FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Mobilní aplikace pro včelaře

Martin Lácha

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Mobilní aplikace pro včelaře

Bc. Martin Lácha

Vedoucí práce
Ing. Ladislav Pešička

PLZEŇ **2024**

Citace v seznamu literatury:
LÁCHA, Martin. *Mobilní aplikace pro včelaře*. Plzeň, 2024. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Ladislav Pešička.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin LÁCHA**
Osobní číslo:	**A21N0057P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Mobilní aplikace pro včelaře**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Nastudujte technologie a oborovu problematiku potřeb včelařů, které lze podpořit softwarovými systémy. Prostudujte a analyzujte vybrané mobilní aplikace zabývající se včelařstvím.
2. Na základě předchozího bodu navrhněte mobilní aplikaci pro platformu Android pro podporu včelařů, umožňující mj. správu vlastních úlů a interakci s dalšími uživateli. Součástí navrženého systému bude kromě mobilní aplikace i komunikační server.
3. Navrženou aplikaci včetně serveru realizujte, ověřte její funkcionalitu a navrhněte vhodná další rozšíření.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Ing. Ladislav Pešička**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **8. září 2023**
Termín odevzdání diplomové práce: **16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2023

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Chlumanech dne 26. dubna 2024

. . . . . .
Martin Lácha

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Cílem této práce je uvést čtenáře do základů včelaření, analyzovat a zhodnotit vlastnosti vybraných aplikací pro včelaře a nakonec navrhnout a implementovat komunikační server a mobilní aplikaci podporující včelaře.

Teoretická část se bude zabývat včelařením (základy, problematika, podpora IT), analýzou existujících aplikací pro včelaře (zhodnocení výhod, nevýhod a funkcí). Praktická část se zabývá návrhem a implementací vlastní mobilní aplikace, která bude obsahovat vybrané funkcionality pro podporu, rozvoj a monitorování včelstev. Komunikační server bude komunikovat s mobilními klienti, které budou poskytovat uživatelské rozhraní. V další části práce je popsán návrh a integrace senzorů pro monitorování aktivity úlu. Instalační příručka obsahuje detailní postup pro instalaci jednotlivých částí systému. Uživatelská příručka popisuje jednotlivé obrazovky a funkcionality mobilní aplikace. Na závěr bude komunikační server a mobilní aplikace otestována a ke každé části budou navržena další rozšíření.

Abstract

The aim of this paper is to introduce the reader to the basics of beekeeping, to analyze and evaluate the advantages and disadvantages of selected applications for beekeepers, and finally to design and implement a communication server and a mobile application supporting beekeepers.

The theoretical part will deal with beekeeping (basics, issues, IT support), and analysis of existing applications for beekeepers (evaluation of advantages, disadvantages, and features). The practical part will describe the design and implementation of a custom mobile application that will include selected functionalities for the support, development, and monitoring of bee colonies. The communication server will provide the data to clients, and the mobile application will be used as a user interface to display the data. The following part of this paper describes the design and integration of sensors for monitoring beehive activity. The installation guide contains a detailed procedure for the installation of each part. The user guide describes the screens and functionalities of the mobile application. Finally, the communication server and the mobile application will be tested and further extensions will be proposed.

Klíčová slova

Včelaření • Mobilní aplikace • Komunikační server • Senzory • Android

Poděkování

Tímto bych velmi rád poděkoval Ing. Ladislavu Pešičkovi, za odborné vedení práce, odborné rady a čas, které mi poskytl při zpracování této diplomové práce.

Také bych rád poděkoval celé své rodině za velkou podporu a trpělivost během zpracování této diplomové práce.

Nakonec bych chtěl poděkovat panu Vladislavovi Turkovi za poskytnutí včelích úlů, které byly použity pro monitování včelstva pomocí senzorů.

Obsah

Úvod 1

Včely hrají v ekosystému klíčovou roli, která má velký vliv na lidskou společnost a přírodu. V posledních letech se však velmi snižuje jejich populace. Důvody pro úbytek včelstev jsou klimatické změny, negativní dopady lidské činnosti, působení parazitů a rozšíření nemocí včel. S vývojem technologií je možné využít různá zařízení (mobilní telefon, vestavěná zařízení, senzory atd.) pro správu a monitorování jednotlivých včelstev.

Cílem této diplomové práce je analyzovat vlastnosti vybraných mobilních aplikací pro včelaře a následně vytvořit vlastní systém včetně mobilní aplikace, obsahujících vybranou funkcionalitu pro podporu a správu včelařských aktivit.

V práci budou analyzovány vybrané existující aplikace a porovnány jejich výhody, nevýhody a funkce. Po analýze bude navržena mobilní aplikace, komunikační server a senzorů pro podporu včelařských aktivit, monitorování stavu a zdraví včelstev. Budou vybrány vhodné nástroje a technologie pro realizaci systému. Dále bude popsána architekturu, zabezpečení a způsob komunikace mezi jednotlivými částmi navrženého systému. V následující části práce bude popsána vlastní realizace jednotlivých částí systému a popis jejich struktury a konfigurace.

Poslední část práce bude popisovat ověření správné funkcionality systému, budou vyhodnoceny výsledky testování a návrhy na rozšíření jednotlivých částí vytvořeného systému.

Včelaření 2

Chovem včel se člověk zabývá už stovky let a je možné se s ním setkat prakticky téměř všude na celém světě. Dříve lidé chovali včely především pro jejich produkty jako jsou med, vosk nebo například propolis. Dnes se této činnosti lidé věnují z dalších důvodů, ať už je to finanční přivýdělek nebo včely zdědili od někoho z rodiny. Motivy včelařů jsou různé a stejně tak i počty jejich včelstev, avšak co je víceméně pro všechny totožné jsou základní principy práce se včelami.

Každý nováček se bude potýkat se stejnými problémy začínajícího včelaře např. výběr nejvhodnějšího stanoviště pro úl, jaké vybavení pořídit, kde a jak si pořídit první včelstvo a jak se o něj nejlépe starat.

Řada věcí bude každému začínajícímu včelaři připadat zpočátku složitá a u mnoha věcí si nebude jistý správným rozhodnutím. To vyžaduje pilné studium a s moderními technologiemi, již není složité si potřebné informace dohledat na internetu. Než se z nováčka stane zkušený včelař, může to trvat i několik let [1] a ostatní včelaři mu určitě poskytnou spoustu cenných rad.

2.1 Včela medonosná

Včela medonosná (Apis mellifera) je jedna z nejznámějších zástupců společenského hmyzu. Jedná se o druh včely nejvíce známý díky schopnosti produkovat med a opylovat květiny. Včely žijí ve společenství, které se nazývá včelstvo. Mezi včelami jsou složité vztahy a také ve včelstvu funguje dokonalá dělba práce. V jednom společenství je zpravidla jedna oplozená matka, viz obrázek 2.1, jejíž úlohou je především klást vajíčka, čímž zajišťuje obnovu včelstva. Včelí matka se v úlu dožívá až 5 let [2].

Dále se ve včelstvu nachází 500-2000 trubců. Jejich úkolem je oplodnit mladou matku a po oplození trubec okamžitě umírá. Jsou odchováni od jara do podletí[1] (polovina července). Líhnou se z neoplozených vajíček do buněk, které jsou větší, než buňky pro dělnice. Před zimou jsou ostatními včelami vyhnány z úlu, aby nespotřebovávali zásoby. V sezóně však pomáhají zahřívat včelí plod, jelikož mají vyšší tělesnou teplotu než klasická dělnice a nemají žihadlo [3].

[1]Období regenerace spojené s posledním vytáčením medu po poslední hlavní snůšce. Včely se připravují na zimu a začínají loupežit u slabých včelstev.

Obrázek 2.1: Včelí matka a dělnice

Dělnic se ve včelstvu nachází v sezóně přibližně 60 tisíc (přes zimu to bývá 20 tisíc). Dělnice žije přibližně 35 dnů [4]. Na zimu se líhnou tzv. dlouhověké včely, které mají za úkol přečkat zimu a délka jejich života může být i několik měsíců. Ve včelstvu vykonávají všechny ostatní práce:

- vyhledávání potravy,
- zpracování medu z nektaru a medovice,
- nošení, zpracování pylu a propolisu,
- nošení vody,
- stavění plástů,
- krmení matky, trubců a plodu,
- střežení vchodu úlu,
- čištění plástů a samotných včel,
- větrání a udržování správné teploty a vlhkosti úlu.

Každá včela za svůj život projde několika stádii, ve kterých má v úlu přidělenou konkrétní činnost, viz tabulka 2.1. Tento proces začíná po vylíhnutí včely:

Den	Stadium	Činnost
po 2–3 hodinách	čistička	čištění buněk a příprava pro matku ke kladení
4. den	krmička	krmení starších larev
6. den	kojička	krmení mladých plodů a matky mateří kašičkou
12. den	stavitelka	stavění včelích plástů (aktivují se voskové žlázy)
18. den	strážkyně česna	střežení vchodu do úlu před nepřáteli (vyvine se jedová žláza)
21. den	létavka	donáší do úlu vodu, nektar, medovici, propolis

Tabulka 2.1: Činnosti včelí dělnice

Produkty

Schopnosti včely produkovat různé produkty jsou zajímavým předmětem vědeckého zkoumání (zdraví a léčby). Každá látka je včelou vytvářena specifickým způsobem a je v úlu používána k odlišnému účelu:

- **Pyl** - je létavkami umístěn do buněk. Včely v úlu následně uskladněný pyl odebírají pro krmení plodu. Včelstvo pro svoji potřebu přinese a spotřebuje za celý rok přibližně 30 kilogramů pylu.
- **Med** - po příletu včely do úlu, předávají létavky nektar z volátka (medného váčku) včelám. Včely přinesený nektar zpracovávají a zahušťují, potom jej ukládají do buněk plástů. Dalším zahušťováním se snižuje procento vody v medu na 16-22%. Poté jej včely zavíčkují. Med včely používají podle potřeby na přežití, ale i jako pohonnou látku k létání. Včelstvo spotřebuje pro svoji potřebu za rok přibližně 70 kilogramů medu.
- **Propolis** - je lepkavá pryskyřičná substance, kterou včely sbírají na pupenech stromů a keřů. Obsahuje vysoký podíl látek bránících vývoji mikroorganizmů. Včely ji používají k tmelení vnitřních stěn úlů, dezinfekci plástů a mumifikaci mrtvých těl vetřelců.
- **Vosk** - je vysoce odolný ester jednoduchých alkoholů a vyšších mastných kyselin. Mladé včely ho produkují jako výpotek tzv. zrcátek na spodní straně článků zadečku. Včely tyto výpotky smíchají slinami a postupně z nich modelují plást, což je svislá tenká vosková stěna osázená z obou stran hustou sítí šestibokých buněk. Buňka má šíři 4,6-5,4 mm u dělnic, u trubčích buněk je to kolem 6 mm. Hloubka buněk je až 15 mm. Buňky jsou stavěny v mírném úhlu cca 5 stupňů vzhůru.

- **Mateří kašička** - je výměšek hltanových žláz mladých včel, kterým je krmena včelí královna po dobu, kdy klade vajíčka a mateří plod po celou dobu svého vývoje. Obsahuje mnoho vitamínů, minerálů i bílkovin.
- **Včelí jed** - je bezbarvá tekutina, která je sekretem jedové žlázy. Je používám včelou k obraně v případě ohrožení.

2.2 Problematika včelaření

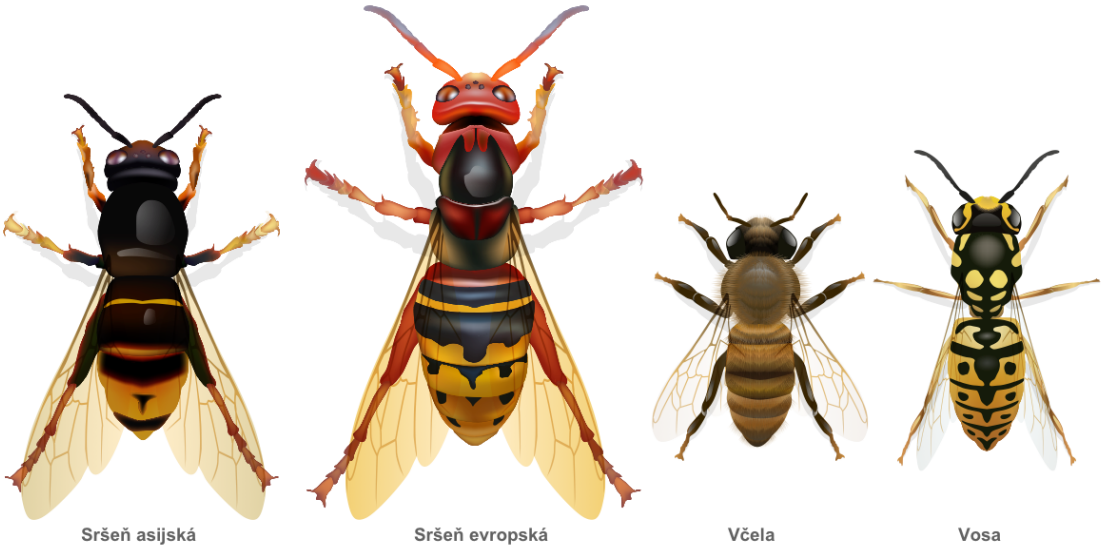
Lidé chovají včely především pro získávání včelích produktů. Další velice důležitý význam chovu včel je opylovací činnost. Hrají klíčovou roli při opylování rostlin, při kterém přenáší pyl z květů, což umožňuje rostlinám produkovat ovoce a semena. Jsou plodiny, u kterých by bez opylení včelami nadále nebyla zajištěna jejich úroda. Na světě je opylováno asi 85% všech kvetoucích rostlin hmyzem a z toho je přibližně 85% opylováno včelami. U ovocných stromů je cca 90% květů opylováno včelami. Z finančního hlediska přinese jedno včelstvo za rok národnímu hospodářství svou opylovací činností až přes sto tisíc korun ročně [3].

Poslední dobou v mnoha částech světa je pozorován výrazný úbytek populace včel. Tento jev je znám jako **Syndrom zhroucení včel** (CCD). Mezi příčiny tohoto jevu patří různé vlivy člověka na životní prostředí, jako jsou například agrotechnická opatření, pesticidy a nebo mnoho parazitů. Varroáza přenáší mnoho škodlivých virů nebo další nebezpečné nákazy jako je mor nebo hniloba včelího plodu, které dokáží zničit celé včelstvo. Včely mají také mnoho nepřátel, kteří jim dokáží ublížit např. vosy, myši, včelojedi. V nedávné době vyšla studie vědců z amerických univerzit, kde je uvedeno, že kvůli teplotním podmínkám čeká některé včelí kolonie kolaps [5]. Počasí a klimatické změny nebo extrémní sucho, které jsou evidovány posledních několik let, mají velký vliv na úbytek včel.

Jedním z největších nebezpečí pro včely je invaze sršně asijské. Porovnání oproti druhům, co žijí v České republice je možné vidět na obrázku 2.2. Tento invazivní druh se specializuje na lov včel. Již se objevila v Evropě a v současné době působí velké škody včelařům na území Francie [6]. Představuje velké riziko pro včely, které často napadá přímo u vstupu do jejich úlu a dokáže jim způsobit značné ztráty nebo dokonce úhyn celého včelstva. Dále může způsobit i velké škody na ovocných stromech či vinohradech, kde se živí sladkou šťávou ze zralého ovoce. Aby se zamezilo invazi, je potřeba jakýkoliv výskyt tohoto druhu okamžitě nahlásit [7].

Vymření včel by mělo vážné dopady na životní prostředí a ekosystém. Došlo by k výraznému poklesu úrody mnoha plodin, ovoce i zeleniny. Planeta by pomalu ztratila pestrobarevnost a biodiverzitu, což by velmi rapidně vedlo k monokulturnosti a postupné destabilizaci celého ekosystému.

Mělo by to obrovské důsledky na celý ekosystém, ekonomiku a lidský život obecně. Je proto velmi důležité se starat a podporovat zdraví a prosperitu včel [8].



Obrázek 2.2: Porovnání velikosti druhů [6]

2.3 Informační technologie ve včelařství

V poslední době pronikají moderní technologie do mnoha oblastí a včelaření tomu není výjimkou. Objevují se technologie pro záznam včelařských aktivit i samotné monitorování událostí v úlu. Samotné včelaření zabere mnoho času, a proto je primárním úkolem technologií především ulehčit a zjednodušit práci včelaře.

Internetové portály umožňují pomoci s povinnou administrativou včelaře vůči úřadům a včelařskému svazu. Mohou také zasílat výstrahy o postřicích ze stran zemědělců. Pro monitorování úlu se začaly používat senzory, které poskytují jejich aktuální hodnoty. Z naměřených dat je možné zjistit aktuální dění v úlu. Informace jako je teplota, vlhkost nebo váha úlu jsou pro včelaře velmi užitečné. Naměřená data se dále mohou využít pro účely včelařských výzkumů. V neposlední řadě se začalo využívat kognitivních technologií, strojového učení nebo analýzy obrazu a zvuku [9].

Je samozřejmé, že včelař stále musí pro určitě aktivity osobně úl kontrolovat, ale díky technologiím je možné množství potřebného času a počet inspekcí značně snížit.

Analýza mobilních aplikací

3

V dnešní době existuje celá řada mobilních aplikací podporující včelaření pro operační systém Android i iOS. Aplikace mohou poskytovat nejrůznější způsoby pro zaznamenávání vývoje včelstev. Dříve se k zaznamenávání používala asi nejvíce tužka a zápisník. Stále se najde mnoho včelařů ze starší i mladší generace, kteří tento způsob i v dnešní době používají.

S rychle rostoucím vývojem technologií se začínají používat pro zaznamenávání také aplikace na mobilních zařízeních. Aplikace poskytují včelaři asistenci na různých úrovních, například samotná včelstva, seznam úlů nebo přehled jednotlivých rámů v nástavku. Mohou také poskytovat grafické znázornění jednotlivých úlů a tím poskytnou rychlý přehled o aktuálním stavu včelstva daného včelaře. Aplikace mimo jiné poskytují základní záznamy o včelstvech jako je počet, stav, ošetření a léčení, krmení, evidence včelích královen a popřípadě i záznamech o sběru množství medu a evidence výnosů.

V rámci diplomové práce byla provedena anlýza několika vybraných aplikací zabývajících se problematikou včelaření. Jednotlivé aplikace byly testovány na telefonu Huawei P30 lite (Android 10).

3.1 APIManager

Základní informace:
Požadovaný operační systém: Android 5.1+, iOS 10+
Poslední aktualizace: 3.3.2024
Verze: 1.4.4
Počet stažení: 10 000+
Jazyky: angličtina

3.1.1 Funkce

Aplikace obsahuje následující funkce:

- správa včelínů, úlů,
- evidence léčení, krmení, matek, inventáře (nástrojů), sběru produktů v jednotlivých včelínech nebo financí,
- události a aktivity v kalendáři s možností vytvoření upozornění,
- úložiště fotek a videí,
- souhrny a zobrazení statistik a grafů,
- podpora QR kódu pro načtení úlu,
- hlasový asistent poskytující užitečné rady,
- časová osa pro zobrazení změn v aplikaci,
- webová aplikace pro zobrazení dat.

Předplatné

V základní verzi aplikace je možné zaznamenat a spravovat až 20 úlů. Pokud bude uživatel potřebovat rozšířit tento počet, jedinou možností je pořídit si roční předplatné aplikace. Na výběr existuje několik variant předplatného, které rozšiřují počet možných úlů, jenž je možné v aplikaci spravovat.

3.1.2 Výhody

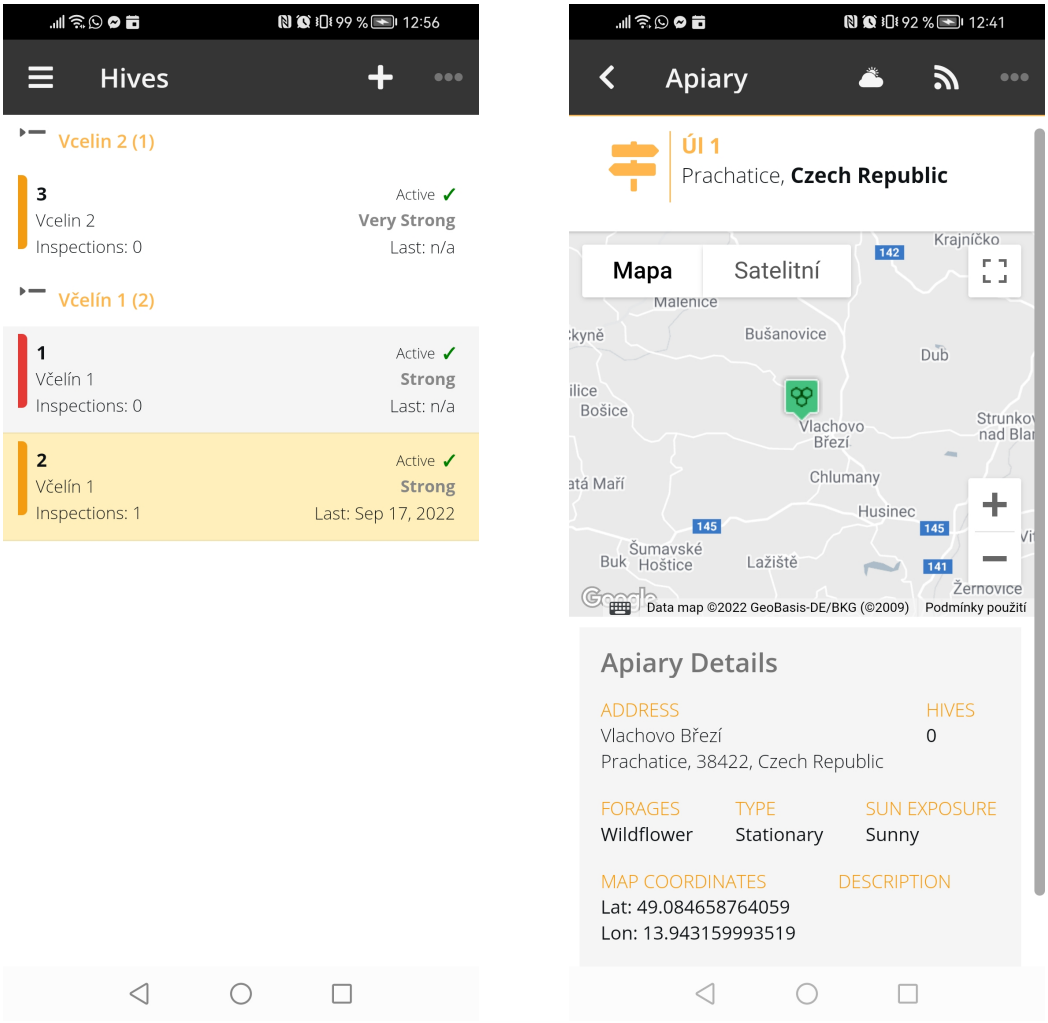
Jednou z hlavních výhod je přehledné grafické rozhraní a přehledné zobrazení seznamu včelínů, viz obrázek 3.1(a), popisující základní informace o včelínu. Lze vytvořit QR kód (poslaný na e-mail) umístit jej na úl a následně ho naskenovat pro rychlejší přidání záznamu. Aplikace obsahuje velmi rozsáhlé možnosti nastavení a přizpůsobení jednotlivých částí. Dále má aplikace možnost integrace Google Maps pro získávání geografické polohy jednotlivých včelínů, viz obrázek 3.1(b). Kontroly se vyhledávají v záznamech pomocí zadaného textu a filtrů. Včelařské inspekce jsou vytvořeny a rozděleny pro jednotlivé dny. Časová osa zobrazuje události se změnami v úlu. Pro jedno zařízení lze vytvořit více uživatelů s různými právy. Lze nastavit periodické notifikace k činnostem jako je například krmení nebo léčení.

3.1.3 Nevýhody

Do aplikace je potřeba se nejprve registrovat a v offline režimu ji lze používat až po prvním přihlášení. Aplikace nabízí pouze anglickou lokalizaci. Zpětné tlačítko občas ukončí celou aplikaci namísto zobrazení poslední obrazovky. Při zobrazení včelínu není možné vidět rovnou jednotlivé úly. Nepřehledný je kalendář událostí bez možnosti detailu jednotlivých dnů. Pro malý obsah dat se statistiky vůbec nezobrazily. Omezené je množství spravovaných úlů (maximálně 20) a při nutnosti zaznamenání dalších je potřeba si pořídit roční předplatné.

3.1.4 Ukázky aplikace

Obrázky 3.1 zobrazují ukázky obrazovek z mobilní aplikace:



(a) Včelíny a úly (b) Detail úlu

Obrázek 3.1: Ukázky aplikace APIManager

3.2 BoxyBEE

Základní informace:
Požadovaný operační systém: Android 5.1+
Poslední aktualizace: 12.4.2021
Verze: 0.91
Počet stažení: 5 000+
Jazyky: angličtina

3.2.1 Funkce

Aplikace obsahuje následující funkce:

- správa včelínů, úlů, nástavků, rámků,
- správa a konfigurace jednotlivých úlů,
- grafické zobrazení struktury úlů a jeho rámků,
- nastavení typu jednotlivých rámů úlů a záznam obsahu (med, vajíčka, prázdné atd.),
- detailní zobrazení úlů s časovou osou,
- souhrn stavu úlů pro jednotlivé dny,
- evidence léčení, nemocí, královen, krmení, škůdců.

Předplatné

Aplikace je dostupná zdarma pouze po dobu 30ti denní zkušební verze a poté je nutné zaplatit měsíční nebo roční předplatné. Měsíční předplatné nepřidává uživateli žádnou funkcionalitu navíc.

3.2.2 Výhody

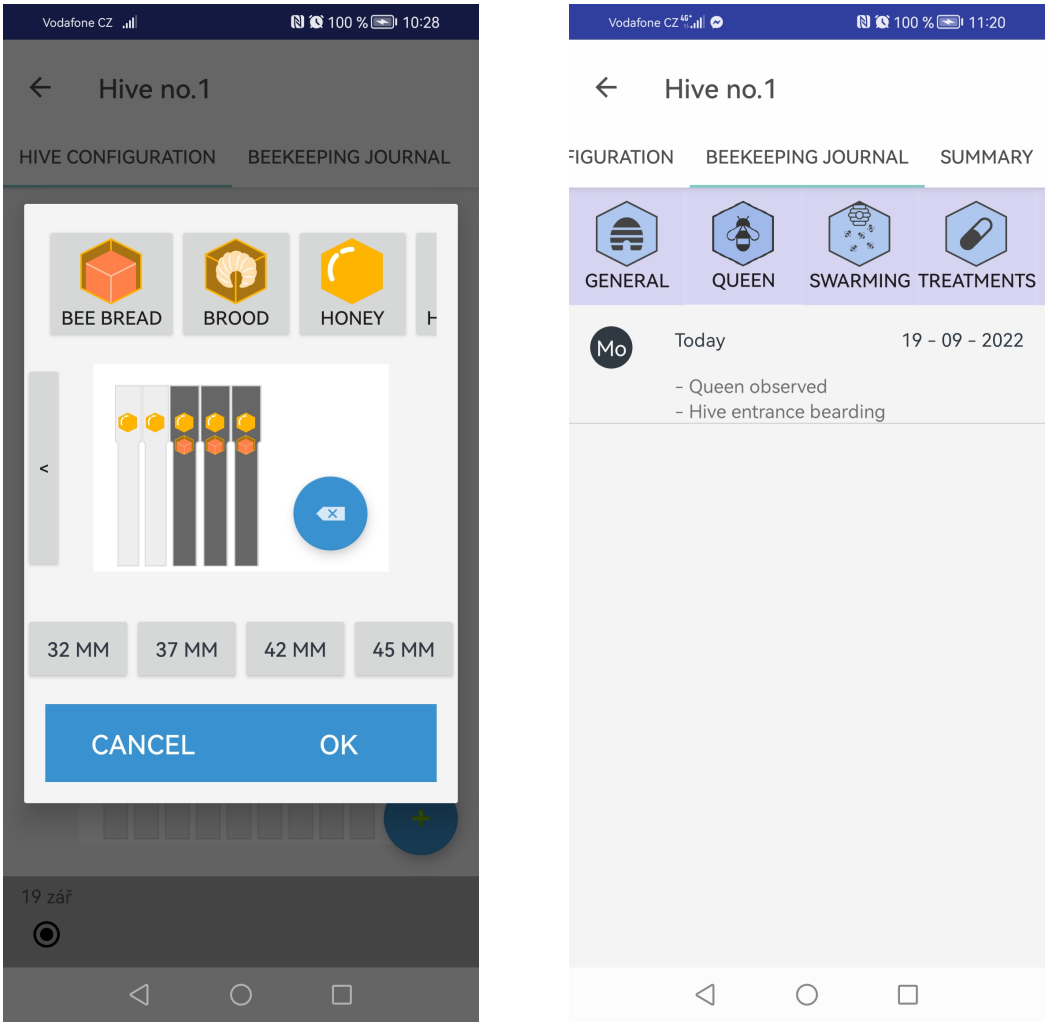
Aplikace je jednoduchá a lehce ovladatelná. Jak vertikální, tak horizontální zobrazení je celkem přehledné. Poskytuje detailní správu úlu a jeho struktury s možností nastavení obsahu jednotlivých rámků, viz obrázek 3.2(a). Umožňuje vytvoření zálohy pomocí Gmail účtu. Obsahuje mnoho filtrů pro zobrazení úlů. Zobrazuje přehledně detail úlu s grafy a časovou osou. Umožnuje vytvářet skupiny. Jeden úl může náležet více skupinám. Časová osa se změnami úlu je přehledná a včelařský deník se záznamy kontrol také, viz obrázek 3.2(b).

3.2.3 Nevýhody

Po expiraci zkušební verze je nutnost platit předplatné. Mnoho funkcí je dostupných až po zaplacení předplatného. Přidání nového úlu do skupiny je zpočátku náročné najít. Není možné zaznamenat fotografie nebo geografickou polohu jednotlivých úlů. Jednotlivé úly jsou v aplikaci reprezentovány pouze podle identifikačního čísla, což je nepřehledné.

3.2.4 Ukázky aplikace

Obrázky 3.2 zobrazují ukázky obrazovek z mobilní aplikace:



(a) Úprava rámů úlu (b) Včelařský deník

Obrázek 3.2: Ukázky aplikace BoxyBee

3.3 Apiary Book

Základní informace:
Požadovaný operační systém: Android 4.1+
Poslední aktualizace: 18.10.2023
Verze: 6.9
Počet stažení: +100 000
Jazyky: angličtina, čeština, další

3.3.1 Funkce

Aplikace obsahuje následující funkce:

- deník se záznamy o prohlídkách, včelínech, úlech, rámcích,
- správa včelínů, úlů, rámků (počet, stav, váha, poznámky),
- plánování činností a prohlídek,
- aktuální počasí a budoucí předpověď,
- evidence léčení, krmení, produktů (medu, vosku atd.), včelích matek, inventáře nástrojů,
- tisk a export dat do formátu PDF a CSV,
- úložiště fotek a videí,
- identifikace NFC a QR kódu,
- možnost vyhledávání a filtrace v záznamech,
- kalkulačka na přípravu cukerného roztoku,
- webová aplikace obsahuje mnoho užitečných informací.

Předplatné

Aplikace je v základní verzi zdarma a nabízí omezenou správu včelínů a úlů (1 včelín, 9 úlů), základní grafy a reporty, hlasového asistenta, notifikace, zálohy a analýzu lokáních dat a doporučení a tipy. Pro uživatele s předplatným se rozšiřuje množství spravovaných včelínů a úlů, možnost přístupu z více zařízení, pokročilejší reporty a analýzy nebo například mentorský program a doporučené postupy.

3.3.2 Výhody

Obsahuje velké množství nastavení aplikace. Umožňuje upravit si vzhled uživatelského rozhraní, aby zobrazovalo jen ty části, které uživatel potřebuje. Umožňuje zálohování do zařízení nebo do cloudového úložiště, vyhledávání podle textu. Poradce poskytuje různá doporučení a odpovědi na dotazy. Spravuje záznamy o částech úlu (přidávání a odebírání nástavků/rámků). Je možné stát se poradcem.

Sdílí informace o stanovištích přes Facebook[1] nebo e-mail. Umožňuje komunikaci s ostatními uživateli (pouze pro uživatele s placenou verzí aplikace). Zobrazuje počasí podle GPS polohy stanoviště, viz obrázek 3.3(a). Aplikaci je možné ovládat pomocí hlasových povelů (počasí, poznámka, vytvoření úlu, provést léčení, zadání nové činnosti, veterinární prohlídka, připomínka). Umí zkopírovat celý úl bez nutnosti ho znovu vytvářet. Nabízí možnost přiražení úlu jinému stanovišti. Obsahuje velký výběr detailního popisu činností např. krmení, viz obrázek 3.3(b). K dispozici je pomocný asistent s doporučeními např. pokud je včelstvo slabé.

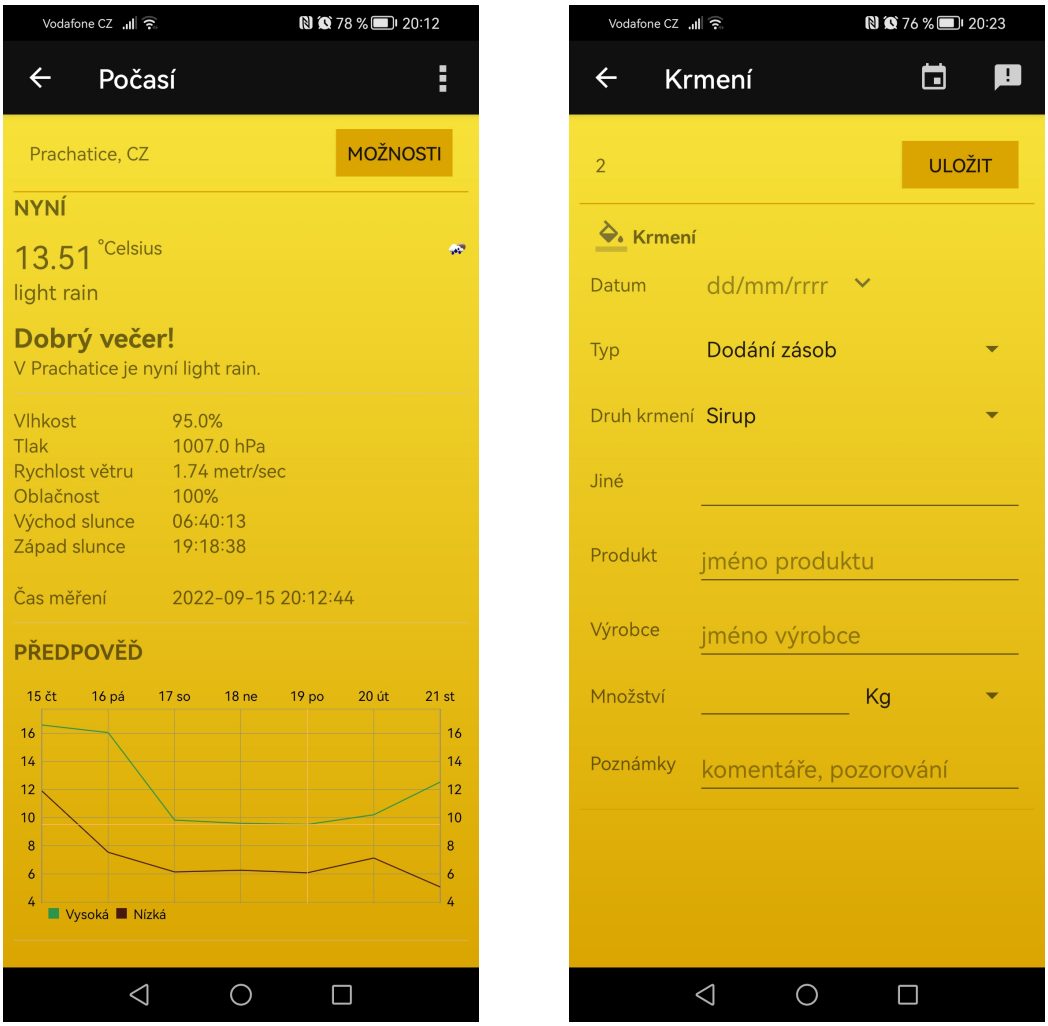
3.3.3 Nevýhody

Uživatelské rozhraní je nepoutavé a občas neintuitivní (přidání nového úlu do stanoviště nebo přidání nového stanoviště). Některé funkce jsou dostupné až po přihlášení a některé dokonce až po zaplacení předplatného (návody pro včelaře, oznámení, poradce, komunikace s dalším uživatelem). Písmena s háčky a čárkami se zobrazují špatně kvůli jiné znakové sadě. Pro bezplatnou verzi je možné spravovat jen jedno stanoviště/včelín. Některá slova nejsou vůbec přeložena.

3.3.4 Ukázky aplikace

Obrázky 3.3 zobrazují ukázky obrazovek z mobilní aplikace:

[1] Sociální síť pro komunikaci a sdílení multimediálních dat.



(a) Počasí (b) Krmení

Obrázek 3.3: Ukázky aplikace Apiary Book

3.4 Chytrý včelař

Základní informace:
Požadovaný operační systém: Android 7.0+, iOS 14.0+
Poslední aktualizace: 6.10.2023
Verze: 1.30.1
Počet stažení: 500 000+
Jazyky: čeština

3.4.1 Funkce

Aplikace obsahuje následující funkce:

- správa jednotlivých stanovišť (název, poloha, adresa, registrační číslo, datum založení, fotografie),
- správa úlů se základními informacemi (název, typ, barva, datum založení, typ včelstva, míra rámku, fotografie),
- deník záznamů o léčení, krmení, prohlídkách úlů (stav plodů, síla včelstva, nálada, nemoci),
- evidence včelstev (informace, stav, krmení, počet, ošetření, léčení), produktů (medu), včelích matek (datum vylíhnutí, původ, linie, úhyn),
- automatické zobrazení počasí podle polohy stanoviště,
- přidávání fotografií ke stanovištím a prohlídkám,
- včelařská kalkulačka pro přípravu cukerných roztoků,
- automatický výpočet spadu roztoče z fotografie podložky úlu,
- nahlášení sršně asijské.

3.4.2 Výhody

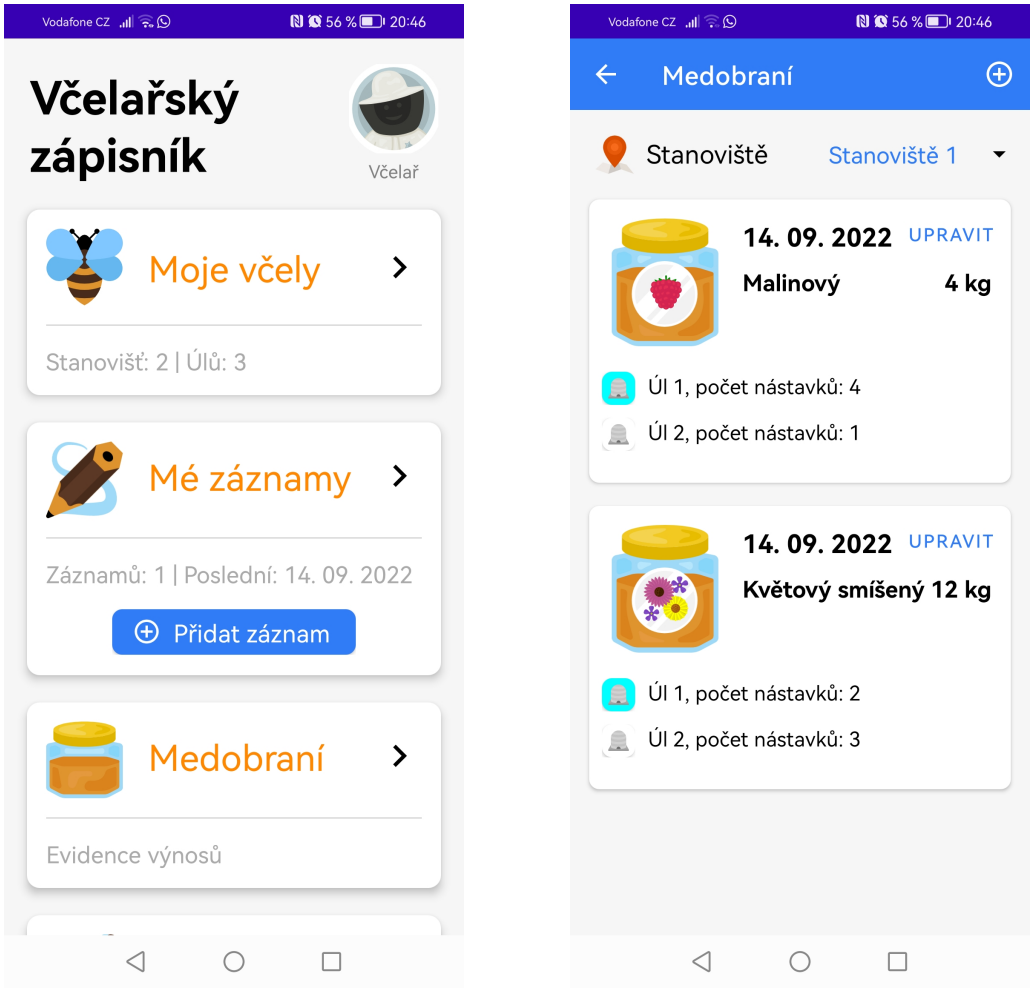
Aplikace pochází od českého vývojářského týmu. Manipulace s aplikací je snadná a uživatelské rozhraní působí velmi poutavě, viz obrázek 3.4(a). Po přihlášení jsou k dispozici všechny funkcionality a navíc je aplikace zcela zdarma. Umožňuje sledovat záznamy na jiném zařízení podle přihlašovacího jména. Zaznamenává poznámky ke stanovištím, úlům, matkám. Velmi rozsáhlý je i filtr pro vyhledávání a zobrazení záznamů. Dovoluje vlastní řazení jednotlivých stanovišť. Je možné přidávání poznámek, fotek či aktuálního počasí na jednotlivá stanoviště. Každé stanoviště může obsahovat libovolný počet úlů. V záznamech o produkci medu je možné zaznamenat nejen typ medu, ale i počet nástavků, viz obrázek 3.4(b). Vývojáři vytvořili několik dalších menších aplikací pro podporu včelařů jako je aplikace pro nahlášení sršně asijské, včelí kalkulačka nebo kalendář včelích matek pro plánování a chov.

3.4.3 Nevýhody

Uživatelské rozhraní občas působí stroze např. přihlašovací a registrační obrazovka. Je nutné přihlášení před použitím aplikace. Není možné používat aplikaci bez přístupu k internetu v offline módu bez předchozího přihlášení. Při registraci nejsou nikde uvedeny požadavky na heslo. Umožňuje zadání velmi slabého hesla (5 znaků). V některých obrazovkách se zobrazuje pouze část textu. Při používání se aplikace dlouho načítala nebo úplně zamrzla. Matku je možné přidat do úlu až po založení a ne při jeho vytváření. Medobraní obsahuje automatický filtr.

3.4.4 Ukázky aplikace

Obrázky 3.4 zobrazují ukázky obrazovek z mobilní aplikace:



(a) Hlavní menu (b) Medobraní

Obrázek 3.4: Ukázky aplikace Chytrý Včelař

3.5 Včelař

Základní informace:
Požadovaný operační systém: Android 4.1+
Poslední aktualizace: 17.1.2016
Verze: 1.5.2
Počet stažení: 1 000+
Jazyky: čeština

3.5.1 Funkce

Aplikace obsahuje následující funkce:

- správa jednotlivých úlů s informacemi o názvu, pořadí, barvě úlu, umístění, popis,
- přidávání fotografií k jednotlivým úlům,
- poznámky s časovou osou změn,
- záznamy o léčení,
- včelařův rok s informacemi obsahujícími rady a informace o jednotlivých měsících,
- informace o značení včelích matek.

3.5.2 Výhody

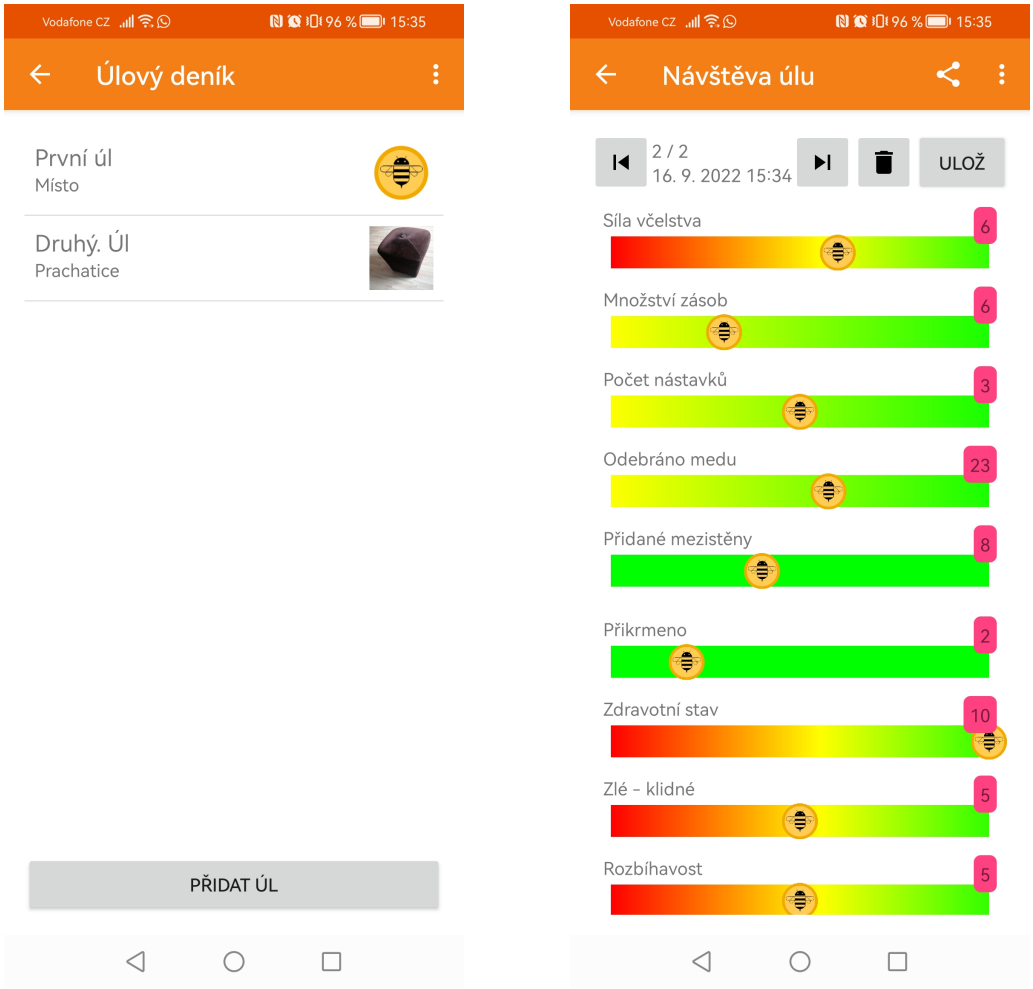
Aplikace má intuitivní uživatelské rozhraní. Parametry prohlídky úlu se zadávají na škále, viz obrázek 3.5(a). Tutoriály v aplikaci obsahovaly mnoho užitečných tipů (co se má splnit do určitého termínu). Deník obsahuje možnost zaznamenávat si prohlídky na jednotlivých stanovištích, viz obrázek 3.5(b) a prohlížet si je v časové ose se zobrazenými změnami. Nabízí možnost určit pořadí jednotlivých úlů.

3.5.3 Nevýhody

Celkové uživatelské rozhraní bylo nedostatečně atraktivní na pohled. Vertikální zobrazení není moc přehledné. Do aplikace je nutné přihlásit se přes Google účet. Aplikace není možné používat bez internetového připojení (offline mód) při prvním použití aplikace. Aplikace se chovala zvláště a přestávala reagovat při některých činnostech (přidání nového záznamu léčení, včelařův rok). Při vytváření úlu nebylo jasné k čemu slouží mapa, jelikož se s ní nedalo pracovat a pohyb po ní nebyl možný. Při vytváření úlu není možné vybrat fotografii z galerie. Aplikace přestala reagovat v určitých chvílích po stisknutí tlačítka zpět. Malé množství atributů úlu. Po vytvoření nového úlu, nejsou informace dále viditelné uživateli. Editace a zobrazení informací o úlu je pro uživatele těžké najít. Není možnost dávat více úlů do jedné skupiny. Aplikace často přestala reagovat a kvůli tomu bylo velmi náročné ji vůbec použít.

3.5.4 Ukázky aplikace

Následující obrázky zobrazují vybrané obrazovky z mobilní aplikace:



(a) Úlový deník (b) Záznam prohlídky

Obrázek 3.5: Ukázky aplikace Včelař

3.6 Včelaření

Základní informace:
Požadovaný operační systém: Android 4.4+
Poslední aktualizace: 15.2.2018
Verze: 2.0
Počet stažení: 1 000+
Jazyky: čeština

3.6.1 Funkce

Aplikace obsahuje následující funkce:

- shrnutí zobrazující statistiky, počet včelstev, produkci medu s grafy o celkovém a průměrném počtu pro daný rok,
- správa úlů (typ, rámky, název), matek, událostí, financí,
- deník se záznamy o úlech,
- záznamy o vytvořených produktech, včelích matkách, událostech, onemocnění,
- tvorba vlastních parametrů (např. temperament včel), hodnocení, typu událostí,
- evidence zákazníků,
- zálohování a obnovení dat.

3.6.2 Výhody

Aplikace funguje i bez internetového připojení. Má jednoduché a přehledné uživatelské rozhraní, viz obrázek 3.6(a). Umožňuje vytvoření vlastních parametrů a rozsahu hodnot daného parametru např. velikost matky se škálou 1-5, kde hodnota 1 představuje malou a 5 velkou. Obsahuje velké množství atributů při vytváření nového úlu nebo včelí matky. Je zde možnost sledovat vlastnosti u včelích matek. Lze vytvářet vlastní události. Řadí záznamy podle mnoha filtrů a kritérií. U včelí matky je možné přidat její matku. Zobrazuje informace pomocí grafů např. průměrný denní spad roztočů[2] na dno úlu. Záznamy je možné řadit podle podle různých kritérií. Nedoporučuje zákazníky v prodejích, např. už mu neprodáváme. Deník se záznamy je zobrazen velmi přehledně společně s důležitými informacemi, viz obrázek 3.6(b).

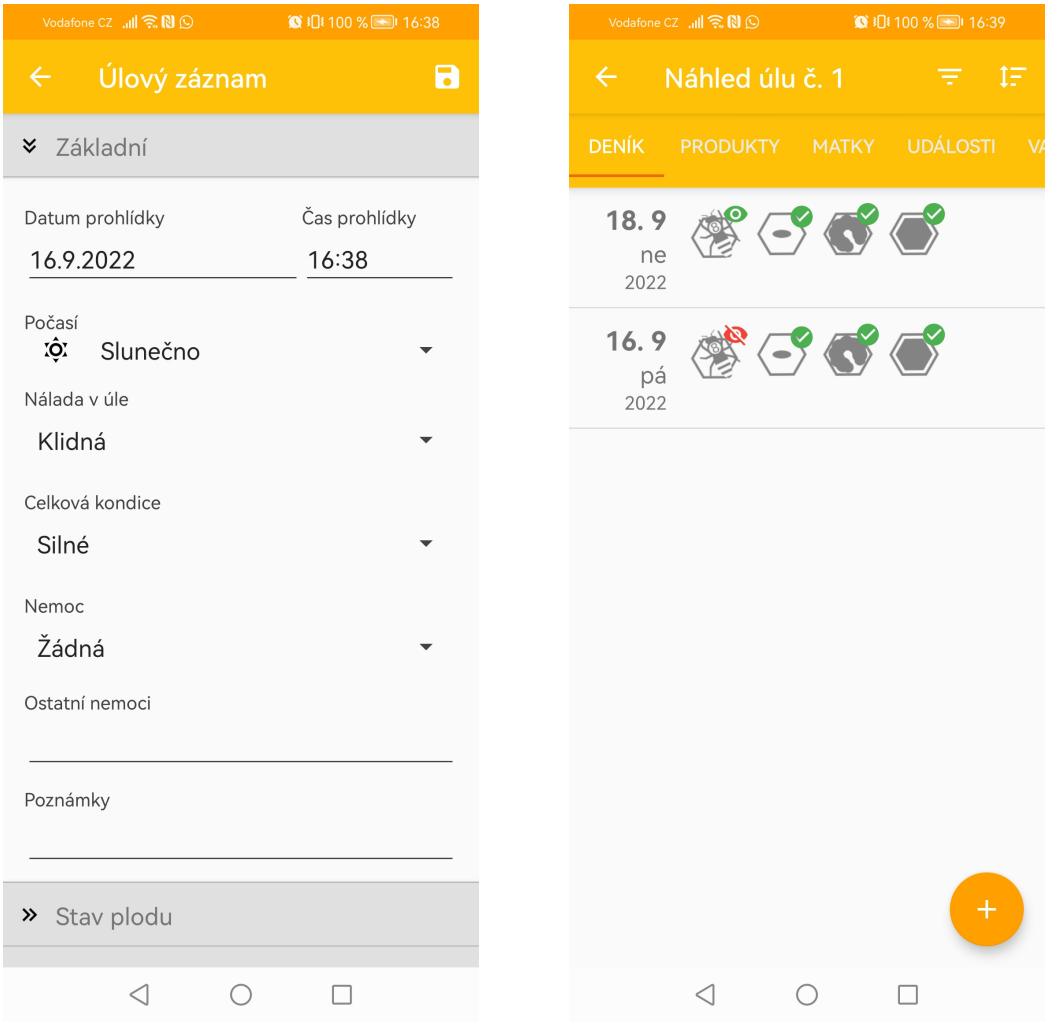
3.6.3 Nevýhody

Aplikace se občas zasekla. Nebylo možné zálohovat data kvůli chybě (Permission denied). Nebylo možné vytvořit novou událost, pokud předtím nebyl vytvořen minimálně jeden vlastní typ události. Zpětné tlačítko v některých obrazovkách nevracelo aplikaci na poslední obrazovku ale vypnulo celou aplikaci nebo byla přesunuta do pozadí. Po jakýchkoliv úpravách bylo nutné změny potvrdit tlačítkem, jinak změny nebyly uloženy.

[2]Spad je množství roztočů Varroa destructor, kteří padají z včel na podložku úlu např. při ošetřování. Měření spadu roztočů může poskytnout včelařům užitečné informace o úrovni infekce ve včelstvu a účinnosti používaných metod ošetření.

3.6.4 Ukázky aplikace

Následující obrázky zobrazují vybrané obrazovky z mobilní aplikace:



(a) Úlový záznam (b) Inspekce

Obrázek 3.6: Ukázky aplikace Včelaření

3.7 Shrnutí

Shrnutí jednotlivých aplikací z několika možných pohledů je znázorněno v tabulce 3.1. Jaké jsou minimální požadavky na jednotlivé platformy, jakou poskytuje funkcionalitu na vedení včelařských záznamů z kontroly včelích úlu, jaké další funkce poskytuje. Dalším hlediskem je orientace v uživatelském rozhraní a zda aplikace obsahuje placené předplatné.

Každá aplikace postrádá, alespoň jednu část, kterou by každá aplikace pro včelaře měla obsahovat, jako je například komunikace mezi včelaři nebo základy včelaření. Většina aplikací obsahuje pouze jeden jazyk (většinou pouze čeština nebo angličtina). Starší generace by určitě ocenila češtinu a naopak mladší raději používá angličtinu.

Ani jedna z vybraných aplikací neobsahuje žádnou možnost integrovat jakékoliv senzory, či jiná zařízení pro monitorování aktuálních dat z úlu. Některé aplikace mají příliš omezenou funkcionalitu, dostupnou pouze při zaplacení předplatného např. maximální počet evidovaných včelínů a úlů. Většina aplikací postrádala jakoukoliv komunikaci mezi jednotlivými včelaři, či novinky a aktuální trendy. Některé aplikace jsou již zastaralé a nedostaly v poslední době žádnou aktualizaci. Zadávání nových záznamů bylo mnohdy nepřehledné z důvodu velkého množství nerelevantních údajů. Jiné aplikace byly dokonce z oficiálních obchodů (Google Play, App Store) staženy a není nadále možné je získat (např. BoxyBee).

Celkově je opravdu omezený počet mobilních aplikací, které se zabývají včelařením, je opravdu omezený počet obsahujících dostatečné parametry. Z tohoto důvodu by bylo vhodné vytvořit aplikaci zdarma s přehledným a jednoduchým uživatelským rozhraním. Aplikace by obsahovala více jazyků a také snadnou správu včelínů, úlů a inspekcí. Dále by umožnila komunikaci včelařů, kde by začátečníci mohli získávat cenné rady od starších zkušenějších uživatelů, či se poradit s případnými problémy. Obsahovala by možnost přidávat jednotlivé uživatele do přátel a dala jim tak možnost nahlédnout do jejich úlů a záznamů. V neposlední řadě by mohla mít možnost integrovat senzory pro monitorování včelstva.

Jelikož je v poslední době existence včel ohrožena například klimatickými podmínkami nebo lidskými dopady, je důležité využít co nejvíce technologických prostředků, které podpoří zdraví a ochranu včel a další rozvoj v oblasti včelaření.

Funkce/Aplikace	APIManager	BoxyBee	Apiary Book	Chytrý včelař	Včelař	Včelaření
Android	5.1+	5.1+	4.1+	7.0+	4.1+	4.4+
iOS	10+	-	-	14.0+	-	-
Webová aplikace	Ano	Ne	Ano	Ano	Ne	Ne
Správa včelínů	Ano	Ano	Ano	Ano	Ne	Ne
Správa úlů	Ano	Ano	Ano	Ano	Ano	Ano
Inventář	Ano	Ne	Ano	Ne	Ne	Ne
Finance	Ano	Ne	Ano	Ne	Ne	Ano
Včelarský deník	Ano	Ano	Ano	Ano	Ano	Ano
Včelí matka	Ano	Ano	Ano	Ano	Ne	Ano
Krmení	Ano	Ne	Ano	Ano	Ne	Ne
Léčení	Ano	Ne	Ano	Ano	Ano	Ano
Medobraní	Ano	Ano	Ano	Ano	Ne	Ano
Události/činnosti	Ano	Ne	Ano	Ne	Ne	Ano
Statistiky	Ano	Ano	Ano	Ne	Ne	Ano
Notifikace	Ano	Ne	Ano	Ne	Ne	Ne
Export dat	Ne	Ne	PDF/CSV	Ne	Ne	Ne
Kalkulačka	Ne	Ne	Ne	Ano	Ne	Ne
Počasí	Ne	Ne	Ano	Ano	Ne	Ne
Tipy/Rady	Ne	Ne	Ano	Ne	Ano	Ne
Mediální soubory	Ano	Ne	Ano	Ano	Ano	Ne
NFC/QR kód	Ano	Ne	Ano	Ne	Ne	Ne
Orientace v UI	Velmi dobré	Ucházející	Dobré	Velmi dobré	Ucházející	Výborné
Předplatné	Nepovinné	Nepovinné	Nepovinné	Ne	Ne	Ne

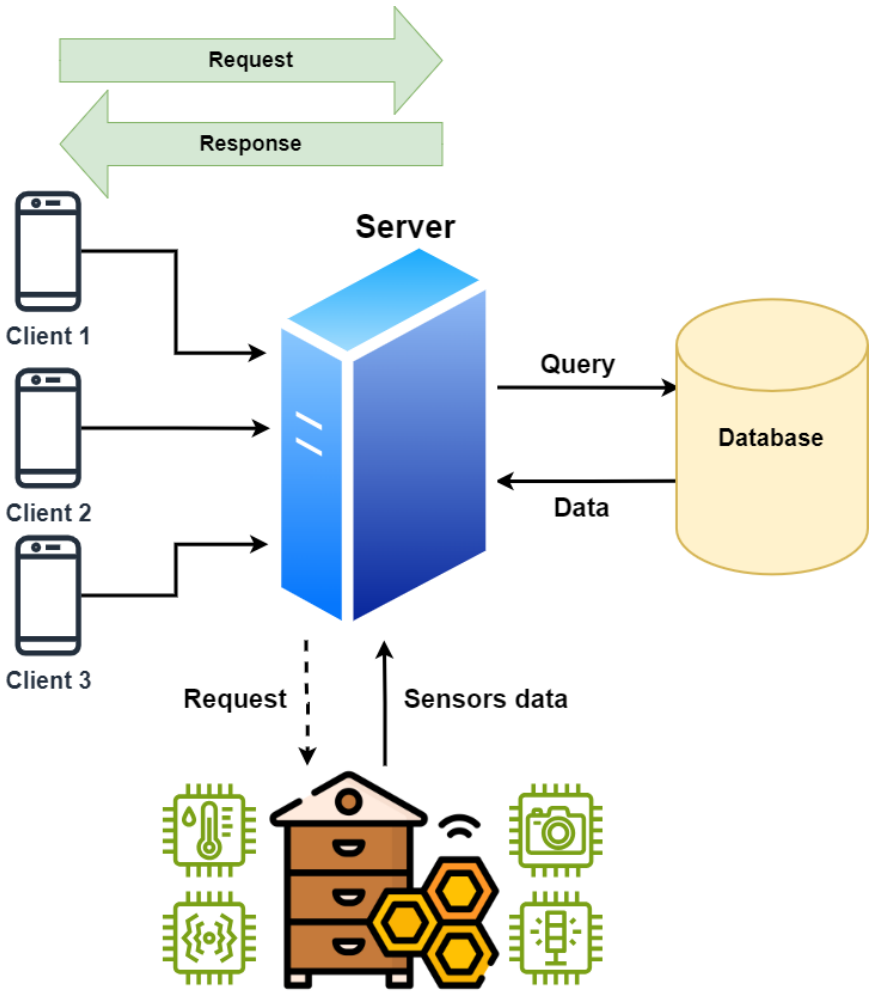
Tabulka 3.1: Souhrn funkcionalit aplikací

Ano - obsahuje danou část, Ne - neobsahuje danou část

Rámcový návrh systému

4

Systém se bude skládat z několika částí, a to z komunikačního serveru, databáze a mobilního klienta. Komunikace mezi klienty a serverem s databází je znázorněna na obrázku 4.1.



Obrázek 4.1: Návrh systému pro podporu včelařů

4.1 Komunikační server

Komunikační server bude v této práci sloužit jako platforma pro síťovou komunikaci a výměnu dat mezi databází a mobilní aplikací. Bude definovat API rozhraní jednotlivých endpointů[1], které budou poskytovat informace o jednotlivých včelínech a úlech včelaře. Jelikož aplikaci bude využívat více uživatelů, proto je potřeba zavést mechanismus pro ověření identity. Dále bude komunikační server přijímat data ze senzorů, které budou monitorovat aktuální stav v úlu. Všechna data budou uložena do databáze. Při požadavku budou mobilnímu klientovi poskytnuta uložená data.

4.2 Mobilní zařízení

Mobilní zařízení bude použito pro získávání a posílání dat komunikačnímu serveru. Může také snadno poskytnout důležité informace v reálném čase a zaznamenávat včelařské operace pomocí jednoduchého intuitivního rozhraní. S rostoucím vývojem technologií a chytrých zařízení se otevírají nové možnosti pro včelaře, jak co nejefektivněji spravovat svá včelstva a monitorovat zdravotní stav včel. Mobilní zařízení v dnešní době vlastní téměř každý a představují tak velmi užitečný nástroj pro toto využití.

4.3 Senzory

Senzory budou monitorovat stav včelího úlu. Výběr senzorů bude záležet na konkrétních hodnotách, které je vhodné monitorovat, viz kapitola 8. Naměřené hodnoty se budou posílat komunikačnímu serveru pro uložení do databáze. V budoucnu by bylo možné, aby komunikační server mohl poslat požadavek a tím získal aktuální hodnoty dat ze senzorů.

[1] Funkce dostupná přes rozhraní, která spouští proceduru provádějící určitý úkol.

Použité technologie 5

V této kapitole bude popsán výběr jednotlivých technologií, které je možné použít při realizaci sytému. Použité technologie budou rozděleny do dvou kategorií, a to pro vývoj a provoz systému.

5.1 Vývoj systému

Technologie pro vývoj se budou zaměřovat především na urychlení vývoje a kvalitu zdrojového kódu.

5.1.1 Framework a programovací jazyk

Pro výběr technologií při vývoji komunikačního serveru je v současné době klíčová například výkonnost, bezpečnost, škálovatelnost a jednoduchost nebo komunitní podpora.

Jako základ pro vývoj výkonného a bezpečného komunikačního serveru bude použit framework[1] Spring Boot. Je založený na programovacím jazyce Java a je jedním z nejpoužívanějších frameworků současnosti, který díky vestavěným funkcionalitám velmi zjednoduší vývoj. Tento framework obsahuje zabudovaný webový server Tomcat, Jetty nebo Undertow, díky čemuž není potřeba nasazovat WAR soubory. Poskytuje automatickou konfiguraci, a tedy programátoři se mohou více soustředit na vývoj samotné aplikace než na konfiguraci.

Spring Boot obsahuje různé knihovny třetích stran, které ušetří programátorovi nejen čas, ale také umožní integrovat další nástroje či technologie, které více rozšíří aplikaci. Dále poskytuje předpřipravené funkce k použití, jako je externí konfigurace nebo kontrola běhu (health check). Framework neklade žádné požadavky na XML konfiguraci. Spring Boot má velmi silnou podporu pro zabezpečení aplikací, nabízí mnoho nástrojů pro autentizaci, autorizaci a další pro bezpečný provoz komunikační platformy. Díky vestavěným nástrojům jako je Lombok, který pomocí anotací

[1] Softwarová struktura pro vývoj, podporu programování a organizaci projektů.

generuje částí jako jsou gettery[2], settery[3], logování[4] a další, ušetří programátorovi mnoho času s manuálním psaním zdrojového kódu.

Daný framework obsahuje velmi rozsáhlou a přehlednou dokumentaci s podrobnými informacemi, ukázkami kódu a videi pro detailní popis. Za tímto frameworkem je také velká komunita vývojářů, připravených poradit s případnými problémy [10].

Spring Boot framework podporuje programovací jazyky Java a Kotlin. Oba jsou v současné době jedny z nejpoužívanějších programovacích jazyků. Java je univerzální jazyk, který je známý díky čitelnosti, přenositelnosti a velkému množství knihoven a frameworků. Java se používá pro:

- Android aplikace,
- webové aplikace,
- vestavěné systémy (Embedded Systems[5]),
- vědecké výpočetní systémy,
- desktopové aplikace.

Kotlin je na druhou stranu podstatně novější jazyk, který je interoperabilní s jazykem Java (Kotlin kód lze použít v projektu Java a naopak). Je známý především stručnou syntaxí, díky které se zdrojový kód lehce píše a udržuje. Nejčastěji se používá pro:

- mobilní aplikace,
- projekty datové vědy (Data Science[6]),
- multiplatformní aplikace.

Každý z programovacích jazyků má své oblasti použití. V porovnání výkonnosti obou jazyků je Kotlin o něco lepší než Java, jelikož se jedná o nový moderní jazyk navržený pro lepší efektivitu kódu. Kotlin má několik funkcí jako odvozování typů a inline funkce, které pomáhají zlepšit výkonnost kódu. Díky snazší syntaxi, lepší čitelnosti a stručnosti je kód méně náchylný k chybám. Avšak při reálném výkonu je mezi jednotlivými jazyky jen nepatrný rozdíl. Více jsou důležité faktory jako je návrh a optimalizace kódu, které mají na výkon aplikace větší vliv bez ohledu na použitý jazyk [11]. Pro realizaci komunikačního serveru bude použit programovací jazyk Java.

[2]Metoda pro získání hodnoty privátní proměnné objektu.
[3]Metoda pro nastavení hodnoty privátní proměnné.
[4]Způsob zaznamenávání událostí, stavů v běžícím systému.
[5]Jednoúčelový počítač ovládaný řídicím systémem, který je v něm zcela zabudován.
[6]Věda za účelem získání cenných informací z velkých objemů dat.

5.1.2 Analýza kódu

Pro statickou analýzu kódu bude použit nástroj SonarLint, který je již integrován do vývojových prostředí (Visual Studio Code, InteliJ IDEA, Eclipse a dalších). Hlavním cílem analýzy je především identifikace potencionálních chyb, nepřesností, duplikací nebo úprava stylu zdrojového kódu, díky čemuž se zlepší kvalita i čitelnost samotného kódu. Analýzu provádí na základě definovaných pravidel (nepoužitá proměnná, neefektivní konstrukce kódu, zranitelností knihoven, nedodržování standardů - styl pojmenování proměnných) [12]. Všechny nálezy statické analýzy jsou následně zvýrazněny ve vývojovém prostředí s popisem a doporučením, jak daný nález opravit.

5.1.3 Vývojové prostředí

Komunikační server bude vyvíjen ve vývojovém prostředí InteliJ IDEA Ultimate. Vývojové prostředí obsahuje širokou škálu nástrojů, funkcí nebo šablon, které byly dříve zmíněné, například podpora Spring Boot, Docker, databází, Maven, Git, SonarLint, které budou použity při vývoji. Pro zvýšení efektivity používá **deep learning model**[7] pro doplňováním celých řádek kódu na základě lokální kontextové analýzy. Na podobném principu funguje i inteligentní doplňování kódu, které poskytuje na základě kontextu návrhy doplnění kódu pro aktuální pozici kurzoru. Jako další užitečné zabudované nástroje jsou **Debugger**[8] pro odstranění chyb nebo **Profiler** umožňující prozkoumat výkon aplikace a identifikovat části náročné na procesor a paměť [13].

Mobilní klient bude vyvíjen ve VS Code (Visual Studio Code), které podporuje integraci s Flutter SDK. Podporuje tvorbu, provoz a ladění aplikace přímo ve vývojovém prostředí bez použití dalších nástrojů. Dále obsahuje mnoho rozšíření speciálně pro vývoj Flutteru (zvýrazněná syntaxe, doplňování kódu a další) [14].

5.1.4 Flutter

Existuje celá řada nástrojů pro návrh, tvorbu a optimalizaci aplikací cílené na mobilní zařízení (telefony, tablety). Na rozdíl od klasického vývoje softwaru počítače se musí brát v úvahu především různé vlastnosti mobilních zařízení, od velikosti obrazovky až po spotřebu baterie a dotykové rozhraní [15]. Pro vývoj mobilních aplikací se v současně době využívá mnoho frameworků,nejvíce používané jsou především Flutter, React Native [16] nebo Xamarin [17].

[7] Typ umělé neuronové sítě.

[8] Nástroj pro ladění a opravu chyb při běhu aplikace, které lze krokovat po jednotlivých příkazech, kde uživatel vidí hodnoty jednotlivých proměnných.

Flutter je open-source[9] obsahující sadu nástrojů pro vývoj uživatelského rozhraní od společnosti Google, který byl vytvořen především pro mobilní zařízení. Umožňuje vytvářet nativní aplikace pro systémy iOS a Android z jednoho zdrojového kódu [15]. Flutter má mnoho výhod pro vývoj mobilních aplikací:

- **Cross-Platform** - umožňuje vyvíjet pro více platforem současně (úspora času a zajištění konzistence).
- **Výkon** - kompiluje do nativního kódu procesoru (s vlastním grafickým enginem), který zajišťuje rychlé a optimalizované uživatelské rozhraní.
- **Hot Reload** - možnost sledovat změny v reálném čase bez nutnosti restartování aplikace (urychluje vývoj).
- **Knihovna balíčků** - obsahuje velké množství existujících balíčků volně k použití.
- **Komunita** - má velkou komunitu vývojářů, kde je možné získat radu s případnými problémy.
- **Bezpečnost** - obsahuje null-safety na syntaktické úrovni kódu a výpočetní vlákna se provádějí v izolovaných sandboxech.

Flutter používá programovací jazyk Dart. Dart využívá kompilátor AOT (Ahead Of Time) a kompilaci JIT (Just in Time). AOT používá ke generování strojového kódu, zatímco JIT se používá při vývoji pro rychlé obnovení bez nutnosti rekompilace [18].

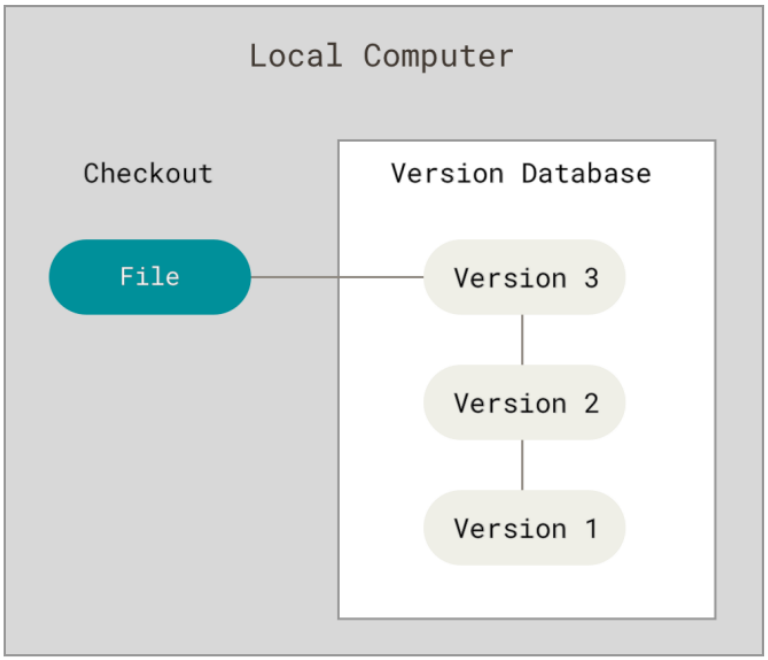
5.1.5 Správa verzí a Git

Správa verzí nebo-li *Version control* (VCS) je postup sledování a správy softwarového kódu. Jedná se o nástroje, které umožňují sledovat a spravovat změny zdrojového kódu v průběhu času [19]. Díky tomu je možné urychlit samotný vývoj. Používají se například pro týmy DevOps, aby jim pomohli co nejvíce zkrátit dobu vývoje a zvýšit úspěšnost nasazení aplikace. Zaznamenává každou změnu zdrojového kódu ve speciálním druhu databáze. Vývojáři mohou pracovat v oddělených větvích, čímž si nebudou zasahovat do práce. Dále je možné se vrátit ke konkrétní verzi kódu. Díky tomu se mohou opravovat rychleji chyby a zároveň se minimalizuje riziko narušení práce ostatních členů týmu [20]. Správa verzí pokrývá následující oblasti:

- Obsahuje kompletní historii změn každého souboru, viz obrázek 5.1.
- Každá změna obsahuje jméno autora, datum a poznámky.
- Souběžná práce více členů týmu je oddělena do větví, které se poté sloučí.

[9] Počítačový software s otevřeným zdrojovým kódem, který může kdokoliv libovolně upravovat a měnit.

- Umožňuje sledovat každou změnu provedenou na softwaru a propojit ji se softwarem pro řízení projektu a sledování chyb (např. Jira).



Obrázek 5.1: Historie změn na lokálním zařízení [20]

5.2 Provoz systému

Technologie pro běh systému se zaměří na nasazení a běh samotného systému. Prioritou takových technologií je minimalizace manuální práce programátora.

5.2.1 Docker

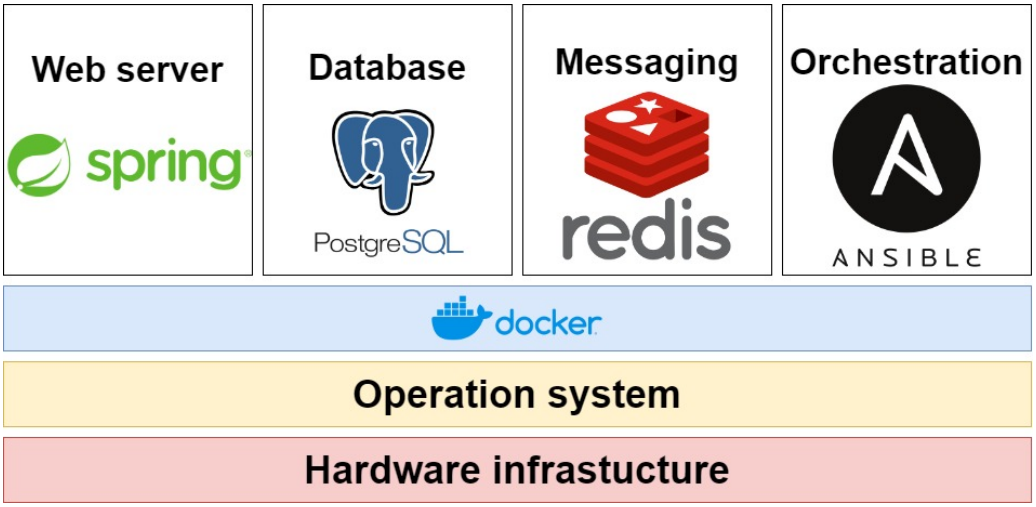
Pro vývoj při rychlém rozvoji moderních softwarových technologií je vhodné použít nástroje pro efektivnější tvorbu, testování a nasazování aplikace. Nástroj se využívá pro vytváření kontejnerů pro jednotlivé aplikace, ale zároveň jako správa vývojových prostředí. Jednotlivé části aplikace jsou odděleny do jednotlivých kontejnerů, které jsou samostatně zapouzdřená prostředí pro běh dané aplikace, viz obrázek 5.2.

Docker je v současné době jeden z nejpoužívanějších kontejnerových softwarových nástrojů. Jednotlivé kontejnery jsou izolované a mají konzistentní prostředí. Díky tomu je zajištěna konzistence v rámci vývoje mezi různými prostředími (vývojové, testovací, produkční). Docker umožňuje zabalení celé aplikace do jednoho kontejneru, kde mohou být již připravené všechny závislosti a tedy mezi jednotlivými typy prostředí nebude žádná nekonzistence. Díky tomu je možné se obejít

bez konfigurace na každém jednotlivém prostředí. Dále Docker poskytuje možnost správy zdrojů a škálovatelnost aplikace. Díky tomu lze aplikaci libovolně škálovat vytvořením více instancí stejného kontejneru nebo úpravou přidělených zdrojů (CPU a paměti) v konfiguraci kontejneru. Díky kontejnerizaci je mnohem jednodušší připravit testovací prostředí se stejným konfiguračním nastavením jaké používá produkční prostředí a tak zjednodušit proces testování a identifikaci problému v rané fázi vývoje [21].

Docker poskytuje nástroje a platformu pro správu životního cyklu kontejnerů:

- Vývoj aplikace a jejích podpůrných komponent pomocí kontejnerů.
- Kontejner představuje jednotku pro vývoj, testování i distribuci aplikace.
- Jednoduché nasazení aplikace na produkční prostředí.



Obrázek 5.2: Ukázka Docker architektury

Image

Základem každého kontejneru je *image* nebo-li obraz. Tento termín se používá také ve virtuálních strojích, kde se jako obraz disku používá soubor, který je reprezentován jako pevný disk. Oproti virtuálním strojům se image spuštěného kontejneru nikdy nezmění a představuje šablonu pouze pro čtení. Místo toho jsou všechny přidané nebo upravené soubory v samostatném souborovém systému, který je následně namapován na hostitelském stroji. Díky neměnícímu obrazu a proměnlivým souborům kontejneru je pak možné z tohoto obrazu vytvořit libovolný počet kontejnerů a spouštět je současně. Velké množství již existujících obrazů je volně dostupné na webu **Docker Hub**. Tyto obrazy je možné libovolně upravovat, či je kombinovat do dalších [22].

Kontejner

Jedná se o spustitelnou instanci image. Každý kontejner je možné spustit, zastavit, přesunout nebo odstranit nezávisle na spuštěné bitové kopii nebo ostatních spuštěných instancí stejného image. Každý kontejner je možné připojit k jedné nebo více sítím nebo z něj vytvořit nový image na základě jeho aktuálního stavu.

5.2.2 Databázový systém

Databáze je organizovaná kolekce dat, která jsou obvykle uložena a zpřístupněna elektronicky pomocí počítačového systému [23]. V moderním informačním systému je ukládání dat nezbytnou součástí. Databázový systém představuje klíčový nástroj pro efektivní uložení, analýzu a zpracování dat. Kolekce dat může představovat podniková data obsahující oblasti jako jsou zaměstnanci, budovy, zásoby, objednávky a další.

V této práci bude databáze klíčovou složkou infrastruktury. Bude se jednat o centrální úložiště všech dat poskytnutých jednotlivými uživateli. Některá data budou v aplikaci sdílená a přístupná ostatním uživatelům. Bude prováděno periodické zálohování dat, které zajistí bezpečnost a ochranu proti nechtěným ztrátám v případě selhání serveru, či jiným neočekávaným událostem. Databáze dále nabízí možnost vyhledávat a filtrovat data, což přispěje efektivnosti celého systému. Pro přístup a práci s daty v databázi se využívá systém řízení báze dat.
Systém řízení báze dat se stará [23]:

- zpracovává aktualizace prováděné koncovými uživateli a aplikacemi,
- zamezuje nežádoucím změnám dat pomocí integrity,
- umožňuje bezpečný přístupu k datům více uživatelům současně,
- v případě poruchy softwaru zajišťuje úplný konzistentní stav dat.

PostgreSQL

Bezplatný objektově-relační databázový systém (open-source), který využívá a rozšiřuje jazyk SQL [24]. Umožňuje vytvářet například vlastní datové typy, funkce a psát kód z různých programovacích jazyků. Při výběru databázového systému bylo přihlédnuto také k tomu, že autor práce má s tímto databázovým systémem nejvíce zkušeností.

5.2.3 Autentizace a autorizace

Autentizace a autorizace jsou dnes považovány za kritické aspekty webových služeb. Ať už se jedná o přihlášení do sociální sítě, bankovního účtu nebo e-mailu. V současné době je velmi jednoduché vydávat se na internetu za někoho jiného. V

digitálním světě je dnes běžné, že webová stránka či samotný účet může být odcizen nebo zneužit hackery. To představuje riziko úniku citlivých informací jako jsou údaje o kreditní kartě nebo bankovním účtu, které mohou být prodány například na "dark webu"[10]. Cílem autentizace je ověření identity uživatele a udržet citlivé informace v bezpečí [25]. Existuje několik možností autentizace:

- **Heslo** - ověření na základě hesla, ze kterého je vytvořen otisk (hash) pomocí algoritmu (např. SHA-1), ten je uložen do databáze a dále se porovnávají jen otisky hesla.
- **Vícefaktorová** - více vrstev a fází autentizace (heslo, jednorázové heslo, e-mailové ověření a další).
- **Certifikát** - identifikace osob, strojů a zařízení, které obsahují digitální identitu včetně veřejného klíče a digitální podpis certifikační autority (důvěryhodná certifikační autorita může vydávat digitální certifikáty ověřující vlastnictví veřejného klíče, které jednotlivé strany věří).
- **Biometrické prvky** - využívá jedinečné biometrické charakteristiky uživatele (otisk prstu, rozpoznání obličeje, rozpoznání řeči, chůze či pohybu uživatele).
- **Token** - po odeslání přihlašovacích údajů obdrží jedinečný zašifrovaný řetězec znaků a namísto opětovného zadávaní údajů může použít token pro přístup ke chráněným datům (Token ověřuje, že byl přístup uživateli již udělen).

V této práci se uživatel bude autentizovat pomocí e-mailové adresy a hesla. Po úspěšném přihlášení bude uživateli vygenerován token, který se bude posílat s dalšími požadavky.

[10] Část internetu přístupná pouze prostřednictvím anonymizačních nástrojů, kde se provozují nelegální aktivity (prodej drog, zbraní apod.).

Návrh komunikačního serveru 6

Cílem této části práce je návrh komunikačního serveru s využitím ověřených postupů. Kapitola popisuje jakým způsobem bude realizována architektura serveru a tvorba aplikačního rozhraní. Dále jakým způsobem se budou přijímat požadavky od klientů a jakým způsobem se bude ověřovat identita jednotlivých uživatelů.

6.1 Architektura

Architektura serveru bude typu klient-server. Tato architektura odpovídá navrhovanému systému, kde jednotliví klienti (uživatelé mobilní aplikace) budou posílat požadavky na data serveru [26].

V současné době se využívá několik architektonických stylů pro psaní API rozhraní jako jsou např. REST, SOAP, GraphQL, gRPC, WebSocket nebo Webhook. Každý architektonický styl má určitě výhody a nevýhody. V této práci bude použit REST API [27]. Jedná se o způsob použití webových standardů pro práci s daty na serveru. REST API je populární a snadno implementovatelný styl, se kterým se denně setkáváme (Youtube, Twitter a další webové služby). Využívá RESTful API, což je styl pro návrh API, který je založen na principu jednoduchých a jednoznačných operací, které pracují s datovými zdroji. Posílaná data jsou ve formátu JSON nebo XML. Architektura REST API je popsána na obrázku 6.1 Nejvíce se tento styl používá na datově orientované aplikace a webové služby, kde je prioritou rychlost a škálovatelnost. REST API využívá metody HTTP:

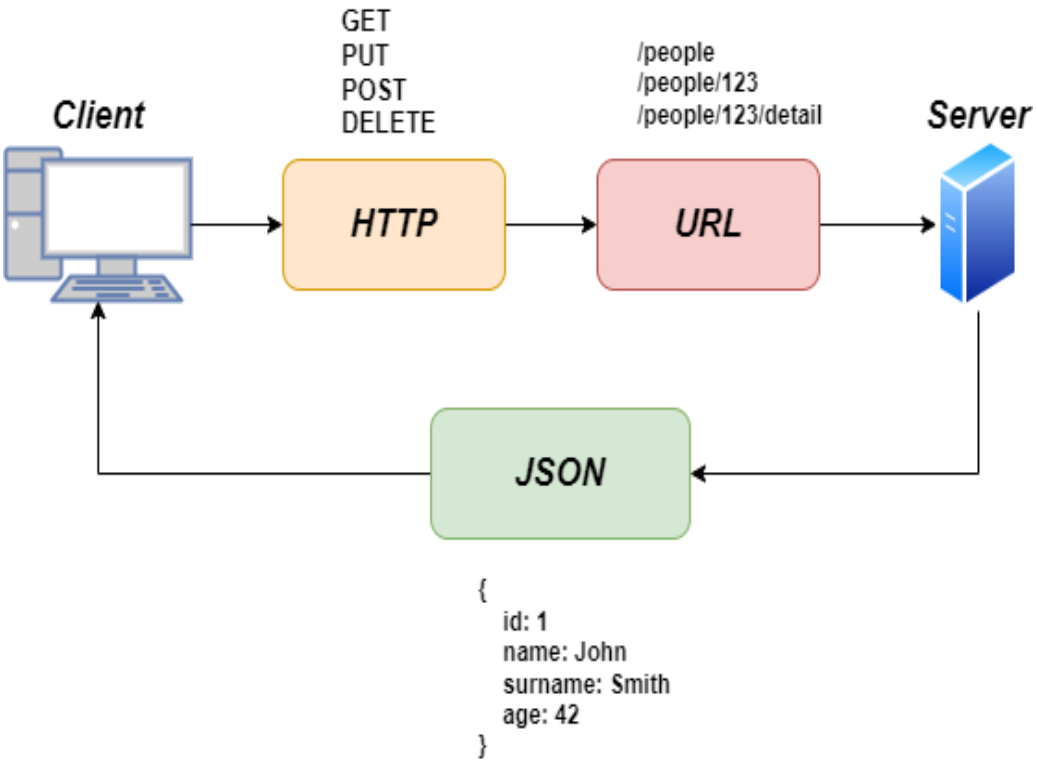
- **GET** - získání dat,
- **POST** - vytvoření dat,
- **PUT** - změna dat,
- **DELETE** - odstranění dat,
- **PATCH** - změna dat, pokud existují.

Výhody

- Jednoduchý a používá se snadno.
- Flexibilní s možností použít různé formáty.
- Je škálovatelný a web-friendly.
- Podporuje bezstavovost a ukládání do mezipaměti.
- Klienti a servery jsou na sobě nezávislí.

Nevýhody

- Nemá jasný kontrakt ani schéma (nepřehlednost, nekonzistence).
- Nepodporuje složité dotazy nebo operace (více malých požadavků).
- Nepracuje dobře s chybami nebo výjimkami, kvůli stavovým kódům HTTP, které nejsou vždy jasné či správné.



Obrázek 6.1: REST architektura (zdroj: [28])

Tvorba samotného REST API se dá shrnout do několika na sebe navazujících kroků [29]:

1. **Identifikace zdrojů** - první krok při tvorbě API je identifikace objektů nebo modelů, které budou zdroji, např. včelín, úl.
2. **Vytvoření URI endpointu** - pokud jsou identifikované zdroje, vytvoří se dále URI pro jednotlivé zdroje například:
   - */apiaries* - vrátí seznam všech včelínů,
   - */apiaries/{id}* - vrátí detail včelínů daného id,
   - */apiaries/{id}/beehives* - vrátí seznam úlů včelínu s daným id.
3. **Reprezentace zdrojů** - po vytvoření schéma endpointů se pro každý vytvoří reprezentace ve formátu XML nebo JSON, která bude daný endpoint vracet, viz ukázka 6.1.
4. **Přiřazení HTTP metod** - nakonec každému URI přiřadíme typ HTTP metody:
   - **GET**
     - *HTTP GET /apiaries* - vrátí seznam všech včelínů,
     - *HTTP GET /apiaries/{id}/beehives/{hiveId}* - vrátí detail úlu s daným hiveId v konktrétním včelíně.
   - **POST**
     - *HTTP POST /apiaries* - vytvoření nového včelínu,
     - *HTTP POST /apiaries/{id}/beehives* - vytvoření nového úlu v konkrétním včelíně.
   - **PUT**
     - *HTTP PUT /apiaries/id* - změna včelínu s daným id,
     - *HTTP PUT /apiaries/{id}/beehives/hiveId* - změna specifického úlu v konkrétním včelíně.
   - **DELETE**
     - *HTTP DELETE /apiaries/id* - odstranění včelínu s daným id,
     - *HTTP DELETE /apiaries/{id}/beehives/{hiveId}* - odstranění úlu s hiveId z včelínu s daným id.

Zdrojový kód 6.1: Příklad reprezentace dat endpointu

```
{
  "id": "1",
  "name": "apiary1",
  "hive_count": 4,
  "hive_ids": [2, 5, 6, 9],
  "type": "base",
  "queen_id": 2,
  "address": {
    "city": "Vlachovo Brezi",
    "address": "Chlumany",
    "zip": "38422",
    "country": "Czech Republic"
  }
}
```

Jako poslední dodatečný krok při vytváření REST API bude přiřazení návratových status kódů jednotlivým endpointům. Součástí každé HTTP odpovědi bude status kód, který informuje klienta o celkovém výsledku požadavku. Protokol HTTP definuje tyto standardní kódy, které se rozdělují do pěti základních kategorií a k nim některé typické příklady [30]:

- **1xx: Informační** - sdílí informace na úrovní přenosového protokolu:
  - *101 (Switching Protocol)* - změna protokolu,
  - *102 (Processing)* - zpracovávání požadavku od klienta.
- **2xx: Úspěšný** - požadavek klienta byl úspěšně přijat:
  - *200 (OK)* - požadavek byl úspěšný,
  - *201 (Created)* - požadavek byl úspěšný a nový zdroj byl vytvořen.
- **3xx: Přesměrování** - přesměruje klienta, aby provedl další akci pro dokončení požadavku:
  - *300 (Multiple Choices)* - požadavek má více než jednu odpověd, je potřeba vybrat jednu z nich,
  - *301 (Moved Permanently)* - adresa URL požadovaného zdroje byla trvale změněna a je uvedena v hlavičce odpovědi.
- **4xx: Chyba klienta** - kategorie chybových stavových kódů na straně klienta:
  - *400 (Bad Request)* - požadavek nemá správnou syntaxi,

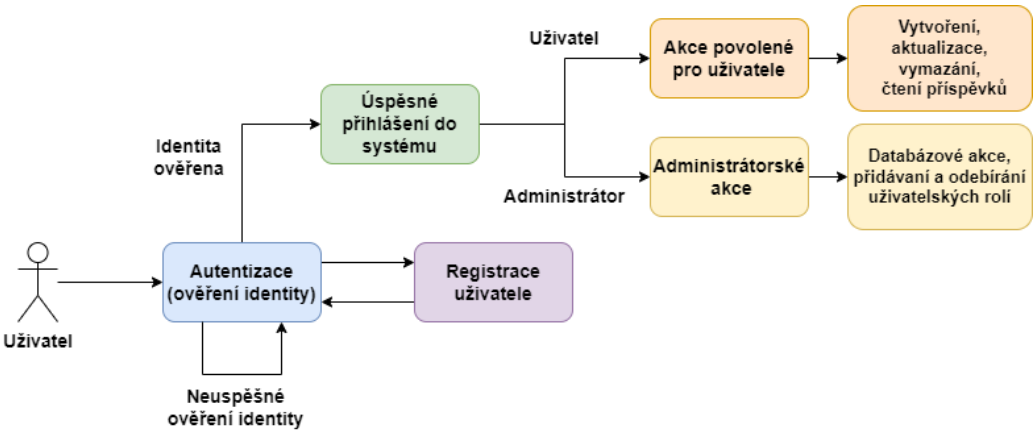
  - *401 (Unauthorized)* - požadavek vyžaduje informace o ověření uživatele.

- **5xx: Chyba serveru** - kategorie chybových stavových kódů na straně serveru:
  - *500 (Internal Server Error)* - na serveru se vyskytl problém, který mu znemožnil zpracovat požadavek,
  - *501 (Not Implemented)* - HTTP metoda není podporována serverem a nelze ji zpracovat.

s

6.2 Ověření identity

Webové aplikace lze zabezpečit různými typy metod autentizací. Pokud nedojde ke správnému ověření identity, bude uživateli zakázán přístup. Stejně tak je tomu pokud je uživatel ověřen, ale není oprávněn, tedy uživateli chybí určitá práva pro danou akci a tento proces se nazývá autorizace. Autentizace a autorizace společně spolupracují a považují se za klíčové pilíře zajištění bezpečnosti webové aplikace. Ve zkratce autentizace je ověření identity uživatele a autorizace ověřuje, zda má tento identifikovaný uživatel povoleno provádět konkrétní úkol nebo ne. Celý proces je popsán na obrázku 6.2.

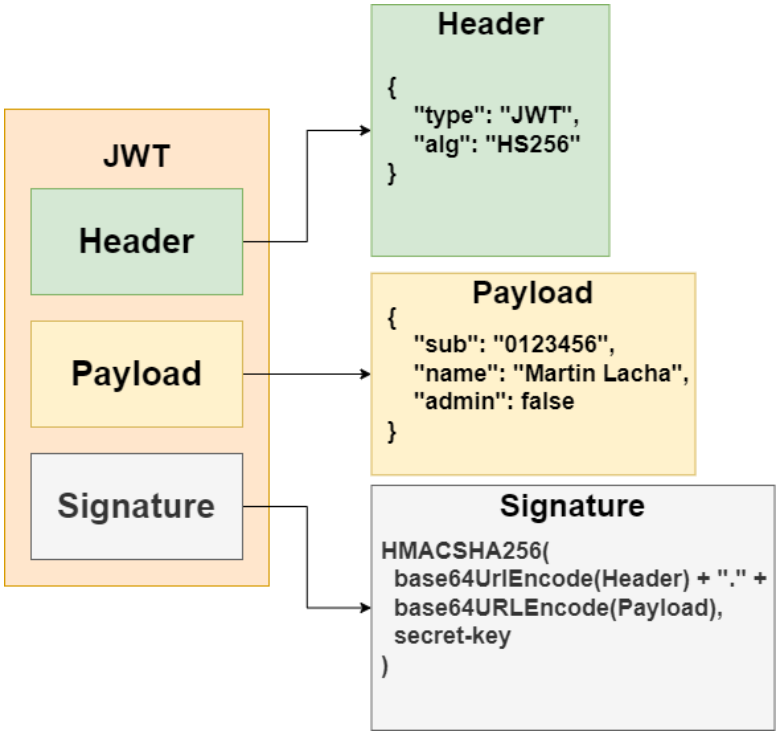


Obrázek 6.2: Koncept autentizace a autorizace

Jako standard pro ověřování a autorizaci uživatelů se často využívá JWT (JSON web token), které je založené na RFC 7519. Tyto tokeny jsou při úspěšném ověření identity předány uživateli, který s dalšími požadavky na server tento token přidá do hlavičky požadavku. Datový obsah tokenu obsahuje typ, šifrovací algoritmus, uživatelská data, datum platnosti a podpis tokenu pro zabezpečení proti úpravě dat. Server zkontroluje token a na základě validity umožní, nebo zakáže uživateli danou akci. Pro případ této práce je nezbytné použít podobné řešení, aby každý uživatel

byl ověřen a mohl přistupovat výhradně k jeho vlastním datům. Bez využití tohoto principu by se každý mohl vydávat za někoho jiného a požadovat data, která mu nepatří. Struktura tokenu se skládá ze tří částí (viz obrázek 6.3) [31]:.

- Header - skládá se z typu tokenu (JWT) a použitého algoritmu (SHA-256)
- Payload - definice entity
  - **Registrované** (registered) - obsahuje předdefinované nepovinné deklarace (sub - subject, exp - doba platnosti).
  - **Veřejné** (public) - obsahuje parametry definované uživatelem (name).
  - **Soukromé** (private) - obsahuje parametry za účelem sdílení informací, které nejsou veřejné ani registrované.
- Signature - podpis tokenu vytvořený z Header, Payload, tajný klíč a algoritmus pro zašifrování (slouží pro ověření, že zpráva nebyla po cestě modifikována a ověření identity uživatele)



Obrázek 6.3: Struktura JSON Web Tokenu

Návrh mobilní aplikace

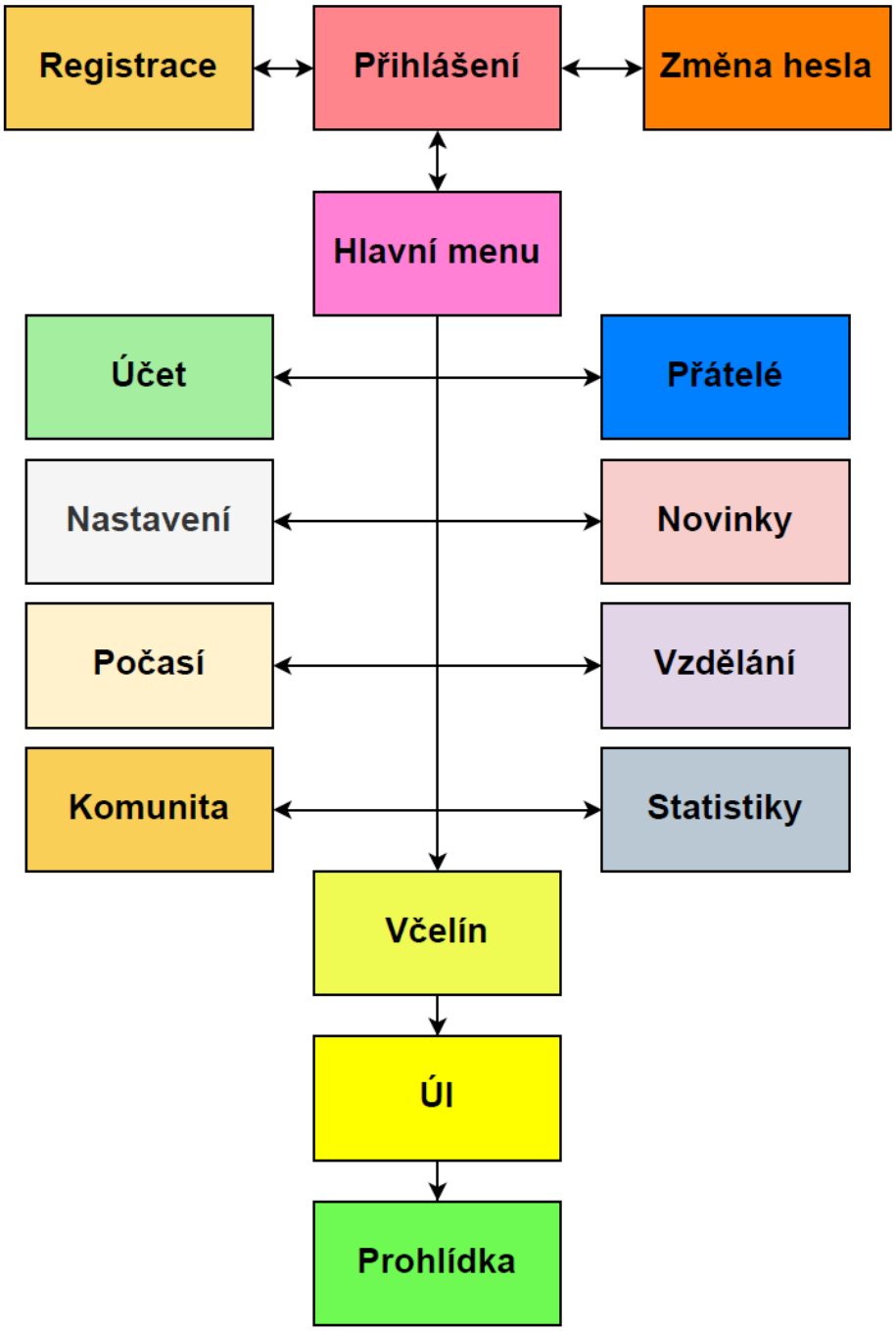
7

Mobilní aplikace bude popsána v další části této diplomové práce a bude navržena pro snadné použití a přehledné zobrazení všech informací. Hlavním cílem bude poskytnout včelaři ucelený nástroj pro správu a monitorování všech svých včelstev. Aplikace bude využívat různé funkce mobilního zařízení, ať už se bude jednat o polohu nebo fotoaparát. Dále bude poskytovat nástroje pro plánování, zaznamenávání a sledování včelařských aktivit.

Klíčové funkce této mobilní aplikace budou zahrnovat:

- správa uživatelského účtu,
- správa včelínů a úlů,
- záznam včelařských operací (stav včelstva, nemoci),
- plánování budoucích aktivit (léčení, krmení),
- diskuzní fórum pro sdílení informací a fotek nebo dotazů,
- získávání aktuálních novinek a trendů ovlivňující zdraví a produkci včel,
- základní znalosti v oblasti včelaření,
- správa přátel (přidávání, odebírání, blokování) a náhled do jejich záznamů,
- zobrazení statistických ukazatelů a grafů s naměřenými hodnotami ze senzorů.

Úvodní obrazovka aplikace bude vyžadovat přihlášení pomocí e-mailové adresy a hesla. Z této obrazovky bude možné registrovat nový účet nebo možnost změnit si heslo k existujícímu účtu. Po úspěšném přihlášení bude uživatel přesunut do hlavního menu, které bude sloužit jako rozcestník do zbytku aplikace. Z hlavního menu bude možné přesunout se do profilu uživatele, nastavení, přehledu včelínů a úlů, počasí, komunity, novinky, edukace a statistik. Obecný návrh jednotlivých obrazovek aplikace je popsán na obrázku 7.1.



Obrázek 7.1: Diagram obrazovek mobilní aplikace

Aby byla aplikace použitelná co největší škálou uživatelů, bude navržena především s důrazem na jednoduchost a uživatelskou přívětivost. Aplikace bude snadno použitelná bez ohledu na technologické dovednosti uživatele. Díky propojení s komunitou se začínající včelaři mohou dotazovat a získávat nezbytné informace od zkušenějších včelařů.

7.1 Cílová platforma

Nejpoužívanější mobilní operační systémy v dnešní době jsou Android a iOS. Podle statistik z roku 2024 operační systém Android využívá přibližně 70% mobilních zařízení a iOS přibližně 28.5% [32]. Díky multiplatformní vlastnosti frameworku Flutter je možné vyvíjet aplikace na operační systém Android i iOS současně z jednoho zdrojového kódu. Primárně se bude vyvíjet a testovat na emulátoru mobilního zařízení a Android zařízení. Pro otestování aplikace na iOS zařízení je potřeba mít navíc nainstalované a nastavené vývojové prostředí Xcode, aktivní Apple Developer Account a používat zařízení s operačním systémem macOS [33]. Pro vývoj mobilní aplikace bude použito vývojové prostředí Visual Studio Code, kde lze jednoduše nastavit cílové zařízení (emulátor, reálné zařízení) pro okamžité nasazení a otestování aplikace bez nutnosti jakékoliv konfigurace.

Návrh monitorování 8

S rychlým rozvojem informačních technologií se otevírají nové možnosti v oblasti monitorování stavu a získávání dat ze včelstev pro včelařského výzkumy a zvýšení samotné produktivity včelstev. Pro moderní včelaření se jako jedna z možností nabízí využití senzorů. Senzory s mobilními zařízeními mohou společně vytvořit velmi silný a užitečný nástroj pro tyto účely. Spojením těchto dvou komponent lze vytvořit monitorovací systém, který by každý včelař mohl mít vždy při sobě a zároveň v jakýkoliv okamžik sledovat aktuální dění v jednotlivých stanovištích. Tímto způsobem lze včelaři ušetřit mnoho času při pravidelných kontrolách, pokud se například úly nachází na špatně dostupných nebo vzdálených místech. Senzory nabízí dobrou přesnost, odolnost, malou spotřebu energie a nízké finanční náklady. Pro senzory, nacházející se mimo úl, je potřeba brát v úvahu meteorologické podmínky (déšť, vlhkost, teplota apod.). Důležitým faktorem je, aby senzory nenarušovaly přirozený ekosystém kolonie. Každé zařízení by mělo splňovat:

- zabírat co nejméně místa a nenarušovat organizaci v úlu,
- ideálně bezhlučné,
- neovlivňovat přirozenou cirkulaci vzduchu,
- neměnit složení vzduchu,
- neuvolňovat/neabsorbovat významné množství tepla,
- nevyzařovat silnější světlo než denní světlo,
- možnost změnit umístění zařízení.

Tato kapitola se zaměří na popis veličin (úlu a okolí), které lze získávat pomocí senzorů a jejich možné využití, návrh blokového zapojení a nakonec možnosti integrace senzorů do vytvářeného systému.

8.1 Senzory

Zavedení senzorové technologie dává uživateli možnost sledovat mnoho fyzikálních a chemických parametrů včelstva a jeho prostředí. Mezi takové aspekty patří především teplota, vlhkost, obsah plynů, video, zvuk vibrace atd. Některé parametry jsou náročné na neustálé monitorování a pro samotného uživatele nemají až takovou hodnotu, jako spíš pro výzkumné účely (zvuk, vibrace) [34]. Některé veličiny budou zároveň analyzovány z hlediska využití v umělé inteligenci jako tomu je například na fakultě strojní, kde například identifikují parazita (roztoče) a určují denní spad [35, 36].

8.1.1 Teplota a vlhkost

Teplotu a vlhkost je dobré sledovat kvůli vlivu prostředí na včelstvo. Důležitá je nejen teplota a vlhkost uvnitř včelího úlu, ale také venkovního prostředí [36]. Mohou mít vliv na zdraví včel, plod a produktivitu [37]. Příliš vysoké nebo nízké teploty mohou ovlivnit chování včel, které mohou být například agresivnější. U plodu může teplota ovlivnit růst a zdraví. Správné hodnoty mohou výrazně ovlivnit úmrtnost a produkci medu (kvalitu, trvanlivost). Příliš vysoká vlhkost může vést ke vzniku plísní a hnilobných procesů, které mohou ohrozit zdraví včelstva.

8.1.2 Vzduch

Vzduch v úlu je komplexní směs různých sloučenin, které uvolňují samotné včely (např. feromony, chemikálie pro odpuzení škůdců a další). Další látky pochází z medu, nektaru, larev, vosku, pylu nebo propolisu a látky z externího prostředí (úl není izolované prostředí) [38]. Hodnoty těchto veličin mohou být zdrojem informací o aktivitě, zdraví a temperamentu včelstva [39].

8.1.3 Váha

Váha se může považovat za ukazatel aktivity včelstva. V období snůšky je ovlivněna sběrem nektaru, pylu a novým potomstvem. Časté události, jako je "vykrádání"[1] ze strany jiného včelstva nebo rojení[2], tento parametr může výrazně ovlivnit a díky notifikacím je možné na tuto událost včelaře upozornit. V zimě zase včely spotřebovávají zásoby, což může postupem času váhu úlu snižovat. Bohužel hmotnost může být zkreslena mnoha faktory jako je vytáčení medu, přidávání a odebírání rámků a nástavků nebo aktuální množství včel v úlu (přes den mnoho včel shání zásoby

[1] Včely navštěvují jiné úly, odkud nenápadně odnáší zásoby do svého úlu.

[2] Jev, kdy stará včelí matka opouští s částí dělnic a zásobami úl, aby našla nový domov (v původním je nahrazena novou matkou).

mimo úl). Dalším faktorem můžou být povětrnostní podmínky, jelikož včely absorbují vodní páru z vlhkého vzduchu během mlhy nebo deště a naopak vysychají na slunci [38].

8.1.4 Zvuk a vibrace

Zvuková analýza se využívá k určení stavu včel bez nutnoti manuální kontroly. Včely společně komunikují pomocí vibrací a zvukových signálů, které generují několika způsoby:

- hrubé pohyby těla,
- vysokofrekvenční svalové kontrakce bez pohybu křídel,
- přitisknutí hrudníku k podložce nebo jiné včele.

Tyto signály úzce souvisejí také s konkrétními událostmi jako je rojení [36]. Zvuk se zaznamenává pomocí mikrofonů, umístěných uvnitř nebo vně úlu. Alternativní možností je použití akcelerometru pro měření vibrací. Byla prokázána korelace mezi událostmi (např. rojení) a zvukem generovaným včelami (změna ve frekvenci a amplitudách po Fourierově transformaci pro zvuk při rojení [40]). Další výhodou snímání zvuku je možnost zjistit přítomnost matky. Studie ukazují, že zvuková analýza poskytuje mnoho užitečných dat, které je možné využít k odvození úplné analýzy zdravotního stavu včelstva [41]. V nedávné době bylo také zavedeno použití strojového učení pro klasifikaci nahraných zvukových vzorků pro identifikaci stavu včelstev [37].

8.1.5 Obraz

Pro sledování videa v rámci včelího úlu se využívají digitální fotoaparáty a videokamery. Díky obrazu je možné vidět okamžitou situaci ve včelím úlu. Obecně pozorování včel v reálném čase nám umožňuje zjistit jak funguje jejich společenství jako celek (kladení vajíček, chov plodu, skladování pylu) nebo může být použita pro edukaci a zvýšení povědomí o včelách. Z hlediska výzkumu může být obraz vhodným vstupem pro trénování umělé inteligence, například pro rozpoznávání vzorů (chování apod.). Je zřejmé že pozorování obrazu nenahradí podrobnou kontrolu, ale může být rozhodující pro vydání včasného varování [38].

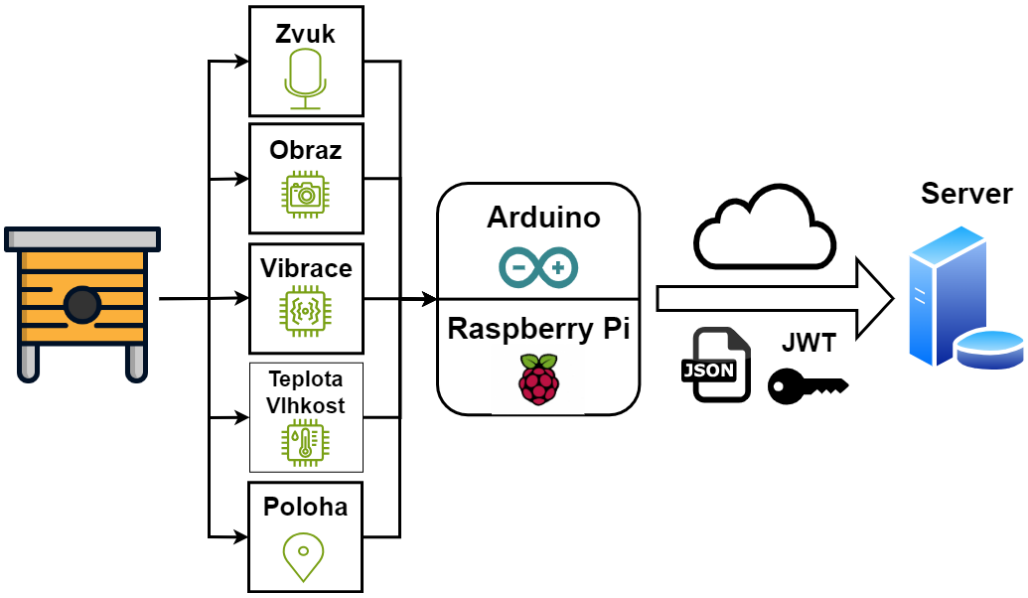
8.1.6 Poloha

Poloha slouží jako kontrola pro případ odcizení celého úlu. Avšak tato událost není v praxi tak častá, dává to uživateli alespoň určitý pocit jistoty. V případě odcizení se mohou škody škody vyšplhat až do několika desítek tisíc korun.

8.2 Návrh integrace senzorů

Cílen toho návrhu je analyzovat a navrhnout integraci senzorů se zbytkem práce (mobilní klient a server). Architektura se bude skládat z těchto částí: senzory připojené ke včelímu úlu, mikropočítač pro sběr, zpracování a odesílání dat a serveru pro zpracování a ukládání dat, viz obrázek 8.1.

Hodnoty veličin z úlu se budou získávat v pravidelných intervalech a následně budou odesílány přes internet na komunikační server, který bude data přijímat a následně je ukládat do databáze. Druhou možností je posílat data nejdříve na mobilní zařízení, ale toto řešení je nadbytečné, jelikož se data nijak v mobilním zařízení nezpracovávají a jen se přepošlou dále na komunikační server. Další nevýhodou je dostupnost mobilního zařízení, kdy nemusí mít signál nebo je vypnuté. Naopak u komunikačního serveru je důležité, aby byl stále dostupný a fuknění, proto bude tato možnost preferována pro realizaci. Dále budou přenesená data v mobilním klientovi zobrazena ve formě záznamů hodnot nebo grafů. Pro pravidelnou komunikaci se může využít komunikačního protokolu WebSocket, který podporuje obousměrnou komunikaci v reálném čase pomocí TCP připojení.



Obrázek 8.1: Diagram propojení systému se senzory

Komunikační server 9

Komunikační server byl vytvořen jako systém pro zprostředkovávání komunikace mezi klienty a ukládání dat. Mezi jeho hlavními úkoly jsou především poskytování dat, správa jednotlivých uživatelů (autentizace, autorizace, registrace apod.), validace, ukládání a zpracování dat do databáze. Pro server byly využity vybrané technologie (viz obrázek 9.1), které jsou v současné době pro tvorbu obdobných serverů často používané.

Obrázek 9.1: Použité technologie pro komunikační server

9.1 Sestavení

Nasazení aplikace je v případě manuální instalace a konfigurace velmi časově náročné. Z toho důvodu byl použit nástroj Docker, který tuto práci maximálně zjednoduší. Serverová část se skládá ze tří služeb: PostgreSQL databáze, serverová aplikace

a nástroj Flyway pro migraci databází. Soubor *compose.yaml* popisuje definici jednotlivých služeb. Dále byla v souboru vytvořena síť, ve které budou jednotlivé služby mít přiřazenou jednu statickou IP adresu. Mezi jednotlivými službami existují závislosti pro správné fungování systému (aplikace nemůže fungovat bez spuštěné databáze).

Databáze

Databáze bude spuštěna v kontejneru s názvem 'beecommunity_postgres_1' s obrazem poslední verze PostgreSQL. Při nasazení nové verze celého systému se nebude znovu nasazovat, aby se neztratila data. V síti bude mít přiřazenou statickou IP adresu **10.1.0.2** a pro prostředí budou nastaveny hodnoty pro název databáze, jméno a heslo pro přístup do databáze. Nakonec bude namapován port z hostitelského počítače (5432) do portu v kontejneru (5432). Konfigurace kontejneru je popsána, viz ukázka 9.1.

Zdrojový kód 9.1: Konfigurace kontejneru databáze

```
postgres:
    container_name: 'beecommunity_postgres_1'
    image: 'postgres:latest'
    restart: "no"
    networks:
        server-network:
            ipv4_address: 10.1.0.2
    environment:
        - 'POSTGRES_DB=beecommunity_db'
        - 'POSTGRES_PASSWORD=beecommunity_password'
        - 'POSTGRES_USER=beecommunity_username'
        - 'TZ=Europe/Prague'
ports:
  - '5432:5432'
```

Flyway

Nástroj bude spuštěn po vytvoření kontejneru databáze. Konfigurace je popsána, viz ukázka 9.2. Obsahuje URL adresu pro migraci databáze. V konfiguraci je definovaná složka s migracemi databáze. V síti bude mít přiřazenou statickou IP adresu **10.1.0.3**. Migrace vytvoří základní strukturu databáze a naplnění počátečními daty.

Zdrojový kód 9.2: Konfigurace kontejneru nástroje Flyway

```
flyway:
    image: flyway/flyway
    command:
        -url=jdbc:postgresql://10.1.0.2:5432/beecommunity_db -
            schemas=public -user=beecommunity_username -password
            =beecommunity_password migrate
    volumes:
        - ./src/main/resources/db/migration:/flyway/sql
    depends_on:
        - postgres
    networks:
        server-network:
            ipv4_address: 10.1.0.3
```

Aplikace

Jako obraz pro kontejner se sestaví projekt v aktuálním adresáři. Popis jak má být obraz vytvořen je popsán v souboru **Dockerfile**. Sestavení je rozděleno na více fází (multi-stage build), které se využívá pro vytvoření efektivních a optimalizovaných obrazů, např. odeberou se zbytečné soubory vytvořené při sestavení aplikace. Jednotlivé fáze jsou ukázány v ukázce 9.3. V první fázi se sestaví aplikace a spustí se jednotkové testy pro ověření funkčnosti. Druhá fáze vytvoří už samotný obraz aplikace bez nadbytečných souborů. Kontejner bude mít přidělenou statickou IP adresu **10.1.0.4** a port z hostitelského počítače (8080) bude namapován do portu v kontejneru (8080). V ukázce 9.4 je popsána konfigurace kontejneru.

Zdrojový kód 9.3: Vícefázové sestavení

```
# Stage 1
# Build the application
FROM maven:3.8.4-openjdk-17-slim AS build
WORKDIR /app
COPY . /app
RUN mvn -B clean package

# Stage 2
# Create the runtime image
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Zdrojový kód 9.4: Konfigurace kontejneru aplikace

```
app:
    build: .
    networks:
        server-network:
            ipv4_address: 10.1.0.4
    ports:
        - "8080:8080"
    depends_on:
        - postgres
```

Celý projekt z **Github** repozitáře se sestaví a spustí na serveru pomocí následujícího příkazu *docker-compose up -d*. Následně bude aplikace přijímat a obsluhovat požadavky ze strany klientů.

9.2 Struktura kódu

Zdrojový kód komunikačního serveru je strukturován do logických celků v samostatných balíčcích, aby se v projektu co nejlépe orientovalo. Kód je rozdělen do následujících částí:

- **config** - třídy pro konfiguraci,
- **controllers.v1.api** - kontrolery pro příjem požadavků,
- **enums** - třídy s výčtovými typy,
- **filters** - filtery pro příchozí požadavky,
- **handlers** - obsluhy při výskytu konkrétní události,
- **jpa** - definice entit, dto a JPA rozhraní tabulek,
- **schedule** - třídy s naplánovanými asynchroními událostmi,
- **services** - rozhraní a implementace obsluhy a zpracování příchozích požadavků,
- **utils** - pomocné třídy.

9.3 Konfigurace

Konfigurace aplikace se dá rozdělit do dvou částí, a to konfigurace objektů **bean** a vlastností (application.properties).

Konfigurační soubor *application.properties* obsahuje definice konfiguračních parametrů použitých v aplikaci Spring frameworku nebo nástrojů, které framework využívá (PostgreSQL, Flyway a další). Pro získání hodnot byla vytvořena třída **PropertiesConfiguration**, která obsahuje metody pro získání hodnot z konfigurace pro použití v aplikaci. Konfigurace je rozdělena do následují částí:

- **Application** - vlastní konfigurace použita v aplikaci:
  - **token-expiration-seconds** - doba platnosti přihlašovacího tokenu v sekundách,
  - **token-month-expiration** - doba platnosti tokenu na měsíc,
  - **secret-key** - klíč pro šifrování tokenů,
  - **enable-token-expiration** - povolení/zakázaní expirace tokenů,
  - **database-name** - jméno databáze.
- **Spring** - nastavení samotného frameworku,
- **Databáze** - konfigurace databáze (URL, jméno, heslo apod.),
- **Flyway** - konfigurace nástroje pro migraci databáze,
- **E-mail** - konfigurace e-mailového serveru.

Konfigurace objektů Bean je už definována v samotné třídě projektu, které se budou využívat napříč v celé aplikaci. Třída **WebConfig** bude obsahovat definice následujících objektů:

- **BCryptPasswordEncoder** - slouží pro kódování hesel (hesla nesmí být uložena v textové podobě z důvodu bezpečnosti)
- **ModelMapper** - slouží k mapování objektů mezi různými datovými typy

9.4 Zabezpečení

Spring Boot framework obsahuje **Spring Security**, která poskytuje zabezpečení (funkcionality pro autentizaci, autorizace a další). Při příchodu požadavku na server, musí požadavek projít filtry v metodě **securityFilterChain**. Pro každý příchozí požadavek se nejprve zjistí, na který endpoint chce přistoupit. V aplikaci jsou některé endpointy přístupné bez nutnosti autentizace a zbytek ji vyžaduje. Všechny endpointy začínají "/api/v1". Je to z důvodu možnosti budoucí aktualizace aplikačního rozhraní na novější verzi. V aplikaci jsou následující endpointy:

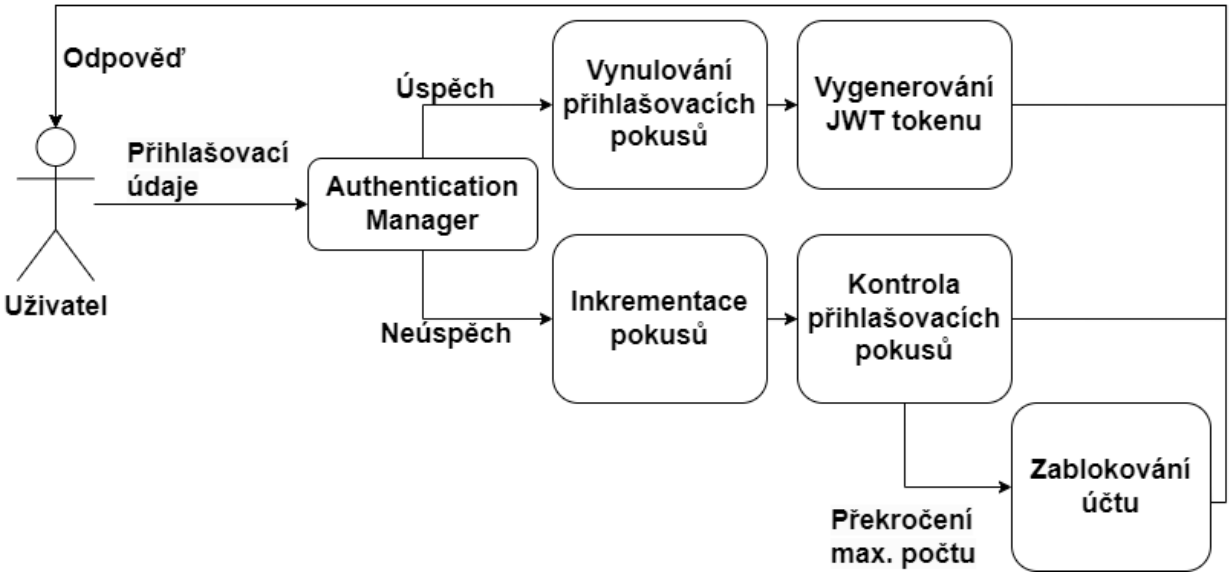
- Nevyžadující autentizaci:
  - */api-docs* - popis aplikačního rozhraní,
  - */api/v1/server/test-connection* - test spojení,
  - */api/v1/user/sign-up* - registrace nového uživatele,
  - */api/v1/user/update-password* - aktualizace hesla účtu,
  - */api/v1/user/reset-password* - změna hesla účtu,
  - */api/v1/hive/sensors* - data ze senzorů.
- Vyžadující autentizaci:
  - */api/v1/apiary* - operace pro včelín,
  - */api/v1/community-post* - operace pro komunitní příspěvky,
  - */api/v1/event* - operace pro události,
  - */api/v1/friends* - operace s ostatními uživateli,
  - */api/v1/hive* - operace s úly,
  - */api/v1/inspection* - operace s kontrolami úlu,
  - */api/v1/news* - operace s novinkami,
  - */api/v1/queen* - operace se včelími královnami,
  - */api/v1/stats* - operace se statistikami,
  - */api/v1/user* - operace s uživatelským učtem.

Dále je nastavená bezstavová politika (není uložen stav serveru) a každý požadavek je nezávislý na předchozích. Stav není potřeba kvůli použití tokenů pro ověření uživatele. Sníží se tím zatížení serveru a komunikační server je možné lépe škálovat, protože se stav nemusí sdílet s ostatními instancemi serveru.

9.4.1 Autentizace

Proces autentizace je popsán na obrázku 9.2. Pro ověření identity je potřeba se přihlásit pomocí uživatelského jména (e-mailové adresy) a hesla na endpointu "/api/v1/user/login". Pro ověření identity **AuthenticationManager** porovná otisk hesla uloženého v databázi s otiskem hesla použitého při přihlášení. Pokud proběhla autentizace neúspěšně bude odchycena ve třídě **AuthenticationUserFailureHandler**, která zvýší počet pokusů uživatele o přihlášení a informuje uživatele, že nebyl autentizován (401 unauthorized). Pokud počet přesáhne tři neúspěšné pokusy, bude účet zablokován (návratový kód 420) a bude vyžadovat resetování hesla.

Úspěšné přihlášení bude odchyceno třídou **AuthenticationUserSuccessHandler**, kde budou pokusy o přihlášení vynulovány (pokud účet není zablokován) a



Obrázek 9.2: Proces autentizace

vytvoří se nový JWT token, viz ukázka kódu 9.5. Jako subjekt bude nastaven uživatelský e-mail. Token bude přiložen do hlavičky v odpovědi uživateli. Token je vygenerován za pomoci tajného klíče šifrovacím algoritmem SHA-256. Ukázka vygenerovaného tokenu je na obrázku 9.3 a dešifrovaný obsah je znázorněn na obrázku 9.4. Tento token bude dále uživatel posílat v každém požadavku, aby předešel neustále autentizaci.

Zdrojový kód 9.5: Vygenerování nového JWT tokenu

```
Jwts
    .builder()
    .setClaims(extraClaims)
    .setSubject(userDetails.getUsername())
    .setIssuedAt(new Date(System.currentTimeMillis()))
    .setExpiration(new Date(expirationTime))
    .signWith(getSignInKey(), SignatureAlgorithm.HS256)
    .compact();
}
```

```
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJtYXJ0aW
4ubGFjaGEyQHNlem5hbS5jeiIsImlhdCI6MTcwN
zgxMzM4MCwiZXhwIjoxNzEwNDA1MzgwfQ.CmOre
aC44PzEXtvfshro7dAt3sR3Wy1Ngqy_7q94B8M
```

Obrázek 9.3: Ukázka JWT tokenu

Obrázek 9.4: Ukázka datového obsahu tokenu

9.4.2 Ověření identity tokenem

Při příchodu požadavku je potřeba zkontrolovat identitu uživatele. Tato činnost je kontrolována pomocí filteru **JwtAuthenticationFilter**. Filtr využívá třídu **JwtServiceImpl** pro vytváření, kontrolu validity a získávání údajů z autorizačního tokenu. Z hlavičky *Authorization* získá token (ukázka 9.6). Pokud požadavek neobsahuje hlavičku, předpokládá se, že uživatel přistupuje na endpoint bez nutnosti ověření identity a tento požadavek je poslán do **securityFilterChain**.

Zdrojový kód 9.6: Extrahování hlavičky s autorizací

```
1 String header=request.getHeader(HttpHeaders.AUTHORIZATION);
```

Pomocí tajného klíče se z tokenu se získá uživatelský e-mail viz ukázka kódu 9.7, pro který se budou porovnávat údaje z databáze a tokenu viz ukázka kódu 9.8. Pokud se údaje shodují je pro uživatele vytvořen objekt **UsernamePasswordAuthenticationToken**, který reprezentuje autentizovaného uživatele. Pokud při validaci tokenu nastala chyba (např. nevalidní nebo vypršela expirační doba) je požadavek poslán do **securityFilterChain**.

Zdrojový kód 9.7: Extrahování e-mailu z tokenu

```
public String extractUsernameFromToken(String token) {
    return extractClaim(token, Claims::getSubject);
}

private <T> T extractClaim(String token, Function<Claims, T>
    claimsResolver) {
    final Claims claims = extractAllClaims(token);
    return claimsResolver.apply(claims);
}

private Claims extractAllClaims(String token) {
    return Jwts
        .parserBuilder()
        .setSigningKey(getSignInKey())
        .build()
        .parseClaimsJws(token)
        .getBody();
}
```

Zdrojový kód 9.8: Validace tokenu

```
// Check validity of the token
if (jwtService.isTokenValid(jwt, userDetails)) {
    UsernamePasswordAuthenticationToken authToken =
        new UsernamePasswordAuthenticationToken(
            userDetails,
            null,
            userDetails.getAuthorities());
    authToken.setDetails(
            new WebAuthenticationDetailsSource()
                .buildDetails(request));
    SecurityContextHolder
        .getContext()
        .setAuthentication(authToken);
}
```

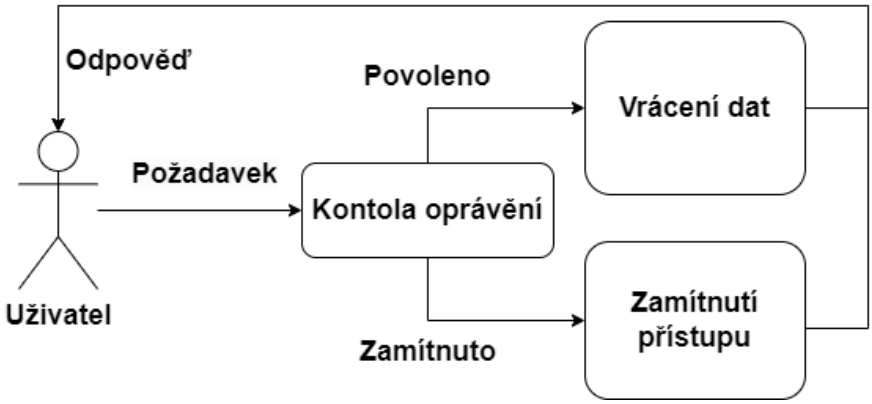
9.4.3 Autorizace

V aplikaci existují celkem tři typy uživatelských rolí. Základní uživatelskou rolí je klasický uživatel, kterou bude mít každý uživatel v systému. Další uživatelskou rolí je administrátor. Oproti uživateli může přiřazovat a odebírat ostatním uživatelům administrátorskou roli nebo měnit uživateli přihlašovací e-mail. Také může přidávat novinky. Jako poslední rolí je super administrátor. Tato role je stejná jako normální administrátor pouze s tím rozdílem, že tato role je jedinečná, nedá se nikomu dalšímu přiřadit a není možné ji odebrat. V systému bude pouze jeden účet, který bude mít tuto roli (správce systému). Je vytvořena z důvodu, pokud by se v aplikaci objevil

uživatel, který by chtěl záměrně škodit, např. odebrat všem ostatním uživatelům administrátorskou roli. Následující seznam popisuje operace jednotlivé uživatelské role:

- **Uživatel**
  - je klasický uživatel.
- **Admin**
  - přidávání, odebírání a aktualizace novinek,
  - přidávání/odebírání administrátorkých práv,
  - změna e-mailu uživatele.
- **Super Admin**
  - stejné operace jako **Admin**
  - jedinečná role, která nemůže být udělena nebo odebrána

Při pokusu o přístup na endpoint, ke kterému uživatel nemá dostatečná práva (uživatelskou roli), je tento požadavek odchycen třídou **ApiAccessDeniedHandler**, která uživatele informuje o nedostatečném oprávnění (403 forbidden). Autorizace je znázorněna na obrázku 9.5.



Obrázek 9.5: Proces autorizace

Realizace autorizace podle rolí je zobrazena, viz ukázka 9.9. Pro přístup na endpoint */api/v1/user/admin* je potřeba, aby uživatel měl **ADMIN**. Ve druhém případě pro endpoint */api/v1/event* musí účet mít roli **USER**.

Zdrojový kód 9.9: Ukázka autorizace pro endpointy

```
1 requestMatchers("/api/v1/user/admin")
2     .hasAuthority(ADMIN.name())
3 requestMatchers("/api/v1/event")
4     .hasAuthority(USER.name())
```

9.5 Datový model a perzistence

Datový model je uveden v příloze D. Při nasazení aplikace (po vytvoření kontejneru databáze) se spustí migrace, která vytvoří databázové tabulky a relace mezi nimi podle definice popsané v souboru "init_db.sql". Po inicializaci se spustí ještě soubor "insert_test_data.sql", který do databáze vloží testovací data.

9.5.1 Databázové operace

Pro práci se záznamy v databázi se využívá JPA rozhraní. Anotace **@Repository** označuje třídu jako Spring Repository, což umožňuje frameworku Spring Boot provádět automatické skenování a vytváření instance tohoto rozhraní. Třídě **JpaRepository** se předá parametr pro entitu, se kterou bude rozhraní pracovat a datový typ primárního klíče, viz ukázka 9.10. Jedná se o rozšíření, které obsahuje předdefinované CRUD operace, rozhraní pro stránkování a řazení. V ukázce jsou definované operace:

- **findByOwnerIdOrderById** - vrátí entitu včelínu podle id uživatele a id včelínu,
- **countById** - vrátí počet včelínů,
- **countApiariesGroupByOwner** - manuálně definovaný **SELECT** pro získání seznamu uživatelů seřazeného dle počtu přiřazených včelínů danému uživateli.

Zdrojový kód 9.10: Ukázka rozhraní JPA

```
@Repository
public interface ApiaryRepository
    extends JpaRepository<ApiaryEntity, Long> {

    List<ApiaryEntity> findByOwnerIdOrderById(Long userId);

    int countByOwnerId(Long userId);

    @Query("SELECT COUNT(r), r.owner.email
        FROM ApiaryEntity r WHERE
        r.owner.userInfo IS NOT NULL
        GROUP BY r.owner
        ORDER BY COUNT(r) DESC")
    List<Object[]> countApiariesGroupByOwner();
}
```

9.5.2 Relační mapování

Pro namapování tabulky z databáze do objektu využívá framework Spring Boot anotace, viz ukázka 9.11. Pro mapování se používají následující anotace:

- **Entity** - označuje třídu jako entitu, která bude mapována na tabulku v databázi.
- **Table** - je tabulka v databázi.
- **Id** - označuje primární klíč.
- **GeneratedValue** - je strategie pro generování hodnot primárního klíče.
- **Column** - je sloupec v databázové tabulce.
- **JoinColumn** - je sloupec v tabulce namapovaný podle cizího klíče v závorce.
- **OneToOne** - označuje kardinalitu vztahu 1:1.
- **OneToMany** - označuje kardinalitu vztahu 1:N.
- **ManyToOne** - označuje kardinalitu vztahu N:1.

Zdrojový kód 9.11: Objektově relační mapování

```
@Entity
@Table(name = "APIARY")
public class ApiaryEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private UserEntity owner;

    @Column(name = "name")
    private String name;

    @Enumerated(EnumType.STRING)
    @Column(name = "environment")
    private ApiaryEnums.EEnvironment environment;
}
```

9.5.3 Záloha databáze

Záloha databáze je při spuštění aplikace naplánována jako asynchronní operace pomocí anotace **@Scheduled**, kde se nastaví CRON[1] pro spuštění zálohy (nastaveno na každý den ve 23 hodin). Samotná záloha se získá pomocí PostgreSQL nástroje *pg_dump* (viz ukázka 9.12), který provede konzistentní zálohu, aniž by blokoval ostatní uživatele.

Zdrojový kód 9.12: Ukázka zálohy databáze

```
// Run backup command on docker database container
var successExitCode = runSingleCommand(
        "docker",
        "exec",
        databaseContainerName,
        "sh",
        "-c",
        String.format("pg_dump -U %s %s > %s",
                propertiesConfiguration.getDatabaseUser(),
                propertiesConfiguration.getDatabaseName(),
                backupName));
```

9.6 Obsluha požadavků

Pokud požadavek projde bezpečností kontrolou, začne se zpracovávat v příslušném kontroleru, viz ukázka 9.13. Anotace **@RestController** označuje třídu jako REST kontroler, který bude zpracovávat HTTP požadavky a vracet odpověď (návratový kód, data). Základní endpoint je uveden v **@RequestMapping** a popřípadě je specifikována dodatečná cesta daného endpointu. Jednotlivé metody mají anotace, které určují o jaký typ metody se jedná (GetMapping - metoda GET atd.). Pokud přijde požadavek na endpoint, který neimplementuje daný typ metody je chyba odchycena třídou **RequestMethodNotSupportedHandler**, která uživateli vrátí chybový kód.

Zdrojový kód 9.13: Ukázka kontroleru událostí

```
@RestController
@RequestMapping("/api/v1/event")
@AllArgsConstructor
public class EventController {
    private final IEventService eventService;

    @GetMapping
    ResponseEntity<LinkedHashMap<String, List<EventDto>>>
   getEvents() {
        return eventService.getEvents();
    }
```

[1]Softwarový daemon, který v operačním systému spouští v určitý čas proces.

```
12      @PostMapping
13      ResponseEntity<Void> createEvent(@RequestBody @Valid
        EventDto eventDto) {
14          return eventService.createEvent(eventDto);
15      }
16  }
```

IEventService představuje rozhraní služby, která slouží jako prostředník mezi kontrolerem a implementační částí (zpracuje samotný požadavek). Každý kontroler má jedno rozhraní, které definuje operace pro zpracování požadavků pro jednotlivé endpointy. Ukázka implementace služby je zobrazena v ukázce 9.14. Každá metoda vrací návratový kód podle výsledku dané operace a popřípadě data v těle odpovědi.

Zdrojový kód 9.14: Ukázka implementace služby

```
1  @Override
2  public ResponseEntity<LinkedHashMap<String, List<EventDto>>>
       getEvents() {
3      var user = UserUtils.getUserFromSecurityContext();
4      var entitiesList =
5          eventRepository.findByOwnerIdOrderById(user.getId());
6      return ResponseEntity
7          .status(HttpStatus.OK)
8          .body(modelMapper.convertEventList(entitiesList));
9  }
```

9.7 Výměna dat

Při komunikaci mezi komunikačním serverem a klientem si budou obě strany vyměňovat data. Použití entit by nebylo vhodné, jelikož mohou obsahovat nadbytečná data nebo je potřeba data upravit.

9.7.1 Datový objekt

Pro výměnu dat mezi dvěma aplikacemi, se používají objekty DTO, viz ukázka 9.15. Objekt reprezentuje nového uživatele a bude obsahovat e-mail uživatele a heslo. Jednotlivé atributy objektu jsou validovány při příchodu požadavku. Pokud některý atribut nesplňuje validaci, je odchycen třídou **RequestValidationHandler**. Pokud některý atribut naopak chybí, je chyba odchycena třídou **RequestBodyMissingHandler**. Příklady anotací pro validaci atributů objektu:

- **E-mail** - hodnota atributu musí být platná e-mailová adresa.
- **NotEmpty** - hodnota atributu musí obsahovat alespoň jeden neprázdný znak (může obsahovat bílé znaky).

- **NotBlank** - hodnota atributu musí obsahovat alespoň jeden znak, který není bílý znak (mezera, tabulátor).
- **Size** - je požadavek na délku hodnoty atributu.

Zdrojový kód 9.15: Ukázka třídy DTO nového uživatele

```
public class NewUserDto {
    @NotEmpty(message = "Email can't be empty.")
    @Email(message = "Not valid email address.")
    private String email;
    @NotEmpty(message = "Password can't be empty.")
    @NotBlank(message = "Password can't be blank.")
    @Size(min = 10, message = "Length of password has to be
   at least 10 characters.")
    private String password;
}
```

9.7.2 Mapování objektů

Pro vyhnutí se manuálnímu mapování entit a objektů DTO, je použita třída **ModelMapper**. Pro mapování se specifikuje, který objekt bude sloužit jako zdroj dat a do jaké třídy se tento objekt bude mapovat. Pokud některou hodnotu nelze namapovat kvůli datovému typu (text na datum), je dodatečně manuálně nastavena. Ukázka obou případů je zobrazena v 9.16.

Zdrojový kód 9.16: Ukázka mapování entity a DTO

```
public ApiaryDto convertApiaryEntity(ApiaryEntity entity) {
    return modelMapper.map(entity, ApiaryDto.class);
}

public SensorsDataEntity convertSensorsDataDto(
    SensorDataDto data) {
    var entity =
        modelMapper.map(data, SensorsDataEntity.class);
    entity.setTime(LocalDateTime.now());
    return entity;
}
```

9.8 Posílání e-mailů

Pokud je uživateli zablokován účet, kvůli překročení maximálního počtu pokusů o přihlášení, může si jej odblokovat pomocí změny hesla. Na e-mail mu bude zaslán kód, který v mobilní aplikaci použije pro změnu hesla. Pro posílání e-mailů z komunikačního serveru, byl v konfiguraci nastaven Gmail účet, který bude posílat

e-maily uživatelům. Konfigurace SMTP je ve Spring Boot popsána v ukázce 9.17 s následujícími položkami:

- **host** - adresa SMTP serveru (Gmail server),
- **port** - port, kde server přijímá spojení,
- **username** - uživatelské jméno Gmail účtu,
- **password** - heslo pro autentizaci na SMTP serveru,
- **properties.mail.smtp.auth** - povolení autentizace na SMTP serveru,
- **properties.mail.smtp.starttls.enable** - povolení použití TLS při komunikaci s SMTP serverem.

Zdrojový kód 9.17: Konfigurace SMTP ve Spring Boot

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=martin.lacha62@gmail.com
spring.mail.password=<password>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

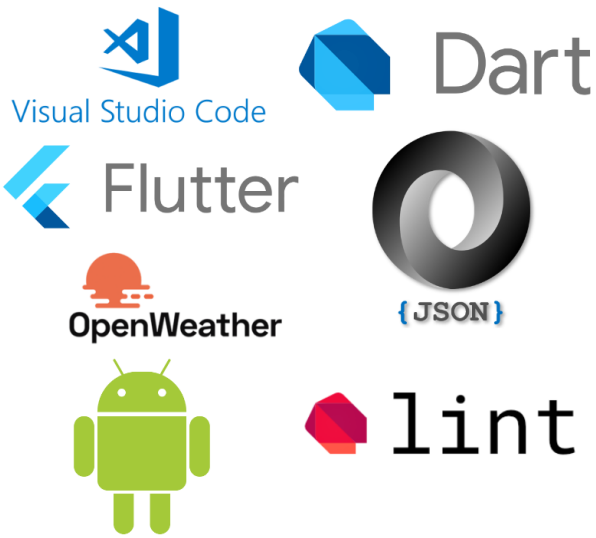
Při požadavku na změnu hesla je vytvořen náhodně vygenerovaný kód, který bude uživateli poslán na e-mailovou adresu. Kód je uložen společně s uživatelským e-mailem na serveru. Následně se uživateli odešle zpráva s vygenerovaným kódem, viz ukázka 9.18.

Zdrojový kód 9.18: Odeslání e-mailu s vygenerovaným kódem

```
SimpleMailMessage mailMessage = new SimpleMailMessage();
mailMessage.setTo(email);
mailMessage.setSubject("BeeCommunity: Reset password code");
mailMessage.setText("Confirm code: " + code);

try {
    emailSender.send(mailMessage);
    log.info("Reset password email sent to {}", email);
    return ResponseEntity.status(HttpStatus.OK).build();
} catch (MailException e) {
    log.warn("Failed to send reset password email to {}",
        email, e);
    return ResponseEntity
        .status(HttpStatus.BAD_REQUEST)
        .build();
}
```

Mobilní aplikace 10

Mobilní zařízení v této práci bude prostředek pro vytváření, úpravu a především zobrazování dat získaných z komunikačního serveru. Pro implementaci byly vybrány technologie (obrázek 10.1), které jsou v současné době velmi populární a používané pro tvorbu mobilních aplikací. V aplikaci byly použity obrázky ze stránek *https://lottiefiles.com* (Lottie Simple License) a *https://www.flaticon.com* (Flaticon License, Merchandising License).



Obrázek 10.1: Použité technologie pro mobilní aplikaci

10.1 Obrazovky

Součástí Flutteru je knihovna **material**, která obsahuje sadu widgetů, nástrojů pro vytváření aplikací. Nabízí základní widgety[1] jako jsou tlačítka, textové pole atd.

[1]Objekt nebo konstrukce reprezentující vizuální část aplikace a její stav.

10.1.1 Stav

Každá obrazovka bude vycházet ze **StatefulWidget**, viz ukázka 10.1. Každá obrazovka bude uchovávat svůj stav, který bude moci dynamicky měnit na základě interakce s uživatelem.

Zdrojový kód 10.1: Deklarace třídy pro vyhledání přátel

```
class FindFriendsPage extends StatefulWidget {
  const FindFriendsPage({Key? key}) : super(key: key);

  @override
  _FriendsFindState createState() => _FriendsFindState();
}
```

Realizace stavu obrazovky je znázorněn v ukázce 10.2. Pro počáteční inicializaci stavu slouží metoda **initState**. Slouží například pro inicializaci hodnot při načtení obrazovky. Metoda **dispose** je naopak volána při zrušení instance třídy a uvolnění zdrojů a metoda **build** slouží pro sestavení vzhledu obrazovky.

Zdrojový kód 10.2: Ukázka stavu obrazovky

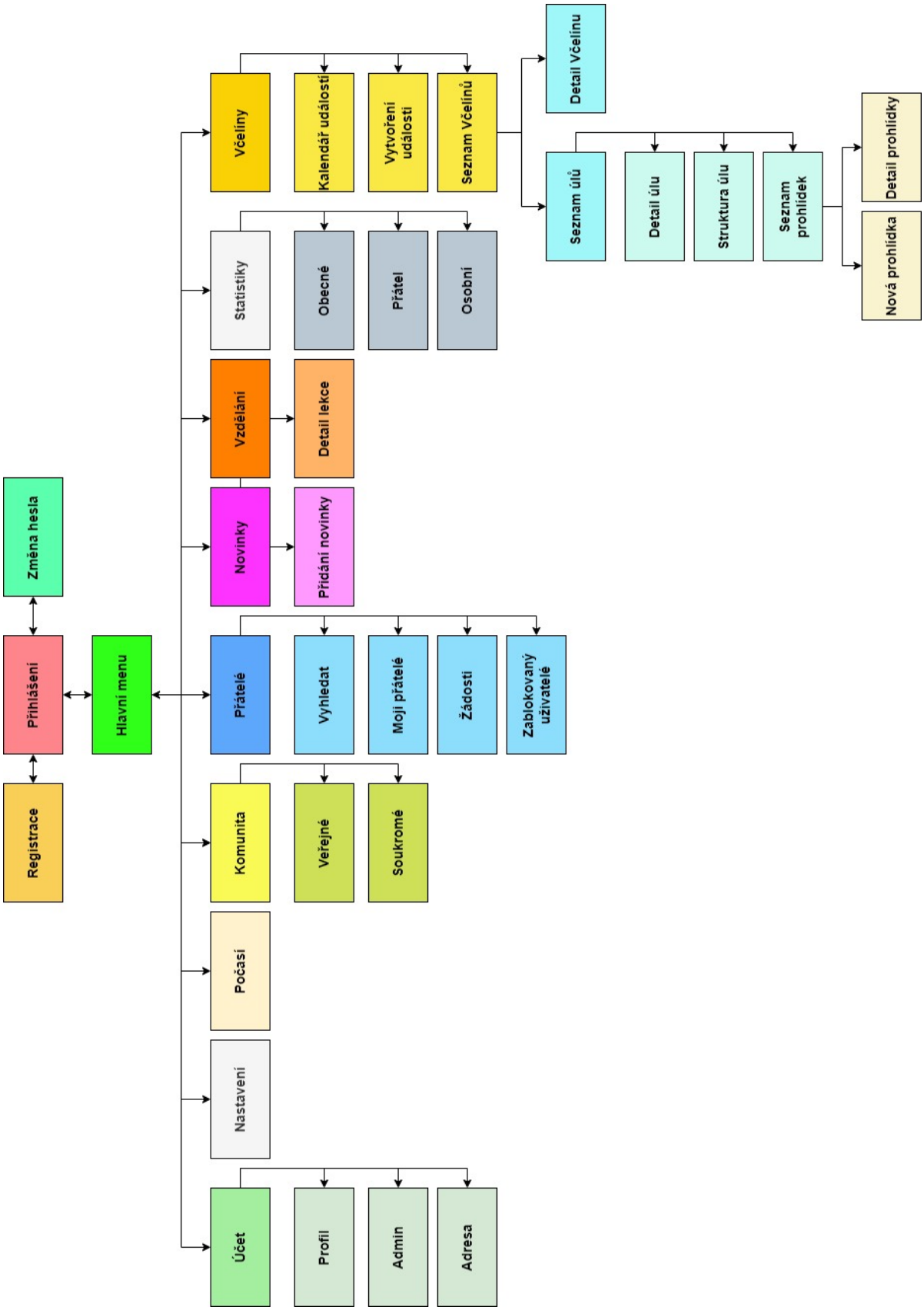
```
class _FriendsFindState extends State<FindFriendsPage> {
  @override
  void initState() {
    // Method called when object is created
  }

  @override
  void dispose() {
    // Method called when object is disposed
  }

@override
  Widget build(BuildContext context) {
    // Method to build screen components
  }
}
```

10.1.2 Rozložení

Jednotlivé rozložení obrazovek je zobrazeno na obrázku 10.2. Primárním cílem bylo vytvořit velmi jednoduché a poutavé uživatelské rozhraní, ve kterém se bude orientovat uživatel jakéhokoliv věku. Pro využívání všech funkcionalit aplikace je potřeba mít internetové připojení.

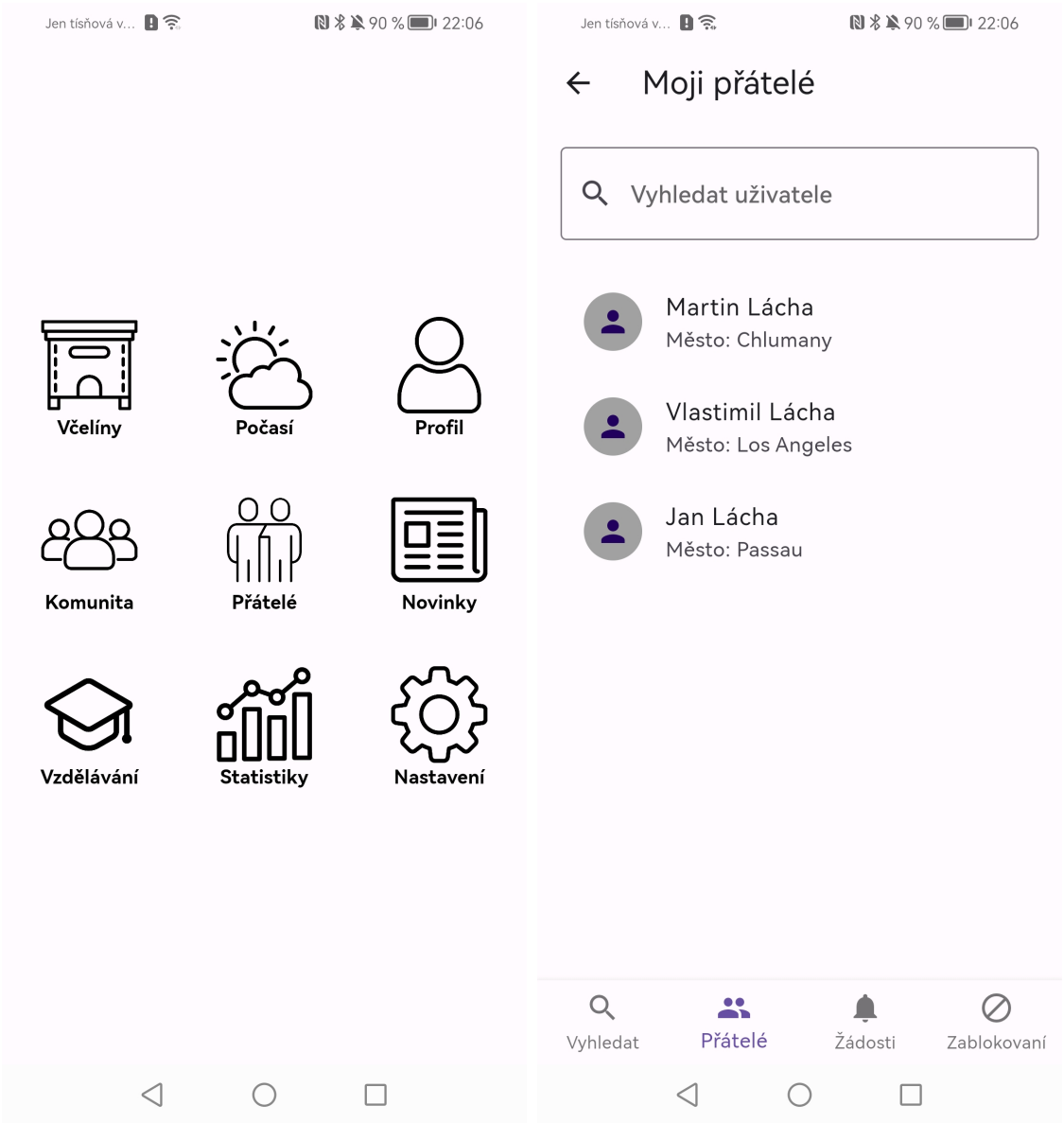
Po úspěšném přihlášení do aplikace je uživateli zobrazeno hlavní menu aplikace, které slouží jako rozcestník do všech částí aplikace, viz obrázek 10.3(a).

Z hlavního menu má zbytek aplikace ploché rozložení (nezanořují se obrazovky do sebe), aby uživatel nemusel příliš složitě hledat některé často používané funkce.



Obrázek 10.2: Rozložení obrazovek v mobilní aplikaci

Jedinou výjimkou je část pro včelíny. Pro tuto část aplikace bylo zvoleno hlubší zanoření obrazovek, jelikož na sebe jednotlivé části logicky navazují a ploché zobrazení by bylo pro uživatele matoucí. Pro navigaci v aplikace se využívá dotyk na jednotlivá tlačítka, ikony a v některých částech horizontální tah prstem, viz obrázek 10.3(b).



(a) Hlavní menu (b) Přátelé

Obrázek 10.3: Ukázky aplikace

Pro navigaci mezi jednotlivými obrazovkami se ve Flutteru používá třída **Navigator**, která si ukládá pořadí stránek, které prošel a tedy je možné zobrazit předchozí obrazovku pomocí tlačítka zpět. Obě situace jsou znázorněny v ukázce 10.3.

Zdrojový kód 10.3: Ukázka změny obrazovky

```
// Navigate to the next page
Navigator.push(context,
    MaterialPageRoute(builder: (context) => nextPage));

// Navigate to the previous page
Navigator.pop(context);
```

10.2 Komunikace se serverem

Data na jednotlivých obrazovkách je při načtení potřeba získat z komunikačního serveru, kde jsou uložena. Aby Android aplikace mohla využívat přístup k internetu a komunikaci se vzdálenými servery, je nezbytné definovat oprávnění v souboru **AndroidManifest.xml**, viz ukázka 10.4.

Zdrojový kód 10.4: Android oprávnění pro přístupu k internetu

```
<uses-permission android:name=android.permission.INTERNET/>
```

10.2.1 Posílání požadavků

Pro získání dat ze serveru je potřeba poslat požadavek. Pokud byl požadavek úspěšně zpracován, klient obdrží požadovaná data. Posílání požadavků zajišťuje třída **RequestSender**, která využívá knihovnu **http** pro práci s posíláním požadavků. Třída obsahuje následující metody:

- **sendRequest** - požadavek s daty,
- **sendRequestWithParams** - požadavek s parametry,
- **sendRequestWithImages** - požadavek s obrázky,
- **testServerConnection** - otestování připojení k serveru,

Každému požadavku jsou přiloženy následující hlavičky pro specifikaci požadavku, viz ukázka 10.5. **Content-Type** určuje formát těla požadavku a **Accept** naopak určuje specifikaci formátu odpovědi. **Access-Control-Allow-Origin** umožňuje přistupovat ke zdrojům z jakékoliv domény.

Zdrojový kód 10.5: Hlavičky požadavku

```
"Content-Type": "application/json",
"Accept": "application/json",
"Access-Control-Allow-Origin": "*"
```

Pokud bude požadavek odesílat data, je potřeba instanci převést do formátu JSON. Každý objekt, který bude posílán na komunikační server obsahuje metodu pro převod hodnot instance do JSON formátu, viz ukázka 10.6.

Zdrojový kód 10.6: Převod parametrů do JSON formátu

```
@override
Map<String, dynamic> toJson() {
    return {
        'title': title,
        'date': date.toString(),
        'activity': activity,
        'type': type,
        'notes': notes,
    };
}
```

Pokud je požadavek posílán na endpoint, který vyžaduje autentizaci, dodatečně se ještě přidává hlavička s JWT tokenem. Podle typu metody se odešle požadavek na komunikační server a čeká na odpověď, viz ukázka 10.7.

Zdrojový kód 10.7: Nastavení autentizačního tokenu a odeslání požadavku

```
if (sendToken) {
    final token = await SharedPreferencesUtils().
    getStringValue(ComponentConstant.bearerToken);
    headers[HttpHeaders.authorizationHeader] = "Bearer
    $token";
}

switch (method) {
case 'GET':
  return await http.get(uri, headers: headers);
case 'POST':
  return await http.post(uri, body: body, headers: headers);
case 'PUT':
  return await http.put(uri, body: body, headers: headers);
case 'DELETE':
  return await http.delete(uri, body: body, headers: headers
   );
default:
  logger.e("Unknown method $method");
}
```

10.2.2 Zpracování odpovědi

Při neúspěšném zpracování požadavku na serveru je uživateli vrácen pouze stavový kód. Ten uživateli upřesňuje z jakého důvodu nebyl požadavek úspěšně zpracován, viz ukázka 10.8.

Zdrojový kód 10.8: Ukázka zpracování stavového kódu

```
@override
void processResponse(http.Response response, context) {
    var statusCode = response.statusCode;
    if (statusCode == HttpStatus.created) {
        Navigator
            .push(context, MaterialPageRoute(
                builder: (context) => const SignInPage()));
    } else if (statusCode == HttpStatus.conflict) {
        ScaffoldMessenger.of(context).showSnackBar(
            PopupMessage.createErrorPopUpMessage(
                LanguageTranslation
                    .of(context)
                    !.value('already_exist_email')));
    } else if (statusCode == HttpStatus.internalServerError)
    {
        ScaffoldMessenger.of(context).showSnackBar(
            PopupMessage.createErrorPopUpMessage(
                LanguageTranslation
                    .of(context)
                    !.value('internal_server_error')));
    }
}
```

Pokud má odpověď serveru obsahovat data (např. získání seznamu přátel), je z těla požadavku dekódován obsah, který je následně převeden do nové instance objektu, viz ukázka 10.9.

Zdrojový kód 10.9: Vytvoření instance objektu z těla odpovědi

```
final json = jsonDecode(utf8.decode(response.bodyBytes));
NewsDetailDto newsDetail = NewsDetailDto.fromJson(json);
```

Implementace namapování hodnot z JSON formátu do nové instance objektu je znázorněna v ukázce kódu 10.10.

Zdrojový kód 10.10: Mapování JSON na instanci objektu

```
factory NewsDetailDto.fromJson(Map<String, dynamic> json) {
    return NewsDetailDto(
        id: json['id'],
        title: json['title'],
        article: json['article'],
        author: json['author'],
        date: json['date']);
}
```

10.3 Konfigurace a Shared Preferences

Konfigurace je načtena ze souboru při spuštění aplikace, viz ukázka 10.11. Třída vrací jedináčka[2] (Singleton) a pro získání hodnoty podle klíče slouží metoda **getConfig**. Konfigurace obsahuje následující hodnoty:

- **salt** - tajný klíč přidaný při šifrování hesel,
- **api_root_path** - URL prefix pro všechny API endpointy,
- **open_weather_api_key** - API klíč pro autentizaci ve službě OpenWeather.

Zdrojový kód 10.11: Načtení konfigurace

```
Map<dynamic, dynamic> _config = {};

Future<AppConfiguration> load(String name) async {
    String jsonContent =
        await rootBundle
                .loadString("assets/config/$name.json");
    _config = json.decode(jsonContent);
    return _singleton;
}

T getConfig<T>(String key) {
    return _config[key] as T;
}
```

V aplikaci jsou konfigurační parametry, které jsou často používané a jejich hodnota může v čase měnit. Adresu komunikačního serveru je dobré mít uloženou, aby jí uživatel nemusel při každém spuštění aplikace zadávat a zároveň není možné ji mít pevně definovanou, pokud se bude časem měnit. Pro uložení takových hodnot je použita knihovna **shared_preferences**. Důležité ukládané konfigurační parametry v aplikaci:

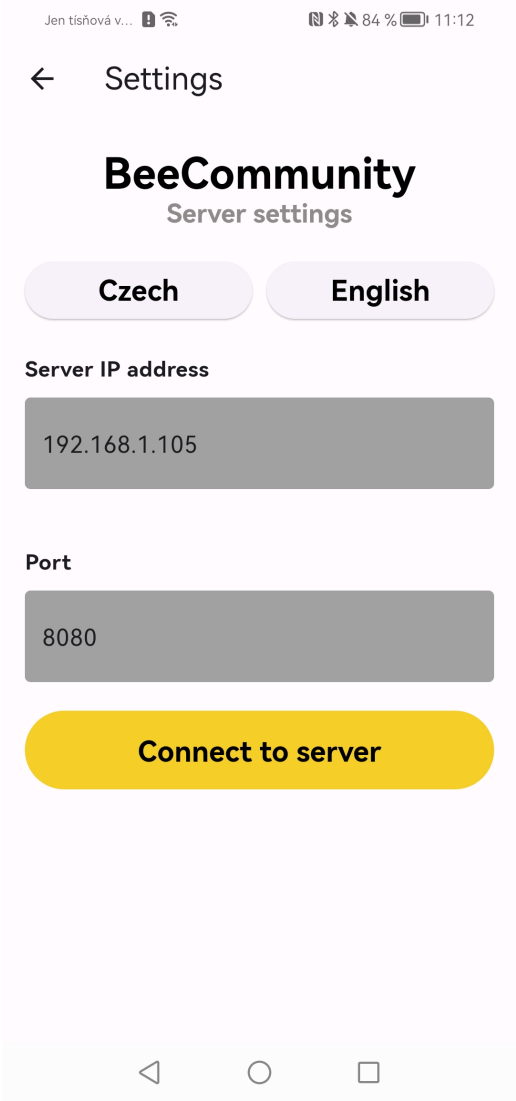
- **server_ip_address** - IP adresa serveru,
- **port** - port serveru,
- **locale** - nastavení jazyka aplikace,
- **bearer_token** - autentizační JWT token.

[2] Návrhový vzor vracející vždy stejnou instanci třídy.

10.4 Jazyk

Aplikace podporuje prozatím dva jazyky. Jako výchozí jazyk je nastavena angličtina a alternativním jazykem je čeština. Jednotlivé jazyky lze libovolně měnit v nastavení aplikace, viz obrázek 10.4.

(a) Čeština



(b) Angličtina

Obrázek 10.4: Nastavení jazyku aplikace

Po ukončení aplikace bude aktuální jazyk uložen do **SharedPreferences** s klíčem *locale*. Překlady pro jednotlivé jazyky jsou uloženy ve složce **assets**, kde pro češtinu jsou texty uloženy v souboru **cs.json** a pro angličtinu jsou uloženy v souboru **en.json**, viz ukázka 10.12.

Zdrojový kód 10.12: Ukázka části souboru en.json

```
{
    "app_title": "BeeCommunity",
    "app_subtitle": "Application for beekeepers",
    "email": "Email",
    "password": "Password",
    ...
}
```

Pro získání textu z konkrétního jazyka slouží třída **LanguageTranslation**. Hodnoty se načtou ze souboru pro daný jazyk a uloží se do mapy, viz ukázka 10.13. Pokud hodnota nebude nalezena v mapě, vrátí se ** $key not found, která upozorní na neexistující hodnotu.

Zdrojový kód 10.13: Načtení textů jazyka

```
static Map<dynamic, dynamic> _localizedValues = {};

static Future<LanguageTranslation> load(Locale locale) async {
    LanguageTranslation translations =
        LanguageTranslation(locale);
    String jsonContent =
        await rootBundle
            .loadString(
                "assets/locale/${locale.languageCode}.json")
    ;
    _localizedValues = json.decode(jsonContent);
    return translations;
}

String value(String key) {
    return _localizedValues[key] ?? '** $key not found';
}
```

Získání konkrétní hodnoty podle klíče například pro text tlačítka je zobrazen v ukázce 10.14, který ukazuje získání hodnoty pro klíč *pickup_image*.

Zdrojový kód 10.14: Vrácení hodnoty podle klíče

```
LanguageTranslation.of(context)!.value('pickup_image')
```

10.5 Grafy

Pro zobrazení grafů se používá knihovna **syncfusion_flutter_charts**. Grafy budou v aplikaci použity pro zobrazení statistik z určitého časového úseku. Použity budou spojnicový grafy pro zobrazení vývoje hodnot např. počet uživatelů v aplikaci nebo

stav úlu, viz obrázek 10.5. Jednotlivé čáry v grafu popisují jednu metriku pro konkrétní úl podle škály popsané v legendě grafu (stav populace, plodu, zásob, zdrojů v okolí atd.). Ukázka implementace spojnicového grafu je znázorněna v ukázce 10.15

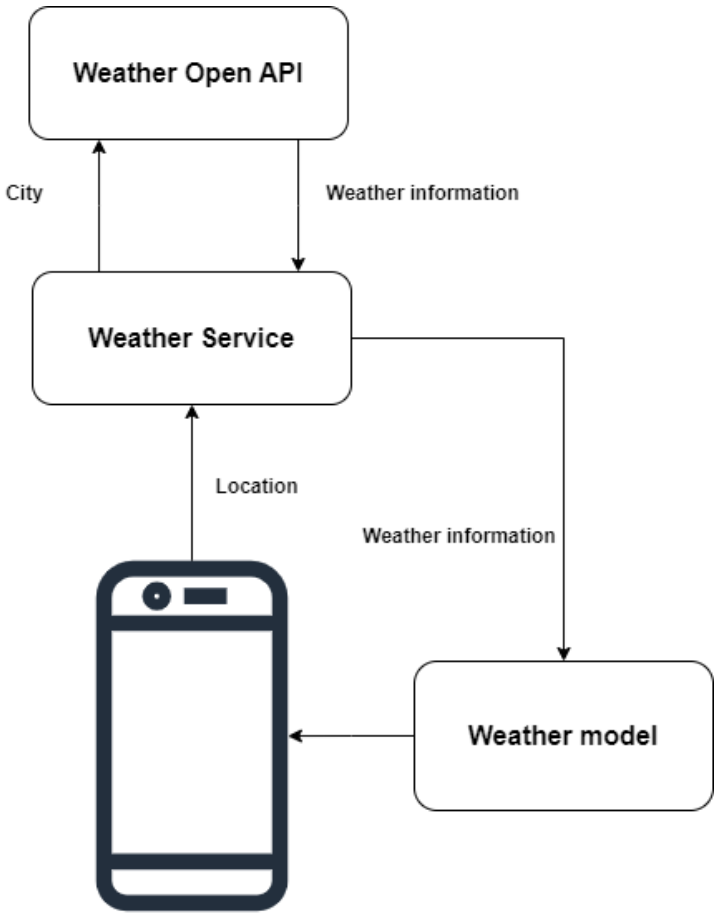
Zdrojový kód 10.15: Ukázka implementace spojnicového grafu

```
SfCartesianChart(
  series: <CartesianSeries>[
    LineSeries<GraphOverviewItem, DateTime>(
        name: LanguageTranslation
                .of(context)!.value(tooltipLabel),
        dataSource: data,
        xValueMapper: (GraphOverviewItem item, _)
            => DateTime.parse(item.date),
        yValueMapper: (GraphOverviewItem item, _)
            => item.count)])
```

Obrázek 10.5: Spojnicový graf zobrazující stav úlu

10.6 Počasí

V aplikaci bude možné získat informace o počasí v místě jak samotného zařízení, tak jednotlivých včelínů. Následující diagram ukazuje proces získávání informací o počasí v aplikaci. Počasí se bude získávat z OpenWeatherAPI, které poskytuje rozhraní pro získání informací o současném počasí a předpovědi daného dne. OpenWeatherMap je online služba, která poskytuje globální data o počasí. Poskytuje data o současném počasí, předpovědi slunečního záření, historická data o počasí a další. Rozhraní poskytuje data ve formátu JSON, XML nebo HTML v metrických nebo imperiálních jednotkách. Pro neplacenou verzi je nastaven limit 1000 požadavků na API za den a při překročení toho limitu se účtuje poplatek za každé další volání[42].



Obrázek 10.6: Diagram načítání informací o počasí

Pro určení aktuálního počasí (v místě mobilního zařízení nebo úlu) je potřeba informace o přesné geografické poloze zařízení. Pro získání polohy je potřeba oprávnění pro získání přesné polohy. Pro Android zařízení je potřeba přidat do *AndroidManifest.xml* oprávnění pro získávání polohy:

- **ACCESS_COARSE_LOCATION** - oprávnění umožňující aplikaci přístup k přibližné poloze.
- **ACCESS_FINE_LOCATION** - umožňuje aplikaci přístup k přesné poloze [43].

Pro operační systém iOS je potřeba přidat do *Info.plist* oprávnění:

- **NSLocationWhenInUseUsageDescription** - oprávnění pro možnost získání polohy, při spuštění aplikace na popředí [44]

O získání geografické polohy se stará knihovna **geolocator**. Pro vrácení současné polohy je potřeba získat oprávnění ze strany uživatele, kterému je oznámeno, že aplikace bude využívat geografickou polohu zařízení, viz ukázka kódu 10.16. Ze třídy **Geolocator** se zavolá metoda **getCurrentPosition**, kde je nastaven parametr **LocationAccuracy.high**, který specifikuje, že požadujeme vysokou přesnost.

Zdrojový kód 10.16: Získání geografické polohy v mobilním zařízení

```
1   Future<Position> getCurrentPosition() async {
2       // Check permission
3       LocationPermission permission =
4           await Geolocator.checkPermission();
5       if (permission == LocationPermission.denied) {
6           permission =
7               await Geolocator.requestPermission();
8       }
9
10      // Get current location
11      return await Geolocator.getCurrentPosition(
12          desiredAccuracy: LocationAccuracy.high);
13  }
```

Při volání rozhraní pro získání počasí je potřeba vlastnit API klíč, který je uživateli vygenerován po registraci. Pro získání informací o počasí se posílá požadavek společně s parametry. V aplikaci budou použity následující endpointy:

- **/data/2.5/weather** - vrací informace o současném počasí, viz ukázka dat 10.17

Každý požadavek bude mít navíc následující parametry:

- **lat** - zeměpisná šířka,
- **lon** - zeměpisná délka,
- **units** - jednotky měření (metric, imperial),
- **appid** - vygenerovaný API klíč.

Zdrojový kód 10.17: Ukázka části dat o počasí

```
{
  "coord": {
    "lon": 13.9654,
    "lat": 49.0666
  },
  "weather": [
    0: {
      "main": "Rain",
      "description": "moderate rain",
    }
  ],
  "name": "Vlachovo Brezi",
}
```

10.7 Fotografie

Fotografie budou v aplikaci užitečné například v rámci diskuzních fór, kdy může uživatel k dotazu nebo při sdílení informace přiložit fotografii. V mobilní aplikaci bude použita knihovna **image_picker** poskytující rozhraní pro výběr a pořizování fotografií z mobilního zařízení. Knihovna vyžaduje zařízení s operačním systémem Android 5.0+ nebo iOS 11+ [45]. Ukázka výběru fotografie úlu ze zařízení je znázorněna v ukázce 10.18. Pro použití fotografií z úložiště nebo pořizování fotografií z fotoaparátu je nutné nastavit následující oprávnění:

- **iOS**
  - **NSPhotoLibraryUsageDescription** - oprávení pro získávání fotografií z galerie
  - **NSCameraUsageDescription** - oprávnění pořizovat fotografie

Zdrojový kód 10.18: Výběr fotografie ze zařízení

```
void pickHiveImage() async {
    final XFile? pickedImage =
        await picker.pickImage(source: ImageSource.gallery);
    setState(() {
      if (pickedImage != null && mounted) {
        setState(() {
          hiveImage = File(pickedImage.path);
        });
    }});
}
```

Integrace senzorů 11

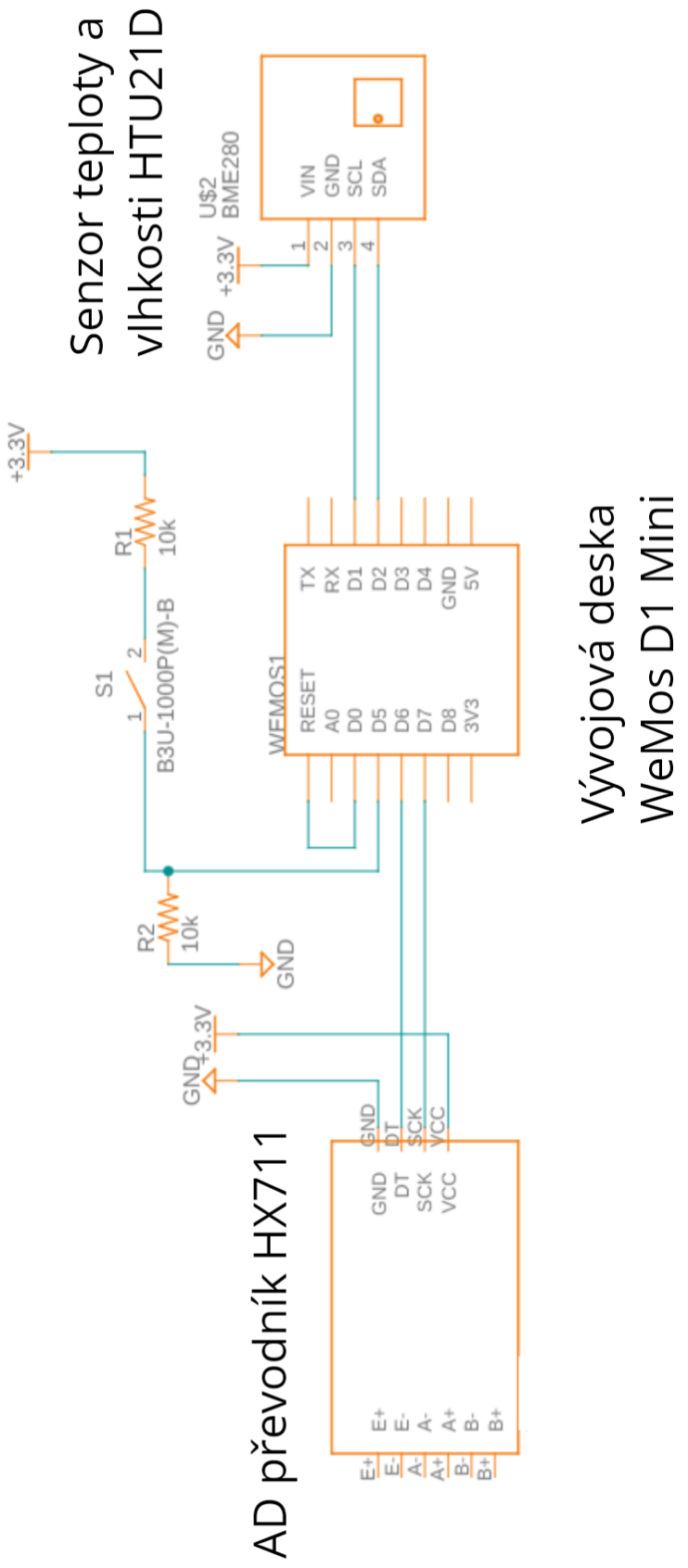
Tato část diplomové práce bude popisovat postup integrace monitorování úlu pomocí senzorů a posíláním dat na komunikační server. Pro vývoj byly využity technologie, viz obrázek 11.1.

Obrázek 11.1: Použité technologie pro monitorování úlu

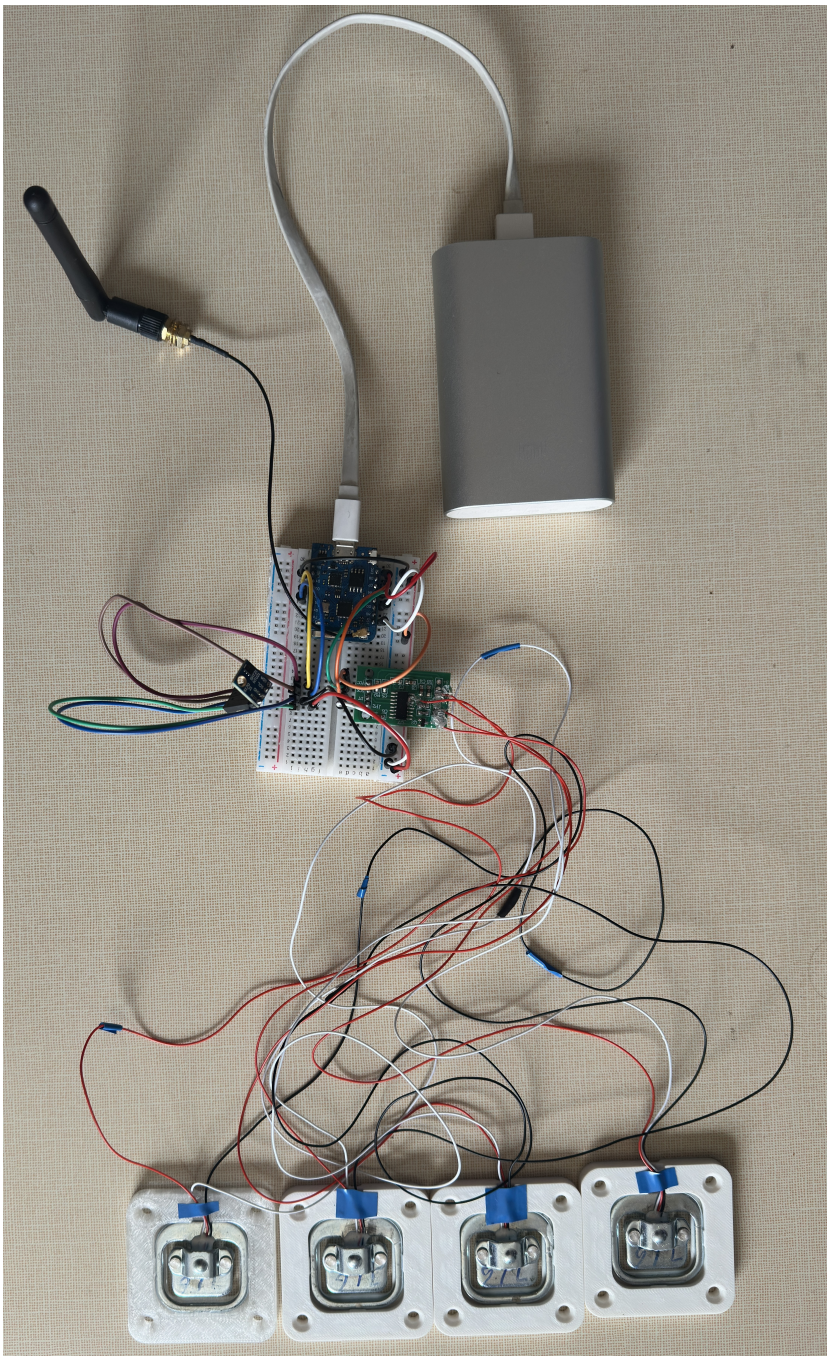
11.1 Komponenty

Na obrázku 11.2 je znázorněno blokové schéma zapojení jednotlivých komponent a jejich popis. Výběr komponent a schéma zapojení bylo konzultováno s panem Štěpánem Koubou. Prototyp zapojení je znázorněn na obrázku 11.3, který se skládá z následujících komponentů:

- **Vývojová deska WeMos D1 Mini** - Vývojová deska s Wi-Fi modulem ESP-8266EX pro bezdrátovou komunikaci a analyzování Wi-Fi sítí.
- **AD převodník HX711** - Modul pro vysoce přesné měření analogového signálu (24-bit s dvěma kanály).
- **Váhový senzor** - Váhový senzor tvořen pružným členem s tenzometrem. Maximální hmotnost je 50 kilogramů.
- **Senzor teploty a vlhkosti HTU21D I2C** - Modul pro měření teploty a vlhkosti v okolí. Dokáže měřit teplotu v rozsahu -40 až +125 stupňů Celsia a vlhkost v rozsahu 0 až 100%.



Obrázek 11.2: Blokové zapojení



Obrázek 11.3: Prototyp zapojení senzorů

11.2 Program

Pro vývoj bylo použito vývojové prostředí **Visual Studio Code** a framework **Platformio**. Vývojová deska obsahuje Micro USB, díky kterému je možné nahrát program z počítače do zařízení. Program bude ve smyčce posílat pravidelně data ze

senzorů na komunikační server. Každý program musí obsahovat metody **setup** a **loop** [46]. V první metodě se provádí nastavení při spuštění, v rámci kterého dojde k připojení na Wi-Fi sít a inicializaci I2C sběrnice a senzorů, viz ukázka 11.1.

Zdrojový kód 11.1: Nastavení Wi-Fi a senzorů

```
// Set device as station
WiFi.mode(WIFI_STA);
// Connect to WiFi
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED)
{
  delay(1000);
  Serial.println("Waiting for connection");
}
// Init I2C and sensors
Wire.begin();
scale.begin(DOUT, CLK);
sht.begin();
// Scale calibration
scale.set_scale(calibration_factor);
scale.set_offset(offset);
```

V metodě **loop** je zdrojový kód, který se bude vykonávat opakovaně ve smyčce, viz ukázka 11.2. Nejprve se aktivuje LED dioda signalizující, že zařízení bude měřit data ze senzorů. Nastaví se kalibrační faktor a offset. Změří se váha úlu a následně se získají hodnoty pro teplotu a vlhkost. Hodnoty se metodou **sendHttpToServer** pošlou přes HTTP na komunikační server. Nakonec se zařízení uspí na hodinu.

Zdrojový kód 11.2: Získávání dat ze senzorů

```
void loop {
    digitalWrite(led, HIGH);
    // Get weight of hive
    weight = scale.get_units();
    delay(10);
    // Read temperature and humidity
    sht.read();
    delay(10);
    hive_temperature = sht.getTemperature();
    hive_humidity = sht.getHumidity();
    delay(10);

    // Print data to serial monitor
    Serial.print("Weight: ");
    Serial.println(weight);
    Serial.print("Hive temperature: ");
    Serial.println(hive_temperature);
    Serial.print("Hive humudity: ");
    Serial.println(hive_humidity);
```

```
21      // Send data to server
22      sendHttpToServer(
23          weight, hive_temperature, hive_humidity);

25      digitalWrite(led, LOW);
26      ESP.deepSleep(HOUR_SLEEP);
27  }
```

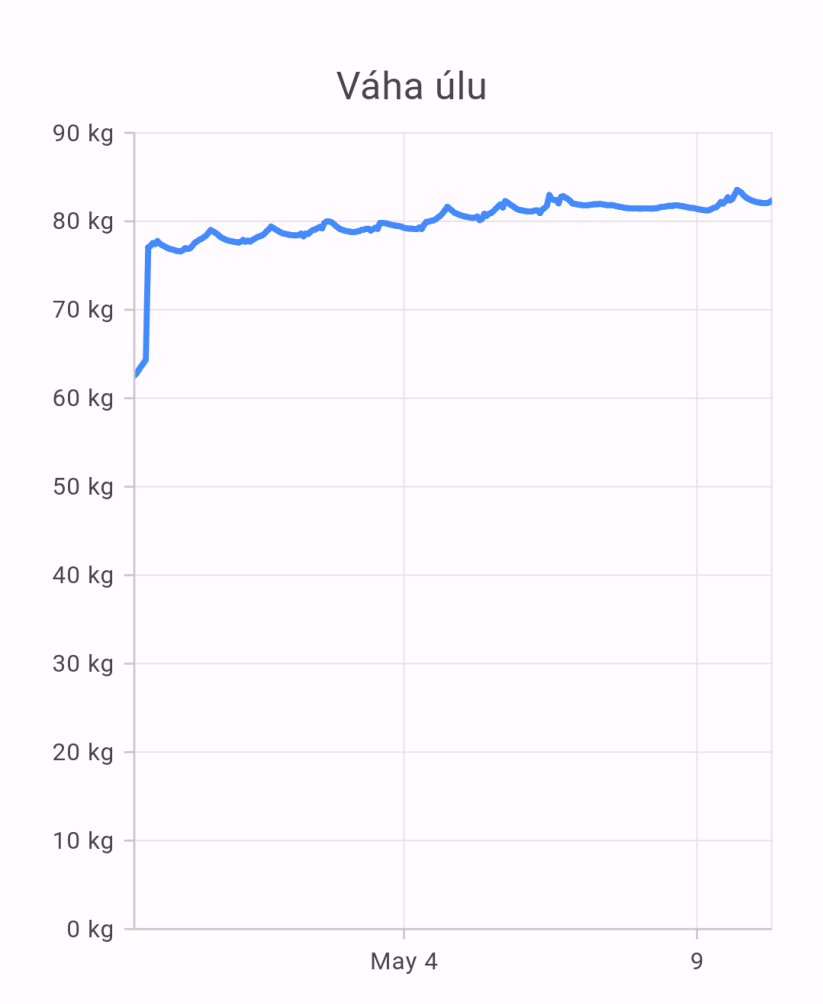
Realizace posílání dat na server je znázorněna v ukázce 11.3. Pro posílání dat se vytvoří JSON data a nastaví se hodnoty pro úl, váhu, teplotu a vlhkost, která se následně převedou do formátu řetězce pomocí funkce **serializeJson**. Specifikuje se cílová adresa podle URL a přidá se hlavička, která popisuje formát dat. Požadavek se odešle a uzavře se spojení.

Zdrojový kód 11.3: Posílání dat na server

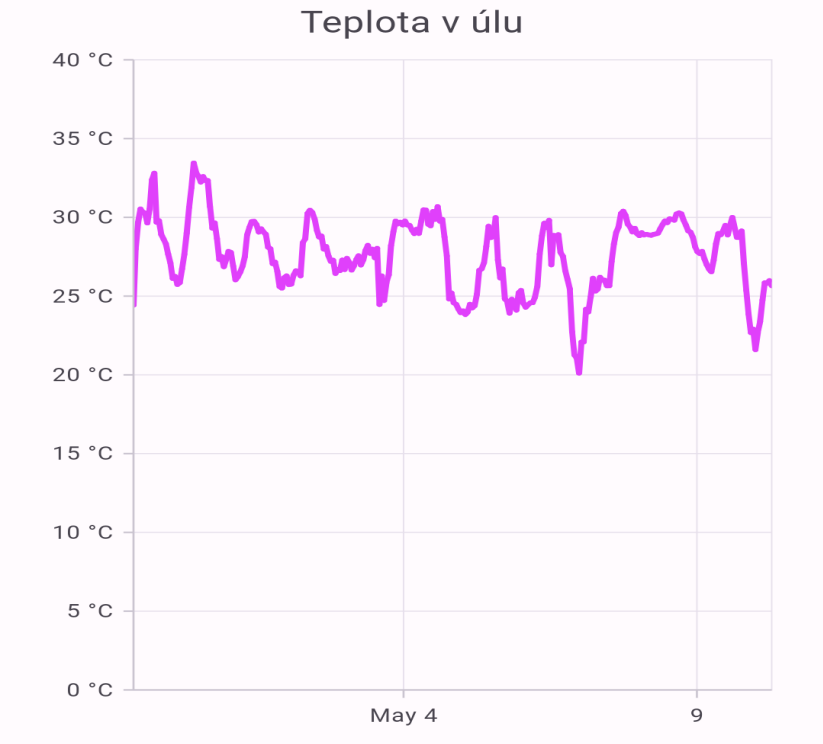
```
1  void sendHttpToServer(String url, float hiveTemperature,
       float hiveHumidity, float weight)
2  {
3    JsonDocument request;
4    request["hiveName"] = "Hive sensors";
5    request["hiveTemperature"] = hiveTemperature;
6    request["hiveHumidity"] = hiveHumidity;
7    request["weight"] = hiveWeight;

9    // Serialize JSON from measured data
10   String data;
11   serializeJson(request, data);

13   // Establishment connection with server
14   http.begin(client, url.c_str());

16   // Specify headers
17   http.addHeader("Content-Type", "application/json");
18   http.addHeader("Accept", "application/json");
19   http.addHeader("Access-Control-Allow-Origin", "*");

21   // Send the request
22   int httpCode = http.POST(data);

24   // Close connection
25   http.end();
26 }
```

11.3 Monitorování stavu úlu

Vytvořený monitorovací systém senzorů byl nainstalován do včelího úlu, kde byly monitorovány hodnoty váhy (obrázek 11.4), teploty (obrázek 11.5) a vlhkosti úlu (obrázek 11.6).



Obrázek 11.4: Ukázka dat váhy úlu



Obrázek 11.5: Ukázka dat teploty úlu

Obrázek 11.6: Ukázka dat vlhkosti úlu

Testování 12

Testování aplikace je nezbytnou částí jakéhokoliv vývoje. Zajišťuje ověření, že aplikace funguje správně, je spolehlivá a splňuje zadané požadavky. Testování jednotlivých částí aplikace bylo prováděno průběžně během celého vývoje. Pro tuto práci bylo použito několik typů testování.

12.1 Unit testy

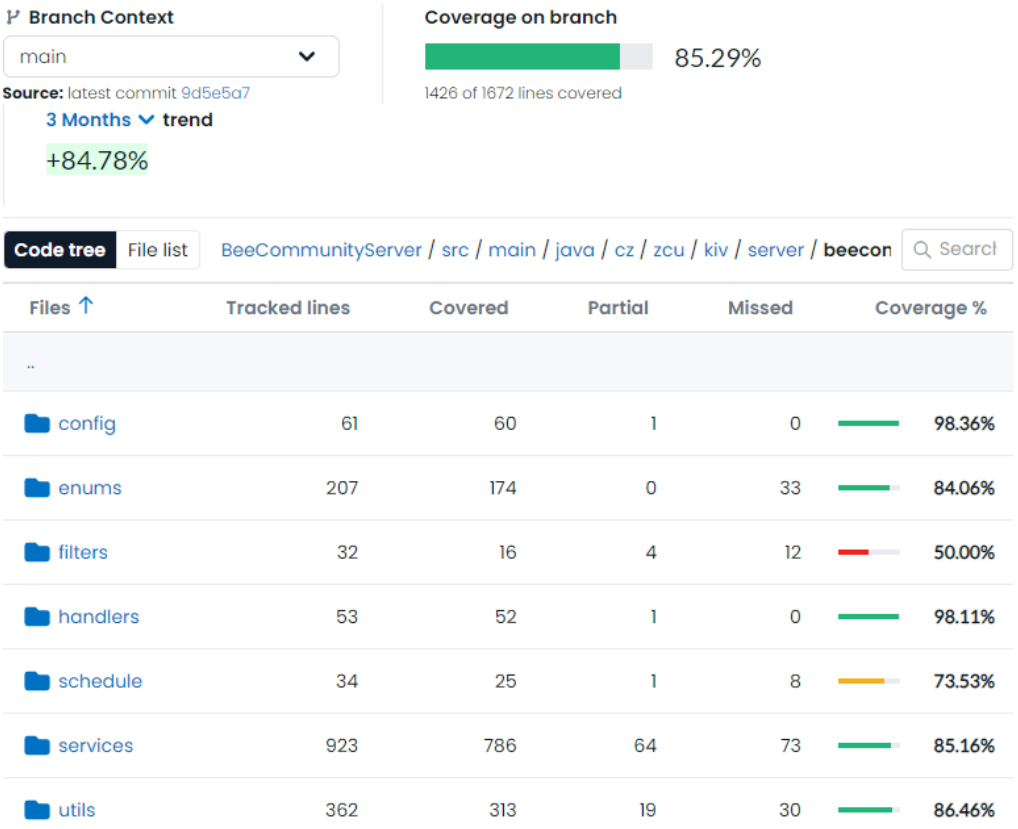
Unit testy nebo-li jednotkové testy jsou základním pilířem otestování funkcionality. Tyto testy se zaměřují na ověření správné funkcionality jednotek kódu nezávisle na ostatních částech aplikace. Jednotkou takového testování jsou například metody a funkce. Jednotkové testování je součást procesu kontinuální integrace a nasazení, které je spuštěno vždy po sestavení aplikace.

Některé třídy byly vynechány z testování, protože neobsahují funkcionalitu nebo se jedná o objekt uchovávající data. Následující části byly vynechány z jednotkového testování:

- **config** - konfigurace,
- **controllers** - vstupní body pro příchozí požadavky, které jej předají pro zpracování implementaci dané služby,
- **enums** - výčtové typy,
- **dto** - objekty pro výměnu dat s klientem,
- **entity** - definice objektů uložených v databázi,
- **repository** - JPA rozhraní pro jednotlivé tabulky s předdefinovanými funkcemi.

Pro analýzu pokrytí testů v projektu byl použit nástroj **JaCoCo**, které byl integrován do projektu. Při spuštění jednotkových testů v rámci sestavení aplikace, analyzuje pokrytí testů ve zdrojovém kódu. Nástroj **CodeCov** použije výsledky z

analýzy pro vizuální zobrazení pokrytí v jednotlivých částech zdrojového kódu [47]. Na komunikačním serveru bylo vytvořeno 258 jednotkových testů pro ověření všech funkcionalit napříč celým komunikačním serverem a celkové pokrytí testů v projektu je 85.29% (1426 z 1672 řádek kódu). Detailní popis pokrytí jednotlivých částí projektu je zobrazen, viz obrázek 12.1.



Obrázek 12.1: Zobrazení pokrytí testů nástroje CodeCov

12.2 Testování endpointů

Dalším důležitým typem ověření správné funkčnosti systému, bylo otestování jednotlivých endpointů. V průběhu vývoje k tomu byl použit nástroj **Insomnia**. Toto testování pomohlo identifikovat a odstranit problémy a chyby v implementaci API rozhraní. Klíčové body otestování každého endpointu byly:

- dostupnost,
- typ požadavku,
- validace parametrů,

- validace vstupních dat,
- autentizace a autorizace,
- stavový kód a data odpovědi.

Po vytvoření nového API endpointu byla otestována nejprve dostupnost tohoto endpointu. Dále byly ověřeny typy povolených metod, které jsou na daném endpointu. Bylo provedeno validování parametrů a vstupních dat, které jsou povinné nebo nepovinné. Většina endpointů vyžaduje ověření identity a proto byla otestována autentizace a autorizace pro jednotlivé typy uživatelských rolí. Nakonec byl testován návratový kód a data v těle odpovědi.

12.3 Případy užití

Pro ověření funkčnosti mobilního klienta bylo použito testování případů užití, které se velmi často používá při vývoji mobilních aplikací. Případy užití jsou scénáře, které popisují interakci mezi uživatelem a mobilní aplikací za účelem otestování použití nebo ověření funkčnosti aplikace. Budou popsány interakce uživatele a očekávané výsledky jednotlivých operací. Toto testování umožňuje systematicky prozkoumat funkční a vizuální oblasti mobilní aplikace a ověřit, zda aplikace správně reaguje na určité scénáře.

V následující části budou popsány nejdůležitější vybrané scénáře, kde je ke každému sepsán cíl, postup a výsledky. Testování bylo provedeno na následujících zařízení:

- Samsung Galaxy A51 (Android 13),
- OPPO Reno5 5G(Android 13),
- Huawei P30 Lite (Android 10),
- Xiaomi Pocophone F1 (Android 10),
- Redmi Note 9 (Android 12),
- Flutter emulator (Android 11) - emulátor,
- Google Pixel 4 (Android 10) - emulátor,
- Google Pixel 7 Pro (Android 5.0) - emulátor,
- Small Phone (Android 4.4) - emulátor.

12.3.1 Instalace a spuštění

Cíl: Úspěšná instalace z instalačního souboru APK a spuštění aplikace.
Předpoklad: Minimální verze Android 5.0 (API level 21).
Postup:

1. Stažení APK souboru do mobilního zařízení.
2. Spuštění instalace aplikace.
3. Ověření, zda byla aplikace nainstalována úspěšně.
4. Spuštění aplikace a ověření, že se zobrazila přihlašovací obrazovka.

Výsledek: Na zařízeních obsahující minimální verzi SDK byla úspěšně nainstalována aplikace.

12.3.2 Nastavení

Cíl: Změnit jazyk aplikace a uložení nastavení serveru.
Předpoklad: Připojení k internetu.
Postup:

1. Spustit aplikaci, stisknutí tlačítka *Nastavení serveru*, stisknutí tlačítka *Angličtina*.
2. Zadání IP adresy serveru do prvního pole.
3. Zadání portu serveru do druhého pole.
4. Stisknutí tlačítka *Připojit se k serveru*.
5. Ověřit, že aplikace přešla do přihlašovací obrazovky a byla zobrazena zpráva *Úspěšné připojení*.
6. Ukončení aplikace.
7. Spuštění aplikace.
8. Ověřit, že nyní je aplikace v angličtině.
9. Stisknutí tlačítka *Server settings*.
10. Ověřit, že pole obsahují IP adresu a port serveru.

Výsledek: Aplikace je po druhém spuštění v anglickém jazyce a IP adresa a port komunikačního serveru jsou vyplněné v polích nastavení.

12.3.3 Registrace a přihlášení

Cíl: Registrovat a přihlásit se pod novým uživatelem.
Předpoklad: Stabilní připojení k internetu a nastavený server.
Postup:

1. Spustit aplikaci, stisknout tlačítko *Zaregistrovat se*.
2. Vyplnit validní e-mailovou adresu, heslo a potvrdit heslo (hesla musí být totožná).
3. Stisknutí tlačítka *Zaregistovat se*.
4. Aplikace přejde zpět do přihlašovací obrazovky a zobrazí se zpráva *Registrace proběhla úspěšně*.
5. Vyplnit e-mailovou adresu, heslo a následně stisknout tlačítko *Přihlásit*.
6. Ověřit změnu obrazovky na hlavní menu aplikace.

Výsledek: Uživatel se může přihlásit do aplikace po úspěšné registraci nového účtu.

12.3.4 Změna hesla

Cíl: Změnit heslo existujícímu uživateli a přihlásit se.
Předpoklady: Aktivní účet s existující e-mailovou adresou, stabilní připojení k internetu.
Postup:

1. Spustit aplikaci, stisknout tlačítko *Zapomněli jste heslo*.
2. Vyplnit e-mailovou adresu existujícího účtu do prvního pole a stisknout tlačítko *Odeslat*.
3. Ověřit zobrazení o odeslání zprávy a z příchozího e-mailu získat potvrzovací kód.
4. V části *Resetovat heslo* vyplnit e-mailovou adresu, heslo a potvrzovací kód z e-mailu a stisknout tlačítko *Resetovat heslo*.
5. Ověřit zobrazení zprávy o úspěsné změně hesla a obrazovky.
6. Vyplnit e-mailovou adresu a nové heslo a stisknout tlačítko *Přihlásit*.
7. Ověřit změnu obrazovky na hlavním menu aplikace.

Výsledek: Uživatel se po úspěšné změně hesla může do svého účtu přihlásit pomocí nového hesla.

12.3.5 Nedostupné internetové připojení

Cíl: Ověřit funkčnost aplikace po ztrátě internetového připojení.
Předpoklady: Připojení k internetu a existující aktivní účet.
Postup:

1. Spustit aplikaci a přihlásit se pomocí e-mailové adresy a hesla.
2. V zařízení vypnout připojení k internetu (Wi-Fi, mobilní data).
3. Procházet části aplikace (Nastavení, Přátelé, Komunita, Včelíny, Počasí, Profil, Vzdělání, Profil).
4. Ověřit, že na jednotlivých obrazovkách se nenačítají data a zobrazuje se chybová zpráva *Nelze se připojit k internetu*.

Výsledek: Aplikace stále funguje pro pohyb napříč aplikací bez možnosti vytváření, načítání a aktualizace dat vytvořené uživatelem.
Poznámka: Aplikace bude reagovat stejně i pro *Letový režim*.

12.3.6 Vytvoření včelínu, úlu a inspekce

Cíl: Vytvoření nového záznamu včelínu s úlem a inspekcí.
Předpoklady: Připojení k internetu a aktivní účet.
Postup:

1. Spustit aplikaci, přihlásit se, stisknout tlačítko *Včelíny* a poté tlačítko plus v pravém dolním rohu.
2. Po zobrazení formuláře vyplnit pole: Název včelínu, Prostředí, Umístění a stisknout tlačítko diskety (uložit) v pravém horním rohu.
3. Ověřit zobrazení zprávy informující, že včelín byl vytvořen a kliknout na záznam nově vytvořeného včelínu.
4. Stisknout tlačítko *Úly* a poté tlačítko plus v pravém dolním rohu.
5. Po zobrazení formuláře vyplnit pole: Název úlu, Zdroj včelstva, Barva úlu a Datum založení a stisknout tlačítko diskety (uložit) v pravém horním rohu.
6. Ověřit zobrazení zprávy *Úl vytvořen* a po na stránce *Úly* ověřit zobrazení nového záznamu s vytvořeným úlem.
7. Stisknout tlačítko *Inspekce* a tlačítko plus a vyplnit pole: Typ inspekce, Datum, Počasí, Populace včelstva, Množství zásob, Zdroje v okolí a Viditelnost plodů a stisknout tlačítko diskety v pravém horním rohu.

8. Ověřit zprávu *Inspekce vytvořena* a obrazovka se změní zpět na Inspekce, kde je vytvořen nový záznam inspekce.

Výsledek: Celý postup vytvořil pro přihlášeného uživatele nový záznam o včelíně s jedním úlem a prvním záznamem kontroly daného úlu.

12.3.7 Přídání nového přítele

Cíl: Vyhledat a přidat existujícího uživatele mezi přátele.
Předpoklady: Připojení k internetu a druhý aktivovaný učet
Postup:

1. Spustit aplikaci a přihlásit se.
2. Stisknout tlačítko *Přátelé* a poté tlačíko *Vyhledat* a zadat do pole jméno nebo e-mail druhého uživatele.
3. Po zobrazení uživatelů kliknout na tlačítko plus pro odeslání žádosti.
4. Na druhém účtu v obrazovce *Přátelé* v záložce *Žádosti* přijmout žádost o přátelství o ověřit oznámení *Žádost o přátelství přijata*.
5. Na stránce *Přátele* obou účtů ověřit, že nový přítel byl přidán.

Výsledek: Po úspěšném přihlášení byl vyhledán druhý učet podle e-mailu, jména nebo příjmení a byla mu zaslána žádost o přátelství. Po přijetí žádosti z druhého účtu bylo možné vidět účet druhého uživatele v obrazovce *Přátelé*

12.3.8 Vytvoření soukromého příspěvku

Cíl: Vytvořit nový příspěvek, který bude viditelný pouze pro přátele.
Předpoklady: Připojení k internetu a druhý aktivovaný účet, kterého má v přátelích a třetí učet, kterého nemá v přátelích.
Postup:

1. Spustit aplikaci a přihlásit se.
2. Stisknout tlačítko *Komunita*, stisknout Přátelé, stisknout tlačítko v pravém horním rohu a stisnout tlačítko *Přidat příspěvek*.
3. Vyplnit pole: Nadpis příspěvku, Text, Typ příspěvku a vybrat fotografii z galerie a stisknout tlačítko *Odeslat*.
4. Ověřit změnu obrazovky zpět na *Veřejné* a po stisknutí tlačítka *Soukromé* ověřit nově vytvořený příspěvek v seznamu a detail po kliknutí na příspěvek.

5. Ověřit z druhého účtu na obrazovce *Soukromé* zobrazení nového příspěvku.

6. Ověřit z třetího účtu na obrazovce *Soukromé*, že nový soukromí příspěvek nebyl přidán.

Výsledek: Uživatel vytvoří nový příspěvek, který je viditelný pro něj a jeho přátele mezi soukromými příspěvky.

Návrh rozšíření 13

Tato kapitola se bude zabývat rozšířeními systému ze stránky softwaru a hardwaru. Pro komunikační server a mobilního klienta budou navrženy další funkcionality. Pro část monitorování budou popsána rozšíření z hlediska přidání dalších typů senzorů.

13.1 Komunikační server

Při získávání dat ze senzorů se mezi rozšířením nabízí použití umělé inteligence, která by analyzovala naměřená data ze senzorů. Podle výsledků by mohla dále poskytovat uživateli rady na základě této analýzy. Také by mohla poskytovat rady na základě současného období (na co si dávat pozor, jak léčit, krmit apod.). Z obrazu a zvuku se může analyzovat, zda včelstvo nezasáhlo onemocnění či parazit. Z analýzy frekvence a amplitudy zvuku je možné rozpoznat rojení včel a na tuto událost upozornit uživatele.

13.2 Mobilní aplikace

Současná aplikace by mohla být rozšířena o množství informací o uložených entitách (včelíny, úly, uživatel, inspekce). Rozšíření části o vzdělávání o další lekce, které by předávaly informace začínajícím včelařům (např. rozpoznání nákazy včel, léčení apod.), kde cílem není nahrazovat v celé šíři různé knihy o včelaření, ale poskytnout nejdůležitější informace na jednom místě. Možnost odesílat soukromé zpráv nebo fotografie mezi jednotlivými uživateli. Zobrazení více možných statistik. Upozornění uživatele ve formě notifikací, které by vycházeli z analyzovaných dat na serveru a tím předejít nežádoucím událostem (rojení včelstva).

V mobilní aplikaci by bylo možné využít i umělé inteligence. Pokud včelař nemá označenou včelí matku, bývá její nalezení mezi ostatními včelami na rámku občas velmi náročné. S využitím mobilního zařízení by umělá inteligence analyzovala obraz a pokusila se včelí matku označit v reálném čase. Dalším využitím umělé inteligence by byla identifikace nemocí či parazitů z videa či fotografie.

13.3 Senzory

Pro účely této práce byl vytvořen prototyp monitorování s použitím vybraných senzorů. Celý model monitorování je možné rozšířit o další senzory, které už byly popsány v kapitole 8. V prototypové verzi byla monitorována váha, vnitřní teplota a vlhkost úlu.

Mezi další rozšíření se nabízí snímání teploty okolního prostředí úlu pro analýzu vlivu okolí na produkci a fungování včelstva. Váha by mohla být monitorována nejen jako jediná hodnota celého úlu, ale také by mohli být snímány hodnoty jednotlivých rámků. Analyzováním směsí obsažených ve vzduchu uvnitř úlu je možné získat další informace o zdraví a aktivitě včel. Ze záznamu zvuku lze vyhodnotit například, zda se včely připravují na rojení, čemuž se chce každý včelař vyhnout. Obraz by mohl poskytnout uživateli informace o současném dění v úlu nebo by mohl být použit pro výukové účely a trénování umělé inteligence.

Závěr 14

Cílem této diplomové práce bylo analyzovat vybrané mobilní aplikace podporující včelařské aktivity a vytvořit vlastní komunikační server a mobilní aplikaci pro platformu Android pro správu a monitorování stavu včelstev.

V diplomové práci byla nejprve provedena analýza vybraných mobilních aplikací, u kterých byly zhodnoceny výhody, nevýhody a dostupné funkce. Další část se zabývala návrhem vlastní mobilní aplikace, komunikačního serveru a monitorovacích senzorů. Popsány byly vybrané technologie, architektura, zabezpečení a komunikace mezi jednotlivými částmi systému.

V diplomové práci byla dále popsána samotná realizace jednotlivých částí. Komunikační server je vytvořen pomocí frameworku Spring Boot, který přijímá a zpracovává požadavky od klientů. Server implementuje REST API rozhraní pro vytvoření, aktualizaci a odstranění dat. Na serveru je řešena také autentizace a autorizace pro zabezpečení uživatelských dat. Mobilní aplikace je vytvořena pomocí frameworku Flutter a slouží pro správu včelařských aktivit. Bylo vytvořeno jednoduché uživatelské rozhraní s ohledem na snadné ovládání. Senzory monitorují váhu, teplotu a vlhkost úlu a posílají periodicky data na komunikační server. V jednotlivých částech implementace byly popsány nejdůležitější části systému (předávání dat, konfigurace, zpracování požadavků a odpovědí atd.).

Dále bylo v diplomové práci provedeno ověření funkčnosti a otestování jednotlivých částí práce pomocí několika typů testování. Mobilní aplikace byla testována na více zařízeních s rozdílnými parametry (velikost, verze operačního systému atd).

Poslední část se zaměřuje na návrh možných rozšíření komunikačního serveru, mobilní aplikace a monitorovacích senzorů.

Vytvořený systém podporuje včelaře tím, že jim umožňuje přehlednou správu vlastních úlů a výměnu informací s dalšími včelaři. Tím může přispět k dalšímu rozvoji včelaření a předávání znalostí mezi včelaři.

Seznam zkratek

API (Application Programming Interface) - aplikační programové rozhraní

CCD (Colony Collapse Disorder) - fenomén, kdy v krátkém časovém úseku většina dělnic opustí královnu a zanechá velké množství potravy a několik chův

CD (Continuous Deployment) - praktika pro kontinuální doručování změn do produkčního prostředí

CI (Continuous Integration) - praktika pro pravidelnou integraci kódu do společného repozitáře

CLI (Command Line Interface) - uživatelské rozhraní pro komunikaci s programy pomocí příkazového řádku

CRUD (Create, Read, Update, Delete) - základní operace nad záznamy (vytvoření, čtení, aktualizace, odstranění)

DevOps (Development and Operations) - postupy a nástroje pro rychlejší poskytování aplikací a služeb

DTO (Data Transfer Object) - objekt pro zapouzdření a posílání dat mezi dvěma aplikacemi

DVCS (Distributed Version Control System) - software pro správu verzí, kde je historie historie změn uložena na každém pracovním stroji

HTTP (Hypertext Transfer Protocol) - internetový protokol pro komunikaci s WWW servery

IP (Internet Protocol) - protokol síťové vrstvy poskytující datagramovou službu pro komunikaci

IT (Information Technology) - věda zabývající se technologiemi (počítač, mobil) pro uchování, zpracování, získávání a přenášení informací

JPA (Java Persistence API) - je standard programovacího jazyka Java pro objektově relační mapování

JSON (JavaScript Object Notation) - způsob zápisu dat nezávislý na počítačové platformě pro přenos dat organizována v polích nebo objektech

JWT (JSON Web Token) - je otevřený standard pro sdílení informací o zabezpečení mezi klientem a serverem

NFC (Near-Field Communication) - bezdrátová technologie pro bezpečnou a rychlou výměnu dat

PDF (Portable Document Format) - souborový formát pro prezentaci a výměnu dokumentů

QR (Quick Response) - maticový 2D kód pro vysokorychlostní čtení

RFC (Request for Comments) - je v informatice označení řady dokumentů popisujících internetové protokoly, systémy apod.

SDK (Software Development Kit) - balíček zdrojových kódu a nástrojů pro tvorbu aplikací pro specifické platformy

SMTP (Simple Mail Transfer Protocol) - e-mailový protokol pro přenos zpráv mezi poštovními servery

TCP (Transmission Control Protocol) - protokol transportní vrstvy v sadě protokolů TCP/IP pro spolehlivé doručování ve správném pořadí

TLS (Transport Layer Security) - kryptografický protokol pro zajištění bezpečné komunikace přes počítačovou síť

UI (User Interface) - označuje veškeré prvkové, interakční a vizuální aspekty aplikace, které uživatel vidí a interaguje s nimi

URI (Uniform Resource Identifier) - jednotný identifikátor zdroje nebo dokumentu na internetu

USB (Universal Serial Bus) - univerzální sériová sběrnice pro připojení periferií

VCS (Version Control System) - centralizovaný softwarový nástroj pro správu změn zdrojového kódu v průběhu času

WAR (Web Application Resource) - archiv pro soubory webové aplikace

WWW (World Wide Web) - systém prohlížení, ukládání a odkazování dokumentů na internetu

XML (Extensible Markup Language) - značkovací jazyk pro vytváření konkrétních značkovacích jazyků

Instalační příručka A

Instalační příručka slouží pro sestavení a zprovoznění běhu jednotlivých částí této práce (komunikační server, mobilní aplikace, senzory). Každá část bude obsahovat potřebné nástroje a postupy, které povedou k úspěšnému spuštění.

A.1 Komunikační server

Pro zprovoznění komunikačního serveru jsou potřebné následující nástroje:

- Git,
- Docker,
- docker-compose.

Pro nasazení komunikačního serveru z Git repozitáře nebo jiného zdroje, slouží následující postup:

1. git clone https://github.com/martinlacha/BeeCommunityServer.git
2. cd BeeCommunityServer
3. docker-compose up -d
4. docker ps -a

Postup je prováděn v příkazové řádce (CLI). Prvním krokem je stažení projektu z Git repozitáře. Změnit složky na BeeCommunityClient. Ve složce BeeCommunityServer se spustí třetí příkaz, který sestaví projekt a spustí jednotkové testy. Po sestavení nasadí a spustí aplikaci na daném zařízení. Pro kontrolu, že komunikační server byl nasazen správně, slouží třetí příkaz. Ukázka úspěšného sestavení a nasazení komunikačního serveru je zobrazena na obrázku A.1. Obrázek A.2 zobrazuje běh jednotlivých kontejnerů. Po těchto krocích je komunikační server zprovozněn a připraven k obsluze požadavků od mobilních klientů.

```
C:\Users\user\Desktop\test\BeeCommunityServer>docker-compose up -d
Creating network "beecommunityserver_server-network" with driver "bridge"
Building app
[+] Building 114.4s (13/13) FINISHED
 Creating beecommunity_postgres_1 ... done
 Creating beecommunity_app_1      ... done
 Creating beecommunity_flyway_1   ... done
```

Obrázek A.1: Ukázka sestavení a nasazení serveru

```
C:\Users\user\Desktop\test\BeeCommunityServer>docker ps -a
CONTAINER ID   IMAGE                  STATUS                     PORTS                    NAMES
6fa0da100dbd   flyway/flyway          Exited (0) 4 minutes ago                            beecommunity_flyway_1
ec7a59a5e10c   beecommunityserver_app Up 4 minutes               0.0.0.0:8080->8080/tcp   beecommunity_app_1
752dd2a1d2af   postgres:latest        Up 4 minutes               0.0.0.0:5432->5432/tcp   beecommunity_postgres_1
```

Obrázek A.2: Výpis Docker kontejnerů

A.2 Mobilní aplikace

Pro sestavení a instalaci mobilní aplikace je potřeba mít nainstalované následující nástroje:

- Flutter SDK,
- Git,
- mobilní zařízení s operačním systémem Android 5.0+.

Pro sestavení aplikace a vytvoření instalačního souboru .apk je potřeba provést následující postup:

1. git clone https://github.com/martinlacha/BeeCommunityClient.git
2. cd BeeCommunityClient
3. flutter build apk --release

Nejprve se naklonuje repozitář z Gitu. Změníme složku na *BeeCommunityClient* a třetí příkaz sestaví instalační soubor *beecommunity.apk*. Přesné umístění souboru je:

- *build/app/outputs/apk/release/beecommunity.apk*.

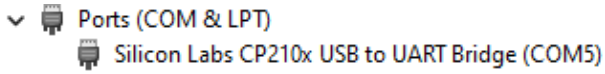
Pro instalaci mobilní aplikace do zařízení je potřeba mít mobilní zařízení s minimálním operačním systémem Android 5.0 a mít v uložený soubor APK, který po stisknutí nabídne možnost nainstalovat aplikaci. Po úspěšné instalaci se vytvoří v zařízení nová ikona pro spuštění mobilní aplikace.

A.3 Monitorování úlu

Pro sestavení a instalaci programu pro monitorování úlu je nezbytné mít nainstalované nástroje:

- Git,
- Arduino IDE,
- ovladač pro typ vývojové desky.

Pro spuštění programu pro monitorování pomocí senzorů je potřeba mít zapojené komponenty popsané v kapitole 11.1. Pro nahrání programu je potřeba nainstalovat příslušný ovladač pro vývojovou desku. Po jeho instalaci a připojení vývojové desky přes Micro USB k počítači, je zařízení viditelné ve správci zařízení mezi porty, viz obrázek A.3. Zařízení je připojené na port **COM5**. Pro sestavení aplikace je použito vývojové prostředí Arduino IDE, který podporuje vývoj vestavěných systému. V nastavení je potřeba vybrat správný typ vývojové desky. Pro tento projekt byla vybrán typ: **LOLIN(WEMOS) D1 mini (clone)**. Aplikace lze sestavit ve vývojovém prostředí přes grafické rozhraní. Po úspěšném nahrání programu je zařízení připraveno monitorovat a posílat data na běžící komunikační server.



Obrázek A.3: Správce zařízení

Uživatelská příručka B

Uživatelská příručka popisuje jednotlivé obrazovky mobilní aplikace a funkce, které se na nich nachází. Aplikace obsahuje následující obrazovky:

Přihlašovací obrazovka

Obrazovka slouží pro přihlášení uživatele do systému pomocí e-mailové adresy a hesla, viz obrázek C.1(a). Tlačítko *Zapomněli jste heslo?* obsahuje dva formuláře. První formulář je pro odeslání potvrzovacího kódu na e-mail uživatelského účtu, které slouží pro změnu hesla. Druhý formulář je samotná změna hesla, viz obrázek C.1(b). Do nastavení se uživatel dostane po stisku tlačítka *Nastavení serveru*, kde je možné nastavit jazyk aplikace a IP adresu a port komunikačního serveru, viz obrázek C.2(a). Pro registraci nového uživatele slouží formulář, na který se uživatel dostane přes tlačítko *Zaregistrovat se*, viz obrázek C.2(b). V aplikaci bylo vytvořeno několik uživatelských účtů (heslo je nastaveno na *password*):

- admin@community.com,
- martin.lacha@seznam.cz,
- vlastimil.lacha@gmail.com,
- random_email@seznam.cz,
- testing@gmail.com.

Hlavní menu

Hlavní menu je zobrazeno po přihlášení do aplikace. Slouží jako rozcestník do zbytku aplikace, viz ukázka C.3(a). Uživatel se může dostat do jednotlivých částí pomocí kliknutí na danou ikonu (Profil, Počasí, Včelíny, Novinky, Přátelé, Komunita, Vzdělání, Statistiky, Nastavení).

Počasí

Obrazovka zobrazí aktuální počasí mobilního zařízení podle jeho geografické lokace, pokud uživatel povolil získávat polohu zařízení. Zobrazení informací o aktuálním počasí je na obrázku C.3(b).

Profil

Na obrazovce *Profil* si může uživatel upravit osobní informace na stránkách *Účet* a *Adresa*, viz obrázek C.4(a). Další stránka slouží pro uživatele s administrátorskou rolí, kde může přiřadit a odebrat uživatelům administrátorskou roli nebo danému uživateli změnit přihlašovací e-mailovou adresu, viz obrázek C.4(b).

Novinky

Obrazovka zobrazuje uživateli seznam o novinkách a trendech v oblasti včelaření společně s autorem a datem publikování, viz obrázek C.5(a). Pro přidání novinky musí mít účet administrátorskou roli. Pro přidání nového příspěvku slouží tlačítko plus v pravém horním rohu, který zobrazí formulář, viz obrázek C.5(b). Novému článku je možné přidat pomocí tlačítek až 3 obrázky.

Přátelé

Slouží pro správu přátel, žádostí a blokování uživatelů. Po kliknutí na ikonu *Přátelé* z hlavního menu se zobrazí seznam přátel, viz obrázek C.6(a). Operace nad jednotlivými uživateli se zobrazí po kliknutí na jejich záznam, viz obrázek C.6(b).

V části *Vyhledat* je možné vyhledat uživatele v aplikaci podle e-mailové adresy, jména nebo příjmení a vyhledanému uživateli pomocí tlačítka plus poslat žádost o přátelství, viz obrázek C.7(a). *Žádosti* zobrazují seznam žádostí od ostatních uživatelů, které je možné přijmout nebo odmítnout. Poslední částí *Zablokovaní* je obrazovka se seznamem blokovaných uživatelů, které je možné tlačítkem zámku odebrat z toho seznamu, viz obrázek C.7(b).

Komunita

Komunita slouží pro sdílení příspěvků a dotazů mezi jednotlivými uživateli. Obrazovka zobrazuje seznam pro *Veřejné* a *Soukromé* příspěvky, viz obrázek C.8(a). Pro přidání nového příspěvku slouží tlačítko *Přidat příspěvek* nacházející se v pravém horním rohu. Po kliknutí na konkrétní příspěvek se zobrazí obrazovka detailu příspěvku s komentáři, viz obrázek C.8(b).

Vzdělání

Obrazovka zobrazující jednotlivé lekce, viz obrázek C.9(a), po kliknutí uživatele zobrazí detail o daném tématu, viz obrázek C.9(b).

Statistiky

Obrazovky poskytují uživateli přehled o statistikách v určitých oblastech. Obecné zobrazují uživateli souhrnné statistiky od všech uživatelů v aplikaci, viz obrázek C.10(a). Část *Přátelé* zobrazuje seznam přátel a po stisknutí se zobrazí statistiky daného uživatele rozdělené na statistiky podle jednotlivých včelínů a úlů, viz obrázek C.10(b). Soukromé jsou totožné jako obrazovky statistiky přítele.

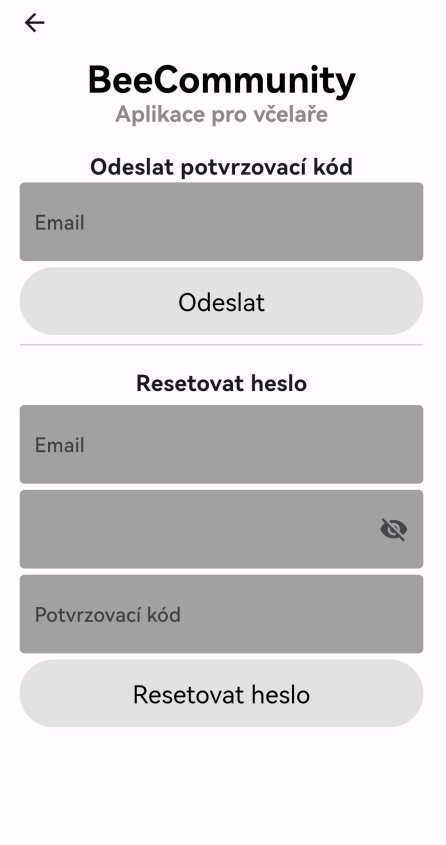
Včelíny

Včelíny představují hlavní část pro zaznamenávání a správu včelařských záznamů. Zobrazuje se seznam jednotlivých včelínů, viz obrázek C.11(a). *To-Do* obsahuje formulář pro vytvoření nové aktivity, viz obrázek C.11(b). *Kalendář* zobrazuje aktivity podle jednotlivých dnů, viz obrázek C.12(a). Při kliknutí na aktivitu se zobrazí detail, viz obrázek C.12(b). V detailu včelína jsou možné vidět informace o daném včelínu a v sekci *Úly* je seznam jednotlivých úlu, které se v něm nachází. Při zobrazení detailu úlu se zobrazí sekce informace, struktura, inspekce a senzory, viz obrázek C.13(a). Struktura úlu je dále znázorněna na obrázku C.13(b). Tlačítkem plus v pravém dolním rohu se zobrazí formulář pro vytvoření nového úlu, viz obrázek C.14(a). V části *Inspekce* jsou zobrazeny všechny kontroly daného úlu, viz obrázek C.14(b). *Senzory* pak zobrazují data naměřená z monitorování úlu.

Ukázky aplikace

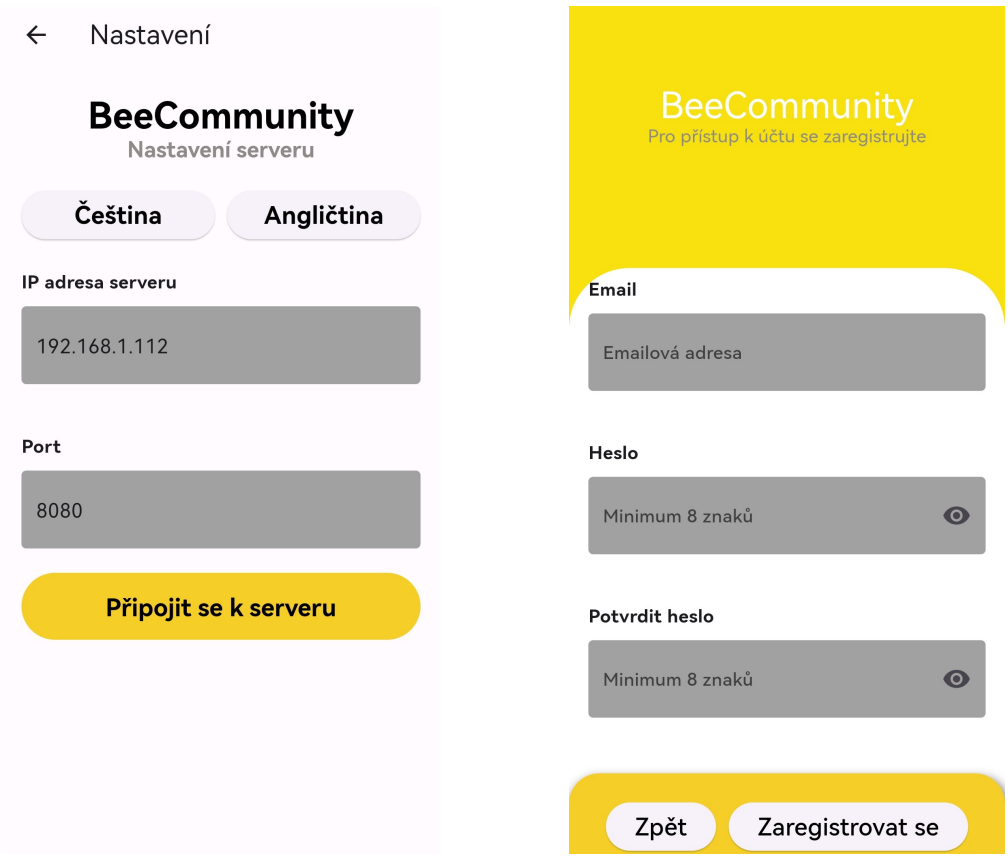
C

(a)



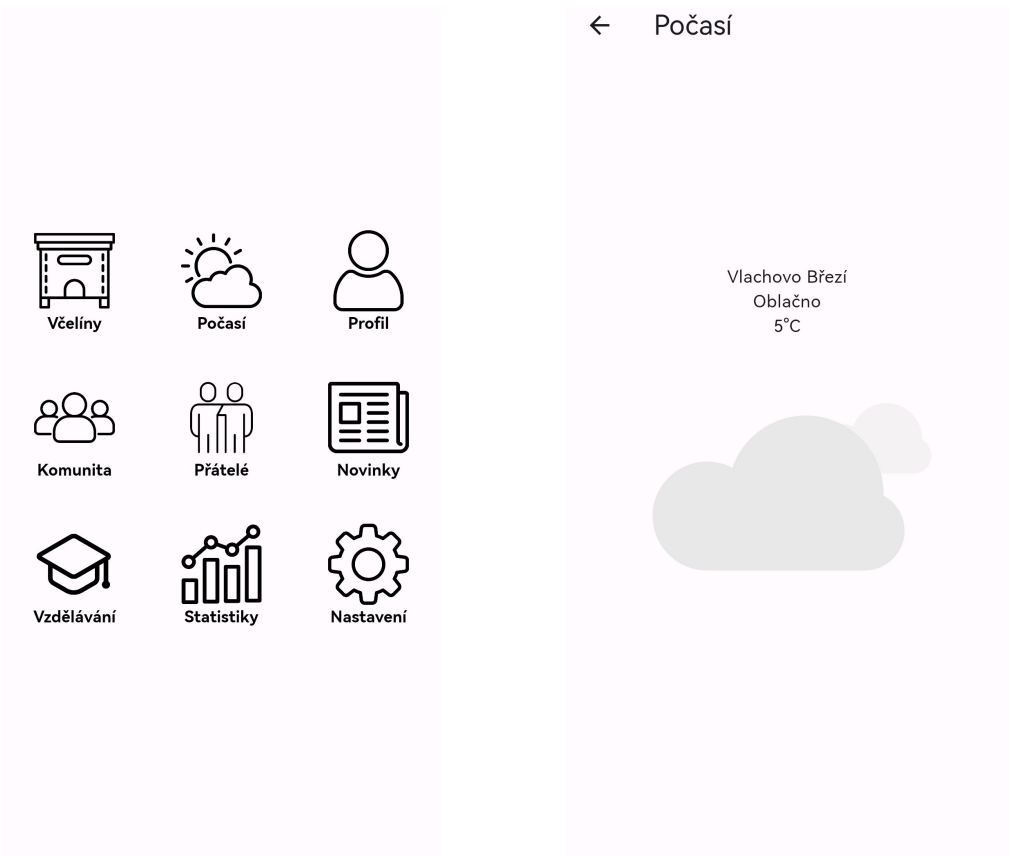
(b)

Obrázek C.1: Přihlašovací obrazovka a změna hesla



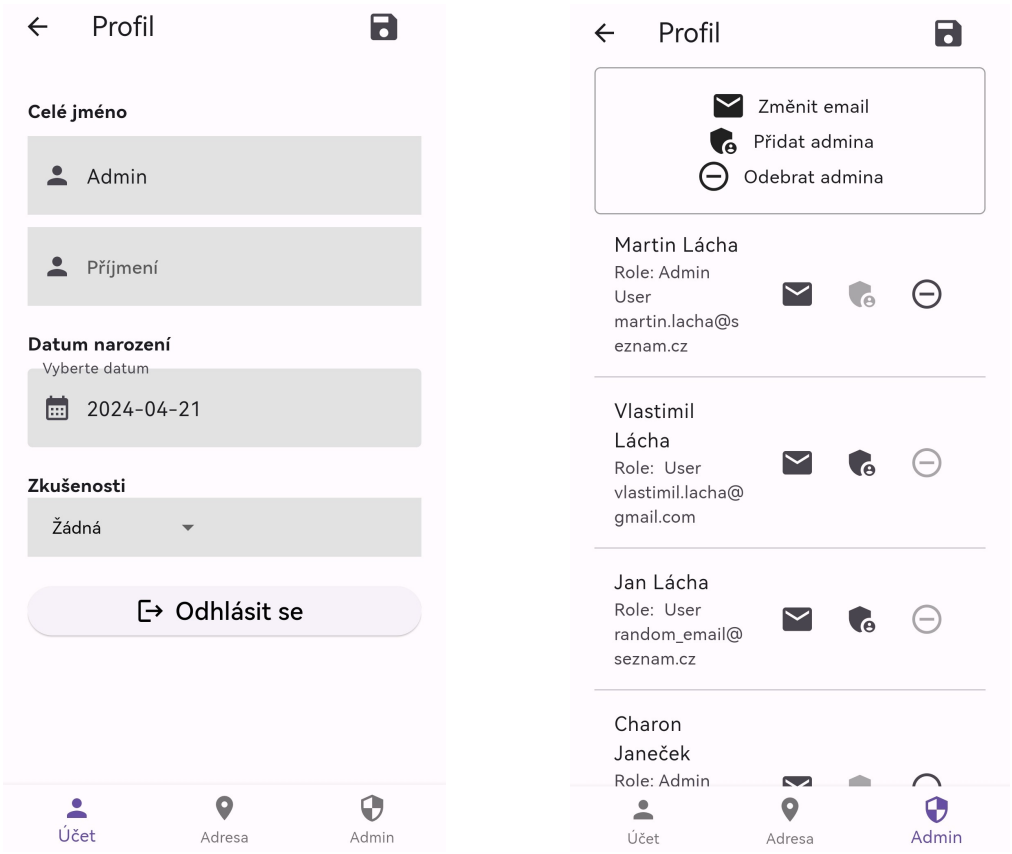
(a) (b)

Obrázek C.2: Nastavení a registrace uživatele



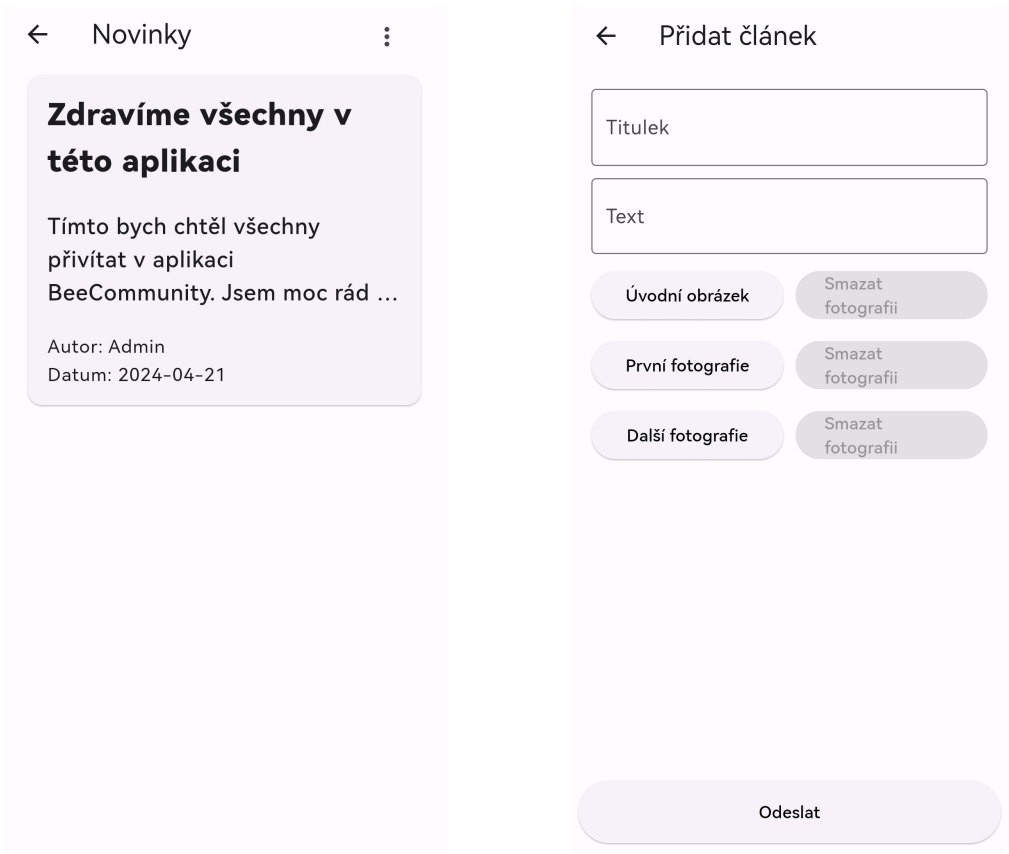
(a) (b)

Obrázek C.3: Hlavní menu a informace o počasí



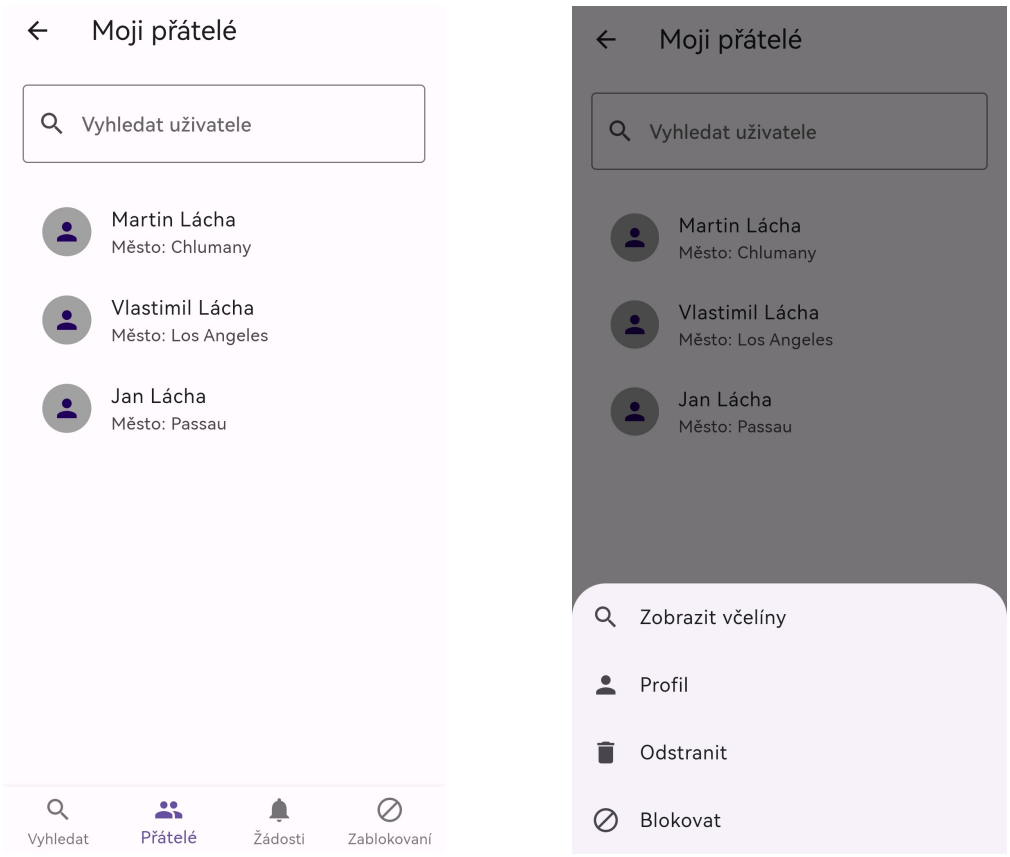
(a) (b)

Obrázek C.4: Profil uživatele a administrátorká obrazovka



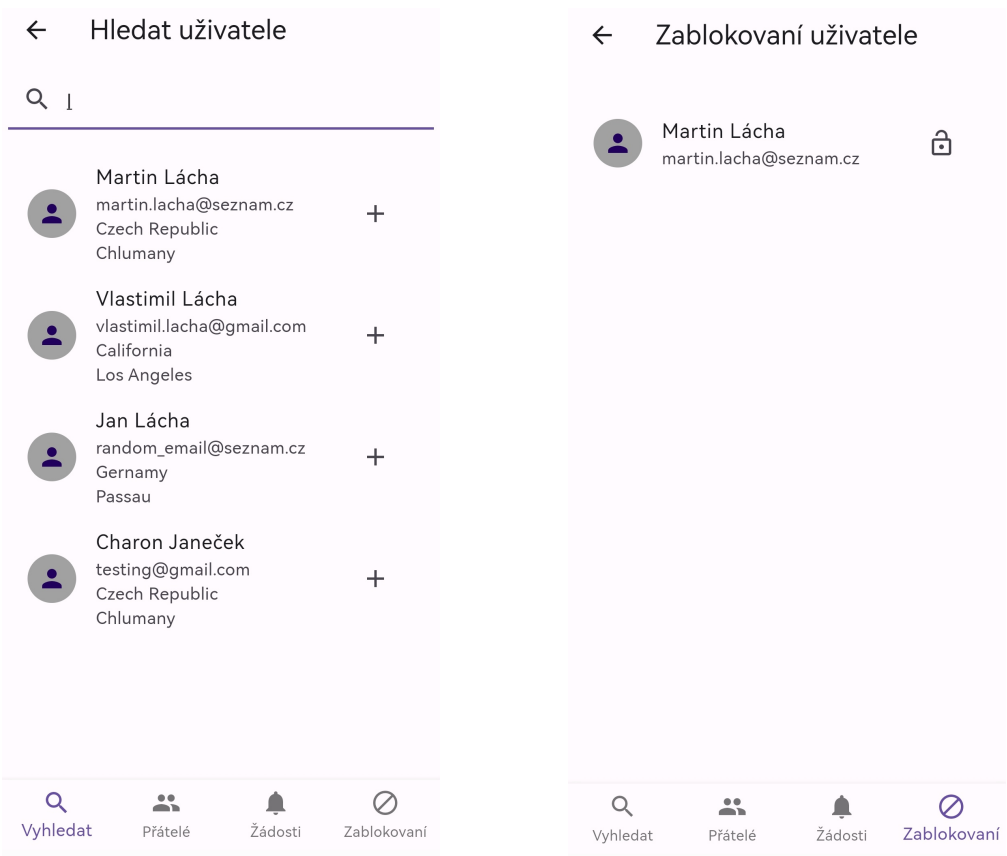
(a) (b)

Obrázek C.5: Seznam novinek a formulář pro přidání článku



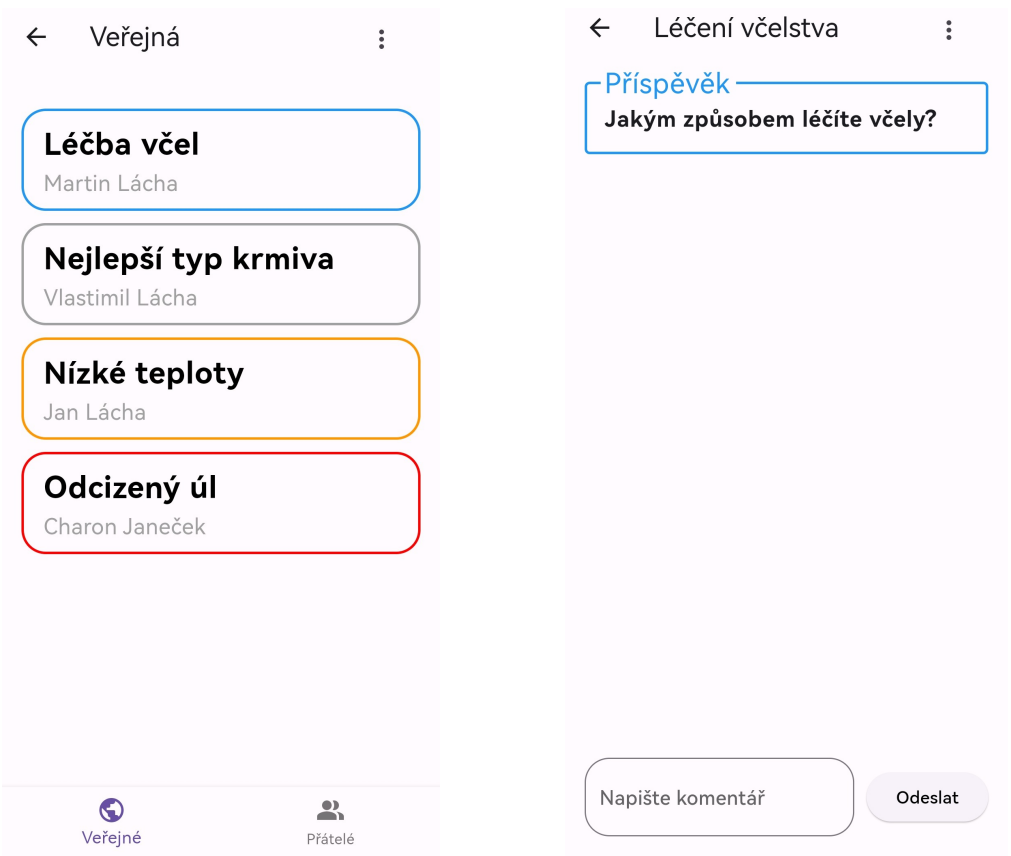
(a) (b)

Obrázek C.6: Seznam přátel a akce s uživatelem



(a) (b)

Obrázek C.7: Vyhledání uživatele a blokovaní uživatelé

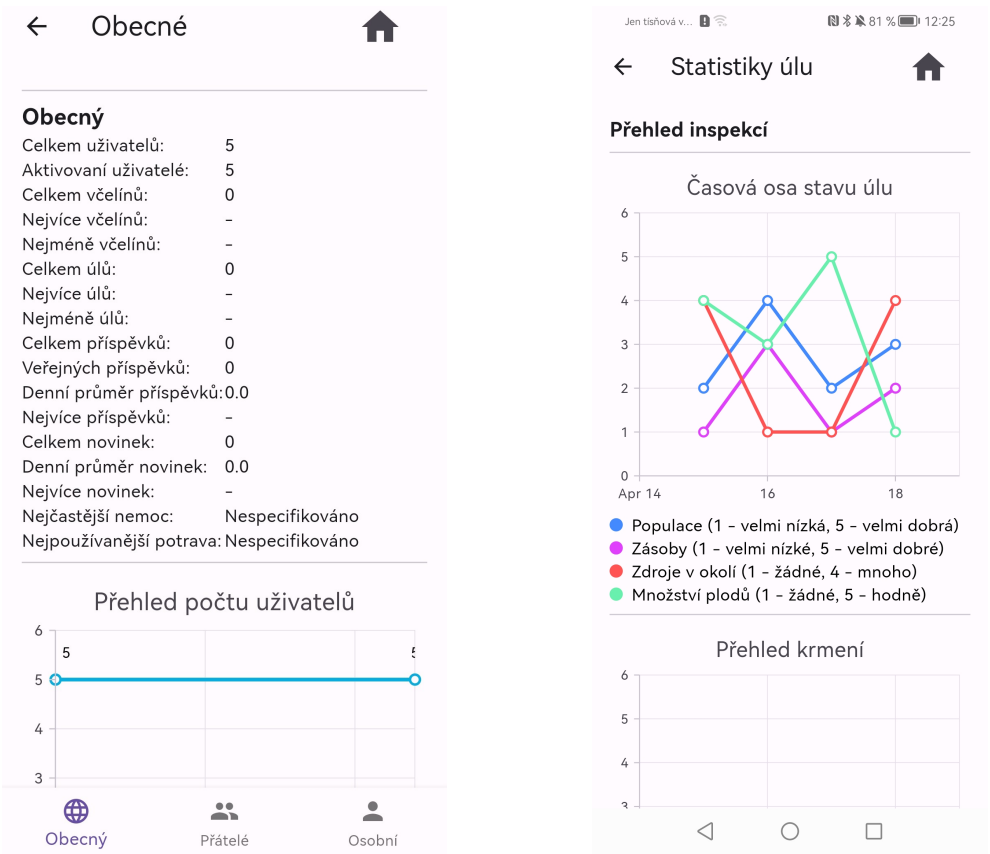


(a) (b)

Obrázek C.8: Veřejné příspěvky a detail příspěvku

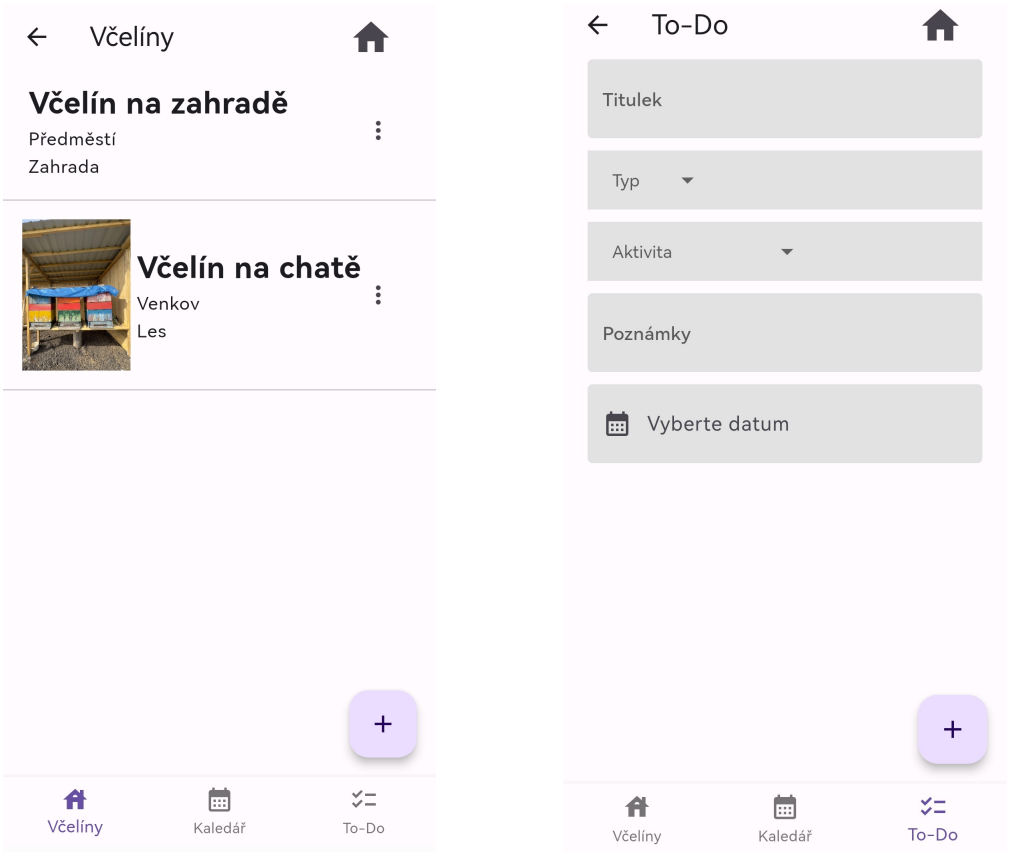
(a) (b)

Obrázek C.9: Vzdělávací lekce a obsah lekce *Druhy*



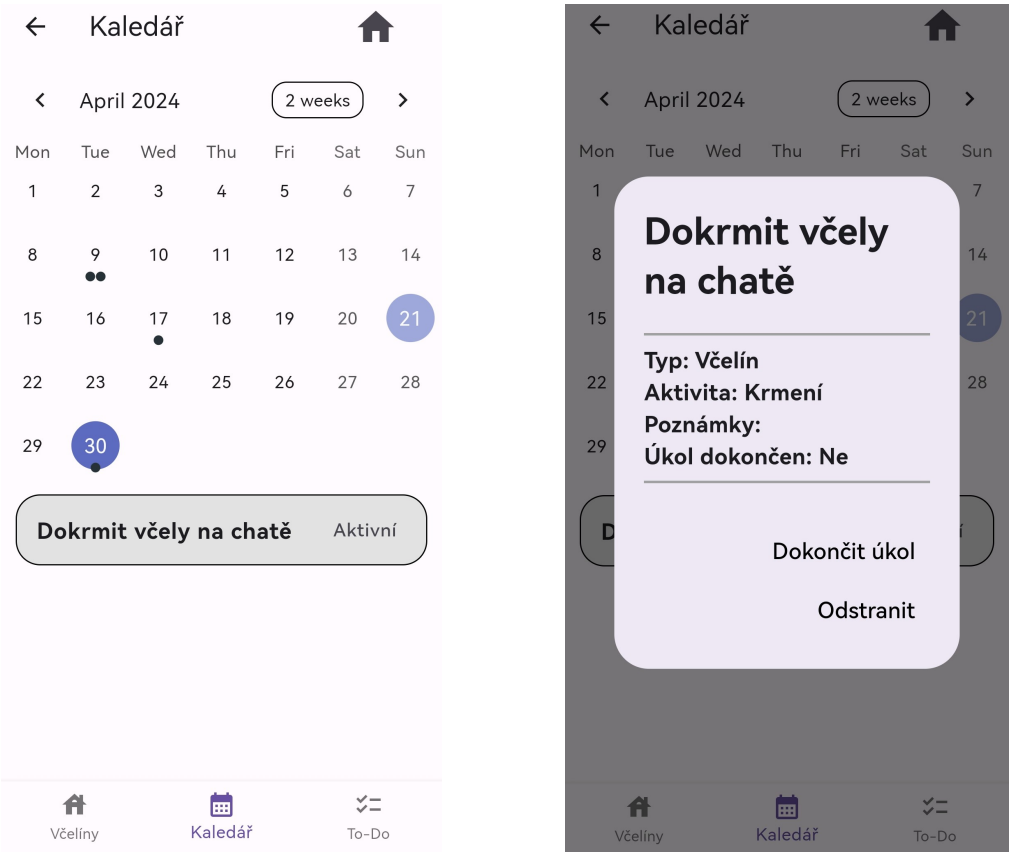
(a) (b)

Obrázek C.10: Obecné statistiky a statistiky včelího úlu



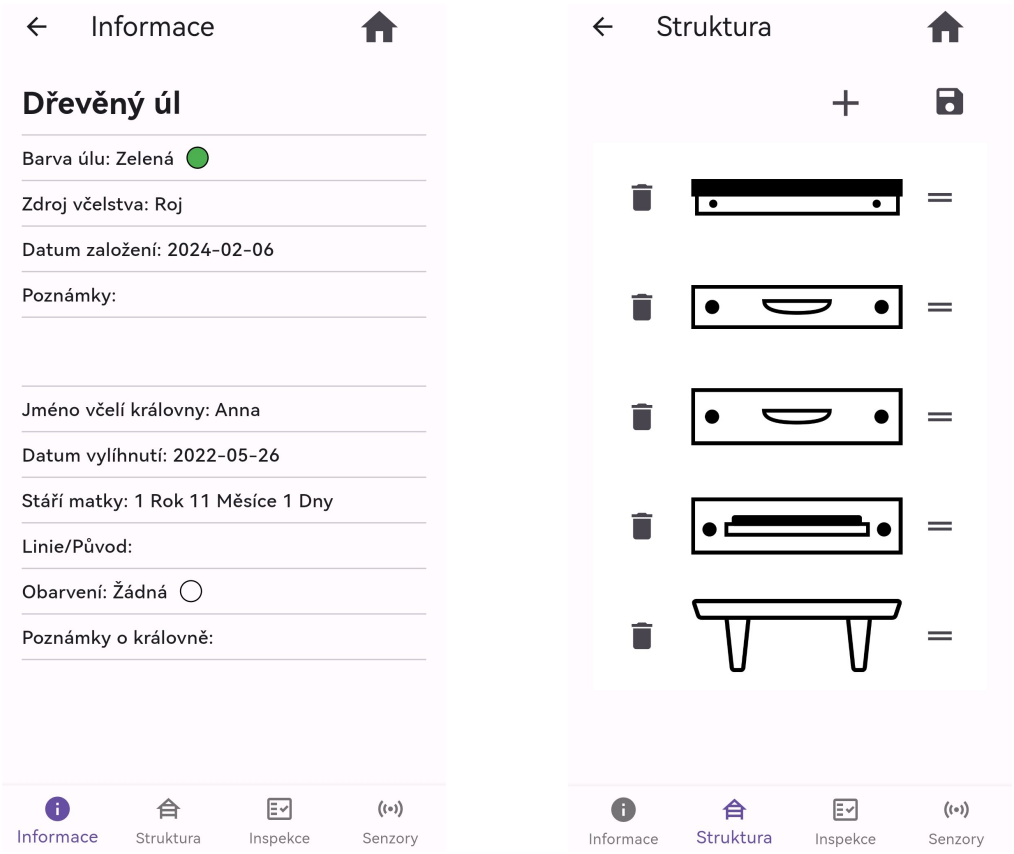
(a) (b)

Obrázek C.11: Seznam včelínů a formulář události



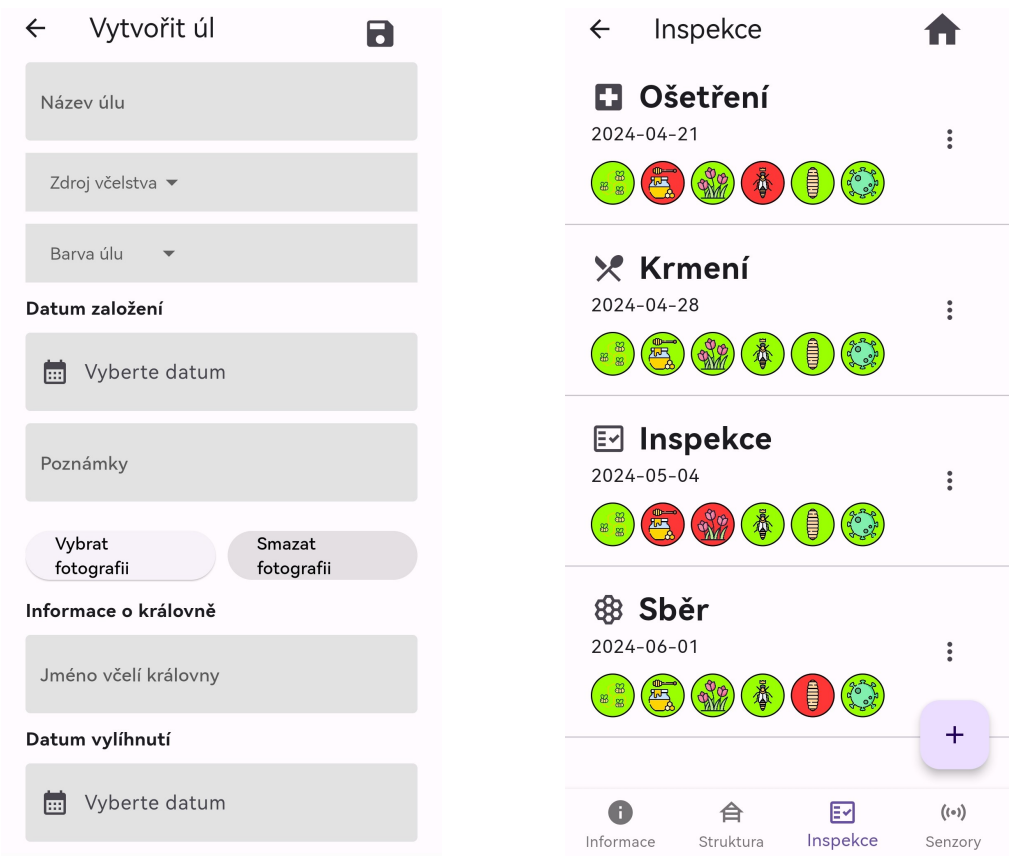
(a) (a)

Obrázek C.12: Kalendář událostí a detail události



(a) (b)

Obrázek C.13: Informace o úlu a struktura úlu

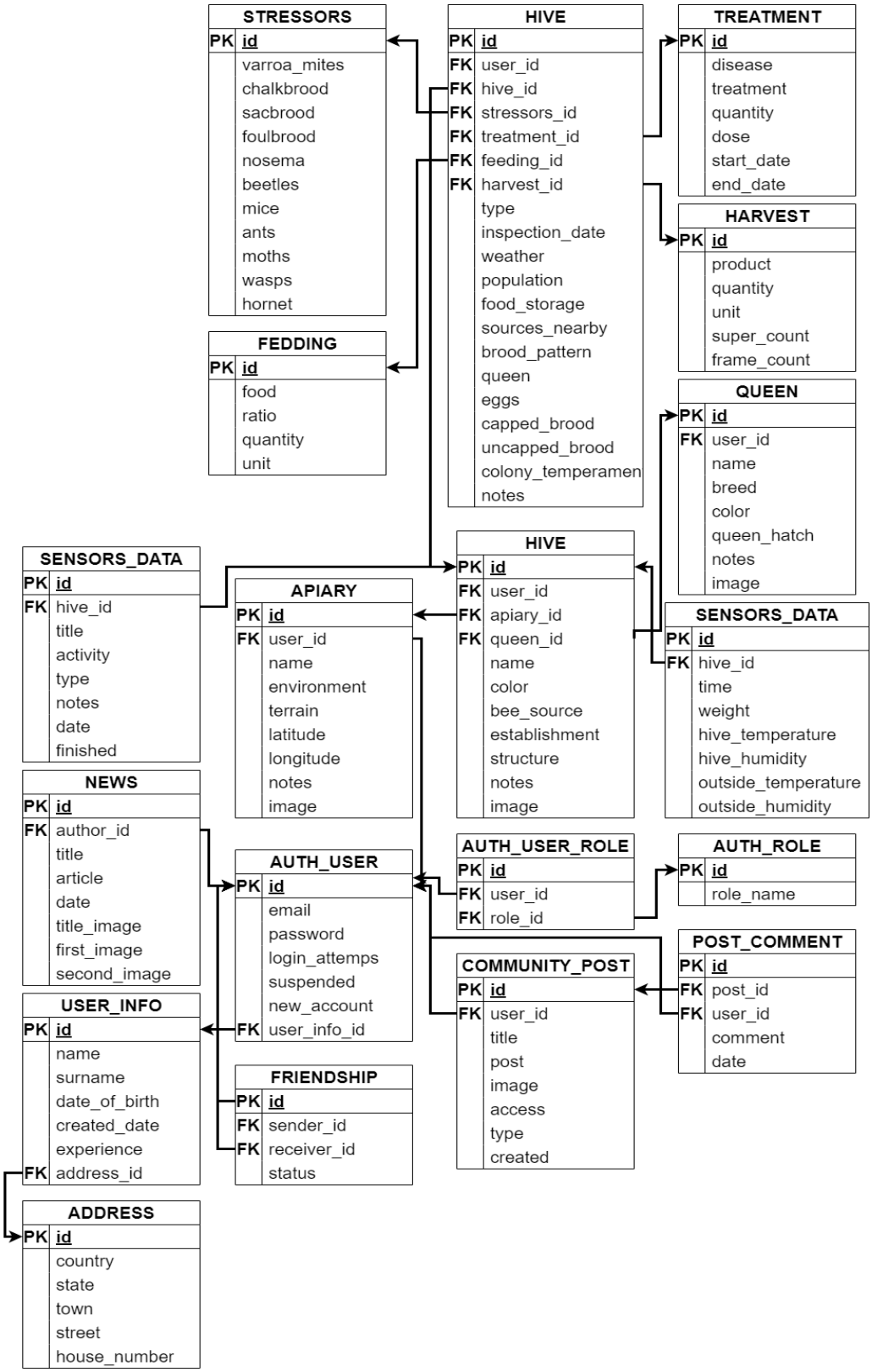


(b) (b)

Obrázek C.14: Formulář úlu a seznam inspekcí

Datový model

D



Obrázek D.1: Datový model

Struktura kódu

E.1 Komunikační server

BeeCommunityServer

- `.github` ..... Konfigurace procesů GitHub Actions (CI, CD)
- `.mvn` ..... Konfigurační soubory Apache Maven
- `db_backup` ..... Složka se zálohami databáze
- `src` ..... Zdrojové soubory projektu
- `.gitignore` ..... Seznam ignorovaných souborů pro Git
- `Dockerfile` ..... Konfigurační soubor pro sestavení projektu
- `README.md` ..... Dokument s popisem projektu
- `codecov.yaml` ..... Konfigurační soubor nástroje CodeCov
- `compose.yaml` ..... Definice kontejnerů při nasazení
- `mvnw` ..... Skript pro spuštění nástroje Maven (Linux)
- `mvnw.cmd` ..... Skript pro spuštění nástroje Maven (Windows)
- `pom.xml` ..... Konfigurace Maven (metadata, závislosti, pluginy)

E.2 Mobilní aplikace

BeeCommunityClient

- `android` ..... Složka pro zdrojové soubory Android aplikace
- `assets` ..... Složka pro statické zdroje (např. obrázky, fonty)
- `ios` ..... Složka pro zdrojové soubory iOS aplikace
- `lib` ..... Složka pro zdrojový kód aplikace
- `.gitignore` ..... Seznam ignorovaných souborů pro Git
- `.metadata` ..... Metadata projektu
- `README.md` ..... Dokumentace projektu
- `analysis_options.yaml` ..... Konfigurace analýzy kódu
- `pubspec.lock` ..... Soubor s uzamčenými verzemi balíčků
- `pubspec.yaml` ..... Konfigurace Dartu (závislosti, obrázky)

E.3 Senzory

- `BeeCommunitySensors`
  - `libraries` ..... Externí zdrojové kódy (knihovny apod.)
  - `BeeCommunitySensors.ino` ..... Zdrojový kód pro Arduino Sketch
  - `README.md` ..... Dokument s popisem projektu

Bibliografie

1. TEW, James E. *Nepostradatelný rádce včelaře*. Great Britain: REBO International, 2015. ISBN 978-80-255-0905-0.
2. TENNANT, Emma; CHADWICK, Fergus. *The Bee Book: Discover the Wonder of Bees and How to Protect Them for Generations to Come*. DK, 2016. ISBN 978-1465443830.
3. *Včela medonosná*. Leoš Dvorský, 2014. Dostupné také z: `https://www.souhorky.cz/ns-vcela-medonosna`.
4. SOCHA, Piotr. *Včely*. Slovart, 2017. ISBN 978-80-7529-333-6.
5. RAJAGOPALAN, Kirti et al. Warmer autumns and winters could reduce honey bee overwintering survival with potential risks for pollination services. *Scientific Reports*. 2024, roč. 14, č. 1. ISSN 2045-2322. Dostupné z DOI: `10.1038/s41598-024-55327-8`.
6. *Porovnání sršně mandarínské, asijské a obecné*. 2023. Dostupné také z: `https://www.pasti.cz/blog/porovnani-srsne-mandarinske-a-srsne-obecne/`.
7. *Invazní sršeň asijská v České republice*. Česká inspekce životního prostředí, 2023. Dostupné také z: `https://www.cizp.cz/aktuality/invazni-srsen-asijska-v-ceske-republice`.
8. *Včely bez nás mohou žít, my bez nic ne*. Kristián Šujan, 2019. Dostupné také z: `https://www.universitas.cz/osobnosti/3636-vcely-bez-nas-mohou-zit-my-bez-nich-ne`.
9. *Moderní technologie pomáhají českým včelařům*. Barbora Venclová, 2017. Dostupné také z: `https://uroda.cz/moderni-technologie-pomahaji-ceskym-vcelarum/`.
10. VMWARE. *Spring Boot*. VMware Tanzu, 2023. Dostupné také z: `https://spring.io/projects/spring-boot`. Documentation.
11. PRO, Programiz. *Kotlin vs Java: Which is better in 2023?* Programiz Pro, 2023. Dostupné také z: `Kotlin%20vs%20Java:%20Which%20is%20better%20in%202023?`. Kotlin and Java Comparsion.
12. *Catching Issues in the IDE with SonarLint*. SonarCloud, 2024. Dostupné také z: `https://docs.sonarsource.com/sonarcloud/improving/sonarlint/`.

13. *Features overview*. JetBrains, 2024. Dostupné také z: https://www.jetbrains.com/idea/features/#features.
14. *Visual Studio Code*. Microsof, 2024. Dostupné také z: https://code.visualstudio.com/.
15. *Top 10 Mobile App Development Frameworks in 2024*. LinkedIn, 2024. Dostupné také z: https://www.linkedin.com/pulse/top-10-mobile-app-development-frameworks-2024-mor-software-jsc-9tlkc.
16. RANDALL, Scott. *Mastering React Native: From Beginner to Expert.* Independently published, 2023. ISBN 979-8861585194.
17. SOLE, Alessandro Del. *Xamarin with Visual Studio: Launch your mobile development career by creating Android and iOS applications using .NET and C (English Edition)*. BPB Publications, 2023. ISBN 978-9355511874.
18. *Why use Flutter: Pros and Cons of Flutter app development*. Daniela Montaño, 2024. Dostupné také z: https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/.
19. CHACON, Scott; STRAUB, Ben. *Pro Git, Second Edition*. Boston, MA, USA: Apress, 2014. ISBN 978-1484200773.
20. *What is Git?* Atlassian Corporation, 2024. Dostupné také z: https://www.atlassian.com/git/tutorials/what-is-git.
21. *Docker overview*. Docker, Inc., 2024. Dostupné také z: https://docs.docker.com/build/cloud/.
22. ÖGGL, Bernd; KOFLER, Michael. *Docker. Practical Guide for Developers and Devops Teams*. Boston, MA, USA: Rheinwerk Verlag GmbH, 2023. ISBN 978-1493223831.
23. ROBERTS, David C. *Introduction to Databases: A Focus on Practical Solutions Kindle Edition*. Maryland, USA: Jada Press; Third Edition, 2024. ISBN 979-8985508512.
24. *About*. PostgreSQL, 2024. Dostupné také z: https://www.postgresql.org/about/.
25. PANT, Piyush et al. Authentication and authorization in modern web apps for data security using Nodejs and role of dark web. *Procedia Computer Science*. 2022, roč. 215, s. 781–790.
26. TERRA, John. *What is Client-Server Architecture? Everything You Should Know*. SimpliLearn, 2023. Dostupné také z: https://www.simplilearn.com/what-is-client-server-architecture-article.

27. NIRAV, Kanani. *Top 6 Most Popular API Architecture Styles You Need to Know (with Pros, Cons, and Use Cases)*. DEV.to, 2023. Dostupné také z: `https://dev.to/kanani_nirav/top-6-most-popular-api-architecture-styles-you-need-to-know-with-pros-cons-and-use-cases-564j`.

28. MANN, Howie. *REST APIs Explained - 4 Components*. Howie Mann, 2023. Dostupné také z: `https://mannhowie.com/rest-api`.

29. GUPTA, Lokesh. *How to Design a REST API*. Lokesh Gupta, 2023. Dostupné také z: `https://restfulapi.net/rest-api-design-tutorial-with-example/`. How to Design a REST API.

30. GUPTA, Lokesh. *HTTP Status Codes*. Lokesh Gupta, 2023. Dostupné také z: `https://restfulapi.net/http-status-codes/`. HTTP Status Codes.

31. *Introduction to JSON Web Tokens*. Auth0, 2024. Dostupné také z: `https://jwt.io/introduction`.

32. *Mobile Operating System Market Share Worldwide*. GlobalStats, 2024. Dostupné také z: `https://gs.statcounter.com/os-market-share/mobile/worldwide`.

33. *Build and release an iOS app*. Flutter, 2024. Dostupné také z: `https://docs.flutter.dev/deployment/ios`.

34. ZACEPINS, Aleksejs et al. Beekeeping in the future—Smart apiary management. In: *2016 17th International Carpathian Control Conference (ICCC)*. IEEE, 2016, s. 808–812.

35. STARÁ, Šárka. *Na střeše Fakulty strojní vznikla chytrá včelí laboratoř*. Dostupné také z: `https://info.zcu.cz/clanek.jsp?id=3445`.

36. KRAUZOVÁ, Eva. *Digitální včelí laboratoř na střeše Fakulty strojní je možné sledovat online*. Dostupné také z: `https://info.zcu.cz/clanek.jsp?id=3445`.

37. CECCHI, Stefania; SPINSANTE, Susanna; TERENZI, Alessandro; ORCIONI, Simone. A smart sensor-based measurement system for advanced bee hive monitoring. *Sensors*. 2020, roč. 20, č. 9, s. 2726.

38. SZCZUREK, Andrzej; MACIEJEWSKA, Monika; BATOG, Piotr. Monitoring system enhancing the potential of urban beekeeping. *Applied Sciences*. 2023, roč. 13, č. 1, s. 597.

39. SZCZUREK, Andrzej; MACIEJEWSKA, Monika. Beehive air sampling and sensing device operation in apicultural applications—methodological and technical aspects. *Sensors*. 2021, roč. 21, č. 12, s. 4019.

40. FERRARI, Sara; SILVA, Mitchell; GUARINO, Marcella; BERCKMANS, Daniel. Monitoring of swarming sounds in bee hives for early detection of the swarming period. *Computers and electronics in agriculture*. 2008, roč. 64, č. 1, s. 72–77.

41. DANIELI, Pier Paolo; ADDEO, Nicola Francesco; LAZZARI, Filippo; MANGANELLO, Federico; BOVERA, Fulvia. Precision Beekeeping Systems: State of the Art, Pros and Cons, and Their Application as Tools for Advancing the Beekeeping Sector. *Animals*. 2023, roč. 14, č. 1, s. 70.
42. OPENWEATHER. *Weather API*. OpenWeather, 2024. Dostupné také z: `https://openweathermap.org/api`. Weather API.
43. ANDROID. *Request location permissions*. Apple, 2024. Dostupné také z: `https://developer.android.com/develop/sensors-and-location/location/permissions`. Android location permission.
44. APPLE. *Requesting authorization to use location services*. Apple, 2024. Dostupné také z: `https://developer.apple.com/documentation/corelocation/requesting_authorization_to_use_location_services`. iOS location permission.
45. *Image Picker plugin for Flutter*. Flutter Dev, 2024. Dostupné také z: `https://pub.dev/packages/image_picker`.
46. ARDUINO. *Getting started arduino*. Dostupné také z: `https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/`.
47. SENTRY. *Code Coverage: More Than a Metric.* Dostupné také z: `https://about.codecov.io/`.

Seznam obrázků

Seznam tabulek

Seznam výpisů