Diplomová práce

F3 Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Integrace senzoru vlhkosti půdy

Diplomová práce

Bc. Kateřina Dlouhá

Vedoucí práce: RNDr. Ondřej Žára
Obor: Otevřená informatika
20. května 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dlouhá** Jméno: **Kateřina** Osobní číslo: **466336**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Integrace senzoru vlhkosti půdy

Název diplomové práce anglicky:

Soil moisture sensor integration

Pokyny pro vypracování:

Hlavním cílem této práce je rozšíření funkcionality existující aplikace z předchozí bakalářské práce 'Progresivní webová aplikace pro monitorování a údržbu rostlin'.
Seznamte se se základními koncepty práce s mikrokontroléry z rodiny 'Internet of Things', následně navrhněte a sestavte autonomní senzor pro měření vlhkosti půdy. Uvažte, jakým způsobem lze takové zařízení připojit k Internetu a jaké jsou možnosti správy jím generovaných dat. Naimplementujte ukládání měřených dat tak, aby tato byla k dispozici výše uvedené webové aplikaci.
Doplňte do webové aplikace funkci sledování vlhkosti půdy. Popište technologii komunikace mezi aplikací a ukládanými daty, zohledněte i otázku zabezpečení těchto dat. Aplikaci dále rozšiřte a vylepšete i s ohledem na plány z předchozí práce, tj. zejména sdílení (Web Share API) a možnost přijímání notifikací (Push API, Web Notifications).
Popište limity stávajícího řešení: míru závislosti na službách třetích stran, spolehlivost a kapacitu datového úložiště, konektivitu a energetickou náročnost půdního senzoru. Navrhněte, jak proces zapojení a konfigurace půdního senzoru učinit uživatelsky vstřícným; otestujte tuto aktivitu v rámci kvalitativního uživatelského testování.

Seznam doporučené literatury:

https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en
.pdf
https://arduino-esp8266.readthedocs.io/en/latest/
https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api

Jméno a pracoviště vedoucí(ho) diplomové práce:

RNDr. Ondřej Žára, Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **11.01.2021** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2022**

RNDr. Ondřej Žára		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studentky

Poděkování

Ráda bych poděkovala všem, kteří se podíleli na závěrečné práci. Jmenovitě, Bc. Marku Janskému, za doplnění mého chabého vzdělání v oblasti elektroniky a konzultace při implementaci senzoru. Vedoucímu práce, RNDr. Ondřeji Žárovi, za vedení práce, cenné rady a materiální podporu pro sestavení senzoru.

Prohlášení

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

V Praze, 20. května 2021.

Abstrakt

Cílem diplomové práce je rozšíření předchozí bakalářské práce 'Progresivní webová aplikace pro monitorování a údržbu rostlin'. Stěžejní změnou je integrace senzoru pro měření vlhkosti půdy a zobrazení naměřených dat. Na základě zpětné vazby z předešlého uživatelského testování použitelnosti rozšířit aplikaci o push notifikace a umožnit sdílení zpráv prostřednictvím Web Share API. Součástí práce je i sestavení prototypu senzoru vlhkosti půdy, kterému předchází rešerše týkající se oblasti Internet of Things. Rozšíření aplikace je následně podrobeno uživatelskému testování použitelnosti.

Klíčová slova: IoT, senzor půdní vlhkosti, MQTT, mikroprocesor ESP8266, Arduino, SPA, PWA, testování použitelnosti

Vedoucí práce: RNDr. Ondřej Žára

Abstract

The aim of this diploma thesis is to extend the previous bachelor thesis 'Progressive web application for plant monitoring and maintenance'. A key change is the integration of a sensor for measuring soil moisture and displaying measured data. Based on feedback from previous usability testing, extend the application with push notifications and enable message sharing via the Web Share API. Part of the work is also the compilation of a prototype soil moisture sensor, which is preceded by a research in the field on Internet of Things. The application extension is then subjected to user usability testing.

Keywords: IoT, soil moisture sensor, MQTT, microprocessor ESP8266, Arduino, SPA, PWA, usability testing

Title translation: Soil moisture sensor integration — Diploma thesis

Obsah

Obrázky

Tabulky

Fragmenty kódu

Kapitola 1
Úvod

Motivací pro závěrečnou práci bylo rozšíření bakalářského projektu na téma Progresivní webová aplikace na údržbu a monitorování rostlin. Tato aplikace umožňovala práci s databází rostlin, vytváření upomínek na zalévání a pořizování snímků rostliny. Co je však pro údržbu rostlin klíčové a může být přínosné, je monitorování stavu vlhkosti půdy.

Díky zpětné vazbě z testování použitelnosti v rámci bakalářské práce byla identifikována potenciální zlepšení aplikace [20], z nichž se některá stala předmětem zadání diplomové práce.

1.1 Cíle práce

Práce se skládá ze dvou klíčových částí. Implementace prototypu senzoru na měření vlhkosti půdy a s tím i spjaté seznámení se s oblastí mikroelektroniky a Internet of Things. Na tuto část naváže rozšíření progresivní webové aplikace o některé z námětů z uživatelského testování. Jmenovitě se jedná o zobrazení aktuálně naměřených dat ze senzoru, vylepšené notifikace s upomínkou na údržbu rostliny, sdílení upomínky na zalévání a přístup k webové aplikaci z vícero zařízení.

Nebylo předpokladem, že by se senzor významně lišil, ba dokonce předčil současná řešení, která jsou dostupná na trhu. Cílem implementace prototypu senzoru bylo hlubší seznámení autora práce s problematikou týkající se Internet of Things a zpřístupnit data ze senzoru pro rozšíření bakalářské práce.

Kapitola 2
Internet of Things

Pojem Internet of Things, dále jen IoT, vznikl v roce 1999, když se Kevin Ashton, britský technologický inovátor, snažil vymyslet vystihující název prezentace pro novou éru v oblasti informačních technologií. Jeho snahou bylo přesvědčit vedení z Procter & Gamble, aby zjednodušili práce ve skladě. Navrhl systém, který pomocí identifikačních štítků a skeneru uměl během chvíle získat potřebné informace o umístění a stavu zboží, což bylo před více než dvaceti lety poměrně revoluční řešení. [37]

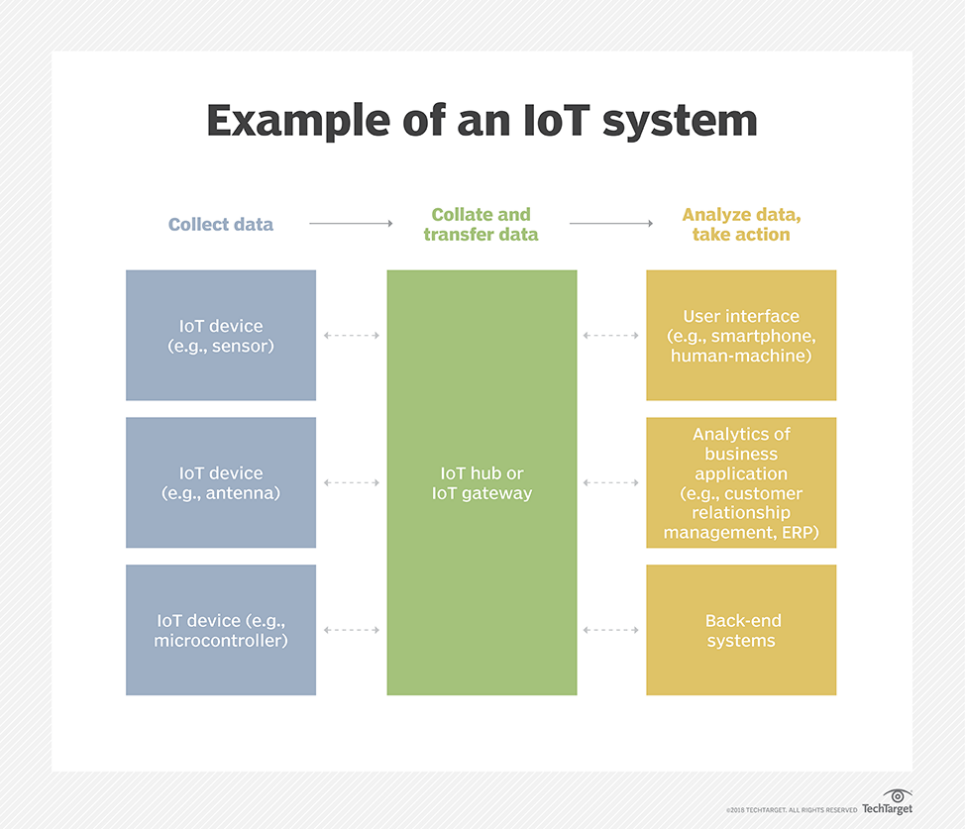
Internet of Things lze chápat jako ekosystém, ve kterém jednotlivé subjekty (též věci či zařízení) mezi sebou sdílí data, na základě kterých mohou být vykonávány příslušné akce podle zabudovaných pravidel. Subjektem může být jakékoliv výpočetní zařízení, digitální stroj, zvíře nebo člověk, kterému je přidělen unikátní identifikátor (UID). Subjekt má tu vlastnost, že umí přenášet data po síti bez nutnosti interakce člověka. Do IoT spadá například osoba s implantátem monitoru srdce, automobil se zabudovaným senzorem upozorňující na nízký tlak v pneumatikách nebo senzory monitorující prostředí rostliny. Velmi zjednodušeně řečeno – kterýkoliv objekt, kterému lze přiřadit IP adresu a umí přenášet data po síti.

Princip IoT může být demonstrován na následujícím příkladu monitoringu zahřívacího stroje, viz obrázek 2.1. V typickém scénáři by teplotní senzor zobrazoval data na analogové nebo digitální obrazovce, která by někdo musel číst a na základě zobrazených hodnot vykonávat určité akce. V případě IoT se odstiňuje veškerá režie spjatá s přítomností fyzické osoby v procesu. Zodpovědnost se deleguje systému, který má za úkol vyhodnotit získaná data ze senzoru a na základě zabudovaných pravidel provést příslušnou akci.

2.1 Komunikace a přenos dat

Celý ekosystém se z pohledu dat skládá ze tří fází:

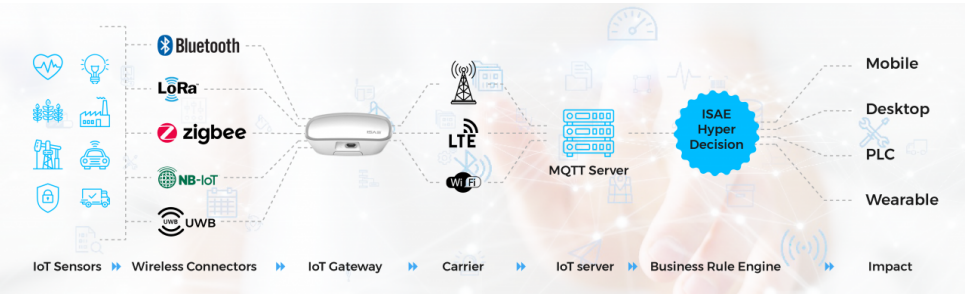
- odeslání,
- shromaždění,
- vyhodnocení.



Obrázek 2.1: Zjednodušený diagram, jak lze chápat IoT z pohledu toku dat [37]

Získaná data jsou shromážděna před připojením k bráně (gateway) IoT nebo jinému zprostředkovateli, který data následně zanalyzuje či odešle k analýze na cloudové úložiště. Data se následně vyhodnotí na cílovém zařízení či zařízeních.

V následujících kapitolách je podrobněji rozebírána architektura celého ekosystému včetně spjatých technologií.



Obrázek 2.2: Detailnější vizualizace architektury ekosystému IoT [33]

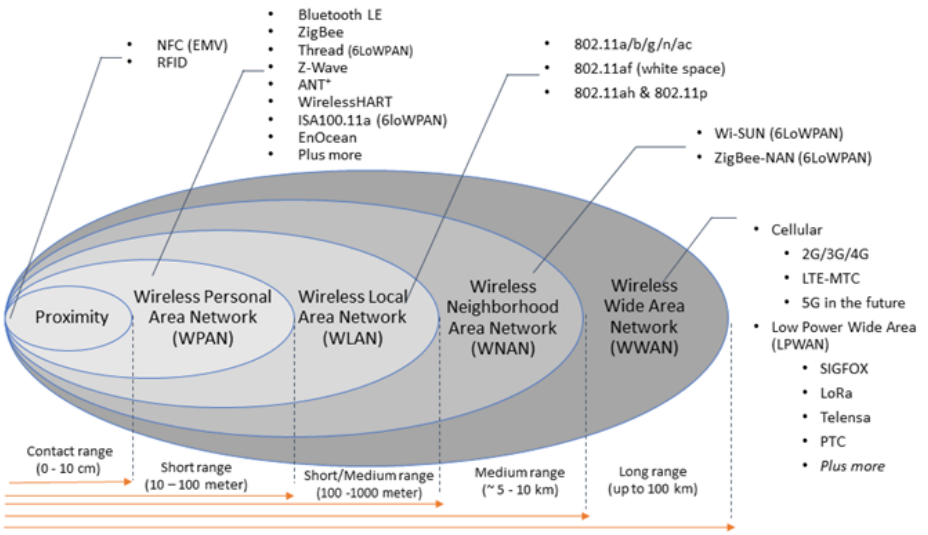
2.2 Připojení k síti

Mezi senzorem a koncovým zařízením probíhá jednosměrná či obousměrná komunikace. V tomto procesu figurují jako prostředníci IoT brána a například MQTT server. Ze senzoru jsou prostřednictvím rádiových frekvencí, jako jsou například BLE, LoRa nebo ZigBee, odesílány informace přímo k bráně, která je následně rozešle pomocí GPRS, Wi-Fi nebo LTE přímo koncovému zařízení či je uloží na cloud.

Tok dat může vypadat zhruba následovně:

- Ze senzorů se pomocí SRWC (Short-Range Wireless Communication) vysílají data, která zachytí místní gateway.
- Gateway prostřednictvím LPWAN (Low Power Wide Area Network) odešle získaná data ze senzoru směrem ke stanici, která umožní přeposílání dat mimo lokální síť. Díky tomu se data z podsítě dostanou do internetu a cloudu.
- Z hlavní stanice se díky LTE nebo Wi-Fi distribuují data směrem ke cloudu.
- Cílové zařízení si potom prostřednictvím LTE či Wi-Fi stáhne data ze serveru a zobrazí je v uživatelsky přívětivé podobě.

Během komunikace s koncovými subjekty probíhá sdílení dat v rámci různých vzdáleností a tudíž i na jiné vrstvě síťové komunikace. Pro stručnou ukázku je zde přiložen diagram 2.3, který člení protokoly bezdrátových sítí podle jejich rozsahu komunikace.



Obrázek 2.3: Členění bezdrátových sítí včetně uvedených protokolů a rozsahů komunikace [45]

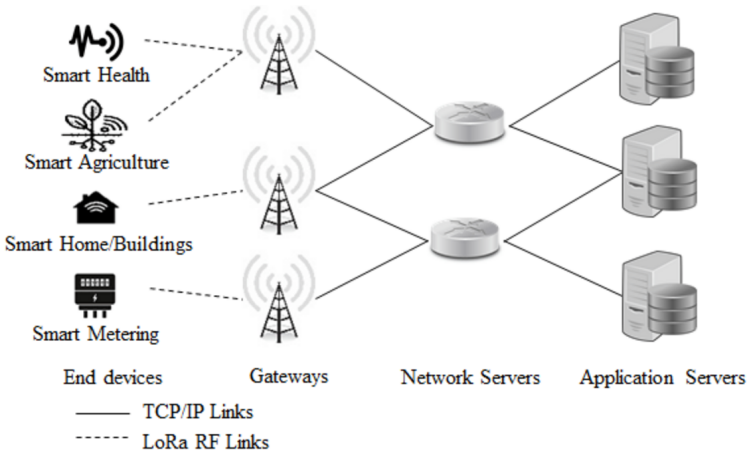
SRWC

Short-Range Wireless Communication, neboli bezdrátová komunikace na krátké vzdálenosti (do zhruba 1 km). Do této kategorie spadají všechny protokoly, které jsou schopny komunikovat na úrovni WPAN (Wireless Personal Area Network) a WLAN (Wireless Local Area Network).

LPWAN

Low Power Wide Area Network lze přeložit jako nízkoenergetickou širokopásmovou síť. Jedná se o bezdrátovou síť, která umožňuje dálkový přenos dat s nízkou bitovou rychlostí mezi zařízeními, které nemají velkou kapacitu baterie, resp. musí být pravidelně dobíjeny – např. senzory. Velkou výhodou této sítě je tím pádem nízká spotřeba energie a nízké provozní náklady, což je pro komunikaci senzor-cloud vítaný benefit.

Do této kategorie například spadají protokoly LoRA, Sigfox, LTE-M nebo NB-IoT.



Obrázek 2.4: Schéma komunikace v případě LPWAN [32]

2.2.1 IoT cloud

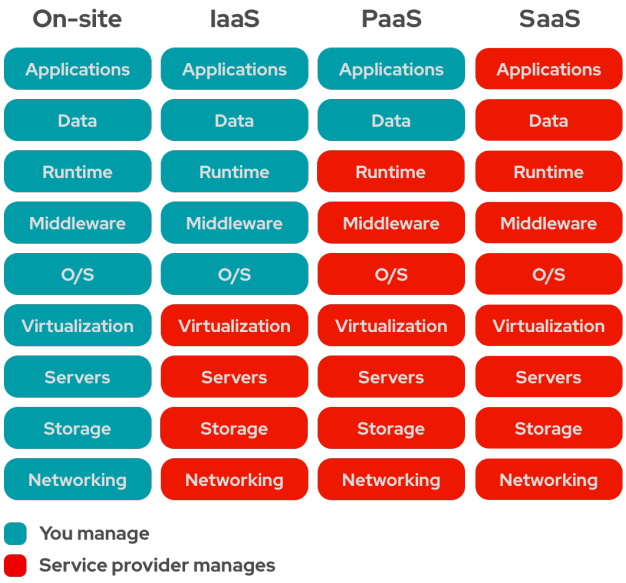
Cloud představuje úložiště dat, ke kterému mají uživatelé nepřetržitý přístup. Data i služby jsou poskytovány prostřednictvím serverů uložených v datových centrech – uživatel tím pádem neví, kde přesně jsou data umístěna.

Senzory IoT mají omezené vlastnosti a jejich primárním úkolem je zejména bezdrátové odesílání, případně přijímání, bitů na fyzické vrstvě. K přenosu dat obvykle používají protokol MQTT. Přijímá a přenáší informace každému, kdo se k jejich odběru přihlásí. MQTT server je obecně označován jako IoT server. Od běžného serveru se liší v tom, že je schopen zpracovávat velmi rychle odesílaná data ze senzorů.

Od roku 2015 se rapidně navýšil počet poskytovatelů vzdálených úložišť pro IoT. [22] Při jejich výběru hraje často roli i hardware, se kterým má cloud komunikovat. Hledaným kompromisem je taková varianta, která umožňuje rychlý a bezpečný přístup k datům.

Na trhu jsou nabízené různé typy cloudu, viz obrázek 2.5. Služby se liší

podle působnosti (regionální, celosvětová), úrovní poskytovaných služeb a formou distribuce, tj. co nabízí – samotný software, hardware či jejich kombinaci. O tom vypovídá distribuční model, který lze rozdělit do třech kategorií:



Obrázek 2.5: Rozdíl mezi jednotlivými cloudovými službami podle distribučního modelu [34]

IaaS (Infrastructure as a Service/Integration as a Service)
Jedná se po poskytování služeb včetně infrastruktury (např. virtualizaci). O veškeré problémy s hardwarem se stará poskytovatel. Příkladem poskytovatelů může být Virtuozzo, KVM, OpenStack.

PaaS (Platform as a Service)
Poskytovatel v tomto případě garantuje kompletní prostředky pro podporu celého životního cyklu (zejména webových) aplikací a služeb. Jsou zde zahrnuty i různé nástroje pro vývoj aplikací (IDE, API) nebo pro údržbu produktu. Řeč je o GitHub, GitLab, Docker, Kubernetes a dalších.

SaaS (Software as a Service)
Produkt je licencován jako služba a pronajímána zákazníkovi. Zákazník si tím pádem nekupuje aplikaci samotnou, ale pouze přístup k ní. Tento produkt nabízí například Verizon, CloudBlue nebo Cloud Scripting.

V současné době existují i další IoT řešení cloudových služeb:

- Microsoft Azure
- Google Cloud Platform
- Oracle Cloud

- IBM Bluemix

2.2.2 Hyper Decision Framework – ISAE

Data z cloudového serveru se odesílají pomocí rámce ISAE. Tento rámec představuje sadu pravidel vytvořených uvnitř rozhodovacího systému. Analyzuje přijaté informace, které následně mapuje proti sadě zadefinovaných pravidel, která se mohou vzájemně překrývat. Pravidlem je myšlena nějaká akce, např.: pokud uživatel vstoupí do místnosti A, rozsvítit světla 1 a 2. Provedení rozhodnutí spojeného s pravidlem je označováno jako *hyper decision framework*.

2.2.3 User Interface

Pro usnadnění práce s ekosystémem by mělo existovat uživatelské rozhraní, které koncové osobě pomáhá komunikovat se senzory. Může se jednat o jednoduchou webovou nebo mobilní aplikaci.

Existují i obecné požadavky na tato rozhraní. Jedná se o zobrazení dat v reálním čase, možnost manipulace se senzory, sdílení dat a další operace, které například získaná data analyzují nebo vizualizují.

2.3 Bezpečnost a soukromí

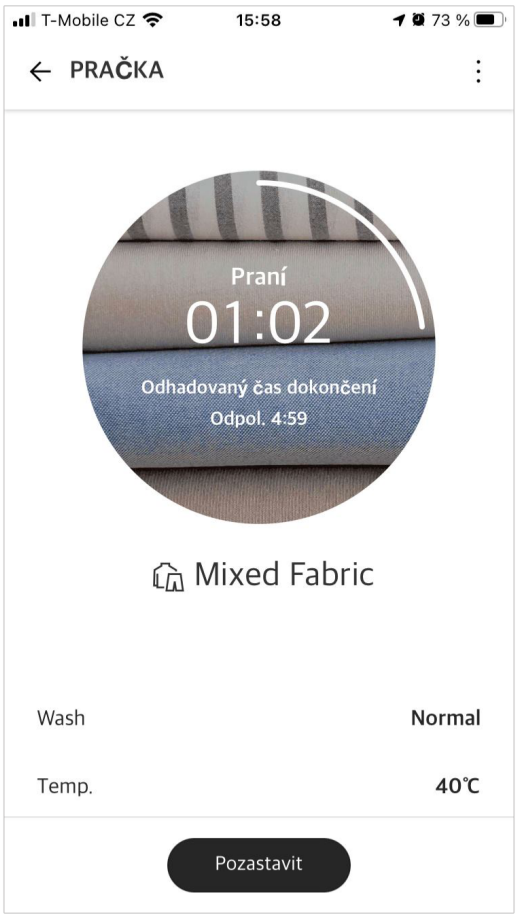
Narušení integrity dat představuje velké bezpečnostní riziko u IoT. I přes to, že některé senzory shromažďují citlivá data, dosavadní výsledky z analýz komunikace ukazují, že je v tomto směru co zlepšovat. [15] Mnoho firem nevěnuje dostatečnou pozornost základním bezpečnostním opatřením, jako je šifrování přenosu dat. Přes tyto bezpečnostní slabiny lze poměrně snadno napadnout zařízení jako jsou kamery nebo senzory zvuku. Útočníci tak mohou odposlouchávat komunikaci a aktivně se jí i účastnit.

Má-li uživatel ve své domácnosti například smart home produkty nebo senzory pohybu, veškerá jimi získaná data jsou zálohována na vzdálených úložištích. Z těchto dat je mnohdy reálné získat nepřeberné množství citlivých informací o členech domácnosti.

V současnou chvíli se postupně zavádí nová bezpečnostní opatření, aby se výše zmíněným rizikům předcházelo. Pomoct by měla jedinečná hesla, která budou mít koncová zařízení a pravidelné aktualizace softwaru. Apel směřuje i na společnosti vyrábějící produkty pro chytrou domácnost, aby se více zabývaly otázkou bezpečnosti a soukromí.

2.4 Výhody a nevýhody

Mezi výhody IoT patří možnost odkudkoliv a kdykoliv přistupovat k informacím ze zařízeních, automatizace úkolů, které pomáhají zlepšovat kvalitu služeb nebo chodu domácnosti a snižují tak nutnost lidské intervence.



Obrázek 2.6: Příklad uživatelského rozhraní IoT ekosystému – LG ThinQ

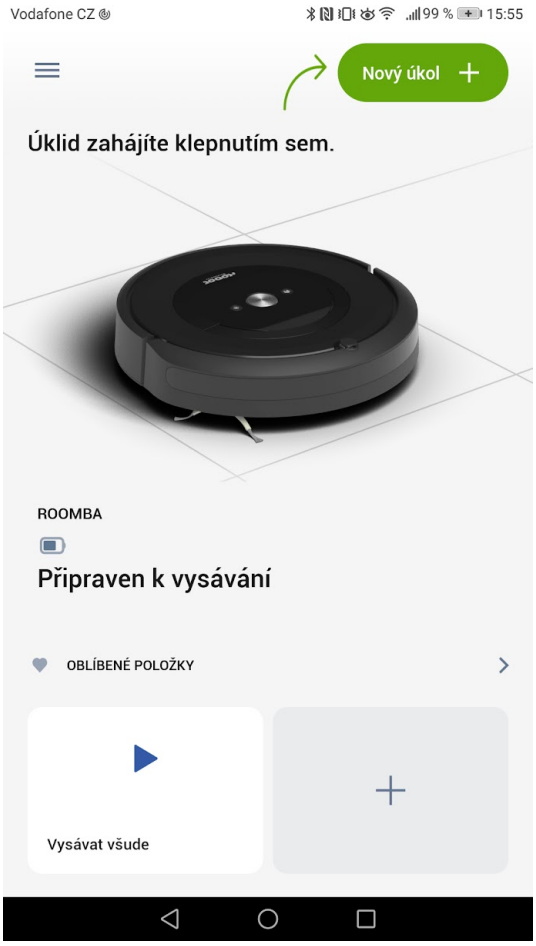
Nevýhodami jsou zmíněná integrita dat a bezpečnost. Pro podniky nastává problém v kolekci dat ze strmě stoupajícího počtu všech zařízení, které používají. Problém je i s kompatibilitou, jelikož v současné době neexistuje žádný mezinárodní standard, který by ji napříč zařízeními různé značky stanovoval. [37] Posledním bodem jsou systémové chyby, které mají často na svědomí i poškození každého zainteresovaného zařízení v komunikaci.

2.5 Reálné využití

Internet of Things je hojně využíváno v potravinářském průmyslu, zdravotnictví, informačních technologií, ve školství i v každodenním životě. Namátkou lze zmínit tyto scénáře:

Sledování palet ve skladě

Mnoho organizací recykluje používané palety. Díky tomu, že sklady každoročně rozšiřují svůj sortiment a tím i požadavky na prostor, umístění BLE senzorů na palety může urychlit jejich hledání v rozlehlých prostorách skladu.



Obrázek 2.7: Příklad uživatelského rozhraní IoT ekosystému – iRobot

Monitorování zaměstnanců

Monitorování zaměstnanců na pracovišti může přispět ke zvýšení produktivity, morálky či bezpečnosti.

Zaměstnanec nosí BLE senzor ve formě identifikační karty, případně čipu. Tato identifikace může autorizovat osobu při přístupu do budov, sektorů nebo zasedacích místností. Mimo jiné může snímat, zda je identifikační čip na těle osoby a sledovat tak, kde se osoba vyskytuje a sledovat její produktivitu.

Monitorování teploty a vlhkosti

Cold chain, neboli chladící řetězec, označuje řadu základních podmínek bezpečnosti potravin a produktů, které je nutné dodržovat k jejich udržení v předepsaném teplotním rozmezí pro dané období. Senzory IoT nachází v této oblasti hojné využití. Díky nim je možné sledovat teplotu a vlhkost uvnitř kontejnerů či provozoven, ve kterých se produkty nachází, a vzdáleně řešit vyskytující se problémy v reálném čase.

2.6 Senzory

Senzory, někdy ještě historicky označovány jako čidla či snímače, představují stěžejní kámen v oblasti IoT. Jedná se o elektronickou součástku, která je schopna měřit veličiny ze vstupních energetických domén okolního světa. Domény mohou být magnetické, mechanické, tepelné, chemické a další. Senzor poskytuje změřené hodnoty dalším vrstvám, které zpracovávají signál pro vyšší komunikační vrstvy.

Senzor na měření vlhkosti půdy

Nejčastěji se vlhkost půdy měří pomocí kapacitních senzorů nebo metody pulsní reflektometrie.

Senzory fungující na základě kapacitní metody se skládají ze dvou nebo více elektrod umístěných přímo v půdě. Pokud jsou elektrody připojeny ke zdroji elektrického napájení, jsou schopny vytvářet elektrické pole. Kapacita kondenzátoru závisí především na vlhkosti půdy.

Obrázek 2.8: Kapacitní senzor pro měření vlhkosti půdy [43]

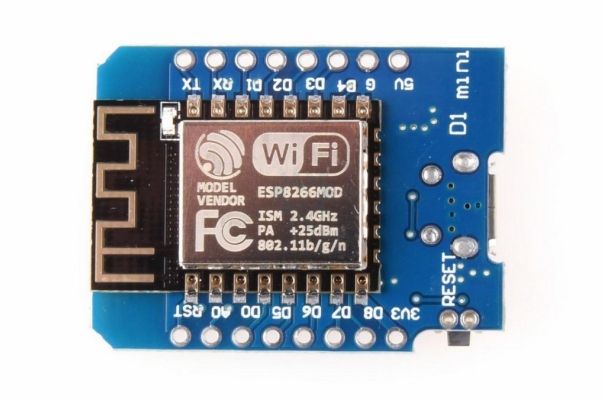
Vedle kapacitních senzorů existují TDR senzory. TDR (Time Domain Reflectometry) je metoda pulsní rerflektometrie. Zařízení se skládá ze dvou až tří elektrod, respektive zářičů, které musí být stejně jako v případě kapacitoru umístěny přímo v půdě. Pomocí rychlosti šíření vysokofrekvenčního elektromagnetického impulsu se určuje permitivita půdy, která je následně přepočítána na půdní vlhkost pomocí empiricky stanovených vzorců. [23]

Obrázek 2.9: TDR senzor pro měření vlhkosti půdy [29]

2.7 Mikrokontrolery pro IoT systémy

Mikrokontrolery (zkráceně MCU) představují pro IoT systémy stěžejní komponentu, která zprostředkovává komunikaci mezi koncovými zařízeními. Jedná se o malé počítače embedované na mikročipu, které se skládají z procesoru, paměti a programovatelného hradla pro vstupní a výstupní periferie. Velmi často dochází k jejich záměně s mikroprocesory, které narozdíl od mikročipů neobsahují žádnou paměť či procesor.

Mikrokontrolery bývají součástí tzv. vývojové desky (*development board*), kde navíc obsahují napájení, podporu pro připojení senzorů a jiné komponenty, které usnadňují práci s MCU. Jsou více než vhodné pro rychlé a snadné prototypování vlastní mikroelektroniky.



Obrázek 2.10: Ukázka vývojové desky WeMos D1 Mini ESP8266 WiFi [28]

Při výběru vhodného mikrokontroleru se mimo pořizovací cenu zvažují následující vlastnosti:

- **Kompatibilita** – Záruka mezi MCU a senzory není samozřejmostí.

Z toho důvodů je nutné ověření kompatibility mezi komponentami. Opomenout by se neměl ani počet dostupných vstupních a výstupních portů (zkráceně I/O porty) i s ohledem do budoucna kvůli případné škálovatelnosti řešení.

- **Paměť** – Stejně jako počet bitů hraje paměť klíčovou roli pro celkový výkon a rychlost zpracování operací. Většina mikrokontrolerů se skládá z RAM, ROM a Flash paměti. Funkcí RAM paměti je čtení a zápis dat, naopak ROM se stará o provádění instrukcí. Flash paměť narozdíl od ROM a RAM pracuje jako offline paměť, tj. pro její fungování není potřeba připojení k napájení.

- **Spotřeba energie** – V oblasti IoT se jedná o velmi důležité kritérium. Cílem každého systému je minimalizovat veškeré energetické náklady z důvodu spolehlivosti zařízení a dlouhodobé životnosti. Za účelem snížení spotřeby se MCU při delší nečinnosti uspávají a probouzí se k provedení úkolů, což výrazně energetickou náročnost.

- **Zabezpečení** – Standardním zabezpečením na většině deskách je šifrování dat, případně aplikování tzv. *shield layers*, neboli štítových vrstvách, které chrání zařízení před narušením integrity.

- **Vývojové nástroje, dokumentace a komunita** – Zejména pro jedince bez zkušeností se jedná o důležité kritérium. Dobrá dokumentace a intuitivní vývojové nástroje usnadňují implementaci, zlepšují porozumění produktu a přispívají k oblibě mezi uživateli.

Nejznámějšími výrobci mikrokontrolerů jsou společnosti Raspberry Pi či Arduino.

2.8 MQTT a přenos dat po síti

Pro přenos dat mezi IoT zařízeními a klientem je zapotřebí přenášet malý objem dat co nejrychleji. Senzory by v procesu komunikace měly pouze přijímat či odesílat data – veškerá logika by se měla odehrávat v jiné části architektury. Většinou se může jednat o řídící systémy, které vyhodnocují a následně zpracují přijatá data od jednoho z koncových zařízení.

Zpočátku se pracovalo s protokolem HTTP, který se jevil jako nejlogičtější volba. Postupem času se ukázalo, že tento protokol má řadu nevýhod pro využití v IoT – nejsou vyřešené úrovně Quality of Service pro dvě zařízení, vyžaduje nutnost serverové strany, která umožní vzájemnou komunikaci mezi koncovými zařízeními a navíc je poměrně komplexní.

MQTT (dříve Message Queuing Telemetry Transport, dnes MQ Telemetry Transport) řeší nedostatky protokolu HTTP. MQTT je jednoduchý, přímočarý a datově úsporný protokol, který umožňuje asynchronně přenášet data mezi zařízeními s nestabilním připojením nebo jsou omezeny na energetickou spotřebu.

Narozdíl od HTTP, jehož architektura je založena na modelu zpracování požadavků a odpovědí, funguje na základě public-subscribe patternu, což přispívá k rychlejší komunikaci mezi zařízeními. Zprávy jsou předávány v binárním formátu, velikost hlavičky je definována na dva bajty a poskytuje lepší zabezpečení přenosu dat, které HTTP jako takové neposkytuje (za to HTTPS ano). Níže je uvedena tabulka 2.1 rozdílů mezi jednotlivými protokoly.

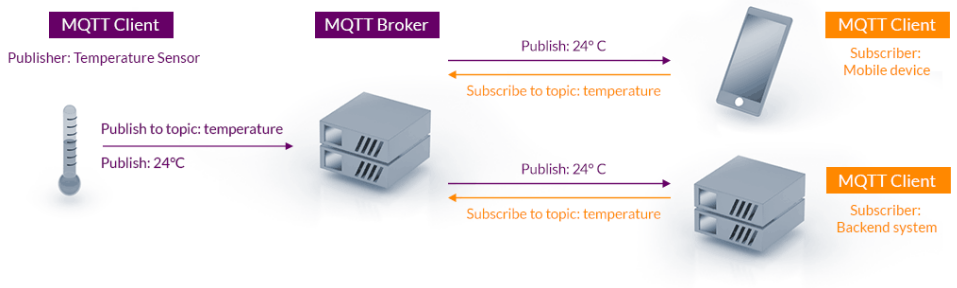
Parametr	**MQTT**	**HTTP**
Architektura	Publish/subscribe model	Request/response model
Komplexita	Méně komplexní	Více komplexní
Běží na	TCP	UDP
Protocol Design	Data centric	Document centric
Velikost zprávy	Méně než pro binární formát.	Více než využívá ASCII formát.
Velikost hlavičky	2 bajty	8 bajtů
Číslo portu	1883	80/8080
Zabezpečení dat	SSL/TLS	Neposkytuje. Výjimkou je HTTPS.

Tabulka 2.1: Porovnání parametrů protokolů HTTP a MQTT [16]

2.8.1 Princip protokolu a přenos zpráv

V komunikaci vystupují tři klíčové subjekty – publisher, subscriber a broker. Publisher je koncové zařízení, které odesílá naměřená data (senzor na měření vlhkosti). Subscriber je odběratel dat ze senzoru (webová aplikace, která zobrazuje naměřená data). Komunikaci mezi těmito participanty řídí broker, který v procesu vystupuje jako server.

Protokol MQTT funguje na principu publish/subscribe, neboli publikace/odběru. Narozdíl od HTTP, kde musí koncové zařízení vyslat žádost o data, vysílá publisher v pravidelných intervalech svá získaná data (například z měření vlhkosti půdy) na adresu brokeru, který následně data odesílá všem odběratelům, kteří se u něj přihlásili k odběru dat. Toto řešení nepředstavuje tak velkou zátěž sítě jako princip request/response (požadavek/odpověď).



Obrázek 2.11: Způsob komunikace s využitím modelu publish/subscribe [8]

Zprávy, které broker odešle danému odběrateli, se odvíjí od tzv. topics,

neboli témat. Témata jsou znaky s kódováním UTF-8, která jsou hierarchické struktury. Topic pro odebírání dat ze senzoru na měření vlhkosti by mohl vypadat například takto:

`bedroom/humidity/sensor1`. .

Odběratel, který se přihlásí k tomuto tématu, bude získávat od brokeru data o naměřené vlhkosti v ložnici ze senzoru č. 1.

V adresách je možné využívat zástupné znaky, které z témat činí regulární výrazy. Jedná se o znaky „#“, „+“ a „$“.

Znak # je ekvivalencí k *, tj. nahrazuje více úrovní. Z adresy `bedroom/#` lze odebírat všechny zprávy ze senzorů, které měří data v ložnici.

Znak + nahrazuje pouze jednu celou úroveň. Adresa `+/humidity` určuje, že odebíraná data budou ze všech místností, kde se měří vlhkost vzduchu.

Posledním znakem je **$**. Pokud jméno tématu začíná tímto znakem, jedná se o speciální téma. Taková témata se používají zejména pro zprávy, které posílá broker.

Kapitola 3

Klientská aplikace

Zadání bakalářské práce spočívalo v implementaci progresivní webové aplikace pro správu a monitorování rostlin. Implementaci aplikace předcházel návrh prototypu, jeho otestování s participanty, následná evaluace všech nálezů a jejich vyhodnocení pro finální verzi prototypu, která byla následně vzorem pro implementační část.

V rámci analýzy a návrhu aplikace byly sestaveny funkční požadavky týkající se správy rostlin, předpovědi o počasí, instalovatelnosti aplikace či správy a zobrazení upomínek.

Aplikace byla z hlediska informační architektury webu rozdělena na obrazovky týkající se rostlin, připomínek, profilu uživatele, předpovědi počasí a informací o aplikaci. Níže na obrázku 3.1 je náhled hlavní stránky.

Veškeré další detaily týkající se funkčních požadavků a výstupu z uživatelského testování jsou uvedeny v bakalářské práci [20].

3.1 Současný stav progresivních webových aplikací

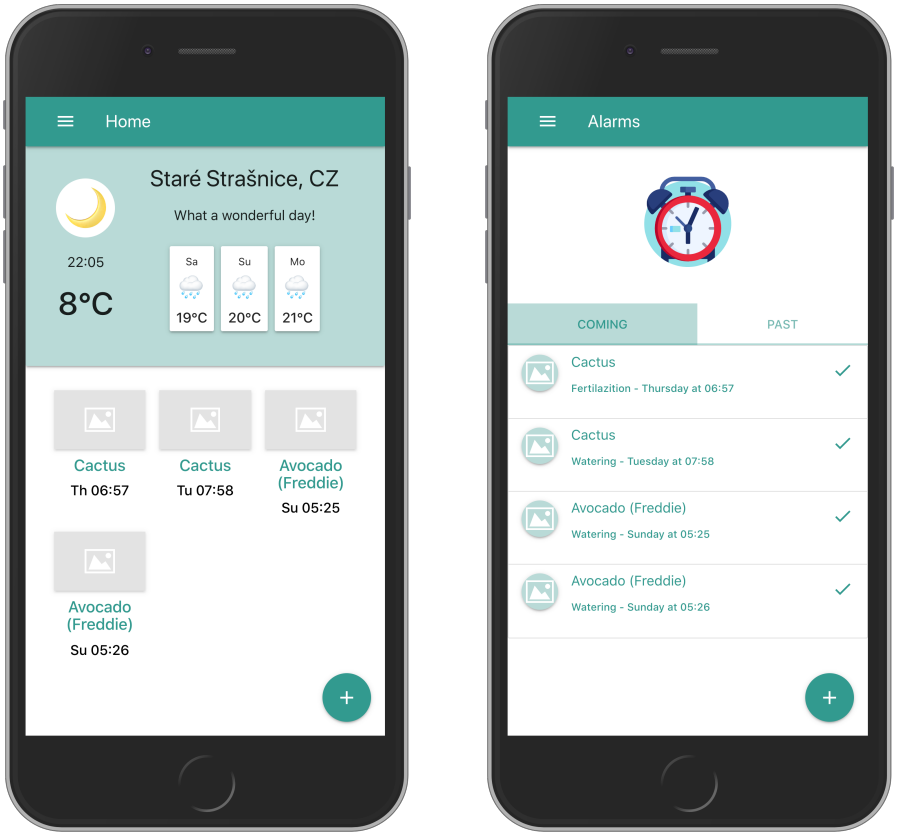
V roce 2019 představoval termín PWA poměrně nový koncept v oblasti vývoje aplikací. Od doby jeho vzniku uplynulo šest let a s tím se i částečně změnilo jeho vnímání mezi vývojáři a uživateli včetně podpory napříč platformami.

Myšlenka tohoto konceptu zůstává stejná, ale její motivace je více orientovaná směrem k uživatelům. S pomocí moderních funkcí dnešních prohlížečů docílit velmi podobného chování jako mají nativní aplikace. Včetně dodržování principu progressive enhancement[1] má být cílem PWA *poskytovat kromě moderních funkcí webových prohlížečů, zejména spolehlivé a instalovatelné aplikace dostupné pro kohokoliv, odkudkoliv a z jakéhokoliv zařízení prostřednictvím jednoho kódu*[2].

V původních dokumentech se vývojáři odkazovali na deset vlastností, které by každá PWA měla mít. V současnou chvíli existují checklisty [14], které mají vývojářům pomoci docílit co nejlepšího možného výsledku. Checklist

[1] Způsob návrhu softwaru založený na myšlence, že základní funkcionalita je dostupná všem uživatelům, ale rozšířená funkcionalita pouze těm, jejichž zařízení ji podporuje.

[2] Originální text: Progressive Web Apps (PWA) are built and enhanced with modern APIs to deliver enhanced capabilities, reliability, and installability while reaching anyone, anywhere, on any device with a single codebase. [14]



Obrázek 3.1: Ukázka původních obrazovek. Vlevo hlavní stránka, vpravo výpis upozornění.

má svou základní a rozšířenou verzi. Součástí základní verze jsou následující vlastnosti:

- rychlost webu a její optimalizace na základě výkonnostních metrik orientovaného na uživatele,
- nezávislost na prohlížeči,
- responzivita,
- uživatelská offline stránka,
- instalovatelnost aplikace.

Pro dosažení ještě lepšího výsledku, které posouvají celkový prožitek z používání aplikace na vyšší úroveň díky používání funkcí moderních prohlížečů, existuje rozšířená (či optimální) verze checklistu. Kritéria jsou následující:

- nezávislost na internetovém připojení,
- přístupnost podle standardu WCAG 2.0,
- dohledatelnost aplikace,

- nezávislost na vstupním zařízení,
- poskytnout uživateli kontext při získávání oprávnění,
- udržování *zdravého kódu*.

3.2 Nový návrh a rozšíření aplikace

Významný dopad na vývojářský prožitek má kvalita kódu jako taková. K definici kvalitního kódu mimo jiné i spadá robustnost či snadná rozšiřitelnost a udržitelnost. Aplikace, která byla výstupem bakalářské práce, tyto předpoklady zcela nesplňovala. Byla implementována jako Single Page Application (dále SPA) bez použití javascriptových knihoven, které by usnadňovaly udržování logiky a stavů aplikace. Z toho důvodu byl žádoucí přepis aplikace s využitím existujících javascriptových frameworků.

Další nevýhodou byla nepřenositelnost aplikace. Implementovaným řešením byla aplikace, která veškerou logiku i ukládání dat realizovala na klientské části webu, tj. v prohlížeči. z toho důvodu nebylo možné přistupovat k datům z různých zařízeních. Přepsání aplikace do klient-server architektury umožňuje lepší oddělení zodpovědností mezi uživatelským rozhraním a serverem, centrální správu dat a snazší rozšiřitelnost aplikace – kterou je například přijímání push notifikací. Naopak značnou nevýhodou jsou zvýšené bezpečnostní nároky, rozsáhlé znalosti vývojáře nejenom z oblasti frontendu, ale i práci s databází, konfiguraci serveru nebo nastavení produkčního prostředí a s tím spjaté Continious Integration & Continious Deployment (CI & CD).

3.2.1 Nové funkční požadavky

Tato kapitola popisuje návrh funkčních požadavků s ohledem na předchozí požadavky definové v bakalářské práci a zadání diplomové práce. Nově přidané požadavky jsou označené *.

FR1 – Aplikace bude zobrazovat notifikace s upomínkou na údržbu rostliny
Uživatel obdrží notifikaci s připomínkou nadcházející údržby rostliny.

FR2 – Aplikace umožní spravovat rostliny
Uživatel bude moct do aplikace ukládat a upravovat profily svých rostlin. Jmenovitě se bude jednat o následující položky:

1. název (případně přezdívku či latinský název),
2. lokaci, kde se rostliny nachází,
3. fotografii,
4. upomínky,
5. senzor.

FR3 – Aplikace umožní spravovat upomínky
Uživatel bude moct přiřazovat konkrétním rostlinám upomínky na jejich spravování.

FR4 – Aplikace bude zobrazovat seznam všech rostlin
Uživateli bude dostupný seznam všech rostlin, které přidal do aplikace.

FR5 – Aplikace bude zobrazovat seznam všech upomínek
Uživateli bude dostupný seznam všech upomínek, které přidal do aplikace.

FR6 – Aplikace bude zobrazovat předpověď počasí
Předpokládá se, že aplikace zobrazí uživateli počasí na následující tři dny podle zvolené lokace.

FR7 – Aplikace bude instalovatelná
Uživatel si bude moct webovou aplikaci nainstalovat na mobilní zařízení.

FR8 – Aplikace bude responzivní *
Aplikace se zcela přizpůsobí velikosti obrazovky daného zařízení.

FR9 – Aplikace bude přenositelná napříč zařízeními *
Uživatel bude moct přistupovat ke svým datům z různých zařízeních i platforem.

FR10 – Aplikace bude zobrazovat aktuálně naměřená data ze senzoru *
Vhodným způsobem budou vizualizována naměřená data.

FR11 – Uživatel bude moct sdílet upomínku na údržbu rostliny *
Uživatel bude moct přes sociální sítě či jiné platformy sdílet upomínku na údržbu rostliny.

3.2.2 Použité technologie

Webová aplikace se nově sestává z klientské a serverové části. Pro sestavení klientské části a snadné udržování stavu aplikace byla zvolena javascriptová knihovna ReactJS. Oproti předchozí implementaci je uživatelské rozhraní postaveno s knihovnou React UI. Serverová část je postavena pomocí jazyka Node.js, aplikačního frameworku Express.js a databázového systému MySQL.

V rámci rozšíření aplikace je nově používání Web Share API a Push Notifications API, které jsou rozepsané v podkapitolách 3.2.3 a 3.2.4.

Klientská část

ReactJS. ReactJS je javascriptová knihovna pro tvorbu uživatelského rozhraní. Vývojářům umožňuje udržovat logiku aplikace a předávat stavy mezi

komponentami, což je následně reflektováno v dynamicky se měnícím DOMu. Z toho důvodů je nutné jej kombinovat s knihovnami pro routování, správou formulářů či UI knihovnami.

React UI. React UI je javascriptová UI knihovna. Narozdíl od Material UI a i ostatních knihoven umožňuje přizpůsobení dílčích komponent prostřednictvím CSS Variables. Vývojářům tak nabízí konzistentní a udržitelnou cestu při úpravě vizuálu.

Serverová část

Node.js. Node.js slouží pro vytváření serverového prostředí webových aplikací. Narozdíl od PHP lze využívat JavaScript nejenom v klientské, ale i serverové části aplikace a vytvářet tak dobře škálovatelné a robustní prostředí.

Express.js. Express.js je framework, který ulehčuje tvorbu webových aplikací a práci s API. V současné době je nedílnou součástí při tvorbě HTTP serverů právě díky jeho jednoduchosti, robustnosti a velmi dobré výkonnosti. Vývojář je odstíněn od rozsáhlých konfigurací a psaní zdlouhavého kódu, které by v případě používání čistého Node.js musel řešit.

MySQL. MySQL umožňuje správu relačních databází. S použitím jazyka SQL umožňuje práci s daty, řízení transakcí a vytváření komplexních databázových struktur.

3.2.3 Web Share API

Web Share API umožňuje sdílení obsahu prostřednictvím nativních aplikací přímo z webové aplikace. Díky tomuto API mohou uživatelé sdílet textové zprávy i soubory s téměř kteroukoliv aplikací, která je k tomu určena. Pro vývojáře se významně sníží režie spjaté s vytvářením prostředí pro sdílení obsahu přes jednotlivé sociální platformy.

Jak je vidět v tabulce 3.1, zejména na desktopových webových prohlížečích je podpora pouze částečná, nicméně pro valnou většinu mobilních platforem je Web Share API podporované.

Desktopové prohlížeče					Mobilní prohlížeče			
Edge	Firefox	Chrome	Safari	Opera	Firefox	Chrome (Android)	Safari	Opera Mini
Částečně	Ne	Částečně	Ano	Ne	Ano	Ano	Ano	Ne

Tabulka 3.1: Současná podpora v prohlížečích Web Share API ze dne 27. 4. 2021 [3]

Pro zprovoznění API existují dva požadavky, kterými jsou:

- provozování webu přes HTTPS (localhost má udělenou výjimku),

- zobrazení dialogu po uživatelské akci (kliknutí, přejetí myší přes element, aj.).

Zavoláním metody `navigator.share(data)` se spustí akce pro sdílení obsahu, jejíž návratovou hodnotou je objekt typu `Promise`. Předávaným parametrem je objekt `data`, který má následující strukturu:

- `url` – odkaz,
- `text` – text,
- `title` – název,
- `files` – pole souborů.

Implementační ukázka s využitím Web Share API je součástí kapitoly 5.

3.2.4 Web Push Notifications

Web Push Notifications je označení pro webové push notifikace. Jedná se o kombinaci tří technologií a to Notifications API, Push API a Service Worker API. Za push notifikaci je označována zpráva, která je odesílána ze serveru do aplikace, kde je nejprve zpracována v service workeru a následně prostřednictvím Web Notifications API zobrazena v tzv. status baru či prohlížeči.

V širším slova smyslu jej lze chápat za komunikační protokol, ve kterém figurují 4 subjekty: uživatel, webová aplikace, service worker a push server. Komunikace je detailně popsána na diagramu 3.2 a probíhá zhruba následovně:

1. Aplikace požádá uživatele o povolení k zobrazování notifikací.
2. Při získání souhlasu zaregistruje aplikace service worker a následně jej použije k vytvoření push notification subscription. Subscription se ukládá jak do aplikace, tak na push server.
3. Pokud proběhla registrace úspěšně, push server nyní může odesílat zprávy směrem k service workeru, který jej následně zobrazí klientovi.

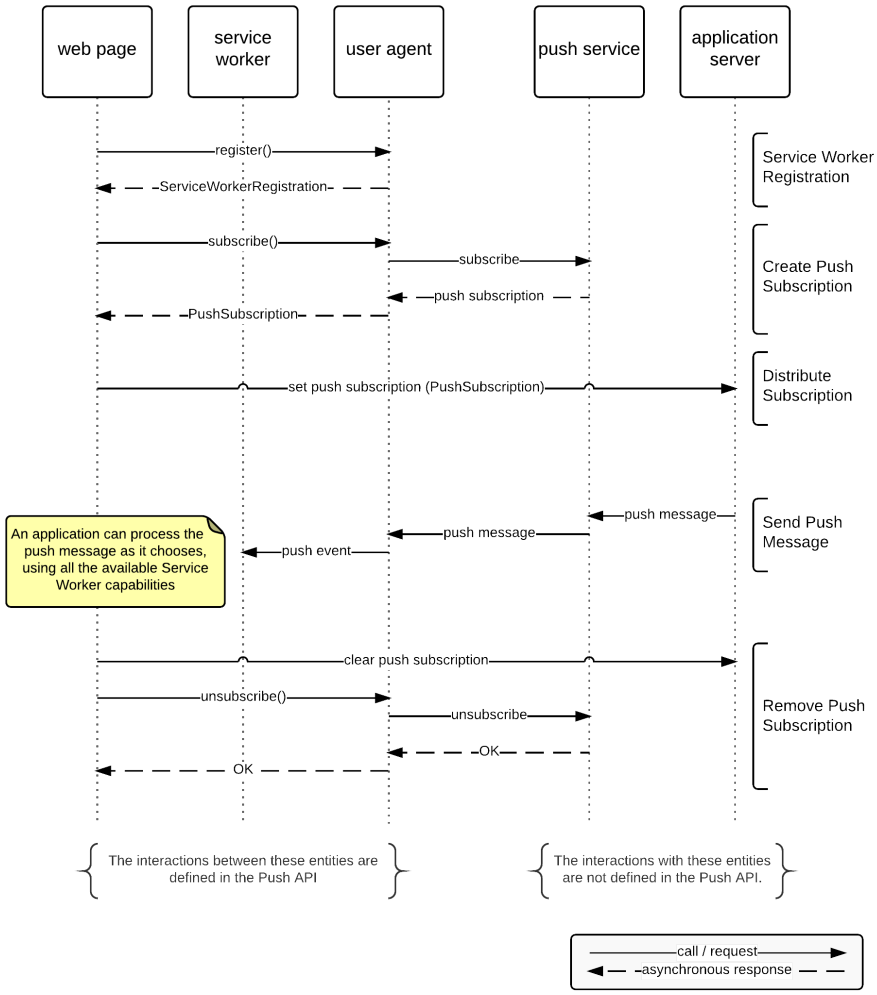
O Notifications API a fungování service workeru je zmínka v bakalářské práci na stránkách 6–10 [14].

Push API

Push API umožňuje odesílat zprávy ze serveru do prohlížeče, i když není webová stránka v aktivním tabu nebo dokonce otevřena v prohlížeči. Oproti Notifications API vyžaduje pro přijímání zpráv registraci service workeru, a tím pádem umožňuje přijímání zpráv ze serveru.

Níže v tabulce 3.2 je uvedena současná podpora Push API napříč prohlížeči.

Implementační ukázka s využitím Web Push Notifications je součástí kapitoly 5.



Obrázek 3.2: Sekvenční diagram Web Push notifikací [44]

Vizualizace dat ze senzoru

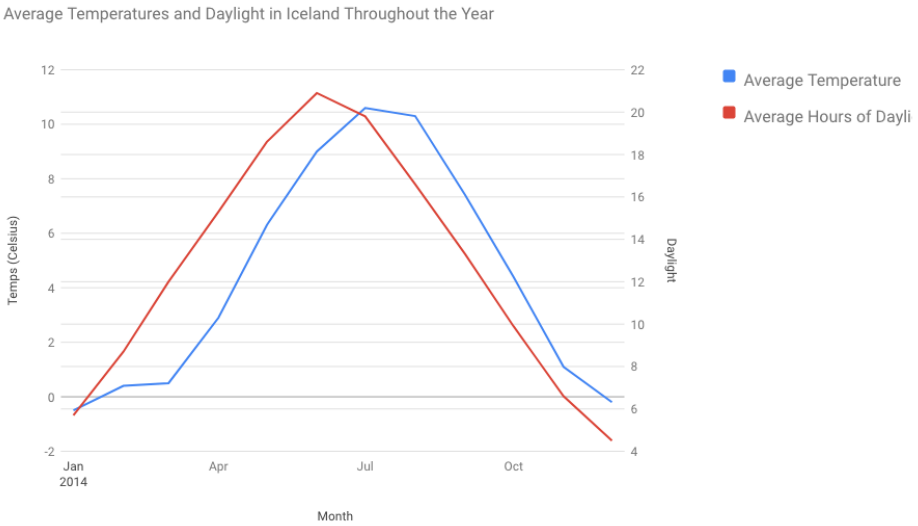
Hlavním přínosem integrace senzoru na měření vlhkosti půdy je možnost sledování aktuálního stavu vláhy rostliny. Data, která budou získávaná ze senzoru, se budou skládat z hodnoty a data, kdy byla hodnota naměřena. Z toho důvodu je nejvhodnějším způsobem vizualizace použití liniového, resp. spojnicového, grafu, ze kterého bude uživatel schopen vyčíst aktuální hodnotu a sledovat i trend přímky.

V případě rozšíření aplikace o integraci dalších senzorů, například na měření teploty, vlhkosti vzduchu či světla, by bylo možné graf jednoduše rozšířit o další přímky, které by byly barevně odlišeny podle typu naměřených dat, viz obrázek 3.3.

Další z variant je zobrazení pouze poslední naměřené hodnoty včetně data naměření, případně použití měřidla, viz obrázek č. 3.4. Ani jedna z těchto variant by nemusela být ideální v případě, že dojde k nečekanému výkyvu

Desktopové prohlížeče					Mobilní prohlížeče			
Edge	Firefox	Chrome	Safari	Opera	Firefox	Chrome (Android)	Safari	Opera
Ano	Ano	Ano	Ne	Ano	Ano	Ano	Ne	Ano

Tabulka 3.2: Současná podpora v prohlížečích Push API ze dne 27. 4. 2021 [4]



Obrázek 3.3: Ukázka liniového grafu s dvěma nezávislýma osama y [7]

u naměřených dat, nebo rozšíření o další senzory.

Obrázek 3.4: Ukázka grafu typy měřidlo [12]

Kapitola 4

Sestavení senzoru na měření vlhkosti půdy

Pro integraci naměřených dat o vlhkosti půdy bylo klíčové sestavení jednoduchého prototypu, který bude schopen měřit a odesílat data.

Představí se jeden z možných způsobů jak data shromažďovat a distribuovat do webové aplikace. V závěru kapitoly bude popsáno implementované řešení z hlediska výhod, nevýhod a energetických požadavků.

4.1 Komponenty prototypu

Pro zajištění funkčního systému byly zapotřebí 3 komponenty:

- **kapacitní senzor** – pro měření vlhkosti půdy,
- **mikrokontroler** – pro posílání dat ze senzoru na server,
- **nepájivé konktaktní pole včetně propojek** – pro spojení mikrokontroleru a kapacitního senzoru.

Obrázek 4.1: WeMos D1 Mini ESP8266 WiFi modul [28]

Kapacitní senzor je možné obstarat na českém i zahraničním trhu. Na českém trhu je dostupný například na e-shopu SOLARSHOP, u zahraničních

Obrázek 4.2: Modul se senzorem na měření vlhkosti půdy [11]

dodavatelů například na AliExpressu. Obdobně tomu je i v případě mikrokontroleru, který je dostupný buď na e-shopu laskarduino.cz, případně opět na AliExpressu. Nepájivé kontaktní pole je dostupné téměř v každých elektrotechnických potřebách včetně propojek. Zcela dostačující je s i s nízkými desítkami pinů.

Pro hrubý odhad celkových nákladů je zde přiložena tabulka 4.1. V celkovém součtu nebyly započítány náklady na dopravu.

Název produktu	**Cena na českém trhu [Kč]**	**Cena na zahraničním trhu [Kč]**
kapacitní senzor	65	10
mikroprocesor ESP8266	128	45
nepájivé pole včetně propojek	120	60
Součet	**313**	**115**

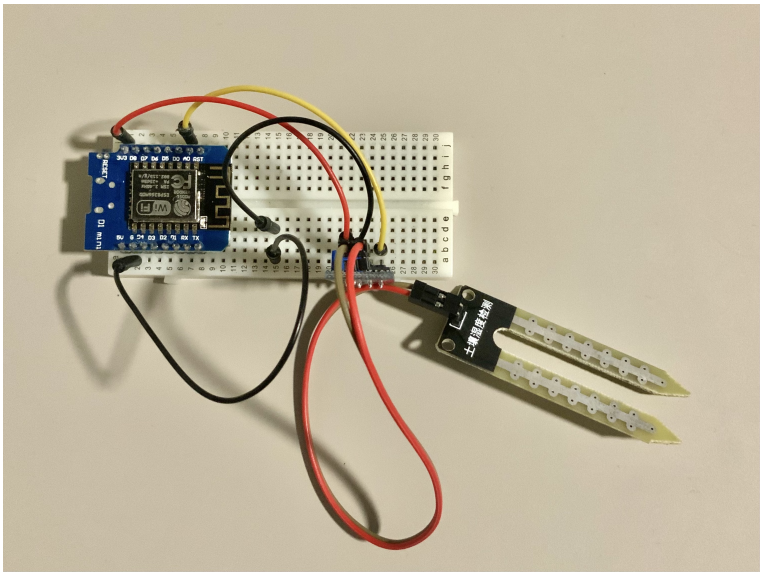
Tabulka 4.1: Přehled finančních nákladů za jednotlivé komponenty

4.2 Implementace jednotlivých kroků

Sestavení funkčního prototypu se sestávalo ze tří kroků:

- zapojení obvodu,
- zprovoznění odesílání dat z mikrokontroleru.
- napojení na Adafruit.

Nejprve byly komponenty propojeny na nepájivém poli. Senzor byl připojen k mikroprocesoru prostřednictvím nepájivého pole pod napětím 3.3 V s analogovým výstupem, viz obrázek 4.3.



Obrázek 4.3: Sestavení zapojení mikrokontroleru s kapacitorem

Po zapojení modulu bylo zapotřebí nakonfigurovat mikroprocesor tak, aby byl schopen se připojit k lokální síti a odesílat data ze senzoru na server Adafruit. K tomu bylo možné použít Arduino IDE, které umožňuje konfigurovat desky kompatibilní s Arduino. Níže je ukázka zdrojového kódu, která obsahuje konfigurační soubor pro připojení k Wi-Fi a účtu na Adafruit IO. Zdrojový kód byl inspirován ze zdroje [46] se souhlasem autora.

```
#define WLAN_SSID       "$WIFI_NAME"
#define WLAN_PASS       "$WIFI_PASSWORD"
#define DELAY 3000

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883

#define AIO_USERNAME    "$USERNAME"
#define AIO_KEY         "$KEY"

#define AIO_FEED_WATER AIO_USERNAME "/feeds/humidity"
```

Fragment kódu 4.1: Hlavičkový soubor pro konfiguraci mikrokontroleru a komunikace s Adafruit IO API

Po předání všech důležitých parametrů do konstruktoru představujícího připojení k Adafruit serveru, bylo zapotřebí zajistit opětovné připojování k lokální síti, která je uvedena v konfiguračním souboru.

```
WiFiClient client;

Adafruit_MQTT_Client mqtt(
    &client,
    AIO_SERVER,
    AIO_SERVERPORT,
    AIO_USERNAME,
    AIO_KEY
);
Adafruit_MQTT_Publish soilMoisture =
    Adafruit_MQTT_Publish(&mqtt, AIO_FEED_WATER);

void connect() {
  Serial.print(F("WiFi: Connecting to "));
  Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }

  Serial.println();
  Serial.println(F("WiFi: connected"));
  Serial.println(F("WiFi IP address: "));
  Serial.println(WiFi.localIP());

  Serial.print(F("MQTT: Connecting to broker... "));
  int8_t ret;

  while ((ret = mqtt.connect()) != 0) {
    Serial.println(mqtt.connectErrorString(ret));
    if (ret >= 0) mqtt.disconnect();
    Serial.println();
    Serial.println(F("MQTT: Retrying connection..."));
    delay(5000);
  }

  Serial.println();
  Serial.println(F("MQTT: connected!"));
}

void setup() {
  Serial.begin(9600);
  Serial.println(F("Initializing"));


  delay(100);
  connect();
}
```

Fragment kódu 4.2: Připojení k lokální síti a implementace komunikace s Adafruit serverem

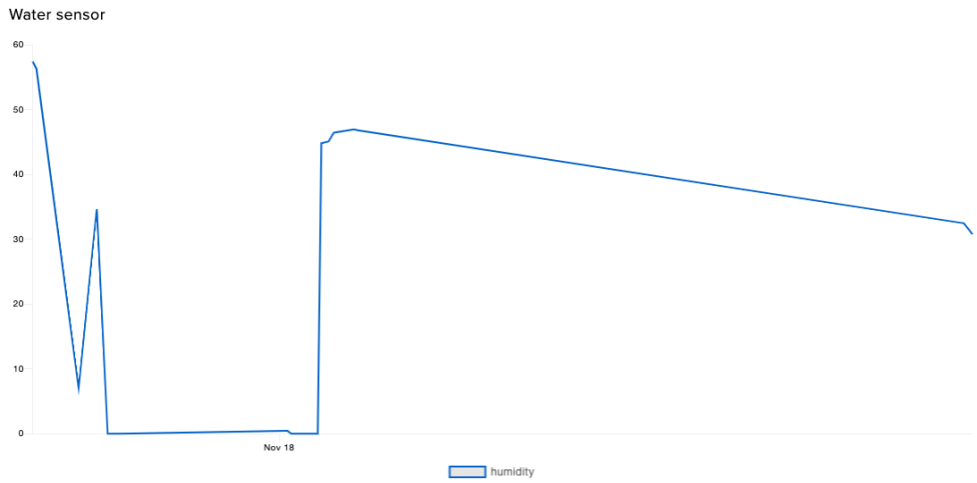
Pokud je spojení k lokální bezdrátové síti úspěšné, lze navázat spojení s MQTT brokerem, který umožní předávání dat. Posléze je každé tři sekundy nad deseti vzorky dat ze senzoru zprůměrována hodnota vlhkosti, která je

převedena na procenta a odeslána na klientský server.

```
void loop() {
  if (!mqtt.ping(3)) {
    if (!mqtt.connected()) { connect(); }
  }


  float averageHumidity;

  for(int i = 0; i < AVERAGE_N; i++){
    int value = analogRead(A0);
    averageHumidity += value;

    delay(10);
  }

  float range = 1024 / 10;
  float humidity = 100 - averageHumidity / range;

  Serial.println("Humidity: ");
  Serial.println(humidity);
  Serial.println(" ");

  soilMoisture.publish(humidity);

  delay(DELAY);
}
```

Fragment kódu 4.3: Ukázka ze zdrojového kódu

Pokud je realizace všech výše uvedených kroků úspěšná, lze sledovat naměřené hodnoty v reálném čase na účtu Adafruit IO, viz snímek 4.4.



Obrázek 4.4: Ukázka vizualizovaných dat ze senzoru

4.3 Zpracování získaných dat ze senzoru

V rámci zvolené architektury řešení je možné naměřená data odesílat prostřednictvím MQTT protokolu IoT brokeru, který bude data ukládat a poskytovat subscriberům.

Serverů pro publikaci IoT dat existuje celá řada. Je možné si implementovat vlastní řešení, případně využít služeb třetích stran. Pokud je žádoucí si implementovat vlastní řešení, je možné zprovoznit vlastní MQTT server například s využitím Raspberry PI a Eclipse Mosquitto. V rámci služeb třetích stran existují platformy jako IFTTT, openHAB nebo Adafruit IO. Pro účely práce bylo zvoleno Adafruit IO.

Adafruit IO je systém pro ukládání dat. Jeho vývojáři si zakládají na tom, aby jej bylo možné použít co možná nejsnadněji s minimem programování a s minimálními datovou spotřebou. IO umožňuje pracovat s klientskými knihovnami jako jsou REST nebo MQTT API.

Naměřená data je možné získat dvěma způsoby:

- stažením ve formátu JSON nebo CSV,
- přístupem k veřejnému API.

Efektivním řešením je využití posledního přístupu, pomocí něhož bude uživatel vždy pracovat s aktuálními daty.

Adafruit IO svá data vystavuje prostřednictvím Adafruit IO HTTP API. Jednou z možností, jak získat data v klientské aplikaci, je využití Fetch API. Po zaslání HTTP requestu na adresu:

`/api/v2/{username}/feeds/{feed_key}/data/chart`,

jehož ukázka je v kódu 4.4, vrátí Adafruit server v případě úspěšného dotazu data ve formátu 4.5.

```
const url = "https://io.adafruit.com/api/v2/dlouhka1/feeds/
    humidity/data/chart";

fetch(url)
  .then(resp => resp.json())
  .then((data) =>
    pre.textContent = JSON.stringify(data, undefined, 2);
  )
  .catch((error) =>
    console.error(error);
  );
```

Fragment kódu 4.4: Získání dat z Adafruit IO pomocí Fetch API [19]

```
{
  "feed": {
    "id": 0,
    "key": "string",
    "name": "string"
  },
  "parameters": {
    "start_time": "2019-02-28T16:17:09Z",
    "end_time": "2019-04-29T16:17:09Z",
    "resolution": 120,
    "hours": 1440,
    "field": "avg"
  },
  "columns": ["date", "avg"],
  "data": [
    [ "2019-03-01T14:00:00Z", "62.579827586206896" ],
    [ "2019-03-02T18:00:00Z", "64.94642857142857" ],
    [ "...", "..."]
  ]
}
```

Fragment kódu 4.5: Struktura dat, které vrací Adafruit IO

HTTP request je možné navíc specifikovat prostřednictvím query parametrů pro případnou filtraci či agregaci dat.

Parametr	Typ	Povinný parametr	Popis
start_time	string	ne	Počáteční čas.
end_time	string	ne	Konečný čas.
resolution	int	ne	Časový rozestup dat.
hours	int	ne	Počet hodin v grafu.
field	string	ne	Aritmetická operace nad daty.
raw	boolean	ne	Navrácení nezpracovaných dat.

Tabulka 4.2: Specifikace parametrů pro HTTP requesty na Adafruit IO [1]

4.4 Vyhodnocení prototypu a návrhy na zlepšení

Navržený prototyp lze považovat za úspěšný proof of concept. Zprovoznění modulu nevyžadovalo velkou časovou investici a bylo tak možné v poměrně rychlém čase ověřit funkčnost řešení. Mezi bezesporné výhody patří i pořizovací cena, která v porovnání s již existujícími řešeními byla několikanásobně nižší.

V případě senzorů na měření vlhkosti půdy v interiérech nejsou na řešení kladena tak velká omezení, jako u senzorů venkovních. Pokud však dochází k zavlažování půdy, je klíčové, aby zanořená část senzoru v půdě byla co nejvíce odolná vůči korozi. Z těchto důvodů jsou vhodnější TDR senzory s vidlicemi, které jsou odolnější vůči této reakci.

Velikost senzoru ovlivňuje přesnost měření. V případě velké rostliny může senzor dosahovat do relativně nízké hloubky a detekovat tak sušší půdu i v případě, že na dně květináče je půda dostatečně vlhká.

Další z nevýhod je nutnost kalibrace pro jednotlivé druhy půdy. Testování senzoru bylo prováděno nad vzorkem převážně písčité půdy, která má jinou permitivitu prostředí než půda s vyšším podílem jílu. Hodnota vlhkosti velmi vysušené půdy se pohybovala kolem 12 %, přičemž po jejím zalití dosahovala 80 %.

4.4.1 Energetická náročnost

Značnou nevýhodou senzoru může být nutnost externího zdroje napájení, což by bylo možné řešit pomocí baterie, která by byla součástí mikrokontroleru. Následující myšlenku je možné si ověřit.

Spotřeba mikrokontroleru se pohybuje mezi 15 μA až 400 mA – záleží na způsobech užití. V nečinném stavu se spotřeba pohybuje kolem 70 mA, což znamená, že při napětí 3.3 V je potřeba příkon okolo 230 mW.

$$W = U \cdot I = 3.3 \cdot 70 = 231\ mW, \tag{4.1}$$

Pokud by byla klíčová otázka spotřeby elektřiny, roční spotřeba by vyšla na 8 Kč, při ceně 4 Kč za 1 kWh.

$$E = W \cdot 365 \cdot 24 = 2.024\ kWh, \tag{4.2}$$

V případě baterie je ale spotřeba omezená její kapacitou. Pokud by byla k dispozici baterie s kapacitou 1 000 mAh, pohybovala by se výdrž baterie okolo 14 hodin.

$$T = \frac{C}{I} = \frac{1000}{70} = 14.3\ h, \tag{4.3}$$

Kupovat nabíjecí baterii s vyšší kapacitu je jedna z možností, ale toto řešení by nemuselo být žádoucí, vzhledem k postupnému snižování kapacity baterie a velmi nízké výdrži. Pokud by se podařilo snížit spotřebu mikrokontroleru (například změnou spotřeby energie v tzv. sleep mode), bylo by možné snížit průměrnou spotřebu téměř o trojnásobek. Což znamená, že by místo původních 14 hodin byla baterie schopna nabíjet až 42 hodin.

4.4.2 Integrita dat

Nejkritičtější a nejvíce zranitelnou částí systému je samotné IoT. Veškerá data, které mezi sebou přenáší koncová zařízení v tomto ekosystému, jsou často sdílena prostřednictvím internetu, aby byla dostupná odkudkoliv namísto lokální sítě, ve které se senzory nachází. V současné chvíli neexistuje organizace či jiná autorita, jež by zastřešovala integritu dat ve světě IoT. Dochází pouze k regulaci a specifikaci požadavků na výkon, použitelnost, energetickou náročnost a jiné hardwarové požadavky.

Při konfiguraci vývojové desky je nutné zadat údaje k lokální síti, což může představovat mnohem větší bezpečnostní riziko v případě nezabezpečené sítě. Útočníci mohou získat přístup k datům, které se používají v rámci chytré domácnosti. Mohou posléze sledovat, která světla a zdali vůbec jsou v domě rozsvícena, získat přístup k osobním údajům či dokonce napadnout některá zařízení za účelem těžby kryptoměn, DDoS[1] útoku či jiných forem kyberútoku.

Pro zabezpečení IoT a jejich zařízení je vhodné používat oddělené sítě. Pro vlastní potřebu pracovat pouze s privátní a zabezpečenou síti a naopak pro IoT zařízení mít oddělenou síť pro hosty. Používání silných hesel či dvoufaktorového ověření v aplikacích, dostatečné zabezpečení routeru případně používání VPN pro sdílení dat je obecně vzato doporučování i komunitou OWASP, která se zabývá zejména bezpečnostní webových aplikací. [41]

[1] DDoS (Distributed Denial of Service) je typ útoku za účelem přetížení serverů, čímž dochází ke kompromitaci přístupu návštěvníků na web.

Kapitola 5

Rozšíření aplikace

Hlavním cílem diplomové práce byla integrace dat naměřených ze senzoru na měření vlhkosti půdy a rozšíření stávající funkcionality o Web Share API a Push notifikace. Následující kapitola by měla seznámit čtenáře s výsledkem implementační části práce a předložit signifikantní části kódu, které naplňují dílčí cíle zadání.

Zdrojový kód a ukázka průchodu aplikací je součástí přílohy závěrečné práce. Aplikace je hostována na platformách Heroku a Vercel na adrese: https://plant-maintenance-app.vercel.app/.

5.1 Ukládání dat

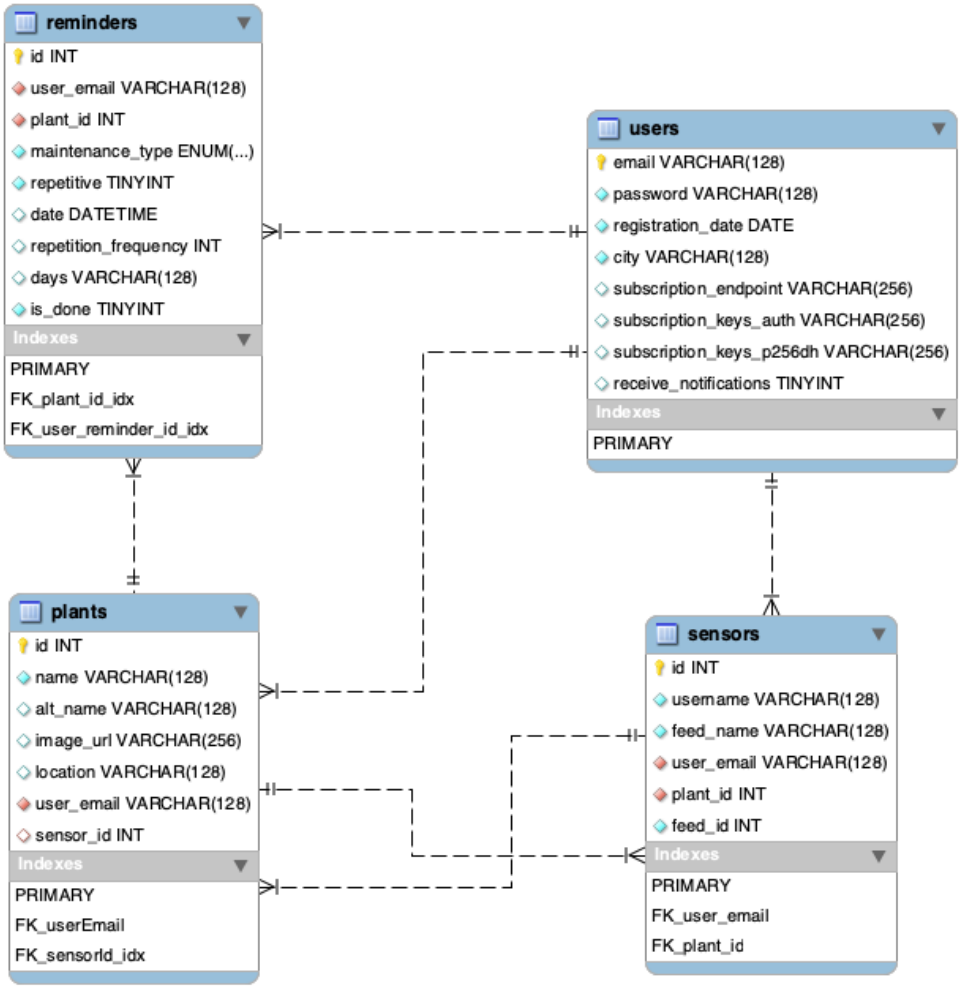
Z důvodů přepsání aplikace do architektury klient-server bylo nutné přesunout veškeré ukládání dat z Web Storage na přístupnější úložiště dat, kterým je například databáze MySQL. Díky tomu je možné ukládat globálně data uživatelů, kteří mohou s aplikací pracovat z více zařízeních, jelikož nejsou vázani na úložiště webového prohlížeče jako tomu bylo v případě bakalářské práce.

V databázovém schématu vystupují následující entity:

- uživatel,
- rostlina,
- upomínka,
- senzor.

Na obrázku 5.1 níže je schéma entitně-relačního modelu, které zachycuje jednotlivé vlastnosti a vazby mezi entitami.

Aplikace i nadále pracuje s Web Storage kvůli ukládání tokenů pro autentizaci a udržování stavu uživatelů. Vyjma klientského a serverového úložiště využívá i službu Cloudinary. Cloudinary je cloudová služba, která umožňuje správu a distribuci médií. Pomocí komplexního API lze jednoduše komprimovat a optimalizovat obrázky či videa, které služba zpřístupňuje z jejich CDN.

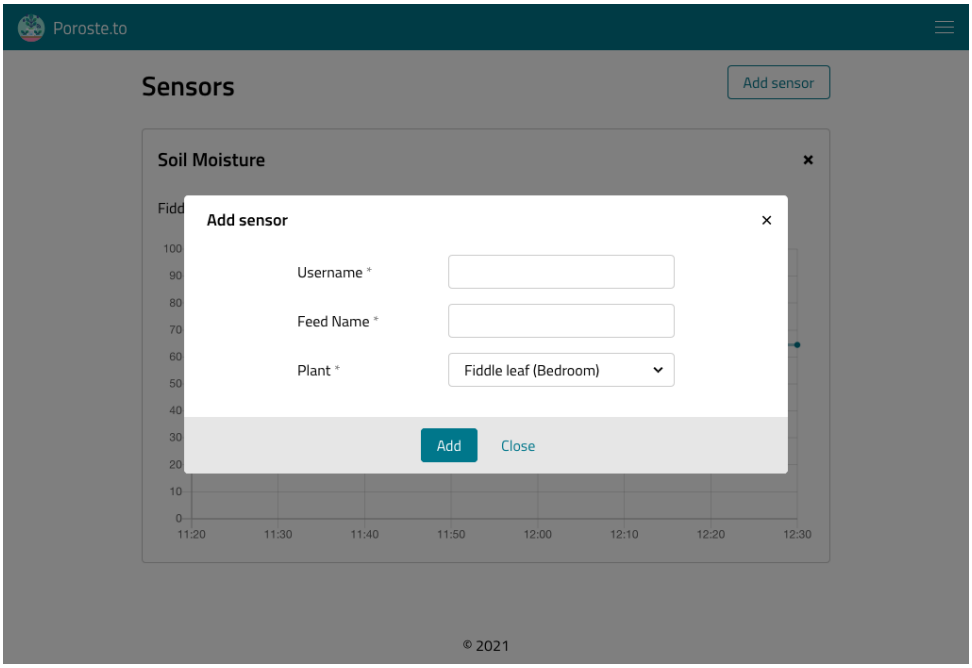


Obrázek 5.1: Databázové schéma

5.2 Integrace senzoru a vizualizace naměřených dat

Uživatel má možnost propojit s aplikací senzor, jehož data jsou nahrávána na serveru Adafruit IO. Pro získání dat je nutné zadat dva klíčové parametry – uživatelské jméno a název zdroje dat. Dále vybrat rostlinu, se kterou je senzor spjat, viz snímek 5.2. Po úspěšném přidání senzoru se uživateli zobrazí data naměřená za posledních pět hodin s časovým rozestupem deseti minut, respektive za poslední hodinu s rozestupem deseti minut na menších obrazovkách pro lepší čitelnost. Ukázka je na snímku č. 5.3.

Sledování stavu vlhkosti půdy je možné nejenom ze stránky s přehledem senzorů, ale i na hlavní stránce v sekci s nadcházejícími upomínkami, viz 5.4. U rostliny, která je propojená s integrovaným senzorem, je možné vidět hrubý odhad aktuálního stavu vlhkosti půdy v podobě statusu vláhy. Ikonky kapek symbolizují vlhkost půdy, jak je vidět na obrázku s legendou č. 5.5.



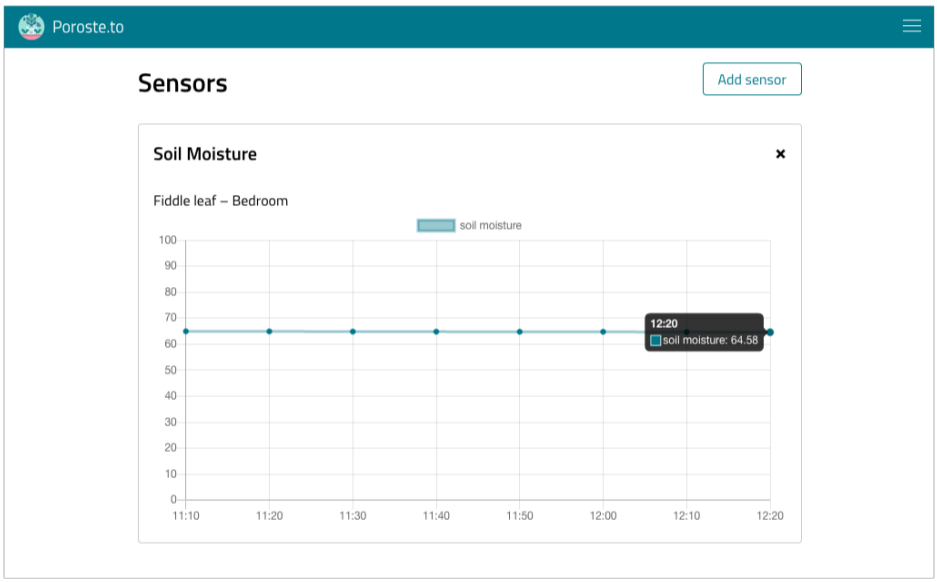
Obrázek 5.2: Obrazovka s přidáním nového senzoru

V případě, že upomínka nebyla dosud označena za splněnou, v levém rohu je číslicí vypsáno zpoždění ve dnech oproti plánovanému termínu.

Předpokladem je, že senzor vrací hodnoty mezi 0–100. Pro určení správné vlhkostní kategorie půdy je jedním z řešení správná kalibrace senzoru. V případě zcela vysušené půdy by měl senzor vracet hodnotu mírně nad 0 a analogicky pro velmi zavlaženou půdu hodnotu blížící se 100. Tuto kalibraci by bylo možné řešit v rámci integrace senzoru, kde by uživatel namapoval na daná rozmezí hodnot stav vlhkosti půdy. Například u některých kaktusovitých rostlin je vhodnější sušší půda, tím pádem by ideální stav vláhy byl u mnohem nižších hodnot než u žíznivějších rostlin.

Uživatel může sledovat stav vlhkosti půdy na stránce s výpisem senzorů, případně na hlavní stránce ve widgetu s nadcházejícími upomínkami. Z uživatelského pohledu by bylo vhodné zobrazovat naměřená data i na detailu rostlinu.

Současná integrace senzoru klade na uživatele jistá omezení. Uživatel může propojit s aplikací senzor, jehož data jsou nahrávána na platformu Adafruit IO. Vhodnějším způsobem by mohlo být poskytnutí vícero rozhraní, se kterými může aplikace komunikovat – thethings.io, Google Cloud IoT, Microsoft Azure IoT, Ubidots aj. Uživatelsky přívětivějším řešením by bylo minimalizovat interakci se senzorem a integrovat jej prostřednictvím QR kódu na něm umístěném. Pro přesnější data z měření by bylo vhodné uživateli zobrazit konfigurační obrazovku, kde by zadal typ půdy, případně další údaje pro kalibraci senzoru.



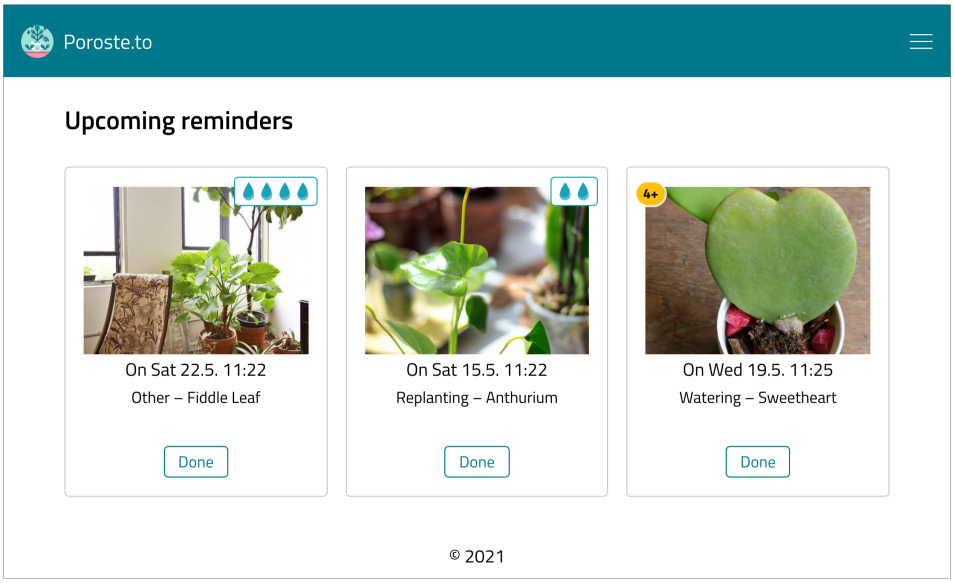
Obrázek 5.3: Obrazovka s vykreslenými daty ze senzoru

5.3 Sdílení upomínek na údržbu rostliny

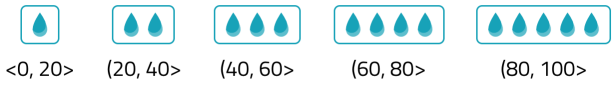
Uživatel aplikace může sdílet upomínku na údržbu rostliny. Sdílení je možné ze stránky s detailem rostliny a pouze u těch upomínek, které doposud nebyly splněny, viz snímek č. 5.6. Následně se uživateli nabídne podle jeho sytémové nabídky okno ke sdílení obsahu jako je na snímku č. 5.7 nebo 5.8.

Předmětem sdílení je zpráva s informacemi o údržby dané rostliny včetně odkazu, který může adresát využít v momentě, kdy má přihlašovací údaje do aplikace, za účelem potvrzení údržby nebo nahlédnutí fotografie. Ukázka zprávy je na snímku 5.10.

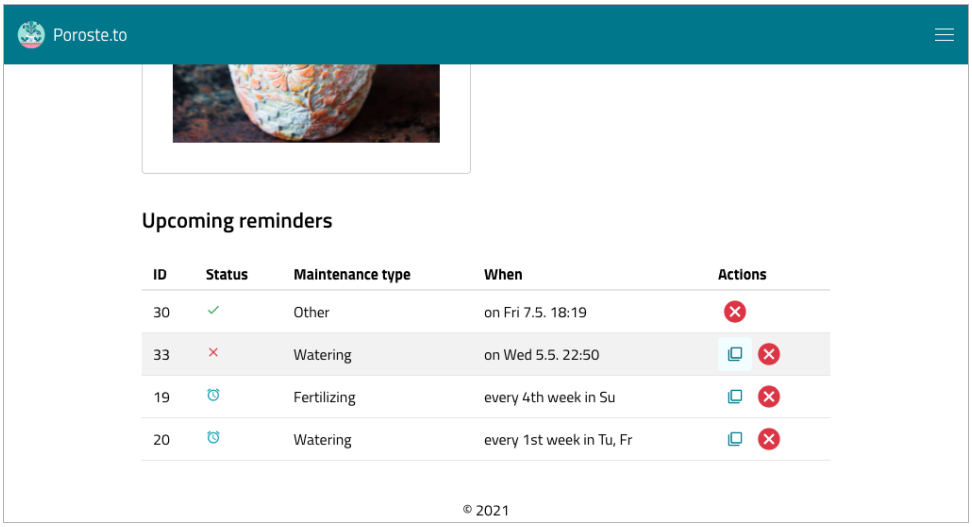
Web Share API není podporované na všech platformách a zařízeních. Z toho důvodu je v kódu přidán fallback, který využívá Clipboard API pro uložení zprávy do schránky. Níže na 5.1 je výňatek ze zdrojového kódu.



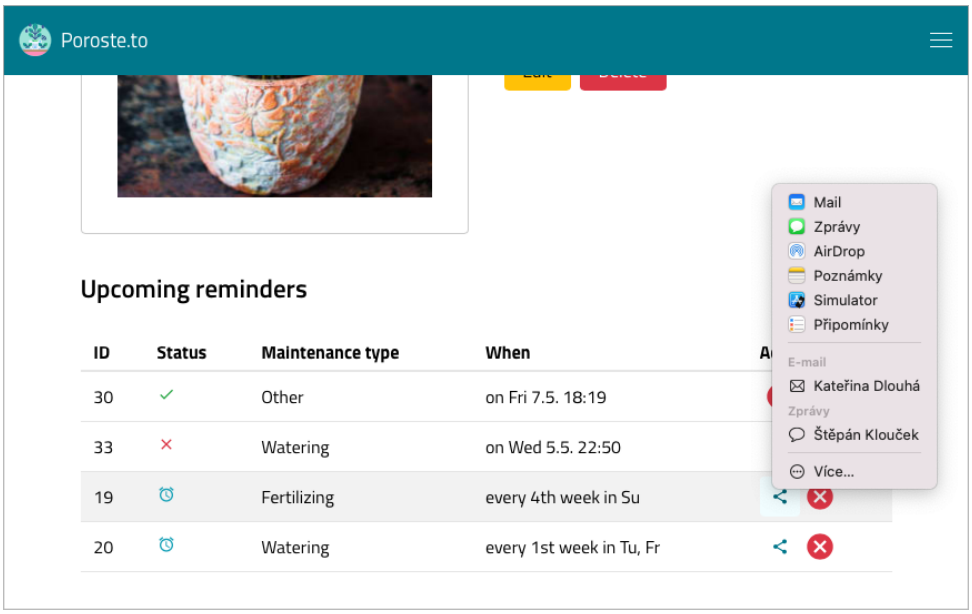
Obrázek 5.4: Obrazovka s nadcházejícími upomínkami včetně aktuálního stavu vlhkosti půdy dané rostliny



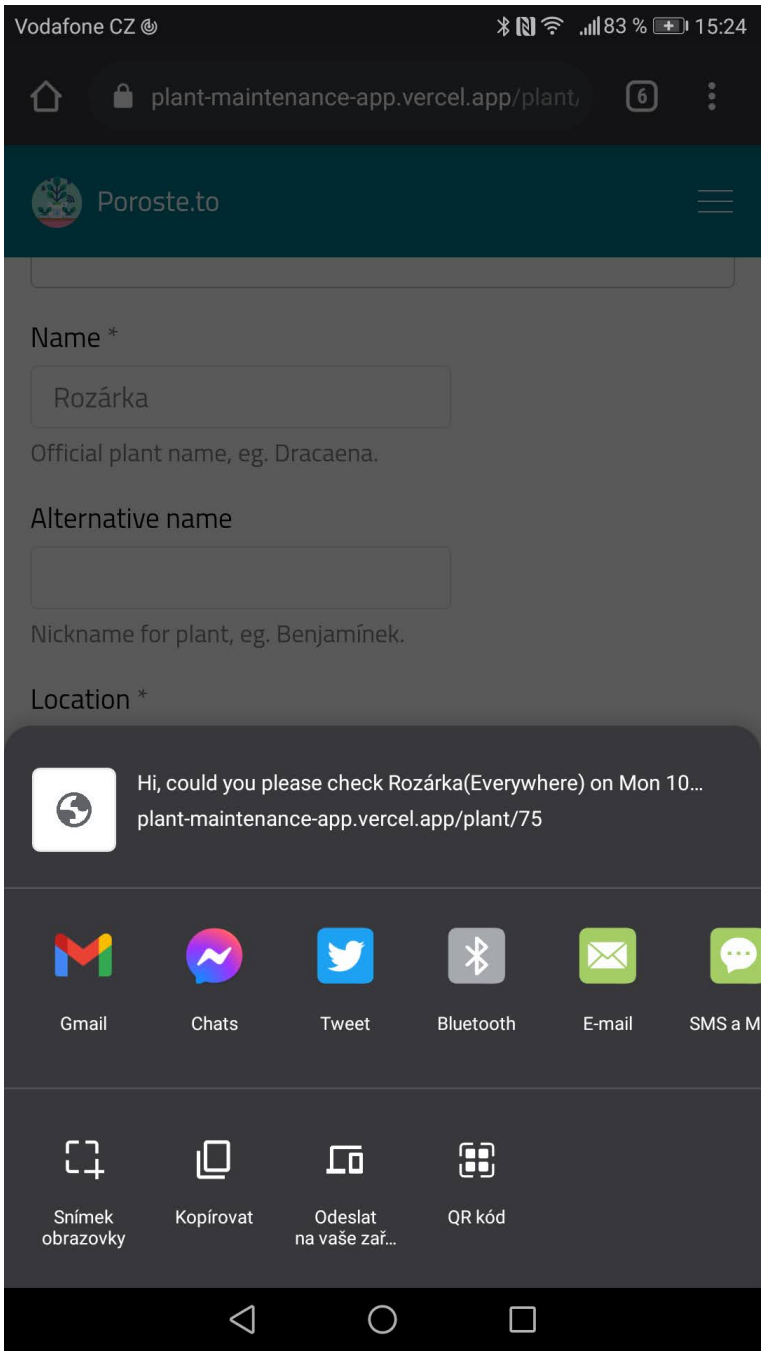
Obrázek 5.5: Legenda k aktuálnímu stavu vlhkosti u komponenty s nadcházejícími upomínkami



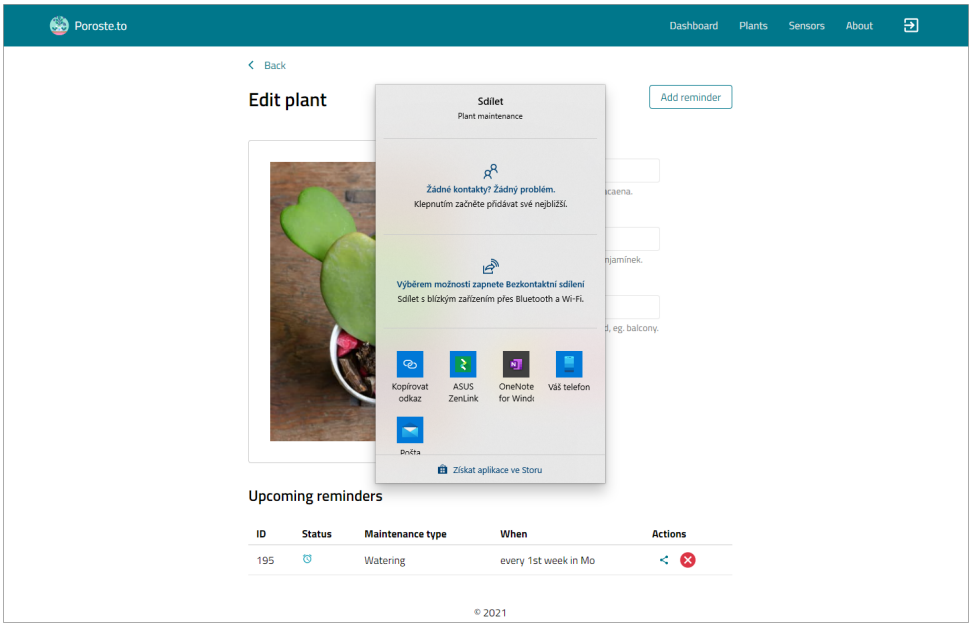
Obrázek 5.6: Ukázka obrazovky s využitím Web Share API



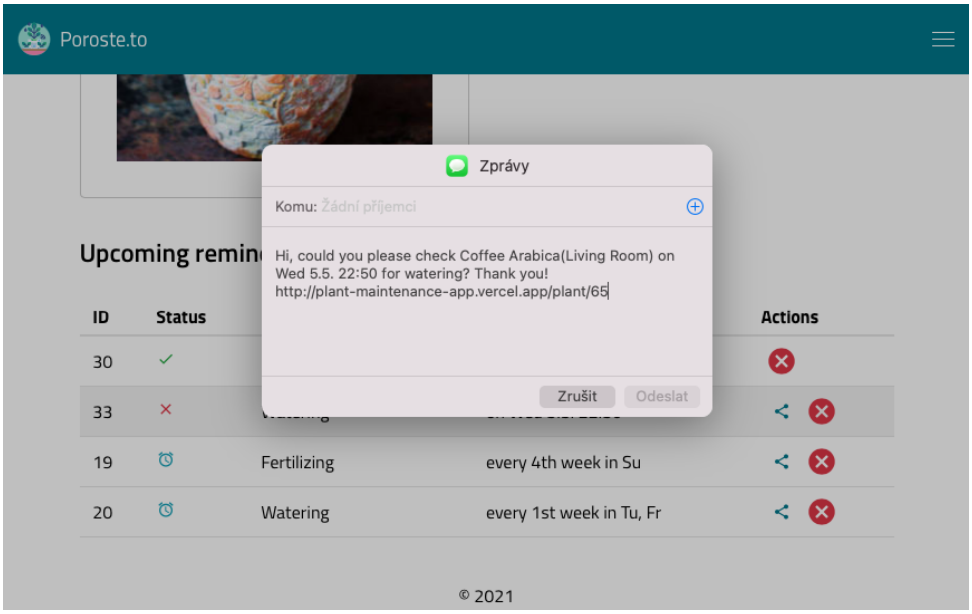
Obrázek 5.7: Sdílení notifikace v prohlížeči Safari



Obrázek 5.8: Sdílení notifikace na platformě Android v prohlížeči Chrome



Obrázek 5.9: Sdílení notifikace na platformě Windows v prohlížeči Chrome



Obrázek 5.10: Sdílení notifikace přes aplikaci Zprávy na platformě macOS

```
const share = (title, text) => {
  if (navigator.share) {
    const data = {
      title,
      text,
      url: document.location.href,
    };
    return navigator.share(data);
  }

  return navigator.clipboard.writeText(text);
};

const handleShare = () => {
  share(
    'Plant maintenance',
    `Hi, could you please check Ficus (Living room) on Wed 5.5.
    12:00 for watering? Thank you!`,
  ).then(() => {
    updateAlert(`Content ${navigator.share ? 'shared' : 'copied'
    }!`, 'success');
  }).catch(() => {
    updateAlert('Something went wrong.', 'error');
  });
};
```

Fragment kódu 5.1: Výňatek ze zdrojového kódu pracující s Web Share a Clipboard API

5.4 Přijímání push notifikací

Pro zobrazení push notifikace je nutné pracovat s Notifications a Push API. Notifications API se stará o zobrazení zprávy na zařízení uživatele. Push API umožňuje přijímat zprávy ze serveru. Přijaté zprávy ze serveru zpracuje service worker.

Z kapitoly 3.2.4 vyplývá, že pro přijímání push notifikací je nutné implementovat tyto kroky:

1. (ověřit podporu push notifikací a service workeru na platformě),
2. (zaregistrovat service worker),
3. získat od uživatele povolení k zobrazení push notifikací,
4. vytvořit potvrzení (či identifikátor) k zasílání oznámení, tzv. push notification subscription,
5. uložit subscription na push server,
6. odeslat zprávu z push serveru do aplikace, resp. service workeru,
7. zobrazit notifikaci.

Aplikace uživateli posílá nesplněné upomínky v podobě push notifikací. K jejich odběru se uživatel může přihlásit na hlavní stránce u sekce s nadcházejícími upomínkami. Tlačítko k odběru notifikací se zobrazuje pouze těm uživatelům, jejichž zařízení podporuje Push API.

V následující kapitole budou popsány implementace jednotlivých kroků. Ještě předtím by bylo vhodné připomenout veškeré artefakty, které v tomto procesu vystupují.

Service worker Jedná se o konfigurovatelnou proxy, jejímž úkolem je přijímání a vykreslení notifikací.

Push service Služba, která funguje jako prostředník mezi klientem a serverem. Stará se o autentizaci a routování požadavků na zobrazení notifikací u klienta i v době jeho nečinnosti.

PushSubscription Též odběr push zpráv. Jedná se o objekt, který poskytuje informace potřebné pro odesílání a šifrování zpráv.

Push server Server, který odesílá push zprávy na adresu, která je součástí vygenerovaného subscriptionu.

5.4.1 Povolení k zobrazení zpráv

Pro zobrazení notifikací musí aplikace obdržet svolení od uživatele. Svolení je nutné pouze za předpokladu, že prohlížeč podporuje service worker a Push API, které mají na starost veškerou logiku kolem notifikací.

```
function isPushNotificationSupported() {
  return 'serviceWorker' in navigator && 'Notification' in window
    && 'PushManager' in window;
}
...
if (isPushNotificationSupported()) {
    createSubscription();
} else {
    setReceivingCommonNotifications(true);
}
```

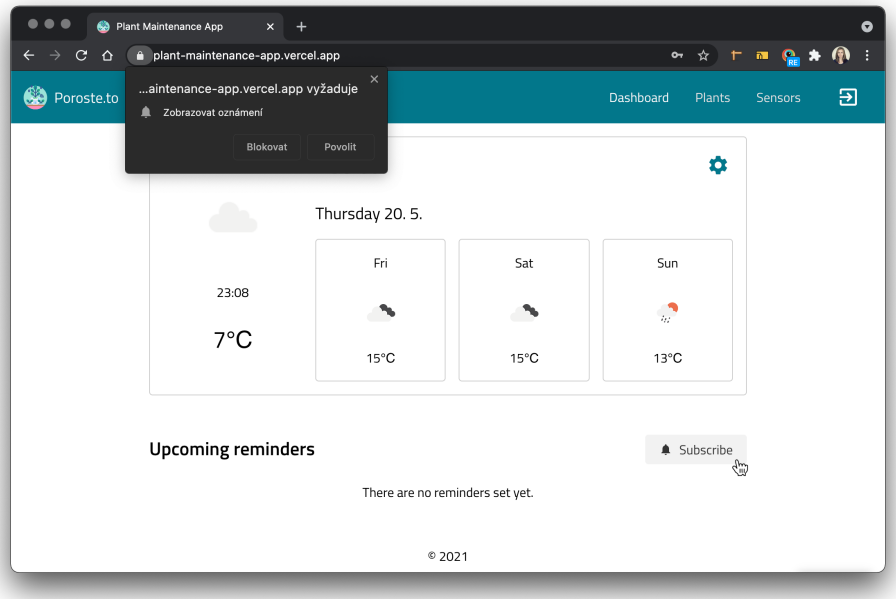
Fragment kódu 5.2: Ověření podpory service workeru a Push API

Kliknutím na tlačítko 'Subscribe' se zavolá kód v ukázce 5.3, který vyvolá zobrazení potvrzovacího okna, viz snímek 5.11. Funkce následně v konstrukci `promise` navrátí souhlas uživatele, který může být buď:

- default – neudělení oprávnění, resp. uživatel se nevyjádřil,
- granted – přijetí,
- denied — zamítnutí.

```
function getUserPermission() {
  return Notification.requestPermission();
}
...
getUserPermission().then((consent) => {
    if (consent === 'granted') {
        handleNotifications();
    } else if (consent === 'denied') {
        console.log('Denied the consent to receive notifications.
    ');
    }
}
```

Fragment kódu 5.3: Volání metody pro získání souhlasu k zobrazení notifikací



Obrázek 5.11: Získání povolení k posílání push notifikací

5.4.2 Přihlášení klienta k odebírání push notifikací

Aby bylo možné zpracovávat push notifikace ze serveru, je nutné vytvořit push notification subscription, neboli jednoznačný identifikátor uživatele pro odběr push zpráv. Service worker vytvoří objekt s názvem `PushSubscription`, který obsahuje veškeré informace k tomu, aby bylo možné odesílat push zprávy – konkrétně koncovou adresu (endpoint) a klíče (keys). Endpoint slouží jako identifikátor klienta a určuje adresu pro odesílání push zpráv.

Z bezpečnostních důvodů je nutné šifrovat komunikaci mezi serverem a klientem. K tomu slouží veřejný a privátní klíč vygenerovaný podle VAPID[1]

[1]Voluntary Application Server Identification for Web Push – jedná se o specifikaci, která opravňuje server k odesílání push notifikací konkrétnímu uživateli a určuje pro push service adresáta zprávy

specifikace. Klíče je možné vygenerovat pomocí npm balíčku `web-push` příkazem `web-push generate-vapid-keys`.

Subscription se vygeneruje zavoláním metody `serviceWorker.pushManager.subscribe()`, která přijímá jako parametr objekt s parametry:

- `userVisibleOnly` – Jedná se o povinný parametr, jehož hodnota se musí rovnat `true`, což udává povinnost zobrazit oznámení pokaždé, když se odešle push zpráva.
- `applicationServerKey` – Též veřejný VAPID klíč, který se používá k dešifrování zpráv.

```
const serverPublicKey = 'Bd7...';

async function createNotificationSubscription() {
  const serviceWorker = await navigator.serviceWorker.ready;

  return serviceWorker.pushManager.subscribe({
    userVisibleOnly: true,
    applicationServerKey: serverPublicKey,
  });
}
```

Fragment kódu 5.4: Vytvoření push notification subscription

Metoda vrací objekt PushSubscription, který má strukturu uvedenou v kódu 5.5.

```
{
  "endpoint": "https://domain.pushservice.com/some-id",
  "keys": {
    "p256dh": "BIPUL12DLfytvTajnr...",
    "auth": "FPssMOQPmLmXWmdSTdbKVw=="
  }
}
```

Fragment kódu 5.5: Struktura objektu PushSubscription

Parametr `endpoint` je unikátní koncová URL svázaná s daným uživatelem. Jedná se o adresu, na kterou push server zasílá zprávy, které následně zpracuje service worker a zobrazí uživateli. Vzhledem k tomu, že každý prohlížeč používá jiný push service, jsou generovány různé adresy. V prohlížeči Chrome vypadá endpoint následovně:

`https://fcm.googleapis.com/fcm/send/fXjr1iIPn00...`

a v prohlížeči Firefox takto:

`https://updates.push.services.mozilla.com/wpush/v2/gAAA...`

Jakmile je tento objekt úspěšně vygenerován, je zapotřebí jej prostřednictvím POST requestu odeslat na server a uložit k příslušnému uživateli.

5.4.3 Odeslání a zobrazení push notifikace

Pro odesílání push notifikací ze serveru se nejčastěji používá knihovna `web-push`. Notifikace by měla respektovat strukturu definovanou Notifications API a spo-

lečně s objektem zprávy by se měl předat i objekt `subscription`, který specifikuje cílového adresáta.

```
function sendPushNotification() {
  webpush
    .sendNotification(
      subscription,
      JSON.stringify({
        title: 'Notifications has been subscribed!',
        text: 'You\'ll be notified in case of low soil moisture.'
    ,
        url: '/sensors',
      }),
    );
}
```

Fragment kódu 5.6: Struktura objektu PushSubscription

Zpráva je odeslána service workeru, který ji zaregistruje pouze v případě, že je implementován posluchač na událost `push`. Zavoláním metody `self.registration.showNotification` se uživateli zobrazí notifikace na jeho zařízení. Aby si mohl zobrazit zprávu, je nutné přidat posluchač i na kliknutí na notifikaci. Ukázka zdrojového kódu je níže na 5.7.

```
function openPushNotification(event) {
  event.notification.close();
  event.waitUntil(clients.openWindow(event.notification.data));
}

function receivePushNotification(event) {
  const {
    image,
    tag,
    text,
    title,
    url,
  } = event.data.json();

  const options = {
    badge: '/icon.png',
    body: text,
    data: url,
    icon: '/icon.png',
    image,
    tag,
    vibrate: [200, 100, 200],
  };

  event.waitUntil(self.registration.showNotification(title,
    options));
}

self.addEventListener('notificationclick', openPushNotification);
self.addEventListener('push', receivePushNotification);
```

Fragment kódu 5.7: Vykreslení push notifikace

5.5 Limity stávajícího řešení a potenciální rozšíření

Současná verze aplikace pracuje se službami třetích stran, jako je Cloudinary pro správu vizuálního obsahu, OpenWeatherMap pro získávání předpovědi počasí a Adafruit IO za účelem správy dat ze senzoru. Všechny tyto služby přináší jistý komfort pro vývoj a sdílení obsahu společně s jistými omezeními, které se odvíjí od ceny služby či integrity dat.

Uživatel je v tuto chvíli silně vázán na platformu, která poskytuje data ze senzoru, což není uživatelsky vstřícným řešením.

Upomínky, které si uživatel může v aplikaci nastavit pro jednotlivé rostliny, jsou zobrazovány prostřednictvím Notifications API a uživateli se zobrazí v momentě, kdy se přihlásí do aplikace. Vhodnějším řešením by bylo prostřednictvím nástroje `cron` procházet v pravidelných cyklech nastavené upomínky a v případě blížícího se termínu správy rostlin, odeslat ze serveru push notifikaci, která uživatelům včas připomene jejich upomínku.

Zajímavým, byť dosud experimentálním, řešením by mohlo být využití Notification Trigger API, které umožňuje plánování notifikací i bez internetového připojení. Pokud chceme zobrazovat notifikaci na základě splnění

nějaké podmínky (příkladem může být poloha uživatele, případně specifický čas), Push API neumí spolehlivě reagovat na tuto akci. Zpráva může být doručena se zpožděním, kvůli síťovému připojení nebo nastavení úsporného režimu baterie. Notification Trigger API předává zobrazování notifikací na starost operačnímu systému, který je doručí v čas, nezávisle na internetové připojení či režimu baterie. Implementace je totožná práci s Notification API – nově je zde parametr `showTrigger`, který představuje podmínku pro zobrazení zprávy. Níže 5.8 je ukázka kódu pro zobrazení notifikace 10 minut před konáním schůzky.

```
async function createAppointment(tag, title, timestamp) {
  const TEN_MINUTES = 10 * 1000;
  const swRegistration = await navigator.serviceWorker.
  getRegistration();

  await swRegistration.showNotification(
      'Appointment reminder',
      {
        tag: timestamp,
        body: 'Your appointment is due in ten minutes!',
        showTrigger: new TimestampTrigger(timestamp -
  TEN_MINUTES),
      }
    );
}
```

Fragment kódu 5.8: Struktura objektu PushSubscription

5.5.1 Potenciální rozšíření

Potenciální funkční a nefunkční požadavky, kterými by bylo možné rozšířit, resp. vylepšit, aplikaci, mohou být:

- **konfigurace vlastního serveru pro zpracování a sdílení dat ze senzoru** díky čemuž by byl uživatel odstíněn od nutnosti konfigurace senzoru,
- **důslednější validace dat a další bezpečnostní opatření**, jak na klientské, tak serverové straně,
- **systémové a unit testy** pro ověření bezproblémového chodu aplikace,
- **přihlašování přes sociální sítě** jako je Google, Facebook či Apple ID,
- **historie údržby rostlin**,
- **galerie fotek či videí u rostliny**,
- **konfigurace dalších senzorů**, např. pro měření teploty nebo vlhkosti vzduchu, případně světelných podmínek,
- **AI pro rozpoznání rostlin** díky čemuž by uživatelé měli veškeré informace o rostlině v aplikaci; v případě dostatečně komplexních dat by bylo přínosné automatizovat nastavování upomínek na údržbu rostlin.

Kapitola 6

Uživatelské testování použitelnosti

V rámci bakalářské práce proběhlo uživatelské testování prototypu i finální aplikace. s ohledem na současné funkční požadavky je cílem tohoto testování ověřit, zdali je nově přidaná funkcionalita uživatelsky přívětivá. Testování použitelnosti bude probíhat formou kvalitativního testování.

Dotazníky včetně získaných odpovědí si lze prohlédnout na adrese:
https://tinyurl.com/dotazniky,
eventuálně jsou součástí přílohy závěrečné práce.

6.1 Cílová skupina a participanti

Cílová skupina zůstává identická. Jedná se o jedince ve věku od 18–35 let s pasivní znalostí anglického jazyka alespoň na úrovni A2, kteří se starají minimálně o jednu rostlinu.

Oslovování participantů bude probíhat prostřednictvím sociálních sítí. Vzhledem k tomu, že testovaná část aplikace není tak rozsáhlá, postačující počet participantů pro nalezení nejzávažnějších problémů je pět.

6.1.1 Vstupní dotazník

Pro selekci participantů z řad oslovených jedinců slouží vstupní dotazník, neboli screener. Níže jsou uvedeny otázky, které v něm zazněly. Otázky označené * jsou povinné.

1. **Uveďte, prosím, své přidělené identifikační číslo.** *
2. **Kolik Vám je let?** *
   a. méně než 18 let
   b. 18–23 let
   c. 24–29 let
   d. 30–35 let
   e. více než 35 let
3. **Studujete, či pracujete?** *

a. Studuji
b. Pracuji
c. Obojí
d. Žádné z výše uvedených

4. **Máte alespoň jednu rostlinu, o kterou pečujete?** *
   a. Ano
   b. Ne
5. **Pokud ano, jakými způsoby se o ni staráte?** *
   a. Zalévání
   b. Přesazování
   c. Zastřihávání
   d. Hnojení
   e. Jiné
6. **Daří se Vám tyto činnosti pravidelně?** *
   a. Ano
   b. Spíše ano
   c. Někdy
   d. Spíš ne
   e. Ne

Všichni participanti neměli doposud žádné zkušeností s aplikací tohoto typu, ale shledávají je užitečnými.

Ocenili by, kdyby aplikace disponovala atlasem rostlin a umožnila jim jednoduše a rychle vyhledat, jak se mají o danou rostlinu starat, kde by měla být umístěna (venku, ve stínu, na světle, atd.) a další informace s ní spjaté. Obdržení notifikací s připomenutím údržby rostliny považují za samozřejmost. Jedna z participantek uvedla, že by ráda skrze aplikaci ovládala zavlažování rostliny a mimo vlhkosti půdy monitorovala i teplotu, intenzitu světla či vlhkost vzduchu.

6.2 Průběh testování

Testování bylo moderované autorkou práce. Probíhalo na základě preferencí participanta, a to buď online formou prostřednictvím platforem se sdílením obrazovky (Zoom), nebo jako osobní setkání v bytě autorky práce. Dotazníky vyplňovali participanti v aplikaci Google Forms.

Pro účely testování bylo nutné vytvořit profil uživatele včetně vytvoření rostlin, jelikož tyto úkoly byly již předmětem testování použitelnosti v bakalářské práci.

Těsně před zahájením testování byl participant požádán o vyplnění pre-test dotazníku, který se skládal z doplňujících otázek. Otázky byly následující:

1. **Máte zkušenosti s aplikací pro údržbu rostlin? Pokud ano, uveďte, prosím, název a jak byste ji celkově ohodnotili.***

2. **Myslíte si, že je užitečné mít aplikaci, které Vám pomáhá s údržbou rostlin a připomíná Vám, kdy se o ně máte starat? ***

3. **Co všechno byste od takové aplikace očekávali? Co vše by podle Vás měla umět, aby byla pro Vás užitečná? ***

Po vyplnění dotazníku byl participantovi předložen seznam úkolů, které by měl v rámci průchodu aplikací zrealizovat. Testovaní probíhalo na vlastním zařízení participanta.

6.2.1 Průchod aplikací

Při průchodu aplikací měli participanti projít následující úlohy. Každý úkol měli následně ohodnotit, z hlediska náročnosti na stupnici od 1 do 5, kde 1 znamenalo 'Velmi jednoduché' a 5 'Velmi obtížné'.

Pod dílčími úkoly jsou uvedeny i očekávané stavy po úspěšném splnění zadání.

Součástí dotazníku byl i popis testovacího prostředí. Funkcionalita, která je předmětem testování, má odlišnou podporu napříč platformami a z toho důvodu je nutné s tím počítat při vyhodnocování zpětné vazby.

1. **Přistupte na stránku https://plant-maintenance-app.vercel.app/.**
   Participant se úspěšně dostane na stránku aplikace.

2. **Přihlaste se do aplikace. Níže jsou uvedené přihlašovací údaje:**
   username: tester@email.com
   password: tester123
   Participant se úspěšně přihlásí do aplikace s výše uvedenými údaji.

3. **Vytvořte v aplikaci upomínku na pravidelné zalévání rostliny.**
   Participant si zobrazí detail libovolné rostliny, kde následně vytvoří opakující se upomínku na zalévání rostliny.

4. **Sdílejte tuto upomínku prostřednictvím aplikace.**
   Participant na stránce nalezne výpis upomínek a prostřednictvím sdílecí ikony přepošle upomínku skrze jím zvolenou nativní aplikaci z nabídky.

5. **Připojte v aplikaci senzor na měření vlhkosti půdy k rostlině Dragon Scale. Údaje jsou:**
   username: dlouhka1
   feedname: humidity
   Participant nalezne formulář pro přidání senzoru a na základě údajů výše připojí do aplikace nový senzor.

6. **Zjistěte, jaká byla poslední naměřená hodnota z nově přidaného senzoru a kdy byla tato hodnota naměřena.**
   Uživatel bude schopen z grafu vyčíst datum a hodnotu posledního měření vlhkosti půdy.

Níže je ve stručnosti uveden průběh testování s jednotlivými participanty. Oslovený respondent s identifikačním číslem P4 nebyl vhodným participantem, proto se nezúčastnil testování.

Participant P1

Participantka je pracující studentka ve věku 24–29 let, která se stará o své rostliny formou zalévání, přesazování, hnojení, zvlhčování a omývání listů. Tyto činnosti se jí daří spíše pravidelně.

Testování probíhalo prostřednictvím platformy Zoom. Uživatelka pracovala s aplikací v prohlížeči Firefox (Windows 10).

Přístup do aplikace proběhl bez problémů. U úkolu č. 3 na přidání upomínky participantka předpokládala, že je možné přidávat upomínku z hlavní stránky.

U přidávání senzoru bylo matoucí, že neprobíhá stejně jako přidávání upomínky, na detailu rostlinu. Později ale participantka zmínila, že je pro ni logičtější tuto akci provádět ve výpisu senzorů.

U vykreslených dat ze senzoru participantka postrádala konkrétní datum, kdy byla hodnota naměřena a možnost zpětně sledovat vlhkost půdy, například o dva dny zpátky.

Participant P2

Participantka je studentka ve věku 18–23 let, která se stará o své rostliny formou zalévání a hnojení. Tyto činnosti se jí nedaří pravidelně.

Testování probíhalo prostřednictvím platformy Zoom. Uživatelka pracovala s aplikací v prohlížeči Safari (iOS).

Přístup do aplikace proběhl bez problémů.

U úkolu č. 3 participantka zmínila, že by očekávala nápovědu od aplikace, jak často by měla květinu zalévat.

Sdílení upomínky bylo pro participantku nejnáročnější. Zadání pro ni bylo nejednoznačné a ikona sdílení byla příliš malá.

Po připojení senzoru byla uživatelka mírně zmatená z vykreslených dat. Na horizontální ose podle ní chybělo celé datum. Uvítala by, kdyby sledovala pouze jeden naměřený záznam za den.

Participant P3

Participantem je pracující muž ve věku 24–29 let, který se o rostliny stará formou nepravidelného zalévání.

Testování probíhalo naživo v bytě autorky práce. Participant testoval aplikaci v prohlížeči Safari (iPadOS).

První čtyři úkoly probíhaly bez problémů. Participant naprosto plynule a intuitivně pracoval s aplikací.

Obdobně jako ostatní participanti nejprve hledal přidávání notifikace na hlavní stránce.

Přidávání senzoru i práci s naměřenými daty hodnotil uživatel jako náročnější, jelikož očekával, že stejně jako notifikace, bude možné propojit senzor s konkrétní rostlinou u stránky s jejím detailem.

Kvůli chybějícím popisku os, nezaokrouhleným naměřeným hodnotám a neintuitivní práci s grafem byl participant zmatený z posledního úkolu.

Participant P5

Participantem je student ve věku 24–29 let, který se převážně o rostliny stará formou nepravidelného zalévání.

Testování probíhalo prostřednictvím platformy Zoom. Uživatel pracoval s aplikací v prohlížeči Chrome (Windows 10).

Z počátku bylo pro uživatele matoucí zobrazení hlavní stránky a pak teprve přihlašovací.

Po přihlášení do aplikace bylo pro participanta náročné vytvořit upomínku. Předpokládal, že na hlavní stránce bude tlačítko na přidání upomínky, což nebylo. Jakmile se ocitl na detailu rostliny, nevšiml si zpočátku tlačítka na přidání připomenutí a hledal tuto akci jinde na stránce. Po té, co klikl na tlačítko 'Add reminder', nebyl si jistý, jakým způsobem je možné přidávat opakující se upomínky. Bylo by vhodné napsat, o jaký interval se jedná (tj. upomínku je možné nastavit na každý druhý týden v měsíci apod.).

Se sdílením upozornění byl pouze drobný zádrhel, jelikož uživatel neměl k dispozici žádnou aplikaci, se kterou by mohl sdílet notifikaci, proto si pouze zkopíroval odkaz, který byl součástí notifikace.

Přidání senzoru proběhlo bez problémů. Participant pouze zmínil, že by bylo vhodné specifikovat jednotku, ve které jsou data naměřena (tj. procenta).

Participantka P6

Participantkou je pracující studentka ve věku 24–29 let, která se stará o své rostliny formou zalévání a zastřihávání. Tyto činnosti se jí nedaří pravidelně.

Testování probíhalo naživo v bytě autorky práce. Participantka testovala na svém mobilním telefonu s operačním systémem Android v prohlížeči Firefox.

Přihlášení do aplikace proběhlo bez problémů. Stejně jako předchozí participanti, bylo pro ni obtížné nalézt přidání notifikace.

Při sdílení notifikace bylo možné odeslat pouze odkaz v ní přiložený. Mobilní prohlížeč Firefox na platformě Android umožňuje sdílet pouze odkaz [2].

Přidání senzoru proběhlo bez problémů. Participantka pouze zmínila, že by bylo vhodné zaokrouhlit naměřenou hodnotu a uvést, v jaké jednotce je naměřena. Zmínila, že by pro ni bylo přínosné mít v grafu vyznačenou horní a dolní hranici vlhkosti půdy, aby věděla, kdy rostlina trpí suchem a kdy je naopak přelitá.

Náročnost jednotlivých úkolů

V tabulce 6.1 je uvedeno hodnocení náročnosti jednotlivých úkolů při průchodu aplikací. Z mediánu lze vyčíst, že nejobtížnější úlohou bylo nalezení tlačítka pro vytvoření notifikace. Jedná se o signifikantní nález, který by bylo vhodné řešit například tlačítkem na hlavní obrazovce, onboarding obrazovkou nebo stránkou s výpisem notifikací, která v této verzi aplikace chybí.

ID	P1	P2	P3	P5	P6	medián
2	1	1	1	1	1	*1*
3	3	1	1	3	4	*3*
4	1	2	1	2	2	*2*
5	1	1	3	1	1	*1*
6	1	1	4	2	2	*2*

Tabulka 6.1: Náročnost jednotlivých úkolů při průchodu aplikací

6.2.2 Post-dotazník

Po průchodu aplikací měli participanti zodpovědět na otázky, které zjišťovaly celkový dojem z aplikace, jak se jim s ní pracovalo, co by v aplikaci zlepšili a proč.

1. **Bylo pro Vás náročné se zorientovat v aplikaci?** *
   a. Ano
   b. Spíše ano
   c. Nevím
   d. Spíše ne
   e. Ne
2. **Který úkol Vám přišel nejnáročnější a proč?** *
3. **Co byste hodnotil/a kladně na aplikaci?** *
4. **Co byste naopak hodnotil/a záporně na aplikaci?** *
5. **Jak na Vás aplikace celkově působila?** *
   a. Velmi příjemně
   b. Dobře
   c. Průměrně
   d. Špatně
   e. Naprosto otřesně
6. **Chtěl/a byste takovou aplikaci používat?** *
   a. Ano

b. Spíše ano
c. Nevím
d. Spíše ne
e. Ne

7. **Pokud byste rád/a něco dodala k aplikaci nebo k testování samotnému, můžete se k tomu vyjádřit zde.**

Ze zpětné vazby je patrné, že uživatelé neměli zásadní problémy se zorientováním se v aplikaci. Nejnáročnější pro ně byl úkol s přidáváním upozornění, proto by uvítali více možností, jak přidávat plánovanou péči o rostliny. Kladně hodnotili vzhled a jednoduchost aplikace a měli by zájem podobnou aplikaci používat.

6.3 Závěr

Testování použitelnosti odhalilo sadu nálezů, jejichž závažnost je ohodnocena na škále 1 až 5, kde 1 označuje minoritní nález, jehož dopad není zásadní a 5 naopak definuje nález, který je markantní a je nutné jej neprodleně opravit. Seznam nálezů včetně potenciálních řešení je uveden v tabulce 6.2.

Významný dopad na testování měla podpora použitých technologií napříč prohlížeči. Zařízení s operačním systémem iOS, případně iPadOS, nepodporují přijímání push zpráv a z toho důvodu se aktivita nestala předmětem testování. Participanti měli možnost se k současnému řešení zpráv vyjádřit a i tak poskytnout zpětnou vazbu.

Napříč prohlížeči bylo i vizuálně rozdílné rozhraní pro sdílení obsahu, jelikož je implementováno zvlášť operačním systémem. Ani tento fakt však nezpůsobil signifikantní problémy při testování daného úkolu.

Pro účely rozsáhlejšího výzkumu by mohlo být zajímavé testovat více způsobů vizualizace dat získaných ze senzoru. Použití spojnicového grafu bylo pro účely práce nejvhodnější volbou, jelikož je tento typ grafu nejznámější a pro uživatele nejsnáze pochopitelný.

ID	POPIS	ZÁVAŽNOST	ŘEŠENÍ
N1	Latence při načtení přihlašovací stránky.	2	Unit testy.
N2	Neintuitivní umístění přidání senzoru.	4	Umístění tlačítka na přidání senzoru na hlavní stránku. Provést uživatele aplikací při registraci.
N3	Neintuitivní umístění přidání upomínky.	3	Umístění tlačítka na přidání upomínky na hlavní stránku. Provést uživatele aplikací při registraci. Přidat stránku se výpisem upomínek.
N4	Nevhodné vykreslení dat ze senzoru.	4	Přidání filtru pro selekci dat (za poslední den, poslední 3 dny, ...). Detailnější označení horizontální a vertikální osy.
N5	Chyba při výběru týdne u opakující se upomínky.	3	Unit testy.
N6	Nedostačující velikost ikony sdílení u upomínky.	2	Zvětšení ikony. Provést uživatele aplikací při registraci.

Tabulka 6.2: Přehled nálezů při průchodu aplikací

Kapitola 7

Závěr

Hlavním cílem práce byla integrace dat ze senzoru na měření vlhkosti půdy. Naměřená data měla být v reálném čase vykreslena v již existující aplikaci, která byla výstupem bakalářské práce. V této práci vyplynuly z testování náměty na vylepšení, které se staly předmětem diplomové práce. Jedná se například o sdílení upomínek prostřednictví Web Share API nebo přijímání push notifikací.

V samotném úvodu práce je představení IoT. Rešerše se zabývá nejenom architekturou a způsobem komunikace prostřednictvím protokolu MQTT, ale i otázkou bezpečnosti a integrity dat. Kapitola pojednává o funkci mikrokontrolerů, které jsou stěžejním prvkem v celé komunikaci.

Čtenář je dále seznámen se současnou definicí termínu progresivní webové aplikace. Kapitola představuje i nově použité technologie zvolené pro rozšíření aplikace, ať klientské či serverové části. Jak Web Share API, tak push notifikace, jsou limitovány podporou platforem. Byť je myšlenka push notifikací poměrně zajímavá v kontextu s webovými aplikacemi, tak tato funkcionalita není dostupná pro uživatele prohlížeče Safari a operačního systému iOS.

Nedílnou součástí byla implementace vlastního prototypu senzoru na měření vlhkosti půdy, který měl posloužit k hlavnímu cíli práce. Samotná implementace je včetně zdrojového kódu obsažená v práci. V závěru kapitoly je zhodnocení prototypu včetně energetické náročnosti. Pro měření vlhkosti půdy se použil kapacitní senzor, který pro účely prototypu je poměrně dostačující, avšak v mnoha ohledech by bylo vhodnější použití senzoru TDR, který je odolnější vůči korozi a byl by schopen měřit větší objem půdy než senzor kapacitní.

Webová aplikace, která byla výstupem bakalářské práce, prošla zásadní změnou. Původně byla aplikace napsaná za pomocí čistého JavaScriptu a veškerá logika se odehrávala v prohlížeči, tj. nedocházelo k perzistentnímu ukládání dat a aplikace tím pádem nebyla přenositelná. Pro realizaci push notifikací je zapotřebí komunikace se serverem. Z těchto důvodů práce navíc zahrnuje oproti zadání přepsání této aplikace. Klientská část je nově napsána za použití frameworku ReactJS a serverová část v jazyku NodeJS a s databází MySQL.

Uživatelské testování použitelnosti ukázalo, že vizualizace dat prostřednictvím spojnicového grafu byla vhodnou volbou. Nutno podotknout, že většina

participantů očekávala více možností s manipulací naměřených dat, jako je například filtrování či skrolování.

Při testování byly nalezeny nedostatky v podobně neintuitivního přidávání senzoru či upomínek do aplikace, což jsou nálezy s vysokou závažností, které je nutné upravit. Ze zpětné vazby vyplynuly zajímavé podněty od respondentů, kterými by bylo vhodné aplikaci vylepšit. Jedná se například o součinnost s atlasem rostlin, díky čemuž by mohli prostřednictvím aplikace zjišťovat informace o vhodné péči pro danou rostlinu. Vítaným funkčním požadavkem by byla možnost automaticky generovaných upomínek na základě typu rostliny a přijímání notifikací ze senzorů.

7.1 Přínos práce

Realizace práce byla pro autorku velmi přínosná. Rešerše z oblasti IoT, která zahrnovala i detailní rozbor senzorů, principu měření vlhkosti půdy (tyto kapitoly nebyly součástí závěrečné práce, nýbrž softwarového výzkumného projektu) včetně programování vývojových desek, bylo vítaným rozšířením obzorů.

Vzhledem k rozsáhlému přepsání práce se autorka teoreticky i prakticky blíže seznámila s komplexním procesem vývoje softwaru, který oproti bakalářské práci zahrnoval implementaci serverové části včetně DevOps procesů.

Výsledný projekt má prostory pro zlepšení, ale už i nyní má jistý potenciál, kterého by bylo možné využít. Z práce vyplynuly jisté návrhy na zlepšení samotného senzoru i aplikace jako takové. Pokud by se podařilo sestavit senzor, který by vyžadoval téměř nulovou součinnost od uživatele při jeho konfiguraci a vyjma sledování vlhkosti půdy by měřil i jiné důležité vlastnosti (jako je vlhkost vzduchu, okolní teplota či intenzita světla), které by se daly využít k lepší péči o rostliny. Vítaným rozšířením by mohlo být propojení s databází rostlin, která by uživateli zobrazovala tipy pro údržbu rostliny, případně jej na základě obdržených dat ze senzorů upozornila na nevhodné prostředí, ve kterém se rostlina nachází.

Literatura

[1] Adafruit IO API Reference. https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api. Navštíveno dne 27. 4. 2021.

[2] [Bug] Web share no longer shares title or text · Issue #11946 · mozilla-mobile/fenix. https://github.com/mozilla-mobile/fenix/issues/11946. Navštíveno dne 15. 5. 2021.

[3] Can I use... Support tables for HTML5, CSS3, etc. https://caniuse.com/web-share. Navštíveno dne 27. 4. 2021.

[4] Can I use... Support tables for HTML5, CSS3, etc. https://caniuse.com/push-api. Navštíveno dne 27. 4. 2021.

[5] How to Choose a Microcontroller for IoT - DZone IoT. https://dzone.com/articles/how-to-choose-a-microcontroller-for-iot. Navštíveno dne 27. 4. 2021.

[6] How to Create Web Push Notifications - Full Tutorial. https://felixgerschau.com/web-push-notifications-tutorial/. Navštíveno dne 15. 5. 2021.

[7] Line Chart | Charts. https://developers.google.com/chart/interactive/docs/gallery/linechart?hl=cs. Navštíveno dne 7. 5. 2021.

[8] MQTT - The Standard for IoT Messaging. https://mqtt.org/. Navštíveno dne 14. 1. 2021.

[9] Navigator.share() - Web APIs | MDN. https://developer.mozilla.org/en-US/docs/Web/API/Navigator/share. Navštíveno dne 27. 4. 2021.

[10] Notification Triggers. https://web.dev/notification-triggers/. Navštíveno dne 15. 5. 2021.

[11] US $0.42 6% OFF|Smart Electronics Soil Moisture Hygrometer Detection Humidity Sensor Module For arduino Development Board DIY

Robot Smart Car|soil moisture|water sensorsoil hygrometer - AliExpress. www.aliexpress.com/item/32704803481.html. Navštíveno dne 9. 1. 2021.

[12] Visualization: Gauge | Charts. https://developers.google.com/chart/interactive/docs/gallery/gauge?hl=cs. Navštíveno dne 7. 5. 2021.

[13] Web Push Notifications: Timely, Relevant, and Precise. https://developers.google.com/web/fundamentals/push-notifications?hl=cs. Navštíveno dne 15. 5. 2021.

[14] What makes a good Progressive Web App? https://web.dev/pwa-checklist/. Navštíveno dne 27. 4. 2021.

[15] The Current State of IoT Security: How We Got Here and What's at Risk. https://www.dogtownmedia.com/the-current-state-of-iot-security/, Mar. 2020. Navštíveno dne 15. 5. 2021.

[16] Difference between MQTT and HTTP protocols. https://www.geeksforgeeks.org/difference-between-mqtt-and-http-protocols/, červenec 2020. Navštíveno dne 14. 1. 2021.

[17] Why IoT Security Is Important for Your Home Network. https://www.kaspersky.com/resource-center/threats/secure-iot-devices-on-your-home-network, Jan. 2021. Navštíveno dne 7. 5. 2021.

[18] I. Andrea, C. Chrysostomou, and G. Hadjichristofi. Internet of things: Security vulnerabilities and challenges. In *2015 IEEE symposium on computers and communication (ISCC)*, pages 180–187. IEEE, 2015. Přečteno dne 15. 5. 2021.

[19] Bc. Kateřina Dlouhá. adafruit-io-data - StackBlitz. https://stackblitz.com/edit/adafruit-io-data?file=index.js. Navštíveno dne 14. 1. 2021.

[20] Bc. Kateřina Dlouhá. Progresivní webová aplikace pro monitorování a údržbu rostlin. https://dspace.cvut.cz/handle/10467/82344, květen 2019. Navštíveno dne 27. 4. 2021.

[21] cdaviddav. How to reduce the ESP8266 power consumption? https://diyi0t.com/how-to-reduce-the-esp8266-power-consumption/, červenec 2019. Navštíveno dne 9. 1. 2021.

[22] F. Dahlqvist. Growing opportunities in the Internet of Things | McKinsey. https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things, 2019. Navštíveno dne 15. 5. 2021.

[23] I. J. Dvořák. Využití metod environmentální geofyziky na vybraných lokalitách krkonošského národního parku. Str. 48–51, https://dspace.cuni.cz/bitstream/handle/20.500.11956/93727/RPTX_2007_2_11310_MSRZ001_252560_0_176402.pdf?sequence=1, 2007. Navštíveno dne 30. 12. 2020.

[24] ELKO EP s. r. o. Elkoep - výrobce elektronických přístrojů • elko ep s.r.o. https://www.elkoep.cz/homepage. Navštíveno dne 28. 12. 2020.

[25] C. Gregersen. A Complete Guide to Microcontrollers for IoT. https://www.nabto.com/iot-microcontroller-guide/, červenec 2020. Navštíveno dne 27. 4. 2021.

[26] Ing. Milan Kříž. Kapacitní měření vlhkosti. Princip měření. Jak funguje? | EKOTECHNIKA - Ing. Milan Kříž. https://www.ekotechnika.com/clanky/kapacitni-mereni-vlhkosti-princip-mereni-jak-funguje/. Navštíveno dne 23. 12. 2020.

[27] A. Isaiah. How to Use the Web Share API. https://css-tricks.com/how-to-use-the-web-share-api/, červen 2019. Navštíveno dne 27. 4. 2021.

[28] laskarduino.cz. WeMos D1 Mini ESP8266 WiFi modul. https://www.laskarduino.cz/wemos-d1-mini-esp8266-wifi-modul/. Navštíveno dne 9. 1. 2021.

[29] laskarduino.cz. Čidlo pro měření vlhkosti půdy HD-38. https://www.laskarduino.cz/cidlo-pro-mereni-vlhkosti-pudy-hd-38/. Navštíveno dne 30. 12. 2020.

[30] M. Malý. Protokol MQTT: komunikační standard pro IoT. https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/. Navštíveno dne 14. 1. 2021.

[31] J. Misra. Sensors and Actuators in IoT – Enabling Industrial Automation. https://bridgera.com/sensors-and-actuators-in-iot/, červen 2017. Navštíveno dne 21. 12. 2020.

[32] Z. I. of Sustainable Development INE. Low Power Wide Area Network (lpwan). https://www.zhaw.ch/en/engineering/institutes-centres/ine/smart-city-guide-main-page/description-of-applications/low-power-wide-area-network-lpwan/, 2019. Navštíveno dne 17. 11. 2020.

[33] QuicSolv. How Iot Works? https://www.quicsolv.com/internet-of-things/how-iot-works/, 2020. Navštíveno dne 15. 11. 2020.

[34] RedHat. Iaas vs Paas vs Saas. https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas, 2019. Navštíveno dne 17. 11. 2020.

[35] RNDr. Tomáš Lintschmann. Měření půdní vlhkosti - PDF Stažení zdarma. https://docplayer.cz/39172393-Mereni-pudni-vlhkosti.html, 2010. Navštíveno dne 23. 12. 2020.

[36] R. B. (RoboticsBD). RTD PT-100 Temperature Sensor. https://store.roboticsbd.com/sensors/493-rtd-pt-100-temperature-sensor-robotics-bangladesh.html. Navštíveno dne 23. 12. 2020.

[37] M. Rouse. What is Iot (Internet of Things) and How Does it Work? https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT, 2020. Navštíveno dne 9. 11. 2020.

[38] Shawn. What is a light sensor? types, uses, arduino guide. https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-types-uses-arduino-guide/, červen 2020. Navštíveno dne 13. 1. 2021.

[39] L. Spyna. React Push notifications (with hooks). https://itnext.io/react-push-notifications-with-hooks-d293d36f4836, Oct. 2020. Navštíveno dne 15. 5. 2021.

[40] L. Spyna. An introduction to Web Push Notifications in JavaScript. https://itnext.io/an-introduction-to-web-push-notifications-a701783917ce, Apr. 2021. Navštíveno dne 27. 4. 2021.

[41] T. O. I. S. Team. OWASP Internet of Things Project - OWASP. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project, 2018. Navštíveno dne 7. 5. 2021.

[42] P. Viorel. A Guide for selecting the right microcontroller for your IoT project. https://iiot-world.com/industrial-iot/connected-industry/a-guide-for-selecting-the-right-microcontroller-for-your-iot-project/, únor 2018. Navštíveno dne 27. 4. 2021.

[43] Vokolo. Senzor vlhkosti půdy - kapacitní. https://www.vokolo.cz/senzor-vlhkosti-pudy-kapacitni/. Navštíveno dne 30. 12. 2020.

[44] W3C. Push API. https://www.w3.org/TR/push-api/#sequence-diagram, 2020. Navštíveno dne 15. 5. 2021.

[45] R. Wilson. Understanding the IoT cloud and how it will change things. https://www.electronicsweekly.com/news/understanding-the-iot-cloud-and-how-it-will-change-things-2015-10/, 2015. Navštíveno dne 17. 11. 2020.

[46] R. O. Žára. ondras/arduino. https://github.com/ondras/arduino. Navštíveno dne 9. 1. 2021.

Příloha A

Seznam použitých zkratek

API – Application Programming Interface

BLE – Bluetooth Low Energy

CDN – Content Delivery Network

CI – Continious Integration

CD – Continious Delivery/Deployment

DDoS – Distributed Denial of Service

DX – Developer Experience

GPRS – General Packet Radio Service

HTTP – Hypertext Transfer Protocol

IaaS – Infrastructure as a Service

IoT – Internet of Things

IP – Internet Protocol

ISAE – International Standard on Assurance Engagements

LoRa – Long Range Technology

LTE – Long-Term Evolution

MCU – Microcontroller

MQTT – MQ Telemetry Transport

PaaS – Platform as a Service

PWA – Progressive Web App

RFID – Radio Frequency Identification

SaaS – Software as a Service

SPA – Single Page Application

SSL – Secure Sockets Layer

TSL – Transport Layer Security

UI – User Interface

UID – User Identifier

UX – User Experience

Příloha B

Elektronická příloha

NÁZEV	OBSAH
Sensor/	Zdrojové kódy a ukázka fungování senzoru.
App/	Zdrojové kódy aplikace.
Usability testing/	Data z testování použitelnosti.
Application-walkthrough.mp4	Video z průchodu aplikací.

Tabulka B.1: Přehled přílohy závěrečné práce.