Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Smart Space

Dokumentácia riadenia projektu

Inžinierske dielo

Tím: 17 - Smart Space
Vedúci tímu: Ing. Peter Kaňuch
Vyhotovil: Jakub Dubec, Milan Kaprál, Tomáš Kovalčík, Jakub Kubica, Oleksandr Lytvyn, Andrej Petričko

Dátum poslednej zmeny: 26.11.2020

Úvod

Počet zariadení pripojených do internetu každým rokom neustále rastie a nejedná sa o len o počítače ako kedysi. V dnešnej dobe sú do internetu pripojené zariadenia z každodenného života ako sú spotrebiče v domácnostiach, chladničky, umývačky riadov, klimatizácie, svetlá a hromada iných zariadení. Vec ktorá je dnes úplne obyčajná, taká ktorá okrem svojej úlohy nič iné nerobí môže o pár rokov byť úplne neobyčajná. Priestor kde sa takéto múdre zariadenia nachádzajú sa nazýva internet vecí (z angl. Internet of things, v skratke IoT). Internet vecí opisuje sieť, ktorá spočíva z navzájom prepojených zariadení, zložených z špeciálneho hardvéru a softvéru, vďaka ktorému dokážu medzi sebou komunikovať podľa vopred určeného protokolu. Táto nová vznikajúca technológia prináša nové možnosti ako efektívnejšie a lepšie využívať zariadenia okolo nás.

Tento projekt je zameraný na IoT riešenie ktoré bude umožňovať efektívnejšie využívanie spoločných priestorov. Od počtu ľudí v miestnosti až po kvalitu vzduchu v miestnosti, to všetko bude výsledkom nášho projektu SmartSpace.

Dokumentácia riadenia projektu

Členovia tímu a ich role

Jakub Dubec

- Odbor: Internetové technológie
- Rola: Team Leader, Technical Leader
- Obľúbená oblasť IT: IoT, Linux, Python

Volám sa Jakub Dubec, pochádzam z Martina, informatike sa venujem už od základnej školy. Na strednej škole som sa venoval vývoju, školského informačného systému (ktorý bol aj následne medzinárodne ocenený - Vernadsky national competition 2013, ISS 2014). Na vysokej škole som sa začal venovať IoT (spolupracujem na projektu komunitného merania ovzdušia). Už piaty rok pracujem vo firme BACKBONE s.r.o., momentálne na pozícii software architekt. Medzi moje obľúbené programovacie jazyky patrí Python a C++. Vo voľnom čase sa venujem open-source projektom (napísal som niekoľko Python knižníc).

Jakub Kubica

- Odbor: Internetové technológie
- Rola: Backend/Frontend Software developer
- Obľúbená oblasť IT: Sieťové technológie

Volám sa Jakub Kubica. Informačným technológiám som sa začal venovať už od strednej školy. Na strednej škole som sa venoval programovaniu Java a C. Taktiež som pracoval s Arduinom a rôznymi modulmi na viacerých projektoch. Na vysokej škole som sa začal venovať sieťam a programovaniu. Konkrétne Python. Vo voľnom čase sa venujem Cisco certifikacií a nadobúdanie vedomosti v sfere sieti.

Milan Kaprál

- Odbor: Internetové technológie
- Rola: Backend/Frontend Software developer, hardver guru
- Obľúbená oblasť IT: IoT, Sieťové technológie

Volám sa Milan Kaprál. Informačné technológie ma sprevádzajú už od strednej školy, kde som sa naučil čo to programovanie je. Programovali sme v jazyku C a Java. Na strednej škole som nabral znalosti ohľadom sieťových technológií. Na vysokej škole som sa začal aktívnejšie venovať programovaniu v jazyku C. Popri bakalárskej práci som nadobudol vedomosti s prácou s dátami a jazykom Python.

Oleksandr Lytvyn

- Odbor: ISS
- Rola: Backend/Frontend Software developer
- Obľúbená oblasť IT: Data Engineering, Backend development

Volám sa Oleksandr Lytvyn, som z Ukrajiny. Informatike, programovaniu a sieťam som začal venovať iba na vysokej škole. Hoci som úspešne absolvoval bakalárske štúdium na FEI TU v Košiciach v odbore počítačové siete, ale v Ing. studiu som sa rozhodol pokračovať v smere ktorý je viac orientovaný na návrh a vývoj softvéru. Ako bakalársku prácu som vytváral rečovú aplikáciu pre Google Asistenta

Andrej Petričko

- Odbor: ISS
- Rola: Backend/Frontend Software developer
- Obľúbená oblasť IT: Neurónové siete, Umelá inteligencia, Spracovanie obrazu

Moje meno je Andrej Petričko a som z Martina. Informatike som sa začal venovať na strednej škole, keďže som mal blízko k počítačovým technológiám. Bohužiaľ na strednej škole sme aj po vybratí informatiky ako maturitného predmetu nemali učivo na nejak vysokej úrovni. Hlbšie do informatiky som sa dostal až na vysokej škole. Popri nej som pri bakalárskom štúdiu pracoval ako backend sw. developer v jazykoch ako python, javascript a php. Najviac z IT ma zaujíma oblasť umelej inteligencie, v ktorej som mal aj bakalársku prácu.

Tomáš Kovalčík

- Odbor: Internetové technológie
- Rola: Backend/Frontend Software developer, Scrummaster
- Obľúbená oblasť IT: Python, Sieťové technológie, Linux, Bezpečnosť

Informatike som sa začal venovať až na vysokej škole. Mám rád programovanie v Pythone ale nebojím sa ani kompilovaných jazykov. Okrem toho mám záľubu v sieťových technológiách. Za posledný rok som nadobudol skúsenosti potrebné na prácu v tíme, kde pracujeme na pomerne veľkom projekte od firmy Cisco. Moje vedľajšie záľuby sú šport a pivo s kamarátmi.

Motivačný dokument

Náš tím má prevažne skúsenosti s programovacími jazykmi ako je Python a C/C++, čo nám dáva ideálny základ na tvorbu natívnych alebo serverových aplikácií. Väčšina z nás má skúsenosti aj POSIX operačnými systémami (prevažne Linux), vďaka čomu vieme riešiť aj konfigurácie serverov (nakoľko Linux momentálne kraľuje tejto oblasti). Členovia nášho tímu sa stretli s relatívne širokov škálou projektov (od vnorených systémov, cez mobilné aplikácie, CISCO riešenia až po strojové učenie). Nebojíme sa ani ukladať dáta na do databáz (PostgreSQL alebo MongoDB). Väčšina z nás má skúsenosti skôr s back-end riešeniami alebo s počítačovými sieťami (CISCO). Sme tak ideálny tím pre tvorbu IoT riešení, prípadne riešení na báze blockchain. Väčšina z nás sa stretla s informatikou už na strednej škole, tak nepatríme medzi “začiatočníkov” a staviame na dlhoročných skúsenostiach.
Flexibilitu nášho tímu dotvára aj fakt, že sme vznikli fúziou dvoch Ing. odborov: IT a ISS, jeden člen nás obohacuje skúsenosťami z FEI TUKE. Veríme, že tak dokážeme vytvárať kreatívnejšie riešenia a postupy.

- efektivne & stručne
- kamaráti POSIXu
- low-level & high-level programovacie jazyky
- fuzia (sieťari, informatici)
- široká škála projektov (súkromných aj profesionálnych)
- skúsenosti z praxe

Zvolená téma: **Transformácia priestorov na bezpečné a inteligentné miesta na prácu [SmartSpace]**

Túto tému sme si vybrali na základe skúsenosti a zloženia našeho tímu. Sme kombinácia sieťarov a informatikov, vieme tak tvoriť kompletné IoT riešenia, ktoré majú hlavu a pätu (ako sa hovorí u nás na salaši). Máme skúsenosti s POSIX operačnými systémami, čo vyhovuje požiadavke na odporúčanú technológiu (práca s operačným IoT systémom RIOT). Každý člen nášho tímu má skúsenosti s programovacím jazykom Python, ktorý je ideálne na implementáciu aplikačných rozhraní, ktoré bude riadiť zariadenia v IoT sieti. V našom tíme ľudí ktorí absolvovali predmety ako mikropočítače, alebo majú zapísané vnorené systémy (rozumieme tak tvorbe a návrhu IoT aplikácií).

Metodiky

Komunikačný kanál

Ako komunikačný kanál sme si zvolili platformu Slack, ktorú využívame na komunikáciu pri pracovaní na tomto projekte. V slacku sme si vytvorili viacero kanálov, každý kanál je určený na špecifickú časť projektu aby sme mohli čo najefektívnejšie pracovať.

Google Calendar & Google Meet

Pomocou nástroja Google Calendár je vytvorená udalosť pre online stretnutie. Túto udalosť vytvorí ScrumMaster v dni, kedy máme naplánované stretnutie (Pondelok, Štvrtok, Piatok). ScrumMaster automaticky pozve do tejto udalosti všetkých členov tímu, vrátane vedúceho projektu. Google Calendar pri vytvorení udalosti taktiež vytvára miestnosť pre videohovory, ktorá je realizovaná pomocou nástroja Google Meet.

Toto riešenie poskytuje vysoký komfort z dôvodu, že aplikácia GoogleMeet automaticky upozorní všetkých účastníkov vytvorenej udalosti niekoľko minút pred začiatkom

Nástroj Google Meet poskytuje dostatočnú kvalitu pre video a audio hovory. Taktiež poskytuje jednoduché a intuitívne pracovanie so zdieľaním obrazovky a možnosti spoločného četu.

Slack

Pre komunikáciu v tíme prostredníctvom textových správ využívame Aplikáciu Slack. Toto riešenie poskytuje jednoduchú komunikáciu, ktorá obsahuje vlastnosti ako:

- Uchovávanie predošlých konverzácií
- Zdieľanie multimediálnych súborov
- Možnosť označiť konkrétnu osobu
- Možnosť písať skupinovo alebo individuálne

Aplikácia poskytuje možnosť vytvorenia kanálov, ktoré majú za úlohu oddeliť konverzácie podľa kategórií. Pre náš tím sú vytvorené tieto kanály:

1. ***# backend*** - kanál, ktorý slúži pre komunikáciu členov pracujúcich na backende
2. ***# frontend*** - kanál, ktorý slúži pre komunikáciu členov pracujúcich na frontende
3. ***# general*** - kanál pre všeobecnú komunikáciu v tíme medzi členmi. V tomto kanále prebiehajú všetky konverzácie, ktoré nepatria do ostatných
4. ***# iot*** - kanál, ktorý slúži pre komunikáciu členov pracujúcich s IoT
5. ***# meetings*** - kanál, ktorý je vyhradený pre komunikáciu ohľadom stretnutí
6. ***# random*** - kanál, ktorý bol vytvorený za účelom komunikácie členov time ohľadom vecí, ktoré sa netýkajú projektu SmartSpace
7. ***# smart-space***- kanál, ktorý slúži predovšetkým na komunikáciu vedúceho projektu s členmi tímu. Vedúci projektu do tohto kanála zadáva neformálne veci týkajúce sa projektu SmartSpace

Ako kanban tabuľu pre metódu Scrum využívame Trello, kde sme si pridali úlohy, ktoré si rozdeľujeme na prehľadnej nástenke.

Testovanie

Testovanie vykonáva každý. Každá naprogramovaná funkcionalita musí mať obsahovať testy. Kód, ktorý nemá testy sa poklada za nefunkčný a nedôveryhodný.

Práca s gitom

Code review

Code review realizujeme v prostredí Gitlab, kde máme viacero repozitárov, určených na prácu v tomto projekte. Celkovo máme momentálne štyri repozitáre.
Code review robia vždy dvaja okrem samotného developera ktorý kód zverejnil.

Verziovanie

Verziovanie kódu a rozdelenie úloh na menšie celky nám v tíme pomáha udržiavať si poriadok kóde a úlohách, ktoré boli vykonané členmi tímu. Verzie kódu uchovávame v niekoľkých vetvách, ktoré sú vysvetlené nižšie.

Branches

master – je to hlavná vetva, v ktorej sa nachádza taká verzia softvéru, ktorá je otestovaná a schválená členmi, ktorý sa podieľali na code review kódu. Do tejto vetvy sa merguju verzie softvéru z ostatných vetiev.

ostatné – slúžia na verziovanie kódu, ktorý sa rieši v aktuálnom šprinte a nie je úplne dokončený.

Pre tieto vetvy platí pravidlo, že môžu existovať pod týmto názvom:

- SMART-[ID]-[názov úlohy], kde:
  - **SMART** – rovnaký názov, ktorý prezentuje projekt SmartSpace, píše sa na začiatku každej vetvy
  - **[ID]** – je to identifikačné číslo úlohy, ktoré je rovnaké ako číslo úlohy v nástroji Trello.
  - **[názov úlohy]** – reprezentuje názov úlohy, ktorý je rovnaký ako názov úlohy v nástroji Trello. Korešponduje s ID úlohy. Pri slovenskom znení úlohy sa píše bez diakritiky. Viacero slov v názve oddeľujeme znakom „-„

- Príklad: SMART-50-pridelenie-device-endpoint

Pull request

Každý pull (merge) request je vykonávaný pri ťahaní zmeny z jednej vetvy do druhej. V tomto prípade to je z ostatných vetiev do master vetvy. Pred vykonaním každého pull requestu sa musí vykonať code review, ako je definované vyššie.

Commit

Každý commit, ktorý je vykonaný obsahuje jednoduchú správu a ak je to potrebné tak sa ku commitu pridá popis, ktorý ho jasne definuje.

Zápisnice

Každý pondelok na meetingu píšeme zápisnicu. Pri písaní sa meníme každý týždeň (metodika round robin). Zápisnicu následne nahrávame do Slacku. Tieto konkrétne zápisnice je si možné pozrieť na našej webovej stránke v sekcii Documents.

Pri vytváraní zápisníc v tíme SmartSpace sú definované tieto pravidlá:

- na vytváranie zápisníc nie je poverený žiadny člen tímu
- vytváranie zápisníc ma na starosti každý člen tímu
- členovia sa striedajú každý šprint podľa priezviska vzostupne
- zápisnica sa vytvára na konci aktuálneho a začiatku nasledujúceho šprintu
- pri písaní zápisnice je každý člen tímu odprezentuje svoju prácu
- pri písaní zápisnice sa zapisuje najprv meno člena v odrážkach sa ku každému členovi vypíše jeho progres v aktuálne končiacom šprinte
- na konci zápisnice sa píšu poznámky vedúceho projektu

Postup pri písaní zápisnice:

1. na začiatku mítingu sa overí, kto bude zápisnicu zapisovať
2. člen zapisuje poznámky do voliteľného nástroja, ktorý uzná za vhodný
3. člen vytvorí nový odstavec začínajúci menom člena, ktorý prezentuje svoj progres
4. v odrážkach sa zapíše čo aktuálne prezentujúci člen za šprint vykonal
5. postup sa opakuje od kroku č.3, až kým odprezentujú všetci členovia tímu
6. na konci člen zapíše poznámky od vedúceho projektu

Definition of done

Pre definovanie používateľského príbehu s označením „hotový“ je potrebné, aby spĺňal nasledujúce kritéria:

- úlohy, ktoré boli definované a schválené sú dosiahnuté
- bol riadne zdokumentovaný so všetkými zmenami, ktoré boli počas riešenia spravené
- boli vykonané všetky potrebné testovania, aby sa overila úplná funkcionalita
- bolo vykonané Code reviews členmi tímu
- bol odprezentovaný členom tímu
- bol odprezentovaný vedúcemu projektu
- bol schválený vedúcim projektu

Pri nasadzovaní a integrovaní používateľského príbehu do systému je potrebné splniť nasledujúce kritéria:

- dokumentácia musí obsahovať:
    - komentáre v zdrojovom kóde
    - popis, akú úlohu daný používateľský príbeh vykonáva
    - aké sú požiadavky pre správne fungovanie používateľského príbehu

- používateľský príbeh musí byť pushnutý na Git
- finálne schválenie vedúceho tímu

Definition of ready

Používateľský príbeh (user story) je považovaný za pripravený v prípade, že sú splnené nasledovné kritériá:

- vedúci projektu jasne definoval všetky požiadavky.
- jednoznačne sa určí člen/členovia tímu zodpovedný/í za vykonanie používateľského príbehu
- boli prediskutované všetky kritéria a povinnosti s členmi tímu, ktorý sa podieľajú na vykonaní daného používateľského príbehu
- boli odprezentované kritéria ostatným členom tímu
- boli stanovené výstupy, ktoré má používateľský príbeh po dokončení vykonávať
- boli stanovené testy, ktoré overia funkčnosť používateľského príbehu

Kanban tabuľa

Pre zapisovanie používateľských príbehov a športových požiadaviek je používaný nástroj Trello. Tento nástroj využíva kanban tabuľu, v ktorej sa rozdeľujú úlohy v prehľadnej nástenke.

Na to aby mohli byť používateľské príbehy zaradené do šprintu a získali označenie „pripravený“ musia byť splnené nasledovné kritéria:

- bol jasne definovaný cieľ podľa vyššie uvedených bodov
- bol vytvorený záznam v Trello
- záznam v Trello obsahoval informácie:
  - riešiteľ/riešitelia
  - opis požiadavky používateľského príbehu
  - výstup používateľského príbehu

Sumarizácie šprintov

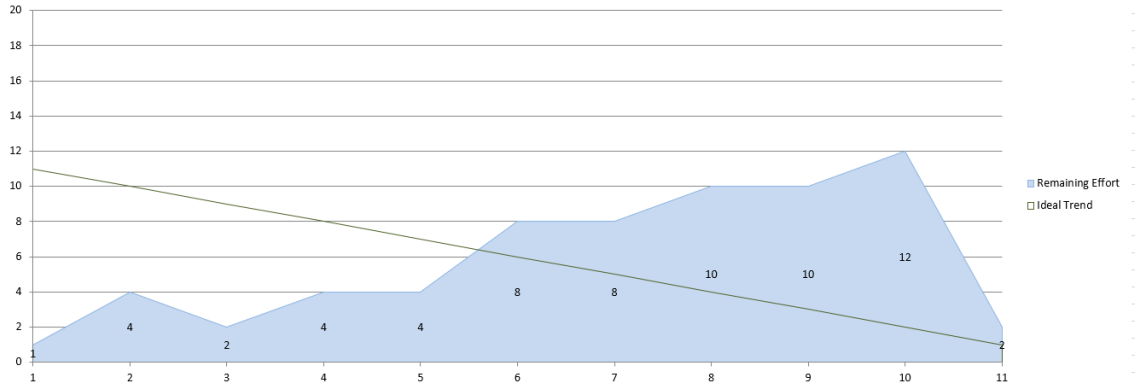
Šprint č. 1

Export z Trella

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
35	Webová stránka	In progress	Andrej Petričko	15h
36	Analýza jazyka frameworku	Hotovo	Jakub Kubica	16h
37	Analýza a výber databázového systému	Hotovo	Oleksandr Lytvyn	10h
38	Analýza HW	Hotovo	Milan Kaprál Jakub Dubec	15h
39	M2M protocols	Hotovo	Jakub Dubec	4h
40	Analýza CUPS a serverových služieb	Hotovo	Tomáš Kovalčík	10h
41	Analýza workera	Hotovo	Tomáš Kovalčík	10h

Burndown graf

Graf zobrazuje priemernú prácu členov tímu počas pracovných dní od začiatku sprintu od prvého dňa po jedenásty deň. Tento graf ukazuje ze prvý šprint nedopadol podľa predstáv, robili sme skôr v poslednom týždni. Od pomyselnej idealnej krivky sme uplne ušli. Tento graf odzrkadľuje aj to aké náročne bolo pre jednotlivých členov nabehnúť na metódu scrum. Možno je to aj tým že ako študenti sme zvyknutí robiť veci na poslednú chvíľu. Aj to je jedna z vecí, ktoré sa musí človek naučiť pri agilnom vývoji. A to je pracovať každý deň podľa plánu.



Graf č. 1

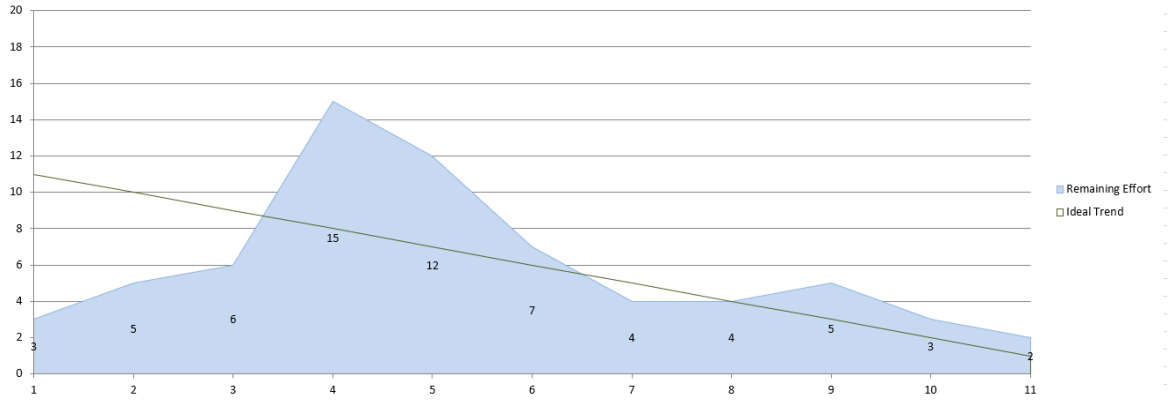
Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	2	9h
Milan Kaprál	1	10h
Tomáš Kovalčík	2	20h
Jakub Kubica	1	16h
Oleksandr Lytvyn	1	10h
Andrej Petričko	1	15h

Šprint č. 2

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
35	Webová stránka projektu	Hotovo	Andrej Petričko	6h
42	Nastavenie servera, služby, docker	Hotovo	Tomáš Kovalčík	20h
43	Základná Django aplikácia	Hotovo	Jakub Kubica	35h
44	Základná React aplikácia	Hotovo	Andrej Petričko	25h
45	Analýza displeja	Hotovo	Milan Kaprál	5h
46	Návrh databázy a dátového modelu	Hotovo	Oleksandr Lytvyn Jakub Dubec	12h
48	Analýza OS pre IoT	Hotovo	Milan Kaprál	10h
50	Schéma zapojenia koncových zariadení	Hotovo	Milan Kaprál	15h
51	CI-CD pipeline	Dokončená ale bude sa ešte upravovať	Tomáš Kovalčík	20h

Burndown graf
Tento graf ukazuje, že druhý šprint nie je úplne ideálny avšak je lepší ako z prveho sprintu, začali sme pracovať počas šprintu takmer od začiatku.



Graf č. 2

Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	1	4h
Milan Kaprál	3	30h
Tomáš Kovalčík	2	40h
Jakub Kubica	1	35h
Oleksandr Lytvyn	1	12h
Andrej Petričko	2	30h

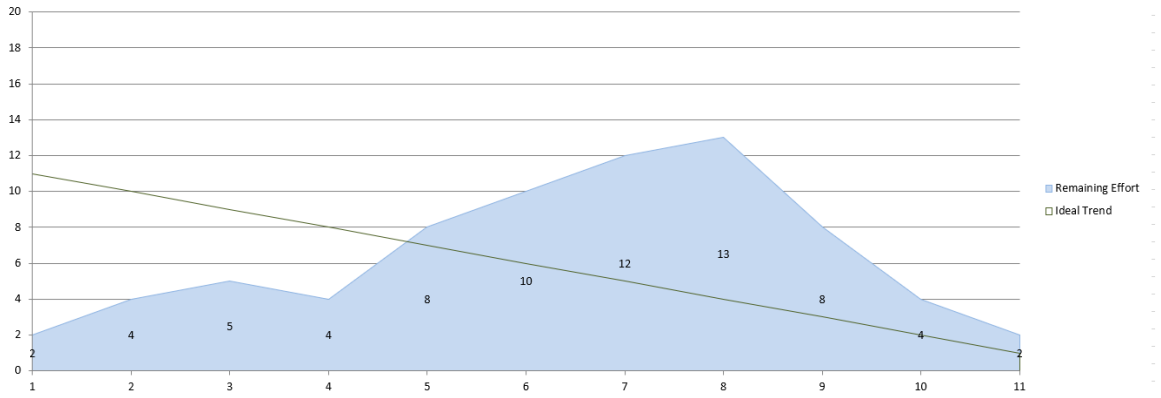
Šprint č. 3

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
47	Návrh api		Andrej Petričko, Jakub Dubec, Jakub Kubica	30h
49	Výber OS	Hotovo	Jakub Dubec	15h
54	API dokumentacia (Swagger)	In progress	Oleksandr Lytvyn	10h
55	(US1.1) fronted - prihlasenie	In progress	Andrej Petričko	15h

	backend - handlovat prihlasenie			
58	(US1.1) frontend - pridanie / odstranenie, backend - handlovat pridávanie, odstranovanie - nového typu zariadenia	Backend je hotový. Frontend nebol pochopený ako si to product owner predstavoval. Praca je presunutá do 4. šprintu	Jakub Kubica	15h
59	(US1.2) frontend - možnosť vybrať typ zariadenia a vedieť zadať počet inštancii. backend - vytvorit X inštancii daného typu	Backend je hotový. Frontend nebol pochopený ako si to product owner predstavoval. Praca je presunutá do 4. šprintu	Tomáš Kovalčík	15h
60	Návrh aplikačného protokolu pre MQTT devices (komunikácia medzi MQTT Broker a DB)	In progress	Milan Kaprál	12h
61	Model v Djangu	Hotovo	Jakub Kubica Tomáš Kovalčík	10h

Burndown graf

Tento graf ukazuje ze tretí šprint, bol vskutku najlepší zo všetkých. Mali sme tam pár problémov a pri niektorých úlohách sme sa zbytočne zdržali. Ideálna krivka nízko z dôvodu, pretože členovia tímu riešili problémy, ktoré zabrali veľa času, ktoré sme nevedeli vopred odhadnúť.



Graf č. 3

Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	2	45h
Milan Kaprál	1	12h
Tomáš Kovalčík	2	25h
Jakub Kubica	3	55h
Oleksandr Lytvyn	1	10h
Andrej Petričko	2	45h

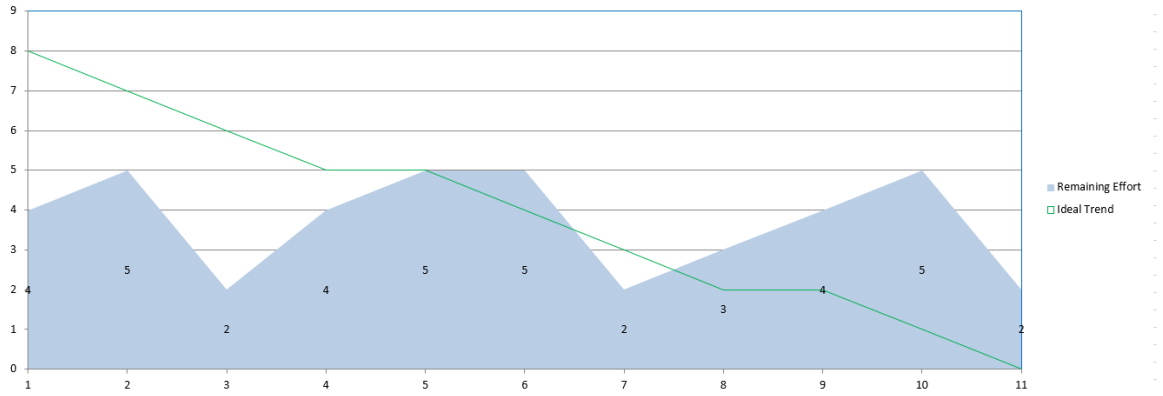
Šprint č. 4

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
54	API dokumentacia (Swagger)	Hotovo	Oleksandr Lytvyn	8h
55	(US1.1) fronted - prihlasenie backend - handlovat prihlasenie	In progress Backend je hotový	Andrej Petričko	5h

58	(US1.1) frontend - pridanie / odstránenie, backend - handlovať pridávanie, odstranovanie - nového typu zariadenia	Hotovo	Jakub Kubica	10h
59	(US1.2) frontend - možnosť vybrať typ zariadenia a vedieť zadať počet inštancii. backend - vytvorit X inštancii daného typu	Hotovo	Tomáš Kovalčík	10h
60	Návrh aplikačného protokolu pre MQTT devices (komunikácia medzi MQTT Broker a DB)	Hotovo	Milan Kaprál	5h
62	Spísanie mílnika	Hotovo	celý tím	5h
63	Update stránky	Hotovo	Andrej Petričko, Tomáš Kovalčík	3h

Burndown graf

Tento graf ukazuje, že tento posledný šprint bol trošku uskákaný. Tento jav, ktorý je možné pozorovať preto, pretože ku koncu semestra sme sa sustredili aj na iné predmety, ktoré mali svoje termíny, ktoré sa pretínali z tímovým projektom. Aj kvôli tomuto nedostatku v našom tíme si myslíme, že sme obstáli veľmi dobre. Výsledky, ktoré z tohto šprintu vzišli splnili očakávania a celkovo boli funkčné.



Sumár podielu práce

Meno	Počet taskov	Celkový čas
Jakub Dubec	2	25h
Milan Kaprál	1	12h
Tomáš Kovalčík	2	25h
Jakub Kubica	2	25h
Oleksandr Lytvyn	1	12h
Andrej Petričko	1	15h

Šprint č. 5

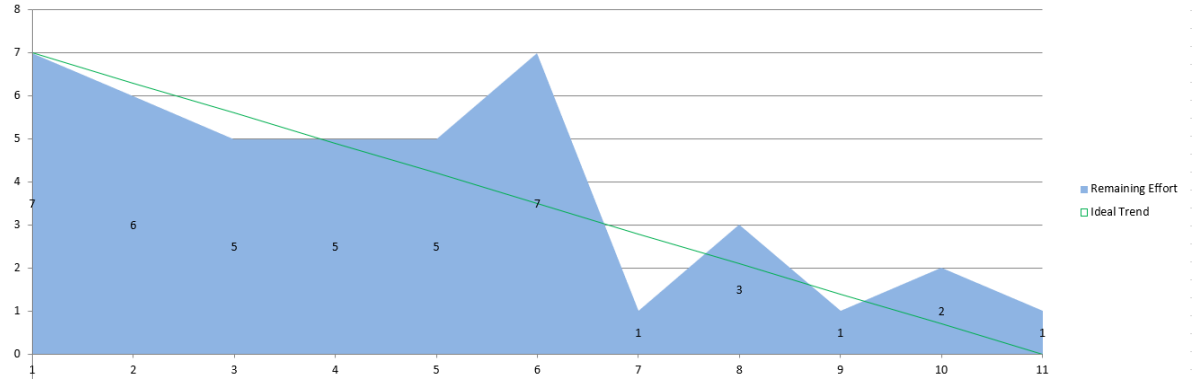
Export z Trella

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
55	(US1.1) fronted - prihlasenie backend - handlovat prihlasenie	Hotovo	Andrej Petričko	15h
69	reactjs -	Hotovo	Jakub Kubica	16h

	zosuladit verzie balikov			
77	analyza ako rozbehat komunikaciu medzi frontendom a backendom na produkcnom serveri	Hotovo	Tomáš Kovalčík	10h
68	Dokoncit - REST API specification	Hotovo	Tomáš Kovalčík Olek	10h
76	NGINX - reverse proxy na frontend	Hotovo	Tomáš Kovalčík	10h
72	definition of ready metodika	Hotovo	Milan Kaprál	10h
75	pisanie zápisníc metodika	Hotovo	Milan Kaprál	8h
73	praca s gitom metodika	Hotovo	Milan Kaprál	12h
71	definition of done metodika	Hotovo	Milan Kaprál	5h
70	rozbehat backend na produkcnom serveri	Hotovo	Tomáš Kovalčík	20h
88	refector	Hotovo	Jakub Dubec	17h

Burndown graf

Tento graf ukazuje, že tento prvy šprint bol trošku uponáhľaný. Jasne sme vedeli čo ideme robiť, avšak ako to u každého projektu býva ku koncu sa nahromadili problémy, čo je možné vidieť na krivke, keďže sme robili nad rámec ako sme naplánovali.



Sumár podielu práce

Meno	Počet taskov	Celkový čas
Jakub Dubec	1	17h
Milan Kaprál	4	45h
Tomáš Kovalčík	4	50h
Jakub Kubica	1	16h
Oleksandr Lytvyn	1	10h
Andrej Petričko	1	15h

Šprint č. 6

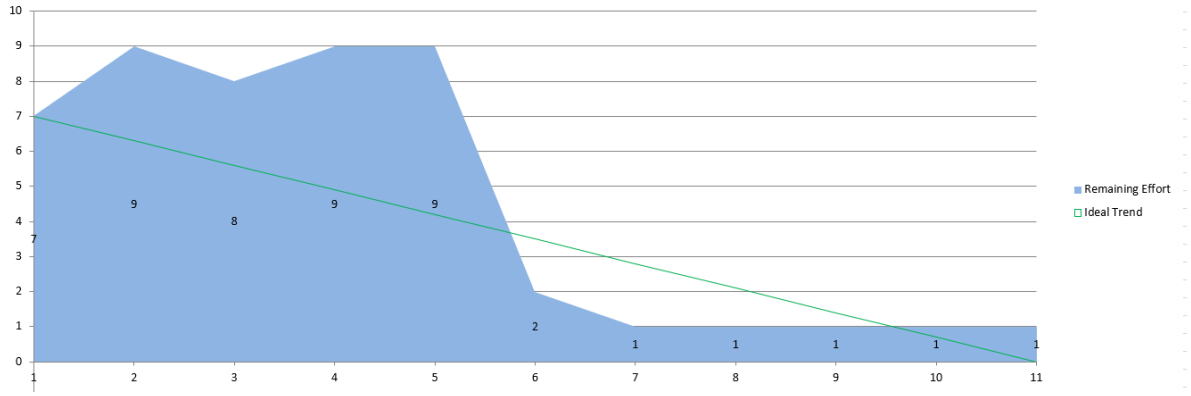
Export z Trella

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
90	BACKEND - Vyžadovanie tokenu pri handlovani requestov	Hotovo	Andrej Petričko	15h
91	FRONTEND - Dashboard dostupný iba po prihlásení.	Hotovo	Andrej Petričko	10h
92	POST /devices	Hotovo	Andrej Petričko	1h
86	REfaktor sucasny	Hotovo	Andrej Petričko	15h

	backend. Zmenit function views na class based views.			
94	Refaktor PUT /devices/{device_id}	Hotovo	Andrej Petričko	1h
93	GET /devices	Hotovo	Andrej Petričko	1h
79	US1.4 - BACKEND - zmena nastavenia konkrétneho zariadenia	Hotovo	Jakub Kubica	15h
96	GET /device_types/{device_type_id}	Hotovo	Jakub Kubica	1h
95	PUT device_types/{device_type_id}	Hotovo	Jakub Kubica	16h
80	us14-frontend-zmena-nastavenia-konkrétneho-zariadenia	Hotovo	Jakub Kubica	5h
81	US1.6 - BACKEND - zobrazit informacie o zariadeni	Hotovo	Oleksandr Lytvyn	10h
	[US2.1] Implementácia merania obsadenosti (firmware + konštrukcia + MQTT broker)	Hotovo	Jakub Dubec	15h
87	Refaktor sucasny backend. Napisat middleware na spracovanie exceptions	Hotovo	Tomáš Kovalčík	10h
97	GET /devices_types	Hotovo	Tomáš Kovalčík	10h

	doplniť do API špecifikácie			
85	Debugging Filtrovania	Hotovo	Milan Kaprál	20h

Burndown graf
V tomto grafe je vidieť väčšiu snahu na začiatku šprintu, pretože sme mali tempo z predchádzajúceho šprintu. V druhom týždni je vidieť mierny pokles práce. Tento jav nastal kvôli tomu, že sme v prvom týždni šprintu spravili väčšinu práce, ktorú sme mali.



Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	1	15h
Milan Kaprál	1	20h
Tomáš Kovalčík	2	20h
Jakub Kubica	4	37h
Oleksandr Lytvyn	1	10h
Andrej Petričko	6	43h

Šprint č. 7

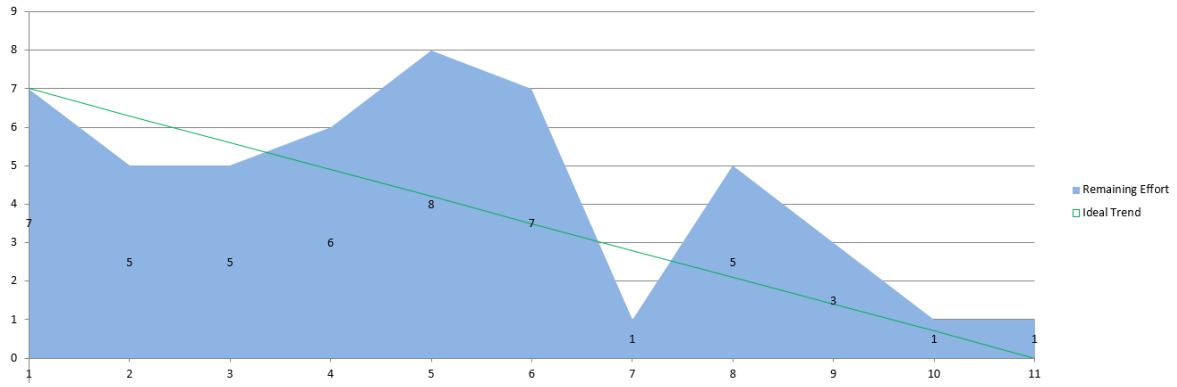
Export z Trella

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
106	Refactor - db	Hotovo	Andrej Petričko	15h

	calls			
115	refactor menu	Hotovo	Jakub Kubica	16h
114	Device type refactor PUT, DELETE a Table	Hotovo	Jakub Kubica	16h
82	US1.6 - FRONTEND - zobrazit detail konkretneho zariadenia	Hotovo	Jakub Kubica	16h
112	DELETE Device - FE	Hotovo	Jakub Kubica	16h
101	Reservation - BE-CRUD	Hotovo	Oleksandr Lytvyn Jakub Kubica	10h 6h
98	Zosuladit FE s novym BE	Hotovo	Tomáš Kovalčík	10h
113	pridaj do ER diagramu DB modelu nove zmeny	Hotovo	Tomáš Kovalčík	10h
	Area - crud	Hotovo	Tomáš Kovalčík	10h
108	measurement - CRUD + oprav ten post na get a namiesto body daj query parametre	Hotovo	Oleksandr Lytvyn	10h
111	Bug - BE	Hotovo	Jakub Dubec	25h
110	Bug - FE	Hotovo	Milan Kaprál	30h

Burndown graf

Nasledujúci Burndown graf ukazuje skákavú krivku. To je preto, pretože v tomto čase sme mali zápočty z iných predmetov, preto aj práca na tímovom projekte bola len vtedy, ked sme si našli čas. Stihli sme ale všetko, čo sme si na začiatku šprintu stanovili.



Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	1	25h
Milan Kaprál	1	30h
Tomáš Kovalčík	3	30h
Jakub Kubica	5	70h
Oleksandr Lytvyn	2	20h
Andrej Petričko	1	15h

Šprint č. 8

Export z Trella

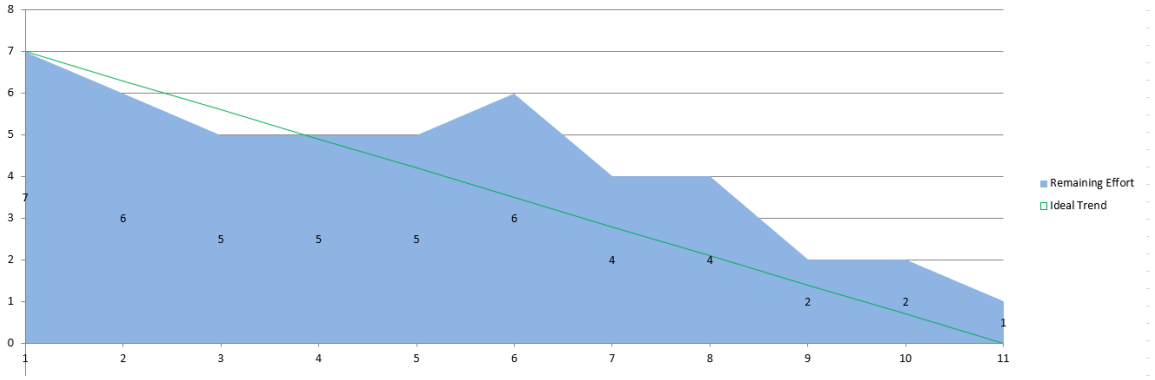
ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
102	BUG - Token na FE	Hotovo	Andrej Petričko	15h
122	US5.2 - FE - zobraziť poslednu nameranu hodnotu a takisto zoznam poslednych hodnot (measurements endpoint)	Hotovo	Jakub Kubica	16h
118	Sensors: CRUD	Hotovo	Oleksandr	10h

			Lytvyn	
85	US1.5 - možnosť filtrovať devicovov	Hotovo	Milan Kaprál	15h
117	API client na backend	Hotovo	Jakub Dubec	4h
119	US1.7 - Vytvoriť django management command na hlasenie stavu zariadenia (chyby, baterka a pod.)	Hotovo	Tomáš Kovalčík	10h
123	fix bugs with device_types and devices	Hotovo	Tomáš Kovalčík	10h
	Vytvoriť CRON job pre US1.7	Hotovo	Tomáš Kovalčík	10h
124	FE - bug - fix update device aby fungoval viac krat	Hotovo	Jakub Kubica	16h
109	GET /devices - ked neexistuje ziaden device tak vrat prázdny list a nie 404	Hotovo	Andrej Petričko	15h
116	data-generator	Hotovo	Jakub Dubec	4h
100	UI / bug - tabulka sa neobnovila pri pridani ani vymazni device typu. Je potrebny refresh	Hotovo	Milan Kaprál	15h
111	UI - ked sa prida device tak nech sa zmena hned odzrkadlí na liste co tam je.	Hotovo	Milan Kaprál	15h

121	US3.5 - vytvorit signal ktory sa pri POST /measurements zavola a vykona nejaku matematiku a rozhodne ci je nutne aby sa poslal email	Hotovo	Andrej Petričko	15h

Burndown graf

Tento graf v tomto šprinte môžeme považovať za ukážkový. To preto, že sme si naplánovali prácu rovnomerne. S prácou sme nemeškali. Na konci šprintu sa práce trochu nahromadilo z dôvodu mergovania BE a FE na gite.



Sumár podielu práce

Meno	Počet taskov	Celkový čas
Jakub Dubec	2	8h
Milan Kaprál	3	45h
Tomáš Kovalčík	3	30h
Jakub Kubica	2	32h
Oleksandr Lytvyn	1	10h
Andrej Petričko	3	45h

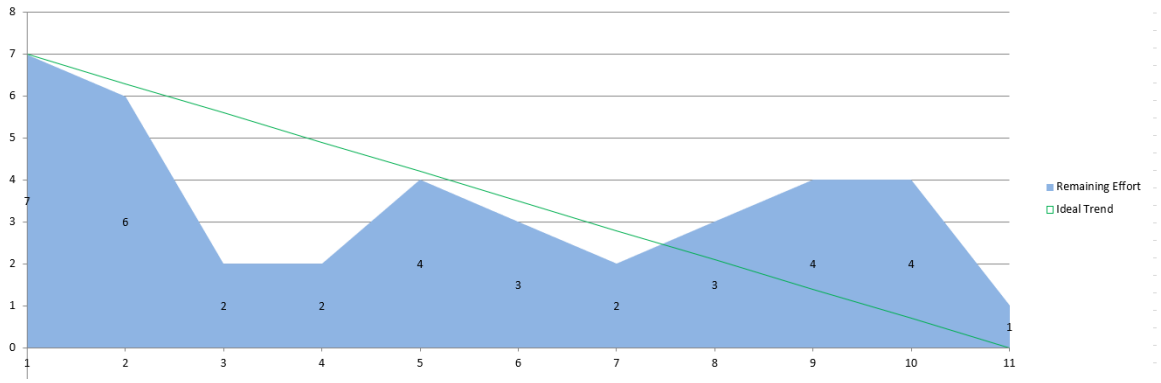
Šprint č. 9

Export z Trella

ID Ulohy	Názov	Stav	Pridelený ku	Potrebný čas
132	FE zodpovednosť	Hotovo	Andrej Petričko	15h
128	Dokumentacia - zodpovednost	Hotovo	Jakub Kubica Milan Kaprál	16h
131	backend zodpovednost	Hotovo	Oleksandr Lytvyn Jakub Kubica	10h 6h
129	vyberiem oponentov - zodpovednost	Hotovo	Tomáš Kovalčík	10h
130	Python Client, HW, MQTT opis zodpovednost	Hotovo	Jakub Dubec	10h
113	Devops	Hotovo	Tomáš Kovalčík	10h

Burndown graf

Ku koncu semestra sa práce nahromadilo nielen pre tímový projekt ale aj pre ostatné predmety. Preto sme zo začiatku šprintu nevenovali toľko pozornosť práci na projekte a uprednostnili sme iné predmety. V druhej časti sme sa snažia dobehnúť prácu na projekte a projektu sme venovali viac času ako v prvom týždni tohto šprintu.



Sumár podielu práce

Meno	**Počet taskov**	**Celkový čas**
Jakub Dubec	1	10h
Milan Kaprál	1	16h
Tomáš Kovalčík	2	20h

Jakub Kubica	2	22h
Oleksandr Lytvyn	1	10h
Andrej Petričko	1	15h

Šprint č. 10

V tomto šprinte sme odovzdali produkt, snazili sme sa dokoncit naležitosti spojene s dokumentaciu. Každý člen tímu doplnil veci, ktoré vedel viac opísať ako predchádzajúci člen. Koniec tohto šprintu predstavuje finálne riešenie nášho produktu.

Globálna retrospektíva zimného semestra

V prvých dvoch šprintoch sme sa naučili pracovať ako tím. Rozdelili sme si úlohy a postupne sme sa začali dostávať pri riešení problémov k hlavným výhodám scrumu. Bolo relatívne náročné sa adaptovať na podmienky pandémie (cítili sme to ako tím), ale ako čas plynul sme sa naučili pracovať aj v týchto podmienkach. Druhy šprint bol uz omnoho lepší, vedeli sme si jednoduchšie zadať úlohy aj ich prediskutovať. Mali sme už technickejšie debaty ohľadom technológií ktoré sme si zvolili a začala sa po nás ukazovať robota. Keďže sa všetci venujeme viac backendu ako frontendu, rozhodli sme sa aby prideľované úlohy týkajúce sa API zahŕňali aj backend aj frontend. Týmto spôsobom si každý vyskúša aj prácu na frontende a nebude tým zaťažený len jeden člen tímu. Treti šprint bol podľa všetkých v tíme najefektívnejší avšak stále bolo pociťovať podmienky pandémie. Pri hodnotení si všetci spomenuli, že to že sme sa nemohli stretnúť osobne vplývalo negatívne na nás celý čas. Čo nás ďalej mrzí je že sme si doteraz nemohli pozrieť priestory. Žiaden pôdorys ani náčrt. Je to vskutku neprijemne. Po štvrtom šprinte, ktorým sme ukončili semester môžeme konštatovať, že sme sa zachovali ako tím. V tomto šprinte boli niektoré úlohy prenesené z predchádzajúceho šprintu, kvôli ich náročnosti alebo časovej tiesni v treťom šprinte. Celkovo hodnotíme zimný semester ako úspešný, pretože sme sa naučili pracovať ako viaccčlenná skupina. Dokázali sme si rozdeliť úlohy podľa obtiažnosti. Na konci každého šprintu sme vedeli objektívne ohodnotiť svoju prácu, aj prácu kolegov v tíme. Vedúci tohto projektu bol s našou odvedenou prácou spokojný a až na pár výnimiek nemal väčšie pripomienky. Projekt sme dotiahli do takého štádia, že v nasledujúcom letnom semestri nebudeme musieť nič dobiehať. S prácou na tomto projekte sme všetci z tímu spokojní.

Globálna retrospektíva letného semestra

V tomto semestri sme mali 5 šprintov. Prvé tri šprinty sme pracovali v konštantnej forme, darilo sa nám rozbehávať komplexné úlohy od produkt ownera. Posledne dva šprinty boli náročnejšie hlavne na čas, keďže to bolo na konci semestra a mali sme toho veľa či už z iných predmetov a taktiež sme trpely únavou po celom roku. Tento semester sa nám veľmi páčil nabrali sme veľa vedomosti a užitočných praktík nie len zo sveta informatiky ale aj riadenia tímu, rozdavania úloh a pracovania na sebe. V poslednom šprinte sme sa zamerali

na dokončenie jednotlivych use casov a hĺbkové testovanie. Výstupom práce v tomto semestri je komplexné riešenie po backedovej ale aj frontendovej stránke.

Zápisnice

Pondelok 5. október 2020 (Jakub Dubec + Tomáš Kovalčík)

1. Prezentacia timu
2. Role (treba este prerozdeliť):
    - komunikácia
    - code-review
    - scrum-master
    - master má byť locked (git-flow)
3. Začiatok 1. šprintu
4. Bude prebiehať analyza (SW aj HW)
    - server
    - ako bude implementované API
    - výber HW a komponentov
5. Hlasovanie a dĺžke šprintu: väčšina je za dva týždne, jeden šprint je ideálne 3MD
6. Vytváranie úloh pre Sprint 1
    - analýza DB servera
    - programovacie jazyk pre API + frameworks
    - výber hardware
    - programovanie hardware (aké jazyky)
7. IoT infraštruktúra
    - IPv6
    - update konfigurácie
    - MQTT-SN

Role

- Jakub Dubec: team leader, technical leader
- Jakub Kubica - backend software developer
- Milan Kapral - backend software developer
- Oleksandr Lytvyn - backend software developer
- Andrej Petricko - full stack software developer
- Tomas Kovalcik - backend software developer, scrummaster

Testing

- Dedikovaneho testera / QA team nemame.
- Každý developer píše testy pre svoje moduly, funkcie ktorá naprogramoval.
- Vytvárať PR iba pre kód ktorý bol otestovaný.

Code review

Rozdelíme sa podla toho kto čo robí.

- Jakub D. Milan - programujú hardvér
- Jakub K. Oleksandr, Andrej, Tomas - pracuju na API a DevOps (konfig servera)

Na to aby sa pull request mohol mergnut je potrebné mať approval od polovice zo skupiny.

Pondelok 12. október (Milan Kaprál)

- rekapitulácia štvrtkového stretnutia
- rozdelenie úloh medzi jednotlivých ľudí
    - Andrej
        - ukážka stránky

        - zobratie úlohy (ako počítať ľudí v miestnosti)
    - Jakub Dubec
        - vybratie vývojovej dosky
        - rozprávanie o zariadeniach, ktoré budeme zabijat
        - analýza protokolov
        - rozpravanie sa o jazykoch, ktoré budeme používať
    - Jakub Kubica
        - analýza jazyka a frameworku (Django & Flask)
        - porovnavania týchto frameworkov
        - pozitíva / negatíva týchto frameworkov
        - navrhnutie používania "Django"
        - Navrhnutie používania verzie jazyka Python (v3.xxx)
    - Oleksandr
        - analýza DB systémov (postgre, mysql vs mb..)
        - analyzovanie time series udajov pre jednotlivé DB
    - Tomáš Kovalčík
        - analýza CUPS
        - komunikacia s API (PyCups)
        - analýza workera (python aplikácia ktorá, vie spracovať MQTT údaje)
    - Peter Kaňuch
        - navrh administrátorskej stránky
        - na konci šprintu na 3-5 minút zhrnutie z daného šprintu pre každého člena a jeho úlohu

Novinky od zákazníka Petra

- ukážka tlačiarní (bude pripojené v najbližšej dobe)
- kávovary asi nebudú v projekte (veľmi drahé) - automat na kartu namiesto kávovarov
- "meranie teploty"
- pri vstupe do budovy bude merať teploty
- pri vyššej teplote bude len zvukový signál (neuchovávať do DB)
- do budúceho týždňa zoznam hardvéru (aká doska, koľko kusov), jednotlivé senzory a koľko kusov
- senzor na kvalitu ovzdušia (co2, humidity,)
- senzor na obsadenosť (infra),
- rezervácia miestnosti (rozmýšľať ako)
- zasadacie miestnosti s displejom a check-inom
    - na displeji (najbližšia schodza, čo sa tam teraz deje, najbližšia rezervácia),
    - check-in (ak je rezervacia na nejaky cas, a neprídem do 10 min (napr), tak sa rezervacia zrusi - vymysliet ako sa to bude robit)
    - vymysliet ako sa budem check-inovať (?? tlačidlo ?? appka ?? web ??)

Otázky

- aké operácie nad databázou sa budú najviac zapisovať (údaje zo senzorov)
- čo ak bude device offline? (zoskupiť dáta, kedy je zariadenie offline a pošle údaje naraz)
- je špecifikácia ku kartovému systemu? (nie, mohla by byť)
- naceňujeme na prototypy (áno) (koľko čoho kúpiť)

Pondelok 19. október (Tomáš Kovalčík)

- Zhrnutie šprintu 1
- Úlohy ktoré boli odprezentované sú hotové
- Počkáme si na stand-up Jakuba o ticket #39 M2M protokoly
- Tiket #38: treba sa dohodnúť ako sa bude merat počet ľudí v miestnosti aj určite jednoznačne, ktoré zariadenia sa majú nakúpiť
- checkin nie je vyriešený

Výsledky analýzy:

- Databáza: PostgreSql
- Back-end: Django
- Front-end: React
- HTTP server: NGINX
- M2M: MQTT-SN
- MQTT client (worker):paho-mqtt

Začiatok šprintu 2 (tento sprint je iba na jeden týždeň 19.10 - 26.10):

- V štvrtok mať aspoň z časti niečo pripravené aby sme vedeli o pripadnych nejastnostiach diskutovať
- rozbehať server a základné služby (nginx, cups, postgresql, mqtt)
- rozbehať základný back-end a front-end
- navrhnúť databázu
- navrhnúť API
- analýza OS pre IoT systém
- analýza displeja

Pondelok 26. október (Jakub Kubica)

- žiaden task nie je hotový na 100%
- 1. týždeň nestačí, predĺžime to o 2 týždne
- dopísať boilerplate a DB
- Jakub K. - základná django app -- dať gitignore, ošetriť hierarchiu priečinkov
- Tomáš - dokončiť docker
- Andrej do konca šprintu jednoduchý dashboard - kde sa bude dať nad tým stavať
- Návrh API - bude to REST - záleží od dátového modelu - odkladáme na neskôr
- analýza displeja - dokument v smart-space - aké piny čo kde pripojíme spísať, analyzovať HW - nakresliť schému s konkrétnymi komponentami a s konkrétnym devicom a linkom
- ktorý konkrétne display je ich tam milión, kúpiť lacnejší nie 100e, stačí 4,3", netreba 7" display
- konkrétny OS vybraný (Milanova Tabuľka)
    - FreeRTOS
    - RIOT (preferencia)
- Jakub D. - rozbehne na ESP32 RIOT a uvidíme
- Milan - analýza HW - začni analýzu nemusíš všetko + DIODku na menenie farby svetla dáme asi viacero
- Alexander - navrhnúť databázový model
- Do štvrtka aspoň niečo pripraviť a na ďalší pondelok(ráno) aspoň nejaký pokrok aspoň čiastočne

Pondelok 2. november (Oleksandr Lytvyn)

Tomáš:

- Dorobiť django endpoint
- Dockerfile na development (na AIP) readme
- Tento tyzden budu sa robit pipeliny

J. Kubica:

- Návrh Django aplikácie

Oleksandr:

- Prerobiť databazu (pridat timestamps, zostavit merania do jednej tabulky)

J.Dubec:

- RIOT na ESP
- Komunikacia na MQTT s ESP
- Networking na ESP, konfigurácia

Pondelok 9. november (Andrej Petričko)

Oleksandr:

- Databázový model
- Tabulka s devices data bola zjednotená pre všetky devices

Jakub Dubec:

- Medzi Devices_types a Sensors spraviť many-to-many cez spojovaciu tabuľku
- Physical_units spojiť s tabuľkou sensors
- Tabuľku Locations nepotrebujeme
- V tabuľke Areas pridať stĺpec Subareas ktorý bude ukazovať na riadok v tej istej tabuľke a device_id
- Measurements je spojená s Devices
- Otázka ohľadom používateľov a ich identifikácie
- Tabuľka Users ide preč
- V tabuľke Reservations zmeniť user_id na owner(VARCHAR)
- Rovnaká úprava v tabuľke printers
- Vytvoriť tabuľku API_KEYS(uid, secret, nazov)

Jakub Dubec

- Riot OS
    - Snaha rozbehať toolchain
    - Niekoľko problémov:
        - Komunikácia na macOS
        - Ručné flashovanie a problémy imagom
        - Limitácie s vláknami
        - V skratke „Je to bordel“
- Vyskúšanie a analýza dvoch iných OS (OpenWRT)

Peťo:

- Aby to bolo user-friendly
- Na jednoduché sensory stačia ESP a na jeden typ senzoru vyskúšať čosi väčšie (EDISON)

Jakub Kubica

- Zakladna Django aplikácia je ready

- Funguje aj admin page
- Do štvrtku vytvoriť DB (modely)

Milan Kaprál
- Prezentácia analýzy senzorov
- Treba dávať bacha na sensor MQ135
- Pozrieť termokameru (možno)

Tomáš Kovalčík
- Server nastavený, stránka beží
- Otázka ohľadom deploy a pipelines – Jakub Dubec odpoveď

Andrej Petričko
- Prezentácia React Appky
- Spokojnosť s nočným režimom

Peter Kaňuch
- Zmena modelu implementácie bude prebiehať jeden task backend + frontend

Pondelok 16. november (Jakub Dubec)

- ORM circular dependency
- reorganizácia dátového modelu
    - odstránenie niektorých FK (device neukazuje na measurements, etc)
    - premenovanie stĺpcov (human_readable -> title)
    - naming (treba zjednotiť názvy)
    - MR pre databázové modely bude dnes
    - token auth FE a API
        - tabuľka token (ktorá má FK user_id): TODO do diagramu
    - Django VS Django DRF

Pondelok 23. november (Milan Kaprál)

Poznámky od Peťa
- dokončiť stránku - podopĺňať zápisnice na stránku a veci ktoré tam majú byť
- do najbližšieho stretnutia odovzdať Petrovi dokument "Míľnik"

Jakub Dubec:
- OPENWRT – nie je možne
- návrh použiť Linux pre ESP32 - sú tam nejaké komplikácie
- návrh pre ESPcka - nepoužívať OS . Pre splnenie požiadaviek nainštalovať OS len na jedno zariadenie
- kvôli tomu že nevieme dostať OS na ESP32 - pokúsiť sa to dostať na inú dosku

Andrej Petričko:
- riešenie autentifikácie
- problémy pri frontende, backend je hotový

Tomáš Kovalčík:
- problém s IPv6 - problém pri zadávaní viacero devicov
- spraví sa tool, ktorý to bude flashovat a ako vstup musí byť ID s frontendu
- vstupy zadávať ručne (názvy zariadení)
- mazanie devicov po jednom.
- ak sa zmaže aj inštancia posledného device, tak nech ostane type

Oleksandr Lytvyn:

- základné metódy pre model (swagger)
- pridanie funkcie: "pridanie devisov"
- treba dorobiť a upraviť nejaké funkcionality
- treba to robiť v snake
- treba upraviť routing

Jakub Kubica:

- pridanie/zmazanie nového devicetype
- vytvorenie formuláru v dashboarde
- kompletne rozbehanie pridanie/zmazanie
- nebeží to ešte na frontende
- upraviť tak, aby som mal pri device tlačidlo "delete" z UserFriendly hľadiska

Milan Kaprál:

- nedokončenie tasku
- doštudovať mqtt-sn kvôli UDP

Zhodnotenie a evaluácia zimného semestra

retrospektívne pozretie sa do minulosti na predchádzajúce šprinty

- zapisovanie do míľnika
- spraví sa zdieľaný doc
- každý za seba do technickej dokumentácie vloží svoje analýzy jeden dokument rozdelený na dve časti

Pondelok 30. november (Tomáš Kovalčík)

Všeobecné informácie:
Úlohy zo šprintu 4 (míľnik, update stránky, analýzy na stránke) sú hotové. Ostali úlohy zo 3 šprintu.

Jakub Dubec:

- micropython ESP 32- firmware - rozbehal MQTT na tom s IPv4, IPv6 sa pozrieš potom, musí sa overi·
- zistil že niektoré zariadenia pracujú na 5V logike. Musí ísť dokúpiť shifter.

Jakub Kubica:·

- list zariadení na FE. Je to skôr US 1.5.

Pondelok 7. december (Tomáš Kovalčík)

Ukončili sme šprint - 4

- treba ešte mergnut posledné zmeny (snáď v priebehu tohto týždňa) - potom ja (TOMAS) aktualizujem trello.
- retrospektíva (každý je už unavený a znechutený zo semestra (celkovo))

Pondelok 1. marec (Jakub Kubica)

Ukončenie šprintu 5

Milan Kaprál
- metodiky, zápisnice, definition of ready/done sú hotové
- dokumenty treba dať na stránku

Tomáš Kovalčík
- navrhol algoritmus a vie prečítať dáta
- ešte treba nejaký firmware na testovanie a hodiť to na git
- na ďalší týždeň prichystá fyzicky prototyp

Andrej Petričko
- skoro hotovo + ešte obaliť routy aby tam pristupoval s routami a pushnut to
- doimplementovat refresh token threshold

Oleksandr Lytvn
- dopísal API špecifikáciu - hotovo

Tomáš Kovalčík
- nejaká chybička, v slacku je fix

Jakub Kubica.
- task done, z readme zmazať yarn

Pondelok 15. marec (Oleksandr Lytvyn)

Ukončenie šprintu 6

Tomáš Kovalčík
- Backend úlohy spravené všetky, len nie je to mergnute
- Devices a Device_Types sú všetko implementované, ešte sa čakajú zmeny od Jakuba K.
- Dokončiť problémy s tokenom na FE
- Areas a Reservations na šprint 7

Andrej Petričko
- User má byť prihlásený pre prácu s API
- Refactoring na Class based views - Done

Oleksandr Lytvyn
- FE task nie je hotový z technických príčin
- Merge request na BE úlohu

Jakub Kubica
- PUT a GET devices je spravene

Milan Kaprál
- Filter Devices podľa Device_Type

Jakub Dubec
- Naprogramovaný Occupancy device
- Služby - komunikácia na MQTT a výstup
- Životný cyklus 6 min
- MQTT consumer - DataWorker
- Zatiaľ ipv4, neskôr ipv6
- DHCP enabled
- Podklady pre implementácie AirQuality device
- Worker Repository created - potrebné dokončiť
- Problémy:
- MQTT nepodporuje TLS
- Device Generator - special task

General:
- Definícia úloh na šprint 7 (podrobnejšie v Trello)
- Chyba v Swaggeri POST -> GET pri measurements from period

Pondelok 29. marec (Jakub Dubec)

Ukončenie šprintu 7

Jakub Dubec
- prezentácia data generátora
- Na ďalší šprint:
- worker s API klientom

Oleksandr Lytvyn
- prezentácia measurements CRUD

Andrej Petričko
- prezentácia hotových taskov

Jakub Kubica
- list of devices UI
- action buttons
- device types (CRUD)

Tomáš Kovalčík
- areas CRUDs

Plánovanie šprintu

- pridávanie devicov je cez autodiscovery
- device types musia byť predom zadefinované, inak sprav ignore
- pridať do generator user access with temperature check
- plánovanie notifikácii o mŕtvych zariadeniach

CRON

- django management command
- upozornenie o vetraní (doplánovať v štvrtok)
- tu by sa mohli použiť Django signals
- posielanie notifikácii
- telegram BOT?
- emaily?
- detaily meraní pre device (edited)

Pondelok 12. apríl (Andrej Petričko)

Ukončenie šprintu 8

Jakub Dubec

- API client je hotový (produkčný/localhost)
- treba implementovať doňho iba metódy čo má robiť
- vytvorené generátory na CO2, humidity, motion a temp
- ukážka funkčnosti a používania generátora

Oleksandr Lytvyn

- Implementácia CRUD operácií na rezervácie
- Implementácia testovania

Andrej Petričko

- oprava buggu na backende, pri get devices list vracia prázdny list
- fixnutý token management na frontende
- úprava prihlasovacieho formulára
- jeden task nespravený (notifikácia measurements)

Jakub Kubica

- implementácia detailu pre device
- measurements v detaily chce zákazník mať ako list
- refactor kódu pre front-end a názvov panelov v dashboarde

Milan Kaprál

- implementácia filtra pre devices podľa type a area
- T. Kovalčik, pre lepší user experience zjednotiť taby device list a filter device

- bug pri filter by area, neukazuje type
- bug fix potrebného refreshu, pri listoch stránka sa refreshuje automaticky

Tomáš Kovalčík
- implementácia emailovej notifikácie pri device status v prípade nízkej batéri
- kontrola batérie je nastavená na jeden krát za deň
- implementácia notifikácie ak zariadenie prestane merať

Úloha od vedúceho na ďalší šprint:
- vytvoriť dáta a pripraviť produkt na testovanie pre druhý tím, zosumarizovať backlog, skontrolovať aby boli dokumentácie kompletné.
- kontrola backlogu a konzultácia user stories.

Pondelok 26. apríl (Milan Kaprál)

Ukončenie šprintu 9

Jakub Dubec
- spísaná dokumentácia k BE (uložená na disku)
- spísaná dokumentácia k MQTT
- spísaná dokumentácia k

Andrej Petričko
- spravené signály

Jakub Kubica
- spravili sme dokumentáciu
- spísané riadenie v dokumentácii
- burndown chart
- doplnený screen zo swaggera
- screenshoty z FE

Milan Kaprál
- spravili sme dokumentáciu
- spísané riadenie v dokumentácii
- burndown charty
- doplnený screen zo swaggera
- screenshoty z FE

Tomáš Kovalčík
- feedback
- zdokerizovanie BE
- začal opisovať DEVOPS

Oleksandr Lytvyn

- dokončenie bugov na BE
- delete na DEVICE a DEVICE_TYPE

Pondelok 10. máj (Tomáš Kovalčík)

Ukončenie šprintu 10

General:

·

Produkt bol

odovzdaný tretej strane

·

Na produkčnom

serveri beží finálna verzia.

· Dokončujeme dokumentáciu a všetky náležitosti k tomu potrebné.

Inžinierske dielo

Cieľ zimného semestra

Cieľom našej práce počas zimného semestra bolo návrh a implementácia API rozhrania s čiastočnou implementáciou frontendu. Takmer na konci môžeme konštatovať, že naše ciele sme splnili. Zo začiatku zimného semestra sme si naštudovali problematiku a možne riešenia, vhodné programovacie jazyky a prostredia. Pripravili sme zoznam HW, ktorý budeme pri tomto projekte potrebovať v letnom semestri. Pripravili sme veľa dokumentácií pre rôzne smery, ktoré nám napomohli pri výbere rôznych technológií. Takmer na konci tohto zimného semestra môžeme povedať, že sme plne pripravený na začiatok letného semestra a nezaťažovať sa s výstupmi resp. úlohami z toho zimného.

Cieľ letného semestra

Hlavným cieľom letného semestra bolo dokončenie celkového projektu včas. Tento cieľ sme si rozdelili na menšie ciele, ktoré pozostávajú z nasledujúcich častí. Dokončenie frontendu podľa požiadaviek product ownera s príjemným grafickým rozhraním. Tento cieľ sa podarilo splniť, pretože product owner bol s funkcionalitou, rozložením prvkov na stránke a vzhľadom stránky spokojný. Ďalším cieľom letného semestra bolo spojenie IoT zariadení so serverom. Toto spojenie predstavuje komunikáciu a preposielanie live údajov zo senzorov, ktoré sa automaticky zobrazia na Admin stránke. Zariadenia boli pripojené do systému a infraštruktúry a bezproblémovo zasielali správne údaje na server, kde sa ukladali do databázy. Týmto považujeme tento cieľ za splnený a plne funkčný. Ostatné časti systému sa postupné dopĺňali alebo opravovali pri nejakom konflikte. Celkovo sme splnili všetky požiadavky, ktoré od nás produkt owner na začiatku zimného semestra požadoval.

Analýza technológie

V tejto kapitole sú opísané technológie, ktoré sme sa rozhodli použiť na vývoj SmartSpace aplikácie.

Serverová implementácia

Backend API

Naša aplikácia bude komunikovať cez REST aplikačné rozhranie. V tomto smere sme sa všetci zhodli jednoznačne keďže RESTful architektura patrí momentalne medzi najpoužívanejší štandard pri vývoji webových aplikácií. Toto naše REST rozhranie implementujeme pomocou frameworku Django, čo je robustný framework zapísaný v jazyku Python. Django nám značným spôsobom uľahčuje prácu s DB vďaka skvele navrhnutému ORM. Je to framework do koča aj do voza. Jedinou nevýhodou tohto frameworku bol fakt že väčšia časť tímu s ním nemala doteraz skúsenosť. Túto slabinu sme ale po prvom šprinte všetci prekonali.

Ako alternatívu sme zvažovali druhý Python framework Flask. Avšak ten neponúka toľko výhod. Kvôli obmedzenému času na tomto projekte nám nevyhovoval, keďže väčšinu vecí čo Django ponúka "out-of-the-box" by sme museli vo frameworku Flask implementovať ručne. Porovnanie Flask a Django je zobrazené v nasledujúcej tabuľke:

Vlastnosti\Názov	Django	Flask
Open-source, free	✓	✓
Veľká podpora(patterns, tools, features and functionality)	✓	✗
Podpora vstavanej databázy(ORM) (Odporúča sa pre použitie SQL databázy)	✓	✗
Admin panel	✓	✗
Vhodný pre NonSQL databázu	✗	✓
Vhodný pre tvorbu servera od nuly(každý detail)	✗	✓
Takmer na všetko rozšírenia tretích strán	✗	✓

Aplikácia napísaná v Django REST frameworku, ktorá komunikuje na PostgreSQL a zabezpečuje:

- CRUD operácie nad dátovým modelom
- agregácie (posledných n-meraní)
- kontrola stavu systému (či a aké senzory posielajú dáta)
- notifikačný systém (správca systému, vie dostať notifikáciu v prípade, že niektoré senzory sa dlho nehlásia, prípadne im dochádza batéria)
- autentifikácia / autorizácia

Projekt je štruktúrovaný do dvoch pod-aplikácií:

- core: ORM mapovanie, správcovské príkazy (CRON), migrácie
- api: RESTful API (CRUD operácie, serializácia, filtre)

Autentifikácia je implementovaná pomocou JSON Web Tokens (JWT). V projekte sú dodržiavané HTTP status kódy a základné REST konvencie. Jednotlivé end-pointy sú deklaratívneho charakteru (imperatívnych je minimum). Súčasťou projektu sú databázové migrácie, ktoré sledujú konzistenciu dátového modelu v PostgreSQL.
Dokumentácia aplikačného rozhrania je opísaná OpenAPI 3.0 štandardom, ktorá sa nachádza priamo v repozitári. Programátor tak vie veľmi pohodlne generovať dummy API alebo kolekciu požiadaviek, čo značne urýchľuje a zjednodušuje vývoj a komunikáciu v tíme.
V rámci vývoja bol dodržiavaný TDD. Konfigurácia aplikačného servera, je načítaná z prostredia (environment variables). Pre zjednodušenie vývoja, projekt dokáže pracovať s .env súbormi.
Využité open-source nástroje a komponenty (závislosti sú spravované pomocou nástroja Poetry):

- python-dotenv: práca s .env súbormi
- django-enum-choices: podpora pre natívne Python enum objekty v ORM

- gunicorn: aplikačný WSGI server
- djangorestframework: RESTful nadstavba nad Django frameworkom
- djangorestframework-simplejwt: JWT autentifikácia
- django-imap-backend: nástroj na testovanie e-mailov

Databázový Systém

V každom projekte je dôležite mať databázu ktora by korešpondovala s požiadavkami samotného systému. Keďže základom väčšiny IoT projektov je pravidelne meranie fyzikálnych veličín v nejakých časových okamihoch, databáza musí byť schopná rýchlo spracovavať časovo zamerané údaje. Je vhodné si zadefinovať dôležité vlastnosti databázového systému(ďalej iba DBS) pre IoT projekt:

- škalovatelnost
- rýchlosť získavania a zápisu údajov
- podpora technológie

K týmto trom vlastnostiam ešte sa pridáva aj skúsenosť členov tímu z určitými databázovými technologiami a rýchlosť implementácie tychto technologii.
Je zrejmé že existuje veľa možností implementácie DBS pre IoT projekty, ale vzhľadom na to že väčšina tímu je oboznámená a vie pracovať s relačnými databázami bolo dohodnuté, že bude použitý **SQL pristup**.
Ako nasledovný krok prieskumu bolo definovať vhodné DBS na pracu s IoT dátami, a to populárne MySQL a PostgreSQL. Každá z týchto DBS má svoje výhody a nevýhody, ktoré sú spracované podrobnejšie v analýze, ale v skratke MySQL - rýchlosť a jednoduché použitie. MySQL je vhodné použiť v prípadoch distribuovanych operácii, pre webové stránky a aplikacie.
Výhodami PostgreSQL sú úplný súlad s SQL štandardom, Open-source a riadený komunitou, možnosť inštalácie rozšírení. PostgreSQL je vhodný na použitie v pripadoch kde potrebujeme vykonávať komplexné operacie, integraciu s inými službami a v systémoch, kde sa vyžaduje vysoká integrita údajov.
TimescaleDB - open-source relačná databáza pre time-series dáta. TimescaleDB ponúka spoľahlivosť, flexibilitu, jednoduché použitie a škálovateľnosť, ktorú vyžadujú aplikácie, infraštruktúra pre analýzu údajov a zložité systémy. TimescaleDB je účelovo zostavená na škálovanie a zvládanie time-series dátového zaťaženia a je navrhnutý ako rozšírenie PostgreSQL.

Výhody TimescaleDB sú:

- Vyššia a stabilnejšia rýchlosť prijímania time-series dát, ako čistý PostgreSQL
- Výkon dopytu je ekvivalentný alebo rádovo vyšší (najmä čo sa týka časovo orientovanych dopytov)
- Časovo orientované funkcie (časovo orientovaná analýza a správa údajov)

Vzhľadom na vyššie uvedené body, pre IoT projekt SmartSpace je vhodným výberom PostgreSQL, databázový systém s možnosťou ďalšieho rozšírenia pomocou TimescaleDB. Pre návrh prototypu projektu, keďže veľa operácií sa nebude vykonávať na DBS, je vhodné použiť čistý PostgreSQL. Ako riešenie pre budúcnosť, keď time-series data búdu spôsobovať veľký vplyv na výkon DBS, bude vhodné pouziť rozšierenie TimescaleDB.

MQTT Worker

Aplikácia, ktorá je zodpovedná za spracovanie dát senzorov, ktoré prídu na MQTT broker. Senzory dáta publikujú a worker, je prihlásený na ich odber. Aktuálne systém pozná nasledujúce MQTT cesty:

URI	**Popis**
tp/occupancy	Informácie o obsadenosti
tp/co2	Merania CO2
tp/humidity	Merania vlhkosti
tp/temperature	Meranie teploty vzduchu

Formát požiadavky je rovnaký pre všetky URI, kanále sú rozdelené len kvôli flexibilite odberu. Na vstupe je reťazec oddelený bodkočiarkov a spolu nesie nasledujúce informácie (v poradí):

- Identifikátor zariadenia (device_id)
- Identifikátor senzoru (sensor_id)
- Identifikátor fyzikálnej jednotky (unit_id)
- Hodnota (data)

Požiadavka vie byť reprezentovaná nasledovne:

Device ID	e6db95be-4190-4b64-b001-7bd53a215b9d (inštancia zariadenia)
Sensor ID	abeedcb8-82b9-4cff-a3b8-a7c5c7d1a227 (identifikátor senzoru, nesie informáciu o chybovosti)
Unit ID	4bcf1b5e-a10d-48bf-b1cf-63d5b7aa4c4c (identifikátor fyzikálnej jednotky)
Data	15.5 (surová hodnota ako reťazec)

Príklad surovej požiadavky podľa tabuľky:
e6db95be-4190-4b64-b001-7bd53a215b9d;*abeedcb8-82b9-4cff-a3b8-a7c5c7d1a227*;*4bcf1b5e-a10d-48bf-b1cf-63d5b7aa4c4c*;*15.5*

Aplikácia po štarte sa prihlási na odber dát z brokera a následne dáta, hneď ako prídu posiela na server pomocou API klienta. Projekt je implementovaný ako konzolový program, ktorý vie byť spúšťaný ako služba operačného systému. Bol navrhnutý tak, aby bol čo najjednoduchší s minimom závislostí:

- paho-mqtt: MQTT klient
- smartspacers-python: API klient

Služba vie bežať dlhodobo a automaticky používa refresh token a aktualizáciu prístupového tokenu.

Python API Klient

Jednoduchá softvérová komponenta, ktorá mapuje API endpointy. Je zodpovedná za komunikáciu so serverom a normalizáciu vstupu a výstupu. Zároveň zabezpečuje autentifikáciu používateľa (na vstupe berie iba meno a heslo). Projekt nie je funkčný ako samostatná aplikácie, je implementovaný ako softvérová komponenta, ktorá musí byť implementovaná v rámci projektu (nedá sa spustiť sama). Modul bol navrhnutý tak, aby reprezentoval štruktúru na strane servera, bol jednoducho rozšíriteľný (vykonávanie HTTP požiadaviek je oddelené od definície jednotlivých koncových bodov).
Projekt je inštalovateľný ako klasický Python modul pomocou štandardných nástrojov:

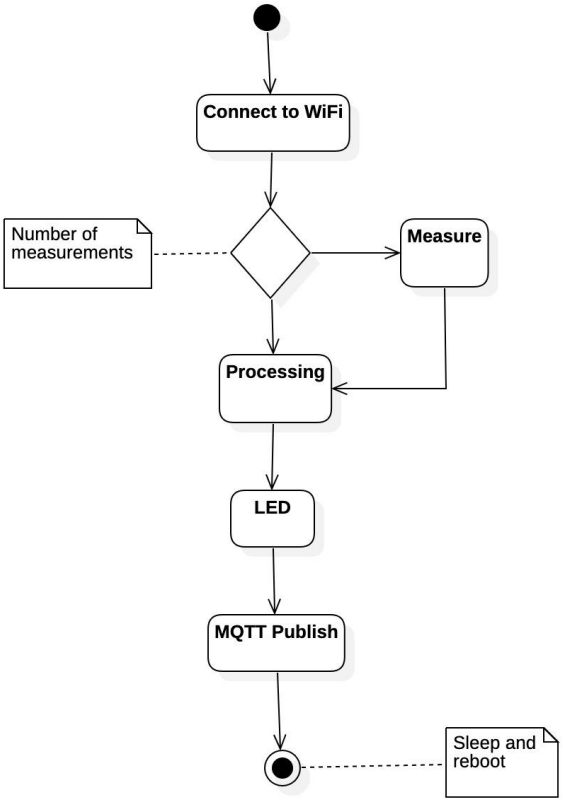
```
# Pip
pip install git+ssh://git@gitlab.com:smartspacers/client-python.git

# Poetry
poetry add git+ssh://git@gitlab.com:smartspacers/client-python.git

# Manual
python setup.py install
```

Motion sensor

Firmvér implementácia pre ESP8266, ktorá meria dáta z MIR senzoru, na základe ktorého jednoduchým algoritmom detekuje obsadenosť kancelárskeho miesta, ku ktorému je zariadenie priradené. Je napísaný v jazyku C nad Arduino frameworkom.
Súčasťou implementácie je aj DEBUG konzola (na UART výstup zariadenia).



Zariadenie sa po štarte pripojí na WiFi, vykoná 60 meraní (jedno meranie trvá vždy aspoň jednu sekundu). Následne vyhodnotí namerané dáta a v prípade, že aspon 10% meraní zachytí pohyb, vyhodnotí miesto ako obsadené (zasvieti LED svetlo). Výsledok posiela na server (vo forme počtu pozitívnych meraní - kedy zariadenie zaznamenalo pohyb). Server má vlastnú formu prepočtu obsadenosti (kvoli flexibilite).

Data generátor

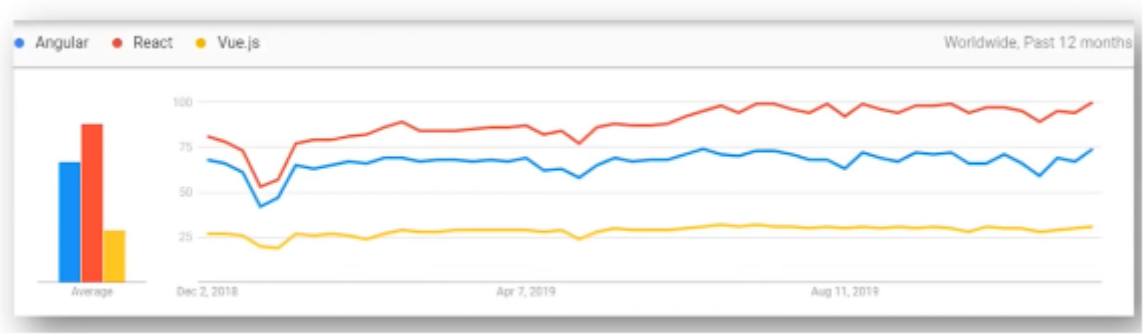
CLI aplikácia, ktorá má za úlohu generovanie čo najviac autentického obsahu za účelom testovania a prezentácie produktu. Skladá sa z dvoch častí:

- Generovanie zariadení (cli.py devices)
- Generovanie meraní (cli.py execute)

Pre správne fungovanie má generovanie meraní prebiehať v pravidelných intervaloch (jedno spustenie reprezentuje merania pre všetky zariadenia vytvorené scriptom devices). Simulujeme tak celú IoT sieť, ktorá vykonáva merania z rôznych miestností (areas). Generátor je schopný všetky hodnoty vytvoriť sám (akceptuje ako vstup prázdnu databázu a všetky potrebné číselníky vytvorí podľa potreby). Na generovanie zariadení využíva Python API klienta, dáta posiela cez MQTT. Vygenerované zariadenia drží v konfiguračnom súbore devices.json.

Klientska implementácia

Naša aplikácia prináša aj grafické rozhranie pre administrátora SmartSpace-u. Frontendové frameworky ponúkajú zaujímavé možnosti. Pozreli sme sa na Vue.js, React a Angular. Vue je zo všetkých najľahšie, tak ide React a najťažší na naučenie a prácu je Angular. Najpopulárnejší zo všetkých je React a má najväčšiu komunitu. Pre grafické zobrazenie dát z databázy a na jednoduchú prácu s nimi sme sa rozhodli spraviť grafické užívateľské prostredie v React aplikácii. Túto technológiu sme si zvolili na základe popularity a z dôvodu že nám prišla zaujímavá. Na nasledujúcom obrázku vidíme popolularitu frontendových frameworkov:



Infraštruktúra

Náš produkt beží na operačnom systéme Ubuntu 18.04 LTS. Naša aplikácia sa skladá z viacerých komponentov. Po dlhšej analýze sme sa rozhodli na nasadenie použiť Docker. Dockerizovaná aplikácia má tú obrovskú výhodu že je nezávislá od platformy a teda development a produkcia je totožná.

Nasadzovanie softvéru bude v budúcnosti jednoduché a bezproblémové.Na “Continuous Integration, Continuous Delivery” využívame GitLab-CI nástroj. Každá zmena v repozitáry spúšťa tzv. “pipeline” ktorá vykoná predstavené úlohy (build, test, *deploy)

*deploy sa vykonáva iba po spojení zmien z development vetiev do hlavnej git vetvy.

Komunikácia na koncové zariadenia

Z dlhodobého hľadiska považujeme za kľúčové vybrať správny aplikačný protokol na prenos dát zo senzorov. Existuje niekoľko zabehnutých populárnych IoT riešení, pri ktorých sa pokúsime špecifikovať ich prípady použitia, vplyv na spotrebu a jednoduchosť implementácie na klientskej a serverovej časti. Výstup z tejto kapitoly použijeme na výber komunikačného protokolu v našom riešení.

Zameriame sa hlavne na protokoly, ktoré fungujú nad TCP/IP architektúre a bežne fungujú nad IEEE 802.11 (WiFI) fyzickou vrstvou.

HTTP

Populárny protokol textovo založený, ktorý bol navrhnutý pre účely WWW v roku 1991. V klient-server architektúre ho označujeme ako request-response protokol. Klient vykonáva požiadavku na ktorú server odpovedá. Na identifikáciu prostriedku používa tzv. Universal Resource Identificator (URI). Na transportnej vrstve sa používa TCP protokol (od verzie HTTP3 UDP).

Na šifrovanie komunikácie používa SSL/TLS na prezentačnej vrstve (od verzie HTTP2 je nevyhnutné použiť šifrovanie). QoS nie je riešený na úrovni protokolu a treba ho implementovať aplikačne alebo na úrovni TCP (limitované). Veľkosť hlavičky je flexibilná a veľkosť tela je závisí od serverovej implementácie.

Z aplikačného uhla pohľadu ho považujeme za relatívne komplikovaný na implementáciu na vnorených zariadeniach. Protokol neudržiava trvalo otvorené TCP okno, preto nie je úplne vhodný na kontinuálne posielanie dát (od verzie HTTP2 vieme využívať jedno TCP spojenie na viac HTTP požiadaviek). Z testovania vychádza, že HTTP by malo zmysel používať iba za predpokladu, že zariadenie po zobudení odošle práve jednu požiadavku na jeden centrálny uzol a po jeho úspešnom odoslaní ide opäť na nejaký čas do režimu spánku.

Na kontinuálne posielanie dát považujeme za nevhodný kvôli neustálemu otváraniu TCP okna. Rovnako považuje nevýhodu podpory iba jedného prijímacieho uzla.

Teoreticky by sa dalo polemizovať o využití v IoT za predpokladu, že použijeme HTTP3 pretože operuje nad UDP. Nakoľko sa jedná stále o experimentálnu technológiu, ktorá nie je globálne podporovaná v štandardných HTTP serveroch túto možnosť zatiaľ vypúšťame.

MQTT

Jeden z najstarších protokolov z kategórie machine-to-machine (M2M), ktorý bol vyvinulo IBM v roku 1999. Operuje nad architektonickým vzorom client – broker, kde zariadenia publikujú alebo sa prihlasujú dátové topics. V praxi to vyzerá tak, že klient publikuje správu na MQTT message broker, ktorý následne túto správu pošle všetkým zariadeniam, ktoré sú prihlásené na daný komunikačný kanál (topic). Klient môže publikovať alebo sa prihlasovať na viac komunikačných kanálov naraz. Dáta sa prenášajú v binárnej forme a zabezpečujú sa pomocou SSL/TLS. Na transportnej vrstve sa štandardne používa TCP.

Protokol je veľmi populárny v IoT riešeniach a to hneď z niekoľkých praktických dôvodov:

- Je veľmi lightweight a jednoduchý na implementáciu na strane klienta
- Prenášajú sa len nevyhnutné dáta
- Veľkosť hlavičky sú iba 2 bajty

- TCP spojenie sa dlhodobo udržiava, nie je potrebné ho znova otvárať pri kontinuálnom posielaní dát
- Súčasťou špecifikácie je QoS

Pri kontinuálnom posielaní dát a udržiavanom otvorenom TCP okne má MQTT ďaleko lepšie výsledky ako HTTP.

V našom prípade, budeme skôr vyžadovať princíp fungovania v režime, že sa meracie zariadenie najprv zobudí zo spánku, pošle merania na broker a vráti sa opäť do režimu spánku. Pre tento prípad je energetická spotreba porovnateľná s HTTP. Na naše účely by nám ideálne vyhovovala komunikácia na báze UDP. Kvôli tomuto vznikla modifikácia s názvom MQTT-SN, ktorá funguje na báze UDP. Je optimalizovaná pre prípady, kedy potrebujeme odosielať správy zo zariadenia pripojené pomocou bezdrôtovej siete na message broker s dôrazom na minimalizáciu prenášaných dát a nízku energetickú náročnosť. Táto implementácia má zatiaľ ale momentálne minimálnu podporu populárnych open-source message brokerov (najčastejšie formou modulu). Za posledné roky ale začína naberať popularitu.

CoAP

Najmladší protokol zo spomínaných, ktorý navrhnutý presne pre účely IoT. Radíme ho do kategórie light-weight M2M protokolov a dokáže operovať na obidvoch zatiaľ spomenutých architektonických vzoroch (client/server a client/broker). Bol navrhnutý tak aby vedel jednoducho fungovať v spolupráci s HTTP RESTful modelom formou jednoduchej proxy.Na identifikáciu prostriedku sa používa URI podobne ako v HTTP. Keď prídu dáta na konkrétnu URI, sú notifikovaný všetky zariadenia, ktoré sú prihlásené na odber podobne ako v MQTT.

Hlavička má veľkosť 4 bajty. Na transportnej vrstve používa UDP a dáta zabezpečuje pomocou DTLS alebo IPSec.

Medzi jeho hlavné výhody patrí podobnosť s RESTful návrhom (prvá veta na stránke projektu), jedná sa stále o mladý protokol, čo je cítiť na podpore existujúcich open-source riešení prípadne na jeho integrácií do zabehnutých nástrojov. Toto považujeme za problém ak chceme implementovať stabilnú a spoľahlivú službu. Jedná sa o veľmi zaujímavé riešenie, ktoré ale ešte potrebuje overenie časom.

Serverové služby

Webový server:

V dnešnej dobe väčšina webových služieb beží na Apache alebo NGINX. Preto nemá význam porovnávať iné webové servery.

NGINX:

- modulárny - dokážeme pridať navyše moduly pre väčšiu funkcionalitu ale tie sa musia pred štartom skompilovať do NGINX. Dynamické pridávanie nie je možné
- Ideálny na veľkú záťaž. Veľmi rýchly, dokáže obslúžiť tisíce požiadaviek naraz.
- event-driven architecture
- menej náročný na systém (CPU, MEMORY)

Apache Server:

- modulárny - modul môže byť pridaný dynamicky
- ak sa zvýši záťaž, môžeme pocítiť nedostatok prostriedkov
- process-driven architektúra
- viac náročný na CPU a pamäť, práve vďaka procesom a vláknam.

Oba tieto webové servery majú svoje výhody a nevýhody. Hlavný faktor, ktorý by sme mali zhodnotiť je architektúra.

Apache spawnuje proces ktorý vie vytvoriť niekoľko vlakien, z ktorých každé vie obslúžiť maximálne jednu požiadavku (spojenie). To môže viesť pri veľkej záťaži k zvýšeným nárokom na prostriedky.

NGINX je schopný obslúžiť s jedným vláknom viacero požiadaviek, tým pádom šetrí výpočtové prostriedky.

Apache má väčšiu komunitu (aj vďaka tomu že je dlhšie na svete) ale NGINX sa každým dňom dostáva do popredia. NGINX je novšia a modernejšia technológia.

Zhodnotenie:
NGINX cesta ktorou by sme mali ísť. Je to moderné, ponúka to viac možností. Hlavne preferujem tento server kvôli event driven architektúre. nginx je na vzostupe.

Common UNIX Printing System - CUPS

Linuxová aplikácia ktorá sa inštaluje (apt install cups).

CUPS je de facto štandard a ine existujúce služby ani zďaleka nepodporujú toľkú funkcionalitu ako CUPS. Takisto je to default na Ubuntu. Keďže máme Ubuntu server tak nám to vyhovuje. komunikáciu medzi nasim backendom(API) a CUPS je možné riešiť pomocou Python knižnice pycups (cez tu kniznicu si vieme pytat aka je queue a pod. veci.)

Internet Printing Protocol ("IPP") je novodobý štandard, umožňuje manažovať tlačové joby a queues. Takisto narozdiel od ostatných protokolov, tento využíva obojsmernú komunikáciu, teda máme väčšiu kontrolu nad tým čo sa práve tlačí a pod.

CUPS vieme ovládať aj cez príkazový riadok. Vhodné na debugging.

MQTT Client

MQTT ako machine to machine protokol
Motivácia:

- Potrebujeme vedieť akým spôsobom budeme posielať dáta z mqtt brokera a zapisovať ich do DB.

Možné riešenie:

- paho-mqtt (pip install paho-mqtt)
- MQtt Client - python aplikácia ktorá bude sprostredkovať komunikáciu medzi DB a brokerom.
- Môj návrh je spustiť túto aplikáciu ako systemd službu. Pri boote sa zapne a bude bežať v nekonečnej slučke.
- Ako systemd službu ju budeme potom vedieť jednoducho vypnúť, zapnúť, reštartovať. V skratke - ovládanie na úrovni systémových služieb, ako nginx, cups, atd.

Aktívne koncové zariadenia

V rámci zimného semestra sme sa zamerali aj na voľbu technológií, ktoré využívame pri implementácií aktívnych koncových prvkov v našej infraštruktúre. Tento proces je možné rozdeliť na dve časti:

1. Výber mikroprocesora (alebo aspoň architektúry) na ktorom postavíme riadenie našich IoT senzorov
2. Výber operačného systému alebo základu pre firmvare

Nevyhnutné na koncové zariadenie (SW aj HW) by sme vedeli zhrnúť v nasledujúcich bodoch:

- nízka energetická náročnosť (ideálne ak vybraný hardvér podporuje režim hlbokého spánku)
- overené riešenia (veľkosť komunity, príklady reálnej implementácie, technológie “overené časom”)
- jednoduchá implementácia softvéru (existencia dokumentácie, SDK)
- cena
- podpora pre WiFi
- podpora IPv6
- open-source (pre SW)
- ISP zbernica
- I2C zbernica
- UART zbernica (pre jednoduchší debugging aspoň 2x)
- Výstupný 5V napájací pin

Zároveň sme definovali aj voliteľné požiadavky:

- podpora pre POSIX operačný OS (zadanie od vedúceho projektu a.k.a klienta)
- analógové piny (aby sme sme nemuseli realizovať ADC)

Na základe hore opísaných podmienok sme uvažovali nad nasledujúcimi HW riešeniami:

- STM32
- ESP32
- Raspberry Pi Zero
- Arduino Uno

Po analýze sme skončili pri výbere ESP32 a to konkrétne nasledujúcich implementácií:

- NodeMCU-32S ESP32
- ESP32-DevKitC-32U

V niekoľkých iteráciach sme skúšali a analyzovali nasledujúce operačné systémy alebo aplikačné rámce pre vnorené aplikácie:

- RiotOS (veľmi komplikované pre náš use-case, na modernom macOS prakticky nepoužiteľné)
- OpenWRT (nemá ovládače pre ESP32)
- OpenRTOS (v princípe platné riešenie ale nespĺňa voliteľnú špecifikáciu pre POSIX operačný systém a na bežné pokrytie prípadov použitia)
- MicroPython (toto by mohla byť cesta)
- ESP-SDK (toto je tiež cesta)

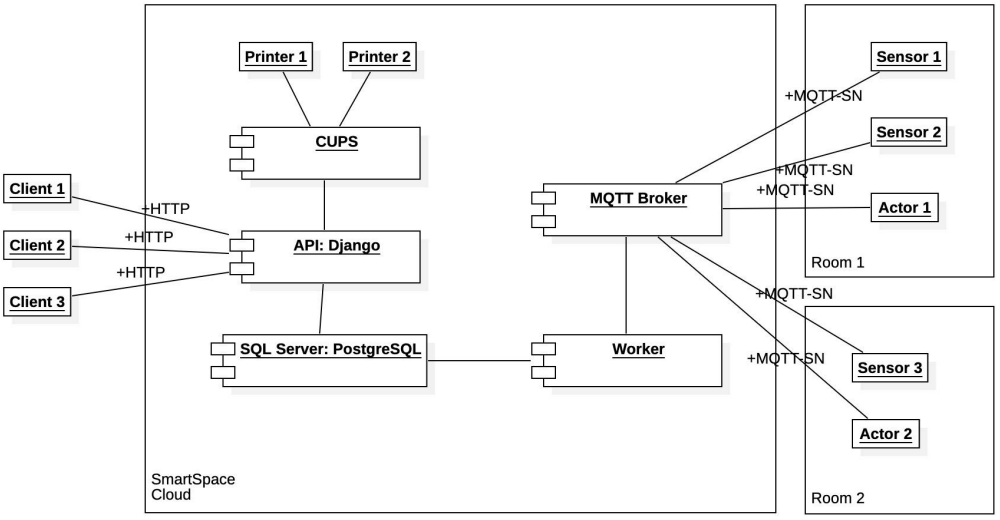
- Yocto Project (asi jediná možnosť ako spustiť Linux jadro na ESP32, komplikované ale splní zadanie vedúceho projektu - klienta)

Architektúra softvéru

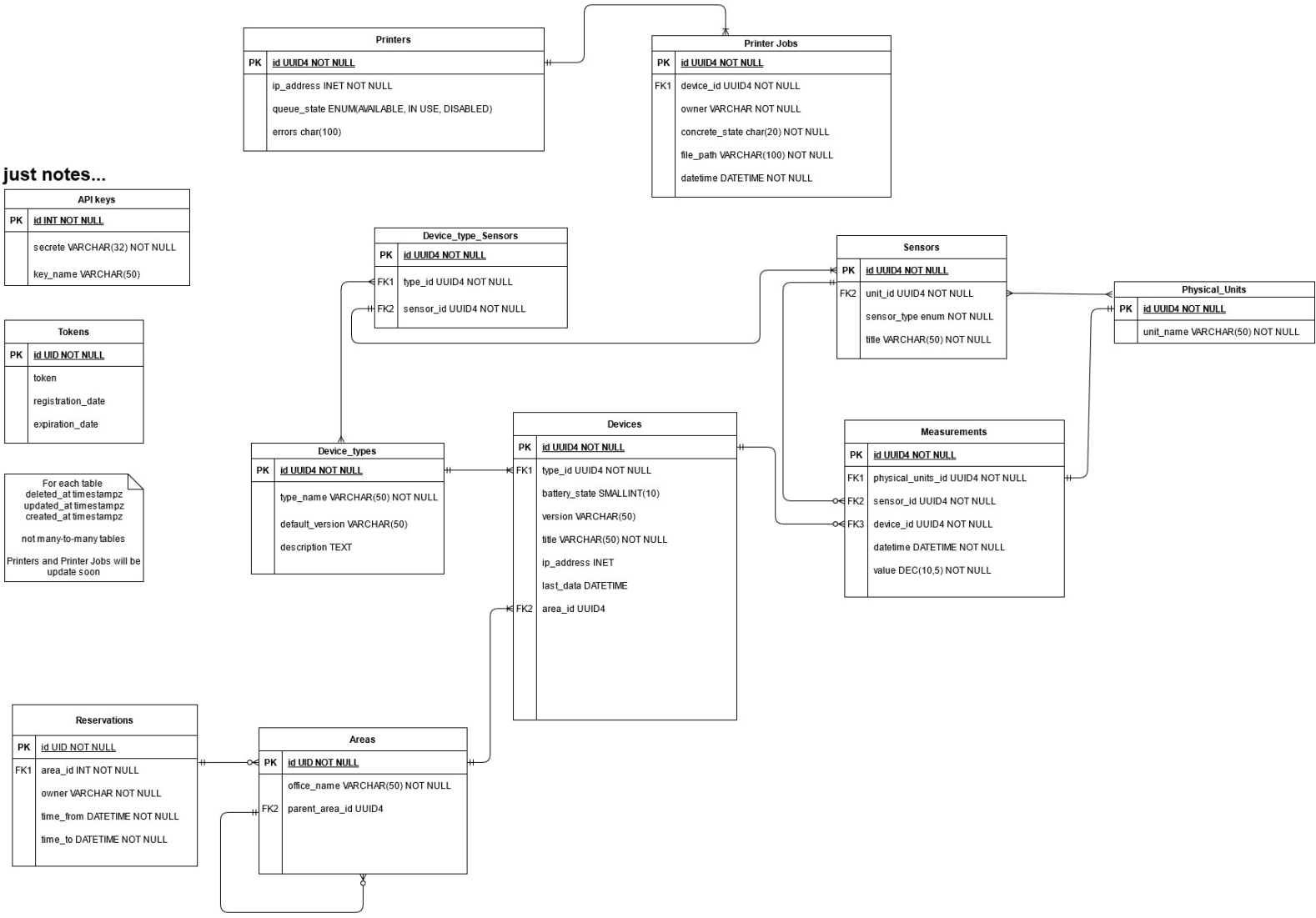
Aplikačné komponenty

Na obrázku nižšie vidíme topológiu softvérových komponentov, ktorá sa skladá z nasledujúcich častí:

1. CUPS - printer server na riadenie zdieľanej tlače
2. API - implementácia biznis logiky a komunikácia na klientov a databázu
3. SQL Server - relačné dátové úložisko
4. MQTT Broker - message broker pre IoT zariadenia
5. Worker - komponenta, ktoré spracováva výstup z IoT zariadení a preposiela na API
6. Sensor - aktívne/pasívne IoT zariadenia
7. Client - používatelia a administrátori



Dátový model



Na obrázku vyššie je predstavený aktuálny dátový model pre system SmartSpace. Pri jeho zostaveni sa vychádzalo hlavne z požiadaviek, ktoré boli kladené na projekt. Hlavné z nich sú:

- Pridávanie nových zariadení a typov zariadení. Manažovanie už existujúcich senzorov a zariadení.
- Rezervacia miestností
- Vzdialená tlač na univerzitných tlačiarňach

Hlavná časť modelu predstavuje prvky potrebné na riadenie zariadení , ktoré vykonávajú merania. Definovali sme nasledujúce prvky:

1. Devices - predstavuje jedntolive zariadenie, ktoré je pripojené k siete, má jeden alebo niekoľko senzorov a má určitý typ.
2. Device_types - predstavuje typy zariadení v našom projekte, obsahuje zakladnu konfiguráciu pre typ a senzory priliehajúce každému zariadeniu.
3. Sensors - jednotlivé senzory, ktoré obsahujú fyzikálne jednotky v ktorých budú senzory merať, type senzora a príslušný názov.

4. Measurements - tato štruktúra predstavuje jednotlivé meranie ktoré sú vykonané na zariadení pomocou konkrétneho senzoru
5. Physical Units - fyzické jednotky na meranie.

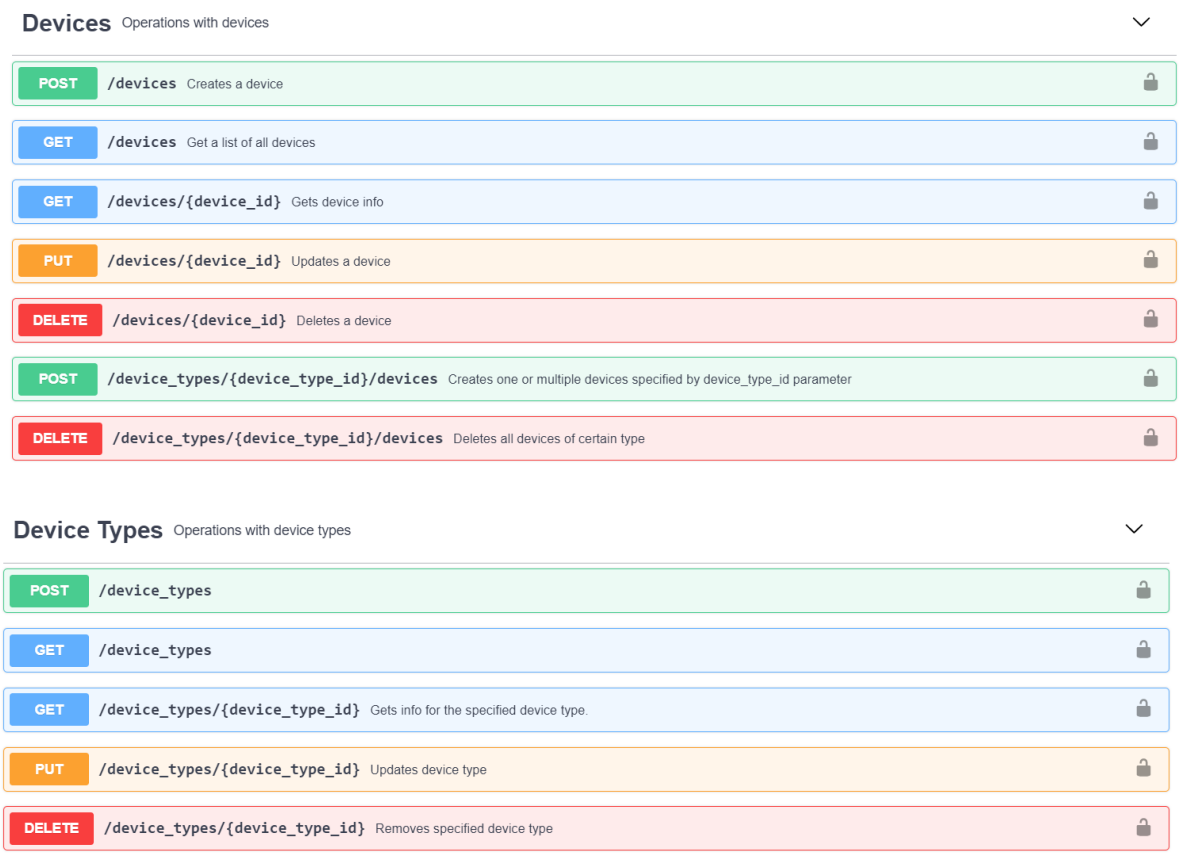
Ďalšia časť obsahuje jednotky, ktoré reprezentujú miestnosti a ich samotnú rezervaciu. Areas predstavuje miestnosť , ktorá má svoje meno, stav a zariadenie z ktorého je prístupná. V niektorých prípadoch potrebujeme rezervovať miesto v miestnosti, a preto miestnosť môže obsahovať odkaz na iný objekt miestnosti, ktorý bude aj miestom.

Posledná časť je venovaná tlačiarňam. Obsahuje zatiaľ iba dva objekty - tlačiareň (printer) a požiadavka na tlač (printer job). Printery obsahujú IPv6 adresu, aktuálny stav, pole pre chybové hlášky. Požiadavka na tlač obsahuje údaje potrebné pre tlač a meno človeka, ktorý tlač zadal. Táto časť je ešte iba v stave návrhu a s veľkou pravdepodobnosťou sa bude meniť v budúcnosti.

Okrem toho dátový model obsahuje API kľúče a Tokeny, ktoré budú slúžiť na autentifikáciu používateľov a bezpečnostný prenos údajov.

Implementované Rest API

Dokumentáciu REST API sme písali v prostredi Swagger. Máme v nej viacero endpointov. Každú tabuľku v dátovom modele sme riešili pomocou CRUD. Riadili sme sa OpenAPI 3.0 špecifikáciou.



Measurements Operations with measurements

GET	/devices/{device_id}/last_measurement	Get last measurment of device
GET	/devices/{device_id}/measurements	Get list of measurments based on time period
POST	/measurements	creates list of measurements

Reservations Reservations operation

POST	/reservations	Creates reservation
PUT	/reservations/{reservation_id}	Updates reservation
DELETE	/reservations/{reservation_id}	Deletes reservation
GET	/reservations/{area_id}	Return all reservations for certain area

Sensors Sensors Operations

POST	/sensors	Creates list of sensors
PUT	/sensors/{sensor_id}	Updates the sensor
GET	/sensors/{sensor_id}	Gets single sensor by id
DELETE	/sensors/{sensor_id}	Deletes single sensor by id
GET	/sensors/filter	Gets multiple sensors by type
DELETE	/sensors/filter	Deletes multiple sensors by type

Authentication

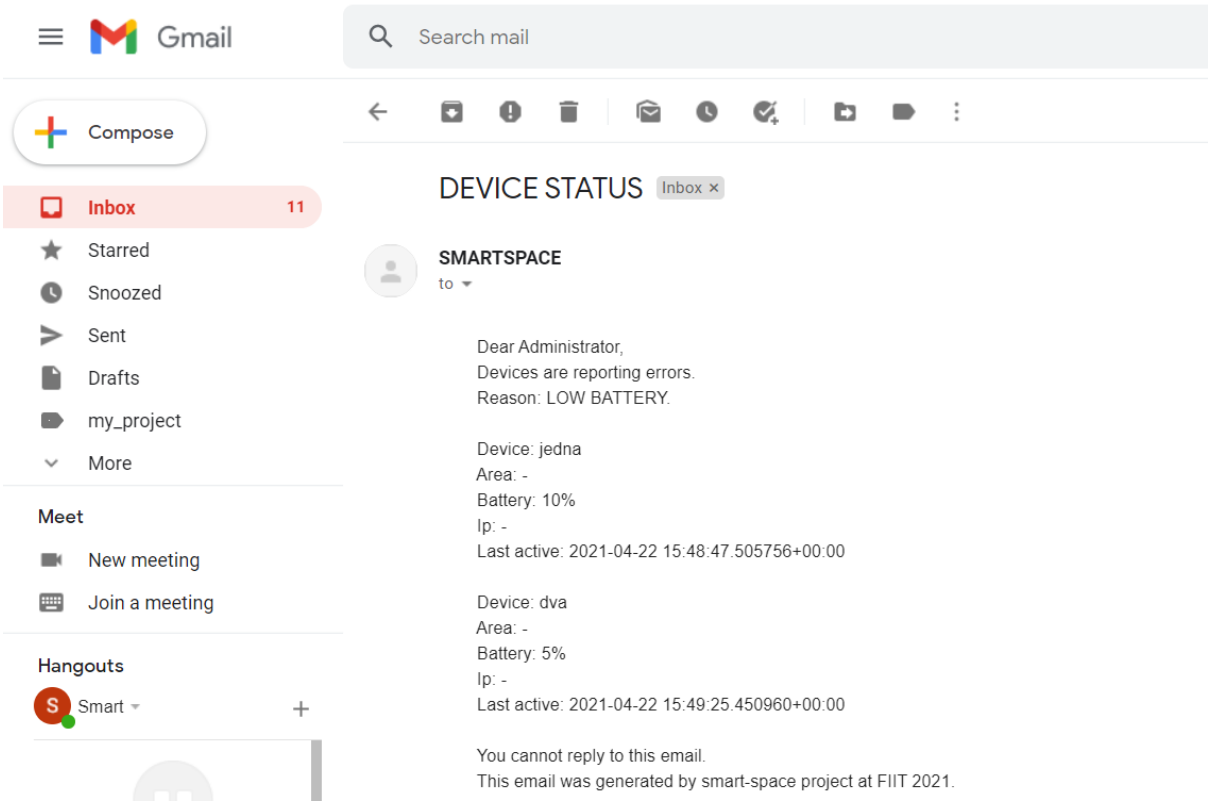
POST	/authenticate	authenticates the user
POST	/authenticate/refresh	refresh of the access token

Areas

POST	/areas	Creates new area
GET	/areas	Get list of all areas
GET	/areas/{area_id}	Gets detailed information about area
PUT	/areas/{area_id}	Updates area
DELETE	/areas/{area_id}	Deletes area

Notifikácie pomocou mailov

Naše riešenie obsahuje automatickú mailovú komunikáciu pre informovanie administrátora o akciach a v stave v sieti. Notifikácia sa posiela 1x do dňa napr. či má dané zariadenie slabú alebo dobrú baterku alebo pre kontrolu posledného dátumu kedy dané zariadenie nameralo nejakú hodnotu. Taktiež tieto notifikácie slúžia na uistenie administrátora, že dané zariadenie je v poriadku. Táto forma komunikácie je efektívna a dáva vedieť administrátorovi aktualny status aj bez toho aby sa pripájal na hlavnú stránku.



Nasadenie aplikácie

Aplikácia pozostáva z niekoľkých komponentov. Každý komponent beží v oddelenom prostredí prostredníctvom dokerizácie.

V produkčnom nasadení boli použité nasledujúce verzie kontajnerov:

- python:3.7-alpine
- node:14.16.1-alpine
- nginx:1.15-alpine
- postgres:12-alpine

Pokiaľ ide o verzie kontajnerov, tak v prípade Node a Nginx išlo o najaktuálnejšie verzie toho času. Verzie Postgresql a Python kontajnera vyšli z analýzy zo zimného semestra. Všetky

kontajnery sú stavané na minimálnej verzii Alpine Linuxu vďaka čomu sa celková implementácia pohybuje len v desiatkach megabajtov.

Samotné nasadenie prebieha manuálne po každom šprinte sa na server z GITLAB repozitára stiahla najnovšia verzia a pomocou nasledujúceho príkazu sa nasadil najnovší kód.

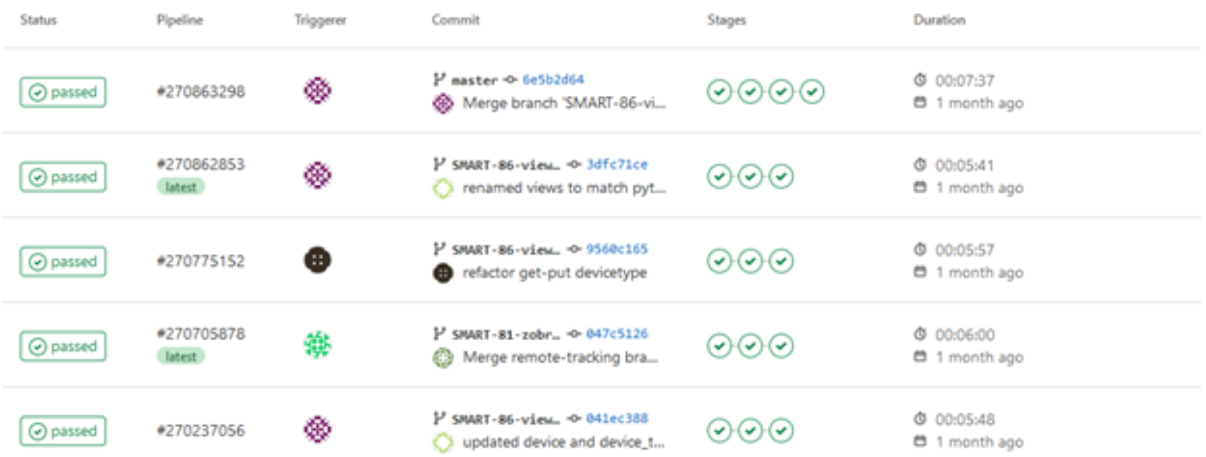
$ docker-compose up --force-recreate --build -d

Pre každý komponent je vytvorený samostatný Dockerfile a na tieto súbory sa odkazuje v jednom docker-compose súbore.

Continuous Integration/Continuous Delivery

V rámci základného CI/CD bola implementovaná testovacia PIPELINE, ktorej úlohou bolo testovať publikovaný kód. Rámci toho sa spúšťali jednotkové testy.

Táto služba bežala v rámci nášho GitLab VCS.



Hardvér komponenty

Pre ovládanie a monitorovanie priestorov bolo potrebné analyzovať a navrhnúť riešenia pre zadané akcie, ktoré sa budú vykonávať:

1. **Monitorovanie obsadenosti miesta**

   Možné riešenie:
   - IR senzor
   - Silový senzor
   - Senzor vibrácií

Po analýze všetkých riešení je najlepšou možnosťou IR senzor. Oproti senzoru vibrácií je jednoduchšou a efektívnejšou voľbou. Oproti silovému senzoru je IR senzor efektívnejším a ekonomickejším riešením.

2. Monitorovanie obsadenosti miestnosti

- kartičkový systém
- dáta z monitorovania obsadenosti miesta
- senzor pohybu pri dverách
- termokamera

3. Uzamknutie miestnosti

- kartičkový systém

4. Monitorovanie kvality ovzdušia miestnosti

Možné riešenie:

- Senzor all-in-one
- osobitné senzory pre vlhkosť, CO2, teplotu

All-in-one senzor, ktorý dokáže merať všetky potrebné parametre pre kvalitu ovzdušia je veľmi dobrým riešením. Výhodou je kompaktnosť, jednoduchosť riešenia. Na druhej strane je toto riešenie oproti jednotlivým senzorom pre meranie kvality ovzdušia nákladnejšie

5. Meranie telesnej teploty pri vstupe do miestnosti

Po diskusii o meraní teploty sme sa zhodli, že teplota človeka sa bude merať pred vstupom do budovy. Po odmerani teploty sa detekuje, či má človek teplotu správnu alebo zvýšenú. Pri zvýšenej teplote bude dodatočným zariadením (reproduktor) oboznámený o tomto stave, nakoľko mu nebude umožnený vstup do budovy. Teplotu si bude môcť odmerať neskôr.

- Reproduktor (piezo)
- Reproduktor (klasický)
- Teplomer pre meranie telesnej teploty

6. Displej pre oznamy

Pri vstupe do miestnosti bude umiestnený LCD displej, ktorý zabezpečí zobrazenie všetkých dôležitých informácií týkajúcich sa miestnosti pri ktorej bude displej umiestnený. Zobrazovať sa bude teplota, kvalita ovzdušia, obsadenosť a prípadné rezervácie v najbližšej dobe.

- 16x2 LCD displej
- LCD dotykové displeje (rôzne veľkosti)

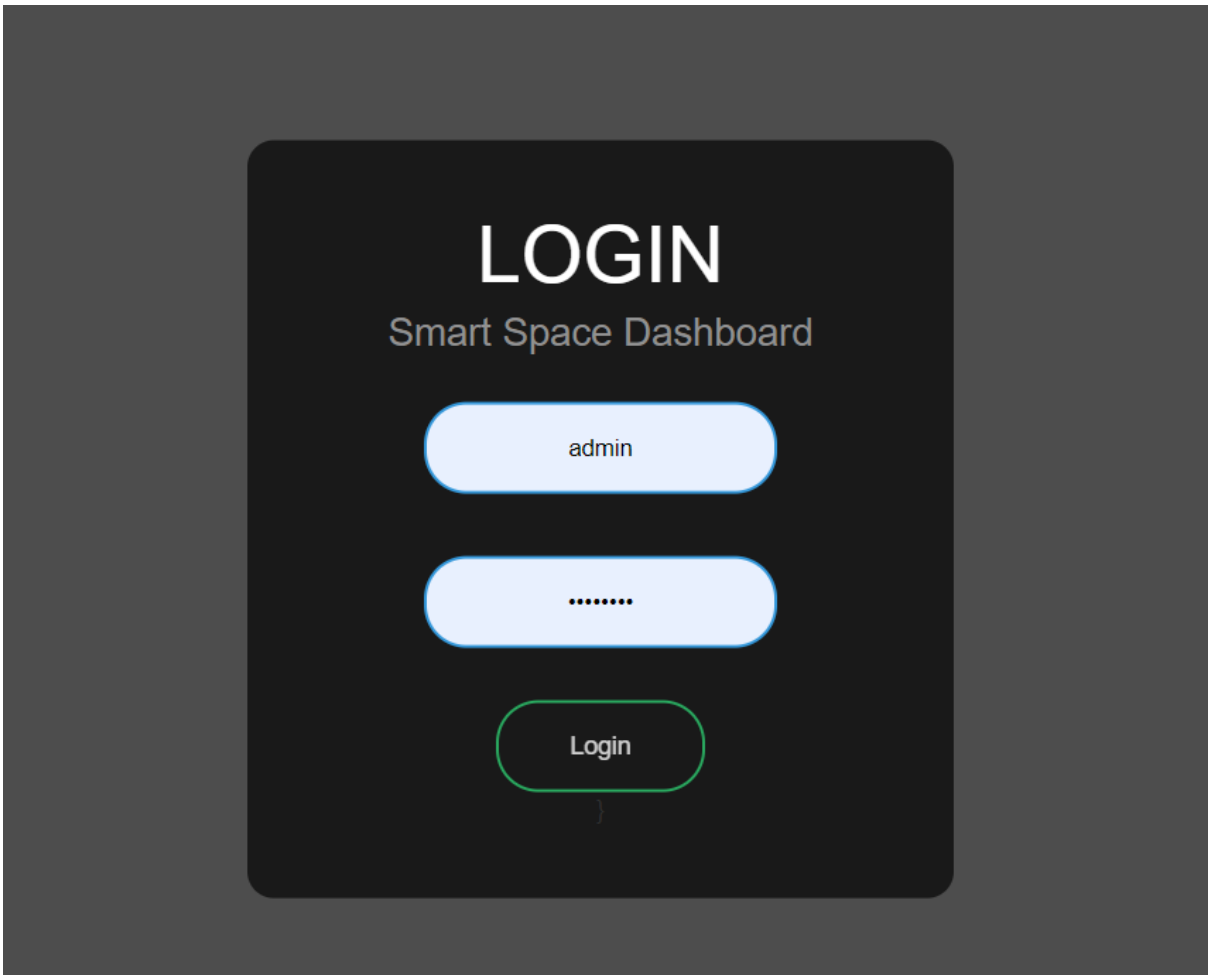
Po analýze možných a dostupných riešení, sa objednal nasledovný hardvér:

Názov	Cena € (ks.)	Počet	Popis
PIR modul HC-SR501	2.13	3	Senzor pohybu a vibrácií
DHT22	6.9	3	Senzor na meranie vlhkosti a teploty
SNS-MQ135	4.97	3	Senzor na koncentráciu plynov (CO, CO2, etc.)
DS18B20	1.97	5	Digitálny termometer
Raspberry Pi 4 Model B	68.9	1	Single-board počítač
WS-16239	25.50	1	Dotykový displej
LS5090T-3F-R4	2.68	2	Reproduktor
Sada rezistorov	3.64	1	Metaloxidové rezistory rôznej impedancie
ASM-1900136-01	4.75	1	Raspberry Pi 4 Model B priesvitný obal
Napájací adaptér	10.32	1	Napájací adaptér 5/3.3V
Samsung MicroSDXC 128GB	19.08	1	MicroSD pamäťová karta (128GB)
Elecrow Electronic Kit	15.95	1	ACD, káble, breadboards, LEDs
NodeMCU-32S	10.8	5	ESP32 vývojová doska (ESP-WROOM-32)
ESP32-DevKitC-32U	9.97	3	ESP32 vývojová doska (ESP32-WROOM-32U)

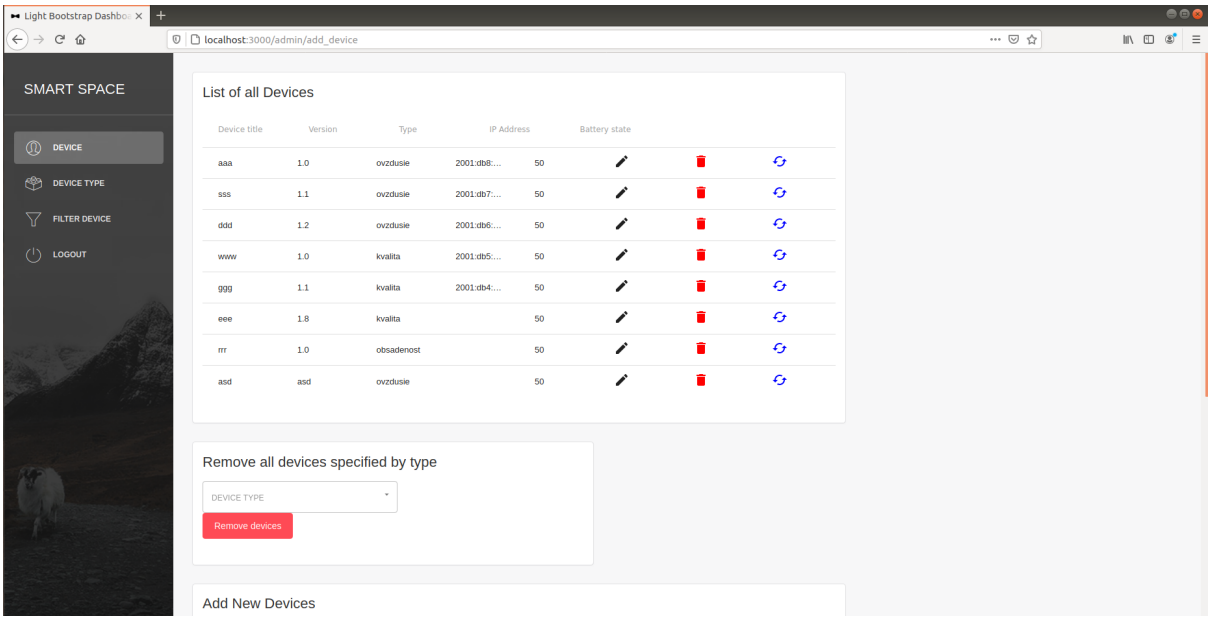
Admin stránka

Výsledkom našej práce je aj Dashboard pre Admin používateľa. Ten si momentálne môže pridať alebo vymazať zariadenia, a takisto definovať nové typy zariadení. Ako template sme použili Material Dashboard React, ktorý je dostupný zadarmo na oficialnej stránke creative-tim.com

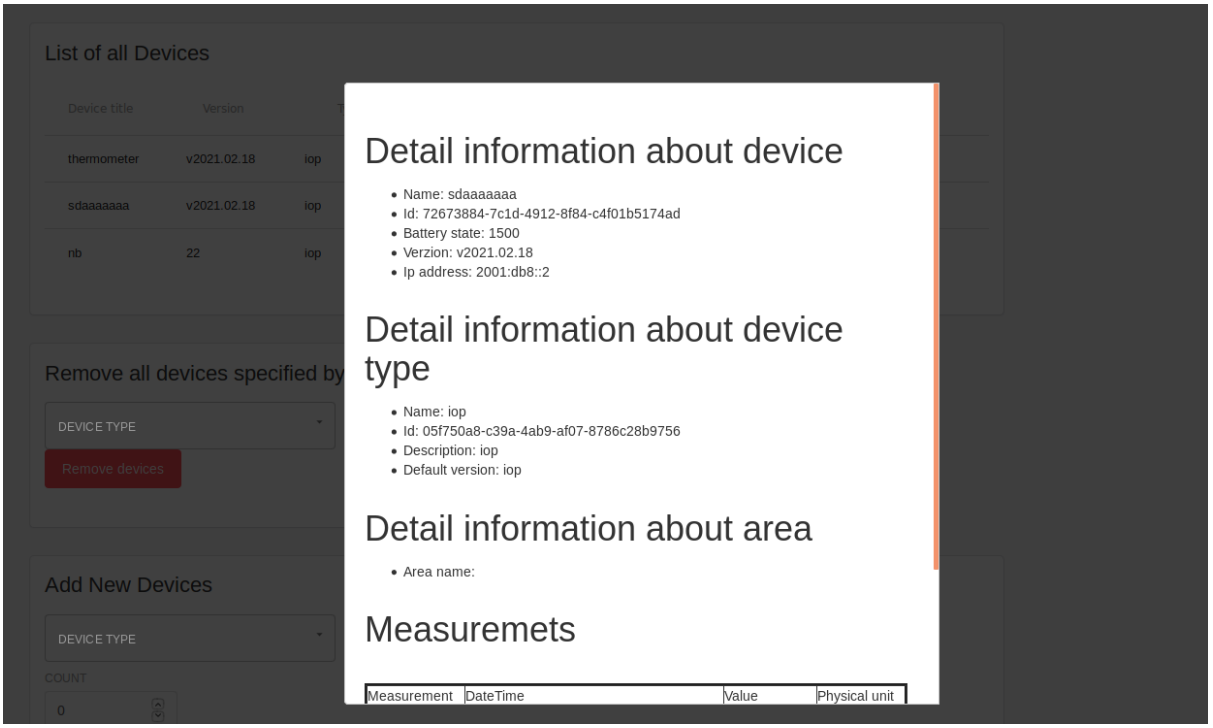
login

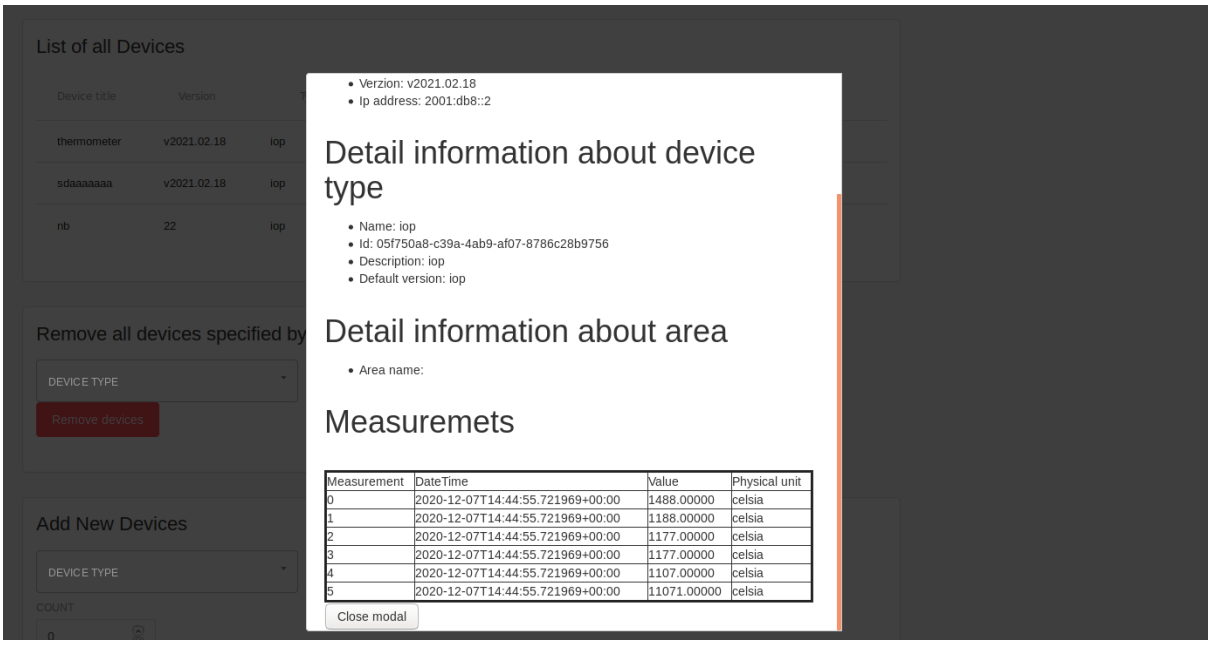


Na obrázku nižšie vidíme kartu v Admin stránke s názvom DEVICE. V tejto karte vie admin sledovať všetky dostupné a aktívne zariadenia, ktoré má nasadené s ich popisom.
Taktiež vie admin vymazať špecifické zariadenia podľa konkrétneho typu.

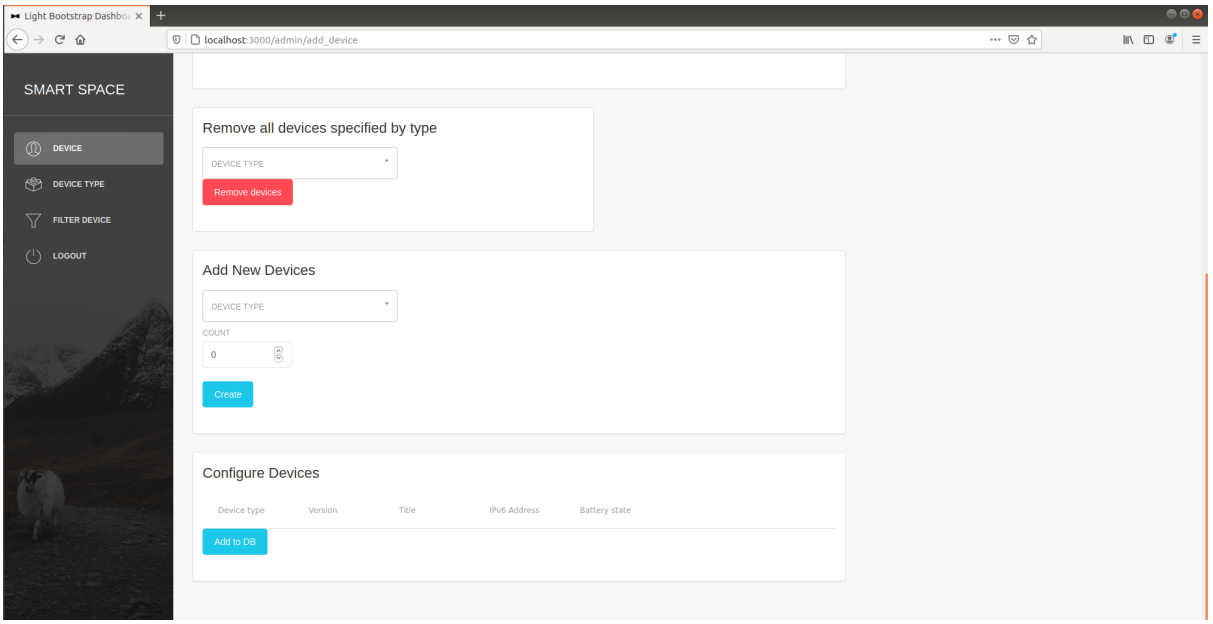


Ak sa chce admin zobraziť konkrétne zariadenie, dokáže otvoriť detail zariadenia. V tomto detaile si admin dokáže zistiť veľmi presné informácie o zariadení, históriu nameraných hodnôt z toho zariadenia, oblasť kde sa zariadenie vyskytuje a aký má typ.

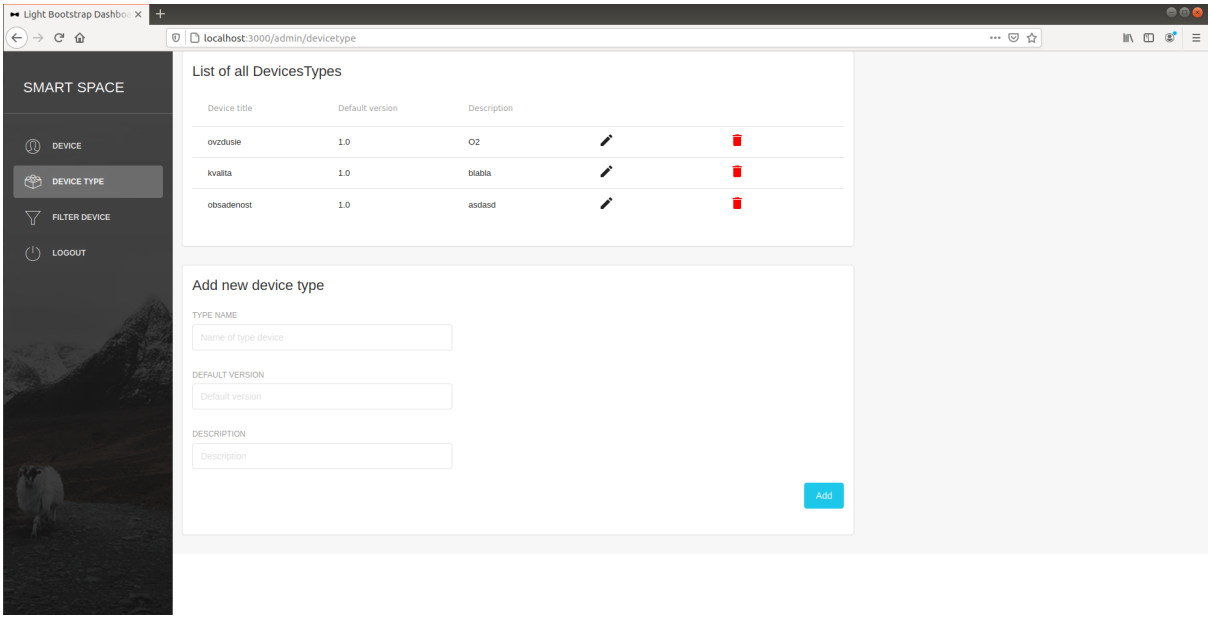




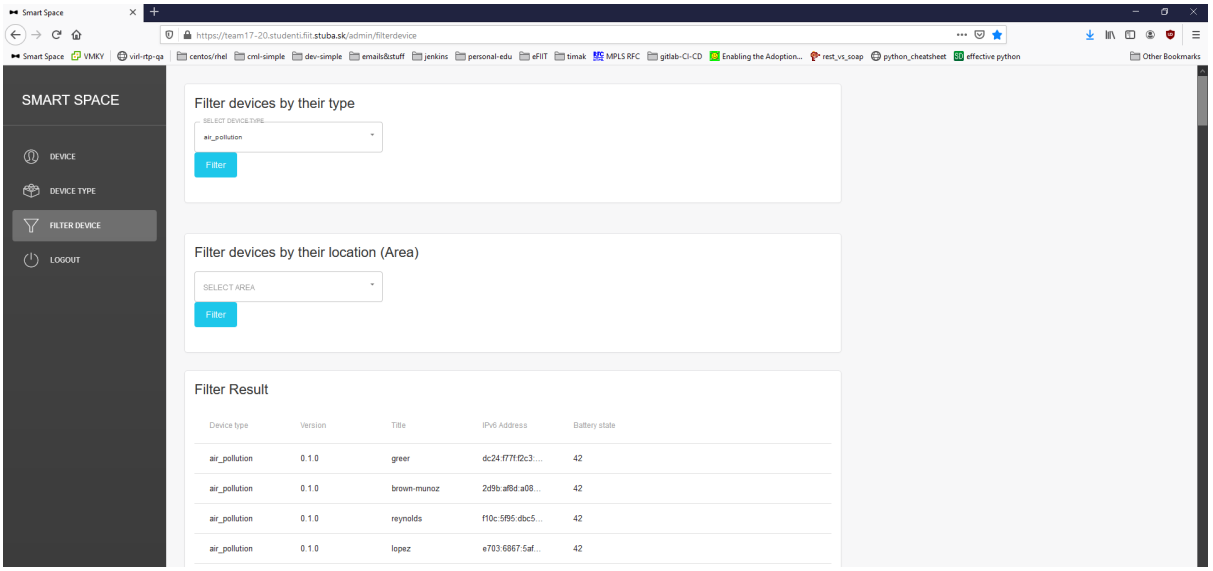
V karte DEVICE po posunutí smerom nadol vie admin pridávať zariadenia na základe vytvoreného typu zariadenia. Pomocou toho vie admin pridať až niekoľko zariadení toho istého typu naraz.
V časti CONFIGURE DEVICE vie novovytvorené zariadenia došpecifikovať podľa potrieb.



V karte DEVICE TYPE vie užívateľ jednoducho pridať nový typ zariadenia, pomocou ktorého bude vedieť ďalej vytvárať jednotlivé zariadenia. Taktiež má admin prehľad o vytvorených typoch, kde ich môže aj vymazať.



V karte FILTER DEVICE vie admin filtrovať svoje zariadenia. Tento filter dokáže uplatniť pre hľadanie na základe typu zariadenia alebo na základe oblasti. Filtrovať sa môže len na základe jednej veci v jednom momente. Vyfiltrované zariadenia si môže pozrieť v tabuľke pod filtrom.



Príloha A: Spätná väzba k projektu SmartSpace

Prosím, vaše hodnotenia rozdeľte do týchto troch hlavných častí. Každú časť vo vašom hodnotení očíslujte (1. Fronted , 2.Backend , 3. Celkový pocit).

1. **Fronted**
    a. popíš čo sa ti páči, respektíve nepáči na hlavnej stránke pre Administrátora, čo by sa mohlo zlepšiť, čo by sa malo pridať a pod.
    b. ako sa naviguje
    c. prihlasovanie, resp. odhlasovanie
    d. notifikácie, chybové hlášky

2. **Backend**
    a. keďže ako konečný užívateľ vieš ťažko hodnotiť efektivitu backendu, prosím opíšte prípadné chybové stavy ako napríklad to, že server vrátil chybový stav 500 - SERVER ERROR

3. **Celkový pocit z produktu**
    a. v tejto časti popíš svoj celkový pocit z produktu a jeho prípadný prínos ak by bol nasadený do praxe. (tento dokument uvažuje, že si bol oboznámený s podstatou tohto projektu)
    b. Na záver ohodnoť projekt na stupnici od 1-5, kde 1 je najhoršie hodnotenie a 5 najlepšie.

Hodnotenie užívateľa A (Product owner)

1. **Frontend**
    - Pekné prihlasovanie do systému.
    - Obrazovky sú prehľadné a zrozumiteľné, takisto navigácia je fajn a človek sa nestratí.
    - Niektoré tlačidlá, vôbec nedávajú náznak že sa dajú klikať a teda užívateľovi sa tak môžu javiť ako nefunkčné. Zároveň to znižuje sebavedomie užívateľa a celkový UX je horší. Konkrétne ide o tlačidlá v listoch.
    - Ikonka --  -- na ktorú keď kliknem vybehne detail zariadenia by mohla vyzerať inak. Momentálne mi to skôr pripadá ako reštart zariadenia ako zobrazenie detailu.
    - Na každú akciu, ktorú užívateľ vykoná by som mal dostať spätnú väzbu, či už v podobe notifikácie alebo dialógového okna. Takto bez notifikácií človek nevie, či sa to vykonalo alebo nevykonalo. Takisto ho to môže dostať do problémov.
    - Detailne zobrazenie zariadenia je veľmi triviálne a ako užívateľ by sa mi páčilo ak by to malo nejaký dizajn, a nech to nie je len modálne okno s textom. Na druhu stranu mi to poskytuje všetky potrebné informácie čo vnímam pozitívne.

- Pridávanie zariadení, typov zariadení je implementované pekne. Čo ale vnímam negatívne je to že po pridaní / odobratí vecí sa zmeny neodrazia automaticky v liste. Je potrebné obnovenie stránky.

2. **Backend**
   - Po skúšaní rôznych kombinácií som nenašiel na žiadnu fatálnu chybu z ktorej by som sa nevedel dostať, čo vnímam veľmi pozitívne.
   - Stranka fungovala svižne a takisto som nikdy nemal pocit že je pomalá.

3. **Celkový pocit**
   Tento projekt to nemal jednoduché v podmienkach akych pracoval, keďže celý tím pracoval z domu a povaha tohto projektu si v niektorých častiach vyžadovala fyzickú prítomnosť. Po konzultácií s tímom, kde mi bolo vysvetlene podrobne o čo ide, verím že väčšina práce bola odvedená na strane ktorú užívateľ nevidí. Či už ide o prácu na databáze, alebo hardvéri.
   Preto tento projekt hodnotím znamkou 4. Teda veľmi dobre.

Hodnotenie užívateľa B (spolužiak)

1. **Front-End**
   - Je trochu zvláštne, že hneď po prihlásení sa používateľovi zobrazí prázdna obrazovka
   - Navigácia je pomiešaná s action tlačidlami
   - Chyby pri prihlasovaní nie sú dialógové okná ale alert od prehliadača (čo je trochu škaredé)
   - Niektoré tlačidlá sú mätúce (restart button, ktorý otvorí detail)
   - Tabulky by mohli byť prepracovanejšie (absencia vyhľadávania a stránkovania)
   - Odstránenie prebieha bez spätnej väzby (žiadne potvrdzovacie okno)
   - Kreatívne riešenie v zmysle delenia navigácie
   - Viac by sa mi páčil domain driven spôsob navigácie
   - Na mobilnom telefóne sa nezobrazí navigácia

2. **Back-end**
   - Štruktúra dát vyzerá rozumne
   - Aplikácia funguje rýchlo
   - Súčasťou aplikácie je posielanie e-mailových notifikácií
   - Moderná architektúra
   - Autodiscovery redukuje čas strávený v administrácii
   - Swagger dokumentácia
   - Nenarazil som na žiadne server-side chyby pri testovaní (5xx)

3. **Celkový pocit**

   Vhľadom na komplikované pandemické podmienky si tím musel poradiť s návrhom laboratória bez fyzického prístupu. Šikovne dokázal nahradiť potrebné vstupy generátorom dát. Navrhované riešenie je modulárne a veľmi jednoducho škálovateľné. Tím sa hral s technológiami tak, aby bol projekt relevantný aj v budúcnosti. Komunikácia s nimi bola efektívna a príjemná. Hodnotím známkou 4.

Hodnotenie užívateľa C (spolužiak)

1. **Front-End**
   - prihlasovacie okno má pekný dizajn a animácie, avšak som si všimol zátvorku na pod prihlasovacím tlačidlom, to celkom kazí dojem
   - vytkol by som, že po prihlásení sa používateľovi zobrazí prázdne okno, privítal by som aspoň nejaký prehľad, upozornenia alebo aspoň uvítací text
   - navigačný bar pre mňa pôsobí intuitívne a prehľadne, odhlasovacie tlačidlo by som avšak presunul na iné miesto, napr. do pra''eho horného rohu
   - tabuľky sú prehľadné avšak mohli by byť lepšie naformátované aby zobrazovali celý názov a nie len prvých pár písmen, tak isto aj typ meraného atribútu
   - pri scrolovaní je tabuľka moc dlhá a presahuje celú stránku, ocenil by som rozdelenie na stránky alebo aspoň nech sa stránka prispôsobí veľkosti tabuľky
   - odstránenie alebo úprava veci v tabuľke je intuitívna a človek to vie rýchlo nájsť
   - pri týchto aktivitách mi avšak chýba istá notifikácia alebo spätná väzba o zmazaní alebo úprave zariadenia, takto po odstránení si moc nie som istý či som ho zmazal a pre istotu ho musím ešte manuálne nájsť
   - chýba mi možnosť vyhľadávania napr. podľa názvu
   - páči sa mi tiež pridanie nového zariadenia, je to rýchle a nenáročné bez nejaký zložitých nastavení
   - stránka s filtrovaním zariadení, by sa dala podľa mňa spraviť aj krajšie. Nastavenie jednotlivých filtrov by som zjednotil do jedného formulára.
2. **Back-end**
   - svižný beh aplikácie, nemal som pocit že by mrzla, načítanie bolo vždy rýchle
   - oceňujem prepracovanú a prehľadnú dokumentáciu k backendu
   - veľkým plusom je posielanie e-mailových notifikácií
   - prehľadná štruktúra kódu
   - po dlhšom skúšaní som nenarazil na žiadnu chybu
3. **Celkový pocit**

   V aktuálnej pandemickej situácii sa mi nápad takéhoto riešenia páči. Možnosť sledovať stav a obsadenosť miestností vie dostatočne uľahčiť správu objektu. Zároveň som si vedomý, problémov ktorým musel tím čeliť pri vývoji tohto projektu. Konzultáciami som zistil, že sa s týmito problémami celkom popasovali. Nemožnosť

stretnúť sa bola určite z počiatku veľkou brzdou avšak chalani sa s tým celkom popasovali a vynaložili značne veľké úsilie. Výsledný projekt sa mi celkom páči ale chýbajú mi tam určite features, ktoré by zlepšili celkový pocit z aplikácie. Hodnotil by som známkou medzi 4 a 5.