VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MOBILNÍ APLIKACE PRO SPRÁVU ZAŘÍZENÍ V IOT

MOBILE APPLICATION FOR MANAGEMENT OF IOT DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **IVANA COLNÍKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Studentka: **Colníková Ivana**
Program: Informační technologie
Název: **Mobilní aplikace pro správu zařízení v IoT**
Mobile Application for Management of IoT Devices
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte oblast IoT a problematiku zpracování a vizualizace dat ze senzorů chytrých zařízení.
2. Prostudujte principy tvorby přehledových obrazovek typu dashboard a problematiku použitelnosti uživatelských rozhraní a uživatelské přívětivosti (UX).
3. Prostudujte princip tvorby webových prezentací optimalizovaných na různé zařízení. Zaměřte se na progresivní webové aplikace (PWA), jejich použitelnost a technologie pro jejich tvorbu.
4. Proveďte analýzu současného stavu prezentace dat ze senzorů chytrých zařízení získávaných z cloudu firmy Logimic. Definujte nedostatky týkající se zobrazení na mobilních zařízeních.
5. Navrhněte úpravy současných dashboardů, tak aby byla zvýšena jejich použitelnost s ohledem na výsledky analýzy z bodu 4 a bylo je možné přívětivě zobrazit na mobilních zařízeních.
6. Implementujte navržené úpravy formou PWA.
7. Ve spolupráci s firmou Logimic otestujte uživatelskou přívětivost implementovaného řešení. Navrhněte možná rozšíření.

Literatura:

- Greengard, S.: *The Internet of Things*. MIT Press, 2015, ISBN 978-026-2527-736.
- Few, S.: *Information Dashboard Design*: The Effective Visual Communication of Data. Sebastopol, USA: O'Reilly, 2006, ISBN 978-059-6100-162.
- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 9780123750303.
- Interní dokumentace firmy Logimic

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 18. října 2021

Abstrakt

Hlavným výstupom tejto práce je vytvorenie Angular modulov do systému firmy Logimic. Tieto moduly predstavujú jednotlivé elementy mobilnej aplikácie pre správu zariadení v IoT. Užívateľské rozhranie aplikácie ponúka užívateľovi lepší prehľad stavov zariadení a senzorov ako súčasný vzhľad. Jednoduchými krokmi môžu poslať servisnú požiadavku na senzor či zariadenie, ktoré je v poruchovom stave. V tejto práci bola prevedená analýza súčasného stavu zobrazenia senzorov, na základe ktorej boli vytvorené návrhy na zlepšenie súčasných nedostatkov vo vzhľade. Implementácia je prevedená pomocou webového rámca Angular s využitím knižnice PrimeNG. Práca zahŕňa testovanie aj samotnú integráciu vytvorených modulov do súčasného systému firmy Logimic.

Abstract

The main goal of this thesis is the creation of Angular modules for the system of Logimic company. These modules represent the elements of the system's mobile user interface and aim to provide an improved user experience in terms of the system's interface and usability. The system provides an overview of sensor states with the ability to easily send service requests. In this thesis I have analyzed the system's current state and proposed suggestions for improvements. The implementation of these improvements were done by using Angular framework and PrimeNG library. The thesis also contains the testing and integration of these modules into the current system of Logimic company.

Kľúčové slová

Angular, PrimeNG, TypeScript, PWA, UX, webová aplikácia, responzivita

Keywords

Angular, PrimeNG, TypeScript PWA, UX, web application, responsivity

Citácia

COLNÍKOVÁ, Ivana. *Mobilní aplikace pro správu zařízení v IoT*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek, Ph.D.

Mobilní aplikace pro správu zařízení v IoT

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Jiřího Hynka, Ph.D. Ďalšie informácie mi poskytol pán Ing. Michal Valný, Ph.D. z firmy Logimic. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Ivana Colníková
9. mája 2022

Poďakovanie

Rada by som sa poďakovala môjmu vedúcemu práce pánovi Ing. Jiřímu Hynkovi, Ph.D. za cenné rady, neustálu dostupnosť pri riešení problémov, ochotu a trpezlivosť. Rovnako by som sa rada poďakovala aj pánovi Ing. Michalovi Valnému, Ph.D. z firmy Logimic, za možnosť spolupráce, cenné rady a čas, ktorý mi na konzultáciach venoval.

Obsah

Kapitola 1

Úvod

Responzivita je čoraz dôležitejšou požiadavkou od klientov pri vytváraní dizajnu. Všetci si uvedomujú, že ľudia trávia na internete čoraz viac času prostredníctvom mobilných zariadení. Tomu sa musia vedieť prispôsobiť aj jednotlivé stránky či aplikácie.

Pre dizajnéra môže byť výzvou z funkčnej aplikácie či webu spraviť responzívnu aplikáciu, ktorá bude spĺňať všetky požiadavky od užívateľa. Častokrát treba zohľadniť, aké dáta zobrazíme užívateľovi a hlavne, ako tieto dáta budeme zobrazovať. Musíme mať vždy na pamäti, že celý návrh musí byť intuitívny, jednoduchý na použitie a hlavne funkčný.

V tejto práci prevediem analýzu a prinesiem návrhy zlepšenia pre webovú aplikáciu firmy Logimic[1], ktorá využíva dashboard na zobrazenie dát z rôznych senzorov a IoT zariadení. Svojim užívateľom ponúkajú prehľad jednotlivých zariadení pomocou vizuálnych prvkov, informujú užívateľa o stave daného zariadenia a o potrebe opravy alebo údržby pomocou farebného indikátora. V mojich návrhoch sa zameriam hlavne na dizajn dashboardu pre mobilné zariadenia. Po sfinalizovaní návrhu začnem tvorbu progresívnej webovej aplikácie, ktorá bude posielať push notifikácie pri zmene stavu zariadenia. Užívateľ bude oboznámený o poruche aj bez otvorenia si samotného dashboardu, čo mu môže priniesť niekoľko výhod, ktoré si popíšeme neskôr v práci.

Kapitola 2 zahrňuje základné vysvetlenie čo sa rozumie pod skratkou IoT, a jeho význam pri návrhu dizajnu dashboardu, kapitola 3 obsahuje päť krokov z UX[2] dizajnu. O každom kroku popíšem, prečo je dôležitý a ako mi pomohol pri vypracovaní návrhov pre náš dizajn. Priblížim v ňom aj využitie už spomínaného dashboardu a na aké prvky si pri jeho dizajne treba dať pozor. Kapitola 4 sa venuje PWA, progresívnej webovej aplikácie, vysvetlím čo to je, aké má výhody a popíšem, ktoré z jej funkcií použijem pri svojej implementácii. Kapitola 5 je zameraná na analýzu súčasného stavu dashboardov firmy Logimic a zároveň bude popisovať aktuálne nedostatky pri zobrazení na mobilných telefónoch. V 6 kapitole budú návrhy riešenia ako zlepšiť aktuálny stav dashboardu z pohľadu UX dizajnu. Kapitola 7 obsahuje implementáciu schváleného návrhu, opis knižníc a rámcov, ktoré som pri práci použila, spomenuté sú aj časti, ktoré boli pre mňa najzložitejšie a opis ako som ich vyriešila. Posledná kapitola 8 zahrňuje testovanie, pri ktorom sa vždy nájdu prvky, ktoré treba zlepšiť, aby bol výsledný produkt čo najprístupnejší pre užívateľov. Každý jeden užívateľ, ktorý sa testovania zúčastnil, mi pomohol moju prácu zlepšiť a vnukol mi nápady na prípadné ďalšie rozšírenie práce.

[1] https://www.logimic.com/

[2] V práci používam skratku UX – predstavujúcu užívateľskú skúsenosť. Preložené z pojmu *User Experience*.

Kapitola 2

Internet vecí

Pojem Internet vecí[1] bol prvýkrát vyslovený Kevinom Ashtonom v roku 1999 pri prezentácii, kde tento pojem použil ako názov jednej z jeho sekcií. Zahrňovalo to predstavu o bežných predmetoch, ktoré budú schopné komunikovať a zdieľať dáta o sebe bez ľudskej interakcie. V dnešných časoch už nejde len o predstavu, ale o neustále sa rozširujúcu infraštruktúru prepojených zariadení [4].

IoT je možné definovať ako sieť fyzických objektov so vstavanou elektronikou, softvérom, senzormi a pripojením na internet. Tieto predmety zbierajú, spracovávajú a vymieňajú informácie medzi sebou [17]. Medzi takéto zariadenia patria teplomery, termostaty, bezpečnostné zámky, osvetlenie, dopravné kamery, radary a veľa ďalších.

IoT nám prináša nespočetné množstvo benefitov do každodenného života. Prostredníctvom internetového pripojenia alebo pomocou technológie bluetooth je dnes možné prepojiť bežné veci okolo nás. Získavame tým väčší prehľad o informáciách a o stavoch daných predmetov. V domácnosti nám dáva možnosť si nastaviť teplotu vykurovania, prináša informácie o stave a intenzite osvetlenia a všetkých iných objektov v rámci konceptu inteligentného bývania. Prispieva aj konceptu inteligentných miest čo umožňuje mať prehľad o stave verejného osvetlenia, spotrebe vody a komunálnych služieb. Obrovský vplyv to má aj v ostatných odvetviach [13].

2.1 Architektúra IoT

Architektúra pre spravovanie a prevádzku IoT zariadení by sa mala vyznačovať s nasledujúcimi vlastnosťami podľa zdroja [20]:

- Škálovateľnosť – musí zvládať spravovať čoraz viac zariadení bez poklesu výkonu.
- Interopereabilnosť – zariadenia od rôznych výrobcov musia vedieť spolupracovať.
- Distributívnosť – dáta zozbierané od rôznych zdrojov by mali byť distribuované a spracovávané rôznymi systémami.
- Šetrnosť – systém by mal využívať čo najmenej zdrojov, pretože zariadenia v nej majú väčšinou len malý výpočetný výkon.
- Bezpečnosť – zabránenie nepovolenému prístupu k systému a následne k dátam

[1] V práci budem ďalej používať skratu IoT – Internet of Things.

Napriek mnohým snahám štandardizácie, v súčasnosti neexistuje pre IoT jednotný referenčný model architektúry. Problémom je veľké množstvo rôznych oblastí aplikácie IoT a snaha výrobcov presadiť svoj vlastný systém na každý z nich. Každá z oblastí zahŕňa vlastné výzvy a riešenia.

Existuje niekoľko architektúr, ktoré boli navrhnuté a postupne rozširované. Spomenuté budú len dve z nich, a to trojvrstvová architektúra a päťvrstvová architektúra. Všetky informácie o architektúre boli čerpané pomocou zdroja [20].

2.1.1 Trojvrstvová architektúra

Ide o všeobecnú vysokoúrovňovú architektúru pozostávajúcu z troch vrstiev. Informácie podľa zdroja [20]:

- **Vrstva vnímania**[2] – táto vrstva reprezentuje fyzickú úroveň zariadení, ich interakciu s okolitým prostredím, zbieraním a spracovávaním informácií z nich. Na tejto úrovni nájdeme objekty, ktoré sú schopné komunikovať s vonkajším svetom, sú vybavené istou výpočetnou kapacitou a tým sa z nich stávajú inteligentné či chytré zariadenia. Inteligentné zariadenie odkazuje na schopnosti auto–identifikácie či auto–diagnostike, kým chytré zariadenie referuje na použitú technológiu.

  Tieto chytré objekty, ktoré predstavujú základné stavebné bloky IoT, stretávame v našom každodennom živote. Príkladom sú napríklad chladnička, televízor, auto, svetlá či iné zariadenia vybavené senzormi a výpočtovou technikou.

  Objekty musia byť schopné jednoznačne sa identifikovať, komunikovať medzi sebou, pripojiť sa a využívať zdroje dostupné na internete, aktualizovať svoj stav a spolupracovať s ostatnými zariadeniami.

  V závislosti od oblasti aplikácie by mali byť objekty adresovateľné a vzdialene konfigurovateľné, lokalizovateľné, môžu mať schopnosť zbierať dáta za pomoci senzorov alebo ovládať ich stav za pomoci akčných členov, na základe údajov získaných zo senzorov a v niektorých prípadoch môžu komunikovať s užívateľom pomocou displeja alebo iným zariadením.

- **Sieťová vrstva** – reprezentuje komunikačnú vrstvu systému, ktorá je zodpovedná za prenos dát poskytnutých predošlou vrstvou do splikačnej vrstvy cez rôzne technológie a protokoly. Na tejto vrstve sú dominantné bezdrôtové technológie, ktoré umožňujú jednoducho rozšíriť vrstvu o nové zariadenia alebo zmeniť polohu už existujúcich zariadení.

  Kvôli rôznorodosti implementovaných technológii zariadení na fyzickej vrstve môže byť použitá sieťová brána. Sieťová brána zoskupuje dáta zozbierané a posielané zariadeniami a senzormi cez rozličné technológie a posiela ich na ďalšie spracovanie pod štandardným TCP/IP protokolom. Brány môžu slúžiť aj ako vyrovnávacia pamäť na dočasné ukladanie dát, na obnovu dát pri výpadku a v niektorých situáciach môžu zbierané dáta prepracovávať pred odoslaním.

- **Aplikačná vrstva** – na tejto vrstve sa za pomoci databáz a iných softvérov ukladajú, filtrujú, analyzujú a spracovávajú dáta z predošlých vrstiev. Formát spracovávaných

[2] Pojem vychádza z anglického názvu *Perception Layer*

dát môže byť binárnej alebo textovej forme. V tejto časti sú dáta poskytnuté na zobrazenie koncovým zariadeniam pomocou tzv. Middleware softvéru, ktorý umožňuje spracovanie rozličných typov dát z predchádzajúcich vrstiev.

V súčasnosti používané technológie sú Cloud a Edge computing. Čo sa týka prvého spomenutého, Cloud computing, ide o úložisko a spracovanie dát poskytnuté ako služba. Naopak Edge computing je časť spracovania dát, ktorá je rozdistribuovaná medzi zariadenia v sieti.

2.1.2 Päťvrstvová architektúra

Niekedy nazývaná aj Middleware architektúra, rozširuje trojúrovňovú architektúru o ďalšie dve úrovne Middleware vrstva a Biznis vrstva [20].

Middleware vrstva slúži pre zaistenie lepšej škálovateľnosti, spoľahlivosti a kvalite poskytnutých služieb. Pomáha integrovať nové aj staršie technológie, a tým pomáha aj uľahčiť vývoj aplikácií nasledovnými spôsobmi spomenutými v zdroji [20]:

- podpora pre rôzne aplikácie, s možnosťou spustenia na rôznych platformách,
- distribúciou výpočetnej kapacity medzi zariadeniami v sieti,
- podpora štandardných protokolov,
- ponúka vysokoúrovňové štandardné rozhranie, umožňujúce spoluprácu rôznych protokolov.

2.1.3 Architektúra na cloude

Príkladom IoT architektúry na cloude je napríklad AWS (Amazon Web Services). Umožňuje chytrým zariadeniam jednoduché a bezpečné pripojenie do cloudu, dokonca aj v offline režime. Využiť sa dajú aj ostatné služby AWS cloudu ako je databáza DynamoDB alebo úložisko Amazon S3.

Architektúra AWS používa sieťovú bránu pre prepojenie IoT zariadení ku cloudu, kde môžu využiť protokol MQTT na komunikáciu. MQTT ponúka výbornú odolnosť proti chybám, využíva málo pamäte, dokáže efektívne využiť dostupnú šírku pásma a dovoľuje komunikáciu jedna k mnohým. MQTT správy sú posielané cez TLS (Transport Layer Security) čím je pokrytá aj bezpečnostná stránka komunikácie.

AWS IoT cloud ponúka radu ďalších bezpečnostných funkcií ako napríklad plne šifrované privátne dáta, autorizáciu či kontrolu prístupu. Informácie boli čerpané zo zdroja [3].

2.2 Komunikačné technológie v IoT

IoT zahŕňa široké spektrum zariadení a technológií. Aby mohol byť nasadený kedykoľvek a kdekoľvek, objekty musia využívať bezdrôtové technológie pre pripojenie a komunikáciu. Každá jedna technológia je vhodná na nasadenie za iných podmienok [27]. Medzi hlavné technológie na vybraných vrstvách patria:

2.2.1 IEEE 802.15.4 PHY

Technológia slúžiaca pre bezdrôtovú komunikáciu [27]. Protokol definuje činnosť na fyzickej vrstve pre siete krátkeho dosahu (Low Rate Wireless Personal Area Networks – LR-WPAN). Využívaný je protokolmi vyšších vrstiev ako je ZigBee, TSMP alebo 6LoWPAN [25].

2.2.2 Wi–Fi

Wi–Fi (Wireless Fidelity) je technológiou, ktorá sa radí medzi bezdrôtové siete. Umožňuje bezpečné prepojenie zariadení v lokálnej sieti a vychádza zo štandardu IEEE 802.11b [2]. Predstavuje základ pre vývoj technológií potrebných pre vznik IoT.

2.2.3 RFID

RFID (Radio Frequency IDentification) popisuje spôsob ukladania a čítania dát pomocou rádiofrekvenčných vĺn [25]. RFID čipy sa dajú klasifikovať do dvoch kategórií aktívne a pasívne. V IoT sú využívané pasívne druhy čipov, kvôli ich dlhej životnosti a malej veľkosti. Pasívny čip sa skladá z antény, polovodičového čipu pripojeného k anténe a obalu aby bol chránený pred vonkajšími vplyvmi. Sú aktivované pri položení čítačky [1]. Objekt vybavený takýmto čipom je možné jednoznačne identifikovať [25].

2.2.4 ZigBee

Bezdrôtová technológia na vytváranie osobných sietí (Personal Area Networks –PAN) pre zariadenia s nízkym výkonom a malou prenosovou rýchlosťou. Rozširuje protokol IEEE 802.15.4 o sieťovú vrstvu a poskytuje možnosť šifrovaného prepojenia zariadení od rôznych výrobcov, pričom ostáva zachovaná kompatibilita aj jednoduché ovládanie. ZigBee je jednoducho ovládateľné počítačom, mobilom či tabletom. Medzi jeho výhody patrí využitie na automatizáciu domácností, šetrí batériu zariadení, vyznačuje sa spoľahlivosťou a najmä bezpečnosťou [29].

2.2.5 Bluetooth

Predstavuje jednu z kľúčových bezdrôtových technológií pre IoT. V rámci osobnej siete podporuje súčasné pripojenie až 255 rôznym koncovým zariadeniam. Bezdrôtová komunikácia, slúžiaca predovšetkým na prepojenie nositeľných technológií ale aj zariadení ako sú tlačiarne, počítačové myšky, mobilné zariadenia, inteligentné hodinky a mnohé ďalšie [27]. Najnovším prínosom vo vývoji tejto technológií je Bluetooth Low Energy (BLE) či Bluetooth Smart technológia.

2.3 Bezpečnosť

Bezpečnosť je dôležitá pre každého v akejkoľvek technológii a so zvyšujúcim množstvom chytrých zariadení je táto oblasť IoT čoraz viac dôležitá. Rôznorodosť komunikačných protokolov a chýbajúce štandardy, vytvárajú bezpečnostné riziká na každej vrstve IoT systému. IoT zariadenia sú častým terčom veľkého množstva útokov a slúžia ako vstupný bod pre útočníka do ďalších častí siete. V celej sekcii je čerpané zo zdroja [11].

V roku 2016 bol zaznamenaný DDoS útok pomocou botnetov, ktoré reprezentovali veľké množstvo IoT zariadení, ako sú tlačiarne, kamery, sieťové brány či detské monitory. V tom roku bola objavená aj bezpečnostná diera v komunikačnom protokole ZigBee, ktorá umožnila pomocou dronov ovládať sadu inteligentných svetiel.

Okrem bezpečnosti z hľadiska technológie je taktiež dôležité aby dáta, ktoré zariadenia zbierajú ostali v súkromí.

2.3.1 Možné útoky na fyzickej vrstve

- Útočník môže fyzicky manipulovať so zariadením, modifikovať jeho softvér alebo inak poškodiť zariadenia.

2.3.2 Možné útoky, ktoré narušia fyzickú zároveň aj sieťovú vrstvu

- Odmietnutie služby – útočník môže zneužiť nízky výkon zariadenia a spraviť ich nedostupnými.
- Smerovacie útoky – modifikácia smerovacích tabuliek pri zbieraní a preposielaní dát.
- Útok pri prenose dát – rôzne útoky, ktoré porušia integritu dát.

2.3.3 Možné útoky na aplikačnej vrstve

- Krádež a únik dát – ak útočník pozná bezpečnostné diery v ponúkanej službe či v aplikácii.

Rozdielom medzi tradičnými zariadeniami a IoT zariadeniami je, že sa na nich nedá aplikovať pridaná bezpečnosť. Zariadenia sú po výrobe uzavreté a musia byť zabezpečené od výroby. Mnohí výrobcovia však kvôli rýchlemu vývoju a za účelom zníženia nákladov na výrobu, nechávajú bezpečnosť na poslednej priečke. Zariadenia väčšinou disponujú iba ochranou na softvérovej úrovni, nechávajúc miesto pre útoky na hardvér zariadenia.

Najväčšie bezpečnostné riziko predstavuje fyzická vrstva, kde kvôli hardvérovým limitáciam, rôznorodosti použitých technológií a nevyspytateľnému prostrediu, do ktorého sa zariadenie nasadí je tažké nájsť efektívne bezpečnostné opatrenia. Transportná vrstva predstavuje najmenšie riziko kvôli neustálemu výskumu a vylepšovaniu bezdrôtových technológií podobne ako aplikačná vrstva, kde sa dajú nájsť protokoly, ako je MQTT alebo CoAP.

2.4 Oblasti aplikácie IoT

IoT je rýchlo sa rozvíjajúca oblasť informačných technológií, čo má za dôvod aj neustále sa rozširujúce možnosti aplikácie v rôznych odvetviach. V súčasnosti by sme aplikáciu IoT mohli rozdeliť do niekoľkých oblastí. V mojej práci budú spomenuté iba niektoré zo zdroja [5]:

2.4.1 Priemysel a agrokultúra

Využitie IoT v oblasti priemyslu a agrokultúry pokrýva širokú škálu možností od logistiky cez výrobu až po monitorovanie procesov či služieb. Relevantný príklad aplikácie sa dá nájsť v sektore logistiky a monitorovania skladu. Identifikácia produktov prebieha pomocou RFID čipov nasadených v jednotlivých výrobkoch. Umožňuje to sledovať stav produktu počas celého výrobného procesu, skladovania až po samotné zakúpenie zákazníkom. Pri skladovaní by bol viditeľný prehľad či bol pre daný produkt zaistený správny postup uskladnenia. V prípade zvýšenej vlhkosti či iného faktoru by bolo zaistené automatické napravenie alebo zaslanie upozornenia, ktoré by sme videli na dashboarde. Týmto spôsobom sa uľahčuje výroba a sledovanie celého procesu. Tým, že sú prístupné informácie o aktuálnom stave produktu v reálnom čase, dáva možnosť zaistiť kvalitnejší výrobný proces, skladovanie a spravovanie produktov. Produkty s nasadeným RFID čipom garantujú svoju integritu [5].

2.4.2 Zdravotníctvo

V zdravotníckom sektore sa pomocou pokročilých senzorov môže sledovať a udržiavať zdravotný stav pacientov bez potreby zasahovania lekárom. Týmto sa uľahčuje strostlivosť o pacienta a zároveň zvyšujeme jeho komfort [5].

Pri akejkoľvek nehode pomocou mobilného zariadenia, ktoré disponuje s GPS lokáciou, je možné zistiť polohu, čo pomáha rýchlejšie poskytnúť alebo privolať pomoc.

2.4.3 Inteligentné mestá

Inteligenté mesto predstavuje koncept, ktorého cieľom je pomocou informačných a komunikačných technológií vytvoriť komplexnú infraštruktúru prepojených zariadení a objektov. Tento systém slúži na vylepšenie poskytovaných služieb mestom pre obyvateľov v rôznych oblastiach, ako je napríklad monitorovanie objektov alebo celková správa mesta [22].

IoT by sa mohlo využívať v inteligentných mestách a priniesol by nasledujúce výhody. Znížil by sa hluk v meste pomocou monitorovania a mapovania v reálnom čase, sledoval by sa stav budov, mostov a iných stavieb pre zvýšenie udržiavateľnosti. Analýza dopravy a predchádzanie dopravným zápcham by zlepšilo dopravnú infraštruktúru mesta, centrálne riadené kamerové systémy pre zvýšenie bezpečnosti, inteligentné osvetlenie prispôsobené počasiu, ktoré by šetrilo energiu a náklady. Občanom by bola poskytnutá kontrola kvality vody a vzduchu [23].

Mestá pomocou týchto technológií dokážu občanom ponúknuť lepšiu dopravnú infraštruktúru, zlepšiť kvalitu komunálnych služieb, zvýšiť bezpečnosť mesta a optimalizovať školské či súdne systémy.

2.4.4 Vizualizácia dát zo senzorov

So zobrazením dát zo senzorov IoT zariadení je úzko spojená aj grafická vizualizácia. Dashboard je najčastejším typom grafického užívateľského rozhrania, ktorý sa používa na zobrazenie dát. Vďaka dobrému dizajnu je čitateľný aj pre ľudí, ktorý nepatria medzi technické typy. To je dôvod, prečo je veľmi dôležitý správny výber zobrazenia informácií pre užívateľa.

Dáta sa často zobrazujú v podobe rôznych diagramov a grafov, vychádzajúce z databázy. Do databázy sú posielané zariadeniami v časových intervaloch, kde sa zhromažďujú a aktualizujú. Následne sa dáta posielajú na koncové zariadenie, kde sú zobrazené a prístupné užívateľovi.

Kapitola 3

Dashboard

V tejto kapitole si bližšie vysvetlíme čo je dashboard a jeho históriu. Ďalej bude popísané, aké informácie sa na ňom zobrazujú, prečo obsahuje viac vizuálnych prvkov ako textu a ako súvisí s UX dizajnom. Spomenuté budú aj najdôležitejšie body, na ktoré treba myslieť pri návrhu dashboardu, aby nebol len vizuálne zaujímavý, ale hlavne nech užívateľovi ponúkne čo najlepšiu užívateľskú skúsenosť. Vysvetlený bude rozdiel medzi UX a UI[1], a popísaný každý krok UX procesu.

3.1 Pojem dashboard

Formálnejšiu definíciu zaviedol napríklad Stephen Few [10]:

Definition 3.1.1. A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.

Dashboard je jedným z najpoužívanejších grafických užívateľských rozhraní, ktoré sa využívajú na zobrazenie kľúčových ukazovateľov výkonnosti, dát a informácií. Využívajú sa na to rôzne kombinácie prvkov, ako sú napríklad texty, grafy, diagramy, widgety. Ide o vizuálnu prezentáciu dát, ktorá musí spĺňať pravidlá užívateľskej prívetivosti UX. Zároveň sa cez dashboard dostávame na rôzne úrovne zobrazení [6].

Kľúčové ukazovatele výkonnosti[2] predstavujú indikátory, merajúce výkon procesov na zariadeniach, výkonnosť jednotlivých zamestnancov alebo organizácie ako celku [9].

3.1.1 Charakteristika

Aby sme predišli nesprávnemu návrhu, v práci som sa riadila nasledujúcimi bodmi. Popisujú načo si dávať pozor pri návrhu dashboardu a čo musí spĺňať. Celá táto podkapitola je prevažne čerpaná zo zdroja [10].

- Dashboard má zobrazovať informácie potrebné na dosiahnutie špecifikovanej úlohy – aby sa to dosiahlo je potrebné agregovať informácie z rôznych zdrojov a poskytnúť užívateľovi iba tie, ktoré sú preňho potrebné, najčastejšie formou KPI.
- Musí predstavovať jednu obrazovku – čo znamená, že informácie sa musia zmestiť na jednu obrazovku, aby užívateľ nemusel informácie hľadať ani skrolovať.

[1] Skratka UI predstavuje užívateľské rozhranie. Preložené z pojmu *User Interface*.
[2] Ďalej v práci budem používať skratku KPI.

- Jednoducho čitateľné informácie – užívateľ už pri rýchlom pohľade musí zistiť všetky potrebné informácie.
- Spôsob vizualizácie by mal odpovedať požadovaným dátam – informácie musia byť zobrazené tak, aby zaberali čo najmenej miesta s využitím ikoniek a farebných indikátorov pre lepšiu čitateľnosť.
- Prispôsobiteľný – v závislosti na skupine používateľov musí poskytnúť relevantné dáta.

Z týchto bodov si môžeme zhrnúť vlastnosti, ktoré musí dashboard spĺňať a nimi sú: **prehľadnosť, jednoduchosť, použiteľnosť** a **čitateľnosť.**

3.2 História

Aj dashboard má svoju históriu. V roku 1980 bol nazývaný *Executive Information Systems(EISs)* a predstavoval podobný koncept ako dnes. Cieľovou skupinou užívateľov boli vtedy tí, ktorí vedeli pracovať s počítačmi na vysokej úrovni [10]. Využíval sa najmä na zobrazovanie dát o firme, jej stave a projektoch. Tieto zobrazenia neboli prispôsobené bežným ľuďom, boli ťažko čitateľné a interaktivita s nimi nebola intuitívna.

V 90. rokoch sa prvýkrát stretávame s KPI, ktoré sú populárne dodnes [10]. Avšak dnes sa dashboardy využívajú aj pre ľudí, ktorí nemajú veľa skúseností s používaním počítača. Zameriavajú sa na jednoduchosť a efektívnosť zobrazenia informácií.

3.3 Delenie dashboardov

Dashboardy sa môžu používať na zobrazenie rôznych typov dát. To je dôvodom, prečo existuje viacero kategórií podľa premenných, ktoré majú byť na danom dashboarde zobrazené. Všetky informácie v tejto sekcii sú čerpané zo zdroja [10].

3.3.1 Klasifikácia podľa účelu

Najdôležitejšie je rozdelenie, ktoré je určené podľa obchodnej činnosti. V tejto kategórií vidíme najväčšie rozdieli po vizuálnej stránke. Ďalej ich vieme rozdeliť na:

- Strategické – najrozšírenejší typ, ktorí využívajú manažéri v organizáciach, poskytujú rýchly prehľad informácií potrebných na rozhodovanie o posunoch v projektoch. Zameriavajú sa na meranie výkonnosti na vysokej úrovni, ktoré môžu zahrňovať podľa zozbieraných dát aj predikcie o vývoji do budúcnosti. Do tohto typu sa hodia jednoduché vizuálne prvky, ktoré majú za úlohu ukazovať dáta zozbierané za určitú časovú dobu. Neobsahujú interaktívne prvky, ide čisto len o prehľad relevantných informácií.
- Analytické – vyžadujú si väčší kontext k dátam ako pri predchádzajúcom type. Môže byť vo forme porovnávaní už nameraných dát alebo dlhšej histórií zobrazených dát. Musí poskytovať interakciu s jednotlivými prvkami, aby si ich mohol užívateľ rozkliknúť a vidieť podrobnejší popis. Napríklad, nestačí vidieť, že klesá teplota, ale musí byť užívateľovi jasné, aký je dôvod, že k tejto poruche prišlo.
- Operačné – poskytujú užívateľovi prehľad dát v reálnom čase. Dôležité je, že pri zmene niektorej z hodnôt do kritickej, je možné jasne vidieť a identifikovať, čo si vyžaduje pozornosť alebo až údržbu. Tento typ musí zahrňovať interaktívne prvky, cez ktoré

vieme získať detailnejšie dáta potrebné na lokalizáciu chyby alebo zmeny hodnoty, ktorá vznikla. Dôraz sa kladie na jednoduchý vzhľad a na výrazné označenie chýb.

3.4 Dáta v dashboarde

Dashboard poskytuje väčšinou kvantitatívne dáta vo forme sumárov, priemerov, štandardných odchýlok a môžu byť zobrazené vo vyššie spomenutých typoch dashboardov. Musia byť jednoznačne odkomunikované tak, aby sa prehľad zmestil na jednu obrazovku a zobrazenie dát odpovedalo otázke "čo sa práve deje?". V závislosti na čase môžeme na dashboardoch zobrazovať dáta v rôznych intervaloch.

Informácie získané z dát môžeme obohatiť porovnávaním. Príkladom by mohlo byť následné zobrazenie grafu, kde budeme mať prehľad nameraných hodnôt teploty za posledný týždeň.

Ďalej ich môžeme obohatiť vyhodnotením a spracovaním, ktorého výstupom je jednoduchá indikácia toho, či je to, čo meriame v dobrom alebo v zlom stave. Dizajn by mal jasne upozorniť užívateľa ak by bolo niečo zlé. Všetky informácie v tejto sekcii sú citované zo zdroja [10].

3.5 Návrh dashboardu

Väčšina z navrhnutých dashboardov má jedinečný vzhľad, nakoľko je ťažké spraviť dizajn, ktorý by bol vhodný pre viacero rozdielnych systémov. Každý systém má iné požiadavky na dáta, ktoré je potrebné zobraziť. Každý dashboard sa zameriava na iný typ koncových užívateľov [6].

Vzhľadom na rýchlo sa rozvíjajúcu dobu je dostupných množstvo týchto prvkov, no nie všetky sú užívateľsky prívetivé. Pri vybratí nesprávneho vzhľadu sa veľmi rýchlo môže stať, že náš koncový užívateľ nepochopí zobrazenie vybraných KPI.

Každý návrh dizajnu sa musí opierať o základné kroky, ktoré vyplývajú z UX dizajn procesu. Na začiatok je vysvetlený rozdiel medzi pojmom UX a UI, ktoré spolu úzko súvisia.

3.5.1 Rozdiel medzi UX a UI

Pri návrhu dizajnu sa často stretneme so skratkami UX a UI. Rozdielom je, že UI – *User Interface* popisuje užívateľské rozhranie. Touto skratkou sa označuje časť dizajnu, ktorá sa venuje rozvoju rozhrania systému, zaoberá sa výberom farebnej palety, typografiou, vizuálnym dizajnom či rozložením jednotlivých prvkov.

Kým UX – *User Experience* v preklade užívateľská skúsenosť je časť dizajnu, ktorá sa zameriava na užívateľa, jeho pocity, zážitky pri interakcii s daným produktom. UX sa zameriava na skúmanie cieľovej skupiny užívateľov, tvorbou náčrtov, prototypov a wireframov [16]. Viaže sa s ním aj pojem užívateľsky orientovaný dizajn – *User–Centered design*, je to iteratívny proces v rámci dizajnu produktu zameraný na užívateľa a jeho potreby.

Dalo by sa povedať, že UX odpovedá na otázku „Ako sa daný produkt používa?“ a UI odpovedá na otázku „Ako daný produkt vyzerá?“. Pri procese návrhu správneho dizajnu sa využíva ako UX, tak aj UI. Tieto časti spolu vytvárajú harmóniu medzi funkčnosťou a vizuálne pekným prevedením.

3.5.2 5 elementov UX dizajnu

Predstavuje to štruktúru krokov, ktorá umožňuje UX dizajnérovi premeniť svoj nápad na hotový produkt. 5 elementov si môžeme predstaviť ako vrstvy, kde každá vrstva je závislá na tej predošlej. Všetky informácie v tejto kapitole sú zo zdroja [12].

- **Stratégia** – najspodnejšia vrstva, predstavujúca ciele dizajnu, založené na potrebách užívateľa.
- **Rozsah** – v tejto vrstve sa určuje typ produktu, ktorý sa ide vytvárať. Definuje sa funkcionalita a obsah daného produktu.
- **Štruktúra** – rozhoduje sa o rozložení dizajnu a ako chceme aby užívateľ pracoval s výsledným produktom.
- **Kostra** – predstavuje rozloženie produktu a jednotlivých jeho elementov. Táto časť už nie je viditeľná pre užívateľa.
- **Povrch** – reprezentuje grafické užívateľské rozhranie, s ktorým užívateľ pracuje.

3.5.3 Proces vytvorenia dizajnu

V tejto podkapitole sú predstavené jednotlivé kroky a úlohy, ktoré k nim prislúchajú a budem ich využívať pri samotnom návrhu dizajnu. Všetky informácie o *Design Thinking Process* sú prevažne čerpané [26].

Pochopenie užívateľa

Prvotným krokom je pochopenie užívateľa alebo cieľovej skupiny, ktorá bude daný produkt používať. V tejto fáze je dôležité pochopiť, ako má daný produkt užívateľ používať a v akých situáciach ho bude vyhľadávať. Prebieha tu množstvo stretnutí, rozhovorov a sledovaní užívateľa pri rôznych procesoch, ktoré si výskumník zapisuje aby ich vedel využiť pri návrhu. Kým nepochopíme užívateľa, nebudeme vedieť spraviť ani vhodný návrh. Cieľom tejto fáze je zozbierať požiadavky na daný produkt a porozumieť užívateľovi.

Stanovenie cieľov

Z predchádzajúcej fáze sme sa naučili porozumieť a lepšie pochopiť užívateľa, čo je potrebné pre tento krok. Tu si výskumníci stanovujú jasnejšie ciele a zameranie, ktoré je potrebné dosiahnuť. Nakoľko výskumníci majú viacero rozhovorov s užívateľmi za sebou a vďaka empatie im lepšie rozumejú a poznajú aj ich prostredie. Tým vedia lepšie definovať problémy a vyriešiť ich na základe poznatkov o svojich užívateľoch. Fáza stanovenia cieľov je ukončená jasne stanovenými požiadavkami, rozsahom a problémami, na ktoré hľadáme riešenie.

V tejto fáze je možné vytvoriť si pomocný diagram nazývaný aj mapa. Takáto mapa so zameraním sa na užívateľa, predstavuje zmapovanie problému, ktorý je nutné riešiť. V rámci nej sa určujú jednoduché kroky, tvoriace daný problém. Postup vytvorenia mapy spočíva v definícii všetkých typov užívateľov, stanovením krokov, s ktorými užívateľ pri danom probléme prechádza [18].

How Might We je ďalšou metódou využívajúcou sa pri analýze. Je vyvinutá spoločnosťou Procter & Gamble [18]. Slovíčko „How“ pomáha zamyslieť sa nad riešením problému, na ktorý nemáme definitívnu odpoveď. Vzbudzuje akúsi istotu a dáva priestor na riešenia

z viacerých pohľadov. Ďalšie slovíčko „Might“, informuje o tom, že riešenia, ktoré nám napadli nemusia byť finálnymi ani jedinými. Posledné slovo „We“ prikladá váhu k tomu, že sa rozhoduje o riešení v tíme [7]. Postup prevedenia tejto metódy, spočíva napísaním si týchto troch slovíčok pod seba. Následne počas tímovej komunikácie si zapisovať otázky, ktoré by mohli viesť k ďalším riešeniam.

Nápady

V tejto fáze sa zapisujú všetky nápady, ktoré by mohli pomôcť vyriešiť problémy, ktoré boli definované v predchádzajúcej fáze. Výskumníci tu majú priestor na vymýšľanie rôznych nápadov a obmedzovaný sú len vlastnou fantáziou. Tým, že výskumníci „hádžu“ nápady môžeme dospieť k inovatívnejším nápadom. Táto fáza zahrňuje rôzne inovačné techniky, skeče, využitie Crazy 8´s[3] [18], návrhy prototypov, ktoré prinášajú ďalší uhol pohľadu. Fáza generovania nápadov je ukončená návrhmi, na ktorých sa výskumníci zhodli, že najlepšie riešia problémy, ktoré si stanovili.

Prototyp

V tejto fáze vytvárame návrh na riešenie z predchádzajúcej fáze. Musíme vytvoriť základný interaktívny model reprezentujúci naše riešenie. Čím realistickejšie prototyp vyzerá, tým lepšiu spätnú väzbu dostaneme od užívateľa a budeme lepšie vedieť zapracovať na nedostatkoch. Tieto prototypy prinášajú tímu veľkú výhodu v tom, že im pomáhajú odhaliť nedostatky, iterovať na svojich nápadoch a neustále vylepšovať návrh. Využívajú sa tu programy ako je Figma[4], Adobe Xd[5] či Sketch[6] a ďalšie podobné. Fáza je ukončená prototypom či už papierovým alebo digitálnym.

Testovanie

Testovanie je spôsob ako získať spätnú väzbu na nápady a prototyp, vytvorených v predchádzajúcich fázach. V testovaní sa znova vraciame k empatii a pochopeniu užívateľa, aby sme sa vedeli lepšie vcítiť do toho, ako na nich prototyp pôsobí. Užívateľovi umožníme preklikať si hotový prototyp. Spätnú väzbu môžeme získať pomocou priamych rozhovorov či dotazníka. Následné porovnanie spätnej väzby s pôvodnými požiadavkami, pomáha vyhodnotiť kvalitu nášho riešenia. Spätná väzba z týchto testovaní pomôže pri ďalšej iterácii na prototype a slúži k overeniu, či všetky stanovené problémy boli vhodne vyriešené.

3.6 Časté chyby pri návrhu

Pred tým ako sa pustím do dizajnu dashboardu, je potrebné zhrnutie niekoľkých bodov, ktorým sa budem snažiť pri návrhu vyhýbať. Samozrejme, je ich nespočetné množstvo, a ja sa budem venovať v tejto práci len zlomku z nich. Všetky informácie v tejto sekcii sú citované zo zdroja [10].

- **Dashboard nie je na jednej strane** – ako bolo spemenuté v podkapitole 3.1.1, dashboard sa musí zmestiť na jednu stranu. Ak by sa kvôli množstvu dát muselo v dashboarde skrolovať, odporúča sa radšej potrebné dáta zobraziť na inej stránke.

[3] Crazy 8´s – metóda na získanie 8. napádov behom 8. minút na riešenie problémov
[4] https://www.figma.com/
[5] https://www.adobe.com/products/xd.html
[6] https://www.sketch.com/

- **Neadekvátny kontext k zobrazeným dátam** – pri návrhu dizajnu je potrebné zamyslieť sa nad tým, ako dáta zobrazovať. Je potrebné premyslieť, aké informácie či už v podobe textov alebo porovnaní k nim použijeme. Príkladom môže byť použitie grafu bez dostatočného množstva textu v legende, čo má za následok, že užívateľ nerozumie čo mu daný graf znázorňuje. Ak sa zobrazí málo informácií, dizajn dashboardu môže pôsobiť prázdno.
- **Zobrazenie nepotrebných dát** – tento bod môže súvisieť s predchádzajúcim. Občas je veľmi tenká hranica medzi tým, či nejaké informácie zahrnúť do dashboardu alebo radšej nie. Príliš veľké množstvo informácii môže viesť k nečitateľnosti a veľmi malej prehľadnosti.
- **Zvolenie si nevhodného merítka** – je potrebné zobraziť dáta tak, aby boli pochopiteľné. Nevhodne zvolené merítko môže viesť k chaotickému dizajnu, kedy užívateľ nevie, čo mu napríklad daný graf ukazuje. Môže prísť aj k výpisu chybných hodnôt.
- **Zbytočné pridávanie prvkov** – nie je vhodné pridávať rôzne prvky zobrazenia rovnakých informácií iba kvôli vizuálnemu efektu. Odporúča sa zachovanie konzistencie zobrazenia, pre jednoduchšiu užívateľskú orientáciu.
- **Zlý dizajn jednotlivých prvkov** – príkladom môže byť zle zvolená paleta farieb, čo môže dospieť k zlému kontrastu farieb a nečitateľnosti. Problém môže byť aj pri použití nevhodných farieb v prvkoch a následné zatienenie farieb indikujúcich kritické stavy [15].
- **Zle zorganizované dáta** – môže to byť zle použité biele miesto, čo môže prispieť k veľkej hustote prvkov alebo neefektívnemu využitiu miesta na stránke. K zlej organizácii dát môžu prispieť aj neprimerané veľkosti zobrazených prvkov.
- **Nezvýraznené dôležité údaje** – už pri prvom pohľade na dashboard by mali byť dôležité údaje vyznačené tak, aby si ich užívateľ ihneď všimol. Takéto zobrazenie vedie k nedostatočnej komunikácii informácii medzi zobrazenými dátami a užívateľom, ktorý tieto dáta musí hľadať.

Kapitola 4

Progresívna webová aplikácia

V tejto kapitole je vysvetlené, čo znamená skratka PWA, ako sa líši od klasických webových či natívnych aplikácií, aké sú jej výhody v porovnaní s natívnymi aplikáciami a aké funkcie nám ponúka. Popísané budú aj limitácie, ktoré so sebou prináša. Vysvetlený bude aj pojem *Service Worker*, predstavujúci základnú časť týchto aplikácií. Spomeniem aj niekoľko webových rámcov, využiteľných na implementáciu progresívnej webovej aplikácie.

4.1 Význam PWA

Skratka PWA je v dnešnej dobe čoraz viac používaná. Táto skratka je pre pojem progresívna webová aplikácia. Jedna z technických definícií o PWA je uvedená v článku [19]:

Definition 4.1.1. We define a PWA as a website that registers a service worker at the browser of a page visitor. Because the service worker is a key technical component that enables native-app experiences including offline usage and push notifications, our definition captures all PWAs designed for various purposes.

4.1.1 Charakteristika progresívnych aplikácií

Okrem technickej definície je viacero možností ako definovať PWA. Definovať by sa mohla aj ako koncept webovej aplikácie, ktorej cieľom je poskytnúť užívateľovi čo najlepšiu užívateľskú skúsenosť natívnej aplikácie. Využívaním najnovších technológií dostupných na platformách sa progresívne webové aplikácie postupne rozširujú o ďalšie a ďalšie funkcie. Z webových stránok sa postupne stávajú aplikácie, ktoré ponúkajú funkcionalitu ako sú push–notifikácie či offline režim [28]. PWA predstavujú webové stránky, optimalizované aj na mobilné zariadenia. Interpretované sú prehliadačom a dokážu obmedzene využívať natívne funkcie zariadenia [19].

Medzi základnú charakteristiku progresívnych aplikácií podľa zdroja [28] sa radí:

- **Progresívnosť** – aplikácia funguje nezávisle od polohy užívateľa aj od webového prehliadača, použitého pre prístup k danej stránke. Využíva funkcie poskytnuté zariadením a prehliadačom.
- **Responzívnosť** – užívateľské rozhranie sa vždy prispôsobí k danej veľkosti zariadenia.
- **Nezávislý na pripojení** – aplikácia dokáže fungovať pri horšom pripojení k internetu či offline.

- **Inštalácia** – užívateľ si vie aplikáciu uložiť na svojej domovskej obrazovke bez nutnosti stiahnutia z obchodu Google Play či App Store.
- **Dostupnosť** – progresívne webové aplikácie sa dajú jednoducho nájsť aj cez vyhľadávače ako je napr. Google.
- **Interaktivita** – využitie funkcií ako sú push-notifikácie pre zvýšenie interakcií s aplikáciou.
- **Bezpečnosť** – progresívne webové aplikácie komunikujú výhradne cez protokol HTTPS.
- **Aktuálnosť** – aplikácie ponúkajú vždy aktualizovaný obsah.

4.2 Natívna a webová aplikácia

Pre vývoj mobilných aplikácií máme dnes dostupných viacero alternatív. Pred popisom progresívnej webovej aplikácie najprv bude definovaný pojem ako natívna a webová aplikácia, na základe ktorých PWA vznikla. Všetky nasledujúce informácie v tejto podsekcii sú čerpané zo zdroja [21].

4.3 Charakteristika natívnej a webovej aplikácie

Natívna aplikácia predstavuje aplikáciu, implementovanú v programovacom jazyku a vo vývojovom prostredí špecifikovanom pre dané zariadenie. Pre zariadenia so systémom Android sa väčšinou používa jazyk Java[1] alebo Kotlin[2] spolu s vývojovým prostredím Android Studio[3]. Takéto aplikácie je možné nahrať do obchodu Google Play, odkiaľ si ju vie užívateľ stiahnuť a nainštalovať na svoje zariadenie.

Natívna aplikácia sa vyznačuje priamym využívaním funkcií dostupných na danom zariadení, ako sú napríklad GPS, kamera, zvuk, rôzne senzory či notifikačný systém. Vysoký výkon a široká funkcionalita natívnych aplikácií je za cenu, že sú dostupné len pre daný vyvíjaný systém. V prípade požiadavky na sprístupnenie natívnej Android aplikácie aj pre iOS, je potrebné vyvíjať a udržiavať dve rôzne aplikácie.

Webové aplikácie sú dostupné na webových stránkach. Dnešným štandardom sú responzívne webové stránky vyvíjané pomocou HTML, CSS a JavaScriptu[4]. Tieto aplikácie sú jednoducho dostupné cez ľubovolný webový prehliadač a to nezávisle na zariadení. Na rozdiel od natívnych aplikácií je výhodou udržiavanie iba jedného zdrojového kódu. Naopak nevýhodou je značne limitovaná funkcionalita.

4.3.1 Porovnanie

V porovnaní webovej aplikácie a natívnej aplikácie trávia ľudia viac času používaním natívnych aplikácií. Z tohto dôvodu bol navrhnutý koncept progresívnej webovej aplikácie. Predstavovať má kompromis medzi dostupnosťou a užívateľskou angažovanosťou s aplikáciou. Takéto aplikácie zaberajú menej miesta v pamäti telefónu, nakoľko sú v podstate

[1] https://www.java.com/en/
[2] https://kotlinlang.org/
[3] https://developer.android.com/studio
[4] https://www.javascript.com/

webové stránky. Vždy zaručia aktuálny obsah bez nutnosti čakania na aktualizáciu ako to môžeme vidieť aj v prípade natívnych aplikácií, informácie boli čerpané zo zdroja [28].

Porovnanie rôznych vlastností medzi natívnou aplikáciou, webovou aplikáciou a PWA, na základe textu v [28].

- **Inštalácia** – natívna aplikácia vyžaduje inštaláciu pomocou Google Play či App Store, zatiaľ čo webovú aplikáciu nie je nutné inštalovať. Čo sa týka PWA, inštalácia je možná prostredníctvom prijatia výzvy na stiahnutie. Táto výzva sa vo väčšine prípadov zobrazuje v dolnej časti mobilného zariadenia pri prehliadaní webovej aplikácie. Po stiahnutí a inštalácii je aplikácia dostupná na domovskej stránke mobilného zariadenia v podobe ikonky.

- **Aktualizácie** – pri natívnej aplikácií je nutné nahrať najnovšiu verziu, ktorá však musí byť stiahnutá užívateľom. Naopak PWA či webová aplikácia disponujú okamžitou aktualizáciou.

- **Veľkosť v pamäti** – natívne aplikácie zaberajú pomerne veľké miesto v pamäti, v porovnaní s webovými aplikáciami a PWA.

- **Funkcionalita bez pripojenia na internet** – dostupnosť offline ponúka natívna aplikácia, pri webovej aplikácii to naopak nie je možné. PWA potrebuje byť pri prvom spustení pripojené na internet a následne je dostupná aj bez pripojenia, za pomoci uloženého obsahu.

- **Užívateľská skúsenosť** – užívateľská skúsenosť je veľmi dôležitá, nakoľko ju užívateľ má hneď pri otvorení aplikácie. Pri natívnej aplikácii môžeme dosiahnuť bezproblémovú užívateľskú skúsenosť pomocou správneho návrhu dizajnu. PWA a webová aplikácia môže byť rušivá rozdielnym typom navigácii, nakoľko sa používajú dve. Jedna pre počítačovú verziu zobrazenia a druhá pre mobilné zariadenia.

- **Push–Notifikácie** – ponúka ich natívna aplikácia aj PWA, na rozdiel od toho, že PWA má push–notifikácie iba pre Android zariadenia. Webové aplikácie ich ponúkajú len za pomoci tretích strán.

- **Dostupnosť v prehliadačoch** – webová aplikácia je štandardne dostupná v prehliadačoch, zatiaľ čo natívnu aplikáciu je možné nájsť iba pomocou obchodu Google Play či App Store. PWA je potrebné optimalizovať pre prehliadače.

4.4 Tri základné komponenty PWA

Webové aplikácie využívajúce PWA sú tvorené tromi základnými komponentmi. *Service Workers*, *App Shell* a *Web App Manifest* umožňujúce požadovanú funkcionalitu.

4.4.1 Service Workers

Service Workers sa radí medzi základné stavebné bloky PWA. Pomocou nich dokážeme využívať funkcionalitu natívnych aplikácií vo webových aplikáciách. Patrí mu zodpovednosť za ukladanie dát do medzipamäte pre sprístupnenie aplikácie aj v offline režime. Umožňuje využiť push–notifikácie nezávisle na platforme a aktualizuje obsah aplikácie na pozadí. Tvorí ho súbor vo formáte JavaScript, nachádzajúci sa medzi požiadavkami aplikácie a serverom.

Interpretovaním týchto požiadaviek dokáže napríklad rozhodnúť o tom, či je potrebné načítať nový obsah alebo stačí načítať obsah uložený v medzipamäti [28].

Ďalšie informácie o *Service Workers* sú čerpané zo zdroja [14]. Beží v samostatnom vlákne, je riadený udalosťami a vyznačuje sa plnou asynchrónnosťou a taktiež vymedzeným rozsahom. Vďaka týmto vlastnostiam zaisťuje, že jeho prípadné zlyhanie či vypnutie, nebude mať vplyv na samotnú aplikáciu. Môžeme ho pri vývoji aplikácie pridávať postupne na zvolené časti aplikácie. Väčšinou je uložený v koreňovom adresári, a tým pádom má prístup ku všetkým častiam aplikácie, čo mu dáva možnosť ju celú spravovať. Nemá prístup k DOM[5], XHR požiadavkám[6] ani k lokálnemu úložisku.

Nasadenie a následné využitie prebieha v štyroch fázach. Keď užívateľ navštívi danú PWA web stránku, spúšťa sa prvá fáza. V nej je potrebné service worker najprv stiahnuť, spracovať a spustiť. To všetko zaisťuje funkcia *register()*, ktorá vracia objekt typu *Promise*. Pomocou nej je možné sledovať a vyhodnotiť stav registrácie. Ak by nastala akákoľvek chyba, objekt *Promise* vracia chybu a registrácia je prerušená. Pri úspešnej registrácii, prechádza do fáze druhej, čo reprezentuje inštaláciu. Táto fáza sa vykoná iba pri prvom spustení, využitím funkcie *waitUntil()*. V nej je vhodné uložiť do medzipamäte všetky potrebné statické súbory, ako sú napríklad obrázky. Ďalej sa presúva do tretej fáze, kde je aktivovaný a dostáva kontrolu nad svojím vymedzeným rozsahom, pomocou funkcie *Clients.claim()*. Po úspešnej inštalačnej a aktivačnej fáze je *Service Worker* v stave kedy môže spracovať požiadavky na serveri. Bez neho webová stránka nevie spracovať žiadne požiadavky a teda jeho nasadenie je prvým krokom pri vytváraní PWA aplikácie.

Ponúka tiež aj udalostné funkcie ako je *fetch()* – slúži na vyžiadanie statických prvkov a dynamického obsahu, môže to byť požiadavka na obrázok, video, CSS, HTML či JS súbor. Ďalšou je funkcia *push()* – slúžiaca na spracovanie push správ a zobrazenie notifikácie užívateľovi a funkcia *sync()* – umožňujúca odložiť vykonanie požadovanej akcie, keď je užívateľ offline alebo má slabšie pripojenie. Požiadavka sa vykoná až pri stabilnom pripojení čím sa zaistí spoľahlivé spracovanie požiadavky. Pri aktualizácii je pôvodný *Service Worker* nahradený novým bez toho, že by si to užívateľ všimol.

4.4.2 Aplication Shell

Aplication Shell predstavuje základnú responzívnu kostru aplikácie načítanú pomocou *Service Worker* a následne uloženú v medzipamäti mobilného zariadenia. Kostra predstavuje základný vzhľad aplikácie, ktorý je po načítaní doplnený dátami. Využitím tohto spôsobu je možné napríklad rýchlejšie načítať aplikáciu. Po otvorení aplikácie sa užívateľovi poskytne naposledy načítaný obsah aplikácie, kým *Service Worker* v pozadí načíta najaktuálnejší obsah. Týmto spôsobom sa nielenže zvyšuje rýchlosť načítania stránok otvorené užívateľom, ale zvyšuje sa aj celková návštevnosť danej stránky [28].

4.4.3 Web App Manifest

Web App Manifest predstavuje súbor vo formáte JSON, obsahujúci informácie o aplikácii ako sú meno, autor, ikonka, farebná téma, popis a podobne. Poskytuje potrebné detailné informácie, pomocou ktorých je možné webovú aplikáciu nainštalovať na domovskú obrazovku mobilného či počítačového zariadenia. Týmto sa zabezpečuje pre užívateľa rýchly a jednoduchý prístup k aplikácii [28].

[5]Document Object Model – predstavuje stromovú štruktúru s HTML elementmi

[6]XMLHttpRequest – rozhranie umožňujúce komunikáciu medzi serverom a klientom pomocou HTTP protokolu

Podľa informácií zo zdroja [14] ním taktiež špecifikujeme konkrétny *Service Worker*, ktorý sa má nainštalovať a spustiť preferovanú orientáciu či jazyk pri spustení, preferované aplikácie pri otváraní odkazov. Môže sa podobne ako pre *Service Worker* určiť rozsah, čo znamená určiť stránky, ktoré budú zobrazené. Stránky nespadajúce do rozsahu sa otvoria obyčajne v preferovanom prehliadači mobilného zariadenia.

Web App Manifest sa pridáva ako HTML tag do sekcie hlavičky.

```
<!DOCTYPE html>
<head>
    .
    <base href="/">
    <link rel="manifest" href="manifest.json"
    .
</head>
```

Obr. 4.1: Ukážka pridania tagu do sekcie hlavičky

PWA aplikáciu je možné uložiť na domovskú stránku po kliknutí na výzvu v dolnej časti stránky. Výzva sa zobrazí automaticky ak sú splnené nasledujúce kritéria:

- Stránka je dostupná výhradne cez HTTPS (vyplývajúce z podmienky pre *Service Workers* a zároveň základná súčať PWA).
- Pre web app manifest musí platiť:
  - vyplnená položka *short_name* a *name*,
  - položka *icons* obsahuje ikonky s rozmermi 192x192px a 512x512px,
  - položka *start_url* má validnú hodnotu,
  - položka *display* nadobúda hodnoty *fullscreen*, *standalone* alebo *minimal–ui*.
- Aplikácia ešte nie je nainštalovaná.
- Splnenie užívateľskej angažovanosti so stránkou (táto položka sa môže zmeniť, a preto je dôležité sledovať aktuálne informácie, momentálne musí platiť, že užívateľ strávi 30 sekúnd na danej doméne).
- Aplikácia má registrovaný *Service Workers* a využíva spracovanie funkcie *fetch()*.

Pri splnení kritérií je automaticky vyvolaná udalosť *beforeinstallprompt (Add to Home Screen A2HS)*. Musí sa sledovať, kedy táto udalosť nastala aby ju bolo možné spracovať. Po zachytení je možné nastaviť, akou formou sa bude táto možnosť uloženia aplikácie prezentovať užívateľovi. Je na každom vývojárovi, aké UI elementy si vyberie pre využitie. Tým, že je možné si udalosť uložiť je dostupné určenie času, kedy zobraziť výzvu užívateľovi. Výzvu je najlepšie zobraziť tak, aby užívateľ nebol rušený pri dôležitej práci.

4.5 Rámce pre vývoj PWA

Progresívne webové aplikácie sú stále viac rozšírené a je možné ich nájsť už na viacerých webových stránkach či aplikáciach. Medzi najpoužívanejšie rámce pre ich vývoj sa

využíva napríklad Vue.js[7]. Vue.js predstavuje progresívny JavaScript rámec pre vytváranie užívateľských rozhraní. Ponúka možnosť postupného a jednoduchého začlenenia do vývoja už existujúcich aplikácií. Primárne je orientovaný na vrstvu zobrazenia, ale aj napriek tomu jeho rozsiahly systém ponúka všetky potrebné nástroje pre vývoj PWA [24].

React[8] vyniká rýchlosťou, eleganciou a schopnosťou jednoducho spravovať veľké aplikácie. Jeho výhodou je systém komponentov, umožňujúci rozdeliť si jednotlivé časti rozhrania na menšie, znovupoužiteľné časti. Je možné ju rozšíriť o nástroje potrebné pre vývoj PWA [8].

Angular[9] predstavuje platformu, ktorú je možné efektívne škálovať a ponúka všetky potrebné nástroje pre tvorbu PWA v jednom rámci. Pomocou Angular CLI môžeme vytvárať alebo konvertovať existujúce webové aplikácie na PWA [14].

V rámci implementácie využijem webový rámec Angular, predstavujúci rozsiahly rámec od spoločnosti Google pre vytváranie škálovateľných webových aplikácií. Založený je na komponentoch. Každý z komponentov tvorí trojica súborov .html, .css a .ts. Rámec je písaný použitím Typescript[10].

[7] https://vuejs.org/
[8] https://reactjs.org/
[9] https://angular.io/
[10] https://www.typescriptlang.org/

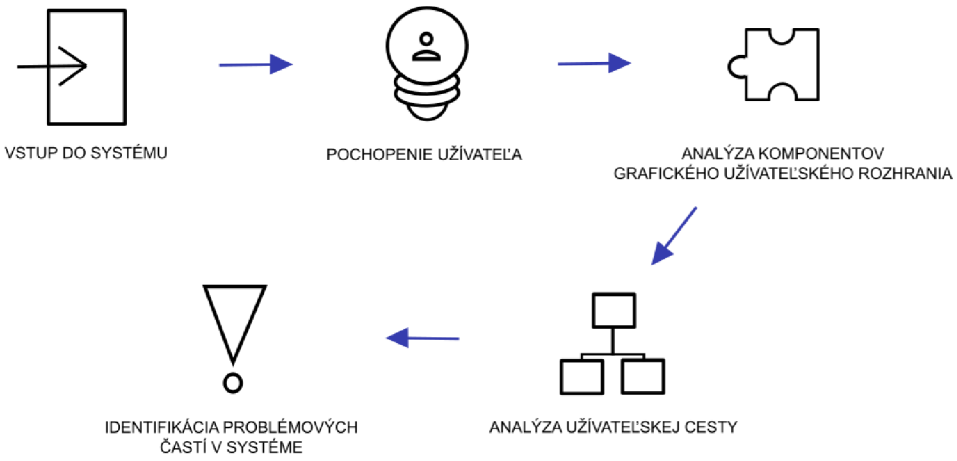
Kapitola 5

Analýza súčasného stavu

Analýzu súčasného stavu zobrazenia dashboardu so stavmi senzorov som previedla vďaka prístupu do systému firmy Logimic. Firma Logimic ponúka dashboardovú technológiu pre mestské senzory. Zbierajú, uchovávajú, filtrujú a zobrazujú dáta zo senzorových sietí s technológiou LoRa, ZigBee, IQRF a zo zariadení ako sú napríklad pouličné osvetlenia, teplotné senzory, senzory hladiny vody a mnohé ďalšie. Typický dashboard, ktorý ponúka firma Logimic, pozostáva zo serverovej backend-ovej časti a webovou aplikáciou predstavujúcou frontend. Serverová časť je zodpovedná za spracovanie, uchovanie dát, ktoré sú následne zobrazované užívateľovi pomocou dashboardu. Pre zabezpečenie škálovateľnosti, robustnosti a bezpečnosti využívajú Amazon Web Services, globálneho poskytovateľa cloudovej infraštruktúry. Lokalizácia zariadení užívateľov predstavuje kľúčovú funkcionalitu ich dashboardov. Dokážu presne lokalizovať každé jedno zariadenie a senzory. Ponúkajú uchovanie nameraných hodnôt zo senzorov či zariadení, čo zabezpečí vytváranie grafov a štatistík. Užívateľ má prehľad o dennom či mesačnom výkone daných zariadení a senzorov.

5.1 Postup analýzy

Pred analýzou som si vytvorila schému, zameranú na užívateľa a jeho interakciu so skúmaným systémom. Pri samotnej analýze som postupovala podľa krokov zobrazených v obrázku 5.1.



Obr. 5.1: Schéma znázorňujúca pochopenie systému pred analýzou

Prvým krokom po vstupe do systému bolo porozumieť súčasnému zobrazeniu na počítačovom monitore. Systém firmy Logimic poskytuje množstvo funkcií a prvkov pre užívateľa. Užívateľom poskytujú už pri prvotnom otvorení systému prehľadné zobrazenie, ukazujúce rôzne indikátory dát zo senzorov.

Po určení si cieľovej skupiny užívateľov a pochopení základných požiadaviek na systém je nasledujúcim krokom analýza komponentov grafického užívateľského rozhrania. Bolo nutné pochopiť, aké akcie sa dajú spraviť s jednotlivými komponentmi a na aké obrazovky budem ako užívateľ presmerovaná. Systém sa skladá z niekoľkých hlavnejších obrazoviek, ktoré užívateľ navštívi pri každom prihlásení sa do systému.

Tým sa dostávame k analýze užívateľskej cesty. Schéma užívateľskej cesty je popísaná už so zakomponovanými zmenami podľa návrhov vyplývajúcich z obrázka 6.1. Schéma reprezentuje všetky stránky, ktoré má užívateľ možnosť v systéme navštíviť.

5.2 Pochopenie užívateľov

Pri analýze bolo potrebné pochopiť, aké interakcie prevádza užívateľ so systémom, aké obrazovky sa mu po rôznych kliknutiach zobrazia a kedy daný systém využíva. Cieľovou skupinou systému sú viaceré typy užívateľov.

Prvým typom sú užívatelia vyžadujúci jednoduchý a intuitívny prehľad o aktuálnych stavoch senzorov či zariadení na hlavnej obrazovke. Ak je farba indikátora červená, užívateľ vie rýchlo zistiť prislúchajúci KPI, hlásajúci poruchu. V prípade poruchy vedia rýchlo zareagovať pomocou servisnej požiadavky.

Druhým typom sú servisní technici, ktorý reagujú na servisné požiadavky posielané od užívateľov prvého typu v prípade, že indikátor senzora či zariadenia hlási poruchu. Pre tento typ užívateľa je dôležitá rýchla lokalizácia daného zariadenia či senzora s poruchou. Z tohto dôvodu ponúka systém viacero vrstiev pohľadu na jednotlivé zariadenia. V nich môžeme nájsť detailnejšie technické časti.

Tretím typom sú administrátori systému. Systém pre nich ponúka správu jednotlivých užívateľov, ktorí majú prístup do daného systému. Môžu pridelovať alebo meniť role registrovaným užívateľom. Pomocou štatistík vidia vyťaženosť servisných technikov alebo počet servisných požiadaviek na dané zariadenie či senzori.

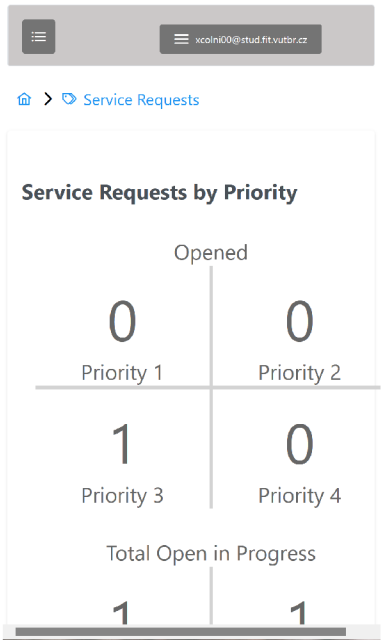
5.3 Požiadavky na systém od užívateľov

Užívatelia požadujú jednoduchý systém, zobrazujúci na domovskej stránke dashboard bez grafov či štatistík. Ako hlavnú stránku požadujú prehľadné a intuitívne zobrazenie stavov senzorov. O intuitívnosť sa stará jednoduché kruhové zobrazenie rozdelené na 4 časti, k tomu príslušná intuitívna legenda. Jednoduchým kliknutím na kruh sa vieme dostať ku KPI a užívateľ prostredníctvom ikoniek vidí, koľko a ktoré z nich fungujú správne, a naopak ktoré z nich potrebujú zásah.

Jednou z požiadaviek bolo, aby sa v systéme dala pozrieť vyťaženosť jednotlivých servisných technikov. K týmto štatistikám sa užívateľ dostane pomocou navigačnej lišty umiestnenej v hornej časti. Vyťaženosť servisných technikov uvidí pomocou intuitívneho grafu a pomocných kartičiek, ukazujúce počty spracovaných a čakajúcich opráv. Užívatelia ďalej vyžadujú od systému jednoduché ovládanie, rýchle vyčítanie informácií a prehľad o polohe zariadení pomocou mapy.

Súčasný stav domovskej stránky zobrazenej na mobilnom zariadení uvádzam na obrázku 5.2. Je z nich vidno, že mobilné zobrazenie aplikácie nie je úplne responzívne.

(a) Stránka zobrazujúca hlavný dashboard s kruhovými indikátormi.



(b) Stránka zobrazujúca počty otvorených servisných požiadaviek.

Obr. 5.2: Mobilné zobrazenie domovskej stránky a stránky znázorňujúce aktuálny počet servisných požiadaviek.

Ďalšia požiadavka vyplýva zo spôsobu využitia môjho implementovaného dizajnu. Výsledný dizajn by firma Logimic využila vo viacerých svojich systémoch, kde vyžaduje spôsob ako meniť farebnú schému, fonty, veľkosť textu či samotné rozloženie jednotlivých prvkov. Z tejto požiadavky vyplýva, že je potrebné dať na jedno miesto všetky potrebné nastavenia na zmenu týchto vlastností. Toto sa dosiahne definovaním globálnych štýlov do jedného súbora, ktorý je ďalej popísaný v podkapitole 7.3.

Nakoľko sa daný systém neustále vyvíja vpred a ponúka užívateľovi priestor na pripomienky o funkcionalite. Od užívateľov pribudlo niekoľko ďalších požiadaviek ako responzívnosť, užívateľsky prívetivejšie zobrazenie na mobilných zariadeniach, možnosť vloženia fotky v sekcii informácie o užívateľovi a následné zobrazenie fotky na dashboarde po úspešnom prihlásení sa.

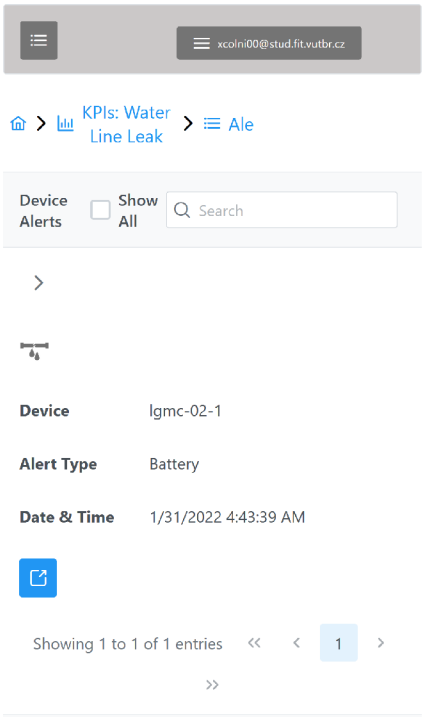
5.4 Identifikácia problémových častí

Ďalším krokom je identifikácia problémových alebo nejasných častí pre užívateľa v systéme. Analýza týchto častí nám dáva priestor na tvorbu lepšieho a užívateľsky prívetivejšieho systému.

Na určité problémy som narazila už pri analýze komponentov grafického užívateľského rozhrania. Pri vyskúšaní responzívneho zobrazenia domovskej stránky sa kruhy nezmenšili, iba sa zoradili pod seba ako je vidieť na obrázku 5.2a.

Nesprávne využitie miesta na hlavnej stránke je zapríčinené aj veľkosťou samotného loga systému. Väčšinou sa stretávame s tým, že navigačná lišta obsahuje logo a príslušné rýchle menu. V tomto systéme sa navigačná lišta využíva na prehľad prihláseného užívateľa, obrázok systémového loga a ikonku, znázorňujúcej rýchle menu ako je vidieť na obrázku 5.2a. Rýchle menu v sebe nesie niekoľko ďalších možností, či už pre nastavenie, zobrazenie stránky pre administrátora a rôzne ďalšie. Logo systému je zobrazené mimo navigačného panela aj so samotnými kruhovými zobrazeniami pre stavy jednotlivých zariadení.

Po prekliknutí sa z domovskej stránky na ďalšiu stránku, sa dostávam na zobrazenie tabuľky všetkých zariadení a mapu s vyznačenými lokalitami. Mapa je pre užívateľa obrovským prínosom. Jednoducho vidí názov ulice, kde sa zariadenie nachádza, čo mu dáva možnosť nájsť zariadenie aj bez toho, že by poznal jeho názov. Čo sa týka tabuľky, obsahuje množstvo informácií ako napríklad meno zariadenia, adresa, pridelený servisný technik v prípade, že bola zahlásená porucha, atď. Problémom pre užívateľa je mobilné zobrazenie tejto stránky zobrazené na obrázku 5.3.



Obr. 5.3: Mobilné zobrazenie stránky s tabuľkou všetkých zariadení

Z tohto obrázku je vidieť, že zobrazenie nie je užívateľsky prívetivé. Pre užívateľa môže prísť k problému s čitateľnosťou textu, pochopením funkcionality prvkov ako napríklad šípka v prvom riadku tabuľky, tlačítko alebo prevedenie zobrazenia užívateľskej cesty pod vrchnou navigačnou lištou.

Medzi hlavné identifikované problémy sa radia responzívnosť celého systému a lepšie rozloženie komponentov na jednotlivých obrazovkách. Na viacerých obrazovkách systém nereaguje na responzívne zobrazenie užívateľsky prívetivým zobrazením ako je zobrazené aj na obrázku 5.2b. Z tohto obrázku je vidieť v dolnej časti obrazovky posuvnú lištu, ktorá indikuje, že prvky sa dostatočne nezmenšili. Je vidno aj príliš veľkú mriežku s veľkými číslicami.

Napriek týmto pár nedostatkom na mobilnom zobrazení ponúka systém užívateľovi komplexný prehľad o zariadeniach či senzoroch, a ponúka mu aj pokročilú funkcionalitu pre ich správu. Všetky definované nedostatky sú len v rámci užívateľskej prívetivosti, ktoré vieme pomocou testovania a vhodného návrhu vylepšiť.

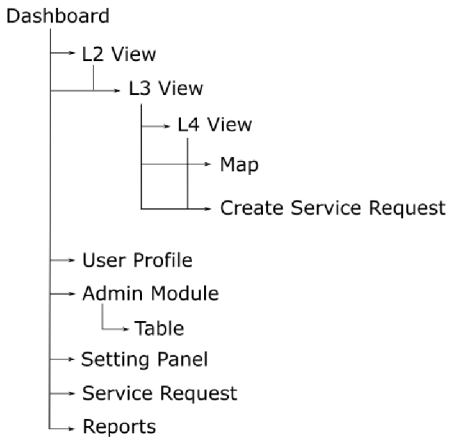
Kapitola 6

Návrh

Počas celej mojej práce som sa snažila pochopiť svojich koncových užívateľov, aké informácie potrebujú mať zobrazené, s akými prvkami užívateľského rozhrania sa im najľahšie manipuluje, poprípade na aké prvky sú už zvyknutí. Vytvorených bolo niekoľko rôznych papierových prototypov, digitálnych prototypov a celý proces bol orientovaný na užívateľa. S týmto dizajnom zameraným na užívateľa je dôležité dbať na kvalitu zmien problémových prvkov a nie na kvantitu zmenených prvkov. Veľké zmeny môžu prispieť k narušeniu užívateľskej prívetivosti, z hľadiska vzhľadu, funkcionality či rozmiestnenia informácií. Počas vytvárania návrhov dizajnu som dodržovala postupy a kroky vyplývajúce z UX, popísané v kapitole 3. Vždy som zohľadňovala aj pripomienky a návrhy na zlepšenie od mojich konzultantov.

6.1 Architektúra

Napriek tomu, že hlavným cieľom tejto práce je návrh a implementácia zobrazenia webovej aplikácie na mobilnom zariadení, musím previesť zmeny užívateľského rozhrania aj na počítačovom zobrazení. Dôvodom je, aby štýl na mobilnom zariadení bol porovnateľný a konzistentný so zobrazením na väčších zariadeniach. Hlavným cieľom systému je poskytnúť užívateľovi jednoduchý a intuitívny prehľad stavov jednotlivých senzorov a ich KPI. Už pri prvotnom pohľade po vstupe do systému musí byť užívateľ schopný zistiť či nenastala porucha na niektorom zo zariadení.



Obr. 6.1: Schéma znázorňujúca užívateľskú cestu mojej aplikácie

Nakoľko systém nebol responzívny a prispôsobený na mobilné zariadenia, návrh som začala vytvorením užívateľskej cesty ukázanej na obrázku 6.1. Táto cesta označuje možnosti užívateľskej cesty po vstupe do systému a pomáha lepšie pochopiť možnosti navigácie v rámci aplikácie. Cesta sa začína na domovskej stránke a postupne sa rozvetvuje a zanoruje, čím sa dostávame k viac detailnejším informáciám. Pomocou týchto vrstiev som lepšie vedela, ako budú jednotlivé stránky prepojené. Definíciou jednotlivých vrstiev som si vopred naplánovala počet možných stránok, potrebných zahrnúť do prvotných návrhov dizajnu.

6.2 Návrh grafických komponentov

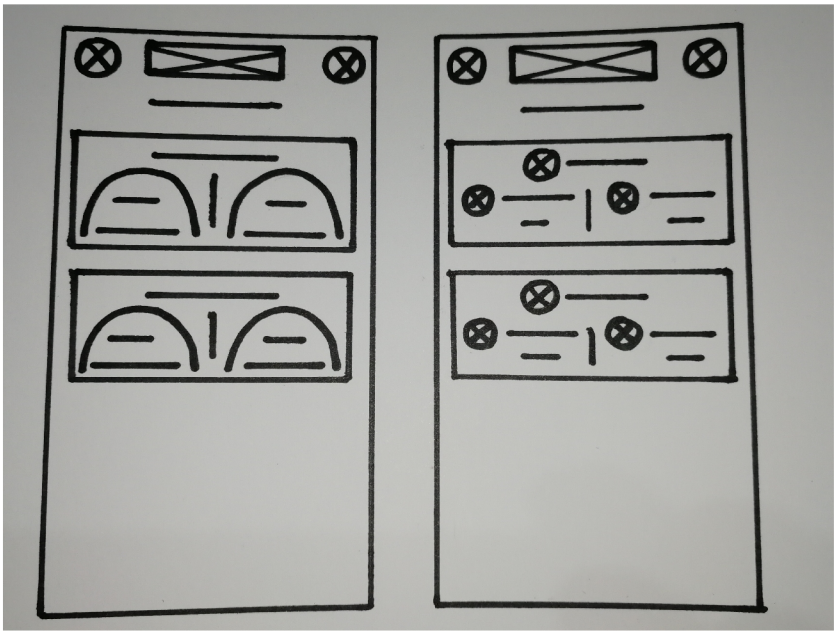
Na začiatku som vypracovala niekoľko návrhov pre zobrazenie stavov senzorov. Svoje návrhy som začala týmto prvkom, nakoľko je nosným pilierom celého systému. Niektoré návrhy boli skoro totožné so súčasným riešením iba s malými odchýlkami a naopak som mala aj návrhy, ktoré boli úplne odlišné. Po prezentácii mojich návrhov vo firme Logimic sme vybrali návrh, ktorý má iba menšie odlišnosti od súčasného riešenia.

Návrh pozostáva z kruhu rozdeleného na štyri časti ako je zobrazené na obrázku 6.2. Každá časť reprezentuje svoj stav farbou, ktorá indikuje užívateľovi či je potrebný nejaký zásah. Taktiež jednotlivé časti reprezentujú istý KPI. Pod krúžkom zobrazujem legendu týchto štyroch častí, ktoré si vie užívateľ jednoducho zmeniť a zobraziť si iné potrebné KPI. Vnútri kruhu je príslušná ikonka a názov daného zariadenia. Zhodli sme sa na ňom aj z dôvodu, že užívatelia sú zvyknutí na súčasné počítačové zobrazenie využívajúce taktiež kruh s legendou, kde je však legenda implementovaná pomocou názvu pre danú sekciu kruhu. Nahradenie kruhového zobrazenia by mohlo užívateľom sťažiť užívateľskú prívetivosť pri otváraní systému cez počítač. Kruhové zobrazenia majú dve skupiny. Prvá skupina zodpovedá zobrazeniu skupiny zariadení podľa typu a druhá zobrazuje skupinu zariadení s KPI.

Obr. 6.2: Schéma znázorňujúca na ľavej strane aktuálnu podobu kruhových indikátorov a na pravej strane je zobrazený môj návrh na zobrazenie kruhového indikátora.

Po rozkliknutí ktoréhokoľvek kruhového zobrazenia skupiny zariadení podľa typu sa užívateľovi zobrazí zoznam jednotlivých zariadení a ich senzorov, ako je vidieť na obrázku 6.3 na ľavej strane. Stavy sú zobrazené pomocou polkruhu s farebným indikátorom, označujúci kategóriu stavu a krátkym názvom.

Po rozkliknutí ktoréhokoľvek kruhového zobrazenia skupiny zariadení s KPI sa užívateľovi zobrazí zoznam s ich príslušnými kľučovými ukazovateľmi výkonnosti, ako je zobrazené na obrázku 6.3 na pravej strane. Stavy sú zobrazené pomocou ikonky s príslušnou farbou a príslušným názvom.



Obr. 6.3: Náčrt zobrazujúci papierový prototyp mobilného zobrazenia stránky po rozkliknutí kruhových indikátorov.

Na mobilnom zobrazení už nemôžem použiť kruhové zobrazenie, z dôvodu nedostatku miesta. Kruhové zobrazenie som teda transponovala na zoznam, tvorený príslušnou ikonkou, názvom a farebným indikátorom o stave senzora či KPI. Týmto spôsobom sa efektívnejšie využíva dostupné miesto a zároveň sa zachováva konzistencia zobrazenia medzi rôznymi veľkosťami zobrazenia.

6.2.1 Navigačná lišta

Ďalším krokom bolo navrhnúť celú obrazovku zobrazenia a rôzne ďalšie naväzujúce obrazovky. Nasledujúcou výzvou pre mňa bolo navrhnúť navigačnú lištu, ktorá môže mať množstvo podôb a môžeme ju mať rôzne rozloženú. V súčasnom počítačovom zobrazení je navigačná lišta vsadená hore a ponúka užívateľovi menu s ďalšími možnosťami. Pekne zobrazuje logo daného systému a mailovú adresu prihlásenej osoby. V tomto zobrazení by som využila skôr bočný navigačný panel, pozostávajúci z loga systému, fotky a mailovej adresy užívateľa, rýchlej voľby k iným nastaveniam či kartám a tlačidlo na odhlásenia sa zo systému. Moje bočné menu ponúka užívateľovi indikátor v podobe zafarbeného obdĺžnika systémovou farbou, ukazujúci, na ktorej stránke sa užívateľ nachádza. Vďaka tomu, že sa bočný panel vysúva na prechod kurzorom, užívateľ má možnosť rýchlej zmeny zobrazenia či rýchlemu návratu na domovskú stránku.

Mobilné zobrazenie navigačnej lišty sa líši tým, že zobrazuje ikonku reprezentujúcu navigačnú lištu. Po kliknutí na túto ikonku sa zobrazí dialógové okno. Štýlom aj obsahom je menu totožné s počítačovým zobrazením.

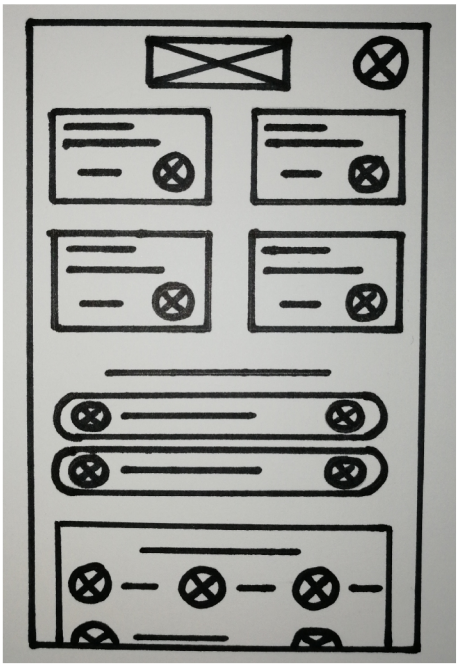
6.2.2 Rýchly prehľad zariadení

Oproti súčasnému stavu som do návrhu pridala na hlavný dashboard 5 kartičiek. Prvé štyri kartičky zaručia užívateľovi rýchly prístup do jednotlivých stavov senzorov podľa kategórie.

Zobrazujem na nej názov danej kategórie, ikonku a počet zariadení, ktoré aktuálne patria do danej kategórie. Po rozkliknutí týchto kartičiek sa dostávam na obrazovku so zoznamom zariadení a senzorov, ktorých stavy korešpondujú s kategóriou na kartičke. Na kartičke je daná kategória zobrazená nielen slovným pomenovaním, ale aj ikonkou a jej prislúchajúcou farbou. Návrh týchto kartičiek vznikol, aby som umožnila rýchlejší prehľad kľúčových informácii pre užívateľa. Jedným pohľadom vie zistiť počet zariadení vyžadujúcich akciu, bez toho, že by si musel v kruhových indikátoroch hľadať príslušné zariadenie a jeho KPI.

Posledná piata kartička, zobrazená na obrázku 7.2, slúži na nastavenie zariadení, zobrazujúcich sa aktuálne na domovskej stránke. Užívateľ dostáva možnosť výberu skupín zariadení a senzorov zobrazených na domovskej stránke. Po výbere zariadenia, užívateľ vyberá štyri kľučové ukazovatele výkonnosti, zobrazené na kruhovom indikátore danej skupiny zariadení. Týmto dávam užívateľovi možnosť interakcie so systémom a taktiež dostáva možnosť prispôsobenia si obsahu, krorý si chce zobraziť na domovskej stránke.

Mobilné zobrazenie zachováva zobrazenie štyroch kartičiek ako je zobrazené na obrázku 6.4. Ponechaním týchto kartičiek na všetkých zariadeniach chcem užívateľovi ponúknuť rýchly prístup bez ohľadu na veľkosť aktuálne používaného zariadenia.



Obr. 6.4: Prototyp mobilného zobrazenia domovskej stránky. Responzívne zobrazenie kartičiek a kruhových indikátorov, ktoré sa usporiadajú do zoznamu.

Piata kartička sa na mobilnom zobrazení už nenachádza. Dôvodom je nedostatok miesta na takýto zložitý prvok, avšak užívateľovi to nahradzujem zobrazením zoznamom všetkých zariadení, ktoré daný systém obsahuje.

6.2.3 Panel pre správcu systému

K tomuto zobrazeniu sa jednoducho preklikneme pomocou bočného panelu. Užívateľovi sa naskytne pohľad na deväť dlaždíc. Tieto dlaždice slúžia ako rýchla voľba pre užívateľa, ktorá zabezpečuje väčšiu prehľadnosť a intuitivitu. Každá z nich nesie príslušný názov kategórie nastavení informácií a ikonku, pre rýchlejšiu orientáciu užívateľa. Po rozkliknutí dlaždíc

máme deväť možných obrazoviek. V každej sa nachádza tabuľka, ktorá ponúka možnosť editácie informácii.

Dlaždicové zobrazenie zachovávam aj na mobilnom zobrazení, s tým, že sa prispôsobuje veľkosť dlaždíc. Následná tabuľka sa transponuje na zoznam jednotlivých parametrov s možnosťou vyrolovania ďalších doplňujúcich informácii.

6.2.4 Profil užívateľa

Stránka pre profil užívateľa a jeho nastavení je rozdelená pomocou tabov do niekoľkých sekcií. Prvou sekciou sú osobné údaje, kde si užívateľ vyplní celé meno, mail, ktorým sa chce prihlasovať do aplikácie a možnosť vloženia a vymazania profilovej fotky. Druhou samostatnou sekciou je zmena hesla, použité pre samotné prihlasovanie sa do systému. Návrh zahŕňa aj spätnú väzbu pri zadávaní nového hesla, ktoré užívateľa informuje o sile novo zadaného hesla. Treťou sekciou sú informácie o bydlisku užívateľa a štvrtou je sekcia, ktorá v sebe nesie systémové nastavenia ako výber jazyka systému a jednotky teploty. Do návrhu som zakomponovala aj prepínač na jednoduché zapnutie systémových notifikácií.

Pri mobilnom zobrazení sa jednotlivé sekcie zobrazia najprv v podobe dlaždíc ako to je aj v prípade panelu pre správcu systému. Po následnom výbere sa užívateľovi objavia príslušné nastavenia na samostatnej stránke.

6.2.5 Servisné požiadavky

Stránka so servisnými požiadavkami v mojom návrhu zobrazuje štyri kartičky rozdelené podľa závažnosti poruchy. Nesie v sebe informácie o počte novovytvorených, práve vykonávaných a už vyriešených servisných požiadavkách za posledné dva či jeden deň. Pre lepší prehľad sa tu nachádzajú dva grafy. Znázorňujú vyťaženosť jednotlivých servisných technikov. V grafe užívateľ vidí koľko servisných požiadaviek musia jednotliví servisní technici ešte vyriešiť. Mobilné zobrazenie sa líši iba absenciou grafov. Tento prvok vynechávam z dôvodu, že graf by sa stal nečitateľným a interakcia s ním by bola pre užívateľa zložitá.

6.2.6 Detailné informácie o zariadeniach

V systéme je možnosť prehľadu podrobnejších informácií o jednotlivých zariadeniach. Obrazovka je rozdelená na dve časti. Na ľavej strane zobrazujem informácie o aktuálnych upozorneniach, vyžadujúcich zásah. Pomocou polkruhov zobrazujem na pravej strane prehľadné štatistiky o celkovom využití. Užívateľ z nich vyčíta všetky informácie o spotrebe už pri prvotnom pohľade. Pomocou vyrolovacích riadkov v dolnej časti obrazovky, ponúkam užívateľovi možnosť rozkliknúť si ďalšie kategórie informácií ako sú technické parametre zariadenia, dokumentácia a ďalšie relevatné podrobnosti.

Mobilné zariadenie má iné rozloženie obrazovky. Nakoľko tu nemám dostatok miesta, rozhodla som sa na každú kategóriu informácii použiť vyrolovací riadok. To znamená, že ak si chce užívateľ pozrieť informácie o aktuálnych upozorneniach, zobrazia sa mu až po samotnom vyrolovaní.

Kapitola 7

Implementácia

Po schválení dizajnových návrhov nasleduje samotná implementácia aplikácie. Výsledná aplikácia je implementovaná pomocou webového aplikačného rámca Angular a použitá je aj knižnica PrimeNG[1], ktorá ponúka už naštýlované komponenty užívateľského rozhrania. Niektoré moje prvky z dizajnu sú implementované práve pomocou týchto predpripravených komponentov z knižnice, ktoré som už len prispôsobila k svojim návrhom. Využitie tejto knižnice bolo prínosné aj z dôvodu plynulejšej integrácie do súčasnej webovej aplikácie, nakoľko sa v nej aktuálne využíva.

Pri vývoji aplikácie som postupovala podľa užívateľskej cesty zobrazenej na obrázku 6.1, ktorú som opísala už vyššie. Postupne som implementovala jednotlivé stránky užívateľskej cesty. Najprv som vytvárala menšie prvky užívateľského rozhrania. Tie následne spolu tvoria jednotlivé stránky aplikácie. Každý prvok s možnosťou viacnásobného použitia a každá stránka, je zapuzdrená vo svojej vlastnej zložke ako samostatný komponent. Z týchto komponentov som vytvorila moduly, ktoré prinášajú pre firmu Logimic možnosť integrovať moje komponenty.

7.1 Komponenty

Webový aplikačný rámec Angular umožňuje vytvárať znovupoužiteľné prvky pomocou takzvaných komponentov. Komponent predstavuje trieda v jazyku TypeScript s dekorátorom *@Component()*, definujúca niekoľko vlastností, ako je *selector*, *templateUrl* a *styleUrls*. Selector je reťazec, predstavujúci názov znovupoužiteľného HTML elementu ako aj názov CSS selektorov pre štýlovanie daný komponent. *TemplateUrl* predstavuje cestu k HTML kódu daného komponentu a styleUrls predstavuje naopak cestu k štýlom daného komponentu. V mojej implementácii každý komponent predstavuje trojica súborov .html, .css a .ts.. Súbor .html udáva HTML kostru komponenty, .css súbor udáva štýl pre daný komponent a .ts súbor obsahuje triedu komponenty, udávajúcu možné nastavenia a premenné pre dynamický obsah či štýlovanie.

7.2 Moduly

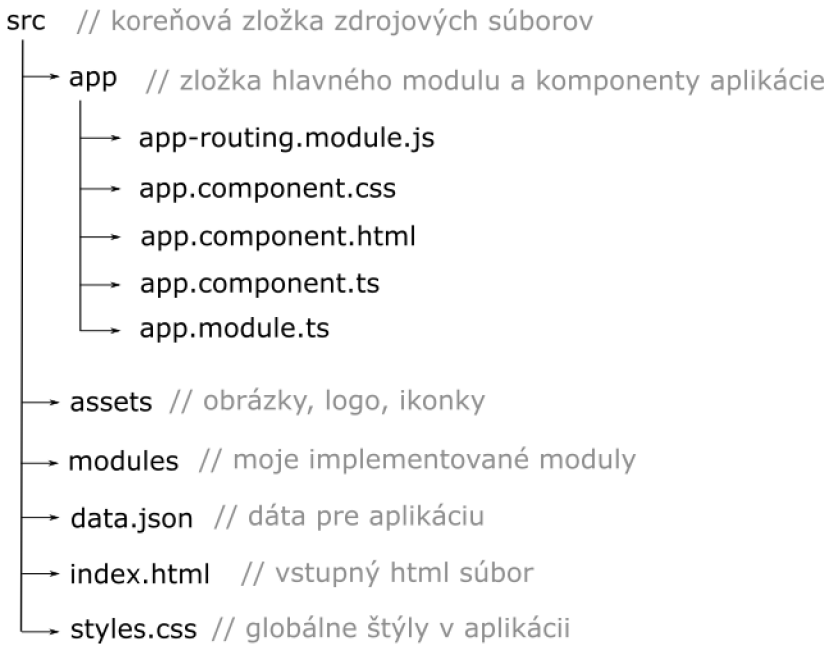
Angular ponúka aj svoj vlastný systém pre vytváranie modulárnych aplikácií nazývajúci sa *NgModules*. Hlavným modulom každej aplikácie je *AppModule*, v ktorom sa pri spustení aplikácie načíta hlavný komponent *AppComponent*.

[1] https://www.primefaces.org/primeng/

Modul predstavuje triedu s dekorátorom *@NgModule()* definujúci nastavenia, vlastnosti a komponenty. Najdôležitejšie vlastnosti sú *declarations*, *imports* a *exports*. V zozname declarations sa definujú komponenty patriace a definované pod daný modul. Zoznam *imports* definuje exportované triedy (komponenty) z iných modulov, využívané daným modulom. Exports je naopak pole komponent, sprístupnené daným modulom v iných moduloch ak je importovaný.

7.2.1 Implementované moduly

Jednotlivé mnou vytvorené komponenty sú obsiahnuté v samostatných moduloch. Tie následne používam v iných moduloch, a tak vytváram výsledné rozloženie stránok. Na obrázku 7.1 je zobrazená štruktúra súborov so zdrojovými kódmi.



Obr. 7.1: Schéma znázorňujúca hierarchiu relevantných zložiek a súborov implementovanej aplikácie

7.2.2 Koreňový modul

Koreňový modul `AppModule` a jeho komponenta `AppComponent` predstavuje vstup do aplikácie a zahŕňa hlavné rozdelenie zobrazenia nasledujúcim spôsobom. Pre počítačové zobrazenie je na ľavej strane obrazovky vyčlenené miesto pre komponent navigačného panela a zvyšný priestor je vyčlenený pre samotný hlavný obsah jednotlivých stránok.

Pri mobilnom zobrazení[2] sa v hornej časti obrazovky nachádza komponent, `mobile-top-bar` ako je zobrazené na obrázku 7.4. Tento komponent zobrazuje užívateľovi logo aplikácie, ikonku znázorňujúcu ponuku menu a ikonku šípky späť s možnosťou vrátiť sa na predchádzajúcu stránku. Pod touto hornou lištou sa nachádza priestor pre hlavný obsah stránky.

[2]Mobilné zobrazenie v mojej práci znamená, ak šírka okna aplikácie je menšia ako 820px.

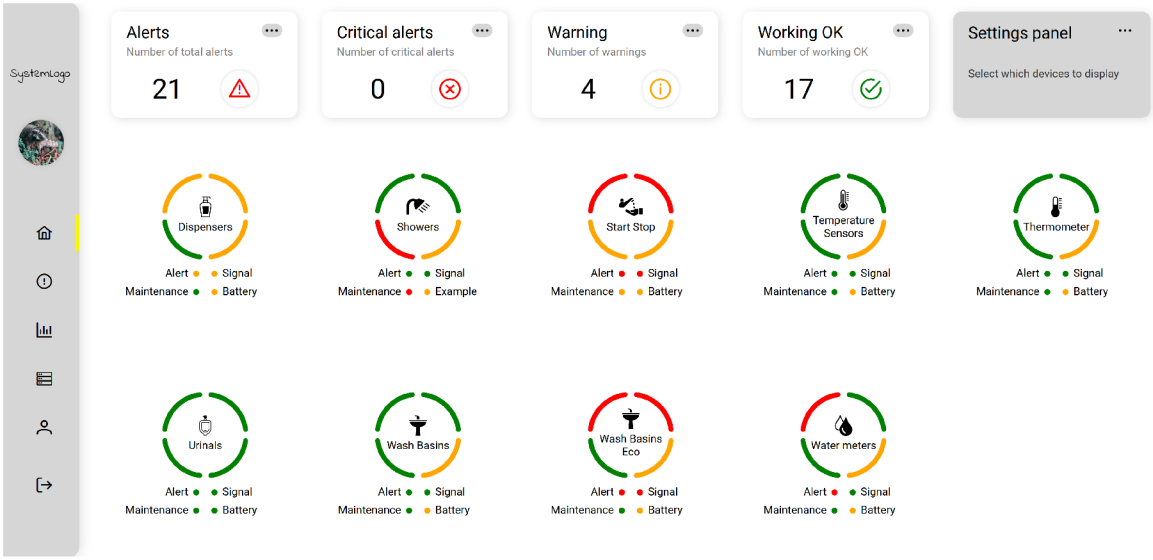
Či už sa jedná o počítačové alebo mobilné zariadenie, prepínanie medzi jednotlivými stránkami zabezpečujem pomocou direktívy `router-outlet`, dostupnej z modulu `RouterModule`.

Direktíva `router-outlet`, slúži ako miesto naplnené obsahom komponenty, určenej na základe URL cesty. Jednotlivé komponenty je možné namapovať k daným cestám v zozname `Routes` zloženého z objektov `Route` definujúcich cestu a príslušný komponent. Navigácia je možná pomocou aplikácie direktívy `RouterLink` na ľubovolný komponent a následne po kliknutí naň sa mení URL cesta, a tým aj komponent, zobrazujúci sa na mieste, kde sa nachádza router-outlet. V mojej implementácii sú jednotlivé objekty typu `Route` deklarované v súbore `app-routing.module.ts`.

Aby som odlíšila názvy svojich komponentov od ostatných, rozhodla som sa ich selektory označovať prefixom `bpc-názov_komponenty`. Následne názvy tried komponentov, reprezentujúcich jednotlivé stránky aplikácie označujem prefixom `BPCNázovTriedyKomponenty`.

7.2.3 Domovská stránka

Domovská stránka aplikácie, viditeľná v momente, kedy je užívateľ úspešne prihlásený. Implementovaná je v moduli `PageDashboardModule`, ako samostatný komponent, s využitím niekoľkých ďalších modulov a ich komponentov.



Obr. 7.2: Zobrazenie hlavného dashboardu na počítačovom zariadení. Na ľavej strane je zobrazený navigačný panel s logom systému, užívateľskou fotkou[3]. V hornej časti sú zobrazené kartičky, pomocou komponenty `bpc-card`. Posledná piata kartička slúži na otvorenie nastavení voľby, slúžiacej na výber zobrazených zariadení a ich jednotlivých KPI. Hlavnú časť tvoria kruhové indikátory, v ktorých je ikonka[4]a prislúchajúci názov zariadenia. Pod nimi sa nachádzajú maximálne štyri vybrané KPI.

Stránka sa skladá z troch logických častí, vyplývajúcich z počítačového zobrazenia na obrázku 7.2. Konkrétne ide o navigačný panel, horný riadok s kartičkami a samotné kruhové znázornenie KPI, v podobe kruhov.

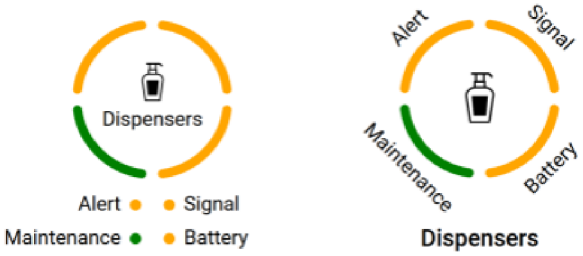
[4]Fotka pre užívateľa je prevzatá zo zdroja https://unsplash.com/
[4]Ikonky použité v kruhových indikátoroch sú prevzaté zo zdroja https://uxwing.com/.

V hornom riadku sú jednotlivé kartičky implementované ako komponenty zapuzdrené v samostatných moduloch. Vlastnosti kartičky je možné nastaviť pomocou tried komponent. Konkrétne sa používajú nasledujúcimi spôsobmi:

Príklad 7.2.1.
```
<moj-komponent názov_vlastnosti="hodnota"></moj-komponent>
```

Príklad 7.2.2.
```
<moj-komponent [názov_vlastnosti]="premenná"></moj-komponent>
```

V príklade 7.2.1 uvádzam ukážku nastavenia týchto hodnôt priamo a príklad 7.2.2 ukazuje nastavenie hodnôt pomocou premennej. Vzhľad kartičiek je prispôsobený pre správne zobrazenie aj na menších zariadeniach pomocou CSS štýlov a to pomocou modulu `Media Query`[5].



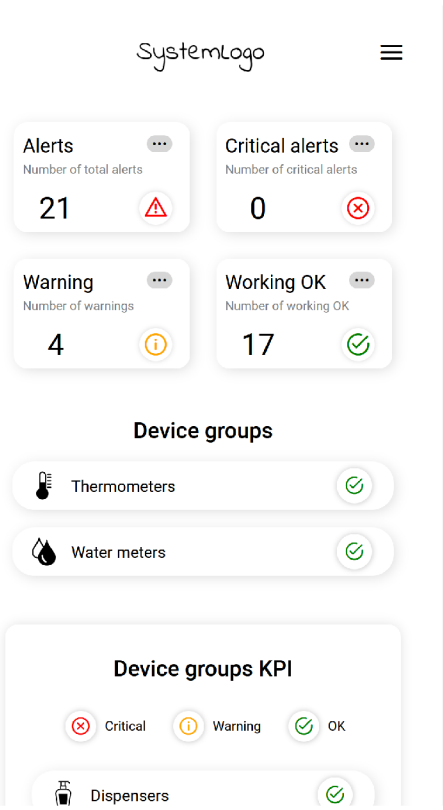
Obr. 7.3: Zobrazené dva typy kruhových indikátorov. Na ľavej strane je zobrazený môj nový vzhľad kruhových indikátorov a na pravej strane je zobrazený nový dizajn pre aktuálne používaný štýl týchto indikátorov.

Kruhové indikátory implementované v module `CircleModule`, znázorňujú štyri KPI ako je zobrazené na obrázku 7.3. Implementované sú pomocou komponentu `p-chart` dostupnej z knižnice PrimeNG. Komponent p-chart dostupný z importovaného modulu `ChartModule`, pôvodne slúžil na znázornenie rôznych typov grafov. Pre implementáciu môjho návrhu som si prispôsobila tento komponent tak, aby bol graf permanentne tvorený štvrťkruhami. Farbu jednotlivých štvrťkruhov je samozrejme možné prispôsobiť si, podľa požiadaviek užívateľa.

Pre jednotlivé kruhové indikátory je možné pomocou vstavanej vlastnosti `ngClass` nastaviť ikonku, reprezentujúcu dané zariadenie, názov, popis jednotlivých KPI a ich farebnú reprezentáciu je možné nastaviť pomocou definovaných vlastností v triede komponent.

Kruhové indikátory sa pri mobilnom zobrazení zobrazujú iným štýlom. Pri mobilnom zobrazení už komponent nevyužíva modul `ChartModule`, ale skladá sa z ikonky zariadenia, názvu zariadenia a farebnej ikonky reprezentujúcej stav KPI. Jednotlivé kruhové indikátory sú vsadené do mriežky, ktorá sa automaticky preskupí, aby sa jej obsah zmestil do dostupného miesta. Aby táto vlasnosť správne fungovala, je potrebné nastaviť šírku prvkov, ktoré budú do mriežky vložené. Komponent mriežky je implementovaný v module `GridDivModule`. Pri mobilnom zobrazení sa mriežka mení na zoznam, kde jednotlivé prvky sú radené pod seba ako je zobrazené na obrázku 7.4.

[5] https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries



Obr. 7.4: Zobrazenie hlavného dashboardu na mobilnom zariadení. Logo a ikonka znázorňujúca menu, predstavuje komponentu `mobile-top-bar`. Prvky pod kartičkami zobrazujú responzívny vzhľad kruhových indikátorov.

7.2.4 Zobrazenie skupín zariadení a ich upozornení

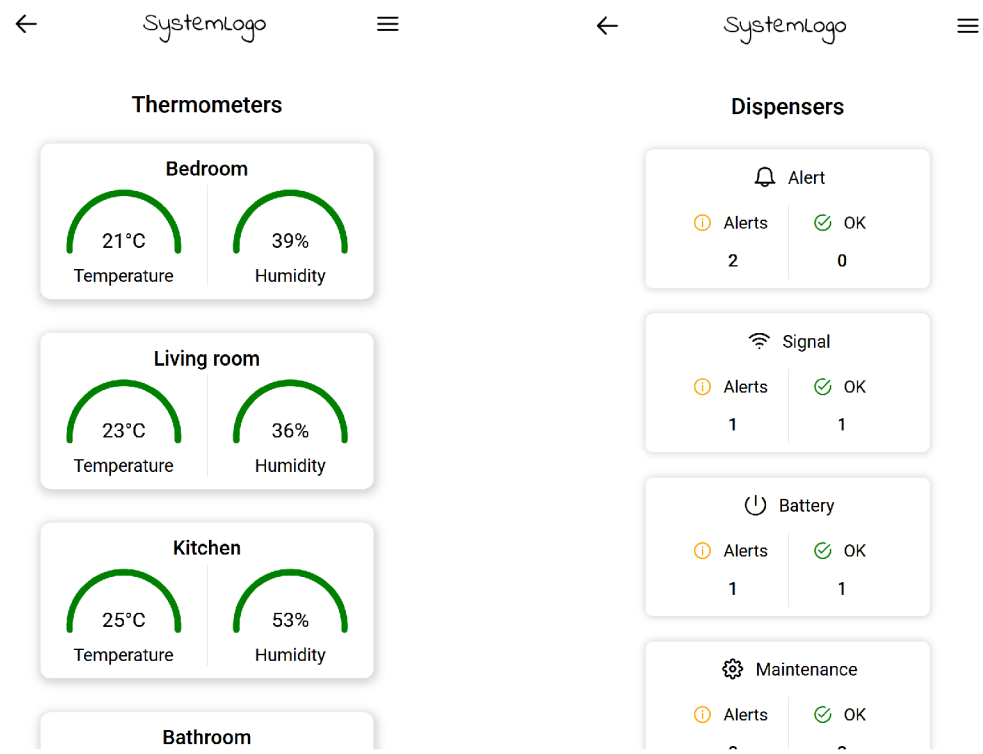
Po kliknutí na kruhový diagram sa z domovskej stránky dostávame na stránku, majúcu dve varianty ako je zobrazené na obrázku 7.5.

Prvá varianta je dostupná z modulu `PageDeviceGroupKPI`. Na počítačovom zobrazení znázorňuje jednotlivé KPI daného zariadenia v podobe obdĺžnikov s názvom a ikonkou reprezentujúcou dané KPI. Ďalej k nim zobrazujem prislúchajúci počet zariadení s upozornením a počet zariadení pracujúcich správne. Pri menšom zobrazení ako je na obrázku 7.5b, sa počty zariadení, u ktorých je upozornenie a tie čo pracujú správne zobrazia pod názvom a ikonkou daného KPI.

Druhou variantou je zobrazenie skupiny zariadení dostupné z modulu `PageDeviceGroup`. Jednotlivé zariadenia sú znázornené pomocou kartičiek s názvom zariadenia a hodnôt ich senzorov. Tieto hodnoty sú zobrazené pomocou dvoch polkruhov s číslom v strede reprezentujúcich hodnotu senzoru, ako je zobrazené na obrázku 7.5a. Toto zobrazenie zabezpečuje komponent z modulu `DoubleHalfCircleModule` vsadený do mriežky, kde opäť využívam komponent z modulu `GridDivModule`.

Komponenty obidvoch variant týchto stránok nás po kliknutí zoberú na rovnaké miesto `PageMapTableModule`. Ide o stránku s tabuľkou, kde sú jednotlivé upozornenia, ktoré k daným zariadeniam prislúchajú. Toto zobrazenie je na obrázku 7.6. Na počítačovom zobrazení sa vedľa tabuľky nachádza ešte aj mapa[6], znázorňujúca polohu zariadení a senzorov.

[6]Mapa využíva knižnicu *geovisto* – https://www.npmjs.com/package/geovisto



(a) Stránka zobrazujúca namerané hodnoty zo senzorov danej skupiny zariadení.

(b) Stránka zobrazujúca jednotlivé KPI danej skupiny zariadení.

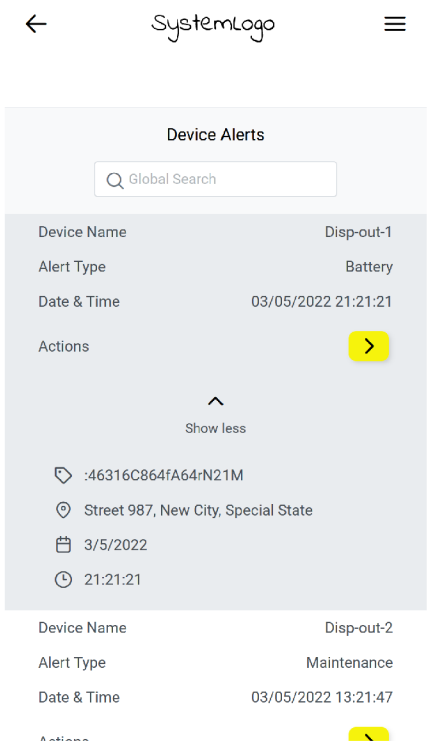
Obr. 7.5: Mobilné zobrazenie stránok skupín zariadení.

Jednotlivé upozornenia v tabuľke nesú informáciu o type a závažnosti upozornenia. Nachádza sa v nich tlačítko s vyskakujúcim dialógovým oknom. Práve to ponúka pre užívateľa ďalšie možnosti, ako je vytvorenie servisnej požiadavky, podrobnejšie informácie o danom zariadení a podobne. Dialógové okno je implementované ako samostatný komponent v module `DialogWindowModule`. Pre jeho implementáciu bola využitá knižnica PrimeNG a jeho komponent `p-dialog`, ktorý som prispôsobila preťažením štýlov knižnice v súbore `style.css`. Prispôsobenie zahŕňalo úpravu zobrazenia dialógového okna pri mobilnom zobrazení. Na menšom zariadení sa dialógové okno vysúva zdola a pri počítačovom zobrazení sa centruje na stred obrazovky. Pri zvolení možnosti *Create Service Request* sa v dialógovom okne modulu `DialogWindowModule`, zobrazí formulár s kolónkami pre vstup od užívateľa. Formulár predstavuje mriežka, vytvorená pomocou komponenty z modulu `GridDivModule`, v ktorej sú vsadené komponenty modulu `InputBoxModule`. Komponent z tohto modulu som implementovala tak, aby ponúkal rôzne typy vstupov. Typ je vybraný pomocou nastavenia vlastnosti `type`, ktorá môže nadobúdať nasledujúce hodnoty.

- text – kolónka pre jednoriadkový textový vstup,
- double-text – dve menšie kolónky pre jednoriadkový textový vstup,
- password – kolónka pre zápis hesla, s možnosťou zobrazovať silu zadaného hesla,

- dropdown – kolónka s výberom z možností,
- switch – prepínač umožňujúci prepnúť medzi dvoma hodnotami,
- textfield – kolónka pre viacriadkový textový vstup.

Všetky typy využívajú komponenty z knižnice PrimeNG prispôsobené preťaženými štýlmi v súbore `style.css`.



Obr. 7.6: Mobilné zobrazenie zobrazujúce stránku `PageMapTableModule`. Nachádza sa tu tabuľka, zobrazujúca informácie o upozornení od daného zariadenia.

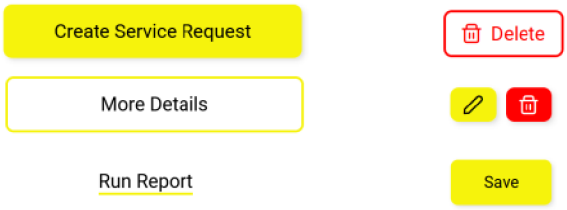
7.2.5 Podrobné informácie o zariadení

Výber možnosti *More Details* pri dialógovom okne presmeruje užívateľa na komponent z modulu `PageDeviceDetails`. Reprezentuje stránku s podrobnými informáciami o danom zariadení.

Tlačítka zobrazené na obrázku 7.7 sú implementované v module `CustomButtonModule`, ktorého komponent využíva prvok pre tlačítko z knižnice PrimeNG. Nastavením vlasnosti `type` sa určuje dôležitosť tlačítka. Táto vlastnosť môže nadobudnút nasledujúce hodnoty:

- primary – tlačítko s pozadím hlavnej farby aplikácie, reprezentuje tlačítko, ktoré by sme chceli aby si užívateľ vybral alebo si ju vyberá najčastejšie,
- secondary – tlačítko s orámovaním reprezentujúce druhú najviac žiadanú možnosť,
- ternary – tlačítko, ktoré má v sebe podčiarknutý text primárnou farbou aplikácie a predstavuje voliteľnú možnosť,

- quaternary – tlačítko, ktoré sa javí ako text, vyjadruje možnosť ktorá sa využíva najmenej.



Obr. 7.7: Ukážka jednotlivých typov tlačítok. Prvý stĺpec zhora dole označuje primárny, sekundárny a ternárny typ. V druhom stĺpci prvý typ tlačítka označuje sekundárny s ďalšími špecifikáciami. Druhý riadok v tomto stĺpci predstavuje tlačítka použité v tabuľke. Posledný typ tlačítka je na stránke s užívateľskými nastaveniami.

Tlačítku je ďalej možné prispôbiť šírku, celkovú velkosť, ale aj pridať ikonku. V ľavej časti stránky máme najčastejšie vyžadované informácie o danom zariadení ako sú kvalita signálu, stav batérie či spotreba. Tieto informácie sú prezentované v podobe polkruhu s hodnotou v jeho strede. Toto zobrazenie implementujem v komponente modulu `HalfCircleModule`. Pre zobrazenie polkruhov opäť využívam komponent importovaný z modulu `GridDivModule`.

Pod hlavnou časťou sa nachádza zoznam s kategóriami informácií s možnosťou vyrolovania jeho obsahu. Tento zoznam je implementovaný pomocou komponenty `p-table` z knižnice PrimeNG, ktorý som si prispôsobila aby neobsahoval hlavičku tabuľky a využila som možnosť `rowexpansion` pre vyrolovanie obsahu.

Pri mobilnom zobrazení sa obsah hlavnej časti presunie a pridá do zoznamu ako dalšia možnosť s vyrolovaním obsahu.

7.2.6 Panel pre správcu

Modul `PageAdminPanel` predstavuje stránku s nastaveniami dostupnými pre správcu systému. Jednotlivé kategórie informácií, ktoré môže správca spravovať sú prezentované pomocou štvorcových dlaždičiek z modulu `SquareModule`.

Responzívne štvorčeky majú ikonku a názov zarovnaný do stredu. Po kliknutí na jeden zo štvorčekov sa užívateľ dostáva na stránku obsiahnutú v module `PageAdminTable`. Stránka obsahuje tabuľku, kde nájdeme informácie, ktoré môže správca ďalej meniť. Pri menšom zobrazení sa jednotlivé riadky tabuľky preskupia do zoznamu s kartičkami. Na nich je vedľa seba zobrazený názov stĺpca tabuľky a jej hodnota.

7.2.7 Servisné Požiadavky

`PageServiceRequestModule` exportuje komponent, reprezentujúci stránku s informáciami o servisných požiadavkách. V hornej časti stránky sa nachádzajú kartičky, predstavujúce komponent z modulu `PriorityCardModule`. Implementácia umožňuje nastaviť ikonku, jej farbu, nadpis a pomocou zoznamu, aj jednotlivé položky informácií na nej. Informácie predstavujú popis a k nej prislúchajúcu hodnotu. Na tejto stránke sa kartičky používajú k vyjadreniu počtu servisných požiadaviek, ktoré sa začali spracovávať, boli spracované a práve sa spracovávajú. Jednotlivé kartičky zobrazujú tieto informácie rozdelené podľa priority servisnej požiadavky.

Pod kartičkami sa nachádzajú dva grafy, znázorňujúce vyťaženosť jednotlivých servisných technikov. V komponente, definovanom v module `GraphModule` sa dajú nastaviť nadpis a samotné dáta. Nastavovať môžeme pomocou vlasností triedy komponent.

Pri menšom zobrazení sa kartičky preskupujú pod seba a zmenšujú sa na základe dostupného miesta. Grafy na mobilnom zariadení nie sú viditeľné z dôvodu, že by z nich užívateľ len ťažko vyčítal relevantné informácie.

7.2.8 Nastavenia pre užívateľa

Komponent stránky užívateľských nastavení je implementovaná v `PageUserProfileModule` moduli. Na počítačovom zobrazení je obsah rozdelený do kariet, respektívne tabov, cez ktoré sa môže užívateľ preklikať. Pre zobrazenie kariet využívam komponent `p-tabview` z knižnice PrimeNG. Na jednotlivých kartách `p-tabview` sa nachádzajú kolónky pre vstup od užívateľa. Tieto kolónky predstavujú komponent z modulu `InputBoxModule` rôznych typov. Každá karta má špecifický zoznam typov kolóniek, načítavajúcich zo zdieľaného súboru typu JSON. Tento súbor slúži na uchovávanie dát pre aplikáciu.

Na mobilnom zobrazení sa kategórie, ktoré sú v kartách zobrazujú v podobe responzívnych dlaždičiek s príslušnou ikonkou a názvom rovnakým spôsobom ako v moduli `PageAdminPanelModule`. Následne po rozkliknutí dlaždičky sa presmeruje na ďalšiu stránku, kde je zobrazený zoznam kolóniek ako v jednotlivých kartách na počítačovom zobrazení.

7.2.9 Navigačný panel

Všetky komponenty pre navigačný panel som implementovala v moduloch `NavbarModule` a `MobileTopBarModule`. `NavbarModule` obsahuje komponent bočnej navigačnej lišty na počítačovom zobrazení. Na navigačnom paneli je možné nastaviť zoznam jednotlivých položiek, logo aplikácie či fotku a meno užívateľa. Vysunutie pri prechode kurzorom je implementované pomocou CSS selektoru `:hover`.

Na mobilnom zobrazení je navigačná lišta zobrazená v strede obrazovky s položkami zarovnanými do stredu. Komponent z modulu `MobileTopBarModule` predstavuje hornú navigačnú lištu na mobilnom zobrazení. Tá obsahuje logo a ikonku znázorňujúcu menu, pre otvorenie navigačného panela. Taktiež je v nej zahrnutá aj ikonka šípky, zobrazujúcej pokiaľ je možnosť vrátiť sa späť.

Tieto moduly navzájom spolupracujú pri mobilnom zobrazení, kde udalosť `open`, vyvolaná v moduli `MobileTopBarModule`. Pri kliknutí na ikonku menu zobrazuje komponenty zahrnuté v `NavbarModule`. Tento modul vysiela udalosť `close`, slúžiacu na jej zatvorenie. Táto udalosť je spracovávaná v moduli `AppModule`.

7.3 Globálne štýly

Každý jeden komponent, ktorý som implementovala zdieľa niektoré globálne štýly definované v súbore `style.css`. Na základe požiadaviek od vývojárov z firmy Logimic som v tomto súbore definovala štýly, pre uľahčenie štýlovania elementov. Jednou z hlavných častí tohto súboru sú definície premenných pre farebnú paletu aplikácie. Definovala som si premenné pre primárnu farbu, sekundárnu farbu a farbu pre zvýraznenie elementu, pri prechode kurzorom. Tieto premenné sú použité v štýloch mojich komponentov a v preťažených štýloch z knižnice PrimeNG. Umožňujú jednoducho previesť zmenu farieb v celej aplikácii.

Ďalším dôležitým nastavením je farba textu a ikoniek. Nakoľko je možné meniť v tomto súbore celkový vzhľad aplikácie, musela som umožniť aj rýchlu zmenu farby textu a ikoniek. Týmto predchádzam problému, že by bolo treba v každom jednom súbore zvlášť prepisovať farby textov a ikoniek.

`Style.css` ďalej obsahuje globálne pomocné štýly pre jednoduchšie nastavenie často používaných vlastností, ako sú zarovnanie, nastavenie výšky či šírky prvku, nastavenie spôsobu jeho zobrazenia či zoradenia jeho obsahu. Ďalej v súbore definujem globálny štýl písma, veľkosti pre jednotlivé kategórie písma či velkosť ikoniek.

V súčasnosti firma Logimic ponúka systém pre viacerých zákazníkov. Vzhľad systému musí byť prispôsobený požiadavkám daného zákazníka. To dosahujú pomocou jedného CSS súbora, kde dokážu meniť všetky dôležité nastavenia. Zmeny v tomto súbore je následne možné vidieť v každej časti systému, čo zabezpečuje rýchlejšie prispôsobenie vzhľadu celkového systému. Pomáha to zachovávať konzistenciu užívateľského rozhrania a znižuje chybovosť pri vykonávaní zmien, nakoľko sa to nastavuje iba raz a nie je nutné meniť viacero súborov.

7.4 PWA

Systém firmy Logimic je dostupný ako webová aplikácia, ktorá je používaná na rôzne veľkých zariadeniach. Na základe toho, vznikla požiadavka prispôsobiť existujúce riešenie obrazovky mobilných zariadení. Pri výbere technológie, ktorá bude použitá, bolo zhodnotených niekoľko možností. Napriek tomu, že natívna alebo hybridná aplikácia by ponúkla väčší výkon a širšiu funkcionalitu, nakoniec sa rozhodlo pre PWA.

Pre PWA sa rozhodlo aj z niekoľkých dôvodov. Prvým je efektívne využitie už existujúceho systému, ktorý je založený na platforme Angular a ponúka možnosť vytvárania PWA. Implementácia PWA ďalej predstavuje cenovo aj časovo výhodnú možnosť oproti ostatným možnostiam. Ďalším dôvodom je, že nebude potrebná správa dvoch rôznych zdrojových kódov, ale je možné využiť už existujúci kód daného systému.

Implementácia progresívnej webovej aplikácie pomocou webového aplikačného rámca Angular zahŕňala pri vytváraní projektu pomocou *AngularCLI* použitie príkazu:

```
ng add @angular/pwa
```

Tento príkaz pridá balík `@angular/service-worker` do projektu a importuje ho do hlavného modulu aplikácie. Taktiež sa vygeneruje súbor `manifest.webmanifest` a modifikuje `index.html` súbor, kde naň pridá odkaz. Stiahne a uloží ikonky pre umožnenie inštalácie webovej aplikácie ako PWA a vytvorí konfiguračný súbor `ngsw-config.json`, špecifikujúci spôsob ukladania obsahu do vyrovnávacej pamäte.

Kapitola 8

Testovanie

Testovanie je proces, kedy sa snažíme nájsť chyby v našom produkte a zároveň overiť, či výsledok spĺňa všetky požiadavky, stanovené v kapitole 5.3. Tieto chyby nám dávajú možnosť stále zlepšovať náš produkt a udržiavať ho vždy aktuálny. Testovať svoj produkt vlastnoručne nie je vždy najlepší nápad. Častokrát neobjavíme veľké množstvo zásadných chýb. To je dôvod, prečo by sme mali zvoliť aj testovanie s užívateľmi. Vždy je lepšie mať produkt otestovaný čo najväčším počtom potencionálnych užívateľov, ktorí by v budúcnosti náš produkt mali alebo mohli používať.

Nad každým kladným aj záporným hodnotením zo strany užívateľov je potrebné sa zamyslieť. Dáva to priestor pochopiť, čo užívateľ vyžaduje a prispôsobiť mu to. Taktiež sa môže stať, že pri testovaní digitálneho prototypu bol užívateľ s funkcionalitou spokojný, ale akonáhle testoval finálny produkt, tak mu tam niečo chýbalo. Aj toto je príklad, prečo je potrebné testovanie produktu v rôznych fázach.

8.1 Testovanie aplikácie

Pri manuálnom testovaní sa manuálne prechádza náš produkt. Toto testovanie robí vývojár behom celého vývoja produktu a užívateľ pri ktorejkoľvek fáze testovania. Manuálne si vie otestovať digitálny prototyp aj finálny produkt. Avšak toto testovanie má aj nedostatky. Jeho prevedenie zaberá množstvo času, je nespoľahlivé, nakoľko žiadny človek nie je neomylný. Ku kladným vlastnostiam patrí jednoduchšie odhalenie dizajnových chýb ako napríklad zle vycentrovaný text, neprimeraná zmena prvkov na menších zariadeniach alebo pretekanie jednotlivých prvkov. Vieme ním odhaliť aj nedostatky z hľadiska použiteľnosti ako je neprimeraná veľkosť písma, zle zvolený font alebo nedostatočný kontrast farieb. V mojej práci som previedla manuálne testovanie.

Každé testovanie mojej mobilnej aplikácie prebiehalo počas pravidelných konzultácií s mojím vedúcim práce a taktiež s mojím konzultantom z firmy Logimic. Vďaka nim som vždy mala ďalšie pohľady a nové poznámky, ako by som svoju prácu mohla zlepšiť. V mojej práci som proces testovania používala od samotného začiatku.

8.1.1 Prvotné testovanie

Prvotné testovanie bolo pri vyhotovení digitálnych prototypov. Testovanie tohto prototypu bolo dôležitým krokom pred implementáciou samotnej aplikácie. Môj digitálny prototyp vytvorený v programe Adobe Xd som sprístupnila na testovanie aj firme Logimic. Mali

umožnené preklikať si jednotlivé stránky a vyskúšať aká bude interakcia užívateľa s aplikáciou.

Počas tohto testovania mi pribudli ďalšie požiadavky na aplikáciu ako napríklad pridanie nahrania profilovej fotky pre užívateľa, zmena hesla či pridanie stránky označenej ako L4, ktorá v sebe nesie ďalšie informácie. Ďalšou požiadavkou bola možnosť zmeny zobrazenia kruhových indikátorov s KPI na domovskej stránke. Jedna možnosť zahŕňa nový dizajn a druhou možnosťou je zachovanie pôvodného štýlu zobrazenia kruhových indikátorov, na ktorý sú koncoví užívatelia už zvyknutí. Po testovaní prototypu som následne do návrhu zakomponovala poskytnutú spätnú väzbu a návrhy na zlepšenie.

8.1.2 Testovanie počas implementácie

Po odsúhlasení digitálneho prototypu som začala samotnou implementáciou. Počas nej som manuálne testovala každú časť aplikácie pri rôznych rozmeroch obrazovky zariadenia. Tým som zisťovala aj hraničné rozmery šírky obrazovky, pri ktorých bolo potrebné zmeniť zobrazenie jednotlivých prvkov. Manuálne testovanie prebiehalo na lokálnej inštancii aplikácie spúšťanej pomocou príkazu `ng serve`.

8.1.3 Záverečné testovanie

Po dokončení väčšej časti implementácie lokálne bola aplikácia sprístupnená na verejnej URL adrese, pomocou ktorej som mohla aplikáciu otestovať na reálnych zariadeniach. V rámci tohto testovania na mobilných zariadeniach som odhalila niekoľko dizajnových chýb, ktoré bolo potrebné na aplikácii upraviť a doladiť, aby bolo mobilné zobrazenie užívateľsky prívetivé. Medzi nájdené chyby patrilo zlé centrovanie textu či zlé rozloženie navigačného panela pri otočení mobilu vodorovne.

Počas celej práce som sa snažila navrhnúť a implementovať jednotlivé komponenty tak, aby sa dali čo najviac prispôsobovať a využívať v rôznych kontextoch. Napriek tomu, mi pri finálnom testovaní aplikácie pribudli požiadavky na parametrizáciu mnohých komponentov. Dôvodom bolo, aby sa v rámci integrácie jednotlivých komponentov do súčasného systému zvýšila flexibilita a možnost ich prispôsobiť. Ako príklad môžem spomenúť skrytie horného riadku kartičiek na domovskej stránke, ktoré boli na ukážke vyššie 7.2.

Testovanie som previedla nielen na rôzne veľkých zariadeniach, ale aj na rôznych operačných systémoch či rôznych webových prehliadačoch. Otestované boli zariadenia s operačným systémom Android aj Apple.

Hotový dizajn a následne aplikácia dostupná na verejnej webovej adrese, bola predstavená a otestovaná dvoma typmi zákazníkov firmy Logimic. Išlo o zákazníkov venujúcich sa inteligentným mestám a so zameraním sa na sanitárnu elektroniku. Spätná väzba po testovaní od zákazníkov mala kladné ohlasy, na základe ktorých bola vyžiadaná postupná integrácia do reálneho systému firmy Logimic. Problémy, ktoré sa pri testovaní objavili súviseli s prehliadačom Safari.

8.2 Integrácia mojich modulov do súčasného systému

Po implementácii modulov a dokončení testovania aplikácie bola sprístupnená nasadením na AWS[1] a ďalej dostupná na verejnej URL adrese. Ďalej mohla nasledovať integrácia modulov do systému firmy Logimic. Integrácia predstavuje dôležitú časť tejto práce. Overí

[1] https://aws.amazon.com/

kvalitu a použiteľnosť mojich vytvorených modulov a ich správanie pri nasadení do iného systému. Dôležitým krokom pred samotným začiatkom je naštudovanie si aplikačnej logiky súčasného systému a jej celkovej architektúry.

8.2.1 Priebeh integrácie

Mojou prvou úlohou bola integrácia hlavnej stránky, zobrazujúcej sa hneď po úspešnom prihlásení sa do systému. Hlavná stránka je zobrazená na obrázku 7.2. Komponenty z modulov, ktoré nevyužívajú žiadnu doplňujúcu knižnicu bolo možné bez problémov integrovať do systému. Problém s nimi nenastal ani po prepojení s aplikačnou logikou systému. Taktiež si zachovali aj responzívnosť, ktorá je hlavným cieľom tejto práce.

Naopak niektoré komponenty z modulov využívajúcich knižnicu PrimeNG sa nezobrazili správne, z dôvodu odlišnosti používanej verzie knižnice. Systém firmy Logimic používal nižšiu verziu, nepodporujúcu niektoré z využívaných vlastností v mojich moduloch. Vyskytnutie problému v odlišnosti verzií knižníc a jeho následné riešenie navýšením verzií, znamená vo všeobecnosti náročný proces. Nakoľko je nutné skontrolovať správnu funkcionalitu všetkých ostatných modulov, ktoré sú na danej verzii knižnice závislé. V mojom prípade, navýšenie verzie knižnice PrimeNG, taktiež vyžadoval navýšiť verziu webového aplikačného rámca Angular. Následne bolo možné pokračovať v integrácii mojich modulov. Proces integrácie bude prebiehať aj po dokončení bakalárskej práce firmou Logimic.

Kapitola 9

Záver

Cieľom tejto práce bolo vytvorenie Angular modulov predstavujúcich jednotlivé elementy mobilnej aplikácie pre správu zariadení v IoT. Tieto moduly sú v súčasnej dobe integrované do reálneho systému firmy Logimic.

V teoretickej časti som spracovala problematiku Internetu vecí a zobrazovania dát na dashboardoch. Ďalej som si naštudovala proces vytvorenia progresívnej webovej aplikácie a jej výhody. Analyzovala som súčasný stav webovej aplikácie a jej správanie pri zapnutí mobilného zobrazenia. Identifikovala som nedostatky súčasného systému. Tie boli neskôr využité pri návrhoch. Prešla som procesom vytvorenia viacerých náčrtov. Následne som vytvorila štyri rôzne prototypy, z ktorých sme spolu s konzultantmi vybrali jeden. Na ňom som v zápätí vylepšovala jednotlivé časti, pridávala funkcionalitu a spracovávala nové požiadavky či pripomienky na zmenu.

V rámci implementácie som vytvorila progresívnu webovú aplikáciu, pomocou webového aplikačného rámca Angular s využitím knižnice PrimeNG pri niektorých prvkoch. Výsledná aplikácia bola prezentovaná a schválená firmou Logimic, na základe testovania s ich zákazníkmi z odvetvia inteligentných miest či sanitárnej elektroniky. Následne bola vyžiadaná integrácia mojich modulov do systému firmy Logimic. Počas celej implementácie som využívala manuálne testovanie a na záver som previedla testovanie aj na rôznych operačných systémoch.

Hlavným výstupom celej mojej práce sú jednotlivé Angular moduly, z ktorých je aplikácia zložená. Tieto moduly predstavujúce prvky užívateľského rozhrania je možné prispôsobiť a budú použité na vylepšenie súčasného systému od firmy Logimic.

Dokončením mojej práce sa však práca na celej aplikácii nekončí. Ďalším krokom je dokončenie integrácie mojich modulov do reálne nasadeného systému firmy Logimic. Tento proces si vyžaduje ďalšie testovanie a mapovanie vytvorených modulov na aplikačnú logiku systému. Na niektoré moje moduly je potrebné vytvoriť nové koncové body na aplikačné programové rozhranie a rozšíriť databázu.

Literatúra

[1] Ahuja, S., Potti, P. et al. An introduction to RFID technology. *Commun. Netw.* 2010, zv. 2, č. 3, s. 183–186.

[2] Al Alawi, A. I. WiFi technology: Future market challenges and opportunities. *Journal of computer science*. Citeseer. 2006, zv. 2, č. 1, s. 13–18.

[3] Ammar, M., Russello, G. a Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*. Elsevier BV. feb 2018, zv. 38, s. 8–27. DOI: 10.1016/j.jisa.2017.11.002. Dostupné z: https://doi.org/10.1016%2Fj.jisa.2017.11.002.

[4] Ashton, K. et al. That 'internet of things' thing. *RFID journal*. 2009, zv. 22, č. 7, s. 97–114.

[5] Borgia, E. The Internet of Things vision: Key features, applications and open issues. *Computer Communications*. Elsevier. 2014, zv. 54, s. 1–31.

[6] Brath, R. a Peters, M. Dashboard design: Why design is important. *DM Direct*. 2004, zv. 85, s. 1011285–1.

[7] Dam, R. a Siang, T. Define and Frame Your Design Challenge by Creating Your Point Of View and Ask "How Might We". *URL Httpswww Interact.-Des. Orgliteraturearticledefine–Frame-Your-Des.-Challengeby-Creat.-Your-Point–View–Ask–Might-We*.
2017.

[8] Domes, S. *Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase*. Packt Publishing Ltd, 2017.

[9] Eckerson, W. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. Wiley, 2005. ISBN 9780471757658. Dostupné z: https://books.google.cz/books?id=rCt-FVy8PvcC.

[10] Few, S. *Information dashboard design: The effective visual communication of data*. O'reilly Sebastopol, CA, 2006.

[11] Frustaci, M., Pace, P., Aloi, G. a Fortino, G. Evaluating critical security issues of the IoT world: Present and future challenges. *IEEE Internet of things journal*. IEEE. 2017, zv. 5, č. 4, s. 2483–2495.

[12] Garrett, J. J. *The elements of user experience: user-centered design for the web and beyond*. Pearson Education, 2010.

[13] Greengard, S. *The internet of things.* MIT press, 2021.

[14] Hajian, M. *Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular.* Apress, 2019.

[15] Johnson, J. *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules.* Morgan Kaufmann Publishers Inc., 2010.

[16] Joo, H. A study on understanding of UI and UX, and understanding of design according to user interface change. *International Journal of Applied Engineering Research.* 2017, zv. 12, č. 20, s. 9931–9935.

[17] Keramidas, G., Voros, N. a Hübner, M. *Components and Services for IoT Platforms.* Springer, 2016.

[18] Knapp, J., Zeratsky, J. a Kowitz, B. *Sprint: How to solve big problems and test new ideas in just five days.* Simon and Schuster, 2016.

[19] Lee, J., Kim, H., Park, J., Shin, I. a Son, S. Pride and prejudice in progressive web apps: Abusing native app-like features in web applications. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.* 2018, s. 1731–1746.

[20] Lombardi, M., Pascale, F. a Santaniello, D. Internet of Things: A General Overview between Architectures, Protocols and Applications. *Information.* Multidisciplinary Digital Publishing Institute. 2021, zv. 12, č. 2, s. 87.

[21] Magomadov, V. Exploring the role of progressive web applications in modern web development. In: IOP Publishing. *Journal of Physics: Conference Series.* 2020, sv. 1679, č. 2, s. 022043.

[22] Mehmood, Y., Ahmad, F., Yaqoob, I., Adnane, A., Imran, M. et al. Internet-of-things-based smart cities: Recent advances and challenges. *IEEE Communications Magazine.* IEEE. 2017, zv. 55, č. 9, s. 16–24.

[23] Ramasamy, L. K. a Kadry, S. Internet of things (IoT).

[24] Rojas, C. *Building Progressive Web Applications with Vue. js: Reliable, Fast, and Engaging Apps with Vue. js.* Apress, 2019.

[25] Sobin, C. A survey on architecture, protocols and challenges in IoT. *Wireless Personal Communications.* Springer. 2020, zv. 112, č. 3, s. 1383–1429.

[26] Steinke, G. H., Al Deen, M. S. a LaBrie, R. C. Innovating information system development methodologies with design thinking. In: Bibliothek, Hochschule Anhalt. *Titel: Proceedings of the 5th Conference in Innovations in IT, Volume Nr. 5.* 2018.

[27] Suresh, P., Daniel, J. V., Parthasarathy, V. a Aswathy, R. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In: IEEE. *2014 International conference on science engineering and management research (ICSEMR).* 2014, s. 1–8.

[28] Tandel, S. a Jamadar, A. Impact of progressive web apps on web app development. *International Journal of Innovative Research in Science, Engineering and Technology*. 2018, zv. 7, č. 9, s. 9439–9444.

[29] Tomar, A. Introduction to ZigBee technology. *Global Technology Centre*. 2011, zv. 1, s. 1–24.