DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, Pardubice

BikeCheck

Příjmení, jméno: Brychta Lukáš

Třída: 4. B

Studijní obor: Informační technologie 18-20-M/01

Školní rok: 2024/2025

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení: *Lukáš Brychta*

Školní rok: *2024/2025*

Třída: 4. B

Obor: *Informační technologie 18-20-M/01*

Téma práce: Mobilní aplikace sledující opotřebení jízdního kola

Vedoucí práce: *Ing. Monika Borkovcová, Ph.D.*

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

Cílem projektu je vytvoření mobilní aplikace pro OS Android, která bude zaznamenávat a spravovat opotřebení MTB komponent a podle toho doporučovat uživateli, kdy je vyměnit nebo zajít do servisu.

Popis funkčnosti mobilní aplikace, která umožňuje:

- Přihlášený uživatel může spravovat svá kola a komponenty.
- Aplikace bude umožňovat:
    - práci s uživatelskými daty, kdy bude aplikace ukládat informace o kolech a komponentách (opotřebení, data o servisech, najetých km a hodin apod.),
    - propojit účet v plánované aplikaci s účtem Strava, což je aplikace na měření jízd,
        - Aplikace ze Strava API bude sbírat data o jízdě jako je délka a čas a podle toho vypočítávat životnost a servisní intervaly komponent, což ulehčí práci uživateli, který nebude muset tyto informace zadávat ručně.
    - notifikační systém pro upozorňování uživatelů na potřebu servisu nebo výměny komponentů,
    - nastavení uživatelských preferencí a personalizace prostředí aplikace,
    - podporu offline režimu a synchronizace dat mezi zařízeními,
    - optimalizaci pro různé typy zařízení.
    -

Aplikace bude splňovat bezpečnostní opatření a zabezpečení dat, včetně autentifikace, autorizace a šifrování datových přenosů. Všechna data se budou měnit v závislosti na datech o jízdách, které uživatel může zadat ručně nebo automatizovaně pokud má účet v aplikaci Strava a zaznamenává si jízdy.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

Září: Analýza existujících mobilních aplikací v oblasti daného tématu a sběr požadavků od potenciálních uživatelů. Návrh architektury a rozhraní mobilní aplikace.

Říjen–Leden: Implementace a vývoj mobilní aplikace s důrazem na plnění definovaných požadavků. Testování a ladění funkcionality aplikace.

Leden: Zahájení zpracování dokumentace aplikace a teoretické části práce.

Únor–Březen: Dokončení zpracování dokumentace a teoretické části práce. Dolaďování a oprava chyb v aplikaci na základě testování. Příprava na prezentaci a obhajobu práce.

Prohlášení

Prohlašuji, že jsem maturitní projekt vypracoval samostatně, výhradně s použitím uvedené literatury.

V Pardubicích dne 31. 3. 2025

_____

(vlastnoruční podpis)

Poděkování

Upřímně děkuji Ing. Monice Borkovcové, Ph.D., za odborné vedení, cenné rady a poskytnutou podporu při zpracovávání maturitního projektu.

Anotace

Cílem projektu BikeCheck je vytvoření mobilní aplikace pro OS Android, která bude zaznamenávat a spravovat opotřebení kola a poskytuje tak uživatelům možnost monitorování a správu cyklistických komponent na základě dat z aplikace Strava. Hlavní zaměření je na synchronizaci a správu kol, a jejich jednotlivé části, s využitím dat o ujetých vzdálenostech.

Tento dokument popisuje vývoj této aplikace a celého projektu, zmiňuje technologický stack projektu, jako je použití Flutteru pro frontend, a popisuje, jak aplikace komunikuje mezi frontendovou a backendovou části aplikace. Dále jsou uvedeny specifické funkce aplikace, jako je sledování opotřebení komponent a historie jízd. Celkově dokumentace poskytuje komplexní přehled o vývoji aplikace BikeCheck, technologických volbách a implementaci funkcí zaměřených na efektivní správu cyklistického vybavení.

Klíčová slova

mobilní aplikace, flutter, cyklistika

Anotation

The aim of the BikeCheck project is to create a mobile application for the Android OS that will record and manage bicycle wear and tear, providing users with the ability to monitor and manage cycling components based on data from the Strava app. The main focus is on the synchronization and management of bicycles and their individual parts, utilizing data on distances traveled.

This document describes the development of this application and the entire project, mentioning the project's technology stack, such as the use of Flutter for the frontend, and describes how the application communicates between the frontend and backend parts of the application. Furthermore, the document highlights specific features of the application, such as tracking component wear and ride history. Overall, the documentation provides a comprehensive overview of the development of the BikeCheck application, technological choices, and the implementation of features aimed at effective management of cycling equipment.

Keywords

mobile application, flutter, cycling

Obsah

Obsah 6
Seznam zkratek, zkratkových slov a pojmů 8
Úvod 9
1 Existující řešení 10
1.1 Srovnání aplikací 10
2 Technologie 12
2.1 Node.js 12
2.2 Express.js 12
2.3 Crypto 12
2.4 Flutter 12
2.5 OAuth 0.2 12
2.6 PostgreSQL 13
2.7 Sequelize 13
3 BikeCheck 14
3.1 Inspirace 14
3.2 Požadavky 14
3.3 Aplikace 15
3.4 Architektura 15
3.5 Backend 16
3.5.1 Autorizace a zabezpečení 17
3.5.2 AES Šifrování 17
3.5.3 PostgreSQL 18
3.5.4 Synchronizace 19
3.6 Frontend 19
3.6.1 Autorizace 19
3.6.2 Struktura aplikace 19
Závěr 23
Literatura 24
Seznam obrázků 26

Příloha 1 – API Endpoint dokumentace 27
Activities – nepoužívá se 27
Bikes 28
Components 29
Service_Intervals 31
Strava 32
User 33

Seznam zkratek, zkratkových slov a pojmů

API – (Application programming interface) Rozhraní umožňující komunikaci mezi různými aplikacemi.

Backend – Serverová část aplikace, která zpracovává data a logiku.

Frontend – Uživatelské rozhraní aplikace, se kterým pracuje uživatel.

Databáze – Systém pro ukládání a správu dat.

Framework – Sada nástrojů a knihoven usnadňující vývoj aplikací.

JSON – (JavaScript Object Notation) Formát pro výměnu dat, čitelný pro lidi i stroje.

Widget – Grafická komponenta v uživatelském rozhraní (např. tlačítko, textové pole).

Open Source – Software s přístupným zdrojovým kódem, volně dostupný k úpravám.

ORM – Nástroj pro práci s databází pomocí objektů místo SQL dotazů.

Strava – Populární sportovní aplikace pro sledování aktivit, zejména cyklistiky a běhu.

Úvod

Cyklistika je velmi oblíbený sport, který se stále rozrůstá. Kromě sportovního využití slouží kolo i jako nenákladný a ekologický dopravní prostředek. Mnoho běžných cyklistů, kteří využívají kolo pouze k přepravě z bodu A do bodu B údržbu a servis svých kol neřeší, nebo ji odkládá na poslední chvíli. Neudržovaná kola pak mohou být nejen neefektivní, ale i nebezpečná v provozu.

Jako vášnivý cyklista, který pravidelně provádí základní servis svého kola, jsem si vědom, že některé servisní úkony vyžadují odborné znalosti a specializované nástroje, a je proto vhodné svěřit je profesionálnímu mechanikovi. Klíčovým faktorem úspěšné údržby je však její včasnost, tzn. pravidelný servis předchází vážnějším problémům a potenciálnímu selhání komponent v kritických situacích. Většina cyklistických komponent má jasně definované servisní intervaly založené na najetých kilometrech nebo hodinách používání. Například řetěz vyžaduje výměnu po 1500-3000 km (v závislosti na typu a podmínkách), brzdové destičky po 500-1500 km, a odpružené vidlice potřebují servis po 50-200 hodinách jízdy.

Sledování těchto intervalů představuje pro běžného cyklistu výzvu, která vyžaduje pečlivé zaznamenávání najetých kilometrů a času pro každou komponentu, což je bez vhodných nástrojů obtížné a časově náročné. Právě zde přichází na řadu specializované většinou mobilní aplikace pro správu cyklistického vybavení.

Cílem tohoto projektu je vytvoření aplikace BikeCheck, která má za úkol pomáhat cyklistům s evidencí opotřebení jejich kol, kdy po přihlášení přes Strava účet získá BikeCheck přístup k datům tohoto účtu. Strava je aplikace pro zaznamenávání sportovních aktivit jako jízda na kole, běh, inline bruslení a zaznamenává ujeté kilometry, jak dlouho aktivita trvala, GPS lokaci a více. Dále umožňuje uživateli přidávat kola a zaznamenávat aktivity k těmto kolům. Na základě získaných dat pak BikeCheck vypočítá opotřebení jednotlivých komponent a upozorní uživatele, když se blíží čas na jejich výměnu nebo servis.

1 Existující řešení

Tato kapitola popisuje podobná řešení, na které se tento projekt soustředí:

BikeManager – BikeManager je aplikace vhodná pro uživatele, kteří chtějí detailně spravovat své komponenty a nevadí jim ruční zadávání. Nabízí možnost manuálního zadání komponenty s vlastním názvem, značkou a dalšími detaily, jako je cena, obchod a komentář. Upozornění na servis lze nastavit podle kilometrů, času nebo periodicky po zvoleném intervalu. Nevýhodou je nepohodlný proces editace, kdy je nutné projít celý formulář i při změně jediné položky. [2]

ProBikeGarage – ProBikeGarage je zaměřený na rychlé a efektivní přidávání komponent. Pokud uživatel vlastní kolo známější značky, aplikace nabízí předpřipravené modely s továrními komponentami, které lze ihned použít nebo upravit. Přidání komponent je jednodušší než u BikeManageru a lze vybírat mezi základními typy nebo značkovými díly s doporučenými servisními intervaly. Uživatelé mohou sledovat své jízdy a zaznamenávat návštěvy servisu. [12]

Maintrack – Maintrack je nejpropracovanější aplikace z těchto tří a nabízí široké možnosti správy kol, komponent a servisu. Umožňuje plánování servisu a automaticky dopočítává další servisní intervaly podle průměrného používání kola. Přestože aplikace nabízí nejvíce funkcí, může být na začátku složitější na orientaci, protože propojení mezi komponentami, servisem a aktivitami je rozsáhlé. Pokud si však uživatel zvykne, získá velmi efektivní nástroj pro komplexní správu svého vybavení. Oproti předchozím aplikacím je dostupná pouze na Apple. [8]

1.1 Srovnání aplikací

Uživatelské rozhraní:

- ProBikeGarage nabízí moderní a intuitivní rozhraní, které bylo v roce 2025 kompletně přepracováno.
- Maintrack je popisován jako intuitivní s krásným designem rozhraní.
- BikeManager má funkční, ale méně pohodlné rozhraní, zejména při editaci komponent.

Integrace se Stravou:

- ProBikeGarage poskytuje bezproblémovou integraci se Stravou s automatickými aktualizacemi po každé jízdě.
- Maintrack také nabízí synchronizaci se Stravou, ale někteří uživatelé zmiňují omezení v mapování kol.
- BikeManager spoléhá více na ruční zadávání.

Uživatelská základna a hodnocení:

- ProBikeGarage má více než 100 000 stažení a hodnocení 4,8 hvězdiček na Google Play
- Maintrack má smíšené recenze, ale mnoho uživatelů oceňuje integraci se Stravou
- BikeManager nemá dostupné údaje o počtu stažení nebo hodnocení

Dostupnost:

- ProBikeGarage je dostupný pro iOS i Android
- Maintrack je dostupný pouze pro iOS zařízení
- BikeManager nemá specifikovanou platformovou dostupnost

Podle recenzí uživatelů je ProBikeGarage obzvláště užitečný pro sledování opotřebení řetězu a intervalů voskování, stejně jako pro sledování kol, pneumatik, kazet a dalších položek, které se přesouvají mezi koly. Jeden uživatel dokonce uvedl, že díky používání aplikace snížil opotřebení řetězu na polovinu.

2 Technologie

Tato sekce se zaměřuje na technologie, které byly využity při vývoji. Každá z vybraných technologií hraje klíčovou roli v realizaci funkčnosti aplikace.

2.1 Node.js

Node.js je JavaScript runtime umožňující spuštění JavaScriptu mimo prohlížeč. Oblíbený pro svou výkonnost a asynchronní programování. Ideální pro aplikace, které potřebují rychlý přístup k databázi nebo API. S použitím express.js frameworku je to populární stack. Node.js je backend aplikace BikeCheck, který zřizuje komunikaci s databází a Strava API. Zároveň se chová jako API pro frontend requesty. [1, 10]

2.2 Express.js

Express.js je jednoduchý, flexibilní framework pro vytváření Node.js webových aplikací a API. Poskytuje základní nástroje a middleware pro zpracování http požadavků, směrování, a zjednodušení práce s odpověďmi a požadavky. Express.js v backendu zajišťuje správu API, definuje směrování pro požadavky a zajišťuje komunikaci mezi frontendem a backendem. [4]

2.3 Crypto

Crypto je nativní modul Node.js pro kryptografické operace jako je hashování, šifrování, dešifrování a generování random sekvencí bitů. BikeCheck využívá Crypto pro šifraci a dešifraci access tokenů. Společně s Node.js nativní podporou typuBuffer propráci s binárními daty, je perfektní volbou pro šifrování a dešifrování.[9, 3, 20]

2.4 Flutter

Flutter je cross-platform framework od Google. Slouží pro vývoj mobilních aplikací. Díky flutteru odpadá nutnost vyvíjet aplikaci zvlášť pro Android a pro IOS. Flutter poskytuje spoustu předdefinovaných widgetů k použití v UI. Flutter pro vykreslování používá vlastné engine, což zajišťuje výkonnost. [7] Flutter používá stavy. Widgety ve fluttru mohou být se stavem nebo bez něj. Widgety beze stavu se používají pro jednoduché statické UI, zatímco Widgety se stavem tento stav můžou měnit a na základě toho překreslovat widgety. [15, 5]

2.5 OAuth 0.2

OAuth je standardní protokol pro autorizaci. Povoluje stránkám a aplikacím po povolení od uživatele využívat data a materiály jiných webů a aplikací bez nutnosti využívat přihlašovací údaje. Data jsou aplikacím zpřístupněna pouze s access tokenem, který získají po autorizaci. Tokeny také mohou mít rozsah, který určuje, co vše může autorizovaná aplikace/web dělat. [11, 21]

2.6 PostgreSQL

PostgreSQL je výkonná open-source relační databáze. Je známá svou škálovatelností a stabilitou. Používá se v mnoha kritických aplikacích, od malých projektů po velké podnikové systémy. [22]

2.7 Sequelize

Sequelize je moderní Node.js ORM (object relational mapping), který umožňuje vývojářům pracovat s různými databázemi, jako jsou Oracle, PostgreSQL, MySQL, MariaDB a SQLite. Umožňuje psát dotazy na databázi v JavaScritpu nebo TypeScriptu, podporuje asociace a dodává k nim předdefinované metody. Není tedy nutné psát SQL dotazy. [13, 14]

3 BikeCheck

3.1 Inspirace

Cyklistika je mnoha lidmi vnímána nejen jako sport, ale i jako vášeň a životní styl. Pro ty, jimiž je jí věnována plná pozornost, ať už na trailech, v bikeparcích nebo na silnici, je považováno za důležité, aby jejich kolo bylo vždy udržováno v perfektním stavu. Správnou údržbou kola je výrazně přispíváno k bezpečnosti, pohodlí i celkové životnosti jednotlivých komponent. Přestože pravidelným servisem je zajišťována klíčová péče, mnoha cyklisty není udržován přesný přehled o tom, kolik toho jejich díly skutečně vydržely a kdy je tím správným časem na výměnu nebo údržbu.

O mé kolo je pečováno s maximální pozorností a jeho perfektní kondice je vyžadována za všech okolností, zejména před návštěvami bikeparků nebo náročných trailů. V dřívější době nebyl při jízdách používán tachometr a záznamy o nich nebyly systematicky vedeny. Tímto přístupem byla zapříčiněna absence přesných dat o kilometráži jednotlivých komponent, čímž bylo značně komplikováno rozhodování o jejich včasné výměně či servisu.

Postupně bylo zjištěno, že mezi cyklistickou komunitou je vysoce ceněna aplikace Strava, kterou je nabízena možnost sledování ujetých tras a výkonů. Kromě toho je jí poskytováno i API, jímž je umožňován přístup k datům o najetých kilometrech a používaném vybavení. Touto možností byla podnícena inspirace k vytvoření aplikace, jíž bude propojena Strava s podrobnějším sledováním opotřebení komponent. Cílem je, aby byly zaznamenávány ujeté kilometry u jednotlivých součástí kola, byly automaticky porovnávány s daty ze Stravy a bylo upozorňováno na jejich opotřebení nebo potřebu servisu. Díky tomu bude uživatelem získán jasný přehled o tom, kdy je považováno za ideální čas na výměnu řetězu, brzdových destiček nebo jiných klíčových dílů, čímž bude zlepšena nejen spolehlivost kola, ale i samotný zážitek z jízdy.

3.2 Požadavky

Na aplikaci jsou tedy kladené požadavky, aby splňovala svůj účel co nejefektivněji a poskytla uživateli všechny potřebné funkce. Aplikace má tedy za potřebí navodit jednoduché a intuitivní UI, které umožní snadnou orientaci a efektivní správu komponent. Dále také musí zařizovat plynulou komunikaci s backendem aby frontend získával správná data potřebná ke zobrazení.

Backend musí spravovat databázi, především pak bezpečné ukádání přístupových tokenů. Veškerá autnetizace musí probíhat přes Strava API, kde probíhá výměna tokenů.

3.3 Aplikace

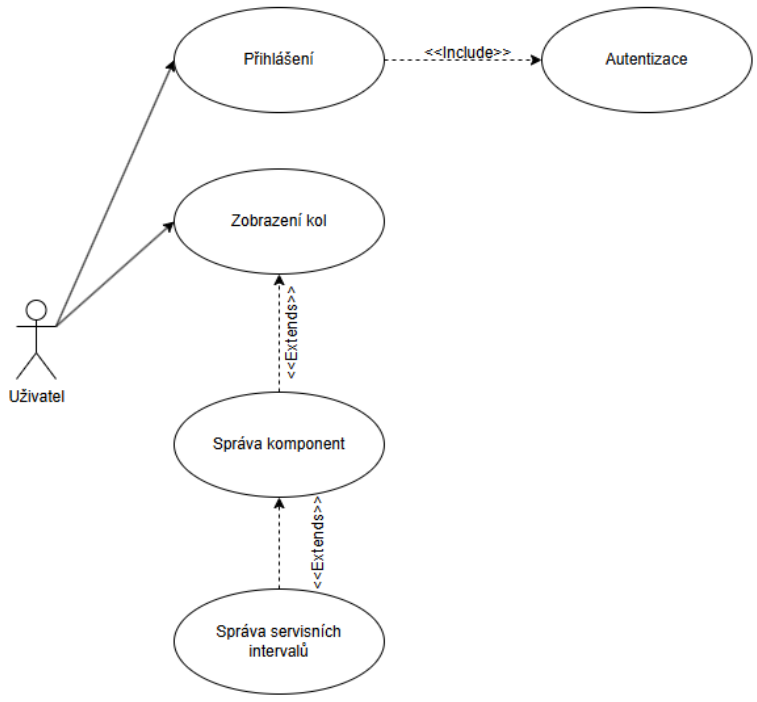
BikeCheck je aplikace pro cyklisty, která jim pomáhá sledovat stav jejich kol a komponent. Cílem aplikace je poskytnout uživatelům jednoduchý a efektivní způsob, jak udržovat své kolo v dobrém technickém stavu pomocí propojení se službou Strava.

Aplikace propojí uživatelův Strava účet a načte data o jeho kolech a ujetých vzdálenostech. Každá komponenta kola je zaznamenaná v databázi a při synchronizaci se sleduje její opotřebení na základě rozdílu mezi zaznamenanou vzdáleností v databázi BikeCheck a daty získanými ze Stravy.

Uživatelé mohou v aplikaci přidávat nové komponenty, kontrolovat jejich stav a získávat přehled o tom, kdy je třeba některé části vyměnit.

BikeCheck kombinuje moderní technologie s potřebami cyklistů a pomáhá jim efektivně spravovat jejich vybavení a prodloužit životnost jejich kol.

Na obrázku 1 můžeme vidět jaké akce může uživatel v aplikaci provádět.

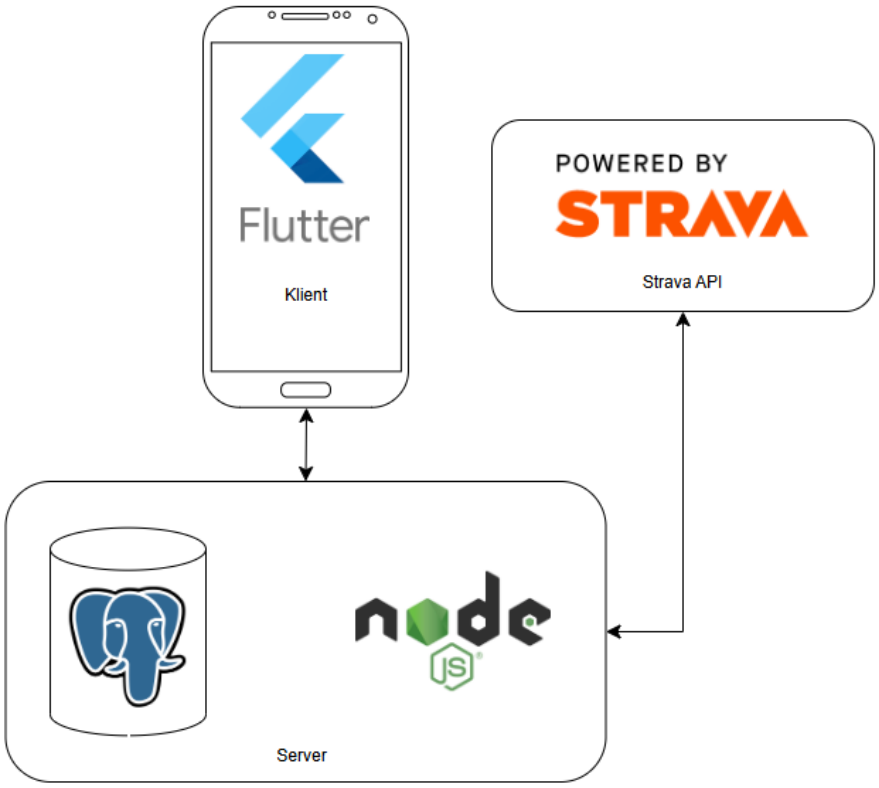


Obrázek 1 Use case diagram

3.4 Architektura

Cílem je tedy vytvořit aplikaci, která bude komunikovat se Strava API a backendem, ukládat data do databáze a výsledky backendu vykreslovat na frontendu.

Jak můžeme vidět na obrázku 2, vyplývají z toho 4 základní prvky, které mezi sebou komunikují



Obrázek 2 Architektura

Flutter klient: Zajišťuje vykreslování uživatelského rozhraní (UI) aplikace. Jedná se o mobilní aplikaci, která běží na zařízení uživatele. Flutter komunikuje s backendem, aby získal data potřebná pro zobrazení.

Strava API: Externí služba, která poskytuje data o aktivitách uživatelů (např. běh, cyklistika). Backend se připojuje k této API, aby získal požadované informace a předal je klientovi.

Node.js backend: Backend je hostován na serveru a slouží jako prostředník mezi klientem a Strava API. Zpracovává požadavky od klienta, komunikuje se Strava API pro získání dat a ukládá tato data do databáze.

PostgreSQL databáze: Databáze je součástí serveru a slouží k ukládání dat získaných ze Strava API. Backend přistupuje k databázi pro čtení nebo zápis dat.

3.5 Backend

BikeCheck backend je postaven na Node.js s Express.js frameworkem a využívá PostgreSQL jako databázi. Jeho hlavním úkolem je spravovat uživatelskou autentizaci, ukládat data o kolech a jejich komponentách a zajišťovat komunikaci se Strava API. Se

Strava API komunikuje přes endpointy uvedené na developerských stránkách Stravy. [16]. Pro zabezpečení tokenů se používá AES šifrování s 256bitovým klíčem.

3.5.1 Autorizace a zabezpečení

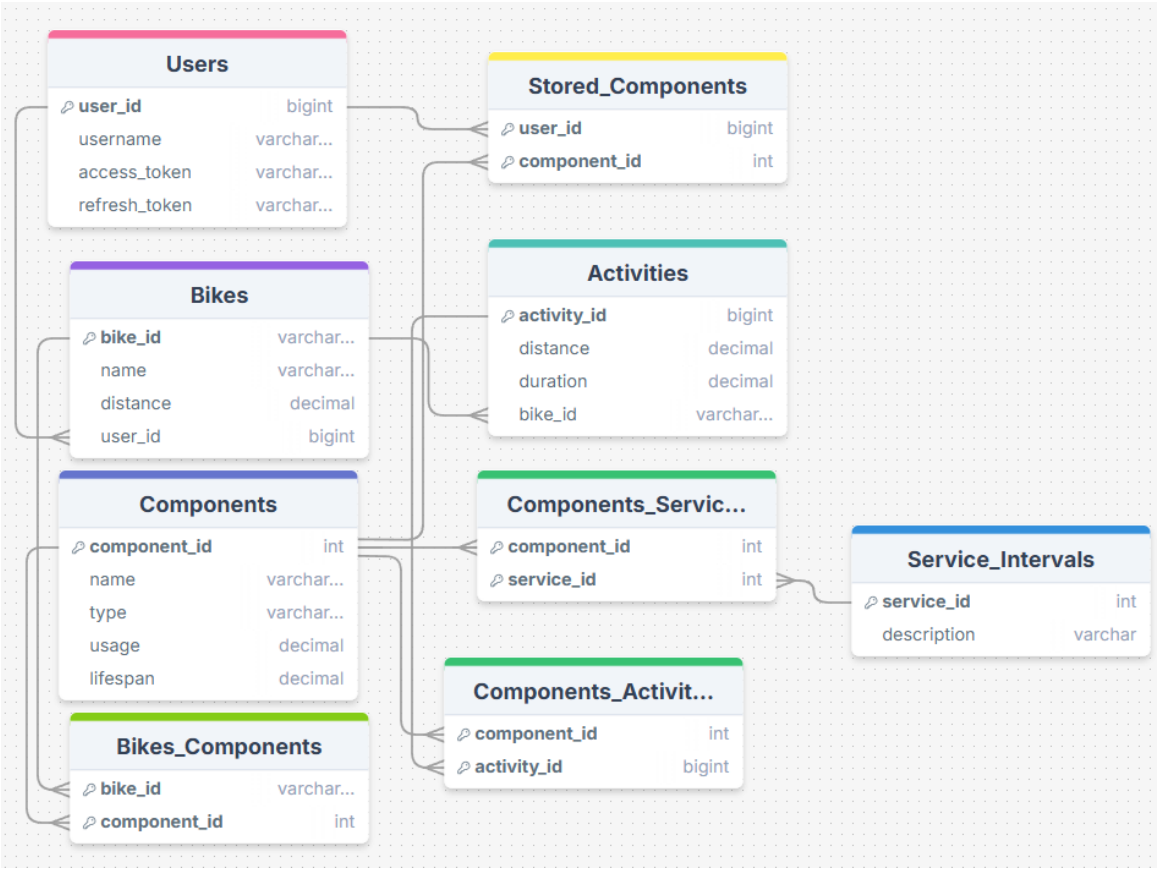
Backend u autorizace řeší výměnu autorizačního tokenu za dlouhodobější access tokan a refresh token. Probíhá to tak, že frontend obdrží autorizační kód díky Strava OAuth, který předá backendu. Backend následně tento kód pošle na Strava API endpoint, který nazpět pošle access a refresh token, přičemž oba tokeny jsou zašifrovány a uloženy do databáze. Poté backend vznese požadavek na Strava API a získá uživatelova kola. Uživatele i kola poté přidá do databáze. Frontendu následně vrátí JSON uživatele.

3.5.2 AES Šifrování

AES 256 cbc (cipher block chaining) data šifruje symetricky. Znamená to tedy, že s klíčem, kterým se data zašifrují, musí být i dešifrována. Pro šifrování a dešifrování je použit 256bitový klíč, který je tedy nepostradatelný a je třeba ho bezpečně zachovat. Má aplikace uchovává klíč na environmentálních hodnotách na serveru. Pro šifrování je kromě klíče používáno IV (inicializační vektor), což je random 128 bitů, které zajistí, že i stejný text, bude pokaždé zašifrován jinak. Cipher block chaining znamená, že data jsou rozdělena na bloky, které jsou na sobě závislé. 1. blok je skombinován s IV, každý další blok je poté kombinován s předchozí kombinací. Následně jsou tyto bloky zašifrované klíčem. [19, 24, 23]

3.5.3 PostgreSQL

Backend využívá relační databázi PostgreSQL pro ukládání informací o uživatelích, kolech a jejich komponentách. Struktura databáze je navržena tak, aby umožňovala efektivní načítání a správu dat.



Obrázek 3 Databázové schéma

Jak je možné vidět na obrázku 3, každá tabulka je propojena cizími klíči, s adekvátními asociacemi, čímž je zajišťována integrita dat a jsou umožňovány efektivní dotazy.

Tabulka Users je s tabulkou Components propojena vztahem M:N, čímž je umožňováno, aby komponenty, které nejsou momentálně instalovány na kole, byly uživatelem ukládány odděleně, přičemž tato skutečnost je zaznamenávána v asociační tabulce. Dále je tabulka Users spojena s tabulkou Bikes vztahem 1:N, čímž je vyjádřeno, že jedním uživatelem může být vlastněno více kol.

Tabulkou Bikes je s tabulkou Components vytvářen vztah M:N, jímž je umožňováno, aby tatáž komponenta byla instalována na více kolech a naopak. Tabulkou Bikes je také vytvářen vztah 1:N s tabulkou Activities, čímž je vyjádřeno, že jednomu kolu může být přiřazeno více aktivit.

Tabulkou Components jsou kromě již zmíněných asociací vytvářeny vztahy M:N s tabulkou Activities, kde je opět uplatňováno, že jednou komponentou může být

absolvováno více aktivit a naopak. Tabulkou Components je také vytvářen vztah M:N s tabulkou Service_Intervals, čímž je vyjádřeno, že jeden servis může být prováděn na více komponentách a naopak.

3.5.4 Synchronizace

Při spuštění aplikace backend načte aktuální data o uživateli a kolech ze Strava API a uloží je do databáze. Komponenty a servisy Strava nezaznamenává, jsou tedy ukládané v databázi a vždy před buildem listu komponent nebo servisů, se načtou z databáze.

3.6 Frontend

BikeCheck frontend je vyvinut ve Flutteru a slouží jako uživatelské rozhraní propojené s backendem. Po úspěšném přihlášení se načítají data uživatele, včetně seznamu jeho kol. Uživatel si může zobrazit podrobnosti o jednotlivých kolech a jejich komponentách, jako jsou řetězy, kazety nebo pláště. Každá komponenta obsahuje informace o svém typu a nájezdu. Ke komponentě je poté možno přidávat servisy s popisem a datem.

S backendem komunikuje přes REST API endpointy viz Příloha 1 – dokumentace endpointů.

3.6.1 Autorizace

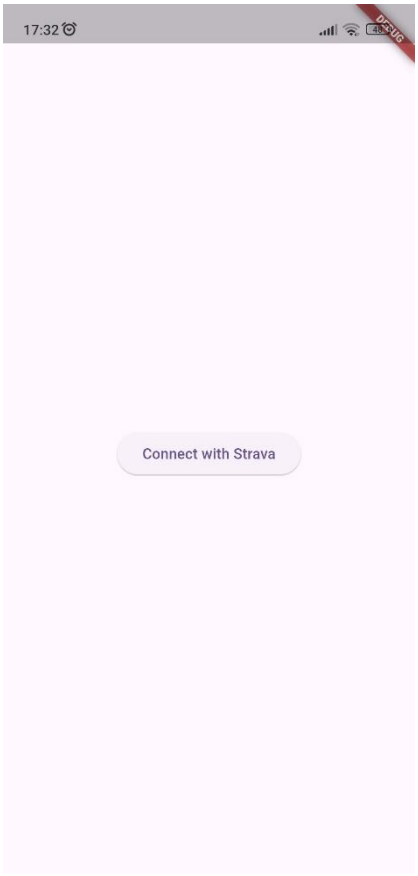
Na frontendu probíhá autorizace uživatele prostřednictvím Strava OAuth. Po úspěšném přihlášení se získaný autorizační kód předá backendu, který jej vymění za přístupový token. Frontend uchovává minimální množství citlivých dat a využívá **Secure Storage** pro bezpečné ukládání přihlašovacích údajů. Veškerá komunikace s backendem probíhá přes zabezpečené HTTPS spojení, čímž se minimalizuje riziko neoprávněného přístupu.

3.6.2 Struktura aplikace

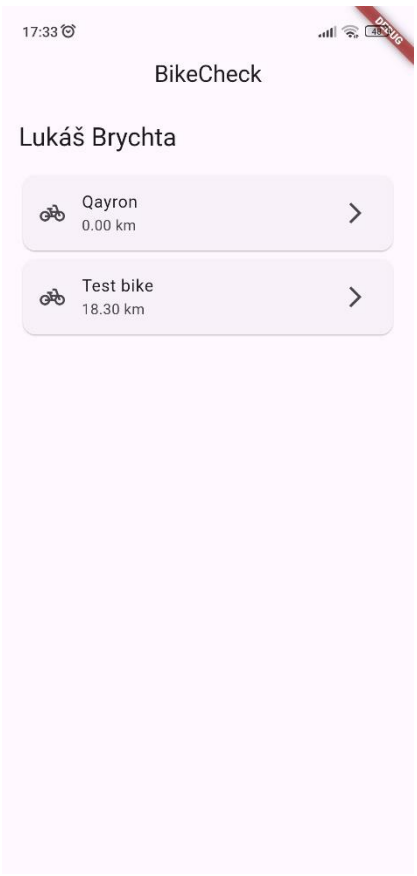
Aplikace je rozdělena do několika hlavních částí:

Autentizace – Zajišťuje přihlášení uživatele přes Strava OAuth a následné odeslání auth tokenu na backend pro výměnu za access a refresh token. viz obrázek 5.

Hlavní obrazovka – Po úspěšné autentizaci, se uživateli ukážou jeho kola, které se vylistovaná zobrazují. viz obrázek 4.



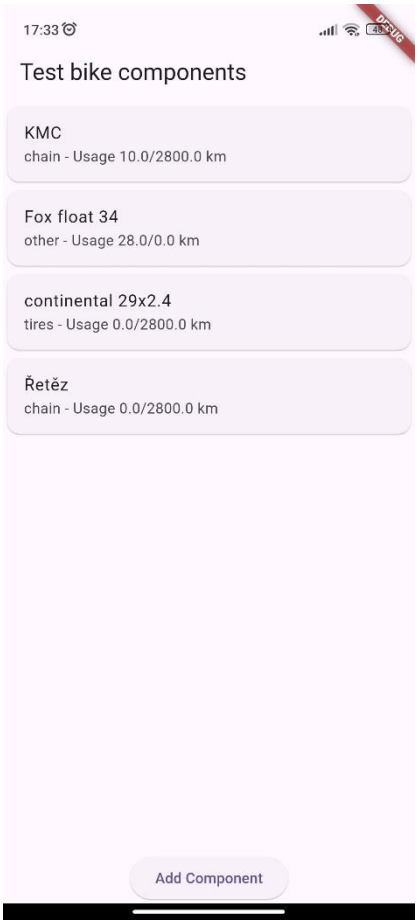
Obrázek 5 Stránka pro autentizaci přes Stravu



Obrázek 4 List kol

List komponent – Zobrazuje list komponent daného kola, kolik má komponenta najeto a její typ. Ve spodu je tlačítko na přidání nové komponenty k danému kolu. viz obrázek 7.

List servisů – Zobrazuje list servisů dané komponenty, čas, kdy byl servis přidán a popis servisního úkonu. Ve spodu je opět tlačítko na přidávání servisů k dané komponentě. viz obrázek 6.

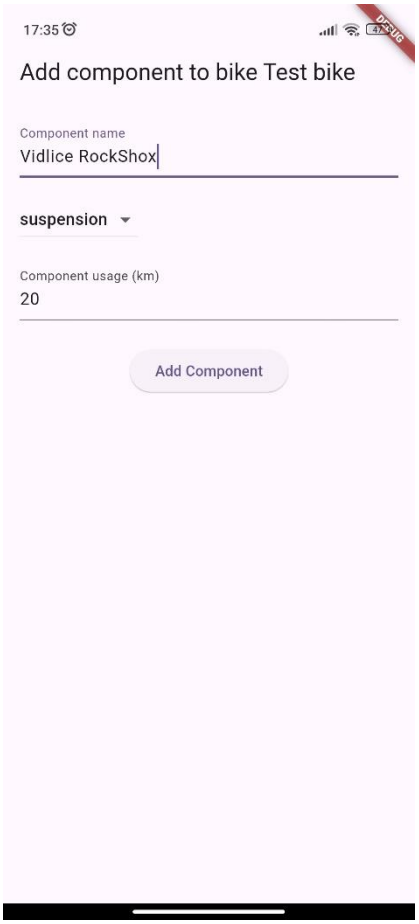


Obrázek 7 List komponent

Obrázek 6 List servisních úkonů

Stránka pro přidání komponenty – Stránka zobrazí formulář, do kterého uživatel zadá hodnoty, které se po přidání přiřadí nové komponentě. viz obrázek 8.

Stránka pro přidání servisního úkonu – Stránka zobrazí formulář, do kterého uživatel zadá hodnoty, které se po přidání přiřadí novému servisnímu úkonu. viz obrázek 9.



Obrázek 8 Přidání komponenty

Obrázek 9 Přidání servisního úkonu

Závěr

Tento maturitní projekt si kladl za cíl vytvořit mobilní aplikaci pro správu jízdních kol a jejich komponent s integrací služby Strava. Aplikace umožňuje uživatelům autorizovat se přes Stravu, zobrazovat si kola, jejich komponenty, sledovat opotřebení a plánovat servisní úkony podle skutečných jízd.

Během vývoje jsem se setkal s několika problémy, především s ORM systémem a propojením databáze. Za náročné bylo považováno správné nastavení vztahů mezi tabulkami a efektivní práce s daty. Komunikace se Strava API byla rovněž složitější, než bylo původně očekáváno, jelikož musela být řešena autentizace a zpracování dat o aktivitách. I přes tyto komplikace byly úspěšně implementovány základní funkce jako přidávání komponent, sledování jejich stavu a zaznamenávání servisních úkonů.

Aplikace BikeCheck může být poskytnuta cyklistům, kteří chtějí mít přehled o stavu svého kola. Pro cyklisty je obecně obtížné pamatovat si, kolik kilometrů bylo najeto na řetězu nebo kdy byly naposledy měněny brzdové destičky. Díky této aplikaci již není nutné tyto informace uchovávat v paměti.

Prací na tomto projektu jsem získal cenné zkušenosti v oblasti programování, práce s databázemi a API. Byl mi také poskytnut vhled do procesu tvorby aplikace od počátečního návrhu až po finální implementaci. Ačkoli byl projekt náročnější, než jsem z počátku předpokládal, jeho dokončením jsem dosáhnul hlubšího porozumění fungování databází a jejich propojení s aplikacemi.

V budoucnu by mohla být aplikace dále vylepšena. Mohly by být přidány grafy zobrazující míru opotřebení komponent nebo funkce předpovídající potřebu servisu na základě stylu jízdy. Za přínosné by mohlo být považováno také přidání možnosti sdílení tipů na údržbu s ostatními uživateli nebo propojení s dalšími sportovními aplikacemi kromě Stravy. Nepochybně by byla oceněna i lepší offline funkce, zajišťující fungování aplikace i bez připojení k internetu, což je situace na cyklovýletech často se vyskytující.

Celkově hodnotím výsledek projektu jako uspokojivý. Byla vytvořena funkční aplikace řešící skutečný problém. Získal jsem zkušenosti, které mohu využít při dalším studiu nebo v profesním životě. Ačkoli aplikace není ještě zcela dokonalá, může být považována za solidní základ pro budoucí vývoj a vylepšení.

Literatura

1. *About Node.js.* Online. Node.js. Dostupné z: https://nodejs.org/en/about. [cit. 2025-03-25].
2. *BikeManager - Cycle maintenance.* Online. Google Play. Dostupné z: https://play.google.com/store/apps/details?id=com.bikemanagerapp&hl=en. [cit. 2025-03-26].
3. *Crypto.* Online. Node.js. Dostupné z: https://nodejs.org/api/crypto.html#crypto. [cit. 2025-03-25].
4. *Express.* Online. Dostupné z: https://expressjs.com/. [cit. 2025-03-25].
5. *Flutter.* Online. Dostupné z: https://flutter.dev/development. [cit. 2025-03-25].
6. *Getting Started with the Strava API.* Online. Strava Developers. Dostupné z: https://developers.strava.com/docs/getting-started/. [cit. 2025-03-25].
7. *How does Flutter Engine work?* Online. Medium. Dostupné z: https://medium.com/@nachiketgohil185/how-does-flutter-engine-work-c1398a3252a4. [cit. 2025-03-27].
8. *MainTrack.* Online. Dostupné z: https://maintrack.app/. [cit. 2025-03-25].
9. *Node.js Crypto Module.* Online. W3Schools. Dostupné z: https://www.w3schools.com/nodejs/ref_crypto.asp. [cit. 2025-03-25].
10. *Node.js Introduction.* Online. W3Schools. Dostupné z: https://www.w3schools.com/nodejs/nodejs_intro.asp. [cit. 2025-03-25].
11. *OAuth 2.0.* Online. OAuth Community Site. Dostupné z: https://oauth.net/2/. [cit. 2025-03-25].
12. *ProBikeGarage.* Online. Dostupné z: https://www.probikegarage.com/. [cit. 2025-03-25].
13. *Sequelize.* Online. Dostupné z: https://sequelize.org/. [cit. 2025-03-25].
14. *Sequelize Basics for Beginners.* Online. DEV Community. Dostupné z: https://dev.to/ceceliacreates/sequelize-basics-for-beginners-part-one-2lc6. [cit. 2025-03-25].
15. *State management.* Online. Flutter Docs. Dostupné z: https://docs.flutter.dev/get-started/fundamentals/state-management. [cit. 2025-03-25].
16. *Strava API v3 API and SDK Reference.* Online. Strava Developers. Dostupné z: https://developers.strava.com/docs/reference/. [cit. 2025-03-25].
17. *Strava Developers.* Online. Dostupné z: https://developers.strava.com/. [cit. 2025-03-25].
18. *Strava Revenue and Usage Statistics (2025).* Online. Business of Apps. 22-01-2025. Dostupné z: https://www.businessofapps.com/data/strava-statistics/. [cit. 2025-03-25].
19. *What is AES-256-CBC?* Online. FenixPyre. Dostupné z: https://docs.anchormydata.com/docs/what-is-aes-256-cbc. [cit. 2025-03-25].
20. *What is Crypto Module in Node.js and How it is used?* Online. Geeks for Geeks. Dostupné z: https://www.geeksforgeeks.org/what-is-crypto-module-in-node-js-and-how-it-is-used/. [cit. 2025-03-27].

21. *What is OAuth 2.0?* Online. Auth0. Dostupné z: https://auth0.com/intro-to-iam/what-is-oauth-2. [cit. 2025-03-25].
22. *What is PostgreSQL?* Online. PostgreSQL. Dostupné z: https://www.postgresql.org/about/. [cit. 2025-03-25].
23. *What Is AES Encryption? The Complete Guide.* Online. 1Kosmos. Dostupné z: https://www.1kosmos.com/authentication/aes-encryption/. [cit. 2025-03-25].
24. *Why You Should Use AES 256 Encryption to Secure Your Data.* Online. Progress. Dostupné z: https://www.progress.com/blogs/use-aes-256-encryption-secure-data. [cit. 2025-03-25].

Seznam obrázků

Obrázek 1 Use case diagram 15
Obrázek 2 Architektura 16
Obrázek 3 Databázové schéma 18
Obrázek 4 List kol 20
Obrázek 5 Stránka pro autentizaci přes Stravu 20
Obrázek 6 List servisních úkonů 21
Obrázek 7 List komponent 21
Obrázek 8 Přidání komponenty 22
Obrázek 9 Přidání servisního úkonu 22

Příloha 1 – API Endpoint dokumentace

Activities – nepoužívá se

Endpoint: GET /activities/components/:component_id/activities

parametry:

- component_id (path) - ID komponenty.

Popis: Vrátí seznam všech aktivit spojených s danou komponentou.

Odpověď: Seznam aktivit patřících ke komponentě.

Endpoint: POST /activities/components/:component_id/activities

parametry:

- component_id (path) - ID komponenty.
- distance (body) - Ujetá vzdálenost (povinné).
- duration (body) - Doba trvání aktivity (povinné).
- bike_id (body) - ID kola použitého pro aktivitu (povinné).

Popis: Vytvoří novou aktivitu a přiřadí ji ke specifikované komponentě.

Odpověď: Vrátí nově vytvořenou aktivitu.

Endpoint: GET /activities/components/:component_id/activities/:activity_id parametry:

- component_id (path) - ID komponenty.
- activity_id (path) - ID aktivity.

Popis: Vrátí podrobnosti o konkrétní aktivitě spojené s komponentou.

Odpověď: Data o aktivitě, pokud existuje.

Endpoint: PUT /activities/components/:component_id/activities/:activity_id parametry:

- component_id (path) - ID komponenty.
- activity_id (path) - ID aktivity.
- (body) – aktualizované podrobnosti aktivity

Popis: Aktualizuje existující aktivitu spojenou s komponentou.

Odpověď: Potvrzení úspěšné aktualizace nebo chybová zpráva, pokud aktivita neexistuje.

Endpoint: DELETE /activities/components/:component_id/activities/:activity_id
parametry:

- component_id (path) - ID komponenty.
- activity_id (path) - ID aktivity.

Popis: Odstraní specifikovanou aktivitu z databáze.

Odpověď: Potvrzení úspěšného smazání nebo chybová zpráva, pokud aktivita neexistuje.

Bikes

Endpoint: GET /bikes/users/:user_id/bikes

Parametry:

- user_id (path) - ID uživatele.

Popis: Vrátí seznam všech kol patřících danému uživateli.

Odpověď: Seznam kol.

Endpoint: POST /bikes/users/:user_id/bikes

Parametry:

- user_id (path) - ID uživatele.
- bike_id (body) - ID kola.
- name (body) - Název kola.
- distance (body) - Ujetá vzdálenost.

Popis: Vytvoří nové kolo pro uživatele.

Odpověď: Informace o vytvořeném kole.

Endpoint: GET /bikes/users/:user_id/bikes/:bike_id

Parametry:

- user_id (path) - ID uživatele.
- bike_id (path) - ID kola.

Popis: Vrátí konkrétní kolo podle jeho ID.

Odpověď: Detaily kola.

Endpoint: PUT /bikes/users/:user_id/bikes/:bike_id

Parametry:

- user_id (path) - ID uživatele.
- bike_id (path) - ID kola.
- (body) - Aktualizované podrobnosti kola.

Popis: Aktualizuje informace o kole.

Odpověď: Potvrzení úspěšné aktualizace.

Endpoint: DELETE /bikes/users/:user_id/bikes/:bike_id

Parametry:

- user_id (path) - ID uživatele.
- bike_id (path) - ID kola.

Popis: Odstraní kolo z databáze.

Odpověď: Potvrzení úspěšného smazání.

Components

Endpoint: GET /components/bikes/:bike_id/components

Parametry:

- bike_id (path) - ID kola

Popis: Načte všechny komponenty přidružené k určitému kolu.

Odpověď: Seznam komponent pro zadané kolo.

Endpoint: POST /components/bikes/:bike_id/components

Parametry:

- bike_id (path) - ID kola
- name (body) - Název komponenty
- type (body) - Typ komponenty
- usage (body) - Použití komponenty
- lifespan (body) - Životnost komponenty

Popis: Vytvoří novou komponentu a přidruží ji k určitému kolu.

Odpověď: Podrobnosti o vytvořené komponentě a potvrzovací zpráva.

Endpoint: GET /components/bikes/:bike_id/components/:component_id

Parametry:

- bike_id (path) - ID kola
- component_id (path) - ID komponenty

Popis: Načte konkrétní komponentu přidruženou k určitému kolu.

Odpověď: Podrobnosti o požadované komponentě.

Endpoint: PUT /components/bikes/:bike_id/components/:component_id

Parametry:

- bike_id (path) - ID kola
- component_id (path) - ID komponenty
- (body) - Aktualizované podrobnosti komponenty

Popis: Aktualizuje konkrétní komponentu přidruženou k určitému kolu.

Odpověď: Potvrzení úspěšné aktualizace.

Endpoint: DELETE /components/bikes/:bike_id/components/:component_id

Parametry:

- bike_id (path) - ID kola
- component_id (path) - ID komponenty

Popis: Odstraní konkrétní komponentu přidruženou k určitému kolu.

Odpověď: Potvrzení úspěšného odstranění.

Endpoint: GET /components/users/:user_id/components

Parametry:

- user_id (path) - ID uživatele

Popis: Načte všechny komponenty uložené určitým uživatelem.

Odpověď: Seznam komponent uložených zadaným uživatelem.

Endpoint: POST /components/users/:user_id/components

Parametry:

- user_id (path) - ID uživatele
- name (body) - Název komponenty
- type (body) - Typ komponenty
- usage (body) - Použití komponenty

- lifespan (body) - Životnost komponenty

Popis: Vytvoří novou komponentu a přidruží ji k úložišti určitého uživatele.

Odpověď: Potvrzovací zpráva o vytvoření komponenty.

Endpoint: GET /components/users/:user_id/components/:component_id

Parametry:

- user_id (path) - ID uživatele
- component_id (path) - ID komponenty

Popis: Načte konkrétní komponentu uloženou určitým uživatelem.

Odpověď: Podrobnosti o požadované komponentě.

Endpoint: PUT /components/users/:user_id/components/:component_id

Parametry:

- user_id (path) - ID uživatele
- component_id (path) - ID komponenty
- (body) - Aktualizované podrobnosti komponenty

Popis: Aktualizuje konkrétní komponentu uloženou určitým uživatelem.

Odpověď: Potvrzení úspěšné aktualizace.

Endpoint: DELETE /components/users/:user_id/components/:component_id

Parametry:

- user_id (path) - ID uživatele
- component_id (path) - ID komponenty

Popis: Odstraní konkrétní komponentu uloženou určitým uživatelem.

Odpověď: Potvrzení úspěšného odstranění.

Service_Intervals

Endpoint: GET /service/components/:component_id/service_intervals

Parametry:

- component_id (path) - ID komponenty

Popis: Načte všechny servisní intervaly pro konkrétní komponentu.

Odpověď: Seznam servisních intervalů pro zadanou komponentu.

Endpoint: POST /service/components/:component_id/service_intervals

Parametry:

- component_id (path) - ID komponenty
- description (body) - Popis servisního intervalu

Popis: Vytvoří nový servisní interval pro konkrétní komponentu.

Odpověď: Detaily vytvořeného servisního intervalu a potvrzovací zpráva.

Endpoint: GET /service/components/:component_id/service_intervals/:service_id

Parametry:

- component_id (path) - ID komponenty
- service_id (path) - ID servisního intervalu

Popis: Načte konkrétní servisní interval pro danou komponentu.

Odpověď: Detaily požadovaného servisního intervalu.

Endpoint: PUT /service/components/:component_id/service_intervals/:service_id

Parametry:

- component_id (path) - ID komponenty
- service_id (path) - ID servisního intervalu
- (body) - Aktualizované údaje servisního intervalu

Popis: Aktualizuje konkrétní servisní interval pro danou komponentu.

Odpověď: Potvrzení úspěšné aktualizace.

Endpoint: DELETE /service/components/:component_id/service_intervals/:service_id

Parametry:

- component_id (path) - ID komponenty
- service_id (path) - ID servisního intervalu

Popis: Odstraní konkrétní servisní interval pro danou komponentu.

Odpověď: Potvrzení úspěšného odstranění.

Strava

Endpoint: POST /strava/auth/tokenexchange

Parametry:

- code (body) - Autorizační kód od Strava

Popis: Provede autentizaci uživatele přes Strava API, vymění autorizační kód za přístupový token. Vytvoří nového uživatele nebo aktualizuje existujícího. Synchronizuje uživatelova kola ze Strava. Ukládá šifrované přístupové tokeny. Automaticky obnovuje expirované tokeny.

User

Endpoint: POST /users/users

Parametry**:**

- user_id (body) - Jedinečné ID uživatele
- username (body) - Uživatelské jméno
- access_token (body) - Přístupový token
- refresh_token (body) - Obnovovací token

Popis**:** Vytvoří nového uživatele v systému. Všechna pole jsou povinná.

Odpověď**:** Vytvořený uživatelský záznam s potvrzovací zprávou.

Endpoint: GET /users/users/:user_id

Parametry**:**

- user_id (path) - ID uživatele

Popis**:** Získání detailních informací o konkrétním uživateli.

Odpověď**:** Celý uživatelský objekt včetně všech údajů.

Endpoint: PUT /users/users/:user_id

Parametry**:**

- user_id (path) - ID uživatele
- (body) – Aktualizované podrobnosti uživatele

Popis**:** Aktualizuje existujícího uživatele. Lze měnit jednotlivá pole nebo všechna najednou.

Odpověď**:** Potvrzení úspěšné aktualizace.

Endpoint: DELETE /users/users/:user_id

Parametry**:**

- user_id (path) - ID uživatele

Popis**:** Trvale odstraní uživatele z databáze.

Odpověď**:** Potvrzení úspěšného smazání.