ITU - technická zpráva

Autoři:

Nevrkla Lukáš (xnevrk03)
Andrei Roshka (xroshk00)
Tomáš Juhász (xjuhas04)

Podrobná specifikace zadání

Abstrakt

Naša aplikácia slúži na spravovanie Bluetooth senzorov a zobrazovanie nimi nameraných hodnôt.
Naše riešenie by malo užívateľovi umožniť uľahčiť prácu so senzormi vďaka jednoduchému prístupu k všetkým potrebným dátam a intuitívnemu rozloženiu UI prvkov, ktoré je oproti konkurencie odľahčene od zbytočností ktoré iba spomaľovali funkčnosť aplikácie.

Analýza užívateľa

Cieľový užívateľ je niekto kto spravuje svoju domácnosť alebo pracovné prostredie a potrebuje monitorovať hodnoty vlhkosti a teploty okolia za pomoci vhodne rozmiestnených Bluetooth senzorov.
Užívateľ je niekto kto potrebuje aplikáciu plne zameranú na prácu so senzormi ktorá neobsahuje žiadne zbytočné prvky užívateľského rozhrania a nevyžaduje nadbytočnosti ako je tvorba účtu.

Typické prípady použitia

Užívateľ bude využívať aplikáciu na jednoduché spárovanie so senzormi, na ich nastavenie a následne na zobrazenie ich nameraných hodnôt.
Bude chcieť vedieť zobraziť dáta všetkých senzorov v prehľadnom formáte a v prípade potreby zobraziť potrebnejší graf obsahujúci dáta na rôznych časových intervaloch od dna až po rok.
Typicky užívateľ potrebuje prístup k týmto informáciám za účelom zlepšenia kvality života a zdravia napr. pri problémoch so spánkom, astmou, bolesťou hrdla. Užívateľ sa taktiež môže snažiť zaobstarať ideálne podmienky pre pestovane rastlín.

Potreby užívateľa

- Potenciálny užívateľ potrebuje byt informovaný o nameraných hodnotách vlhkosti a teploty v pravidelných intervaloch.
- Potrebuje jednoduchý a rýchly prístup k dátam všetkých senzorov poprípade ich vývoj v čase na grafe.

- Užívateľ sa chce vyhnúť zbytočnému preklikávaniu obrazoviek a nepotrebných prvkov užívateľského rozhrania.
- Užívateľ nechce strácať čas čakaním na načítanie jednoduchých dát.

Návrh architektury a GUI

Návrh architektury

Aplikaci jsme rozdělili na grafickou část, na API a na data.
Již existující knihovny pro komunikaci s námi použitými senzory nepodporovaly mobilní platformu a tak jsme zatím nevytvořili API, které by četlo data z reálných senzorů. Proto jsme vytvořili falešná data, která by měla simulovat reálné podmínky. Tyto data jsou uložena v interní paměti daného zařízení, kam by je ukládalo i reálné API.
Naše API dále zprostředkovává čtení a editování těchto dat, přičemž simulujeme čekací doby (hledání dostupných senzorů, čtení historických dat, …).

Návrh rozložení GUI

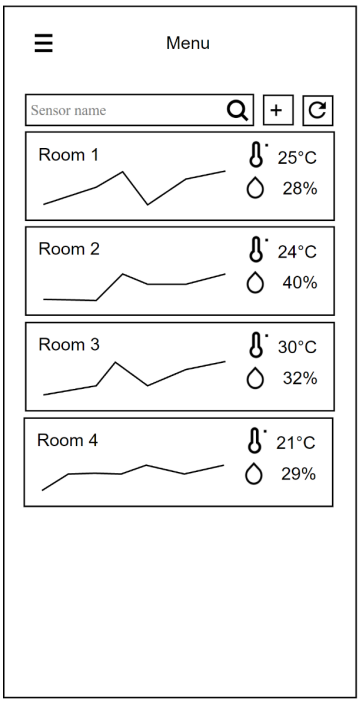
Jelikož v originální aplikaci Xiaomi Home byl největším problémem zdlouhavý přístup k datům, rozhodli jsme se aktuální data umístit rovnou na hlavní obrazovku. Ta by tedy měla sloužit zejména pro zobrazení všech připojených senzorů a jako rozcestník do dalších obrazovek. Jako nejvhodnější implementací se jeví seznam senzorů s jejich aktuálními daty.

Další obrazovka by měla obsahovat podrobnější informace z daného senzoru. Přesněji: aktuální teplotu a vlhkost, historické teploty a vlhkosti. Také by zde měla být možnost upravení / smazání daného senzoru. Implementace by měla obsahovat převážně graf historických hodnot.

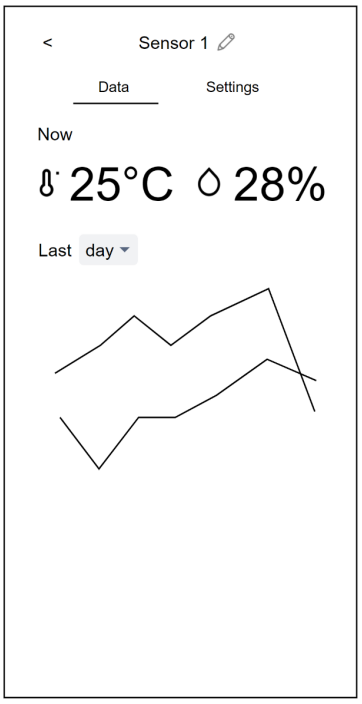
Další nedílnou součástí aplikace by mělo být nastavení. Zde by měla být především možnost nastavit intervaly automatické obnovy dat na pozadí.A příslušné změny jednotek teploty, či lokalizace.

Předběžný návrh GUI by mohl vypadat následovně:

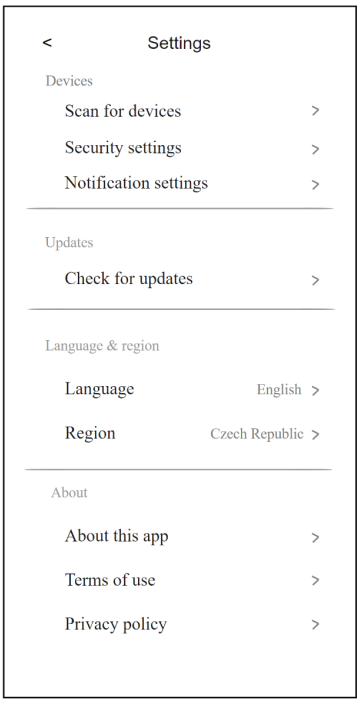
Hlavní obrazovka



Podrobnější přehled



Nastavení



Popis použitých nástrojů

- NodeJS - backend JavaScript runtime prostředí pro spouštění JS mimo prohlížeče.
- npm - balíčkový manažer pro NodeJS.
- React Native - UI knihovna, umožňující používat React spolu s nativními možnostmi platforem.
- React Navigation - knihovna pro navigace podle obrazovek.
- Expo - knihovna usnadňující spuštění a buildování React Native aplikaci.
- Native Base - nadstavba nad React Native, předdefinující obecné UI komponenty.
- Victory - knihovna pro vytváření grafů.

Popis implementace

Struktura projektu

Kořenový adresář obsahuje různé konfigurační soubory vygenerované automaticky a `index.js` - což je hlavním vstupním bodem NodeJS aplikace. Tam se provádí registrace kořenové komponenty React Native aplikaci App, která je definována v `App.js`. Všechny ostatní zdrojové soubory obsahuje složka `src`.

Složka `src` obsahuje tři hlavní podsložky podle rozdělené logiky aplikace:

- `components` obsahuje všechny dílčí UI komponenty, ze kterých se skládají stránky aplikace
- `screens` obsahuje definice jednotlivých obrazovek, které uvnitř sebe používají komponenty z `components`

- `utils` obsahuje pomocnou logiku
    - `api` - logika pro api jako funkcí pro generování falešných dat a pro simulování zpožděného o náhodný čas přístupu k datům senzoru
    - `redux` - logika pro globální stav aplikace a práci s daty, obsahující výchozí stav dat a funkcí pro práce s nim
    - `storage` - logika pro práce s interní paměti zařízení, na kterém běží aplikace
    - `charts.js` - pomocné funkcí pro zpracování dat ze senzoru do podoby potřebné pro vykreslení grafů
    - `utils.js` - obecné pomocné funkcí, např. převedení Celsia na Fahrenheity

Práce s daty

S daty pracujeme pomocí knihovny Redux pro správu globálního stavu aplikace. Globální stav je jediným zdrojem dat pro všechny komponenty. Stav je rozdělen na dvě logické části (tzv. slices):

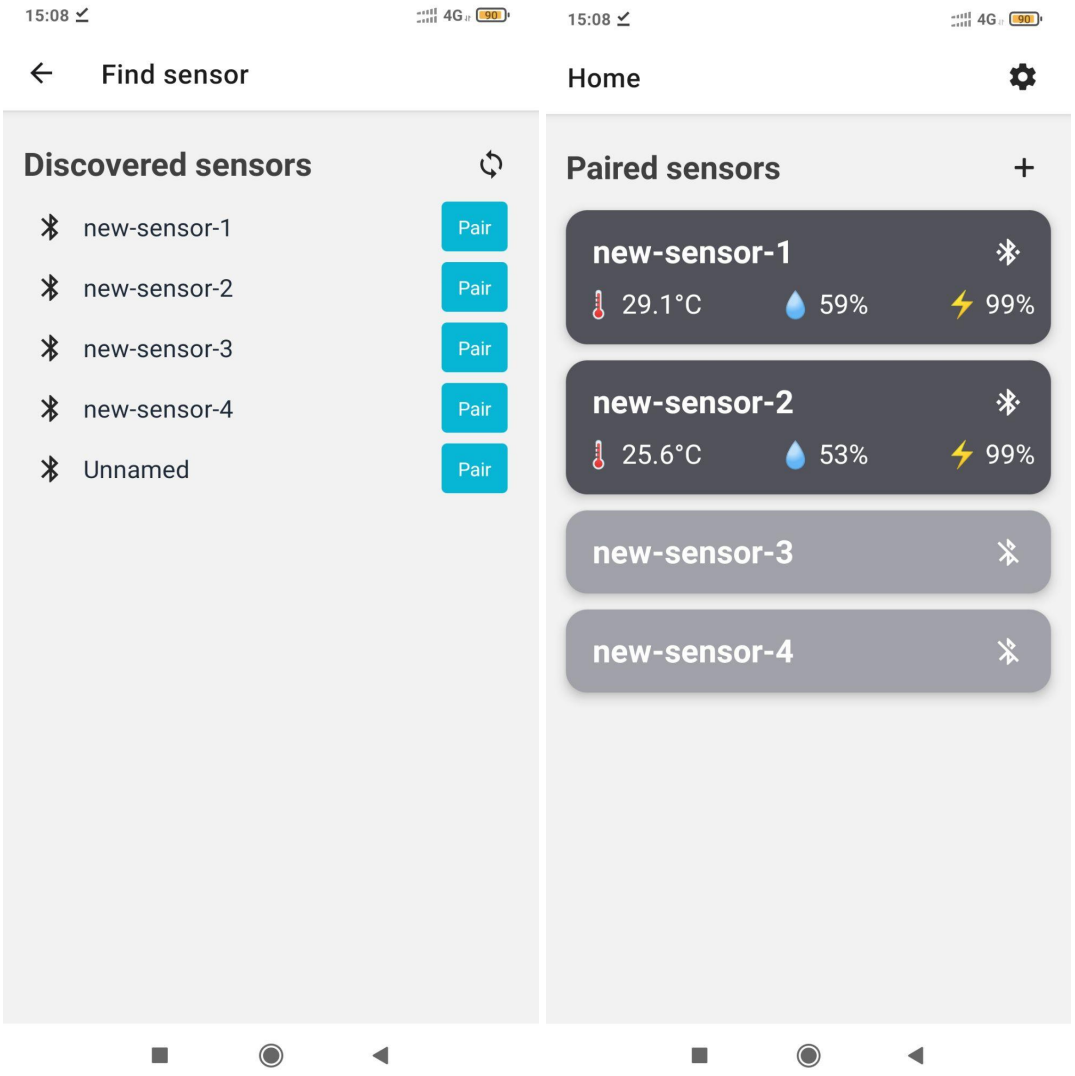
- `sensorsSlice.js` pro data objevených senzorů přístupných pro spárování, již spárovaných senzorů a jejich nastavení jako uživatelské zadané jméno senzoru
- `settingsSlice.js` pro nastavení aplikace jako například interval dotazování senzorů, jednotky měření (Celsia nebo Fahrenheity), jazyk a případně jiné.

Potřebovali jsme současně udržovat konzistentní data v několika místech - současný stav aplikace a interní paměť zařízení. Abychom eliminovali chyby při práci s daty na různých místech aplikace, byly vytvořené redux části (tzv. slices), které mimo dat obsahují i logiku pro práci s daty (tzv. actions). Mimo spravování stavu aplikace se tato logika také zabývá dotazováním API a ukládáním dat do interní paměti - na což existují speciální asynchronní akce (tzv. async thunk). Tím pádem robustnost práce s daty naší aplikace je na velmi vysoké úrovni. Data se mění na jediném místě a případné hledání chyb je velmi snadne.

Podřízené komponenty buď získávají data přímo dotazováním globálního stavu, nebo dostávají to jako vstupní data při vykreslení (tzv. props) od rodičovské komponenty, která tyto data již získala.

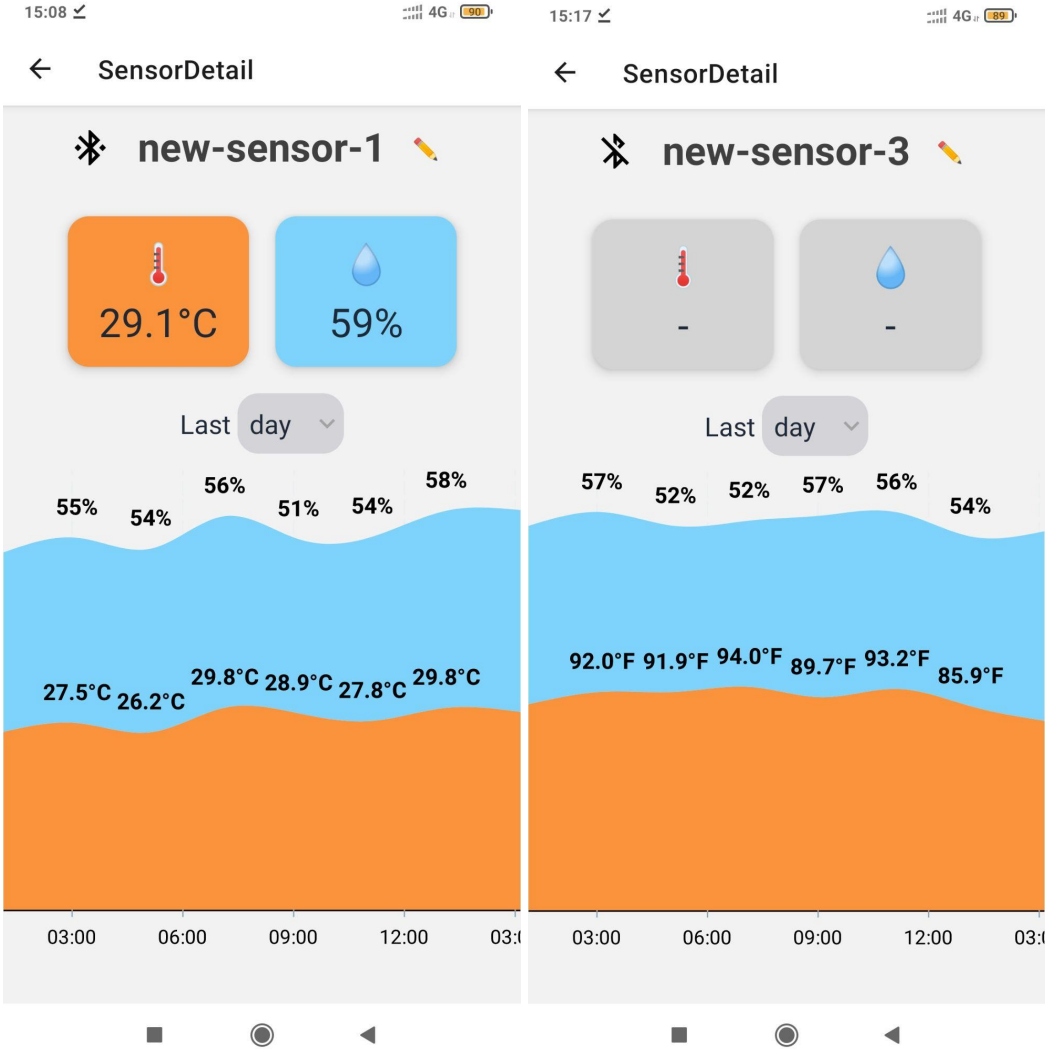
Screenshoty výsledné aplikace

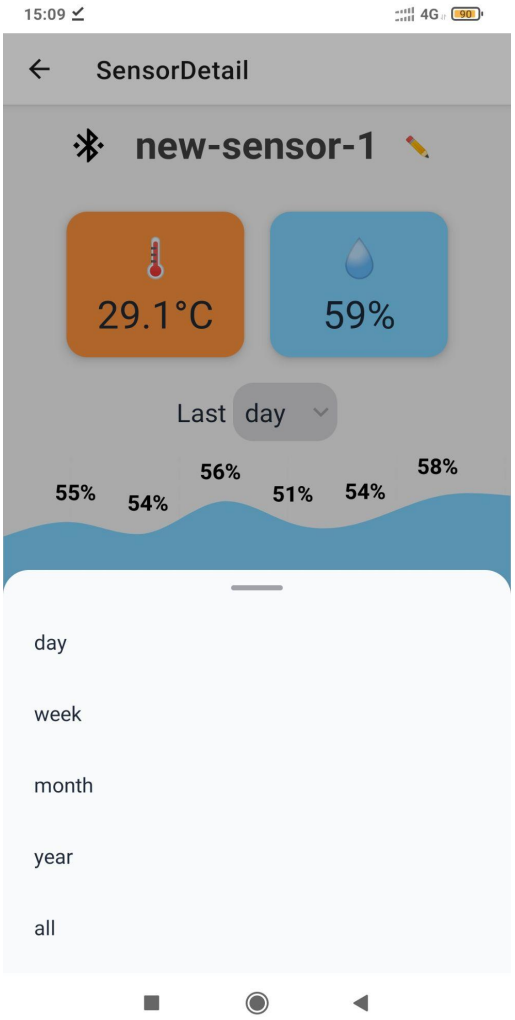
Spárování s novými senzory a domovská obrazovka

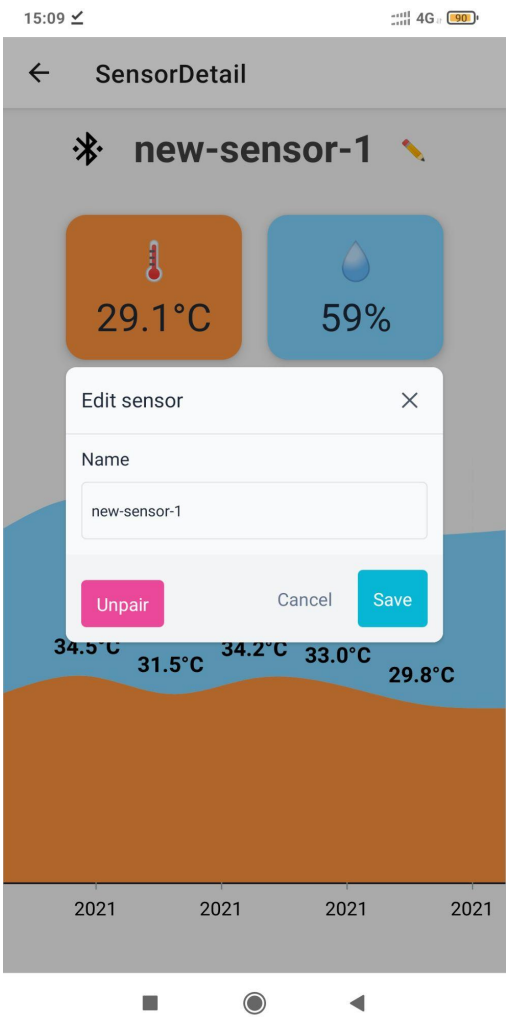


Detail senzoru

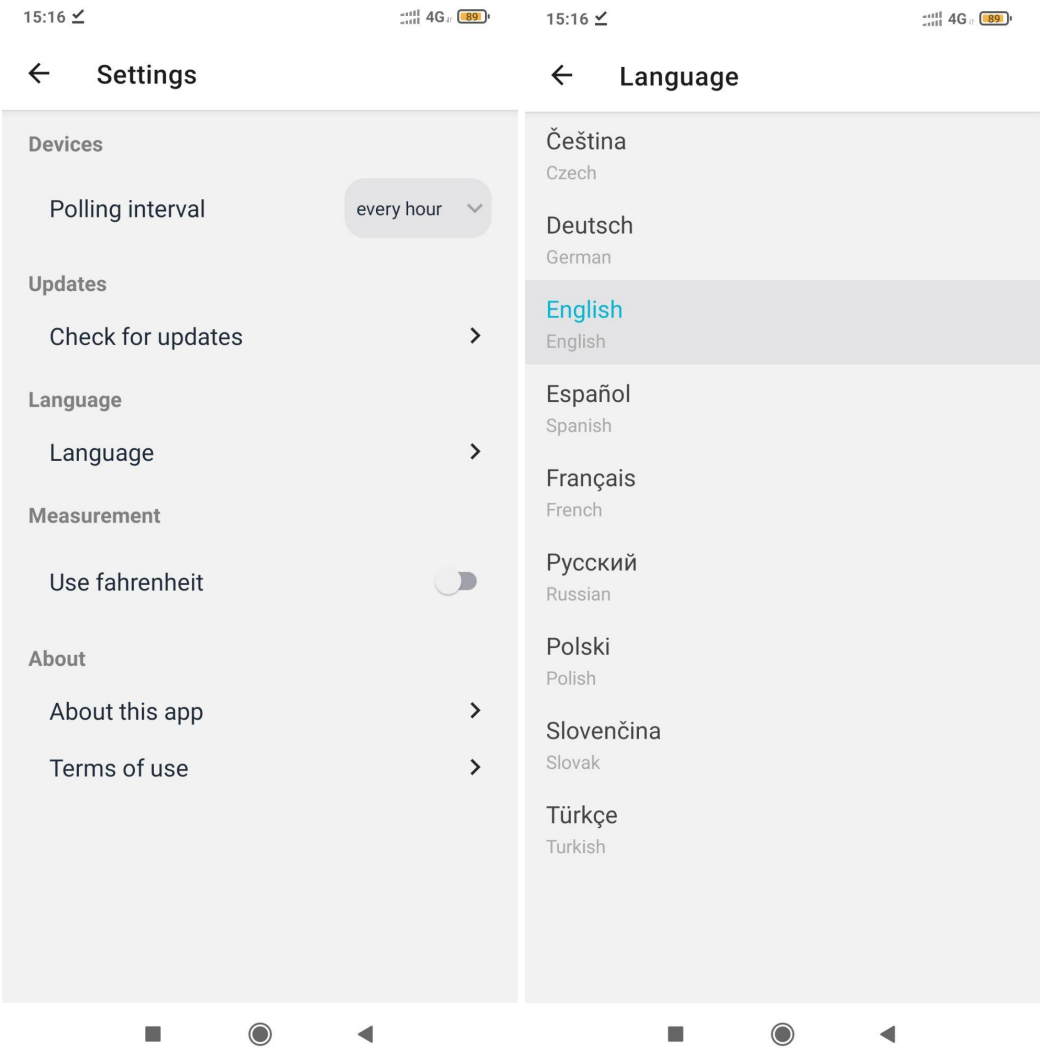
Stupně x Fahrenheity, připojený x nepřipojený senzor




15:09
SensorDetail
new-sensor-1
29.1°C
59%
Last day
55%
54%
56%
51%
54%
58%
day
week
month
year
all


15:09
SensorDetail
new-sensor-1
29.1°C
59%
Edit sensor
Name
new-sensor-1
Unpair
Cancel
Save
34.5°C
31.5°C
34.2°C
33.0°C
29.8°C
2021
2021
2021
2021

Natavení aplikace



Použitá literatura

- Originální aplikace Xiomi Home: https://play.google.com/store/apps/details?id=com.xiaomi.smarthome&hl=cs&gl=US
- Použitý senzor teploty a vlhosti: https://www.xiaomi.cz/xiaomi-mi-temperature-and-humidity-monitor-2/
- Dokumentace React Native: https://reactnative.dev/docs/getting-started
- Dokumentace Victory knihovny: https://formidable.com/open-source/victory/
- Dokumentace Native base: https://nativebase.io/