Gymnázium Arabská, Praha 6, Arabská 14
Obor programování

Weathery

Filip Hruška, Filip Růžička, Ondřej Salát

Duben, 2023

Prohlašujeme, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů udělujeme bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V ..... dne .....

Filip Hruška .....

Ondřej Salát .....

Filip Růžička .....

Abstrakt

Tato práce popisuje tvorbu a výsledek ročníkového projektu na téma vytvoření stanice na měření aktuálního počasí.

Obsah

1 Úvod

Cílem tohoto projektu bylo vytvořit měřící stanici na měření aktualního počasí a následné zobrazení dat pomocí grafů na webové stránce.

1.1 Výběr projektu

Toto téma jsme si vybrali, abychom získali zkušenosti s vytvářením projektu, který obsahuje propojení hardwaru se sofwarem, konkrétně propojení stanice s backendem a následné zobrazení dat na webové stránce.

Prvoplánově jsme zamýšleli vytvořit web pro inzerci produktů, ale tento nápad se nám nezdál ideální, neboť neobsahoval programátorsky zajímavé prvky a nezdál se nám příliš tvůrčí.

1.2 Původní zadání

Aplikace bude mít tři části

1. operace a vyrobení hardwaru, který bude měřit počasí (teplota, vlhkost, rychlost a směr větru, déšť) hardware bude obsluhovat Arduino, přičemž si čidla vyrobíme sami (rychlost a směr větru, měření deště)
2. backend - sbírá data o počasí a ukládá je do databáze, backend bude fungovat i jako API pro jine vývojáře na zjištění aktuálního počasí
3. frontend - webová aplikace zobrazující počasí (mapka, na které budou vidět všechny stanice a bude se dát zobrazit podrobnější data o dané stanici).

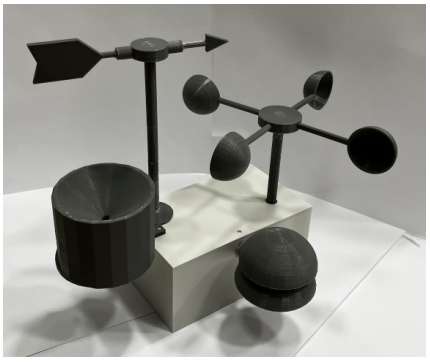
2 Hardware

Při vytváření stanice bylo nejprve potřeba zvolit vhodný micropočítač pro připojení měřicích komponentů. Nakonec jsme vybrali platformu Arduino, konkrétně model ESP8266WiFi, který nám umožnil nejen zapojit všechny různé komponenty potřebné pro měření, ale i připojení k internetu.

Komponenty potřebné k sestavení stanice byly vytisknuty na 3D tiskárně. Modely potřebné pro tisk byly vytvořeny v programu TinkerCad (model pro měření deště byl použit opensource model [1]). Samotná krabička pro micropočítač byla vyrobena z plastových desek, které byly vytvořeny na CNC frézce.

Stanice při spuštění zavolá funkci setup, která zinicializuje proměnné potřebné pro běh programu. Dále stanice měří počasí a po každých pěti minutách stanice pošle data na server, kde budou dále zpracovány.

Obrázek 1: Stanice



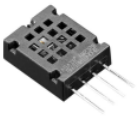
Obrázek 2: Stanice

2.1 Propojení se serverem

Po zapnutí se stanice připojí k internetu za pomoci WiFi a zaregistruje na backendu (popsáno v kapitole 3.2). Jakmile stanice dostane od serveru odpověď s JSON Web Tokenem, uloží si ho a při každém odeslání dat na server ho použije k prokázání totožnosti.

2.2 Měření teploty a vlhkosti

Pro měření teploty a vlhkosti byl použit Digitální teploměr Adafruit AM2320 Senzor[2]. V tomto případě nebylo zapotřebí vymýšlet jak cokoliv měřit, protože senzor je připojen přes I2C interface, ze kterého program přímo čte již naměřené hodnoty.



Obrázek 3: Teploměr a vlhkoměr

2.3 Měření tlaku

Měření tlaku je zhruba stejně přímočaré jako měření teploty, což znamená, že se o měření stará senzor[3], který je zapojen přes I2C interface, ze kterého program čte hodnotu tlaku v jednotce Pa.

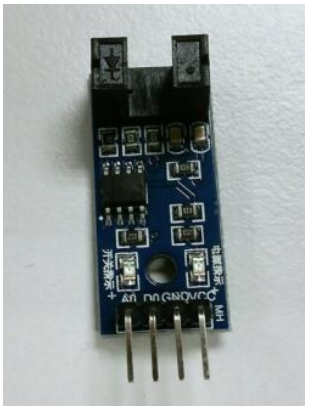
Obrázek 4: Tlakoměr

2.4 Měření rychlosti větru

V případě měření rychlosti větru už nestačilo jen koupit senzor, ale bylo potřeba vymyslet jak převést momentum otáčející se osy s lopatkami na reálnou rychlost. Tento problém je vyřešen následujícím způsobem. Na ose je nasazen disk kruhovitého tvaru, který má v sobě dírky, skrz které optický senzor vysílá paprsek. Pokud se paprsek přeruší, znamená to, že se mechanismus otočil o jednu dírku. Program tedy počítá kolikrát se paprsek přeruší a spočítá skutečnou rychlost větru pomocí následující rovnice.

$$v = n * d/CasMereni \tag{1}$$

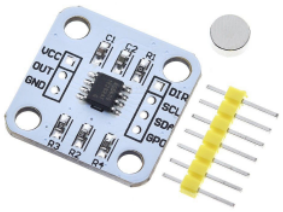
Kde v = rychlost větru, n = počet přerušení, d = vzdálenost, kterou vítr urazí za jedno přerušení.



Obrázek 5: Optická závora

2.5 Měření směru větru

Měření směru větru je realizováno pomocí magnetického rotačního senzoru, který dokáže měřit odchylku magnetického pole. To bylo využito na měření odchylky větru tak, že na ose, na které je přidělaná střelka, kterou vítr otáčí, je zespod přidělán malý magnet a senzor tedy měří jeho odchylku.

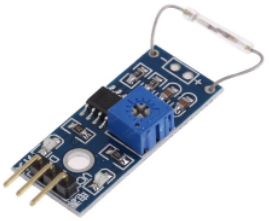


Obrázek 6: Magnetický rotační senzor

2.6 Měření deště

Při vytváření součástky na měření deště vzniklo hned několik nápadů jak déšť měřit. Nejprve vznikl nápad, že by skrz trychtýř sbírající vodu prokapávala voda a optický senzor by počítal kapky a tím by měřil déšť. Tento způsob by však nefungoval, protože optický senzor by nedokázal rozpoznat padající kapku.

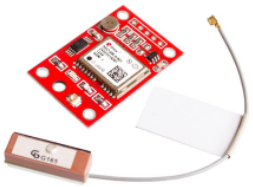
Po delším hledání na internetu jsem však našel řešení[1], které by tento problém vyřešilo a nebyl by potřeba žádný optický senzor. Toto řešení spočívá v tom, že voda kape do malé kolébky, která se v jeden moment překlopí a voda kape do druhé strany. Tímto překlopením kolébka zaktivuje senzor rozpoznání magnetického pole, protože na kolébce je přidělán malý magnet.



Obrázek 7: Senzor magnetického pole

2.7 Měření GPS

Z důvodu, že stanice potřebuje mít informace o své poloze, je ve stanici GPS modul, který získává aktuální GPS souřadnice. Tento senzor má být zapojen na RX/TX piny, ale z důvodu, že na naší arduino desce už nezbývali volné RX/TX piny, GPS modul musel být zapojen na jiné dva piny a data se z pinů čtou softwarově pomocí knihovny SoftwareSerial. S tímto řešením také ale nastaly první problémy, protože program začal náhodně padat a dlouhou jsme nemohli přijít na příčinu. Po delším zkoumání jsme přišeli na to, že knihovna SoftwareSerial zapisuje data do paměti bez ohledu na to, zda je paměť volná, a tudíž začala přepisovat data programu, což způsobilo padání programu. Tento problém je dosud nevyřešen, protože už nebyl čas na vytvoření vlastní knihovny, která by tento problém řešila. Další možným řešením by bylo použít jinou desku, která by měla více RX/TX pinů, avšak toto řešení už také nebylo možné.



Obrázek 8: GPS modul

2.8 Prostor pro zlepšení

Největším nedostatek je momentálně neúplná funkce GPS modulu, což by do budoucna chtělo nějakým způsobem vyřešit. Další možnost pro vylepšení by mohlo být přidání kompasu, protože v tento moment je potřeba natočit stanici správným směrem, aby měřila směr větru přesně.

3 Backend

Backend, neboli serverová část aplikace, našeho projektu slouží primárně k shromažďování a následnému zpracovávání dat. Naše API dále umožňuje bezplatný přístup k těmto datům. Bac-

kend byl vyvinut v Pythonu pomocí knihovny FastApi a pro ukládání dat byla použita databáze PostgreSQL. Pro odesílání e-mailů byla použita knihovna smtplib a pro komunikaci s platební branou byla použita knihovna stripe.

Backend může být abstrahován do tří klíčových částí, z nichž každá má specifické úkoly:

- zpřístupnění dat pro Frontend/Developery (pomocí API)
- zajištění komunikace mezi stanicí a databází
- spravování objednávek, objednané prostřednictvím webové stránky (více popsáno v kapitole 4)

3.1 Zpřístupnění dat

O zpřístupnění naměřených dat se starají veřejné endpointy, které může kdokoli bezplatně použít, ať už pomocí naší webové stránky, či pomocí našeho API. V případě použití těchto endpointů, není potřeba žádná autorizace a uživateli vrátí vyžádaná data ve formátu JSON. Mezi tyto veřejné endpointy patří:

- GET /stations
- GET /now/gps
- GET /stats/gps

První endpoint (/stations) jednoduše vrátí GPS souřadnice všech stanic, které jsou v ten moment zapojené. Za tímto endpointem se schovává v podstatě jeden SQL dotaz, který z databáze vytáhne všechny aktivní stanice a nakonec funkce vrátí JSON.

Listing 1: Příklad požadavku /stations

```
{
  "message": "ok",
  "stations": [
    {
      "gps": "50.0993194_14.3596525"
    },
    {
      "gps": "49.7454400_14.0578025"
    }
  ]
}
```

Druhý endpoint (/now/gps) bere jako parametr GPS souřadnice stanice a podle nich z databáze vytáhne poslední naměřená data dané stanice. Za tímto endpointem se opět schovává v podstatě jen jeden SQL dotaz, který z databáze vytáhne poslední naměřené hodnoty konkrétní stanice a nakonec funkce vrátí JSON.

Listing 2: Příklad požadavku /now/gps

```
{
  "message": "ok",
  "time": "24-12-2022 4:20:00",
  "temperature": -3,
  "humidity": 43,
  "pressure": 100000,
  "wind_speed": 13,
  "wind_direction": "N",
  "rain": 4
}
```

Poslední endpoint (/stats/gps) má za úkol vracet dlouhodobá zprůměrovaná data v časovém rozmezí, které je určeno pomocí query parametrů, to znamená ”date_from”, ”date_to”a nepovinný parametry ”freq”, který pevně udává frekvenci průměrování dat. Data se průměrují vzhledem k velikosti časového intervalu. Data v časovém rozmezí, které je menší než tři dny se průměrují po hodině. Data v časovém rozmezí, které je menší než tři měsíce se průměrují po dnech. Data v rozmezí, které je menší než rok se průměrují po týdnech a pokud je časové rozmezí větší než rok průměr se dělá z dvou týdnů. Pokud je potřeba průměrovat bez ohledu na velikost časového rozmezí, stačí nastavit hodnotu parametru query ”freq”a data se budou průměrovat vzhledem k hodnotě ”freq”následovně:

- 1 -> data se průměrují po 5 minutách
- 2 -> data se průměrují po 30 minutách
- 3 -> data se průměrují po hodině
- 4 -> data se průměrují po dnech
- 5 -> data se průměrují po týdnech
- 6 -> data se průměrují po dvou týdnech
- 7 -> data se průměrují po čtyřech týdnech

O samotném průměrování hodnot se stará samotná databáze pomocí funkce AVG. Celý časový interval se nejprve rozdělí po časových úsecích, které se buď automaticky, nebo podle již zmiňované hodnoty ”freq”. Následně se provede SQL dotaz do databáze, který vytáhne zprůměrované hodnoty pro každý časový úsek. Nakonec funkce vrátí tyto hodnoty ve formátu JSON.

Listing 3: Příklad požadavku /stats/gps

```
{
  "message": "ok",
  "data": [
    {
      "time": "1-12-2022 00:00:00",
      "temperature": -3.6,
      "humidity": 41,
      "pressure": 100200,
      "wind_speed": 13,
      "wind_direction": "N",
      "rain": 4,
      "avrage_of": 288
    },
    {
      "time": "2-12-2022 00:00:00",
      "temperature": -2.2,
      "humidity": 49,
      "pressure": 100100,
      "wind_speed": 4,
      "wind_direction": "S",
      "rain": 0,
      "avrage_of": 288
    }
  ]
}
```

3.2 Komunikace se stanicí

Pro komunikaci stanice se serverem stanice využívá dva endpointy:

- POST /station/register
- POST /station/update

V případě prvního endpointu (/station/register) jde primárně o autorizace stanice, aby data na server nemohl posílat neautorizovaný uživatel. Tento POST požadavek očekává v parametru sériové číslo stanice a jeho GPS souřadnice. Backend nejprve ověří, zda sériové číslo odpovídá registrovaným stanicím a pokud ano, tak vytvoří JSON Web Token (dále jen JWT), kterým se stanice dále prokazuje pří dalších interakcích se serverem. Dále si stanici uloží do databáze do seznamu aktivních stanic.

Druhý endpoint (/station/update) slouží k přidání naměřených dat stanicí do databáze. V tomto případě server očekává v hlavičce požadavku již zmiňovaný JWT, podle kterého server pozná, o jakou jde stanici a naměřená data, která stanice odeslala ve formátu JSON, zapíše do databáze.

3.3 Správa objednávek

Celý náš objednávkový systém slouží k tomu, aby si kdokoliv mohl zakoupit naší stanici, čím zajistí, že bude mít možnost měřit počasí kdekoliv bude potřebovat. Při vytváření objednávkového systému bylo potřeba nejprve vybrat nějakou platební bránu, která by nám umožnila přijímat platby od zákazníků. V našem případě jsme vybrali platební bránu Stripe.

Pro spravování objednávek backend používá 5 endpointů

- POST /payment-webhook
- POST /login
- GET /orders
- GET /order/id
- POST /update/id

První endpoint (/payment-webhook) funguje jako webhook[1], který se zavolá v moment, kdy platební brána vyhodnotí, že objednávka byla zaplacena. V moment kdy se zavolá webhook, do databáze se přidá nová objednávka. V ten moment se také odešle zákazníkovi e-mail o tom, že objednávka byla přijata. Zároveň příjde e-mail i adminům s upozorněním, že byla vytvořena nová objednávka.

Endpoint (/login) slouží k autorizaci adminů. Admin se pomocí webové stránky může přihlásit a spravovat jednotlivé objednávky.

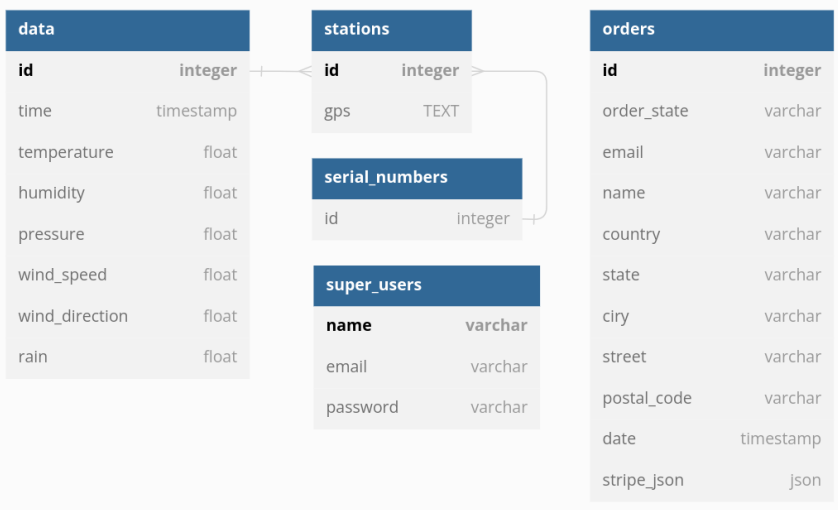
Endpoitn (/orders) je endpoint, který vrací ID všech objednávek. Tento endpoint je omezen pouze pro adminy.

Endpoitn (/order/id) vrací informace konkrétní objednávky.

Poslední endpoint (/update/id) slouží opět jen pro adminy a slouží k upravování stavu objednávky.

3.4 Databáze

Backend používa již zmiňovanou databázi PostgreSQL, ve které je pět tabulek zobrazených na obrázku č.9.



Obrázek 9: Model databáze

4 Frontend

Frontend slouží jako nástroj pro vizualizaci aplikace. Patří sem veškeré soubory pro vykreslování stránek, obrázky, mapa a grafy. Uživatel může jednoduše přepínat mezi stránky aplikace, kterými jsou: domovská stránka, mapa a údaje o vývojářích.

[1]Webhook je způsob, jakým se automaticky přenášejí data mezi webovými aplikacemi. Pokud se v jedné aplikaci stane určitá událost, webhook ji zachytí a přenese do druhé aplikace.

Frontend aplikace obsahuje soubory vue.js, což je javascriptový framework, který se stará o dynamické a reaktivní zobrazení uživatelského rozhraní. Vue.js je velmi populární a používaná technologie i velkými firmami, která funguje na bázi jednoduchých šablon a komponent. Vue.js je nováčkem na trhu, ale disponuje rychlostí a velkými možnostmi oproti ostatním frameworkům. Tato technologie usnadňuje uživatelům jednoduché použití a přehlednou dokumentací. Kvůli rozsáhlému používání jsme usoudili, že vue.js je správná volba.

Pro odbavování HTTP požadavků jsme použili knihovnu axios, která propojuje backend s frontendem na základě jejich požadavkům. Disponuje rychlostí, jednoduchým odbavováním a odesíláním požadavků a propracovanou dokumentací, která je základem pro správné použití. Jde o technologii používanou všemi úrovni vývojářů. Aplikace umožňuje odesílání asynchroních požadavků, transformací dat a správou případných chyb. Za pomocí čtení url stránky posílá axios odpověď. Tímto způsobem se dostanou data z databáze až do grafů na frontendu. Pro vykreslení mapy s body jsme zvolili knihovnu leaflet, která je podrobně popsána v kapitole mapa stanic. Pro zpracování grafů jsme zvolili knihovnu chart.js, která je popsána v kapitole grafy.

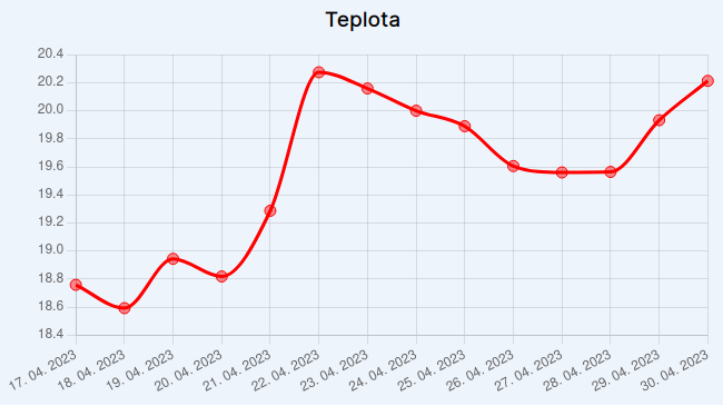
4.1 Grafy

Pro přehledné vykreslení dat jsme zvolili grafy. Na stránce je vykresleno pět grafů podle jejich typu(teplota, déšť, rychlost větru, směr větru, tlak).

Na ose x jsou zobrazeny časové údaje a ose y hodnoty pro daný čas. Použité grafy jsou z opensource knihovny charts.js, která umožňuje přehledně a jednoduše grafy nastylizovat a upravovat dle potřeb.

Uživatel má možnost měnit časový úsek vykreslení dat pomocí tlačítka pod souřadnicemi stanice. Na výběr má z předdefinovaných hodnot (např. den, týden, rok), ale pomocí výběru "vlastní"si může nastavit své vlastní časové rozmezí. Pro rozpoznání grafů je nad každým grafem jméno, které určuje, co je v daném grafu ukázáno. Grafem je křivka, která plynule propojuje dva sousední body.

Knihovna chart.js nebyla naší první volbou, jelikož jsme v předchozích verzích používali ApexCharts.js, ale knihovna nám nevyhovovala v možnostech vykreslení grafů, nekompletní dokumentací a její intuitivností ve smyslu implementování. Z těchto důvodů jsme knihovnu nahradili charts.js, která má o poznání přehlednější dokumentaci a obsahuje více potřebných funkcí.



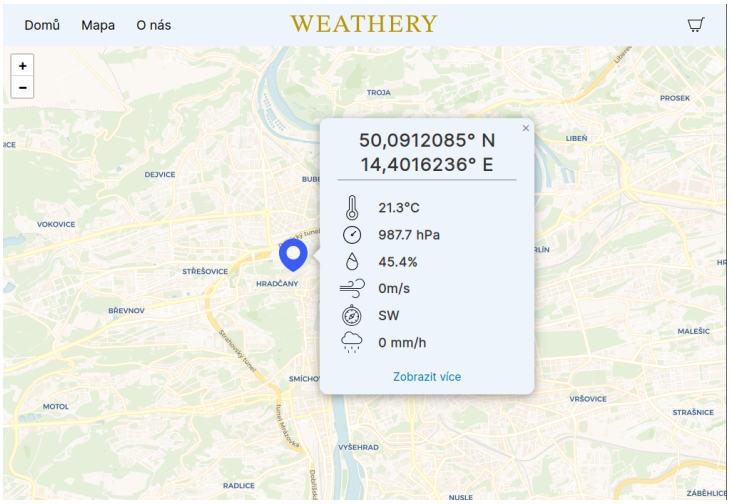
Obrázek 10: Graf s teplotami

4.2 Mapa stanic

Aby si mohl uživatel zobrazit data, musí se dostat na stránku s mapou, ve které je vidět bod. Po kliknutí na bod se dozví poslední naměřené údaje a může se prokliknout pomocí odkazu na

podrobnější údaje v podobě grafů.

Pro vykreslení bodů do mapy jsme použili opensource knihovnu leaflet.js, která disponuje velkou škálou možností pro vykreslení bodů, množin a tvarů. Avšak vybrání této knihovny také nebylo jednoduché. Prvoplánově jsme zamýšleli použít známé mapy jako jsou google maps nebo apple maps, ale implementace těchto knihoven byla pro naše účely zbytečně složitá. Dále tyto knihovny obsahují zdlouhavě a neintuitivní dokumentace, což je při práci s nimi nepříjemné. Knihovna leaflet je populární a rozsáhle používána pro mobilní aplikace. Použití této mapy lze pozorovat například u dopravce české dráhy, sociální sítě Facebook a flickr.



Obrázek 11: Mapa stanic

Pro nastavení mapy jsme zvolili světlé pozadí s málo kontrastním vykreslením vodních ploch, obytných oblastí, silnic, přírodních míst a hranic států. Uživatel má možnost libovolně mapu oddalovat a přibližovat, přičemž výchozím bodem mapy je Praha, na kterou je mapa po spuštění zaměřená.

4.3 Design

Design znamená vizuální podoba stránky. Naším cílem bylo vytvořit intuitivní prostředí pro uživatele, ve kterém se dokáže rychle a jednoduše pohybovat.

Prvním setkáním s designem webu je domovská stránka, která obsahuje funkci detekující kolečko myši. Právě tímto pohybem se stránka odhaluje a postupně mění. V průběhu animace se uživatel dozví informace o dlouhodobějších změnách počasí a srovnání počasí napříč roky.

Na stránce nákup se uživatel dozví podobu stanice a je obeznámen s registrací a podrobnostmi.

Obrázek 12: Nákup stanice

5 Závěr

Na závěr bychom chtěli zhodnotit projekt weathery. Původní zadání jsme splnili a s výsledkem jsme spokojeni, přičemž jsme přidali další funkce, které projekt rozšiřují. Práce na projektu nás, členy týmu, obohatila o zkušenosti pracování v týmu na jednom projektu, rozvržení práce, komunikace a zajištění prostředků na sdílení práce.

Například jsme k projektu přidali funkci, která umožňuje uživatelům nákup stanice a následné zařazení do databáze, aby se stanice ukazovala na mapě. na tuto funkci se váže admin prostředí, přes které je možné vidět jednotlivé objednávky a jejich stav.

Odkazy

[1] *Měření deště*. URL: https://www.thingiverse.com/thing:3779456.

[2] *Teploměr a vlhkoměr*. URL: https://www.adafruit.com/product/3721.

[3] *Tlakoměr*. URL: https://www.adafruit.com/product/2651.

Seznam obrázků