Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

SmartSpace2

Inžinierske dielo
Tímový projekt 2021/2022

Tím 03

Ing. Peter Kaňuch
Bc. Róbert Fajd, Bc. Adam Gajdošík, Bc. Lukáš Kurtiniak, Bc. Matúš Mikuláš, Bc. Juraj Ďurej, Bc. Juraj Skákala

http://team03-21.studenti.fiit.stuba.sk/
xgajdosika@stuba.sk

Obsah

1. Big Picture

1.1 Úvod

Tento dokument poskytuje popis k aktuálnemu stavu tímového projektu SmartSpace2 z pohľadu inžinierskeho diela. Obsahuje globálne ciele na zimný semester, diagramy a opis systému. Keďže táto téma je pokračovaním témy z minulého roku, obsahuje aj analýzu existujúceho riešenia.

1.2 Globálny cieľ na ZS

Počas zimného semestra 2021/22 sme si definovali tieto ciele, ktoré chceme splniť:

- získanie znalostí v systéme SCRUM a jeho využívanie popri vývoji
- úprava existujúceho kódu pre naše potreby
- analýza a vytvorenie systému pre 3 typy senzorov
    - na zaznamenávanie pohybu
    - na meranie kvality ovzdušia
    - na meranie teploty a vlhkosti
- zaznamenávanie meraní zo senzorov do databázového systému
- spustenie existujúcej časti na serveri
- vytvorenie webovej stránky pre interakciu so systémom
- vytvorenie tímovej stránky

1.3 Globálny cieľ na LS

V ZS sme analyzovali existujúce časti po predchádzajúcom tíme, ktorý na tomto projekte pracoval. Zistili sme, že je potrebné vytvoriť kód pre všetky senzory a zariadenia, ktoré budú slúžiť na meranie. Tento kód sme navrhli a z časti je už aj implementovaný. Taktiež sme začali pracovať na implementácií webového portálu pre administrátora systému, kde bude vedieť spravovať všetky jeho časti. Počas práce na webe sme odhalili viacero nedostatkov v API endpointoch, ktoré sme počas ZS nestihli opraviť a teda na nich budeme pracovať v LS.

Počas letného semestra 2021/22 sme si definovali tieto ciele, ktoré chceme splniť:

- doladenie chýb v riadení, ktoré sme identifikovali po ZS
- implementácia a testovanie kódu pre navrhované typy senzorov
    - refaktoring kódu
    - prispôsobenie pre upravené API endpointy
    - testovanie v miestnostiach
- úprava problematických API endpointov
    - po pôvodnom tíme sú niektoré funkcionality nelogické alebo nefunkčné
- zmena databázového modelu
    - databázový model v súčasnom stave neumožňuje jednoduché spojenie meraní s konkrétnymi senzormi a oblasťami
- doimplementácia webovej stránky pre admina
    - kompletné menežovanie zariadení, senzorov a oblastí

- vytvorenie kiosku pre správu systému priamo v konkrétnych miestnostiach
- testovanie kompletného systému a všetkých jeho častí

1.4 Celkový pohľad na systém

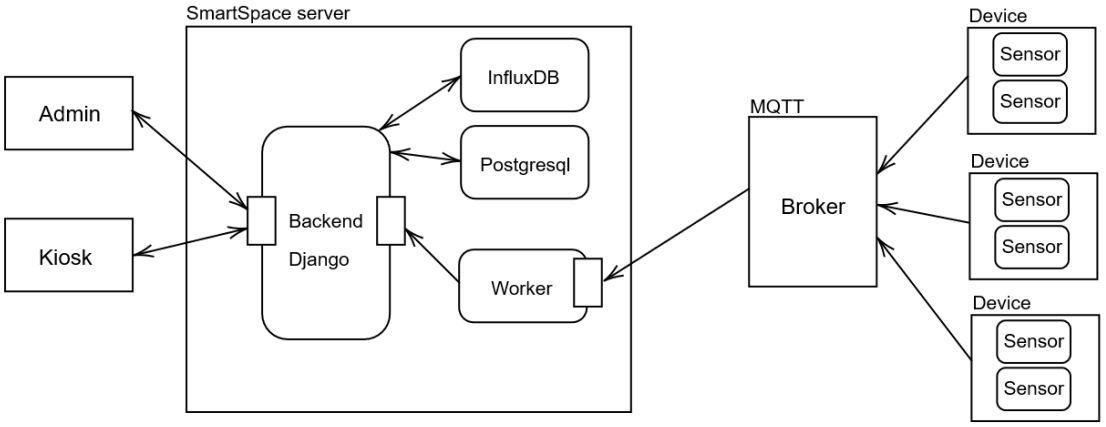
Systém sa skladá z troch častí.

Backend

Prvá časť je jadrom celého systému a jedná sa o backend. Backend má za úlohu spracovať a ukladať informácie a merania zo senzorov a taktiež poskytovať funkcionalitu rezervovania pre klientov. Interakcia s backendom je realizovaná prostredníctvom API volaní, ktoré sú vykonávané z ostatných častí systému ako administrátorský portál, kiosk alebo worker. Backend sa skladá z troch častí.

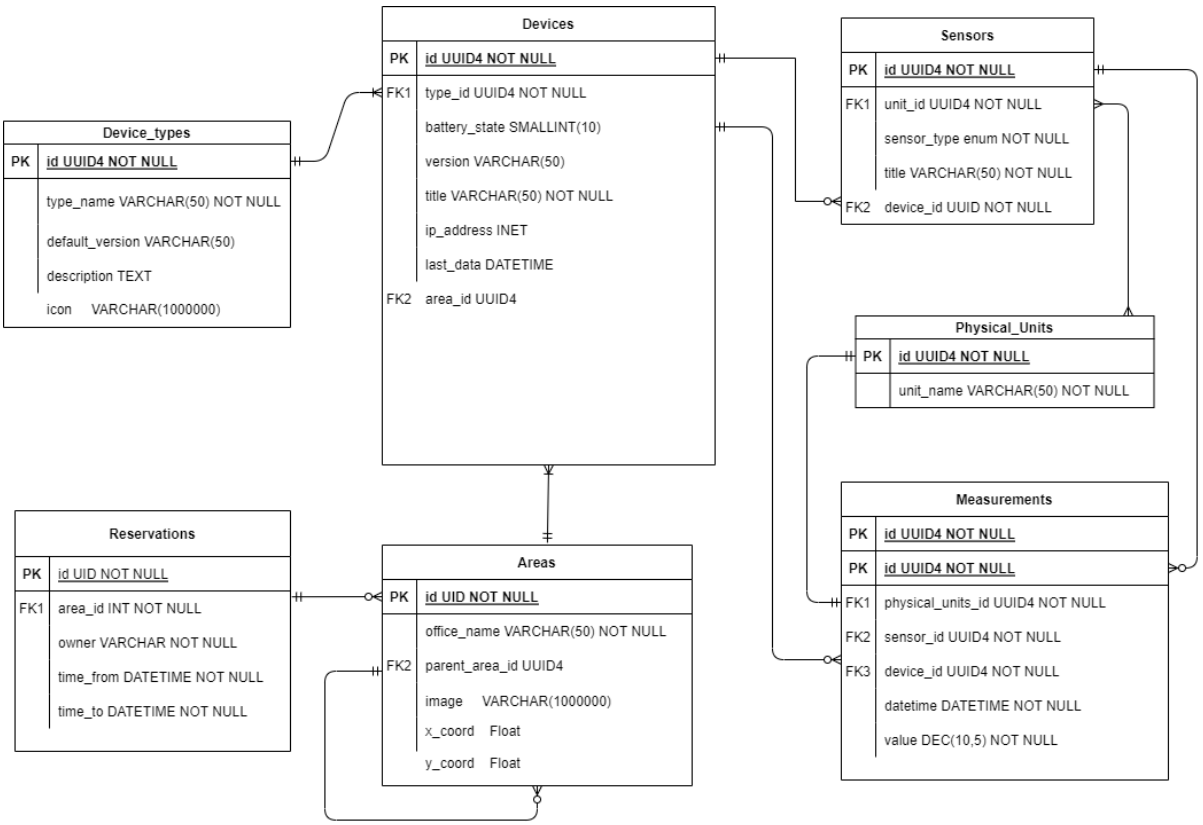
1. Webová aplikácia - Django
2. Relačná databáza - Postgresql
3. Time-series databáza - InfluxDB

Všetky dáta okrem meraní sú v backende reprezentované ako modely, ktoré sú uložené v relačnej databáze. Influx je používaný na dáta z meraní. Na spracovanie žiadostí systém využíva REST django framework a všetky dáta prenášané v žiadostiach a odpovediach sú vo formáte JSON. Architektúra celej backend časti systému je na nasledujúcom obrázku:



Obrázok 1: Architektúra aplikácie

Na obrázku je dátový model systému. Je v ňom zobrazená relácia medzi meraniami a modelmi sensor a device. Aj napriek tomu, že merania sú uložené v nerelačnej databáze, pre správne fungovanie systému ich jednoznačne priradiť k pôvodcom daného merania.



Obrázok 2: Dátový model

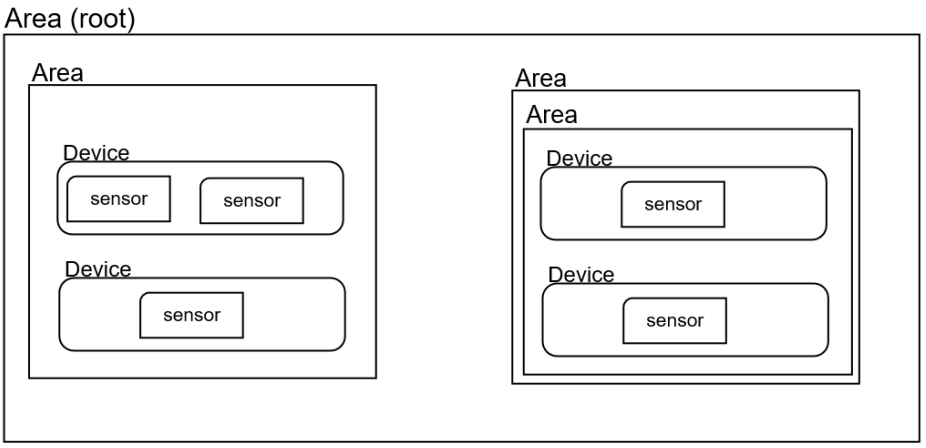
Aktuálna funkcionalita, ktorú backend umožňuje je vytváranie, upravovanie a mazanie oblastí, zariadení, senzorov a rezervácií.

Oblasť (v modeli je to Area) je miestnosť, budova alebo aj časť miestnosti, ktorá je nejako logicky samostatná. Miestnosť v sebe môže mať zariadenia alebo aj ďalšie oblasti. Oblasť, ktorá nie je vnorená v žiadnej inej sa nazýva “root area”. Ak vyhodnocujeme priemernú teplotu alebo inú štatistiku, vyhodnocujeme ju v rámci danej oblasti. Takisto aj rezervácie sú vytvárané pre konkrétnu oblasť. Oblasť môže reprezentovať aj konkrétne miesto pri stole, čo umožňuje v prípade potreby rezervovať aj konkrétne miesto pri stole.

Zariadenia (modely Device) sú mikropočítače, na ktorých sú umiestnené senzory. Zariadenie môže mať na sebe viacero alebo aj žiadny senzor. V implementácií sme pracovali s Raspberry Pi 4 a ESP32. Pri vytváraní zariadenia je nevyhnutné určiť v akej oblasti sa nachádza aby systém vedel pri vyhodnocovaní stavu miestnosti brať do úvahy aj merania z daného zariadenia.

Senzory (model Sensor) sú pripojené ku konkrétnemu zariadeniu. Pri vytváraní senzoru v systéme je potrebné určiť, na ktorom zariadení sa nachádza. Všetky merania sa ukladajú do InfluxDB spolu s ID zariadenia a ID senzoru, ktoré toto meranie vytvoril. Každý senzor meră v určitej fyzikálno-matematickej jednotke, ktorú má k sebe priradenú v databáze.

Príklad toho, ako môže vyzerať konfigurácia oblastí, zariadení a senzorov je na nasledujúcom obrázku:



Obrázok 3: Oblasti, zariadenia a senzory

Celá dokumentácia (vygenerovaná pomocou nástroja Sphinx) k zdrojovému kódu backendu je priložená v technickej dokumentácií - backend. Je možné ju otvoriť pomocou prehliadača. Všetky volania API sú v technickej dokumentácií - swagger.

Zariadenia a senzory

Naše riešenie obsahuje viacero mikrokontrolérov typu ESP32, do ktorých sa nahrá kód určený na získavanie meraní jednotlivých senzorov. Kód obsahuje aj údaje o WIFI mene a hesle, ktorými sa pripojí na MQTT broker, kde bude posielať získané údaje.

Máme 3 typy senzorov:

- Air Quality - Tento senzor predstavuje model MQ-135, nakoľko nepoznáme jednotlivé hodnoty odporu a prúdu a preto nie je možné presne namapovať výstup senzora na hodnoty ppm. Jediné čo sa nám podarilo zo senzora získať sú hodnoty napätia a tie sme experimentálne namapovali stupnicu od 0 po 10.
- Motion - Tento senzor predstavuje model HC-SR501, ktorý podľa danej konfigurácie vracia bínárne hodnoty toho, či je zachytený pohyb alebo nie. Podľa týchto hodnôt meraných napríklad každú sekundu je možné určiť, či je dané miesto/miestnosť obsadená.
- Temperature & Humidity - Tento senzor predstavuje model DHT22, ktorý dokáže merať teplotu aj vlhkosť prostredia. Tieto hodnoty vracia podľa prednastavenej konfigurácie.

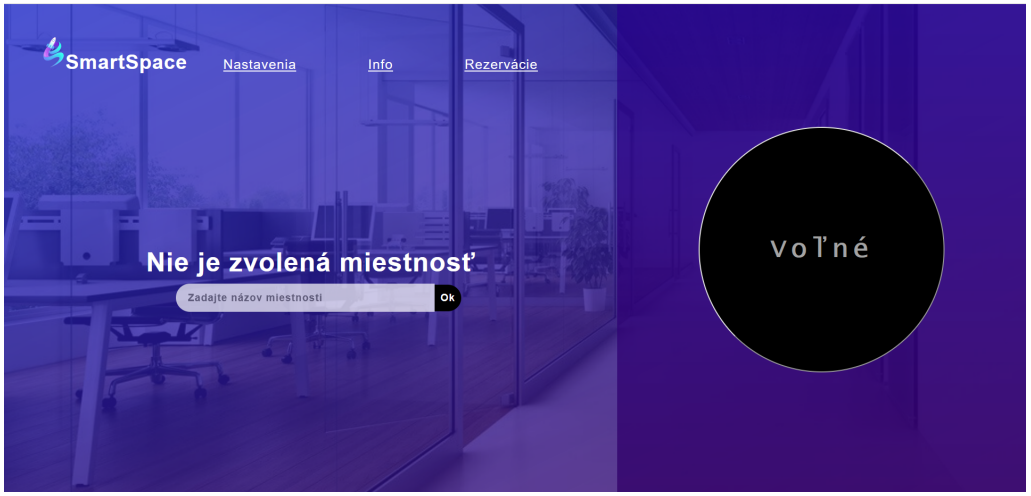
ESP32 zariadenia odosielajú hodnoty na MQTT broker, prostredníctvom ktorého sa ukladajú do InfluxDB v požadovanom tvare. Samotnú architektúru riešenia je možné vidieť na obrázku č. 1.

Identifikovanými problémami pri senzoroch a zariadeniach bola nedostatočná dokumentácia vnútornej schémy pri MQ-135 senzore, kde nameranú hodnotu bolo potrebné premapovať na základe 12-bitového ADC (Analog Digital Converter) prevodníka aby sme dostali aspoň výstupné napätie. Premapovanie bolo potrebné pretože referenčné napätie v tomto AQ senzore je 5V a referenčné napätie ESP32 je 3.3V aj keď je napájané 5V napätím.

Ďalším problémom pri ESP32 bolo neschopnosť zariadenia vykonávať analógové meranie a zároveň byť pripojený na WIFI, nakoľko obe tieto funkcionality využívajú už spomínaný ADC prevodník a preto bolo nutné podať dodatočnú knižnicu, ktorá dokáže úplne vypnúť a zapnúť WIFI modul.

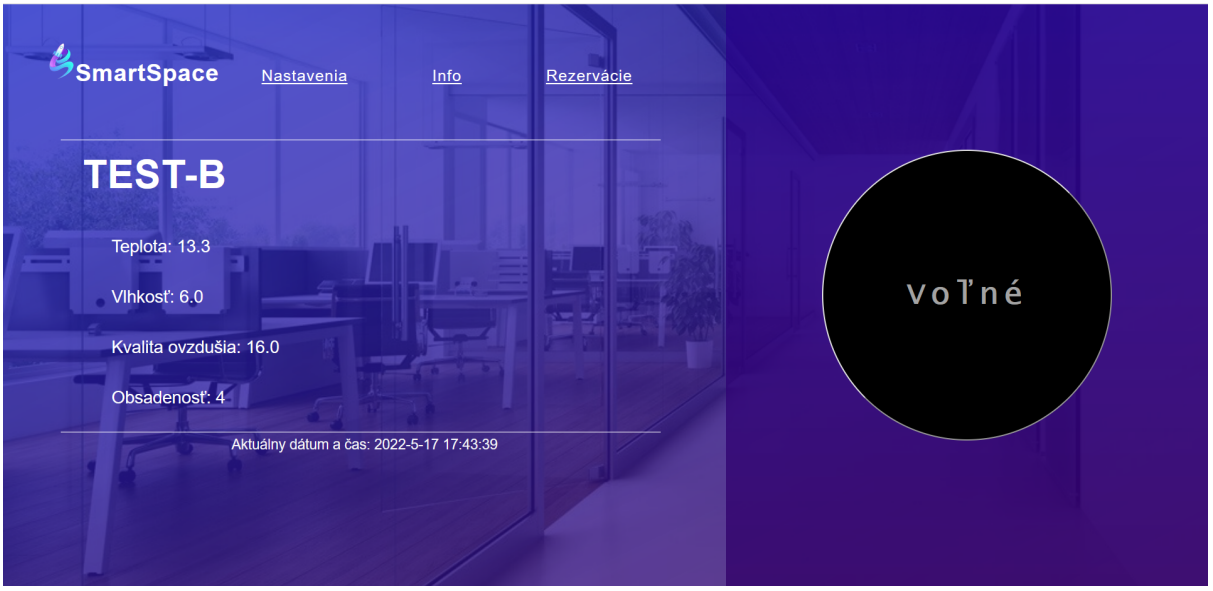
Kiosk

Treťou časťou sú klienti. Každý klient vie sledovať stav miestností a prípadne vie urobiť rezerváciu miesta za pomoci zariadenia Kiosk, ktoré je nasadené pred vstupnými dverami do miestnosti. Na tomto zariadení je potrebné na začiatku zvoliť názov miestnosti, pre ktorú chceme zobrazovať namerané hodnoty.



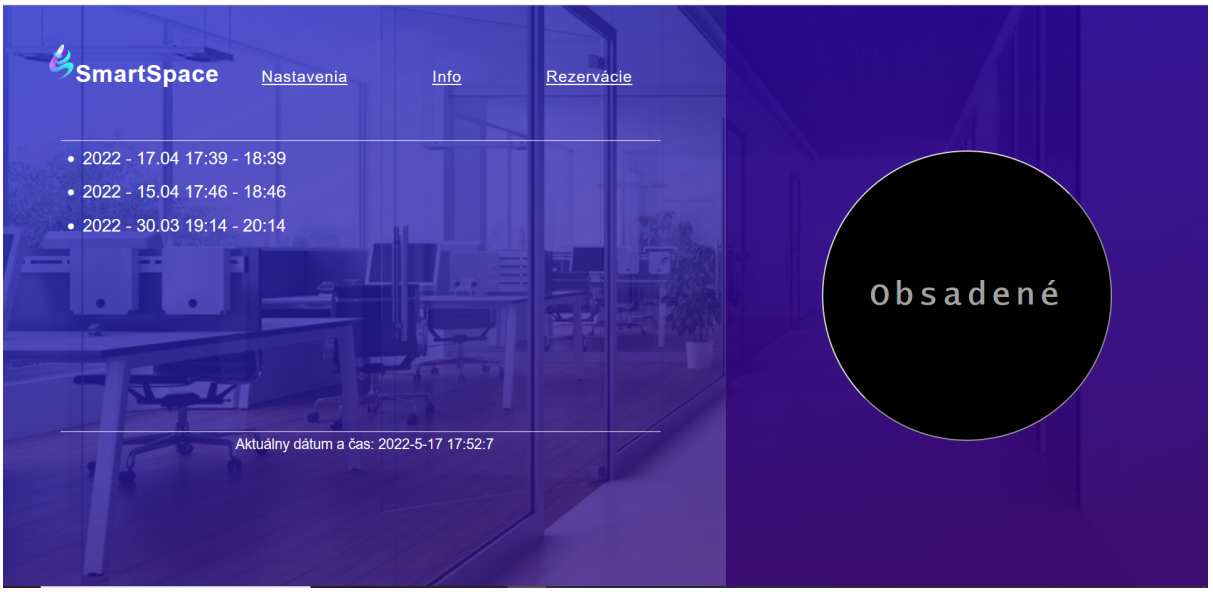
Obrázok 4: Hlavné menu na zariadení Kiosk

V prípade kliknutia na podstránku s názvom “Info” v hlavnej navigácii, sa zobrazia namerané hodnoty pre zvolenú miestnosť. Všetky dáta sa aktualizujú každých 15 sekúnd a vznikajú ako priemerná hodnota z posledných meraní každého senzora nasadeného v miestnosti.



Obrázok 5: Zobrazenie nameraných hodnôt v zariadení Kiosk

Poslednou časťou tejto aplikácie je možnosť pre zobrazenie všetkých rezervácii pre zvolenú miestnosť miestnosť. Informácie o všetkých rezerváciách sú taktiež aktualizované každých 15 sekúnd.



Obrázok 6: Zobrazenie rezervácii v zariadení Kiosk

Ako je možné vidieť aj na obrázku vyššie, tak miestnosť je v danom momente označená ako obsadená. V prípade obsadenia miestnosti priamo cez zariadenie Kiosk, je vytvorená hodinová rezervácia.

Dashboard

Poslednou časťou nášho riešenia je dashboard portál, ktorý slúži ako webové rozhranie pre zobrazenie zariadení v miestnostiach. Obsahuje aj funkcionalitu pre mapovanie konkrétnych fyzických senzorov na vytvorené zariadenie. Kľúčovým prvkom aby bolo možné zobrazovať zariadenia v miestnosti je nutnosť vedieť miestnosti vytvárať. To môžeme docieliť v sekcii Areas, ako je zobrazené na obrázku nižšie. Miestnosti sú organizované do stromovej štruktúry s rodičmi a potomkami.

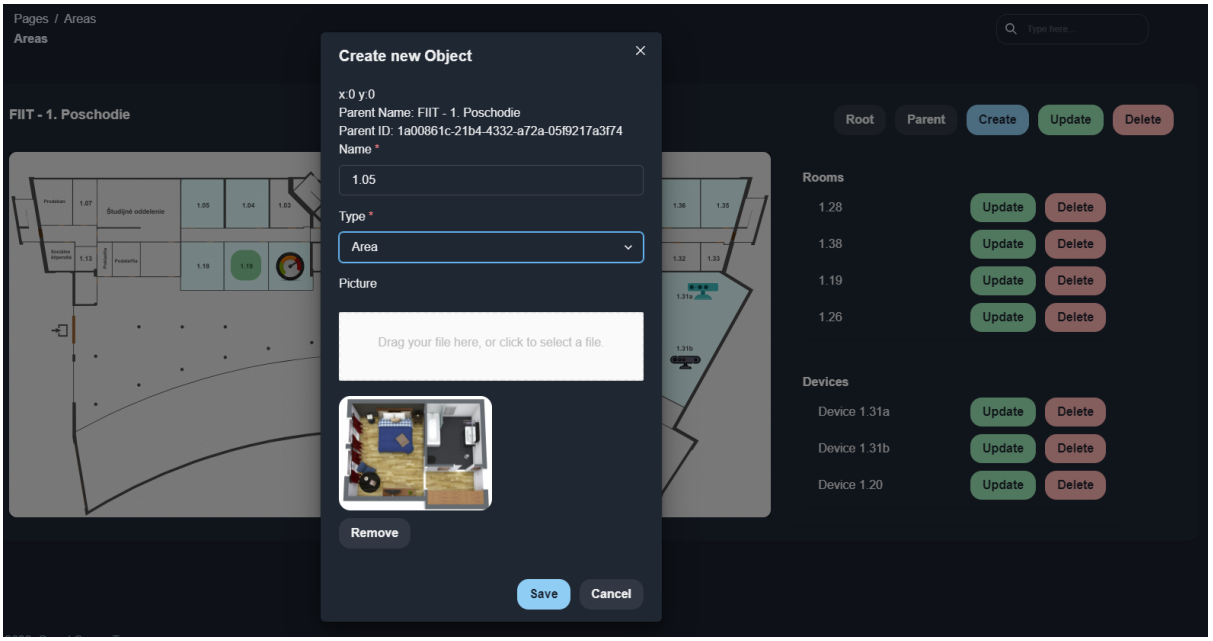
Podradenú miestnosť je možné zobraziť kliknutím na patričnú miestnosť v zozname alebo klikom na jej reprezentáciu v obrázku (zelený štvorec).

Obrázok 7: Zobrazenie oblasti

Horný panel obsahuje viacero tlačidiel: Root, Parent, Create, Update, Delete. Tlačidlom Parent je možné prejsť do priameho rodiča, ktorej je miestnosť v rámci systému umiestnená. Root tlačidlom sa presunieme v stromovej štruktúre úplne do najvrchnejšieho uzla. V tomto uzle sa nachádza zoznam zariadení a miestnosti, ktoré nie sú asociované k žiadnemu rodičovi. Napríklad takýmto spôsobom je možné v systéme viesť evidenciu miestnosti viacerých budov alebo nezávislých lokácií a tak užívateľ má voľnú ruku aký manažment oblastí si zvolí.

Obrázok 8: Zobrazenie oblastí

Miestnosti je možné vytvárať tlačidlom Create. Okno, ktoré sa nám otvorí slúži aj na manažment zariadení, preto je nutné zvoliť možnosť typ vytváraného objektu ako Area. Je nutné dodať, že tlačidlom Create sa vytvára miestnosť v miestnosti, v ktorej sa akurát nachádzame v rámci stromovej štruktúry. Miestnosť je taktiež možné vytvoriť klikom na patričnú pozíciu v obrázku ak aktuálna miestnosť má nahratý obrázok.

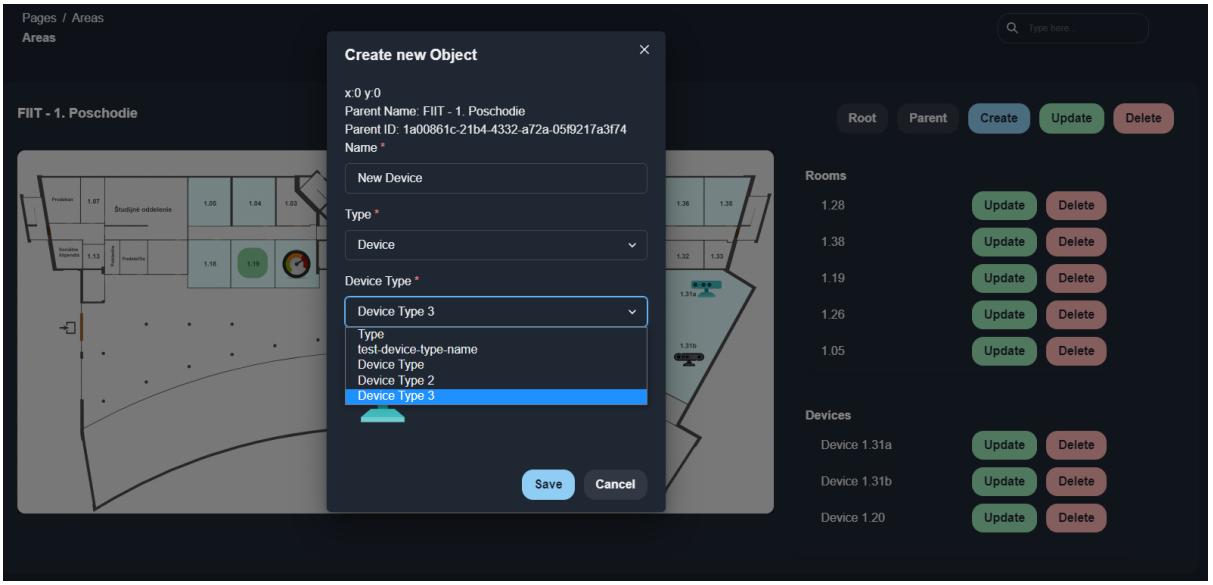


Obrázok 9: Vytvorenie oblasti

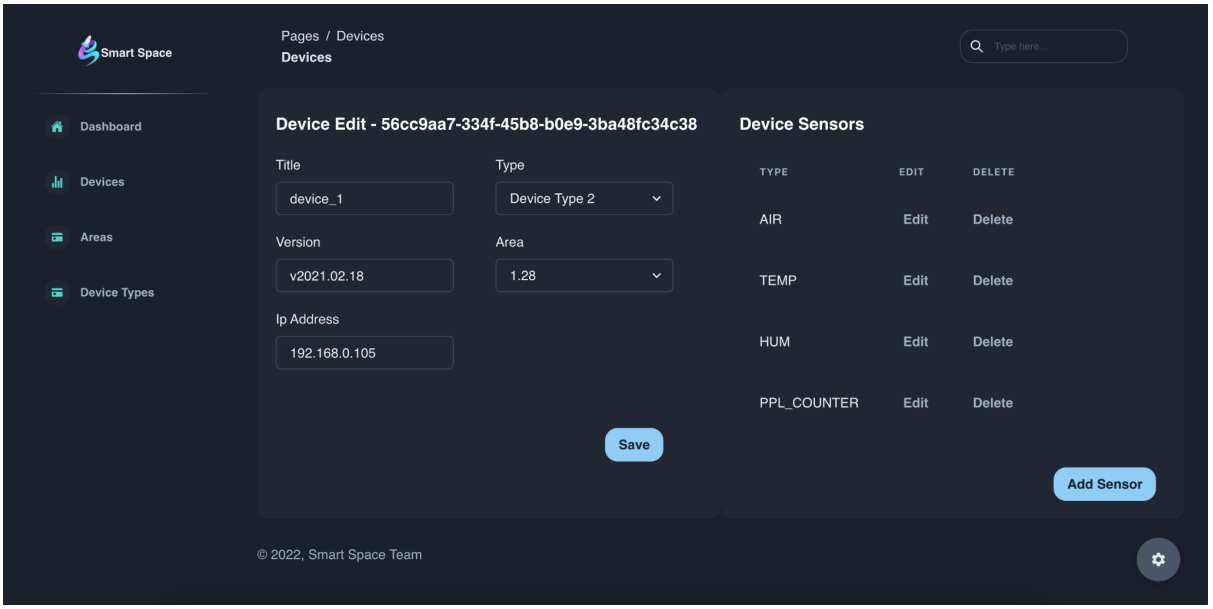
V rámci zobrazení miestností, je tiež možné vytvárať zariadenia podobne ako Areu avšak aby bolo možné vytvoriť zariadení je potrebné mať v systéme vytvorený patričný typ zariadenia. Manažment zariadení je možné nájsť vo webovom rozhraní v sekcii Device Types. Logika zobrazovania, vytvárania, mazania a úpravy je obdobná ako pri miestnostiach.

Obrázok 10: Zoznam typov zariadení

Pri vytváraní alebo úprave zariadenia sa obrázok nenahráva, ale automaticky je reprezentovaný obrázkom daného zvoleného typu. Zmenu pozície miestnosti alebo zariadeniu je možné realizovať funkcionalitu Drag-and-drop na jeho reprezentáciu v rámci obrázka.

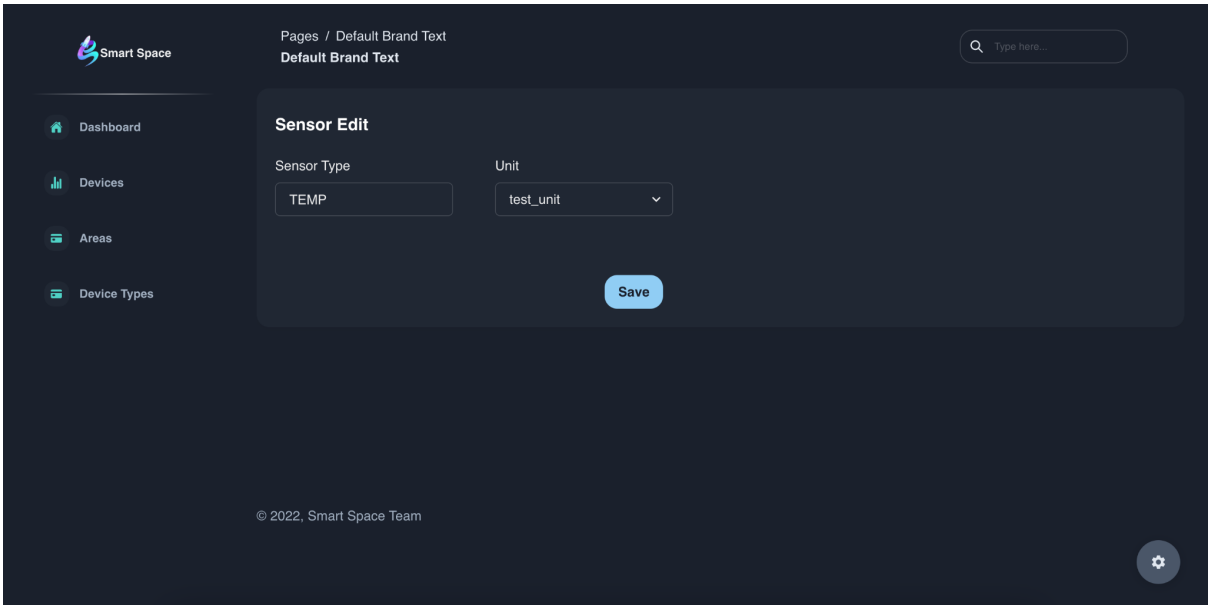


Obrázok 11: Priradenie typu zariadenia



Obrázok 12: Úprava zariadenia

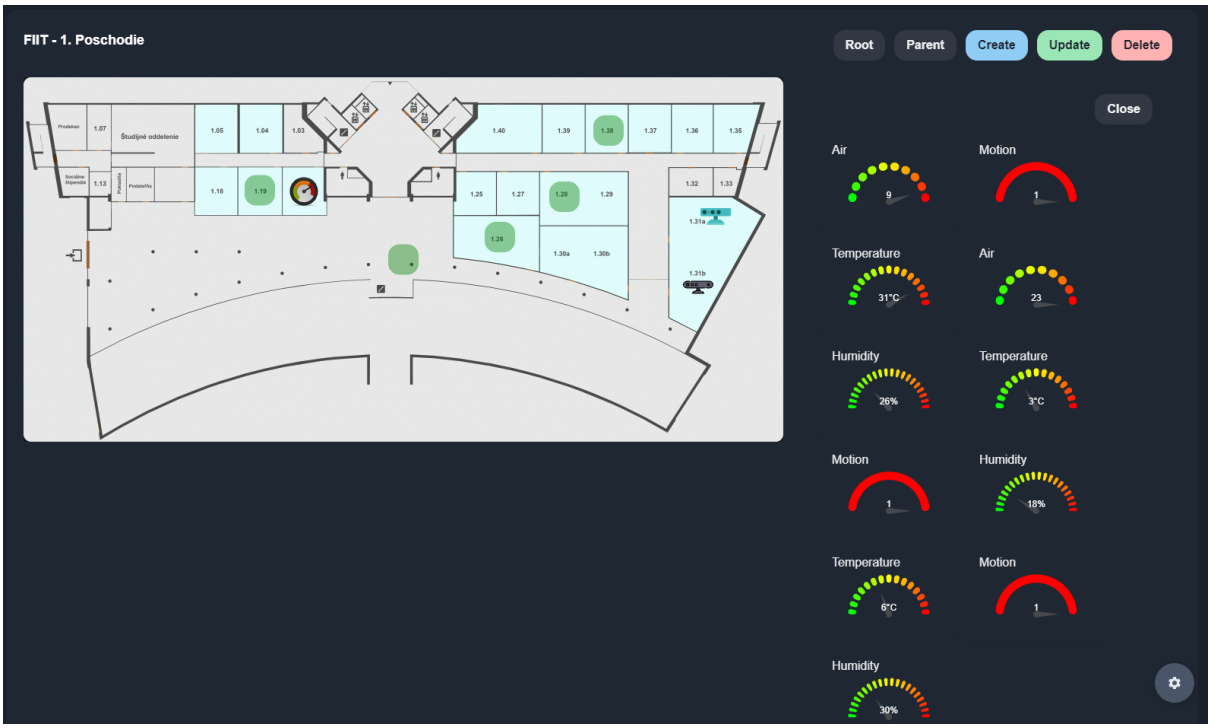
Ak sa nachádzame v edite zariadenia, tak môžeme v pravom panely vidieť zoznam všetkých senzorov priradených danému zariadeniu. Tieto senzory vieme upraviť alebo zmazať.



Obrázok 13: Úprava senzoru

Ak sme na stránke edit zariadenia stlačili tlačidlo create alebo edit pri konkrétnom senzore tak sme sa dostali na formulár vytvorenia/úpravy konkrétneho senzoru v ktorom vieme určiť typ senzoru a jeho jednotku a uložiť ho.

Po priradení senzorov vytvorenému zariadeniu, je možné v manažmente miestností rozkliknúť jeho detail. Detail môžeme zobraziť dvomi spôsobmi a to kliknutím naň v zozname alebo jeho grafickú reprezentáciu v rámci obrázka miestnosti. V podrobnostiach sú zobrazené údaje z jeho priradených senzorov, ktoré sú aktualizované periodicky.



Obrázok 14: Zobrazenie senzorov zariadenia

Vymazanie miestnosti tlačidlom Delete sa rekurzívne odstránia zo systému aj potomkovia.

2. Moduly systému

2.1 Analýza

Systém, ktorý máme, sa dá rozdeliť na viacero častí. Frontend tvorí osobitnú časť, a backend je zložený z viacerých prvkov, ktoré spolu interagujú. Medzi tieto prvky patria napríklad senzory, zberače údajov, generátor testovacích dát, samotný spracovávač dát a iné. Tieto časti sú popísané v nasledujúcich podkapitolách.

2.1.1 Backend

Hlavnou časťou backendu je webová aplikácia vytvorená vo frameworku Django. V zásade jej úlohou je poskytovanie API pre klientske zariadenia a pre Workera, ktorý spravuje zariadenia. API je dostupná na route /api. Volania API sú dobre vizualizované pomocou Swagger-a. **Všetky volania api:**

Práca so zariadeniami:

Metóda	**URI**	**Popis**
POST	/devices	Vytvorenie zariadenia
GET	/devices	Zoznam všetkých zariadení
GET	/devices/{device_id}	Informácie o zariadení
PUT	/devices/{device_id}	Aktualizácia informácii o zariadení
DELETE	/devices/{device_id}	Vymazanie zariadenia
POST	/devices_types/{device_id}/ devices	Vytvorenie viacerých zariadení so špecifickým typom
DELETE	/devices_types/{device_id} devices	Vymazanie všetkých zariadení so špecifickým typom

Tabuľka 1: Volania API pre zariadenia

Práca s typmi zariadeni:

Metóda	**URI**	**Popis**
POST	/devices_types/	-
GET	/devices_types/	-

GET	/devices_types/{device_type_id}	Získa informácie o type zariadení
PUT	/devices_types/{device_type_id}	Aktualizácia informácii o type
DELETE	/devices_types/{device_type_id}	Odstránenie typu zariadenia

Tabuľka 2: Volania API pre typy zariadení

Práca s meraniami:

Metóda	URI	Popis
POST	/measurements	Vytvorenie zoznamu meraní
GET	/devices/{device_id}/last_measurement	Posledné meranie zariadenia
GET	/devices/{device_id}/measurements	Zoznam meraní zariadenia na základe časového úseku

Tabuľka 3: Volania API pre merania

Práca s rezerváciami:

Metóda	URI	Popis
POST	/reservations	Vytvorenie rezervácie
PUT	/reservations/{reservation_id}	Aktualizácia rezervácie
DELETE	/reservations/{reservation_id}	Vymazanie rezervácie

Tabuľka 4: Volania API pre rezervácie

Práca so senzormi:

Metóda	URI	Popis
POST	/senors	Vytvorenie zoznamu senzorov
PUT	/senors/{sensor_id}	Aktualizácia informácii o senzore
GET	/senors/{sensor_id}	Získanie informácii o senzore
DELETE	/senors/{sensor_id}	Vymazanie senzora

GET	/sensor/filter	Získanie informácii o viacerých senzoroch
DELETE	/sensor/filter	Vymazanie viacerých senzorov

Tabuľka 5: Volania API pre senzory

Práca s autentifikáciou:

Metóda	URI	Popis
POST	/authenticate	Autentifikácia používateľa
POST	/authenticate/refresh	Obnovenie prístupového tokenu

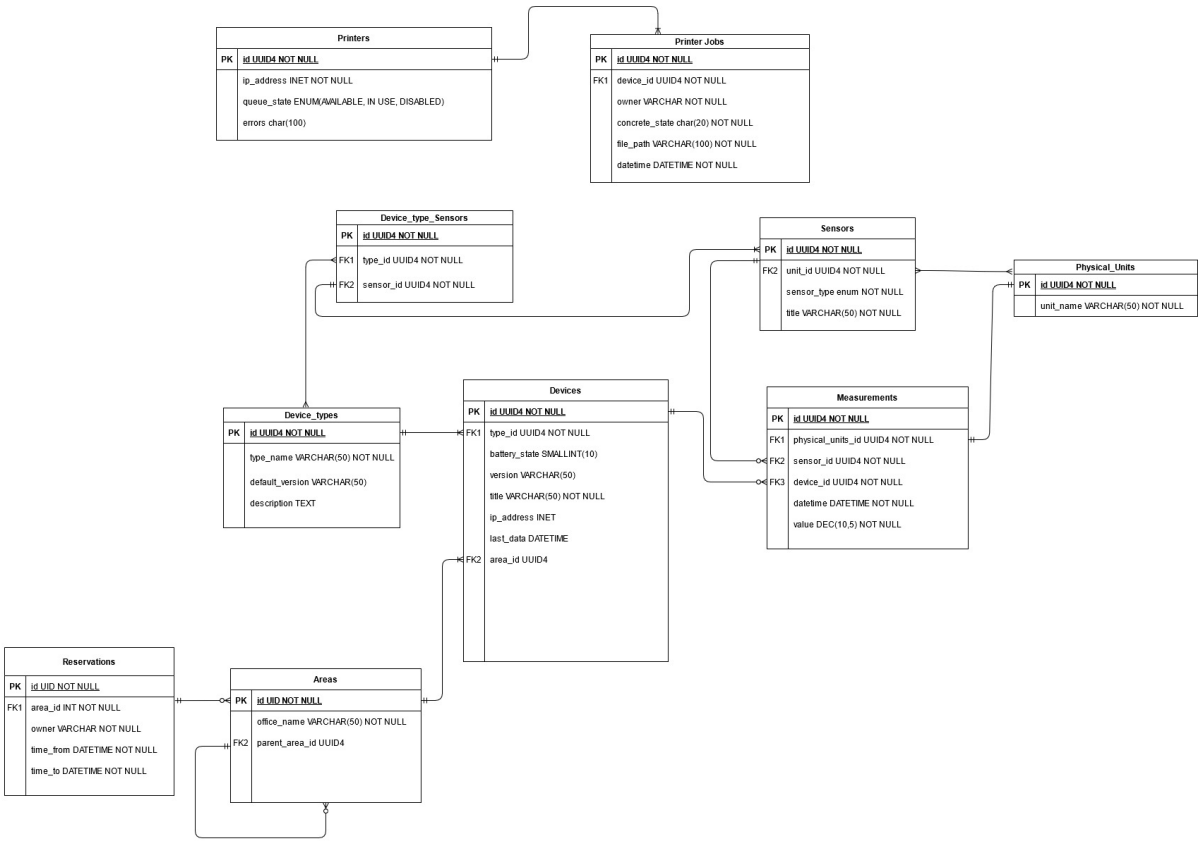
Tabuľka 6: Volania API pre autentifikáciu

Práca s oblasťami:

Metóda	URI	Popis
POST	/areas	Vytvorenie novej oblasti
GET	/areas	Získanie zoznamu oblastí

Tabuľka 7: Volania API pre oblasti

Databázový systém použitý na uskladnenie dát je PostgreSQL. Jedná sa o relačnú databázu. Backend pracuje s databázou pomocou modelov z modulu django.db. Dátový model bol nasledovný:



Obrázok 15: Aktuálny dátový model

2.1.2 Worker

Na spracovanie dát zo senzorov je vytvorený Worker, ktorý tieto dáta zbiera, upravuje a posiela na backend. Funguje na jednoduchom princípe - odoberá kanály, kam senzory posielajú merania. Keďže merania môžu byť v rôznom tvare podľa typu senzoru, používa sa viac kanálov - osobitne pre

- motion
- temperature and humidity
- air quality

Tieto merania sa následne odosielajú na backend cez triedu measurements. Worker dáta nevyhodnocuje, len ich posiela ďalej v preferovanom tvare.

2.1.3 Data-generator

V projekte sa nachádza aj dátový generátor, ktorý je určený na rýchle generovanie náhodných dát, imitující ľubovoľný počet akýchkoľvek senzorov, ktoré sú zadefinované v konfiguračnom súbore.

Generátor sa skladá z nasledujúcich tried:
Generators

V súbore generators sú vytvorené 2 generátory, ktoré na základe konfiguračného súboru vracajú náhodné hodnoty pre RangeSenzor a Occupancy senzor.

Class RangeGenerator

Volaná z „**execute.py**“, generuje hodnoty pre co2,humidity a temperature.

```
    driver = RangeGenerator(unit)
    client.publish(f"tp/{unit['name']}", f"data={driver.execute()};device={device['id']}")
elif unit['name'] == 'occupancy':
```

Obrázok 16: Miesto volania RangeGenerátora

```
class RangeGenerator:
    def __init__(self, config):
        self._config = config

    def execute(self) -> int:
        steps = self._config['params']['measurements']
        result = 0
        for i in range(0, steps):
            result += random.randrange(
                self._config['params']['min'],
                self._config['params']['max'],
                self._config['params']['delta']
            )
        return result // steps
```

Obrázok 17: Podstatná časť zdrojového kódu RangeGenerátora

Class OccupancyGenerator

Volaná z „**execute.py**“, určená na generovanie hodnôť obsadenosti miestnosti.

```
elif unit['name'] == 'occupancy':
    driver = OccupancyGenerator(unit)
    client.publish(f"tp/{unit['name']}", f"data={driver.execute()};device={device['id']}")
```

Obrázok 18: Miesto volania OccupancyGenerátora

```
class OccupancyGenerator:
    def __init__(self, config):
        self._config = config

    def execute(self) -> int:
        steps = self._config['params']['measurements']
        result = 0
        for i in range(0, steps):
            if random.random() < self._config['params']['probability']:
                result += 1
        return result
```

Obrázok 19: Podstatná časť zdrojového kódu OccupancyGenerátora

init.py

Načítanie konfiguračného súboru

```python
class BaseCommand(ABC):
    def __init__(self, configuration: dict):
        self._config = configuration

    @abstractmethod
    def execute(self, **kwargs):
        pass
```

Obrázok 20: Zdrojový kód scriptu __init.py__

Config.py

Vypísanie konfiguračného súboru

```python
class ConfigCommand(BaseCommand):
    def execute(self, **kwargs):
        print(self._config)
```

Obrázok 21: Zdrojový kód scriptu Config.py

Devices.py

Súbor určený na inicializáciu zariadení, zadefinovaných v konfiguračnom súbore.

- Prideľovanie IP adries, verzie, stavu batérie...
- Výsledné zariadenia sú následne uložené do json súboru „devices_file“

Execute.py

Pre každý virtuálny device zavolá potrebné generátory a namerané hodnoty odosiela pomocou MQTT protokolu workerovi.

```python
# Run little Satan, run!!
for device in device_file['devices']:
    for unit in device_types[device['device_type_id']].get('units'):
        if unit['name'] in ['co2', 'humidity', 'temperature']:
            driver = RangeGenerator(unit)
            client.publish(f"tp/{unit['name']}", f"data={driver.execute()};device={device['id']}")
        elif unit['name'] == 'occupancy':
            driver = OccupancyGenerator(unit)
            client.publish(f"tp/{unit['name']}", f"data={driver.execute()};device={device['id']}")
```

Obrázok 22: Podstatná časť zdrojového kódu Execute.py

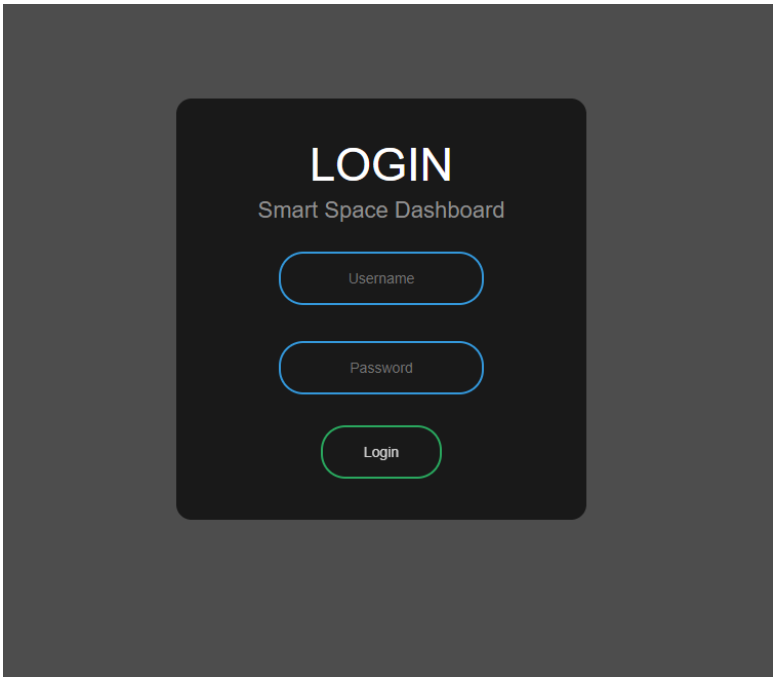
2.1.4 Frontend

V projekte sa už nachádzal existujúci frontend, ktorý bol vytvorený v reacte, ktorý používal class komponenty v ECMAScript 6(ES6) bez Typescriptu a grafickú šablónu. Stránky a ich komponenty, ktoré sa v projekte už nachádzali:

2.1.4.1 Login Page

Súbor: /src/layouts/Login.jsx
Komponent: Home

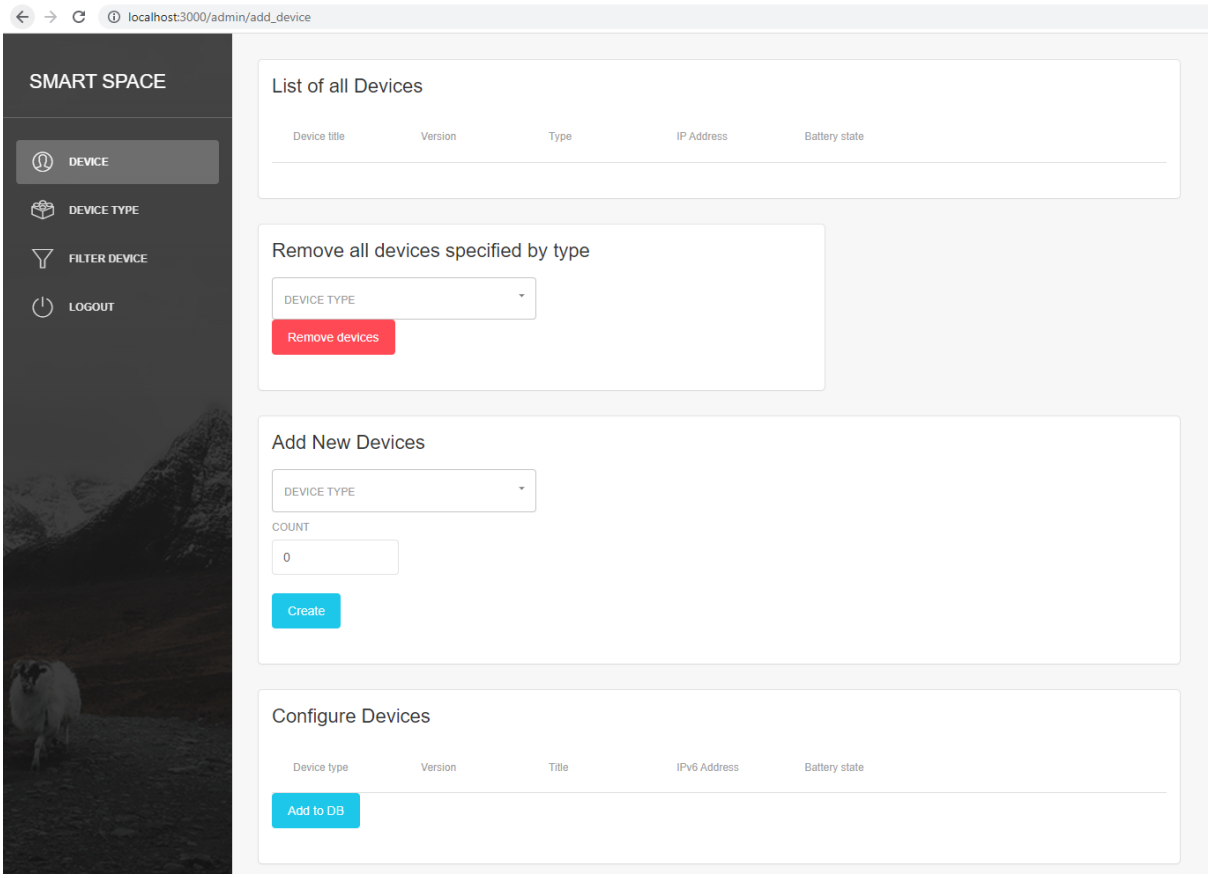
- Post request prihlásenie
  - endpoint: /api/authenticate
  - data:
    - username
    - password



Obrázok 23: Login obrazovka pôvodného projektu

2.1.4.2 Device screen

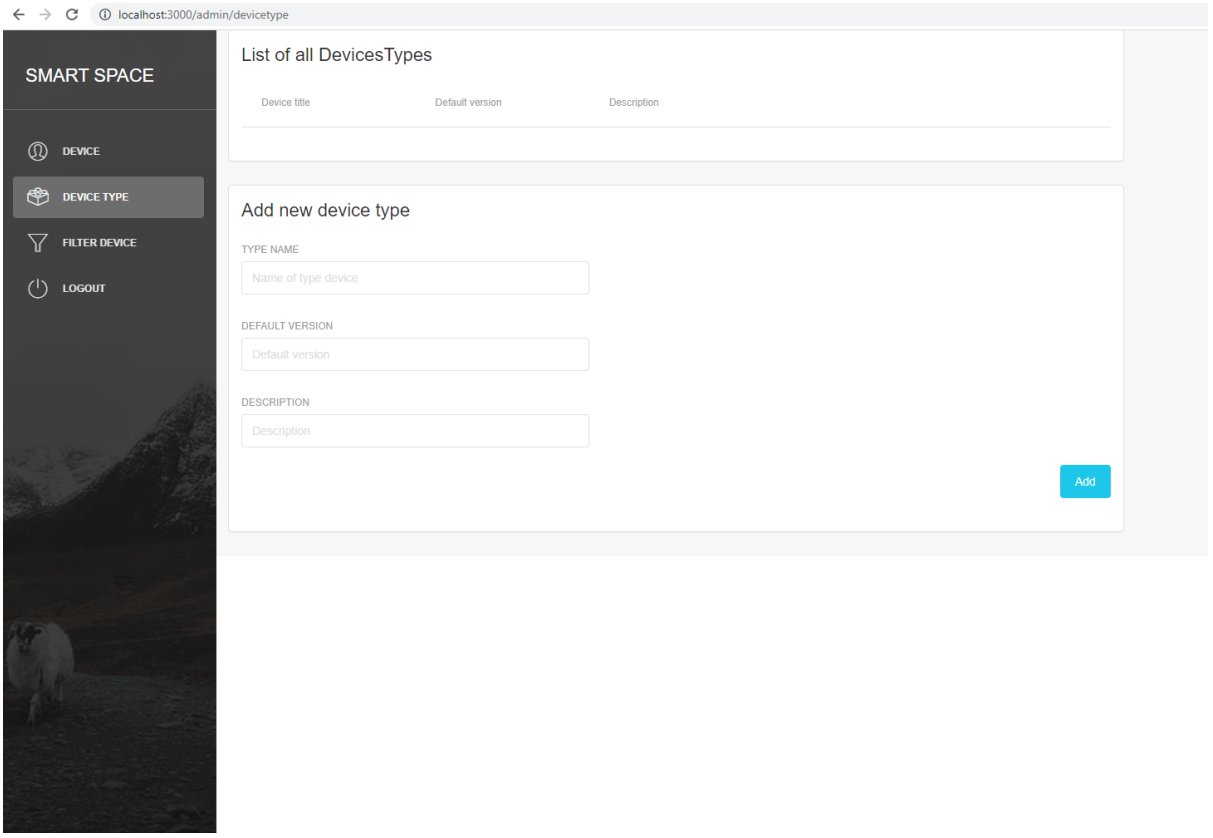
Súbor: /src/views/AddDevice.jsx
Komponent: AddDevice



Obrázok 24: Device obrazovka pôvodného projektu

- Získanie všetkých typov zariadení
  - get request
  - endpoint: /api/device_types
- Odstránenie všetkých zariadení podľa typu
  - delete request
  - endpoint: /device_types/${device_type.id}/devices
- Pridať nové zariadenie
  - nefunkčné
  - bez implementovaného requestu

2.1.4.3 Device types screen

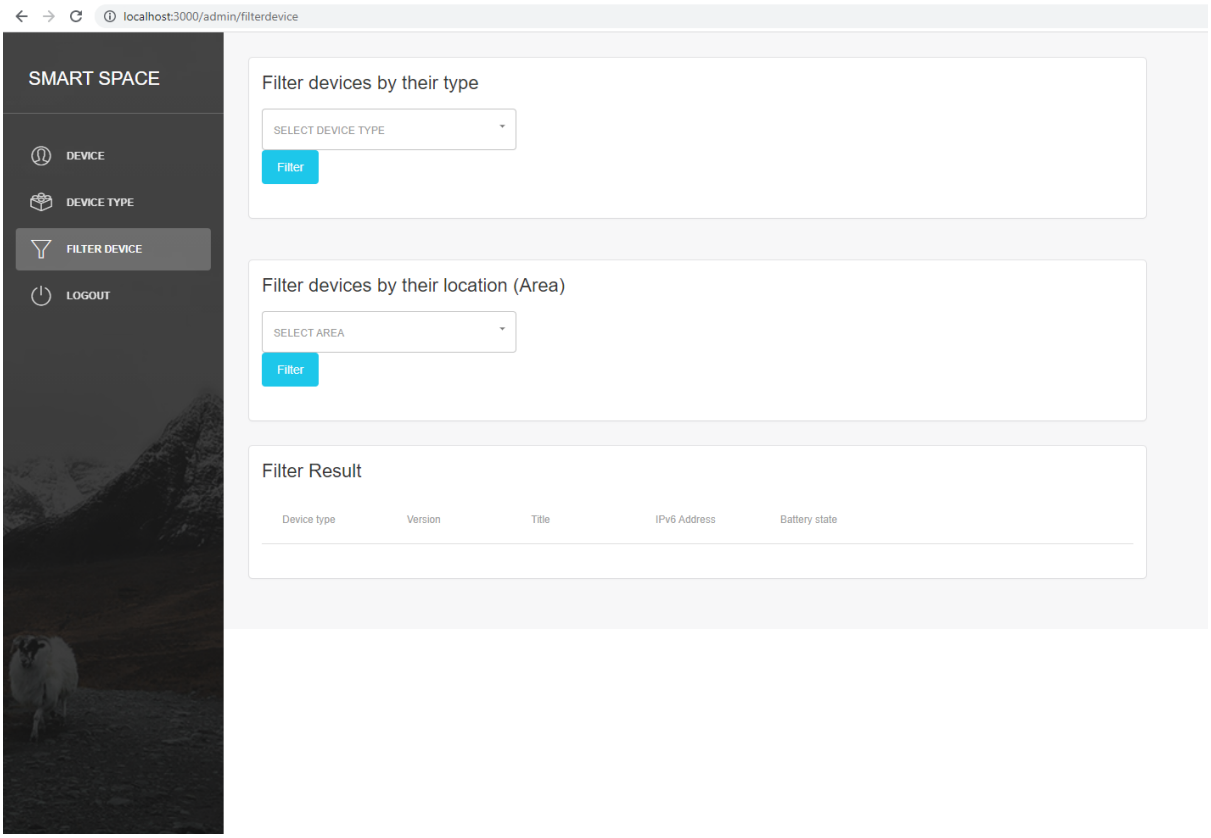


Obrázok 25: Device type obrazovka pôvodného projektu

- Zobrazuje zoznam všetkých typov zariadení
- Pridať nový typ zariadenia:
    - post request na /api/device_types s dátami

2.1.4.4 Filter Devices screen

- GET /api/device_types - filtrovanie podľa typu zariadenia
- GET /api/areas - filtrovanie podľa polohy zariadenia
- GET /api/devices - Zobrazenie sfiltrovaných zariadení



Obrázok 26: Device filter obrazovka pôvodného projektu

Z vyššie uvedených obrazoviek a komponentov v našom projekte nepoužijeme nič, pretože je jednoduchšie a prehľadnejšie pre nás začať robiť celého klienta odznova. Celý frontend sa bude vytvárať odznova v novšej verzii JS a novej komponentovej knižnici. Prepoužijeme však niektoré API volania, napríklad získanie všetkých zariadení/filtrovanie zariadení.

2.1.5 Hardware

Pre tento projekt máme dostupný hardware uvedený v nasledujúcej tabuľke:

Názov	Použitie	Počet
Raspberry Pi 4 - Model B 8GB RAM	zobrazovanie informácii a ovládanie	2
NodeMCU-32S ESP32	vývojová doska	3
ESP32-DevKitC-32U	vývojová doska	1
Waveshare 4,3” DSI LCD	displej pre Raspberry Pi	2
Multicomp Raspberry Pi-4 Premium Case	škatuľka pre Raspberry Pi	2
Raspberry Pi USB-C Power Supply	napájanie pre Raspberry Pi	2

DHT22 Temperature-Humidity	senzor na meranie teploty a vlhkosti	3
PIR Motion Sensor SOD00101S	senzor na detekcia pohybu	2
MQ-135 Air Quality	senzor na meranie kvality ovzdušia	2

Tabuľka 8: Dostupný hardware

2.1.5.1 Senzory

V projekte využívame 3 typy senzorov, ktoré v nasledujúcich podkapitolách analyzujeme. Pri analýze sme zistili, že senzory môžu využívať buď protokol MQTT alebo COAP. Hlavný rozdiel je ten, že CoAP funguje na princípe client - server, pričom MQTT formou publish-subscribe.

2.1.5.1.1 Motion sensor

Na meranie pohybu sa používa senzor typu HC-SR501. Tento typ senzora funguje na princípe, že každý objekt vyžaruje určité teplo, ktoré tento senzor pomocou infračerveného snímača zaznamenáva.

Má 2 možnosti jednoduchej modifikácie

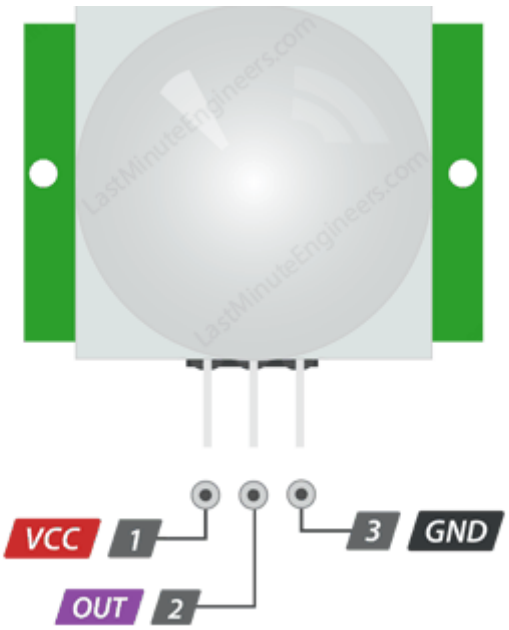
- **citlivosť** - dá sa nastaviť až do 7 metrov
- **čas** - ako dlho po zaznamenaní pohybu zostane v tomto móde

Takisto má svoje limitácie pri spustení, kym začne správne fungovať. Potrebuje 30 až 60 sekúnd na to, aby sa prispôsobil miestnosti a až následne vykonáva presné merania. Po každom meraní mu trvá 6 sekúnd, kým je schopný vykonať ďalšie meranie.

Technické detaily

- vstup: DC 4,5V-20V
- spotreba v nečinnosti <50 μA
- výstup: 3.3V / 0V
- dosah <120 °, do 7 m
- veľkosť 32 x 24 mm

Pinout



Obrázok 27: Pinout motion senzoru

2.1.5.1.2 Air quality sensor

Na meranie kvality ovzdušia používame senzor MQ-135, ktorý dokáže detekovať viacero druhov plynov ovplyvňujúcich kvalitu ovzdušia a to najmä Oxid uhličitý(CO_2), Oxidy dusíka(NO_x), amoniak (NH_3), benzén, acetón a podobne.

Obrázok 28: MQ-135 Air quality sensor

Senzor funguje na základe merania odporu tenkej vrstvy Oxidu ciničitého(SNo_2), ktorá mení svoj odpor na základe koncentrácie vyššie spomínaných plynov.

Obsahuje dva výstupy:

- Digitálny - Funguje na základe hodnoty nastavenej potenciometrom, v prípade dosiahnutia potrebnej koncentrácie zasvieti LED svetlo a digitálny výstup začne

namiesto 0V odosielať 5V. To znamená že digitálny výstup pozná len 2 hodnoty a to keď je koncentrácia pod hranicou a keď je nad hranicou potenciometra.

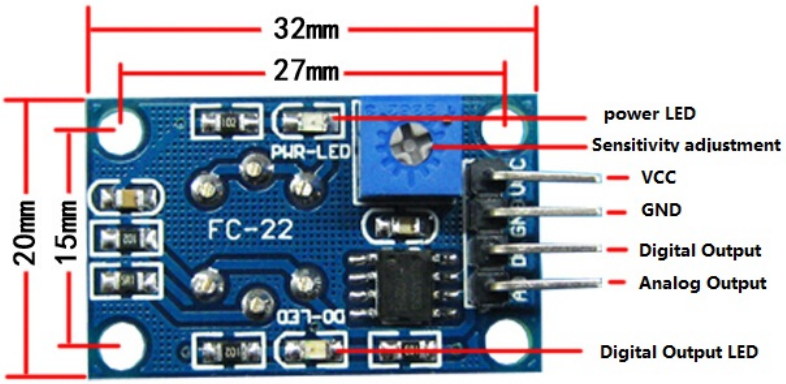
- Analógový - Odosiela výstupné napätie na základe koncentrácie plynov

Senzor je pred použitím vhodné nechať spustený 24 hodín pre správnu kalibráciu a následne pred každým ďalším spustením nechať nahriať po dobu 60-120 sekúnd aby sa senzor fungoval správne.

Technické detaily

- Vstupné napätie 5V
- Analógový výstup od 0V vyššie, záleží od koncentrácie
- Digitálny výstup buď 0V alebo 5V (na základe nastavenia potenciometra)
- Spotreba prúdu sa pohybuje okolo 120 mA
- Veľkosť 32mm(dĺžka) x 20mm(šírka) x 22mm(výška)

Pinout



Obrázok 29: Pinout Air Quality senzoru

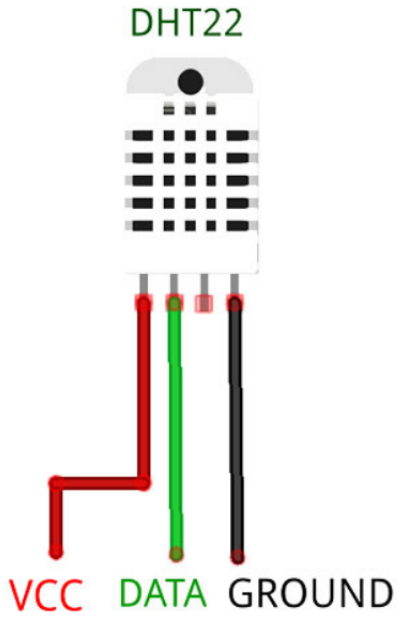
2.1.5.1.3 Humidity and Temperature sensor

Na meranie vlhkosti využívame základný senzor DHT22. Využíva kapacitný snímač vlhkosti a termistor na meranie okolitého vzduchu, ktorý vysiela signály na dátový pin. Ide o jednoduché zariadenie na využívanie, no vyžaduje si dobré načasovanie keďže jeho nevýhodou je, že je schopné zaznamenávať dáta len každé 2 sekundy.

Technické detaily

- 3 až 5V na vstup aj výstup
- 2.5mA prúd počas konverzie
- Rozsah pre meranie vlhkosti je 0-100% (2% - 5% nepresnosť)
- Rozsah pre meranie teploty -40 - 80°C (0,5°C nepresnosť)
- Veľkosť 27mm x 59mm x 13,5mm

Pinout

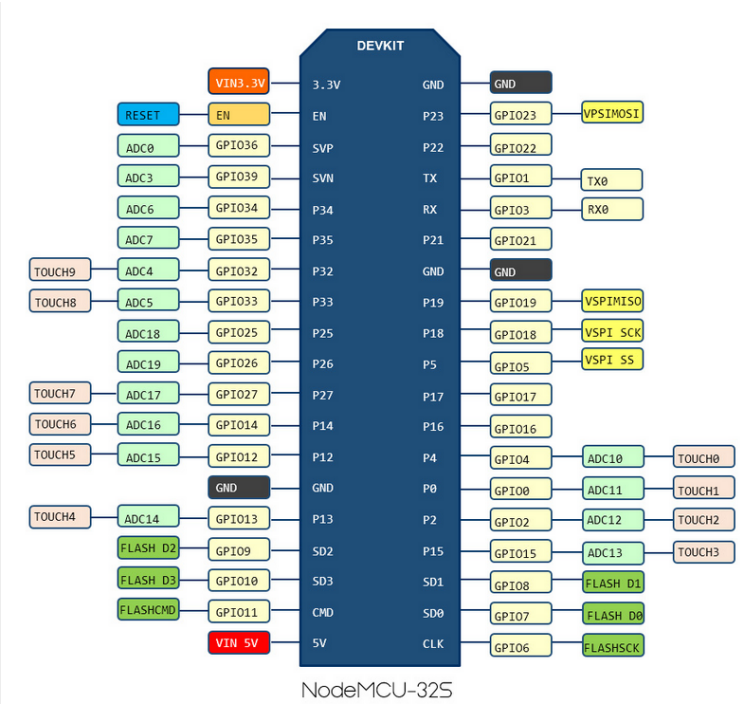


Obrázok 30: Pinout Temperature senzoru

2.1.5.2 Vývojové dosky

V našom projekte využívame vývojové dosky typu NodeMCU-32S ESP32 s možnosťou pripojenia na Wifi a BT.

Pinout



Obrázok 31: Pinout vývojovej dosky

Ďalším typom vývojových dosiek ktoré využívame sú ESP32-WROOM-32U (Espressif), ktoré majú rovnaký pinout ako predchádzajúca doska.

2.2 Návrh

V nasledujúcich kapitolách je rozpísané, ako máme navrhnuté jednotlivé moduly systému.

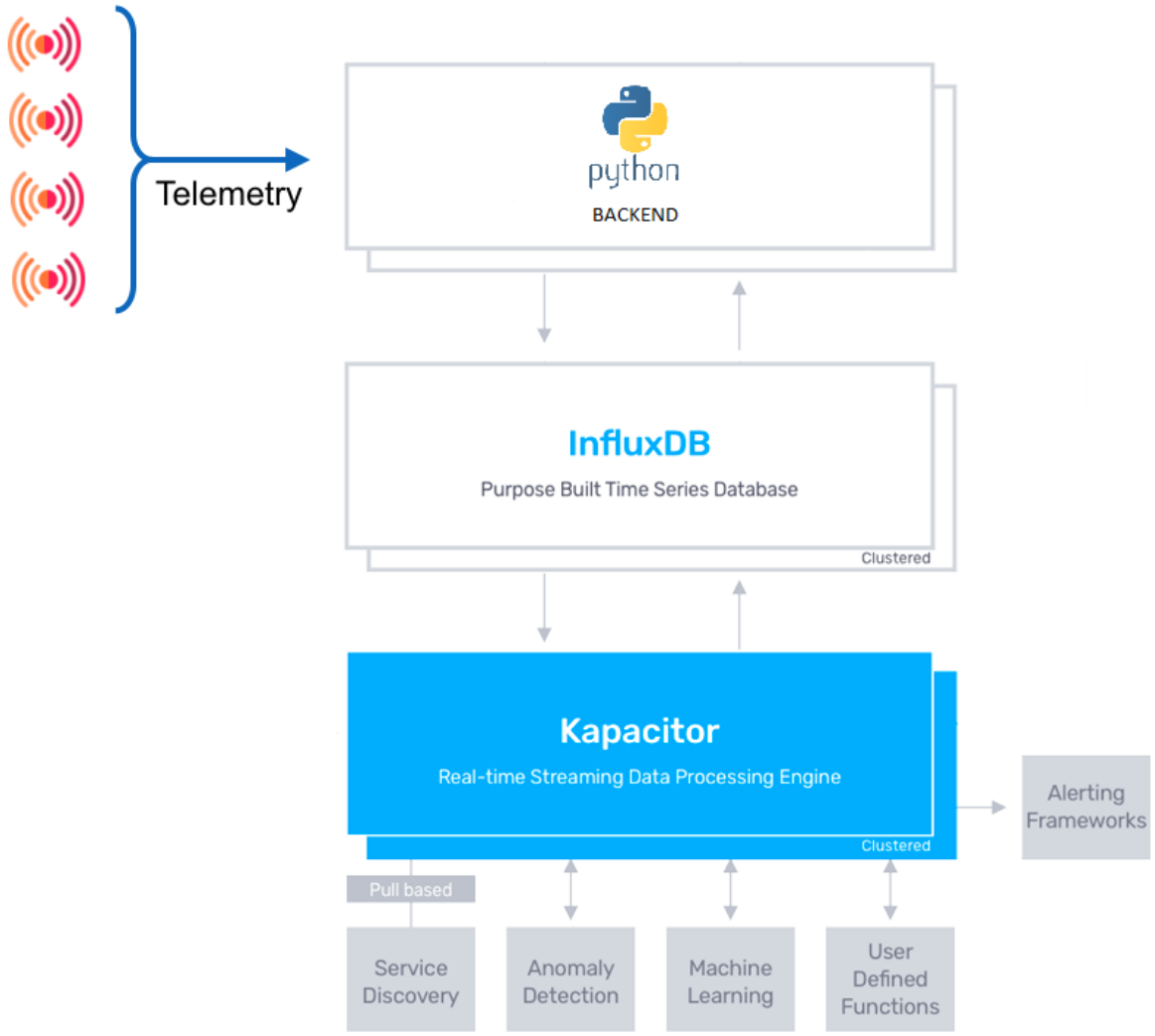
2.2.1 Backend

Aktuálna implementácia pracuje s relačnou databázou PostgreSQL. Tento prístup je výhodný pre určité funkcionality ako napríklad prácu so zariadeniami, oblasťami alebo rezerváciami. Na skladovanie nameraných dát zo senzorov je však výhodnejšie použiť time-series databázu. Tento typ databázy nám umožní zvládať veľký prísun informácii zo senzorov, aj v prípade, že sa infraštruktúra značne rozrastie. Okrem rýchlejšieho prístupu k dátam taktiež uľahčí ich analýzu a zefektívni ukladanie, čím ušetríme priestor na disku. InfluxDB je open-source time-series databázový systém, ktorý je často používaný v prostredí IoT. Má aktívnu podporu, je rýchly a poskytuje jednoduchú api na prácu s dátami. Taktiež podporuje nami používaný jazyk a to Python. Ďalšou našou požiadavkou je, aby databázový systém mal image pre Docker, čo InfluxDB splňuje.

Obrázok 32: Docker image pre InfluxDB

Návrh nového dátového modelu sa je rozdielny oproti pôvodnému len v tom, že všetky merania zo senzorov budú uložené v InfluxDB (obrázok v kapitole 1.3).

Okrem samotnej databázy InfluxDB budeme používať engine pre inteligentné spracovanie dát Kapacitor. Kapacitor je natívny dátový engine pre spracovanie údajov v samotnej Influx databáze. Primárnu úlohu bude pre nás zachytávať anomálie a neprijateľné hodnoty zo senzorov pri čom bude posielať upozornenia na nami vytvorený webhook prípadne Slack kanál. Skripty kapacitoru sú písané v jazyku TICKscript.



Obrázok 33: Prepojenie backendu, Influxu a Kapacitora

2.2.2 Frontend

V predošlej implementácii frontendu bol použitý react a komponentová knižnica. Nebudeme však pokračovať s touto implementáciou, ale vytvoríme celý frontend úplne odznova za použitia javascriptového frameworku React JS, Typescript a vizuálnej komponentovej knižnice purity UI. Pre zabezpečenie globálneho manažmentu stavov použijeme js knižnicu Redux. Autentifikácia bude v aplikácia vyriešená pomocou session token storage.

Návrh jednotlivých základných pohľadov na stránky webu sú znázornené nižšie.

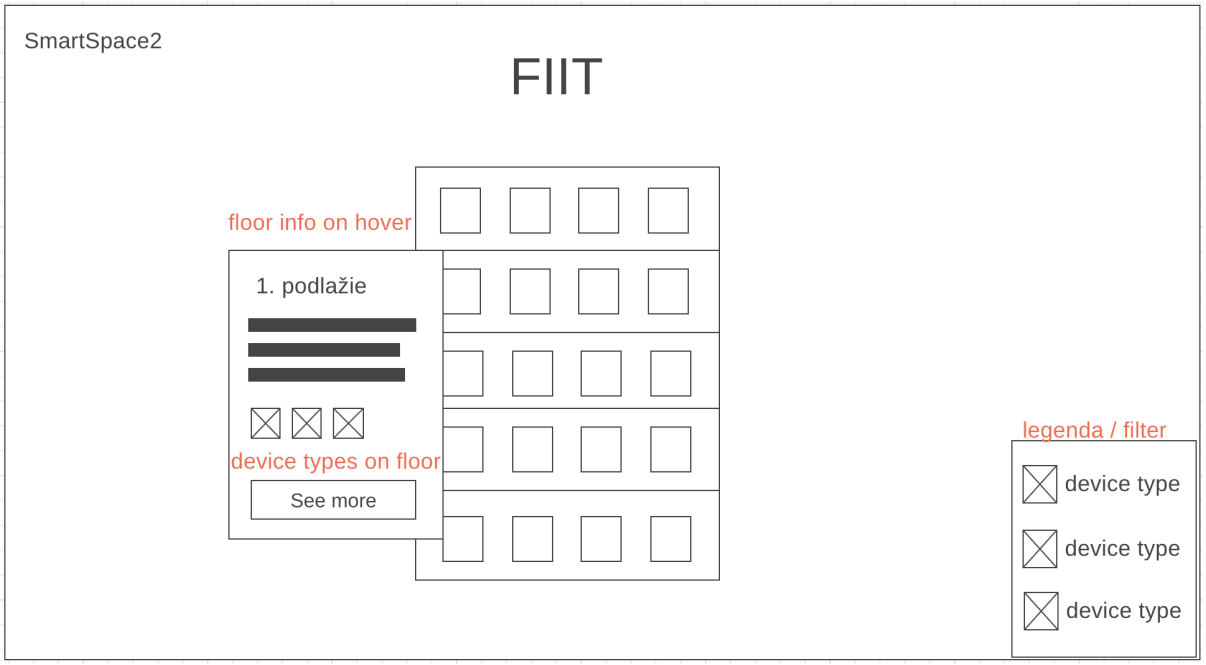
2.2.2.1 Dashboard screen

Bude zobrazovať sumarizáciu a štatistiky o jednotlivých zariadeniach v podobe grafov.

Obrázok 34: Návrh hlavnej obrazovky dashboardu

2.2.2.1 Floor plan screen

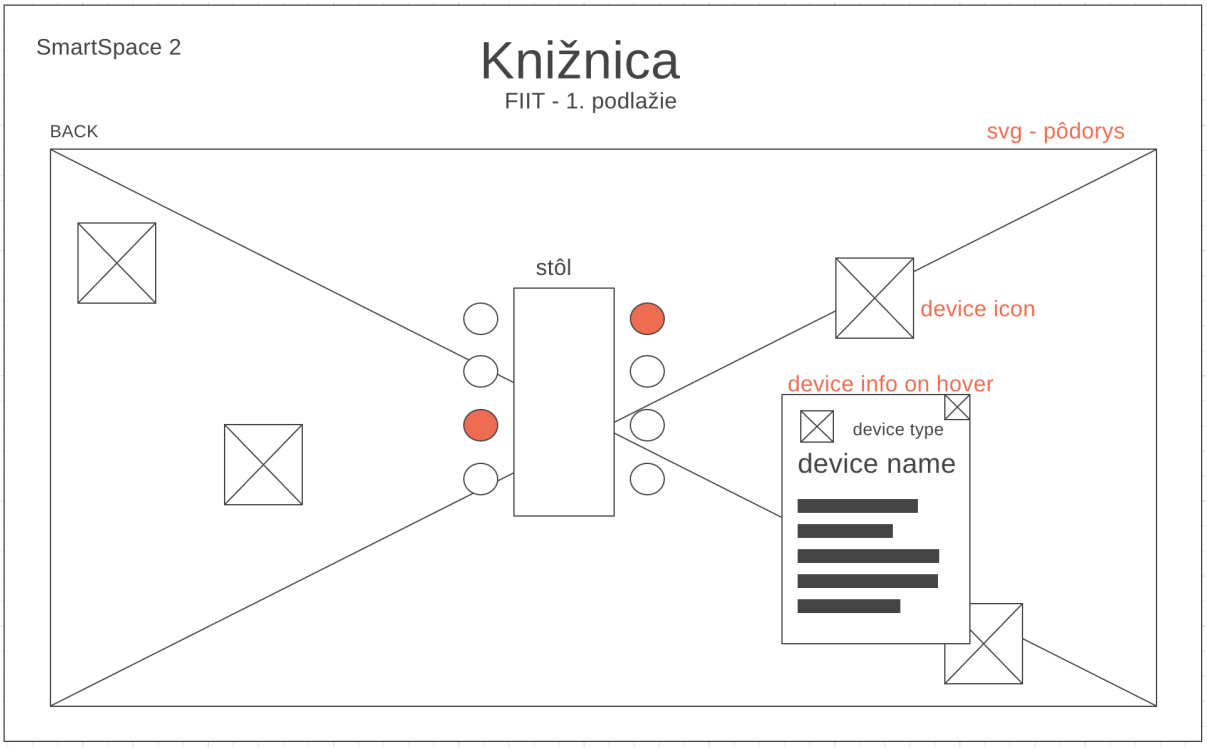
Bude obsahovať zoznam položiek - pôdorysov (Môže to byť mapa budovy, poschodia, miestnosti..) a jednotlivé objekty v nich (napríklad zariadenie, alebo vnorený objekt ako miestnosť). Jednotlivými pôdorysmi sa bude dať preklikávať a pridávať vnorené.



Obrázok 35: Návrh zobrazenia podlaží budovy

2.2.2.1 Devices screen

Bude obsahovať zoznam zariadení v miestnosti, pri ktorých sa na hover zobrazia podrobnosti o zariadení. Taktiež bude možné pridávať nové zariadenia a premiestňovať existujúce.



Obrázok 36: Návrh zobrazenia miestnosti so senzormi

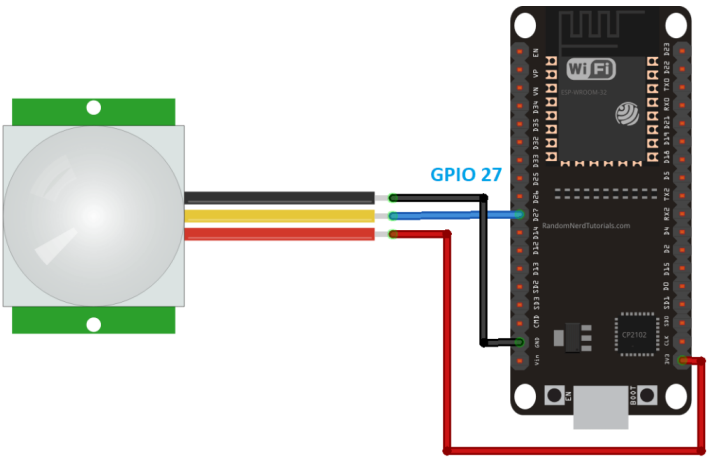
2.2.3 Senzory

2.2.3.1 Motion sensor

Tento senzor bude spolu s ESP32 v krabičke umiestnený pri každom mieste, kde chceme zaznamenávať pohyb. Primárne to bude pod stolom, aby sme vedeli, či na danom mieste niekto sedí.

Napojený bude na 5V na ESP32.

Schéma zapojenia



Obrázok 37: Zapojenie motion senzora

2.2.3.2 Air quality sensor

Senzor bude pripojený na ESP32, kde bude napájaný 5V a bude zachytávaný jeho analógový výstup. Keďže ESP32 obsahuje 12 bitový ADC(Analog to Digital Converter) a jeho referenčné napätie je 3.3V tak výstup zo senzora, ktorý sa bude pohybovať od 0V vyššie bude mapovaný na 4096 možných hodnôt. V nasledujúcej tabuľke je ukážka výstupného napätia a hodnoty, ktorú bude vracať ESP32 na základe prevodu:

Napätie analógového výstupu	Hodnota z ESP32 na základe 12 bitového ADC
0V	0
1.65V	2048
3.3V	4096
viac ako 3.3V	4096

Tabuľka 9: Číselná reprezentácia výstupného napätia

V prípade, že chceme z hodnoty v ESP32 dostať hodnotu napätia je potrebné danú číselnú reprezentáciu vydeliť 4096 a vynásobiť 3.3.

Senzor bude v neustálej prevádzke a bude každú minútu odosielať nameranú hodnotu, pri čom sa bude kontrolovať jej rozdiel oproti bežnej hodnote.

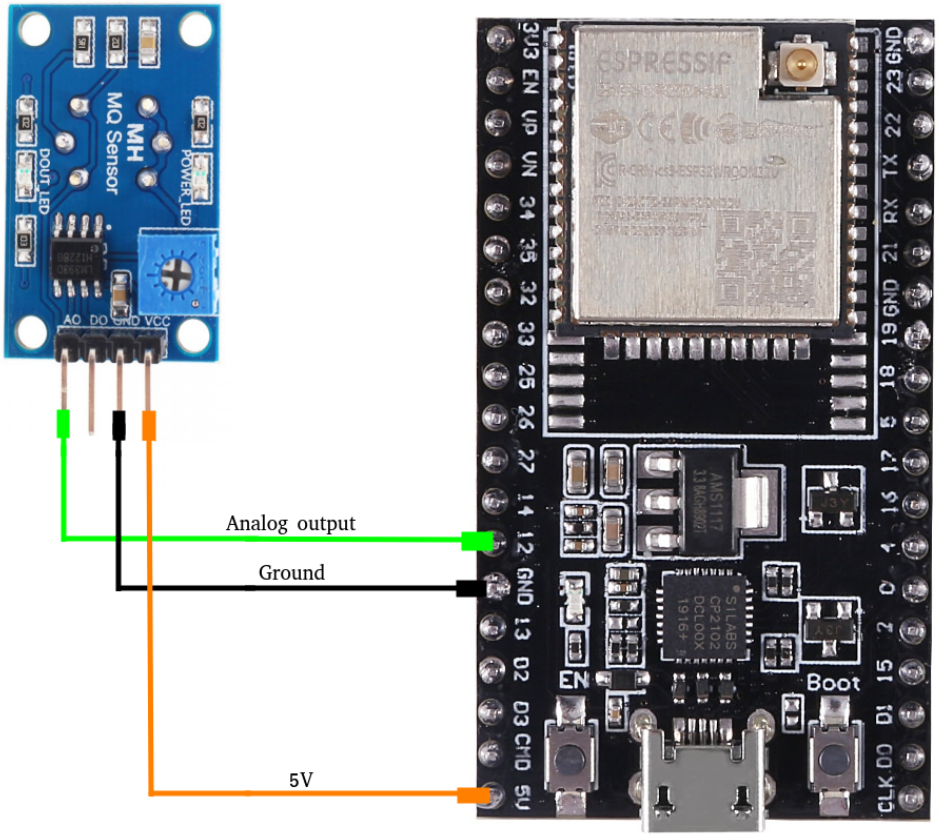
Keďže senzor dokázal vracať len hodnoty napätia a nie konkrétny hodnoty ppm, rozhodli sme sa experimentálne určiť pri ktorých hodnotách bude tento výstup považovaný za nebezpečný. Experiment sa vykonával postupným meraním hodnôt v rôznych prostrediach a pri rôznych látkach. Medzi tieto faktory ovplyvňujúce experiment patria látky ako:

- Acetón
- Propan-Bután

Medzi testované prostredia patria:

- Prostredie vonku v prírode, mimo obývaných prostredí
- Prostredie vo vetranej izbe v husto obývanom prostredí
- Prostredie v nevetranej izbe v husto obývanom prostredí
- Prostredie v zadymenom(tabakové výrobky) nevetranom priestore

Z týchto hodnôt následne vytvoríme tabuľku namapovanú na jedenásť dielne spektrum na lepšiu priblíženie nameraného napätia pre potenciálnych používateľov.



Obrázok 38: Zapojenie Air Quality senzora

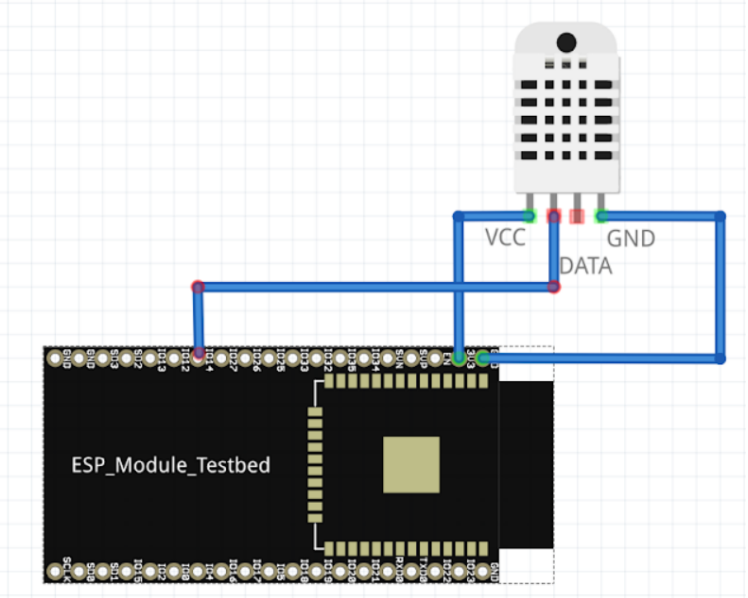
2.2.3.3 Humidity and Temperature sensor

Ide o senzor, ktorý bude v krabičke spolu s ESP32 napájaný batériou a umiestnený pod pracovným stolom v strede miestnosti, čo nám zaručí že údaje nebudú nijakým spôsobom ovplyvnené nepriaznivými faktormi, akými je napríklad rozdielna teplota/vlhkosť v blízkosti okna.

2.2.4 Kiosk

Rozhodli sme sa že pre zlepšenú interakciu používateľa s “Inteligentnou miestnosťou” bude potrebné aj zariadenie, ktoré bude nasadené pred vstupnými dverami do miestnosti. Malo by ísť o tablet/kiosk zariadenie, na ktorom budú zobrazené dáta o nameraných hodnotách zo senzorov v miestnosti, bude možné zobraziť všetky rezervácie pre danú miestnosť a taktiež aj obsadiť danú miestnosť za pomoci tohto zariadenia. Toto zariadenie bude implementované v JavaScriptovom frameworku Vue.js.

Schéma zapojenia



Obrázok 39: Zapojenie Temperature senzora

2.3 Implementácia

2.3.1 Backend

Ukladanie meraní zo senzorov a generátora bolo prerobené do time-series databázy Influx. Inlux poskytuje jednoduché vkladanie a získavanie dát prostredníctvom vlastnej API. Taktiež je jednoduché ho manažovať prostredníctvom prehľadného webového rozhrania.
Aby sa dáta ukladali do InfluxDB ale stále sa používali rovnaké API volania, bolo potrebné prerobiť MeasurementView tak, aby nepracovalo s ORM ale presmerovala dáta do Influx API. Z pohľadu generátorov dát, senzorov a ani klientov sa nič nezmenilo. Stále sa používajú rovnaké API volania do backendu avšak backend dáta iba presmeruje do druhej API.

API backendu bola rozšírená o nasledujúce volania

- /api/measurements/<area_id>. Tento endpoint umožňuje získať posledné merania zo všetkých senzorov v danej oblasti. Táto funkcionalita je kľúčová pre použitie v kiosku a taktiež pre zobrazovanie aktuálneho stavu v miestnostiach na webovej stránke.
- /api/physica_units. Vracia všetky možné fyzikálne jednotky, ktoré sú v systéme definované. Vznikol z dôvodu, že pri vytváraní senzorov je nutné zadať ID fyzikálnej jednotky, v ktorej daný senzor meră.
- /api/areas?root=true. Vracia všetky oblasti, ktoré sú root, čiže všetky čo nepatria do žiadnej inej oblasti.
- /api/areas/<area_id>/devices. Vráti všetky zariadenia, ktoré sa nachádzajú v danej oblasti. Je to potrebné pre priradenie novovytvoreného senzoru ku konkrétnemu devicu.
- /api/areas?office_name. Vracia area_id pre konkrétne meno oblasti. Túto funkcionalitu bolo potrebné implementovať pre kiosk.
- /api/devices/<device_id>/sensors. Vracia ID všetkých senzorov, ktoré patria k danému devicu.

Bolo nutné upraviť už existujúce enpointy pre vytváranie a úpravu area a device_type, z dôvodu pridania možnosti mať obrázok. V area bolo taktiež potrebné pridať polia x_coord a y_coord pre uloženie umiestnenia v parent area. Okrem toho, že museli byť upravené serializéry, taktiež bolo potrebné rozšíriť tieto modely. Obrázky sa ukladajú ako pole znakov v base64 kódovaní.

Konečný stav api endpointov je nasledovný. Každý api endpoint je obsiahnutýv swagger-i v technickej dokumentácií:

Práca so zariadeniami:

Metóda	**URI**	**Popis**
POST	/devices	Vytvorenie zariadenia
GET	/devices	Zoznam všetkých zariadení
GET	/devices/{device_id}	Informácie o zariadení
PUT	/devices/{device_id}	Aktualizácia informácii o zariadení
DELETE	/devices/{device_id}	Vymazanie zariadenia
POST	/devices_types/{device_id}/ devices	Vytvorenie viacerých zariadení so špecifickým typom
DELETE	/devices_types/{device_id} devices	Vymazanie všetkých zariadení so špecifickým typom

Tabuľka 10: Volania API pre zariadenia

Práca s typmi zariadeni:

Metóda	URI	Popis
POST	/devices_types/	-
GET	/devices_types/	-
GET	/devices_types/{device_type_id}	Získa informácie o type zariadení
PUT	/devices_types/{device_type_id}	Aktualizácia informácii o type
DELETE	/devices_types/{device_type_id}	Odstránenie typu zariadenia

Tabuľka 11: Volania API pre typy zariadení

Práca s meraniami:

Metóda	URI	Popis
POST	/measurements	Vytvorenie zoznamu meraní
GET	/devices/{device_id}/last_measurement	Posledné meranie zariadenia
GET	/devices/{device_id}/measurements	Zoznam meraní zariadenia na základe časového úseku

Tabuľka 12: Volania API pre merania

Práca s rezerváciami:

Metóda	URI	Popis
POST	/reservations	Vytvorenie rezervácie
PUT	/reservations/{reservation_id}	Aktualizácia rezervácie
DELETE	/reservations/{reservation_id}	Vymazanie rezervácie
GET	/reservations/{area_id}	Všetky rezervácie pre danú oblasť

Tabuľka 13: Volania API pre rezervácie

Práca so senzormi:

Metóda	URI	Popis
POST	/senors	Vytvorenie zoznamu senzorov

PUT	/senors/{sensor_id}	Aktualizácia informácii o senzore
GET	/senors/{sensor_id}	Získanie informácii o senzore
DELETE	/senors/{sensor_id}	Vymazanie senzora
GET	/sensor/filter	Získanie informácii o viacerých senzoroch
DELETE	/sensor/filter	Vymazanie viacerých senzorov

Tabuľka 14: Volania API pre senzory

Práca s autentifikáciou:

Metóda	URI	Popis
POST	/authenticate	Autentifikácia používateľa
POST	/authenticate/refresh	Obnovenie prístupového tokenu

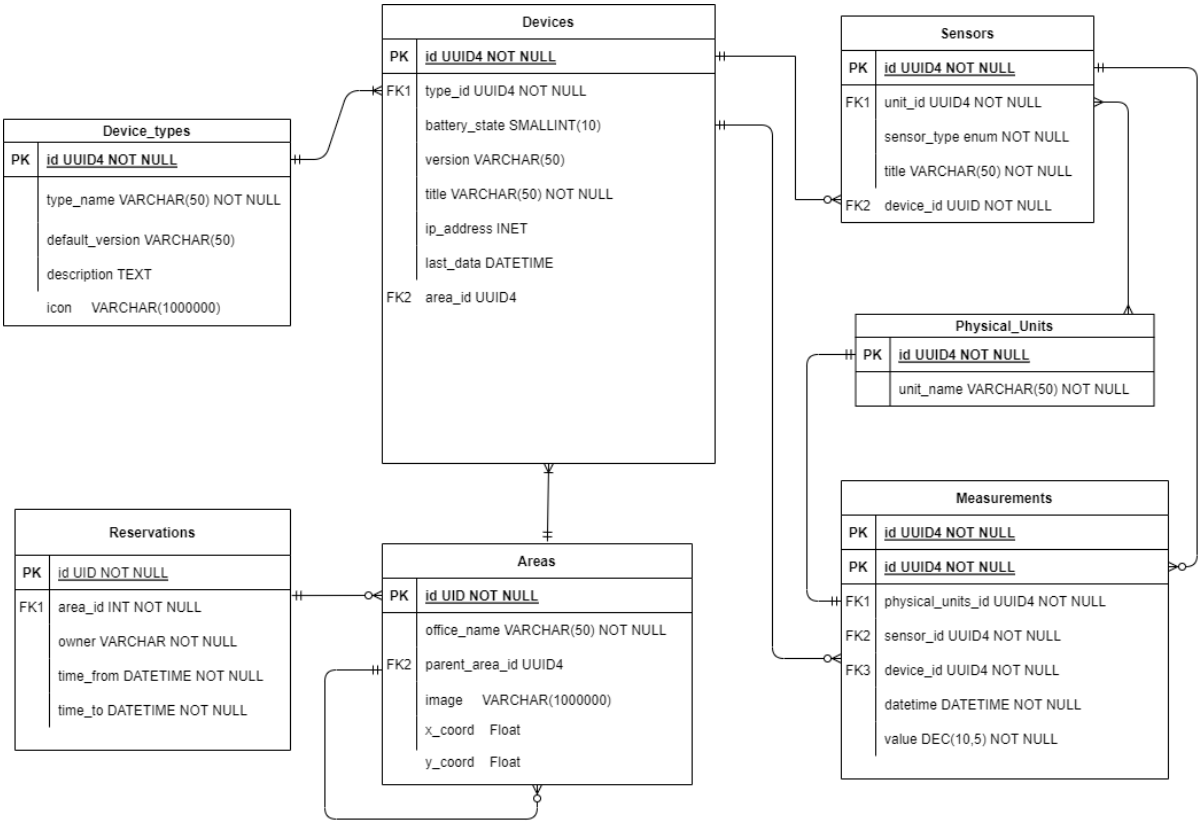
Tabuľka 15: Volania API pre autentifikáciu

Práca s oblasťami:

Metóda	URI	Popis
POST	/areas	Vytvorenie novej oblasti
GET	/areas	Získanie zoznamu oblastí
GET	/areas?office_name=<názov oblasti>	Získanie identifikátora oblasti pre konkrétny názov
GET	/areas?root=<True>	Získanie všetkých oblastí, kde *parent_id* obsahuje prázdny reťazec
GET	/areas/{areas_id}	Získanie informácii o oblasti a všetkých oblastiach, ktoré sú jej priamym potomkom
PUT	/areas/{areas_id}	Aktualizácia oblasti
DELETE	/areas/{areas_id}	Vymazanie oblasti
GET	/areas/{areas_id}/devices	Získanie jednotlivých zariadení pre konkrétnu oblasť

Tabuľka 16: Volania API pre oblasti

Takto vyzerá výsledný upravený dátový model, Bolo nevyhnutné zmeniť relácie medzi senzormi, device a device_type. Senzor nebolo možné namapovať na konkrétne zariadenie, kvôli tomu, že predchádzajúca relácia bola cez tabuľku device_type. V súčasnosti má každý senzor ako cudzí kľúč primárny kľúč konkrétneho senzoru. To znamená, že senzor patrí ku konkrétnemu zariadeniu a jedno zariadenie môže mať nekonečno senzorov:



Obrázok 40: Dátový model po úprave

2.3.2 Frontend

Pre implementáciu frontendu sme si pripravili ukážkové wireframes, ktoré v aplikácii budeme používať. Následne sme implementovali niektoré obrazovky s menšími modifikáciami podľa návrhu.

2.3.2.1 Vizuál

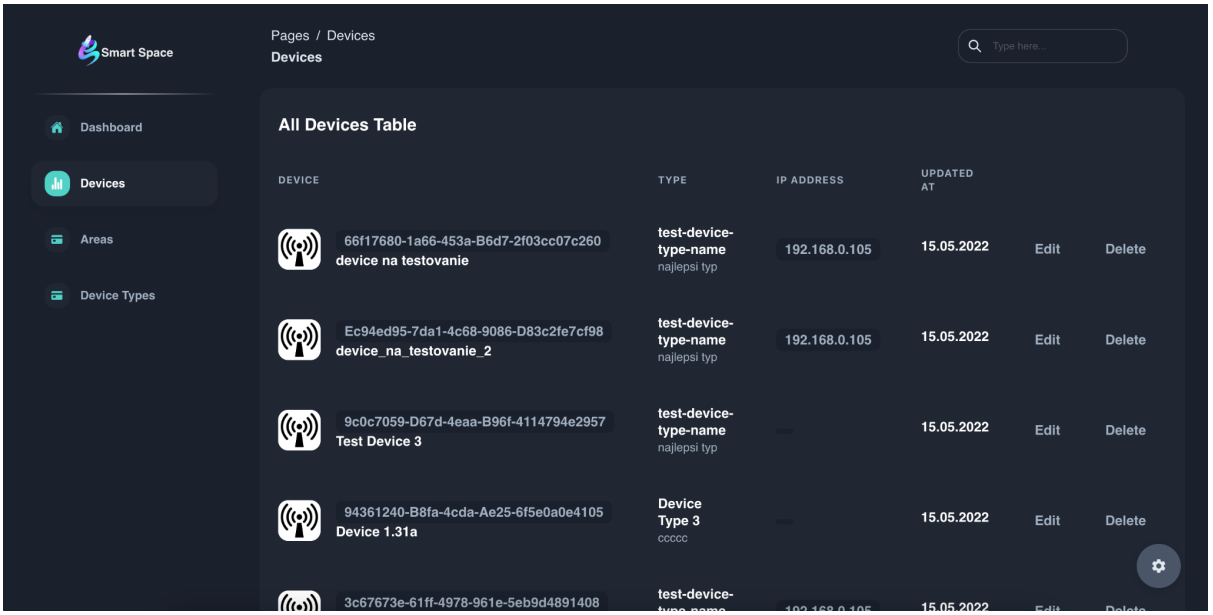
Dashboard

Úvodná stránka s logom našej aplikácie.

Obrázok 41: Dashboard

Device list

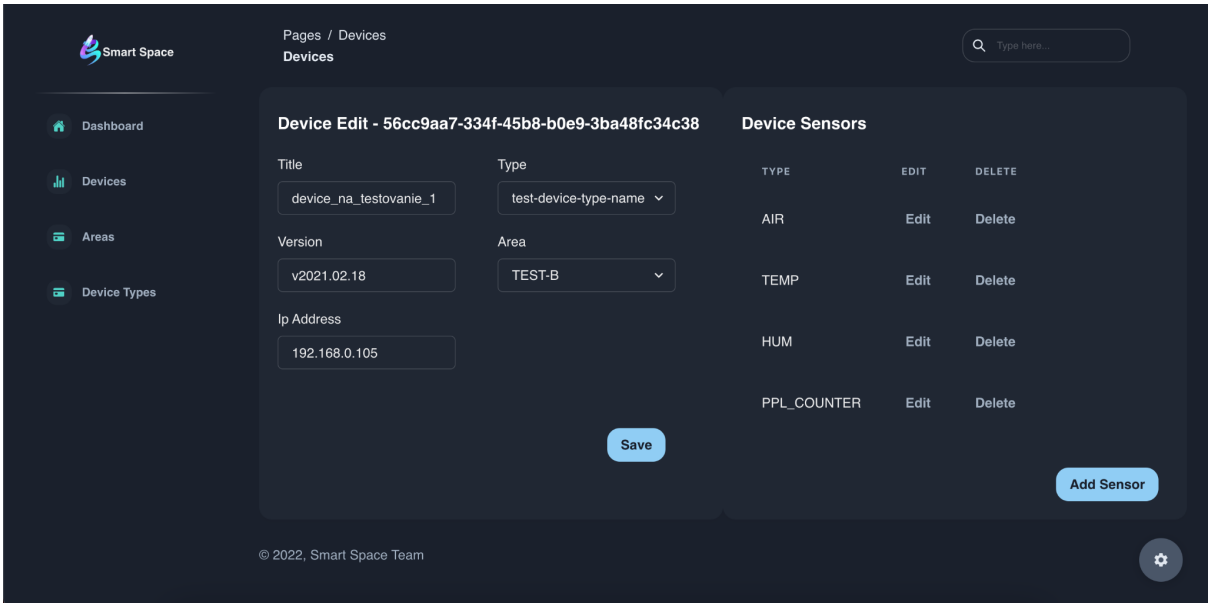
Obsahuje zoznam všetkých zariadení s možnosťou ich úpravy/zmazania.



Obrázok 42: Zoznam zariadení

Device update

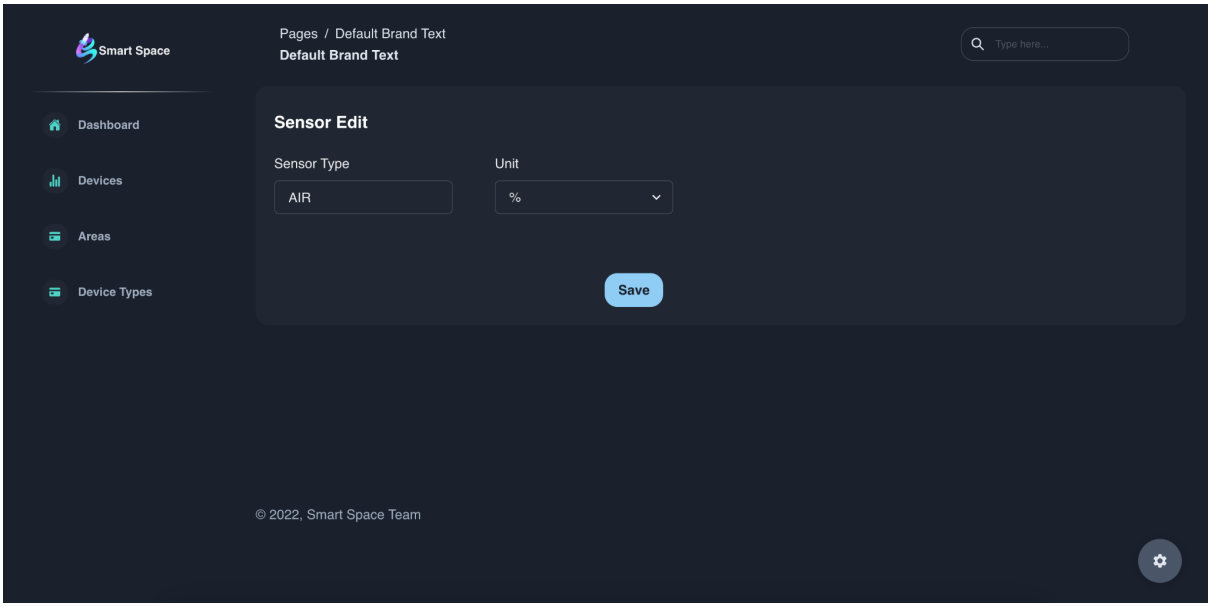
Obsahuje formulár na úpravu existujúceho zariadenia. Taktiež obsahuje zoznam všetkých senzorov, ktoré má zariadenie priradené s možnosťou ich úpravy/vytvorenia/zmazania.



Obrázok 43: Edit zariadenia

Sensor Create/Update

Obsahuje formulár na vytvorenie/úpravu existujúceho senzoru.



Obrázok 44: Edit senzoru

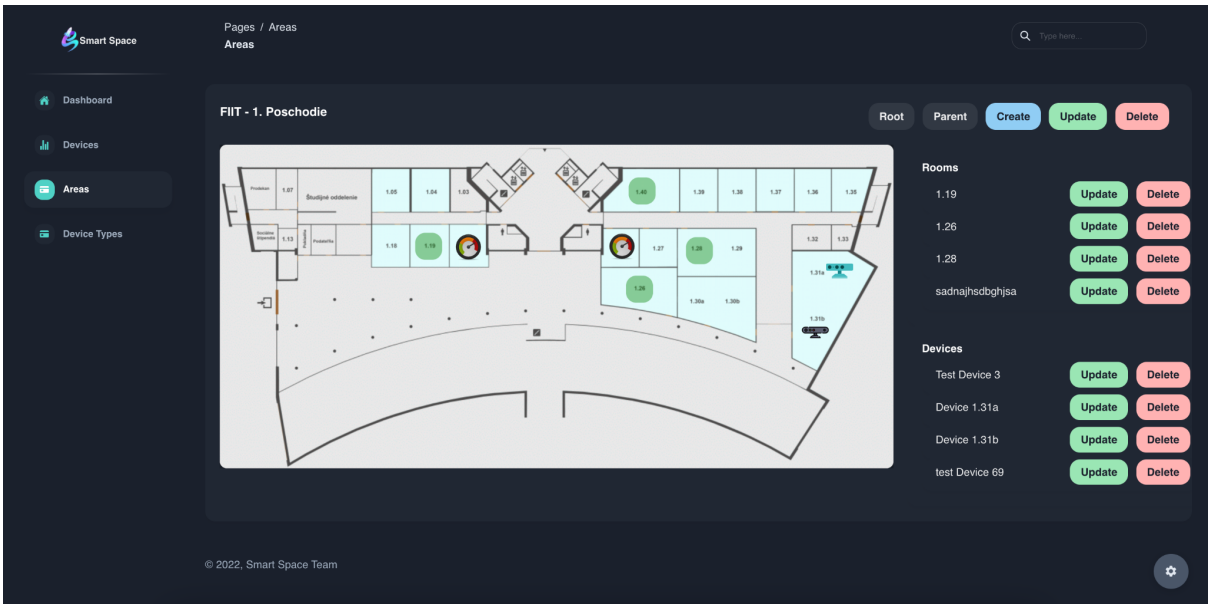
Area and Device list

Obsahuje zoznam všetkých základných(root) oblastí/miestností a zariadení s možnosťou ich úpravy/vytvorenia/zmazania.

Obrázok 45: Zoznam oblastí a zariadení

Area Update

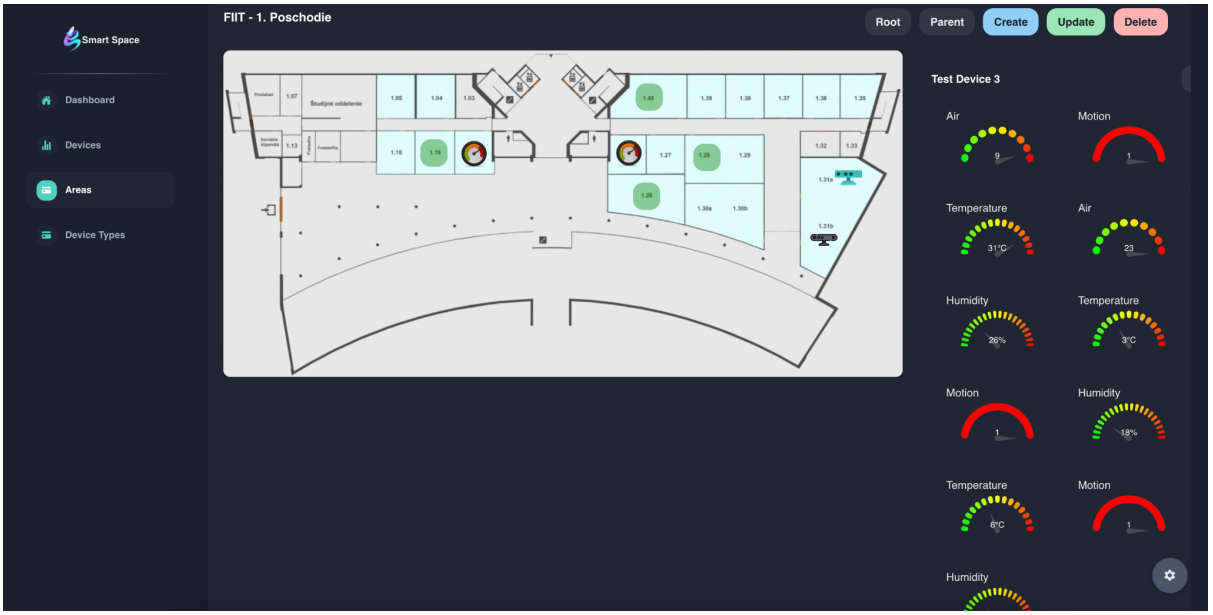
Obsahuje detail oblasti/miestnosti s jej fotografiou a všetkými zariadeniami/oblasťami, ktoré sa v tejto oblasti nachádzajú(takzvané child areas/devices) s možnosťou ich úpravy/vytvorenia/zmazania. Oblasť/zariadenie je možné pridať/upraviť kliknutím na konkrétne miesto pôdorysu oblasti.



Obrázok 46: Detail oblasti

Area device details

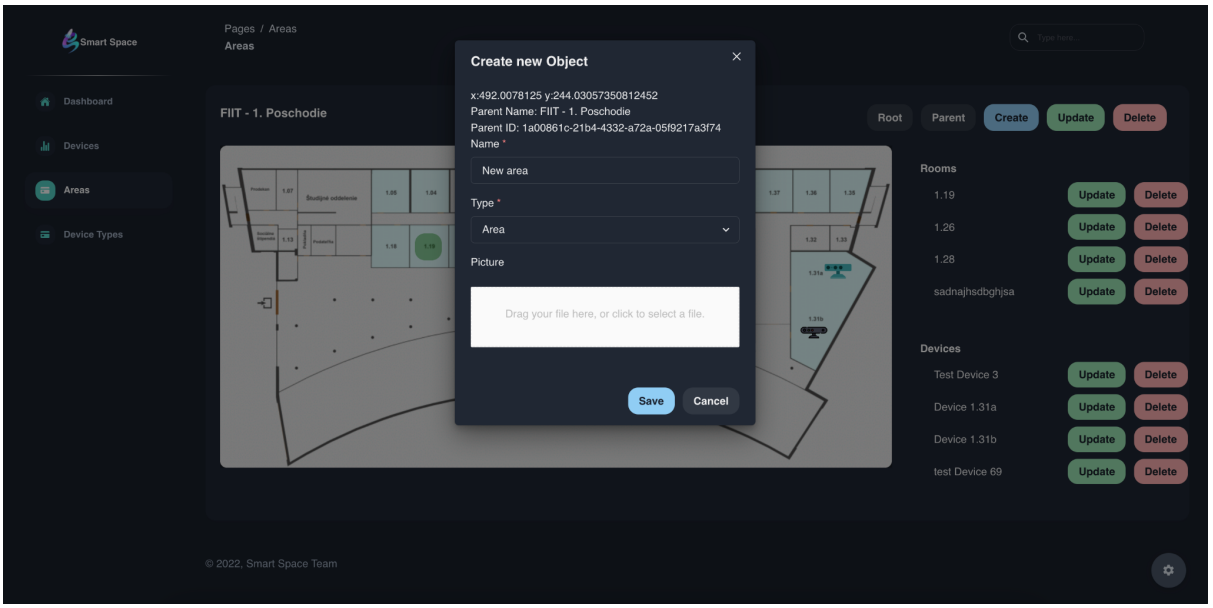
Obsahuje detail zariadenia nachádzajúceho sa v danej oblasti s meraniami zo senzorov.



Obrázok 47: Detail zariadenia so senzormi

Area child Create/Update

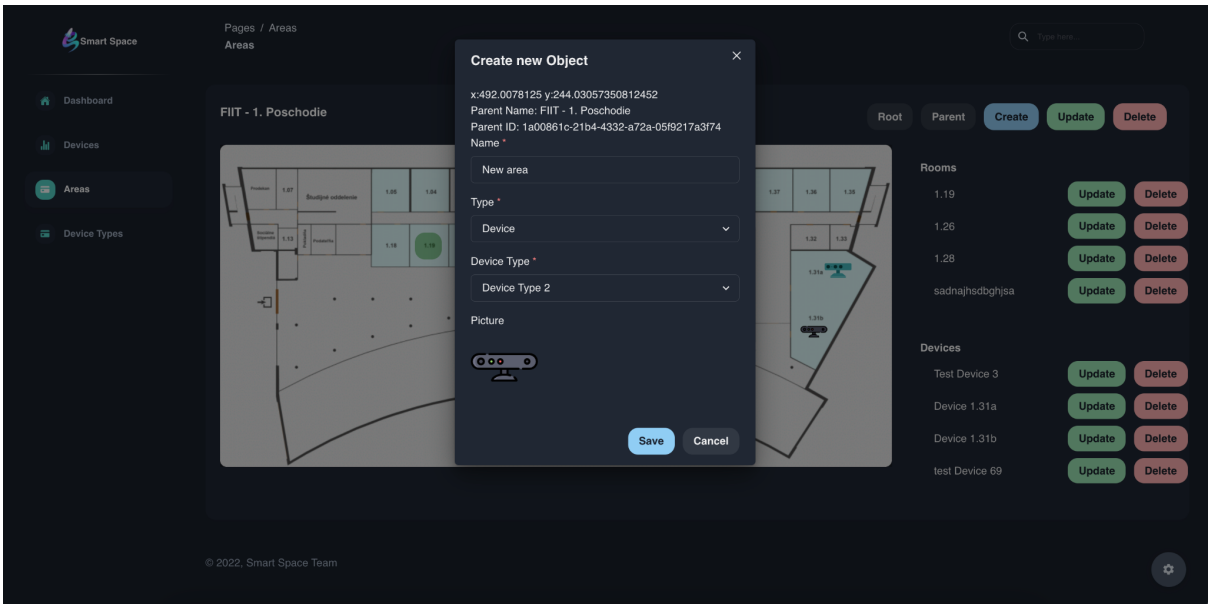
Obsahuje modal na vytvorenie/úpravu oblasti pre konkrétnu otcovskú oblasť.



Obrázok 48: Vytvorenie oblasti

Area Device Create/Update

Obsahuje modal na vytvorenie/úpravu zariadenia pre konkrétnu otcovskú oblasť.



Obrázok 49: Vytvorenie zariadenia

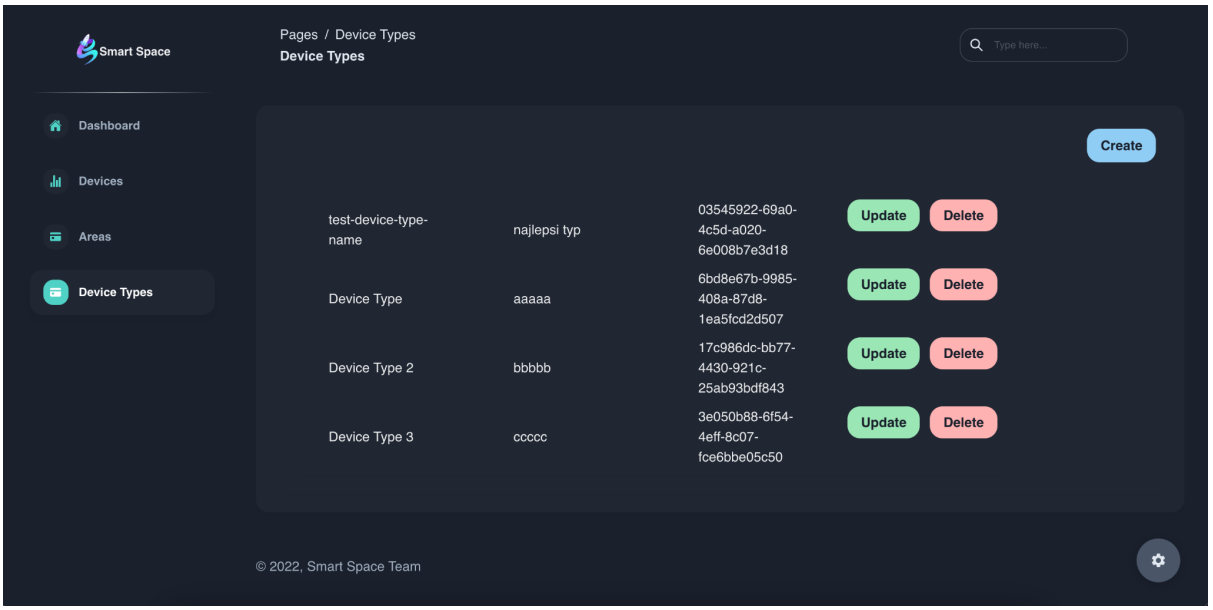
Device detail

Obsahuje merania z priradených senzorov.

Obrázok 50: Detail zariadenia so všetkými senzormi

Device type list

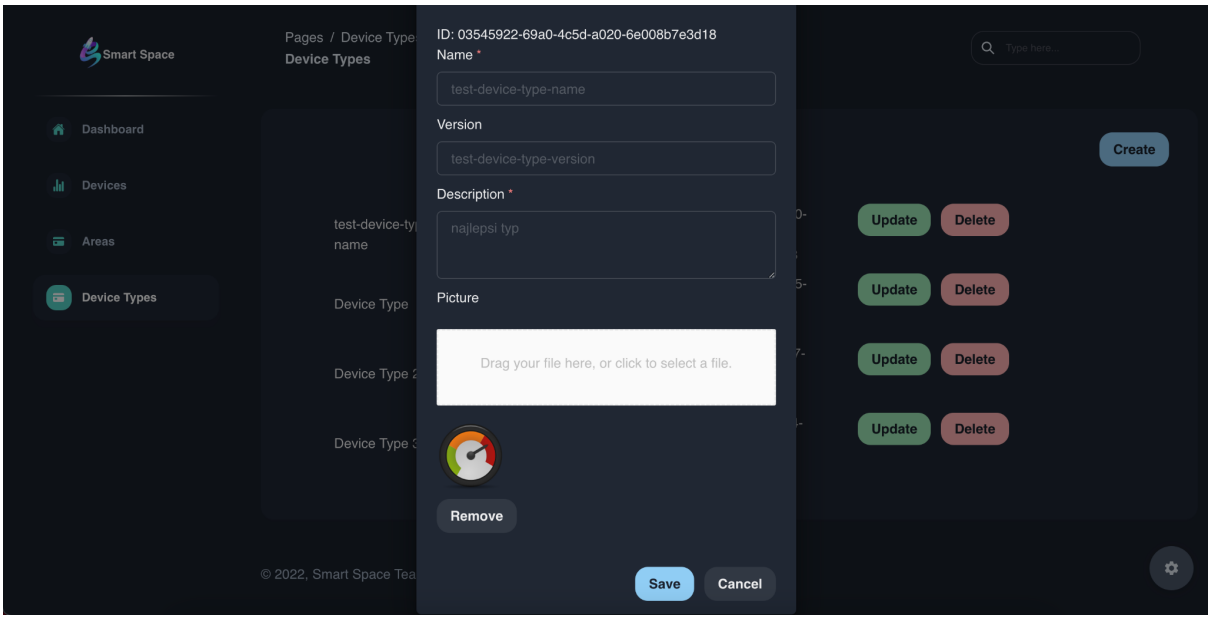
Obsahuje zoznam všetkých typov zariadení s možnosťou ich vytvorenia/úpravy/zmazania.



Obrázok 51: Zoznam typov zariadení

Device type Create/Update

Obsahuje formulár na vytvorenie/úpravu existujúceho typu zariadenia.



Obrázok 52: Úprava typu zariadenia

2.3.3 Senzory

Každý senzor bude dáta odosielať cez MQTT protokol. Zdrojový kód pre senzory je vytváraný v prostredí Arduino IDE. Všetky senzory majú spoločné to, že sa pripájajú na wifi a publikujú namerané dáta. Tieto dáta sú ďalej spracované pomocou Workera a posielané do backendu.

2.3.3.1 Motion sensor

Senzor funguje na nasledovnom princípe - počas zadaného počtu sekúnd sa snaží zachytiť pohyb a zaznamenáva si, či pohyb zachytil alebo nie. Ak zachytí pohyb vo viacerých rôznych meraniach, znamená to, že dané miesto je obsadené. Merania vykonáva v pravidelných intervaloch.

Dáta publikuje v tvare
"data=" + data + ";device=" + deviceID + “;sensor=" + sensorID

Po odoslaní dát je následne podľa nastavenia X minút nečinný (5).

2.3.3.2 Air quality sensor

Senzor každú minútu odmeria hodnotu výstupného napätia a odošle ju. Senzor beží nepretržite a po odoslaní dát cez MQTT je nečinný po dobu jednej minúty. Ako výstup používa výlučne analógový výstup nakoľko je oveľa presnejší ako digitálny výstup, ktorý dokáže odosielať len 2 hodnoty.

Dáta publikuje v tvare
"data=" + data + ";device=" + deviceID + “;sensor=" + sensorID

Odosielané dáta obsahujú kladnú celočíselnú hodnotu od 0 po 4096, ktorá reprezentuje výstupné analógové napätie senzora. Hodnota je počítaná automaticky ADC(analog-to-digital-converter) prevodníkom, ktorý má rozsah 12 bitov.

Na základe nameraných hodnôt počas experimentu sme pre látky a prostredia v návrhu namerali nasledovné hodnoty

Látka / prostredie	Nameraná hodnota
Acetón - množstvá	menej ako 2.0
Acetón - množstvá	2.0 - 2.4
Acetón - množstvá	viac ako 2.4
Propán-Bután - množstvá	menej ako 2.0
Propán-Bután - množstvá	2.0 - 2.4
Propán-Bután - množstvá	viac ako 2.4
Vzduch v mestskom prostredí	1.85 - 1.88
Vzduch v nevetranej izbe	1.88 - 1.91
Vzduch v zadymenom prostredí	1.91 - 2.00

Tabuľka 17: Namerané hodnoty pri testovacích látkach

Hodnota	Spodná hranica napätia	Horná hranica napätia	Škála napätie pre danú hodnotu
0	0	1.85	-
1	1.86	1.90	0.04
2	1.91	1.94	0.03
3	1.95	1.97	0.02
4	1.98	2.01	0.03
5	2.02	2.10	0.08
6	2.11	2.25	0.14
7	2.26	2.50	0.24
8	2.51	2.85	0.34
9	2.86	3.29	0.43
10	3.30	-	-

Tabuľka 18: Hranice napätí k jednotlivým hodnotám

2.3.3.3 Temperature and Humidity sensor

Tento senzor bude schopný zaznamenávať hodnoty o vlhkosti a teplote v ľubovoľných intervaloch, kde najkratší možný interval sú 2 sekundy. Ako základný interval merania je prednastavený na 1 minútu. Po nameraní potrebných dát sa ESP prepne do úsporného režimu.

Dáta publikuje rovnako ako predchádzajúce senzory v tvare:

Dáta odosielané zo senzoru určeného na meranie vlhkosti:
payload = "data=" + String(h) + ";device=" + String(DEVICE) + ";sensor=" + String(H_SENSOR_ID);

Dáta odosielané zo senzoru určeného na meranie teploty:
payload = "data=" + String(t) + ";device=" + String(DEVICE) + ";sensor=" + String(T_SENSOR_ID);

2.3.4 Kiosk

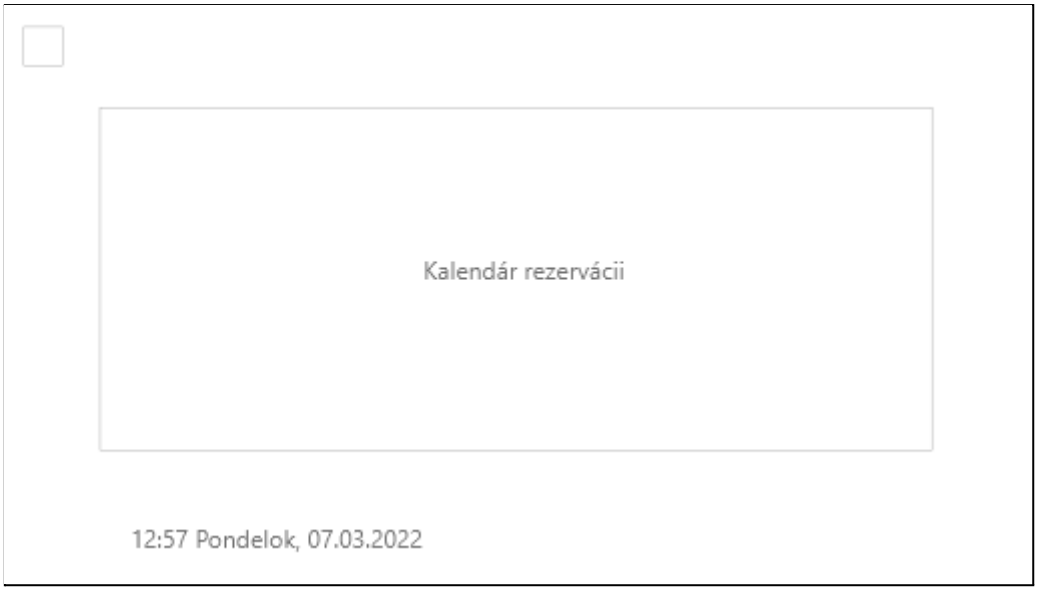
2.3.4.1 Návrh

V prípade zariadenia Kiosk, sa na začiatku vytvorili potrebné wireframy v aplikácii AdobeXD, ktoré mali zobrazovať vyzuálny návrh aplikácie.

Wireframe zobrazujúci rozloženie hlavnej stránky Kiosku, na ktorej sú zobrazené namerané hodnoty z miestnosti, a taktiež tlačidlo umožňujúce obsadenie miestnosti.

Obrázok 53: Dashboard

Ďalším vytvoreným wireframe-om bola podstránka určená pre zobrazenie aktívnych rezervácii pre danú miestnosť.

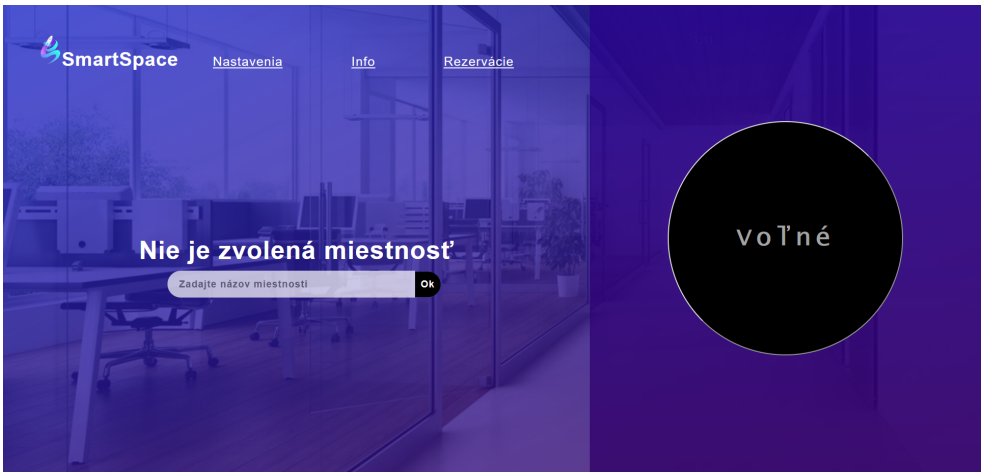


Obrázok 54: Kalendár s rezerváciami

2.3.4.2 Vizuál

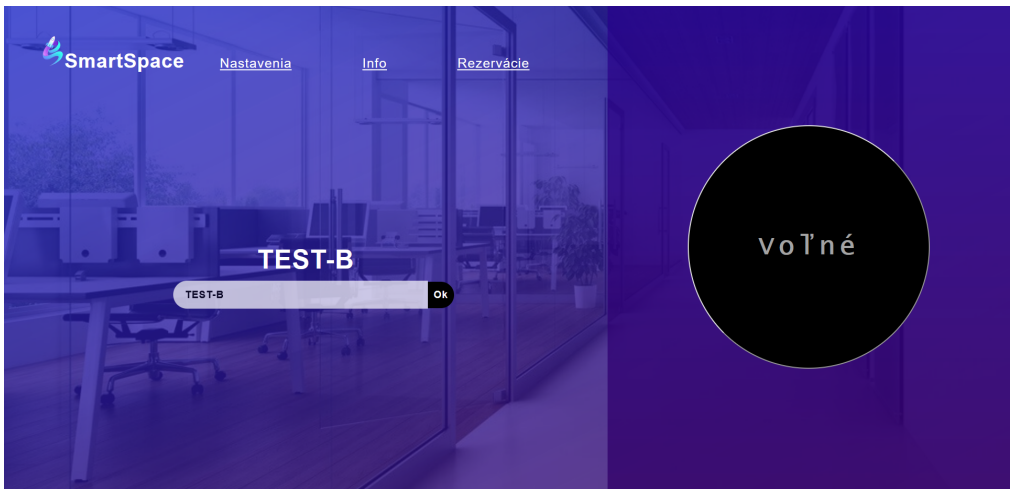
Webová aplikácia pre Kiosk bola implementovaná v JavaScriptovom frameworku Vue.js.

Po načítaní aplikácie sa zobrazí domovská stránka, kde je potrebné zadať názov miestnosti pre ktorú sa budú zobrazovať namerané dáta.



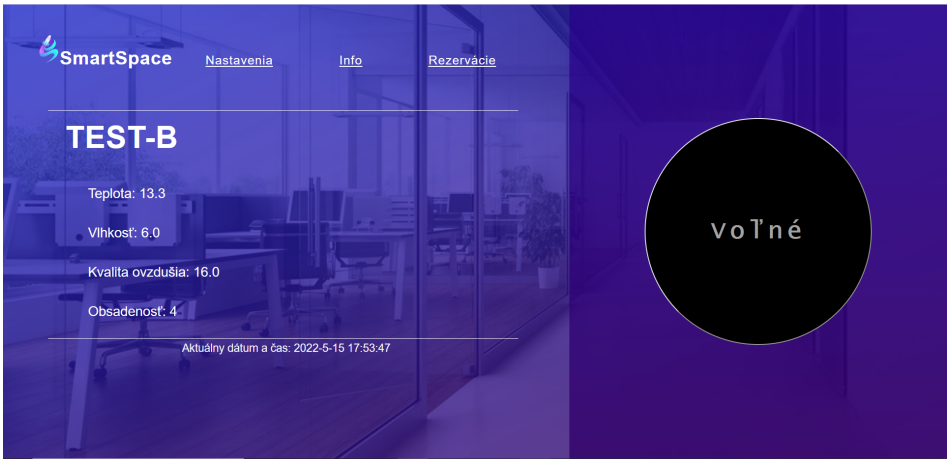
Obrázok 55: Hlavná obrazovka

Po zvolení správneho mena miestnosti, nám aplikácie umožňuje v navigácii prekliknúť na podstránku info, kde sú zobrazené dáta o miestnosti alebo na podstránku Rezervácie, kde sa dajú pozrieť dátumy, kedy je pre danú miestnosť vytvorená rezervácia. Za pomoci tlačidla s nápisom Voľné je možné danú miestnosť v danom momente obsadiť a vytvorí sa aj hodinová rezervácie v prípade že pre daný čas ešte nie je rezervácia vytvorená.



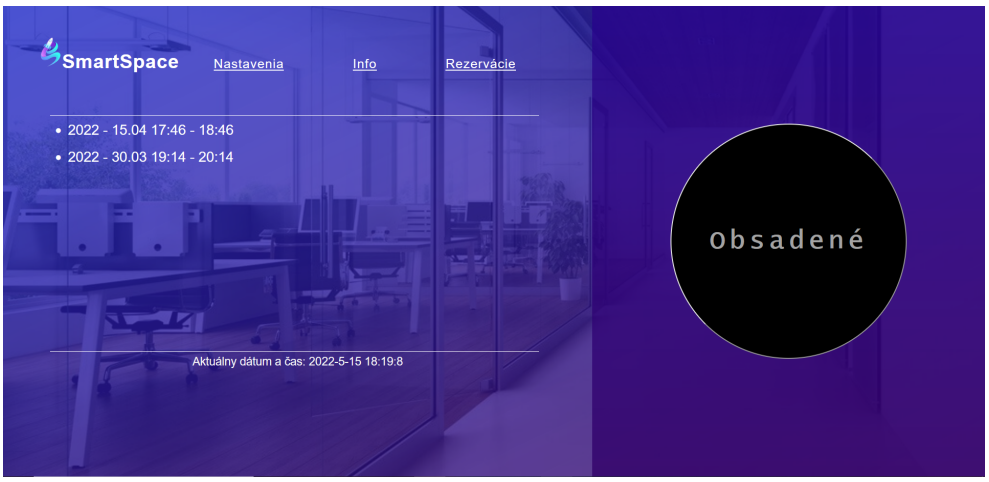
Obrázok 56: Zvolená miestnosť

Ako už bolo spomenuté, podstránka s názvom Info je určená na zobrazenie nameraných hodnôt z príslušných senzorov danej miestnosti



Obrázok 57: Zobrazené namerané hodnoty

V prípade podstránky Rezervácie sa zobrazujú vytvorené rezervácie pre danú miestnosť.



Obrázok 58: Výpis rezervácií

2.4 Testovanie

Keďže veľkú časť projektu tvoria senzory, testovanie sa nedá robiť automaticky. Preto pri testovaní skúšame každý senzor osobitne pomocou Arduino IDE -> Serial Monitor. Ak funguje správne, následne sa otestuje publikovanie dát na jednoduchom kóde, ktorý má len za úlohu prijímať dáta a vypisovať ich aj s ID zariadenia, ktoré tieto dáta poslalo. Ak úspešne prejde aj tento krok, dáta sa začnú posielať už na Workera, kde sa predpokladá, že všetko funguje už správne a senzor sa môže zapojiť do prevádzky.
Worker je pred prijímaním dát zo senzorov otestovaný pomocou generátora dát. Tento generátor vytvára rovnaké dáta ako senzory ale vieme si ho upravovať podľa svojho uváženia.
Na testovanie API endpointov sme vytvorili vlastný skript, ktorý postupne vyskúša všetky používané endpointy. Postupne prejde cez celú logiku ako vytváranie oblastí, vytváranie zariadení, senzorov a získavanie dát. Skúša operácie ako vytvorenie, upravenie a vymazanie z databázy. Tento spôsob testovania nám umožní vyskúšať nie len logiku jednotlivých endpointov ale taktiež dostupnosť celého servera. V gitlab pipeline sme vytvorili jeden testovací pipeline, ktorý vyskúša pri operáciách commit, push alebo merge skúsi vytvoriť kontajner z backendu a tak otestujeme, či daný kontajner budeme vedieť spustiť aj na servery.

Prílohy

Technická dokumentácia

Technická dokumentácia obsahuje dokumentáciu z backendu, kiosku a swaggera. Tieto dokumentácie sú umiestnené v osobitnom súbore/files/tech_doc.zip, keďže ich nebolo možné pridať sem.

Inštalačná príručka

SmartSpace2

Všetky časti nášho systému, okrem samotných IoT zariadení je možné spustiť pomocou Docker kontajnerov. Systém sa skladá z nasledujúcich častí

1. Backend - databázy, webová aplikácia na sprostredkovanie API
2. Web - administrátorský portál
3. Worker - program, ktorý zachytáva všetky merania senzorov z MQTT protokolu a posiela ich na backend
4. IoT zariadenia - zariadenia ako ESP32 a Raspberry Pi spolu s pripojenými senzormi
5. Kiosk - webová aplikácia

Inštalácia backendu

Backend je možné spustiť pomocou Dockeru.
Inštalácia Dockeru a docker-compose na Linuxe s balíkovým manažérom *apk*

- sudo apt install docker.io
- sudo apt install docker-compose

Z gitlabu naklonujeme potrebné repozitáre:

- git clone https://gitlab.com/smartspace_2/backend-api
- git clone https://gitlab.com/smartspace_2/devops

Vzniknú dva nové adresáre *backend_api* a *devops*. Z adresára *devops* potrebujeme súbor *docker-compose*:

- cp devops/docker-compose-docker-compose.yml .

Následne je možné vytvoriť image podľa daného *docker-compose.yml* súboru:

- sudo docker-compose -f docker-compose.yml build

Pre spustenie celého backendu už použijeme príkaz:

- sudo docker-compose -f docker-compose.yml up

V prípade, že chceme backend buildnúť a spustiť jedným príkazom môžeme použiť:

- sudo docker-compose -f docker-compose.yml up –build

Inštalácia web

Frontend administrátorský portál je možné spustiť viacero spôsobmi. Je implementovaný v React frameworku takže sa používajú nástroje podporujúce práve tento framework.

Frontend aplikácia je dostupná na adrese https://gitlab.com/smartspace_2/react-app/:

- git clone https://gitlab.com/smartspace_2/react-app/

Lokálne spustenie:

- Potrebné nástroje pre spustenie:
    a. npm - odskúšané na verziách *v8.5.5*, *v8.8.x*, *v8.9.0*
    b. node - odskúšané na verziách *v12.18.3*, *v16.15.0*
- V root priečinku aplikácie spustíme príkazy:
    a. *'npm install'* alebo *'npm i'*
    b. *'npm start'*

Spustenie ako Docker kontajner:

- Potrebné nástroje pre spustenie:
    - Docker
    - Docker Compose
- V root priečinku aplikácie spustíme príkazom: '*docker-compose up [-d]*'

Produkčný build balíček obsahujúci statické súbory:

- Potrebné nástroje pre spustenie:
    a. npm - odskúšané na verziách *v8.5.5*, *v8.8.x*, *v8.9.0*
    b. node - odskúšané na verziách *v12.18.3*, *v16.15.0*
- V root priečinku aplikácie spustíme príkazy:
    a. *'npm install'* alebo *'npm i'*
    b. *'npm run build'*
- Web stránka je potom dostupná v priečinku *<root>/build*. Ak po načítaní index.html zostane stránka v prehliadači prázdna, treba vyskúšať úplný reload (ctrl + F5).

IoT zariadenia

Zdrojový kód, ktorý použijeme na zariadení stiahneme z prislúchajúceho git repozitára a nahráme ho na zariadenie pomocou Arduino IDE. V zdrojovom kóde je potrebné zmeniť nasledujúce hodnoty:

```
// WiFi
char* WIFI_SSID = "LAPTOP-JKG3BO8M 2132";
char* WIFI_PASSWORD = "12345678";

// MQTT - change only MQTT_TOPIC
const char* MQTT_SERVER = "broker.mqttdashboard.com";
const int MQTT_PORT = 1883;
const char* MQTT_TOPIC = "tp/co2";
const char* MQTT_USER = "";
const char* MQTT_PASSWORD = "";

// App
const int INPUT_PIN = 0;
```

```
const char* DEVICE = "34037832-a6b6-11ec-b909-0242ac120002";
const char* SENSOR = "182414ec-4a66-445a-b379-987fa120c1ac";
const char* VERSION = "0.1.0";
```

Po zmenení hodnôt a nahratí kódu na zariadenie cez Arduino IDE už stačí zariadenie len spustiť a automaticky začne odosielať merania cez MQTT.

Inštalácia Kiosku

Kiosk je potrebné naklonovať z online repozitára.

- git clone https://gitlab.com/smartspace_2/kiosk.git

Po naklonovaní je potrebné vytvoriť Docker image pre Kiosk s názvom **tp-kiosk**

- cd kiosk
- docker build -t tp-kiosk .

Po vytvorení Docker image s názvom **tp-kiosk**

- docker run -it -p 8080:8080 --name kiosk tp-kiosk

Kiosk stránka je dostupná na porte 8080

Worker

Worker je inštalovaný pomocou Dockera. Je potrebné naklonovať git repozitár pomocou príkazu:

- git clone https://gitlab.com/smartspace_2/worker

Vstúpime do vytvoreného adresára.
Pomocou dockera vytvoríme image a spustíme kontajner:

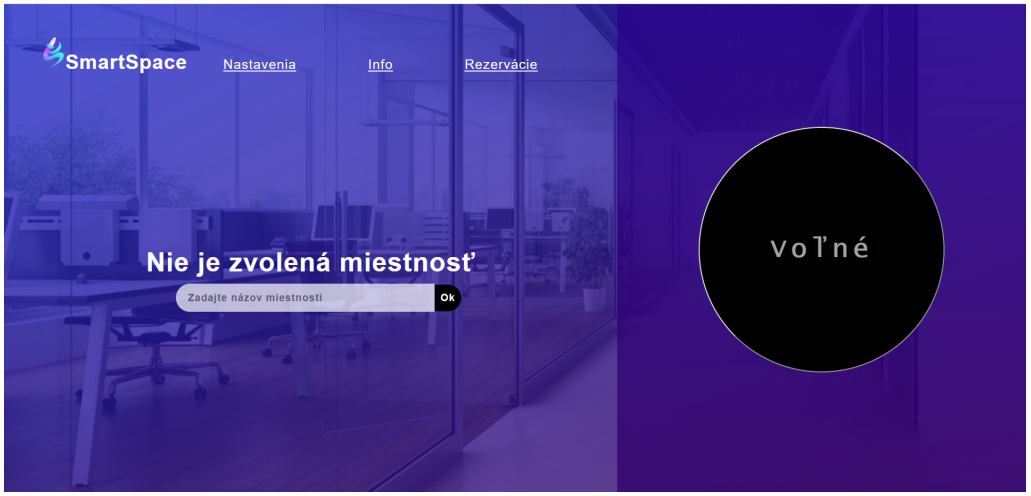
- docker build -t worker .
- docker run –name worker worker

Používateľská príručka

Kiosk

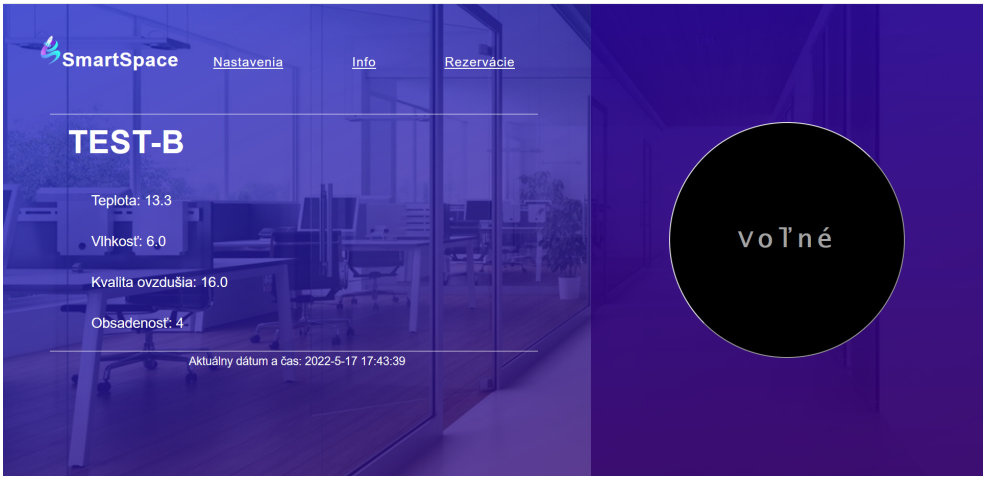
Zariadenie Kiosk, ktoré sa nachádza pred vstupom do miestnosti je určené na zobrazovanie nameraných hodnôt pre konkrétnu miestnosť.

Nastavenia
V prvom rade je potrebné na zariadení zvoliť miestnosť, pre ktorú sa majú potrebné informácie zobrazovať. Toto nastavenie je možné vykonať po kliknutí na podstránku “Nastavenia”.



Obrázok 1: Hlavné menu na zariadení Kiosk

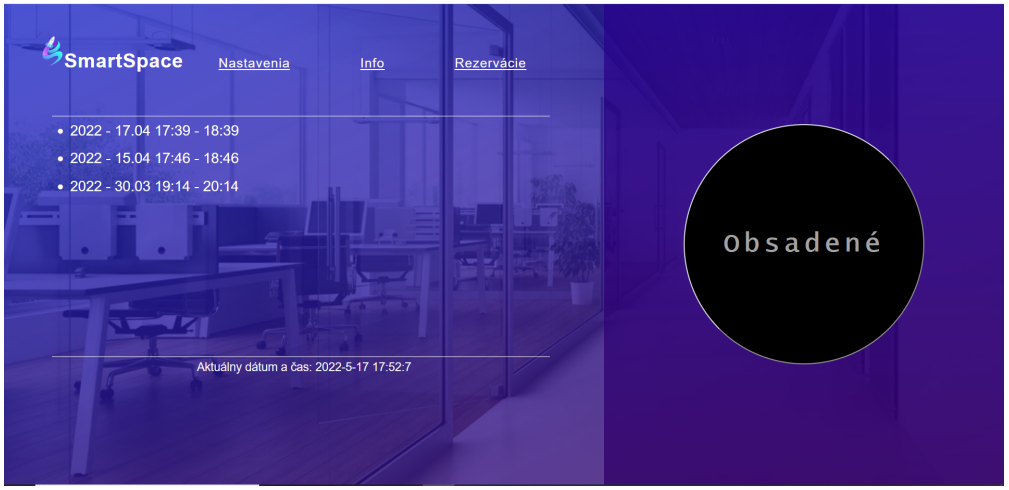
Informácie o miestnosti
Po zvolení danej miestnosti v nastaveniach, je možné zobraziť namerané dáta na podstránke s názvom “Info”, kde je možné vidieť hodnoty ako Teplota, vlhkosť, Kvalita ovzdušia a počet osôb v miestnosti.



Obrázok 2: Zobrazenie nameraných hodnôt v zariadení Kiosk

Rezervácie

Posledná podstránka s názvom ”Rezervácie” je určená na zobrazenie všetkých rezervácii pre danú miestnosť. Na tejto podstránke sú zobrazené rezervácie vzniknuté za pomoci webovej aplikácie ale taktiež aj rezervácie vzniknuté za pomoci samotného zariadenia Kiosk.



Obrázok 3: Zobrazenie rezervácii v zariadení Kiosk

Na tomto zariadení je rezerváciu možné vykonať za pomoci kliknutia na tlačidlo s nápisom “Voľné”, ktoré sa následne zmení na “Obsadené” a vytvorí sa hodinová rezervácia pre danú miestnosť.

Webová aplikácia

Dashboard

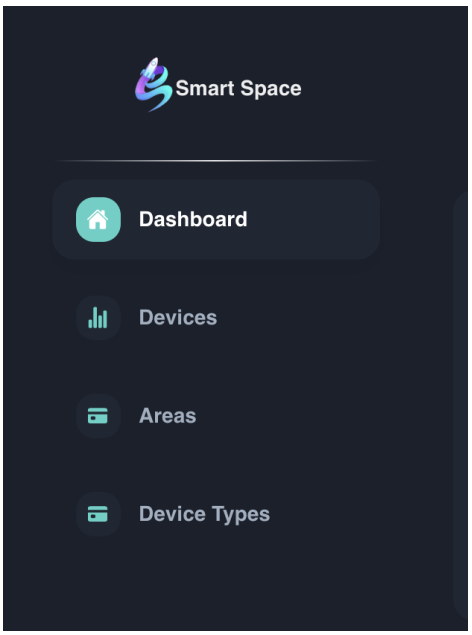
Úvodná stránka s logom našej aplikácie. Na ktorú sa môžme dostať prechodom na url: http://team03-21.studenti.fiit.stuba.sk/dash/#/admin/dashboard
Na stránke sa nachádza menu ktorým sa dostaneme na ďalšie podstránky aplikácie.

Obrázok 4: Dashboard

Menu

Pomocou menu sa vieme dostať nasledujúcich sekcií:

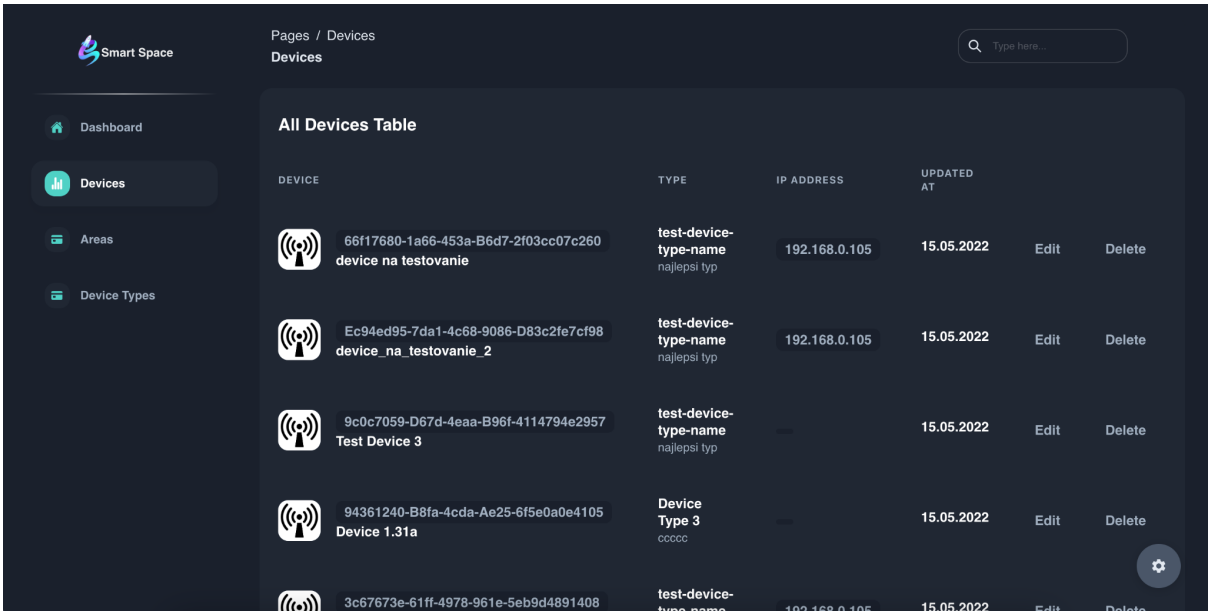
- **Dashboard** (Úvodná stránka s logom našej aplikácie)
- **Devices** (Stránka so zoznamom všetkých zariadení s možnosťou ich úpravy/zmazania)
- **Areas** (Stránka so zoznamom všetkých základných(root) oblastí/miestností a zariadení s možnosťou ich úpravy/vytvorenia/zmazania.)
- **Device Types** (Stránka so zoznamom všetkých typov zariadení s možnosťou ich vytvorenia/úpravy/zmazania)



Obrázok 5: Menu

Device list

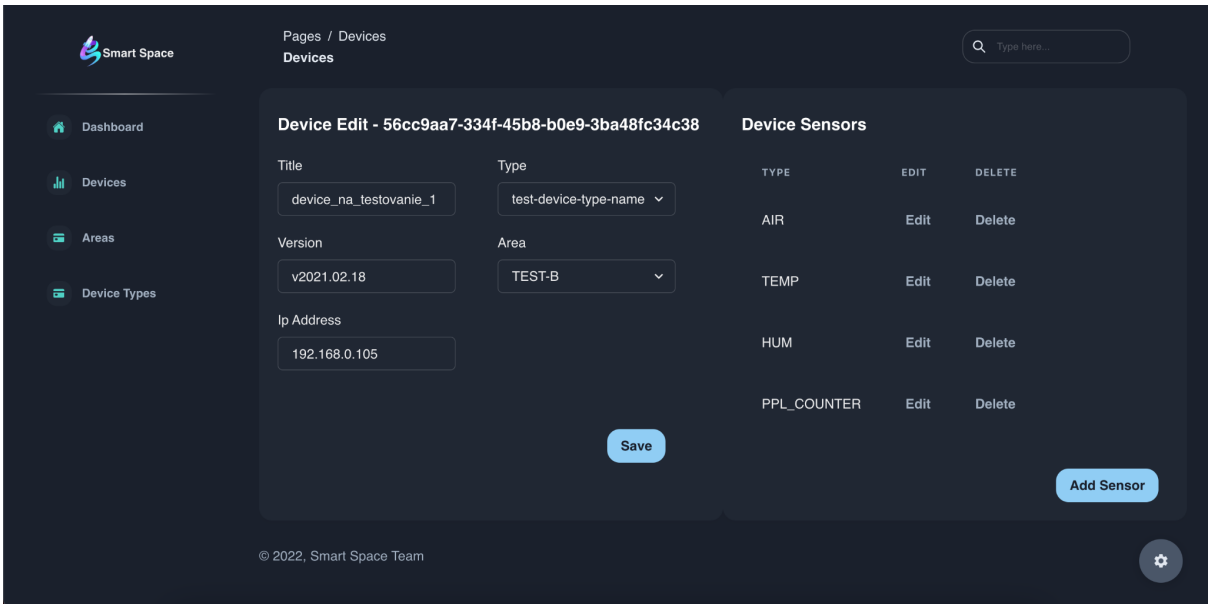
Obsahuje zoznam všetkých zariadení s možnosťou ich úpravy/zmazania. Tlačidlo Edit nás presunie na úpravu zariadenia a tlačidlo Delete zmaže konkrétne zariadenie.



Obrázok 6: Zoznam zariadení

Device update

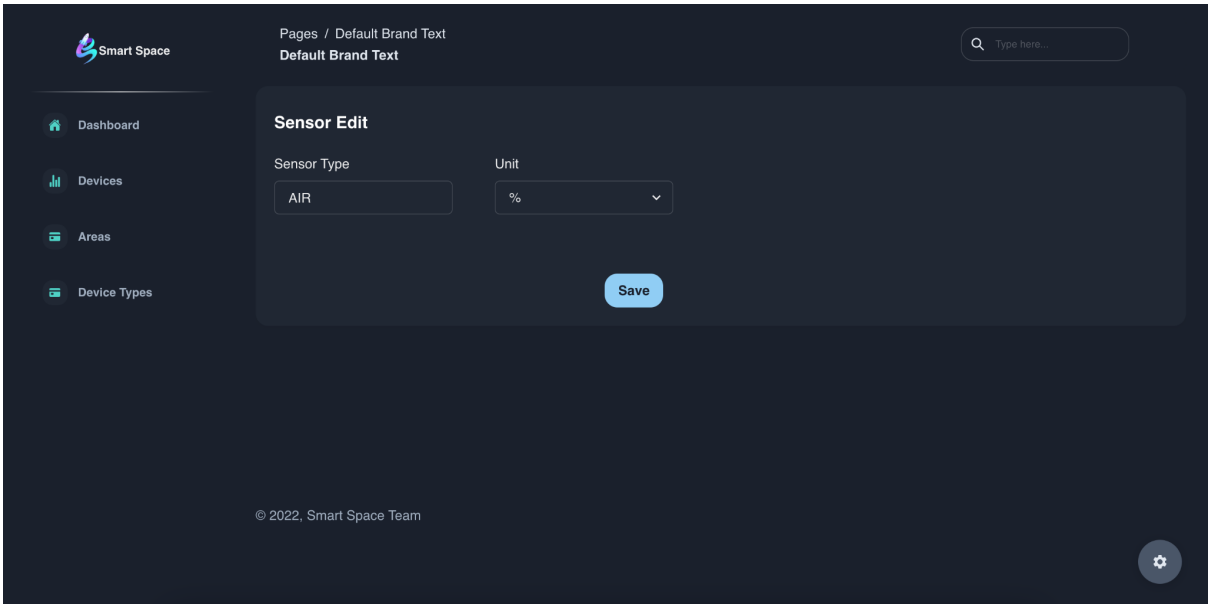
Obsahuje formulár na úpravu existujúceho zariadenia, ktorý sa po vyplnení údajov uloží pomocou tlačidla Save. Taktiež obsahuje zoznam všetkých senzorov, ktoré má zariadenie priradené s možnosťou ich úpravy/vytvorenia/zmazania. Tlačidlo Edit nás presunie na úpravu senzoru a tlačidlo Delete zmaže konkrétny senzor.



Obrázok 7: Edit zariadenia

Sensor Create/Update

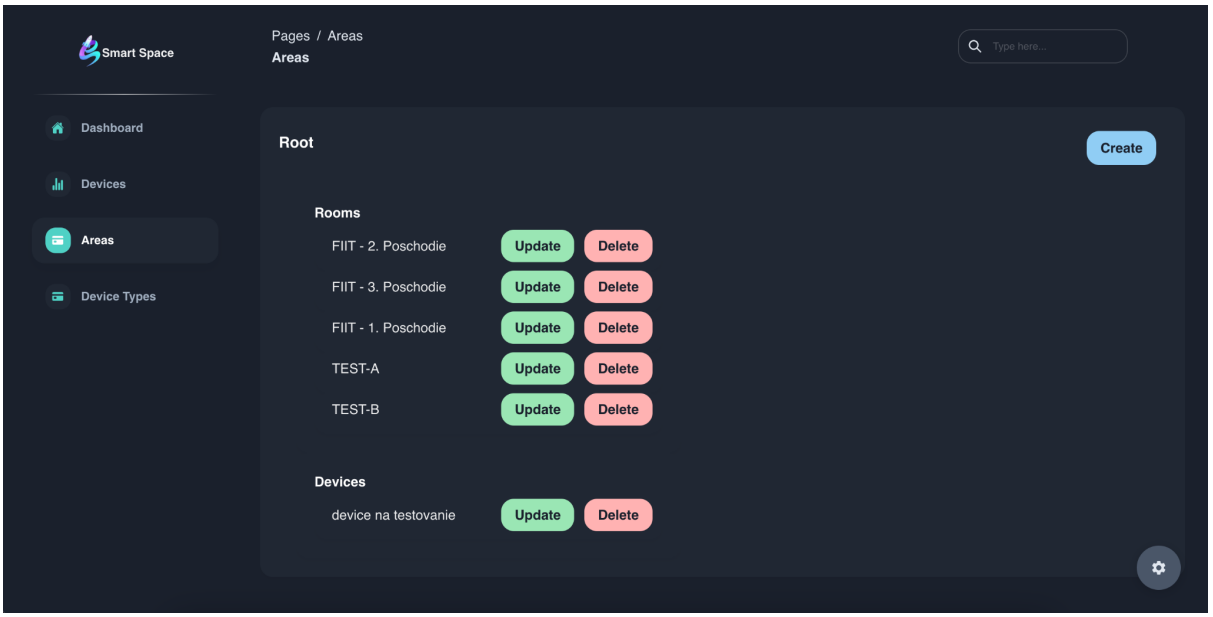
Obsahuje formulár na vytvorenie/úpravu existujúceho senzoru ktorý sa po vyplnení údajov uloží pomocou tlačidla Save.



Obrázok 8: Edit senzoru

Area and Device list

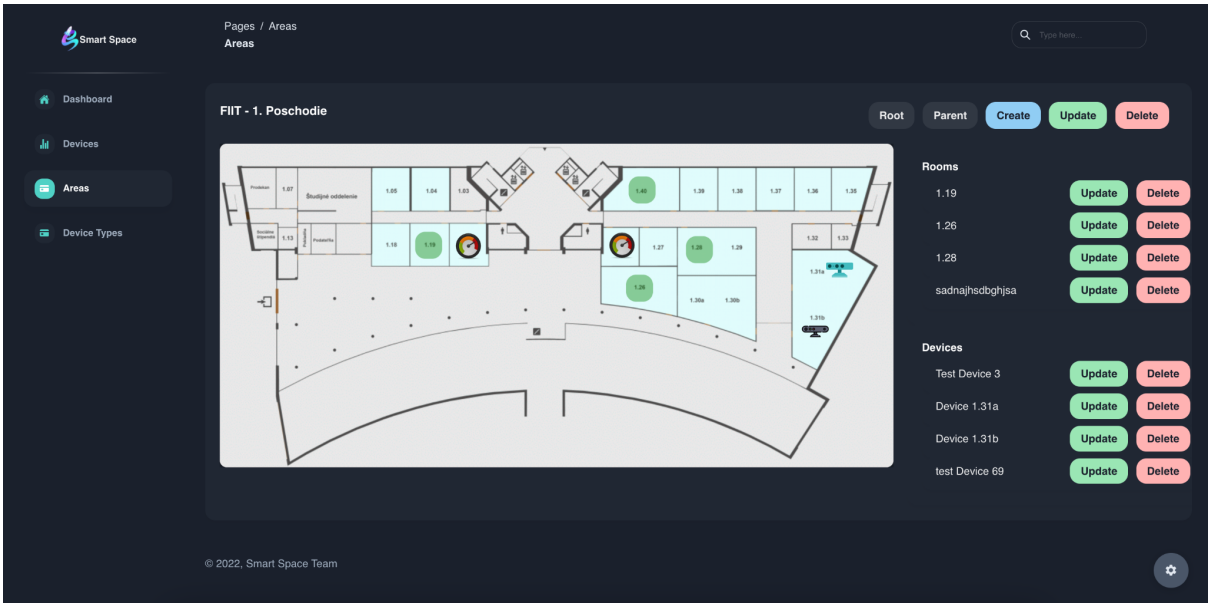
Obsahuje zoznam všetkých základných(root) oblastí/miestností a zariadení s možnosťou ich úpravy/vytvorenia/zmazania. Tlačidlo Create nám zobrazí modal v ktorom máme možnosť vytvoriť oblasť/zariadenie. Tlačidlo Update nás presunie na úpravu oblasti/zariadenia a tlačidlo Delete zmaže konkrétnu oblasť/zariadenie.



Obrázok 9: Zoznam oblastí a zariadení

Area Update

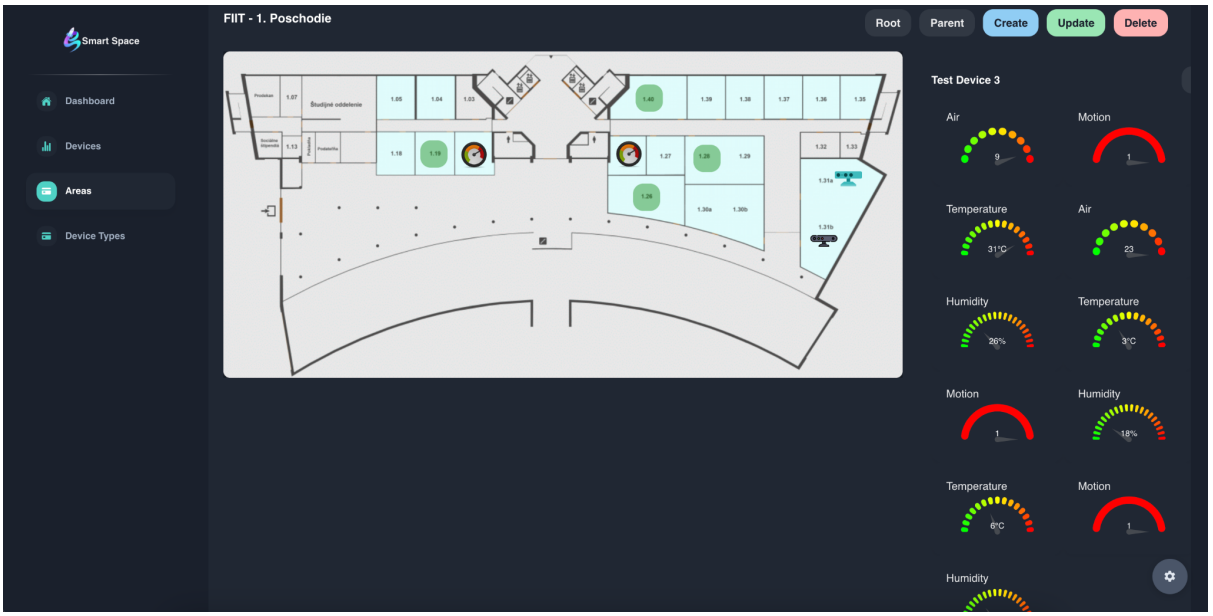
Obsahuje detail oblasti/miestnosti s jej fotografiou a všetkými zariadeniami/oblasťami, ktoré sa v tejto oblasti nachádzajú(takzvané child areas/devices) s možnosťou ich úpravy/vytvorenia/zmazania. Oblasť/zariadenie je možné pridať/upraviť kliknutím na konkrétne miesto pôdorysu oblasti. Polohu oblasti/zariadenie na pôdoryse je možné meniť potiahnutí danej ikonky. Tlačidlo Update nás presunie na úpravu oblasti/zariadenia a tlačidlo Delete zmaže konkrétnu oblasť/zariadenie. Tlačilo Root nás presunie na root (základnú) oblasť a tlačidlo Parent nás presunie na rodiča oblasti/zariadenia v ktorom aktuálne sme. Kliknutím na ikonku zariadenia sa presunieme do jeho detailu so senzormi.



Obrázok 10: Detail oblasti

Area device details

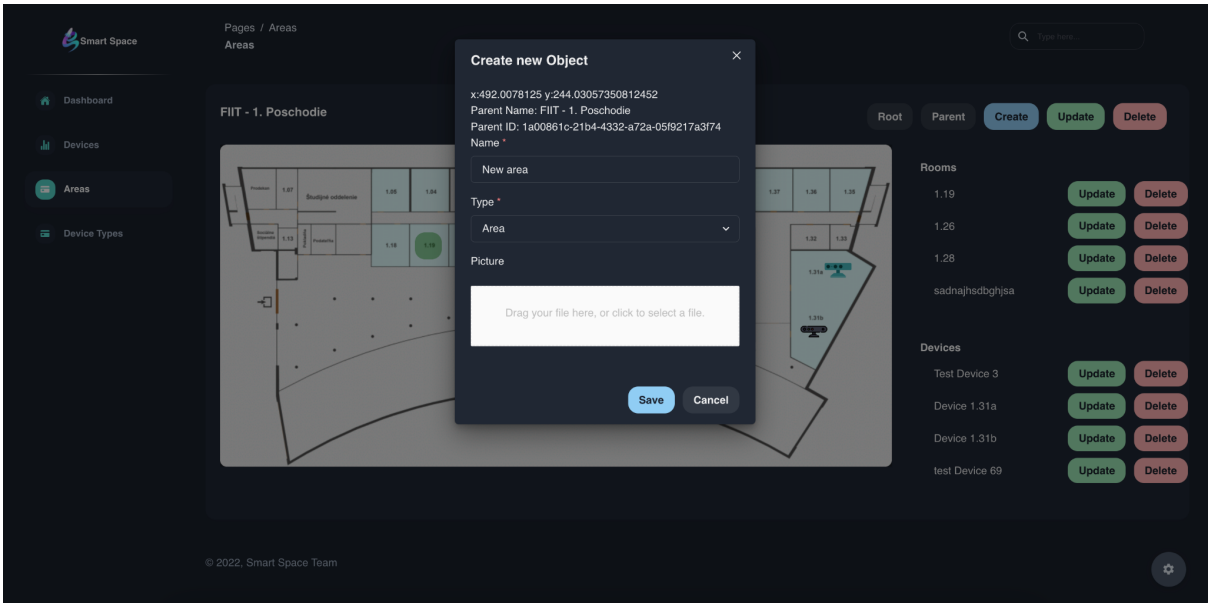
Obsahuje detail zariadenia nachádzajúceho sa v danej oblasti s meraniami zo senzorov. Tlačidlo Update nás presunie na úpravu zariadenia a tlačidlo Delete zmaže konkrétne zariadenie. Tlačilo Root nás presunie na root (základnú) oblasť a tlačidlo Parent nás presunie na rodiča zariadenia v ktorom aktuálne sme. V pravej časti obrazovky môžme vidieť rôzne senzory s ich aktuálnymi hodnotami.



Obrázok 11: Detail senzoru v oblasti

Area child Create/Update

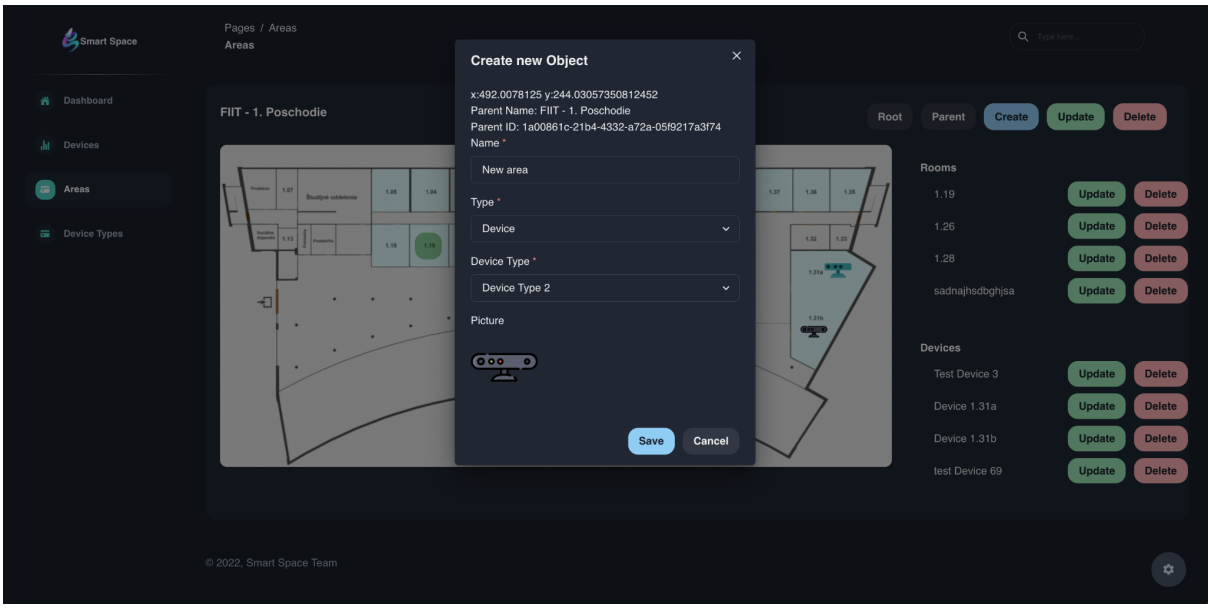
Obsahuje modal, ktorý sa nám zobrazí po ťuknutí na pôdorys na vytvorenie/úpravu oblasti pre konkrétnu otcovskú oblasť. Po vyplnení formulára a nahratí obrázka sa formulár tlačidlom Save uloží.



Obrázok 12: Vytvorenie oblasti

Area Device Create/Update

Obsahuje modal, ktorý sa nám zobrazí po ťuknutí na pôdorys na vytvorenie/úpravu zariadenia pre konkrétnu otcovskú oblasť. Po vyplnení formulára a nahratí obrázka sa formulár tlačidlom Save uloží.



Obrázok 13: Vytvorenie zariadenia

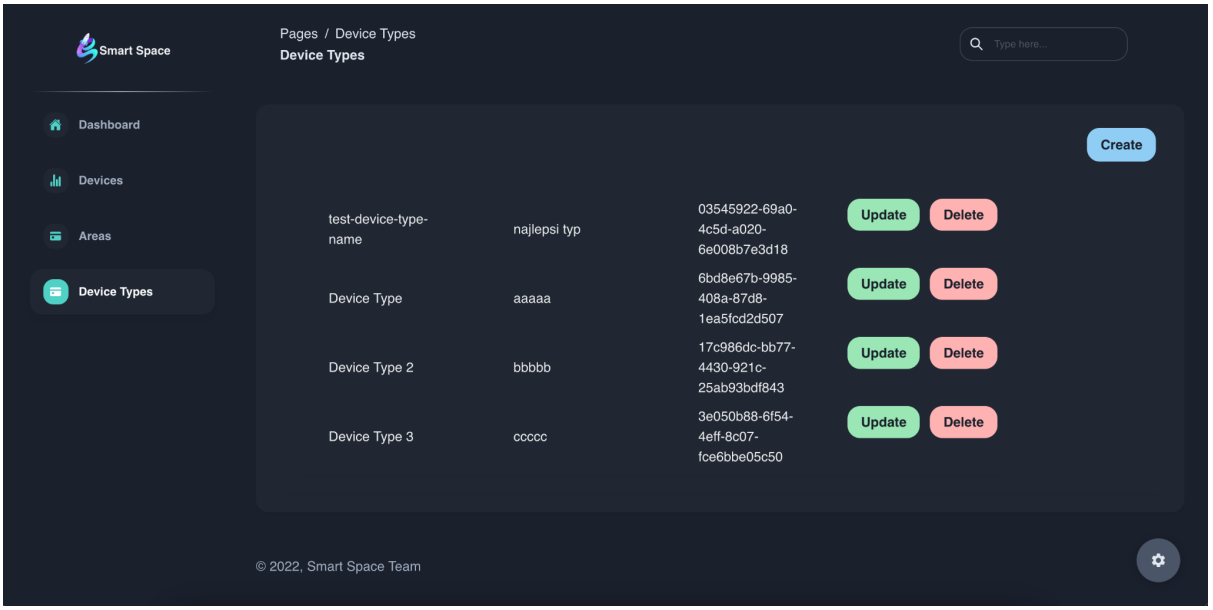
Device detail

Obsahuje merania z priradených senzorov.

Obrázok 14: Detail zariadenia so všetkými senzormi

Device type list

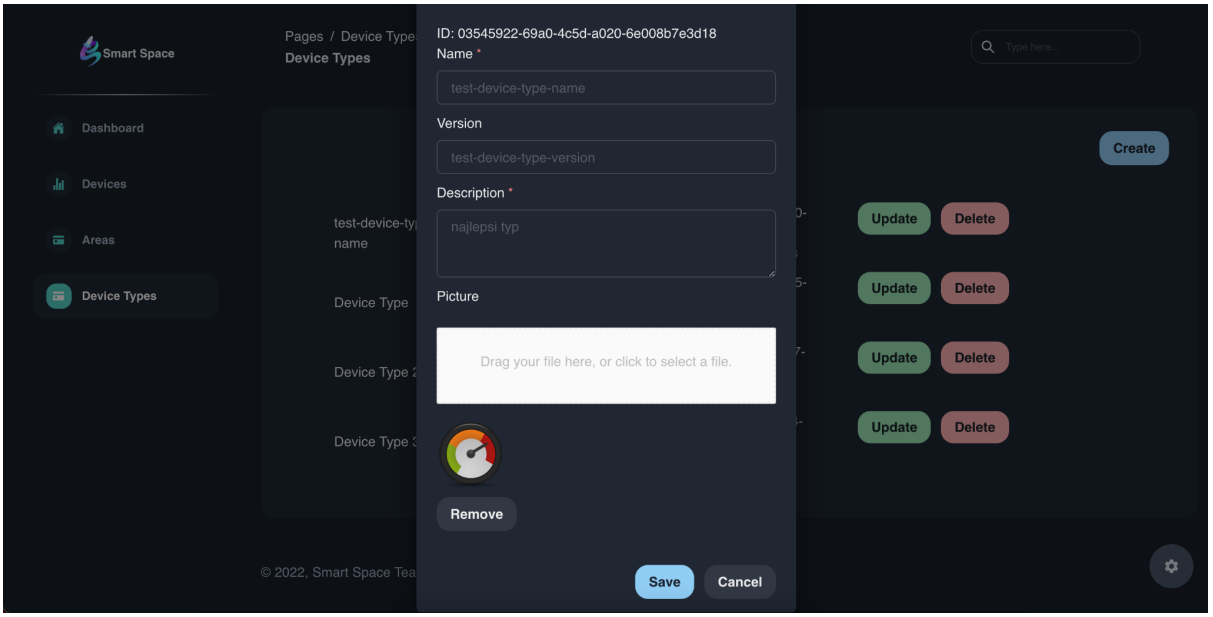
Obsahuje zoznam všetkých typov zariadení s možnosťou ich vytvorenia/úpravy/zmazania. Tlačidlo Update nás presunie na úpravu typu zariadenia a tlačidlo Delete zmaže konkrétny typ zariadenia. Tlačidlo Create nám zobrazí modal s formulárom v ktorom vieme vytvoriť nový typ zariadenia.



Obrázok 15: Zoznam typov zariadení

Device type Create/Update

Obsahuje formulár na vytvorenie/úpravu existujúceho typu zariadenia. Po vyplnení formulára a nahratí obrázka sa formulár tlačidlom Save uloží. Obrázok vieme pomocou tlačidla Remove odstrániť a nahrať nový.



Obrázok 16: Edit typu zariadenia

DocuWiz (tím č. 18): Posudok na prototyp tímu SmartSpace2 (tím č. 3)

Úvod

Tento dokument je výstupom z testovania prototypu od teamu č. 3 SmartSpace2. Testovanie bolo vykonané teamom č. 18 Document Wizard. Hodnotené boli aspekty ako prístup teamu k práci, kvalita vytvoreného prototypu a celkové fungovanie teamu počas práce na projekte.

Hodnotenie prototypu

Cieľom hodnoteného projektu bolo vytvorenie systému na monitorovanie kvality prostredia v kancelárskych a co-workingových priestoroch pomocou prepojenia IoT zariadení. Tieto IoT zariadenia majú za úlohu monitorovať stav v miestnosti pomocou senzorov ako napríklad senzor na meranie teploty, kvality ovzdušia a vlhkosti. Ďalším meraným faktorom je obsadenosť miestností, ktorá je prepojená aj s rezervačným systémom pre obsadenie miesta v konkrétnych miestnostiach pre konkrétny čas.

Prototyp, ktorý sme dostali na testovanie pozostával z viacerých IoT zariadení spolu s pripojenými senzormi. Súčasťou prototypu bola webová stránka, ktorá slúži ako portál pre admina, ktorý v ňom vie vytvárať nové zariadenia, senzory, miestnosti a taktiež vie monitorovať namerané hodnoty. Poslednou súčasťou prototypu je tzv. kiosk, ktorý slúži ako rozhranie pre interakciu s rezervačným systémom priamo v konkrétnej miestosti a taktiež na zobrazenie jej stavu ovzdušia.

Počas testovania backendovej časti aplikácie sme zistili, že niektoré endpointy, ktoré boli zapísané v dokumentácií na získavanie dát nefungujú - buď neexistujú alebo nevracajú žiadne hodnoty. Táto chyba však nemá dopad na zvyšok systému, keďže reálne využívané endpointy fungovali. Ďalšou chybou bolo, že v niektorých prípadoch kiosk nevypočítaval priemerné hodnoty zo všetkých senzorov v danej oblasti. Napríklad v niektorých prípadoch bola priemerná teplota vypočítaná len z troch senzorov namiesto štyroch. Jedným z identifikovaných problémov, je nejednoznačnosť hodnôt zo senzoru na meranie kvality ovzdušia. Zobrazené hodnoty sú na stupnici od 0 do 10 a ako používatelia sme nevedeli určiť čo si pod týmito hodnotami predstaviť. Pri testovaní portálu admina sme odhalili, že je možné vytvoriť zariadenie bez pridelenej ikony. Takýto obrázok následne nie je viditeľný na mape oblasti. Celá komunikácia cez web nie je zabezpečená a prebieha čisto prostredníctvom HTTP. Dalším problémom, ktorý sme identifikovali je zlé a neprehľadné zobrazovanie chýb pri vytváraní a upravovaní senzorov alebo zariadení. Tieto chyby sú vo viacerých prípadoch zobrazované ako JSON a používateľ tak nemusí pochopiť, čo chybu spôsobilo.

Hodnotenie práce tímu

Samotnú prácu tohto tímu sme hodnotili na základe výsledkov v aplikácii Jira a taktiež aj na základe dostupných zápisníc zo šprintov, ktoré si tento tím vytváral každý týždeň. Zo

zápisníc, ktoré sú dostupné aj na stránke tímu bolo vidieť, že všetci členovia sa pravidelne zúčastňovali každého stretnutia, na ktorých analyzovali priebeh samotného šprintu, vzniknuté problémy a taktiež si definovali úlohy na ďalší šprint. Na základe výstupov v aplikácii Jira sme si všimli že v tíme mali úlohy rozdelené približne rovnomerne a vo väčšine prípadoch sa im podarilo úlohy plniť aj v dohodnutom termíne aj keď niektoré úloha sa medzi šprintmi prenášali. Podľa vygenerovaných grafov je vidieť že tím mal počas celého semestra konzistentné rozloženie úloh.

Celkové hodnotenie

Výslednú prácu tímu č.3 (SmartSpace2) hodnotíme veľmi pozitívne a to hlavne z dôvodu kvalitne organizovanej práce a dosiahnutým veľmi dobrým výsledkom. Tímu sa podarilo vytvoriť “inteligentnú miestnosť”, ktorá disponuje rôznymi vychytávkami ako je analýza stavu prostredia za pomoci viacerých senzorov, možnosť rezervácie si tejto miestnosti cez internet alebo za pomoci tabletu nasadeného pred vstupom do miestnosti (na tomto tablete majú taktiež zobrazené dáta ako napríklad teplota v miestnosti, kvalita ovzdušia, vlhkosť a počet osôb v miestnosti). Čo nás zaujalo je, že za pomoci administrátorského rozhrania je možné túto miestnosť veľmi jednoducho rozšíriť o ďalšie senzory a taktiež je možné vytvoriť aj ďalšie inteligentné miestnosti v budove. Počas testovania sme narazili na menšie problémy, na ktoré nás ale tento tím vopred upozornil a spísali sme ich do časti “Hodnotenie prototypu”.