VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

AUTENTIZACE POMOCÍ MOBILNÍ APLIKACE S VYUŽITÍM TECHNOLOGIE BLE

AUTHENTICATION USING MOBILE APPLICATION AND BLE TECHNOLOGY

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Martin Žigrai

AUTHOR

VEDOUCÍ PRÁCE — Ing. Patrik Dobiáš

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Martin Žigrai ***ID:*** 231307

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Autentizace pomocí mobilní aplikace s využitím technologie BLE

POKYNY PRO VYPRACOVÁNÍ:

V rámci této práce se student seznámí s vývojem mobilních aplikací se zaměřením na použití kryptografických funkcí a bezdrátovou komunikaci pomocí technologie BLE. Cílem je vytvořit multiplatformní aplikaci s jednoduchým uživatelským rozhraním, která bude zaměstnanci ve fiktivní firmě sloužit k přístupu k vozidlu. Aplikace by měla umožňovat rezervaci vozu a následnou autentizaci uživatele při přístupu do vozu. Pro bezdrátovou komunikaci a potřebné kryptografické funkce student využije existující knihovny. Dílčím cílem je také měření a vyhodnocení výpočetních a paměťových nároků použitých kryptografických primitiv.

DOPORUČENÁ LITERATURA:

[1] MENEZES, Alfred, Paul C VAN OORSCHOT a Scott A VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] Townsend, K. and Davidson, R. and Cufí, C. Getting Started with Bluetooth Low Energy. O'Reilly 2014. ISBN 978-1491949511.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Patrik Dobiáš

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Hlavnou myšlienkou tejto bakalárskej práce je návrh a vytvorenie multiplatformovej mobilnej aplikácie. Táto mobilná aplikácia má slúžiť na rezerváciu vozidla a autentizáciu pre prístup do vozidla. Cieľom aplikácie je poskytnúť, každému kto túto aplikáciu využíva jednoduché a spolahlivé prostredie, v ktorom si môžu rýchlo a jednoducho vyhľadať a rezervovať dané vozidlo. Mobilná aplikácia komunikuje s daným vozidlom pomocou technológie BLE. Pre zabezpečenie bezpečnosti a dôvernosti komunikácie, boli využité kryptografické primitíva ako AES a SHA.

Súčasťou bakalárskej práce je prehľad o vývoji mobilných aplikácií, nástrojoch pre vývoj, prehľad kryptografických primitív alebo technológie Bluetooth. Ďalšou časťou je návrh mobilnej aplikácie, prehľad kryptografických knižníc vo Flutteri a implementácia mobilnej aplikácie.

KĽÚČOVÉ SLOVÁ

AES, Android, Android Studio, BLE, Dart, Firebase, Flutter, GCM, iOS, Multiplatformová mobilná aplikácia, SHA.

ABSTRACT

The main idea of this bachelor's thesis is the design and development of a cross-platform mobile application. This mobile application is intended for vehicle reservation and authentication for accessing the vehicle. The goal of the application is to provide a simple and reliable environment for users to quickly and easily search for and reserve a specific vehicle. The mobile application communicates with the vehicle using BLE technology. To ensure the security and confidentiality of communication, cryptographic primitives such as AES and SHA have been utilized.

The bachelor's thesis includes an overview of mobile application development, development tools, an overview of cryptographic primitives, and Bluetooth technology. Another part is the design of the mobile application, an overview of cryptographic libraries in Flutter, and the implementation of the mobile application.

KEYWORDS

AES, Android, Android Studio, BLE, Dart, Firebase, Flutter, GCM, iOS, Multiplatform mobile application, SHA.

Vysadené pomocou balíčku `thesis` verzie 4.07; `http://latex.feec.vutbr.cz`

ŽIGRAI, Martin. *Autentizace pomocí mobilní aplikace s využitím technologie BLE*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 59 s. Bakalárska práca. Vedúci práce: Ing. Patrik Dobiáš,

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Martin Žigrai
VUT ID autora:	231307
Typ práce:	Bakalárska práca
Akademický rok:	2022/23
Téma závěrečnej práce:	Autentizace pomocí mobilní aplikace s využitím technologie BLE

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu bakalárskej práce pánovi Ing. Patrikovi Dobiášovi, za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Každý z nás sa už v živote stretol s rôznymi mobilnými aplikáciami, sú súčasťou nášho každodenného života. Tieto mobilné aplikácie sa používajú na rôznych typoch mobilných zariadení a taktiež sú zamerané na plnenie rôznorodých úloh takmer vo všetkých typoch odvetví. Výnimkou nie je ani zdieľaná doprava.

Táto práca sa zameriava na vývoj mobilnej aplikácie, ktorá bude umožňovať zamestnancovi rezerváciu vozidla a následnú autentizáciu zamestnanca pre prístup do vozidla. Táto mobilná aplikácia využíva kryptografické funkcie a bezdrôtovú komunikáciu pomocou technológie BLE (Bluetooth Low Energy).

V prvej kapitole je uvedený teoretický úvod do vývoja aplikácií zameraných na mobilné zariadenia. Ďalej je rozobraný tzv. natívny a multiplatformový vývoj. A nakoniec je popísaný programovací jazyk Dart a programovacie rozhranie Flutter.

Druhá kapitola bakalárskej práce je zameraná na skúmanie kryptografických primitív. V tejto kapitole sú popísané hashovacie funkcie, symetrická a asymetrická kryptografia a ich súčasti.

Tretia kapitola hovorí o technológiach Bluetooth a BLE. Ďalej popisuje ich spoločné vlastnosti a ich rozdiely.

Štvrtá kapitola je zameraná na návrh mobilnej aplikácie. Je tu zahrnutý takzvaný UML Use Case diagram, ER diagram, ktorý zobrazuje návrh databázy, taktiež je tu návrh uživateľského rozhrania a štruktúra mobilnej aplikácie.

Piata kapitola prezentuje kryptografické knižnice vo Flutteri. Tieto knižnice implementujú rôzne kryptografické primitíva. V tejto kapitole sú jednotlivé knižnice otestované a porovnané na základe času vykonávania implementovaných funkcií.

Posledná kapitola popisuje implementáciu mobilnej aplikácie. Je tu popísaný postup implementovania konkrétnych funkcionalít, ako napríklad autentizácia pomocou Firebase, práca s dátami, komunikácia s GATT serverom a využívanie kryptografie v mobilnej aplikácii.

1 Úvod do vývoja mobilných aplikácií

Táto časť opisuje teoretický úvod do vývoja mobilných aplikácií. Najprv je popísaný pojem mobilná aplikácia. Následne je porovnaný natívny a multiplatformový vývoj. A nakoniec programovací jazyk Dart a programovacie rozhranie Flutter.

1.1 Mobilná aplikácia

Mobilná aplikácia, taktiež označovaná ako „appka“, je počítačový program, alebo softwarová aplikácia, ktorá je určená pre mobilné zariadenia. Týmito zariadeniami môžu byť napríklad telefóny, tablety alebo hodinky. Mobilné aplikácie slúžia na to, aby poskytovali užívateľom podobné služby, aké sú dostupné na desktopových počítačoch alebo webových prehliadačoch. Mobilné aplikácie sú ľahko poúžívateľné, užívateľsky priateľské, lacné a taktiež spustiteľné takmer na všetkých typoch mobilných zariadení [23].

Hlavnou myšlienkou mobilných aplikácií bola podpora produktivity a preto boli vytvárané aplikácie typu email, kalendár, databáza kontaktov a rôzné iné aplikácie podobného charakteru. Neskôr sa mobilné aplikácie rozšírili aj do iných oblastí a vznikli mobilné hry, aplikácie využívajúce GPS a lokalizačné služby, aplikácie zamerané na donášku jedla, aplikácie pre zdieľanú dopravu a iné [23, 29].

Mobilné aplikácie sa na základe počtu operačných systémov, na ktorých môžu operovať delia na:

- natívne mobilné aplikácie,
- multiplatformové alebo cross-platformové mobilné aplikácie.

1.1.1 Vývoj natívnych mobilných aplikácií

Natívna mobilná aplikácia je taká, ktorá je navrhnutá a naprogramovaná na konkrétny typ operačného systému. Tieto aplikácie môžu byť navrhnuté napríklad pre operačný systém Android alebo iOS. Aplikácie tohto typu sú naprogramované pomocou SDK (Software Development Kit) daného operačného systému a majú prístup k hardwarovým prostriedkom mobilného zariadenia. Medzi tieto prostriedky patrí napríklad: kamera, GPS, bluetooth, úložisko zariadenia a rôzne biometrické senzory [33].

Natívne mobilné aplikácie majú svoje výhody a taktiež nevýhody. Tieto výhody a nevýhody sú uvedené v nasledujúcej tabuľke [27].

Tab. 1.1: Výhody a nevýhody natívnych apliácií.

Výhody	**Nevýhody**
Vyššia ochrana pred zneužitím.	Nutnosť naprogramovať samostatnú aplikáciu pre každú platformu.
Ľahšia údržba aplikácie pre jednu platformu.	Náročnejšia údržba pre viacero platforiem.
Menší výskyt chýb.	Nutnosť znalosti viacerých programovacích jazykov pre rôzne platformy.

1.1.2 Vývoj multiplatformových mobilných aplikácií

Multiplatformový mobilný vývoj je prístup, ktorý umožňuje vytvoriť mobilnú aplikáciu, ktorú je možno spustiť na viacerých operačných systémoch. V takýchto typoch aplikácií je možné využiť časť zdrojového kódu alebo aj celý zdrojový kód. To znamená, že vývojári môžu vytvárať mobilné aplikácie, ktoré fungujú na platformách Android a iOS bez nutnosti písania kódu pre každú platformu zvlášť [25].

Ako aj pri natívnom prístupe, tak aj pri vývoji multiplatformových aplikácií nájdeme výhody a nevýhody, ktoré sú uvedené v nasledujúcej tabuľke [25].

Tab. 1.2: Výhody a nevýhody multiplatformových apliácií.

Výhody	**Nevýhody**
Znovupoužitelnosť kódu.	Nižší výkon.
Úspora času.	Nižšia bezpečnosť.
Efektívne riadenie zdrojov.	

Existuje viacero možností vývoja multiplatformových mobilných aplikácií. Medzi tieto možnosti vývoja sa radia programovacie rozhrania ako napríklad Xamarin, NativeScript, React Native alebo Flutter. Všetky z týchto vymenovaných rozhraní sa môžu slobodne vyvíjať, sú dobre testované a bolo pomocou nich vytvorených mnoho aplikácií. Taktiež sú tieto rozhrania používané vo veľkých organizáciach. Napriek rozsiahlej aplikovateľnosti všetkých rozhraní, vieme iba pomocou Flutteru vyvinúť aplikáciu, ktorá dokáže fungovať na Webe, počítačoch a mobilných zariadeniach zároveň [32].

1.2 Programovací jazyk Dart

Dart je programovací jazyk pre vývoj rýchlych aplikácií na akejkoľvej platforme. Hlavnou myšlienkou tohto programovacieho jazyka je poskytnúť najproduktívnejší programovací jazyk pre vývoj na viacerých platformách. Tento jazyk bol vyvinutý spoločnosťou Google, konkrétne navrhnutý Larsom Bakom a Kasperom Lundom [13].

Dart je typovo bezpečný, používa takzvanú kontrolu statického typu, aby sa zabezpečilo, že sa hodnota premennej zhoduje so statickým typom premennej. Typový systém programovacieho jazyka Dart je tiež flexibilný a umožnuje používať takzvaný dynamický typ v kombinácií s runtime kontrolami, čo vie byť užitočné pri kóde, ktorý musí byť dynamický. Taktiež je tu zakomponovaná tzv. sound null safety, čo znamená, že premenné nemôžu mať hodnotu „Null“, pokiaľ sa to implicitne nedovolí. Týmto sa dokáže zamedziť rôznym nečakaným zlyhaniam aplikácie [11].

Tento jazyk je kompilovaný, objektovo-orientovaný, založený na triedach s garbage-collectorom, ktorý je nevyhnutný pre alokovanie a dealokovanie pamäti. Taktiež sa syntax podobá na syntax jazykov Java, C#, C++ alebo na iné objektovo-orientované jazyky. [11, 13].

1.3 Open-source framework Flutter

Flutter je open-source používateľské rozhranie SDK na vývoj softvéru, ktoré vytvorila v roku 2016 spoločnosť Google. Flutter je hlavne využívaný na vývoj multiplatformových aplikácií pre Android, iOS, Linux, macOS, Windows a iné operačné systémy. Táto platforma využíva progrmovací jazyk Dart, vďaka čomu je vývoj rýchlejší a jednoduchší oproti tradičným metódam [16].

Je jedinečný v tom, že namiesto využívania tzv. web views alebo OEM widgetov, tak vykresluje každý pohladový komponent pomocou vlastného vysokovýkonného vykreslovacieho enginu. Táto vlastnosť umožnuje vytvárať aplikácie, ktoré sú rovnako výkonné, ako natívne aplikácie [39].

Flutter umožňuje tzv. stateful hot-reload počas vývoja aplikácie, čo je považované ako hlavný faktor pre zrýchlenie vývoja. Dart VM bez zmeny vnútornej štruktúry aplikácie, tzn. že všetky prechody a akcie budú zachované po tomto hot-reloade [39].

1.3.1 Architektúra Flutteru

Flutter je navrhnutý ako vrstvený systém s možnosťou rozšírenia. Jeho architektúra funguje ako séria nezávislých knižníc, z ktorých každá je závislá od základnej vrstvy.

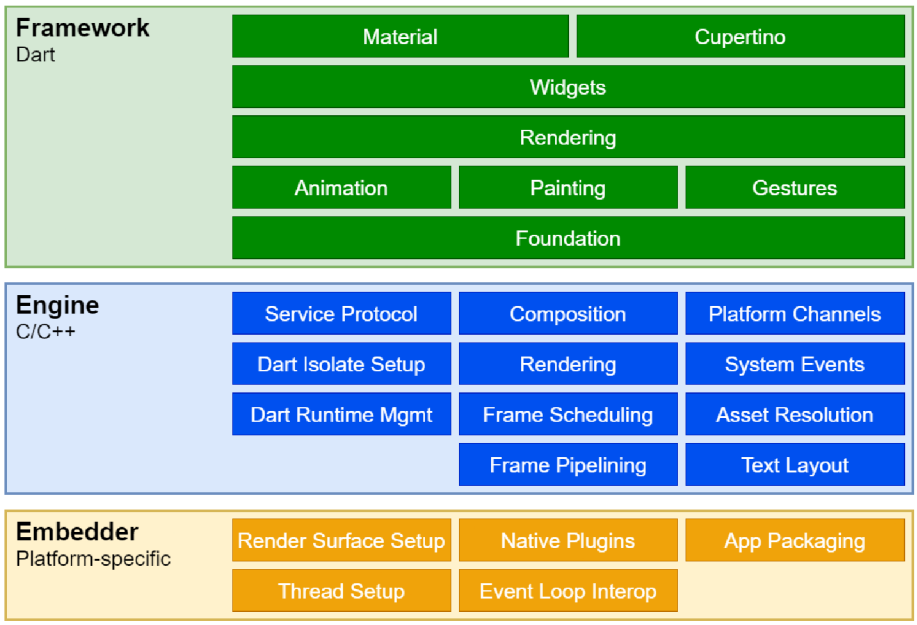
Žiadna z vrstiev nemá prístup k nižšej vrstve a každá časť úrovne rámca je navrhnutá tak, aby bola vymeniteľná a voliteľná [15].

Pre koordináciu so základným operačným systémom a pre prístup k službám ako napríklad zobrazovacie plochy, dostupnosť a vstupy, riadenie udalostí je tu takzvaný embedder, ktorý je špecifický pre danú platformu. Embedder býva napísaný v rôznych programovacích jazykoch. Výber jazyku sa líši od platformy, napríklad pre Android to môže byť Java alebo C++, pre iOS a macOS objektové C, pre Windows a Linux to je C++ [15].

Za jadro Flutteru sa považuje Flutter engine, ktorý je vo väčšine napísaný v jazyku C++. Jeho hlavnou funkciou je podpora primitív nevyhnutných pre podporu všetkých aplikácií Flutteru. Taktiež je zodpovedný za vykreslovanie scén, poskytuje nízkoúrovňovú implementáciu základného API Flutteru vrátane grafiky, rozloženia textu a mnoho iných procesov. Flutter engine komunikuje s Flutter frameworkom pomocou dart:ui, ktorý obaluje kód C++ do Dart tried [15].

Vývojári zvyčajne komunikujú s Flutterom pomocou Flutter frameworku, ktorý je napísaný v jazyku Dart. Nachádza sa tu sada platforiem, layoutov a rôznych knižníc. Flutter framework obsahuje [15]:

- základné triedy a stavebné bloky pre animáciu a maľovanie,
- vrstvu vykresľovania, ktorá poskytuje abstrakciu na riešenie rozloženia,
- vrstvu widgetov, ktorá je abstrakcia kompozície,
- knižnice Material a Cupertino, ktoré ponúkajú komplexné sady ovládacích prvkov.



Obr. 1.1: Architektúra Flutteru [15].

1.4 Vývojové prostredie Android Studio

Vývojové prostredie Android Studio je oficiálné IDE pre vytváranie aplikácií pre operčný systém Android. Toto vývojové prostredie je založené na vývojovom prostredí IntelliJ IDEA. Okrem výkonného editoru, obsahuje aj množstvo nástrojov, ktoré sú určené na zvýšenie produktivity pri vývoji aplikácií, ako napríklad [18]:

- flexibilný stavebný systém založený na Gradle,
- rýchly emulátor plný funkcií,
- funkcia, ktorá bez nutnosti reštartovania aplikácie, doplní novo pridaný kód,
- testovacie nástroje a frameworky,
- nástroje na testovanie výkonu, použiteľnosti, kompatibility verzií a iných problémov.

Jedným z dôležitých testovacích nástrojov Android Studia je Android Studio Profiler. Android Studio Profiler je vysokovýkonný nástroj, ktorý poskytuje v reálnom čase údaje o stave CPU, pamäti, sieti, batérii a iných perifériach. Tento nástroj nám pomáha hlbšie pochopiť ako naša aplikácia využíva dané prostriedky [19].

1.5 Firebase

Platforma Firebase bola pôvodne vytvorená ako realtime databáza slúžiaca pre mobilné a webové aplikácie v roku 2011 Andrewom Leeom a Jamesom Taplinom. Táto platforma bola v roku 2014 odkúpená spoločnosťou Google a odvtedy je jej neodeliteľnou súčasťou [4].

Firebase ponúka veľké množstvo nástrojov a služieb, ktoré môže vývojár využiť pri vývoji mobilných ale aj webových aplikácií. Táto platforma je typu Backend–as–a–Service. V jej ponuke sú nástroje pre autentizáciu, vytvorenie databázy alebo ukladanie rôznych súborov [4].

2 Kryptografické primitíva

Kryptografické primitíva sú nízkoúrovňové algoritmy, ktoré sa môžu použiť ako stavebné bloky pre zabezpečovacie systémy. Kryptosystémy sú súborom týchto stavebných blokov, ktoré implementujú konkrétne bezpečnostné funkcie, ako napríklad šifrovanie alebo jednocestné hashovacie funkcie.

Tieto primitíva sú navrhnuté tak, aby spoľahlivo vykonávali jednu vopred definovanú úlohu. Vytváranie spoľahlivých a funkčných kryptografických primitív je zdĺhavý a náročný proces. Takže z tohto dôvodu je veľmi zriedkavé, aby sa vytváralo nové kryptografické primitívum, ktoré by vyhovovalo potrebám kryptografického systému. Analógiou ku kryptografickým primitívam môžu byť programovacie jazyky. Tak ako si programátor nevytvára vlastný programovací jazyk pri písaní nového programu, tak ani dizajnér kryptosystémov nevytvára nové primitíva. Namiesto vytvárania niečoho nového sa v oboch prípadoch využijú už existujúce možnosti, aby sa vyhlo časovo náročnej a chybám náchylnej práci [7].

2.1 Jednocestné hashovacie funkcie

Hashovacie funkcie sú jedným zo základných primitív v modernej kryptografii. Hashovacia funkcia je výpočtovo efektívna funkcia, ktorá mapuje reťazce rôznorodej dĺžky na reťazce s pevnou dĺžkou, taktiež nazývané ako hash [28].

Pre takúto funkciu, ktorá má výstup n-bitový hash (napr. $n = 128$ alebo 160) a požadované vlastnosti, tak pravdepodobnosť, že sa náhodne zvolený reťazec namapuje na n-bitový hash je 2^{-n}. Hlavnou myšlienkou je to, že hash slúži ako otlačok vstupného reťazca. Pre kryptografické systémy je zvyčajne hashovacia funkcia h volená tak, že tu je kladený požiadavok na to aby bolo nemožné nájsť dva vstupy, ktoré budú mať rovnaký výstup hashovacej funkcie (t. j. dva kolidujúce vstupy x a y, tak že $h(x) = h(y)$). Taktiež je tu kladený požiadavok na to aby sa výstup hashovacej funkcie nedal naspäť previesť do počiatočného reťazca (t. j. z hashu y, $h(x) = y$) [28].

Najbežnejším využitím hashovacích funkcií v kryptografii je pri digitálnych podpisoch a pri zaručení integrity dát. Pri digitálnych podpisoch sa dlhá správa hashuje verejne známou hashovacou funkciou a podpísaný je iba hash, čiže výstup tejto funkcie. Pri overovaní pravosti podpisu, príjmajúca strana správu zhashuje a posudzuje či je prijatý podpis pre tento hash správny. Táto metóda šetrí čas a priestor v porovnaní s priamym podpisom, pričom by bolo nutné rozdeliť správu do blokov vhodnej veľkosti a taktiež by bolo nutné podpísať každý blok zvlášť [28].

Hashovacie funkcie dokážu zabezpečiť integritu dát, tak že sa vypočíta zo vstupu v určitom časovom okamžiku hash. Týmto spôsobom sa zabezpečuje integrita dát.

Ak chceme integritu overiť, či nebolo manipulované s dátami alebo neboli zmenené, tak sa v ďalšom časovom okamžiku, pomocou rovnakej hashovacej funkcie znova vypočíta hodnota hash a následne sa porovná s pôvodnou hodnotou. Táto vlastnosť sa dá uplatniť pri ochrane proti vírusom alebo pri distribúcii softwaru [28].

Tretím využitím hashovacích funkcií je ich využitie v protokoloch zahŕňajúce apriórne záväzky, spolu s niektorými schémami digitálneho podpisu a identifikačných protokolov [28].

Väčšina hashovacích funkcií je verejne známa a nezahrňujú žiadne tajné a verejné klúče. Ak sa hashovacie funkcie využívajú na to, či bol zmenený vstup správy, tak sa nazývajú kódy detekcie modifikácií. Podobné týmto funkciám sú aj hashovacie funkcie, ktoré potrebujú tajný klúč a poskytujú overenie pôvodu a integritu dát a nazývajú sa kódy na autentifikáciu správ [28].

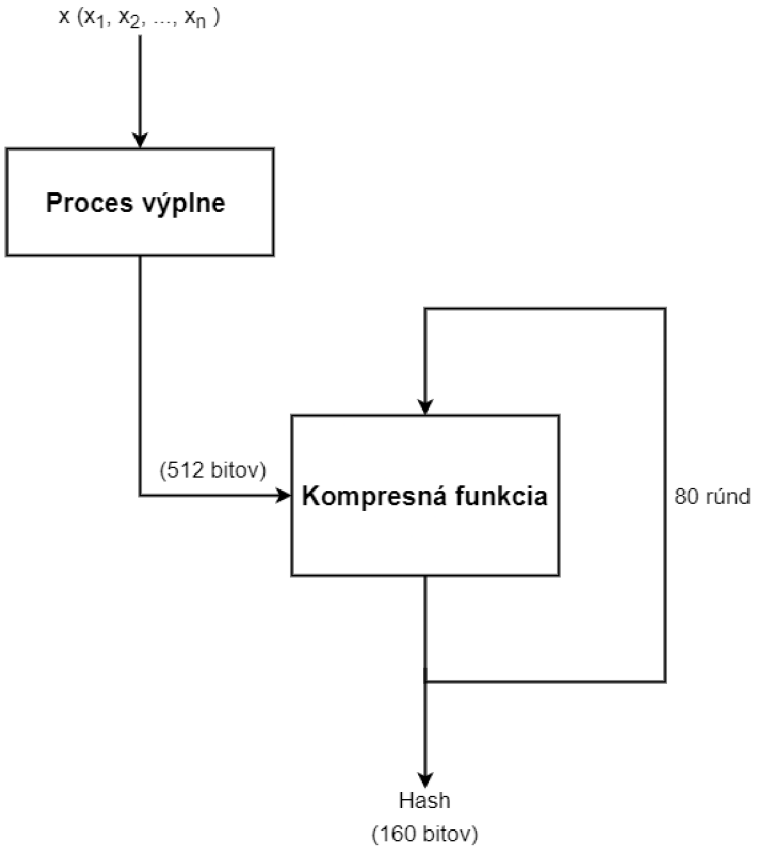
2.1.1 Rodina hashovacích funkcií SHA

Označenie SHA (Secure Hash Algorithm) patrí rodine kryptografických funkcií, ktoré slúžia na zabezpečenie dát. Zloženie tejto funkcie pozostáva z bitových operácií, modulárneho sčítania a kompresných funkcií. Hlavnou myšlienkou tejto funkcie je, že sa pomocou nej dáta pretransformujú na reťazec s pevnou veľkosťou, ktorý sa líši od originálu. Tieto funkcie sú navrhnuté ako jednocestné, čo znamená, že po vytvorení hashu z pôvodného reťazca, je skoro nemožné z tohto hashu odvodiť pôvodný reťazec. Z tejto rodniny napríklad pochádza SHA-1, SHA-2 alebo SHA-3, pričom bola každá z týchto funkcií postupne navrhnutá so silnejším šifrovaním [26].

Zvyčajným využitím funkcie SHA je šifrovanie hesiel, pretože na strane servera je bezpečnejšie ukladať zhashované heslo daného užívateľa a nie konkrétne heslo. Je to bezpečnejšie z toho hľadiska, že keď príde k útoku na databázu, tak útočník získa iba hodnoty hashov daných hesiel a nie skutočné heslá. Takže ak útočník vloží hodnotu hashu do formuláru pre heslo, tak hashovacia funkcia skonvertuje túto hodnotu a premení ju na inú hodnotu a po porovnaní týchto hodnôt príde k zamietnutiu prístupu. Pri hashovacích funkciách SHA sa objavuje takzvaný lavínový efekt, kde pri malej zmene vo vstupe spôsobí veľkú zmenu vo výstupe alebo napríklad pri veľmi odlišných reťazcoch sa vytvárajú podobné výstupné hodnoty. Tento lavínový efekt spôsobí to, že výstupné hodnoty hash neposkytujú žiadne informácie, ktoré sa týkajú vstupného reťazca, ako napríklad pôvodná dĺžka reťazca. Taktiež sú tieto funkcie zabezpečené mechanizmami na detekciu falšovania dát, čo znamená, že pri veľmi málo viditeľnej zmene, hodnota hashu bude iná ako hodnota hashu pôvodného súboru a manipulácia so súborom bude ľahko viditeľná [26].

SHA-1

Medzi jednu z prvých a najznámejších hashovacích funkcií z rodiny SHA patrí hashovacia funkcia SHA-1. Táto hashovacia funkcia, tak ako aj ostatné funkcie z rodiny SHA bola vytvorená americkou národnou bezpečnostnou agentúrov NIST v roku 1995. Je využívaná v bezpečnostných protokoloch a aplikáciach ako napríklad TLS, SSL, SSH alebo IPsec. Napriek tomu, že táto funkcia je stále vysoko používaná, boli v roku 2005 zistené určité slabiny, ktoré by mohli ohroziť túto funkciu. Hlavnou slabinou bol vysoký výskyt kolízií, čo znamená, že dva odlišné vstupy sú mapované na rovnaký výstup, čiže hash [26].



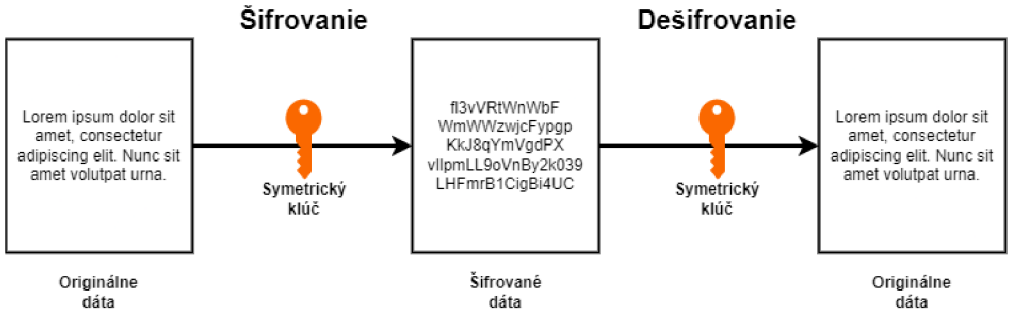
Obr. 2.1: Obecná schéma hashovacej funkcie SHA-1 [3].

Proces hashovania pomocou funkcie SHA-1 by sa dal rozdeliť do dvoch častí. V prvej časti nastáva proces výplne, kde sa správa rozdelí na n častí o veľkosti 448 bitov a ku každej časti sa pridá výplň o veľkosti 64 bitov, čím vznikne blok o dĺžke 512 bitov. Tento blok ďalej putuje do ďalšej časti, čiže prechádza cez kompresnú funkciu. Kompresná funkcia má 80 rúnd a každou rundou prechádzajú všetky bloky. Po vykonaní týchto 80 rúnd získavame výstup o veľkosti 160 bitov [3].

2.2 Symetrická kryptografia

Symetrická kryptografia funguje na princípe znalosti zdieľaného tajomstva pre výmenu šifrovaných dát medzi komunikujúcimi stranami. Pre šifrovanie a dešifrovanie dát sa používa rovnaký kľúč a z tohto dôvodu sa šifry z tejto kategórie nazývajú symetrickými šiframi. Zjednodušene povedané, odosielateľ dát šifruje tieto dáta pomocou tajnej frázy, čiže hesla a príjemca so znalosťou tajnej frázy dokáže dešifrovať dáta [9].

Symetrické šifrovanie je obojsmerný proces. Pri kombinácii kľúč a nešifrovaný text, budú symetrické šifry vždy vytvárať rovnaký šifrovaný text, obr. 2.2. Tento princíp funguje rovnako aj v opačnom poradí čo znamená, že pri použití rovnakého kľúča na dešifrovanie šifrovaného textu bude vytvorený pôvodný text, obr. 2.2 [9].



Obr. 2.2: Zjednodušený model šifrovania dát pomocou symetrického kľúča.

Hlavnou výhodou symetrickej kryptografie je jej relatívne vysoká rýchlosť a taktiež relatívne ľahká hardwerová implementácia. Pri použití symetrickej kryptografie je zaručený určitý stupeň autentifikácie, pretože sa dáta šifrujú jedným symetrickým kľúčom a nemožno ich dešifrovať pomocou iného symetrického kľúča. Komunikujúce strany musia udržiavať symetrický kľúč v tajnosti, aby komunikujúce strany mali istotu, že si vymieňajú dáta medzi sebou a nie medzi stranou, ktorej sa komunikácia netýka [22].

Za veľkú nevýhodu symetrickej kryptografie sa dá považovať problematickosť výmeny tajného kľúča, pretože káždá výmena tajného kľúča musí mať určitý stupeň zabezpečenia. To znamená, že tu je nutnosť zašifrovať tajný kľúč iným tajným kľúčom a príjemca už musí disponovať týmto tajným kľúčom, ktorým si vie dešifrovať zašifrovaný tajný kľúč. A tu môže nastať ďalší problém s nekonečnou závislosťou na inom kľúči [22].

Šifry symetrickej kryptografie sa delia na dve skupiny:

- Blokové šifry,
- Prúdové šifry.

Bloková šifra je šifrovacia schéma, ktorá funguje na princípe delenia dát na bloky fixnej dĺžky t nad abecedou A a pomocou tajného kľúča šifruje celý blok. Blokové šifry sa považujú za najznámejšie techniky symterickej kryptografie [28].

Prúdová šifra sa dá považovať za veľmi jednoduchú blokovú šifru kde má blok dĺžku rovnú jednej. V prípade prúdových šifier sa každý symbol šifruje samostatne. Tieto šifry sa využívajú v prípadoch kde je vysoká chybovosť prenosu alebo v prípadoch kde sa dáta musia spracovávať po jednotlivých symboloch, napr. ak má zariadenie nedostatočnú pamäť alebo je ukladanie dát do vyrovnávacej pamäti obmedzené [28].

Medzi najznámejšie šifry symetrickej kryptografie patria [34]:

- AES,
- DES,
- IDEA,
- Blowfish,
- RC4, RC5, RC6.

2.2.1 Symetrická šifra AES

Symetrická šifra AES je založená na algoritme Rijndael, ktorý predstavuje symetrickú blokovú šifru, ktorá dokáže spracovať dátové bloky o veľkosti 128 bitov s kľúčami o veľkosti 128, 192 a 256 bitov. Algoritmus Rijndael bol pôvodne navrhnutý tak, aby zvládal spracovať aj bloky a kľúče o väčšej veľkosti ale pri tomto štandarde sú používané vyššie spomenuté veľkosti [1].

Autormi algoritmu Rijndael sú Joana Daemena a Vincent Rijmena, ktorý tento algoritmus prihlásili do súťaže o federálny algoritmus AES od organizácie NIST. Organizácia NIST v roku 2001 schválila tento algoritmus pre šifru AES ako najvhodnejšiu možnosť [1].

Proces šifrovania pomocou šifry AES, znázornený na obr. 2.3 funguje tak, že sa na začiatku vstup vloží do dvojdimenzionálneho stavového poľa. Po prvotnom pridaní rundového kľúča, je stavové pole transformované implementovaním rundovej funkcie, ktorá sa opakuje na závislosti veľkosti kľúča (10, 12 alebo 14-krát), pričom posledná runda sa odlišuje od ostatných rúnd [30].

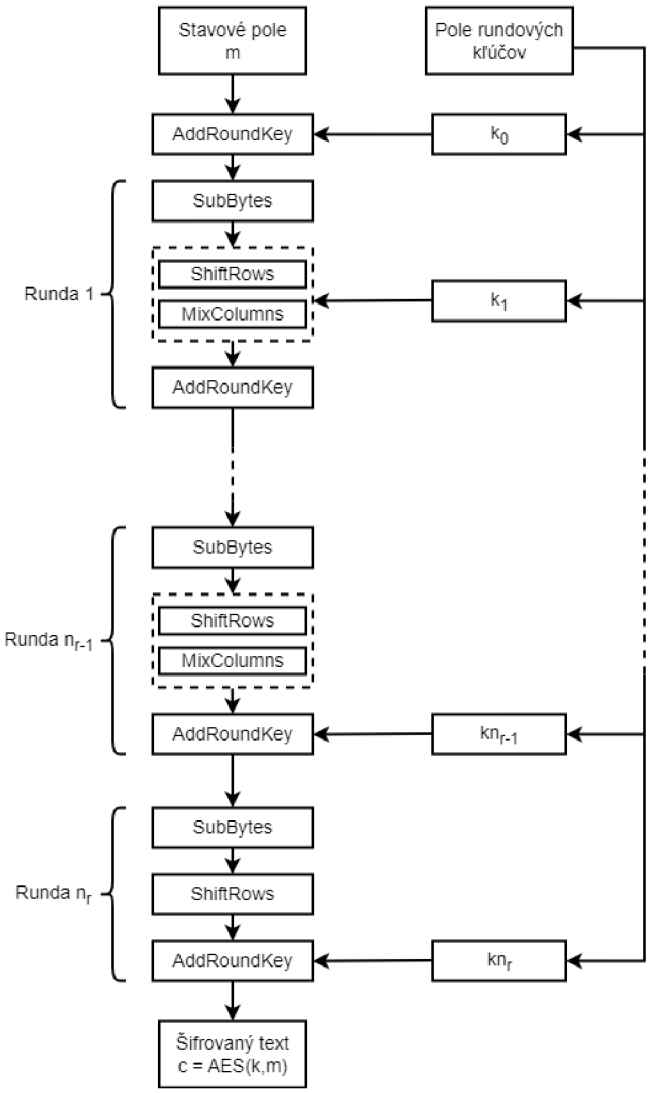
Rundová funkcia je parametrizovaná na základe rundového kľúča, ktorý je vyberaný z jednorozmerného poľa. Jeden rudnový kľúč má veľkosť štyri bajty a je odvodený zo symetrického kľúča pomocou funkcie nazývanej Kľúčová Expanzia. Rundová funkcia sa ďalej delí na podčasti a to SubBytes (zámena bajtov), ShiftRows (prehodenie riadok), MixColumns (kombinovanie stĺpcov), AddRoundKey (pridanie rundového kľúča) [30].

V časti **SubBytes** sa vykonáva nelineárna bajtová substitúcia nezávisle na každej časti stavového poľa pomocou substitučnej tabuľky (S-box). Zjednodušene povedané, nastáva tu zámena každého bajtu v stavovom poli pomocou zameňovacieho boxu [30].

V časti **ShiftRows** sa bajty v riadkoch stavového pola cyklicky posúvajú o rôzne počty bajov, pričom prvý riadok ostáva bez posunu [30].

MixColumns zabezpečuje to, že sa kombinujú 4 bajty v každom stĺpci stavového pola. Väčšinou sa vezmú 4 bajty ako vstup a vracia sa 4 bajtový výstup, kde každý vstupný bajt ovplyvní všetky vstupné bajty [30].

AddRoundKey zabezpečuje, že každý bajt stavového pola skombinovaný pomocou operácie XOR s rundovým kľúčom [30].

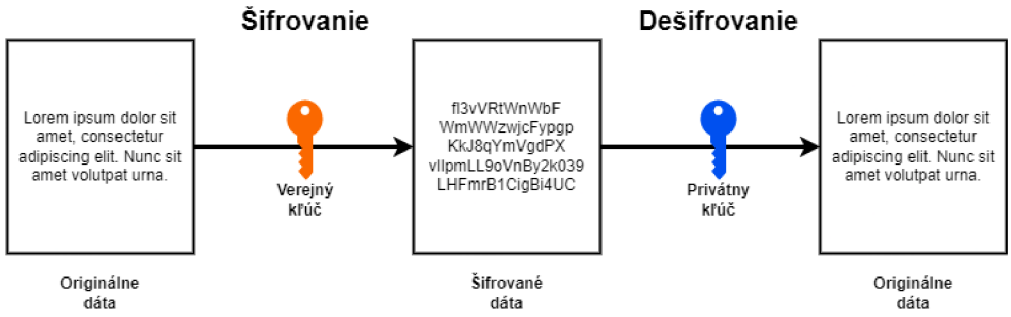


Obr. 2.3: Štruktúra symetrickej šifry AES.

2.3 Asymetrická kryptografia

Asymetrická kryptogafia je taktiež známa pod názvom Kryptografia verejného kľúča. Kryptografia verejného kľúča na rozdiel od symetrickej kryptografie zahŕňa pár kľúčov a nie iba jeden privátny kľúč. Tento pár kľúčov sa skladá z verejného a privátneho kľúča a sú prepojené s entitou, ktorá potrebuje preukázať svoju identitu alebo podpísať či zašifrovať dáta. Podobne ako pri symetrickej kryptografii, tak aj tu je privátny kľúč udržiavaný v tajnosti ale verejný kľúč býva verejne prístupný. Pri asymetrickej kryptografii sa dáta šifrujú verejným kľúčom a dešifrovať ich vieme iba pomocou príslušného privátneho kľúča [21].

Pomocou asymetrickej krypografie vieme zabezpečiť zamaskovanie posielaných dát medzi dvomi komunikujúcimi stranami. To sa dá zabezpečiť, tým že odosieľatel šifruje dáta pred nasledovným odoslaním. A príjemca dát ich po prijatí dešifruje. Pre útočníka je potom veľmi obtiažné zašifrované dáta opätovne dešifrovať. Taktiež je tu zaistená nepopierateľnosť, to znamená že odosielateľ dát nemože po dlhšom čase tvrdiť, že tieto dáta neodoslal [21].



Obr. 2.4: Zjednodušený model šifrovania dát pomocou asymetrickej kryptografie.

Ako je zobrazené na obr. 2.4 tak vlastník privátneho kľúča, môže voľne distribuovať verejný kľúč aby len on mohol dešifrovať dáta, ktoré boli zašifrované týmto verejným kľúčom. Z tohto vyplýva, že ak chce jedna komunikujúca strana poslať zašifrované dáta druhej strane, tak ich musí zašifrovať verejným kľúčom, ktorý získala od druhej strany a druhá strana ich dešifruje pomocou svojho privátneho kľúča [21].

Asymetrická kryptografia na rozdiel od tej symetrickej vyžaduje viac výpočtov. Takže kryptografia verejného kľúča sa zvyčajne nepoužíva pre spracovanie veľkých objemov dát. Asymetrická kryptografia rieši problém distribúcie tajného kľúča pri symetrickej kryptografii, takže vhodnou kombináciou týchto dvoch prístupov vieme šifrovať väčšie objemy dát [21].

Systémy asymetrickej kryptografie sa delia do troch hlavných kategórií a to na:

- asymetrické kryptosystémy typu IF,
- asymetrické kryptosystémy typu DL,
- asymetrické kryptosystémy typu EC.

2.3.1 Asymetrické kryptosystémy typu IF

Asymetrické kryptosystémy typu IF (Integer Factorization) predstavujú kryptosystémy, ktoré sú založené na obtiažnosti riešenia problému faktorizácie veľkých čísel. Ak máme dve rôzne a dostatočne veľké prvočísla p a q, vieme celkom ľahko vypočítať ich súčin, čiže $n = p \cdot q$. Pokiaľ ale disponujeme iba číslon n, potom je takmer nemožné nájsť čísla p a q, z ktorých sa toto číslo skladá [8].

Kryptosystém RSA sa dá považovať za najrozšírenejšieho reprezentanta zo skupiny asymetrických kryptosystémov typu IF. Názov kryptosystému RSA predstavuje počiatočné písmená vynálezcov tohto kryptosystému, čiže Rivest, Shamir a Adleman, ktorý ho v roku 1978 publikovali. Tento kryptosystém môže byť použitý pre podpisovanie a taktiež môže byť použitý pre šifrovanie [8].

2.3.2 Asymetrické kryptosystémy typu DL

Tak ako aj kryptosystémy typu IF, tak aj kryptosystémy typu DL (Discrete Logarithm), sú založené na podobnej problematike. Bezpečnosť týchto kryptosystémov je založená na obtiažnosti riešenia problémov dikrétneho logaritmu. Ak disponujeme dvomi celými kladnými číslami x, g a veľkým prvočíslom p, tak je pomerne jednoduché vypočítať mocninu $y = g^x \bmod p$. Pokiaľ poznáme čísla y, g a p z uvedenej rovnice, tak je skoro nemožné nájdenie exponentu x. Parameter x musí byť súkromný a parametre y, g a p môžu byť verejne známymi [8].

Asymetrické kryptosystémy typu DL majú využitie hlavne v oblasti podpisov a prípadne sa využívajú v oblasti ustanovenia kľúča. V oblasti šifrovania neni tento typ veľmi využívaný, pretože dĺžka kryptogramu je oproti dĺžke správy približne dvojnásobná. Predstaviteľom tohto typu kryptosystému je El-Gamalov algoritmus [8].

2.3.3 Asymetrické kryptosystémy typu EC

Asymetrické kryptosystémy typu EC (Elliptic Curve) by sa dali definovať ako kryptosystémy, ktorých bezpečnosť záleží na obtiažnosti riešenia problému diskrétneho logaritmu na eliptickej krivke. Ak disponujeme celým kladným číslom u a bodom G na eliptickej krivke, tak z neho vieme celkom ľahko vypočítať bod $U = u \cdot G$. Ak disponujeme iba bodmi G a U, tak nájdenie hodnoty čísla u z danej rovnice je

pre vhodne zvolenú eliptickú krivku v obecnom prípade skoro nemožné. Parameter u musí byť súkromný a parametre eliptickej krivky a body G a U sú verejne známe [8].

Ako aj kryptosytémy typu DL tak aj kryptosystémy tohto typu sa hlavne využívajú v oblasti podpisovania akebo prípadne pre zostrojenie kľúča. Taktiež sa tieto dva typy kryptosystémov podobajú principiálne, pretože obe využívajú takzvané cyklické grupy. Tieto kryptosystémy sa navzájom od seba odlišujú tým, že DL kryptosystémy využívajú multiplikatívnu grupu na množine celých čísel a EC kryptosystémy využívajú aditívnu grupu bodov eliptickej krivky. Algoritmy, ktoré využívajú tento typ kryptosystému sú napríklad ECDH alebo ECDSA [8].

3 Prenos dát pomocou Bluetooth

3.1 Technológia Bluetooth

Technológia Bluetooth by sa dala definovať ako rádiofrekvenčná bezdrôtová technológia krátkeho dosahu, ktorá slúži pre bezdrôtovú komunikáciu medzi rôznymi zariadeniami. Táto technológia umožňuje komunikovať zariadeniam ako napríklad počítače, telefóny alebo smart hodinky bez nutnosti pripojenia sa káblom medzi danými zariadeniami. Frekvenčné pásmo, ktoré technológia bletooth využíva je v rozmedzí od 2400 MHz do 2483,5 MHz [24].

Pomenovanie Bluetooth je inšpirované menom dánskeho kráľa Haralda Blaotland, ktorý žil v roku 908 nášho letopočtu. Technológia Bluetooth bola vyvinutá konzorciom pod vedením spoločnosti Ericsson a na jej vývoji sa ďalej podieľali spoločnosti ako napríklad Toshiba, IBM, Nokia alebo Intel [37, 24].

Pre komunikáciu pomocou technológie Bluetooth je nutné mať na zariadení takzvaný bluetooth vysielač. Taktiež sú tu podporované vysielacie výkony od –20 dBm (0,01 mW) do 20 dBm (100 mW). Ďalej technológia Bluetooth špecifikuje, že zariadenia, ktoré túto technológiu využívajú musia byť schopné dosiahnuť minimálnu citlivosť príjmača od –70 dBm až po –82 dBm, ale zvyčajne dosahujú oveľa vyššie citlivosti príjmača a to –95 dBm alebo lepšie [37, 24].

Komunikácia medzi zariadeniami pomocou Bluetooth býva zvyčajne typu peer-to-peer, čo znamená, že sa každé zariadenie v komunikácii považuje za rovnocenné. Pri komunikácii pomocou tejto technológie sa používa termín pikonet, ktorý znamená, že sú dve alebo viacero zariadení prepojené do malej ad hoc siete. V tomto type sieti je jedno zo zariadení ako hlavné („Master“) zariadenie a zvyšné zariadenia ako podriadené („Slaves“). V pikonet sieťach môžu byť pripojené minimálne dve zariadenia a maximálne až osem. V jednej pikonet sieti môže byť maximálne jedno hlavné zariadenie. Je to hlavne z toho dôvodu, že technológia Bluetooth podporuje typy spojenia point-to-point a point-to-multipoint. Taktiež je tu možnosť prepojiť medzi sebou dva alebo viacero pikonetov, čo vytvorí rozptylovú sieť a každé hlavné zariadenie z jednotlivých pikonetov funguje ako most medzi týmito pikonetmi [37, 24].

Ako spoločné rozhranie medzi zariadením ktoré využíva Bluetooth a jadrom Bluetooth je Host Controller Interface (HCI). Host Controller Interface zabezpečuje aby boli rôzne hardvérové implementácie medzi sebou kompatibilné. Na prepojenie služieb základného pásma s protokolmi vyššej úrovne slúži Logical Link Control and Adaptation Protocol (L2CAP). Protokolmi vyššej úrovne sa myslia protokoly ako napríklad Service Discovery Protocol (SDP), Radio Frequency Communication (RFCOMM) alebo Telephony Control Protocol (TCS) [37, 24].

Technológia Bluetooth taktiež zahŕňa množstvo bezpečnostných opatrení. Napríklad pri iniciovaní komunikácie medzi dvomi zariadeniami, prechádzajú tieto zariadenia procesom nazývaným párovanie. Pri tomto procese nastáva výmena kľúčov a je tu zaručené, že si dané zariadenia môžu dôverovať. Tieto klúče umožňujú chrániť prenášané dáta napríklad šifrovaním, čím sa docieli to, že nebudú čitateľné pre ostatné zariadenia. K niektorým zariadeniam sa dá pripojiť iba s pomocou znalosti tajnej frázy alebo kódu [17].

3.2 Bluetooth Low Energy

Bluetooth Low Energy, je štandard pre bezdrôtovú osobnú sieť s malou spotrebou energie ktorý vyšiel v roku 2010. Hlavným cieľom tohto štandardu je prepájať zariadenia na relatívne krátku vzdialenosť. Pri vývoji BLE bolo zohľadňované aj jeho možné využitie v oblasti IoT, čo určite má konkrétne dôsledky na návrh tohto štandardu. Napríklad zariadenia IoT sú väčšinou umiestňované na odlahlé miesta a nemajú stály prístup elektrickej energie, takže BLE využíva menšie množstvo energie [2, 5].

Tab. 3.1: Rozdiely BLE a klasického Bluetooth

	Klasické Bluetooth	**BLE**
komunikácia	kontinuálna, obojsmernná	málo dát jedným smerom
dosah	100 m	$<$ 100 m
spotreba energie	1 W	0,01-0,50W
prenosová rýchlosť	1-3 Mbit/s	125kbit/s - 2 Mbit/s
latencia	100 ms	6 ms
prenos zvuku	áno	nie

Klasický Bluetooth a BLE sú si navzájom podobné ale navzájom nekompatibilné protokoly. Komunikácia pomocou BLE narozdiel od klasického Bluetooth prebieha v malých dávkach, ktoré sú posielané iba raz za čas. Typická komunikácia pomocou BLE prebieha, tak že sa pravidelne zapína vysielač, ktorý prenáša alebo príjma niekoľko bajtov alebo kilokajtov dát a potom sa vracia do takzvaného režimu spánku. Pri klasickom Bluetooth sa pripojenie udržiava dlhšiu dobu, z dôvodu zabezpečenia nižšej latencie pri prenose a prenášajú sa naraz väčšie objemy dát [2, 5].

Generic Access Profile

Generic Access Profile slúži na riadenie pripojenia a oznamovanie v technológii Bluetooth. Hlavnou myšlienkou GAP je zviditeľňovanie zariadení pre ostatné zariadenia a taktiež určuje ako môžu zariadenia navzájom interagovať [35].

GAP definuje dve hlavné skupiny zariadení a tými sú Centrálne zariadenia a Periférne zariadenia. Pod pojmom centrálne zariadenie sa rozumie zvyčajne mobilný telefón alebo tablet, ktorý sa pripája k perifémemu zariadeniu a využíva väčšie množstvo výpočetných a pamäťových prostriedkov. Periférne zariadenie je definované ako malé zariadenie zvyčajne s malým výkonom a obmedzenými zdrojmi, ktoré sa pripája k výkonnejšiemu centrálnemu zariadeniu [35].

Oznamovanie pomocou GAP môže prebiehať pomocou dvoch spôsobov a to pomocou spôsobu Advertising data payload a Scan Response payload. Oba spôsoby sú identické a správy vysielané pomocou týchto spôsobov môžu obsahovať až 31 bajtov dát. Povinným spôsobom je iba Advertising data payload, pretože vysielaný obsah dáva informáciu o existencii daného zariadenia centrálnym zariadeniam v okolí. Scan Response payload je voliteľný spôsob, o ktorý môžu centrálne zariadenia požiadať. Táto možnosť umožňuje dizajnérom zariadení vložiť do oznamovaného obsahu viacero informácií, ako napríklad názov zariadenia [35].

Generic Attribute Profile

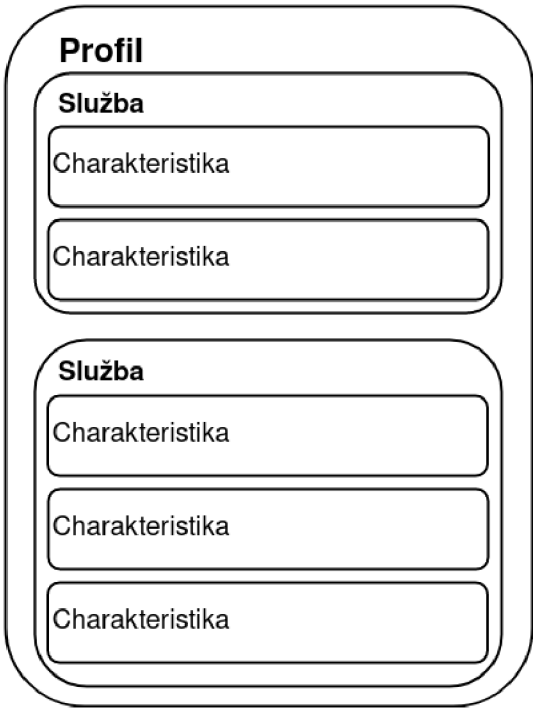
Generic Attiribute Profile definuje spôsob akým si BLE zariadenia vymieňajú navzájom dáta. Nasadenie GATT nastáva vtedy keď sa vytvorí spojenie medzi dvomi zariadeniami, ktoré prešli oznamovacím procesom riadeným pomocou GAP. BLE zariadenia obsahujú databázu známu ako tabuľka atribútov. V tejto tabuľke sa nachádzajú GATT služby, charakteristiky a štrukturálne detaily a hodnoty deskriptorov. Táto tabuľka je klúčová pre fungovanie zariadení BLE založených na GATT. Položky tejto tabuľky sú identifikované pomocou atribučných identifikátorov, ktoré majú veľkosť 16 bitov [36, 38].

Pri využívaní GATT môže byť periférne zariadenie pripojené iba k jednému centrálnemu zariadeniu. Periférne zariadenie prestane po pripojení oznamovať svoju existenciu ostatným zariadeniam počas toho kým je pripojené k jednému centrálnemu zariadeniu [36, 38].

Periférne zariadenie je taktiež označované ako GATT Server a ukladá definície služieb a charakteristík. Centrálne zariadenie je označované ako GATT Client a posiela žiadosti na GATT Server. GATT Client je inicializátorom tranzakcií a príjma odpovede, ktoré prichádzajú zo strany GATT Servera [36, 38].

Tranzakcie GATT sa skladajú z vnorených objektoch nazývaných Profily, Služby a Charakteristiky. Profil je preddefinovaná kolekcia služieb vytvorená spoločnosťou

Bluetooth SIG alebo dizajnérom periférnych zariadení. Služby delia dáta na logické časti a skladajú sa zo špecifických častí nazývaných charakteristiky. Služby sa od seba odlišujú od jedinečného číselného ID, ktoré má veľkosť 16 alebo 128 bitov. Charakteristiky predstavujú dátovú jednotku, ktoré môžu ukladať rôzne parametre. Taktiež ako služby, tak aj charakteristiky sa navzájom od seba odlišujú pomocou jedinečného číselného ID, taktiež označovaného ako UUID [36, 38].



Obr. 3.1: Štruktúra GATT profilu [36].

4 Návrh mobilnej aplikácie

Pred samotným krokom programovania mobilnej aplikácie je nutné pristúpiť k jej návrhu. Pri návrhu mobilnej aplikácie sa špecifikuje účel danej aplikácie, požiadavky aplikácie, entity ktoré sa tu budú nachádzať, dáta ktoré sa budú spracovávať a ukladať a technológie s ktorými bude vývojár pracovať. Pri špecifikácii podbodov návrhu mobilnej aplikácie vieme docieliť to, že sa zjednoduší nasledujúci krok programovania a tým sa docieli následná úspora času.

4.1 Model prípadov použitia

Nielen pri návrhu mobilných aplikácií ale aj všeobecne pri návrhu akéhokoľvek softvérového systému sa využívajú takzvané UML Use Case diagramy. Tieto UML Use Case diagramy vyjadrujú možné interakcie užívateľa s danou aplikáciou a taktiež pomáhajú lepšie identifikovať funkčné požiadavky systému.

V mobilnej aplikácii, ktorou sa zaoberá tento projekt vystupuje entita užívateľa, alebo lepšie povedané zamestnanca firmy, ktorá disponuje flotilou vozidiel. V prvom kroku je nutné zistiť, že s akými prípadmi použitia by sa daný užívateľ mohol stretnúť. Základnými operáciami by pre daného užívateľa mohli byť operácie typu prehľadávanie zoznamu vozidiel, vytvorenie a zrušenie rezervácie alebo manipulácia s vozidlom.

Prvým prípadom použitia, s ktorým by sa vedel užívateľ stretnúť je **prehľadávanie zoznamu vozidiel.** Užívateľ by mal po prihlásení sa do aplikácie možnosť manuálne prehľadávať zoznam vozidiel, v ktorom by sa nachádzali rôzne informácie o danom vozidle.

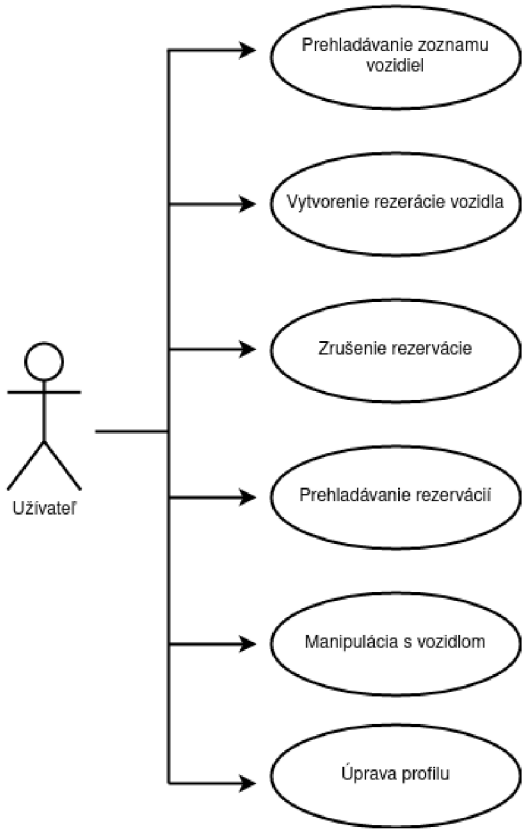
Druhým prípadom použitia by mohlo byť **vytvorenie rezervácie pre dané vozidlo.** Pri vytváraní rezervácie by si užívateľ vybral daný termín, v ktorom by chcel mať vybrané vozidlo zarezervované. Taktiež by mal možnosť zobrazenia už zarezervovaných termínov aby sa vyhlo prípadnému prekrytiu s inými rezerváciami a aby vedel v akom termíne sú vozidlá nedostupné.

Ako tretí prípad použitia by sa dalo definovať **zrušenie rezervácie pre dané vozidlo.** Pomocou tejto funkcionality by mal užívateľ možnosť zrušiť rezerváciu vozidla v danom termíne a tým by sa uvoľnila dostupnosť vozidla pre ďalších užívateľov.

Taktiež ako prehľadávanie zoznamu vozidiel tak by mal mať užívateľ možnosť **prehľadávania svojich rezervácií.** Tento zoznam by zobrazoval vopred zoradené užívateľom vytvorené rezervácie. Z tohto zoznamu by sa užívateľ dozvedel nielen informácie o vozidle, ale aj informáciu o čase rezervácie vozidla.

Nasledujúcim prípadom použitia po prehľadávaní rezervácií, by mohla byť **manipulácia s vozidlom.** Pod pojmom manipulácia s vozidlom sa rozumie, že si užívateľ bude vedieť vozidlo odomknúť pomocou mobilnej aplikácie s využitím technológie BLE.

Posledným prípadom použitia by mohla byť **úprava užívateľského profilu.** Užívateľ by mal možnosť editovania rôzných údajov. Vyššie spomenuté prípady použitia sú graficky znázornené na obr. 4.1 .



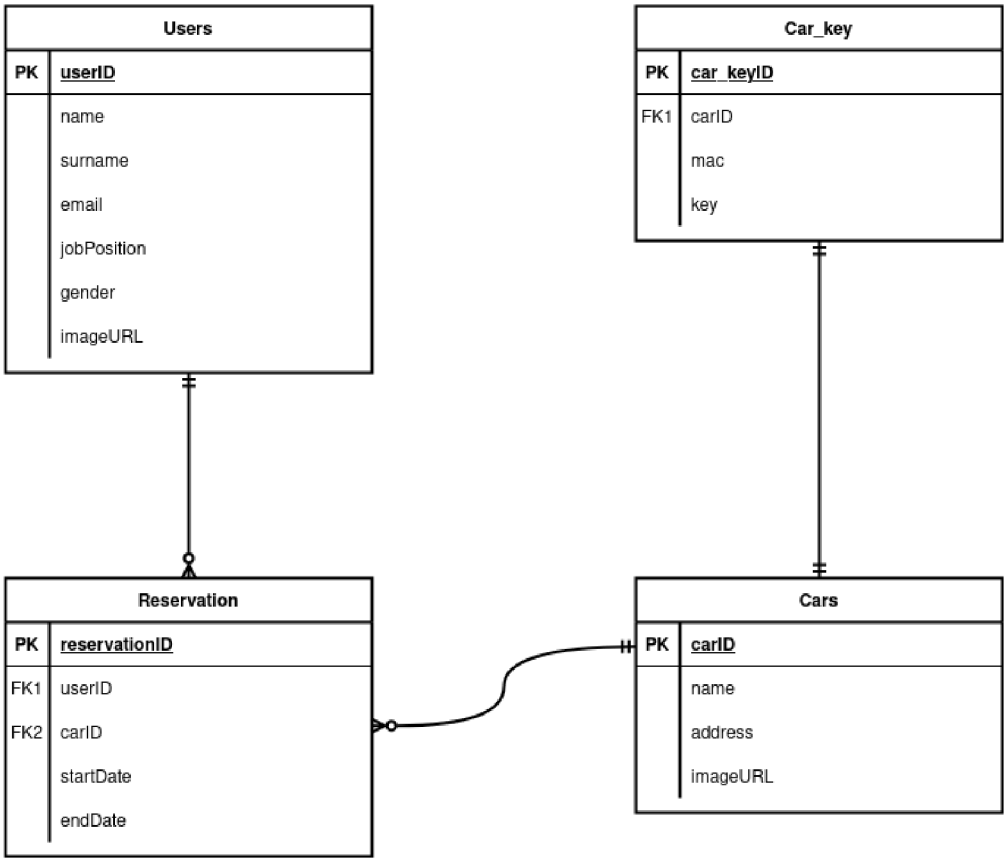
Obr. 4.1: Model prípadov použitia mobilnej aplikácie.

4.2 Dátový model

Mobilná aplikácia pracuje s rôznymi dátami, ktoré je nutné uchovať. Pre správne uchovanie týchto dát slúži databáza. V tomto prípade je nutné navrhnúť databázu, ktorá by bola schopná uchovávať štrukturované dáta, čo by umožnilo efektívne vyhľadávanie, aktualizáciu a získavanie informácií z aplikácie.

Na obr. 4.2 je zobrazený ER diagram, ktorý graficky znázorňuje návrh databáze vhodnej pre použitie v rámci tohto projektu, ktorú si v nasledujúcej časti opíšeme.

V tejto databáze je uchovávaných viacero entít, ktoré vystupujú v rámci mobilnej aplikácie. Týmito entitami sú užívateľ, vozidlo, rezervácia a kľúč k autu.



Obr. 4.2: ER diagram databáze.

Podľa tohto návrhu by sa entita užívateľa ukladala do tabuľky **Users.** Tu by sa ukladali základné údaje o užívateľovi, ako napríklad jeho meno a priezvisko, email, pracovná pozícia, pohlavie a URL adresa profilovej fotky užívateľa. Každý užívateľ by mal taktiež jedinečný identifikátor.

Informácie o vozidlách by sa ukladali do tabuľky **Cars.** Do tejto tabuľky by sa ukladali údaje o vozidlách, ako napríklad model vozidla, adresa miesta, kde sa vozidlo momentálne nachádza, URL adresa obrázka daného modelu vozidla a jedinečný identifikátor pre dané vozidlo.

Tabuľka **Reservation** má za úlohu uložiť informácie o rezerváciach. Táto tabuľka by uchovávala identifikátory užívateľa a vozidla pre spoľahlivé priradenie užívateľa a vozidla k danej rezervácii. Ďalej by sa tu nachádzali atribúty, ktoré popisujú začiatok a koniec rezervácie a taktiež jedinečný identifikátor, tak ako aj pri ostatných entitách.

Kľúč k danému vozidlu by sa ukladal do tabuľky **Car key.** V tejto tabuľke by

bol uložený jedinečný identifikátor užívateľa pre priradenie kľúču k vozidlu a tatiež by tu bola uložená fyzická adresa pre dané bluetooth zariadenie a samotný kľúč.

Tabuľka Reservation predstavuje entitno-vzťahovú množninu, kde pomocou cudzích kľúčov userID a carID prepája užívateľa s vozidlom. Užívateľovi by bolo umožnené vytvoriť viacero rezervácií a entita vozidla by sa mohla vyskytovať vo viacerých rezerváciach. Pre tabuľku Car key bol zvolený vzťah 1:1, z toho dôvodu, že záznam ukladaný v tejto tabuľke je práve spojený iba s jedným vozidlom.

4.3 Návrh užívateľského rozhrania

Užívateľské rozhranie by sa dalo definovať ako súbor prvkov a funkcií, ktoré umožňujú užívateľovi komunikovať s počitačovým programom alebo systémom. Je to prostredie, v ktorom sa odohráva interakcia medzi užívateľom a technológiou.

Pri návrhu užívateľského rozhrania pre aplikáciu bol kladený dôraz na intuitívne usporiadanie jednotlivých prvkov. Tento dôraz bol kladený z toho dôvodu, aby sa užívateľ mohol jednoducho a bez námahy orientovať v prostredí a intuitívne vykonávať požadované úkony. Užívateľské rozhranie pre mobilnú aplikáciu bolo navrhnuté pomocou webovej aplikácie Figma.

Prihlasovací a registračný formulár

Prihlasovací a registračný formulár sa považujú za dôležité prvky užívateľského rozhrania. Tieto prvky umožňujú vstúpiť užívateľom do systému alebo sa pomocou nich zaregistrovať.

Aplikácia je navrhnutá tak, aby sa neprihlásenému užívateľovi po prvotnom spustení zobrazil prihlasovací formulár. Tento formulár, zobrazený na obr. 4.3 sa skladá z dvoch textových polí a dvoch tlačítok. Textové polia slúžia na vloženie užívateľových údajov, ako je email a heslo a pomocou nich sa vie prihlásiť do aplikácie. Ak užívateľ zadá správne prihlasovacie údaje tak sa môže pomocou tlačítka **Sign in** presunúť na ďalšiu stránku. Ak užívateľ ešte nemá vytvorený účet tak sa vie pomocou tlačítka **Register now** presunúť na registračný formulár.

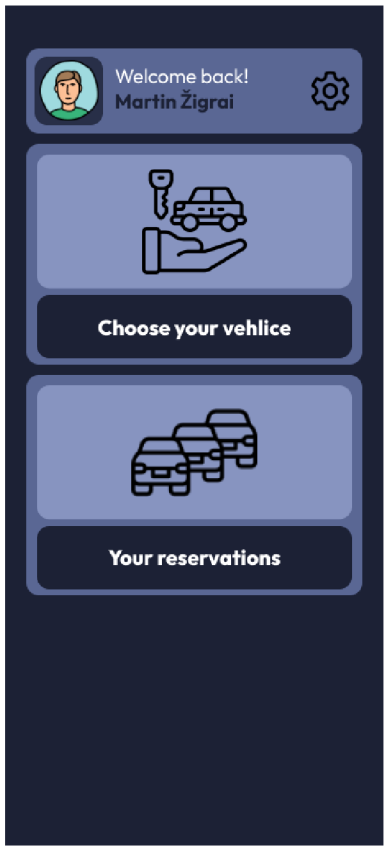
Registračný formulár, znázornený na obr. 4.3 sa skladá zo šiestich textových polí a piatich tlačítok. Užívateľ pri registrácii bude musieť zadať do textových polí svoje údaje, ako napríklad email, meno, priezvisko, pracovnú pozíciu, heslo a potvrdenie hesla. Tri tlačítka **Male**, **Female** a **Other** znázorňujú výber pohlavia. Ak užívateľ správne vyplní registračné údaje tak sa pomocou tlačítka **Register** bude môcť zaregistrovať. Ak užívavateľ bude mať účet vytvorený tak sa pomocou tlačítka **return to the login page** vráti naspäť na prilasovací formulár.

Obr. 4.3: Prihlasovací a registračný formulár.

Hlavné menu

Po úspešnom prihlásení užívateľa do aplikácie bude zobrazené hlavné menu, ktoré je zobrazené na obr. 4.4 . Hlavné menu bolo navrhnuté tak aby sa delilo na tri časti.

- **Profilová lišta** zobrazuje profilovú fotku užívateľa a jeho názov, respektíve jeho meno a priezvisko. Taktiež sa tu nachádza tlačítko, ktoré slúži na presmerovanie užívateľa do nastavení.
- **Výber vozidla** je časť hlavného menu, ktorá po kliknutí na tlačítko presmeruje užívateľa na zoznam dostupných vozidiel.
- **Rezervácie užívateľa** je posledná časť hlavného menu, pomocou ktorej sa vie užívateľ dostať na zoznam jeho rezervácií.



Obr. 4.4: Hlavné menu.

Výber vozidla

Ďalšou dôležitou súčasťou mobilnej aplikácie, ktorá slúži na rezerváciu vozidiel by mal byť prehľad zoznamu vozidiel. Z tohto dôvodu bolo nutné zakomponovať tento prehľad aj do návrhu aplikácie, ktorou sa zaoberá tento projekt.

Ako je zobrazené na obrázku 4.5, tak sa tento zoznam skladá z viacerých kariet. Každá karta obsahuje fotografiu vozidla, informácie o vozidle a tlačítko. Užívateľ z jednotlivých kariet môže vyčítať jednotlivé informácie, ako názov vozidla a adresa kde sa vozidlo nachádza. Ďalší prvok, tlačítko, slúži na presmerovanie užívateľa na výber termínov pre rezerváciu daného vozidla.

Na obrázku 4.5 sa taktiež nachádza návrh stránky, ktorá slúži na výber termínu pre rezerváciu. Táto stránka bola navrhnutá tak aby užívateľ vedel, že ktoré vozidlo si chce zarezervovať a z tohto dôvodu bol pridaný názov vozidla do vrchnej časti stránky. Ďalšou časťou je kalendár, ktorý zobrazuje dostupné a nedostupné termíny rezervácie. A poslednou časťou je tlačítko, ktoré vykoná danú rezerváciu.

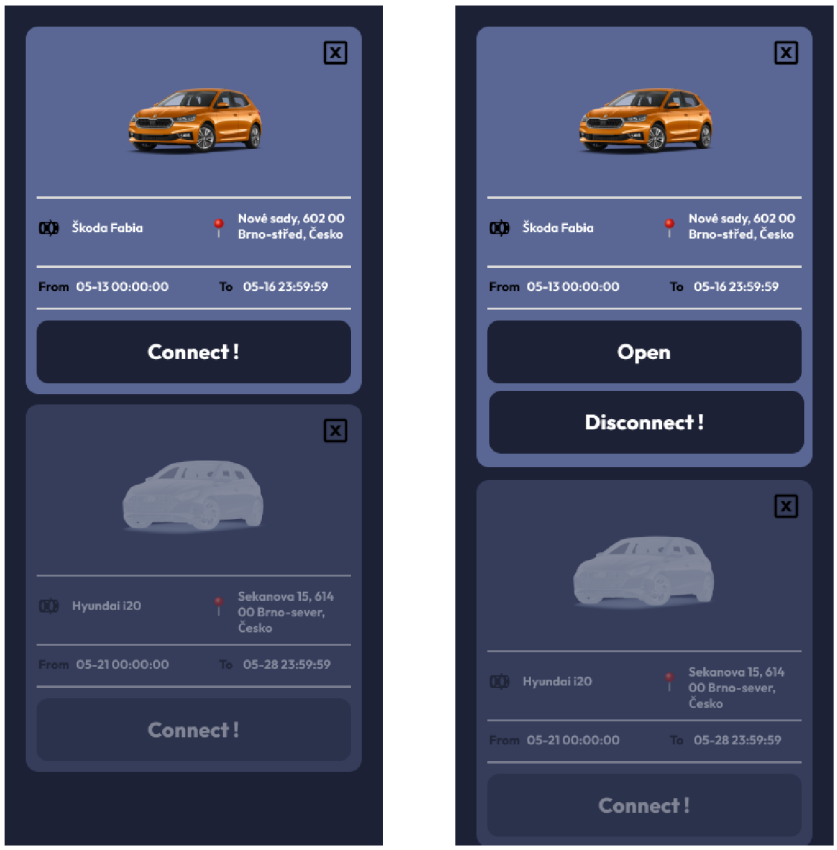
Obr. 4.5: Prehľad zoznamu vozidiel a výber termínov pre rezerváciu.

Rezervácie užívateľa

Poslednou a asi aj najhlavnejšou súčasťou mobilnej aplikácie je prehľad zoznamu rezervácií užívateľa. V tejto časti bude prebiehať interakcia medzi vozidlom a užívateľom a z tohto dôvodu sa tak aj pristupovalo pri návrhu tejto stránky.

Ako môžeme vidieť na obr. 4.6, tak sa táto stránka skladá z viacerých kariet, v konečnom dôsledku bude počet kariet závisieť od počtu rezervácií. Jedna karta predstavuje aktuálnu rezerváciu a ďalšia rezerváciu, ktorá je neaktuálna. Rezerváciu bude možné zrušiť pomocou tlačítka v pravom hornom rohu. Podobne ako v predošlích návrhoch, tak aj tu sa nachádzajú jednotlivé informácie o vozidle a ešte sa tu nachádzajú časové údaje o začiatku a konci rezervácie. Pomocou tlačítka **Connect** sa bude môcť užívateľ pripojiť k vozidlu pomocou technológie BLE.

Druhá časť obrázku 4.6 zobrazuje stav karty po pripojení sa k vozidlu. Po pripojení sa tlačítko **Connect** schová a zobrazia sa dve nové a to tlačítka **Open** a **Disconnect**.



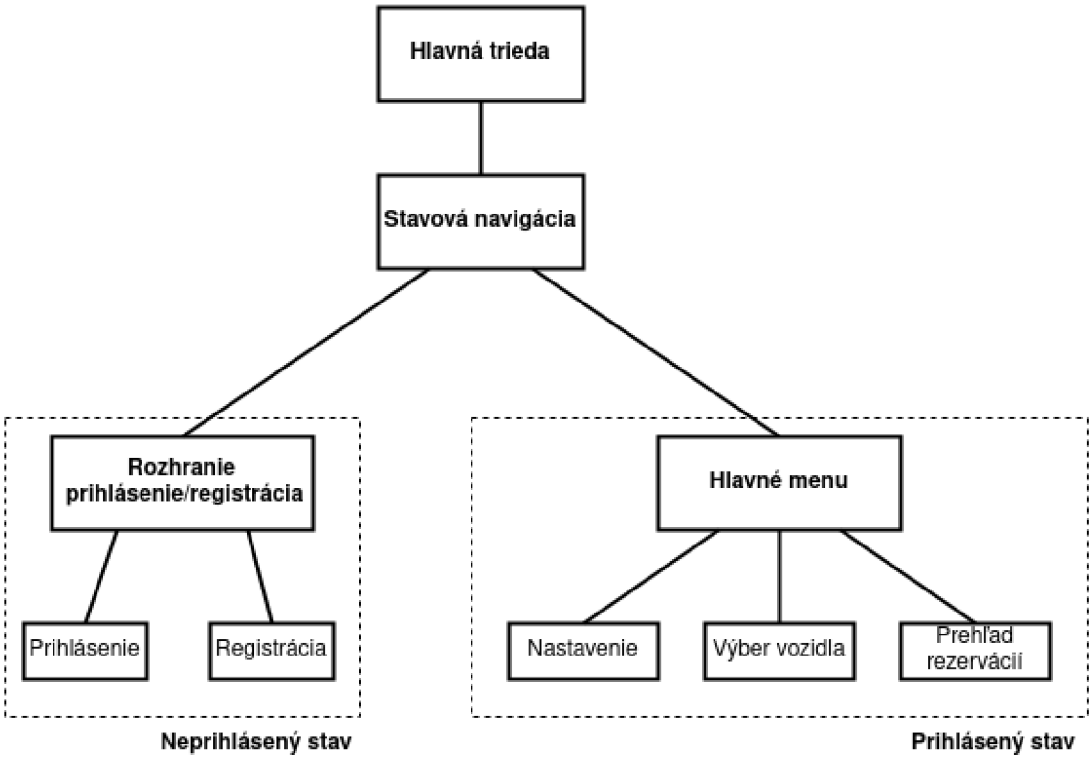
Obr. 4.6: Prehľad zoznamu užívateľovych rezervácií.

4.4 Štruktúra mobilnej aplikácie

Štruktúrou aplikácie sa myslí organizačná schéma, ktorá popisuje jednotlivé časti aplikácie a vzťahy medzi nimi. Definuje ako sú jednotlivé stránky alebo widgety usporiadané.

Na obr. 4.7 je znázornená štruktúra mobilnej aplikácie, ktorou sa zaoberá tento projekt. Táto štruktúra znázorňuje hlavnú triedu, ktorá je vstupným bodom aplikácie, kde sa inicializuje a spúšťa táto aplikácia. Časť nazvaná ako stavová navigácia je zodpovedná za sledovanie stavu prihlásenia užívateľa. Ak užívateľ nie je prihlásený tak má prístup k rozhraniu pre pihlásenie alebo registráciu. V tejto časti sa môže pomocou prihlasovacieho a registračného formuláru, prihlásiť alebo zaregistrovať.

Ak je užívateľ prihlásený do aplikácie, tak je presmerovaný do hlavného menu. Z hlavného menu má prístup k jednotlivým stánkam, ako sú nastavenie, výber vozidla alebo prehľad rezervácií.



Obr. 4.7: Štruktúra mobilnej aplikácie.

5 Kryptografické knižnice vo Flutteri

5.1 Prehľad knižníc

Pre open-source framework Flutter existuje množstvo knižníc, ktoré sa dajú využiť v rôznych projektoch. V týchto knižniciach sú zakomponované funkcie, ktoré umožnujú vývojárom rýchlo a jednoducho implementovať určité funkcionality do aplikácií, ako napríklad animácie, navigácia alebo podpora pre rôzne zariadenia a operačné systémy. Tieto funkcie môžu ušetriť vývojárom mnoho času a úsilia a pomocou nich vieme vytvárať kvalitnejšie aplikácie. Väčšina z týchto knižníc je dostupná na webstránke `https://www.pub.dev/`.

Vo frameworku Flutter je taktiež možné využívať knižnice z iných programovacích jazykov. Toto je možné s využitím Dart Foreign Function Interface, ktorý nám umožňuje volať natívne funkcie C/C++ na rôznych platformách.

V nasledujúcej časti sú popísané knižnice, ktoré obsahujú rôznorodé kryptografické primitíva, ktoré budú otestované a vyhodnotené. Knižnice s najlepšími výsledkami budú potom využité pri vývoji mobilnej aplikácie.

Knižnica Libgroupsig

Knižnica Libgroupsig je open-source kryptografická knižnica pre tvorbu a správu skupinových podpisov. Táto knižnica je napísaná v jazyku C a poskytuje funkcie pre generovanie a verifikáciu skupinových podpisov, taktiež aj pre správu identít členov skupiny. Používa sa pri vývoji aplikácií, ktoré vyžadujú podpisovanie dokumentov skupinov ľudí, ako napríklad pre elektronické hashovanie alebo pre správu dokumentov v kolektívnom prostredí [20].

Knižnica Libgroupsig imlementuje schémy z oblasti skupinových podpisov, ako napríklad [20]:

- DL21,
- KLAP20,
- GL19,
- PS16 alebo
- BBS04.

Knižnica Crypto

Knižnica Crypto je kryptografická knižnica napísaná v programovacom jazyku Dart. Túto knižnicu možno využiť v rámci aplikácie vytvorenej vo Flutteri. Knižnica poskytuje hlavne hashovacie funkcie, ako napríklad SHA-1, SHA-256 alebo hashovaciu funkciu MD5 [12].

Knižnica Cryptography

Podobne ako knižnica Crypto, tak aj knižnica Cryptography je napísaná v jazyku Dart a je určená pre aplikácie vyvíjaných v Darte a Flutteri. Táto knižnica je považovaná za ľahko použiteľnú a rýchlu. Táto knižnica poskytuje funkcie, ako napríklad AES, RSA, SHA alebo Ed25519 [14].

Knižnica Encrypt

Encrypt je knižnica, ktorá poskytuje vysokoúrovňové API pre knižnicu pointycastle. Táto knižnica poskytuje funkcie pre šifrovanie dešifrovanie a podpis, ako napríklad AES a RSA [10].

Knižnica Ninja

Knižnica Ninja predstavuje kryptografickú knižnicu napísanú v programovacom jazyku Dart. Poskytuje funkcie pre šifrovanie, dešifrovanie, podpis a overenia správy. Funkcie, ktoré táto knižnica obsahuje sú hlavne AES a RSA [31].

Knižnica Pointycastle

Pointycastle je knižnica pre šifrovanie a dešifrovanie, ktorá vychádza z knižnice Bouncycastle pre Javu. Táto knižnica implementuje veľkú sadu algoritmov. Implementuje napríklad hashovacie funkcie SHA, algoritmy symetrickej kryptografie AES alebo algoritmy asymetrickej kryptografie RSA a ECDSA [6].

5.2 Proces testovania

Pre zabezpečenie čo najlepšieho uživateľského zážitku je nutné dosiahnuť čo najlepšiu mieru, v akej aplikácia reaguje na požiadavky a vykonávané úlohy. Tento projekt obsahuje značné množstvo kryptografických funkcií, ktoré sú implementované v rôznych knižniciach. Tieto knižnice majú odlišné časy vykonávania jednotlivých funkcií a preto je nutné tieto knižnice porovnať a vyhodnotiť, ktorá z nich je najlepšia z časového hľadiska.

Tieto parametre boli testované pre zariadenia s operačnými systémami Android a iOS, konkrétne pre Xiaomi Redmi 9 pro a Iphone 12 mini. Na testovanie bola použitá knižnica Benchmark harness, ktorá obsahuje funkcie, ktoré umožňujú testovať čas vykonávania jednotlivých kryptografických funkcií. Meranie pomocou tejto knižnice prebieha tak, že sa daná funkcia volá desaťkrát a následne sú tieto časy vykonávania spriemerované a dostávame jeden výsledný čas.

Pri meraniach jednotlivých fukncií, bola spracovávaná rovnaká správa o veľkosti šestnásť znakov. Týmto sa nastavila rovnaká obtiažnosť pre každú jednu funkciu. Každá funkcia bola zmeraná desaťkrát, zmerané časy boli spriemerované a výsledný priemerný čas bol zapísaný do tabuľky.

Časy vykonávania sú merané pre funkcie, ako napríklad SHA-1, AES, RSA, ECDSA alebo Ed25519. Nie každá knižnica obsahuje rovnaké počty implementácií týchto funkcií, takže sú tieto výsledky porovnávané samostatne podľa jednotlivých funkcií a v rámci knižníc. Výsledné hodnoty sú merané v jednotkách µs, ms, s a sú zaznamenané v nasledujúcich tabuľkách.

Tab. 5.1: Časy vykonávania hashovacích funkcií SHA zmerané na OS Android

	Pointycastle	**Crypto**	**Cryptography**
	Čas vykonávania $[\mu s]$	**Čas vykonávania** $[\mu s]$	**Čas vykonávania** $[\mu s]$
SHA-1	118,03	28,35	226578
SHA-224	276,02	38,17	226550
SHA-256	276,14	39,62	226542
SHA-384	425,9	48,92	226535
SHA-512	421,09	50,22	226528
SHA3-224	2250,84	-	-
SHA3-256	2245,7	-	-
SHA3-384	2276,07	-	-
SHA3-512	2236,31	-	-

Tab. 5.2: Časy vykonávania hashovacích funkcií SHA zmerané na OS iOS

	Pointycastle	**Crypto**	**Cryptography**
	Čas vykonávania $[\mu s]$	**Čas vykonávania** $[\mu s]$	**Čas vykonávania** $[\mu s]$
SHA-1	19,08	12,63	1259,32
SHA-224	40,56	15,21	1259,32
SHA-256	40,52	15,18	1259,32
SHA-384	288,15	31,79	1259,32
SHA-512	292,62	31,77	1259,32
SHA3-224	497,97	-	-
SHA3-256	509,7	-	-
SHA3-384	519,1	-	-
SHA3-512	513,89	-	-

Tab. 5.3: Časy vykonávania kryptografickej funkcie AES zmerané na OS Android

	Pointycastle	**Encrypt**	**Cryptography**
	Čas vykonávania $[\mu s]$	**Čas vykonávania** $[\mu s]$	**Čas vykonávania** $[\mu s]$
AES cbc	426,36	4945,50	1144787
AES ecb	1667,87	5287,3	-
AES gcm	7226,93	-	675581
AES ccm	2471,65	-	-

Tab. 5.4: Časy vykonávania kryptografickej funkcie AES zmerané na OS iOS

	Pointycastle	**Encrypt**	**Cryptography**
	Čas vykonávania $[\mu s]$	**Čas vykonávania** $[\mu s]$	**Čas vykonávania** $[\mu s]$
AES cbc	110,25	2352,19	11326,90
AES ecb	101,41	2377,25	-
AES gcm	447,33	-	3777,80
AES ccm	289,05	-	-

Tab. 5.5: Časy vykonávania kryptografickej funkcie RSA zmerané na OS Android

	Pointycastle	**Ninja**
	Čas vykonávania $[s]$	**Čas vykonávania** $[s]$
RSA 1024	3,55	16,04
RSA 2048	18,46	23,02
RSA 3072	80,70	135,35

Tab. 5.6: Časy vykonávania kryptografickej funkcie RSA zmerané na OS iOS

	Pointycastle	**Ninja**
	Čas vykonávania $[s]$	**Čas vykonávania** $[s]$
RSA 1024	0,41	0,44
RSA 2048	2,49	7,53
RSA 3072	8,67	12,83

Tab. 5.7: Časy vykonávania kryptografickej funkcie ECDSA zmerané na OS Android

	Pointycastle
	Čas vykonávania $[ms]$
ECDSA 192	762,03
ECDSA 224	1056,29
ECDSA 256	1277,50
ECDSA 384	2995,55
ECDSA 521	6457,34

Tab. 5.8: Časy vykonávania kryptografickej funkcie ECDSA zmerané na OS iOS

	Pointycastle
	Čas vykonávania $[ms]$
ECDSA 192	102,0
ECDSA 224	141,54
ECDSA 256	177,62
ECDSA 384	434,66
ECDSA 521	882,27

Kryptografická funkcia Ed25519 bola implementovaná knižnicou cryptography. Časy vykonávania tejto funkcie pre operačný systém Android bol 1277,64 ms. Čas vykonávania tejto funkcie pre operačný systém iOS bol 7,55 ms.

5.3 Vyhodnotenie nameraných výsledkov

Ako môžeme vidieť z jednotlivých tabuliek, tak sa časy vykonávania na jednotlivých operačných systémoch značne líšia. Táto skutočnosť môže byť spôsobená nižším výpočetným výkonom konkrétneho Android zariadenia. Taktiež môžeme vidieť, že sa líšia časy vykonávania jednotlivých funkcií.

Hashovacie funkcie SHA boli porovnávané v rámci knižníc Pointycastle, Crypto a Cryptography. Funkcie z knižnice Crypto dopadli z pohľadu času vykonávania najlepšie u oboch zariadení s operačnými systémami Android a iOS. Druhou v poradí bola knižnica Pointycastle a posledná knižnica Cryptography.

Kryptografická funkcia AES, bola meraná pre rôzne prevádzkové režimy a to konkrétne pre cbc, ecb, gcm a ccm. Nie každá knižnica obsahovala konkrétne prevádzkové režimy tejto šifry a z tohto dôvodu niektoré režimy nemožno porovnať. Najlepšie výsledky zo všetkých knižníc dosahuje knižnica Pointycastle u oboch operačných systémoch.

Pri kryptografickej funkcii RSA boli volené rôzne veľkosti kľúčov a to veľkosť 1024, 2048 a 3072 bitov. Táto funkcia s týmito konkrétnymi veľkosťami kľúčov bola porovnávaná v rámci knižníc Pointycastle a Ninja. Ako je zrejmé z tabuľky 5.5 a tabuľky 5.6, tak knižnica Pointycastle je rýchlejšia ako knižnica Ninja.

Ako vyplýva z tabuliek 5.7, 5.8 a z výpisu časových honôt pre funkciu Ed25519, tak nemajú byť s čím porovnané pretože zvyšné knižnice neobsahujú implementácie týchto funkcií.

6 Implementácia mobilnej aplikácie

Implementácia predstavuje proces prevodu konceptuálnych návrhov, myšlienok alebo plánov na funkčnú aplikáciu. Nasledujúca kapitola opisuje implementáciu jednotlivých častí, z ktorých sa aplikácia skladá.

6.1 Autentizácia pomocou Firebase

Aplikácia je navrhnutá tak, aby sa užívateľovi po prvotnom spustení zobrazil prihlasovací formulár. Z tohto tôvodu bolo nutné vyriešiť v prvom kroku autentizáciu užívateľa pre prístup k uživateľskému účtu.

Aby mobilná aplikácia správne komunikovala s Firebase, tak bolo nutné pridať konfiguračné súbory vygenerované pre jednotlivé operačné systémy Android a iOS. Tieto konfiguračné súbory obsahujú potrebné informácie, ako napríklad identifikátor projektu, kľúče a iné nastavenia, ktoré mobilná aplikácia potrebuje pre správnu komunikáciu s Firebase službami.

Firebase autentizačná služba poskytuje veľké množstvo spôsobov autentizácie, ako napríklad pomocou emailu, Google, Facebook alebo Apple účtu alebo anonyme. Pre tento projekt bola zvolená autentizácia pomocou emailu. Heslo užívateľa sa ukladá v zhashovanej forme, ktoré spracováva hashovacia funkcia Scrypt.

Na strane aplikácie bol vytvorený prihlasovací a registračný formulár. Pomocou nich bude užívateľ schopný posielať svoje prihlasovacie údaje pre prihlásenie a registračné údaje pre vytvorenie nového užívateľského účtu. Ďalej boli implementované metódy signInWithEmailAndPassword a createUserWithEmailAndPassword z knižnice Firebase_auth pre správne využívanie autentizačnej služby.

Výsledok tejto operácie sleduje stavová navigácia, ktorá je zobrazená na obr. 4.7. Tá sleduje či je pokus o prihlásenie alebo registráciu úspešný. Na základe úspešného výsledku presmeruje užívateľa na hlavné menu a na základe neúspešného ho vráti späť na prihlasovací alebo registračný formulár.

6.2 Práca s dátami

Mobilná aplikácia potrebuje pre svoju funkčnosť dáta. Aplikácia nepotrebuje dáta iba získavať ale v určitých prípadoch ich aj generuje. Tieto dáta je nutné ukladať, tak aby boli pre užívateľa čo najdostupnejšie. Pre zabezpečenie najlepšej dostupnosti je nutné ukladať tieto dáta v databáze.

Pre tento projekt bola zvolená databáza Cloud Firestore, čo je realtime Firebase databáza. Cloud Firestore je NoSQL databáza takže dáta nie sú ukladané do tabuliek ale do JSON objektov.

Databáza bola vytvorená podľa návrhu zobrazeného na obr. 4.2 . V Cloud Firestore boli vytvorené jednotlivé kolekcie, ktoré slúžia ako skupiny dokumentov a umožňujú užívateľovi štruktúrovať a hierarchicky usporiadavať jeho dáta. Dáta v dokumentoch závisia od kolekcie, v ktorej sú ukladané.

Na strane mobilnej aplikácie bolo nutné importovať knižnicu Cloud_firestore, ktorá obsahuje funkcie a metódy potrebné k získavaniu dát z Cloud Firestore. Dáta sú získavané pomocou metód vytvárajúcich stream, ktorý sleduje zmeny v konkrétnom dokumente alebo v celej kolekcii. Týmto spôsobom sa získavajú aktuálne dáta z databázy a tým pádom vie aplikácia reagovať na ich zmeny v reálnom čase. Metóda vytvárajúca stream, pre získanie užívateľových dát je zobrazená v ukážke 6.1 .

Výpis 6.1: Metóda vytvárajúca Stream

```
1 Stream<MyUser> get myUserData {
2     return userCollectionReference.doc(user.uid)
3     .snapshots()
4     .map(_myUserFromSnapshot);
5   }
```

Na tieto zmeny reaguje StreamProvider z knižnice Provider. Knižnica Provider obsahuje funkcie, ktoré slúžia na správu stavov jednotlivých widgetov. Ak StreamProvider zistí zmenu sledovaných dát tak automaticky obnoví daný widget, pre ktorý je nastavený.

Užívateľ generuje dáta hneď v niekoľkých prípadoch. Prvým prípadom generovania dát je registrácia. Pri procese registrácie užívateľ posiela svoje dáta do kolekcie Users, ktorá uchováva informácie o užívateľoch. Druhým prípadom je vytvorenie rezervácie vozidla. Užívateľ si vyberá termín, v ktorom chce mať dané vozidlo zarezervované a tento termín ukladá spolu s identifikátorom vozidla a užívateľa do kolekcie Reservation.

Užívateľ má možnosť mazania alebo úpravy dát z konkrétnych kolekcií. Keď sa užívateľ rozhodne, tak môže vymazať svoju konkrétnu rezerváciu. Alebo ak bude užívateľ chcieť tak má možnosť úpravy dát v kolekcii Users, kde si môže upraviť svoje údaje.

6.3 Komunikácia s GATT serverom

GATT server (podrobnejšie popísaný v sekcii 3.2) predstavuje BLE zariadenie, ktoré poskytuje a spracováva charakteristiky. Tento GATT server by sa dal označiť ako súčasť vozidla, pomocou ktorej by sa vedel užívateľ autentizovať do daného vozidla na základe správnych charakteristík.

Ako GATT server bol využitý už existujúci program napísaný v programovacom jazyku Java a vytvorený F. Texlom. Tento GATT server príjma, odosiela, uchováva a spracováva 6 charakteristík. Po úspešnom prijatí jednotlivých charakteristík prebehne na tomto serveri proces, ktorý rozhodne o tom, či bude užívateľovi umožený prístup k vozidlu alebo nie.

GATT server musel byť z dôvodu kompatibility aplikácie upravený. Bola pridaná možnosť zapísania charakteristiky bez odpovede. Táto možnosť bola pridaná pretože nie všetky charakteristiky posielajú odpoveď naspäť do mobilnej aplikácie a knižnica, ktorá je využívaná na strane mobilnej aplikácie by bez tejto možnosti neodosielala dané charakteristiky správne.

Ďalšou zmenou bolo upravenie metódy, ktorá konvertuje hexadecimálny reťazec na pole bajtov. Táto metóda bola síce implementovaná v pôvodnej verzii, ale nevyhovovala potrebám mobilnej aplikácie. Keby táto zmena nebola vykonaná, tak by prijaté dáta neboli rovnaké ako dáta vygenerované aplikáciou. To by malo za následok nesprávne vyhodnotenie užívateľských parametrov a tým pádom by nemal prístup k vozidlu. Nová metóda je zobrazená v ukážke 6.2 .

Výpis 6.2: Metóda pre konvertovanie hexadecimálneho reťazca

```
1  public static byte[] hexStringToByteArray(String s) {
2          if (s.length() % 2 != 0) {
3              throw new IllegalArgumentException(
4              "Invalid_hexadecimal_string");
5          }
6          int len = s.length() / 2;
7          byte[] data = new byte[len];
8          for (int i = 0; i < len; i++) {
9              int index = i * 2;
10             int byteValue = Integer.parseInt(
11             s.substring(index, index + 2), 16
12             );
13             data[i] = (byte) (byteValue & 0xFF);
14         }
15         return data;
16     }
```

Na strane mobilnej aplikácie sa pre komunikáciu s GATT serverom využíva knižnica Flutter reactive BLE. Táto knižnica umožňuje vytvárať spojenia s GATT serverom, odosielať a prijímať dáta, odhaľovať dostupné služby a charakteristiky na serveri, spravovať notifikácie a indikácie a vykonávať ďalšie operácie súvisiace s komunikáciou pomocou GATT protokolu.

Aby sa uskutočnilo spojenie medzi GATT serverom a mobilnou aplikáciou, tak bolo nutné vyriešiť jednotlivé kroky. V prvom kroku sa vykonala inicializácia BLE adaptéra, pre prísup BLE funkcionality z aplikácie. Nasledujúcim krokom bolo vytvorenie metódy, ktorá umožňuje skenovanie BLE zariadení v okolí a na základe skenu sa pripojí k danému GATT serveru. Zariadenie, na ktorom beží mobilná aplikácia skenuje BLE zariadenia v okolí, na základe UUID služby charakteristickej pre daný GATT server. Ak je služba objavená tak sa zariadenia môžu pripojiť, ak uplynie 10 sekúnd a GATT server sa nepodarí nájsť tak sa skenovanie ukončí a užívateľ bude informovaný o danej situácii.

Ak je uskutočnené spojenie medzi GATT serverom a zariadením, na ktorom beží mobilná aplikácia, tak môže prebehnúť výmena charakteristík. Charakteristiky sú čítané a zapisované pomocou metód a funkcií z knižnice Flutter reactive BLE. Podrobnejší prehľad o generovaní a spracovaní daných hodnôt z charakteristík je popísaný v nasledujúcej sekcii 6.4 .

6.4 Využitie kryptografie pre prístup do vozidla

Aby GATT server poslal odpoveď o tom, že je užívateľovi umožnený prístup k vozidlu je nutné vygenerovať správne parametre. Na tomto serveri je implementovaný protokol, ktorý je zložený z hashovacích funkcií SHA a symetrickej šifry AES v šifrovacom režime GCM.

Na strane mobilnej aplikácie bolo teda nutné využiť tieto dve kryptografické primitíva. Na základe výsledov z testu kryptografických knižníc (pozri kapitolu 5) bola zvolená knižnica Crypto pre hashovaciu funkciu SHA a pre symetrickú šifru AES bola zvolená knižnica PointyCastle.

V prvom kroku získa užívateľ reťazec, ktorý má požadovanú dĺžku z databáze. Z tohto reťazca sa zoberie prvých 32 bitov a následne sa zhashuje pomocou funkcie SHA. Typ funkcie SHA závisí od veľkosti kľúča, ktorý sa bude používať pri šifrovaní funkciou AES. Výsledok hashovania je tiež následne zmenšený na štvrtinovú veľkosť kľúča, ktorý sa bude používať pri šifrovaní a je zapísaný na GATT server do charakteristiky pomenovanej HatuCharacteristic.

Druhým krokom je vytvorenie reťazca, z ktorého bude vygenerovaný symetrický kľúč pre kryptografickú funkciu AES. Reťazec, ktorý sme získali z databáze a reťazec, ktorý sme v predošlom kroku vygenerovali zhashujeme pomocou hashovacej funkcie SHA. Taktiež typ funkcie SHA závisí od veľkosti použitého kľúča. Následne výsledok hashovacej funkcie je zmenšený na 1/8 veľkosti použitého symetrického kľúča.

Tretím krokom je vygenerovanie finálneho symetrického kľúča pre funkciu AES v režime GCM. Najprv je nutné prijať náhodnú hodnotu zo servera. Potom je nutné vygenerovať náhodnú hodnotu na strane aplikácie a následne ju zapísať na GATT

server do charakteristiky nonceUserCharacteristic. Finálny kľúč je vytvorený zhashovaním reťazca vytvoreným v druhom kroku, danej konštanty, náhodnej hodnoty zo strany servera a náhodnej hodnoty zo strany aplikácie. Taktiež je tu použitá hashovacia funkcia SHA, ktorej typ závisí od zadanej veľkosti kľúča. Následne je výsledok hashovacej funkcie zmenšený na 1/8 zadanej veľkosti kľúča.

Štvrtým krokom je generovanie inicializačného vektora, ktorý využíva symetrická šifra AES. Po vygenerovaní tohto parametru je zaslaný na GATT server do charakteristiky IvCharacteristic.

Finálnym krokom je vytvorenie šifrovaného textu. Ako bolo už vyššie spomenuté, tak v tomto kroku je využívaná symetrická šifra AES v režime GCM. Pomocou tejto šifry bol zašifrovaný reťazec získaný z databáze. V procese šifrovania bol využitý inicializačný vektor, ktorý bol vygenerovaný v predošlom kroku a symetrický kľúč vygenerovaný v treťom kroku. Následne bola výsledná šifra zaslaná na GATT server a zapísaná do charakteristiky CryptogramCharacteristic. Posielanie jednotlivých charakteristík je zobrazený vo výpise 6.3 .

Výpis 6.3: Posielanie charakterík na GATT server

```
1  //nahodná hodnota vygenerovaná na strane aplikácie
2  await interactor.sendCharacteristicValueWithResponse(
3      'bc3ba145-f588-4f86-8bc4-fb925a23dc31',
4      deviceMac,
5      byte_userNonce.toList());
6
7  //hatu
8  await interactor.sendCharacteristicValueWithResponse(
9      'b828f7a3-157a-4a4e-9c9b-3feb19b8e90d',
10     deviceMac,
11     byte_hatu.toList());
12
13 //inicializačný vektor
14 await interactor.sendCharacteristicValueWithResponse(
15     '27938afb-f6f0-4e19-a4b2-2545da6bad40',
16     deviceMac,
17     i.codeUnits);
18
19 //šifra
20 await interactor.sendCharacteristicValueWithResponse(
21     '23cab78f-28d9-4ecb-bfd1-1bba7b486fa3',
22     deviceMac,
23     s.codeUnits);
```

Server tieto parametre spracováva a následne posiela odpoveď. Na strane servera prebieha podobný proces ako na strane mobilnej aplikácie. Ak server vyhodnotí, že prijaté parametre sú správne tak zasiela užívateľovi odpoveď true, ak sú nesprávne vracia false.

Táto funkcia bola otestovaná na zariadeniach Xiaomi Redmi 9 pro a Iphone 12 mini. Priemerná doba, za ktorú zariadenie s operačným systémom Android vygenerovalo a poslalo dané hodnoty je 618,83 ms. Pre zariadenie s operačným systémom iOS bola táto doba 371,06 ms.

Záver

V dnešnom svete technológií sa stali mobilné zariadenia neoddeliteľnou súčasťou nášho života. Na týchto mobilných zariadeniach bežia mobilné aplikácie, ktoré vykonávajú rôzne úkony. Jedným z týchto úkonov môže byť aj rezervácia vozidla. Účelom tejto práce bolo vyvinutie multiplatformovej mobilnej aplikácie, ktorá by umožnila zamestnancovi firmy prístup k vozidlu. Aplikácia poskytuje užívateľovi rozhranie, ktoré umožňuje rezerváciu vozidla a jeho následnú autentizáciu pre prístup k vozidlu.

Na úvod bolo potrebné oboznámiť sa s vývojom mobilných aplikácií a technológiami potrebných k vývoju. Po oboznámení sa s vývojom mobilných aplikácií a technológiami potrebnými k vývoju bol zvolený framework Flutter ako základný stavebný blok. Okrem toho bol zvolený Firebase pre poskytovanie spoľahlivých serverových služieb.

V ďalšej kapitole sú popísané kryptografické primitíva, ktoré slúžia ako stavebné bloky v procese autentizácie užívateľa pomocou technlógie BLE. Správne použitie a implementácia je nevyhnutná pre zabezpečenie dôvernosti, integrity a autenticity dát v prostredí, kde je bezpečnosť kľúčovou požiadavkou.

Tretia kapitola popisuje technológiu Bluetooth. V tejto kapitole je vysvetlená funkcionalita technológie Bluetooth a technológie BLE a taktiež sú tu popísané ich vzájomné rozdiely.

V štvrtej kapitole je predstavený konceptuálny návrh. Konceptuálny návrh hovorí o prípadoch užitia, ktoré sa vyskytujú v mobilnej aplikácii. Ďalej predstavuje návrh databáze, užívateľského rozhrania a štruktúru mobilnej aplikácie. Tento konceptuálny návrh je dôležitou súčasťou tohto projektu, pretože predstavuje popis jednotlivých častí, z ktorých je mobilná aplikácia vytvorená.

Piata kapitola je zameraná na výber knižníc, ktoré implementujú funkcionalitu kryptografických primitív. Kryptografické knižnice sú dôležitým nástrojom pri implementácii bezpečnostných mechanizmou a preto je nutné vybrať spoľahlivé a overené knižnice. Tieto knižnice sú testované a na základe výsledkov navzájom porovnávané a vyhodnocované.

Posledná kapitola, hovorí o prevode konceptuálnych návrhov zo štvrtej kapitoly na fungujúcu mobilnú aplikáciu. Taktiež boli v tejto kapitole využité všetky poznatky z predchádzajúcich častí. Je tu popísaný kompletný proces implementácie.

Literatúra

[1] *Advanced Encryption Standard.* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 16 November 2022 [cit. 2022-11-20]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Advanced_Encryption_Standard>.

[2] AFANEH, Mohammad. *Bluetooth Low Energy (BLE): A Complete Guide.* Novelbits [online]. novelbits, 6 September 2022 [cit. 2023-05-24]. Dostupné z URL: <https://novelbits.io/bluetooth-low-energy-ble-complete-guide/>.

[3] ANAND, Aditya. *Breaking Down: SHA-1 Algorithm.* InfoSec Write-ups [online]. InfoSec Write-ups, 6 November 2019 [cit. 2022-11-18]. Dostupné z URL: <https://infosecwriteups.com/breaking-down-sha-1-algorithm-c152ed353de2>.

[4] APPMASTER. *What is Firebase?.* AppMaster [online]. AppMaster, 2020-23, 3 Február 2023 [cit. 2023-05-24]. Dostupné z URL: https://appmaster.io/blog/what-is-firebase>.

[5] AVSYSTEM. *Everyone is using Bluetooth Low Energy – should you?.* Avsystem [online]. Avsystem, 30 April 2021 [cit. 2023-05-24]. Dostupné z URL: https://www.avsystem.com/blog/bluetooth-low-energy-ble/>.

[6] BOUNCYCASTLE.ORG. *Pointycastle 3.6.2.* Pub.dev [online]. pub.dev, 9 September 2022 [cit. 2022-12-10]. Dostupné z URL: <https://pub.dev/packages/pointycastle>.

[7] BRADEN, Abby. *Cryptographic Primitive.* Webopedia [online]. Darien, Connecticut: INT Media Group, 11 December 2020 [cit. 2022-11-09]. Dostupné z URL: <https://www.webopedia.com/definitions/cryptographic-primitive/>.

[8] BURDA, Karel. *Aplikovaná kryptografie.* Brno: VUTIUM, 2013. ISBN 978-80-214-4612-0.

[9] BURNETT, Mark a James C. FOSTER. *Hacking the Code.* Oxford, UK: SYNGRESS, 2004. ISBN 9780080478173.

[10] CAVALCANTE, Leo. *Encrypt 5.0.1.* Pub.dev [online]. pub.dev, 8 August 2021 [cit. 2022-12-10]. Dostupné z URL: <https://pub.dev/packages/encrypt>.

[11] DART. *Dart overview.* In: Dart [online]. Mountain View, Kalifornia, USA: Dart, 2011- [cit. 2022-10-16]. Dostupné z URL: <https://dart.dev/overview>.

[12] DART.DEV. *Crypto 3.0.2.* Pub.dev [online]. pub.dev, 27 Apríl 2022 [cit. 2022-12-10]. Dostupné z URL: <https://pub.dev/packages/crypto>.

[13] *Dart (programming language).* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 4 Október 2022 [cit. 2022-10-16]. Dostupné z URL: <https://en.wikipedia.org/wiki/Dart_(programming_language)>.

[14] DINT.DEV. *Cryptography 2.0.5.* Pub.dev [online]. pub.dev, 14 November 2021 [cit. 2022-12-10]. Dostupné z URL: <https://pub.dev/packages/cryptography>.

[15] FLUTTER. *Flutter architectural overview.* In: Flutter [online]. 20 Máj 2022 [cit. 2022-11-07]. Dostupné z URL: <https://docs.flutter.dev/resources/architectural-overview>.

[16] *Flutter (software).* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 16 Október 2022 [cit. 2022-10-16]. Dostupné z URL: <https://en.wikipedia.org/wiki/Flutter_(software)>.

[17] FRANKLIN, Curt a Chris POLLETTE. *How Bluetooth Works.* How Stuff Works [online]. How Stuff Works, 1998-, 10 February 2021 [cit. 2022-12-02]. Dostupné z URL: <https://electronics.howstuffworks.com/bluetooth.htm>.

[18] GOOGLE. *Meet Android Studio.* In: Developers [online]. 11 Júl 2022 [cit. 2022-11-07]. Dostupné z URL: <https://developer.android.com/studio/intro#code_completion>.

[19] GOOGLE. *The Android Profiler.* In: Developers [online]. 28 Júl 2021 [cit. 2022-11-07]. Dostupné z URL: <https://developer.android.com/studio/profile/android-profiler?gclsrc=ds&gclsrc=ds>.

[20] IBM. *Supported schemes.* In: Github [online]. San Francisco, Kalifornia, USA: GitHub, 2008-, 3 Apríl 2021 [cit. 2022-12-10]. Dostupné z URL: <https://github.com/IBM/libgroupsig/wiki/Supported-schemes#available-schemes>.

[21] IBM. *Public key cryptography.* IBM [online]. IBM, 1911-, 17 November 2022 [cit. 2022-11-26]. Dostupné z URL: <https://www.ibm.com/docs/en/ztpf/2022?topic=concepts-public-key-cryptography>.

[22] IBM. *Symmetric cryptography.* IBM [online]. IBM, 1993-, 17 November 2022 [cit. 2022-11-19]. Dostupné z URL: <https://www.ibm.com/docs/en/ztpf/2022?topic=concepts-symmetric-cryptography>.

[23] ISLAM, Rashedul; ISLAM, Rofiqul; MAZUMDER, Tohidul. *Mobile application and its global impact.* International Journal of Engineering & Technology, 2010, 10.6: 72-78.

[24] JASON, Marcel. *3 Key Factors That Determine the Range of Bluetooth.* Bluetooth [online]. Bluetooth, 1998-, 17 October 2022 [cit. 2022-12-02]. Dostupné z URL: <https://www.bluetooth.com/blog/3-key-factors-that-determinethe-range-of-bluetooth/>.

[25] KOTLIN. *What is cross-platform mobile development?.* In: Kotlin [online]. Kotlin, 2012-, 6 September 2022 [cit. 2022-10-15]. Dostupné z URL: <https://kotlinlang.org/docs/cross-platform-mobile-development.html>.

[26] LANDMAN, Nathan, Christopher WILLIAMS, Eli ROSS a Jimin KHLM. *Secure Hash Algorithms.* Brilliant [online]. Brilliant, 2012-, 18 November 2021 [cit. 2022-11-18]. Dostupné z URL: <https://brilliant.org/wiki/secure-hashing-algorithms/>.

[27] MAGENEST. *WHAT IS NATIVE APP? DEFINITION, TIPS AND BEST EXAMPLE – UPDATED 2022.* In: Magenest [online]. magenest, 2021-, 30 November 2021 [cit. 2022-10-15]. Dostupné z URL: <https://magenest.com/en/native-app/>.

[28] MENEZES, A. J., Paul van OORSCHOT a Scott A. VANSTONE. *Handbook of applied cryptography.* Boca Raton: CRC Press, 1997. xiii, 780. ISBN 0-8493-8523-7.

[29] *Mobile app.* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 3 Október 2022 [cit. 2022-10-15]. Dostupné z URL: <https://en.wikipedia.org/wiki/Mobile_app>.

[30] NIST. *Federal Information Processing Standards Publication 197.* NIST, 2001.

[31] PUB.DEV. *Ninja 3.0.8.* Pub.dev [online]. pub.dev, 15 Marec 2022 [cit. 2022-12-10]. Dostupné z URL: <https://pub.dev/packages/ninja>.

[32] PAYNE, Rap. *Beginning App Development with Flutter: Create Cross-Platform Mobile Apps.* New York, NY: Apress, 2019. ISBN 978-1-4842-5180-5.

[33] SAUCE LABS. *Native vs. Web vs. Hybrid vs. Progressive Web Apps: Key Differences for Development and Mobile Testing.* In: Sauce Labs [online]. San Francisco (CA): Sauce Labs, 2008-, 28 Október 2021 [cit. 2022-10-15]. Dostupné z URL: <https://tinyurl.com/y3y94n82>.

[34] SMIRNOFF, Peter a Dawn M. TURNER. *Symmetric Key Encryption - why, where and how it's used in banking.* Cryptomathic [online]. Aarhus, DK: Cryptomathic, 1986-, 18 Január 2019 [cit. 2022-11-19]. Dostupné z URL: <https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking>.

[35] TOWNSEND, Kevin. *GAP.* Adafruit [online]. Adafruit, 24 Január 2014 [cit. 2023-05-24]. Dostupné z URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap#device-roles-621417>.

[36] TOWNSEND, Kevin. *GATT.* Adafruit [online]. Adafruit, 24 Január 2014 [cit. 2023-05-24]. Dostupné z URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt>.

[37] VERMA, Madhvi a Satbir SINGH. *An Overview of Bluetooth Technology and its Communication Applications.* International Journal of Current Engineering and Technology [online], 2015 [cit. 2022-12-02]. Dostupné z URL:<https://www.researchgate.net/publication/276251559_An_Overview_of_Bluetooth_Technology_and_its_Communication_Applications>.

[38] WOOLLEY, Martin. *Bluetooth Core Specification v5.1.* [online]. 28 Január 2019 [cit. 2023-05-24]. Dostupné z URL: <https://davidhoglund.typepad.com/files/1901_feature_overview_brief_final.pdf>.

[39] WU, Wenhao. *React Native vs Flutter, Cross-platforms mobile application frameworks.* Helsinki, FIN, 2018.

Zoznam symbolov a skratiek

AES Advanced Encryption Standard

BBS04 Boneh, Boyena and Sacham Short Group Signatures 4

BLE Bluetooth Low Energy

dBm Decibel-milliwatts

DES Data Encryption Standard

DL Discrete Logarithm

DL21 Diaz a Lehmann 21

EC Elliptic Curve

ECDH Elliptic Curve Diffie–Hellman

ECDSA Elliptic Curve Digital Signature Algorithm

ER Entity-Relationship

GAP Generic Access Profile

GATT Generic Attribute

GPS Global Positioning System

GL19 Garms and Lehmann 19

HCI Host Controller Interface

IDE Integrated Development Environment

IDEA International Data Encryption Algorithm

IF Integer Factorization

IoT Internet of Things

IPsec Internet Protocol Security

JSON JavaScript Object Notation

KLAP20 Kim, Lee Abdalla a Park 20

L2CAP Logical Link COntrol and Adaptation Protocol

MHz Megahertz

mW Milliwatt

NIST National Institute of Standards and Technology

OEM Original Equipment Manufacturer

OS Operačný systém

PS16 Pointcheval and Sanders 16

RC4 Rivest Cipher 4

RC5 Rivest Cipher 5

RC6 Rivest Cipher 6

RSA Rivest–Shamir–Adleman

RFCOMM Radio Frequency Communication

SDK Software Development Kit

SHA Secure Hash Algorithm

SSL Secure Sockets Layer

SSH Secure Shell

SDP Service Discovery Protocol

TLS Transport Layer Security

TCS Telephony Control Protocol

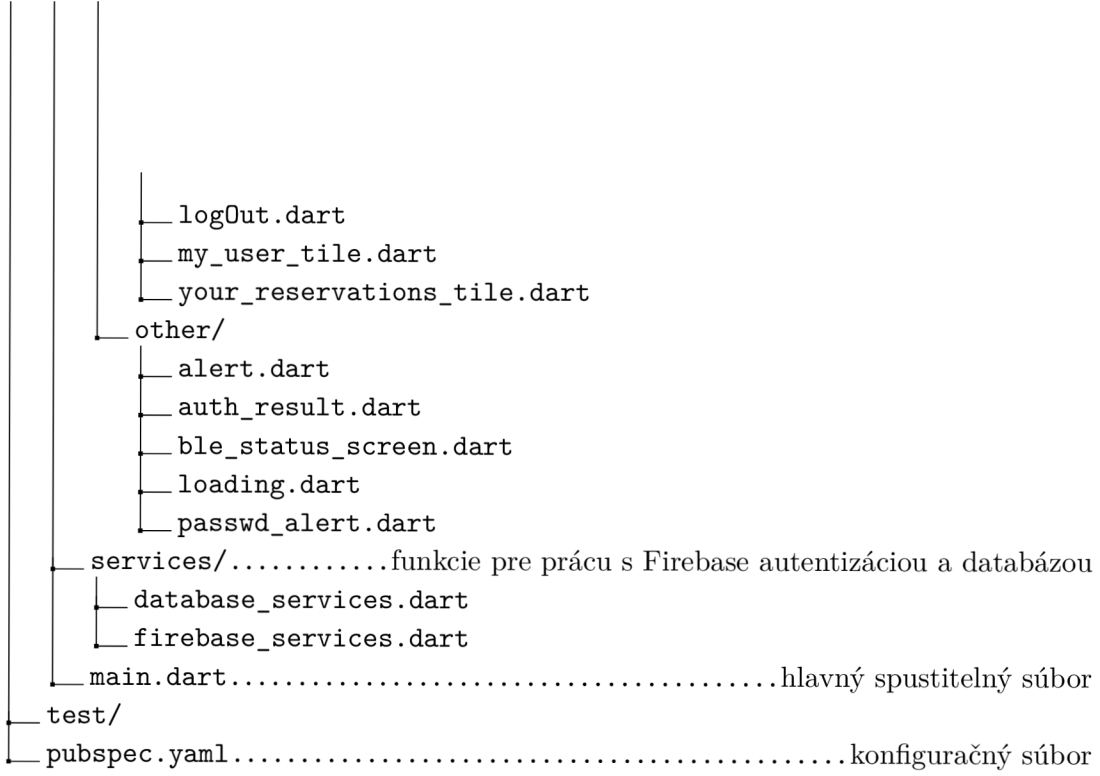
UML Unified Modeling Language

UUID Universally unique identifier

A Obsah elektronické přílohy

V elektronickej prílohe je obsiahnutý kompletný zdrojový kód mobilnej aplikácie. Naseldujúca stromová štruktúra obsahuje popis hlavných adresárov.

- `/` ..... koreňový adresár priloženého archívu
  - `dart_tool/`
  - `.idea/`
  - `android/`
  - `assets/`
  - `build/`
  - `ios/`
  - `lib/` ..... zložka obsahujúca podstatné súbory pre funkciu aplikácie
    - `auth/` ..... umiestnenie logiky stavovej navigácie
      - `auth_page.dart`
      - `main_page.dart`
    - `ble/` ..... funkcie pre prácu s BLE
      - `bleStatusMonitor.dart`
      - `connector.dart`
      - `interactor.dart`
    - `crypto/` ..... funkcie pre generovanie kryptografických parametrov
      - `crypto_core.dart`
    - `model/` ..... triedy pre ukladanie entít
      - `car.dart`
      - `car_key.dart`
      - `globals.dart`
      - `my_user.dart`
      - `reservation.dart`
    - `pages/`
      - `authentication/` ..... prihlasovací a registračný formulár
        - `login.dart`
        - `registration.dart`
      - `home/`
        - `car_reservation/` ..... časť pre rezerváciu vozidla
          - `calendar.dart`
          - `calendar_provider.dart`
          - `car_reservation.dart`
          - `car_reservation_tile.dart`
        - `reservations/` ..... časť pre zobrazenie rezervácii užívateľa
          - `reservations.dart`
          - `reservations_provider.dart`
          - `reservations_tile.dart`
        - `settings/` ..... časť pre editovanie uživateľovych údajov
          - `settings.dart`
        - `chooseTile.dart`
        - `home.dart`


logOut.dart
my_user_tile.dart
your_reservations_tile.dart
other/
alert.dart
auth_result.dart
ble_status_screen.dart
loading.dart
passwd_alert.dart
services/.....funkcie pre prácu s Firebase autentizáciou a databázou
database_services.dart
firebase_services.dart
main.dart.....hlavný spustitelný súbor
test/
pubspec.yaml.....konfiguračný súbor