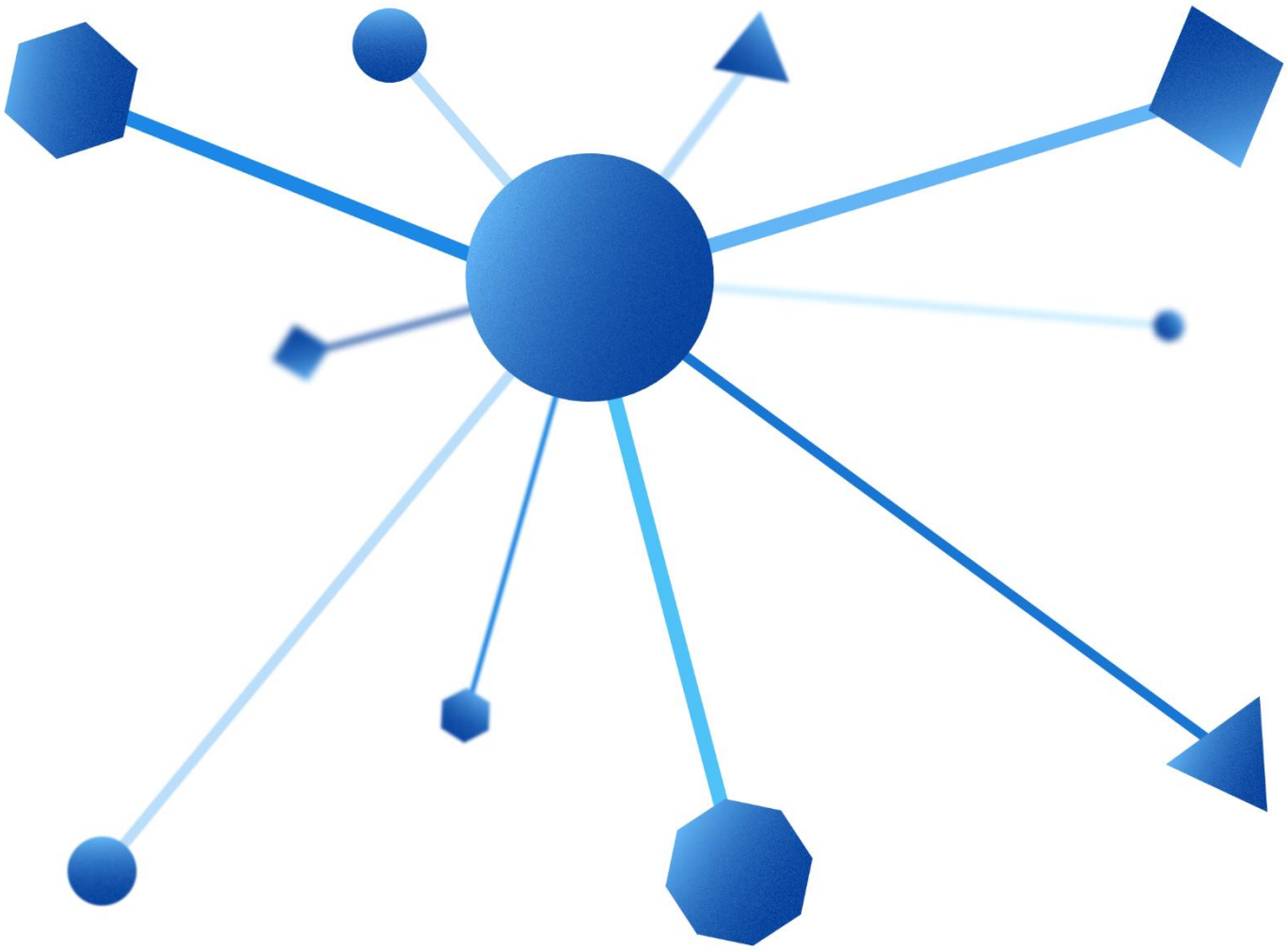


LoRaNet

Курсов проект по:

Проектиране и Интегриране
на Софтуерни Системи

Януари - 2022

Никола Тотев ФН 62271

Ивайло Петков ФН 62329

Съдържание

1. Въведение

a. Цел

Целта на проекта е да се създаде софтуерна система която събира данни за околната среда и ги предоставя на потребителите чрез мобилно или уеб приложение.

b. Резюме

За постигане на целта използваме модерни технологии и платформи като LoRa, React, Blazor Server, The Things Network и Flutter. В този документ сме описали защо сме направили този избор на технологии, архитектурата на системата, как компонентите са реализирани и внедрени, както възможни приложения и как си разпределихме работата.

c. Дефиниции и съкращения

Тъй като системата е сложна и използва множество различни платформи, за яснота в тази точка ще изредим важните дефиниции и съкращения

LoRa – Long Range – Това е патентована техника за широкообхватна мрежова модулация с ниска мощност. Това е физическия слой на една LoRa мрежа.

LoRaWAN – Long Range Wide Area Network - LoRaWAN е комуникационен протокол и мрежова архитектура, която се намира на върха на физическия слой на LoRa.

API – Application Programming Interface

Signal R - SignalR е безплатна софтуерна библиотека с отворен код за Microsoft ASP.NET, която позволява на сървърния код да изпраща асинхронни известия до уеб приложения от страна на клиента.

TTN – The Things Network е LoRaWAN платформа, която е критичен компонент за всяко LoRaWAN решение. Използван от хиляди компании и разработчици по целия свят, той сигурно управлява приложения, крайни устройства и gateways. Създадена е от The Things Industries.

ASP.NET - ASP.NET е framework за сървърни уеб приложения, предназначена за създаване на динамични уеб страници.

2. Използвани технологии и платформи

LoRa и LoRaWAN за връзка между сензорните станции и TTN.

TTN е външна платформа която използваме да свързване на нашия сървър със LoRa мрежата която се състои от сензорни станции и LoRa Gateways.

Използваме Blazor Server App (ASP.NET) за създаване на уеб сървъра, който приема заявки от TTN и предоставя API и сокети за клиентските приложения.

За база от данни използваме MSSQL.

Потребителските интерфейси са разработени с React и Flutter.

Електрониката на сензорните станции е базирана на платформата Arduino.

Технологията 3Д принтиране използвахме за създаване на тялото на станцията.

За внедряването на системата се използва хостинг доставчика „SmarterASP.NET“ и Google Play Store.

За улесняване на внедряването на системата на различни среди добавихме docker-compose файл, съдържащ нужната конфигурация за създаване на контейнери за Blazor Server App и React клиента.

3. Описание на архитектурата

a. Основни компоненти

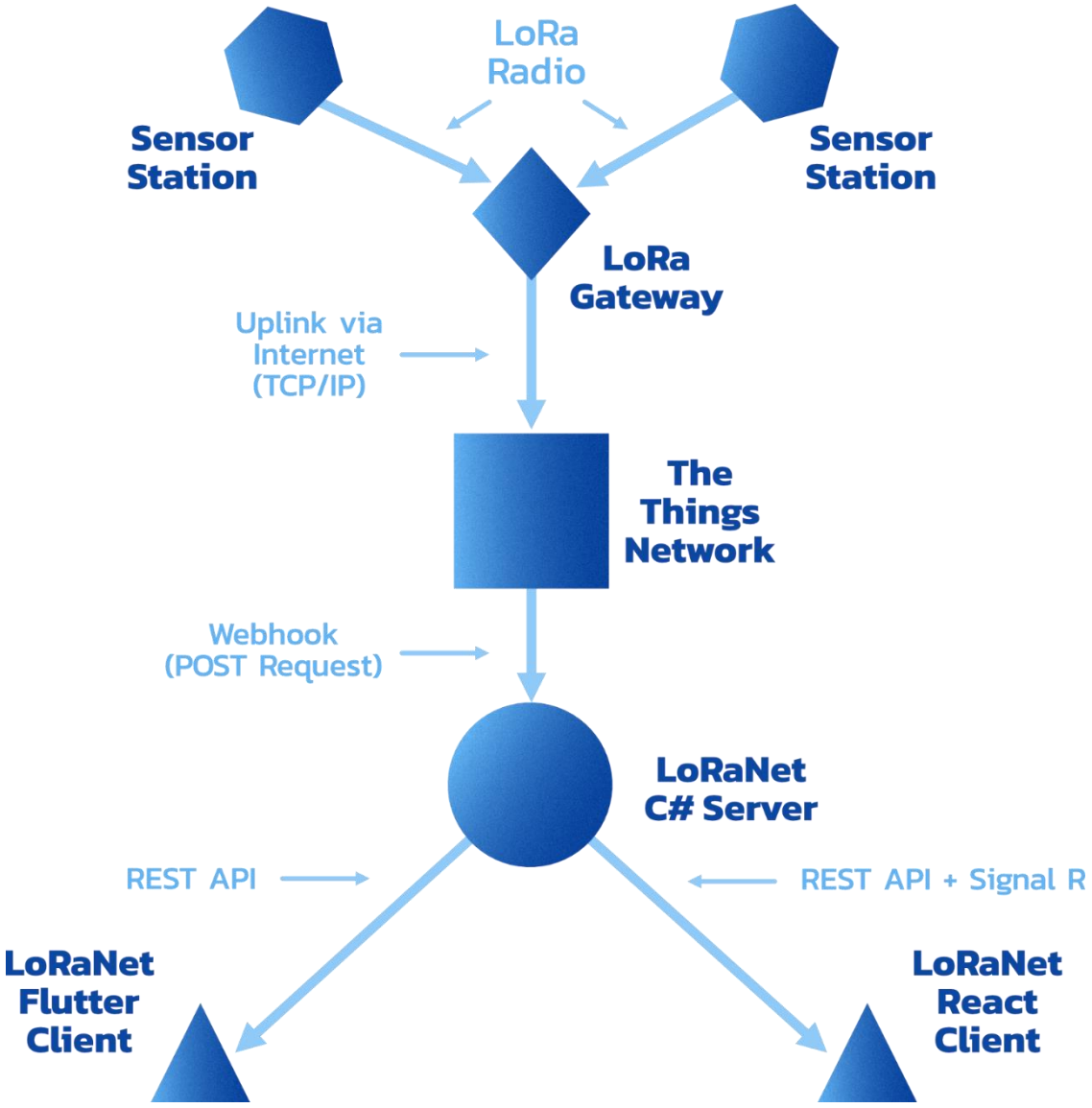
Целта на тази точка е да запознае читателя с основните компоненти на системата. Тук има само кратко описание и изреждане на функциите. Подробно описание на реализацията се намира в точката „Реализация на системата“

- **Сензорни станции** – Сензорните станции съдържат сензорите от които се събират данни, както и микроконтролера, който служи за управление на сензора и изпращане на данни. Измерванията от сензора се изпращат с LoRa до LoRa Gateway компонента.
- **LoRa Gateway** – Gateway компонента служи за приемане на данни от сензорните станции и да ги изпраща към TTN платформата/компонента.
- **TTN (The things network)** – TTN Платформата е външната система, с която нашата се интегрира. TTN приема данните от LoRa Gateway компонента и за всеки пакет от данни, от всяка станция, генерира webhook, който се изпраща до нашия сървър.
- **C# Blazor Server App** – C# Сървъра отговаря за приемане на заявки от TTN и запазването на данните в базата от данни. Освен на връзката с TTN сървъра също обслужва клиентските приложения като това става чрез два метода, REST API и Signal R връзки (друго наименование е сокети).
- **MSSQL База от данни** – Базата от данни пази данните, като избрахме MSSQL поради лесното свързване с C# и лесната работа с базата от SQL Studio. Особеностите за това как сървъра работи с базата от данни са описани в точката „Реализация на системата.

- **React уеб клиент** - React клиента е предназначен за използване като уебстраница на компютър. Решихме да имаме уеб клиент за да може да демонстрираме работа със сокети и REST API.

- **Flutter мобилен клиент** – Flutter клиента е предназначен за използване от телефон. Предимството на Flutter Framework е, че може да се компилира за Android и iOS използвайки еднакъв код. Избрахме да имаме мобилно приложение да демонстрираме REST API връзка със сървъра от мобилно устройство, тъй като голяма част от потенциални потребители на подобна система биха я достъпвали от телефон.

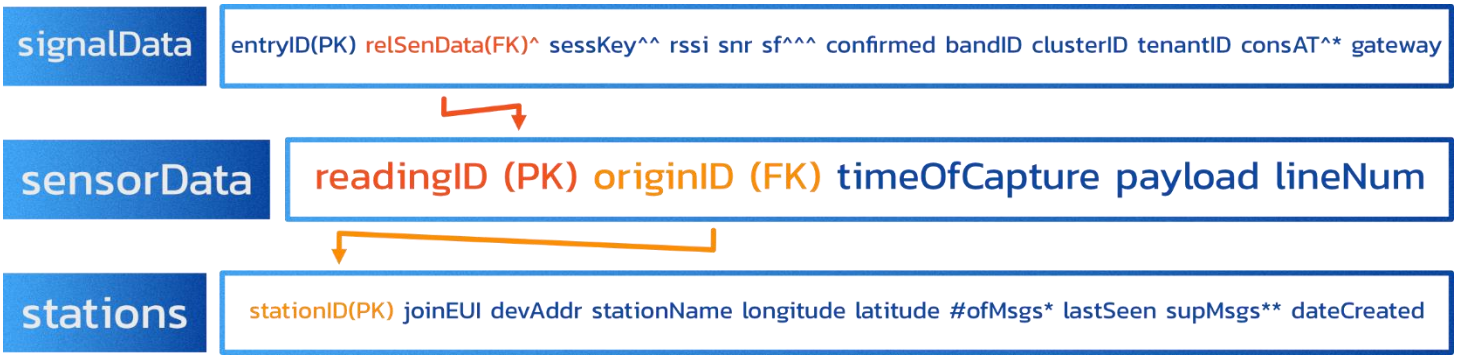
b. Начин на свързване



4. Реализация на системата

a. База от данни

Базата от данни е структурирана по следния начин:



*^ relatedSensorData ^^sessionKey ^^^spreadingFactor ^*ConsumedAirtime | *numberOfMessages*
***supportedMessages*

Таблицата "**sensorData**" съдържа измерванията от сензорите. За Primary Key използваме колоната **readingID**, като тази колона е Foreign Key за таблицата "**signalData**". Foreign Key за "**sensorData**" е "**originID**" което съответства на "**stationID**". Това позволява лесно да се намерят данни от определена станция. Самите данни се намират в колоната „**payload**" като са във формат Dictionary<String, String> която е кодирана в json низ.

Таблицата „**signalData**" съдържа за всяко измерване метаданни свързани с информация за LoRa мрежата. В следващи версии на системата тази таблица може да е много полезна за диагностика, като може да помогне да се определи кои станции имат добра свързаност и кои не. Foreign Key за тази таблица, както беше споменато в преди е Primary Key на "**sensorData**".

Таблицата „Stations" съдържа информация за всички станции в мрежата. При свързване на нова станция, тя автоматично се добавя в базата от данни. Най-важните данни са „**stationID**", GPS координатите "**longitude**" и "**latitude**", "**lastSeen**" и „**supportedMessages**". „**lastSeen**" показва кога за последно станцията е изпратила данни. Тази колона помага бързо да се установи дали дадена станция работи. Колоната „supportedMessages" е списък List<string> кодиран в json низ, който съдържа какви измервания дадена станция може да прави. Благодарение на това, в системата може да има различни станции, но лесно да се намират всички достъпни измервания.

За по-голяма сигурност, използваме „**storedProcedures**” за изпълняване на заявки към базата. Тези процедури са предварително компилирани SQL заявки който може да се използват с или без параметри. По-голямата сигурност идва от това, че този начин на изпълняване на заявки намалява шанса за SQL Injection и също така позволява на сървъра да се даде роля, която може да изпълнява само „**storedProcedures**” и по този начин дори и сървъра да бъде атакуван, няма admin level достъп до базата от данни.

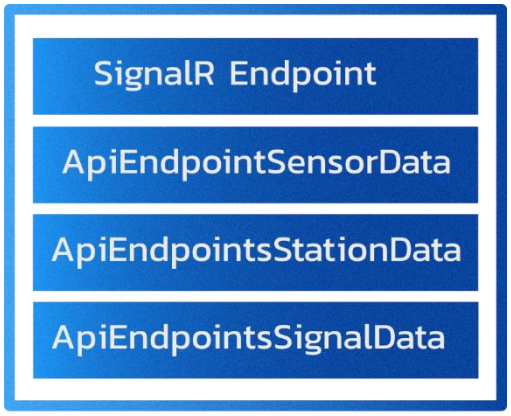
На следващите две фигури може да видите пример за “storedProcedure” както и C# кода, който се използва за извикването ѝ.

```
create procedure dbo.spSensorData_AddEntrySensorReadings
  @readingID varchar(255),
  @originID varchar(255),
  @payload text,
  @timeOfCapture datetime
as
begin
    insert into sensordata (readingID,originID,payload,timeOfCapture)
    values (@readingID,@originID,@payload,@timeOfCapture);
end
```

```
public void AddEntrySensorReading(DbModel_SensorReadingEntry entry)
{
    using (IDbConnection connection = new SqlConnection(connectionString))
    {
        connection.Execute(
            sql:"dbo.spSensorData_AddEntrySensorReading @readingID, @originID, @payload, @timeOfCapture",
            param:new
            {
                readingID = entry.readingID,
                originID = entry.originID,
                payload = entry.payload,
                timeOfCapture = entry.timeOfCapture
            });
    }
}
```

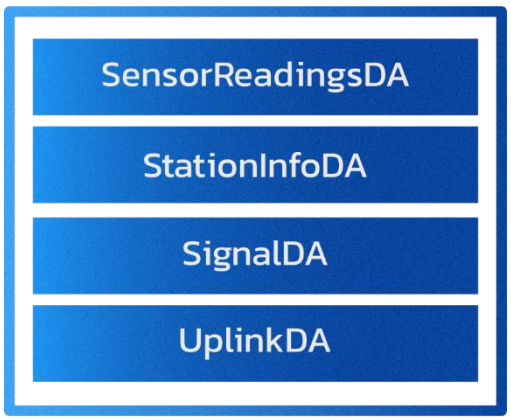
b. Уеб Сървър

Уеб сървъра е реализиран на C# използвайки Blazor Server App. Това е вид проект от семейството на ASP.NET. В тази точка ще представим основните компоненти на сървъра, както и най-важните последователността на най-важните процеси.



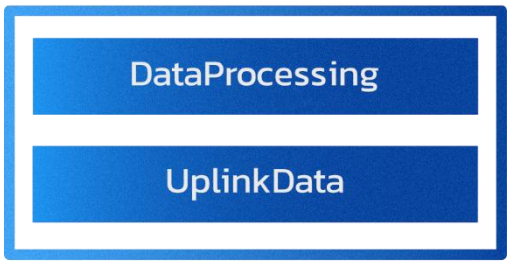
Enpoints

Endpoints – тази група компоненти са отговорни за връзката с LoRaNet клиентите. Разделени са в 4 логически групи. За SignalR Endpoint-а е създаден отделен клас SocketHub.



Database Logic

Database Logic – чрез тези класове се осъществяват всички операции с базата от данни. Отново са разделени в 4 логически групи.
DA = Data Access



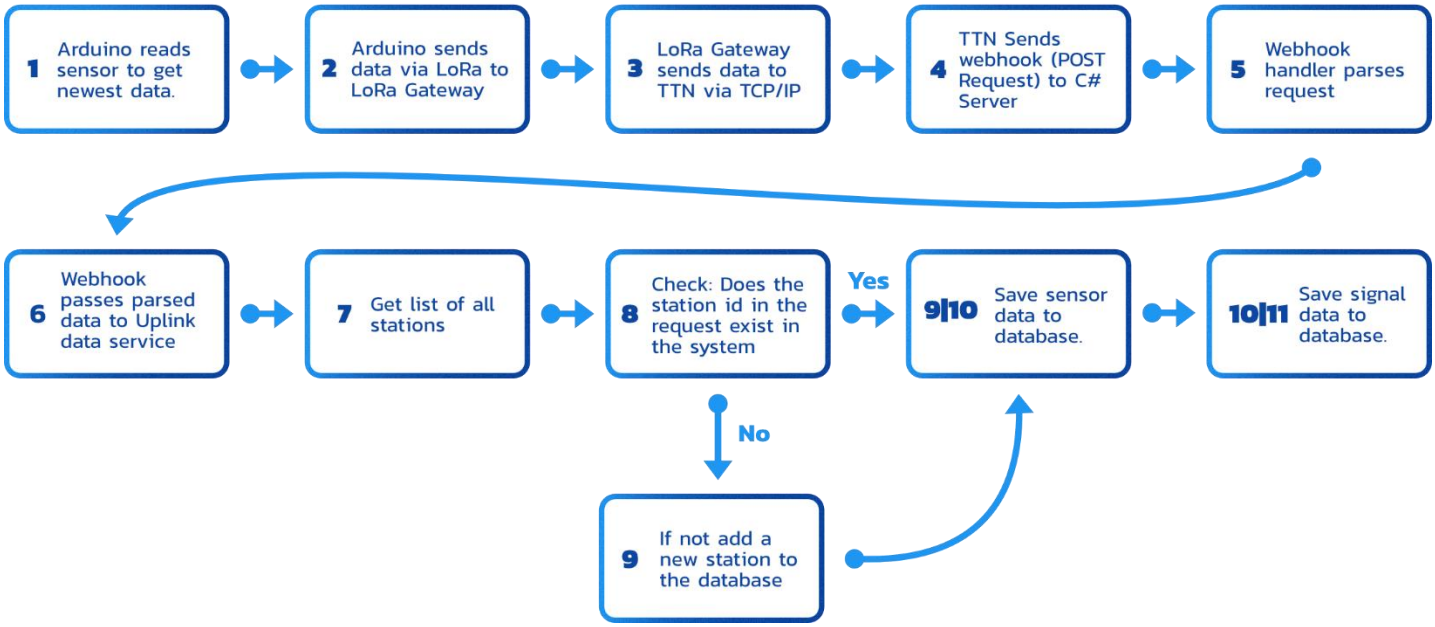
Services

Services – тези модули съдържат функции/услуги, които се използват от останалите модули. Пример за това е оформяне на данните, които за взети от базата от данни.

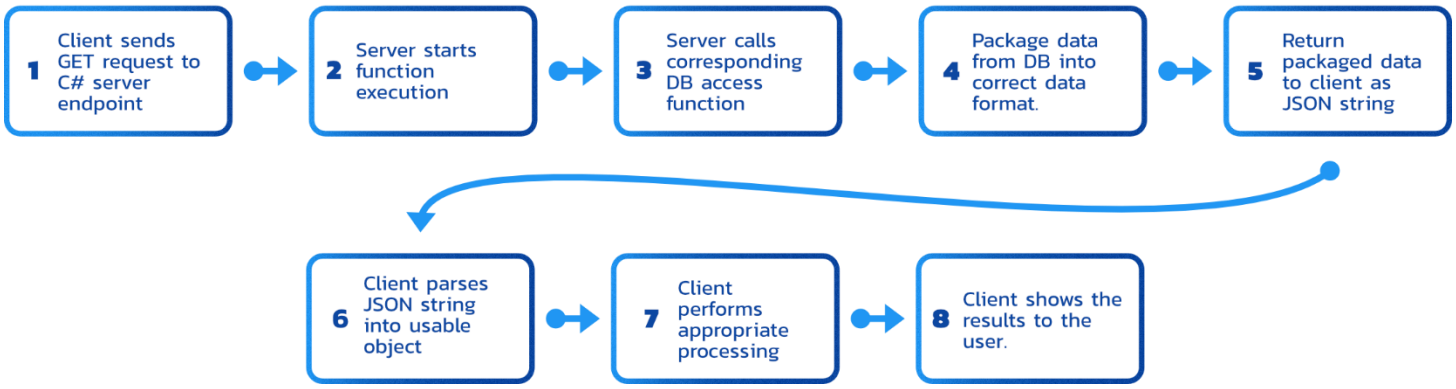
Webhook Handler

Webhook Handler – Този модул е начина по който сървъра се свързва с The Things Network. Той приема POST заявката, десериализира данните и ги предава на останалите компоненти.

New sensor reading flow



Generic client GET request to API



c. Сензорни станции

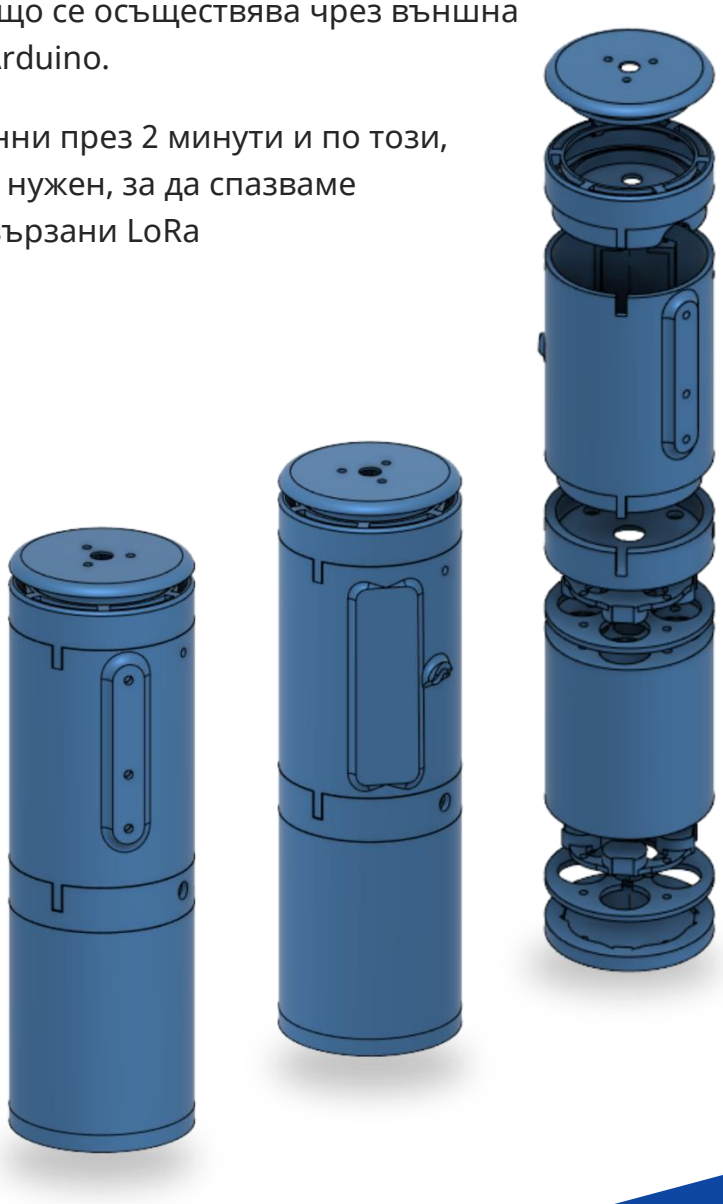
Сензорните станции са устройства разработени специално за този проект. Базирани са на микроконтролера Arduino MKR 1300/1310 LoRa Connectivity. Останалата структура е направена в програмата Fusion 360 и 3D принтирана.

Целта на изработката на тези устройства е да се покаже възможно най-добре как едно реално решение би изглеждало и работело.

Фърмуера за станциите е написан на C++ използвайки Arduino IDE, като за работа със сензорите използваме външни библиотеки предоставени от производителите на съответните сензори.

Работата с LoRa модема също се осъществява чрез външна библиотека създадена от Arduino.

Всяка станция изпраща данни през 2 минути и по този, като този времеви лимит е нужен, за да спазваме правилата за излъчване свързани LoRa честотата (868MHz).



d. Уеб приложение

Уеб приложение е разработено на typescript като е използван React. За стилизиране е използвана библиотеката Material UI, а за създаване на графиките за различните измервания използва библиотеката @devexpress/dx-react-chart. Станиците на приложението са следните: „Home”, „Stations” и „History”.

```
export default function App() {
  return (
    <BrowserRouter>
      <ServerSocketProvider>
        <CssBaseline />
        <Header />
        <Routes>
          <Route path="/home" element={<Home />} />
          <Route path="/history" element={<History />} />
          <Route path="/stations" element={<Stations />} />
          <Route path="*" element={<Navigate replace to="/home" />} />
        </Routes>
      </ServerSocketProvider>
    </BrowserRouter>
  );
}
```

„ServerSocketProvider” съдържа логиката за свързване със сървъра посредством библиотеката SignalR. Цялата логика, както и връзката е изнесена в един context (използван е useContext react hook), за да може да се използва в повече от една страница.

„Home” страницата взима средната стойност от последните измервания на температурата и влажността, като се обновява всяка секунда при изпращане на данните от сървъра чрез сокета.

„Stations” страницата показва всички станции, които се намират в системата. Потребителят може да види всички типове измервания, които станцията прави и да избере едно от тях и да вижда последните данни от това измерване, отново те се обновяват използвайки сокета. Също така е възможно да се видят данните от измервания в определен период от време, който може да се зададе чрез Start Date и End Date под формата на линейна графика. За визуализирането на графиките създадохме компонент Charts, който да използва @devexpress/dx-react-chart за създаване на нужните графики.

```
<Charts
  startDate={startDate!}
  endDate={endDate!}
  typesOfMeasurement={selectedMeasurements}
  stations={stationList!}
/>
```

„History" страницата дава възможност да се видят всички измервания в даден период време. Потребителят има възможност да избере повече от един тип измерване и на графиката се визуализират данните от всички станции, които поддържат този тип измервания.

e. Мобилно приложение

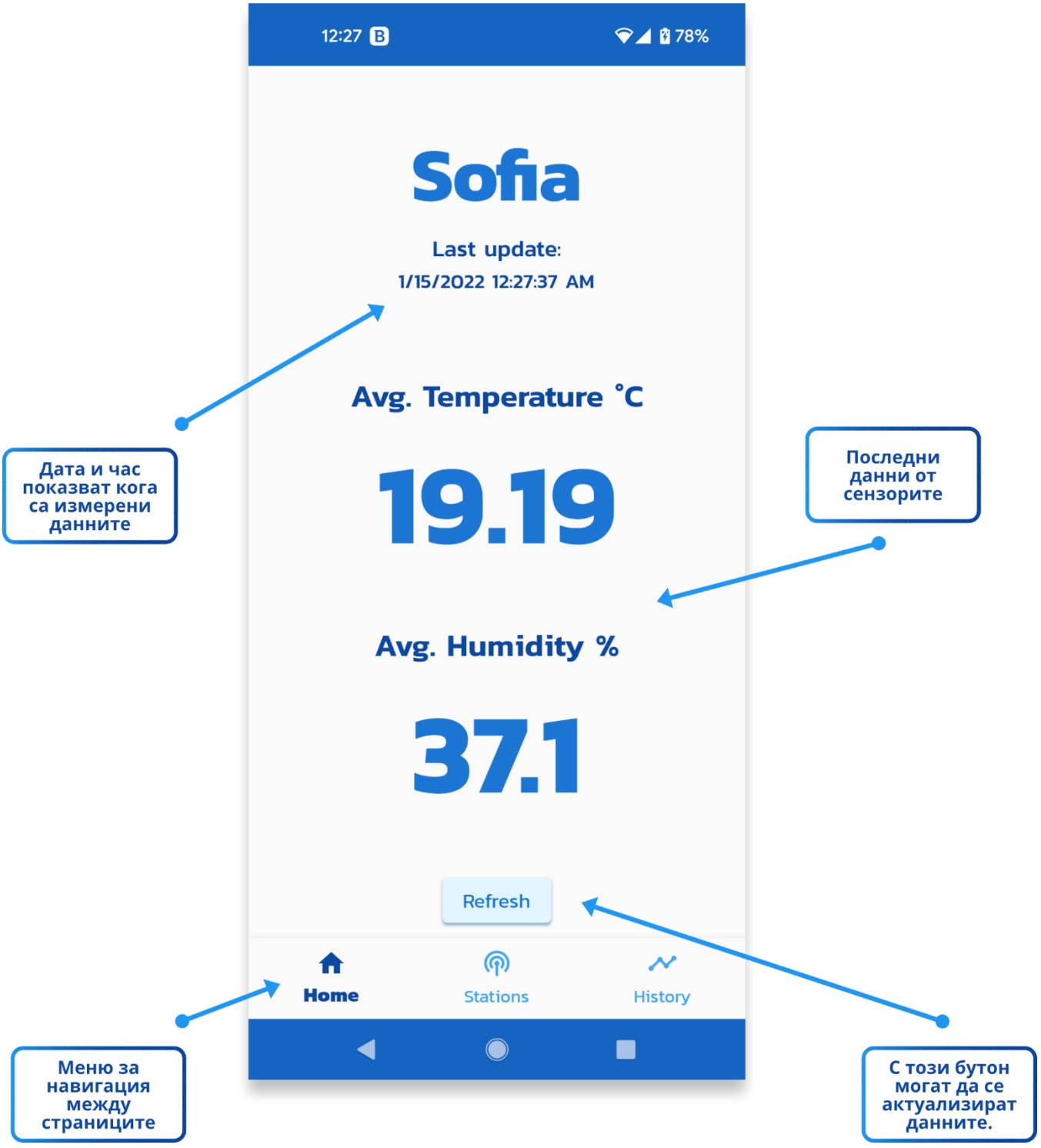
Мобилното приложение е разработено с Flutter. Framework разработен от Google за създаване на мултиплатформени приложения. Основните страници на приложението са „**Home**" и „**Stations**".
„**Home**" показва средната температура за града в който системата работи. Температурата се изчислява като се вземе последното измерване на температура от всички станции и се направи средноаритметично. Влажността се изчислява по същия начин.

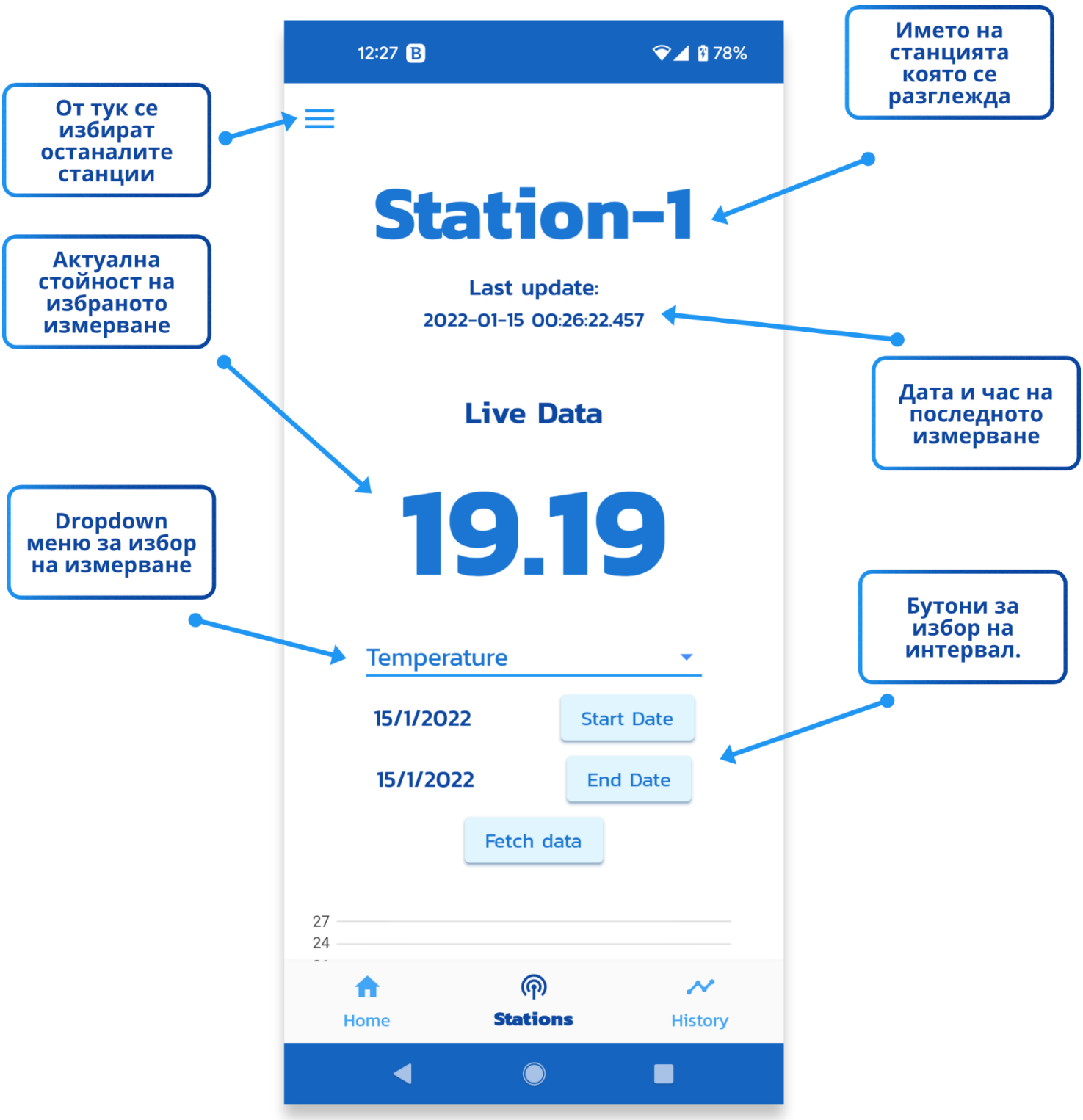
„**Stations**" страницата дава възможност на потребителите да видят измерванията от една специфична станция. Има възможност да се видят последните данни, както и минали данни в даден интервал от време, който може да се зададе от потребителя. Данните от миналото се показват в графика тип линия. От dropdown менюто потребителя може да види какви други измервания прави дадената станция и при избиране на елемент от списъка се зареждат съответните данни.

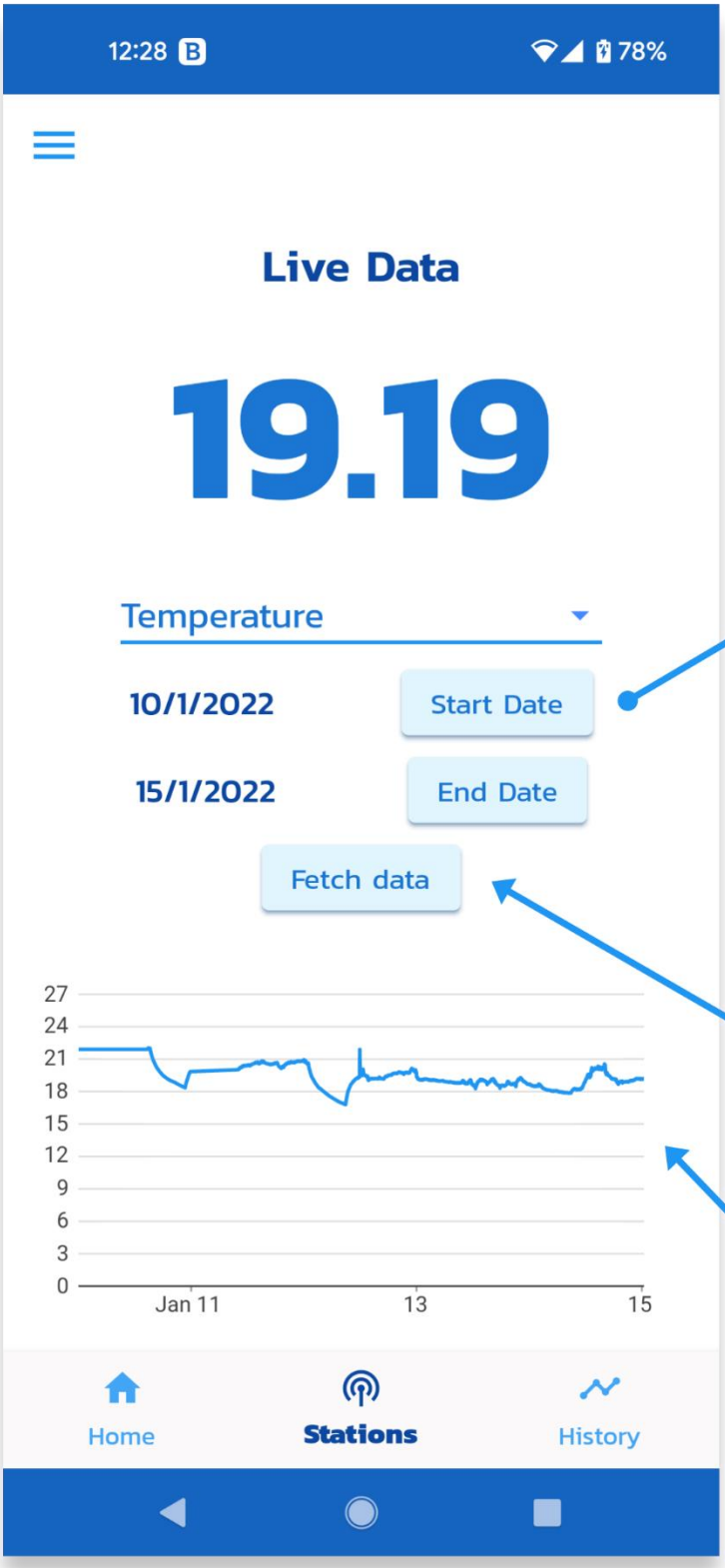
Мобилното приложение взима данните от сървъра единствено чрез REST API.

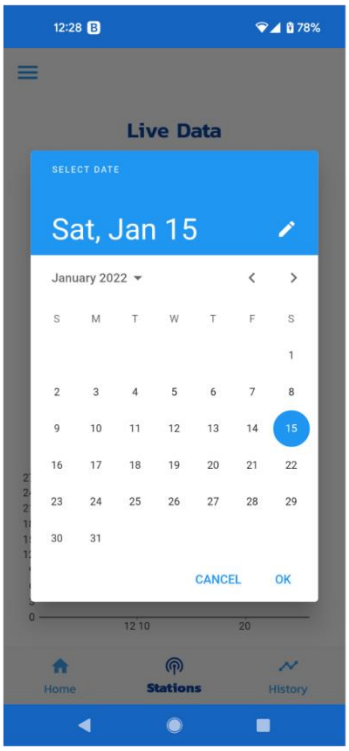

Sofia
Last update:
1/15/2022 12:27:37 AM
Avg. Temperature °C
19.19
Avg. Humidity %
37.1
Refresh
Stations
History
Home
Station-1
Last update:
2022-01-15 00:26:22.457
Live Data
19.19
Temperature
Start Date
End Date
Fetch data
15/1/2022
15/1/2022
Live Data
19.19
Temperature
10/1/2022
15/1/2022
Start Date
End Date
Fetch data
Jan 11
Live Data
SELECT DATE
Sat, Jan 15
January 2022
Live Data
SELECT DATE
Sat, Jan 15
Enter Date
01/15/2022
CANCEL
OK
End Date
Fetch data
15/1/2022

UI DEMO


12:27
78%
Sofia
Last update:
1/15/2022 12:27:37 AM
Avg. Temperature °C
19.19
Avg. Humidity %
37.1
Refresh
Home
Stations
History
Дата и час показват кога са измерени данните
Последни данни от сензорите
Меню за навигация между страниците
С този бутон могат да се актуализират данните.


От тук се избират останалите станции
Актуална стойност на избраното измерване
Dropdown меню за избор на измерване
Името на станцията която се разглежда
Дата и час на последното измерване
Бутони за избор на интервал.
12:27
78%
Station-1
Last update:
2022-01-15 00:26:22.457
Live Data
19.19
Temperature
15/1/2022
Start Date
15/1/2022
End Date
Fetch data
27
24
Home
Stations
History

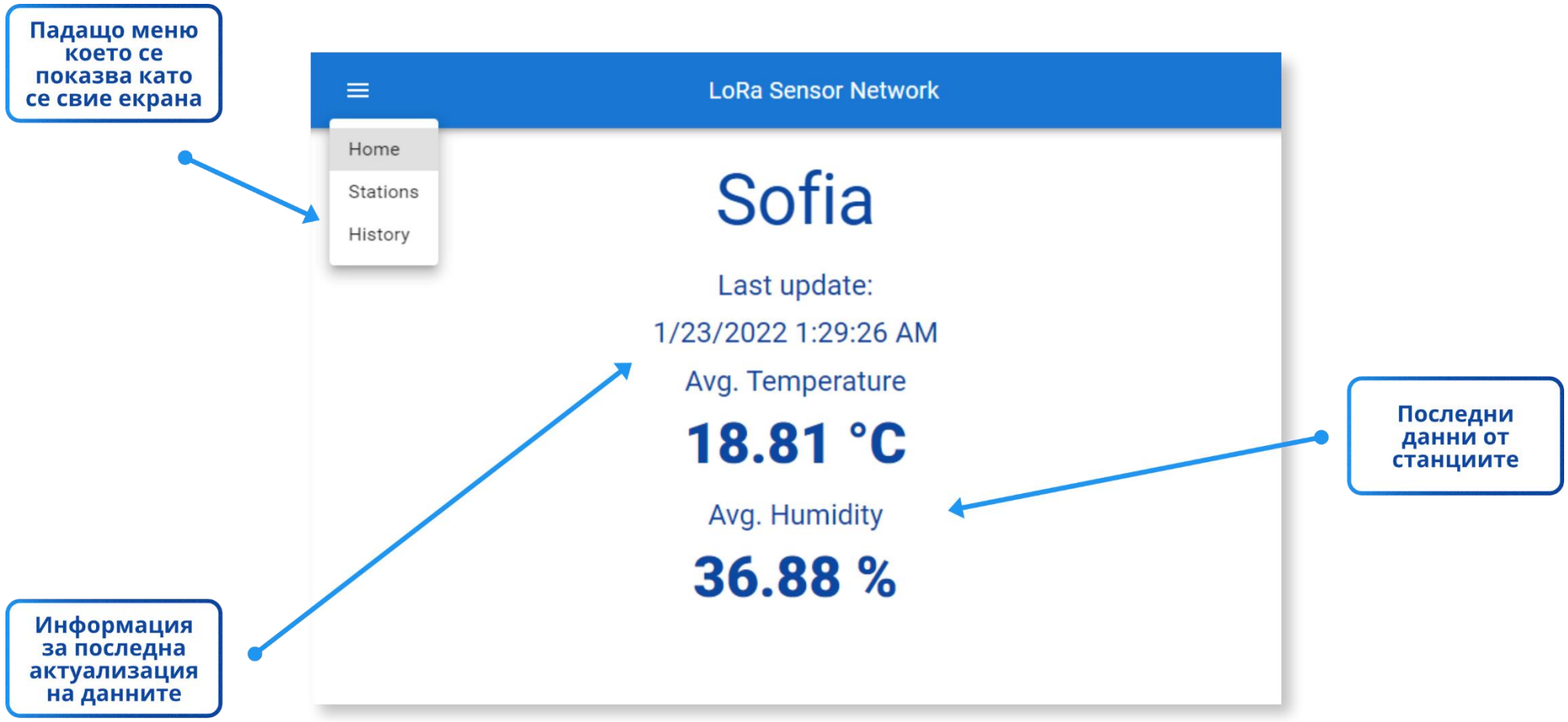


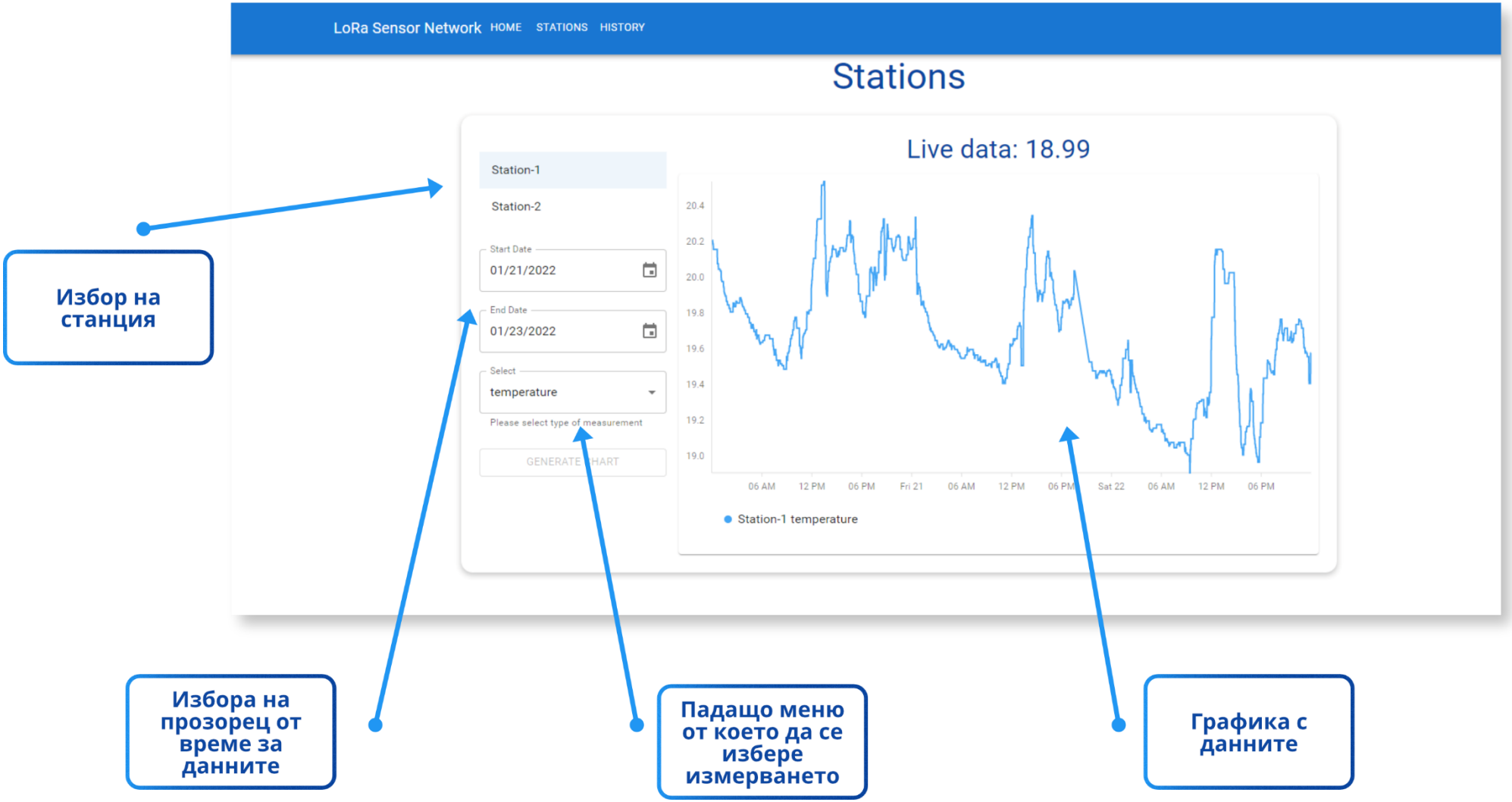


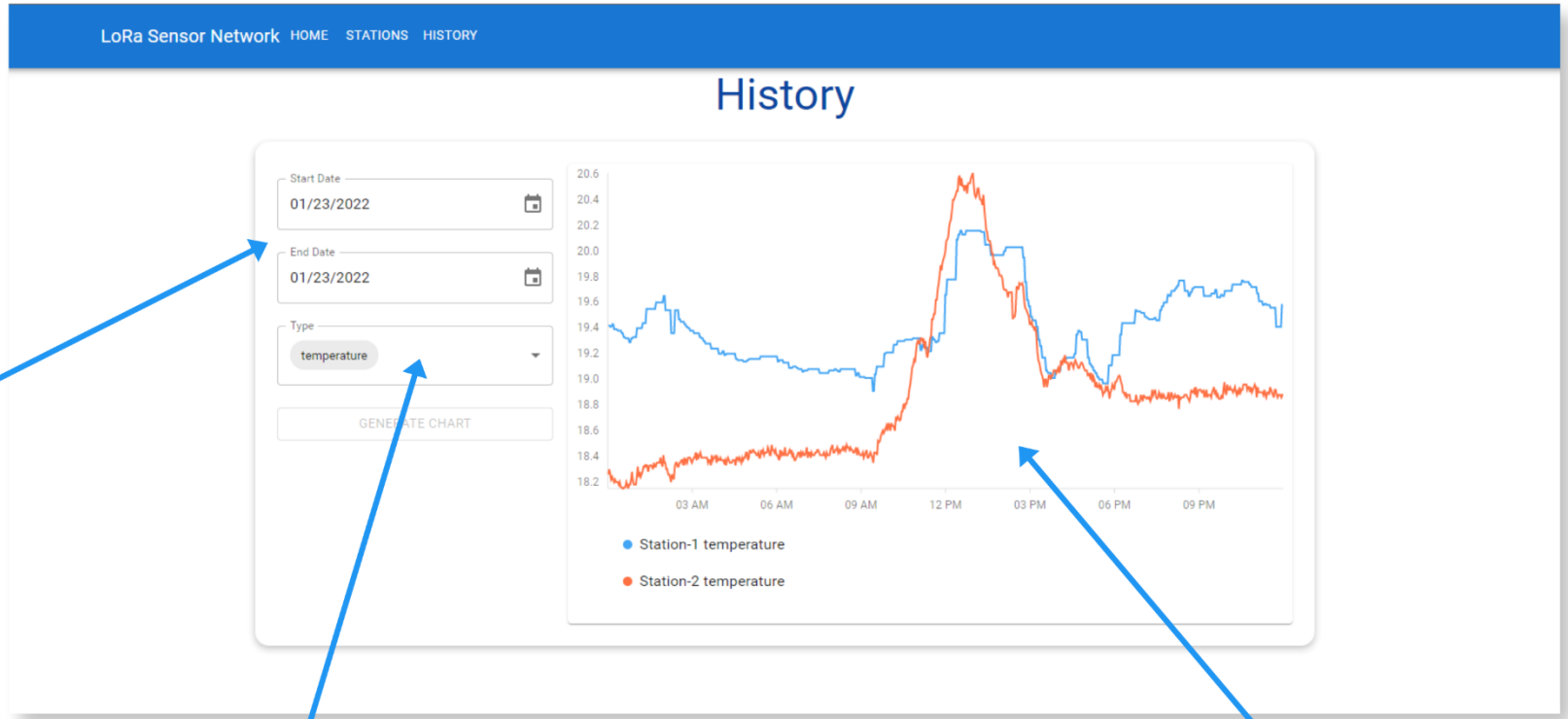
Меню за избиране на дата

Бутон за зареждане на данните

Line Chart с данни от зададения интервал


Падащо меню което се показва като се свие екрана
LoRa Sensor Network
Home
Stations
History
Sofia
Last update:
1/23/2022 1:29:26 AM
Avg. Temperature
18.81 °C
Avg. Humidity
36.88 %
Последни данни от станциите
Информация за последна актуализация на данните


LoRa Sensor Network HOME STATIONS HISTORY
Stations
Station-1
Station-2
Start Date
01/21/2022
End Date
01/23/2022
Select
temperature
Please select type of measurement
GENERATE CHART
Live data: 18.99
20.4
20.2
20.0
19.8
19.6
19.4
19.2
19.0
06 AM
12 PM
06 PM
Fri 21
06 AM
12 PM
06 PM
Sat 22
06 AM
12 PM
06 PM
Station-1 temperature
Избор на станция
Избора на прозорец от време за данните
Падащо меню от което да се избере измерването
Графика с данните



Избор на прозорец за данните

Избор на измервания които да се показват

Графика

5. Внедряване на системата

a. Внедряване на базата от данни

Базата от данни се хоства на платформата SmarterASP.NET. Работа с хостваната база от данни се осъществява чрез SQL Management Studio.

В процеса на разработка заявките се пазят в отделни файлове. В процеса на подготовка за внедряване, всичките заявки се събират в един файл, след което се изпълняват наведнъж на production базата.

След внедряването на базата от данни, **appsettings.json** файла на сървъра се променя да съдържа правилния **connection string** за връзка с production базата от данни.

b. Внедряване на сървъра

Сървъра също се хоства на платформата SmarterASP.NET. Внедряването се прави от Visual Studio от „Publish" опцията на проекта. Използва се автоматичен процес на внедряване, който използва конфигурационен файл, който се сваля от сайта на доставчика на хостинг услугата.

След процеса на публикуване на сървъра в хостинг средата, с инструмента „SoapUI" се тестват API endpoints, за да се потвърди че всичко работи както се очаква.

Създаден е Dockerfile, който съдържа нужната конфигурация за създаване на docker контейнер, с чиято помощ може да се качи и на друго място или да се вдигне втора инстанция на сървъра при нужда.

c. Внедряване на фърмуера

Внедряването на фърмуера се осъществява с програмата за разработване (Arduino IDE). Автоматично се компилира кода и се инсталира на микроконтролера.

Верификацията, че всичко работи правилно, се реализира чрез малък OLED екран, който първоначално изписва до къде е стигнал кода в процеса на инициализация и след това редовно се актуализира статуса на станцията.

d. Внедряване на клиентските компоненти

Внедряването на мобилното приложение е свързано само с публикуването му в Google Play Store за да е достъпно до крайните потребители.

За внедряването на React клиента може да се използва вече генерираната Build директория или да се създаде Docker контейнер.

6. Разпределение на дейности

Дейност/Задача	Никола	Ивайло
Документация точка 1	●	
Въведение		●
Използвани технологии и платформи	●	●
Описание на архитектурата	●	●
Реализация на системата	●	●
Flutter UI Demo	●	
React UI Demo		●
Внедряване на системата	●	●
Разпределение на дейности	●	
Приложения и потенциал за развитие		●
Работа по станциите	●	
Настройване на The Things Network	●	
Работа по C# сървъра	●	●
API Endpoints	●	
Signal R Endpoints		●
База от данни	●	●
React клиент		●
Flutter клиент	●	

7. Приложения и потенциал за развитие

Разработената система е добра основа за по-сложи и големи системи. Системата има потенциал да стане по-използваема при добавяне на функционалност за регистриране и логване на потребители и свързване на конкретни сензори с профилите на тези потребители.

Подобни системи могат а намерят приложение в много отрасли като земеделие, строителство, охрана, управление/администриране на сгради, поддръжка на инсталации като електрическа или ВиК мрежи, умни домове/градове.

За всеки един от тези сектори може да се разработят станции, които отговарят на съответните изисквания на отрасъла. След което лесно може да се адаптират клиентите и сървъра да работят с новите данни.

Възможно бъдеще развитие за тази система е да се добавят още сензори към станциите и да се използва за наблюдение на статуса на домашна градина.

Линкове към LoRaNet

https://github.com/NikolaTotev/LoRa_Sensor_Network - Github repo

http://nikolatotev-001-site2.ctempurl.com/ - React клиент

https://nikolatotev-001-site1.ctempurl.com/ - LoRaNet Server