa suprascrie metoda "run()" în care citește datele din fișierul corespunzător. În metoda "run()" se suprascrie din nou metoda "run()", însă de această dată din metoda "post()" din "mainHandler". Aici se actualizează câmpurile unde sunt afișate datele și firul de execuție pentru citire este pus pe pauză pentru o secundă (Secvența de cod 3.5).

Listing 3.5: Actualizarea datelor în timp real.

```
mainHandler.post(new Runnable() {
    @Override
    public void run() {
        tvbp.setText(bp);
        tvect.setText(ect);
        tvfl.setText(fl);
        tvimp.setText(imp);
        tvmaf.setText(maf);
        tvait.setText(ait);
        tvtp.setText(tp);
        tvdct.setText(dtc);
        tvta.setText(ta);
        tvrpm.setText(rpm);
        tvkm.setText(km);
    }
});
try {
```

```
    Thread.sleep(1000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

În fiecare fişier corespunzător fiecărei mașini există câmpul DCT_NUMBER (Figura 4.5) în care este afişat automat numele erorii generate de către calculatorul de bord al mașinii (dacă este cazul). Astfel, aplicaţia doar citeşte din fişier valoarea din câmpul respectiv, iar dacă acesta nu este nul (are valoare scrisă în el) atunci, în mod automat, aplicaţia va înştiinţa proprietarul printr-o notificare pe dispozitivul pe care este instalată aceasta și eroarea este salvată în baza de date ca în Figura 3.6.

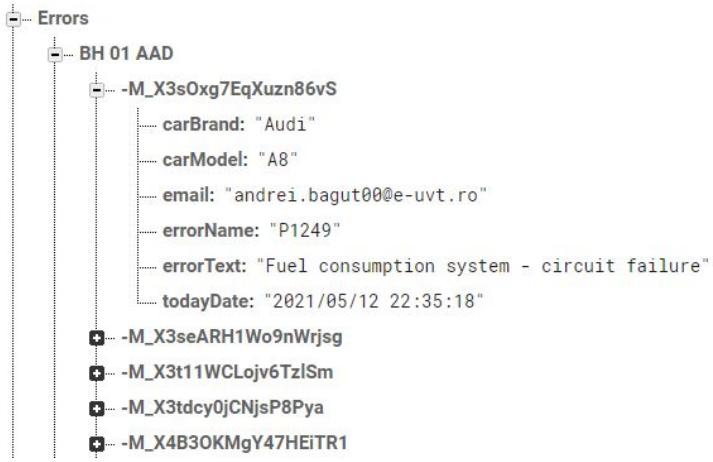


Figura 3.6: Erorile în baza de date.

În cazul în care o eroare apare în mod repetat, aplicația va trimite notificări o dată la 3 secunde, dar în baza de date este salvată fiecare apariție a erorii. În momentul în care se găsește câmpul DCT_NUMBER nevid, se efectuează o despărțire a șirului de caractere din câmpul respectiv. Acest lucru se realizează în eventualitatea în care există mai multe erori odată, iar aplicația trebuie să creeze în baza de date o înregistrare pentru fiecare eroare întâlnită.

Listing 3.6: Separarea erorilor

```
if(!dtc.isEmpty() )
{
    String [] dtcParts = dtc.split("␣");
    for (int i=0;i<dtcParts.length;i++)
    {
        checkNotification(dtcParts[i],x,carModel,carBrand,carLp,
           email);
    }
}
```

Separarea se realizează prin rularea secvenței de cod 3.6. În fișiere, în cazul în care la un moment dat apar mai multe erori simultan, acestea sunt despărțite prin “ ” (ex: “P0000 P1111 P2222“). Datorită acestui lucru, șirul de caractere se desparte după spații, iar mai apoi pentru fiecare entitate găsită în urma despărțirii se verifică dacă

se poate afișa notificare sau doar să se salveze în baza de date.

Un alt modul care este gestionat de subsistemul aplicației și cel al bazei de date este cel care se ocupă cu afișarea istoricului erorilor care au fost detectate de către calculatorul de bord al mașinii. Tot în acest modul, utilizatorul poate să filtreze erorile după numele acestora.

La filtrarea după numele erorii, utilizatorul poate să aleagă doar dintre numele erorilor care s-au detectat (el poate să vadă erorile cu numele P1100 dacă acestea au fost detectate, în schimb deși este tratat cazul apariției erorii P0923 și nu a fost detectată de mașină până în acel moment, utilizatorul nu va avea P0923 ca și opținue de filtrare)

Opțiunile de filtrare după numele mașinii sunt alcătuite doar din numele erorilor pe care computerul de bord le-a detectat. Să presupunem că avem două erori: P1111 și P0000. În istoricul erorilor apare P1111, dar P0000 nu apare. Utilizatorul va putea filtra doar după eroarea P1111 deoarece a fost detectată cândva în trecut, în schimb nu va putea efectua o filtrare dupa P0000 deoarece aceasta nu a fost detectată până în acel moment, iar rezultatul filtrării ar fi nul.

Listing 3.7: Formarea spinnerului pentru filtrare.

```
ArrayList<String> setErrors = new ArrayList<String>();
setErrors.add("All");
myRef2.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for (DataSnapshot ds : snapshot.getChildren()) {
            String errorName = ds.child("errorName").getValue().
                                                    toString();
            if (!setErrors.contains(errorName))
                setErrors.add(errorName);
        }
        mySpinner = (Spinner) findViewById(R.id.spinner1);
        myAdapter = new ArrayAdapter<String>(getApplicationContext
            (),
                android.R.layout.simple_list_item_1, setErrors);
        myAdapter.setDropDownViewResource(
                android.R.layout.simple_spinner_dropdown_item
            );
        mySpinner.setAdapter(myAdapter);
}
```

În secvența de cod 3.7 se poate observa că, pentru a determina numele erorilor care au fost detectate la mașina respectivă, se parcurge fiecare element din baza de date a erorilor (Figura 3.6), iar numele fiecărei erori este reținut într-un vector o singură dată. Odată ce vectorul este instanțiat, urmează ca spinner-ul din care utilizatorul va selecta numele erorii după care vrea să filtreze, să fie inițializat cu elementele vectorului.

Afișarea istoricului unei erori selectate de către utilizator se realizează prin parcurgerea bazei de date a erorilor, sunt afișate doar cele care au numele ales de către utilizator.

Capitolul 4

Teste și simulări

Testele și simulările efectuate asupra aplicației s-au realizat asupra unui cont în care au fost adăugate două mașini. Configurația hardware pentru fiecare mașină este compusă dintr-un microcontroler de tip Arduino UNO, un LED, două rezistențe, conectori și un modul Bluetooth de tip HC-05. Aplicația a rulat pe sistemul de operare Android. Dispozitivul folosit la teste a fost Huawei MATE 20 Lite.

4.1 Utilizarea aplicației

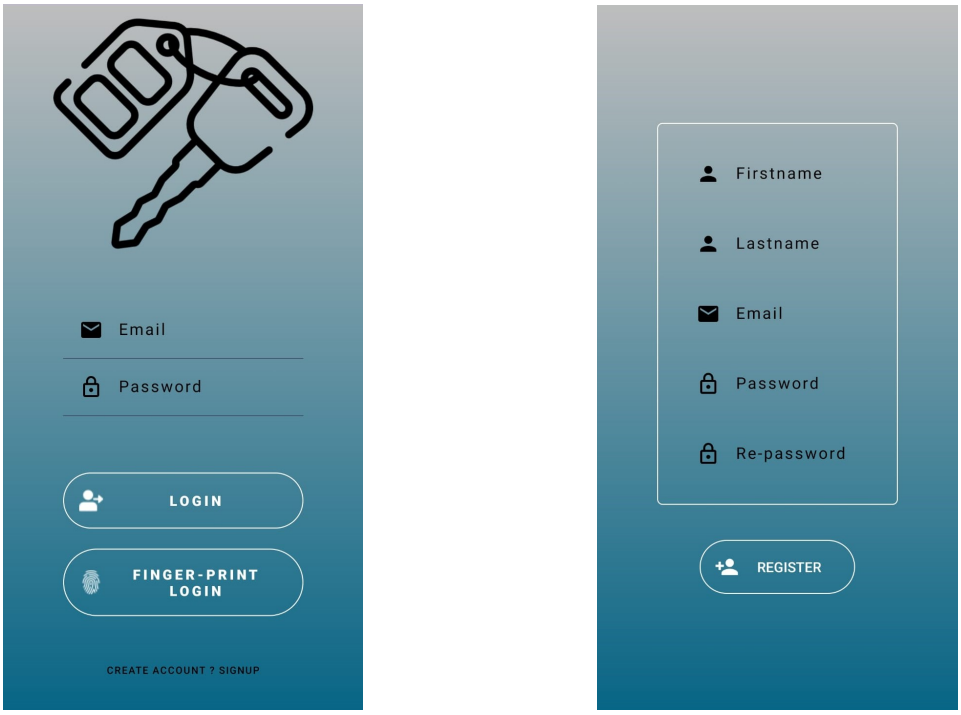
Primul contact pe care utilizatorul îl are cu aplicația Smart Key este cu pagina de logare a acesteia (Figura 4.1a). Acesta are nevoie de un cont personal pentru a folosi aplicația. Pentru a se putea înregistra, el trebuie să acceseze butonul "Create account? Signup". Astfel, va fi redirecționat către zona de creare a conturilor (Figura 4.1b).

El are la dispoziție două metode de login. Acest lucru se întâmplă deoarece aplicația poate rula pe dispozitive care nu au senzori biometrici făcând logarea cu amprentă imposibilă. Un alt motiv este faptul că există anumite persoane care deși au un telefon mobil care suportă logarea cu amprentă, ei preferă logarea clasică din anumite motive.

Prima metodă este cea clasică, în care utilizatorul trebuie să introducă manual datele contului său în câmpul corespunzător (Figura 4.1a). Cea de-a doua metodă de login este realizată prin amprentă, utilizatorul nemaitrebuind să introducă datele contului manual, ea fiind accesată de la butonul "Finger-print login".

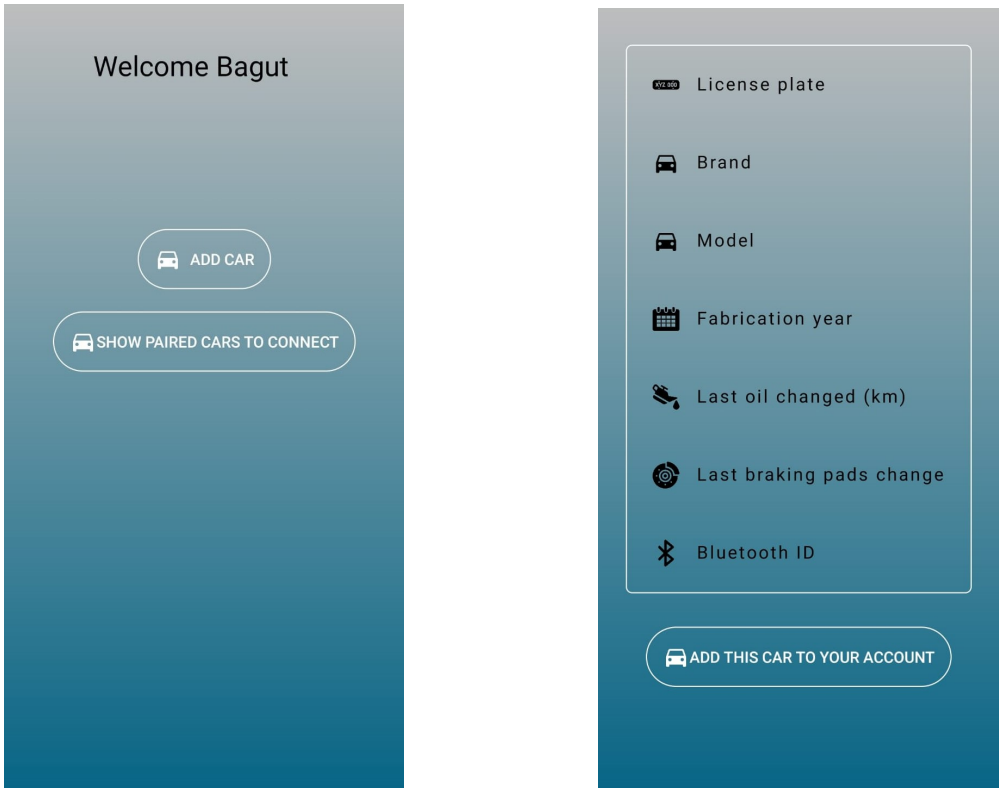
După logare, utilizatorul este redirecționat către meniul aplicației (Figura 4.2a), unde are două butoane. Primul buton, "Add car", se referă la adăugarea unei mașini în cont, pentru a putea folosi aplicația în loc de cheia mașinii, pentru blocarea și deblocarea ușilor și pentru a putea citi datele de la senzorii acesteia în timp real. Prin apăsarea acestui buton, utilizatorul va fi redirecționat spre formularul de adăugare a mașinilor (Figura 4.2b) în care trebuie să completeze anumite date ale mașinii, precum numărul de înmatriculare sau anul în care a fost fabricată mașina.

Primul pas pentru conectarea la o mașină este realizarea împerecherii dispozitivului mobil de pe care rulează aplicația cu modulul Bluetooth instalat pe mașina respectivă. Cel de-al doilea buton prezent în acest meniu este "Show paired cars to connect". Prin apăsarea acestuia, aplicația îi va arăta utilizatorului, doar modulele Bluetooth corespunzătoare mașinilor pe care le are adăugate în cont.



(a) (b)

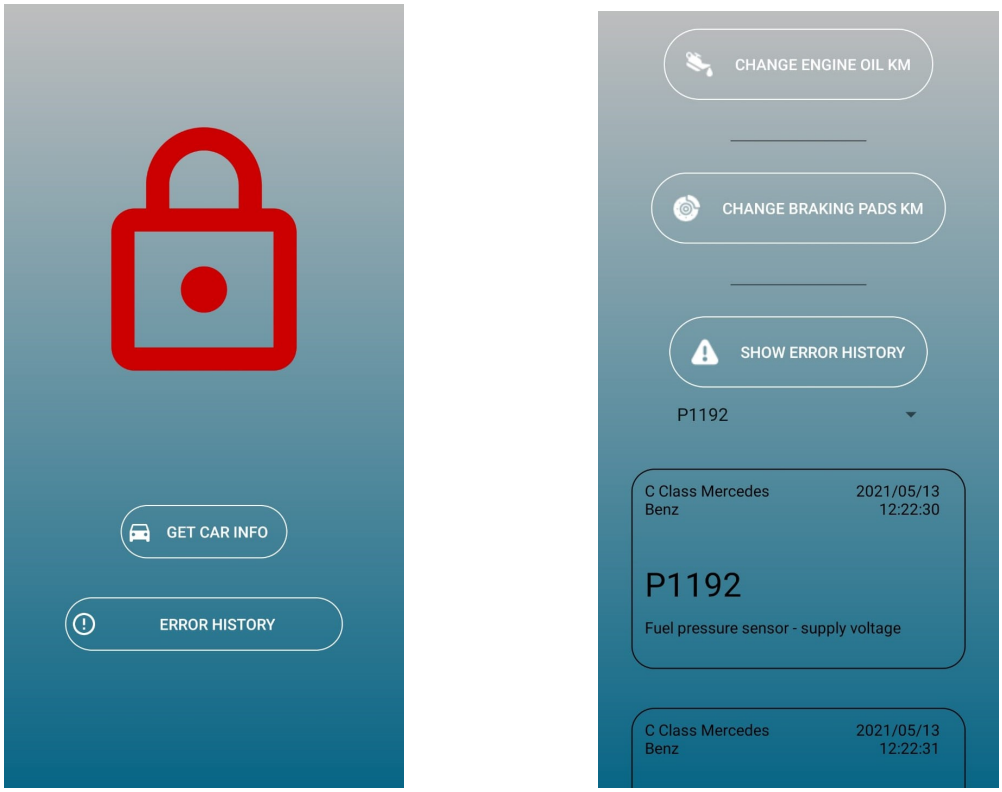
Figura 4.1: Prima pagină (a); Pagina pentru înregistrare(b)



(a) (b)

Figura 4.2: Meniul contului (a); Formular pentru adăugarea unei mașini noi (b).

După conectarea la o mașină, utilizatorul este redirecționat către meniul mașinii (Figura 4.3a). În acest meniu este prezentă cheia de blocare a ușilor mașinii (Subsecțiunea 4.4). Pe lângă această cheie mai sunt prezente două butoane. Prin apăsarea butonului "GET CAR INFO" utilizatorul va fi redirecționat către pagina de unde poate vedea datele de la senzorii mașinii pe care le primește în timp real (Secțiunea 4.3).



(a) (b)

Figura 4.3: Meniul mașinii (a); Istoricul erorilor (b).

Prin utilizarea butonului "ERROR HISTORY", el este redirecționat către pagina în care poate să vadă istoricul erorilor pe care le-a generat mașina respectivă (Figura 4.3b). Pentru a fi mai ușor de găsit o eroare specifică, acestea se pot filtra după numele erorii. Tot din acestă pagină, utilizatorul poate să își actualizeze numărul de kilometri la care s-a efectuat schimbul de ulei și de plăcuțe de frână.

4.2 Setul de date

Setul de date [28] pe care lucrează aplicația dezvoltată are scopul de a simula cum ar trebui să primească datele în timp real de la calculatorul de bord al mașinii (OBD2). Acesta ar trebui să trimită datele de la senzorii mașinii către Arduino-ul conectat pe mașina respectivă, iar acesta să le trimită mai departe catre aplicație.

Setul de date inițial este format din datele citite de la computerele de bord a 14 mașini distincte și anumite date despre mașina respectivă. Are în componență 47515 de înregistrări. Setul de date conține marca mașinii, modelul acesteia, anul de fabricație, puterea motorului, tipul de transmisie (automată, manuală sau semi-automată), un

număr de identificare unic pentru fiecare mașină, presiunea barometrică, temperatura lichidului de răcire al mașinii, nivelul combustibilului din rezervor, procentul la care este utilizată puterea motorului, temperatura aerului ambiental, turația motorului, presiunea colectorului de admisie, debitul masei de aer, tipul de combustibil, temperatura de admisie a aerului, viteza, timpul în care a stat motorul pornit, procentul de apăsare al pedalei de accelerație, timpul necesar arderii combustibilului, eroarea generată de OBD2 (doar dacă este cazul), minutul și ora din zi, respectiv data în care s-a efectuat citirea.

Acest set de date are prea multe date inutile pentru aplicația dezvoltată. Un exemplu de câmpuri inutile pot fi datele (câmpurile pentru minutul, ora, ziua, luna și anul în care a fost detectată eroarea respectivă). Datorită acestui lucru am efectuat o filtrare a parametrilor din setul inițial, păstrând doar datele de la anumiți senzori, relevante pentru aplicație .

Am păstrat datele generate de senzorul de temperatură pentru lichidul de răcire, cel pentru detectarea presiunii barometrice, pentru detectarea nivelului combustibilului de rezervor, pentru aflarea turației motorului; datele de la senzorul de presiune al galeriei de admisie, de la senzorul pentru detectarea poziției pedalei de accelerație și datele de la debitul masic de aer. Pe lângă această filtrare am adăugat și alte date precum numărul de kilometri înregistrați la momentul respectiv în calculatorul de bord al mașinii.

În baza de date virtuală ar trebui ca pentru fiecare mașină în parte a fiecărui utilizator al aplicației să existe câte o relație (tabelă) care să conțină datele mașinii. Datorită acestui lucru datele de la senzorii fiecărei mașini sunt stocați în fișiere diferite. După ce am efectuat operațiile de mai sus, setul de date arată ca în figura de mai jos (Figura 4.4):

ENGINE_POWER	BAROMETRIC_PRESSURE(KPA)	ENGINE_COOLANT_TEMP	FUEL_LEVEL	ENGINE_R	INTAKE_MANIFOLD_PRESSURE	MAF	AIR_INTAKE_TEMP	THROTTLE_POS	DTC_NUMBER	TIMING_ADVANCE	KILOMETERS
3	96	96	20,80%	750	43	2.18	49	18%		63.90%	25000
3	96	95	21,20%	625	62	2.32	49	18%		61.20%	25100
3	96	93	20,80%	6000	27	19.65	46	26%	P1335	65.10%	25200
3	96	91	17,60%	1105	70	2.45	47	18%		67.80%	25300
3	96	0	14,50%	975	68	2.35	48	18%	P1147	67.80%	25400
3	96	90	13,70%	883	53	2.2	50	18%		67.50%	25500
3	96	90	13,70%	842	53	2.14	50	18%		64.70%	25600
3	96	90	13,70%	746	57	2	0	18%	P1148	64.70%	25700
3	96	91	14,10%	727	76	2	50	18%		65.10%	25800
3	96	92	13,70%	1863	46	13.71	44	23%		69.00%	25900
3	96	92	16,10%	1486	86	9.09	44	21%		73.70%	26000
3	96	92	16,50%	896	76	0	45	18%	P1146	66.70%	26100
3	96	91	16,90%	834	41	2.29	48	18%		66.30%	26200
3	96	92	18,40%	2125	37	16.98	41	24%		69.40%	26300
3	96	92	20,40%	1177	38	2.51	47	18%		65.90%	26400
3	96	91	23,50%	1985	33	16.7	43	24%		65.10%	26500
3	96	91	25,50%	950	34	2.39	47	18%		67.80%	26600
3	96	90	25,90%	1732	36	10.98	42	22%		69.40%	26700
3	96	90	26,30%	1901	39	13.85	42	23%		68.20%	26800
3	96	90	25,90%	1657	32	7.64	42	22%		72.90%	26900
3	96	90	25,90%	803	57	2.39	45	18%		65.90%	27000
3	96	90	25,10%	2020	42	17.87	42	25%		66.30%	27100

Figura 4.4: Set de date CSV.

Ultima prelucrare a setului inițial de date este reprezentată de transformarea fișierelor de tip CSV în fișiere de tip JSON (Figura 4.5). Această conversie se realizează pentru a se putea citi datele mai ușor în aplicație.

```
[
  {
    "ENGINE_POWER": 5,
    "BAROMETRIC_PRESSURE(KPA)": 100,
    "ENGINE_COOLANT_TEMP": 80,
    "FUEL_LEVEL": "48,60%",
    "ENGINE_RPM": 1009,
    "INTAKE_MANIFOLD_PRESSURE": 49,
    "MAF": "4,49",
    "AIR_INTAKE_TEMP": 59,
    "THROTTLE_POS": "25%",
    "DTC_NUMBER": "",
    "TIMING_ADVANCE": "56,9%",
    "KILOMETERS": 14000
  },
  {
    "ENGINE_POWER": 5,
    "BAROMETRIC_PRESSURE(KPA)": 100,
    "ENGINE_COOLANT_TEMP": 80,
    "FUEL_LEVEL": "48,60%",
    "ENGINE_RPM": 1003,
    "INTAKE_MANIFOLD_PRESSURE": 52,
    "MAF": "4,51",
    "AIR_INTAKE_TEMP": 59,
    "THROTTLE_POS": "25%",
    "DTC_NUMBER": "",
    "TIMING_ADVANCE": "56,5%",
    "KILOMETERS": 14100
  },
```


Figura 4.5: Set de date JSON.

4.3 Citirea datelor în timp real

Pentru a putea prelucra datele din computerul de bord al maşinii, aplicaţia trebuie să le primească. Acest lucru este simulat prin citirea datelor din fişierul corespunzător maşinii respective. Totodată, pentru a putea simula și trecerea timpului în lumea reală, 24 de înregistrări citite la un interval de timp prestabilit înseamnă 24 de ore în viaţa reală.

Valoarea normală pentru temperatura lichidului de răcire este între 75 și 85 de grade Celsius. Știind acest lucru, utilizatorul poate vedea în cât timp ajunge motorul la temperatura optimă. Dacă timpul de încălzire este unul îndelungat, acesta este conștient de faptul că mașina are o problemă la circuitul de răcire și trebuie să o ducă la un service auto pentru a rezolva această problemă.

Datorită faptului că în pagina destinată pentru dispunerea datelor în timp real (Figura 4.6) utilizatorul poate să vadă exact cum se actualizează valorile pe câmpurile datelor de la senzori, acesta poate preveni defecțiuni grave ale blocului motor.

4.4 Detectarea erorilor

Detectarea erorilor se realizează la nivelul computerului de bord al mașinii. Aplicația primește numele erorii în momentul în care aceasta apare la fel cum primește și datele de la senzorii mașinii. Aceasta doar arată numele erorii respective în câmpul Dct Number din Figura 4.6.

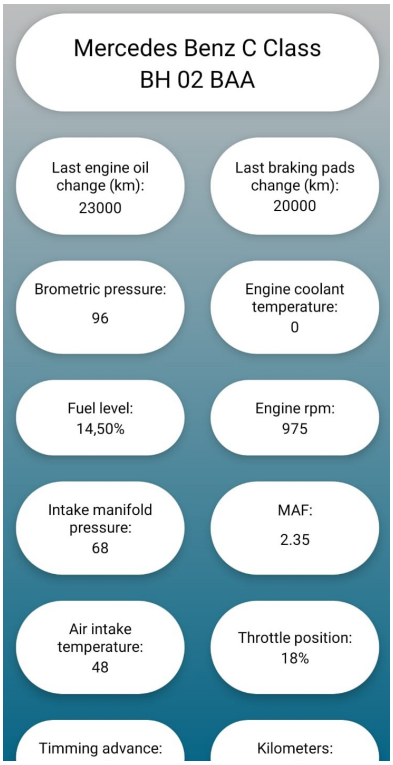


Figura 4.6: Citirea datelor în timp real.

În cazul în care o eroare se repetă de mai multe ori consecutiv, pentru a nu trimite notificări în mod excesiv, aplicația va înștiința utilizatorul la un interval de timp de 3 secunde în aplicație (3 ore în lumea reală). Totuși, pentru ca utilizatorul să aibă istoricul erorilor complet, acestea se vor salva tot timpul în baza de date, indiferent de câte erori la fel apar consecutiv.

În eventualitatea în care proprietarul mașinii dorește să o vândă, el trebuie să aibă acces la toate erorile pe care le-a generat mașina de-a lungul timpului. Vizionarea tuturor erorilor este importantă și în cazul în care autovehiculul este dus la mecanic deoarece acesta poate detecta mult mai repede cauza erorilor. Astfel, el nu va mai trebui să piardă timpul cu căutarea erorilor, putând să se concentreze pe rezolvarea lor. Datorită acestor fapte, aplicația va salva câte o înregistrare în baza de date la fiecare apariție a erorilor. Înregistrarea este compusă din marca și modelul mașinii, numele și o descriere scurtă a erorii, data exactă la care s-a detectat și emailul contului din care face parte autovehiculul.

4.5 Notificare pentru schimbarea de ulei

În cazul schimbării de ulei, fiecare mașină are un anumit interval de kilometri la care constructorul recomandă efectuarea schimbului. Pentru a vedea care este intervalul recomandat pentru o mașină specifică, aplicația este conectată la bazele de date ale constructorilor în care se găsește informația dorită. Acest lucru este simulat din nou printr-un fișier de tip JSON, în care se află doar modelul de mașină și intervalul specific (Figura 4.7).

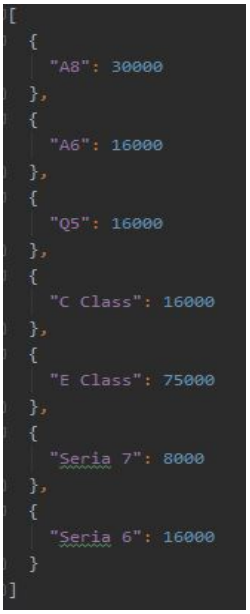


Figura 4.7: Interval de schimb de ulei.

Notificările sunt afișate o singură dată în fiecare zi (o dată la 24 de înregistrări citite). Repetarea notificării are scopul de a-l ține mereu la curent pe proprietarul mașinii în legătură cu numărul de kilometri (Figura 4.8a) pe care îi mai poate parcurge înainte de a fi schimbat. În cazul în care conducătorul depășește acel interval din varii motive, aplicația va afișa notificări, păstrând același interval (24 de ore), însă textul va fi schimbat (Figur 4.8b) astfel încât utilizatorul să își poată da seama că schimbul de ulei trebuie efectuat pentru a evita diferite defecțiuni grave la motor.

(a) (b)

Figura 4.8: Notificare când numărul de kilometri nu este depășit (a); Notificare când numărul de kilometri este depășit (b).

4.6 Notificare pentru schimbarea plăcuțelor de frână

În cazul schimbării plăcuțelor de frână, intervalul de schimbare recomandat de către majoritatea marilor constructori de mașini este de 40000 de kilometri. La fel ca și în cazul notificărilor care înștiințează utilizatorul în legătură cu schimbarea de ulei, notificările pentru schimbarea plăcuțelor de frână apar la același interval de timp. În notificare este specificat numărul de kilometri rămași până în momentul efectuării schimbului (Figura 4.9a). În cazul în care este depășit acest interval se va schimba notificarea, păstrându-se intervalul la care apare, astfel încât proprietarul să realizeze

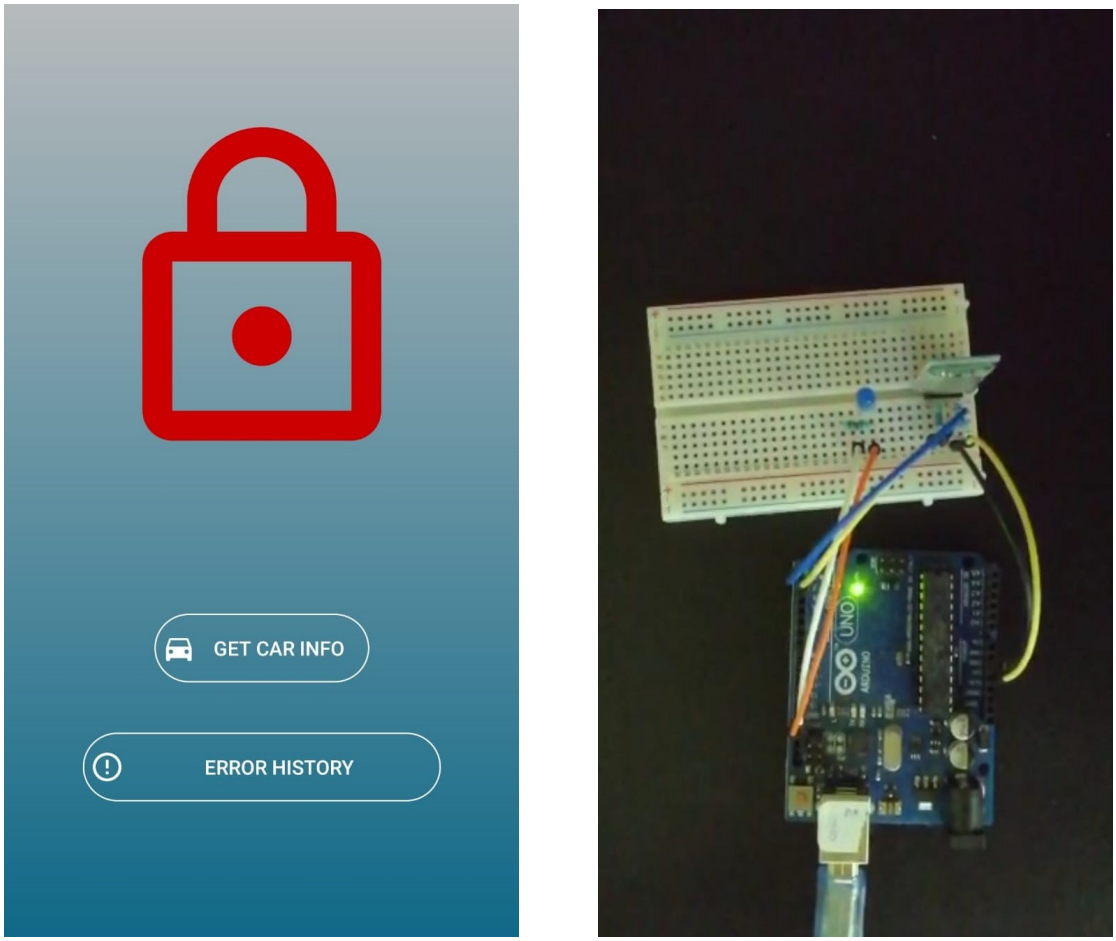
că este important să efectueze schimbul plăcuțelor pentru a nu se supune la riscuri inutile.

(a) (b)

Figura 4.9: Notificare când numărul de kilometri nu este depășit (a); Notificare când numărul de kilometri este depășit (b).

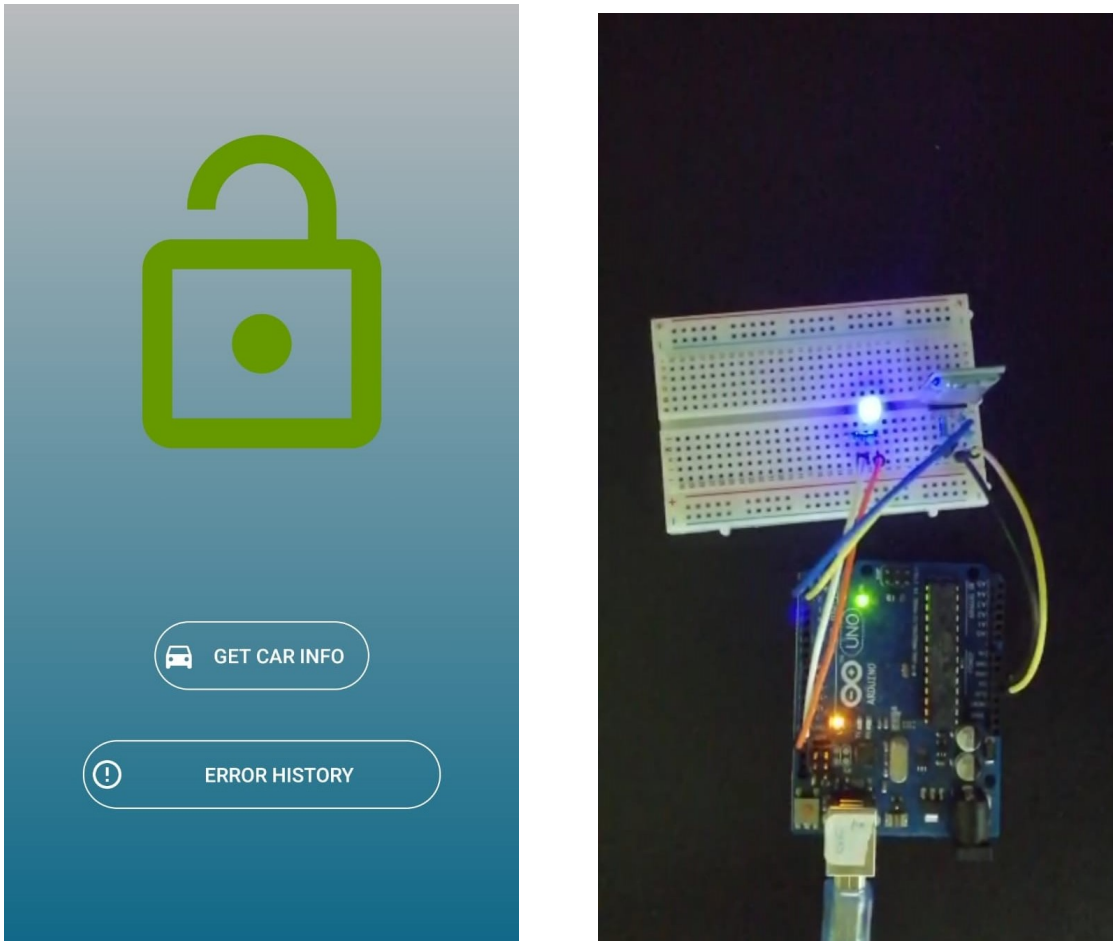
4.7 Blocarea și deblocarea ușilor

Blocarea și deblocarea ușilor mașinii se realizează prin intermediul meniului mașinii. Acest lucru este simulat printr-un LED, care în momentul în care ușa mașinii ar fi blocată este stins, respectiv în momentul în care mașina ar fi deblocată este deschis.



(a) (b)

Figura 4.10: Cheia mașinii când aceasta este blocată (a); Stare 1 LED (b).



(a) (b)

Figura 4.11: Cheia mașinii când aceasta este deblocată (a); Stare 2 LED (b).

În momentul în care butonul respectiv este roșu (Figura 4.10a), înseamnă că mașina este blocată deci LED-ul este stins (Figura 4.10b). Pentru a putea debloca mașina, utilizatorul apasă pe buton, iar acesta devine verde (Figura 4.11a) și LED-ul se aprinde (Figura 4.11b).

În mod implicit, în momentul în care utilizatorul accesează cheia mașinii după ce redeschide aplicația, mașina va figura ca fiind închisă. Astfel dacă utilizatorul lasă mașina deblocată și închide aplicația, în momentul în care revine la cheia mașinii prin redeschiderea aplicației, mașina va figura ca fiind blocată. Pentru a rezolva această problemă, el trebuie să apese din nou butonul respectiv de două ori. La prima apăsare mașina va figura ca fiind deblocată, iar la a doua apăsare mașina se va bloca.

Capitolul 5

Concluzii și direcții viitoare

5.1 Concluzii

Deși există foarte multe aplicații similare cu Smart Key, aplicația dezvoltată are mai multe avantaje decât celelalte aplicații deja existente pe piață. Din aceste avantaje enumerăm: aplicabilitatea mărită, posibilitatea de urmărire a mai multor parametrii ai mașinii, avantajul economic dar și semnalarea imediată a problemelor mașinii.

Aplicabilitatea mărită a aplicației Smart Key comparată cu alte aplicații similare deja existente pe piață (ex Mercedes Me, My BMW app) se datorează faptului că ea permite introducerea în aplicație a unui număr nelimitat de mașini. Acestea pot fi dezvoltate de către orice producător din industria constructoare de mașini.

În ceea ce privește numărul parametrilor urmăriți cu ajutorul aplicației, Smart Key oferă utilizatorului posibilitatea de a urmări cu ușurință numărul de kilometri la care este recomandată schimbarea de ulei de motor și de plăcuțe de frână. Acest lucru se realizează prin salvarea numărului de kilometri la care s-a efectuat ultimul schimb de ulei de motor, respectiv de plăcuțe de frână. În funcție de intervalul de kilometri la care se recomandă (de către constructor) înlocuirea uleiului de motor și al plăcuțelor de frână, aplicația dezvoltată va trimite în mod regulat notificări. Datorită acestui sistem, în marea majoritate a cazurilor utilizatorul va efectua schimburile la timp.

Din punct de vedere economic, întrucât aplicația permite și introducerea mașinilor mai vechi în baza de date, Smart Key este avantajoasă în special pentru firmele de transport public, firmele de curierat sau cele de taxi. Astfel, firma va putea beneficia de avantajele utilizării unei astfel de aplicații chiar dacă mașinile sunt vechi, nemaifiind necesar să se cumpere mașini noi care să se încadreze în criteriile aplicațiilor deja existente pe piață. Datorită costurilor mici de utilizare ale aplicației, aceasta devine accesibilă unui eșantion mai mare de persoane.

Aplicația primește datele de la calculatorul de bord al mașinii în timp real, inclusiv erorile detectate de către acesta. Mai apoi, aplicația analizează aceste date și le transmite mai departe utilizatorului. În momentul în care este detectată o eroare, aplicația îl înștiințează pe acesta prin notificări legate de natura erorii.

În plus, aplicația dezvoltată are și o funcție specială care permite utilizarea ei ca și o cheie virtuală. Cu ajutorul acesteia, ușile mașinii pot fi blocate și deblocate. Având în vedere acest lucru, în cazul închiderii cheii în mașină, cu ajutorul aplicației, utilizatorul poate să deblocheze ușile și să își recupereze cheile în cel mai scurt timp posibil.

5.2 Limitări ale aplicației

Pe lângă avantajele pe care le oferă utilizarea Smart Key, există și unele limitări ale acesteia. Cea mai importantă limitare a aplicație este că în momentul în care utilizatorul poate uita mașina deblocată și să se îndepărteze la o distanță considerabilă de ea, aceasta rămâne deschisă. În momentul în care se pierde conexiunea, mașina va rămâne deblocată, proprietarul putând să o blocheze doar dacă se reîntoarce la mașină.

Un alt dezavantaj apare în cazul în care managerul firmei de transport dorește să vadă anumite date despre o mașină. Acest lucru este realizabil doar dacă se conecteză la mașina respectivă, lucru imposibil dacă aceasta este în teren în acel moment.

Din cauza limitărilor Firebase, aplicația are toate datele stocate într-o sigură bază de date. Acestea ar trebui să fie stocate în baze de date diferite. Ar trebui să existe o bază de date pentru utilizatori, una pentru mașini și una în care să fie stocate erorile detectate de fiecare mașină.

5.3 Direcții viitoare

Ca și direcții viitoare de dezvoltare ale aplicației îmi propun separarea acesteia în două module. Primul modul să fie destinat șoferilor și să conțină cheia virtuală, datele citite în timp real și istoricul erorilor. Cel de-al doilea modul să fie destinat managerului firmei sau directorului de transport din firma respectivă. Acest modul ar urma să conțină posibilitatea de a vedea datele mașinii în timp real pentru a putea verifica dacă angajații își fac treaba și posibilitatea de a vedea istoricul erorilor generate de orice mașină înregistrată la firmă.

În momentul în care un angajat utilizează mașina, el poate să vadă anumite date în timp real. Acestea ar urma să fie salvate într-o bază de date. În momentul în care o persoană autorizată din firmă dorește să vadă datele unei mașini, aplicația folosită de el citește din baza de date ceea ce este salvat când un angajat utilizează mașina. Astfel s-ar putea realiza cel de al-doilea modul al aplicației.

Un alt mod prin care se poate îmbunătăți aplicația în viitor este gestionarea puterii conexiunii dintre telefonul utilizatorului și mașină. În acest caz, s-ar elimina posibilitatea ca utilizatorul să uite mașina deblocată. Pentru aceasta, ar trebui ca aplicația să aibă în vedere puterea semnalului dintre cele două dispozitive în timp real. Dacă semnalul ar scădea sub o anumită valoare, atunci înseamnă că utilizatorul este la o anumită distanță față de mașină; iar dacă aceasta este deblocată, atunci aplicația ar bloca-o automat.

Bibliografie

[1] W. E. FORUM, "These are the countries with the most vehicles per person," 2015. URL: https://www.weforum.org/agenda/2015/10/these-are-the-countries-with-the-most-vehicles-per-person/ [accesat: 2021-31-05].

[2] "Different types of sensors used in automobiles." URL: https://www.elprocus.com/different-types-of-sensors-used-in-automobiles/ [accesat: 2021-10-05].

[3] GEOTAB, "What is obdii? history of on-board diagnostics," 2020. URL: https://www.geotab.com/blog/obd-ii/ [accesat: 2021-02-15].

[4] M. Jard, H. Shah, and A. Bhaskar, "Empirical evaluation of bluetooth and wifi scanning for road transport," 01 2013.

[5] H. Xuguang, "An introduction to android," 2009. URL:https://walidumar.my.id/buku.elektronik/networking.umum/Introduction%20to%20Android.pdf [accesat: 2021-04-29].

[6] J. Fernando, "What is an api ? how to call an api from android ?," 2016. URL: https://droidmentor.com/api-call-api-android/ [accesat: 2021-04-29].

[7] Y. A. Badamasi, "The working principle of an arduino," in *2014 11th international conference on electronics, computer and computation (ICECCO)*, 2014.

[8] R. Sanderson, "Introduction to remote sensing," *New Mexico State University*, 2010.

[9] Stratstone, "Mercedes me," 2021. URL: https://www.stratstone.com/mercedes-benz/mercedes-me/ [accesat: 2021-01-15].

[10] BMW, "My bmw app," 2021. URL: https://www.bmw.ro/ro/topics/offers-and-services/my-bmw-app-overview.html [accesat: 2021-01-15].

[11] B. GROUP, "Bmw announces bmw digital key plus with ultra-wideband technology coming to the bmw ix," 2021. URL: https://www.press.bmwgroup.com/global/article/detail/T0324128EN/bmw-announces-bmw-digital-key-plus-with-ultra-wideband-technology-coming-to-the-bmw-ix [accesat: 2021-01-14].

[12] Economica.net, "Câte maşini sunt în românia, câte sunt mai vechi de 10 ani şi câţi oameni au murit în accidente rutiere – ins/drpciv,"

2020. URL: https://www.economica.net/cate-masini-sunt-in-romania-masini-vechi-parc-auto-accidente-morti-accidente-auto$_1$85398.*html*[*accesat* : 2021 − 01 − 14].

[13] Phillips, "Phillips hue." URL: https://www.philips-hue.com/ro-ro [accesat: 2021-01-15].

[14] Uptodown, "Universal tv remote." URL: https://universal-tv-remote.uptodown.com/android [accesat: 2021-01-15].

[15] A. Katzenbach, "Automotive," in *Concurrent engineering in the 21st century*, pp. 607–638, Springer, 2015.

[16] S. Bennett, *A History of Control Engineering 1930-1955*. INSTITUTION OF ENGINEERING T, Dec. 1993.

[17] K. Yoo, K. Simpson, M. Bell, and S. Majkowski, "An engine coolant temperature model and application for cooling system diagnosis," *SAE Transactions*, vol. 109, pp. 950–960, 2000. URL: http://www.jstor.org/stable/44634280.

[18] "Coudri de eroare audi." URL: https://4mgv.com/coduri-de-eroare-audi/ [accesat: 2021-02-05].

[19] V. Varshney, R. K. Goel, and M. A. Qadeer, "Indoor positioning system using wi-fi bluetooth low energy technology," in *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, pp. 1–6, 2016.

[20] S. Tanasa, *Java : de la 0 la expert*. Iasi: Polirom, 2011.

[21] R. Meier, *Professional Android 4 application development*. John Wiley & Sons, 2012.

[22] "Android api." URL: https://www.quora.com/What-is-API-for-android-mean-What-software-apps-do-I-need-to-do-an-API-programming-Is-there-any-website-for-tutorial [accesat: 2021-04-29].

[23] T. N. H. J. Toby Teorey, Sam Lightstone, *Database modeling and design : logical design*. Amsterdam Boston: Morgan Kaufmann Publishers, 2011.

[24] L. Moroney, Moroney, and Anglin, *Definitive Guide to Firebase*. Springer, 2017.

[25] D. Stebenson, "What is firebase? the complete story, abridged.," 2018. URL: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [accesat: 2021-03-05].

[26] H. Yahiaoui, *Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase*. Packt Publishing Ltd, 2017.

[27] M. Banzi, *Getting started with Arduino*. Sebastopol, CA: O'Reilly, 2011.

[28] "Obd-ii datasets." URL: https://www.kaggle.com/cephasax/obdii-ds3 [accesat: 2021-03-03].