FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	iOS aplikace pro správu rostlin
Student:	Bc. Lukáš Litvan
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout, implementovat a otestovat mobilní aplikaci pro operační systém iOS, která bude sloužit jako nástroj pro správu pěstovaných rostlin. Součástí práce je také vhodně prozkoumat možnosti, které by tento typ aplikace měl podporovat (např. zalití, sdílení atp.).

Postupujte v těchto krocích:
1. Analyzujte existujících aplikace, které se věnují obdobnému tématu
2. Na základě analýzy navrhněte vhodnou architekturu aplikace i její funkcionality s ohledem na cílovou platformu iOS
3. Dle návrhu implementujte výsledné řešení
4. Řešení řádně otestujte
5. Proveďte všechny potřebné kroky k reálnému zpřístupnění aplikace pro budoucí uživatele
6. Zhodnoťte dosažené výsledky, navrhněte možná budoucí vylepšení

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 23. června 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

iOS aplikace pro správu rostlin

Bc. Lukáš Litvan

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

3. ledna 2023

Poděkování

Především bych chtěl poděkovat Ing. Jiřímu Hunkovi za jeho cenné rady, připomínky a trpělivost při vedení mé diplomové práce. Dále také své rodině, přítelkyni a kamarádům za podporu, a to nejen při psaní této práce, ale i v průběhu celého studia. Děkuji také všem účastníkům uživatelského testování.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelum. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 3. ledna 2023 . . . . . .

Odkaz na tuto práci

Litvan, Lukáš. *iOS aplikace pro správu rostlin*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Tato diplomová práce se zabývá vývojem mobilní aplikace pro iOS, která slouží jako správce pěstovaných rostlin. V aplikaci si uživatelé mohou přidávat rostliny, které pěstují. Tyto rostliny lze organizovat do týmů a míst. Následně k těmto rostlinám mohou přidávat události, fotografie a nastavovat připomínky. Jedinečnou funkcionalitou aplikace, kterou se odlišuje od ostatních řešení, je sdílení rostlin mezi více lidmi. Práce popisuje analýzu, návrh, implementaci serverové a klientské části aplikace. Dále se věnuje samotnému nasazení a vydání aplikace. Nakonec rozebírá uživatelské testování, kterému byla aplikace podrobena.

Klíčová slova pěstování, rostliny, připomínky, mobilní aplikace, iOS

Abstract

This thesis deals with the development of a mobile application for iOS that serves as a plant manager. In the app, users can add the plants they grow. These plants can be organized into teams and places. They can then add events, photos and set reminders to these plants. A unique functionality of the app that sets it apart from other solutions is the sharing of plants between multiple people. This thesis describes the analysis, design, implementation of the server and client side of the application. It also covers the application deployment and release. Finally, it discusses the user testing that the application was subjected to.

Keywords growing, plants, reminders, mobile application, iOS

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódů

Úvod

Téměř každý si jistě zkusil pěstovat vlastní rostlinu, a to buď na venku na zahrádce, či doma v květináči. Pro mnohé je pěstování rostlin koníček, na který nedají dopustit. Samotná péče o rostliny není v mnoha případech těžká věc, spíše jde o provádění jednoduchých operací, především o zalévání, hnojení, občasné přesazení nebo udržování vlhkosti rozprašovačem. Vše ovšem závisí na konkrétním typu rostliny. Někteří lidé (zejména začátečníci nebo lidé, co nemají tolik času) občas zapomenou rostlinu zalít nebo ji přelijí, a to se ji může stát osudným.

Cílem této práce je analýza, návrh, implementace mobilní aplikace pro operační systém iOS, která by měla sloužit jako pomocník, či deník pěstitele rostlin/květin. Aplikace umožní uživatelům udržovat seznam rostlin, o které pečují, nastavit si notifikace k upozornění naplánovaných akcí. Hlavní funkcí aplikace by měla být možnost spolupráce uživatelů, což znamená, že uživatelé budou moci vytvářet týmy, ve kterých budou mít společné rostliny. Každý potom uvidí, kdo provedl kterou akci a budou moci dostávat různé notifikace ohledně provedených akcí či různé připomínky.

Motivací k vytvoření této aplikace je zkušenost s vlastním pěstováním pokojových rostlin v bytě, ve kterém bydlím se svojí přítelkyní. Svých pár rostlin zaléváme často sporadicky a občas dojde k tomu, že někdo z nás zapomene květinu zalít nebo se stane, že je květina zalita moc. Vhodným řešením by tedy bylo navržené řešení, které by umožnilo organizaci a připomínání akcí (formou notifikací).

Tato práce začíná kapitolou, která se věnuje analýze (kvantitativní a kvalitativní výzkum, analýza existujících řešení) a specifikaci požadavků na aplikaci. Ta je následovaná kapitolou, která popisuje metody softwarového vývoje. Na základě požadavků je poté v další kapitole proveden návrh aplikace, a to serverové i klientské části (iOS aplikace). Následuje kapitola věnující se samotné implementaci obou částí aplikace. Výsledná implementace je v poslední kapitole otestována a jsou navržena řešení k odhaleným problémům.

Kapitola 1

Analýza

V této kapitole jsou nejdříve shrnuty poznatky ohledně kvantitativního a kvalitativního výzkumu, který se týká danému tématu a poté jsou analyzována existující řešení (mobilní aplikace). Aplikace jsou ohodnoceny na základě Nielsenovy heuristiky a zároveň jsou shrnuty jejich silné a slabé stránky. S pomocí těchto poznatků jsou poté navrženy funkční a také nefunkční požadavky na výslednou aplikaci. Požadavky jsou kategorizovány podle modelu FURPS.

1.1 Kvantitativní výzkum

Kvantitativní výzkum spočívá ve sběru a analýze dat za účelem vysvětlení jevů pomocí čísel. Informace ze vzorku se používají k zobecnění nebo k předpovědím o populaci, v tomto případě o cílové skupině. Nejčastějšími metodami kvantitativního výzkumu jsou především dotazníky či různé ankety.

Zaměřil jsem se zejména na již provedené rešerše, dotazníky a jejich výsledky, které se týkají daného tématu. Vyhledával jsem tedy pomocí internetového vyhledávače Google termíny jako „*survey about plant care*“ či „*plant care statistics*“. Články [1, 2], které se zabývají rozborem různých dotazníků a statistik ohledně pokojových rostlin a zahradničení, nabízejí sadu relevantních a často i zajímavých informací. Výsledky říkají, že 7 z 10 mileniálů (označení pro lidi narozené zhruba od roku 1981 do roku 1995) se identifikují jako „*plant parent*“, tedy rodič rostlin. Takovýto rodič v průměru také zničil sedm pokojových rostlin, které si přinesl domů. S tím souvisí obavy z různých aspektů samotného pěstování. Tabulka 1.1 právě tyto největší starosti dotázaných ukazuje. Nejvíce mají dotázaní mileniálové starosti s tím, zda rostliny mají dostatek světla a vody. Zajímavým údajem je také, že zhruba 66 % amerických domácností vlastní alespoň 1 pokojovou rostlinu a že průměrná domácnost utratí za zahradnické potřeby 608,54 amerických dolarů ročně. Dále byly v rámci výzkumu vyzdvihnuty nejrůznější pro člověka pozitivní benefity, pokud se nachází v prostředí s rostlinami či se o ně přímo starají. Vyzdvihl bych ještě

Mileniálové	Starosti s péčí o rostliny
60 %	Zajištění dostatečného slunečního světla
56 %	Zajištění dostatečného množství vody
48 %	Udržení rostlin při životě
37 %	Stresování rostlin při přemisťování
21 %	Hledání hlídání v době, kdy nejsou doma

Tabulka 1.1: Tabulka nejčastějších starostí mileniálů při péči o rostliny

fakt, že trend v pěstování rostlin roste. Především došlo k nárůstu v poslední době v souvislosti s pandemií Covid-19. Tato pandemie vytvořila 18,3 milionu nových zahrádkářů v USA. Zajímavou a trochu úsměvnou informací je rostoucí zájem o pěstování konopí, za což může postupná legalizace v několika amerických státech.

1.2 Kvalitativní výzkum

Kvalitativní výzkum využívá nestatistické metody k získání informací o různých skutečnostech. Oproti kvantitativnímu výzkumu nepracuje s čísly a obvykle se provádí v přirozeném prostředí. K získání hlubšího porozumění se využívají rozhovory (s lidmi) nebo pozorování dané skutečnosti.

Provedl jsem tedy rozhovor (cca 30 minut) s třemi osobami z mého okolí, o kterých jsem si myslel, že by mohli mít s pěstováním rostlin zkušenost. Jednalo se o jednoho muže ve věku 24 let a dvě ženy ve věku 24 a 26 let. Rozhovory probíhaly dosti volně a v přátelském duchu. Snažil jsem se především zjistit jejich zkušenost s pěstováním, o kolik, o jaké rostliny a jak se o ně starají. Ptal jsem se ohledně využití aplikací, které slouží jako pomocník s pěstováním rostlin a co by daná aplikace měla umět, aby ji chtěli sami využívat.

Protože se jednalo o vcelku mladé osoby, jsou zvyklí žít poměrně časově náročnými životy, a tak se sami starají pouze o zhruba 3-4 rostliny, z čehož všechny jsou rostliny pokojové a poměrně nenáročné. Starání se o jejich rostliny se skládá v zásadě z pravidelného a vhodného zalévání, popřípadě občasného stříhání a zbavování se odumřelých částí. Jednou za čas (velmi ojediněle) rostlinu přesadí. Bylo zjištěno, že sami žádnou podpůrnou aplikaci nepoužívají. Důvodem je nevědomost a takovémto druhu aplikace a také fakt, že nevnímají pěstování rostlin jako jednu z podstatných priorit svého života. Podle jejich názoru by aplikace měla být velmi jednoduchá a hlavně by neměla člověka zdržovat. Největší přínos by podle nich byl zejména v upomínkách na zalévání a možnosti si prohlížet různé rostliny a jak se o ně vhodně starat. Na otázku, jak by se jim líbila funkcionalita ohledně týmové spolupráce, odpověděli, že se jedná celkem o zajímavý koncept.

1.3 Analýza existujících řešení

Aplikace, které řeší daný problém, existuje několik, zde je vybráno 5 (dle mého názoru) nejzajímavějších pro porovnání. Existující aplikace byly hledány v obchodu s mobilními aplikacemi pro iOS – *App Store*, pomocí těchto klíčových slov *plant companion*, *plant diary*, *plant reminders* a dalších. Dále také pomocí webového vyhledávače Google, kde byly použity výrazy typu „*Best plant care apps*“. Každá z vybraných aplikací je popsána, jsou shrnuty její výhody a nevýhody a také je ohodnocena její použitelnost (z hlediska uživatelského rozhraní) na základě Nielsenovy heuristiky (ta je popsána níže v 1.3.1).

1.3.1 Nielsenova heuristika [3, 4]

Nielsenova heuristika patří mezi nejpoužívanější heuristiky použitelnosti pro návrh uživatelského rozhraní. Nielsen je vypracoval spolu s Rolfem Molichem v roce 1990 a konečnou sadu heuristik, které se používají dodnes, vydal v roce 1994. Tyto heuristiky vytvořili na základě pozorování a odborných znalostí získaných během své dlouholeté praxe. Samotné Nielsenovy heuristiky jsou sadou deseti obecných zásad (neurčují konkrétní pravidla použitelnosti), jejichž dodržení by mělo vést k vytvoření přístupnějších, uživatelsky přívětivějších a intuitivnějších produktů. Těmito zásadami jsou:

1. **Viditelnost stavu systému** Systém by měl uživatele vždy informovat o tom, co se děje, a to prostřednictvím vhodné zpětné vazby v dostatečně krátké době.

2. **Shoda mezi systémem a reálným světem** Systém by měl být uživateli co nejvíce srozumitelný (používat slova, fráze a pojmy, které uživatelé znají z reálného světa).

3. **Uživatelská kontrola a svoboda** Uživatelé často dělají chyby a potřebují mít možnost se jednoduše vrátit nebo napravit danou chybu.

4. **Konzistence a standardy** Uživatelé by neměli přemýšlet, zda různá slova, situace nebo akce znamenají totéž. Systém by měl dodržovat konvence platformy a stejné prvky by měly na různých místech dělat to stejné.

5. **Prevence chyb** Systém by měl předvídat chyby a ideálně jim předcházet (například pomocí potvrzovacích dialogů před důležitými akcemi).

6. **Rozpoznání místo vzpomínání** Zviditelnění objektů, akcí a možností, které uživatel může provést. U složitějších akcí je vhodný krátký textový popis. Uživatel by si neměl pamatovat informace z jedné části dialogu do druhé.

7. **Flexibilita a efektivita** Systém by měl pro pokročilé uživatele umožnit provádět akce rychleji (například klávesové zkratky). Tyto funkce by ale měly být skryty pro začínající či občasné uživatele.

8. **Estetický a minimalistický design** Uživatelské rozhraní by mělo zobrazovat pouze relevantní informace, akce či prvky. Nadbytečné informace by mohly způsobovat uživateli zmatení a odvedení pozornosti.

9. **Diagnostika, rozpoznání a oprava chyb** Chybová hlášení by měla být srozumitelná pro uživatele, přesně označovat problém a navrhovat řešení.

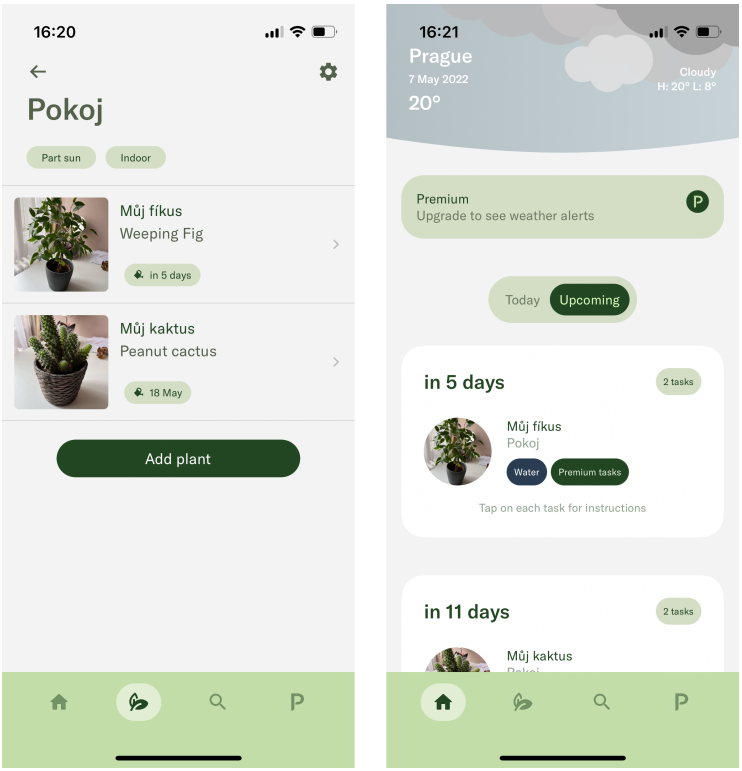
10. **Nápověda a dokumentace** Ideálně by měl být systém intuitivní a použitelný bez nápovědy. To ale ne vždy jde, a tak je vhodné poskytnout nápovědu a dokumentaci (jasnou a snadno vyhledatelnou).

1.3.2 Planta

Aplikace Planta 1.1 od švédských vývojářů (Stromming AB) umožňuje uživateli vybrat si rostliny z celkem obsáhlé databáze (pokojových, ale i zahradních). U každé rostliny uvádí užitečné informace o potřebné péči, například kolik světla, nebo jakou vlhkost či teplotu rostlina ideálně potřebuje a další. Dále také ukazuje, zda se rostlina hodí uživateli do dané místnosti, a to na základě informací, které po registraci uživatel vyplní. Jde například o zkušenosti uživatele ohledně pěstování a jak moc se chtějí pěstování věnovat. Uživatel si může vytvářet různé místnosti, do kterých pak umisťuje vybrané rostliny.

V neplacené verzi umožňuje pouze nastavení a sledování zalévání jednotlivých rostlin. Samotný plán zalévání je sestaven na základě dané rostliny a informací o místnosti a uživateli. Na hlavní (domovské) obrazovce potom může sledovat, které úkoly/akce má dnes udělat a také rozpis akcí na další dny. Dále lze ke svým rostlinám přidávat fotografie a sledovat tak postupný vývoj v čase. V placené verzi (měsíční poplatek v řádu několika eur, který závisí na vybraném plánu) poté nabízí více funkcí, mezi které patří rozpoznávání rostlin, doporučování rostlin, další akce a plány (hnojení, přesazování), rozpoznávání nemocí nebo měření světla pomocí kamery. Tyto placené funkce nebyly však otestovány.

Aplikace je zpracována vizuálně pěkně a umožňuje jistě uživatelům zjednodušení či organizaci správy o své rostliny. Co však aplikace neumožňuje, je starání se o rostliny společně, respektive v rámci týmu. Nelze mít tedy situaci, kde by jedna rostlina byla sdílena mezi více uživateli, kteří by viděli např. historii akcí, které byly provedeny. Ostatně v jedné z recenzi v obchodě *App Store* uživatel sděluje, že by nebyl špatný nápad mít tuto možnosti sdílení. Odpovědí vývojářů na tuto recenzi je poskytnutí uživatelského účtu (jména a hesla) druhému člověku, což ovšem není úplně ideální možnost a navíc nevyužitý potenciál.

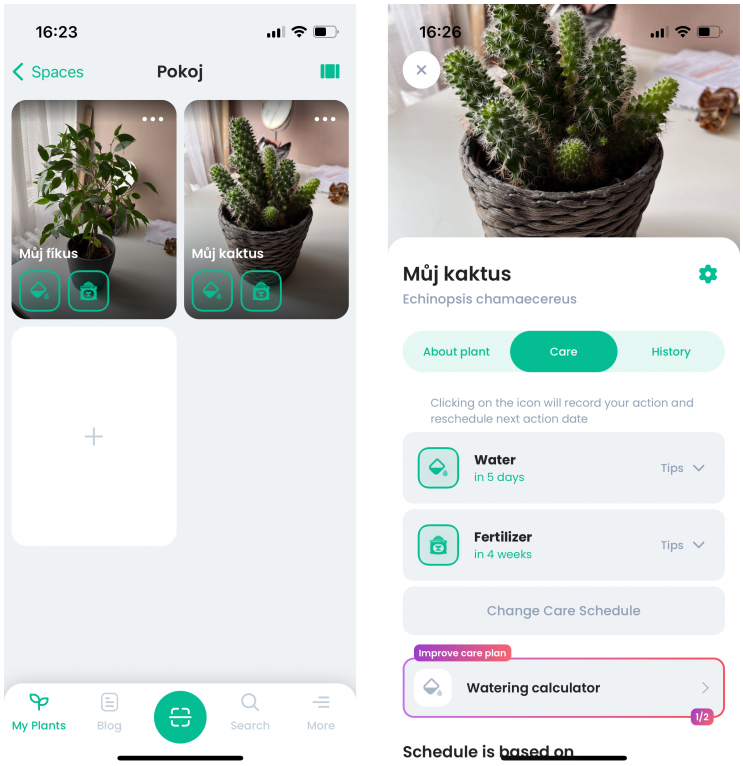


Obrázek 1.1: Ukázka aplikace Planta

1.3.3 PlantIn

Druhou aplikací je aplikace PlantIn 1.2. Obdobně jako aplikace Planta nabízí uživatelům možnost vytvářet místa/místnosti, do kterých pak můžou ukládat z dostatečné databáze rostlin. Na detailu rostliny je několik obrázků dané rostliny, pár klíčových informací (toxicita, vhodná teplota, světlo a vlhkost) a následně další užitečné informace v textové formě. Po přidání rostliny je sestaven plán péče, obdobně jako u aplikace Planta. Aplikace zohledňuje různé informace (světlo, teplotu, roční období, lokaci) k modifikace a vytvoření prvotního plánu. Následně si uživatel může manuálně upravit intervaly akcí a povolit/nastavit notifikace. Své rostliny pak lze různě pojmenovávat a změnit jejich výchozí obrázek.

Dále aplikace obsahuje obrazovku s názvem blog, která nabízí různé články ohledně pěstování, návody nebo různé zajímavosti ze světa rostlin. Aplikace nabízí rozpoznávání rostlin pomocí fotoaparátu. Po úspěšném rozpoznání rostliny se uživatel může dostat na samotný detail rostliny, kde najde již zmíněné informace. Dále také nabízí rozpoznávání nemocí či odbornou pomoc od zkušeného botanisty. Aplikace poskytuje možnost zakoupení prémiového plánu, který nabízí jen pár vylepšení (lepší/chytřejší plán, různé vzhledy/barvy apli-



Obrázek 1.2: Ukázka aplikace PlantIn

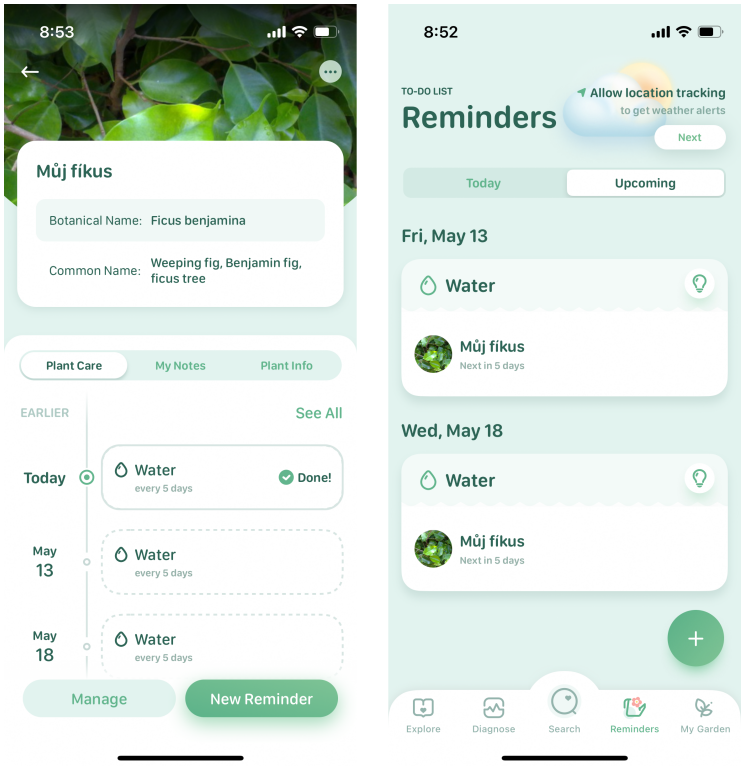
kace) od neplacené verze.

Zároveň jako u aplikace Planta je zde absence spolupráce nebo sdílení rostlin mezi více uživateli. Jinak je aplikace poměrně jednoduchá a hezky vzhledově zpracovaná.

1.3.4 Blossom

Obdobně jako předchozí aplikace umožňuje aplikace Blossom 1.3 vyhledávat z poměrně obsáhlé databáze rostlin, dále také rozpoznávat rostliny pomocí fotoaparátu. Detail rostliny je vzhledově hezký, na vrchu jsou přehledně zobrazeny nejdůležitější informace, následuje galerie obrázků dané rostliny a poté rozbalovací boxy s textovými informacemi, které se vztahují k péči dané rostliny. Rostliny lze přiřadit do uživatelem vytvořených místností. Připomínky/notifikace se nastavují zcela ručně, aplikace nevytvoří sama plán jako u aplikací Planta a PlantIn. Na druhou stranu je toto ruční nastavení přehledné a umožňuje uživateli plnou kontrolu.

Dalšími funkcemi jsou také, po vzoru předešlých aplikací, rozpoznání chorob/nemocí, odborná pomoc od zkušených odborníků, nabízení článků ohledně pěstování. Aplikace nabízí zakoupení předplatného (prémium), které zpří-



Obrázek 1.3: Ukázka aplikace Blossom

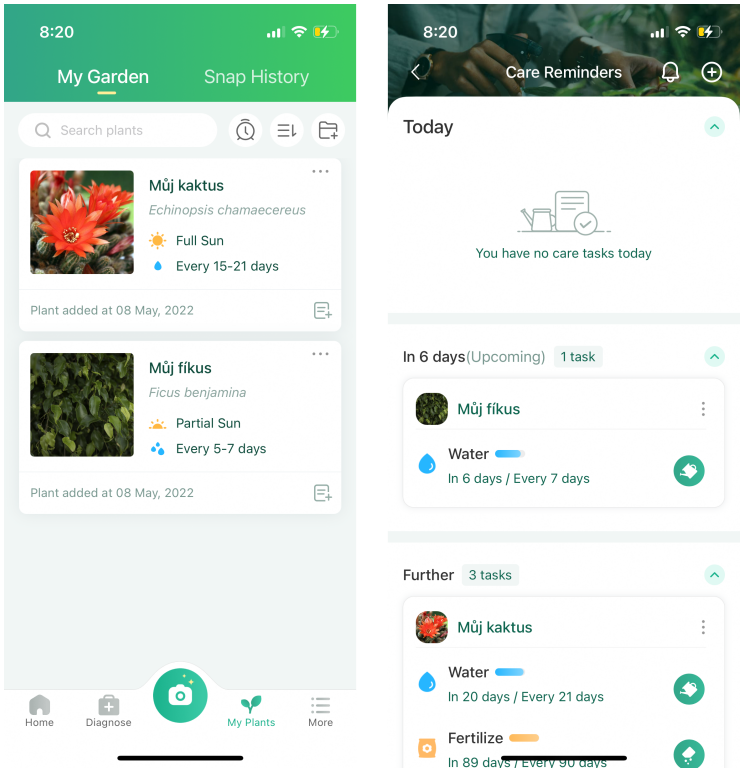
stupní veškeré funkce aplikace a neomezené rozpoznávání (rostlin a jejich nemocí).

Aplikace je vzhledově dle mého názoru nejpovedenější, ale místy trpí „přeplácaností“ prvků, což může být na začátku trochu matoucí. I zde je absence týmové spolupráce či sledovaní postupu.

1.3.5 PictureThis

Další aplikací je PictureThis 1.4. Aplikace jako hlavní funkci nabízí identifikaci rostliny pomocí fotoaparátu, další funkce jsou podobné předchozím aplikacím. Ukládání rostlin, nastavení upomínek na zalévání/hnojení, diagnostiku nemocí či pomoc od expertů. Jedinečnými funkcemi oproti ostatním je rozpoznání hmyzu či ptáků, nicméně toto se netýká tématu.

Opět je zde model zpoplatnění (prémium), které odemkne všechny funkce a odstraní limity. Aplikace opět nenabízí ukládání rostlin v týmu a sdílení upomínek/postupu.

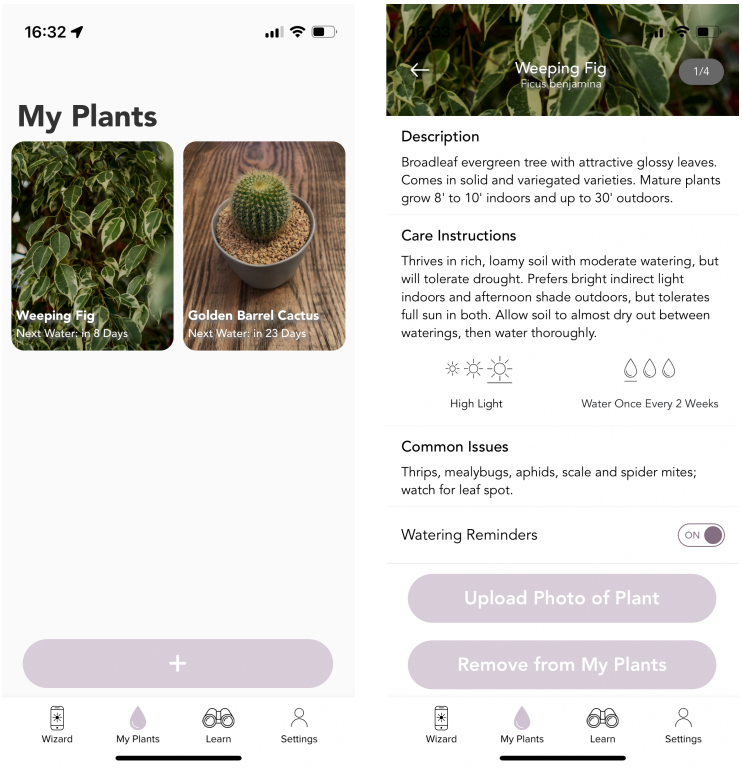


Obrázek 1.4: Ukázka aplikace PictureThis

1.3.6 Florish

Aplikace Florish 1.5 je ukázkou jednoduchosti a minimalismu. Nenabízí z daleka tolik, co předešlé aplikace. Umožňuje vybírat z celkem omezené nabídky rostlin a zobrazuje pouze základní informace. Po vložení rostliny může uživatel dostávat připomínky k zalévání. Dále také umožňuje rozpoznání úrovně světla na daném místě pomocí kamery a vyplnění krátkého dotazníku, na jehož základě umí aplikace doporučit rostlinu uživateli. Na obrazovce *Learn* obsahuje pár článků či návodů ohledně rostlin a péče o ně. Aplikace je zcela zdarma.

Aplikace sice umožňuje přidávání fotografií rostlin (zřejmě k rozšíření databáze fotografií), nicméně je u svých rostlin nelze měnit, stejně tak jako rostliny přejmenovat. Není divu, že se zde nenachází možnost sdílení rostlin více uživatelů mezi sebou. Aplikace je hodně jednoduchá a byla zvolena zejména kvůli zmíněné jednoduchosti a minimalistickému zpracování.



Obrázek 1.5: Ukázka aplikace Florish

1.3.7 Ostatní

Prozkoumány byly i další aplikace, většina z nich měla obdobné funkce jako zmíněné aplikace. Zde je jejich výčet:

- Gardenia
- Greg
- Lovely
- PlantPal
- Plantnote

1.3.8 Hodnocení pomocí Nielsenovy heuristiky

V tabulce 1.2 se nachází hodnocení jednotlivých bodů Nielsenovy heuristiky u každé z pěti výše podrobněji rozebraných konkurenčních aplikací. Při hodnocení byly udělovány známky na stupnici od 1 (nejlepší) do 5 (nejhorší). Nejlepší hodnocení dosáhla aplikace Planta, která obdržela 21 bodů (méně bodů znamená lepší hodnocení). 22 bodů poté získaly aplikace PlantIn a Blossom.

Nejhorší výsledek, 28 bodů, pak patří aplikacím PictureThis a Florish. Uživatelské rozhraní všech hodnocených aplikací vypadá vcelku solidně, žádná vyloženě neobsahuje kritické chyby v uživatelském návrhu.

	Planta	PlantIn	Blossom	PictureThis	Florish
Viditelnost stavu systému	2	1	2	2	2
Shoda mezi systémem a reálným světem	1	2	3	3	3
Uživatelská kontrola a svoboda	2	3	2	4	3
Konzistence a standardy	3	2	2	3	2
Prevence chyb	2	2	2	2	4
Rozpoznání místo vzpomínání	2	2	1	3	2
Flexibilita a efektivita	2	3	3	2	3
Estetický a minimalistický design	2	2	3	3	2
Diagnostika, rozpoznání a oprava chyb	3	2	2	3	3
Nápověda a dokumentace	2	3	2	3	4
Součet	21	22	22	28	28

Tabulka 1.2: Tabulka hodnocení konkurenčních aplikací pomocí jednotlivých zásad Nielsenovy heuristiky

1.3.9 Souhrn

Jak je vidět z analýzy existujících řešení, aplikace mají poměrně vypracované funkce ohledně rozpoznávání rostlin pomocí strojového učení. Některé algoritmus samy spravují, některé používají externí API. Některé aplikace umí sestavit automaticky plán zalévání, či jiných činností v závislosti na nastavených parametrech uživatele a daných místností.

Tím, čím se bude výsledná aplikace odlišovat, je možnost vytvářet skupiny, ve kterých budou její členové moci provádět a sdílet akce a události. Dále pak budou moci nastavit a dostávat notifikace ohledně různých akcí (zalévání atd.), a to buď jako připomínku nebo přímo notifikaci o jejím provedení některým ze členů skupiny.

1.4 Požadavky

V této sekci jsou formulovány požadavky na aplikaci, které vycházejí z předešlé analýzy. Požadavky se obecně rozdělují do dvou skupin, jimiž jsou požadavky funkční a nefunkční (vysvětlení je popsáno v 1.4.2 a 1.4.3). U každého požadavku je nutné určit název a stručně ho popsat. Dále je také vhodné přiřadit prioritu, složitost a kategorii. Pro kategorie byl využit model *FURPS*. Pro priority lze využít metodu *MoSCoW*, která jednotlivým požadavkům přiřazuje jednu ze čtyř priorit – M (Must have), S (Should have), C (Could have), W (Won't have). Ta však použita nebyla (cílem bylo splnit všechny požadavky) a jednotlivé priority požadavků byly ohodnoceny slovně.

1.4.1 FURPS [5]

FURPS je zkratka pro Functionality, Usability, Reliability, Performance a Supportability (funkčnost, použitelnost, spolehlivost, výkonnost a podporovatelnost). Právě těchto 5 termínů tvoří jednotlivé kategorie, které se používají pro kategorizaci požadavků na systémy. Model FURPS byl vyvinut ve společnosti Hewlett-Packard, je stále používán v současnosti a vznikla i jeho modifikace FURPS+, která k základním pěti kategoriím přidává čtyři další.

Funkčnost (Functionality) Jedná se o nejdůležitější požadavky. Ty se týkají jednotlivých funkcí, které musí daný systém splňovat. Tedy v podstatě to, co by měl systém umět a kvůli čemu má být vyvinut/existovat.

Použitelnost (Usability) Vypovídají o schopnosti systému být jednoduše používán uživateli. Jedná se o vzhled, dojem a vliv na uživatele, přístupnost.

Spolehlivost (Reliability) Týkají se spolehlivosti, dostupnosti, předvídatelnosti, přesnosti.

Výkon (Performance) Popisují, jak má být systém výkonný a škálovatelný, což zahrnuje jak rychle by měl systém odpovídat, jaké a kolik prostředků bude systém využívat.

Schopnost být udržována (Supportability) Věnují se udržitelnosti systému, čili jak je systém čitelný, jaká je zvolena architektura, jestli se dá jednoduše spravovat a provádět modifikace.

1.4.2 Funkční požadavky

Funkční požadavek (anglicky *functional requirement*, zkratka FR) je specifikace funkce či služby, kterou musí daná aplikace splňovat. Jde o popis dané funkce, jak se systém bude chovat při daných uživatelských vstupech a jaký bude výsledný výstup. Právě specifikace funkčních požadavků je jedním ze

základních pilířů pro návrh aplikace, neboť ujasňuje, co má aplikace dělat a umět. Všechny tyto požadavky spadají do kategorie Funkčnost (Functionality) v modelu FURPS. [6]

Níže jsou vypsány jednotlivé funkční požadavky, včetně názvu, priority, obtížnosti a stručného popisu.

FR1 Vytvoření a správa uživatelského účtu

Priorita: Vysoká
Složitost: Vysoká

Uživatel se bude moci registrovat nebo se přihlásit. Uživatelský účet pak bude identifikovat jednotlivé uživatele aplikace, budou k němu přiřazovány jednotlivé týmy, rostliny, provedené akce a další. Svůj uživatelský účet/profil si poté bude moci měnit či upravovat.

FR2 Tvorba a správa týmů

Priorita: Vysoká
Složitost: Střední

Pro sdílení záznamů o rostlinách bude klíčové vytvořit funkcionalitu týmové spolupráce. Uživatel bude moci vytvořit tým, do kterého následně může pozvat další uživatele. Společně tak budou mít možnost kooperace či spolupodílení se na péči o rostliny.

FR3 Vyhledávání a zobrazení informací o rostlinách

Priorita: Vysoká
Složitost: Střední

Aplikace bude obsahovat databázi rostlin, přičemž u každé rostliny se uživatel dozví informace o ní. Nad touto databází bude možné jednoduše vyhledávat podle jména rostliny (vyhledávací textové pole v mobilní aplikaci).

FR4 Přidávání a správa míst

Priorita: Střední
Složitost: Nízká

Bude existovat možnost vytvářet, editovat místa, která budou přiřazená jednotlivým záznamům o rostlině.

FR5 Přidávání nových rostlin do svého/týmového seznamu rostlin

Priorita: Vysoká
Složitost: Střední

Na detailu rostliny si uživatel bude moci vybranou rostlinu přidat do své nebo týmové sbírky rostlin. Jedná se o konkrétní záznam o rostlině, uživatel může mít více rostlin stejného druhu.

FR6 Sledování a správa svých/týmových rostlin

Priorita: Vysoká
Složitost: Vysoká

Své nebo týmové rostliny bude moci sledovat, myšleno především události/akce, které byly provedeny nebo jsou naplánovány. Případně bude moci záznam o rostlině upravit (například uživatelský název).

FR7 Přidávání fotek ke svým/týmovým rostlinám

Priorita: Střední
Složitost: Nízká

Pro sledování rostlin bude uživatel moci přidávat fotografie, aby měl možnost následně porovnávat růst a vývoj dané rostliny.

FR8 Nastavení plánu akcí/událostí

Priorita: Vysoká
Složitost: Vysoká

U každého záznamu rostliny bude uživatel mít možnost nastavit jeden z připravených plánů nebo si ho sám manuálně sestavit (pomocí vhodných připomínek).

FR9 Dostávání a nastavení notifikací

Priorita: Střední
Složitost: Střední

Aby uživatel nezapomněl na nějakou akci nebo aby byl informován, že někdo například zalil rostlinu, bude si moci nastavit notifikace.

FR10 Zobrazení nadcházejících událostí/akcí

Priorita: Střední
Složitost: Nízká

Uživatel si bude moci zobrazit obrazovku s přehledem nadcházejících akcí.

1.4.3 Nefunkční požadavky

Nefunkční požadavek (anglicky *non-functional requirement*, zkratka NFR) je požadavek na systém, který se netýká přímo jeho byznysové funkcionality. Typicky se jedná o kvalitativní parametry systému, může jít například o dostupnost, použitelnost, responzivita, zabezpečení a další. Jejich specifikace následně ulehčí a ujasní některé věci ohledně samotného vývoje (architektura, technologie). [7]

Zde je vypsán výčet nefunkčních požadavků na aplikaci (název, priorita, obtížnost, kategorie podle FURPS a krátký popis).

NFR1 Klient-server architektura

Priorita: Vysoká
Složitost: Nízká
Kategorie: Schopnost být udržována (Supportability)

Jedná se o jeden ze standardních modelů architektury, kdy server je strana, která spravuje a poskytuje data, která následně zobrazuje a prezentuje klient uživateli (typicky webová, mobilní nebo desktopová aplikace). Tento model architektury bude při návrhu a implementaci výsledné aplikace využit.

NFR2 REST API

Priorita: Vysoká
Složitost: Střední
Kategorie: Schopnost být udržována (Supportability)

Server bude komunikovat s mobilní aplikací pomocí protokolu HTTP s využitím architektonického vzoru REST.

Mobilní aplikace pro systém iOS

Priorita: Vysoká
Složitost: Vysoká
Kategorie: Schopnost být udržována (Supportability)

Klientská část aplikace bude implementována pro mobilní aplikace s operačním systémem iOS, tedy pro mobilní zařízení od společnosti Apple.

NFR4 Jednoduchost používání aplikace

Priorita: Střední
Složitost: Vysoká
Kategorie: Usability (Použitelnost)

Bude kladen důraz, aby byla mobilní aplikace jednoduchá na používání, což lze zajistit vhodným návrhem vzhledu a správným a logickým propojením obrazovek pomocí interakčních prvků.

NFR5 Svižnost aplikace

Priorita: Střední
Složitost: Střední
Kategorie: Výkon (Performance)

Ideálně by aplikace neměla mít dlouhé odezvy v komunikaci a obecně se zasekávat. Aplikace by rozhodně neměla padat a v případě problému zobrazovat smysluplné hlášky.

NFR6 Vysoká dostupnost serveru

Priorita: Střední
Složitost: Nízká
Kategorie: Spolehlivost (Reliability)

Mobilní aplikace by měla být schopna zobrazovat data neustále (kdykoliv na požádání), je tedy potřeba, aby byl server dostupný ideálně 24/7.

Kapitola 2

Metodiky vývoje software

Softwarový vývoj není jednoduchá disciplína. Jedná se o poměrně složitý proces, který se skládá z mnoha podprocesů/částí. Ty zároveň nejsou striktně dány, neboť každý vývojový proces se liší, především podle vyvíjeného produktu. Proto se v průběhu let začaly vytvářet různé metodiky pro vývoj software, které popisují vhodné systematické kroky a postupy při vývoji. Cílem těchto metodik je především zjednodušení samotného vývoje, ve smyslu poskytnutí jasné struktury. Vhodná struktura přináší větší efektivitu, přehlednost, kontrolu a usnadňuje plánování a komunikaci. Je ovšem nutné nevybrat slepě určitou metodiku jen proto, že patří mezi aktuální trendy nebo proto, že ji používá ona společnost při jejich vývoji. Volba by měla brát v potaz přímo daný produkt (rozsáhlost, požadavky, technologie atd.), jenž má být vyvinut, a také tým, který se bude na vývoji podílet (velikost, preference, zkušenost atd.). Zároveň každá metodika má svoje kladné i záporné stránky. V následujících sekcích jsou popsány nejznámější a nejpoužívanější metodiky. [8, 9]

2.1 Waterfall [10]

Jednou z prvních a zároveň také jednou z nejpoužívanějších metodik je *Waterfall*, v překladu vodopádový model. Jedná se o sekvenční vývojový model, který je rozdělen do několika hlavních fází (jejich počet se může lišit, standardně se jich uvádí 5), přičemž každá závisí na té předchozí (lze si představit vodopád s jednotlivými patry, viz obrázek 2.1) a nelze se vracet zpět. Jednotlivé fáze jsou:

Requirements (Požadavky) Fáze, při které dochází k analýze, specifikaci požadavků a definici funkcí, které má výsledný produkt obsahovat.

Design (Návrh) Vývojový tým, vzhledem k požadavkům (z předchozí fáze) definuje vhodnou softwarovou architekturu, vybírá vhodné technologie.

Implementation (Implementace) Na základě návrhu je software/produkt implementován.

Verification (Verifikace) V této fázi je ověřeno a otestováno, zda je implementace funkční a odpovídá všem požadavkům, které byly specifikovány. Zároveň je také daná implementace vydána.

Maintenance (Údržba) Vydáním produktu ovšem práce většinou nekončí. V této fázi jde o dodatečné testování, opravu nově vzniklých problémů či údržbu a upgrade strojů.

Výhody

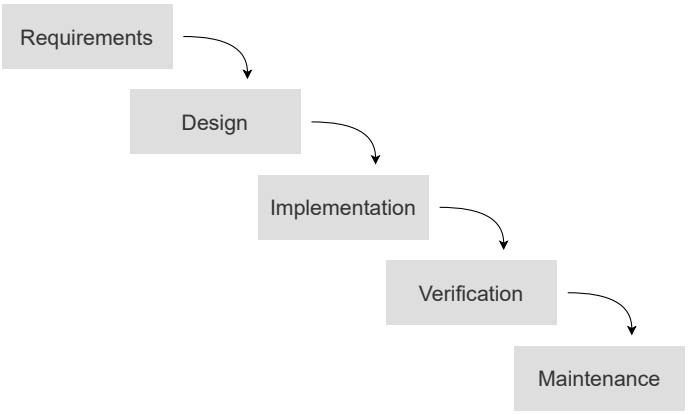
- Linearita přispívá k snadnějšímu pochopení, zejména pro nové vývojáře.
- Všechny specifikace a výstupy jsou popsány před zahájením vývoje.
- Protože je vše jasně definováno a popsáno v každé fázi, není zde prostor pro špatnou komunikaci informací.

Nevýhody

- Riziko odchýlení projektu od cíle z důvodu absence zpětné vazby od zákazníků v počátečních fázích.
- Testování probíhá až na konci vývoje. Některé chyby, které zejména vznikly na počátku vývoje, mohou být problematické na opravu.
- Striktnost a malá flexibilita metodiky není vhodná pro komplexnější projekty.
- Důraz na dokumentaci místo dodávání řešení, která řeší problémy uživatelů.

2.2 Agile [11]

Agilní metodiky patří v poslední době k nejvíce populárním metodikám. Místo lineárního postupu, jako u vodopádové metodiky, se soustředí na správnou komunikaci se zákazníkem a uspokojování jeho potřeb. Nelpí tolik na tvorbě dokumentace a dodržování striktních postupů. Vývoj probíhá v tzv. sprintech, které zpravidla trvají 1 - 4 týdny. V těchto sprintech se vyvíjí dílčí části produktu, které berou v potaz podněty zákazníka a zároveň se průběžně testuje. Tyto sprinty se neustále opakují, jedná se tedy o iterativní proces. V průběhu času vznikly různé modifikace a zpřesnění této metodiky, například Scrum a další.



Obrázek 2.1: Znázornění metodiky Waterfall

Výhody

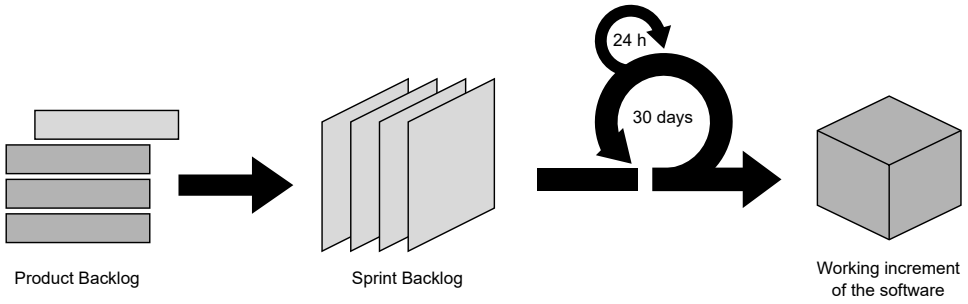
- Minimum chyb díky opakovanému testování a ladění.
- Častá komunikace se zákazníkem umožňuje rychle reagovat na jeho požadavky a řešit případné problémy.
- Dosažení kvalitnějšího konečného výsledku.

Nevýhody

- Časová náročnost kvůli časté komunikaci
- Díky nedůrazu na dokumentaci může být vývoj v pozdějších fázích problematičtější.
- Ztráta soustředěnosti kvůli častým požadavkům.
- Díky nestrukturovanému přístupu jsou potřeba zkušení a nezávislí vývojáři.

2.3 Scrum [12]

Tato metodika spadá pod agilní metodiky a patří mezi nejflexibilnější. Dodržuje agilní koncepty a rozděluje vývojový tým do rolí. První rolí je Product Owner (vlastník produktu). Ten je zodpovědný za komunikaci s klientem a zajišťuje, že vývojový tým dodržuje termíny a doručuje požadované funkcionality. Dále se v týmu nachází Scrum Master, který slouží jako prostředník mezi vlastníkem produktu a vývojovým týmem. Ten má zároveň za cíl zajistit správný chod scrum procesu, odstiňuje ostatní členy týmu od negativních vlivů, organizuje schůzky/porady a povzbuzuje tým. Vývojový tým (obvykle 3



Obrázek 2.2: Znázornění metodiky Scrum

- 9 vývojářů) pak pracuje na aktuálním inkrementu daného sprintu. Součástí scrum procesu je plánování nadcházejícího sprintu a denní scrumy (krátké porady).

Výhody

- Časová a ekonomická efektivita.
- Krátké iterace umožňují rychle reagovat na požadavky klienta.
- Pravidelné schůzky/porady zajišťují, že všichni členové ví, co se zrovna děje a kdo na čem pracuje.

Nevýhody

- Nutnost zkušených vývojářů a také odhodlanost následovat metodiku scrum.
- Denní scrum porady mohou být pro vývojáře frustrující.
- Nevhodné pro rozsáhlé projekty.

2.4 Feature Driven Development (FDD) [13]

Stejně jako scrum, tato metodika je založena na agilním přístupu. Hlavní myšlenkou je vytvoření základního systémového modelu, ze kterého je poté vytvořen seznam funkcí (feature list). Separátně se poté na každé funkci prochází plánováním, návrhem a samotným vývojem. Práce na jednotlivých funkcí by zpravidla neměla trvat déle než 2 týdny.

Výhody

- Dá se pracovat na více funkcích zároveň a díky tomu je tato metodika vhodná pro rozsáhlé projekty.
- Rozdělení komplikovaných úloh na menší, což vede k větší efektivitě.
- Není potřeba častých porad/schůzek, komunikace především v textové formě.

Nevýhody

- Nevhodné pro malé projekty.
- Závislost na hlavním vývojáři, který má kromě vývoje na starost i distribuci a koordinaci úloh mezi ostatní vývojáře.
- Absence kvalitní dokumentace.

2.5 Prototyping [14]

Tato metodika je založena na vytvoření prototypu aplikace (základní klíčové funkce), který je poté dokola testován a přepracováván do té doby, než je dosaženo přijatelného výsledku. Postupuje se zpravidla tak, že ze začátku se specifikují požadavky, dojde k rychlému návrhu a následnému vytvoření prototypu (základní klíčové funkce, není dokonalý). Poté dochází k již zmiňovaným cyklům testování a znovuvytváření lepšího prototypu. Nakonec je finální produkt řádně otestován, nasazen do produkce a následně je prováděna údržba. Pro úplnost je třeba dodat, že existují různé varianty této metodiky (*rapid throwaway*, *evolutionary*, *incremental*, *extreme*), které se liší ve vlastnostech prototypů nebo způsobem, jakým jsou vytvářeny.

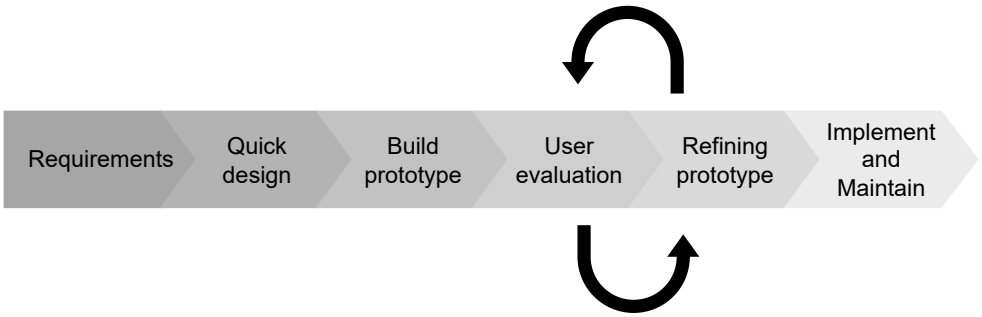
Výhody

- Přímočará metodika, jednoduchá na pochopení.
- Klient velmi brzy vidí „nějaký“ výsledek a může hned poskytovat zpětnou vazbu. Zároveň se tak dají brzy podchytit potenciální problémy.

Nevýhody

- Celý proces vytváření, testování, vylepšování je značně časově náročný.
- Klient může z dosavadních prototypů nabýt představy, že produkt je téměř hotový, nebo naopak přestane mít zájem ho dodělat, protože není s dosavadním výsledkem spokojený.

- Zaostávající dokumentace z důvodu častých změn požadovaných od zákazníka.



Obrázek 2.3: Znázornění metodiky Prototyping

2.6 Lean [15]

Tato metodika se zrodila ve firmě Toyota a stojí na dvou základních pilířích. Prvním je vytváření nových funkcí či provádění změn, které přináší největší hodnotu, zaberou nejméně úsilí a času. Zadruhé je kladen důraz na respektování lidí. Lean se mimořádně hodí při přepracovávání již hotových, zejména dlouho používaných projektů (přechod na nové technologie, předělání uživatelského rozhraní atd.).

Výhody

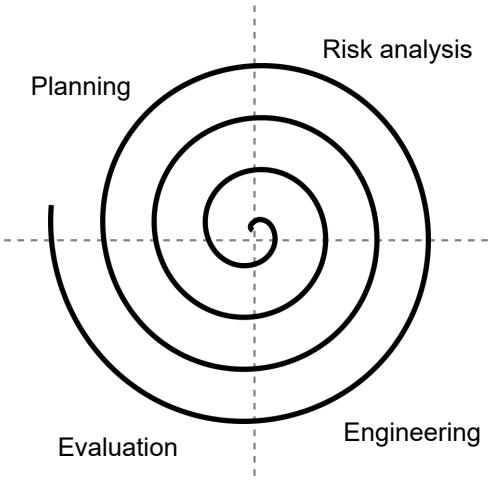
- Minimální životaschopný produkt (MVP) je dodán poměrně rychle.
- Náklady na vývoj jsou poměrně nízké.

Nevýhody

- Nutnost precizní dokumentace a její porozumění.
- Vyžaduje velmi zkušené jednotlivce s dostatečnými znalostmi. Nelze se učit „za pochodu“, neboť to přináší risk.

2.7 Spiral [16]

Ve spirálové metodice se proces vývoje skládá ze čtyř fází. Vývojáři těmito fázemi průběžně procházejí ve spirále/cyklu. Po každém průchodu postupují do nové iterace vývoje, přičemž cílem je v každé iteraci postupně zdokonalovat produkt.



Obrázek 2.4: Znázornění metodiky Spiral

Plánování Definice cílů v dané fázi vývoje.

Analýza rizika Snaha o předvídání rizika a snaha navrhnout jejich řešení.

Vývoj Návrhu a vývoj produktu na základě předchozích fází.

Vyhodnocení Vyhodnocení stavu projektu a plánování další iterace.

Výhody

- Možnost vyhnout se riziku díky důkladné analýze.
- Velká flexibilita.
- Vhodné pro velké projekty.
- Prostor pro zpětnou vazbu od zákazníků.

Nevýhody

- Použití této metodiky může být dosti nákladné.
- Úspěch projektu je závislý na analýze rizik, která zároveň vyžaduje odborné znalosti.
- Nefunguje příliš dobře pro menší projekty.

2.8 Výběr metodiky

Důležitým faktorem, který hrál roli ve výběru metodiky, je jednočlenný vývojový tým (já, autor této práce). Není tak nutné řešit spolupráci mezi více lidmi. Nejedná se zároveň o komplexní projekt a náklady na vývoj jsou nízké (převážně jen čas a zaplacení služeb pro finální nasazení). I z důvodu přímočarosti, jasné struktury a snadné uchopitelnosti byla použita především vodopádová metodika (Waterfall), která je místy podle potřeby lehce obohacena o prvky jiných metodik.

Kapitola 3

Návrh

3.1 Persony

Personou se rozumí smyšlená fiktivní osoba, která reprezentuje uživatele aplikace. Persony slouží k vytvoření představy, co uživatel chce, jaké má potřeby, jaké jsou jeho cíle či jak se uživatel bude chovat. Pomáhají tak identifikovat uživatele, kteří aplikaci budou hojně využívat, či uživatele, pro které aplikace vhodná není. Při vytváření person jde o zobrazení uživatele, která reprezentuje vytyčenou skupinu lidí. Typicky se tvoří 3 různé persony, přičemž jde o typického a příležitostného uživatele a uživatele, který aplikaci téměř nepoužije (negativní uživatel). Persona by měla působit jako reálný člověk, dává se jí jméno, věk, pohlaví a dále také jaké jsou třeba její záliby, historie a typický den. [17, 18]

3.1.1 Persona A - typický uživatel

Jméno Jana Nováková

Věk 29

Pohlaví Žena

Koníčky vaření, cestování, pěstování rostlin

Životní historie Jana je pracovitá, energická, milující matka a manželka. Žije v malém domku na okraji Prahy. Naplno využívá svůj volný čas. Dělá zásadně činnosti, které ji baví. To, co ji nebaví, přenechává ostatním. Ráda se učí novým věcem. Ve volném čase ráda sportuje, vaří a stará se o. Jana vystudovala střední obchodní školu. Ihned po škole začala pracovat v marketingové agentuře. Následně odešla na mateřskou dovolenou, přičemž po ní se vrátila zpět do předešlé práce. Nyní aspiruje na pozici vedoucí oddělení.

Typický den Jana vstává okolo šesté hodiny, spolu jako rodina se nasnídají, připraví se a vypraví se autem do práce, kde cestou vysadí svého syna ve školce. Cestou z práce syna ve školce opět vyzvedne a spolu dorazí domů. Zbytek dne Jana tráví se svým synem a manželem doma. Jako vášnivý pěstitel rostlin se často (mimo zimní období) pohybuje po jejich menší zahrádce, na které spolu pěstují nejrůznější druhy rostlin a květin. Také uvnitř jejich domu lze nalézt velké množství pokojových rostlin. Občas ji přijde, že rostlin má přehršel a stane se jí, že se o ně nestíhá starat, nebo že na nějakou sem tam zapomene.

3.1.2 Persona B - příležitostný uživatel

Jméno Petr Klíč

Věk 46

Pohlaví Muž

Koníčky fotbal, auta

Životní historie Petr už má něco za sebou, leccos ho už nepřekvapí. Má své zajeté koleje, ale zároveň se nebrání vyzkoušet i něco nového. Používá aktivně chytrý telefon, který mu slouží hlavně jako komunikační zařízení. Ve svém bytě má pár pokojových rostlin, které sporadicky (když si vzpomene) zalije. Už několikrát se mu stalo, že zapomněl zalít svůj fíkus, a tak mu v létě uschl.

Typický den Petr je mužem středního věku, je otcem dvou dětí. Petr žije sám, s manželkou se rozvedl, když mu bylo 36 a děti už se odstěhovaly do vlastního. Bydlí ve větším městě v menším bytě a kousek od něj vlastní garáž. Každý pracovní den vstává v půl sedmé ráno, nasnídá se, pustí si ranní zprávy v televizi a následně jde do zaměstnání (provozní elektrikář). Obědvá v práci a domů se vrací kolem čtvrté. Po práci chodí dvakrát týdně hrát s přáteli rekreačně fotbal, či se stará v garáži o své milované auto. Večer občas zajde s přáteli na pivo do hospody nebo sleduje fotbalová utkání v televizi. O víkendech odpočívá a jezdí za ním na návštěvu děti.

3.1.3 Persona C - negativní uživatel

Jméno Helena Dvořáková

Věk 72

Pohlaví Žena

Koníčky televize, procházky v přírodě

Životní historie Helena je pesimistická, poctivá a obětavá důchodkyně. Nerada se učí nové věci. Nerada mění své názory. Je přesvědčená, že si za svůj život všechny své názory na svět vytvořila a nechce je již měnit. Helena vystudovala učitelství a celý svůj život učila na druhém stupni základní školy zeměpis a dějepis. Vždy byla vnímána ve škole jako ta „hodná“ učitelka. Žije sama v malém domku na okraji malého městečka. Manžel ji bohužel před pár lety umřel při autonehodě.

Typický den Helena vstává ráno kolem šesté hodiny. Připraví si snídani a sní si ji v doprovodu ranních televizních zpráv. Dopoledne obvykle tráví doma, kouká na televizi či si čte noviny. Pokud nemá připravený oběd z předešlých dní, tak si ho s dostatečným předstihem uvaří a pak ho následně sní. Po obědě vyráží na krátkou procházku do přírody nebo se jde starat o svoji zahrádku, neboť si ji chce ještě užít, dokud může. Večer tráví buď sama u televize nebo chodí na návštěvy k jejím přátelům, kde rozebírají nejrůznější věci, které se staly. O víkendech za ní jezdí příbuzní.

3.2 Případy užití a scénáře

Význam termínu případ užití (anglicky *use case*) má široké pojetí. Ve vývoji softwaru lze jednoduše říci, že jde v podstatě o specifikaci a popis klíčové funkcionality aplikace. Tento popis je vytvořen na základě cílů, kterého chce daná entita dosáhnout (typicky uživatel nebo systém), a je reprezentován sadou kroků, které vedou k jejich dosažení. Jednotlivé případy užití by měly ideálně popisovat pouze jednoduché akce/činnosti. Samotný formát, jak mají případy užití vypadat, není striktně daný, níže je použit tzv. Fowlerův styl. Ten obsahuje název, krátký popis a jednotlivé scénáře (sled kroků), které se v daném případu nachází. U případu užití nemusí být jeden jediný scénář, obvykle jich může existovat více (například scénář, ve kterém dojde k chybě). [19, 18]

Případy užití lze zachytit v diagramu případů užití (jeden z UML diagramů chování). Ten má modelovat funkcionality systému pomocí aktérů (anglicky *actor*) a případů užití. Aktéři jsou entity, které jsou spojeny s jednotlivými případy užití (aktér se tedy může účastnit daných případů užití). Aktéři jsou zobrazeni jako jednoduchý panáček a případy užití jsou zobrazeny jako elipsa s jeho názvem uvnitř. Dohromady jsou tyto dva prvky spojeny čárou, která značí, že daný aktér je spojen s daným případem užití. [20]

Níže se nachází popis aktérů, samotný diagram 3.1 a popis jednotlivých případů užití.

Aktéři

Nepřihlášený uživatel Tento aktér zachycuje uživatele, který nemá přístup k hlavním funkcím aplikace. Pro přístup k nim se musí nejdříve přihlásit, nebo pokud nemá uživatelský účet, tak se zaregistrovat.

Přihlášený uživatel Má přístup k hlavním funkcím aplikace. Nepřihlášený uživatel se stává tímto aktérem přihlášením nebo registrací.

Systém Stará se automatizované funkce (posílání notifikací).

Případy užití

UC1 Registrace Možnost vytvořit si svůj uživatelský účet, který bude využívat a vystupovat pod ním v aplikaci. Tato možnost je dostupná, když v aplikaci není přihlášený žádný uživatel. Registrace probíhá formou registračního formuláře, kde potenciální uživatel vyplní informace potřebné k registraci.

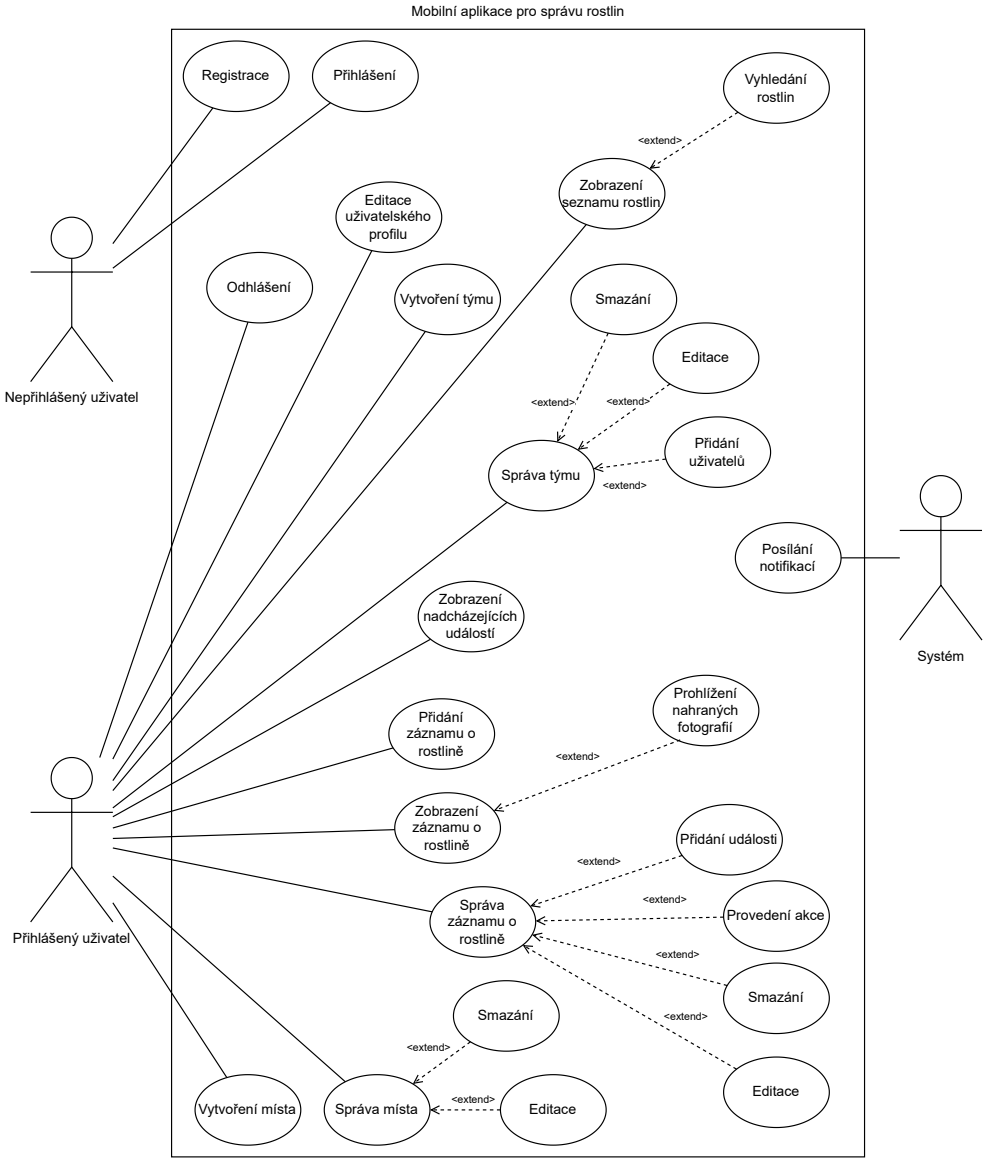
1. Potenciální uživatel na úvodní obrazovce klikne na tlačítko/odkaz sloužící k registraci uživatele.
2. Aplikace uživateli zobrazí registrační formulář.
3. Uživatel vyplní potřebné položky formuláře a odešle požadavek na registraci kliknutím na tlačítko.
4. Systém ověří, zda zadané informace jsou v pořádku. Pokud je tomu tak, uživatel je registrován a rovnou přihlášen. Naopak, pokud dojde k nějaké chybě, uživatel je upozorněn.

UC2 Přihlášení Přihlašovací proces uživatele, který už má vytvořený účet. Jednoduché vyplnění přihlašovacího formuláře s jeho následným odesláním pro ověření.

1. Na úvodní obrazovce (pokud není v aplikaci uživatel přihlášen) je uživateli zobrazen přihlašovací formulář.
2. Uživatel vyplní přihlašovací formulář.
3. Uživatel klikne na tlačítko k odeslání požadavku na přihlášení.
4. Zadané informace systém ověří a přihlásí uživatele do aplikace (uživatel existuje a heslo souhlasí) nebo v případě chyby na ni upozorní.

UC3 Odhlášení Uživatel očekává, že se může z aplikace odhlásit. Odhlášení je možné jen pokud je uživatel již přihlášen.

1. Uživatel se nejdříve proklikne do nastavení.
2. Následně klikne na odhlašovací tlačítko.



Obrázek 3.1: Diagram případů užití

3. Uživateli je zobrazeno potvrzovací okno, zda si přeje se opravdu odhlásit.
4. Při potvrzení je uživatel odhlášen.

UC4 Editace a nastavení uživatele Uživatel by měl mít možnost upravovat si svůj uživatelský účet (například změna hesla) a nastavit si funkcionality aplikace (např. zda bude dostávat notifikace).

1. Uživatel se nejdříve proklikne na obrazovku s nastavením.
2. Následně klikne na tlačítko, které slouží ke změně hesla.
3. Uživateli je zobrazen formulář pro vyplnění nového hesla (součástí je stávající heslo pro potvrzení).
4. Při potvrzení a splnění požadavků na nové heslo je uživateli změněno heslo.

UC5 Vytvoření týmu Každý uživatel si může vytvořit tým, ve kterém bude společně s dalšími uživateli sdílet informace o pěstovaných rostlinách.

1. Uživatel vybere akci pro vytvoření týmu.
2. Uživatel vyplní jméno týmu a potvrdí vytvoření.

UC6 Správa týmu Členové týmů mají možnost spravovat daný tým. Uživatel očekává, že může měnit jméno týmu, přidávat a odebírat členy týmu, popřípadě tým smazat.

1. U týmu uživatel vybere akci pro správu týmu.
2. Následně je zobrazeno jméno, aktuální členové a tlačítko pro smazání týmu. Jméno může uživatel změnit a členy přidávat a odebírat.

UC7 Zobrazení seznamu rostlin Seznam rostlin v databázi aplikace, ve kterém lze vyhledávat.

1. Uživatel zobrazí obrazovku se seznamem rostlin.
2. V seznamu může uživatel scrollovat a vyhledávat pomocí jména.
3. Kliknutím na některou rostlinu v seznamu se může uživatel dostat na její detail.

UC8 Zobrazení detailu rostliny Uživatel bude moci zobrazit detail o konkrétní rostlině. Ten může zobrazit vybráním ze seznamu rostlin nebo proklikem z konkrétního záznamu o rostlině (týmové či osobní).

1. Uživatel zobrazí detail klikem na položku v seznamu rostlin nebo tlačítkem na konkrétním záznamu o rostlině.
2. Uživateli je zobrazen detail o rostlině, na kterém nalezne různé detaily ohledně dané rostliny.

UC9 Vytvoření místa Uživatel bude moci vytvořit místo (místnost, část zahrady), do kterého pak lze přiřadit jednotlivé záznamy o rostlinách.

1. Uživatel vyvolá akci k vytvoření místa.
2. Uživatel vyplní jednoduchý formulář k vytvoření místa a následně ho potvrdí kliknutím na tlačítko.

UC10 Správa místa Místo bude moci být editováno.

1. Uživatel vyvolá akci k editaci místa.
2. Je zobrazena editovací obrazovka místa.
3. Uživatel vyplní požadované změny a potvrdí je tlačítkem.

UC11 Přidání záznamu o rostlině Uživatel bude moci přidat záznam o konkrétní rostlině do svých nebo týmových rostlin. Toto lze provést z detailu o rostlině.

1. Uživatel se nachází na detailu rostliny a klikne na přidávací tlačítko.
2. Uživateli se zobrazí obrazovka, která slouží k vytvoření záznamu o rostlině.
3. Uživatel vyplní požadované informace k vytvoření záznamu a klikne na tlačítko k vytvoření daného záznamu.
4. Daný záznam o rostlině je systémem vytvořen a je rovnou uživateli zobrazen.

UC12 Zobrazení záznamu o rostlině Uživatel by měl být schopen zobrazit svůj/týmový záznam o rostlině. V záznamu by měl viděl budoucí události/akce, historii událostí/akcí, které byly kým u daného záznamu provedeny, galerii fotografií daného záznamu.

1. Uživatel rozklikne daný záznam o rostlině.
2. Na příslušné obrazovce si může uživatel prohlížet detail záznamu.

UC13 Správa záznamu o rostlině Jednotlivé záznamy o rostlinách by měly být možné spravovat uživatelem/i. Mělo by být možné zaznamenat provedení akce, změnit jméno a náhledovou fotografii, nahrát/odebrat fotografie.

Provedení akce:

1. Uživatel klikne na tlačítko k provedení akce.
2. Uživatel vybere akci a potvrdí provedení.
3. Akce je zaznamenána a přidána.

Změna informací:

1. Uživatel rozklikne tlačítko pro editaci informací o záznamu.
2. Uživatel může vyplnit nové jméno záznamu, nahrát novou náhledovou fotografii a následně potvrdit změnu tlačítkem.

Nahrání fotografie záznamu rostliny:

1. Uživatel klikne na záznamu o rostlině na tlačítko k přidání fotografie.
2. Uživatel vyfotí nebo vybere fotografii k nahrání a potvrdí ji.

Odebrání fotografie v galerii:

1. Uživatel zobrazí na záznamu o rostlině galerii.
2. Uživatel zobrazí fotografii, kterou chce odebrat.
3. Uživatel klikne na tlačítko k odebrání fotografie.
4. Je zobrazeno potvrzovací okno, které uživatel potvrdí nebo zruší.

UC14 Zobrazení nadcházejících událostí Zobrazení obrazovky, na které uživatel bude vidět jeho nadcházející události neboli akce jako chronologický seznam. Tato obrazovka bude tzv. domácí obrazovkou (první zobrazená obrazovka po otevření/přihlášení do aplikace).

1. Uživatel zobrazí obrazovku s nadcházejícími událostmi, které jsou zobrazeny jako seznam.
2. Uživatel může scrollovat událostmi a při kliku na událost se dostane k vytvoření/potvrzení naplánované akce konkrétního záznamu o rostlině.

UC15 Posílání notifikací Systém bude uživatelům posílat notifikace. Jedná se o připomínkové notifikace k akcím a notifikace o provedení akce uživatelem.

Notifikace o provedení akce:

1. Uživatel provede akci (například zalije rostlinu).
2. Systém pošle notifikaci všem ostatním členům v týmu (mají-li povolený tento druh notifikace), že uživatel provedl danou akci.

Připomínková notifikace:

1. Uživatel má naplánovanou událost s nastaveným připomínáním pomocí notifikací.
2. V daný okamžik je uživatel notifikován o dané události.

Ověření, zda případy užití pokrývají funkční požadavky, je zobrazeno v tabulce 3.1. Řádky jsou jednotlivé případy užití a sloupce jsou jednotlivé funkční požadavky. Písmeno *X* v buňkách zachycuje, kterému funkčnímu požadavku odpovídá který případ užití.

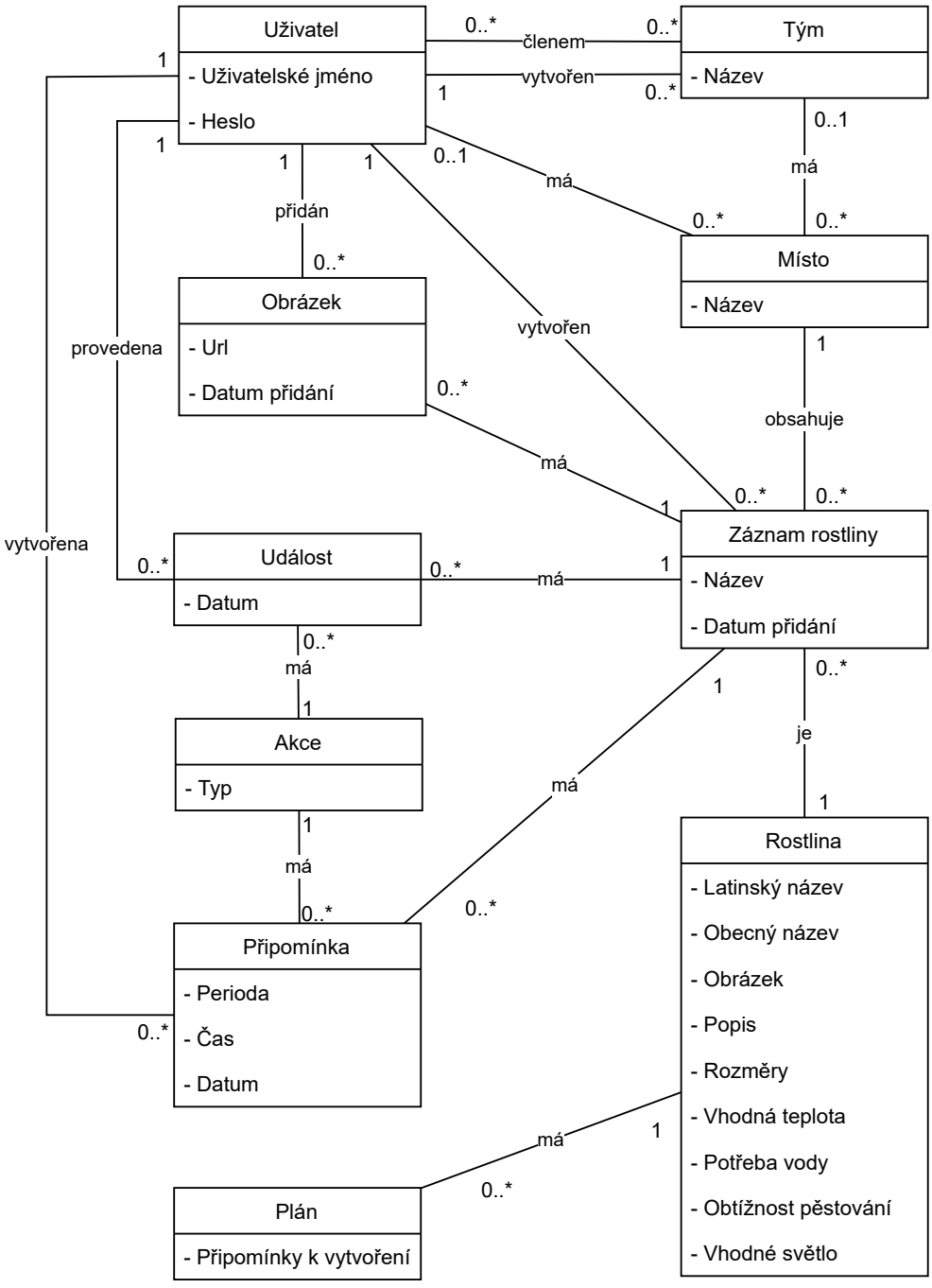
	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10
UC1	X									
UC2	X									
UC3	X									
UC4	X									
UC5		X								
UC6		X								
UC7			X							
UC8			X							
UC9				X						
UC10				X						
UC11					X					
UC12						X				
UC13						X	X	X		
UC14									X	
UC15										X

Tabulka 3.1: Tabulka pokrytí funkčních požadavků

3.3 Doménový model

Často může být obtížné si představit, jak bude problém strukturován a následně zpracován do technologické podoby. Proto se pro prvotní zachycení objektů a jejich vztahů mezi nimi se často používá doménový model. Ten je znázorněn zjednodušeným UML diagramem tříd. Jednotlivé entity jsou zobrazeny jako obdélníky, které mají uvnitř svůj název a důležité atributy. Entity jsou spojeny čárami, které znázorňují vztah mezi nimi. Spojující čáry mohou mít různé typy podle vztahu a na každém konci mají uvedenou multiplicitu. U atributů se nespecifikuje jejich datový typ a celý doménový model by měl být platformě nezávislý. Na obrázku 3.2 se nachází doménový model, který zachycuje doménu aplikace. Následuje popis jednotlivých entit v samotném doménovém diagramu. [21, 22]

Uživatel Jedna z hlavních entit celého systému. U každého uživatele je nutné uchovávat jeho uživatelské jméno a heslo, respektive jeho hash. Noví uživatelé budou přidání po úspěšné registraci. Každý uživatel může vytvořit a být členem jednoho nebo více týmů. Dále k němu patří žádné až



Obrázek 3.2: Doménový model

více osobních míst a záznamů rostlin, které si vytvoří. Dále je také spojen s žádnou nebo více konkrétními událostmi, které dokončil. Nakonec k uživateli mohou patřit jednorázové a opakující se připomínky. Těch může mít více a nebo také žádnou.

Tým Entita, která reprezentuje skupinu uživatelů. Je spojena s konkrétním uživatelem, který daný tým vytvořil. Má jednoho až více členů (uživatelů) a žádné nebo více míst. Každý tým má také svůj název.

Místo Místo je uživatelem vytvořená entita, která označuje, kde se rostlina nachází. Místo má svůj název a je spojen vždy buď s jedním uživatelem nebo jedním týmem. Každé místo pak může obsahovat nula až více záznamů rostlin.

Záznam rostliny Hlavní entita, jež reprezentuje konkrétní rostlinu (ne tedy o jaký druh rostliny se jedná). Každý takový záznam má svůj název, obrázky a datum přidání. Záznam je spjat s jedním konkrétním druhem rostliny (entita Rostlina), místem a uživatelem, který jej vytvořil. Dále je také spojen s žádnou nebo více událostí, jednorázovou a opakující se připomínkou.

Rostlina Entita popisující druh rostliny. K atributům patří latinský a obecný název, její popis, obrázek. Dále také rozměry, teplotní rozpětí (krajní hodnoty intervalu – minimum a maximum), ve kterém je rostlina schopna žít, kolik vody potřebuje, vhodné světlo a obtížnost pěstování.

Událost Tato entita zachycuje jednotlivé události neboli také provedené akce. Má svůj typ/druh (zalití, přesazení atd.) a datum provedení. Je spojena s uživatelem, který ji provedl a s konkrétním záznamem o rostlině.

Připomínka Jedná se o entitu, která popisuje aktivní připomínku. Ta obsahuje akci, kterou připomíná, a datum a čas. Dále může mít specifikovanou periodu, s jakou se připomínka opakuje. Je spárovaná s konkrétní rostlinou a konkrétním uživatelem, který ji vytvořil.

Akce Jednoduchá entita, která znázorňuje akci (typ akce), kterou lze provést na rostlině. Může se vázat k události nebo k připomínce.

Obrázek Jedná se o entitu obsahující informaci o obrázku konkrétního záznamu rostliny. Obsahuje URL obrázku, datum vytvoření a je spojena s konkrétním záznamem rostliny a uživatelem, který obrázek vytvořil/pořídil.

Plán Entita, která slouží pro definici jednotlivých plánů pro danou rostlinu. Každý plán má sadu připomínek na různé akce.

3.4 Architektura a technologie

V této sekci je popsána architektura navrhované aplikace a také technologie, které budou použity pro vývoj a implementaci. Popsána je architektura jako celek a poté jednotlivě každá část aplikace, neboli také vrstva (databáze, serverová část, klientská část).

3.4.1 Třívrstvá architektura

Třívrstvá architektura (anglicky *Three-Tier Architecture*) je v současnosti jedna z nejpoužívanějších architektur ve vývoji softwaru, která vychází z klient-server architektury. Klíčovou vlastností je rozdělení aplikace na tři části (vrstvy), přičemž tyto části nejsou na sobě závislé. Každá část běží odděleně a má přístup jen ke zdrojům, které opravdu potřebuje. Díky nezávislosti může vývoj jednotlivých částí aplikace probíhat zvlášť. Na obrázku 3.3 se nachází zjednodušený diagram, který tuto architekturu popisuje. [23, 24]

Prezentační vrstva Vrstva, se kterou se setkává uživatel. Jedná se o klientské aplikace, které zaznamenávají uživatelské vstupy a na jejich základě zobrazují a prezentují data. Tyto aplikace jsou závislé na platformě, typicky se dnes vyvíjejí webové a mobilní aplikace. Komunikuje s aplikační vrstvou (serverem), ale zároveň nemá přímý přístup k datům v databázi.

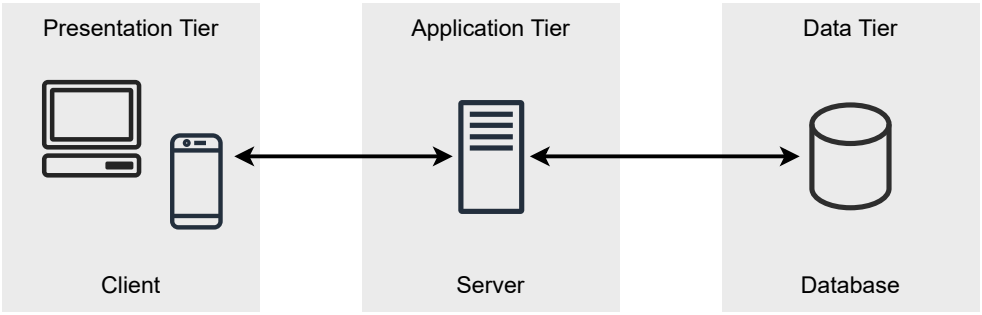
Aplikační vrstva Tato vrstva, také nazývaná jako logická, prostřední, či byznysová vrstva, se stará o hlavní funkce aplikace. Zde probíhají hlavní výpočty a logika nad daty, které napřímo získává a ukládá do databáze. Nabízí rozhraní pro komunikaci s klientskými aplikacemi (prezentační vrstvou).

Datová vrstva Entity starající se o ukládání, řízení, audit a poskytování dat.

Tato architektura bude použita při vývoji aplikace. Prezentační vrstva bude obsahovat klientskou aplikaci pro iOS. Ta bude komunikovat s aplikační vrstvou pomocí REST API. Data poté budou ukládána v databázi, která patří do datové vrstvy.

3.4.2 Datová vrstva

Výběr databázového systému, který bude použit k uložení perzistentních dat, je poměrně přímočarý. Nejdříve je nutné zvolit o jaký typ databáze by se mělo jednat. Vzhledem k datům, které se budou ukládat a zároveň kvůli jejich množství (nejedná se o žádná enormní data), je vhodné použít standardní relační databázi. V tomto segmentu se nachází 4 nejpopulárnější možnosti, jimiž jsou relační databázové systémy Oracle, Microsoft SQL Server, MySQL, PostgreSQL [25]. První dva zmíněné systémy nejsou zástupci otevřeného softwaru (open source), zatímco zbytek ano. Zároveň vzhledem k použití ORM



Obrázek 3.3: Třívrstvá architektura

Prisma (viz 3.4.3.2) odpadá Oracle databáze, jelikož není podporována. Dále už je volba spíše o osobní preferenci a zkušenosti. Ačkoliv je vnitřek každého systému odlišný, používají dotazovací jazyk SQL a pracuje se s nimi obdobně. S ohledem na budoucí nasazení aplikace (na cloudovou platformu Heroku) tak konečná volba padla na PostgreSQL. Zároveň ovšem není problém kdykoliv databázový systém v případě potřeby změnit, a to kvůli využití nástroje Prisma.

3.4.2.1 PostgreSQL

PostgreSQL je pokročilá open source relační databáze, která podporuje dotazování v jazyce SQL (relační) i JSON (nerelační). Jedná se o vysoce stabilní systém pro správu databází, za kterým stojí více než 20 let komunitního vývoje, který přispěl k jeho vysoké úrovni odolnosti, integrity a správnosti. PostgreSQL se používá jako primární datové úložiště nebo datový sklad pro mnoho webových, mobilních, geoprostorových a analytických aplikací. Nejnovější hlavní verzí v době psaní této práce je PostgreSQL 14. PostgreSQL má bohatou historii podpory pokročilých datových typů a vysokou úroveň optimalizace výkonu. Dále také umožňuje běh uložených procedur napsaných v různých programovacích jazycích (Python, Perl, C). [26, 27]

3.4.3 Aplikační vrstva

Technologií, které lze v dnešní době použít k implementaci serverové části aplikace, je nepřeberné množství. Vývojáři si mohou vybrat z různých frameworků a programovacích jazyků s nimi spjatých. Vhodné je, vzhledem k nefunkčnímu požadavku vytvořit REST API rozhraní pro komunikaci, vybrat vhodný framework, který tvorbu REST API umožňuje a co nejvíce usnadňuje. Protože se nejedná a aplikaci výpočetně náročnou, není potřeba se orientovat na takové technologie, které se na rychlost a výkonnost přímo specializují. Důležité je se spíše zaměřit na framework, který umožňuje psát přehledný a znovupoužitelný

kód. Nakonec volba padla na framework *NestJS* [28], který používá prostředí *Node.js*. Dalším nástrojem, který byla zvolen, je *Prisma*, což je technologie pro objektově relační mapování (ORM). *Prisma* usnadňuje práci s daty a dokáže spolu s *NestJS* výborně spolupracovat, neboť obě dvě technologie si zakládají především na typové bezpečnosti (použití programovacího jazyka *TypeScript*).

3.4.3.1 NestJS

NestJS je progresivní framework, pomocí kterého lze vytvářet aplikace na straně serveru. Autorem tohoto frameworku je polský vývojář Kamil Myśliwiec. *NestJS* samotný staví na frameworcích *Express* nebo *Fastify*, a přidává modulární organizaci a širokou škálu dalších knihoven. Jedná se o open-source software, používá programovací jazyk *TypeScript* a struktura kódu výrazně připomíná framework *Angular*. Podporuje tvorbu aplikačního rozhraní REST a GraphQL. Klíčové vlastnosti, o které *NestJS* usiluje, jsou efektivita, spolehlivost a škálovatelnost aplikací. Velkou výhodou je velmi přehledná a propracovaná dokumentace a živá komunita. [28, 29, 30]

Hlavními komponentami NestJS aplikace jsou:

Controller Controllery jsou zodpovědné za zpracování příchozích požadavků a vracení odpovědí klientovi.

Provider Slouží k zapouzdření a abstrakci logiky jiných tříd. Lze je vkládat do jiných tříd pomocí dependency injection (vkládání závislostí).

Module Moduly slouží ke sdružování souvisejících funkcí, controllerů a providerů.

3.4.3.2 Prisma

Prisma je poměrně nová open-source knihovna pro Node.js, která slouží k objektově relačnímu mapování (ORM). Objektově relační mapování je technika, která propojuje relační databázi a objektově orientovaným programovacím jazykem pomocí automatické konverze dat. Umožňuje přistupovat k databázi prostřednictvím metod a objektů daného programovacího jazyka (TypeScript nebo JavaScript v případě knihovny Prisma), aniž by bylo nutné psát dotaz v samotném databázovém jazyce. Prisma se skládá ze tří částí.

Prisma Client Automaticky generovaný a typově bezpečný databázový klient.

Prisma Migrate Výkonný nástroj pro migraci databázových schémat. K popisu databázového schématu používá deklarativní syntaxi. Také ukládá celou historii migrací a umožňuje snadno se vracet a znovu opakovat migrace.

Prisma Studio Nástroj pro zobrazení a editaci dat v prohlížeči. Nabízí moderní UI a snaží se být jednoduchý a přehledný.

Hlavním způsobem, jak interagovat s projektem Prisma z příkazového řádku je použití Prisma CLI (Command Line Interface). Tak lze vytvářet nové prostředky projektu, generovat klienta a analyzovat stávající struktury databáze pomocí introspekce a automaticky vytvářet modely. Knihovna Prisma je přímo připravená pro framework NestJS a výborně spolupracuje s jeho modulární architekturou. [31, 32]

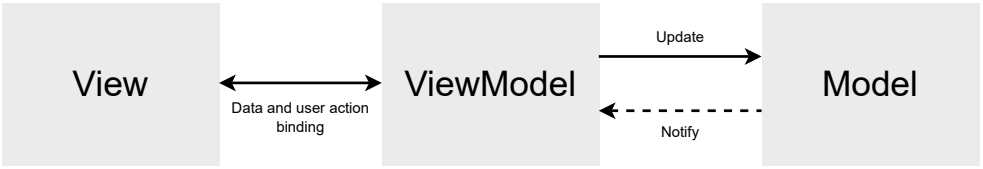
3.4.4 Prezentační vrstva

V současnosti se vývoj mobilních aplikací dělí na nativní a multiplatformní. Nativní mobilní aplikace jsou vytvářeny s nástroji specifickými pro danou platformu. Vzhledem k tomu tak dokážou poskytnout skvělý pocit z používání, protože jsou výkonnější, optimalizované a také používají na míru šité vizuální prvky pro danou platformu. Nevýhodou jsou vyšší náklady (lidské i peněžní) na vývoj, neboť je potřeba vytvořit separátní aplikace pro každou platformu. Proto vznikly frameworky, se kterými lze vytvářet aplikace na více platforem najednou. Mezi tyto frameworky se řadí například *React Native* nebo *Flutter*. Výhodou je jediná *codebase*, což šetří prostředky a urychluje vývoj. Aplikace ale nemusí být optimalizované a mohou vypadat a chovat se lehce jinak na každé platformě. Z důvodu lepší optimalizace/výkonu, použití nativních vizuálních prvků a protože není potřeba implementovat aplikaci na jiné platformy, jsem se rozhodl pro nativní vývoj. [33]

3.4.4.1 SwiftUI

Tradičním a historickým způsobem, jakým lze psát nativní aplikace pro iOS, je použití frameworku UIKit. Ten je podporován pro jazyky Objective C a Swift a kód je psán imperativním způsobem. Stále je hojně využívaný, neboť je odladěný a má za sebou mnoho let vývoje. V roce 2019 Apple ovšem představil nový framework s názvem SwiftUI. Ten nabízí deklarativní psaní kódu a reaktivní přístup. Nějakou dobu nebylo stále vhodné tento framework používat, protože byl dosti problémový a pro reálné aplikace se nedal téměř použít. Po třech letech si ale dovolím tvrdit, že se dostal do stavu, kdy se stal použitelným a v mnoha ohledech dokáže předčit samotný UIKit. A pokud by se ovšem vyskytl nějaký problém s implementací určité věci, lze ji vytvořit v UIKitu a vložit do SwiftUI. [34]

Nejnovější verze operačního systému iOS je v době psaní této práce verze 16. Na tuto verzi zároveň bude implementována výsledná aplikace. Vzhledem k faktu, že Apple poskytuje aktualizace poměrně starým zařízením (5 let staré) a zároveň mezi uživateli je tendence udržovat systém aktuální, omezení se pouze na nejnovější verzi by nemělo být kritické. Použití nejnovější verze



Obrázek 3.4: Architektura MVVM (Model View ViewModel)

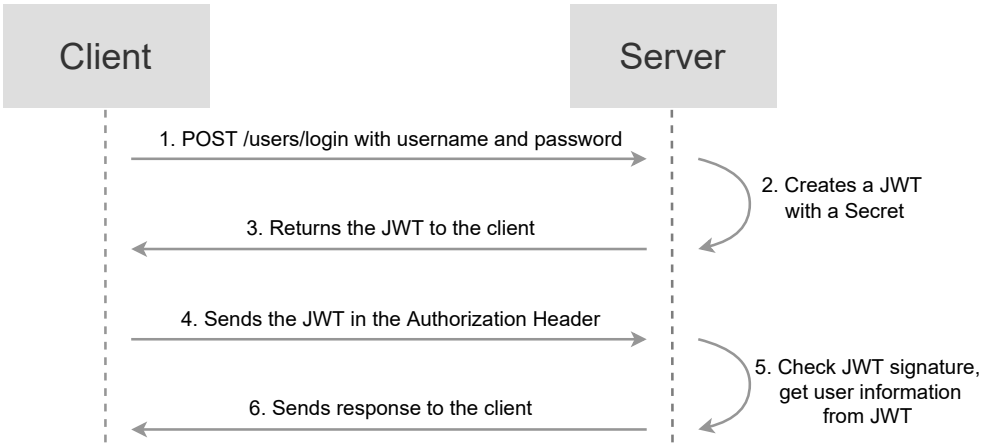
zároveň znamená, že lze použít nově přidané prvky a nástroje. Statistiky obchodu s aplikacemi App Store [35] navíc udávají, že zhruba 8 měsíců po vydání nové verze bývá zvykem, že je právě tato nejnovější verze nainstalována na 82 procentech zařízení.

Dále Apple pomalu upouští od vývoje frameworku UIKit a soustředí si na SwiftUI, který by měl být primárním frameworkem pro tvorbu nativních aplikací pro mobilní zařízení od společnosti Apple. I proto byl také framework SwiftUI zvolen pro následnou implementaci klientské části aplikace.

3.4.4.2 MVVM

Vybranou architekturou pro vývoj iOS aplikace je architektura MVVM (Model View ViewModel). Vychází z architektury MVC (Model View Controller) a zároveň se jí velmi podobá. Hlavním důvodem, proč se používají tyto architektury (a spoustu dalších), je oddělení částí aplikace pro lepší přehlednost a na základě toho i rozšiřitelnost kódu. V případě MVVM se jedná o rozdělení na tři části – Model, View, ViewModel (viz 3.4). Model obsahuje data, se kterými aplikace pracuje. View reprezentuje veškeré prvky uživatelského rozhraní, čili to, jak aplikace vypadá. Uprostřed je ViewModel, který slouží jako prostředník mezi View a Modelem. Ten provádí veškerou dodatečnou logiku a drží stav aplikace. Typicky je View spojen s ViewModelem pomocí konstrukce zvané *binding*. View pak dokáže reagovat na změny v přidruženém ViewModelu a překreslovat tak uživatelské rozhraní. [36]

SwiftUI přímo nestaví na žádné architektuře, nicméně architektura MVVM se výborně hodí pro použití. Ve SwiftUI existují Views, ve kterých se pomocí deklarativního kódu popisují prvky na obrazovce. Samotné View pak drží referenci (a je spojen pomocí *binding* konstrukce) na přidružený ViewModel. Všechny uživatelské vstupy/akce jsou delegovány na ViewModel, kde probíhá samotná logika a v případě potřeby se mění stav aplikace, na kterou reaktivně reaguje View (překreslením změněných prvků). Modely jsou pak jen jednoduché třídy, které uchovávají a zapouzdřují data. S těmi pracuje výhradně ViewModel. [37]



Obrázek 3.5: Interakce mezi klientem a serverem při použití JWT

3.4.5 Autentizace

Protože samotné rozhraní bude na internetu veřejně dostupné, je potřeba zajistit zabezpečení, aby kdokoliv nemohl přistupovat k datům. Prvním způsobem, jak zabezpečit přístup je tzv. *Basic* autentizace. Ta spočívá v poslání uživatelského jména a hesla zakódovaného pomocí Base64 v hlavičce požadavku. Tento způsob je velmi jednoduchý, ale není doporučován z hlediska bezpečnosti. Dále se používá autentizace pomocí API klíče. V hlavičce nebo přímo v URL požadavku je posílán řetězec znaků (klíč) vygenerovaný serverem. Dalším způsobem je *Bearer* autentizace, při které se posílá v hlavičce tzv. access token, který jasně identifikuje tázající se entitu. Tento token vydává sám server a může mít definovanou expirační dobu. Proto se také spolu s tímto tokenem vystavuje refresh token, pomocí kterého lze dostat od serveru nový platný access token. Pokud aplikace používá data aplikace třetích stran, využívá se autentizace pojmenovaná *OAuth 2.0*, jejíž koncept je podobný *Bearer* autentizaci, dochází také k vystavování a ověřování tokenů, ale o toto se nestará samotná aplikace, ale právě třetí strana. [38, 39]

Pro aplikaci byla vybrána *Bearer* autentizace, přičemž samotný token bude řešen jako JWT (JSON Web Token) [40]. Token bude tedy zakódovaná (Base64) trojice hlavička-data-podpis (header-payload-signature). V hlavičce se nachází typ a hashovací funkce, v datech vydavatel, subjekt, čas vypršení a další libovolná data. Podpis je poté vytvořen z hlavičky a dat pomocí specifikované hashovací funkce a tajného klíče. Na obrázku 3.5 je popsána interakce klienta a serveru při použití JWT tokenu.

3.5 Návrh API rozhraní

Z nefunkčních požadavků vyplývá použití architektonického vzoru REST pro návrh API. REST, z anglického *Representational State Transfer*, popisuje pravidla, dle kterých se mají aplikační rozhraní vytvářet. Komunikace poté probíhá tím způsobem, že entita, která chce získat/modifikovat data, pošle požadavek (request) na konkrétní endpoint pomocí HTTP protokolu a následně dostane zpět odpověď. Každý požadavek obsahuje informaci o použité HTTP metodě, v případě dodržení pravidel REST se jedná typicky o metody GET (získání), POST (vytvoření), PUT (modifikace celku), PATCH (modifikace části), DELETE (smazání). Dále také požadavek obsahuje samotný endpoint, HTTP hlavičky a tělo (body), ve kterém jsou obsaženy samotná data. Data lze poslat v různých formátech (formát je specifikován v hlavičce požadavku), mezi nejpoužívanější formáty patří JSON, XML nebo pouze text (plain text). [41, 42]

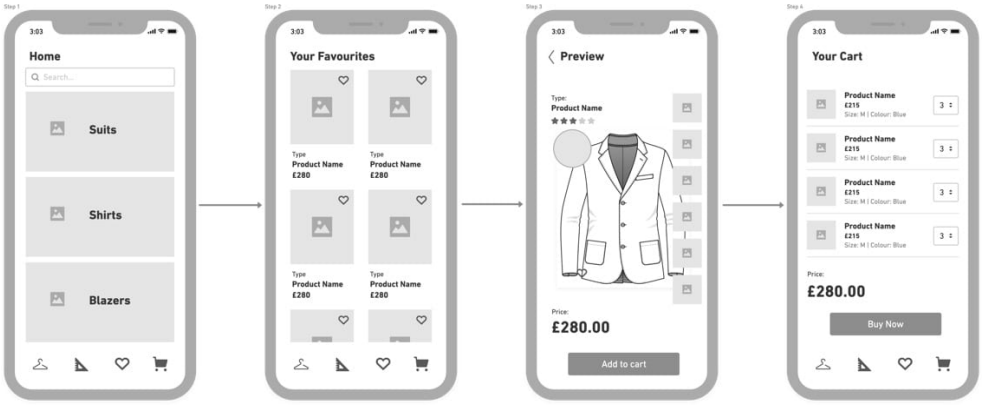
Server by měl podporovat veškeré nutné operace, které bude využívat klientská aplikace (v tomto případě mobilní aplikace). Jedná se tedy o autentizaci, CRUD operace nad jednotlivými entitami a specifické operace (například nastavení plánu).

3.6 Uživatelské rozhraní

Uživatelské rozhraní (zkratka UI) je ve zkratce souhrn všech grafických prvků a jejich uspořádání, které slouží pro komunikaci mezi uživatelem a aplikací či webem (obecně se strojem). S UI úzce souvisí termín UX. UX je zkratka pro uživatelskou zkušenost (User Experience), jde o propojení a interakci grafických prvků, tak aby průchod aplikací byl snadný a intuitivní a aby dosažení požadovaného výsledku vyžadovalo minimální úsilí ze strany uživatele. Dobré UI a UX je stejně tak klíčové, jako samotné funkce dané aplikace. Ty můžou být prvotřídní, ale pokud je aplikace zpracována vizuálně špatně a neintuitivně, uživatel si může vytvořit odpor k jejímu používání. Tento fakt je ještě více umocněn u aplikací, které uživatelé používají primárně dobrovolně (nepoužívají je pro práci, ale ve volném čase). [43]

3.6.1 Prototypování

Tímto termínem je myšlen proces, při kterém jednotlivec či tým designérů realizují svoje nápady do papírové či digitální podoby ve formě tzv. prototypu. Hlavním cílem jeho tvorby je vyzkoušet a otestovat návrh uživatelského rozhraní před vytvořením skutečného produktu, neboť přepracovávat hotový produkt je mnohem nákladnější než jeho prototyp. Pro tvorbu prototypu lze využít různé nástroje a software nebo také pouze tužku a papír. Jednotlivé prototypy mají podle potřeby různý stupeň přesnosti (fidelity) a dělí se na



Obrázek 3.6: Lo-Fi prototyp [46]

Lo-Fi a Hi-Fi. Vytvořené prototypy slouží k prezentaci a konzultaci s klienty a jako podklad pro vývojáře. [44]

3.6.1.1 Lo-Fi prototyp

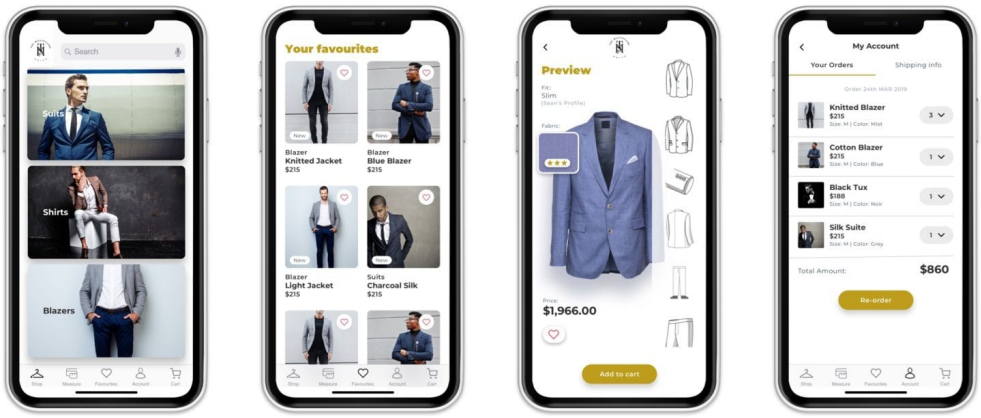
Jedná se o prototypy s nízkou přesností či věrností. Jde o prvotní pohled na budoucí aplikaci a důraz je kladen zejména na rozmístění, typ a velikost grafických prvků. Neřeší se barvy, nevkládají se reálné hodnoty, texty ani obrázky. Lo-Fi prototyp lze vytvořit jen za pomocí papíru a tužky, ale také specializovanými nástroji. Výhodou tohoto prototypu je zejména rychlost zhotovení a jeho nízká cena. Samotný prototyp se může následně dotvořit (nebo může sloužit jako podklad) v Hi-Fi prototyp. Příklad Lo-Fi prototypu je znázorněn na obrázku 3.6. [45]

3.6.1.2 Hi-Fi prototyp

Takový prototyp by měl znázorňovat téměř finální vzhled výsledného produktu (nemusí však být funkční). Zachycuje konkrétní vzhled a barvu prvků, propojení jednotlivých obrazovek, včetně animací a přechodů. Pro jeho tvorbu lze využít mnoha nástrojů nebo prototyp přímo implementovat pro danou platformu. Tento prototyp je na tvorbu časově mnohem náročnější a nákladnější a někdy se lze bez něj obejít. Výhodou je na druhou stranu ovšem zobrazení téměř finální podoby, kterou klient nakonec dostane. Na obrázku 3.7 je zobrazen takovýto prototyp. [45]

3.6.1.3 Volba druhu prototypu

V rámci této závěrečné práce byl zvolen a zhotoven prototyp, který spadá spíše do kategorie Hi-Fi prototypů, ale zároveň se nejedná úplně o jeho plnohod-

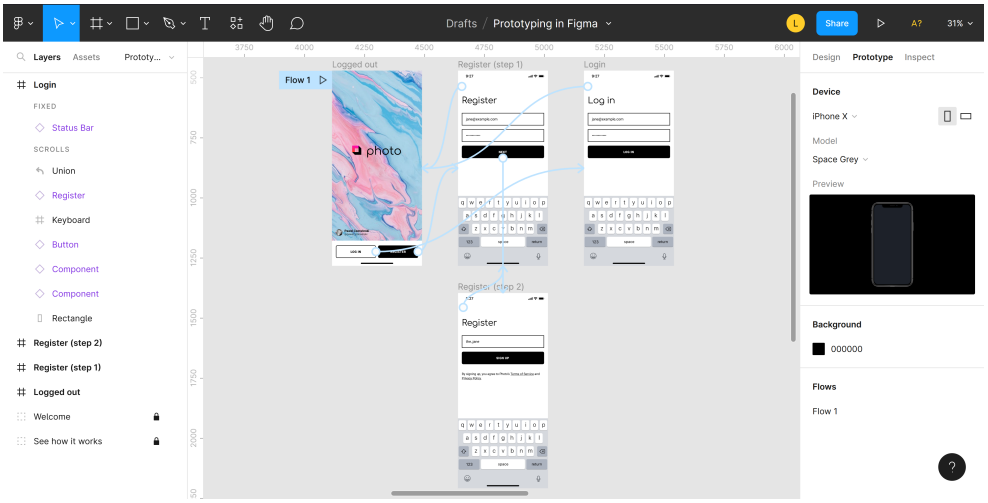


Obrázek 3.7: Hi-Fi prototyp [46]

notného zástupce. Vzhledem k absenci klientské strany, nebylo potřeba dbát tolik na tvorbu dokonalého prototypu. Stačilo sestavit a poskládat grafické prvky do uceleného celku, který bude sloužit jako reference k implementaci. K tomu by postačila tužka a papír nebo nějaký jednoduchý nástroj. Protože jsem ale zvolil nástroj Figma, který je popsán v následující sekci, naskytla se možnost velmi jednoduše vytvořit i klikatelný prototyp. Výsledný prototyp (popsán níže) jsou tedy vzájemně propojené (klikatelné) obrazovky, které obsahují nebarevný nástin grafických prvků.

3.6.2 Figma

Figma je webová aplikace pro tvorbu/editaci grafiky a návrh uživatelského rozhraní. Lze s ní vytvářet nejrůznější grafické počiny od vytváření drátěných modelů webových stránek, přes navrhování rozhraní mobilních aplikací, jejich prototypování až po tvorbu reklamních bannerů. Aplikace funguje přímo v prohlížeči a pro její používání je potřeba mít uživatelský účet. Výhodou toho je, že uživatel má přístup ke svým projektům (ty jsou uloženy v cloudu) z libovolného počítače nebo platformy. Velkou výhodou je také týmová kolaborace v reálném čase na stejném projektu a dále také veliká komunita, kdy každý může přispět sdílením svých projektů (jako inspirace pro ostatní) nebo vytvořením šablony či pluginu. Figma je dostupná komukoli zadarmo s několika omezeními (maximálně 3 projekty, omezený přístup k historii verzí, omezené zabezpečení), které pro nenáročnou práci stačí. Náročnější uživatelé si poté mohou zaplatit za některé z placených plánů. Na obrázku 3.8 se nachází ukázka rozhraní aplikace Figma (na demo projektu, který se věnuje prototypování). [47]



Obrázek 3.8: Screenshot z nástroje Figma

3.6.3 Vytvořený prototyp

Prototyp byl vytvořen, jak již bylo zmíněno v aplikaci Figma. Zjednodušený diagram klíčových obrazovek a jejich přechodů je vyobrazen na obrázku 3.9. Nejsou na něm zachyceny veškeré přechody (například navigace zpět) a některé obrazovky jsou spojeny do jedné. Klikatelný prototyp je dostupný na [48], popřípadě všechny samotné obrazovky (včetně kontextových nabídek) se nachází na přiloženém médiu ve formátu PDF. Ukázku několika obrazovek a kontextových nabídek lze nalézt v obrázku 3.10.

První obrazovkou, se kterou se nový uživatel setká, je přihlašovací obrazovka (Sign In). Na té se může přihlásit a dostat se do aplikace (pokud již má vytvořený účet), nebo se překliknout na registrační obrazovku (Sign Up). Pokud se úspěšně registruje, zobrazí se mu Onboarding, což je krátké seznámení s klíčovými funkcemi aplikace. Po jeho dokončení se dostane do samotné aplikace.

Samotná aplikace se skládá ze 4 hlavních obrazovek, respektive oddělených sekcí. Těmi jsou obrazovky *Tasks*, *My Plants*, *Search Plants* a *Profile*. Mezi těmito sekcemi lze přepínat pomocí dolního panelu, který je známý pod anglickým spojením *Bottom Tab Bar*. Tento panel patří obecně mezi nejoblíbenější navigační prvky mobilních zařízení (iOS i Android) a nachází se ve většině nejpopulárnějších aplikací. Výhodou je jeho poloha, která umožňuje jednoduché ovládání pomocí palce. Ačkoliv se jedná o velmi jednoduchý grafický prvek, je potřeba si dát záležet a nepodcenit jeho návrh. Dolní panel by měl ideálně obsahovat 3 až 5 tlačítek, které neprovádějí akce, ale pouze přepínají mezi oddělenými obrazovkami/sekcemi. Samotná tlačítka by měla mít výstižnou ikonku a pod ní se může nacházet popisek (nutný, pokud není nebo

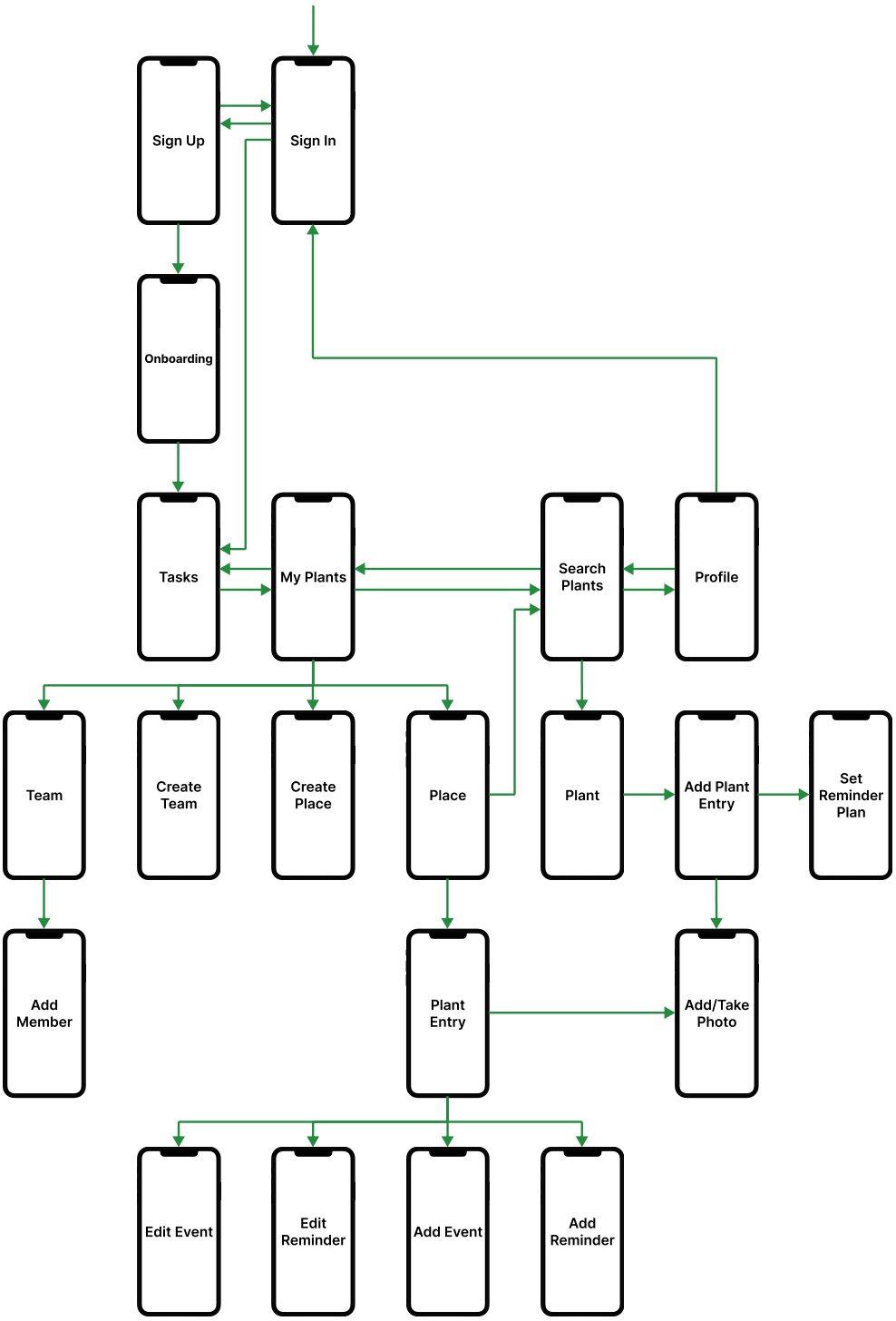
neexistuje dosti reprezentativní ikonka). Také by panel neměl být posunovací a neměl by zbytečně mást uživatele složitými animacemi. [49]

První obrazovkou je obrazovka *Tasks*, na které se nachází seznam nadcházejících úkolů, respektive budoucích událostí, které budou v daný čas připomenuty. V dolním panelu se tato obrazovka nachází úplně nalevo a je reprezentována ikonkou papírového seznamu úkolů. Cílem je zobrazit uživateli jednoduchý přehled úkolů a umožnit mu jednoduše jednotlivé úkoly dokončit.

Obrazovka *My Plants*, kterou reprezentuje ikonka dvou lístků, poskytuje ucelený pohled na uživatelovy rostliny. Na této obrazovce se vpravo nahoře nachází tlačítko pro vytvoření týmu, místa či přidání záznamu rostliny. Níže se pod sebou nachází horizontálně posuvný seznam míst, které přísluší jednotlivým týmům. Do detailu týmu se uživatel dostane kliknutím na jméno týmu, odkud poté může spravovat daný tým. Místo je zobrazeno zaobleným čtvercem, na kterém se nachází až 4 záznamy rostlin, které se nachází v daném místě (jsou k němu přiřazeny). V samotném místě jich ale ovšem může být více než 4, jedná se pouze o náhled. Po posledním místě se vždy nachází tlačítko (zaoblený čtverec se symbolem plus) pro vytvoření místa v daném týmu. Po rozkliknutí místa se zobrazí obrazovka s jeho detailem, na které se nachází příslušné rostliny (jako zaoblené čtverce s fotkou a popiskem) a také lze dané místo spravovat. Po kliknutí na rostlinu se zobrazí obrazovka s detailem záznamu rostliny. Na té se nachází historie událostí/úkonů, nastavené připomínky a galerie. Tlačítkem plus lze přidávat události, připomínky a fotografie (každá tato akce má vlastní obrazovku). Ty lze také poté upravovat a mazat.

Další hlavní obrazovkou (ikonka lupy s malým lístkem) *Search Plants*, na které se nachází vertikálně posuvný seznam druhů rostlin. Po rozkliknutí daného druhu se zobrazí obrazovka s detailem, na které se nachází obrázek, základní informace a tlačítko pro přidání mezi své rostliny (vytvoření záznamu rostliny). Přidávání probíhá na vlastní obrazovce, na které se nachází položky pro vyplnění či vybrání uživatelem (obrázek/fotografie, jméno, tým a místo). Po této obrazovce následuje obrazovka s nastavením doporučeného připomínkového plánu.

Poslední obrazovkou je *Profile* (ikonka profilu, neboli hlava s půlkou těla). Ta slouží k zobrazení informací o přihlášeném uživateli. Dále nabízí nastavení notifikací, měrných jednotek (metrické, imperiální). Na této obrazovce se lze z účtu odhlásit nebo ho smazat.



Obrázek 3.9: Zjednodušený diagram přechodů obrazovek

Obrázek 3.10: Ukázka obrazovek a kontextových nabídek vytvořeného prototypu

Kapitola 4

Realizace

V této kapitole je popsána realizace aplikace (vycházející z návrhu). Je rozdělena do dvou sekci, první se věnuje serverové části a druhá části klientské, tedy mobilní aplikaci pro iOS. Nejprve byla implementována serverová část aplikace se základními koncovými body pro komunikaci (REST API). Ty umožňovali autentizaci a CRUD operace nad jednotlivými entitami, respektive jejich vytvoření, čtení, úprava a mazání. Následně byla podle navrženého uživatelského rozhraní vytvořena mobilní aplikace, přičemž byly v případě potřeby vytvářeny nové koncové body či dělány další menší úpravy v serverové části.

Vývoj probíhal s využitím řady nástrojů, jejichž popis se nachází níže.

Visual Studio Code Jedná se o bezplatný open source editor zdrojových kódů postavený na frameworku Electrum (multiplatformní desktopové aplikace), který vyvíjen společností Microsoft. V základu se jedná o jednoduchý editor, který ale podporuje debuggování, zvýrazňování syntaxe, automatické doplňování či refaktoring. Síla tohoto editoru však tkví v podpoře doplňků (extensions), které lze pár kliky instalovat a vytvořit tak téměř plnohodnotné vývojové prostředí (IDE). Dle výsledků ankety Developer Survey pro rok 2022 [50], kterou každoročně zpracovává webová stránka Stack Overflow, jde o nejpoužívanější IDE mezi vývojáři s podílem 74,48% [51, 52]

Xcode Pro vývoj nativních aplikací pro zařízení společnosti Apple zde existuje pouze jediné vývojové prostředí a tím je Xcode. Toto vývojové prostředí bylo vytvořeno a je spravováno společností Apple a je ho možné nainstalovat a spustit pouze na zařízeních s operačním systémem macOS. Nabízí všechny potřebné nástroje a prostředky pro vývoj software na většinu zařízení od společnosti Apple (vývoj pro iOS, iPadOS, macOS, watchOS, tvOS). [53]

Docker Docker je open-source nástroj pro kontejnerizaci (virtualizace na úrovni operačního systému), doručení a spouštění aplikací na různých

platformách. Docker byl v rámci této závěrečné práce použit pro vytvoření a spuštění kontejneru s PostgreSQL databází pro vývoj a pro end-to-end testování. [54, 55]

Insomnia Jde o open source API klienta, který umožňuje vytvářet, posílat a testovat REST, SOAP, GraphQL a gRPC dotazy. Další součástí je i editor pro tvorbu API dokumentace ve formátu OpenAPI. Tento nástroj byl využit k ručnímu testování REST dotazů v průběhu vývoje. [56]

DataGrip Nástroj DataGrip od společnosti JetBrains, je integrované vývojové prostředí pro databáze a SQL, s jehož pomocí se lze připojit k databázi, psát dotazy, prohlížet data, importovat nebo exportovat data a provádět všechny potřebné manipulace. Vedle nástroje Prisma Studio (prohlížeč databáze, který je součástí Prisma ORM) byl DataGrip použit pro prohlížení a úpravu dat v databázi během vývoje. [57]

Git Git je distribuovaný open source systém pro správu verzí, který umožňuje ukládat kód, sledovat historii revizí, slučovat změny kódu a v případě potřeby se vracet k dřívějším verzím kódu. Kód byl za pomoci tohoto systému verzován a samotný repozitář byl uložen na serveru Github. [58]

4.1 Serverová část

Jak již bylo zmíněno v návrhu, serverová část byla naimplementována ve frameworku NestJs, přičemž pro mapování persistentních dat a práci s nimi byla využita open source knihovna Prisma. Výsledná souborová struktura této části aplikace je popsána v 4.1. Níže v této sekci se nachází popis dalších využitých knihoven, datového schématu, jednotlivých modulů, klíčových mechanismů a detaily ohledně finálního nasazení.

4.1.1 Podpůrné knihovny

Vedle knihoven, se kterými v základu přichází framework NestJs, bylo využito i několik dalších knihoven, které usnadňují a řeší určité aspekty aplikace. Jedná se zejména o knihovny usnadňující zabezpečení, validaci dat a odesílání notifikací. Zde se nachází jejich seznam a krátký popis.

argon2 Balíček, pomocí kterého lze hashovat data hashovací funkce Argon2, která shrnuje nejnovější poznatky v oblasti návrhu paměťově náročných funkcí. Tato hashovací funkce byla prohlášena za vítěze soutěže Password Hashing Competition [59] v červnu roku 2015 a jejími tvůrci jsou výzkumníci z Lucemburské univerzity. [60]

```
server ..... adresář se soubory serverové části
├── prisma ..... adresář se soubory vztahující se ke knihovně prisma
│   ├── migrations ..... adresář s migračními skripty
│   ├── schema.prisma ..... soubor s definicí databázového schématu
│   └── seed.ts ..... skript pro naplnění databáze
├── src ..... adresář se zdrojovými kódy samotného NestJs projektu
│   ├── decorators ..... adresář s vlastními dekorátory
│   ├── middleware ..... adresář s vlastním middleware
│   ├── modules ..... adresář s moduly
│   └── main.ts ..... hlavní spouštěcí soubor
├── test ..... adresář se sadou e2e testů
├── .env ..... proměnné prostředí
├── .eslint.js ..... ESLint konfigurační soubor
├── .gitignore ..... soubory ignorované verzovacím systémem git
├── .prettierrc ..... Prettier konfigurační soubor
├── docker-compose.yml ..... soubor s definicemi docker kontejnerů
├── nest-cli.json ..... NestJs konfigurační soubor
├── package-lock.json .... soubor s přesnými informacemi o závislostech
├── package.json ..... manifest závislostí projektu
├── Procfile ..... instrukce pro spuštění na platformě Heroku
├── README.md ..... základní informace
├── tsconfig.build.ts ... konfigurační soubor pro TypeScript (sestavení)
└── tsconfig.ts ..... konfigurační soubor pro TypeScript
```

Obrázek 4.1: Souborová struktura serverové části

node-apn Jedná se o balíček, který podporuje komunikaci a posílání notifikací skrze APNs (Apple Push Notification service). Byl použit komunitně spravovaný fork originálního balíčku, neboť ten již není aktivně vyvíjen. [61]

class-transformer, class-validator Balíčky pro transformaci plain objektů na instance tříd a validaci proměnných tříd pomocí dekorátorů. Tyto balíčky následně využívá NestJs ve svých *Validation Pipes*, které zajišťují automatickou validaci jednotlivých dotazů. [62, 63]

helmet Helmet je kolekcí menších middlewarových funkcí pro aplikace založené na Express.js (s nímž NestJs v základu pracuje), které nastavují HTTP hlavičky. Tím lze ochránit aplikace před některými známými webovými zranitelnostmi. [64]

parse-duration Balíček pro konverzi řetězců popisující dobu trvání (v lidsky čitelné podobě) na milisekundy. Příkladem buď například výraz „1h“ nebo „30m 10s“. [65]

passport Autentizační middleware pro Node.js, kompatibilní s Express.js. Jeho účelem je ověřování požadavků, které provádí prostřednictvím pluginů, které jsou známy pod pojmem strategie. Mezi ně patří například přihlašování uživatelským jménem a heslem, pomocí OAuth nebo OpenID či autentizaci skrz tokeny.

pactum Balíček pro automatizaci testování REST API vhodný pro nejrůznější druhy testů, včetně integračních či end to end. [66]

4.1.2 Databázové schéma

V souboru *schema.prisma* byly popsány jednotlivé modely (entity) a vztahy mezi nimi. Následně byl z tohoto schématu pomocí Prisma vygenerován migrační skript, který dokáže zanést dané schéma do zvolené databáze (vytvoří příslušné tabulky). Pokud by bylo potřeba udělat ve schématu nějaké změny, je možné vygenerovat další migrační skript, který dokáže aplikovat pouze změny nového schématu oproti starému. Daná migrace pak byla aplikována na instanci PostgreSQL databáze, která běžela v Docker kontejneru na mém počítači. Současně s aplikováním migrace byl vygenerován klient, který na základě daného schématu obsahuje rozhraní (včetně typů), pomocí kterého se lze připojit k databázi a provádět databázové operace s definovanými modely (entitami). Výsledná struktura databáze PostgreSQL po aplikování migrace je zobrazena na obrázku 4.2.

4.1.3 Moduly

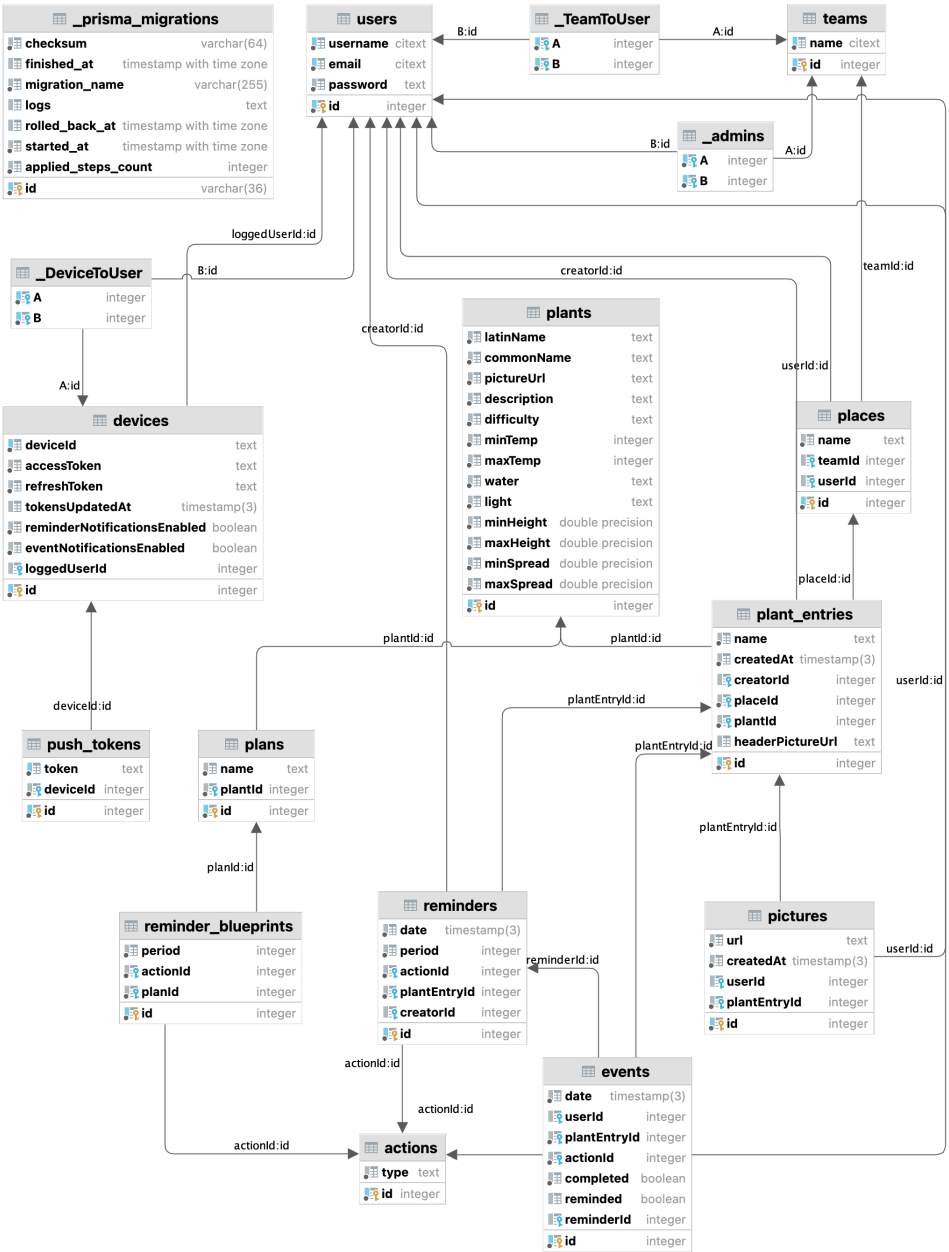
Aplikace používá celkem 14 modulů. Hlavním modulem je *App*, který obsahuje veškeré zbylé moduly. Jeho součástí je zaregistrování již připraveného modulu *Schedule*, který zprostředkovává funkcionality kolem pravidelného volání funkcí. Ten je poté využit k posílání připomínkových notifikací. Vedle hlavního *App* modulu běží modul *Swagger*, který automaticky generuje OpenAPI dokumentaci z kódu. Ta se nachází na cestě `/docs`. Níže jsou popsány moduly, které jsou součástí hlavního modulu *App*.

4.1.3.1 Prisma

Klíčový modul, který zprostředkovává ostatním modulům *PrismaService*, což je provider umožňující použití vygenerovaného Prisma klienta. Jeho definici lze vidět v kódu 4.1. PrismaService dědí od třídy PrismaClient (vygenerovaný klient) a implementuje OnModuleInit (v metodě onModuleInit, která je volána při inicializaci modulu, dojde k připojení klienta do databáze).

4.1.3.2 Auth

Modul zodpovědný za autentizaci uživatelů. Podle návrhu byla implementována autentizace pomocí JWT, kdy po úspěšné registraci či přihlášení je vy-



Obrázek 4.2: Struktura databáze

```
import { INestApplication, Injectable, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';

@Injectable()
export class PrismaService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
  }

  async enableShutdownHooks(app: INestApplication) {
    this.$on('beforeExit', async () => {
      await app.close();
    });
  }
}
```

Výpis kódu 4.1: Třída PrismaService

generována dvojice tokenů, které opravňují uživatele dotazovat se na zbylých koncových bodech. Tyto tokeny jsou uloženy do databáze k příslušnému zařízení (device). Byly vytvořeny 2 autentizační strategie *JwtAccessTokenStrategy* a *JwtRefreshTokenStrategy*, které dědí od již připravené třídy PassportStrategy a implementují metodu *validate*. Ta definuje, zda má dotaz v *Authorization* hlavičce validní access, respektive refresh token. Tyto strategie jsou následně použity v třídách *JwtAccessTokenGuard* a *JwtRefreshTokenGuard*, které umožňují controllerům pomocí dekorátoru specifikovat, že se má použít daná autentizace. Zpřístupněny jsou tyto koncové body:

POST /auth/register Registrace nového uživatele.

POST /auth/login Přihlášení uživatele.

POST /auth/refresh Obnova tokenu.

POST /auth/logout Odhlášení uživatele (zneplatnění tokenů).

4.1.3.3 User

Poskytuje rozhraní pro správu uživatelů. Provider *UserService* je využit v modulu Auth pro vytváření uživatelů. Controller v tomto modulu spravuje tyto koncové body:

GET /users Získání všech uživatelů.

GET /users/{id} Získání konkrétního uživatele.

DELETE /users/delete Smazání uživatele (id uživatele, který bude smazán, se nachází v access tokenu).

4.1.3.4 Team

Zprostředkovává správu týmů, včetně vytváření, získávání, editace, mazání, přidávání a odebírání členů, udělování a odebírání práv. Dostupné endpointy jsou:

POST /teams Vytvoření nového týmu.

GET /teams Získání všech týmů, jichž je přihlášený uživatel členem.

GET /teams/summary Získání souhrnu všech týmů, jichž je přihlášený uživatel členem. Tento souhrn obsahuje přímo i informace o místech a záznamech rostlin. Tento endpoint slouží pro získání všech potřebných informací v mobilní aplikace na obrazovce „My Plants“.

GET /teams/{id} Získání konkrétního týmu.

PUT /teams/{id} Editace konkrétního týmu.

DELETE /teams/{id} Smazání konkrétního týmu.

GET /teams/{id}/members Získání všech členů (uživatelů) konkrétního týmu.

POST /teams/{id}/members Přidání uživatele mezi členy týmu.

DELETE /teams/{id}/members/{id} Odebrání uživatele z týmu.

POST /teams/{id}/members/give-admin-rights Přidání uživatele mezi členy týmu.

POST /teams/{id}/members/remove-admin-rights Přidání uživatele mezi členy týmu.

4.1.3.5 Place

Modul pro spravování míst. Protože místo může patřit buď samotnému uživateli nebo týmu, bylo potřeba vytvořit dva separátní koncové body pro vytváření. Pomocí jednoho lze vytvořit místo patřící ke konkrétnímu uživateli (osobní) a pomocí druhého místo, které patří týmu. Koncovými body jsou:

POST /places/user Vytvoření nového místa pro uživatele.

POST /places/team Vytvoření nového místa pro tým.

GET /places Získání všech míst.

GET /places/{id} Získání konkrétního místa.

PUT /places/{id} Editace konkrétního místa.

DELETE /places/{id} Smazání konkrétního místa.

4.1.3.6 Plant

Jednoduchý modul pro získávání různých druhů rostlin. V odpovědích jsou posílány i jednotlivé připomínkové plány. Endpointy jsou následující:

GET /plants Získání všech rostlin.

GET /plants/{id} Získání konkrétní rostliny.

4.1.3.7 PlantEntry

Modul pro správu záznamů rostlin. Základní operace s danou entitou plus endpoint na nastavení připomínkového plánu. Koncové body:

POST /plant-entries Vytvoření záznamu rostliny.

GET /plant-entries Získání všech záznamů rostlin.

GET /plant-entries/{id} Získání konkrétního záznamu rostliny.

PUT /plant-entries/{id} Editace konkrétního záznamu rostliny.

DELETE /plant-entries/{id} Smazání konkrétního záznamu rostliny.

POST /plant-entries/{id}/setPlan Nastavení připomínkového plánu konkrétnímu záznamu rostliny.

4.1.3.8 Event

Tento modul řeší vytváření, získávání, úpravu a mazání událostí. Má přístup k modulům Prisma, User (vyhledání uživatelů, pro které je daná událost relevantní) a Notification (posílání notifikací). Zpřístupněnými koncovými body jsou:

POST /events Vytvoření nové události, která je označená jako provedená. Při vytvoření nové události je poslána notifikace ostatním členům v týmu, kteří mají povolené *event* notifikace.

GET /events Získání událostí. Query parametrem lze vyfiltrovat pouze dokončené nebo nedokončené události.

GET /events/{id} Získání konkrétní události.

PUT /events/{id} Editace konkrétní události.

DELETE /events/{id} Smazání konkrétní události.

4.1.3.9 Reminder

Tento modul slouží pro vytváření, získávání, úpravu a mazání připomínek. Zpřístupněné endpointy jsou:

POST /reminders Vytvoření připomínky.

GET /reminders Získání všech připomínek, které patří přihlášenému uživateli.

GET /reminders/{id} Získání konkrétní připomínky.

PUT /reminders/{id} Editace konkrétní připomínky.

DELETE /reminders/{id} Smazání konkrétní připomínky.

4.1.3.10 Picture

Vytváření, získání, úprava a mazání obrázků. Jednotlivé obrázky se nahrávají na cloudové úložiště Firebase Storage přímo v klientské části aplikace a zde se vytváří pouze záznam s odkazem na zdroj, kde je obrázek uložen. Jednotlivé koncové body jsou následující:

POST /pictures Vytvoření nového obrázku záznamu rostliny.

GET /pictures Získání všech obrázků.

GET /pictures/{id} Získání konkrétního obrázku.

DELETE /pictures/{id} Smazání konkrétního obrázku.

4.1.3.11 Notification

Modul, který zprostředkovává nastavení a posílání notifikací uživatelům. Provider *NotificationService* poskytuje funkci *sendEventNotification*, která umožňuje poslat event notifikaci. Tuto funkci poté využívá provider *EventService* v *Event* modulu. Také je zde definována pravidelná úloha (Cron) *handleReminderNotifications*, která se stará o posílání připomínkových push notifikací. Kontrola připomínek probíhá každou minutu. Notifikace je poslána pomocí balíčku *node-apn* Kód této úlohy je vidět na 4.2. Zároveň jsou poskytnuty následující endpointy:

POST /notifications/addPushToken Přidání „push“ tokenu, který je unikátní pro každé iOS zařízení a je zřízeno po zaregistrování daného zařízení k push notifikacím.

POST /notifications/enable-reminder-notifications Zapnutí připomínkových notifikací.

```
@Cron('0 * * * * *') // every minute
async handleReminderNotifications() {
  for (let event of await this._getEventsToRemind()) {
    let usersLinkedToEvent = await this._getUsersLinkedToEvent(event.id);
    await this._sendNotificationToUsers(
      usersLinkedToEvent,
      'Sprinkled',
      'It is time to ' + event.action.type.toLowerCase()
      + ' the ' + event.plantEntry.name + '.',
      NotificationType.REMINDER,
    );
    await this._markEventAsReminded(event.id);
  }
}
```

Výpis kódu 4.2: Pravidelná úloha, která posílá připomínkové notifikace

POST /notifications/disable-reminder-notifications Vypnutí připomínkových notifikací.

POST /notifications/enable-event-notifications Zapnutí notifikací o provedení události.

POST /notifications/disable-event-notifications Vypnutí notifikací o provedení události.

4.1.4 End to end testy

Pro kontrolu fungující aplikace, byla vytvořena základní sada end to end testů (E2E), která testuje autentizaci a CRUD operace nad entitou user. End to end testy kontrolují, jestli aplikace funguje jako celek tak, za jakým účelem byla vytvořena (testuje komplexní procesy z pohledu uživatele). V kontextu implementované aplikace (serverová část) se tedy jedná o správné odbavování požadavků a vracení správných odpovědí pro dané vstupy.

Pro toto testování je tedy nutné spustit aplikaci včetně databáze, do které má během testů přístup. Testy používají balíček *Jest* [67], který je výchozím testovacím frameworkem NestJs projektu, a dále také balíček *PactumJS* (viz 4.1.1) pro jednoduché a efektivní testování samotného rozhraní REST. Testovací databáze ve formě Docker kontejneru je vytvořena před samotným během testů a ta je navíc naplněna testovacími daty. Příkladem testu, který testuje správnou registraci uživatele, buď kód 4.3.

4.1.5 Nasazení

Pro nasazení aplikace do ostrého provozu byla použita platforma jako služba (PaaS) Heroku [68]. Aplikace jsou nasazovány na virtuální počítače (na stroje

```
describe('Register', () => {
    it('should register a new user', () => {
      const dto: CreateUserDto = {
        username: 'newUser',
        email: 'newUser@gmail.com',
        password: 'passw0rd',
        deviceId: '06ab9f3b-302e-4cf3-93f1-8549e242caf4',
      };
      return pactum
        .spec()
        .post('http://localhost:3001/auth/register')
        .withJson(dto)
        .expectStatus(201)
        .expectJsonMatch({
          id: 3,
          username: 'newUser',
          accessToken: regex(JWT_TOKEN_REGEX),
          refreshToken: regex(JWT_TOKEN_REGEX),
        });
    });
});
```

Výpis kódu 4.3: Příklad testu ověřující validní registraci uživatele

poskytované Amazon Web Services), v Heroku terminologii zvané jako *dynos*. Tyto virtuální počítače mají přiděleny prostředky (výkon) podle plánu, který si uživatel zvolí (a zaplatí). Protože se jedná o PaaS, není potřeba se starat o infrastrukturu strojů, na samotných *dynos* je už připravené prostředí pro spuštění aplikací nejrůznějších technologií. Stačí doručit kód (například skrze GitHub) a specifikovat instrukce (Procfile), které aplikaci spustí. Dalším důvodem, mimo jednoduchost nasazení, je také fakt, že Heroku nabízí i hosting PostgreSQL databáze. Heroku tedy stačí na kompletní nasazení aplikace a databáze.

Do 28. listopadu 2022 nabízela služba Heroku omezené bezplatné plány, které stačily na základní experimenty či malé projekty. Z finančních důvodů však tyto plány byly odstraněny a nahrazeny novými placenými (avšak cenově dostupnými) plány. Heroku zároveň ale podporuje studenty a nabízí jim kredity v ceně 156 dolarů (ekvivalent 13 kreditů za každý měsíc po dobu jednoho roku), které lze jakkoliv využít na různé plány. Aby daný student kredity dostal, je potřeba se ověřit GitHub účtem, který má přístup k *Github Student Developer* balíku. Tento balík lze získat propojením GitHub účtu s univerzitním účtem. Tato nabídka byla využita a bylo zřízeno *dyno* (v plánu basic) na provoz serverové části aplikace a PostgreSQL databáze (v plánu mini). V kontextu serverové části byly nastaveny veškeré proměnné prostředí k úspěšnému připojení k produkční PostgreSQL databázi (url databáze), posílání notifikací skrze APNs (bundle id, key, team id, druh prostředí) a také byly nastaveny expirační doby access (1 hodina) a refresh (7 dní) tokenu. Databáze byla také

zároveň naplněna celkem šestnácti druhy rostlin.

V GitHub repozitáři byl v rámci GitHub Actions zřízena akce pro spuštění jednotkových (unit) a E2E testů a následného nasazení aplikace na Heroku (na již vytvořené a připravené *dyno*). Tato akce se spouští vždy při odeslání (push) kódu na hlavní větev repozitáře nebo manuálně kliknutím na webu. Akce (jobs) jsou definovány v souboru *backend-test-and-deploy.yml*, který je umístěn v adresáři *.github/workflows*. Tento adresář se musí nacházet v kořeni repozitáře, aby GitHub konfigurace akcí rozeznal.

4.2 Klientská část

Aplikace byla implementována dle návrhu ve frameworku SwiftUI s použitím několika dalších knihoven. Vzhled aplikace vychází ze zhotoveného Hi-Fi prototypu, ve kterém byly provedeny místy menší úpravy a vylepšení. Souborová struktura aplikace se nachází na 4.3.

```
client-ios ..... adresář se soubory klientské části
├── sprinkled-ios ..... samotná mobilní aplikace
│   ├── Assets ..... adresář s prostředky (obrázky, barvy, ikony)
│   ├── Services ..... adresář se službami vkládané pomocí DI
│   ├── States ..... adresář se stavy předávané jako objekty prostředí
│   ├── Model ..... adresář s Modely
│   ├── ViewModel ..... adresář s ViewModely
│   ├── View ..... adresář s obrazovkami (Views)
│   ├── AppDelegate.swift .... třída pro zpracovávání událostí životního
│   │                          cyklu aplikace
│   ├── Errors.swift ..... chyb (výjimky)
│   ├── Extensions.swift ..... rozšíření
│   ├── ImagePicker.swift ..... obrazovka pro výběr obrázků
│   ├── LaunchScreen.storyboard ..... načítací obrazovka
│   ├── sprinkled-ios.entitlements ..... oprávnění
│   ├── SprinkledApp.swift ..... hlavní třída aplikace
│   ├── Styles.swift ..... styly
│   ├── TestData.swift ..... testovací data
│   ├── UIElements.swift ..... grafické prvky
│   └── Utils.swift ..... pomocné funkce
├── sprinkled-ios-tests ..... jednotkové testy
└── sprinkled-ios.xcodeproj ..... soubor s konfigurací projektu
```

Obrázek 4.3: Souborová struktura klientské části

4.2.1 Podpůrné knihovny

Aplikace používá několik knihoven, které usnadňují práci, umožňují používat externí služby či přidávají grafické prvky nebo funkcionality. Těmito knihovnami jsou:

Kingfisher Knihovna pro práci se vzdálenými obrázky. Umožňuje stahování, zobrazování a ukládání obrázků z webu do mezipaměti. Výhodou je, že již stažené obrázky se nemusí opakovaně stahovat, pokud se v mezipaměti nachází. Samotná konfigurace ukládání a limity lze přizpůsobit, ale v mnoha případech stačí použít výchozí nastavení. [69]

Firebase iOS SDK Sada komponent pro práci s *Firebase*. V rámci *Firebase* byl využit nástroj *Firebase Storage* pro ukládání obrázků/fotografií.

Swift Collections Balíček (o který se stará samotný Apple), který obsahuje několik doplňujících datových struktur, které nejsou součástí *Swift Standard Library*. Jedná se o obousměrnou frontu, seřazenou množinu a slovník (v nadcházejících verzích i například bitová množina a pole či prioritní fronta). Právě seřazený slovník je využit na obrazovce *Tasks*, ve kterém jsou uchovány nadcházející úkoly. [70]

JWTDecode Knihovna pro dekódování JWT tokenu. Z textového řetězce je vytvořen objekt, ze kterého lze jednoduše získávat informace obsažené v tokenu. [71]

PopupView Knihovna, která poskytuje několik druhů vyskakovacích a plovoucích oken, které lze různě konfigurovat. S pomocí této knihovny byly vytvořeny chybové a potvrzovací hlášky. [72]

SwiftUI-Shimmer Jednoduchý View modifikátor, který danému View přidá „blyštící“ se animaci, která se výborně hodí k reprezentaci načítajících se grafických prvků (spolu s použitím modifikátoru *redacted*). [73]

4.2.2 Dependency Injection

Dependency Injection (Vkládání závislostí) je technika pro předávání závislostí mezi třídami. Tato technika je svázána s návrhovým vzorem Inversion of Control, ve kterém je cílem uvolnit pevné vazby mezi objekty. V dependency injection existuje tzv. injektor, což je třída, která je zodpovědná za konstrukci, uchovávání a následnému vkládání instancí tříd (*services*) jiným třídám.

Velice jednoduchá a elegantní implementace tohoto mechanismu byla převzata z [74]. Struktura `DependencyInjector` 4.4 obsahuje slovník, do kterého jsou ukládány jednotlivé instance objektů, které je potřeba vkládat jiným objektům. Dále obsahuje metody `register` pro registraci instancí (vloží ji do slovníku pod klíčem jména třídy) a `resolve` pro jejich získávání. Vedle této

```
struct DependencyInjector {
  private static var dependencyList: [String:Any] = [:]

  static func resolve<T>() -> T {
    guard let t = dependencyList[String(describing: T.self)] as? T else {
      fatalError("No provider registered for type \(T.self)")
    }
    return t
  }

  static func register<T>(dependency: T) {
    dependencyList[String(describing: T.self)] = dependency
  }
}
```

Výpis kódu 4.4: Třída DependencyInjector, která spravuje závislosti.

```
@propertyWrapper struct Inject<T> {
  var wrappedValue: T

  init() {
    self.wrappedValue = DependencyInjector.resolve()
  }
}

@propertyWrapper struct Provider<T> {
  var wrappedValue: T

  init(wrappedValue: T) {
    self.wrappedValue = wrappedValue
    DependencyInjector.register(dependency: wrappedValue)
  }
}
```

Výpis kódu 4.5: Vytvoření obalení proměnných (property wrappers) pro jednoduchou registraci služeb (services) a vkládání závislostí.

struktury jsou definována dvoje obalení proměnné (property wrappers) 4.5 – `Inject` a `Provider`, které tyto metody používají, a slouží právě k samotné registraci a vkládání závislostí.

V aplikaci je pak vkládání závislostí pomocí této implementace řešeno tak, že po úspěšném spuštění aplikace jsou vytvořeny a zaregistrovány (v třídě `AppDelegate`) potřebné služby (services) pomocí obalení proměnné `Provider`. Následně v jednotlivých ViewModelech, které danou závislost potřebují, je vytvořena proměnná a přidáno obalení `Inject`. Tím je do dané proměnné vložen objekt ze seznamu závislostí v injektoru a lze s ním jakkoliv nakládat.

Služby, které byly vytvořeny a jsou takto dostupné vkládáním, jsou tyto:

API Slouží pro komunikaci se serverovou částí aplikace pomocí REST API. Posílá HTTP požadavky a v odpovědích získává data ke zobrazení.

```
protocol StorageManagerProtocol {
  func uploadImage(image: UIImage) async throws -> URL
}

final class StorageManager: StorageManagerProtocol {
  let storage = Storage.storage()

  func uploadImage(image: UIImage) async throws -> URL {
    let storageRef = storage.reference().child("images/\(UUID()).jpg")
    let data = image.jpegData(compressionQuality: 0.2)
    guard let data else {
      throw StorageManagerError.conversionError
    }
    let metadata = StorageMetadata()
    metadata.contentType = "image/jpg"
    _ = try await storageRef.putDataAsync(data, metadata: metadata)
    return try await storageRef.downloadURL()
  }
}
```

Výpis kódu 4.6: Protokol a třída NotificationManager, která umožňuje ukládat obrázky na cloudové uložiště Firebase Storage.

StorageManager Služba pro ukládání obrázků/fotografií na cloudové uložiště Firebase Storage. Obsahuje jedinou metodu `uploadImage`, která komprimuje a nahraje obrázek ve formátu JPEG a vrátí URL adresu, na které je daný obrázek dostupný. Kód lze vidět na 4.6.

NotificationManager Služba, která spravuje notifikace. Pomocí této služby lze požádat o povolení notifikací a také zapínat a vypínat jednotlivé druhy notifikací (připomínkové a událostní).

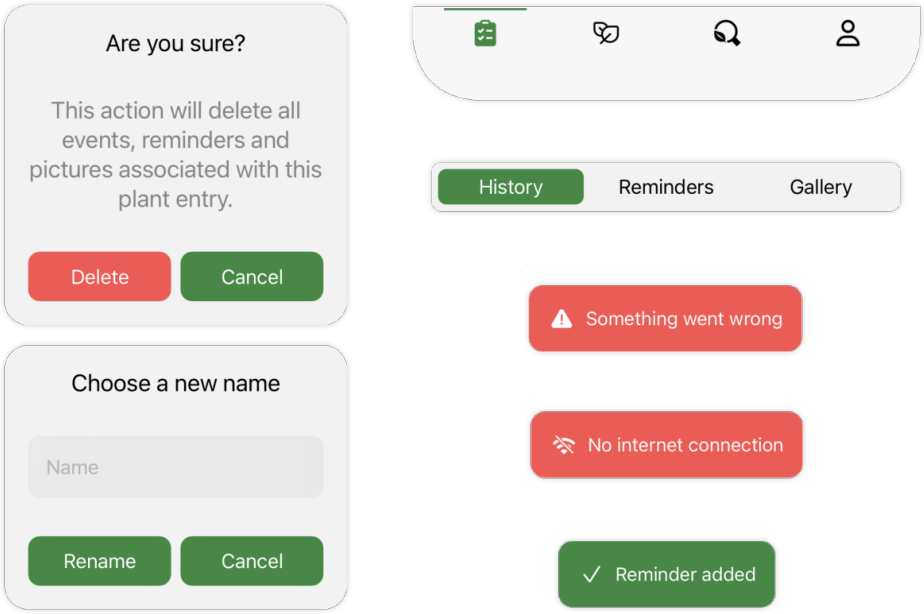
4.2.3 Identita a grafické prvky

Logo aplikace je černé písmeno „S“ na bílém pozadí. Na vrchním konci písmena se pak nachází menší zelený list. Vedle tohoto loga bylo vytvořeno logo textové, které neobsahuje pouze první písmeno názvu, ale název celý. To je použito na přihlašovací a registrační obrazovce. Loga lze vidět na obrázku 4.4.

Obrázek 4.4: Klasické a textové logo

Aplikace je laděna do kombinace 4 hlavních barev – černá, bílá, šedá, zelená. Zelená barva je použita zejména pro prvky, se kterými lze nějakým způsobem interagovat (poklepat na ně), nebo pro indikaci aktivních prvků (vybraný tab, sekce). Šedá barva slouží jako pozadí některých prvků a také jako barva informativního textu. Zároveň jsou v této barvě zobrazeny načítající se objekty. Text a pozadí má barvu černou a bílou (záleží na zvoleném barevném režimu – světlý nebo tmavý). Font textu je ponechán základní, tedy *SF Pro*. Barvu, kterou lze také nalézt, je červená. Ta se nachází na některých tlačítkách, které provádějí destruktivní akce (mazání), a jako pozadí chybových hlášek.

Aplikace používá ve většině případech základní grafické prvky, které jsou přítomny ve *SwiftUI*. Některé tyto prvky ale nevyhovovaly, co se týče grafického provedení (nedají se vhodně stylizovat), a tak byly vytvořeny vlastní verze těchto prvků (tab bar, segmented control). Zároveň byly vytvořeny ty, které se v základu vůbec nenacházejí (vlastní modal). Pro chybové hlášky byl využit balíček *PopupView*. Různé grafické prvky aplikace jsou zobrazeny na obrázku 4.5.



Obrázek 4.5: Ukázka grafických prvků

4.2.4 Přihlášení a registrace

Přihlašovací a registrační obrazovku lze vidět na ukázce 4.7. Na vrchu obou obrazovek se nachází textové logo a pod ním textová pole pro vyplnění přihlašovacích či registračních informací. Protože rozkliknutí textového pole otevře klávesnici, bylo potřeba zajistit, aby uživatel viděl, co píše a mohl se mezi textovými poli přepínat. U přihlašovací obrazovky klávesnice nijak neinterferuje s textovými poli. U registrace bylo přidáno schovávání loga při otevřené klávesnici. Pokud by logo zůstalo, dolní textová pole by byla překryta klávesnicí.

4.2.5 Onboarding

Cílem onboardingu je seznámit nového uživatele s funkcemi aplikace a nabídnout mu možnost povolení notifikací. Skládá se celkem ze šesti částí, přičemž tlačítkem „Next“ se uživatel může posunout vpřed. Může zároveň také použít *swipe* gesto, kterým se může dostat jak na následující tak na předchozí část. V poslední části onboardingu se text tlačítka změní na „Get Started“, které onboarding zavře a přesune uživatele do aplikace. Nad tlačítkem se nachází ukazatel, který pomocí teček ukazuje počet částí a v které části se uživatel právě nachází. Ukázka onboardingu se nachází na obrázku 4.6.

4.2.6 Správa týmů a míst

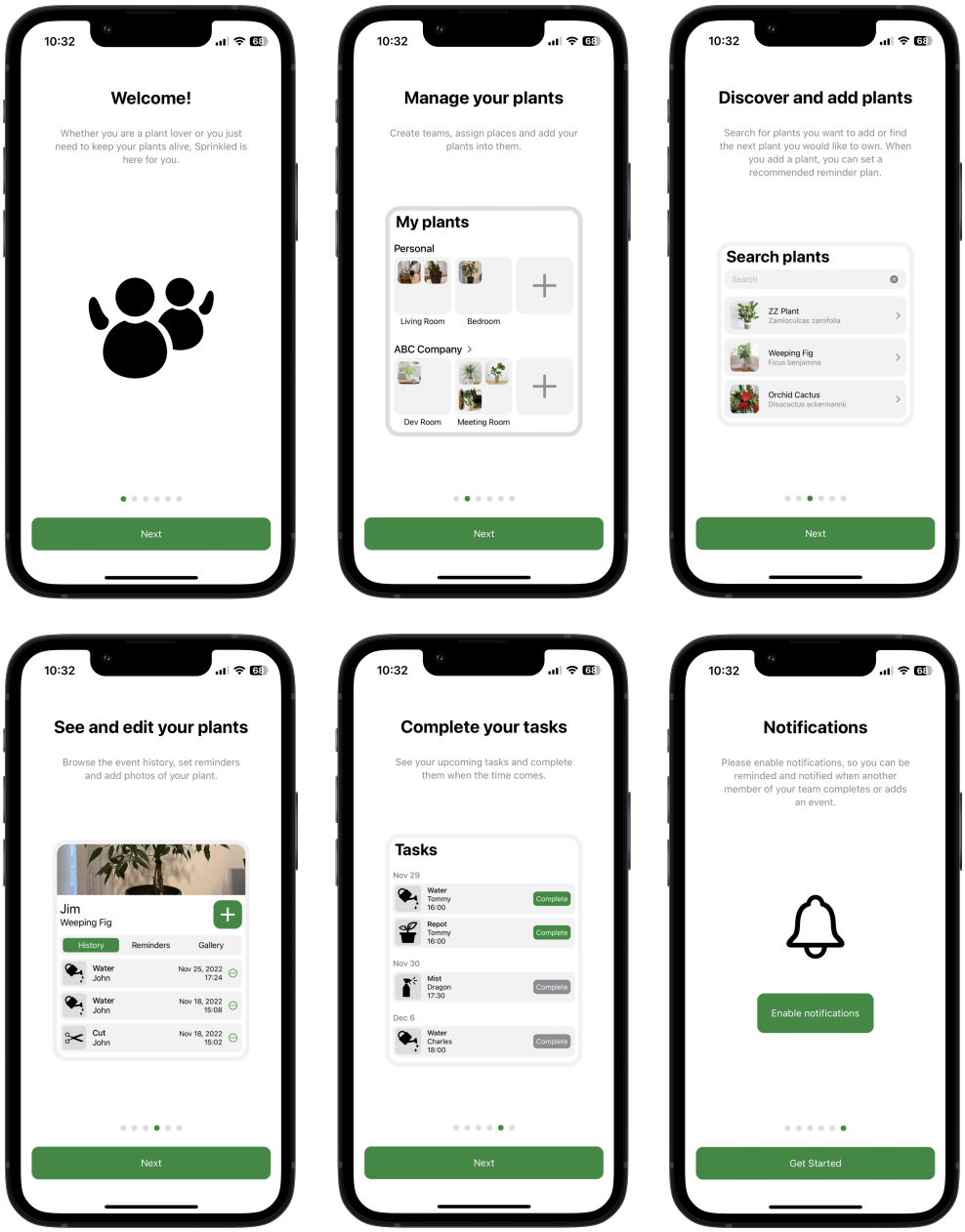
Spravovat týmy a místa může uživatel na obrazovce „My plants“. Ta vypadá téměř identicky jako návrh, změnila se ikonka tlačítka pro vytváření týmů, míst a přidávání rostlin. Je shodné s tlačítkem pro přidávání na detailu záznamu rostliny.

Do detailu týmu se lze dostat kliknutím na jméno týmu (dostatečným indikátorem, že na něj lze kliknout, by měla být zelená šipka, která se vedle jména týmu nachází). V detailu týmu může uživatel vidět členy, pokud má práva, může členy přidávat, odebírat nebo jim může dávat/brát práva.

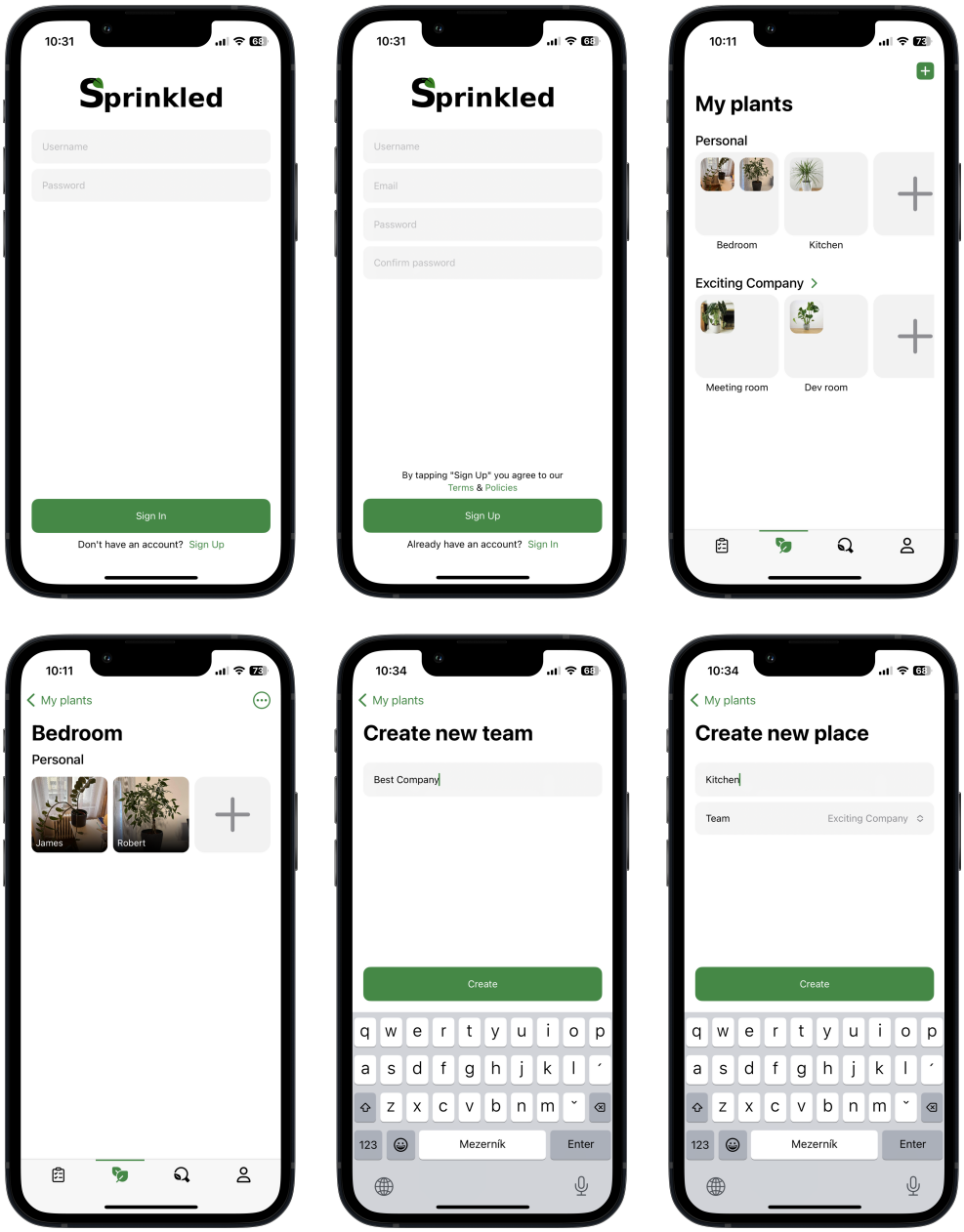
Na obrazovce s detailem místa se nachází jednotlivé záznamy rostlin (jako dlaždice s náhledovým obrázkem a názvem), do kterých se může prokliknout. Také zde může místo přejmenovat nebo odstranit.

4.2.7 Vyhledávání a přidávání rostlin

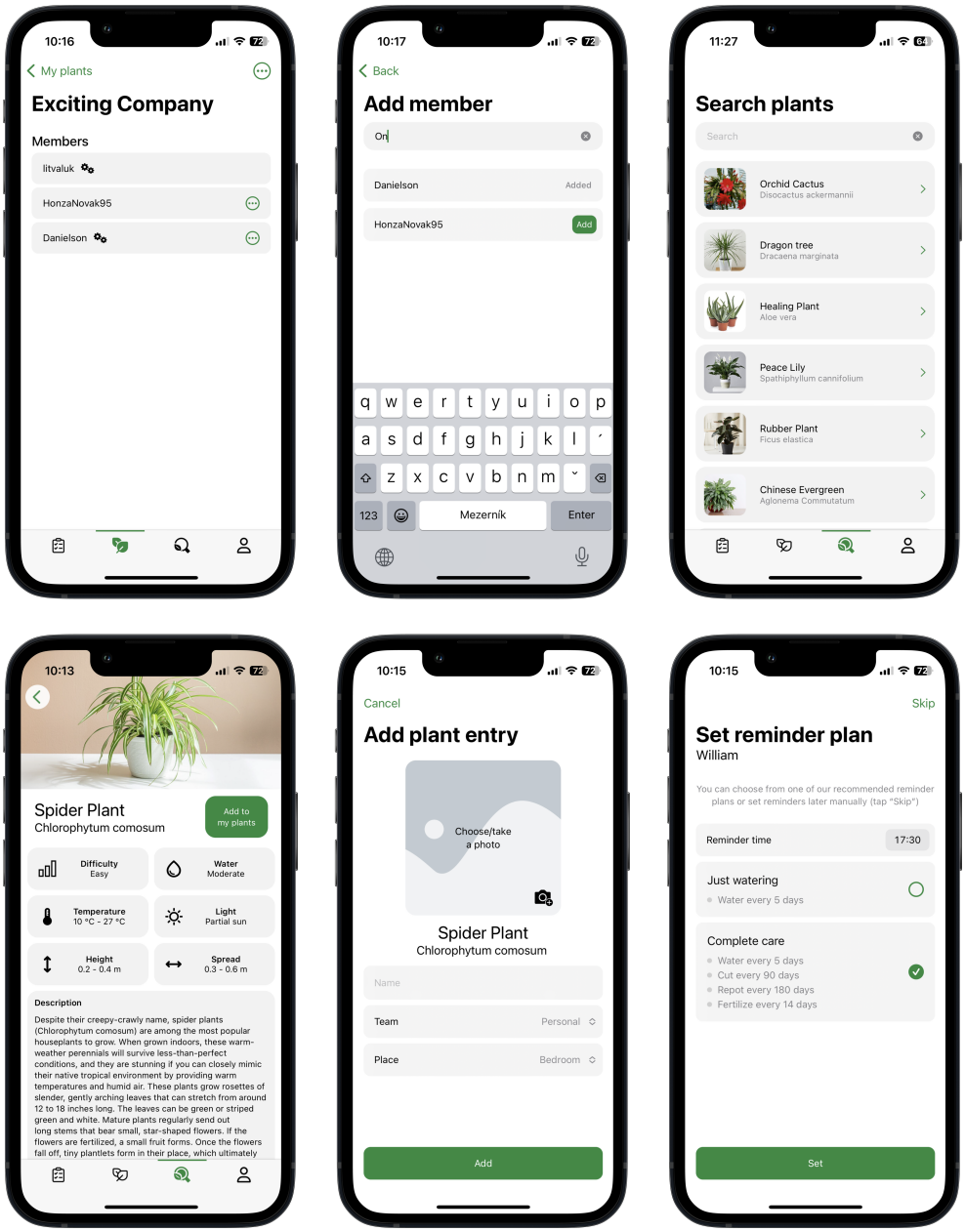
Na obrazovce se seznamem rostlin lze procházet jednotlivé druhy rostlin. Pomocí vyhledávacího pole lze vyhledávat jak podle obecného, tak i latinského názvu. Po kliknutí na jakýkoliv druh rostliny se uživatel dostane do jeho detailu, kde si lze prohlédnout základní informace a může si danou rostlinu přidat do svých rostlin. Na přidávací obrazovce může uživatel nahrát fotografii, specifikovat jméno své rostliny, přiřadit tým a místo. Po přidání rostliny se uživateli zobrazí obrazovka s nastavením připraveného připomínkového plánu



Obrázek 4.6: Ukázka obrazovek - onboarding



Obrázek 4.7: Ukázka obrazovek - přihlášení, registrace, moje rostliny



Obrázek 4.8: Ukázka obrazovek - správa týmu, vyhledávání a přidávání rostliny

(ten ovšem nastavovat nemusí). Proces vyhledávání a přidávání rostlin je zachycen v ukázce 4.8.

4.2.8 Správa záznamu rostliny

Z obrazovky místa se lze dostat na záznamy rostlin, které se v daném místě nachází. Obrazovka záznamu rostliny vedle obrázku, jména, druhu rostliny a tlačítka pro přidávání událostí, připomínek a fotek obsahuje 3 sekce. Sekci, ve které se uživatel nachází, lze měnit přepínačem (*Picker*). Protože základní přepínač (ve variantě *segmented control*) se nehodil ke vzhledu aplikace, byla vytvořena jeho vlastní implementace.

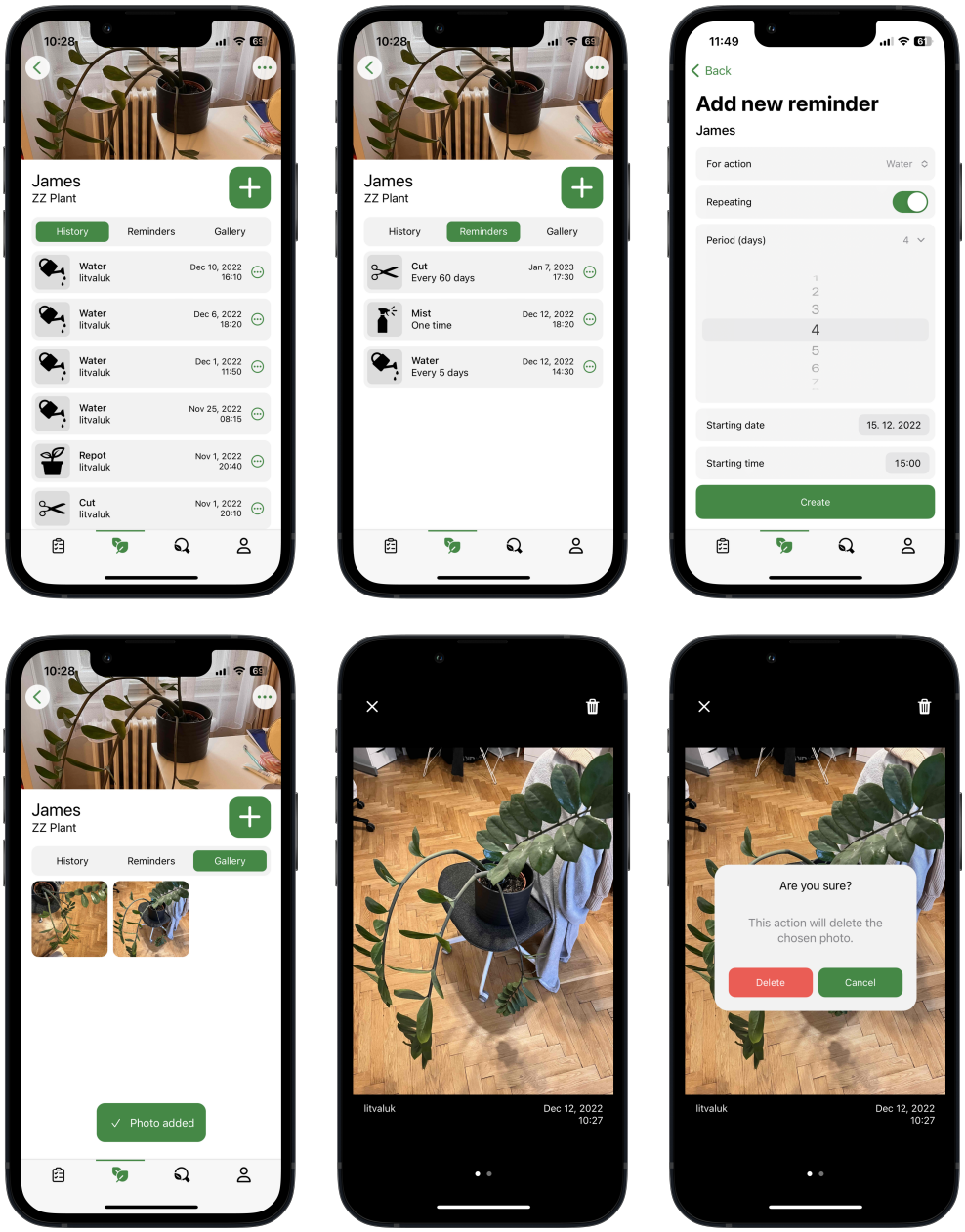
Seznam událostí a připomínek vypadá obdobně, každá položka obsahuje název akce a příslušnou ikonku. Pod názvem je v případě události zobrazeno jméno uživatele, který danou událost provedl. U připomínky je to pak interval, jak často se daná akce má vykonávat (pokud se jedná o jednorázovou připomínku, je zobrazen popisek „One time“). Na pravé straně položky je poté buď datum a čas provedení události nebo v případě připomínky její nadcházející spuštění (tedy poslání připomínkové notifikace). Úplně napravo se nachází tlačítko, které vyvolá menu, ve kterém lze událost či připomínku upravit nebo smazat. Upravovací obrazovka je téměř totožná obrazovce přidávací, pouze se mění několik popisků. Před samotným smazáním je uživatel vyzván k potvrzení akce pomocí vyskakovacího okna.

Galerie se skládá z dlaždic, obsahující náhledy jednotlivých fotografií. Po rozkliknutí fotografie se uživatel dostane k detailu fotografie. Zde se nad samotnou fotografií nachází tlačítko pro zavření (také je možné fotografii zavřít jejím potáhnutím pryč) nebo smazání fotografie a pod ní je jméno uživatele (který fotografii přidal) a také datum přidání. Samotnou fotografii lze přiblížit tzv. *magnifying* gestem. Pokud je fotografií více, lze *swipe* gestem mezi nimi procházet.

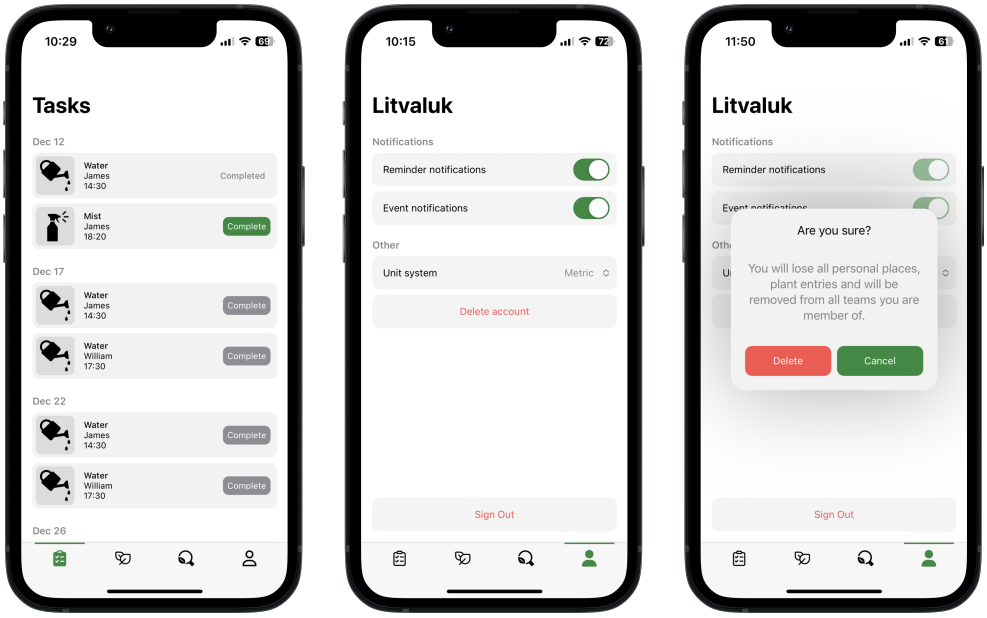
4.2.9 Úkoly

Samotné úkoly (řešeny jako nedokončené události) jsou načteny ze serveru do datové struktury `OrderedDictionary` (z knihovny Swift Collections). Úkoly jsou tak seskupeny a uloženy pod klíčem, kterým je textový řetězec vyjadřující datum ve formátu „`MMM d`“. Z tohoto seřazeného slovníku je v samotném View vytvořeno seřazené pole (dvojice datum a příslušný seznam událostí), které je následně zobrazeno.

Oproti návrhu se jednotlivé úkoly liší tím, že na pravé straně se nachází tlačítko „Complete“, které po jeho stisknutí označí daný úkol jako dokončený (z nedokončené události se stane dokončená a ta je zobrazena v historii příslušné rostliny). Zároveň, pokud se jedná o úkol, který přísluší týmové rostlině, jsou o jeho dokončení ostatní členové notifikováni (pokud mají zapnuté notifikace o událostech).



Obrázek 4.9: Ukázka obrazovek - správa záznamu rostliny



Obrázek 4.10: Ukázka obrazovek - úkoly a profil/nastavení

4.2.10 Profil a nastavení

Obrazovka, ve které uživatel může zapínat nebo vypínat druh notifikací (pomocí prvku *Toggle*), které mu jsou posílány. Dále si také může přepnout měrné jednotky, které se zobrazují v aplikaci. Může si zde také smazat svůj účet nebo se odhlásit. Ukázka této obrazovky je součástí obrázku 4.10.

4.2.11 Notifikace

O samotné posílání notifikací se stará serverová část, nicméně je potřeba u každého zařízení získat tzv. *push* token. Pro získání tohoto tokenu je potřeba zavolat metodu `registerForRemoteNotifications` na singletonu `UIApplication`. Následně se zavolá v `AppDelegate` specifická metoda o úspěšné registraci *push* notifikací, ve které se push token nachází v parametru dané metody. Tento token je poté odeslán na server a uložen v databázi.

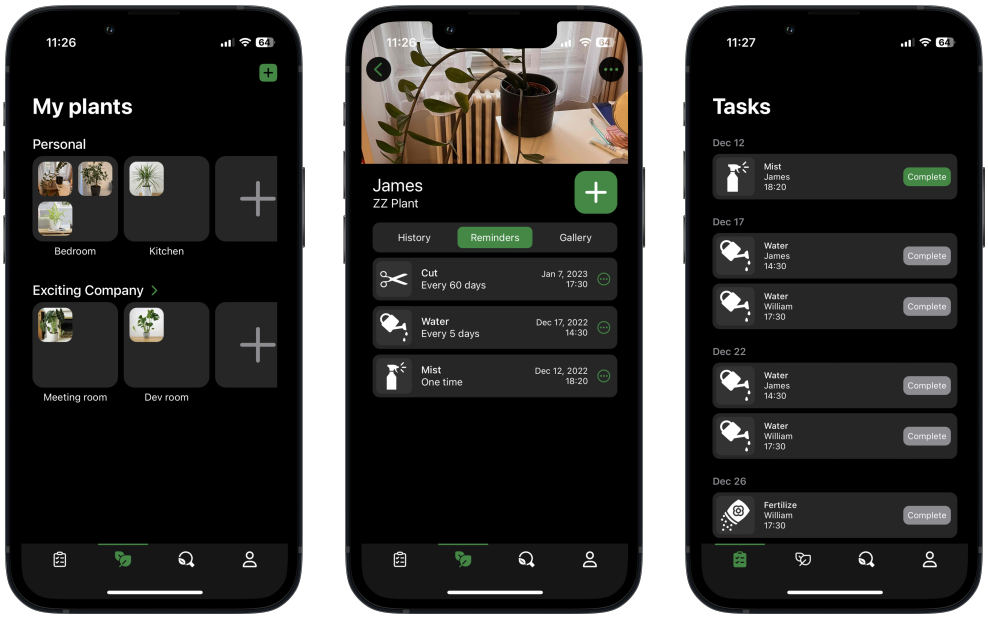
Na jednotlivá zařízení tedy chodí 2 druhy notifikací:

Událostní Tuto notifikaci obdrží ta zařízení, na kterých je právě přihlášený uživatel (zároveň má povolené událostní notifikace), jež se nachází v týmu, ve kterém právě někdo přidal událost nebo dokončil úkol u některé z rostlin. Obsahuje uživatele, o jakou akci se jedná a na jaké rostlině byla provedena.

Připomínkové Je poslána v okamžik, kdy aktuální čas odpovídá času některé z připomínek. Je poslána na zařízení všech příslušných uživatelů, kteří mají povolené připomínkové notifikace.

4.2.12 Tmavý režim

Protože operační systém iOS od verze 13 podporuje tmavý režim, je zvykem, že aplikace tento režim podporují. V případě aplikace „Sprinkled“ tomu není jinak. Implementačně se nejedná a složitý úkon, v projektu je potřeba definovat jednotlivé barvy, které aplikace používá v rámci *Assets* (prostředky). Každé barvě je poté přidána světlá a tmavá varianta. Aplikace už pak mezi těmito barvy dokáže podle zvoleného režimu přepínat. Ukázku tří obrazovek ve tmavém režimu lze vidět na obrázku 4.11.



Obrázek 4.11: Ukázka obrazovek v tmavém režimu (dark mode)

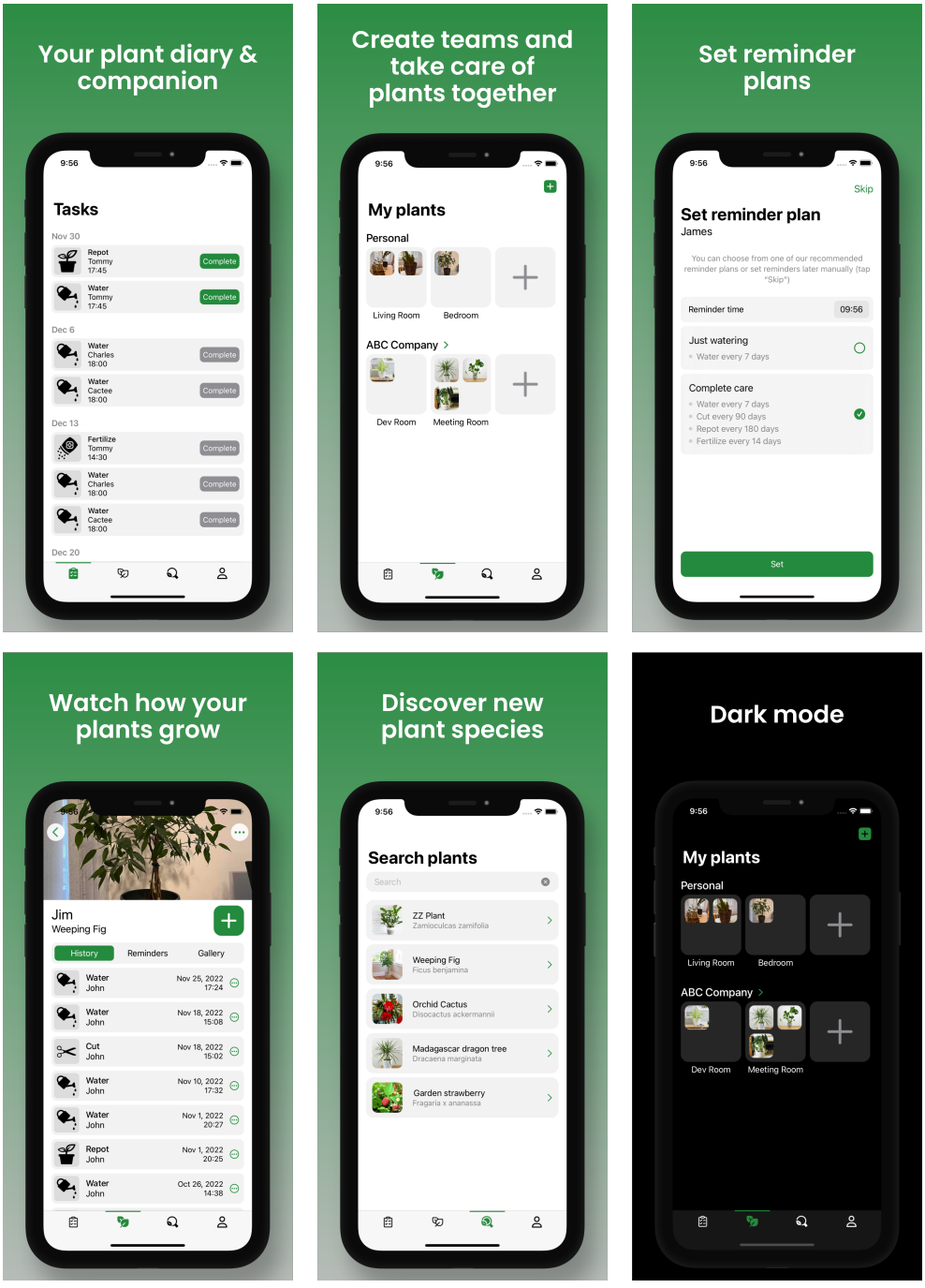
4.2.13 Zveřejnění aplikace

Pro nahrání a zveřejnění aplikace na App Store bylo potřeba zřídit si Apple účet s aktivním vývojářským programem (Apple Developer Program). Ten je pro jednotlivce zpoplatněn sumou 99 dolarů ročně. V nástroji App Store Connect, který slouží pro správu aplikací v App Store, byla vytvořen záznam o aplikaci s názvem „Sprinkled“. Pomocí XCode byl z implementace aplikace vytvořen archiv, který byl na App Store Connect nahrán. V tomto stavu bylo

možné aplikaci interně testovat v aplikaci TestFlight, aniž by bylo potřebné podstupovat jakýkoliv schvalovací proces.

Pro zpřístupnění aplikace veřejnosti bylo potřeba vyplnit popis, klíčová slova, informace o využívání citlivých dat, poskytnout data o dostupnosti, nahrát náhledové fotografie (viz 4.12), nastavit věková omezení a cenu (aplikace je dostupná zadarmo). Také bylo ještě potřeba v aplikaci vytvořit účet a vyplnit tyto přihlašovací údaje, aby aplikaci mohl pracovník z Apple vyzkoušet a schválit.

Aplikace byla jednou vrácena z důvodu neposkytnutí možnosti smazání účtu. Apple totiž ve své směrnici (App Store Review Guidelines) vyžaduje, aby aplikace, které umožňují vytváření účtů, zároveň také poskytly možnost pro mazání účtů. Tato funkcionalita byla tedy do aplikace dodělána a následně aplikace schvalovacím procesem prošla. Aplikace je dostupná v App Store na [75].



Obrázek 4.12: Náhledové obrazovky v App Store

Kapitola 5

Uživatelské testování

Tato kapitola se věnuje uživatelskému testování. Uživatelské testování (také známe jako *Usability Testing*) je jednou z oblíbených metod testování UX. Malá skupina uživatelů (z cílové skupiny) plní úkoly v rámci aplikace, zatímco moderátor pozoruje jejich chování a dělá si poznámky. Cílem tohoto testování je zjistit, zda jsou uživatelé, kteří aplikaci vidí poprvé, schopni se zorientovat a provést zadané úkoly. Tímto lze odhalit problémy v návrhu uživatelského rozhraní a ty případně opravit.

Předpokladem bylo zvolit testery, kteří jsou aktivními uživateli mobilních zařízení s operačním systémem iOS. Tím bylo zaručené, že umí zařízení používat a zároveň mohou aplikaci otestovat na vlastním zařízení. Zároveň také musí umět anglicky, neboť aplikace podporuje (prozatím) pouze angličtinu. Bylo vybráno celkem 5 (doporučený počet dle [76]) účastníků, kteří spadají mezi typické a příležitostné uživatele. Testování proběhlo na již nasazené a vydané aplikaci. A bylo rozděleno na tři části.

Pre-test Seznámení testerů s testovacím procesem, krátké popsání aplikace a optání se na několik otázek, které se týkají jejich vztahu k rostlinám, jestli nějaké vlastní a starají se o ně a případně jestli používají, či zkusili použít nějakou aplikaci pro péči o rostliny. Následně byly podány instrukce k nainstalování aplikace z obchodu s aplikacemi *App Store*.

Testovací scénáře Samotné testování aplikace, bylo rozděleno do 11 scénářů, které na sebe navazují.

Post-test Zadání několika otázek, které zjišťují celkové pocity testerů z aplikace, co se jim líbilo/nelíbilo, jejich nápady na zlepšení a zda by aplikaci využili.

U každého testování byla zaznamenávána obrazovka zařízení a zvuk z externího mikrofonu. Jednotlivé nahrávky se nachází na přiloženém médiu.

5.1 Pre-test

Nejdříve byli testeři požádáni, aby se představili. Jednalo se o dvě ženy a tři muže ve věkovém rozpětí 24 až 25 let. Jeden muž a žena byli čerstvě vystudovaní zubaři a zbytek studenti architektury. Poté jim byly položeny následující otázky:

- Vlastníte a staráte se (nyní nebo v minulosti) o nějaké rostliny?
- Jakým způsobem jste se o ně staral/a? Co všechno jste musel/a dělat?
- Vyzkoušel/a jste či používáte nějakou mobilní aplikaci, která vám pomáhá se starat o rostliny?

Všichni až na jednoho sdělili, že aktuálně vlastní a starají se o 2 nebo více rostlin. Povětšinou se jedná o nenáročné rostliny, které intuitivně zalévají. Dva testeři navíc rostliny občas zastřihávají, přesazují nebo otáčejí za sluncem. Žádný y testerů nevyužil specializovanou aplikaci, která by pomáhala se staráním o rostliny. Jeden vyzkoušel v historii použít klasické připomínky (základní aplikace), nicméně tento postup se neosvědčil.

Poté jim bylo vysvětleno, jak si mají aplikaci stáhnout a nainstalovat, načež u každého vše proběhlo bez problémů.

5.2 Testovací scénáře

Testerům byly postupně předkládány kartičky se zadáním jednotlivých testovacích scénářů. Tyto scénáře jsou zkonstruovány tak, aby byly splnitelné samostatně bez doptávání. Zásah moderátora a poskytnutí pomoci by totiž znamenalo, že aplikace nefunguje zcela správně nebo je část uživatelského rozhraní špatně navržená.

5.2.1 Registrace

Tento scénář je zaměřen na prvotní spuštění aplikace a následné vytvoření nového uživatele. Po úspěšné registraci se novému uživateli zobrazí onboarding. U onboardingu je nutné sledovat, zda si ho testeři opravdu čtou, či ho rovnou přeskočí. K provedení tohoto scénáře není potřeba žádného předpokladu.

Zadání

Otevřete aplikaci, vytvořte si nový účet a dostaňte se na obrazovku „My plants“. Registrujte se pod uživatelským jménem *Franta01* a použijte email *franta01@gmail.com*. Heslo zvolte libovolné.

Očekávaný průchod

1. Otevření aplikace
2. Kliknutí na klikatelný text „Sign Up“
3. Zadání informací potřebných k registraci
4. Kliknutí na zelené tlačítko „Sign Up“
5. Prohlédnutí onboardingu
6. Zavření onboardingu

Vyhodnocení

S tímto scénářem nebyl víceméně žádný problém, pouze v jednom případě tester zadával registrační údaje jako přihlašovací, načež pak zjistil, že se nejdříve musí do registrační obrazovky prokliknout. Vhodným řešením by mohlo být zobrazení registrační obrazovky jako první obrazovky při otevření aplikace. Zároveň registrace bude zřejmě častějším úkonem než přihlášení. Všichni si po registraci pečlivě prošli onboarding a pouze 2 si povolili notifikace.

5.2.2 Přidání rostliny

V tomto scénáři by měl být testovací subjekt sám schopný přidat rostlinu do seznamu svých rostlin. To se může zprvu zdát jednoduché, ale cesta, jak se k výsledku dostat, nemusí být úplně přímočará. Subjekt se musí co nejrychleji zorientovat v rozhraní (měl by už mít ponětí, co lze v aplikaci dělat, pokud si prohlédl onboarding), přidat místo a do něj vytvořit záznam o rostlině. Předpokladem pro splnění je již přihlášený uživatel.

Zadání

Doma v obývacím pokoji máte fíkus (obecný anglický název Weeping Fig, latinsky Ficus benjamina). Přidejte ho tedy do svých rostlin.

Očekávaný průchod

1. Zobrazení obrazovky „My plants“
2. Kliknutí na přidávací tlačítko (symbol plus v zaobleném čtverci) u týmu „Personal“ nebo na malé tlačítko v pravém horním rohu (symbol plus v zaobleném čtverci) a následné zvolení možnosti „Create new place“
3. Vyplnění jména místa a zvolení týmu
4. Kliknutí na tlačítko „Create“

5. Kliknutí na právě vytvořené místo
6. Kliknutí na přidávací tlačítko (symbol plus v zaobleném čtverci)
7. Vyhledání rostliny a kliknutí na ni
8. Kliknutí na tlačítko „Add to my plants“
9. Vybrání/vyfocení fotografie rostliny, vyplnění jména, přiřazení týmu a místa
10. Kliknutí na tlačítko „Add“
11. Vybrání a potvrzení (tlačítkem „Set“) připraveného plánu nebo přeskočení (tlačítkem „Skip“)

Vyhodnocení

Všem testerům se podařilo tento úkol splnit, ale ve třech případech nebyl postup a výsledek podle očekávání. Po kliknutí na tlačítko pro přidání místa se totiž domnívali, že budou rovnou přidávat rostlinu, a tak jako jméno místa vyplnili název rostliny (Weeping Fig). Toto místo přidali a následně do něj přidali rostlinu, což proběhlo bez problémů. Zároveň si uvědomili, že přidali nejdříve místo. I když je v onboardingu zmíněné, že se vytvářejí místa a v nich rostliny, bylo by vhodné více odlišit, co se právě přidává, nebo více informovat uživatele (vhodným způsobem), že před samotným přidáním rostliny by si měl nejprve vytvořit místo. Dalším zjištěním v tomto scénáři bylo, že dva účastníci si nepřidali k rostlině fotografii a někteří zápasili se zavřením klávesnice, která překrývala část obrazovky. Očekávali, že ji zavřou poklepáním mimo klávesnici (což nefungovalo).

5.2.3 Vytvoření týmu

Scénář sleduje proces vytvoření týmu a následného přidání členů a jejich správu. Pro jeho splnění je předpokladem být přihlášený v aplikaci.

Zadání

Pracujete v kanceláři s dalšími dvěma kolegy, kde se společně staráte o několik rostlin. Vytvořte tedy tým (název *Práce*) a přidejte do něj své dva kolegy, kteří už mají v aplikaci vytvořený účet. Jejich uživatelská jména jsou *HonzaNovak95* a *Danielson*. Následně kolegovi *Danielson* dejte administrátorská práva, aby i on mohl přidávat a upravovat členy týmu.

Očekávaný průchod

1. Zobrazení obrazovky „My plants“
2. Kliknutí na na malé tlačítko v pravém horním rohu (symbol plus v zaobleném čtverci) a následné zvolení možnosti „Create new team“
3. Vyplnění jména nového týmu
4. Potvrzení vytvoření týmu kliknutím na tlačítko „Create“
5. Kliknutí na název právě vytvořeného týmu
6. Kliknutí na tlačítko v pravém horním rohu (tři tečky uvnitř kolečka) a vybrání možnosti „Add member“
7. Vyhledání a přidání požadovaných uživatelů stiskem tlačítka „Add“.
8. Návrat na týmovou obrazovku gestem nebo stiskem šipky (levý horní roh).
9. U uživatele *Danielson* kliknout na tlačítko (symbol plus v zaobleném čtverci), které vyvolá menu, následně vybrat možnost „Give admin rights“

Vyhodnocení

S tímto scénářem nebyl žádný zásadní problém. Tři účastníci ho splnili přesně podle očekávání a u dvou se stalo, že nejdříve přidali místo s názvem „Práce“. Nicméně poměrně rychle si uvědomili, že přidali právě místo a ne tým. Poté se vrátili zpět a úspěšně tým vytvořili. S přidáváním účastníků a nastavováním administrátorských práv nebyly žádné potíže.

5.2.4 Přidání míst a rostlin do týmu

Jedná se o prověření provedení obdobného scénáře s přidáním rostliny s tím rozdílem, že jsou tentokrát přidávány rostliny do týmu. Scénář má sledovat jak rychle se člověk naučil a zapamatoval proces přidávání rostliny a zda je schopen rozlišit osobní rostliny od týmových. Předpokladem je vytvořený tým s názvem „Práce“.

Zadání

V kanceláři máte kulkas (obecný anglický název ZZ plant, latinsky Zamioculcas Zamiifolia) a dračinec (obecný anglický název Madagascar Dragon Tree, latinsky Dracaena marginata). Přidejte je do svého týmu do nové místnosti s názvem „Kancelář“ a nenastavujte u nich žádný připravený plán.

Očekávaný průchod

1. Zobrazení obrazovky „My Plants“
2. Kliknutí na přidávací tlačítko (symbol plus v zaobleném čtverci) u týmu „Práce“ nebo na malé tlačítko v pravém horním rohu (symbol plus v zaobleném čtverci) a následné zvolení možnosti „Create new place“
3. Vyplnění jména místa a zvolení týmu „Práce“
4. Kliknutí na tlačítko „Create“
5. Kliknutí na právě vytvořené místo „Kancelář“
6. Kliknutí na přidávací tlačítko (symbol plus v zaobleném čtverci)
7. Vyhledání rostliny (ZZ plant) a kliknutí na ni
8. Kliknutí na tlačítko „Add to my plants“
9. Vybrání/vyfocení fotografie rostliny, vyplnění jména, přiřazení týmu a místa
10. Kliknutí na tlačítko „Add“
11. Přeskočení nastavení připraveného plánu (tlačítkem „Skip“)
12. Návrat na obrazovku s místem „Kancelář“ gestem nebo stiskem šipky (levý horní roh).
13. Zopakování kroku 6 až 11, pro přidání rostliny (Madagascar Dragon Tree)

Vyhodnocení

Protože účastnící už v druhém scénáři rostlinu přidávali a byli tedy seznámeni s tímto procesem, přidání dvou nových rostlin do týmu jim nedělalo žádný problém. Pouze v jednom případě bylo špatně pochopené zadání, kdy „do svého týmu“ bylo pochopeno jako do svých osobních rostlin místo do týmu „Práce“. Následně došlo k objevení chyby, kdy po vytvoření místa nešlo v tomto místě ve výběru rostlin rozkliknout detail rostliny. Tuto chybu se avšak nepovedlo reprodukovat a zjistit tak její příčinu. Také se opět projevovaly nejasnosti kolem zavírání klávesnice při přidávání rostliny.

5.2.5 Manuální nastavení připomínek

Scénář testuje manuální přidání připomínek k rostlině (jednorázová a opakující se). Aby tento scénář mohl vykonat, je potřeba mít již vytvořený záznam o rostlině (z předešlých scénářů se jedná o rostlinu dračinec v místě „Kancelář“ v týmu „Práce“).

Zadání

Neboť jste si nenastavil/a žádný plán u rostliny dračinec (Dragon Tree), chcete si nyní nastavit připomínku (anglicky reminder) na zalévání (anglicky water) každých 5 dní (od zítřka) a také připomínku na přesazení (anglicky repot) příští středu v 14:00. Vytvořte takové připomínky u této rostliny.

Očekávaný průchod

1. Zobrazení obrazovky „My plants“
2. Kliknutí na místo „Kancelář“
3. Kliknutí na záznam rostliny, zastupující rostlinu „ZZ Plant“
4. Kliknutí na zelené tlačítko plus v zaobleném čtverci a vybrání možnosti „Add reminder“
5. Vyplnění informací k vytvoření připomínky
6. Vytvoření připomínky stiskem tlačítka „Create“
7. Zopakování kroků 4 až 6 k vytvoření druhé připomínky

Vyhodnocení

Vše proběhlo v pořádku a podle očekávání. Nejprve se testeři dostali do záznamu požadované rostliny a připomínky nastavili. Jeden tester si při přidávání druhé připomínky spletl připomínku a událost – nejprve zadával událost. Protože nemohl vytvořit událost s budoucím datem, uvědomil si, že se jedná zřejmě také o připomínku a tu poté správně vytvořil.

5.2.6 Editace připomínek

Tento scénář ověřuje, zda je uživatel schopen upravovat připomínky. Předpokladem k provedení této akce je mít vytvořený záznam o rostlině a u něj přidanou alespoň jednu připomínku. Očekává se, že z minulého scénáře se uživatel nachází na obrazovce se záznamem rostliny dračinec.

Zadání

Právě jste si uvědomil/a, že příští středu nejste doma a nemůžete tak rostlinu dračinec přesadit. Upravte datum u dané připomínky (místo příští středy nastavte příští čtvrtek).

Očekávaný průchod

1. Přepnutí se do sekce připomínek stiskem „Reminders“
2. Kliknutí na ikonku tří teček v kolečku a vybrání možnosti „Edit“
3. Změnění data (z příští středy na příští čtvrtek)
4. Potvrzení změny stiskem tlačítka „Edit“

Vyhodnocení

U testera, u kterého se našla chyba s vytvářením místa a následně nemožného prokliku na detail rostliny, nešlo zmáčknout tlačítko, které otevře menu s úpravou či smazáním připomínky. Proto tento krok byl u něj přeskočen a bylo mu vysvětleno, jak by se tento scénář měl provést (zároveň sdělil, že přesně takto by to čekal). U ostatních editace připomínky proběhla bez problému.

5.2.7 Přidávání událostí

Jedná se o scénář, který testuje přidání události. Pro provedení je potřeba mít vytvořený záznam rostliny. Uživatel by se měl nacházet na obrazovce se záznamem rostliny (po provedení předchozího scénáře).

Zadání

Najednou jste si vzpomněl/a, že jste minulý pátek v 16:00 jste dračinec pohnojil/a. Přidejte tuto událost k této rostlině.

Očekávaný průchod

1. Kliknutí na zelené tlačítko plus v zaobleném čtverci a vybrání možnosti „Add event“
2. Vyplnění informací k vytvoření události
3. Vytvoření události stiskem tlačítka „Create“

Vyhodnocení

Všichni testeři splnili tento scénář bez problému. Dva z nich se překlikli (zadali údaj špatně nebo na některý zapomněli), načež tuto chybu napravili upravením dané události.

5.2.8 Mazání událostí

V tomto scénáři by měl být tester schopen smazat již vytvořenou událost. Pro splnění tohoto scénáře je potřeba mít vytvořený záznam rostliny a v něm přidanou událost. Z předchozích scénářů se jedná o dračinec, který má vytvořenou událost pohnojení (fertilize).

Zadání

Omylem jste se spletl/a, nepohnojil/a jste dračinec ale fíkus. Odstraňte vytvořenou událost. Novou (správnou) událost nepřidávejte.

Očekávaný průchod

1. Přepnutí se do sekce historie událostí stiskem „History“
2. Kliknutí na náhled fotografie
3. Kliknutí na tlačítko s ikonou koše
4. Potvrzení smazání fotografie kliknutím na tlačítko „Delete“

Vyhodnocení

Žádný problém nenastal, všichni testeři tento scénář splnili přesně podle očekávaného průchodu.

5.2.9 Přidávání fotografií

Scénář testuje proces přidávání fotografií, který by měl být jednoduchý a přímočarý. Předpokladem k provedení tohoto scénáře je mít vytvořený záznam o rostlině (dračinec).

Zadání

Protože vaši příbuzní vám nevěří, jak se daří vašim rostlinám, budete si chtít začít vést galerii, abyste jim při vašem společném setkání mohl ukázat, jak váš dračinec roste. Nahrajte k dračinci první fotografii.

Očekávaný průchod

1. Kliknutí na zelené tlačítko plus v zaobleném čtverci a vybrání možnosti „Add photo“
2. Vybrání „Camera“ nebo „Photo library“ ve spodním panelu.
3. Vyfocení fotografie nebo vybrání fotografie z knihovny fotografií zařízení

Vyhodnocení

Scénář byl splněn všemi bez problému, jeden tester dokonce vyzkoušel obě možnosti přidání fotografie (vyfocení a přidání z galerie).

5.2.10 Mazání fotografií

Tento scénář ověřuje, zda je uživatel schopen smazat fotografie. Předpokladem pro tento scénář je mít vytvořený záznam rostliny a v něm přidanou fotografii.

Zadání

Vložil/a jste omylem špatnou fotografii, smažte ji a nahrajte novou.

Očekávaný průchod

1. Přepnutí se do sekce fotografií stiskem „Gallery“
2. Kliknutí na náhled fotografie
3. Kliknutí na tlačítko s ikonou koše
4. Potvrzení smazání fotografie kliknutím na tlačítko „Delete“

Vyhodnocení

U zařízení s menším displejem bylo tlačítko pro mazání fotografie obtížně stisknutelné. Účastníci testu se museli správně trefit do určitého místa, aby se tlačítko stisklo. Mimo tento fakt nedošlo k jiným komplikacím a scénář byl všemi splněn.

5.2.11 Provedení/dokončení úkolů

Posledním scénářem je ověření, zda se uživatel dokáže orientovat v nadcházejících úkolech. Měl by být schopen si je prohlédnout a dokončit je kliknutím. Tento scénář vyžaduje přítomnost 2 jednorázových připomínek, které notifikují uživatele později v daném testovacím dni.

Zadání

Zkontrolujte seznam úkolů, zda není potřeba nějaký dnes dokončit. Pokud ano, předpokládejte že jste dané úkoly splnil/a a označte je za dokončené v aplikaci.

Očekávaný průchod

1. Zobrazení obrazovky „Tasks“
2. Vyhledání úkolů k dokončení a dokončení stiskem tlačítka „Complete“

Vyhodnocení

S tímto scénářem nebyl žádný problém, všichni ho dokončili podle očekávaného průchodu.

5.3 Post-test

Po splnění všech scénářů byly položeny a diskutovány následující otázky:

- Jak byste ohodnotil/a celkový dojem z aplikace na stupnici od 1 (nejlepší) do 10 (nejhorší)?
- S čím jste měl/a největší problém při plnění zadaných úloh?
- Co naopak bylo skvělé, respektive je nějaká část aplikace, která se vám líbí?
- Změnil/a byste některou část aplikace nebo vše bylo podle očekávání?
- Využil/a byste tuto aplikaci ve vašem životě?

Celkový dojem testeři ohodnotili vcelku kladně. Jednotlivá hodnocení jsou v tabulce 5.1, po zprůměrování vyšla hodnota 2.

Většina testerů popisovala, že při druhém úkolu nejprve tápali v tom, jaká je hierarchie obsahu. Nebylo jim ihned jasné, že na začátku jsou týmy, ve kterých jsou místa a teprve až v nich jsou jednotlivé rostliny. Dále někteří zmiňovali, že se jim nedařilo zavřít klávesnici při přidávání rostlin a jeden tester zmínil, že na tlačítku, které potvrzuje změny, je zvyklý vídat spojení „Save changes“ místo „Edit“.

Obecně se testerům líbil vzhled aplikace, že je jednoduchý a poměrně přehledný. Jeden z testerů zmínil, že se podobá základním předinstalovaným iOS aplikacím. Celkově se testerům líbil koncept sdílení, tedy že více lidí se podílí na pěstování stejných rostlin. Všichni by si dokázali představit, že by aplikaci používali ve svém životě.

Tester #1	Tester #2	Tester #3	Tester #4	Tester #5
1	2	3	2	2

Tabulka 5.1: Tabulka hodnocení celkového dojmu (1 (nejlepší) – 10 (nejhorší)

5.4 Zjištěné nedostatky a jejich řešení

Ačkoli aplikace při uživatelském testování dostala velmi pěkné hodnocení, bylo nalezeno několik nedostatků a chyb. Ke zlepšení požitku z používání aplikace je více než vhodné tyto nedostatky odstranit. Některé nedostatky lze vyřešit

pouze mírným zásahem do kódu, jiné však potřebují promyšlenější řešení. Níže se nachází seznam těchto nedostatků a jejich řešení či návrh, jak tento nedostatek řešit.

Na první pohled nedostatečná orientace v hierarchii tým – místo – záznam rostliny

Priorita: Střední
Složitost: Vysoká

Jedná se o komplexnější problém, který lze řešit různými způsoby. Řešením by mohlo být přidání nebo upravení některých popisků na obrazovce „My plants“. V úvahu připadá i přejmenování této obrazovky na něco, co více souvisí s týmem. Určitě by nebylo špatné vylepšit onboarding tak, že by byl kladen důraz právě na pochopení této hierarchie, nebo vytvořit interaktivní tutoriál.

Nejasný rozdíl mezi událostí a připomínkou

Priorita: Střední
Složitost: Střední

Lze řešit například důkladnějším vysvětlením v rámci onboardingu.

Problém se zavíráním klávesnice při přidávání rostliny

Priorita: Střední
Složitost: Nízká

Jedná se spíše o implementační chybu, která jde poměrně jednoduše opravit. Klávesnici by se měla zavřít při poklepání mimo ni.

Neexistující výchozí obrázek při přidání rostliny bez obrázku

Priorita: Střední
Složitost: Nízká

U záznamu rostliny nebyl nastaven žádný výchozí obrázek, tudíž aplikace zobrazovala tento obrázek jako načítající se do nekonečna. Řešením je přidat buď obrázek daného druhu rostliny (ze seznamu rostlin) nebo použít obecný zástupný obrázek pro rostlinu.

Vyplňování registračních informací na přihlašovací obrazovce

Priorita: Nízká
Složitost: Nízká

Protože registrační obrazovka je využívána častěji než ta přihlašovací, je vhodné právě registrační obrazovku nastavit jako prvotní obrazovku při startu aplikace (a nepřihlášeném uživateli).

Očekávání vhodnějších popisků u tlačítek

Priorita: Nízká
Složitost: Nízká

Jak bylo testerem doporučeno, u akcí, které něco upravují, by text tlačítek obsahovat slovní spojení „Save changes“ namísto „Edit“.

Závěr

Cílem této závěrečné práce bylo analyzovat, navrhnout a následně implementovat mobilní aplikaci pro operační systém iOS, která by měla sloužit jako pomocník či deník při starání se o rostliny. Nakonec výslednou aplikaci řádně otestovat.

První kapitola se zprvu zabývala kvantitativním a kvalitativním výzkumem a analýzou existujících řešení. Existující aplikace byly ohodnoceny Nielsenovo heuristikou a byly shrnuty jejich silné a slabé stránky. Na základě získaných informací byly specifikovány funkční a nefunkční požadavky aplikace. V druhé kapitole jsou popsány metodiky softwarového vývoje. Jedna z těchto metodik byla poté vybrána pro následný vývoj a tento výběr byl také odůvodněn.

Kapitola třetí se věnovala návrhu samotné aplikace. Nejprve byly vytvořeny persony, reprezentující tři druhy skupin uživatelů. Poté byly specifikovány případy užití a scénáře. Prvotní zachycení objektů a entit bylo zachyceno v doménovém modelu. Ten pomohl představit si rozsah celé domény a vztahy mezi jednotlivými objekty. Následně byla popsána architektura aplikace od datové, přes aplikační až po prezentační vrstvu. Součástí popisu architektury je i popis použitých technologií. V závěru této kapitoly je poté navrženo uživatelské rozhraní, a to jako prototyp vytvořený v nástroji Figma.

Samotné implementaci se věnuje kapitola s názvem realizace. Ta byla rozdělena do dvou sekcí, kdy první se věnovala serverové části a ta druhé části klientské, tedy mobilní aplikaci pro operační systém iOS. Obě tyto sekce popisovali využité nástroje, knihovny, strukturu jednotlivých projektů a také detaily ohledně nasazení či zveřejnění.

Výsledná aplikace byla otestována v rámci uživatelského testování (pátá kapitola), během kterém bylo zjištěno několik nedostatků, mezi kterými jsou například na první pohled špatná orientace v hierarchii obsahu nebo problémy se zavíráním klávesnice. K těmto nedostatkům byla navržena potenciální řešení a některá z nich byla rovnou implementována. Na druhou stranu se ale testerům líbil vzhled aplikace a koncept týmového starání se rostliny.

Aktuální stav a budoucnost

Aplikaci se podařilo implementovat dle všech vytyčených požadavků a je volně přístupná ke stažení na *App Store*. Od 2. prosince 2022 (vydání aplikace) do 3. ledna 2023 (vydání této závěrečné práce) se aplikace ve výsledku vyhledávání objevila celkem 1549krát, samotnou stránku aplikace potom otevřelo 83 uživatelů a byla stažena 25krát. Nutno podotknout, že aplikace nebyla nijak propagována.

Z výsledků uživatelského testování vyplývá, že i přes několik nedostatků, je aplikace funkční a v jisté míře použitelná. Na druhou stranu se v aplikaci nachází prostor pro různá vylepšení. Zaprvé je nutné obohatit samotnou databázi rostlin. Uživatel by v ní v ideálním případě měl vždy najít všechny své rostliny. K tomu je potřeba sehnat všechny potřebné informace, včetně fotografií daných rostlin. Do aplikace by navíc mohla být přidána funkce, kdy by uživatelé mohli navrhovat rostliny k přidání.

Vylepšení, které by mohlo být přidáno, je identifikace rostliny pomocí fotografie. Tato funkcionalita není ale úplně triviální, proto by bylo vhodné použít službu třetí strany, například [77]. Aplikace by však musela být nějakým způsobem monetizována, aby se tuto službu vyplatilo použít.

Dále by také stálo za to, vylepšit a zautomatizovat samotný proces vytváření připomínkových plánů. Brát v úvahu některé další faktory, jako třeba světlo nebo vzdálenost od okna atd. Dále pak také například filtrování rostlin při vyhledávání.

Způsobů, jak monetizovat aplikace, je několik. Lze zpoplatnit samotné stažení, vytvořit předplatné nebo vložit do aplikace reklamy. Nejvhodnějším řešením, v případě aplikace *Sprinkled*, by bylo nejspíše vytvořit prémiový plán (předplatné), který by odemkl veškeré funkce aplikace. Uživatelé, kteří by si plán neplatili, by stále mohli aplikaci využívat, ale byli by například v některých ohledech omezeni (absence rozpoznávání rostlin, maximální počet místností, rostlin, připomínek atd.).

Bibliografie

1. *Houseplant Statistics in 2022 (incl. Covid & Millennials)* [online]. 2022-05-09 [cit. 2022-05-14]. Dostupné z: https://gardenpals.com/houseplant-statistics.
2. *Gardening Statistics in 2022 (incl. Covid & Millennials)* [online]. 2022-05-09 [cit. 2022-05-14]. Dostupné z: https://gardenpals.com/gardening-statistics.
3. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online]. 1994-04-24 [cit. 2022-09-29]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.
4. TEAM, DMC. *10 heuristík použiteľnosti* [online]. 2016-01-27 [cit. 2022-09-29]. Dostupné z: https://www.digitalmag.sk/10-heuristik-pouzitelnosti/.
5. GEKHT, Nikolay. *Create Better Backlog and Engage the Development Team with FURPS.* [Online]. 2020-02-26 [cit. 2022-09-30]. Dostupné z: https://gehtsoftusa.com/blog/create-better-backlog-and-engage-the-development-team-with-furps/.
6. MARTIN, Matthew. *What is a Functional Requirement in Software Engineering? Specification, Types, Examples* [online]. 2022-04-23 [cit. 2022-05-14]. Dostupné z: https://www.guru99.com/non-functional-requirement-type-example.html.
7. MARTIN, Matthew. *What is Non-Functional Requirement in Software Engineering? Types and Examples* [online]. 2022-04-23 [cit. 2022-05-14]. Dostupné z: https://www.guru99.com/non-functional-requirement-type-example.html.
8. NIKOLAIEVA, Aliona. *8 Best Software Development Methodologies* [online] [cit. 2022-09-01]. Dostupné z: https://www.uptech.team/blog/software-development-methodologies.

9. RAULT, Jerome. *Top 7 Software Development Methodologies Every Developer Needs to Know* [online] [cit. 2022-09-01]. Dostupné z: https://www.laneways.agency/top-7-software-development-methodologies.

10. LEERON HOORY, Cassie Bottorff. *What Is Waterfall Methodology? Here's How It Can Help Your Project Management Strategy* [online]. 2020-03-25 [cit. 2022-09-01]. Dostupné z: https://www.forbes.com/advisor/business/what-is-waterfall-methodology.

11. SINGH, Virender. *Agile Methodology* [online]. 2021-09-04 [cit. 2022-09-02]. Dostupné z: https://www.toolsqa.com/agile/agile-methodology.

12. *What Is Scrum Methodology? & Scrum Project Management* [online] [cit. 2022-09-01]. Dostupné z: https://www.digite.com/agile/scrum-methodology.

13. SINGH, Virender. *Feature Driven Development (FDD) : An Agile Methodology* [online]. 2021-07-07 [cit. 2022-09-02]. Dostupné z: https://www.toolsqa.com/agile/feature-driven-development.

14. MARTIN, Matthew. *Prototyping Model in Software Engineering: Methodology, Process, Approach* [online]. 2022-07-09 [cit. 2022-09-02]. Dostupné z: https://www.guru99.com/software-engineering-prototyping-model.html.

15. JENA, Satyabrata. *Lean Software Development (LSD)* [online]. 2021-10-13 [cit. 2022-09-02]. Dostupné z: https://www.geeksforgeeks.org/lean-software-development-lsd/.

16. MARTIN, Matthew. *Spiral Model: When to Use? Advantages and Disadvantages* [online]. 2022-07-09 [cit. 2022-09-02]. Dostupné z: https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html.

17. SEKARPUTRI, Janitra Ariena. *Four Main Keys of Persona in Software Development* [online]. 2020-02-25 [cit. 2022-05-21]. Dostupné z: https://medium.com/inside-ppl-b7/four-main-keys-of-persona-in-software-development-d4df627aeb96.

18. PAVLÍČEK, Josef. *Přednáška 3: UI design steps* [online] [cit. 2022-05-21]. Dostupné z: https://docs.google.com/presentation/d/1CldSHyC8D8cN2LGrqUVJ2U5eEKn5z9MpkFPmzwZX4Zc.

19. *Use Cases* [online] [cit. 2022-05-27]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/use-cases.html.

20. ČÁPKA, David. *Lekce 2 - UML - Use Case Diagram* [online] [cit. 2022-05-27]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram.

21. ČÁPKA, David. *Lekce 4 - UML - Doménový model* [online] [cit. 2022-06-03]. Dostupné z: `https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram`.
22. FOWLER, Martin. Domain model. In: *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002, s. 116–124. ISBN 0321127420.
23. RABELO, Juliano. *Three-Tier Architecture* [online] [cit. 2022-05-28]. Dostupné z: `https://www.techopedia.com/definition/24649/three-tier-architecture`.
24. *Třívrstvá architektura (Three-tier architecture)* [online] [cit. 2022-05-28]. Dostupné z: `https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture`.
25. *DB-Engines Ranking* [online] [cit. 2022-05-28]. Dostupné z: `https://db-engines.com/en/ranking`.
26. *PostgreSQL* [online] [cit. 2022-05-28]. Dostupné z: `https://www.postgresql.org`.
27. PETERSON, Richard. *What is PostgreSQL? Introduction, Advantages & Disadvantages* [online]. 2022-04-16 [cit. 2022-05-28]. Dostupné z: `https://www.guru99.com/introduction-postgresql.html`.
28. *NestJS* [online] [cit. 2022-05-29]. Dostupné z: `https://nestjs.com/`.
29. *NestJS – backend framework pro webové aplikace* [online]. 2021-03-27 [cit. 2022-05-29]. Dostupné z: `https://visionslabs.io/nestjs-backend-framework-pro-webove-aplikace`.
30. MCARTHUR, Ann. *Why Use Nest.js* [online]. 2022-11-16 [cit. 2022-05-29]. Dostupné z: `https://devcycle.com/blog/why-use-nest-js`.
31. *Prisma* [online] [cit. 2022-05-30]. Dostupné z: `https://www.prisma.io/`.
32. PRASAD, Madhusha. *A Next-Generation ORM: Prisma* [online]. 2021-12-18 [cit. 2022-05-30]. Dostupné z: `https://medium.com/sliit-foss/a-next-generation-orm-prisma-3c3f2b46bd5b`.
33. MARCHUK, Anastasiya. *Native vs Cross-Platform Development: Pros & Cons Revealed* [online] [cit. 2022-05-30]. Dostupné z: `https://www.uptech.team/blog/native-vs-cross-platform-app-development`.
34. JEOREN, L. *UIKit vs. SwiftUI: How to Choose the Right Framework for Your App* [online]. 2021-03-27 [cit. 2022-05-30]. Dostupné z: `https://getstream.io/blog/uikit-vs-swiftui`.
35. *App Store* [online] [cit. 2022-05-30]. Dostupné z: `https://developer.apple.com/support/app-store`.
36. PING, Quek Wei. *Model-View-ViewModel Pattern explained* [online]. 2020-12-20 [cit. 2022-06-01]. Dostupné z: `https://isaacquek97.medium.com/model-view-viewmodel-pattern-explained-a9a6a4c06773`.

37. KHAN, Aasif. *MVVM and SwiftUI* [online]. 2021-12-03 [cit. 2022-06-01]. Dostupné z: https://www.appypie.com/mvvm-swiftui-how-to.

38. *4 Most Used REST API Authentication Methods* [online]. 2019-07-26 [cit. 2022-06-13]. Dostupné z: https://blog.restcase.com/4-most-used-rest-api-authentication-methods.

39. LO, Victoria. *Introduction to REST API Authentication Methods* [online]. 2020-08-22 [cit. 2022-06-13]. Dostupné z: https://lo-victoria.com/introduction-to-rest-api-authentication-methods.

40. *JWT* [online] [cit. 2022-06-13]. Dostupné z: https://jwt.io.

41. LO, Megan. *RESTful API 101* [online]. 2021-07-05 [cit. 2022-06-13]. Dostupné z: https://medium.com/geekculture/restful-api-101-b61671e5a3ea.

42. *What is a REST API?* [Online]. 2020-05-08 [cit. 2022-06-13]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

43. TEAM, Indeed Editorial. *What Is User Interface (UI)?* [Online]. 2022-06-01 [cit. 2022-11-21]. Dostupné z: https://www.indeed.com/career-advice/career-development/user-interface/.

44. BABICH, Nick. *Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each* [online]. 2017-11-29 [cit. 2022-11-21]. Dostupné z: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use.

45. SHUHALII, Avrora. *What is the difference between low and high fidelity prototypes?* [Online]. 2020-10-16 [cit. 2022-11-21]. Dostupné z: https://bootcamp.uxdesign.cc/what-is-the-difference-between-low-and-high-fidelity-prototypes-b1f3612f85f7.

46. *Prototypes* [online]. 2021 [cit. 2022-11-21]. Dostupné z: https://www.jyst.agency/services/prototypes/.

47. *Figma* [online] [cit. 2022-11-21]. Dostupné z: https://www.figma.com/.

48. *Sprinkled Figma Prototype* [online] [cit. 2022-12-01]. Dostupné z: https://www.figma.com/file/jtO9UMSyElHImRQy0quigO/.

49. BABICH, Nick. *Bottom Tab Bar Design Best Practices* [online]. 2022-09-30 [cit. 2022-12-01]. Dostupné z: https://uxplanet.org/bottom-tab-bar-design-best-practices-ef3ee71de0fc.

50. *2022 Developer Survey* [online] [cit. 2022-12-02]. Dostupné z: https://survey.stackoverflow.co/2022/.

51. *Visual Studio Code* [online] [cit. 2022-12-02]. Dostupné z: https://code.visualstudio.com/.

52. CANGEMI, Denis. *The Reasons why you Must Use Visual Studio Code* [online]. 2020-07-29 [cit. 2022-12-02]. Dostupné z: https://blog.devgenius.io/the-reasons-why-you-must-use-visual-studio-code-b522f946a849.

53. *Xcode* [online] [cit. 2022-12-02]. Dostupné z: https://developer.apple.com/xcode/.

54. *Docker* [online] [cit. 2022-12-02]. Dostupné z: https://www.docker.com/.

55. GUPTA, Shashvat. *Docker 101: All you wanted to know about Docker* [online]. 2020-07-01 [cit. 2022-12-02]. Dostupné z: https://towardsdatascience.com/docker-101-all-you-wanted-to-know-about-docker-2dd0cb476f03/.

56. *Insomnia* [online] [cit. 2022-12-02]. Dostupné z: https://insomnia.rest/.

57. *DataGrip* [online] [cit. 2022-12-02]. Dostupné z: https://www.jetbrains.com/datagrip/.

58. *Git* [online] [cit. 2022-12-02]. Dostupné z: https://git-scm.com/.

59. *Password Hashing Competition* [online] [cit. 2022-12-03]. Dostupné z: https://www.password-hashing.net/.

60. *argon2* [online] [cit. 2022-12-03]. Dostupné z: https://github.com/ranisalt/node-argon2/.

61. *node-apn* [online] [cit. 2022-12-03]. Dostupné z: https://github.com/parse-community/node-apn/.

62. *class-transformer* [online] [cit. 2022-12-03]. Dostupné z: https://github.com/typestack/class-transformer/.

63. *class-validator* [online] [cit. 2022-12-03]. Dostupné z: https://github.com/typestack/class-validator/.

64. *Helmet* [online] [cit. 2022-12-03]. Dostupné z: https://helmetjs.github.io/.

65. *parse-duration* [online] [cit. 2022-12-03]. Dostupné z: https://github.com/jkroso/parse-duration/.

66. *PactumJS* [online] [cit. 2022-12-03]. Dostupné z: https://pactumjs.github.io/.

67. *Jest* [online] [cit. 2022-12-05]. Dostupné z: https://jestjs.io/.

68. *Heroku* [online] [cit. 2022-12-05]. Dostupné z: https://www.heroku.com/.

69. *Kingfisher* [online] [cit. 2022-12-10]. Dostupné z: https://github.com/onevcat/Kingfisher.

70. *Swift Collections* [online] [cit. 2022-12-11]. Dostupné z: https://github.com/apple/swift-collections.

71. *JWTDecode* [online] [cit. 2022-12-10]. Dostupné z: https://github.com/auth0/JWTDecode.swift.

72. *PopupView* [online] [cit. 2022-12-11]. Dostupné z: https://github.com/exyte/PopupView.

73. *SwiftUI-Shimmer* [online] [cit. 2022-12-11]. Dostupné z: https://github.com/markiv/SwiftUI-Shimmer.

74. HERTACH, Hannes. *SwiftUI Dependency Injection in 20 lines of code* [online]. 2020-05-01 [cit. 2022-12-11]. Dostupné z: https://medium.com/@hanneshertach/swiftui-dependency-injection-in-20-lines-b322457065a5.

75. *Sprinkled* [online] [cit. 2022-12-11]. Dostupné z: https://apps.apple.com/us/app/sprinkled/id6443843055.

76. NIELSEN, Jakob. *How Many Test Users in a Usability Study?* [Online]. 2012-06-03 [cit. 2022-12-14]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/.

77. *Plant.id API* [online] [cit. 2022-12-25]. Dostupné z: https://web.plant.id/plant-identification-api/.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

APNs Apple Push Notification service

CLI Command Line Interface

CRUD Create, Read, Update, Delete

DI Dependency Injection

E2E End-to-end

FR Functional Requirement

gRPC Google Remote Procedure Calls

Hi-Fi High Fidelity

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

JSON JavaScript Object Notation

JWT JSON Web Token

Lo-Fi Low Fidelity

MVC Model-view-controller

MVP Minimum Viable Product

MVVM Model-view-viewmodel

NFR Non-functional Requirement

ORM Object-relational Mapping

PaaS Platform as a Service

PDF Portable Document Format

REST Representational State Transfer

SOAP Simple Object Access Protocol

SQL Structured Query Language

UC Use Case

UI User Interface

UML Unified Modeling Language

UX User Experience

XML Extensible Markup Language

Příloha B

Obsah přiloženého média

- `src`
  - `backend` ..... zdrojové kódy serverové části aplikace
  - `ios` ..... zdrojové kódy iOS aplikace (klientská část)
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `test-recordings` ..... záznamy z uživatelského testování
- `thesis.pdf` ..... text práce ve formátu PDF