НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і
спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050102 «Комп'ютерна інженерія»

на тему: «Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android»

Виконав:
студент IV курсу, групи КВ-52
Юрович Іван Васильович _____

Керівник:
к. т. н. доц. каф. СПСКС
Клятченко Ярослав Михайлович _____

Консультант з нормоконтролю:
к. т. н. доц. каф. СПСКС
Клятченко Ярослав Михайлович _____

Рецензент:
_____
_____ _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студента

Юровича Івана Васильовича

1. Тема проекту «Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android», керівник проекту доц. каф. СПСКС, к.т.н. Клятченко Ярослав Михайлович, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту «14» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - Аналіз існуючих рішень діагностики автомобілів та обґрунтування теми дипломного проекту;
   - особливості розробки Android додатку;
   - особливості реалізації Android додатку для діагностики автомобіля;
   - особливості реалізації серверу для збереження та обміну даними.

5. Перелік обов'язкового графічного матеріалу:
   - архітектура модуля Bluetooth (схема структурна);
   - архітектура модуля Network (схема структурна);
   - процедура роботи з мережею на клієнтській стороні (схема алгоритму);
   - процедура роботи з мережею на серверній стороні (схема алгоритму).

– презентація за темою роботи.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М. доц. каф. СПСКС, к.т.н.		

7. Дата видачі завдання «__» ____ 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	17.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Аналіз існуючих рішень	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення програмного продукту	03.03.2019	
6.	Відлагодження програмного продукту	10.03.2019	
7.	Підготовка матеріалів другого розділу дипломного проекту	25.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	10.04.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	20.04.2019	
11.	Підготовка графічної частини дипломного проекту	25.05.2019	
12.	Оформлення документації дипломного проекту	31.05.2019	

Студент І.В. Юрович

Керівник проекту Я.М. Клятченко

АНОТАЦІЯ

Об’єкт розробки – створення комп’ютерної системи для моніторингу автомобілів оснащених інтерфейсом OBDII.

Комп’ютерна система дозволяє: збирати дані з різних апаратних модулів автомобіля використовуючи протокол OBDII; зберігати дані у файли CSV формату; здійснювати передачу даних бездротовими каналами зв’язку. Передбачена можливість підключення одного смартфона до іншого для дистанційного зчитування даних. В процесі розробки була використана технологія бездротового зв’язку Bluetooth та стандарт промислової мережі CAN. В якості бази даних для серверу використовувалась MySQL.

В ході розробки:

– проведено аналіз методів побудови клієнт-серверних систем;
– сформульовані вимоги до системи моніторингу автомобіля;
– розроблено програмний модуль для взаємодії з Bluetooth пристроями;
– розроблено програмний модуль для обміну повідомленнями через протокол OBDII;
– розроблено програмний модуль для роботи з серверними та клієнтськими сокетами;
– розроблено серверний додаток для передачі даних між клієнтами та збереження даних у базу даних;
– розроблено користувацький мобільний додаток для моніторингу автомобілів, збереження та обміну даними між користувачами.

Упровадження цієї системи дозволить станціям технічного обслуговування (СТО) автомобілів дистанційно проводити моніторинг автомобілів. Це зменшить витрати коштів та часу для аналізу поломки автомобіля як зі сторони користувача так і зі сторони СТО.

Ключові слова:

Комп’ютерна система моніторингу автомобілів, станція технічного обслуговування (СТО), сокет, Bluetooth, OBDII, CAN, MySQL, Android, CSV.

ABSTRACT

The object of this project is creating a computer system for car monitoring that has OBDII interface.

The computer system allows: to collect data from various hardware modules of the car, using the protocol OBDII; storing data in a CSV file format; transmit data via wireless communication channels. It is possible to connect one smartphone to another for remote data processing. In the development process, Bluetooth wireless technology and the CAN standard were used. MySQL is used as the database for the server.

During development:

- the analysis of methods of constructing client-server systems were carried out;
- requirements for the car monitoring system were formulated;
- developed a software module for interacting with Bluetooth devices;
- a software module for messaging through the OBDII protocol has been developed;
- a software module for working with server and client sockets has been developed;
- developed server application for data transmission between customers and data storage in the database;
- mobile application for monitoring cars, storing and exchanging data between users has been developed.

Implementation of this system will allow the service station of cars to remotely monitor the cars. This is a reduction in the cost of time and time to analyze car breakdown, both from the user's side and through the side of the service station.

Keywords:

Computer monitoring system of cars, service station, socket, Bluetooth, OBDII, CAN, MySQL, Android, CSV.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Додаток для дистанційного	4		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Технічне завдання.			
	А4	ІАЛЦ. 045490.003 ТП	Додаток для дистанційного	2		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Відомість технічного			
			проекту.			
	А4	ІАЛЦ. 045490.004 ПЗ	Додаток для дистанційного	61		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Пояснювальна записка.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Юрович І.В.			Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android. Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.005 Д1	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Архітектура модуля			
			Bluetooth.			
			Схема структурна.			
	А4	ІАЛЦ. 045490.006 Д2	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Архітектура модуля			
			Network.			
			Схема структурна.			
	А4	ІАЛЦ. 045490.007 Д3	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Процедура роботи з			
			мережею на клієнтській			
			стороні.			
			Схема алгоритму.			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.008 Д4	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Процедура роботи з			
			мережею на серверній			
			стороні.			
			Схема алгоритму.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

ЗМІСТ

					ІАЛЦ. 045490.002 ТЗ		
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***			
Розроб.		Юрович І.В			***Технічне завдання*** Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android	***Літ.***	***Лист*** 1 / ***Листів*** 4
Перев.		Клятченко Я.М.					
Н. контр.		Клятченко Я.М.				**КПІ ім. Ігоря Сікорського, ФПМ, КВ-52**	
Затв.		Тарасенко В.П.					

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android».

Галузь застосування: збирання та обмін даними між станціями технічного обслуговування автомобілів та їх клієнтами для пришвидшення процесу діагностики автомобіля.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи ступеня «бакалавр комп'ютерної інженерії», затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення мобільного додатку для збору, обробки, аналізу та обміну даних з автомобілів, обладнаних інтерфейсом OBDII.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Android починаючи з версії 6.0 Marshmallow (охоплення 75.5% користувачів Android);

- можливість зчитування даних з автомобіля за допомогою Bluetooth ELM контролера, використовуючи OBDII інтерфейс;
- можливість логування даних у CSV файли;
- можливість збереження даних на сервері;
- можливість передачі даних (показників) між користувачами;
- можливість передачі повідомлень між користувачами (чат).

5.2. Вимоги до програмного та апаратного забезпечення користувача

- пристрій або емулятор з операційною системою Android 6.0 і вище;
- для обміну даними між користувачами потрібна наявність доступу до мережі Internet (GPRS, EDGE, 3G, 4G, Wifi);
- ELM контролер з Bluetooth модулем.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	17.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Аналіз існуючих рішень	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення програмного продукту	03.03.2019	
6.	Відлагодження програмного продукту	10.03.2019	
7.	Підготовка матеріалів другого розділу дипломного проекту	25.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	10.04.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	20.04.2019	
11.	Підготовка графічної частини дипломного проекту	25.05.2019	
12.	Оформлення документації дипломного проекту	30.05.2019	
13	Попередній огляд матеріалів диплому на кафедрі	31.05.2019	

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045490.004 ПЗ	Додаток для дистанційного	61		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Пояснювальна записка.			
	А4	ІАЛЦ. 045490.005 Д1	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Архітектура модуля			
			Bluetooth.			
			Схема структурна.			
	А4	ІАЛЦ. 045490.006 Д2	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Архітектура модуля			

			прин		ІАЛЦ.045490.003 ТП		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Юрович І.В.			Додаток для дистанційного діагностування автомобіля на базі OBD-II під мобільну платформу Android. Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Network.			
			Схема структурна.			
	А4	ІАЛЦ. 045490.007 Д3	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Процедура роботи з			
			мережею на клієнтській			
			стороні.			
			Схема алгоритму.			
	А4	ІАЛЦ. 045490.008 Д4	Додаток для дистанційного	1		
			діагностування автомобіля			
			на базі OBD-II під			
			мобільну платформу			
			Android.			
			Процедура роботи з			
			мережею на серверній			
			стороні.			
			Схема алгоритму.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

1. Комп’ютерна система моніторингу автомобілів – засоби для зчитування даних з основних керуючих систем автомобіля;
2. Сокет – програмний інтерфейс, що забезпечує передачу даних між процесами;
3. Станція технічного обслуговування (СТО) – установа, що здійснює технічне обслуговування транспортних засобів;
4. Android – операційна система для мобільних пристроїв;
5. Bluetooth – технологія бездротової передачі даних;
6. CSV – формат файлів, в якому збереження даних досягається за рахунок розділення даних комами;
7. MySQL – система керування реляційними базами даних;
8. OBDII - інтерфейс для діагностування сучасних автомобілів;
9. SSL – криптографічний протокол для встановлення безпечного з’єднання;
10. TLS – покращена версія SSL.

ВСТУП

Активний ритм життя призводить до того що люди стають залежними від багатьох технічних засобів сьогодення. Зараз вже важко уявити рутинні справи без транспортних засобів. А, отже, і питання зручного та швидкого діагностування ТЗ набуває великого значення.

Найбільш простим і зручним способом мобільного діагностування автомобільних транспортних засобів є сканери з використанням інтерфейсу OBDII. Такі сканери бувають як з провідним інтерфейсом (USB) так і з безпровідним (Bluetooth, WiFi). Автомобілі з OBDII інтерфейсом почали виготовляти починаючи з 1996 року. А на даний момент цей інтерфейс є обов'язковим для всіх виробників США та Євросоюзу.

На даний момент вже існує багато методів діагностування автомобілів за допомогою інтерфейсу OBDII. Більшість з них вміють робити діагностування автомобіля, зчитувати та скидати помилки і навіть робити логування показників системи. Проте, всі вони потребують глибоких знань структури ТЗ, що спричиняє складнощі а іноді і неможливість користуватися такими додатками людям без досвіду технічного обслуговування автомобільних транспортних засобів.

Додаток, розроблений у цьому дипломному проекті, дозволяє зручно і швидко проводити діагностування автомобілів всім категоріям користувачів. Для професіоналів є всі необхідні інструменти для повної діагностики автомобіля, а для користувачів, які не хочуть розбиратися у технічній стороні ТЗ, є можливість надати віддалене керування діагностикою професіоналам. Таким чином, користувач, автомобіль якого зламався посеред дороги, може з'єднатися з СТО і отримати всю необхідну інформацію щодо поломки автомобіля не викликаючи майстра.

Отже, з боку клієнта не потрібно чекати механіка для діагностування причини поломки, що є досить довгим і затратним процесом. А, з боку СТО,

механік може визначити поломку не виїжджаючи до клієнта. Це дозволить спростити і оптимізувати витрати ресурсів для процесу діагностування і усунення поломки автомобіля.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДІАГНОСТИКИ АВТОМОБІЛІВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1.Комп’ютерна діагностика автомобіля

Комп’ютерна діагностика автомобіля (OBD, on-board diagnostics) - це діагностика різноманітних систем автомобіля, яка проводиться блоком керування автомобіля. Результати діагностики відображаються для користувача автомобіля, наприклад на панелі приладів, а також використовуються автомеханіками і діагностами. Системи OBD впроваджуються з 1980-х років, OBDII - з 1996 року. Сучасні варіанти використовують стандартизовані цифрові порти для надання поточних даних і видачі ряду стандартних кодів проблем (DTC, diagnostic trouble code).

Стандарти інтерфейсів:

1. ALDL (Assembly Line Diagnostic Link) - діагностична система автомобілів, розроблена компанією General Motors, яка є попередником стандарту OBD-I. Ця система не являла собою чіткий стандарт і через це була введена як специфікація забезпечення зв’язку з транспортним засобом. Існує три різні роз’єми ALDL: 5-контактний роз'єм, 10-контактний і 12-контакний, - останній має широке поширення на автомобілях GM. Більш ранні версії використовували швидкість передачі 160 біт / с, в той час як пізніші - 8192 біт / с і використовували двосторонній зв'язок з Power-train Control Module (PCM).
2. OBD-I (On-Board Diagnostic Link) - бортова діагностика, що спонукала автовиробників розробляти надійні системи контролю за викидами (Emission control system).
3. OBD 1.5 - часткова реалізація OBD-II, яку General Motors використовував на деяких автомобілях в 1994 і 1995 роках (General Motors не використовували термін OBD 1.5 в документації на ці автомобілі, вони просто називалися OBD і OBD-II секції в інструкції по експлуатації).

4. OBD-II - бортова діагностика, стандарт якої був спроектований у середині 90-х років, що дозволяє отримувати повний контроль над двигуном автомобіля. Дозволяє здійснювати діагностування частин кузова і додаткових пристроїв, а також проводити моніторинг мережі управління транспортним засобом. В даному інтерфейсі виробники використовують такі протоколи для з'єднання з транспортним засобом:
    - ISO 9141-2;
    - ISO 14230 Keyword Protocol 2000;
    - SAE J1850 VPW;
    - SAE J1850 PWM;
    - ISO 15765-4 CAN (Controller Area Network).
5. EOBD - європейська бортова діагностична система, заснована на специфікації OBD-II. Ця система була введена під час розробки вимог моніторингу та скорочення викидів від автомобілів EURO 3, відповідно до «Directive 98/69 / EC of the European Parliament» від 13.10.1998 р.
6. EOBD2 - Термін EOBD2 є маркетинговим терміном, який використовувався деякими виробниками транспортних засобів, щоб звернути увагу на наявність специфічної функції від виробника, яка фактично не є частиною OBD або EOBD стандарту. В даному випадку “E” розшифровується як “Розширений” (Enhanced).
7. JOBD (Japan On-Board Diagnostic) - версія OBD-II для автомобілів, які були продані у Японії.

1.2. OBD-II діагностичний роз'єм

Специфікація OBD-II передбачає стандартизований апаратний інтерфейс - колодку діагностичного роз'єму (DLC - Diagnostic Link Connector), що відповідає стандарту SAE J1962, з 16-ма контактами (2x8) в формі трапеції для підключення діагностичного обладнання до автомобіля. На відміну від роз'єму OBD-I, який іноді зустрічається під капотом автомобіля, роз'єм OBD-II

зобов'язаний бути в районі рульового колеса, або в межах досяжності водія. SAE J1962 визначає розташування виходів на роз'ємі:

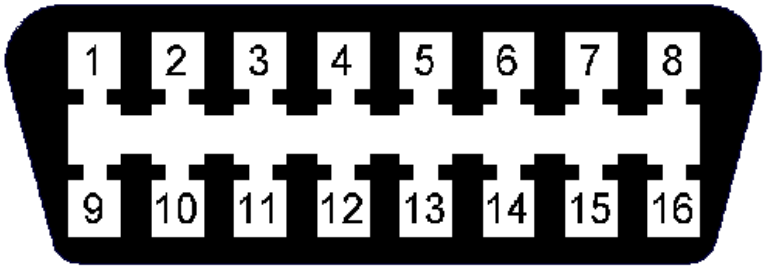

Рисунок 1.1 - Розташування виходів на роз'ємі OBDII

Призначення кожного з виходів:

1. OEM (протокол виробника).

2. Шина + (Bus positive Line). SAE-J1850 PWM, SAE-1850 VPW.

4. Заземлення кузова.

5. Сигнальне заземлення.

6. Лінія CAN-High високошвидкісної шини CAN Highspeed (ISO 15765-4, SAE-J2284).

7. K-Line (ISO 9141-2 и ISO 14230).

9. Лінія CAN-Low, низькошвидкісної шини CAN Lowspeed.

10. Шина — (Bus negative Line). SAE-J1850 PWM, SAE −1850 VPW.

14. Лінія CAN-Low високошвидкісної шини CAN Highspeed (ISO 15765-4, SAE-J2284).

15. L-Line (ISO 9141-2 и ISO 14230).

16. Живлення +12В від АКБ.

Призначення невизначених контактів залишається на розсуд виробника.

1.3. OBDII діагностичні дані

OBD-II забезпечує доступ до даних різних систем автомобіля, в тому числі з блоку управління двигуном (Engine control unit), і є цінним джерелом інформації при усуненні неполадок в автомобілі. Стандарт SAE J1979 визначає спосіб запиту різних діагностичних даних і списку стандартних параметрів

через Parameter's Identificators (Ідентифікатори параметрів), які можуть бути доступні в ECU. Виробники не зобов'язані виконувати всі перераховані в J1979 PID. Вони можуть включати в OEM власні PID (ІНП, ідентифікаційний номер параметру). Окремі виробники, часто розширюють OBD-II коди, додатковим набором власних OBD-II Non-Standard PIDs. Існує досить обмежений обсяг інформації, що є суспільним надбанням, для Non-Standard PIDs. Первинне джерело інформації для нестандартних PIDs для всіх виробників - інститут ETI (Equipment and Tool Institute), але інформація доступна тільки його членам.

1.4. Опис ІНП та структури повідомлень OBDII

OBDII повідомлення умовно розподіляється на дві частини: ідентифікатор та дані. Дані, в свою чергу, поділяються на такі параметри як довжина, режим (mode), ІПН (PID), і байти даних Ah, Bh, Ch, Dh (шістнадцяткові значення).

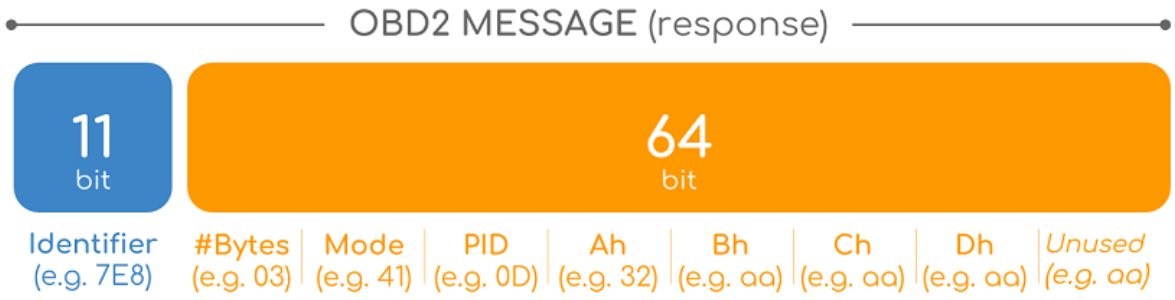

Рисунок 1.2 - Структура OBDII повідомлення

Приклад запиту/відповіді CAN повідомлення для ІПН “Швидкість автомобіля” зі значенням 50 км/год виглядає наступним чином:

Запит: 7DF 02 01 0D 55 55 55 55 55

Відповідь: 7E8 03 41 0D 32 aa aa aa aa

В даному прикладі 32 це шістнадцяткове значення 50

Розглянемо детальніше кожну частину повідомлення:

- Ідентифікатор – складається з 11 біт і використовується для розподілу “повідомлень запиту” (ID 7DF) і “повідомлень відповіді” (ID 7E8 - 7EF). Наприклад, типовим ІНП для відповіді з головного блоку двигуна є 7E8.

- Довжина - відображає довжину в кількості байтів решти даних. Для прикладу швидкості транспортного засобу, це 02 для запиту (оскільки тільки 01 і 0D слідують далі), в той час як для відповіді це 03, тому що далі йдуть 41, 0D і 32.
- Режим – для запитів це поле буде в межах 01-0A, а для відповідей 41-4A. Існує 10 режимів які описані в SAE J1979 OBD2 стандарті. Режим 1 показує поточні дані і, наприклад, використовуються для перегляду швидкості транспортного засобу в реальному часі, обертів двигуна і т.д. Інші режими використовуються наприклад для того, щоб показувати або очищати збережені діагностичні коди несправностей або показувати дані стоп-кадру.
- ІНП - для кожного режиму існує список стандартних ІНП-модулів OBD2, наприклад в режимі ІНП 01, 0D - швидкість транспортного засобу. Кожен PID має опис, а деякі мають формулу перетворення або задані мінімуми / максимуми.

  Для прикладу, формула швидкості це просто A, тобто, байт даних Ah (який поданий у шістнадцятковому форматі) перетворюється в десяткове число з одиницями км / год (тобто 32 стає 50 км / год). A, наприклад, для обертів двигуна (RPM, PID 0C), формула буде наступна:

  (256 * A + B) / 4.

  Ah, Bh, Ch, Dh - це байти даних у шістнадцятковому форматі, які необхідно перетворити в десятковий формат, перш ніж вони будуть використані в розрахунках ІНП-формули.

1.5. OBD-II коди помилок

Кожен з OBD-II кодів помилок складається з п’яти символів: букви і чотирьох цифр [11]:

1. P00xx - Контроль систем сумішоутворення і системи додаткового зниження токсичності вихлопної системи.

2. P01xx - Контроль систем сумішоутворення.
3. P02xx - Контроль систем сумішоутворення.
4. P03xx - Система запалення і системи контролю пропусків займання.
5. P04xx - Допоміжні системи контролю емісії.
6. P05xx - Контроль швидкості автомобіля, системи холостого ходу та інших систем.
7. P06xx - Блоки управління ECM / PCM / TCM та інші системи
8. P07xx - Трансмісія.
9. P08xx - Трансмісія.
10. P09xx - Трансмісія.
11. P10xx - Коди, що встановлюються виробником. Залежать від марки автомобіля.
12. P20xx - Коди, що встановлюються виробником. Залежать від марки автомобіля.
13. B00xx - Кузов (подушка безпеки, центральний замок, електростекло-підйомник).
14. C00xx - Шасі (ABS, ESP, TCS, система курсової стійкості).
15. U10xx - Міжблокова шина обміну даних (CAN-шина) (CAN-II).
16. U25xx - Міжблокова шина обміну даних (CAN-шина) (CAN-II).

Символи xx посилаються на окремі коди помилок всередині кожної підсистеми.

1.6. OBD-II режими діагностики систем

Основні можливості протоколу OBD-II, відповідно до SAE J1979 [12]:

1. Режим $01: Діагностичні дані силового приводу (Current Powertrain Diagnostic Data, Live Data, Data Stream).
2. Режим $02: Доступ до збережених («заморожених») даних (Freeze Frame, FF).

3. Режим $03: Зчитування кодів параметрів які впливають на токсичність викидів (Emission Related Powertrain).
4. Режим $04: Видалення діагностичної інформації (Clear / Reset Emission Related Diagnostic Information) і кодів несправності.
5. Режим $05: Результати перевірки кисневих датчиків (Oxygen Sensor Monitoring Test Results).
6. Режим $06: Результати перевірки («вторинних») компонентів які не перевіряються не безперервно (On-Board Monitoring Test Results for Non-Continuously Monitoring Systems).
7. Режим $07: Результати перевірки систем які безперервно перевіряються (Monitoring Test Results for Continuously Monitored Systems).
8. Режим $08: Запит виконання управління виконавчими пристроями (Request Control of On-Board System Test or Component).
9. Режим $09: Зчитування ідентифікаційної інформації автомобіля (Request Vehicle Information).
10. Режим $0A: Помилки, які були видалені. Permanent DTC's (Cleared DTC's) - Diagnostic Trouble Codes.

Виробникам транспортних засобів не потрібна підтримка всіх режимів. Кожен виробник може визначати додаткові режими вище $ 09 (наприклад, режим 22, як це визначено SAE J2190 для Ford / GM, режим 21 для Toyota).

1.7. Поняття якості мобільного додатку

Перш ніж перейти до аналізу існуючих рішень потрібно сформулювати критерії за якими будуть оцінюватися додатки.

Мобільний додаток можна оцінювати по різним критеріям, основними з яких є: графічний інтерфейс, користувацький досвід, функціональні можливості та швидкодія.

Графічний інтерфейс (UI, user interface) – загальний вигляд користувацького інтерфейсу додатку. Визначає наскільки добре підібрана

палітра кольорів, шрифти та компонування елементів інтерфейсу. Також, важливим аспектом є дотримання тенденцій у виробництві графічних інтерфейсів.

Користувацький досвід (UX, user experience) – загальне враження від додатку. Визначає наскільки інтуїтивно зрозумілий інтерфейс та наскільки просто користувачеві було досягнути цілі.

Функціональні можливості – визначають наскільки корисним є додаток та які особливі функції відрізняють додаток від інших схожих додатків.

Швидкодія – наскільки швидко працює додаток.

1.8. Аналіз існуючих додатків для діагностики автомобіля

Для аналізу було обрано три найпопулярніші додатки в Android Play Store: Torque, CarScanner, OBD Auto Doctor.

- Torque.

1. Графічний інтерфейс. Додаток має застарілий дизайн, який не відповідає теперішнім дизайнерський рішенням та вимогам Material Design від Google.
2. Користувацький досвід. Важкий у освоєнні інтерфейс. Іконки з параметрами можна кастомізувати, але дуже не зрозуміло як і де це робити. Новому користувачеві важко розібратися з елементами керування.
3. Функціональні можливості. Додаток вміє робити базові речі: зчитувати коди помилок та робити логування параметрів системи.
4. Швидкодія. Додаток має задовільну швидкодію. Під час користування додатком ніяких нарікань не виявлено.

Скріншоти додатку:

Рисунок 1.3 - Скріншот Torque

Рисунок 1.4 - Скріншот Torque

- Car Scanner

1. Графічний інтерфейс. Додаток має приємний, лаконічний дизайн, який відповідає вимогам Material Design. До головних функцій є легкий доступ на головному екрані у вигляді кнопок з іконками і надписами.

2. Користувацький досвід. Всі основні функції винесені на головний екран, що є зручним у користуванні і легким у освоєнні для нових користувачів. Проте, додаток містить рекламні банери.
3. Функціональні можливості. Додаток вміє зчитувати коди помилок та робити логування параметрів системи та показвати їх у зручному для аналізу вигляді (графіки, спідометри і таблиці). Також, додаток може читати коди помилок.
4. Швидкодія. Додаток має задовільну швидкодію. Під час користування додатком ніяких нарікань не виявлено.

   Скріншоти додатку:

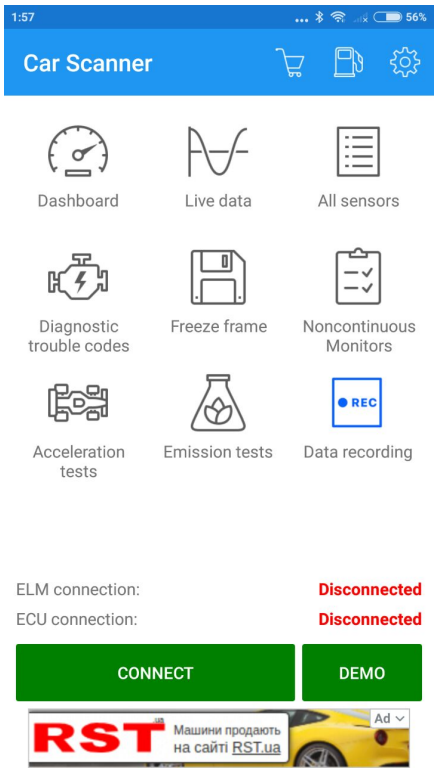

Рисунок 1.5 - Скріншот Car Scanner

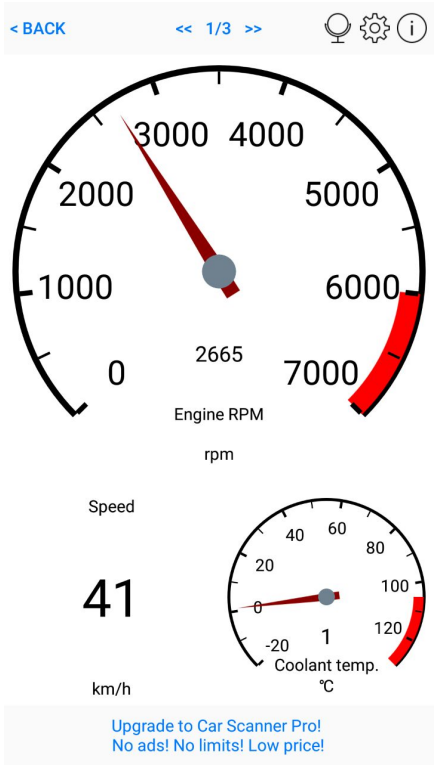

Рисунок 1.6 - Скріншот Car Scanner

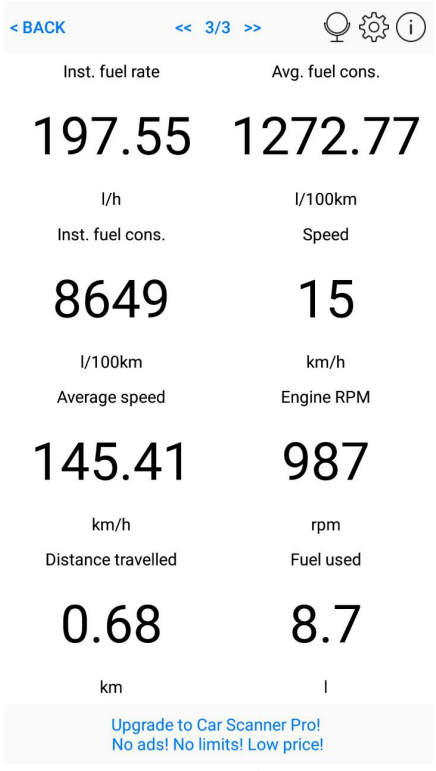

Рисунок 1.7 - Скріншот Car Scanner

- OBD Auto Doctor.

1. Графічний інтерфейс. Додаток відповідає вимогам Material Design від Google і має приємний користувацький інтерфейс.
2. Користувацький досвід. Додаток має головне меню внизу екрану, що є дуже зручним рішенням. Але, спершу не зрозуміло як користуватися додатком, адже багато елементів не мають підпису про те, яку саме дію вони виконують.
3. Функціональні можливості. Додаток вміє зчитувати основні параметри а також читати та скидати помилки.
4. Швидкодія. Додаток має задовільну швидкодію. Під час користування додатком ніяких нарікань не виявлено.

Скріншоти додатку:

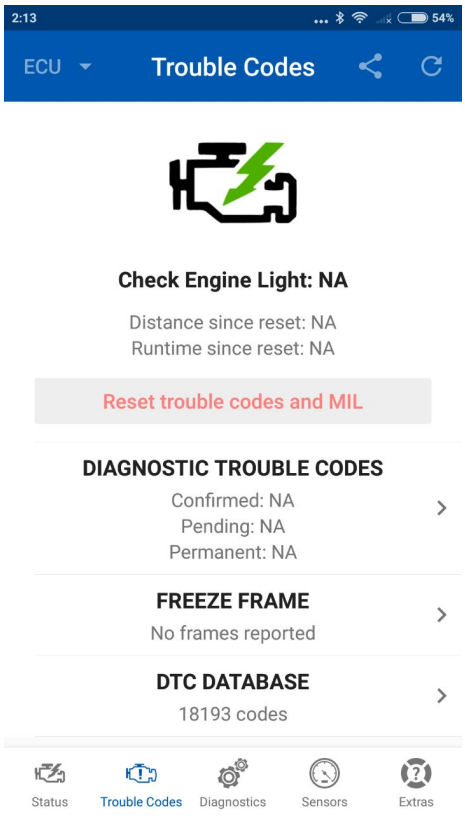

Рисунок 1.8 - OBD Auto Doctor

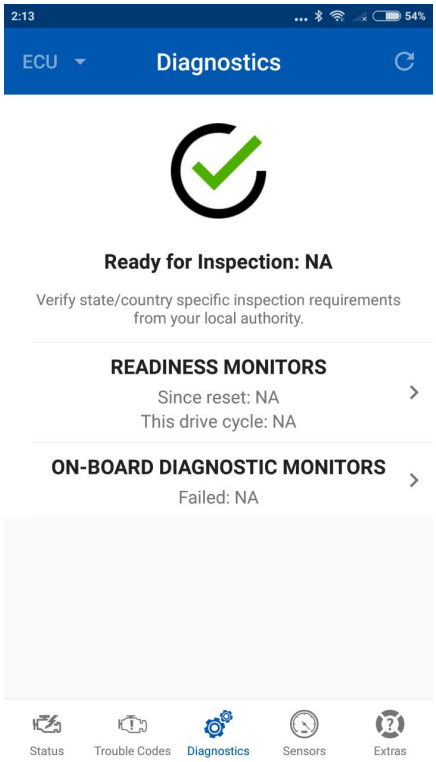

Рисунок 1.9 - OBD Auto Doctor

Висновки:

Найкращим додатком по сукупності всіх параметрів виявився “Car Scanner”, адже він надає великий функціонал у поєднанні з красивим і логічним дизайном. За допомогою “Car Scanner” можна зчитувати коди помилок, проводити моніторинг основних параметрів у реальному часі а також робити їх логування. Проте, основним недоліком всіх цих додатків є те, що користувач без досвіду в автомобільній механіці не матиме достатньо знань для діагностики автомобіля без допомоги спеціаліста.

1.9. Обґрунтування теми дипломного проекту

Метою даного дипломного проекту є створення сервісу, яким буде зручно користуватися як людям без досвіду в технічному обслуговуванні автомобіля, так і професіональним автомеханікам. Основною особливістю розробленого додатку є те, що автомеханіки зможуть дізнатися дані, необхідні для діагностики транспортного засобу, віддалено - під’єднавшись до пристрою

який в свою чергу під’єднаний до OBDII приймача. Упровадження цієї системи дозволить станціям технічного обслуговування (СТО) автомобілів дистанційно проводити моніторинг автомобілів. Це зменшить витрати коштів та часу для аналізу поломки автомобіля як зі сторони користувача так і зі сторони СТО.

2. ОСОБЛИВОСТІ РОЗРОБКИ ANDROID ДОДАТКУ

2.1. Про мобільну платформу Android

Android - це система для мобільних пристроїв, що була розроблена компанією Google. Вона побудована на модифікованій версії ядра операційної системи Linux, та інших програм з відкритим вихідним кодом, і призначена, в основному для мобільних пристроїв, які мають сенсорний екран, таких як смартфони та планшети. Також, Google розробила Android TV для телевізорів, Android Auto для автомобілів і Wear OS для наручних годинників, кожна зі спеціалізованим інтерфейсом користувача. Варіації Android також використовуються на ігрових консолях, цифрових камерах, ПК та іншій електроніці. [2]

Комп'ютерний інженер Енді Рубін вибрав android.com як свій персональний сайт, а його колеги використовували Android як прізвисько на роботі. Ім'я Андрій і іменник Android поділяють грецький корінь andros, що означає людина. Це зрештою стало назвою компанії, яку він заснував, і назвою операційної системи, яку вони розробили. [5]

Компанія Android Inc. була заснована в Пало-Альто, Каліфорнія, у жовтні 2003 року Енді Рубіном, Річем Майнером, Ніком Сірсом та Крісом Уайтом. Рубін назвав проект Android "величезним потенціалом у розробці розумніших мобільних пристроїв, які краще знають місцезнаходження та переваги свого власника". Ранніми намірами компанії було розробити потужну операційну систему для цифрових камер, і це стало основою для його інвесторів у квітні 2004 року. Потім компанія вирішила, що ринок камер не є достатньо великим для своїх цілей, і вже через п'ять місяців віддала свої зусилля і підняла Android як операційну систему, що конкуруватиме з Symbian і Microsoft Windows Mobile (на той час iOS від Apple ще не існувало). [8]

Первинно розроблений компанією Android Inc., яку Google придбав у 2005 році, Android був випущений у 2007 році, а перший комерційний Android-

пристрій був представлений у вересні 2008 року. Операційна система з того часу мала багато оновлень, останнє стабільне з яких випущене у серпні 2018 і має назву Andorid 9 “Pie”, що в перекладі з англійської означає “пиріг”. Взагалі, всі версії Android називалися іменем солодощів. Попередні версії мали такі назви: Apple Pie (яблучний пиріг), Banana Bread (бананoвий хліб), Cupcake (кекс), Donut (пончик), Eclair (еклер), Froyo (заморожений йогурт), Gingerbread (пряник), Honeycomb (медовий стільник), Ice Cream Sandwich (сендвіч з морозивом), Jelly Bean (желе), KitKat, Lollipop (льодяник), Marshmallow (зефір), Nougat (нуга), Oreo, Pie (пиріг). [9]

Android також пов'язаний з набором запатентованого програмного забезпечення, розробленого компанією Google, що називається Google Mobile Services (GMS), що дуже часто поставляється попередньо встановленим на пристроях. GMS зазвичай включають веб-переглядач Google Chrome і Google Search і завжди містять основні програми для таких послуг, як Gmail, а також магазин додатків і платформа цифрового розповсюдження Google Play і пов'язана з ними платформа розробки.

Android є найпопулярнішою ОС у світі на смартфонах з 2011 року і на планшетах з 2013 року. Станом на травень 2017 року вона має понад два мільярди активних користувачів щомісяця, а саном на грудень 2018 магазин Google Play містить понад 2.6 мілярди додатків.

2.2. Архітектура Android платформи

Android є системою з відкритим сирцевим кодом, що побудована на основі програмного стеку системи Linux для підтримки широкого спектру пристроїв різних форм-факторів.

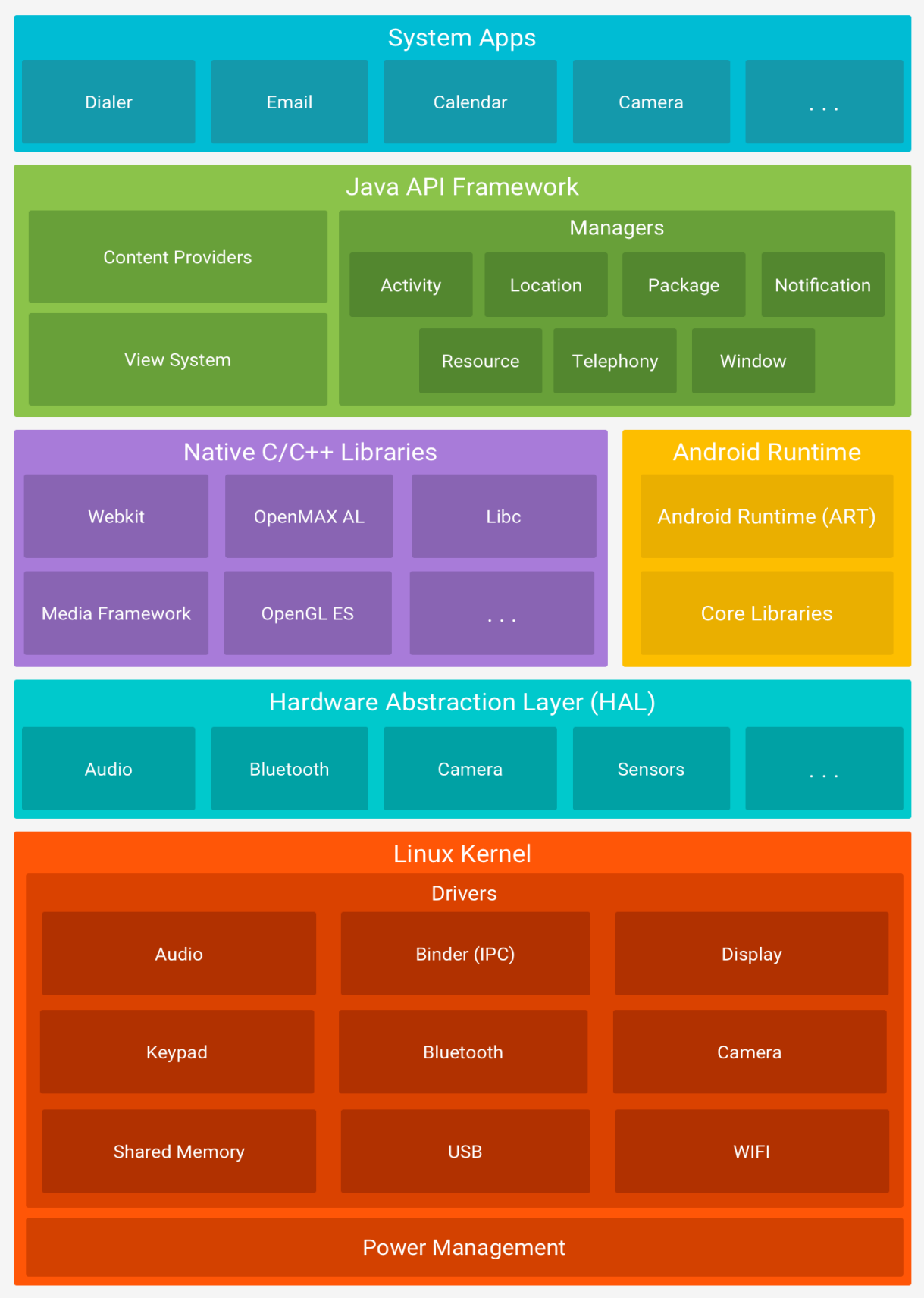

Рисунок 2.1 - Структура Android системи

2.2.1. Linux ядро

Основою платформи Android є ядро Linux. Наприклад, Android Runtime (ART) спирається на ядро Linux для основних функціональних можливостей, таких як потоки та управління низьким рівнем пам'яті.

Використання ядра Linux дає змогу Android використовувати переваги ключових функцій безпеки і дозволяє виробникам пристроїв розробляти драйвери апаратного забезпечення для вже добре відомого ядра.

2.2.2. Шаблон абстракції обладнання (HAL)

Шаблон абстракції обладнання (Hardware Abstraction Layer, HAL) надає стандартні інтерфейси, які розширюють апаратні можливості пристрою до більш високого рівня Java API. HAL складається з декількох модулів, кожен з яких реалізує інтерфейс для конкретного типу апаратного компонента, такого як камера або модуль Bluetooth. Коли API платформа здійснює виклик для доступу до апаратного забезпечення пристрою, система Android завантажує модуль бібліотеки для цього компонента обладнання.

2.2.3. Android Runtime

Для пристроїв, що працюють під Android версією 5.0 (рівень API 21 або вище), кожна програма працює у своєму власному процесі та з власним екземпляром Android Runtime (ART). ART написаний для запуску декількох віртуальних машин на пристроях з низькою пам'яттю, виконуючи файли DEX, формат байт-коду, розроблений спеціально для Android, оптимізований для мінімального споживання пам'яті. Збірка інструментів, наприклад, Jack, компілює джерела Java в байт-код DEX, який може працювати на платформі Android.

Деякі з ключових можливостей ART включають наступне:

- завчасна (Ahead-of-time, AOT) компіляція та компіляція під час роботи програми (just-in-time, JIT);

- оптимізована збірка сміття (garbage collection, GC);
- на Android 9 (рівень API 28) і вище, перетворення файлів у форматі Dalvik Executable (DEX) пакета програм на більш компактний машинний код;
- підтримка кращого налагодження (анг. debug), включаючи спеціальний профайлер вибірки, докладні діагностичні виключення та звіти про аварійне завершення роботи, а також можливість встановити точки спостереження для моніторингу конкретних полів.

До версії Android 5.0 (рівень API 21), Android був оснований на віртуальній машині Dalvik. Якщо програма працює добре на ART, то вона буде працювати і на Dalvik, але зворотне не стверджується.

Android також включає в себе набір основних бібліотек, які забезпечують більшу частину функціональності мови програмування Java, включаючи деякі мовні особливості Java 8, які використовує Java API.

2.2.4. Нативні (Native) C/C++ бібліотеки

Багато основних системних компонентів і служб Android, таких як ART і HAL, побудовані з коду, який вимагає нативних бібліотек, написаних на мовах C і C++. Платформа Android надає API Java-платформи для розкриття функціональних можливостей деяких з цих нативних бібліотек для програм. Наприклад, можна отримати доступ до OpenGL ES за допомогою API OpenGL для платформи Android, щоб додати підтримку для малювання та маніпулювання 2D і 3D графікою у додатку.

Якщо розробляється програма, яка потребує коду C або C++, можна скористатися NDK для Android, щоб отримати доступ до деяких з цих бібліотек платформи безпосередньо.

2.2.5. Java API Framework

Весь набір функцій ОС Android доступний через API, написане на мові Java. Ці API формують “будівельні” блоки, необхідні для створення додатків Android, спрощуючи повторне використання основних компонентів і служб модульної системи, які включають наступне:

- велику та розширювану систему перегляду (View System), яку можна використовувати для створення користувацького інтерфейсу програми, включаючи списки, сітки, текстові поля, кнопки та навіть вбудований веб-переглядач;
- менеджер ресурсів (Resource Manager), що надає доступ до ресурсів системи, таких як локалізовані рядки, графічні файли та файли компонування;
- менеджер сповіщень (Notification Manager), який дозволяє програмам відображати спеціальні сповіщення в рядку стану;
- менеджер активності (Activity Manager), який керує життєвим циклом програм і надає загальний стек навігації;
- постачальники вмісту (Content Providers), які дозволяють програмам отримувати доступ до даних з інших програм, наприклад, програми "Контакти", або обмінюватися власними даними.

Розробники мають повний доступ до тих самих API інтерфейсів, які використовують системні програми для Android.

2.2.6. Системні додатки

Android поставляється з набором основних програм для електронної пошти, SMS-повідомлень, календарів, веб-перегляду, контактів тощо. Програми, включені до платформи, не мають спеціального статусу серед програм, які користувач вибирає для встановлення. Таким чином, додаток стороннього виробника може стати користувальницьким веб-браузером за

промовчанням, SMS-переглядачем або навіть клавіатурою за замовчуванням (але, все ж, є деякі винятки, такі як додаток налаштувань системи).

Системні програми функціонують як програми для користувачів, а також надають ключові можливості, до яких розробники можуть отримати доступ з власних програм. Наприклад, якщо програма хоче надіслати SMS-повідомлення, то не потрібно самостійно створювати цю функцію, а, натомість, можна скористатися вже встановленим SMS додатком для відправки за вказаною адресою.

2.3. Користувацький інтерфейс Android

2.3.1. Material Design

Material Design це принцип побудови графічного інтерфейсу програмного забезпечення який запровадила компанія Google. Ідея дизайну полягає у тому, що графічний інтерфейс наслідує правила поведінки паперових карток у реальному житті. Material Design був створений задля того, щоб перетворити взаємодію користувача з інтерфейсом у зручну та звичну для людини форму. Основні особливості Material Design:

1. Система тіней. Для того, щоб створити візуальну ілюзію простору між додатками та екраном пристрою була введена система тіней. Таким чином, всі екрани, вікна, іконки та інші об’єкти графічного дизайну обрамляються тінями.
2. Анімація. Збільшена реальність анімації за рахунок руху об’єктів з різною швидкістю та з різними прискореннями. Також, додано пружне стрибання об’єктів.
3. Іконки. Для забезпечення кращої взаємодії з користувачем, іконки стали інтерактивними. Це дозволяє виконувати різні жести над іконками, які призведуть до різної поведінки. Наприклад, при довгому натисканні на іконку можна видалити її, або подивитися параметри додатку за який відповідає ця іконка.

4. Об’єктна залежність. Суть об’єктної залежності полягає у тому, що об’єкти можуть впливати на поведінку інших об’єктів.

Компанія Google активно просуває Material Design. Усі додатки компанії побудовані за цією методологією. Інші виробники ПЗ також користуються Material Design для проектування графічних інтерфейсів.

Material Design активно використовується у мобільній платформі Android починаючи з версії 5.0 Lollipop.

2.3.2. Макети

Макети (анг. Layouts) визначають структуру для користувацького інтерфейсу додатків. Всі елементи у макетах побудовані за допомогою ієрархії View та ViewGroup об’єктів. View об’єкт визначає собою частину графічного інтерфейсу, яка візуалізується під час створення екрану, з якою користувач можу взаємодіяти. ViewGroup це невидимий для користувача контейнер, який містить у собі інші View або ViewGroup об’єкти.

2.3.3. Сповіщення

Сповіщення мобільній платформі Android це повідомлення, яке показується за межами графічного інтерфейсу додатку для того, щоб нагадати користувачеві про якусь дію, показати сповіщення комунікації з іншими людьми або іншу своєчасну інформацію від додатку. Користувачі також можуть взаємодіяти зі сповіщеннями щоб відкрити додаток, або здійснити якусь дію безпосередньо зі сповіщення.

2.4. Bluetooth

2.4.1. Загальні відомості про Bluetooth

Bluetooth - технологія передачі даних по бездротовому зв'язку, що була спроектована у 1998 році такими компаніями: Ericsson, IBM, Intel, Nokia, Toshiba.

Bluetooth розробляється компанією Bluetooth Special Interest Group (SIG), яка налічує понад 30 000 компаній які працюють у сфері телекомунікацій, обчислень, мереж і побутової електроніки. IEEE стандартизував Bluetooth як IEEE 802.15.1, але більше не займається підтримкою стандарту. Bluetooth SIG контролює розробку специфікації, керує програмою кваліфікації та захищає торгові марки. Виробник повинен дотримуватися стандартів Bluetooth SIG, щоб продавати пристрої з Bluetooth.

Розвиток радіотехнологій «короткого зв’язку», згодом названих Bluetooth, був започаткований у 1989 році Нілсом Рідбеком, технічним директором Ericsson Mobile в Лунді, Швеція, і Йоханом Уллманом. Мета полягала в розробці бездротових гарнітур. Технологія придумана голландським інженером-електриком Яапом Хаартсеном, який працював у телекомунікаційній компанії Ericsson у 1994 році. Перший користувацький пристрій з Bluetooth був випущений в 1999 році. Це була “Hands-free” мобільна гарнітура, яка отримала нагороду "Best of show Technology Award". Перший мобільний телефон з Bluetooth був Ericsson T36, який предаставили у 2001 році.

Bluetooth працює на частотах від 2402 до 2480 МГц. Дана технологія розділяє дані для передачі на пакети і передає кожен пакет на один з 79 призначених каналів Bluetooth. Кожен канал має пропускну здатність 1 МГц.

Bluetooth - це пакетний протокол з архітектурою Master / Slave. Один Master пристрій може передавати дані на сім Slave пристроїв у пікомережі. Обмін пакетами ґрунтується на базовому тактовому сигналі, визначеному майстром, який вказує на інтервали у 312,5 мкс. Два такти складають слот у 625 мкс, а два слоти складають слот-пару 1250 мкс. У простому випадку однослотових пакетів, Master передає дані в парних слотах і отримує в непарних слотах. Slave, навпаки, отримує в парних слотах і передає в непарних слотах.

Пакети можуть бути довжиною 1, 3 або 5 слотів, але у всіх випадках передача з Master пристрою розпочинається на парних слотах, а Slave пристрою на непарних слотах.

2.4.2. Bluetooth у платформі Android

Платформа Android включає підтримку мережевого стека Bluetooth, що дозволяє пристрою бездротово обмінюватися даними з іншими пристроями Bluetooth. Фреймворк надає доступ до функціональності Bluetooth за допомогою інтерфейсу Android Bluetooth. Це API дозволяє додаткам бездротово підключатися до інших пристроїв Bluetooth, використовувати режими "точка-точка" та широкомовного розсилання.

За допомогою API-інтерфейсу Bluetooth для Android можна виконувати такі дії:

- пошук інших пристроїв Bluetooth;
- запитувати Bluetooth адаптер пристрою про вже “спарені” пристрої;
- встановлювати канали RFCOMM;
- підключитися до інших пристроїв;
- передавати даних на інші пристрої та приймати дані з них;
- керувати декількома з'єднаннями.

Для того, щоб Bluetooth-пристрої передавали дані між собою, вони повинні спочатку сформувати канал зв'язку, використовуючи процес спарювання. Один пристрій, який можна виявити (Slave), доступний для вхідних запитів на з'єднання. Інший пристрій (Master) знаходить відкритий пристрій за допомогою процесу пошуку служб. Після того, як Slave пристрій приймає запит на створення пари, ці два пристрої завершують процес з'єднання, де вони обмінюються ключами безпеки. Пристрої кешують ці ключі для подальшого використання. Після завершення процесів сполучення і парування два пристрої можуть обмінюватися інформацією. Коли сеанс завершено, пристрій, який ініціював запит сполучення (Master), вивільняє канал, який

пов'язував його з Slave пристроєм. Надалі, два пристрої залишаються з’єднаними, тому вони можуть автоматично підключатися під час майбутньої сесії, якщо вони знаходяться в діапазоні видимості один одного, і жоден пристрій не видалив зв'язок.

Для того, щоб отримати список пристроїв, які знаходяться у межах видимості поточного пристрою, потрібно “підписатися” на повідомлення “ACTION_FOUND”. Це повідомлення надсилає Android, коли знаходить новий пристрій Bluetooth. Разом з цим повідомленням, система надає екзмепляр класу BluetoothDevice, який містить у собі основну інформацію про знайдений Bluetooth пристрій. Для того, щоб з’єднатися з пристроєм і почати передавати інформацію, потрібно виділити сокет, через який пристрої будуть обмінюватися даними.

3. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ANDROID ДОДАТКУ ДЛЯ ДІАГНОСТИКИ АВТОМОБІЛЯ

3.1. Користувацький інтерфейс

Коли користувач вперше вмикає додаток, перед тим як показати головний екран, користувачеві показується сповіщення про запит надання прав доступу до внутрішнього сховища мобільного пристрою (Рисунок 3.1). Якщо користувач не надасть дозвіл, то в майбутньому, коли він захоче зберегти дані у файлову систему пристрою, це сповіщення буде показане ще раз, адже без цих прав доступу додаток не зможе нічого зберігати в пам’ять пристрою. Також, запитується дозвіл на визначення точного місцезнаходження користувача (Рисунок 3.2). Дозвіл точного місцезнаходження користувача потрібен для коректної роботи Bluetooth.

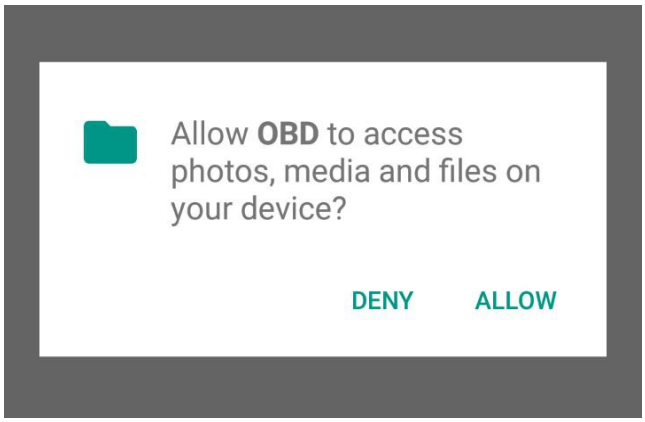

Рисунок 3.1 - Запит на надання прав доступу до сховища пристрою

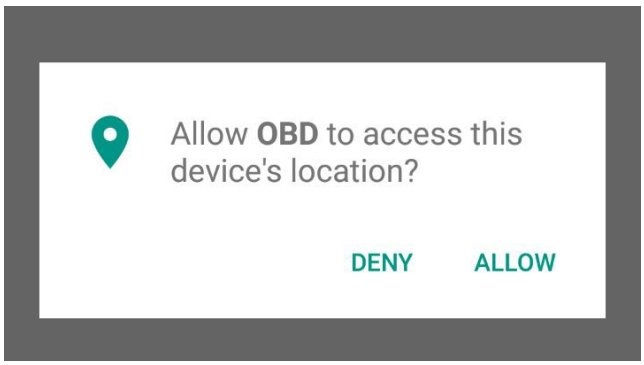

Рисунок 3.2 - Запит прав доступу на місцезнаходження пристрою

Коли користувач дасть відповідь на сповіщення, буде показано основне меню додатку (Рисунок 3.3). В основному меню знаходиться нижнє меню, які відповідає за перемикання основного контенту на головному екрані. Для цього реалізовано контейнер, вміст якого може динамічно змінюватися під час роботи додатку. Таким чином, користувач може перемикати вміст головного екрану, не втрачаючи історію роботи з цим екраном. Тобто, якщо користувач щось робив у вікні 1, переключився на вікно 2, а потім повернувся назад на 1 вікно, то стан вікна 1 буде той, який був на момент, коли користувач покинув це вікно.

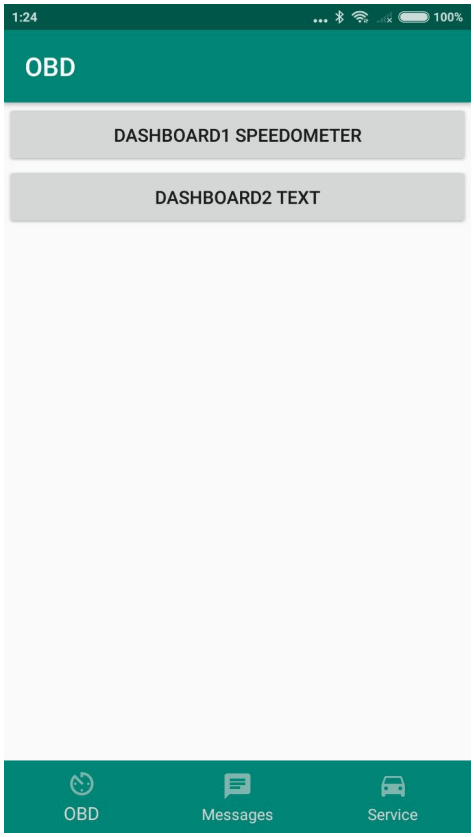

Рисунок 3.3 - Головне меню додатку

3.1.1. Пункт меню “OBD”

При запуску програми активується перше вікно (перший пункт у нижньому меню), яке надає можливості вибору способу моніторингу автомобіля, яке називається “OBD”

На вибір користувача є два режими:

1. Режим показу даних у вигляді спідометра (рисунок 3.4). Цей режим надає можливість слідкувати за показниками швидкості та обертами двигуна у вигляді спідометра та тахометра. Цей режим буде звичним і зручним для водіїв, адже він повторює фізичні пристрої на керуючій панелі автомобіля. Також, даний екран може грати роль стандартного тахометра, у випадку якщо він не робочий, або показує неправильну інформацію.
2. Режим показу даних у вигляді таблиці (рисунок 3.5). Даний режим є зручним для збору та аналізу даних декількох показників, адже може одночасно показувати до восьми значень у режимі реального часу. Цей режим буде особливо корисний автомобільним механікам, які хочуть одночасно відслідковувати поведінку різних показників автомобіля.

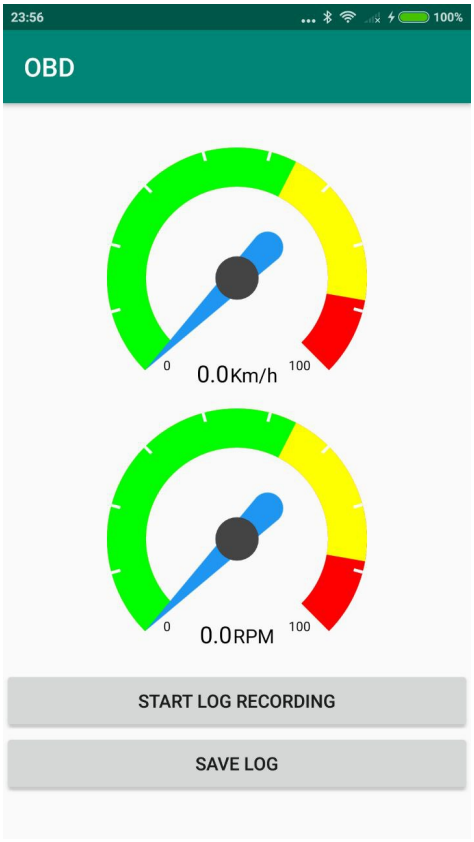

Рисунок 3.4 – Екран показу швидкості та обертів двигуна у вигляді спідометру

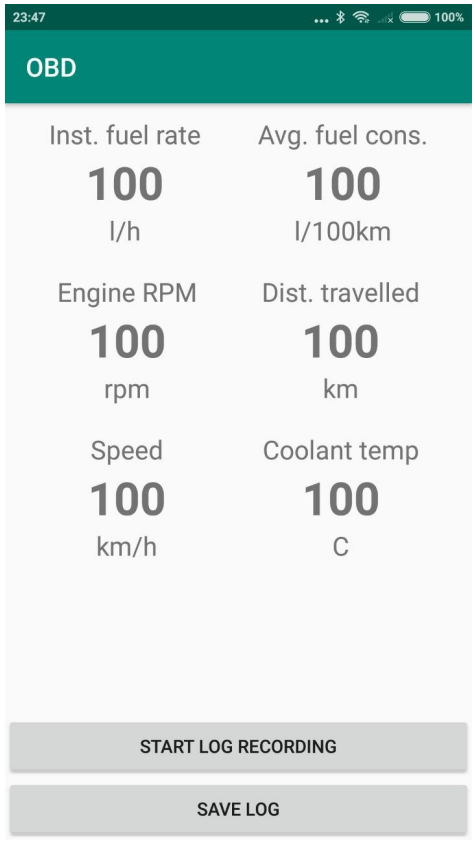

Рисунок 3.5 - Екран показу даних у вигляді таблиці

Після того, як користувач натисне на одну з вище перелічених кнопок, відкриється екран пошуку Bluetooth пристроїв (рисунок 3.6), що є в зоні видимості.

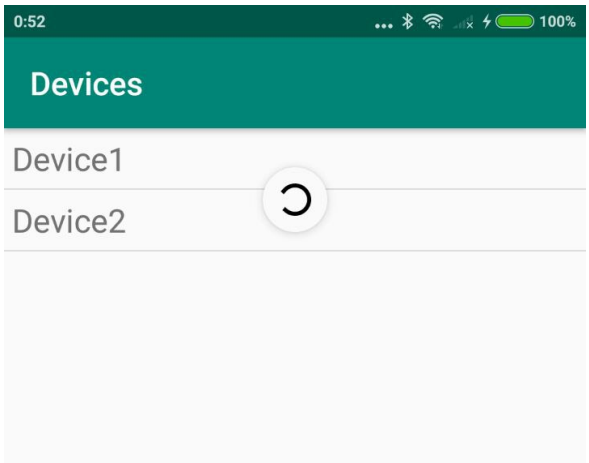

Рисунок 3.6 – Екран пошуку Bluetooth пристроїв

У кожному з вище перелічених режимів також є кнопки, які надають можливість логування даних. Дані зберігаються у форматі CSV у зовнішнє

сховище додатку, тому їх можна вільно переглядати чи редагувати за межами додатку. Наприклад, користувач може може підключити телефон до персонального комп’ютера у режимі накопичувача, і користуватися збереженими файлами, бо вони вільні для зчитування та редагування.

Для того, щоб розпочати логування даних потрібно натиснути на кнопку “START LOG RECORDING”. Після того як користувач розпочне логування даних, всі дані будуть зберігатися в режимі реального часу на зовнішню пам'ять пристрою, а кнопка “START LOG RECORDING” змінить свою назву на “STOP LOG RECORDING” (рисунок 3.7, 3.8). Для того, щоб зупинити записування логів, користувач повинен натиснути на кнопку “STOP LOG RECORDING”, яка, після натискання, змінить свою назву назад на “START LOG RECORDING”.

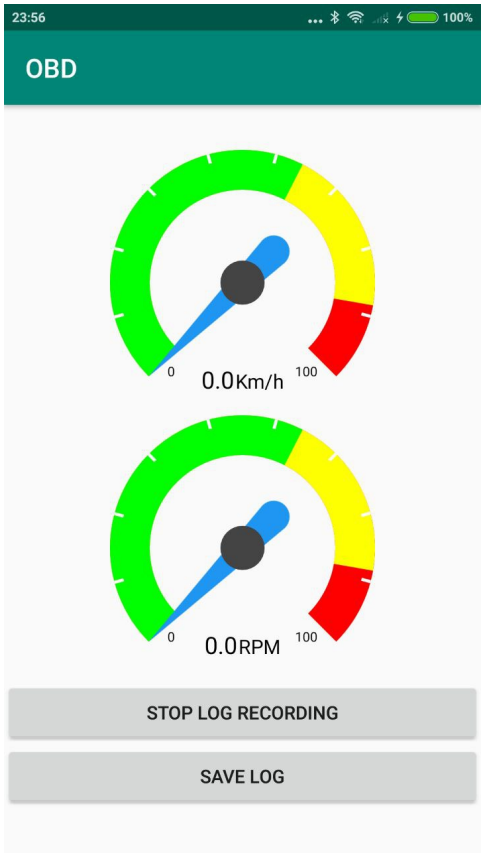

Рисунок 3.7 – Екран показу швидкості та обертів двигуна у вигляді спідометру з кнопкою закінчення логування

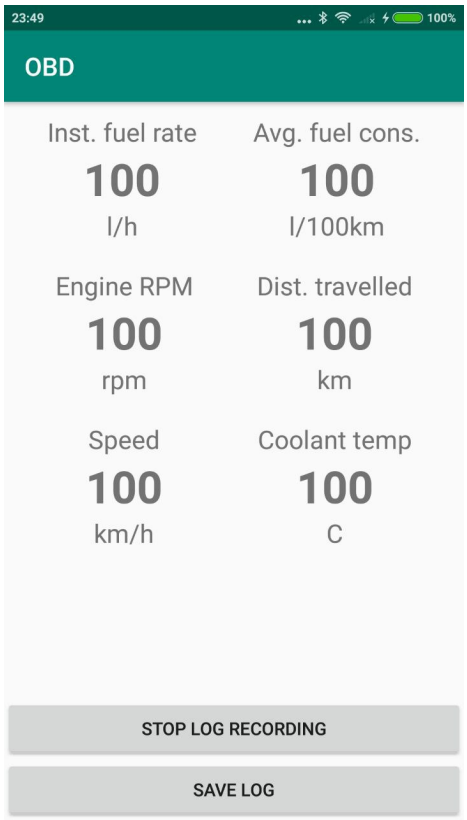

Рисунок 3.8 - Екран показу даних у вигляді таблиці з кнопкою закінчення логування

Для того, щоб зберегти записані логи, на кожному з вище перелічених екранів є кнопка для збереження даних, яка має назву “SAVE LOG”. Натиснувши на цю кнопку, створиться CSV файл, назва якого відповідатиме даті та часу збереження логів.

3.1.2. Пункт меню “Messages”

Даний екран призначений для того, щоб користувачі могли обмінюватися даними між собою. При запуску додатку відбувається перевірка на те чи був користувач раніше аутентифікований. Якщо так, то додаток дістає зі сховища збережений унікальний ідентифікатор користувача та надсилає його на сервер для аутентифікації. Якщо ж користувач раніше не був аутентифікований, або була стерта вся інформація додатку (користувачі можуть очищати пам’ять, яку

використовують додатки), то показується вікно для вводу особистого унікального ідентифікатора користувача та його паролю (рисунок 3.9).

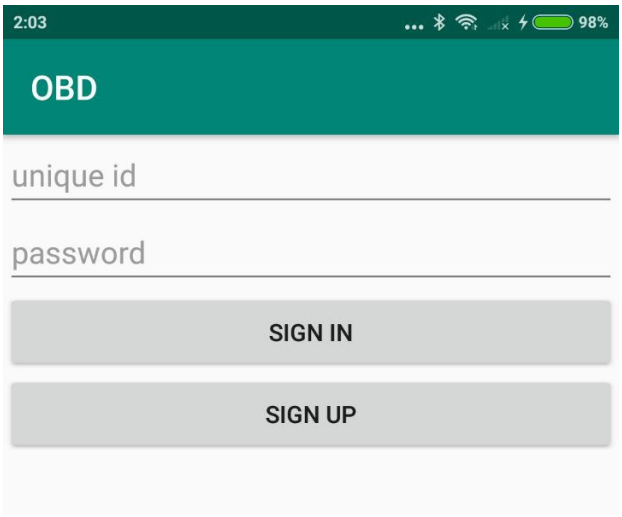

Рисунок 3.9 – Екран для входу в систему

Для того, щоб аутентифікуватися, користувач повинен увести свій унікальний ідентифікатор та пароль до свого облікового запису. Якщо одне з полів буде порожнім, то аутентифікація не почнеться, а користувачеві буде показано повідомлення про помилку (рисунок 3.10).

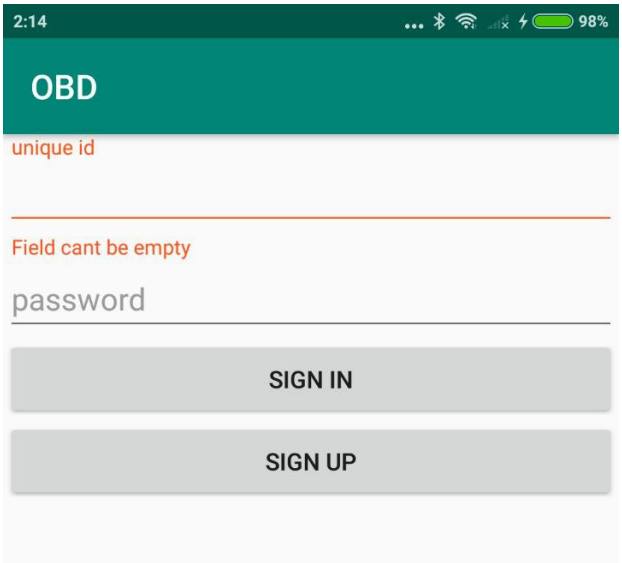

Рисунок 3.10 – Екран для входу в систему при спробі аутентифікації з порожніми полями

Якщо користувач ще не зареєстрований у системі, потрібно натиснути на кнопку “SIGN UP”, яка покаже вікно для реєстрації користувача (рисунок 3.11). Для того, щоб користувач зареєструвався, він повинен ввести наступні дані:

- логін;
- поштову скриньку;
- номер телефону;
- пароль.

Якщо якесь поле буде порожнім, користувач отримає повідомлення про помилку, як показано на рисунку 3.10. Поле паролю не буде показуватися як звичайний текст, а буде замінено на крапки, щоб ніхто не підглянув пароль користувача.

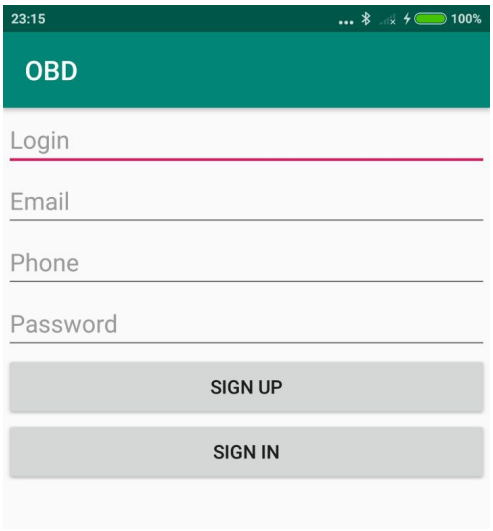

Рисунок 3.11 – Екран для реєстрації у системі

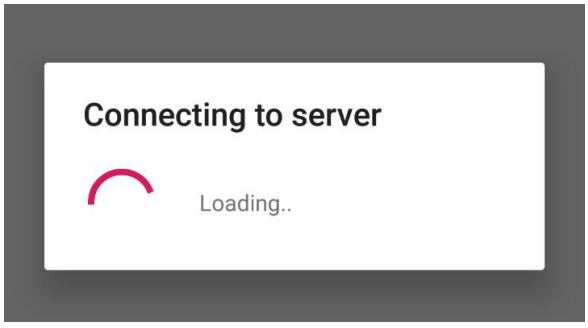

Рисунок 3.12 – Екран очікування з’єднання з сервером

Коли користувач увів всі потрібні дані, він повинен натиснути на кнопку “SIGN IN”, якщо користувач вже був зареєстрований, або “SIGN UP” якщо користувач реєструється. На даному етапі встановлюється з’єднання з сервером, тому користувачеві показується сповіщення про те, що йде з’єднання з сервером (рисунок 3.12). Після того, як з’єднання з сервером встановлене і аутентифікація пройшла успішно, відкриється список онлайн користувачів (рисунок 3.13). У даному списку відображаються усі онлайн користувачі, з якими можна почати обмінюватися даними. Список можна листати вниз-вгору, якщо кількість онлайн користувачів перевищує кількість користувачів які можуть одночасно відображатися в списку.

Для того, щоб розпочати передачу даних з користувачем зі списку, потрібно натиснути на бажаного користувача. Після цього відкриється екран для передачі даних у вигляді чату (рисунок 3.14). Користувачі можуть обмінюватися текстовими повідомленнями, фотографіями та файлами.

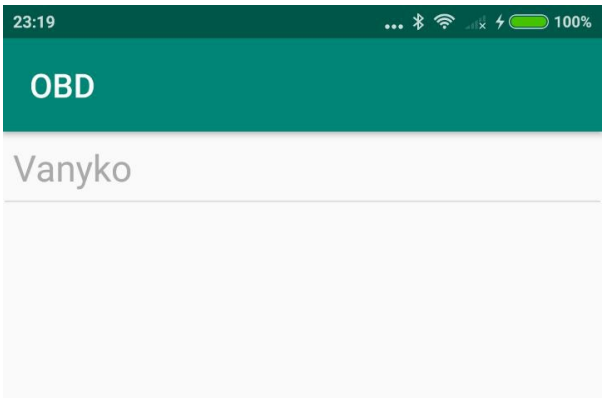

Рисунок 3.13 – Екран списку онлайн користувачів

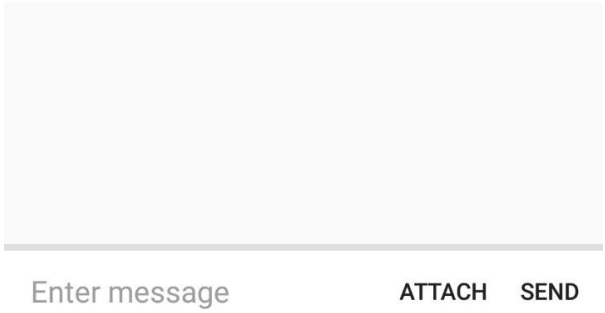

Рисунок 3.14 – Екран передачі даних

Для того, щоб відправити текстове повідомлення, достатньо надрукувати його на клавіатурі (як показано на рисунку 3.15) і натиснути кнопку “SEND”. Після цього повідомлення надсилається на сервер, а вже потім до користувача, якому воно було надіслане. Усі повідомлення зберігаються в історії листування. Повідомлення, які відправив користувач показуються з правої сторони зеленим кольором, а повідомлення які прийняв користувач показуються з лівої сторони оранжевим кольором (рисунок 3.15).

Для того, щоб прикріпити картинку, або файл потрібно натиснути на кнопку “ATTACH”. Коли користувач натисне на цю кнопку, йому відкриється додаток для вибору файлів з пам’яті пристрою (рисунок 3.16). Користувач повинен обрати бажаний файл та натиснути кнопку “Send”. Тоді, файл відправиться до іншого користувача, який зможе його відкрити та редагувати. Коли користувач отримує файл, він автоматично зберігається у сховищі пристрою, у загальній теці файлів додатку.

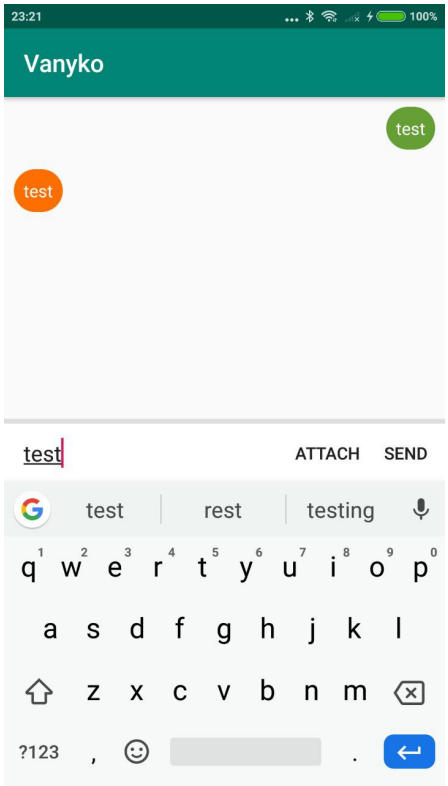

Рисунок 3.15 – Екран обміну текстовими повідомленнями

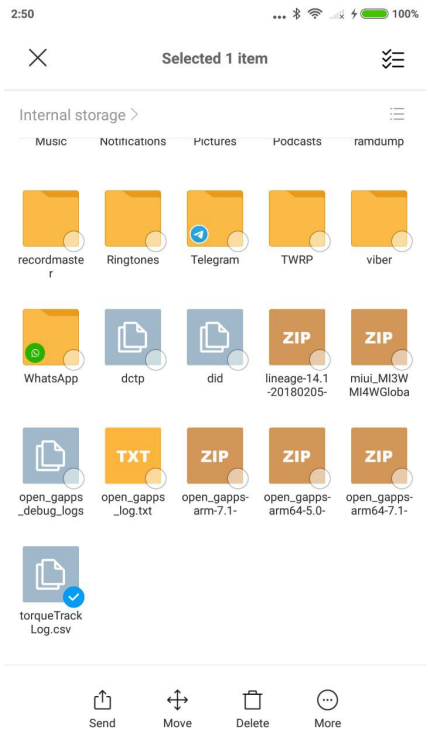

Рисунок 3.16 – Екран вибору файлів з пам’яті пристрою

3.1.3. Пункт меню “Service”

Основне призначення екрану “Service” (рисунок 3.17) – редагування інформації особистого профілю, яка доступна іншим користувачам для перегляду.

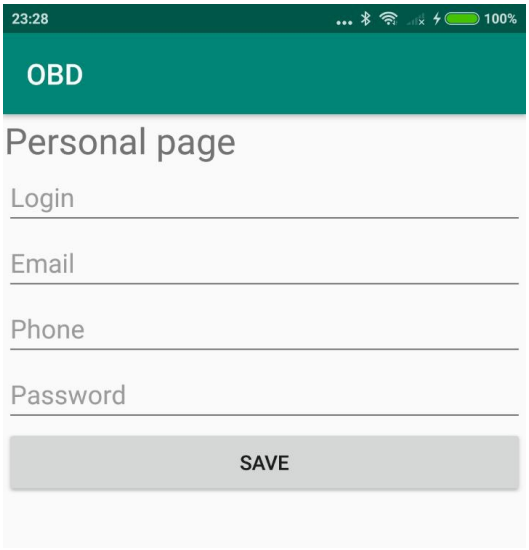

Рисунок 3.17 – Екран редагування особистої інформації користувача

3.2. Архітектура

Додаток складається з декількох Android модулів, кожен з яких має своє призначення:

- storage;
- bluetooth;
- network;
- obd;
- app.

3.2.1. Модуль “storage”

Модуль “storage” містить у собі класи для роботи зі сховищем даних. Android надає декілька варіантів для збереження даних програми. Щоб обрати потрібне сховище, треба визначитися з такими речами, як, наприклад, скільки місця потрібно даним, які дані потрібно зберігати, і чи повинні дані бути приватними для даної програми або доступними для інших програм і користувачів. Список доступних сховищ у Andriod:

Internal file storage - для збереження приватних файлів додатку у файловій системі пристрою.

External file storage - для збереження файлів у спільній зовнішній файловій системі. Зазвичай використовується для спільних файлів користувачів, таких як фотографії.

Shared preferences - для збереження приватних примітивних даних у вигляді пар ключ-значення.

Databases – для збереження структурованих даних в приватній базі даних.

Для збереження даних з OBD використовується “External file storage”. Використання цього типу сховища надає вільний доступ користувачеві до збережених файлів. Таким чином, користувач, при підключенні до комп’ютера у режимі “монтування” матиме вільний доступ до збережених файлів.

Дані логування зберігаються у вигляді CSV (comma separated values), адже цей формат є простим у використанні, не займає багато пам’яті і може відкриватися у більшості програм які працюють з таблицями (наприклад Microsoft Excel).

Дані про налаштування додатку зберігаються у сховищі “Shared preferences”. Цей тип сховища є дуже зручним для збереження налаштувань. Дані в цьому сховищі зберігаються у вигляді пар ключ-значення. Недоліком цього сховища є те, що воно надає можливість зберігати лише примітивні типи даних, таких як Integer, Float, String та інші. А перевагою є те, що це сховище захищене від зовнішнього втручання і має вбудоване шифрування. Тому, воно найбільше підходить для збереження налаштувань.

3.2.2. Модуль “bluetooth”

Модуль “bluetooth” містить у собі застосунки для роботи з Bluetooth в мобільній платформі Android.

Після з’єднання, модуль “bluetooth” повертає модулеві “app” сокет для передачі даних на через Bluetooth.

Управління Bluetooth з’єднанням реалізовано за допомогою шаблону проектування “Одинак” (анг. Singleton). Одинак — це породжувальний шаблон проектування, який гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього. Це надає зручний та уніфікований доступ до управління Bluetooth з’єднанням з різних частин програми.

У додатку 1 показана структура модуля Bluetooth. Розглянемо детальніше кожен з класів.

ConnectedThread унаслідується від класу Thread для того, щоб виконуваний код оброблявся на окремому потоці. Окремий потік потрібен для того, щоб виконувати ресурсозатратні операції, такі як з’єднання, читання і писання у Bluetooth сокет. Також, цей клас зберігає BluetoothSocket і його вхідний (InputStream) та вихідний (OutputStream) потоки даних. Задача цього

класу – керування з’єднання з сокетом та його вхідним і вихідним потоками даних. Потік працює у режимі зчитування, тобто, у режимі постійного очікування повідомлення. Коли повідомлення зчитане, СщттусеувЕркуфв повідомляє MyBluetoothService про нове повідомлення і знову переходить у режим очікування нового повідомлення.

MyBluetoothService – обгортка над ConnectedThread, щоб обмежити прямий доступ до потоку який обробляє BluetoothSocket. Даний клас зберігає у собі посилання на ConnectionCallbacks – інтерфейс, який описує основні функції для роботи з Bluetooth сокетом.

BluetoothManager – основний клас модуля “bluetooth”. Цей клас надає основний інструментарій для роботи з Bluetooth. За допомогою цього класу можна писати у сокет, отримувати повідомлення з сокета, з’єднуватися з пристроєм, перевіряти чи з’єднання сформоване і закривати з’єднання. Також, цей клас надає можливість пошуку пристроїв з увімкненим Bluetooth, які знаходяться в межах видимості Bluetooth приймача. Даний клас зберігає посилання на BluetoothCallbacks – інтерфейс, що містить основні функції для роботи з Bluetooth. BluetoothCallbacks містить у собі наступні функції:

- onConnectionSucceed();
- onConnectionLost();
- onMessageReceived(byte[] bytes);
- onNewDeviceFound(BluetoothDevice device);
- onDiscoveringStateChanged(boolean isDiscovering).

3.2.3. Модуль “network”

Модуль “network” забезпечує взаємодію додатку з сервером. Основними задачами даного модуля є з’єднання з сервером, передача та прийом даних та обробка станів мережі. Модуль опрацьовує такі події як: успішне / не успішне

з’єднання, успішна / не успішна передача / прийом даних, успішне / не успішне роз’єднання.

Управління Internet з’єднанням займається клас, який побудований за допомогою архітектурного шаблону “Одинак”, і називається – ConnectionManager (менеджер з’єднання). Все управління поза модулем “network”, відбувається через менеджер з’єднання.

Для того, щоб встановити з’єднання з сервером потрібно вказати його адресу. Як адресу можна використовувати напряму IP-адресу, так і доменне ім’я. Також, щоб встановити з’єднання потрібно вказати менеджеру з’єднання номер порту, який буде використовуватися. У процесі встановлення з’єднання створюється логічне з’єднання з сервером.

У додатку 2 показана структура модуля Network. Розглянемо детальніше кожен з класів.

Класс Connection – це обгортка над стандартним сокетом у мові Java. Екземпляр класу Connection містить у собі посилання на стандартний сокет Java і описує основні функції для роботи з ним. У даному класі реалізовано функції для зчитування з сокета, запису даних у сокет а також змінну-прапорець яка вказує чи відкрите з’єднання. Для того, щоб отримувати дані одразу, коли вони були передані у сокет з сервера, клас Connection унаслідується від класу Thread (як і у випадку з ConnectedThread у модулі “bluetooth”), і в окремому потоці постійно (поки з’єднання відкрите) виконує операцію зчитування з сокету. Для того, щоб повідомляти про отримані дані, клас Connection зберігає посилання на інтерфейс ConnectionInterface, який описує основні стани з’єднання. Таким чином, коли Connection зчитує нові дані, він викликає функцію onDataRead і передає туди зчитане повідомлення, яке на даному етапі представлене набором байтів.

ConnectionManager – клас, який керує з’єднаннями. Даний клас надає можливості для повного керування життєвим циклом з’єднання. За допомогою цього класу можна створити та закрити з’єднання. Для створення з’єднання

потрібно вказати наступні параметри: хост (доменне ім’я, або IP адресу), номер порту сервера з яким потрібно встановити з’єднання, та чи повинно дане з’єднання використовувати шифрування (SSL / TSL). Також, цей клас надає можливості для надсилання, та зчитування даних з сокету.

Як відомо, операція встановлення з’єднання є досить довгою, тому може статися така ситуація, коли користувач спробує надіслати повідомлення не дочекавшись встановлення з’єднання. Щоб уникнути втрати даних, ConnectionManager містить у собі чергу повідомлень, які потрібно надіслати на сервер. У випадку, якщо з’єднання ще не було встановлено, а користувач намагається надіслати дані, ці дані поміщаються у чергу, і після встановлення з’єднання клас ConnectionManager надсилає всі дані, які були збережені у черзі.

3.2.4. Модуль “obd”

Модуль “obd” містить у собі засоби для роботи з інтерфейсом OBDII. У цьому модулі реалізовані константи, які відповідають різним командам інтерфейсу OBDII, а також класи для обробки відповіді цих команд. Наприклад, для запиту значення обертів двигуна (яке відображається на тахометрі автомобіля), потрібно використати команду "01 0C”. А для обчислення відповіді скористаємося формулою для обрахунку обертів двигуна ((A*256)+B)/4.

Даний модуль також містить так званий OBDManager (менеджер ОБД), задача якого полягає у взаємодії з пристроєм через інтерфейс OBDII. Для того, щоб OBDManager почав зчитувати дані, йому на вхід потрібно передати вхідний (input) і вихідний (output) потоки даних. Це можуть бути як і потоки з Bluetooth сокету, так і з Internet сокету. ОБД менеджер може зчитувати дані як одинично, так і циклічно. Для цього, йому потрібно вказати необхідні команди для зчитування, і запустити його. Після успішного запуску, OBDManager проводить ініціалізацію OBDII контролера, для того, щоб прибрати зайві

символи, дублювання, а також, задати необхідні параметри передачі. Після успішної ініціалізацію менеджер починає зчитувати вказані дані.

Як було визначено емпіричним шляхом, зчитування даних відбувається на різних машинах з різною швидкістю. Велику роль в швидкості передачі грає комп'ютер машини. В середньому, на машині економ класу 2010-х років виробництва, зчитування відбувається зі швидкістю 3-4 параметри в секунду. Отже, мінімальна затримка якої вдавалося досягнути складала 0.25с. Таким чином, можна безперервно зчитувати декілька параметрів з затримкою кадру в N * 0.25, де N – кількість параметрів, або зчитувати один параметр з затримкою в 0.25с.

3.2.5. Модуль "app"

Основний модуль додатку - "app". Цей модуль містить користувацьку логіку додатку, весь графічний інтерфейс, і зв'язує між собою інші модулі.

Основою графічного інтерфейсу в системі Android є Activity (операція). Activity – це компонент, з яким користувач може взаємодіяти для того, щоб виконати ту чи іншу дію, як наприклад, набрати номер телефону чи зробити фотографію. Для кожної Activity надається вікно для відображення відповідного користувацького інтерфейсу. Зазвичай, вікно відображається на весь екран, проте, бувають випадки коли вікна поділяють екран між собою.

Додаток складається з багатьох Activity, кожна з яких грає окрему роль. У додатку для Android оов'язково повинна існувати головна Activity, яка показується одразу після запуску додатку. В свою чергу, операція може викликати інші операції для виконання різних дій. Кожен раз, коли викликається нова операція, попередня зупиняється, але не зникає, система зберігає її в стеку ("стек переходів назад"). Стек переходів назад працює за принципом "останнім зайшов – першим вийшов". Тому, коли користувач закінчує поточну операцію, поточна операція видаляється зі стеку, і відновлюється попередня операція, яка була на вершині стеку.

Не менш важливою складовою графічного інтерфейсу Android є фрагменти (Fragment). Фрагмент являє собою частину користувацького інтерфейсу в операції. Вони створені для того, щоб декілька фрагментів компонувати в одну операцію, для побудови багатофункціонального користувацького інтерфейсу з можливістю перевикористання різних фрагментів. Таким чином, фрагмент можна розглядати як частину операції. Кожен фрагмент має свій окремий життєвий цикл і самостійно обробляє дії користувача. Ще одною важливою особливістю фрагментів є те, що їх можна додавати і видаляти прямо під час роботи програми.

В розробленому додатку основною операцією є користувацьке меню. Тобто, головна операція містить у собі інтерфейс меню, а також контейнер для основних пунктів меню, які представлені у вигляді фрагментів. За допомогою меню, яке розташоване внизу екрану, користувач має змогу переключатися між фрагментами, які не перестворюються щоразу, як користувач їх вмикає, що забезпечує високу швидкодію і плавний інтерфейс додатку.

Модуль “app” також реалізовує роботу з мережею на прикладному рівні. Розглянемо структуру роботи з мережею детально.

Клас NetworkManager виконує роль обгортки над класом ConnectionManager і може відкривати та закривати з’єднання а також надсилати дані. NetworkManager декодує дані з масиву байтів у прикладний клас DecodedMessage, який містить у собі дані в зручному для використання вигляді. DecodedMessage складається з двох класів: MsgHeader і MsgContent. MsgHeader містить у собі дані із заголовку повідомлення у вигляді набору різних параметрів, а MsgContent містить у собі саме дані, які були передані, у вигляді масиву байтів. Після декодування даних NetworkManager передає їх до класу, який реалізує інтерфейс MessagesListenerInterface.

Для того, щоб організувати взаємодію з сервером, повідомлення повинні будуватися з однаковою структурою. Тому, для того, щоб клієнт та сервер

могли розуміти одне одного, був розроблений протокол для повідомлень. Протокол має наступний вигляд:

SOH<Header length>STX<Content length>STX<Header><Content>

Символи SOH та STX виконують роль розділювачів. У десятковому представлення символи SOH та STX мають значення 1 та 2 відповідно.
Header length – це довжина заголовку (Header) у десятковому форматі.
Content length – це довжина контенту (Content) у десятковому форматі.
Header – заголовок, у JSON форматі. У заголовку передаються наступні дані:

- keyAction – строка, яка описує основне призначення повідомлення. Допустимі значення keyAction:
  - “echo” – використовується для тестування серверу. Якщо надіслати дане повідомлення на сервер, то сервер поверне у відповідь точно те саме повідомлення.
  - “msg” – використовується для передачі текстових повідомлень від одного користувача до іншого.
  - “usersListRequest” – використовується для запиту у сервера списку онлайн користувачів.
  - “usersListResponse” – надсилається сервером, у відповідь на запит “usersListRequest”. У контенті надається список усіх онлайн користувачів у форматі масиву JSON об’єктів.
  - “loginRequest” – надсилається сервером, для того, щоб повідомити клієнта, що сервер очікує аутентифікації користувача.
  - “loginResponse” – надсилається клієнтом, у відповідь на “loginRequest”. У контенті надається унікальний ідентифікатор користувача, який був збережений після успішної реєстрації, а також, пароль користувача.
  - “loginAck” – надсилається сервером у відповідь на “loginResponse”. У контенті міститься логічний прапорець, який вказує чи була успішною аутентифікація.

- “registerRequest” – надсилається сервером для того, щоб зареєструвати користувача. У контенті містяться наступні дані:
  - login – ім’я користувача;
  - email – електронна пошта користувача;
  - phone – номер телефону користувача;
  - password – пароль користувача для наступної аутентифікації.
- “registerResponse” – надсилається сервером у відповідь на “registerRequest”. У контенті міститься логічний прапорець який вказує чи була реєстрація успішною а також унікальний ідентифікатор користувача для подальшої аутентифіеації.

Клас MessagesHandler – це обробник повідомлень. Даний клас реалізує інтерфейс MessagesListenerInterface, і підписується на обробку нових повідомлень з NetworkManager. Також, MessagesHandler зберігає посилання на обробники таких подій, як аутентифікація, оновлення списку онлайн користувачів та отримання новгого повідомлення. При надходженні нового повідомлення, MessagesHandler аналізує заголовок повідомлення і, в залежності від заголовку повідомлення, інформує слухача, який повинен обробляти це повідомлення. В основному, на події які надлисає MessagesHandler підписуються різні Activity, або класи, що виконують управління відповідними даними.

3.3. Тестування

Щоб упевнитися у працездатності додатку, перед наданням у використання, його потрібно протестувати. Для успішного тестування програмного засобу потрібно перевірити працездатність усіх його функцій. Це досягається або за рахунок людини-тестувальника, яка вручну проводить тестування програмного забезпечення, або за рахунок написання автоматизованих тестів, які являють собою окрему програму, яка виконує певні дії над додатком, який тестується.

Додаток, що розроблявся у даному дипломному проекті, було протестовано вручну, на декількох “середовищах”. Середовищем називають набір конкретних засобів (як програмних, так і апаратних), які потрібні для функціонування додатку. У даному випадку, під середовищем розуміється набір мобільних пристроїв, автомобілів, а також модулів для діагностування автомобілів на базі OBDII. Перелік автомобілів на яких було протестовано додаток:

- Volkswagen Golf IV 2008 дизель 2.0;
- Mitsubishi Lancer 2010 бензин 1.6.

У якості OBDII сканера було обрано пристрій Viecar. Сканер Viecar має багато переваг, серед яких можна визначити наступні:

- висока швидкодія;
- наявність Bluetooth 4.0;
- підтримка всіх протоколів OBDII;
- компактний корпус;
- стильний дизайн.

Під час тестування були протестовані все функції, на обидвох автомобілях. Зчитування даних OBD відбувається без помилок і великих затримок. В середньому на цих автомобілях час зчитування одного параметру складає 0.3 секунди.

Мобільні додатки були протестовані на версіях Android: 6.0, 7.0, 8.0, 9.0. Пристрої, що використовувалися для тестування були від виробників Samsung та Xiaomi.

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ СЕРВЕРУ ДЛЯ ЗБЕРЕЖЕННЯ ТА ОБМІНУ ДАНИМИ

Для того, щоб користувачі могли обмінюватися інформацією між собою через Internet, потрібно реалізувати сервер.

Сервер – це програмно-апаратний комплекс, що може автономно працювати та надавати свої послуги великій кількості користувачів. Зазвичай, сервер знаходиться у віддаленому місці, недоступному для користувачів.

Сервери можуть працювати без втручання людини протягом довгого проміжку часу. Але, існують ситуації, коли спеціаліст (системний адміністратор) повинен втрутитися у роботу серверу, наприклад для оновлення програмної чи апаратної складової, або при поломці. Для того, щоб зробити сервер якомога надійнішим, було розроблено два окремі програмні додатки. Таким чином, якщо станеться неполадка на одному сервері, інший сервер продовжить свою роботу у звичному режимі. Це відповідає теперішнім стандартам розробки серверного забезпечення. Відмовостійкість серверу є дуже важливим показником, адже якщо користувачів багато, то відмова серверу може спричинити масу незручностей для користувачів, а для підприємства – великі втрати коштів.

4.1. Сервер для взаємодії з базою даних

Даний сервер призначений для зберігання основних даних користувачів, таких як дані особистої сторінки та записи логів та помилок. Зберігання даних на сервері є дуже важливою функцією, адже воно забезпечує резервне копіювання даних з мобільних пристроїв. Тому, якщо користувач видалить додаток, чи зітре його дані, він зможе отримати копію даних, що зберігається на сервері. Також, на сервері зберігаються дані особистої сторінки користувача, для того щоб зробити можливим процедуру реєстрації та аутентифікації.

Щоб зберігати дані користувачів, повинна бути база даних, у якій ці дані будуть записуватися. В якості системи керування базою даних була обрана MySQL. MySQL – одна з найпопулярніших систем керування реляційними базами даних. Переваги даної системи є простота, швидкодія, кросплатформенність та наявність зручної документації.

Для доступу до даних, які зберігаються у БД, реалізовано REST api. REST (скор. англ. Representational State Transfer, «передача репрезентативного стану») – рекомендації проектування архітектури мережевих протоколів, які забезпечуєть зручний та гнучкий доступ до даних, які зберігаються на сервері. REST архітектура визначає чотири основні HTTP методи для доступу до даних:

- “GET” – використовується для отримання даних;
- “PUT” – використовується для редагування даних;
- “POST” – використовується для створення нових даних;
- “DELETE” – використовується для видалення даних.

Розглянемо детально дані, які можуть зберігатися на сервері.

- User – структура, яка містить дані облікового запису користувача:
  - id – унікальний ідентифікатор користувача;
  - login – ім’я користувача, яке доступне для перегляду іншими користувачами;
  - phoneNumber – номер телефону користувача;
  - email – поштова скринька користувача;
  - password – пароль для входу в особистий кабінет;
- Car – структура, яка містить дані про транспортний засіб користувача:
  - id – унікальний ідентифікатор автомобіля;
  - brand – назва фірми виробника автомобіля;
  - model – назва моделі автомобіля;
  - vinCode – VIN код автомобіля;
  - engine – дані про двигун автомобіля;

- ecuId – ідентифікатор блоку управління автомобілем;
- firmwareVersion – версія прошивки автомобіля;

- CarLog – структура, що містить у собі файли логів;
- CarErrorLog – структура, що містить у собі файли помилок.

4.2. Сервер для передачі даних між клієнтами

Другий сервер реалізовує передачу даних між користувачами у режимі реального часу. Користувачі можуть передавати текстові повідомлення, фотографії а також файли.

Основною складовою частиною даного серверу є серверний сокет. Серверний сокет призначений для отримання з'єднання від клієнтського сокету, та виділення нових ресурсів для кожного нового з'єднання.

При запуску даного серверу стартує два мікро-сервери: BasicServer і SSLServer. Основна задача цих мікросерверів – отримувати нові з'єднання від клієнтський сокетів.

BasicServer реалізує основний функціонал для роботи з серверним сокетом. Даний клас, при запуску, створює серверний сокет і в циклі починає очікувати на підключення до нього. Так, як операція очікування на нове підключення є блокуючою для потоку, на якому вона виконується, інші операції в даному потоці не будуть виконуватися. Тому, сервер може взаємодіяти тільки з одним клієнтом. Щоб цього уникнути, BasicServer унаслідується від класу Thread для того, щоб виконувати операцію очікування на нове підключення в окремому потоці, не блокуючи роботу всього серверу.

Коли серверний сокет отримує підключення, виділяється окремий клієнтський сокет для цього підключення, і створюється об'єкт класу Connection. Клас Connection – це обгортка над стандартними сокетами у мові Java. Він надає можливості зчитування, запису за закриття сокету.

Після того, коли об’єкт класу Connection створений, він передається у конструктора класу UserHandler. Клас UserHandler містить у собі основну логіку роботи з клієнтом. Даний клас створюється для кожного нового підключення, і грає роль менеджера підключення. Для того, щоб реалізувати роботу з багатьма клієнтами, кожен екземпляр класу UserHandler унаслідується від класу Thread. Таким чином, запити кожного користувача обробляються одночасно і окремо від інших користувачів.

UserHandler отримує нове повідомлення у вигляді масиву байтів. Для того, щоб зрозуміти суть повідомлення, потрібно його декодувати. Як було сказано раніше, дані структуруються по наперед визначеному протоколу. Щоб декодувати дані, UserHandler звертається до класу MessageProtocolUtils і викликає у нього функцію decodeRawMessage. Ця фунекція призначена для того, щоб переводити масив байтів у клас DecodedMessage використовуючи прикладний протокол повідомлень.

Після створення UserHandler, BasicServer записує посилання на даний екземпляр у список userHandlers. Список userHandlers потрібен для того, щоб відслідковувати життєвий цикл кожного з’єднання. Також, цей список потрібен для того, щоб перенаправляти повідомлення від одного користувача до іншого. Наприклад, коли UserHandler приймає повідомлення, яке повинно бути надіслане до іншого користувача, то UserHandler викликає у об’єкта BasicServer функцію sendDataToUserWithID, і передає туди ідентифікатор користувача, до якого треба надіслати повідомлення, і саме повідомлення, у вигладі масиву байтів. BasicServer шукає у списку userHandlers клієнта з відповідним ідентифікатором, і надсилає йому повідомлення. Отже, по суті, userHandlers це список який містить у собі всіх користувачів, які на даний момент під’єднані до серверу.

При підключенні нового користувача, всі клієнти отримують сповіщення про те, що змінився список онлайн користувачів, і отримують новий список користувачів.

При відключенні існуючого користувача від серверу, UserHandler об’єкт, який відповідав за керування даним з’єднанням, очищує всі свої дані, та передає класу BasicServer сповіщення про те, що дане з’єднання більше не активне. Коли BasicServer отримує таке сповіщення, то він видаляє зі списку userHandlers об’єкт, що відповідав даному з’єднанню, і, як і у випадку з підключенням, інформує всіх інших користувачів про зміну списку онлайн користувачів.

BasicServer функціонує таким чином, що дані у незмінному вигляді передаються з сервера та на сервер. Таким чином, якщо зловмисник захоче перехопити будь які дані, то він зробить це без великих зусиль. Такий варіант передачі не підходить для обміну даними, які містять конфіденційну інформацію. Для того, щоб забезпечити безпечний обмін даними, було створено додатковий сервер, який має назву SSLServer.

SSLServer – сервер, який використовується для безпечного обміну даними між користувачами. SSLServer базується на використанні TSL протоколу, і виконує шифрування за допомогою асиметричного алгоритму з відкритим ключем.

Асиметричний алгоритм шифрування з відкритим ключем базується на використанні двох ключів. Кожен з ключів може використовуватися для шифрування повідомлення, проте лише один з них для розшифрування. Ключ, який використовується для розшифрування називається приватним, і повинен залишатися в таємниці. Інший ключ, за допомогою якого можна виконувати лише шифрування даних, вважається публічним, і може бути опублікованим.

При встановленні з’єднання з сервером SSLServer, окрім встановлення базових параметрів передачі під час рукостискання, також узгоджуються алгоритми шифрування та виконується обмін ключами. Спочатку клієнт надсилає список підтримуваних шифрів і хеш функцій. З цього списку сервер обирає найбільш захищений шифр, та хеш функцію які він підтримує. Після того, як необхідні параметри були обрані, сервер надсилає до клієнта ці

параметри у вигляді сертифікату. Зазвичай, сертифікат містить назву серверу, довірений Центр Сертифікації та свій публічний ключ для подальшого шифрування даних. Деякі клієнти після цього зв’язуються з центром сертифікації, який видав сертифікат, і перевіряють чи даний сертифікат є справжнім. Але, сертифікат який використовується у даній дипломній роботі був створений на локальному сервері, і не може бути підтверджений центром сертифікації, тому клієнт пропускає перевірку сертифікату. Після цього починається генерація ключів сеансу. Для того, щоб згенерувати ключі сеансу, клієнт генерує випадкове число і зашифровує його за допомогою відкритого ключа серверу, і надсилає його на сервер. Якщо приватний ключ не було перехоплено, то тільки сервер може розшифрувати дане повідомлення. Зловмисник може знати тільки публічний ключ, але розшифрувати повідомлення за допомогою публічного неможливо. На основі випадкового числа, обидві сторони (клієнт та сервер) створюють ключові дані для шифрування та розшифрування даних. На цьому рукостискання закінчується, і з’єднання вважається захищеним.

Користувацькі дані шифруються і розшифровуються тільки за допомогою ключових даних, які були створені під час встановлення рукостискання.

ВИСНОВКИ

Усі автомобілі, які заріз виробляються, підтримують діагностику на базі OBDII інтерфейсу. В деяких країнах, на законодавчому рівні, зобов'язують автовиробників влаштовувати підтримку OBDII інтерфейсу у свої автомобілі.

Дуже багато автомобілістів вже добре знайомі з даним інтерфейсом, і, навіть, мають свої власні OBDII адаптери. Майже всі адаптери, які існують на масовому ринку базуються на основі чіпів ELM, адже ці чіпи підтримують всі існуючі протоколи для роботи з OBDII інтерфейсом.

У результаті аналізу існуючих рішень для діагностування автомобілів на мобільній платформі Android, було визначено, що їх основний недолік – необхідність бути поруч з автомобілем для проведення діагностики. Це означає, що автомеханікам потрібно їхати до клієнта, щоб провести діагностику автомобіля, або, навпаки, клієнту потрібно їхати в сервісний центр для проведення діагностики.

Головною метою даного дипломного проекту було створення додатку, який дозволяє проводити віддалене діагностування автомобілів. Це дозволить зменшити витрати коштів і часу для діагностування автомобіля як для власника автомобіля, так і для сервісного центру.

Для того, щоб використовувати додаток, що розроблений у даному дипломному проекті, достатньо мати Android телефон, з доступом в Internet, і OBDII адаптер, який підтримує передачу даних, використовуючи технологію Bluetooth.

Розроблений додаток, дозволяє робити діагностування основних показників автомобіля, їх логування та збереження у сховищі телефону. Також, особливістю даного додатку є те, що він надає можливість віддаленого діагностування автомобіля. Для цього варто лише під'єднати телефон до OBDII адаптера, і надсилати дані іншому користувачеві, який, наприклад, може бути автомеханіком. Таким чином, автомеханіки зможуть віддалено проводити

діагностування та моніторинг автомобіля, без прив’язки до місцезнаходження транспортного засобу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Васильев А. Программирование на Java для начинающих / Алексей Васильев., 2017. – (Питер).
2. Дарвин Я. Android. Сборник рецептов. Задачи и решения для разработчиков приложений / Ян Дарвин., 2016.
3. Нимейер П. Программирование на Java для начинающих / П. Нимейер, Д. Леук., 2014. – (Эксмо).
4. Сьерра К. Изучаем Java / К. Сьерра, Б. Бейтс., 2016. – (Эксмо).
5. Филлипс Б. Android. Программирование для профессионалов / Б. Филлипс, К. Стюарт, К. Марсикано., 2016. – (Питер).
6. Шилдт Г. Java. Полное руководство / Герберт Шилдт., 2016. – (Вильямс). – (Полный справочник; № 10).
7. Эккель Б. Филосовия Java / Брюс Эккель., 2016. – (Питер).
8. Griffiths D. Head First Android Development: A Brain-Friendly Guide / D. Griffiths, D. Griffiths., 2015.
9. Lake I. Professional Android / I. Lake, L. Meier., 2018.
10. Phillips B. Android Programming: The Big Nerd Ranch Guide / B. Phillips, B. Hardy., 2012.
11. OBD-II (Check Engine Light) Trouble Codes [Електронний ресурс] – Режим доступу до ресурсу: https://www.obd-codes.com/trouble_codes/.
12. OBD-II PIDs [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: http://obdcon.sourceforge.net/2010/06/obd-ii-pids/.