STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č.18: Informatika

RidersTrack – mobilní aplikace pro motorkáře

Matěj Mlynář
Pardubický kraj

Pardubice, 2024

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č.18: Informatika

RidersTrack – mobilní aplikace pro motorkáře

RidersTrack – mobile app for motorcycle riders

Autoři: Matěj Mlynář
Škola: DELTA – Střední škola informatiky a ekonomie, s.r.o.
Ke Kamenci 151, 530 03 Pardubice
Kraj: Pardubický
Konzultant: Bc. Vlaďka Janů

Pardubice, 2024

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 22.3.2024 ……

Matěj Mlynář

Poděkování

V první řadě bych chtěl poděkovat mé vedoucí projektu Bc. Vlaďce Janů za odborné vedení a věcné připomínky, které jsem od ní v průběhu vývoje obdržel. Dále bych chtěl poděkovat všem lidem, kteří se podíleli na testovací fázi aplikace.

Anotace

Tato práce se především zabývá návrhem a realizací mobilní aplikace, která umožňuje uživatelům zobrazovat statistiky ze svých motocyklových jízd a sdílet tyto záznamy s ostatními uživateli. Aplikace také poskytuje správu pohonných hmot s výhodou přehledného zobrazení těchto dat na grafech, umožňující uživatelům monitorovat průměrnou spotřebu paliva a náklady. Systém je postaven na frontendové technologii Flutter, backendovém frameworku Express a databázovém systému MongoDB.

Klíčová slova

Flutter; MongoDB; Express; Google Maps API; Linux, Motorkáři; Motocykly, Statistiky

Annotation

This thesis mainly focuses on the design and implementation of a mobile application that allows users to view statistics from their motorcycle rides and share these records with other users. The application also provides fuel management with the benefit of clearly displaying this data on graphs, allowing users to monitor average fuel consumption and costs. The system is built on the frontend technology Flutter, the Express backend framework and the MongoDB database system.

Keywords

Flutter; MongoDB; Express; Google Maps API; Linux Motorcycles; Motorcycle riders, Stats

Obsah

1 Úvod 1
2 Teoretická část 2
2.1 Motivace k realizaci projektu 2
2.2 Podobná existující řešení 2
2.3 Pojmy 2
2.4 Technologie 4
2.5 Nástroje 5
3 Praktická část 5
3.1 Architektura 5
3.2 Backend 7
3.3 Mobilní aplikace 14
3.4 Webová stránka 21
4 Závěr 22

1 Úvod

V dnešním světě se motorkářská komunita rozrůstá stále rychleji. Ať už se jedná o mladší nebo starší členy komunity, mezi motorkáři roste zájem o aplikaci, která by umožnovala zaznamenávání a sdílení jednotlivých motorkářských výletů mezi ostatní uživatele s možností sledování detailních statistik. Motorkáři zároveň vyžadují možnost vést svůj „deník“ palivových záznamů, aby mohli sledovat průměrnou spotřebu paliva, náklady na pohonné hmoty za motorkářskou sezónu a celkový přehled svých finančních výdajů v této oblasti.

Cílem této práce je navržení a naprogramování mobilní aplikace, která by byla dostupná pro platformy iOS i Android a byla by zaměřena na motorkáře, kteří chtějí sledovat a sdílet své motorkářské jízdy. Aplikace by umožnovala záznam motorkářských tras, přidávání neomezeného množství motorek do virtuální garáže a spravovat pro ně palivové záznamy. Důraz je kladen na přehledné a atraktivní uživatelském prostředí.

Tato aplikace má za cíl nabídnout motorkářům užitečnou aplikaci, která by zlepšovala celkový motorkářský zážitek.

2 TEORETICKÁ ČÁST

2.1 Motivace k realizaci projektu

Sám jako motorkář jsem si povšiml, že motorkáři často čelí problému při hledání aplikace, která by sjednocovala všechny potřebné funkcionality. Na trhu chybí univerzální aplikace a to znamená, že uživatel musí stahovat více aplikací, což může být mnohdy nepřehledné a nepraktické pro samotného uživatele.

Hlavní motivací bylo především reálné využití aplikace a zlepšení celkového zážitku motorkářů poskytnutím užitečné aplikace pro širokou motorkářskou komunitu.

2.2 Podobná existující řešení

V rámci rešerše bylo zjištěno, že již existuje několik aplikací podobného zaměření, ale většina z nich se zabývá pouze specifickou funkcionalitou. Motorkáři jsou tak nuceni mít ve svém mobilním zařízení staženo větší množství aplikací, které mezi sebou nejsou ale nijak propojené, a to se pro uživatele může jevit jako nepřehledné. Některé, již využívané aplikace slouží k záznamu tras, zatímco jiné se specializují na správu palivových hmot.

SuperBiker

Aplikace vyniká především v záznamu motorkářských tras, obsahuje detailní zobrazení statistik a data vizualizuje v přehledném prostředí. Její nevýhodou je ale absence správy palivových hmot a nedisponuje interakcí mezi uživateli.

MyCar

Aplikace, která je primárně mířena pro automobily, ale dá se využívat i pro motocykly. Vyniká především správou pohonných hmot nebo sledování nákladů a servisních intervalů.

2.3 Pojmy

2.3.1 Framework

Framework je soubor předem vytvořených nástrojů, knihoven, pravidel a konvencí, které usnadňují a urychlují vývoj softwaru. Framework poskytuje strukturu a základní funkcionality, což umožňuje vývojářům zaměřit se především na implementaci specifických funkcí nebo vlastností své aplikace, místo toho, aby se zabývali opakujícími se úkoly a nízko úrovňovými detaily.

Můžeme se setkat s různými druhy frameworků podle toho, jaké oblasti vývoje se týká:

- „**Webové** frameworky zaměřené na vývoj webových stránek a aplikací.
- **Mobilní** frameworky pro vývoj aplikací pro chytré telefony.
- **UI** frameworky zaměřené na vývoj uživatelského rozhraní.
- Frameworky **pro databáze** usnadňující jejich vývoj a práci s daty a další.“ [1]

2.3.2 NoSQL

„Namísto typické tabulkové struktury relačních databází obsahují databáze NoSQL data v rámci jedné datové struktury, například dokumentu JSON. Protože tato nerelační databázová konstrukce nevyžaduje schéma, nabízí rychlou škálovatelnost pro správu velkých a typicky nestrukturovaných souborů dat.“ [2]

2.3.3 CROSS-PLATFORM

Crossplatform vývoj (nebo též multiplatformní vývoj) je přístup k softwarovému vývoji, při kterém jsou aplikace navrhovány a vyvíjeny tak, aby mohly být spuštěny na více než jednom operačním systému nebo platformě. Cílem crossplatform vývoje je minimalizovat duplicitu práce a umožnit vývojářům vytvářet aplikace, které mohou být nasazeny na různé platformy s co nejmenšími úpravami.

Pro dosažení tohoto cíle se často využívají frameworky a technologie, které umožňují vytváření aplikací, které mohou být spuštěny na různých operačních systémech, jako jsou iOS, Android nebo další. To může zahrnovat použití multiplatformních frameworků, jako je například Flutter nebo React Native, které umožňují vytvářet mobilní aplikace a sdílí kód mezi různé platformy.

Crossplatform vývoj je klíčovým aspektem v dnešní době, kdy uživatelé přistupují k aplikacím z různých zařízení s různými operačními systémy.

2.3.4 Frontend

Část softwaru nebo webové aplikace, která prezentuje data poskytnutá z backendové části systému a zobrazí je návštěvníkům nebo uživatelům. Při vytváření frontendu se vývojáři soustředí např. na vizuál dané stránky webu nebo aplikace a její uživatelskou přívětivost.

2.3.5 Backend

Hlavní část softwaru, kterou běžný uživatel nevidí, ale která zajišťuje fungování aplikace. Například správu objednávek a skladu e-shopu nebo redakční systém blogu. Backend slouží i k zabezpečení přístupu k datům.

2.3.6 Cron

CRON je nástroj používaný pro automatizaci plánovaných a pravidelně se opakujících úkolů při správě softwaru. Tento nástroj umožňuje specifikovat konkrétní časové úseky, kdy má dojít k provádění určitých akcí, jako je například zálohování dat, aktualizace a další údržbové

činnosti. CRON přispívá k efektivnímu a pravidelnému řízení rutinních úkolů v rámci správy softwarových systémů.

2.3.7 VPS

VPS (Virtual Private Server) je virtuální instance operačního systému, která je hostována na fyzickém serveru a poskytuje uživatelům izolovaný a konfigurovatelný prostor pro provozování aplikací a služeb.

2.3.8 MVC

Softwarová architektura rozdělující software na datový model, uživatelské rozhraní a řídicí logiku aplikace – Model definuje práci s daty, pohled (view) data vizualizuje a kontroler stanovuje pravidla, která drží celou aplikaci pohromadě.

2.3.9 Middleware

Middleware je softwarová vrstva, která se nachází mezi různými částmi aplikace nebo systému a zprostředkovává jejich vzájemnou interakci. Slouží k zpracování žádostí, dat nebo akcí při přenosu mezi různými vrstvami aplikace. Middleware může poskytovat funkcionalitu, jako jsou autentizace, autorizace, manipulace s daty, zabezpečení a další.

2.4 Technologie

2.4.1 Flutter

Flutter je open-source framework vyvinutý společností Google, umožňující vývoj mobilních aplikací pro iOS, Android a další platformy z jednoho zdrojového kódu. Jeho klíčovou vlastností je hot-reload, který umožňuje okamžitě vidět změny v kódu během vývoje.

2.4.2 Typescript

TypeScript je nadstavba jazyka JavaScript, poskytující statické typování, interface a další konstrukce, což usnadňuje rozvoj komplexních webových aplikací. Kód napsaný v TypeScriptu je kompilován do standardního JavaScriptu.

2.4.3 Node.js

Node.js je prostředí pro běh JavaScriptových aplikací na straně serveru. Postavený na V8 (engine) od společnosti Google. Node.js umožňuje vývojářům psát serverový kód v JavaScriptu a zpracovávat asynchronní operace efektivně. Tím se poskytuje rychlé a škálovatelné řešení pro vývoj serverových a síťových aplikací.

2.4.4 MongoDB

MongoDB je NoSQL databázový systém, který ukládá data ve formátu BSON. Nabízí škálovatelnost a flexibilitu a je často využíván pro ukládání a manipulaci s velkým objemem nestrukturovaných dat.

2.4.5 GIT

Git představuje systém pro správu verzí, umožňující sledovat změny v kódu, verzovat projekt a koordinovat spolupráci mezi vývojáři. Tento nástroj zabezpečuje bezpečný a efektivní vývoj softwaru, poskytující funkcionality pro sledování a správu historie projektu, efektivní práci v týmu a řešení potenciálních konfliktů v kódu.

2.5 Nástroje

2.5.1 WebStorm

WebStorm je vývojové prostředí navržené především pro webové aplikace. WebStorm je součástí produktové řady JetBrains IDE, která je známá svým zaměřením na produktivitu vývojářů a poskytování rozsáhlých nástrojů pro zjednodušení vývoje.

2.5.2 Android Studio

Android Studio je vývojové prostředí speciálně pro vývoj mobilních aplikací. Je vyvinuté společností JetBrains ve spolupráci s Google, Android Studio přináší širokou škálu funkcí a nástrojů, které podporují efektivní vývoj mobilních aplikací pro Android i iOS (např. Flutter, Java).

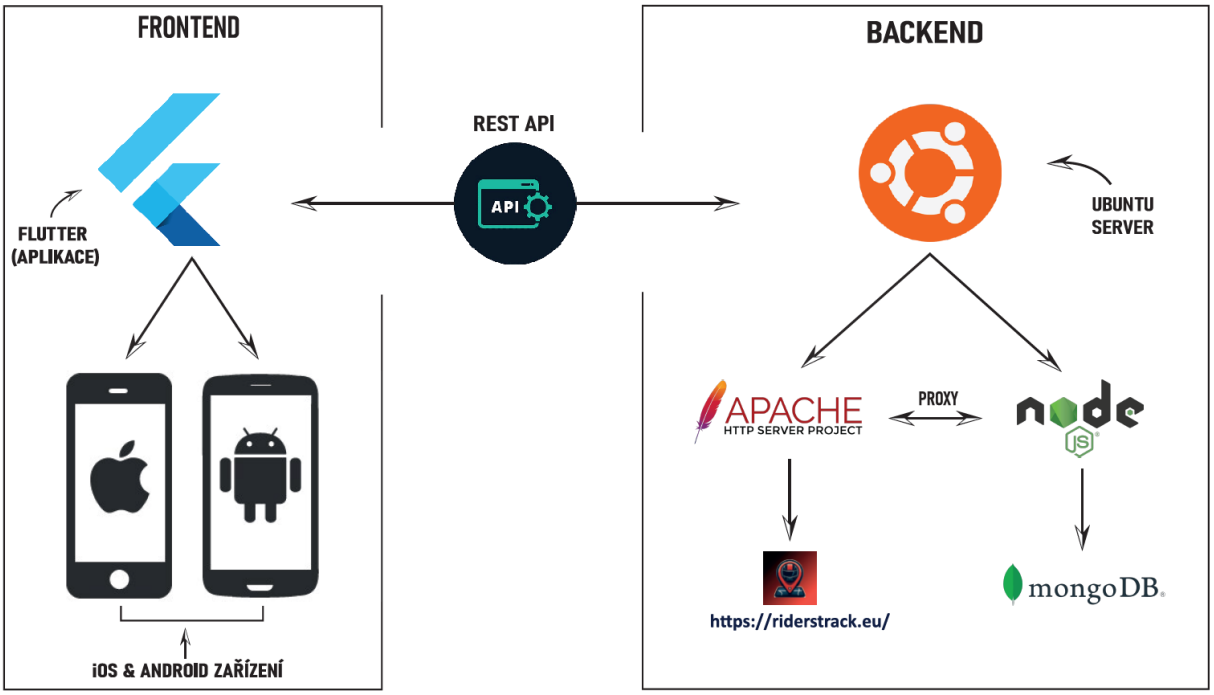
2.5.3 Postman

Postman je oblíbený nástroj pro testování a vývoj HTTP API. Jedná se o uživatelsky přívětivé rozhraní, které poskytuje prostředky pro odesílání HTTP požadavků, testování odpovědí a organizaci a dokumentaci API projektů. Postman usnadňuje práci s API a zjednodušuje proces testování, ladění a sdílení API i mezi týmy.

3 PRAKTICKÁ ČÁST

3.1 Architektura

Architektura celkového systému včetně aplikace se skládá z několika částí. Uživatel interaguje s mobilní aplikací, která se dotazuje na backendový server, nasazený na linuxovém serveru, na kterém se nachází i samotná databáze. Podrobná grafická vizualizace architektury systému je vidět na obr. 3.1.



Obr. 3.1: Architektura systému

Navržený systém se skládá ze dvou hlavních částí. První z nich je frontend, pro který byl nakonec využit framework Flutter s programovacím jazykem Dart. Flutter byl zvolen především díky multiplatformnímu řešení a efektivním vývoji. Tato část je prakticky tváří systému, kterou vidí a ovládá uživatel. Byla navržen tak, aby poskytovala plynulý a uživatelsky přívětivý zážitek a byla pro uživatele přehledná.

Flutter také poskytuje optimalizaci pro samotný běh aplikace, jelikož nabízí dva druhy widgetů (komponent).

Stateless widget

- Widget, který nemá žádný vnitřní stav, takže jeho stav (data) zůstává konstantní po celou dobu života komponenty.
- Je jednou inicializovaný, jeho stav nemůže být změněn. Stateless widget je ideální pro prezentaci statických informací nebo pro situace, kdy nemusí reagovat na uživatelské interakce nebo změny dat.
- Příklady stateless widgetů zahrnují textové popisky, ikony nebo například statické obrázky.

Stateful widget

- Stateful widget udržuje interní stav, který může být během života widgetu měněn. Tento stav může reagovat na uživatelské interakce, vstupy nebo změny v aplikaci.
- Při změně stavu volá metodu build(), což způsobuje aktualizaci uživatelského rozhraní. Stateful widget je vhodný pro složitější widgety, které potřebují reagovat na dynamické změny třeba od uživatele.

- Příklady stateful widgetů zahrnují formuláře, interaktivní prvky a všechny widgety, které potřebují udržovat a aktualizovat svůj vnitřní stav.

Druhou klíčovou částí systému je backend naimplementovaný v TypeScriptu, který pro ukládání dat využívá spolehlivou databázi MongoDB. Tato část hraje klíčovou roli v poskytování dat pro frontend a zajišťuje efektivní správu dat.

3.2 Backend

3.2.1 Využité technologie

Pro backendovou část projektu byl zvolen a naprogramován Node server v kombinaci s frameworkem Express. Pro implementaci bylo využito TypeScriptu, který poskytuje výhody typování a tím zajišťuje lepší kontrolu nad kódem. Tato volba přispívá k celkové stabilitě a rychlosti vývoje backendu.

Databázový systém, který byl zvolen pro úložiště dat je MongoDB. Tato volba vychází především z její flexibility a schopnosti snadného propojení s backendovou částí projektu. Využití této databáze přináší efektivní řešení pro situace, kdy daná struktura dat může být dynamická.

3.2.2 Použití MVC architektury

V rámci projektu byla zvolena architektura Model-View-Controller-Service (MVC+S) především pro organizaci, strukturování kódu a zajištění celkové přehlednosti v projektu, která je vizuálně zobrazena na obr. 3.2.

Model

Modely představují datovou vrstvu aplikace a jsou zodpovědné za manipulaci s daty. Obsahují také datový stav aplikace. V této části projektu jsou definovány a implementovány všechny datové struktury, business logika a operace, které souvisejí s manipulací s daty.

- Definice datových struktur a jejich vlastností.
- Implementace logiky pro práci s daty a validace.
- Interakce s databází a manipulace s daty.
- Držení stavu aplikace.

Kontroler (Controllers)

Kontrolery představují logickou část aplikace, která reaguje na uživatelské vstupy a provádí potřebné akce. Jsou prostředníkem mezi modelem, službou a pohledem, zajišťují synchronizaci a koordinaci mezi daty a uživatelským rozhraním. Kontrolery iniciují služby pro vytvoření API requestů.

- Přijímání uživatelských požadavků a vstupů.
- Koordinace s modelem a aktualizace dat podle potřeby.
- Odesílání odpovědí pohledu a uživatelskému rozhraní.

Služby (Services)

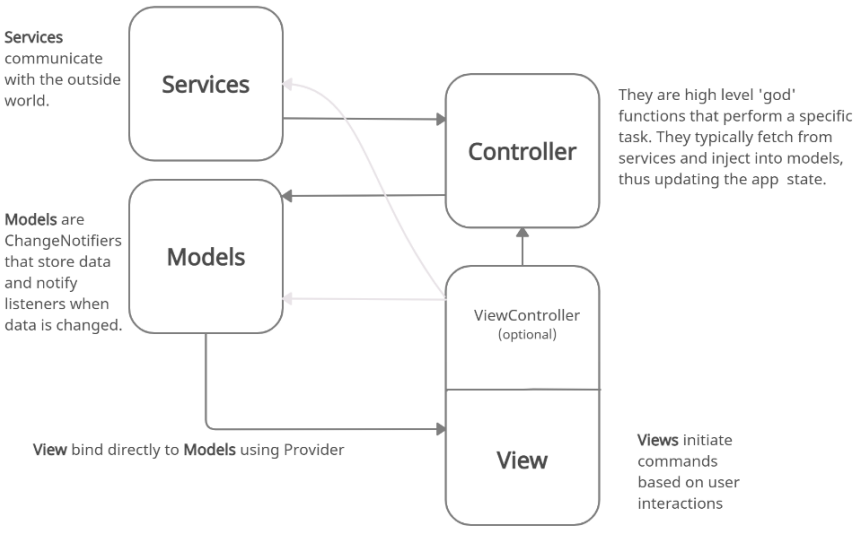
Služby úzce spolupracují s backendem. Jejich hlavní funkčnost je získávat data od backendového serveru a následně tyto informace poskytovat jako odpověď kontrolerům, které službu požádaly.

- Získávání dat od backendu.
- Zpracování dat.
- Optimalizace komunikace.

Pohledy (Views)

Pohledy mají pouze jediný úkol, a to je data vizualizovat. Jsou také jedinou částí architektury, s kterou uživatel přijde do kontaktu. Pohled nejčastěji komunikuje s kontrolerem, který mu z modelu poskytuje data.

- Vizualizace dat pro uživatele.
- Reagování na změny v datech a aktualizace zobrazení.
- Předávání uživatelských akcí kontroleru.



Obr. 3.2: Architektura MVC+S, zdroj: [3]

3.2.3 Komunikace s frontendem

V rámci integrace frontendu s backendem je klíčovým prvkem využití REST API. Backendová část projektu poskytuje REST API, což umožňuje jednoduchou a efektivní komunikaci s frontendovou částí aplikace.

API je navrženo podle principů REST architektury, což znamená, že komunikace probíhá pomocí standardních HTTP metod, jako jsou GET, POST, PUT a DELETE.

Pro ověření správné funkcionality API byl využit nástroj Postman. Tento nástroj umožňuje testování různých HTTP požadavků, validaci odpovědí a ladění API, což zajišťuje, že API je schopné adekvátně reagovat na různé scénáře a poskytovat očekávané výsledky.

3.2.4 Nasazení na server

Pro klíčovou funkčnost systému je nezbytné nasazení backendu a databáze na server, což zajišťuje celkové využití aplikace v praxi. Pro tento účel bylo využito virtuálního privátního Linux serveru (VPS) s operačním systémem UBUNTU. Na tomto serveru byly také spuštěny rovněž tzv. „CRON“ úlohy, které každý měsíc automaticky zažádají o SSL certifikát pro propagační web, který je taktéž součástí projektu. Na serveru byla nakonfigurována i služba PM2, která zajišťuje neustálý provoz serveru. V případě, kdyby se na serveru stala neočekávaná chyba a server by spadnul, služba PM2 je schopna Node server zpět nastartovat.

Výraznou výhodou využívání virtuálního privátního serveru spočívá v jeho ekonomické efektivitě. Nízké měsíční náklady na pronájem VPS služby umožňují uživatelům snížit své náklady a získat přístup k robustnímu serverovému prostředí za dostupnou cenu.

Další klíčovou výhodou VPS je rychlost a jednoduchost vytvoření nového pracovního prostoru. Díky virtuálnímu charakteru těchto serverů lze nové instance spustit velmi rychle a snadno, což přináší flexibilitu a škálovatelnost pro uživatele, kteří potřebují okamžitý přístup k novým serverovým zdrojům bez komplikovaných a zdlouhavých procesů.

3.2.5 Kolekce koncových bodů (endpointů)

Bylo nakonfigurováno velké množství jednotlivých endpointů pro zajištění komunikace mezi frontendem a backendem. Níže jsou na ukázku popsány 3 endpointy. Zbytek endpointů je definován obdobným způsobem.

Motorkářské záznamy

Route: /createRideRecord

Popis: Uloží motorkářskou jízdu s potřebnými daty

```
Struktura URL: https://api.riderstrack.eu/api/rideRecord/createRideRecord
```

Metoda: POST

Parametry

Jméno parametru	**Datový typ**	**Popis**
motorcycleId	String	Id vozidla, pro které je záznam přiřazen
date	Date	Datum, kdy byl záznam pořízen
totalDistance	Double	Celková vzdálenost trasy
duration	Int	Doba trvání záznamu (vteřiny)
maxSpeed	Double	Maximální rychlost dosažena v průběhu záznamu
positionPoints	Array	Pole objektů, které obsahuje všechny záznamy o poloze. Struktura jednotlivého bodu je vidět na obr. 3.7.
isPublic	Bool	Hodnota, která určuje, zda má být záznam veřejný

Tabulka 1: Parametry endpointu pro vytvoření záznamu jízdy

Kontrola verze aplikace

Route: /checkAppVersion

Popis:

> Jelikož se aplikace nachází v testovací verzi a zatím není dostupná na obchodech jako je například AppStore nebo Google Play, tak je potřeba uživatele informovat o nové verzi pomocí vlastního řešení. V rámci tohoto řešení je porovnána verze aplikace, kterou má uživatel nainstalovanou s verzí, která je nahrána na serveru. V případě, že se verze aplikací neshodují, je uživatel informován o dostupnosti stažení nové verze, která je nahrána na webu.

```
Struktura URL: https://api.riderstrack.eu/api/general/checkAppVersion
```

Metoda: GET

Parametr

Jméno parametru	Datový typ	Popis
version	String	Verze mobilní aplikace, kterou má uživatel staženou

Tabulka 2: parametry endpointu pro kontrolu verze aplikace

Palivové záznamy

Route**:** /api/fuelRecord/createFuelRecord

Popis: Vytvoří palivový záznam

```
Struktura URL: https://api.riderstrack.eu/api/fuelRecord/createFuelRecord
```

Metoda: POST

Parametry

Jméno parametru	Datový typ	Popis
motorcycleId	String	Id vozidla, pro které je palivový záznam přiřazen
liters	Double	Množství paliva, které bylo načerpáno
totalPrice	Double	Celková cena načerpání pohonných hmot

date	Date | null	Datum, kdy tankování proběhlo. Volitelný parametr
distance	Double | null	Vzdálenost, kterou uživatel najel od posledního tankování. Volitelný parametr
consumption	Double | null	Spotřeba vozidla. Volitelný parametr

Tabulka 3: Parametry endpointu pro záznam palivových záznamů

3.2.6 Zabezpečení endpointů

Backend si sám pomocí middleware ověřuje, jaké requesty jsou platné a jaké ne. Jak je vidět na obr. 3.3, byl naimplentován middleware, který je navržen k ověření existence a správnosti přenášeného tokenu v hlavičce HTTP žádosti (authorization). Pokud je token platný, middleware přidává hodnotu tokenu do objektu „Request“ a pokračuje voláním funkce next(). V opačném případě, když token chybí nebo je neplatný, middleware odpovídá s HTTP stavovým kódem 403 (Forbidden).

```
1 usage  Matěj Mlynář
interface RequestWithToken extends Request{
    token: string
}

5+ usages  Matěj Mlynář *
export const checkToken = (req : RequestWithToken, res : Response, next : NextFunction) : void  => {
    const header : string|undefined = req.headers['authorization'];
    if(typeof header !== 'undefined'){
        const bearer : string[] = header!.split( separator: ' ');
        req.token = bearer[1];
        next();
    }
    else{
        res.sendStatus( code: 403);
    }

}
```

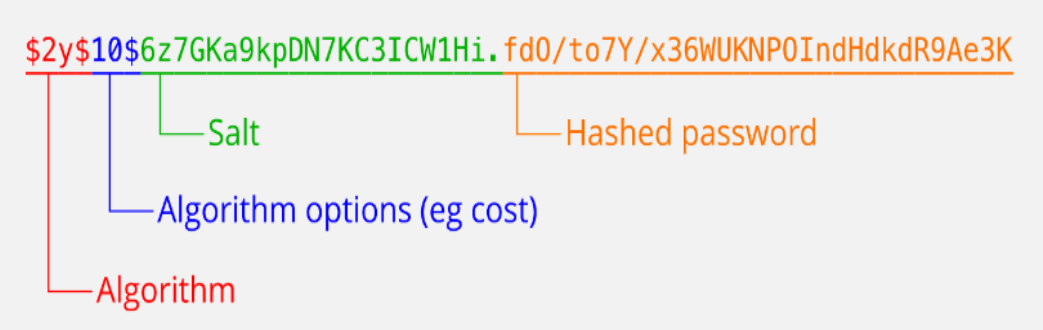
Obr. 3.3: Zabezpečení endpointů

3.2.7 Zvolené backendové knihovny

Nedílnou součástí příprav, během které byl navrhován backend, bylo zvolení vhodných knihoven, které pomohly v realizaci projektu a celkově přispěly k zjednodušení vývoje a časové úspory. Balíčky jsou různého druhu, počínaje od autorizace, přes parser HTTP hlaviček, až například po knihovnu pro šifrování hesel.

3.2.7.1 Bcrypt

Bcrypt je knihovna, která slouží k bezpečnému hashování hesel. Hlavním účelem knihovny je chránit uživatelská hesla před například brute-force útoky, kdy se útočník snaží získat původní heslo zkoušením různých kombinací. Bcrypt vytváří tzv. hash z hesla, což je jednosměrná matematická funkce, která převádí heslo na unikátní a nečitelný řetězec znaků. Tento hash obsahuje informace o soli (náhodně generovaný řetězec), což zvyšuje odolnost proti útokům. Níže na obr. 3.4 je zobracena celková struktura hesla, které proběhlo fází šifrováním.



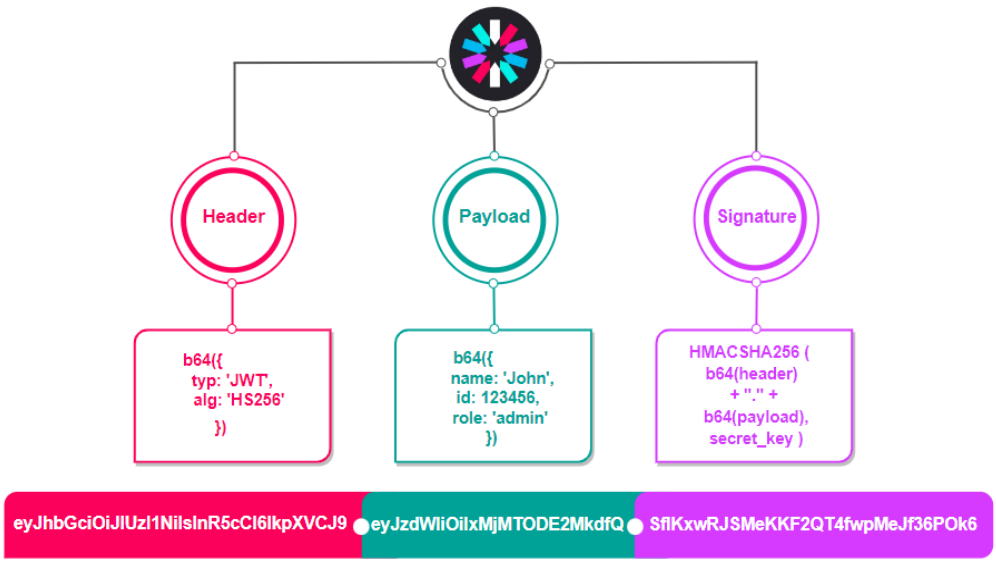
Obr. 3.4: Ukázka hesla po použití knihovny Bcrypt, zdroj: [4]

3.2.7.2 Jsonwebtokens (JWT)

„JSON Web Token (JWT) je otevřený standard (RFC 7519), který definuje kompaktní a autonomní způsob bezpečného přenosu informací mezi stranami ve formě JSON objektu. Tato informace může být ověřena a důvěryhodná, protože je digitálně podepsaná. JWT může být podepsán pomocí tajného klíče (algoritmem HMAC) nebo pomocí páru veřejný/privátní klíč s využitím RSA nebo ECDSA.“ [5]. Celá struktura tokenu vyplývá z obr. 3.5.

JWT se skládá ze tří částí:

- **Header** (Hlavička): Obsahuje informace o typu tokenu a použitém šifrovacím algoritmu.
- **Payload** (Náplň): Obsahuje uživatelská data nebo informace o oprávnění.
- **Signature** (Podpis): „Podpis slouží k ověření, že zpráva nebyla po cestě změněna, a v případě tokenů podepsaných soukromým klíčem může také ověřit, že odesílatel JWT je tím, za koho se vydává.“ [5]



Obr. 3.5: Struktura JWT tokenu, zdroj: [6]

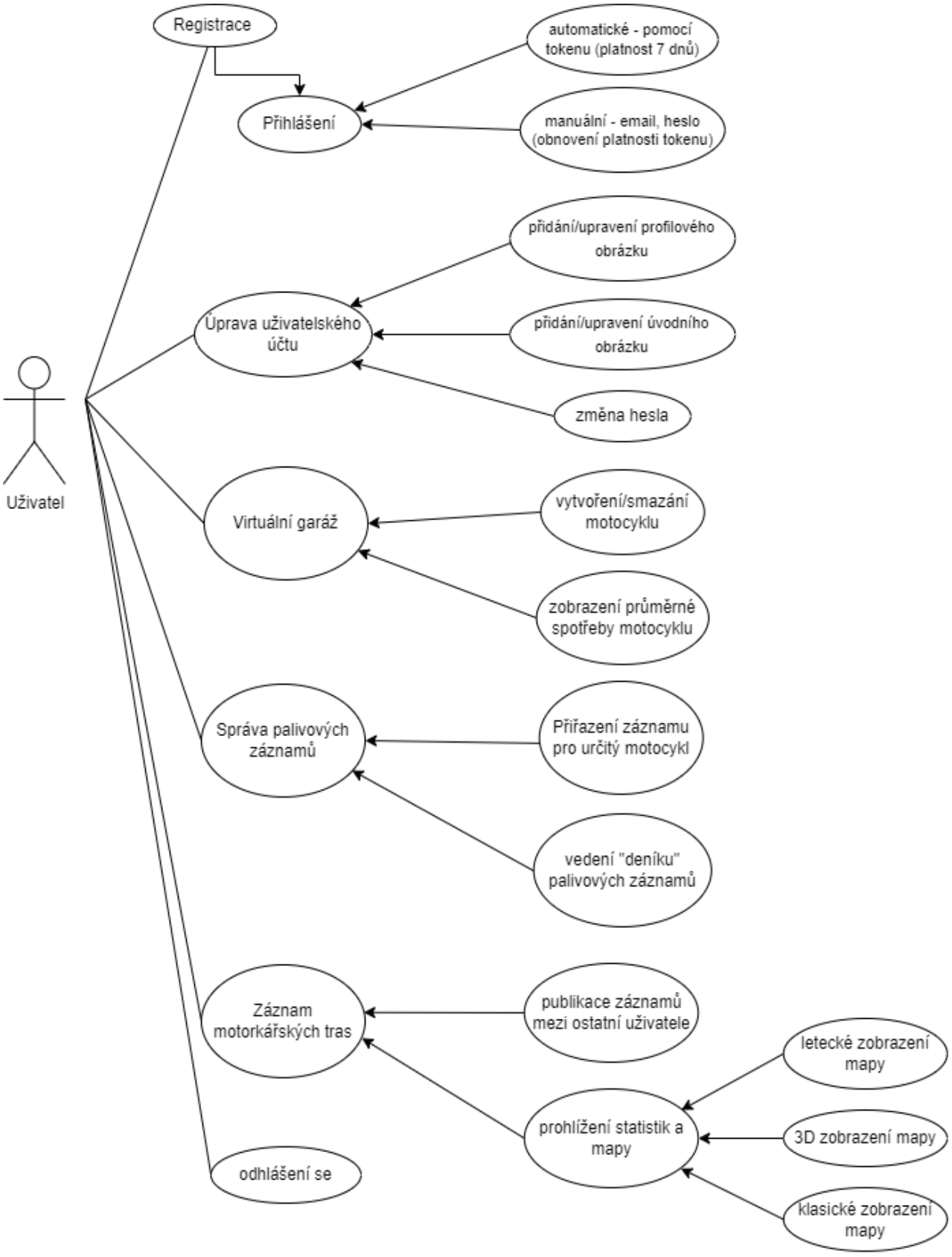
3.2.7.3 Mongoose

Mongoose je knihovna pro modelování objektových dat pro Node.js a MongoDB. Jedná se o nástroj navržený pro snadnou a intuitivní práci s MongoDB. Mongoose poskytuje abstrakci nad MongoDB, což umožňuje vývojářům definovat schéma dat, vytvářet dotazy a manipulovat s databází pomocí objektově orientovaných modelů.

3.3 Mobilní aplikace

V rámci projektu byla navržena a zrealizována multiplatformní mobilní aplikace RidersTrack, která je navržena s ohledem na potřeby motorkářů, kteří chtějí sledovat a sdílet své motorkářské jízdy. Aplikace nabízí unikátní možnost zaznamenávat motorkářské trasy a umožnit uživatelům detailní analyzování své jízdy prostřednictvím maximální rychlosti, doby trvání a ujeté vzdálenosti. Trasy jsou přehledně a interaktivně zobrazeny na Google mapách v 3D, leteckém nebo klasickém zobrazení, což zvyšuje atraktivitu záznamů pro uživatele. Aplikace obsahuje také správu pohonných hmot a přidávání neomezeného množství motocyklů do virtuální garáže.

Na obr. 3.6 je znázorněn Use case diagram samotné mobilní aplikace.



Obr. 3.6: Use case diagram mobilní aplikace

Aplikace byla rozdělena na jednotlivé důležité části, se kterými může uživatel interagovat. Níže jsou tyto části podrobně popsány.

3.3.1 Záznam motorkářských tras

Hlavní funkcionalitou aplikace je záznam motorkářských tras, který uživatel spustí přímo z hlavní obrazovky pomocí výrazného tlačítka. Zaznamenávání polohy uživatele probíhá i v případě minimalizace aplikace nebo při otevření jiné aplikace. Je možné zaznamenávat polohu i při vypnutém displeji. Tato funkčnost vyžadovala implementaci několika různých řešení. Hlavním problémem bylo zajistit, aby tato funkce fungovala spolehlivě na obou mobilních platformách (Android, iOS). Implementace pro iOS byla zároveň o něco složitější, jelikož vyžadovala důkladné přidání různých oprávnění a podrobné popsání, proč daná oprávnění aplikace potřebuje. Ukázka obrazovky, která obsahuje zaznamenanou trasu uživatelovy polohy je vidět na obr. 3.8 a obr. 3.9.

Získávání polohy uživatele probíhá každých 5 sekund a nevyžaduje připojení mobilního telefonu k internetu. Tato časová perioda byla pečlivě zvolena tak, aby zajistila dostatečně detailní záznam trasy a zároveň minimalizovala energetickou náročnost.

Na obr. 3.7 je vidět struktura každého bodu na mapě, který obsahuje podrobné informace o uživatelově poloze a pohybu.

```
Position newPoint = Position(
    longitude: position.longitude,
    latitude: position.latitude,
    timestamp: position.timestamp,
    accuracy: position.accuracy,
    altitude: position.altitude,
    altitudeAccuracy: position.altitudeAccuracy,
    heading: position.heading,
    headingAccuracy: position.headingAccuracy,
    speed: position.speed,
    speedAccuracy: position.speedAccuracy);

locationPoints.add(newPoint);
```

Obr. 3.7: Struktura GPS bodu



Obr. 3.8: Zobrazení mapy v 3D režimu

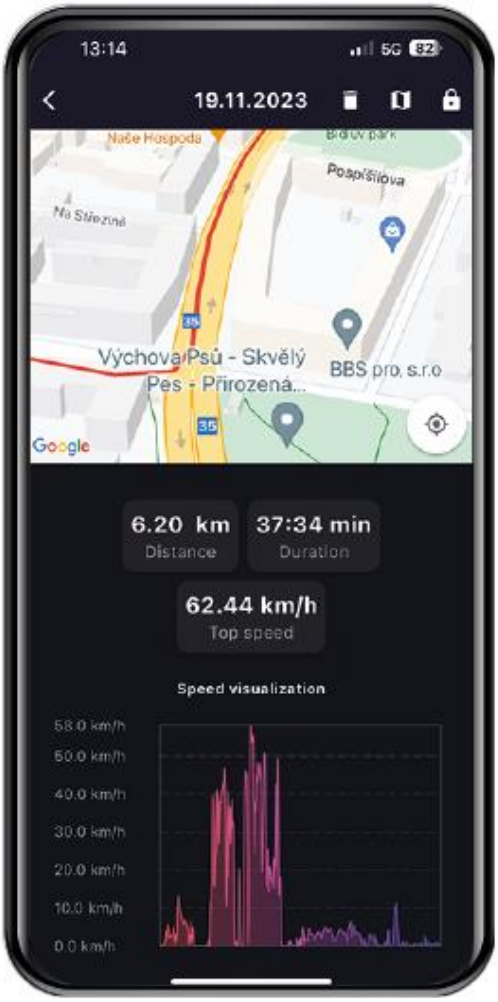


Obr. 3.9: Zobrazení mapy v leteckém režimu

3.3.2 Detail motorkářských záznamů

Detail motorkářských záznamů obsahuje také podrobné zobrazení statistik, které byly během jízdy zaznamenány. Patří sem celková vzdálenost, doba trvání nebo třeba maximální rychlost. Nesmí chybět také graf, který zobrazuje celkový vývoj rychlosti. Pro uživatele je zajímavé to, že graf je interaktivní, tedy že po něm může uživatel prstem přejíždět a aplikace na mapě zobrazí nový bod v místě, kde se danou rychlostí pohyboval. (obr. 3.10)

Tyto údaje jsou pro motorkáře atraktivní a mají za důsledek zlepšení požitku z dané projížďky.

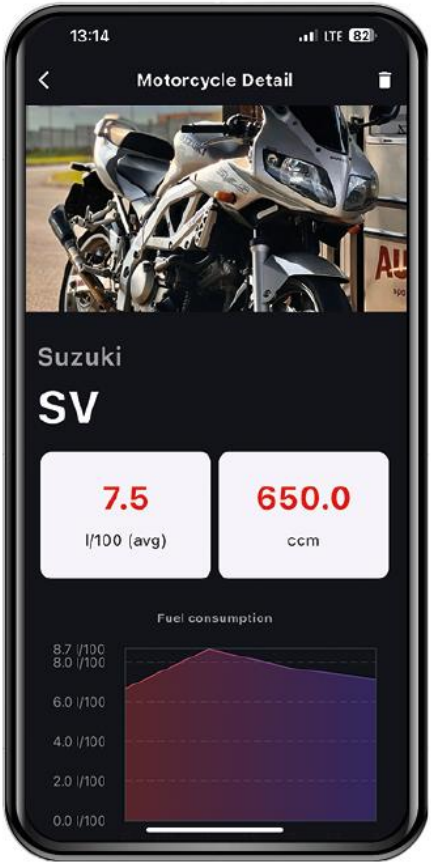


Obr. 3.10: Detailní zobrazení statistik

3.3.3 Virtuální garáž

Aplikace obsahuje funkci přidávání neomezeného množství motocyklů do virtuální garáže, kde se pro každý motocykl počítá průměrná spotřeba z přiřazených údajů. Obrazovka se seznamem palivových záznamů je zobrazena na obr. 3.12.

Pro zlepšení srozumitelnosti a pro přehlednost jsou hodnoty průměrné spotřeby z jednotlivých jízd prezentovány v grafu, který je přístupný uživateli po přechodu na obrazovku s detailem motocyklu. Jak je vidět na obr. 3.11, bylo pečlivě dbáno na to, aby hodnoty byly pro uživatele co nejvíce přehledné a snadno srozumitelné.

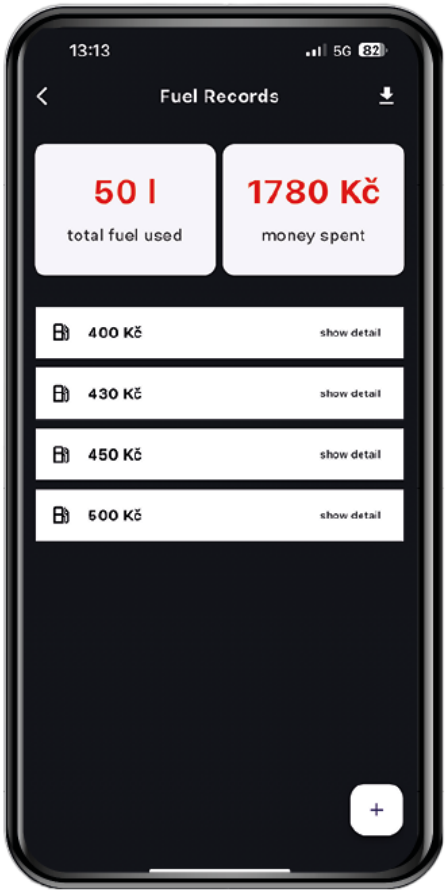


Obr. 3.11: Detail motocyklu – zobrazení průměrné spotřeby

3.3.4 Správa pohonných hmot

V rámci aplikace si uživatel může vytvářet svůj osobní „deník“ palivových záznamů, což je klíčová funkce pro ty, kteří chtějí sledovat spotřebu paliva a náklady spojené s pohonnými hmotami. Funkce osobního deníku umožňuje uživatelům získat přehled o celkové spotřebě paliva a nákladech za pohonné hmoty. Každý záznam je nutné přiřadit konkrétnímu motocyklu, který je součástí virtuální garáže. Jednotlivé záznamy si uživatel může rozkliknout a zobrazit si tak detailní zobrazení.

Bohužel není motorkářský průmysl rozvinut natolik, aby uživatelům umožnil připojení na řídící jednotku motocyklu. Původní plán totiž spočíval v tom, že tyto údaje bude uživateli zasílat motorka samotná a tím by bylo docíleno maximální přesnosti a zároveň jednoduchosti v použití pro uživatele. Na tuto funkcionalitu je bohužel potřeba počkat, až to budou samotné motocykly umět.



Obr. 3.12: Správa pohonných hmot

3.3.5 Stopky

Při zaznamenávání jízdy je na obrazovce uživateli zobrazena aktuální doba trvání záznamu, která se každou vteřinu pravidelně aktualizuje. Tím je uživatel vždy informován, že záznam probíhá. Důležité je zajištění funkčnosti i v případě, kdy je aplikace minimalizována, což představovalo následující možná řešení.

Prvním řešením bylo implementovat úlohy v nativních jazycích, konkrétně Swift/Objektové C pro iOS a Java pro Android. Toto řešení by sice zajistilo správný běh stopek na pozadí, avšak nese s sebou vyšší časovou náročnost kvůli nutnosti práce v nativních prostředích.

Druhým řešením, které se ukázalo jako vhodnější pro daný kontext, bylo reagování na změnu stavu aplikace. V případě, že uživatel minimalizuje aplikaci (životní stav aplikace se změní na „paused“), jsou stopky zastaveny a aktuální čas spolu s počtem zaznamenaných vteřin jsou uloženy do paměti mobilního telefonu. Když uživatel znovu spustí záznam trasy, aplikace si získá data, která byla uložená v paměti a vypočítá rozdíl mezi aktuálním časem a časem, kdy byl záznam přerušen. Tento časový rozdíl je následně přičten k předchozímu součtu vteřin. Toto řešení přináší efektivnější a rychlejší implementaci s nižší časovou náročností.

3.4 Webová stránka

Kromě implementace mobilní aplikace spolu s backendovou částí bylo v rámci projektu také vytvoření a uvedení do provozu webové stránky, která plní klíčovou roli jako informativní nástroj pro mobilní aplikaci. Zde mohou návštěvníci snadno získat přehled o výhodách, které aplikace přináší, a navíc si zde mohou stáhnout testovací verzi.

Webová stránka běží na serveru Apache, kde je dále nakonfigurována CRON úloha. Tato úloha pravidelně, jednou za měsíc, pomocí certifikačního bota obnovuje SSL certifikát, čímž zajišťuje bezpečné připojení a důvěryhodnost webové stránky.

4 ZÁVĚR

Mobilní aplikace byla úspěšně naimplementována a celý systém je již nasazen na virtuálním privátním serveru. V současné době probíhá fáze testování aplikace mezi motorkářskou komunitou, která poskytuje cennou zpětnou vazbu a umožní včasné identifikování potenciálních problémů.

Zaměřil jsem se nejen na současnost, ale také na budoucnost projektu. Klíčovým prvkem byla škálovatelnost a otevřenost pro další rozvoj. V blízké době plánuji integrovat nové funkcionality do mobilní aplikace, které uživatelům poskytnou ještě bohatší a interaktivnější zážitek. Jedním z cílů je rozšíření aplikace, aby se z ní stala menší sociální síť pro motorkáře. V plánu je vytvořit prostředí, kde si uživatelé budou moci sdílet své zážitky, tipy na výlety a zajímavé trasy. Nové funkce zahrnují také možnost přidávání reakcí a komentářů na jednotlivé záznamy, což pomůže dosáhnout ještě většího propojení a interakce mezi uživateli.

Závěrem lze říct, že se úspěšně povedlo navrhnout a naimplementovat celý systém pro motorkářskou komunitu a zároveň mám již navrženou další práci na projektu. Věřím, že jsem na správné cestě k vytvoření inovativní platformy pro motorkáře, která nejen uspokojí jejich potřeby, ale také poskytne prostředí pro sdílení zážitků a budování motorkářské komunity.

Citovaná literatura

[1] „Slovník IT pojmů: Framework,“ MasterDC, [Online]. Available: https://www.master.cz/help/slovnik/framework/. [Přístup získán 23 Únor 2024].

[2] „Co jsou to NoSQL databáze?,“ IBM, [Online]. Available: https://www.ibm.com/topics/nosql-databases. [Přístup získán 4 Únor 2024].

[3] „FAQ: Bezpečné hashování hesel,“ php, [Online]. Available: https://www.php.net/manual/en/faq.passwords.php. [Přístup získán 4 Březen 2024].

[4] „Úvod do Json Web Tokenů,“ JWT, [Online]. Available: https://jwt.io/introduction. [Přístup získán 4 Březen 2024].

[5] P. Halani, „Úvod do MVCS architektury,“ pvha, 7 Září 2021. [Online]. Available: https://pvha.hashnode.dev/mvcs-architecture. [Přístup získán 4 Březen 2024].

[6] D. Mishra, „Bezpečnost: Dekódování JWT tokenu a ověření pomocí Javy,“ Metamug, 02 Říjen 2021. [Online]. Available: https://metamug.com/article/security/decode-jwt-java.html. [Přístup získán 4 Březen 2024].

Seznam obrázků

Obr. 3.1: Architektura systému 6
Obr. 3.2: Architektura MVC+S, zdroj: [3] 8
Obr. 3.3: Zabezpečení endpointů 12
Obr. 3.4: Ukázka hesla po použití knihovny Bcrypt, zdroj: [4] 13
Obr. 3.5: Struktura JWT tokenu, zdroj: [6] 14
Obr. 3.6: Use case diagram mobilní aplikace 15
Obr. 3.7: Struktura GPS bodu 16
Obr. 3.8: Zobrazení mapy v 3D režimu 17
Obr. 3.9: Zobrazení mapy v leteckém režimu 17
Obr. 3.10: Detailní zobrazení statistik 18
Obr. 3.11: Detail motocyklu – zobrazení průměrné spotřeby 19
Obr. 3.12: Správa pohonných hmot 20

Seznam tabulek

Tabulka 1: Parametry endpointu pro vytvoření záznamu jízdy 10
Tabulka 2: parametry endpointu pro kontrolu verze aplikace 11
Tabulka 3: Parametry endpointu pro záznam palivových záznamů 12

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	Matěj Mlynář
Školní rok:	2023/2024
Třída:	4.B
Obor:	Informační technologie 18-20-M/01
Téma práce:	Rider's Track: aplikace pro záznam motorkářských tras a statistik
Vedoucí práce:	Bc. Vlaďka Janů

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem projektu je vytvořit motorkářskou mobilní aplikaci, která umožní měření statistik a zaznamenávání tras na mapě během motocyklových výletů. Aplikace dále nabídne možnost vytvoření jednoho nebo více motocyklů, ke kterým bude možnost přiřadit a zaznamenávat motorkářské trasy, své změřené statistiky a bude poskytovat přehled o tankování paliva (včetně průměrné spotřeby a nákladů na pohonné hmoty).

Příklad použití aplikace:
Uživatel před svou jízdou zapne aplikaci a spustí měření. Aplikace následně bude měřit a zaznamenávat celkové statistiky včetně trasy, po které se uživatel pohyboval.

Dále vznikne jednoduchá webová stránka, která bude sloužit jako „reklama" a informační zdroj pro tuto aplikaci.

Projekt bude mít reálné využití. Inspirace z aplikace Super Biker, kterou již aktivně využívám, ale pár věcí mi na ní chybí a vadí.

Základní použité technologie:

- Flutter
- Dart
- React
- Firebase nebo jiná databáze

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- Dokončení základního návrhu a designu stránky a mobilní aplikace (září)
- Práce na vývoji webové stránky a mobilní aplikace (říjen - listopad)
- Dokončení funkčního prototypu webové stránky a mobilní aplikace (prosinec)
- Testování, ladění aplikace a chyb (leden - únor)
- Práce na dokumentaci (leden - březen)