



Vyšší odborná škola
a Střední průmyslová škola elektrotechnická
Plzeň, Koterovská 85

DLOUHODOBÁ MATURITNÍ PRÁCE S OBHAJOBOU

Téma: *Aplikace pro chytrou domácnost*

Autor práce:

Adam Míka

Třída:

4. H

Vedoucí práce:

Mgr. Jakub LINDAUER

Dne:

30. 4. 2022

Hodnocení:



Vyšší odborná škola
a Střední průmyslová škola elektrotechnická
Plzeň, Koterovská 85

Zadání dlouhodobé maturitní práce

Žák: Adam MÍKA
Třída: 4. H
Studiální obor: 18-20-M/01 Informační technologie
Zaměření: Správa počítačových sítí
Školní rok: 2021–2022

Téma práce: ***Aplikace pro chytrou domácnost***

Pokyny k obsahu a rozsahu práce:

- ❖ Teoretické seznámení s IoT prvky
- ❖ Naprogramování IoT prvků
- ❖ Návrh a design responzivní aplikace
- ❖ Programování aplikace

Určení částí tématu zpracovávaných jednotlivými žáky:

Požadavek na počet vyhotovení maturitní práce: 2 výtisky

Termín odevzdání: **22. dubna 2022**

Čas obhajoby: **15 minut**

Vedoucí práce: **Mgr. Jakub LINDAUER**

Projednáno v **katedře VTT** a schváleno ředitelkou školy.

V Plzni dne: 30. října 2021

Ing. Naděžda Mauleová, MBA, v.r.
ředitelka školy

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Plzni dne:

Podpis:

Obsah

1 Teoretické seznámení s IoT prvky	7
1.1 Zařízení na měření vlhkosti a teploty	7
1.1.1 WeMos D1 R2	7
1.1.2 DHT 11	7
1.2 Zařízení Lampa.....	7
1.2.1 WeMos D1 mini ESP32.....	7
1.2.2 RGB LED dioda	8
2 Programování IoT prvků	9
2.1 Programovací platforma	9
2.1.1 Arduino IDE	9
2.2 Princip posílání dat.....	9
2.2.1 MQTT protokol	9
2.2.2 Princip posílání dat	10
2.2.3 Formát poslaných a přijatých dat.....	10
2.3 Amazon Web Services IoT	11
2.3.1 Informace o AWS IoT	11
2.3.2 Zaregistrování IoT zařízení.....	11
2.4 Programování zařízení	13
2.4.1 Programovací jazyk Arduina	13
2.4.2 Zařízení na měření teploty a vlhkosti	13
2.4.3 Zařízení Lampa.....	14
3 Návrh a design responzivní aplikace.....	16
3.1 Autentifikační formuláře.....	16
3.1.1 Přihlášení	16
3.1.2 Registrace	16
3.1.3 Zapomenuté heslo.....	16
3.2 Hlavní stránka	17
3.2.1 Obrázek.....	17
3.2.2 Design Zařízení na měření teploty a vlhkosti.....	17
3.2.3 Zařízení Lampa.....	18
3.2.4 Design navigace FAB	18
3.3 Design stránky pro výběr čidla.....	19
3.4 Design vytvoření Lampy.....	19
3.5 Design vytvoření Zařízení na měření teploty a vlhkosti	20
3.6 Design výběru barev.....	20
3.7 Design stránky Nastavení	21
3.8 Paleta barev.....	21
4 Programování aplikace.....	22
4.1 Backend	22
4.1.1 Amazon Web Services Amplify	22
4.1.2 Konfigurace a instalace AWS Amplify	22
4.1.3 Amplify Authentification.....	23
4.1.4 Amplify PubSub	24
4.1.5 GraphQL API.....	24
4.1.6 GraphQL schéma	25
4.1.7 Vytvoření Lampy.....	26
4.1.8 Změna barvy Lampy.....	26

4.1.9	Posílání dat do zařízení Lampa.....	27
4.1.10	Mazání zařízení Lampa.....	28
4.2	<i>Fronend</i>	28
4.2.1	React Native UI knihovna	28
4.2.2	Vypisování jednotlivých zařízení	29

Úvod

Práce se zabývá zhotovením reálného produktu, a to konkrétně mobilní aplikace pro chytrou domácnost. Využití má jednoduché. Zpríjemnit život v domácnosti a vždy být informován, jaká teplota či vlhkost je v daném okamžiku je doma, ať už se nachází uživatel někdekoliv s internetovým připojením.

Mobilní aplikace pro chytrou domácnost bude programována v React Native. Jako backend byl zvolen Amazon Web Services Amplify. Aplikace bude umět v reálném čase snímat vlhkost a teplotu ze senzoru DHT11. Z aplikace si uživatel bude moci rozsvítit, nebo změnit barvu RGB ledky. Pro přenos přes MQTT protokol bude používán Amazon Web Services IoT. Pomocí tohoto poskytovatele clouдовých služeb bude komunikovat s IoT prvky v řádu milisekund. Součástí je vypracování grafického návrhu, tak aby aplikace pro koncové uživatele byla „user friendly“.

1 Teoretické seznámení s IoT prvky

1.1 Zařízení na měření vlhkosti a teploty

Zařízení se skládá z desky WeMos D1R2 a čidla DHT 11. Propojený pomocí vodičů s konektory. Funkce zařízení je měřit teplotu a vlhkost. Všechny naměřená data se následně odesílají a zobrazují se uživateli v reálném čase.

1.1.1 WeMos D1 R2

WeMos D1 R2 WiFi ESP je programovatelná deska přes Arduino IDE. Je kompatibilní s Arduino platformou. Je postavené na modulu ESP 8266. připojení k Wi-Fi. Oproti Arduinu je rychlejší a výkonnější. Používá MicroUSB k připojení do počítače, nahrávání kódu nebo napájení. Na trhu jsou různé verze WeMos D1. Například: MINI, R1 nebo R2.

Důvodem výběru je široké využití. Obsahuje moderní Wi-Fi modul ESP8266.

Hlavní důvod je snadné programování.

1.1.2 DHT 11

Digitální senzor DHT11 je senzor vlhkosti a zároveň také teploty vzduchu. Disponuje svým širokým rozsahem od 20% až po 90% s přesností na 5% a rozsah teploty 0 ~ 60 °C s přesností na ± 2 °C. Maximální proud 2,5mA. Napájet lze přes 3.3 nebo 5 V. Nepřekračovat frekvenci více než 1Hz to znamená, že může poslat data každé dvě sekundy.

1.2 Zařízení Lampa

Zařízení Lampa se skládá z vývojové desky WeMos D1 mini ESP32, RGB LED diody, univerzálního plošného spoje, 2 female haderů a tří 220 ohmových rezistorů. Lampa dokáže měnit barvu podle nastavené v aplikaci.

1.2.1 WeMos D1 mini ESP32

Programovatelná vývojová deska s ESP32 čipem vlastní podporu Wi-Fi a Bluetooth. Wi-Fi mikrokontroler ESP32 vyvinula firma Espressif Systems. Snadno programovatelná pomocí Arduino IDE. Napájecí napětí činí 3,0 V ~ 3,6 V (VCC), 3,0V – 5,0V (USB). Provozní napětí vyskytující se na GPIO pinech je 3,3V. V klidném režimu spotřebovává deska 30mA až 80mA. Wi-Fi a Bluetooth spotřebovává 100mA až 240mA. Můžu se používat periferní rozhraní: UART, I2C, GPIO, DAC. Wi-Fi běží na standardu 802.11 o frekvenci 2,4GHz.

Obsahuje Bluetooth s verzí 4.2. Provozní teplota se může pohybovat mezi -40°C a $+85^{\circ}\text{C}$. K dispozici je až 40 GPIO pinů.

Pro projekt byl hlavním důvodem výběru velikost desky. Na svoji velikost má široké využití a spoustu funkcí. Dalším důvodem je velmi nízká cena, která také rozhodovala.

1.2.2 RGB LED dioda

Dioda o velikosti 5 mm obsahuje 4 vodiče s roztečí 1,25mm. Barva pouzdra je transparentní. Maximální proud může dosáhnou až 20 mA. Napětí v propustném směru pro červenou 1,79V pro zelenou 2,32V a pro modrou 2,57V.

Důvodem výběru RGB LED diody byl její funkce. Oproti klasické LED diodě je pro koncového uživatele příjemnější, měnit barvu podle potřeby.

2 Programování IoT prvků

2.1 Programovací platforma

2.1.1 Arduino IDE

Vývojové prostředí Arduino IDE (Integrate Development Enviroment). Je typické svou jednoduchostí a velkou škálou druhů hardwarů se kterým je kompatibilní. Mimo jiné obsahuje též textový editor a několik tlačítek například: kontrola kódu, nahrání kódu do desky, atd.. Po nahrání lze zkontolovat výstup dat pomocí serial monitor.

Všechny projekty se ukládají do pracovního adresáře. Při ukládání se vytvoří pro každý projekt podsložka. Arduino soubor má příponu *cpp*.

2.2 Princip posílání dat

2.2.1 MQTT protokol

MQTT (Message Queue Telemetry Transport) je protokol určený pro přenos dat mezi vzdálenými místy s malou datovou kapacitou. Je to otevřený standard publish-subscribe protokol. Pracuje na protokolu TCP. Vlastní klient-server architekturu. To znamená, že každý senzor je klientem a připojuje se na server, který se nazývá broker. Broker v tomto projektu představuje AWS IoT Core.

Komunikace mezi brokerem a klientem probíhá ve formě zpráv. Každá MQTT zpráva se posílá na adresu, která se nazývá *topic*. Klienti mohou posílat zprávy na více než jeden topic a každý může z více topiků data přijímat. Odesílání dat se říká *publish*. Datům, které daný senzor přijímá se říká *subscribe*. Příjemcem je zařízení Lampa. Odesílatelem je zařízení Zařízení pro měření teploty a vlhkosti vzduchu. Toto zařízení posílá každé dvě sekundy data přímo do aplikace. V projektu byl použit topic, díky kterému lze do budoucna přidávat více zařízení. Topic používaný pro Zařízení na měření teploty a vlhkosti má tvar:

device/čísloZařízení/data

Topic používaný pro Zařízení na měření teploty a vlhkosti s 15 minutovým odesíláním dat určený do databáze má tvar:

device/čísloZařízení/database

V projektu se data posílat do databáze nebudou, ale do budoucna to bude připravené. Další topic používaný pro Lampu slouží jako příjem dat do senzoru. Liší se na konci, kde je nastaveno sub (subscribe).

device/čísloZařízení/sub

Čísla zařízení jsou 1 a 2. Více zařízení nebylo použito z důvodu, že zařízení byly pravděpodobně již od výroby nefunkční. Čísla zařízení je díky takto upravenému topiku velmi přehledná a připravena pro větší počet zařízení.

MQTT poskytuje tři stupně QoS

- Doručení zprávy jednou bez potvrzení
- Doručení zprávy aspoň jednou, a to s potvrzením
- Doručení zprávy přesně jednou

Pro větší bezpečnost MQTT nabízí zašifrování pomocí SSL/TLS. Volitelně se pak může zvolit heslo a jméno uživatele. Tento protokol operuje na TCP/IP.

2.2.2 Princip posílání dat

Data se posílají v reálném čase přes AWS IoT. V AWS IoT Core v sekci *test* je možné vidět všechny data a také topiky poslané ze zařízení. Data posланé ze zařízení pak putují do přes cloudové služby do AWS Amplify. Tam pomocí knihovny PubSub se vypisují data do aplikace.

2.2.3 Formát poslaných a přijatých dat

Formát dat lze vidět na obrázku, který pochází z AWS IoT Core ze sekce pro testování.

The screenshot shows the AWS IoT Core interface with the 'Test' section selected. In the 'Subscriptions' section, there is one entry for the topic '#'. The message payload is displayed as a JSON object:

```
{ "lamp": "on", "red": 255, "green": 255, "blue": 0 }
```

Obrázek 1 Ukázka formátu dat poslaných z Lampy

Formát dat je posílan ve formátu json. Z aplikace se posílají data ve formátu:

```
{  
    "lamp": "on",  
    "red": 255,  
    "green": 255,  
    "blue": 0  
}
```

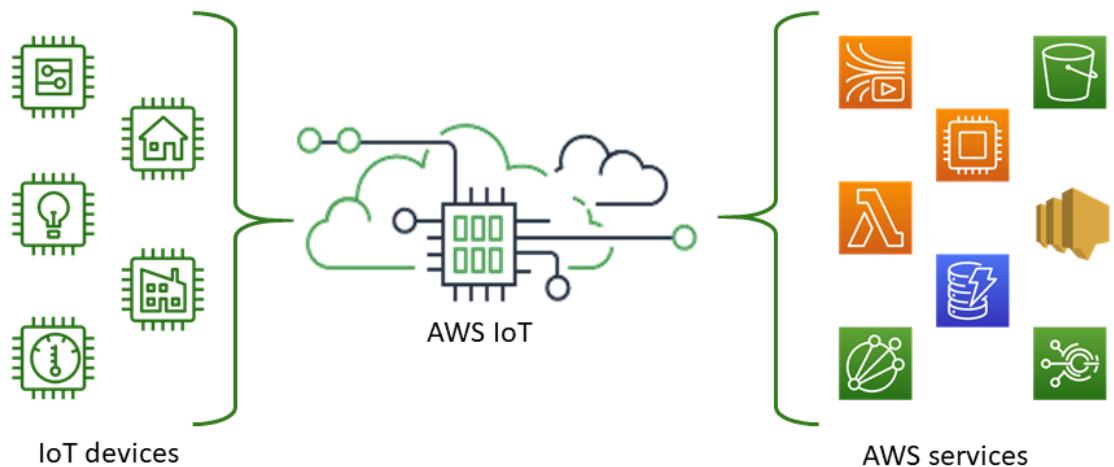
První řádek označuje stav Lampy. Druhý, třetí a čtvrtý řádek je obsahuje intenzitu dané barvy definované od 0 do 255.

2.3 Amazon Web Services IoT

2.3.1 Informace o AWS IoT

AWS IoT poskytuje clouдовých služeb, které propojí IoT zařízení se službami.

Projekt je konkrétně propojen se službou AWS Amplify, která poskytuje backend pro aplikaci.



Obrázek 2 AWS IoT

Služba AWS IoT Core připojí IoT zařízení do AWS IoT a dalších služeb. AWS IoT Core zahrnuje výchozí bránu zařízení a broker pro zprávy. AWS IoT Core se připojí a zpracovává zprávu mezi IoT zařízením cloudem.

2.3.2 Zaregistrování IoT zařízení

Každé zařízení by mělo mít svoji registraci. Registrace je důležitá, už pro vlastní pořádek, ale také pro bezpečí. Při každé registraci se generují klíče, které se zadávají buď přímo do

kódu, nebo do vedlejšího souboru. Zařízení používají X.509 certifikát pro ověření pomocí AWS IoT Core. K certifikátu jsou pripojovány AWS IoT policies. Tyto zásady AWS IoT určují, které operace může provádět jako subscribe nebo publish na MQTT topiky. Pro registraci je zapotřebí postupovat podle následného postupu:

- Vybere se záložka *Secure* a následně *Policies*.
- Pojmenuje se zásada. (PubSub_policy)
- Pole *Policy effect* se nechá v základu, tedy *Allow*. To znamená, že všichni klienty smějí ke svému certifikátu provádět akci uvedené v poli *Policy action*.
- V poli *Policy action* je potřebí vybrat *iot:Connect*. Policy action je nutné mít z důvodu spouštění programu Device SDK.
- *Policy resource* pole se zadá *. Tím povolíme jakéhokoliv klienta (zařízení)
- Tento postup se zopakuje ještě 3krát. Zadá se *iot:Receive*, *iot:Publish*, *iotSubscribe*. Ke každé rádce se přidá do pole *Policy resource* *. Hvězda (*) je používána pro jednoduchost. Pokud by byla potřeba větší bezpečnosti, je potřeba dodat přesně jaké zařízení může připojit a posílat zprávy podle specifikování klienta ARN. Klient ARN by měl mít formát:

`arn:aws:iot:používaný-region:používaný-aws-účet:client/id-zařízení`

Připojená zařízení jsou reprezentovány jako *thing object* v AWS IoT registry. Zařízení nebo

logické entity jsou představovány *thing object*. Fyzická zařízení nebo senzor to můžou být.

Vytvoření *thing object*:

- V AWS IoT se rozbalí záložka *Manage* a vybere se *Things*
- Na stránce *Create things* se vybere *Create single thing*.
- Zadá se jméno (device_1). Jméno později nepůjde změnit, proto je důležité vybírat jméno s ohledem do budoucna.
- Na další stránce *Configure device certificate* se vybere doporučovaná volba *Auto-generate a new certificate*.
- Dále se vybere zásada, která byla vytvořena dříve.
- Po vytvoření se objeví nabídka na stažení certifikátů a klíčů. Do kódu pro zařízení je potřeba stáhnout *Private key*, *Device certificate* a *Root CA certificate*.

2.4 Programování zařízení

2.4.1 Programovací jazyk Arduina

Zařízení byla naprogramována v jazyce C++, s přídavnými a speciálními metodami a funkcemi. Jazyk v Arduinu je charakteristický tím, že má dvě základní funkce. Funkce `Setup()` a `Loop()`. Ve funkci `Setup()` se obvykle nastavují piny použitého zařízení, nastavení parametrů sériové komunikace a dalších aktivit, u kterých je potřeba provést jen jednou. Funkce `Loop()` je jádrem kódu. Zde se vykonává všechn kód, který je potřeba neustále provádět. Jak už napovídá z názvu, jedná se o nekonečnou smyčku, která se neustále furt dokola provádí. Může obsahovat četný ze vstupů, posílání dat, přijímání dat atd.

2.4.2 Zařízení na měření teploty a vlhkosti

Na připojení do AWS IoT Core byla použita předloha kódu od uživatele *HarringayMakerSpace* a jeho projektu *awsiot* na stránce github.com. Kód byl zcela přepsán až do dnešní podoby. Byly tam následně doplněny dva certifikáty a jeden klíč z IoT device. V kódu byly použity knihovny *PubSubClient*, *ESP8266WiFi*, *ArduinoJson* a *DHT sensor*. Všechny knihovny spadají pod volné využití jak pro své projekty, tak pro komerční využití.

V první části kódu se deklarují konstanty, které se využijí pro připojení k internetu a připojení k AWS IoT. Na těchto pár řádků jsou napsány citlivé informace. První slouží pro připojení k internetové síti domácnost. Název *ssid* označuje jméno domácí Wi-Fi sítě. *Password* zase heslo. Další řádek *awsEndpoint* znamená konečný bod, který se nachází ve spodní sekci *Settings*. První certifikát, který se doplní je *device certificate*. Další následuje klíč *private key* a potom certifikát *Root CA certificate*. Ve funkci `setup()` se provádí připojení do Wi-Fi sítě a AWS IoT. Ve funkci `loop()` se nalézá ta nejdůležitější část kódu. Nejprve se provede kontrola, zda je zařízení stále připojené do AWS IoT. Když ano, kód pokračuje načítáním dat ze senzoru DHT 11. Načítá se teplota a vlhkost vzduchu. Tam se provádí kontrola, zda senzor v pořádku načítá.

```
float humidity = dht.readHumidity();
float temp = dht.readTemperature();
float f = dht.readTemperature(true);

if (isnan(humidity) || isnan(temp) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
}
```

Pokračuje úprava do json formátu. To se provádí pomocí knihovny Arduino Json. Teplota se bude odesílat jako *DHTTemperature*, vlhkost se bude odesílat jako *DHTHumidity*.

```
StaticJsonDocument<256> doc;
doc["DHTTemperature"] = temp;
doc["DHTHumidity"] = humidity;

char out[200];
int result = serializeJson(doc, out);
```

Po rozřazení se data odesílají do dvou topiků. Jeden topic se přijímá v aplikaci, kde se data vypisují v reálném čase každé dvě sekundy: device/2/data. Druhý topic byl určen do databáze. Tento topic se odesílá v intervalu každých 15 minut.

```
if(millis() >= time_1 + DATA_INTERVAL) {
    time_1 +=DATA_INTERVAL;
    pubSubClient.publish("device/1/data", out);
}
if(millis() >= time_2 + DATABASE_INTERVAL) {
    time_2 +=DATABASE_INTERVAL;
    pubSubClient.publish("device/1/database", out);
}
```

2.4.3 Zařízení Lampa

Pro zařízení Lampa byl použit kód, který sloužil jen jako připojení. Kód následně prošel spoustou změn. Kód, ze kterého bylo se vycházelo, patří uživateli *aws-samples*. Byl použit z projektu *aws-iot-esp32-arduino-examples* na stránce *github.com*. Knihovny, které byly použity pro tento kód jsou: *ArduinoJson*, *WiFiClientSecure*, *PubSubClient* a *WiFi*. Všechny knihovny povoleno pro jakékoliv používání.

Všechny citlivé informace jsou uložené v souboru *secrets.h*. Ten obsahuje potřebné věci pro připojení do Wi-Fi sítě, awsEndpoint, certifikáty a klíč. Ve funkci *setup()* se vyskytuje podmínka *while()* díky které se ví kdy je zařízení připojené. Zároveň se zde nastaví piny, na kterých je RGB LED dioda připojená. V hlavní části kódu ve funkci *loop()* se kontrola zda je klient připojen. Pokud ano, tak přijímá data z aplikace.

```
void pubSubCheckConnect() {
    if ( ! pubSubClient.connected() ) {
        Serial.print("PubSubClient connecting to: ");
        Serial.print(AWS_IOT_ENDPOINT);
        while ( ! pubSubClient.connected() ) {
            Serial.print(".");
        }
    }
}
```

```

    pubSubClient.connect("Device1");
    Serial.println(" connected");
    pubSubClient.subscribe(SUBSCRIBE_DATA);
    pubSubClient.loop();
}

```

Následuje funkce pracuje s obdrženými daty. Data se roztrídí pomocí knihovny ArduinoJson do proměných pojmenovaných podle barev.

```

StaticJsonDocument<200> doc;
deserializeJson(doc, lamp);
int redValue = doc["red"];
int greenValue = doc["green"];
int blueValue = doc["blue"];

```

Poslední část kódu se zabývá stavem zařízení. Jestli bude zapnuté nebo vypnuto. Pokud se zapne, nastaví se barva světla. Pokud se vypne, nebude nastavena žádná barva. Pokud se stane chyba a nebude ve stavu *on* ani *off*, bude svítit bílá barva jako upozornění.

```

if(String(topic) == "device/1/sub") {
    if(lamp == "{\"lamp\":\"on\"}") {
        Serial.println("{\"lamp\":\"on\"}");
        analogWrite(redPin, redValue);
        analogWrite(greenPin, greenValue);
        analogWrite(bluePin, blueValue);
    }else if(lamp == "{\"lamp\":\"off\"}") {
        analogWrite(redPin, 0);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 0);
    }else{
        analogWrite(redPin, 255);
        analogWrite(greenPin, 255);
        analogWrite(bluePin, 255);
    }
}

```

3 Návrh a design responzivní aplikace

3.1 Autentifikační formuláře

3.1.1 Přihlášení

Stránka na přihlášení se skládá z uživatelského jména a hesla. Tato stránka je vygenerovaná od AWS Amplify Auth. Z této stránky lze přejít na stránku Zapomenuté heslo a Registrace. Údaje každého uživatele je možné vidět v Amplify studiu.

3.1.2 Registrace

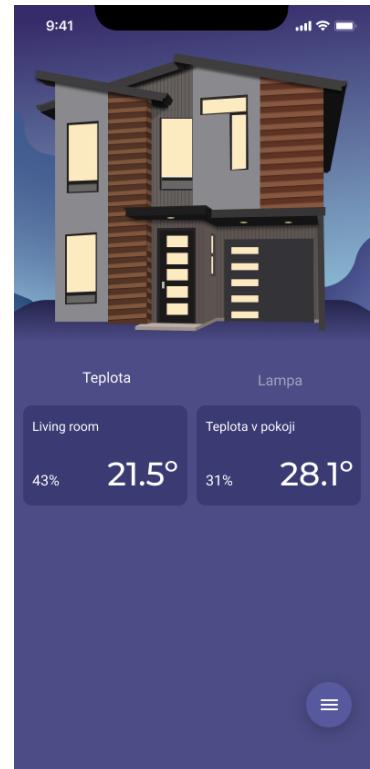
Stránka, která slouží pro registraci kolonku pro celé jméno, uživatelské jméno, email, heslo a telefonní číslo. Inputy už jsou předem ošetřeny před nezadáním žádné hodnoty nebo hodnoty, která není v souladu s pravidly. Telefonní číslo je potřeba zadat, ale pro další použití není potřeba ani celé jméno. Tyto údaje jsou jen pro správce aplikace. Email je potřeba proto, aby se mohlo ověřit správné zadání email. V případě, že by byl špatně zadán email by se nedalo resetovat heslo. Po registraci aplikace vyzve aplikace k zadání uživatelského jména a ověřovacího kódu. Tento kód přijde během pár sekund automaticky vygenerovaným emailem.

3.1.3 Zapomenuté heslo

Při potíži na stránce přihlášení, je možné změnit heslo. K tomu je potřeba zadat uživatelské jméno, na které se chce daný uživatel přihlásit. Po zadání uživatelského jména aplikace vyzve k potvrzení kódu z emailu. Zadává se 6-ti místní kód. Zadáním kódu se resetuje heslo a opět se vrátí na přihlašovací stránku.

3.2 Hlavní stránka

Hlavní stránka byla nadesignovaná tak, aby zbytečný věci nepřekáželi uživateli a působily „user friendly“. Zde hrála velkou roli přehlednost. Jednotlivá zařízení ukazují nejdůležitější věci. Obrázek zase ukazuje, o jakou aplikaci se jedná. Mezi zařízeními jako Teplota a Lampa lze posouvat bud' prstem, nebo kliknutím na nápis, který slouží jako tlačítko. Navigace se nachází dole vpravo. Zabírá málo místa, tak uživateli nijak nepřekáží.



3.2.1 Obrázek

Obrázek byl vytvořen pomocí vektorů v designovém softwaru Figma. Jako předloha sloužil obrázek z internetu. Jedná se pouze ilustrační obraz. Funkce obrázku je informovat uživatele o jakou aplikaci se jedná. Díky obrázku aplikace získává motiv a nevypadá prázdně. Obrázek se skládá z domu, který má působit moderním stylem. Nachází se v noční „krajině na kopečku“. V pozadí jsou „hory“.

3.2.2 Design Zařízení na měření teploty a vlhkosti

Zařízení obsahuje jen ty nejdůležitější věci. Název zařízení, který může obsahovat místo, kde se nachází. Funkce je informativní a aby se daly jednotlivá zařízení od sebe odlišit. Dále obsahuje velkými číslicemi teplotu vyjádřenou ve stupních cesia. Malými číslicemi vlhkost vyjádřenou procenty. Velikost číslic je podle důležitosti. Teplota je velkými číslicemi proto, protože uživatele upoutá jako první. Vlhkost je malými proto, protože je méně důležitá. Ne každý si může představit, jaká vlhkost je pro místo správná. Toto zařízení se dá odstranit pomocí delším podržením. Je to udělané proto, aby si uživatel omylem nesmazal zařízení. Po podržení na 2 sekundy se ukáže alert. V alertu se dotazuje, zdá smazat zařízení, či nikoliv.

3.2.3 Zařízení Lampa

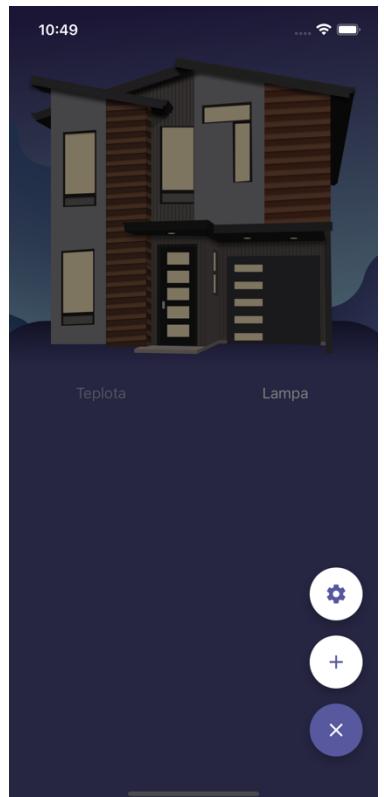
Zařízení obsahuje informativní název, kde se právě nachází. Pod názvem je šipka, která odkazuje na stránku pro výběr barvy. Tato šipka je vektorová. Šipka funguje jako tlačítko. Její rozměry jsou větší než se zdá. To proto, aby uživatel mohl správně stisknout tlačítko a neklikl omylem vedle. Tento komponent je ohrazený tenkým rámečkem. Díky rámečku uživatel vidí, která barva je právě nastavená. Při defaultním vytvoření obsahuje černou. To znamená, že nebude zařízení při stisknutí svítit. Celý komponent funguje jako tlačítko. Při stisknutí, změní barvu na světlejší (viz druhý zařízení *Lampa v pokoji*). Opětovným stisknutím se barva vrací do základní a znamená to, že zařízení je vypnuto. Zařízení lze smazat dlouhým podržením. Tady je to udělané nejen pro to, že by došlo k omyleu, ale proto aby se předešlo chybnému zapnutí lampy. Díky tomuto mechanismu se nachází méně věcí. Stává se tak přehlednějším.

3.2.4 Design navigace FAB

FAB (Floating Action Button) je tlačítko rozbalující menu. Při aktivaci tohoto tlačítka, obrazovka ztmavne. Díky tomu víme, že nikam jinam než na rozbalené menu nemůžeme klinout. FAB rozbalí dvě tlačítka společně s tím hlavním. Tlačítka obsahují jednoduché vektorové ikony. První tlačítko ze shora, odkazuje na nastavení. Druhé odkazuje na stránku pro přidání zařízení. Třetí tlačítko se ze tří čar změní na krížek. Při kliknutí na krížek, se ztmavlý display opět zesvětlí a tlačítka se zabalí. Díky FAB se ušetří spoustu místa.



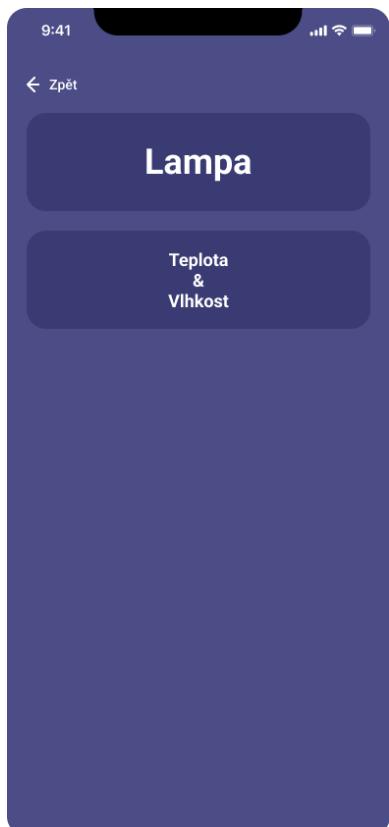
Obrázek 4 Hlavní stránka s Lampou



Obrázek 5 design FABu

3.3 Design stránky pro výběr čidla

Na stránce jsou dvě tlačítka. Tlačítko Lampa odkazuje na stránku pro vytvoření nové Lampy. Druhé tlačítko odkazuje na stránku pro vytvoření Zařízení na měření teploty a vlhkosti. Šipka v levém horním rohu s textem Zpět, odkazuje na předchozí stránku, takže na hlavní stránku.



Obrázek 6 Výběr zařízení

Stránka obsahuje opět tlačítko odkazující na přechozí stránku. Uprostřed nachází nadpis Lampa, který informuje co se vytváří. Pod nadpisem je vysvětleno, co je potřeba udělat před zadáním čísla zařízení a názvem. Vše je barevně odlišeno kvůli přehlednosti. Stránka obsahuje dva inputy. Tyto inputy jsou od sebe odlišeny takzvanými „placeholder“. Po klepnutí na input, se objeví klávesnice. Při zadávání čísla zařízení se objeví numerická klávesnice. Při zadávání názvu bude k dispozici klasická klávesnice. Název má ještě ošetření velkého písmene. To znamená, že název bude vždycky začínat velkým písmenem. Oba inputy jsou musí obsahovat hodnoty. Pokud nebude obsahovat hodnotu, vyskočí alert, který upozorní na input, kde se nevyskytuje zadá hodnota. Taktéž oba inputy jsou ošetřeny před zadáním a příliš písmen nebo číslic. U čísla zařízení je možné zadat pouze číslo obsahující dvě čísla. Název muže obsahovat až 25 písmen. Toto je proto, že při zadání více písmen, se název překrývá tlačítko na hlavní stránce. Poslední tlačítko znamená, že se hodnoty obsahující inputy vepíšou do databáze a vytvoří se tlačítko. Zároveň tlačítko Potvrdit přesune uživatele na hlavní stránku



Obrázek 7 Vytvoření Lampy

3.5 Design vytvoření Zařízení na měření teploty a vlhkosti

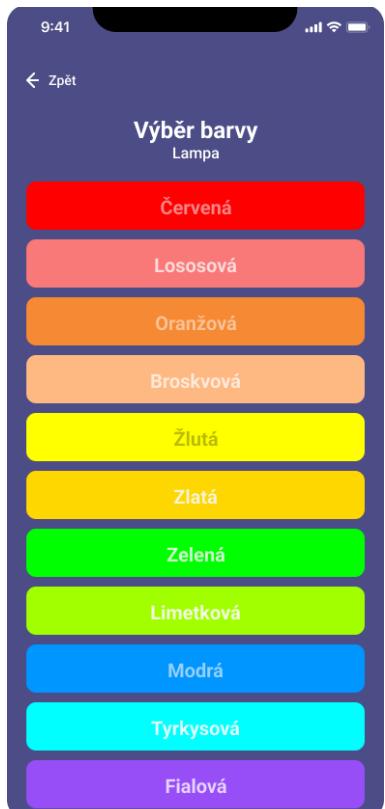
Stránka určená pro vytvoření Zařízení na měření teploty a vlhkosti se nijak zvlášť neliší od stránky pro vytvoření Lampy. Jediná odlišnost je v popisu se nachází v postupu připojení zařízení. Tato odlišnost se vyskytuje v příkladu pojmenování.

3.6 Design výběru barev

Stránka, která slouží pro výběr barev, obsahuje tlačítko zpět, nadpis, podnadpis a 12 tlačítek vyznačující se odlišnou barvou. Nadpis označuje název stránky. Podnadpis obsahuje název zařízení Lampa, pro které uživatel nastavuje barvu. Tlačítka, která označují barvu, určují jakou barvou bude zařízení Lampa svítit. Názvy tlačítek jsou pojmenovány kreativně podle barvy vyznačujícího se pojmenovaného předmětu. Je to netypické a proto by se to mohlo uživateli zamlouvat. Barvy byly vybírané vždycky podle klasických barev a k něm byla zvolena barva světlejšího motivu. Všechny barvy a jak vypadají když je Lampa rozsvícená je možné vidět v příloze.



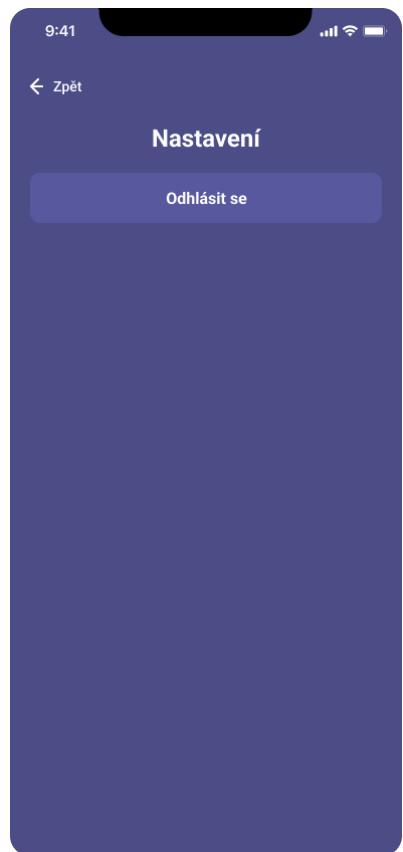
Obrázek 9 Design vytvoření teploty a vlhkosti



Obrázek 8 Design výběru barev

3.7 Design stránky Nastavení

Nastavení se nalézá opět tlačítko Zpět s funkcí vrátit uživatele zpět. Uprostřed stránky je nadpis Nastavení, jako informace na jaké stránce se nachází. Důležitou funkci však poskytuje tlačítko Odhlásit se. Díky němu může uživatel se odhlásit. Tím pádem se vrátí zpět na stránku Přihlášení.



3.8 Paleta barev

Barevný motiv, který byl zvolen, je tmavě fialový až modrý. Tlačítka se odlišují světlejším motivem než pozadí. Tmavší motivy se stávají čím dál populárnější, proto byl zvolen tmavý motiv. Paletě barev jsou uvedeny všechny barvy i jejich pojmenování, které je potřeba zadat programátorovi.

Obrázek 10 Design Nastavení



Obrázek 11 Paleta barev

4 Programování aplikace

4.1 Backend

4.1.1 Amazon Web Services Amplify

AWS Amplify je sada účelových nástrojů se spousty funkcí, které usnadňují dělání webů a mobilních aplikací rychleji a snadněji. Amplify slouží především pro konfiguraci backendu, ale také poskytuje připojení odkudkoliv pomocí AWS konzole a připojení aplikace v řádu minut. Dělání backendu je též možno pomocí grafické vizualizace, pro ty méně pokročilé, nebo až po nainstalování v schema.graphql, které se nachází ve složce amplify > backend > api.

Amplify CLI (Amplify Command Line Interface) - slouží pro vytváření, konfiguraci, mazání, správu cloudových služeb AWS. Podporuje více prostředí. Po každém vytvoření v terminálu, vytvoří se i v cloudové aplikaci. Všechna vytvořená backendová prostředí lze vidět v Amplify Console. Používání Amplify CLI v terminálu je jednoduché. Stačí napsat *amplify (configure, mock)* přidávání funkcí slouží *amplify add (auth, api, hosting, function, storage)*

4.1.2 Konfigurace a instalace AWS Amplify

Amplify bylo nainstalováno podle dokumentace. [5] Nejdříve je zapotřebí nainstalovat Amplify CLI přímo do vytvořeného projektu v React Native. Amplify CLI se instaluje globálně, takže bude možná zapotřebí přidat před příkaz *sudo*.

```
npm install -g @aws-amplify/cli
```

Poté je potřeba nastavit Amplify CLI.

```
amplify configure
```

Poté se otevře stránka, kde vyzvou k přihlášení nebo registraci. Po přihlášení se vybere AWS region, potom uživatelské jméno. Amazon vyzve k vytvoření uživatele na jejich stránce. Důležitá bude až ta poslední čtvrtá, tam se zkopírují klíče a vloží do konzole.

Zapne se aplikace a je čas začít s nastavováním Amplify.

Amplify init může určit na základě adresáře výchozí hodnoty. Pokud by vyhovovaly, stačí je potvrdit odpověď yes. V průběhu tohoto procesu se vytvoří tři zdroje.

- IAM role pro neověřené uživatele
- IAM role pro ověřené uživatele
- S3 bucket

```
amplify init
```

4.1.3 Amplify Authentification

Tento Amplify framework používá Amazon Cognito. Amazon Cognito poskytuje autentifikace. Je to robustní uživatelská adresářová složka, která zpracovává registrace, přihlašování, obnovy účtů.

Přidání funkce autentifikace se provádí pomocí příkazu:

```
amplify add auth
```

Po příkazu vás to vyzve, jestli chcete mít autentifikaci a zabezpečení přednastavené. Následuje otázka, jak se bude uživatel přihlašovat. V tomto projektu používám username. Je možné ještě nastavovat dál, ale není potřeba. Ukončení nastavování pak stačí zadat příkaz

```
amplify push
```

Následně se všechno nastavení odešle do Amazon cloutu. Aby vše fungovalo stačí nainstalovat potřebné moduly.

```
npm install aws-amplify amazon-cognito-identity-js @react-native-community/netinfo @react-native-async-storage/async-storage
```

Při používání klasického React Native je zapotřebí nainstalovat dané moduly zvlášť pro iOS. K tomu slouží příkaz:

```
npx pod-install ios
```

Kód, dostupný z dokumentace je Amplify Auth. Kód se vloží do App.js, případně do index.js.

```
import Amplify, { Auth } from 'aws-amplify';
import awsconfig from './aws-exports';
Amplify.configure(awsconfig);
```

Pro viditelnost přihlašovacích oken se musí do App.js, nebo index.js přidat úplně nahoru:

```
import { withAuthenticator } from 'aws-amplify-react-native';
```

A dolů kde se exportuje funkce, zabalit ji do:

```
export default withAuthenticator(App)
```

Při správném provedení by mělo ukázat se přihlašovací formulář pomocí username a password.

4.1.4 Amplify PubSub

Aby data posílané do IoT Core mohly vypisovat v aplikaci, bylo nutné propojit aplikaci s backendem IoT Core. K tomu bylo potřeba naistalovat potřebné knihovny.

```
npm install @aws-amplify/api @aws-amplify/pubsub
```

Po nainstalování se musí vytvořit novou zásadu v AWS IoT Core. Akce musí obsahovat iot:*. Tím říkáme, že přijímám jakékoli akci. Potom se vloží do kolonky Resource ARN

```
arn:aws:iot:<region>:<iot_id_účtu>:*
```

Efekt se povolí. Touto zásadou říkáme, povolíme plný přístup ke všem zařízením. Po založení nové zásady je potřeba tu zásadu pomítnout do aplikace, aby se aplikace mohla připojit ke všem zařízením. To se provede přidání řádky do kódu.

```
Auth.currentCredentials().then(creds => console.log(creds));  
Pomocí tohoto příkazu se zjistí, jaké cognito identity id je používáno
```

Před posledním zadání je potřeba připojit zásady pro komunikaci s IoT. Je nutné přejít do Cloud Formation. Tam je root stack projektu. Při roskliknutí se vybere resources a auth role. Pak atach policies AWSIoTDataAccess a AWSIoTCofigAccess.

Následně se přidá příkaz do terminálu v projektu.

```
aws iot attach-policy --policy-name 'myIoTPolicy' --target '< cognito_identity_id>'
```

4.1.5 GraphQL API

V projektu byla použita GraphQL API. Díky tomuto jazyku určený pro API, může uživatel komunikovat se serverem. GraphQL API je prostředí na straně severu pro provádění dotazů pomocí typového systému, který je již definován pro data. Není vázán na žádnou konkrétní databázi. Služba GraphQL je vytvořena definováním typů a polí. Na typech poskytuje funkce pro každé pole na každém typu.

4.1.6 GraphQL schéma

Projekt obsahuje dvě tabulky. Tyto tabulky se jmenují *CreateDHT* a *CreateLamp*. První tabulka obsahuje *id*, které je definováno automaticky. Další atribut je *dhtID*. Tam se doplňuje číslo zařízení, které se vloží po zadání. Poslední atribut *dhtTitle* je název. Druhá tabulka obsahuje také unikátní id. Výjimkou jsou atributy *red*, *green* a *blue*. Tyto atributy nemusí být doplněny, ale při vytvoření se doplní nula. Tyto atributy slouží k změně barvy. Data se tam připisují, jakmile na stránce Výběr barvy vybere barva. Tyto tabulky bohužel nemá každý nový uživatel unikátní. Tabulky jsou sdílené mezi uživateli. Proto obsahují práva, která povolují jejich využívání.

```
type CreateDHT @model @auth(rules: [{allow: public}]) {
    id: ID!
    dhtID: String!
    dhtTitle: String!
}

type CreateLamp @model @auth(rules: [{allow: public}]) {
    id: ID!
    lampID: String!
    lampTitle: String!
    red: String
    green: String
    blue: String
}
```

4.1.7 Vytvoření Lampy

Vytvoření lampy probíhá zadáním potřebných dat do inputů. Data se pomocí React hooks a to konkrétně `useState()` vloží do proměnných `lampID` a `lampTitle`. Díky volání funkce `Submit` stisknutím tlačítka *Potvrdit*, se také provede následující vytvoření nového itemu. Tato funkce obsahuje `DataStore`. To je především potřeba získat z knihovny *aws-amplify*. Funkce je zakončena voláním navigace, která přesune uživatele na hlavní stránku

```
await DataStore.save(  
  new CreateLamp({  
    lampID,  
    lampTitle,  
  }),  
);  
setLampID('');  
setLampTitle('');  
navigation.navigate('home');
```

4.1.8 Změna barvy Lampy

Změna barvy se provádí na stránce Výběr barvy. Nejdříve je potřeba se připojit na přesně daný *item*. To se provádí pomocí `useEffect()`. Pomocí konstanty `route` se připojím v tabulce `CreateLamp` na *item*.

```
const route = useRoute();  
  
useEffect(() => {  
  if (!route.params?.id) {  
    return;  
  }  
  DataStore.query(CreateLamp, route.params.id).then(setLamp);  
}, [route.params?.id]);
```

Poté už lze měnit barvy. V projektu je to udělatné tak, že každý tlačítko má svoji funkci, kde se je už barva předdefinována. Při stisku tlačítka se odešle a tím se přepíše hodnota na novou. Funkce konkrétně pro červenou barvu je ukázkou toho, jak vypadá každá funkce pro každé tlačítko. Barva se mění jedině v proměnné `color`. Pomocí `tinycolor()` změní stávající hodnotu na `rgb`. To se potom rozdělí do pole. Následně se jednotlivé hodnoty v poli dosadí do proměnných v tabulce. Při stisku volá funkci `setRed`, data se odešlou a hodnota se změní.

```

async function setRed(item) {
    let color = tinycolor('#ff0000').toRgbString();
    let sep = color.indexOf(',') > -1 ? ',' : ' ';
    color = color.substr(4).split(')')[0].split(sep);
    await DataStore.save(
        CreateLamp.copyOf(lamp, item => {
            item.red = color[0];
            item.green = color[1];
            item.blue = color[2];
        }),
    );
    navigation.navigate('home');
}

```

4.1.9 Posílání dat do zařízení Lampa

Předtím, aby zařízení mohlo v pořádku posílat data je nutné přidat *endpoint* a *region*. *Endpoint* slouží jako předloha, nikoli k plnému využití zde.

```

Amplify.addPluggable(
    new AWSIoTProvider({
        aws_pubsub_region: 'eu-central-1',
        aws_pubsub_endpoint:
            'wss://xxxxxxxxxxxx.iot.eu-central-1.amazonaws.com/mqtt',
    }),
);

```

Určování stavu zajišťuje podmínka. Pokud zařízení je aktivní, změní se barva na světlejší motiv. Tím se odešle zpráva na příslušný topik obsahující číslo zařízení. Pokud bude špatně zadané, tak se nevykoná žádná akce. Když by stav byl vypnutý, odešlou se do zařízení pouze barvy obsahující nulu.

```

PubSub.configure();
if (isOn) {
    console.log('is on');
    PubSub.publish(`device/${lampItem.lampID}/sub`, {
        lamp: 'on',
        red: Number(lampItem.red),
        green: Number(lampItem.green),
        blue: Number(lampItem.blue),
    });
} else {
    console.log('is off');
    PubSub.publish(`device/${lampItem.lampID}/sub`, {
        lamp: 'off',
        red: 0,
        green: 0,
        blue: 0,
    });
}

```

4.1.10 Mazání zařízení Lampa

Klasické zapínání probíhá jen tehdy jeli prst na tlačítka Lampa méně jak dvě sekund. Pokud ten prst je přidržen aspoň na dvě sekundy, tak se objeví alert tázající se zda to zařízení chce uživatel opravdu smazat. Pokud ne, nic se nestane. Pokud ano, odstraní se kompletně z databáze. Vrátit se to nedá. V alertu se vyskytuje i přesné jméno zařízení, pro lepsí uživatelskou orientaci.

```
const deletePressed = () =>
  Alert.alert(
    'Odstranit zařízení',
    `Opravdu chcete odstranit zařízení ${lampItem.lampTitle}`,
    [
      {
        text: 'Zrušit',
        onPress: () => console.log('Cancel Pressed'),
        style: 'cancel',
      },
      {
        text: 'OK',
        onPress: async () => {
          try {
            await DataStore.delete(lampItem);
          } catch (e) {
            console.log('Delete failed: $e');
          }
        },
      },
    ],
  );
};
```

4.2 Fronend

4.2.1 React Native UI knihovna

Díky této knihovně *react-native-ui-lib* byly všechny barvy uloženy do proměnných. Jejich použití bylo snadný. Stačilo importovat umístění a pak dát *colors.barva*.

4.2.2 Vypisování jednotlivých zařízení

Vypisování jednotlivých dat probíhalo pomocí FlatGrid. Všechny komponenty zařízení byly umístěny do mříže. Všechno reagovalo v reálném čase bez žádného čekání.

```
<FlatGrid
    itemDimension={130}
    data={dhtData}
    spacing={10}
    renderItem={({item}) => <SensorDHT dht={item} />}
    showsVerticalScrollIndicator={false}
/>
```

Závěr

V projektu jsem vybudoval svojí vlastní aplikaci pro chytrou domácnost. Zároveň jsem sestavil zařízení a naprogramoval do nich kód. Aplikace je schopna snímat data v reálném čase a odkudkoliv na světě.

Naučil jsem se používat nejmodernější technologie a programování v jednom z nejrozšířenějším jazyku pro mobilní aplikace. Také jsem si zkoušel práci s IoT zařízeními. Programovat v jazyku C.

Do programování v react native mě dostal youtuber jménem notJust.dev. Jen jemu mohu poděkovat za vysvětlení základních principů programování v react native, ale taky v Amazon Web Services.

Seznam použité literatury

Informace o ESP32 D1 Mini desce [online]. [cit. 2022-04-15]. Dostupné z:
<https://makersportal.com/shop/esp32-d1-mini-bluetoothwifi-board>

Infromace o RGB LED diodě [online]. [cit. 2022-04-15]. Dostupné z:
<https://dratek.cz/docs/produkty/1/1298/1434544602.pdf>

BARTOŠ, Štefan. *Web application for collection and visualization of sensor data: MQTT definice*. Brno, jaro 2020. Magisterská práce. Masarykova Univerzita Fakulta Informatiky.

Informace o AWS Amplify. *Amplify Framework Documentation* [online]. [cit. 2022-04-21]. Dostupné z: <https://docs.amplify.aws/>

Dokumentace AWS IoT. *What is AWS IoT?* [online]. [cit. 2022-04-21]. Dostupné z:
<https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>

GraphQL API definice. *Introduction to GraphQL* [online]. [cit. 2022-04-21]. Dostupné z:
<https://graphql.org/learn/>

Seznam obrázků

Obrázek 1Ukázka formátu dat poslaných z Lampy	10
Obrázek 2 AWS IoT	11
Obrázek 3 Hlavní stránka s teplotou	17
Obrázek 4 Hlavní stránka s Lampou	18
Obrázek 5 design FABu	18
Obrázek 6 Výběr zařízení.....	19
Obrázek 7 Vytvoření Lampy	19
Obrázek 8 Design výběr barev	20
Obrázek 9 Design vytvoření teploty a vlhkosti	20
Obrázek 10 Design Nastavení	21
Obrázek 11 Paleta barev.....	21
Obrázek 12 modrá LED	33
Obrázek 13 limetková LED.....	33
Obrázek 14 zelená LED	33
Obrázek 15 zlatá LED	33
Obrázek 16 žlutá LED	33
Obrázek 17 oranžová LED	33
Obrázek 18 broskvová LED	33
Obrázek 19 lososová LED.....	33
Obrázek 20 červená LED	33
Obrázek 21 růžová LED.....	34
Obrázek 22 fialová LED	34
Obrázek 23 tyrkysová LED.....	34

Přílohy



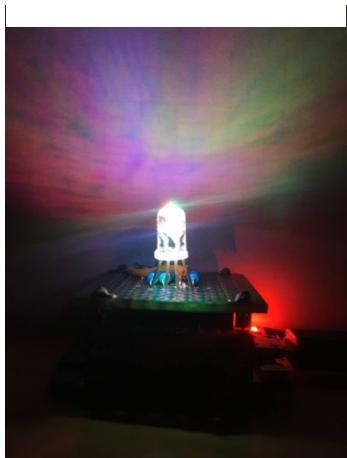
Obrázek 20 červená LED



Obrázek 18 brosvková LED



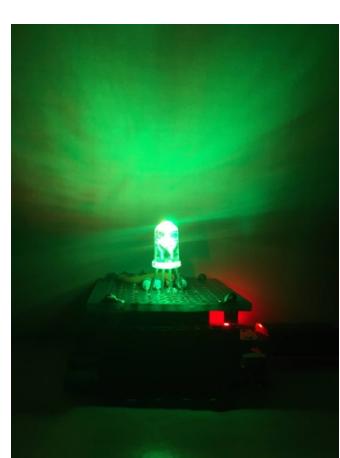
Obrázek 14 zelená LED



Obrázek 19 lososová LED



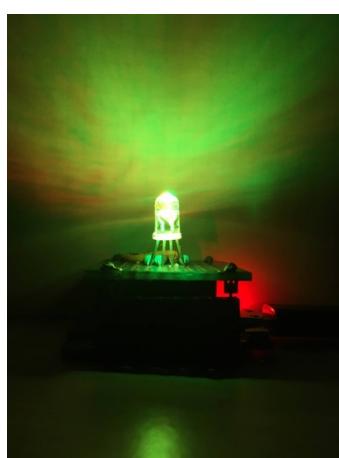
Obrázek 16 žlutá LED



Obrázek 13 limetková LED



Obrázek 17 oranžová LED



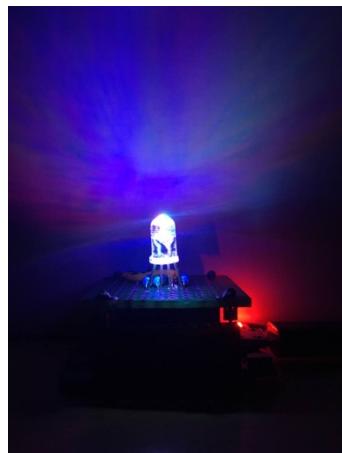
Obrázek 15 zlatá LED



Obrázek 12 modrá LED



Obrázek 23 tyrkysová LED



Obrázek 22 fialová LED



Obrázek 21 růžová LED