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 projektoch 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.

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.

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 samotneho developera ktorý kód zverejnil.

Meetings

Pred každým meetingom si posielame pozvánku do kalendára, kde sa oboznamujeme s účasťou na danom meetingu. Na meetingy preferujeme platformu Google meet. Meetingy realizujeme 3x týždenne a to pondelok, štvrtok a piatok. Pri meetingoch s vedúcim projektu realizujeme aj zápisnicu, ktorá obsahuje najdôležitejšie body zo stretnutia.

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.

Sumarizácie šprintov

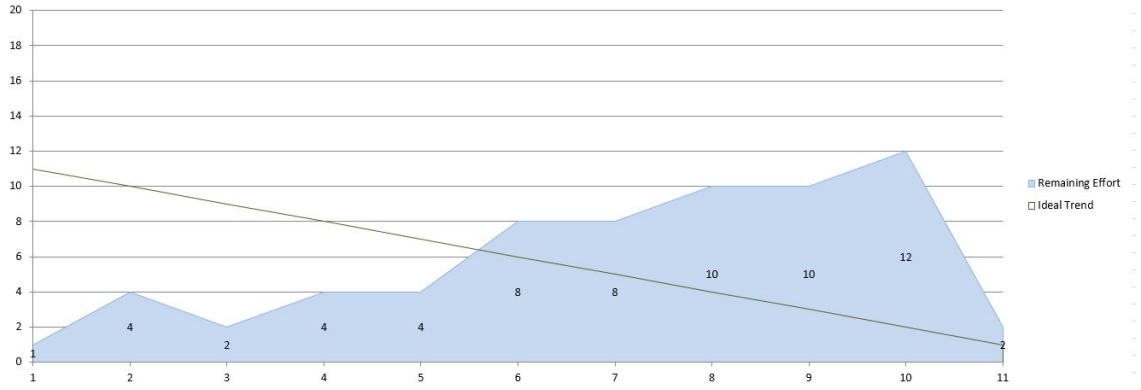
Šprint č. 1

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

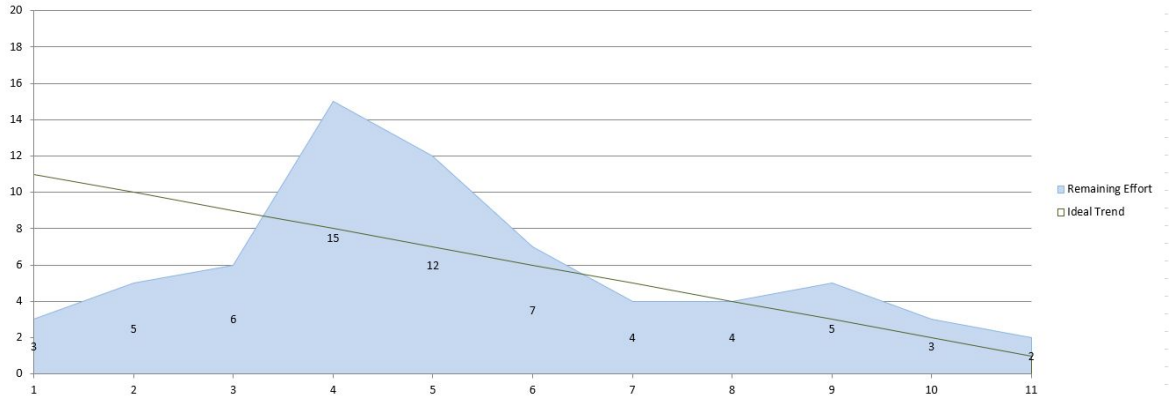
Sumár podielu práce

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

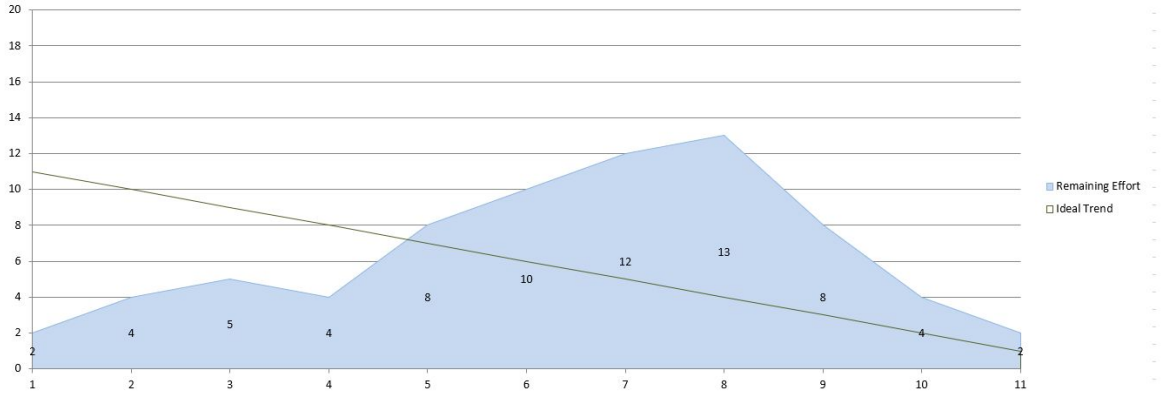
Šprint č. 3

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	25h
Milan Kaprál	1	12h
Tomáš Kovalčík	2	25h
Jakub Kubica	2	25h
Oleksandr Lytvyn	1	12h
Andrej Petričko	1	15h

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.

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 tickete #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 pripadných 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 ma 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

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.

Analýza technológie

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

Serverová implementácia

REST 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	✗	✓

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.

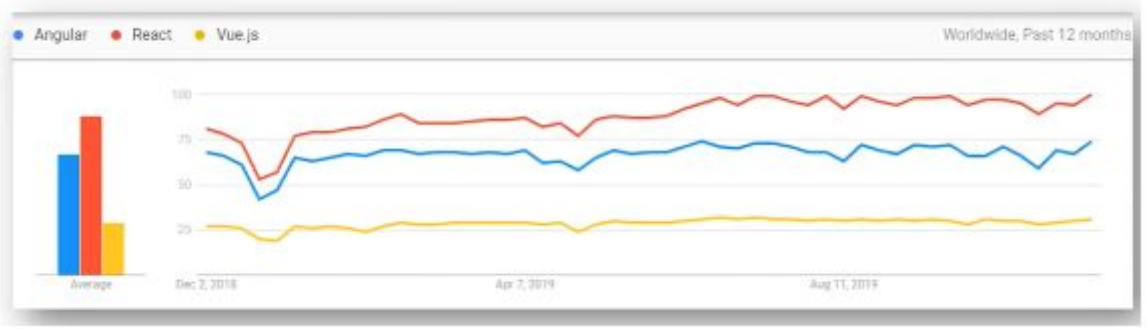
Výhody TimescaleDB sú:

- 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.

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 u 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 využijeme 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

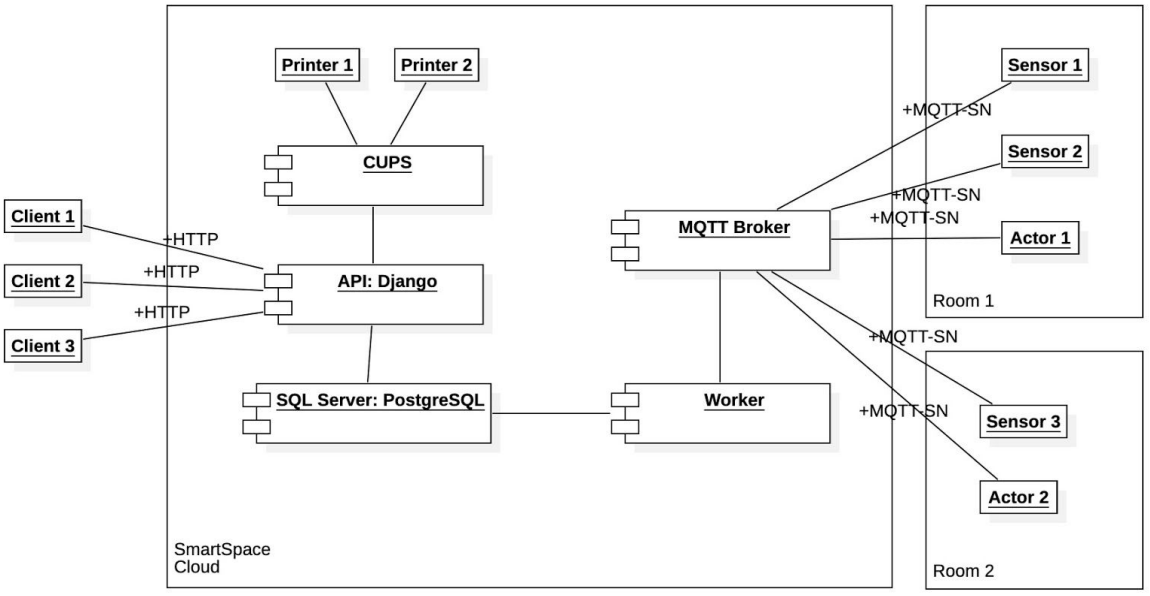
V niekoľkých iteráciach sme skúšali a analyzovali nasledujúce operačné systémy alebo aplikačné rámce pre vnorené aplikácie:

- 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)

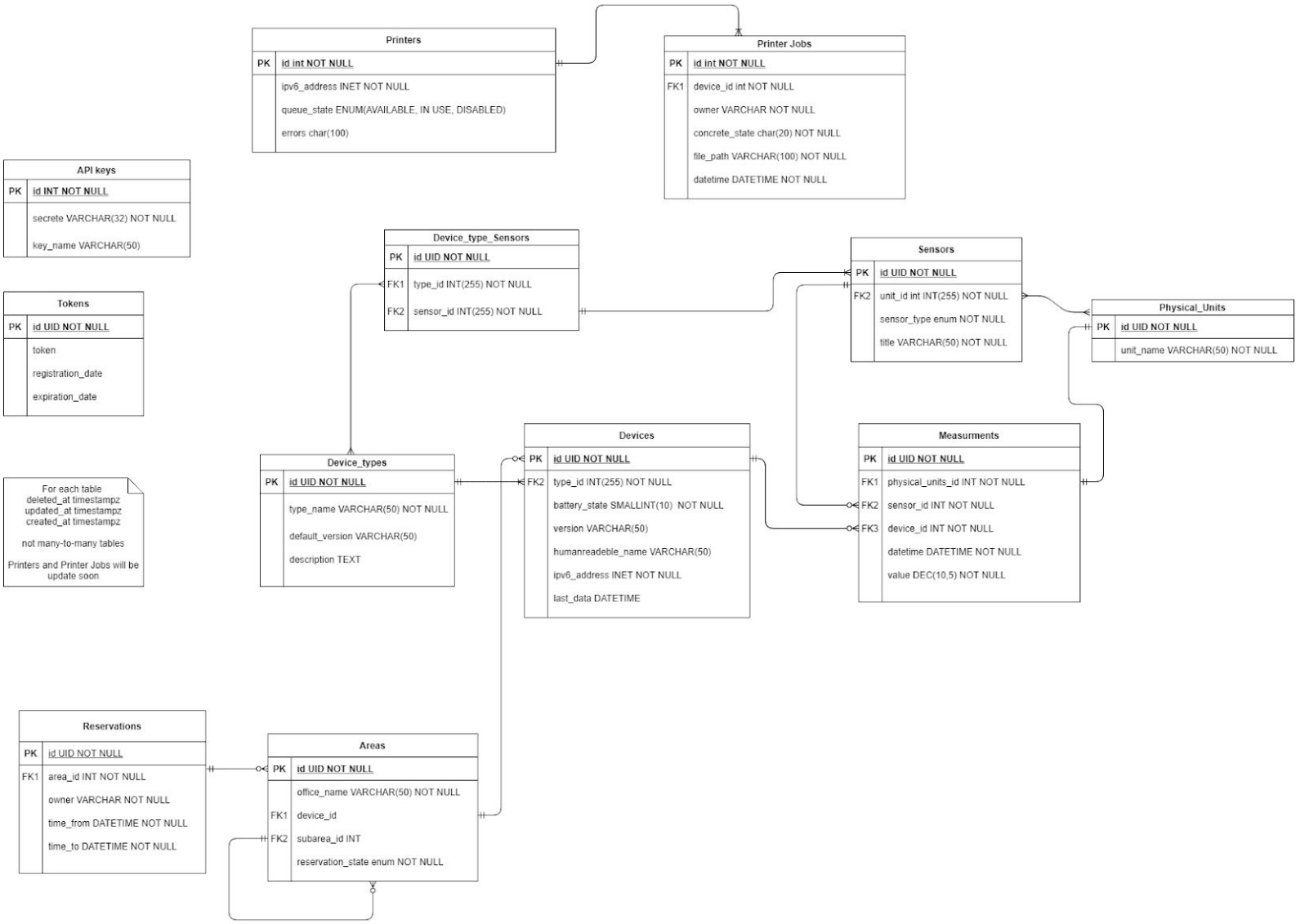
- Yocto Project (asi jediná možnosť ako spustiť Linux jadro na ESP32, komplikované ale splní zadanie vedúceho projektu - klienta)

Architektúra softvéru

Aplikačné komponenty



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.

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)