

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

---

# **Internet vecí v našich životoch [IoT]**

## **Inžinierske dielo – moduly systému**

---

**Tím:** č. 20  
**Pedagogický vedúci tímu:** Ing. Tomáš Kováčik, PhD  
**Členovia tímu:** Barbora Čelesová, Tomáš Koreň, Jakub Pullmann, Michal Puškáš, Matúš Sosňak, Peter Štofaňák, Jozef Val'ko,  
**Akademický rok:** 2017 / 2018

## Obsah

1	Úvod.....	1
2	Modul – Arduino.....	2
2.1	Analýza.....	2
2.1.1	Technológia Sigfox .....	2
2.1.2	Analýza frekvencie zberu a odosielania údajov .....	3
2.1.3	Analýza existujúcich riešení.....	5
2.1.4	Arduino platforma .....	6
2.1.5	Analýza senzorov .....	8
2.1.6	IoT zariadenie.....	9
2.1.7	Analýza merania stavu batérie.....	10
2.1.8	Analýza posielaných správ .....	11
2.2	Návrh zapojenia.....	12
2.3	Implementácia .....	13
2.3.1	Modul merania teploty a vlhkosti.....	13
2.3.2	Modul detegovania prevrátenia/odcudzenia.....	13
2.3.3	Modul merania stavu batérie .....	14
2.3.4	Sigfox modul .....	14
2.4	Testovanie .....	15
2.4.1	Testovanie merania fyzikálnych veličín pomocou Arduina .....	15
2.4.2	Testovanie SigFox modemu .....	16
3	Modul – Server.....	17
3.1	Analýza.....	17
3.1.1	Carriots .....	17
3.1.2	Analýza Sigfox callbackov.....	19
3.1.3	SigFox Cloud.....	20
3.2	Návrh.....	21
3.2.1	Carriots service.....	21
3.2.2	Použitie Carriots API.....	22
3.2.3	Databáza PostgreSQL.....	23
3.2.4	Zabezpečenie spojenia.....	24
3.3	Implementácia .....	25
3.3.1	API pre komunikáciu medzi používateľom a serverom Včelička.....	25
3.4	Testovanie .....	31
3.4.1	Testovanie API na získanie zariadení používateľa.....	31
3.4.2	Testovanie API na získanie posledného merania .....	31

4	Modul – Webová stránka produktu .....	32
4.1	Návrh.....	32
4.1.1	Titulná stránka.....	32
4.1.2	Stránka po prihlásení včelára.....	33
4.1.3	Stránka po prihlásení administrátora.....	34
4.2	Implementácia .....	34
4.2.1	Zabezpečenie komunikácie .....	34
4.2.2	Webová stránka .....	34
4.2.3	Podstránka kontakt .....	35
4.3	Testovanie .....	35
4.3.1	Testovanie zabezpečenia komunikácie medzi klientom a web stránkou produktu .....	35
4.3.2	Testovanie prihlásenia na webovej stránke .....	36
4.3.3	Testovanie registrácie používateľa.....	36
4.3.4	Testovanie funkčnosti webového rozhrania pre správu databázy na serveri Včelička..	37
5	Modul – Android.....	38
5.1	Návrh mobilnej aplikácie .....	38
5.1.1	Úvodné logo a prihlasovacia obrazovka aplikácie .....	38
5.1.2	Prehľad úľov a bočný panel aplikácie .....	38
5.1.3	Registrácia používateľa a chybové okno pri odhlasovaní .....	39
5.1.4	Objednávanie včelieho úľa a zobrazenie detailu úľa.....	40
5.2	Implementácia .....	40
5.2.1	Prihlasovacia stránka.....	40
5.2.2	Registračný formulár.....	41
5.2.3	Objednávka zariadenia .....	41
5.2.4	O projekte.....	42
5.2.5	Dôležité triedy .....	42
5.3	Testovanie .....	43
5.3.1	Testovanie „Main Activity“ .....	44
5.3.2	Testovanie „Login Activity“ .....	45
5.3.3	Testovanie „HiveDetails“ .....	46
5.3.4	Testovanie registrácie používateľa v Android aplikácii .....	47
5.3.5	Testovanie vytvorenia objednávky používateľom.....	48
6	Bibliografia.....	49

# 1 Úvod

Účelom tohto dokumentu je popísať moduly systému, ktoré vznikli počas našich piatich šprintov. Moduly sme rozdelili na Arduino, Server, Webová aplikácia produktu a Android aplikácia. Každý jeden modul obsahuje svoju analýzu, návrh riešenia, implementáciu a testovanie, ktoré boli vykonané.

Prvý modul – Arduino obsahuje aj celkový pohľad na technológiu sigfox, frekvenciu zberu údajov, existujúce riešenia ale následne aj analýzu samotného arduina, použitých senzorov, IoT zariadení a analýzu posielaných správ.

Modul Server obsahuje analýzu Carriots a tiež Sigfox callbackov potrebných na získavanie údajov zo Sigfox cloudu. Návrh a implementácia sú venované našim API, ktoré slúžia ako brána medzi používateľom a našim serverom.

Modul webovej stránky je venovaný opisu webovej stránky produktu, čiže stránky pre včelárov. Návrh obsahuje mockupy stránky a v implementácii sú popísané jednotlivé časti, ktoré boli implementované počas piatich šprintov.

Modul Andorid opisuje návrh, implementáciu a testovanie aktivít v Android aplikácii. Návrh bol podobne ako pri webovej stránke vytvorený pomocou mockupov. Implementácia popisuje funkcionality jednotlivých aktivít Android aplikácie.

## 2 Modul – Arduino

Táto kapitola obsahuje analýzu, návrh, implementáciu a testovanie spojené s modulom Arduino a technológiou Sigfox.

### 2.1 Analýza

Obsahom prvého šprintu bolo zanalyzovanie si oblasti, ktorou sa ideme zaoberať. Ako prvé sme sa zoznámili s technológiou Sigfox, ktorá je ďalej popísaná. V nadväznosti na to, sme sa rozhodli ako hardvérový prostriedok využiť Arduino. Inšpiráciou nám boli už existujúce riešenia, ktorú sme v skratke tiež spomenuli pri tomto module.

#### 2.1.1 Technológia Sigfox

Jedná sa o sieť, pomocou ktorej môžu IoT zariadenia medzi sebou komunikovať. Sigfox využíva na komunikáciu úzke frekvenčné kanály (UNB) a bunky tejto siete umožňujú pokryť oveľa väčšie územie než základné stanice pri GSM. Táto technológia je vybudovaná na bunkovom systéme podobne ako GSM a využíva topológiu hviezda. Dosah každej základňovej stanici je cca 200 km pri priamej viditeľnosti, v teréne kde je viacero prekážok je to 50 km a v lesových oblastiach je to cca 2-10 km. Sieť pracuje vo frekvenčnom pásme 868 MHz v Európe a 902 MHz v USA. Najvyšší vysielač výkon je povolený 25 mW.



Obrázok 1 Architektúra siete Sigfox

Medzi základné prvky Sigfox architektúry patria:

- Sensory – slúžia na meranie určitej veličiny. Sensory môžu byť nastavené tak, že buď posielajú dáta do jedného hlavného senzora a až potom ďalej alebo posielajú údaje priamo každý zvlášť.
- Sigfox stanice – slúžia na zaistenie dostupnosti siete Sigfox. Existuje ich viacero a každá stanica pokrýva určitú oblasť.
- Sigfox úložisko – sa používa na ukladanie používateľských dát.

- Používateľské zariadenia – prijímajú dáta zo senzorov a obvykle z nich robia buď nejaké štatistiky. Všeobecne slúžia ako zobrazovacie zariadenia pre senzory.

V tejto technológii je implementovaný protokol, ktorý umožňuje prenášať malé objemy dát a má nízku spotrebu energie, čo je pre IoT ideálne. Každý paket je veľký 12 bajtov a každé koncové zariadenie využívajúce technológiu Sigfox by malo poslať denne maximálne 140 takýchto paketov, čo predstavuje jednu správu za približne 10 minút a celkovú dennú kapacitu 1680 bajtov. Paket nemá žiadnu predpísanú štruktúru. Na adresovanie zariadení sa využívajú interné identifikátory, ktoré emituje certifikačná autorita, pričom zariadenie sa zaregistruje u prevádzkovateľa pri aktivácii služby. Problémom pri tomto protokole je negarantovanie prijatia správy. Aby bola ale pravdepodobnosť doručenia čo najvyššia, tak sa správa posiela vždy trikrát a to zakaždým na inú náhodne vybranú frekvenciu. Vysielač potom prijatú správu pošle na server prevádzkovateľa a ten skontroluje jej validitu. Následne sa správa sprístupní klientovi, ktorému je určená buď cez REST API, prípadne sa správa zašle na server klienta cez http callback.

Na Slovensku poskytuje technológiu Sigfox len jediný mobilný operátor, ktorým je SimpleCell Networks Slovakia. Táto sieť je určená výhradne pre komunikáciu medzi IoT zariadeniami.

### 2.1.2 Analýza frekvencie zberu a odosielania údajov

#### Čo máme k dispozícii:

- Uplink (zo zariadenia do Sigfox siete)
  - max 140 správ za deň (1 správa/10 min)
  - v správe môžeme poslať 12 bajtov dát
  - 3x poslaná každá správa
  - žiadne overenie prijatia
- Downlink (zo Sigfox siete na zariadenie)
  - každý paket 8 bajtov
  - každá správa má k využitiu 8 bajtov dát

Posielanie údajov je značne obmedzené maximálnym počtom odoslaných správ za deň (140) a veľkosťou jednej správy (12 B). Frekvencia zberu je obmedzená len veľkosťou pamäte Arduina. Odosielať môžeme konkrétnu hodnotu merania alebo priemer viacerých meraní (opakujúcich sa hneď po sebe, alebo po určitom časovom intervale napr. 10 min) pre zvýšenie presnosti merania. Hodnotu z akcelerometra nie je potrebné posielat' pravidelne, ale stačí keď ju odošleme len keď dôjde k zmene (prevrhnutie úľa). Na hodnotu z akcelerometra stačí 1 bit (prevrhnutý/neprevrhnutý). Či je úľ prevrhnutý by vyhodnocovalo Arduino na základe meraní z akcelerometra. Niektoré namerané hodnoty môže byť potrebné posielat' častejšie ako iné. Včelár (koncový používateľ), by si mohol v systéme vybrať, ktoré hodnoty potrebuje merať častejšie (prispôsobenie podľa osobných preferencií včelára). Treba nechať „rezervu“ (napr. 20 správ) pre odosielanie správ z meraní, ktoré prekročili hraničné hodnoty, keďže tieto

dáta sú pre včelára dôležitejšie ako tie čo sa posielajú pravidelne. Keďže doručenie správy v Sigfox nie je zaručené, môžeme posielat' informácie o meraniach, ktoré prekročili hraničné hodnoty viac krát po vybranom časovom intervale (napr. 2x po 10 min). Hraničné hodnoty sú pre včelára dôležitejšie ako tie čo sa posielajú pravidelne. K dispozícii máme uplink 4 správy/deň, ktoré sa dajú použiť podľa potreby. Napríklad na odoslanie príkazu na zmenu frekvencie merania vybranej veličiny, alebo na žiadosť o odoslanie vybranej veličiny (ak chce včelár vedieť nejakú z hodnôt okamžite).

## Monitorovanie teploty

Pomocou monitorovania teploty dokážeme zistiť:

- Vhodný čas ošetrenia včelstva proti škodcom (Klieštik včelí)
- Prítomnosť včelej matky
- Oplodnenie včelej matky

Ako hraničný stav sme zvolili teplotu vo vnútri úľa 35°C.



Obrázok 2 Graf teploty v úli

## Monitorovanie váhy:

Bežný úľ váži 20-25kg na jar a až 70-80kg v jesenných obdobiach. Pomocou monitorovania hmotnosti dokážeme zistiť:

- V znáškovom období množstvo medu
- V mesiacoch máj a jún je nutná váha pre zistenie pridávanie medníkov, aby včely mali priestor ukladať med. Ak včely zistia, že majú málo miesta, dochádza k rojeniu a môže sa stať, že si včely vychovávajú novú včeliu matku a polka včelstva si odíde hľadať nové obydlie a zoberie so sebou aj časť medu
- V júli prichádza príprava včelstva na zimu. Po vytočení medu musíme včelám vrátiť zásoby na zimu, aby prežili do jari
- Pomocou váhy zistíme, že už znáškové obdobie skončilo a že môžeme pridať zásoby vo forme cukrového roztoku

### **Monitorovanie vlhkosti:**

Keď včely naplnia úl medom, začnú sa zhlukovať a zvýši sa vlhkosť. Pri určitom stupni vlhkosti nastane vhodný okamžik pre vytočenie medu. Počas obdobia chovu mladých včiel sú stredné úrovne vlhkosti v úli medzi 50 a 60 percentami. Ak je vlhkosť príliš nízka, vajcia nebudú vyliahnuté a larvy sa vysušia. Ak je vlhkosť príliš vysoká, včely sú náchylnejšie na hubové choroby.

### **2.1.3 Analýza existujúcich riešení**

V tejto kapitole sme sa zaoberali existujúcimi riešeniami, ktoré sú podobné ako náš projekt. Z každého riešenia sme sa sčasti inšpirovali a pridali podobnú funkcionality do nášho návrhu.

#### **Arduino Beehive monitor**

Autor Marc Curtis vytvoril monitorovací kit včelieho úľa na základe Arduina, senzorov a posielanie dát cez GSM sieť. Na zber dát použil platformu Xively. Jeho cieľmi boli:

- upozornenia cez SMS alebo e-mail ak hodnoty prekročia hraničné hodnoty
- merať hmotnosť úľa na zisťovanie množstva medu a množstva včiel v úli
- monitorovanie vstupu do úľa
- monitorovať frekvenciu bzučania v úli
- detegovať pohyb úľa, vietor, prevalenie úľa alebo vandalizmus

Základným kameňom jeho projektu bolo Arduino Duo a teplotný a vlhkosťný senzor DHT22. Keďže úle sú mimo dosahu akéhokoľvek Wi-Fi smerovača, bolo potrebné zabezpečiť posielanie dát. To vyriešil GPRS modulom pre Arduino. Dáta boli posielané cez GSM/GPRS sieť do Xively. Ďalším cieľom bolo zabezpečiť postačujúce napájanie. Všetky moduly fungujú perfektne pokiaľ sú pripojené k pevnému napájaniu. Na druhej strane, 9V batéria vydržala v jeho prípade 2 hodiny, kým stratila schopnosť poskytovať postačujúce množstvo elektrickej energie. Tento problém vyriešil solárnym panelom a lítiovou batériou, ktoré dokopy vytvárajú teoreticky sebestačnú energetickú jednotku. Na stránke projektu [1] má aj výsledky meraní a z nich vyplýva, že včely si v úli udržiavajú priemernú teplotu okolo 34°C.

#### **Bee Smart TM**

Beebot [2] je vzdialená diagnostická a monitorovacia stanica pre každý včelí úl. Zhromažďuje informácie o zdraví úľa a produktivite pomocou nameranej teploty, vlhkosti a akustickej analýze zvuku každých 15 minút. Pri riešení využili bezdrôtovú váhu HiHive, ktorá sleduje zmenu hmotnosti a poskytuje maximálny prehľad úľa. Bola špeciálne navrhnutá pre úle a náročné podmienky. Na jedno nabitie vydrží až 3 mesiace. Bee Smart tiež sleduje pohyb a informuje včelárov o prípadnom



premiestnení alebo prevrátení úľa. Riešenie poskytuje záznamy v reálnom čase, šetrí čas včelárov, zjednodušuje prevádzku, znižuje náklady a napomáha zvýšiť výnos.

Všetky informácie sa zostavia do správy a posielajú sa na cloud tri krát denne. Samotné dáta prechádzajú sériou algoritmov a upozorňujú používateľov v prípade potreby. Týmto je zabezpečené minimalizovanie počtu manuálneho sledovania situácie zo strany včelárov. Beebot využíva na komunikáciu Wi-Fi. Komunikačný modul je aktívny iba 8 minút z 24 hodín, čo nespôsobuje význačné rušenie včiel. Zariadenie je napájané z batérie, ktorá vydrží po dobu 6 mesiacov.

## **ProBee**

Ide o súbor zariadení pre elektronické monitorovanie včelstiev, prezentáciu výsledkov a ich vyhodnocovanie, spoločne s evidenciou všetkých aktivít včelára. Systém sa skladá z niekoľkých samostatných častí a to napríklad zvukový a teplotný senzor, úľová váha, vibračný senzor s GPS vysielateľom na sledovanie polohy a úľová kamera. Všetky tieto časti zasielajú informácie na server, kde sú ukladané a vyhodnocované. Takisto obsahuje možnosť zasielania upozornení v prípadoch, že sa napríklad hranične zmení teplota alebo váha. Systém ProBee je napojený na výkonné počítačové centrum IBM, kde sú analyzované akustické merania. Výsledky sa následne vracajú naspäť do ProBee, kde sú názorným spôsobom zobrazené alebo zaslané používateľovi [3].

### **2.1.4 Arduino platforma**

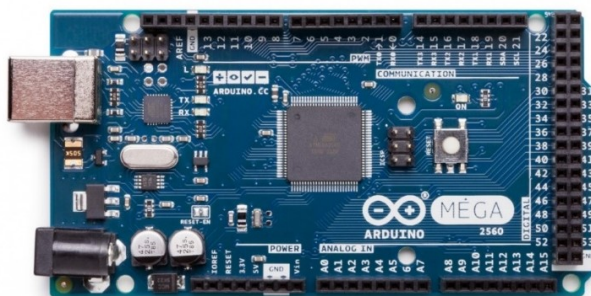
Je to voľne šíriteľná elektronická platforma založená na ľahko použiteľnom hardvéri a softvéri. Arduino dosky dokážu čítať vstupy - svetlo pomocou snímača, stlačené tlačidlo alebo Twitter správu - a zmeniť ho na výstup - aktiváciu motora, zapnutie LED, publikovanie niečoho online. Môžete povedať svojej doske, čo má robiť tým, že pošlete sadu inštrukcií na mikrokontrolér. Použijete programovací jazyk Arduino (založený na Wiring) a softvér Arduino (IDE) založený na Processing. V priebehu rokov bolo Arduino mozgom tisícov projektov, od každodenne používaných objektov po zložité vedecké nástroje. Na tejto platforme s otvoreným zdrojom sa zhromaždila celosvetová komunita tvorcov - študenti, umelci, programátori a odborníci, ktorých príspevky prispeli k neuveriteľnému množstvu prístupných poznatkov, ktoré môžu byť veľmi nápomocné pre začiatočníkov i odborníkov. Arduino sa narodilo v Ivrea Interaction Design Institute ako jednoduchý nástroj pre rýchle prototypovanie, zameraný na študentov bez znalostí z oblasti elektroniky a programovania. Akonáhle sa dostal do širšej komunity, Arduino doska sa začala meniť, aby sa prispôsobila novým potrebám a výzvam, rozšírila svoju ponuku od jednoduchých 8-bitových dosiek k produktom pre aplikácie IoT, prenosné, vnorené prostredia a 3D tlač. Všetky dosky Arduino sú voľne šíriteľné, umožňujú používateľom budovať ich nezávisle na sebe a nakoniec ich prispôbovať konkrétnym potrebám. Softvér je tiež voľne šíriteľný a rastie prostredníctvom príspevkov používateľov na celom svete [4].

Arduino sa používa v tisícoch rôznych projektov a aplikácií vďaka jednoduchému dizajnu. Softvér Arduino je ľahko použiteľný pre začiatočníkov, ale je dostatočne flexibilitný aj pre pokročilých

používateľov. Beží na systémoch Mac, Windows aj Linux. Učítelia aj študenti ho používajú na vytváranie nízko nákladových vedeckých nástrojov, na preukázanie zásad chémie a fyziky alebo na programovania a robotiku. Návrhári a architekti vytvárajú interaktívne prototypy, hudobníci a umelci ju používajú na inštaláciu a experimentovanie s novými hudobnými nástrojmi. Výrobcovia ju, samozrejme, používajú na výstavbu mnohých projektov vystavených napríklad v spoločnosti Maker Faire. Arduino je kľúčovým nástrojom naučiť sa nové veci.

Existuje mnoho ďalších mikrokontrolérov a platforiem mikrokontrolérov, ktoré sú k dispozícii pre fyzickú výpočtovú techniku. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard a mnoho ďalších ponúkajú podobnú funkčnosť. Všetky tieto nástroje zabalia komplikované detaily programovania mikrokontrolérov do ľahko použiteľného balíka. Arduino tiež zjednodušuje proces práce s mikrokontrolérmi. Výhody, ktoré prináša pre učiteľov, študentov a zainteresovaných amatérov v porovnaní s inými systémami:

- **Lacný** – dosky Arduino sú relatívne lacné v porovnaní s ostatnými platňami s mikrokontrolérom. Najlacnejšia verzia modulu Arduino môže byť zostavená ručne a dokonca aj predmontované moduly Arduino stoja menej ako 50 dolárov.
- **Univerzálny** – Softvér Arduino (IDE) pracuje na operačných systémoch Windows, Macintosh OSX a Linux. Väčšina mikrokontrolérových systémov je obmedzená na Windows.
- **Jednoduché a prehľadné programovacie prostredie** – Softvér Arduino (IDE) je pre začiatočníkov ľahko použiteľný, ale je dostatočne flexibilný, aby ho mohli aj pokročilí používatelia využiť. Pre učiteľov je to pohodlne založené na spracovateľskom programovacom prostredí, takže študenti, ktorí sa učia programovať v tomto prostredí, budú oboznámení s tým, ako funguje IDE Arduino.
- **Voľne šíriteľný softvér** – Softvér Arduino sa uverejňuje ako nástroje s otvoreným zdrojovým kódom, ktoré môžu skúsení programátori rozšíriť. Jazyk sa dá rozšíriť prostredníctvom knižníc C++ a Ľudia, ktorí chcú porozumieť technickým detailom, môžu urobiť skok z Arduina do programovacieho jazyka AVR-C, na ktorom je založený. Podobne môžete pridať kód AVR-C priamo do programov Arduino.
- **Voľne šíriteľný hardvér** – Plány Arduino sa uverejňujú pod licenciou Creative Commons, takže skúsení dizajnéri obvodov môžu vytvoriť vlastnú verziu modulu, rozšíriť ju a zdokonaľiť. Dokonca aj relatívne neskúsení užívatelia môžu vytvoriť verziu modulu, aby pochopili, ako to funguje a ušetrili peniaze.



Obrázok 3 Arduino MEGA 2560

## 2.1.5 Analýza senzorov

Táto kapitola opisuje základné vlastnosti senzorov teploty, vlhkosti a akcelerometra. Tieto údaje sme použili pri návrhu riešenia.

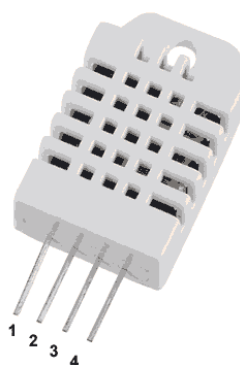
### Senzor teploty a vlhkosti DHT22

Je to presný digitálny senzor teploty a vlhkosti prostredia. Ponúka širší rozsah teplôt a presnejšie merania ako jeho predchodca, DHT11. Na prácu so senzorom je potrebná knižnica. Na odfiltrovanie šumu z prostredia, výrobca odporúča pripojiť pull-up rezistor na DATA pin.

Základné špecifikácie:

- Operačné napätie: 3,3V – 6V
- Meranie vlhkosti: 0-100%RH  $\pm$ 2%RH
- Meranie teploty: -40~80°C  $\pm$ 0.5°C
- Nízka spotreba energie pri pomerne stabilných meraniach

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



Obrázok 4 Senzor teploty a vlhkosti DHT22

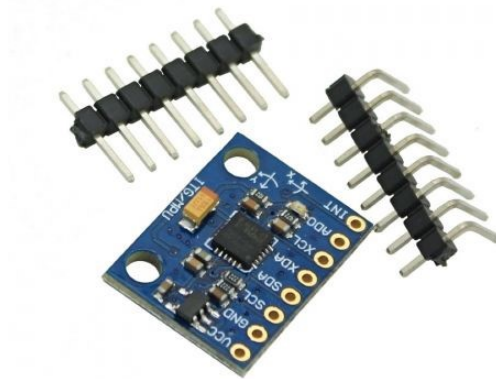
### Akcelerometer

Tento senzor obsahuje 3-osý gyroskop spolu s 3-osím akcelerometrom. Senzor je založený na čipe MPU-6050.

Základné špecifikácie:

- Napájanie: 3 - 5V
- Komunikácia: komunikačný štandard IIC

- Vstavaný 16 bitový AD prevodník, 16 bitový výstup dát
- Rozsah gyroskopu: + 250 500 1000 2000 ° / s
- Rozsah zrýchlenia:  $\pm 2 \pm 4 \pm 8 \pm 16$  g
- Veľkosť: 21 \* 15 \* 1.2 mm
- Hmotnosť: 3 g



Obrázok 5 Akcelerometer a gyroskop

### 2.1.6 IoT zariadenie

Táto kapitola sa zaoberá základnou špecifikáciou IoT zariadenia, ktoré rozšíri naše Arduino o prístup do Sigfox siete.

#### LPWAN SIGFOX anténa

Tento modul obsahuje lacný a malý UART modem spolu s 5dBi anténou. Modem ľahko spolupracuje s Arduino, Raspberry a aj s inými mikrokontrolérmi, ktoré majú Tx a Rx piny. Je určený pre LPWAN (Low Power Wide Area Network) siete. Tento typ siete je určený pre nízkoenergetické prenosy malých objemov dát a je vhodný pre zariadenia napájané z batérií.

#### Základné špecifikácie:

- Typ: rozširujúci kit
- Vývojová platforma: vlastná, Arduino, Raspberry Pi, Mbed
- Kompatibilita s AT príkazmi: áno
- Vývojové prostredie (IDE): vlastná, Arduino, Atmel Studio, mbed
- Sensory: teplota

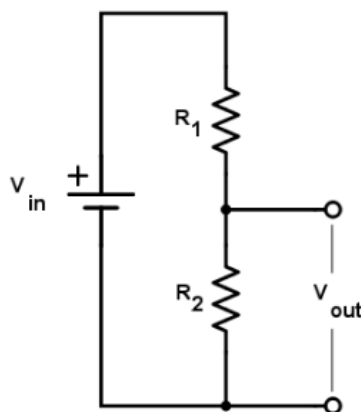


Obrázok 6 UART modem a anténa

### 2.1.7 Analýza merania stavu batérie

Stav batérie je možné vypočítať na základe hodnoty napätia batérie. Analógové vstupy môžu byť použité na meranie DC napätia medzi 0 a 5V. Pre meranie väčšieho rozsahu napätia je možné použiť dva odpory, pomocou ktorých vytvoríme napäťový delič. Delič napätia zníži merané napätie v rozsahu analógových vstupov Arduina. Následne použitím analógového vstupu je možné merať napätie väčšie ako 5V.

Obvod napäťového deliča pozostáva z dvoch odporov v sérii, ktoré rozdelí vstupné napätie na napätie merateľné analógovým vstupom Arduina. Schéma zapojenia takéhoto napäťového deliča je zobrazená na nasledujúcom obrázku, kde  $V_{OUT}$  je výstupne napätie (zmenšene napätie, ktoré je výstupom deliča),  $V_{IN}$  - vstupne napätie,  $R_1, R_2$  - hodnoty odporov a  $\frac{R_2}{R_1+R_2}$  pomer zmenšenia napätia.



Obrázok 7 Schéma zapojenia napäťového deliča

$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2}$$

Arduino obsahuje port, respektíve pin, ktorý je možné použiť na napájanie Arduina z externého zdroja napájania, alebo ak je Arduino napájané pomocou napájacieho konektora je možné pomocou tohto pinu napätie zdroja získať. Takéto zapojenie je možné vidieť na obrázku 8. Arduino prečíta hodnoty z analógového vstupu a následným vhodným prekonvertovaním získa hodnotu napätia zdroja.

## 2.1.8 Analýza posielaných správ

Pre odoslanie údajov nameraných zo senzorov je dostupná správa s maximálnou veľkosťou 12 znakov respektíve bytov. Pre posielanie celých čísel je potrebné minimálne 24 bytov. Preto je potrebné navrhnuť spôsob pre efektívne využitie maximálnej veľkosti posielanej správy.

### Rozsah hodnôt senzorov

- Vlhkosť -  $\langle 0, 100 \rangle$
- Teplota -  $\langle -50, 80 \rangle$
- Stav batérie -  $\langle 0, 100 \rangle$
- Hmotnosť -  $\langle 0, 100 \rangle$
- Poloha úľa -  $\langle 0, 1 \rangle$
- Hodnoty so senzorov zakódujeme do hexadecimálneho tvaru.

### Pridelenie bit-ov hodnotám (Dostupných 12 znakov)

- Vlhkosť - 2 x 7 bit-ov
- Teplota - 2x 8 bit-ov
- Hmotnosť - 7 bit-ov
- Stav batérie - 7 bit-ov
- Poloha úľa - 1 bit

### Vzorové dáta

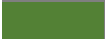






Tabuľka 1 Rozloženie bitov správy

Bit	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Bin	1	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1
Hex	8				5				9				5			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bin	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	1
Hex	0				2				F				9			

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bin	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hex	0				0				0				0			

Tabuľka 2 Legenda k tabuľke rozloženia bitov správy

Farba	Rozsah bitov	Senzor
	0-7	Teplota vonku
	8-15	Teplota vnútri
	16-22	Vlhkosť vonku
	23-29	Vlhkosť vnútri
	30-36	Stav batérie
	37-43	Hmotnosť
	47	Poloha úľa

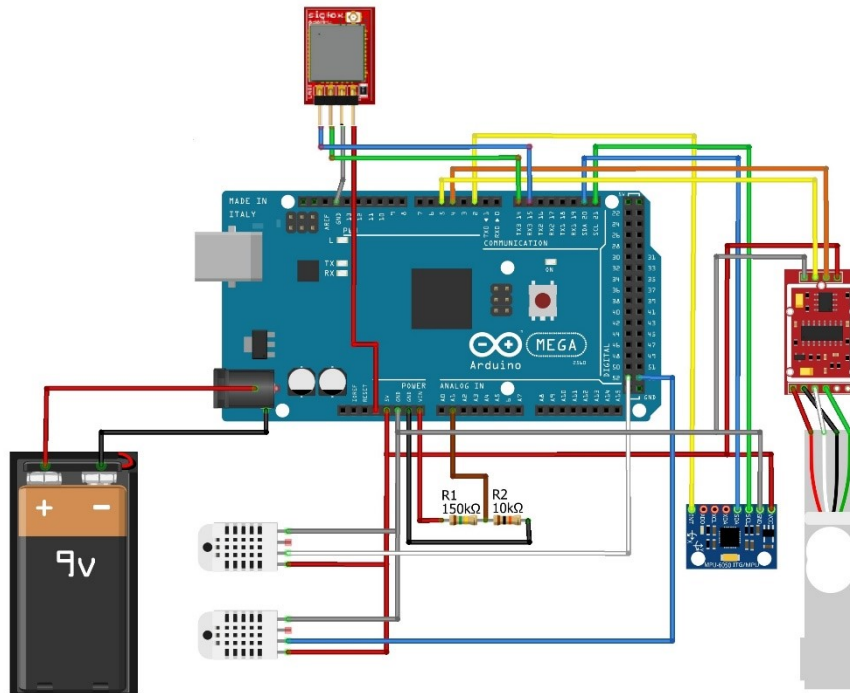
Výsledná správa: 859502F90000

## 2.2 Návrh zapojenia

Pre meranie hmotnosti, teploty a odosielanie týchto údajov využijeme Arduino MEGA 2560. Arduino bude napájané 9V zdrojom napätia. Na Arduino budú pripojené dva senzory DHT22 na meranie vonkajšej aj vnútornej teploty a vlhkosti, akcelerometer MPU6050 pre meranie polohy úľa na základe ktorej bude detegované prevrátenie/odcudzenie úľa, analógovo – digitálny 24bit prevodník HX711 a hmotnostný senzor na meranie váhy úľa za účelom detekcie úľa plného medu a LpWan SigFox node modemu pre odosielanie nameraných údajov na SigFox Cloud. Zapojenie senzorov na Arduino je znázornené na nasledujúcom obrázku.

Priradené piny jednotlivým senzorom:

- DHT22 (1)
  - Digitálny pin 53
  - Napájacie piny +5V, GND
- DHT22 (2)
  - Digitálny pin 52
  - Napájacie piny +5V, GND
- HX711
  - PWM pin 4, 5
  - Napájacie piny +5V, GND
- MPU6050
  - Digitálny pin 2
  - Digitálne (Comuniacion pins SCL, SDA) piny 20, 21
  - Napájacie piny +5V, GND
- LpWan SigFox node
  - Digitálne (Comuniacion pins TX3, RX3) piny 14, 15



Obrázok 8 Zapojenie senzorov na Arduino

## 2.3 Implementácia

Táto kapitola popisuje implementáciu modulu Arduino. Zameriava sa hlavne na správnu funkcionálnu jednotlivých senzorov a spracovanie nameraných dát.

### 2.3.1 Modul merania teploty a vlhkosti

Pre meranie teploty a vlhkosti využívame digitálny senzor DHT22, ktorý dokáže merať obe veličiny. Na meranie hodnôt zo senzorov používame knižnicu DHT Sensor Library verzie 1.3.0. Pre získavanie hodnôt inicializujeme každý senzor, kde zadáme digitálny port na, ktorý je pripojený senzor a typ senzoru, ktorý používame. `<DHT dht(52, DHT22)>` a `<DHT dht2(53, DHT22)>`. Knižnica poskytuje funkcie `readHumidity()` a `readTemperature()`, pomocou ktorých získame potrebné hodnoty, ktoré môžeme ďalej spracovať. Na obrázku je znázornený výstup zo senzorov.

```
-----DHT22-----
Humidity(IN): 50.40 %, Temperature(IN): 23.10 C
Humidity(OUT): 46.70 %, Temperature(OUT): 25.40 C
-----DHT22-----
```

Obrázok 9 Výstup zo senzorov

### 2.3.2 Modul detegovania prevrátenia/odcudzenia

Pre detegovanie prevrátenia/odcudzenia používame akcelerometer MPU-6050, ktorý meria osy x, y a z. Pre meranie využívame knižnicu MPU-6050 a I2Cdev. Pre komunikovanie MPU-6050 používame I2C



protokol. Pre detegovanie používame vstavaný 1024 bytový FIFO buffer. Hodnoty senzora MPU-6050 sa ukladajú do buffra, ktoré sú následne čítané Arduino. Buffer je používaný spolu so signálom prerušenia. Ak MPU-6050 vloží dáta do buffra, signál prerušenia, signalizuje Arduino, že bola vložená hodnota do buffra a čaká na prečítanie Arduino. Hodnoty akcelerometra a gyroskopu sú nazývané „raw“ hodnoty. Predtým než získame údaje z akcelerometra, zariadenie je potrebné inicializovať pomocou funkcie `<mpu.initialize()>` a nastaviť potrebné offsety pre zvýšenie presnosti `mpu.setXGyroOffset(220)`. Zariadenie je pripravené pre čítanie hodnôt z akcelerometra. Arduino v slučke kontroluje či vzniklo prerušenie, ak áno prečíta vstup z FIFO buffra. Údaje získané z buffra zobrazí v stupňoch pomocou funkcie `mpu.dmpGetYawPitchRoll(ypr, &q, &gravity)`. Na základe získaných uhlov detegujeme prevrátenie/odcudzenie úľa. Na obrázku je znázornený výstup z akcelerometra MPU-6050.

```

-----Accelerometer-----
X: -0.30°, Y: -0.07°
State: hive is on right place
-----Accelerometer-----

-----Accelerometer-----
X: -0.24°, Y: 23.11°
State: hive was moved
-----Accelerometer-----

```

Obrázok 10 Výstup z akcelerometra

### 2.3.3 Modul merania stavu batérie

Pre meranie stavu batérie používame napäťový delič, ktorý bol spomenutý vyššie. Hodnoty jednotlivých rezistoroch sú  $R_1 = 10k\Omega$ ,  $R_2 = 150k\Omega$ , čím získam pomer  $\frac{1}{16}$ . Ako zdroj napätia používame batériový box s napätím 9V. Napäťovým deličom získame možnosť merať napätia zdroja, ktoré je väčšie ako 5V. Pre získanie napätia zdroja napájania získame 16 hodnôt meraných z analógového vstupu, ktoré sčítame čím získame hodnotu reprezentujúcu napätia zdroja. Túto hodnotu následne konvertujeme a kalibrujeme pre získanie hodnoty napätia zdroja vo voltoch. Z hodnoty napätia následne získame percentuálne ohodnotenie stavu batérie na základe vzorca:

percentage = (voltage - 6) \* 33.3; Na obrázku je znázornený výstup.

```

-----Battery-----
Voltage: 8.99 V
Percentage: 99 %
-----Battery-----

```

Obrázok 11 Výstup z batérie

### 2.3.4 Sigfox modul

Pri zapnutom zariadení a pripojenom k PC je možné zadávať pomocou *Monitoru sériového portu* príkazy pre SigFox modem. Program následne zobrazí odpoveď z modemu. Pre komunikáciu so SigFox modemom využívam sériový port „Serial3“ a pre odosielanie príkazov využívam funkciu „Serial3.println(command)“ pre prijatie odpovede používam funkciu „Serial3.readString()“.

## Príklady príkazov

AT – overenie správneho zapojenia modemu a napájania

AT+SSF= - odoslanie správy

AT+SSF=A123 - aktualizovanie stavu pripojenia na stránke SigFox backend

AT+ST? – aktuálna teplota

AT+SV? – aktuálna spotreba

## 2.4 Testovanie

Táto kapitola opisuje jednotlivé testy funkčných modulov systému. V každom teste je podrobne popísaná akcia a porovnaná reakcia testera s očakávanou reakciou.

### 2.4.1 Testovanie merania fyzikálnych veličín pomocou Arduina

Tabuľka 3 Testovanie merania fyzikálnych veličín pomocou Arduina

<b>ID:</b>	1	<b>Názov:</b>	Meranie fyzikálnych veličín pomocou Arduina	
<b>Úroveň splnenia testu:</b>	Musí		<b>Tester:</b>	Jozef Vaľko
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie funkčnosti prototypu pre meranie veličín			
<b>Vstupné podmienky:</b>	Dostupný HW			
<b>Výstupné podmienky:</b>	Všetky merania prebehli v poriadku			
<b>Kr ok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Používateľ zapojí senzory podľa schémy zapojenia	Správne zapojenie senzorov na Arduino	Správne zapojenie senzorov na Arduino	
2	Používateľ spustí „Monitor Sériového portu“	Spustenie „Monitor Sériového portu“	Spustenie „Monitor Sériového portu“	
3	Používateľ zadá ľubovoľnú klávesu	Zadanie ľubovoľnej klávesy	Zadanie ľubovoľnej klávesy	
4	Používateľ overí meranie teploty a vlhkosti	Prečítanie a skontrolovanie hodnoty	Prečítanie a skontrolovanie hodnoty	
5	Používateľ overí meranie stavu batérie	Prečítanie a skontrolovanie hodnoty	Prečítanie a skontrolovanie hodnoty	
6	Používateľ overí funkčnosť akcelerometra	Zmena polohy prototypu a zistenie stavu úľa	Zmena polohy prototypu a zistenie stavu úľa	

## 2.4.2 Testovanie SigFox modemu

Tabuľka 4 Testovanie SigFox modemu

<b>ID:</b>	2	<b>Názov:</b>	Otestovanie SigFox modemu	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Peter Štofaňák	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie funkčnosti komunikácie so SigFox-om			
<b>Vstupné podmienky:</b>	Dostupný HW			
<b>Výstupné podmienky:</b>	Overenie funkčnosti prebehlo v poriadku			
<b>Kr ok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Používateľ zapojí anténu do Arduina	Správne zapojenie antény	Správne zapojenie antény	
2	Používateľ pripojí Arduino k PC	Správne pripojenie Arduina k PC	Správne pripojenie Arduina k PC	
3	Používateľ spustí „Monitor Sériového portu“	Spustenie „Monitor Sériového portu“	Spustenie „Monitor Sériového portu“	
4	Používateľ zadá ľubovoľnú klávesu	Zadanie ľubovoľnej klávesy	Zadanie ľubovoľnej klávesy	
5	Používateľ zadá príkaz = AT\$SF=A123	Používateľ zadal správne príkaz	Používateľ zadal správne príkaz	
6	Používateľ overí stav pripojenia na SigFox Backend	Overenie stavu pripojenia	Overenie stavu pripojenia	

## 3 Modul – Server

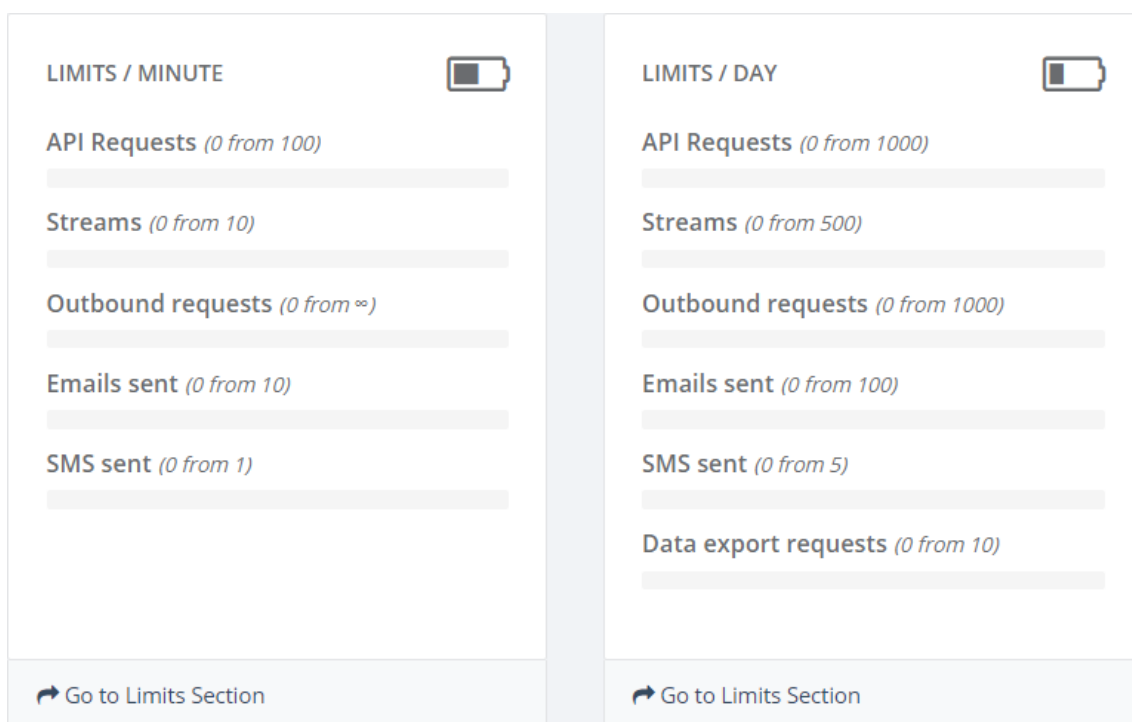
V tejto kapitole je popísaný modul server nášho systému. Je tu rozpísaná analýza niektorých častí a potom návrh a implementácia funkcionalít, ktoré zabezpečujú komunikáciu medzi používateľom a serverom, resp. komunikáciu s Carriots serverom.

### 3.1 Analýza

Táto kapitola sa zaoberá analýzou častí servera a to hlavne Carriots a Sigfox. Analyzovali sme službu Carriots a takisto Sigfox spolu s callbackmi, ktoré nám umožnia získavať údaje.

#### 3.1.1 Carriots

Carriots je Platform as a Service (PaaS) navrhnutá pre Internet vecí (IOT). Carriots umožňuje zbieranie a ukládanie rôznych druhom dát zo zariadení používateľov. Umožňuje vytvárať aplikácie s vlastným SDK (Software development kit). Používateľ môže jednoducho integrovať vlastné aplikácie s externými IT systémami cez API, webové služby a prostredie hostingu. Carriots podporuje vývoj a rozšírenie od malých prototypov po tisíce zariadení. Limity, platné pre voľné členstvo zobrazuje



Obrázok 12 Limity pre voľné členstvo služby Carriots

#### Dôvody použitia

- redukuje čas vývoja projektu
- podporuje triggers: odosielanie emailov, sms správ

## Hlavné výhody

- voľný prístup umožňuje manažovať až 2 zariadenia
- podpora REST API a Groovy SDK
- dokumentácia API a SDK
- podpora komunikácie so Sigfox cloudom

The screenshot shows the Sigfox web interface for configuring a callback for an Arduino Carriots kit. The interface is divided into a sidebar on the left and a main content area. The sidebar contains navigation options: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING TRANSFERRED, STATISTICS, EVENT CONFIGURATION, and CALLBACKS. The main content area is titled "Device type Arduino Carriots kit - Callback edition" and contains a form for setting up a callback. The form includes the following fields and options:

- Callbacks:** Type: DATA, BIDIR; Channel: URL; Send duplicate: ; Custom payload config:  (with a help icon).
- URL syntax:** `http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...`. Available variables: device, time, duplicate, snr, station, data, avgSnr, lat, lng, rssi, seqNumber, ack, longPolling. Custom variables: .
- Uri pattern:** `https://api.carriots.com/runner/Bidirectional_SIGFOX_test@projectName/run?persist=tr`
- Use HTTP Method:** POST
- Send SNI:**  (Server Name Indication) for SSL/TLS connections
- Headers:** Carriots-apikey: apikeretrievedfromcarriotsplatformwithpermissiontobeide (with a help icon). Header: ; Value:
- Content type:** application/json
- Body:**

```
{
  "id": "{device}",
  "lat": {lat},
  "lng": {lng},
  "data": "{data}",
  "bidi": true
}
```

The "Ok" button at the bottom of the form is circled in red.

Callbacks edition

Obrázok 13 Nastavenie Callbacks pre Carriots v Sigfox cloude

## Funkcie platformy

- **Manažment zariadení** - kontrola a interakcia so zariadeniami na diaľku. Kontrola stavu, zmena konfigurácie.
- **Listeners** - pri získaní dát alebo uložení je možné vytvárať udalosti s použitím IF-THEN-ELSE štruktúr.
- **Rules** - vytváranie Groovy skriptov, ktoré môžu byť použité v Listeneroch
- **Triggers** - umožňuje poslať dáta do externých systémov prostredníctvom emailu alebo sms
- **Export dát** - umožňuje spoluprácu s ďalšími IT systémami, vytvorenie exportovacích súborov, alebo použitie REST API pre manažovanie výstupných dát
- **Vytváranie logov** - zaznamenávanie interakcií s platformou

### 3.1.2 Analýza Sigfox callbackov

Táto kapitola opisuje analýzu Sigfox callbackov, ktoré používame v našom systéme na získavanie dát zo Sigfox cloudu na Carriots server. Existujú dva spôsoby posielania dát na server Carriots zo SigFox cloudu:

- Posielanie POST request na špeciálnu URL, ktorá umožňuje špecifikovať parametre
- Posielanie POST message, ktorá ma hlavičku a telo

#### Používanie špeciálnej adresy

Carriots poskytuje špeciálnu webovú adresu na príjem údajov zo služby SIGFOX Cloud. Stačí poslať požiadavku POST s niektorými parametrami.

#### Odoslanie dát na Carriots

Tabuľka 5 Odoslanie dát na Carriots

<b>URL</b>	http://sigfox.carriots.com/streamforms
<b>HTTP metóda</b>	POST

#### Parametre

Tabuľka 6 Atribúty dát a ich opis

Atribút	Opis
<b>apikey (povinný)</b>	Apikey sa používa na určenie privilégií a viditeľnosti requestov v rámci Carriots platformy
<b>protocol (povinný)</b>	Komunikačný protokol. Používa sa na kontrolné účely
<b>at (povinný)</b>	Unix časová pečiatka. Carriots vyplní „at“ časovou pečiatkou dátumu prijatia
<b>device (povinný)</b>	Jedinečný identifikátor zariadenia
<b>data (povinný)</b>	Pole

#### Kódy odozvy

Tabuľka 7 Kódy odozvy

Kód	Opis
<b>200</b>	OK
<b>401</b>	Error
<b>404</b>	Zdroj nenájdený
<b>503</b>	Služba nedostupná

## Príklad takejto URL

[http://sigfox.carriots.com/streamforms?apikey=c4564dsf&protocol=v1&at=now&device=sigfoxHive@carriots&data\[value1\]=1&data\[value2\]=2](http://sigfox.carriots.com/streamforms?apikey=c4564dsf&protocol=v1&at=now&device=sigfoxHive@carriots&data[value1]=1&data[value2]=2)

### 3.1.3 SigFox Cloud

Nastavenie posielania „Callbacks“ na SigFox Cloude, ktoré bude automaticky preposielať niektoré udalosti. Callback sa nastavuje na skupinu zariadení nie na konkrétne zariadenie. Callbacks sa spúšťajú, keď príde nová správa od zariadenia alebo keď sa deteguje strata komunikácie so zariadením

#### Callback typy

Existujú 3 typy callback

- DATA
- SERVICE
- ERROR

K dispozícii je taktiež súbor spoločných premenných a súbor špecifických premenných pre každý callback typ. Tieto premenné sa nahradia ich hodnotou pri volaní callback.

#### Data

Tento callback typ definuje príjem používateľskej správy zo zariadenia.

#### Špecifické premenné

Tabuľka 8 Špecifické premenné

Premenná	Opis
<b>rssi</b>	RSSI – sila signálu
<b>data</b>	Používateľské dáta v hexa tvare

#### Service

Tento callback definuje príjem správy zo zariadenia o stave prevádzky.

#### Error

Tento callback definuje straty komunikácie pre dané zariadenie.

## Špeciálne premenné

Tabuľka 9 Špeciálne premenné

Premenná	Opis
<b>info</b>	Informácie o chybe v prípade straty komunikácie obsahujú posledný dátum prijatej správy
<b>severity</b>	ERROR ak je chyba v zariadení, WARN ak je chyba v sieti

## Spoločné premenné

Tabuľka 10 Spoločné premenné

Premenná	Opis
<b>time</b>	Časová pečiatka udalosti
<b>device</b>	Identifikátor zariadenia
<b>signal</b>	Pomer signálu k šumu
<b>avgSignal</b>	Priemerný pomer signálu k šumu vypočítaný z posledných 25 správ
<b>station</b>	Identifikátor základnej stanice
<b>lat</b>	Zemepisná šírka
<b>lng</b>	Zemepisná dĺžka

Parametre musia byť uzavreté v zátvorkách {}.

## 3.2 Návrh

Táto kapitola opisuje návrh jednotlivých častí servera a to službou Carriots, použitie Carriots API a návrhom databázy v PostgreSQL.

### 3.2.1 Carriots service

Na ukladanie dát sme sa rozhodli používať službu Carriots. Dáta v Carriots sú uložené vo formáte JSON v nerelačnej databáze.

Merania budú v Carriots databáze uložené v nasledovnom formáte:

```
"Merania": [  
  { "cas": "18.10.2017.22.37", "hodnota": 35, "typ": "IT"},  
  { "cas": "18.10.2017.22.37", "hodnota": 35, "typ": "OT"},  
  { "cas": "18.10.2017.22.37", "hodnota": 80, "typ": "H"},  
  { "cas": "18.10.2017.22.37", "hodnota": true, "typ": "P"}  
]
```

Obrázok 14 Formát merania v databáze



## Získanie dát:

Pre účely získavanie dát používa Carriots REST API volania. Pre získanie meraní je potrebné uskutočniť metódu GET na URL <https://api.carriots.com/streams/>. Takýto request vráti merania zo všetkých zariadení.

<b>Content-Type</b>	application/json; charset=utf-8
<b>Carriots.apiKey</b>	582155c4a8a15467d4fbede176862673f4c5a5137b911e8a7cbf5034ff7c38ce

Obrázok 15 Hlavičky požiadaviek pre API volanie

V prípade, že chceme získať dáta z konkrétneho zariadenia, pridáme URL parameter „device“. V takom prípade bude vytvorený dopyt na adresu, v ktorej názov zariadenia označujeme ako „device\_name“: [https://api.carriots.com/streams/?device=„device\\_name“](https://api.carriots.com/streams/?device=„device_name“)

Carriots API tiež poskytuje filtrovanie podľa času merania, umožňuje sort získaných výsledkov, nastavenie limitu alebo offset. Všetky dostupné parametre, ktoré poskytuje Carriots API sú dostupné v dokumentácii na adrese [https://www.carriots.com/documentation/api/data\\_management](https://www.carriots.com/documentation/api/data_management).

## 3.2.2 Použitie Carriots API

### Použitie carriots API pre vytvorenie streams z SigFox Cloudu

Tabuľka 11 Použitie carriots API pre vytvorenie streams z SigFox Cloudu

<b>URL:</b>	<a href="http://api.carriots.com/streams/">http://api.carriots.com/streams/</a>
<b>HTTP metóda:</b>	POST

### Parametre

Tabuľka 12 Atribúty

Atribút	Opis
<b>protocol (povinný)</b>	Komunikačný protokol. Používa sa na kontrolné účely
<b>at (povinný)</b>	Unix časová pečiatka. Carriots vyplní „at“ časovou pečiatkou dátumu prijatia
<b>device (povinný)</b>	Jedinečný identifikátor zariadenia
<b>data (povinný)</b>	Pole
<b>checksum (voliteľný)</b>	Kontrolný kód kontrolného súčtu
<b>persist (voliteľný)</b>	Stream persistence (0,1)

## Kódy odozvy

Tabuľka 13 Kódy odozvy

Kód	Opis
200	OK
401	Error
404	Zdroj nenájdený
503	Služba nedostupná

### Príklad request

POST <http://api.carriots.com/streams/>

```
{
  "protocol": "v1",
  "device": "defaultDevice@carriots",
  "at": "now",
  "persist": "true",
  "data": {
    "temperature": "20",
    "humidity": "50"
  }
}
```

### Príklad response

```
{
  "response": "OK"
}
```

### 3.2.3 Databáza PostgreSQL

Pre uchovávanie údajov o používateľoch a im patriacim zariadeniam sme sa rozhodli použiť databázu PostgreSQL, ktorá je nainštalovaná na webovom školskom serveri, dostupnom na adrese <http://147.175.149.151/>.

Údaje v databáze nám umožnia uskutočniť REST API volania zo servera na Carriots pre získanie nameraných údajov pre konkrétneho používateľa. Každý používateľ má pridané zariadenia (tabuľka devices) tak ako sú vytvorené v službe Carriots. V službe Carriots sú uložené všetky namerané údaje. V databáze na serveri udržujeme len názov zariadenia zhodný s Carriots a GPS koordináty, ktoré v službe Carriots nie je možné pridať a držať ich na serveri je výhodné.

### Tabuľka users:

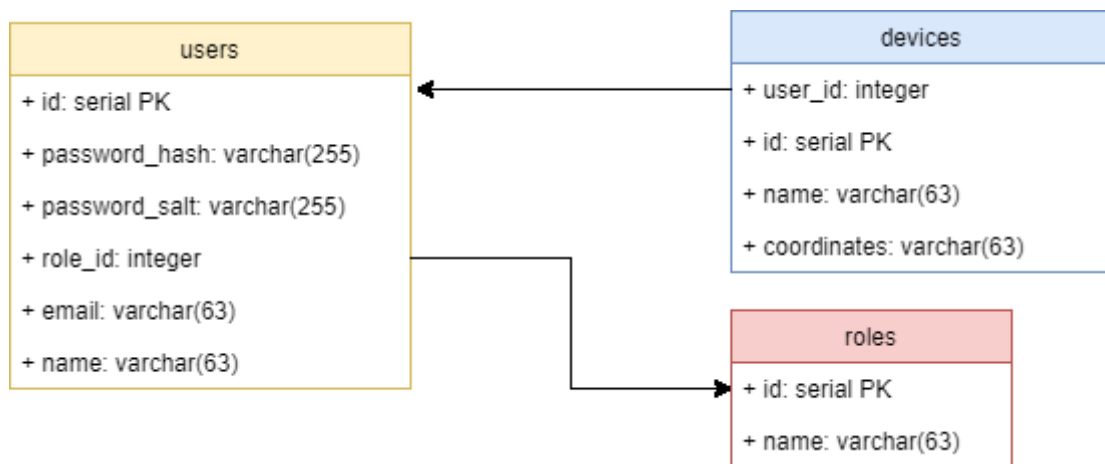
- Id: Primárny kľúč
- Password\_hash: Hodnota hashu hesla
- Password\_salt: Hodnota soli, použitej pri vytvorení hodnoty hashu
- Role\_id: Cudzí kľúč do tabuľky "roles"
- Email: Email používateľa - slúži na prihlasovanie
- Name: Meno a priezvisko používateľa

### Tabuľka devices:

- Id: Primárny kľúč
- User\_id: Cudzí kľúč do tabuľky "users"
- Name: Názov zariadenia, tak ako je uložený v službe Carriots
- Coordinates: Gps súradnice zariadenia

### Tabuľka roles:

- Id: Primárny kľúč
  - Enumerácia, ktorá značí typ role
    - 1: Administrátor
    - 2: Včelár
    - 3: Pozorovateľ
- Name: Označuje názov danej role



Obrázok 16 Návrh databázy včelárov

### 3.2.4 Zabezpečenie spojenia

Po zadaní používateľského mena a hesla sa bude ešte pred odosielaním údajov heslo šifrovať nasledovne. Vezme sa čistý text hesla a zahashuje sa spoločne so saltom, ktorý bude uložený v android aplikácii. Ako hashovacia funkcia bola vybraná sha3-512, ktorá je momentálne považovaná za jednu z najbezpečnejších. Takto zahashovaný text bude následne odoslaný do siete. API na serveri potom prijme dané údaje a porovná z databázy najprv používateľské meno a následne jednotlivé hash. Týmto krokom

sa zabráni odchyteniu hesla resp. sniffingu. Útočník získa len hash, spolu so saltom a nie čistý text hesla. Nemôže ani hádať, ktorý hash bol použitý keďže tam je taktiež pridaný salt. Pre prácu so sha3-512 by bolo vhodné mať najnovšiu verziu t.j. php 7.0. Samozrejme je možné použiť kľudne aj iný programovací jazyk, napr. Javascript. Tento algoritmus bude ale úplne účinný až vtedy, keď sa ako komunikačný protokol bude používať šifrovací protokol TLS medzi Java aplikáciou a API na serveri.

### 3.3 Implementácia

Táto kapitola opisuje implementáciu API volaní na našom serveri.

#### 3.3.1 API pre komunikáciu medzi používateľom a serverom Včelička

Tímový server poskytuje okrem webového sídla tímu aj ďalšiu funkcionálnosť pre náš projekt. Je to napríklad overenie používateľa v databáze. Na serveri je implementovaná funkcionálnosť na prihlásenie a registráciu používateľa v jazyku PHP. Jednotlivé triedy spolu vytvárajú API na prihlásenie a registráciu používateľa. Zabezpečujú overenie používateľa v databáze, ktorý sa chce prihlásiť do mobilnej alebo webovej aplikácie a aj prídanie nového používateľa pri registrácii do nášho systému. Ďalšou funkcionálnosťou sú API pre získavanie údajov konkrétneho používateľa a aj API na vytvorenie objednávky zariadenia.

#### API pre prihlásenie

Cesta k získaniu prístupu: /login/user

- HTTP metóda: POST
- Parametre požiadavky:
  - {email}=email používateľa
  - {password}=heslo používateľa
- Vracia JSON
  - V prípade nezadaného emailu alebo hesla:
    - {"error":true,"error\_msg":"Required parameters email or password is missing!"}
  - V prípade zle zadaného emailu alebo hesla
    - {"error":true,"error\_msg":"Login credentials are wrong. Please try again!"}
  - V prípade správneho zadania všetkých údajov
    - {"error":false,"id": id používateľa,"role\_id": rola používateľa ,"user":{"name": meno používateľa},"email": email },"token": token}

## API pre registráciu

Cesta k registrácii používateľa: register/user

- HTTP metóda: POST
- Parametre požiadavky:
  - {name}= meno používateľa
  - {email}= email používateľa
  - {password}= heslo používateľa
- Vracia JSON
  - V prípade nezadania jedného z povinných parametrov
    - {"error":true,"error\_msg":"Required parameters (name, email or password) is missing!"}
  - V prípade ak používateľ zadá email, ktorý sa už používa
    - {"error":true,"error\_msg":"User already existed with email"}
  - V prípade úspešnej registrácie
    - {"error":false,"uid":**id používateľa**,"user":{"name": **meno používateľa**,"email": **email používateľa**,"created\_at": **čas registrácie**}}

## API pre získanie dát

Na serveri sú implementované funkcie pre získanie zariadení meraní z Carriots. Tieto funkcie overujú autentifikáciu používateľa pomocou technológie JSON Web Tokens<sup>1</sup>.

### Poskytnutie názvov zariadení pre klienta:

Cesta: /db/devices

- HTTP metóda: POST
- Request:

```
+ Request (application/json)
{
  "user_id": "3",
  "token": "xyz"
}
```

---

<sup>1</sup> <https://jwt.io/>

- Response valid:

+ Response 200 (application/json)

```
{
  "data": [
    {
      "name": "DeviceBratislava@fiitp20.fiitp20"
    },
    {
      "name": "DeviceTomas@fiitp20.fiitp20"
    }
  ]
}
```

- Response invalid:

+ Response 401 (application/json)

```
{
  "error": true
}
```

### Poskytnutie informácií o zariadeniach

Cesta: /db/devices/info

- HTTP metóda: POST
- Request:

+ Request (application/json)

```
{
  "user_id": "3",
  "token": "xyz"
}
```

- Response valid:

+ Response 200 (application/json)

```
{
  "data": [
    {
      "name": "DeviceBratislava@fiitp20.fiitp20",
      "uf_name": "Pod cintorinom",
      "location": "Bratislava",
      "coordinates": ""
    },
    {
      "name": "DeviceTomas@fiitp20.fiitp20",
      "uf_name": "Pri FIITke",
      "location": "Bratislava",
      "coordinates": ""
    }
  ]
}
```

Response invalid:

+ Response 401 (application/json)

```
{
  "error": true
}
```

### Poskytnutie posledného merania

Cesta: /api/measurements/actual

- HTTP metóda: POST
- Request:

+ Request (application/json)

```
{
  "device_name": "DeviceTomas@fiitp20.fiitp20",
  "token": „xyz“
}
```

- Response valid:

+ Response 200 (application/json)

```
{
  "data": [
    {
      "cas": "19.10.2017.22.37",
      "hodnota": 32,
      "typ": "IT"
    },
    {
      "cas": "19.10.2017.22.37",
      "hodnota": 14,
      "typ": "OT"
    },
    {
      "cas": "19.10.2017.22.37",
      "hodnota": 50,
      "typ": "H"
    },
    {
      "cas": "19.10.2017.22.37",
      "hodnota": true,
      "typ": "P"
    },
    {
      "cas": "19.10.2017.22.37",
      "hodnota": 30,
      "typ": "W"
    },
    {
      "cas": "19.10.2017.22.37",
      "hodnota": true,
      "typ": "P"
    }
  ]
}
```



- Response invalid:

```
+ Response 401 (application/json)
```

```
{
  "error": true
}
```

### Poskytnutie všetkých meraní:

Cesta k získaniu zariadení: /api/measurements/all

- HTTP metóda: POST

Request aj Response telá sú rovnaké ako v predchádzajúcom volaní pre získanie aktuálneho volania.

Validný Response vracia pole všetkých meraní.

### API pre vytvorenie objednávky

Cesta: /order/new

- HTTP metóda: POST
- Request:

```
+ Request (application/json)
```

```
{
  "name": "Janko",
  "email": "janko@gmail.com",
  "phone": "0455443555",
  "device_count": 9,
  "notes": "Dobry den. Dakujem. Dovidenia.",
  "user_id": 3,
  "token": "xyz"
}
```

- Validný reponse: 200
- Invalidný reponse: 401

### 3.4 Testovanie

Táto kapitola obsahuje testovania funkcionality jednotlivých API na serveri.

#### 3.4.1 Testovanie API na získanie zariadení používateľa

Tabuľka 14 Testovanie API na získanie zariadení používateľa

<b>ID:</b>	3	<b>Názov:</b>	Testovanie API na získanie zariadení používateľa	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Tomáš Koreň	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie správnej funkčnosti získania zariadení používateľa			
<b>Vstupné podmienky:</b>	Internetové pripojenie, Url, Rest Api Client			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Zapnutie REST API klienta (napr. Advanced REST Client)	Zapnutie klienta	Zapnutie klienta	
2	GET na URL: http://147.175.149.151/bee/public/db/devices/3	Vrátenie zariadení používateľa s ID=3	Vrátenie zariadení používateľa s ID=3	
3	GET na URL: http://147.175.149.151/bee/public/db/devices/1	Vrátenie null	Vrátenie zariadení používateľa s ID=3	

#### 3.4.2 Testovanie API na získanie posledného merania

Tabuľka 15 Testovanie API na získanie posledného merania

<b>ID:</b>	4	<b>Názov:</b>	Testovanie API na získanie posledného merania	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Tomáš Koreň	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie správnej funkčnosti získania posledného merania			
<b>Kr ok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	GET na URL: http://147.175.149.151/bee/public/api/measurement s/DeviceTomas@fiittp20.fiittp20	Zobrazenie posledného merania pre zariadenie DeviceTomas@fiittp20.fiittp20	Zobrazenie posledného merania pre zariadenie DeviceTomas@fiittp20.fiittp20	
2	GET na URL: http://147.175.149.151/bee/public/api/measurement s/DeviceBratislava@fiittp20.fiittp20	Zobrazenie posledného merania pre zariadenie DeviceBratislava@fiittp20.fiittp20	Zobrazenie posledného merania pre zariadenie DeviceBratislava@fiittp20.fiittp20	

## 4 Modul – Webová stránka produktu

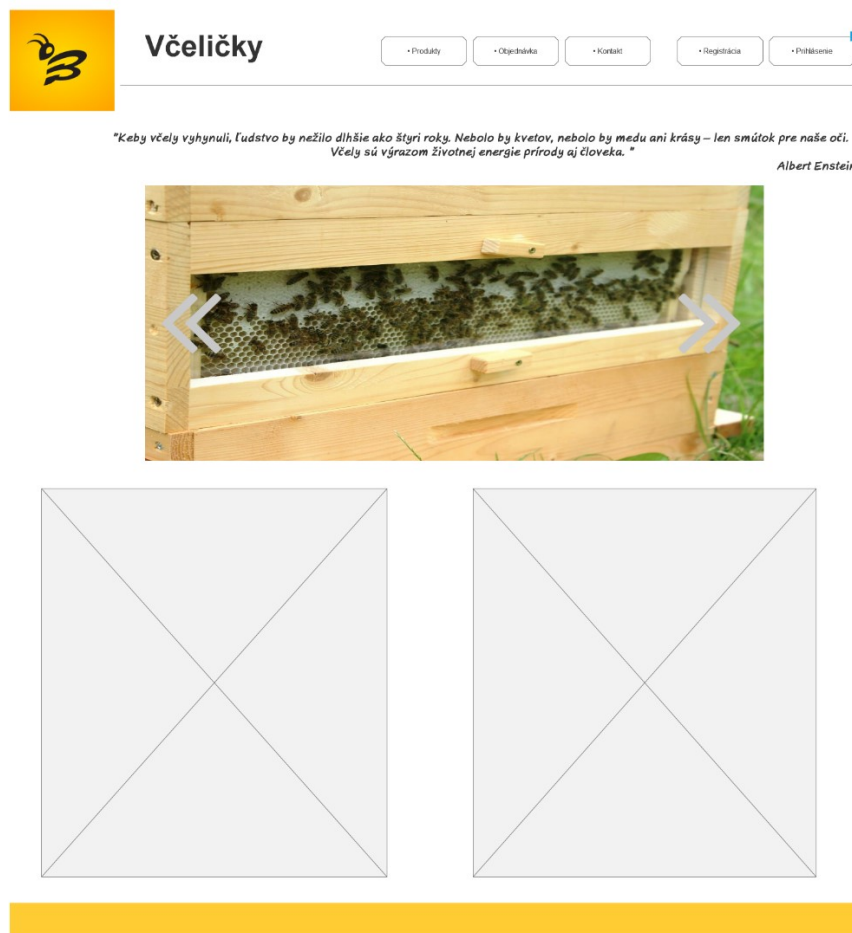
Tento modul opisuje návrh, implementáciu a testovanie webovej stránky produktu. Je to stránka určená pre včelárov, čiže našich zákazníkov a takisto pre správcov systému.

### 4.1 Návrh

V nasledujúcej časti je zobrazená základná vizuálna predstava, ako by mala vyzerat' stránka pre včelárov. Na jej základe bude postavená následná implementácia.

#### 4.1.1 Titulná stránka

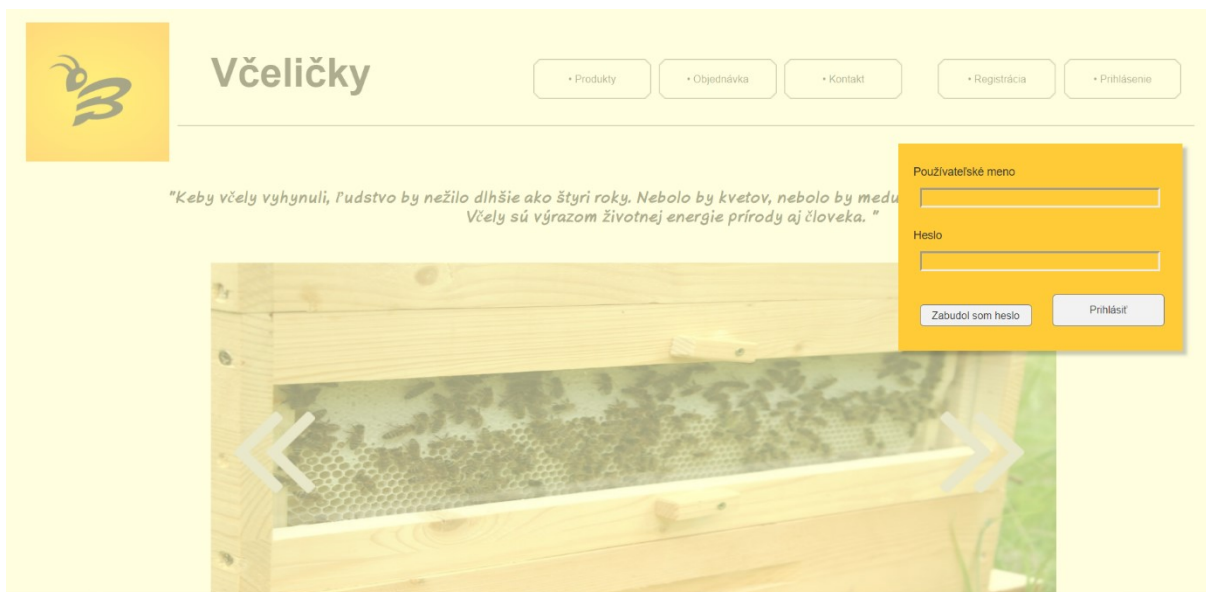
Základným prvkom stránky bude prezentácia obrázkov, ktorá má za úlohu zaujať nových ale aj stálych používateľov stránky. Jej obsahom budú zo začiatku tematické obrázky včiel, úl'ov alebo nášho riešenia. Po spustení stránky do reálnej prevádzky sa tam budú môcť nachádzať napríklad odkazy na nadchádzajúce spoločné udalosti alebo prípadné zmeny/novinky v projekte. Pod prezentáciou obrázkov bude umiestnený stručný zaujímavý popis, ktorý bude mať za úlohu nového používateľa nadchnúť a presvedčiť ho o investícii do nášho produktu.



Obrázok 17 Titulná stránka

Horný panel titulnej stránky bude obsahovať naše logo, nadpis stránky a päť ďalších položiek. V časti produkty sa bude nachádzať popis hardvérových zariadení použitých v našom riešení a ich spôsob využitia a implementovania. Ďalšou časťou bude objednávka pre zákazníka a kontakty na nás. Poslednými prvkami bude registrácia nového používateľa a prihlásenie.

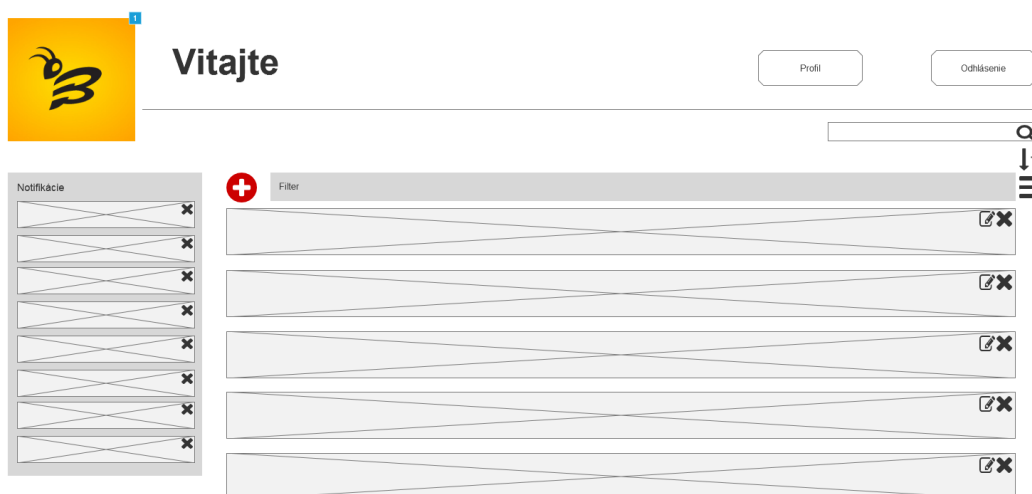
Nasledujúci obrázok zobrazuje jednoduchý formulár, ktorý bol zvolený na prihlásenie používateľov. Používatelia sa budú prihlasovať pomocou svojho emailu a zadaného hesla. V prípade, že zabudnú heslo, formulár im poskytne možnosť pomocou kontrolnej otázky overiť svoju totožnosť.



Obrázok 18 Titulná stránka – prihlásenie

#### 4.1.2 Stránka po prihlásení včelára

Podľa zadaných prihlasovacích údajov systém rozozná, či ide o včelára alebo administrátora. Včelár si bude môcť v hornom paneli zobrazovať svoj profil. Hlavným obsahom bude prehľad úľov, ktoré má včelár pridelené. V každom momente si bude môcť jednotlivé záznamy pridávať, upravovať, mazať či filtrovať alebo vyhľadávať medzi nimi. Podstatnou súčasťou sú aj notifikácie, ktoré sú umiestnené v pravej časti.



Obrázok 19 Stránka po prihlásení včelára

### 4.1.3 Stránka po prihlásení administrátora

Z grafického hľadiska bude kompozícia stránky po prihlásení administrátora veľmi podobne implementovaná ako stránka po prihlásení včelára, s tým rozdielom, že bude obsahovať záznamy o včelároch. Tie bude môcť spravovať a po rozkliknutí každého záznamu bude mať k dispozícii informácie o samotných úľoch včelára.



Obrázok 20 Stránka po prihlásení administrátora

## 4.2 Implementácia

Táto kapitola opisuje implementáciu jednotlivých častí webovej stránky produktu.

### 4.2.1 Zabezpečenie komunikácie

Do webovej stránky produktu bol implementovaný protokol TLS, ktorý slúži na šifrovanie komunikácie medzi klientom a serverom. Pri implementácii bol najprv pomocou príkazu „`sudo a2enmod ssl`“ povolený protokol `tls`. Následne bolo potrebné reštartovať `apache` príkazom „`sudo service apache2 restart`“. Po vykonaní týchto krokov bolo potrebné vytvoriť certifikát, aby po spustení web stránky nevykazoval antivírusový program chybu, že je na stránke nedôveryhodný certifikát. Ku konfiguračnému súboru `ssl` bolo teda potrebné nakopírovať certifikát a privátny kľúč. Potom bolo potrebné upraviť konfiguračný súbor `default-ssl.conf` a doplniť v ňom meno servera, doménu, alias a taktiež cestu k certifikátu a privátnemu kľúču. Po vykonaní týchto krokov bol tento konfiguračný súbor povolený a nakoniec bolo potrebné reštartovať `apache`.

### 4.2.2 Webová stránka

Hlavná časť webovej stránky je implementovaná vo frameworku `Slim3`. Na verziovanie `PHP` balíkov sa používa `composer`. Na zabezpečenie responzibility sa používa `Bootstrap4`. Na verziovanie `javascript` knižníc sa používa `npm`. Na správu `javascript` a `css` súborov sme použili nástroj `gulp`, pomocou ktorého sa spustia skripty, ktoré majú za úlohu uľahčiť správu a kompresiu týchto súborov. Používateľ sa na webovú stránku prihlási pomocou používateľského e-mailovej adresy a hesla. Server overí používateľa

v databáze, a ak používateľ zadal správne prihlasovacie údaje prihlási ho na stránku. Používateľ si následne môže v aplikácii pozrieť hodnoty namerané senzormi na jeho úľoch. Dáta zo senzorov sa získavajú zo serveru včelíčka cez jeho REST API.

### 4.2.3 Podstránka kontakt

Účelom tejto podstránky je umožniť potencionálnym zákazníkom, aby nás mohli v prípade záujmu kontaktovať. Stačí ak na webovej stránke vyplnia formulár s meno, e-mailom, telefónom a taktiež správou. Podstránka bola implementovaná pomocou webového frameworku bootstrap v prostredí phpstorm. Na podstránku bude v budúcnosti doimplementované overenie údajov a taktiež odosielanie údajov na mail.

## 4.3 Testovanie

Táto kapitola opisuje jednotlivé testy funkčných modulov systému. V každom teste je podrobne popísaná akcia a porovnaná reakcia testera s očakávanou reakciou.

### 4.3.1 Testovanie zabezpečenia komunikácie medzi klientom a web stránkou produktu

Tabuľka 16 Testovanie zabezpečenia komunikácie medzi klientom a web stránkou produktu

<b>ID:</b>	5	<b>Názov:</b>	Testovanie zabezpečenia komunikácie medzi klientom a web stránkou produktu	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Matúš Sosňák	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie zabezpečenia komunikácie medzi klientom a web stránkou produktu			
<b>Vstupné podmienky:</b>	Prístup k internetu, program Wireshark, ip adresa web stránky produktu			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Spustenie programu Wireshark a navolenie konkrétneho rozhrania na ktorom bude bežať komunikácia	Otvorenie nového okna, kde sa zobrazujú aktuálne dáta prúdiace po sieti	Otvorenie nového okna, kde sa zobrazujú aktuálne dáta prúdiace po sieti	
2	Navolenie filtra na ip adresu 147.175.149.151	Okno s aktuálnymi dátami by malo byť prázdne, keďže s daným serverom ešte neprebíha komunikácia	Okno s aktuálnymi dátami je prázdne	
3	Spustenie webového prehliadača a zadanie url: <a href="https://team20-17.studenti.fiit.stuba.sk/BeeWebPage/public">https://team20-17.studenti.fiit.stuba.sk/BeeWebPage/public</a>	Zobrazenie web stránky produktu	Zobrazila sa web stránka produktu	
4	Skontrolovanie Wiresharku	Zobrazenie šifrovaných dát protokolom TLS	Zobrazili sa šifrované dáta protokolom TLS	

### 4.3.2 Testovanie prihlásenia na webovej stránke

Tabuľka 17 Testovanie prihlásenia (Web)

<b>ID:</b>	6	<b>Názov:</b>	Testovanie funkcie prihlásenia na Web stránke včelička	
<b>Úroveň splnenia testu:</b>	Musí		<b>Tester:</b>	Matúš Sosňák
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie korektnosti funkcie prihlásenia používateľa			
<b>Vstupné podmienky:</b>	Internetové pripojenie, Url webu, Prihlasovacie údaje,			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Spustenie web stránky včelička	Zobrazenie hlavnej web stránky	Zobrazenie hlavnej web stránky	
2	Kliknutie na tlačidlo prihlásenie	Zobrazenie formulára pre zadanie prihlasovacích údajov	Zobrazenie formulára pre zadanie prihlasovacích údajov	
3	Zadanie správnych prihlasovacích údajov (login, heslo). Stlačenie tlačidla prihlásiť sa	Úspešné zobrazenie web stránky prihláseného používateľa	Úspešné zobrazenie web stránky prihláseného používateľa	
4	Zadanie nesprávnych prihlasovacích údajov (login, heslo). Stlačenie tlačidla prihlásiť	Zobrazenie okna s chybovou hláškou: „Chyba! Nesprávne prihlasovacie údaje“	Žiadna reakcia	

### 4.3.3 Testovanie registrácie používateľa

Tabuľka 18 Testovanie registrácie používateľa

<b>ID:</b>	7	<b>Názov:</b>	Testovanie registrácia používateľa	
<b>Úroveň splnenia testu:</b>	Musí		<b>Tester:</b>	Michal Puškáš
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie správnosti registrácie používateľa			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Vynechanie alebo nesprávne vyplnenie polí v registračnom formulári	Upozornenie používateľa chybovou hláškou a zabránenie odoslaniu údajov	Upozornenie používateľa chybovou hláškou a zabránenie odoslaniu údajov	
2	Zaregistrovanie používateľa kliknutím na tlačidlo "Vytvoriť účet"	Pridanie riadku v databáze s novým používateľom	Pridanie riadku v databáze s novým používateľom	

#### 4.3.4 Testovanie funkčnosti webového rozhrania pre správu databázy na serveri Včelička

Tabuľka 19 Testovanie funkčnosti webového rozhrania pre správu databázy na serveri Včelička

<b>ID:</b>	8	<b>Názov:</b>	Testovanie funkčnosti webového rozhrania pre správu databázy na serveri Včelička	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Matúš Sosňak	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie funkčnosti webového rozhrania pre správu databázy na serveri Včeličky			
<b>Vstupné podmienky:</b>	Prístup k internetu, url rozhrania databázy, prihlasovacie údaje			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Spustenie webového prehliadača a zadanie url: <a href="https://team20-17.studenti.fiit.stuba.sk/phppgadmin">https://team20-17.studenti.fiit.stuba.sk/phppgadmin</a>	Otvorenie stránky pre prihlásenie do rozhrania databázy	Bola otvorená stránka pre prihlásenie do rozhrania databázy	
2	Korektné zadanie používateľského mena a hesla	Žiadna reakcia	Žiadna reakcia	
3	Potvrdenie údajov	Úspešné prihlásenie a zobrazenie údajov databázy	Úspešné prihlásenie a zobrazenie údajov databázy	



## 5 Modul – Android

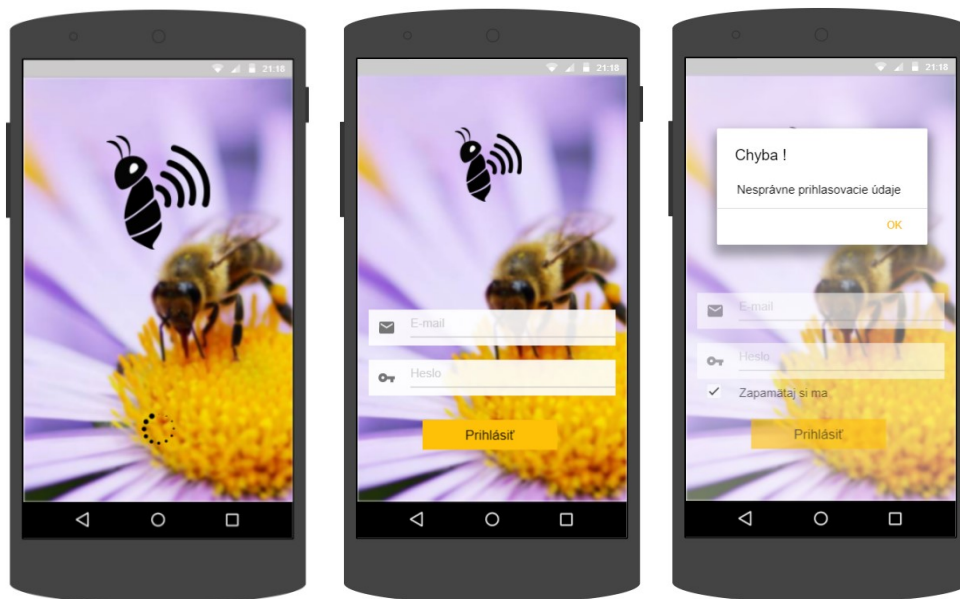
Tento modul opisuje návrh a implementáciu mobilnej aplikácie určenej pre operačný systém Android. Je tu popísaný návrh v podobe mockupov aplikácie a aj implementácia jednotlivých aktivít, ktoré sú nakoniec otestované.

### 5.1 Návrh mobilnej aplikácie

Pomocou webovej aplikácie proto.io boli vytvorené mockupy android aplikácie. V druhom šprinte vznikol návrh úvodnej obrazovky, prihlasovacej obrazovky a hlavná obrazovka s prehľadom úľov. V treťom šprinte bol pridaný bočný panel a jeho obsah. Vo štvrtom šprinte sme pridali mock-upy registrácie používateľa, formuláru na objednanie úľa, mockup odhlásenia používateľa a zobrazenie detailu úľa.

#### 5.1.1 Úvodné logo a prihlasovacia obrazovka aplikácie

Počas prvých sekúnd zapínania aplikácie je zobrazené logo ako hlavná dominanta obrazovky. Rovnako ako pri webovej aplikácii, používateľ sa prihlasuje so svojim emailom a zadaným heslom. Posledná, tretia obrazovka zobrazuje chybové okno, ktoré sa zobrazí po nesprávnom zadaní prihlasovacích údajov.

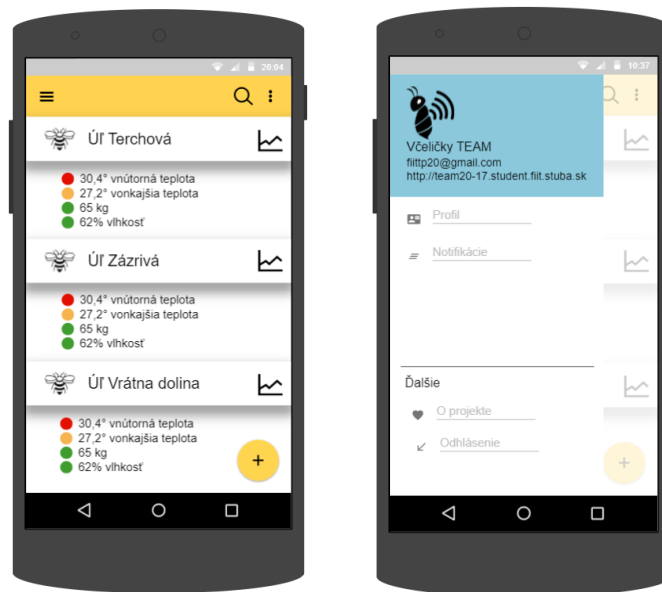


Obrázok 21 Úvodné logo a prihlasovacia obrazovka aplikácie

#### 5.1.2 Prehľad úľov a bočný panel aplikácie

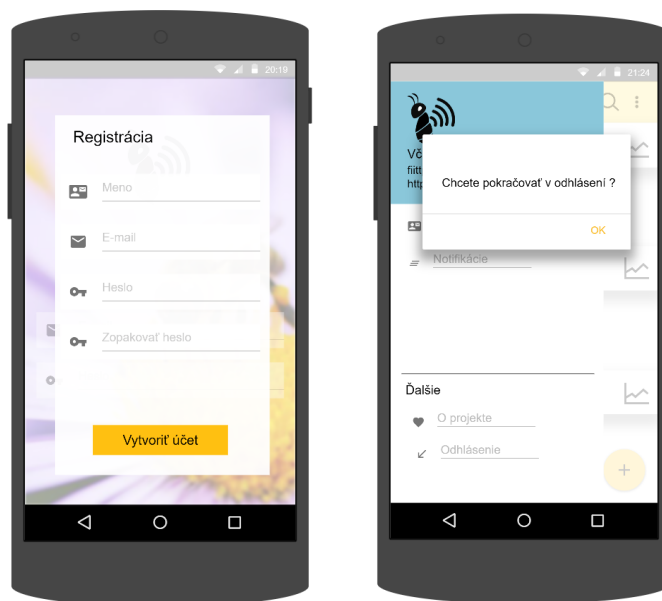
Hlavná obrazovka bude obsahovať prehľad úľov včelára. Nadpisom budú oddelené jednotlivé úle a v jednoduchom zobrazení budú obsahovať informácie o teplotách, váhe a vlhkosti. Farebné guľičky budú notifikovať aktuálny stav a uľahčia tak včelárovi prehľad.

Bočný panel bude obsahovať základné informácie o nás ako tvorcoch projektu a v hlavnej časti profil včelára, historické notifikácie, informácie o projekte a možnosť odhlásenia.



Obrázok 22 Prehľad úlov a bočný panel aplikácie

### 5.1.3 Registrácia používateľa a chybové okno pri odhlasovaní



Obrázok 23 Registrácia používateľa a chybové okno pri odhlasovaní

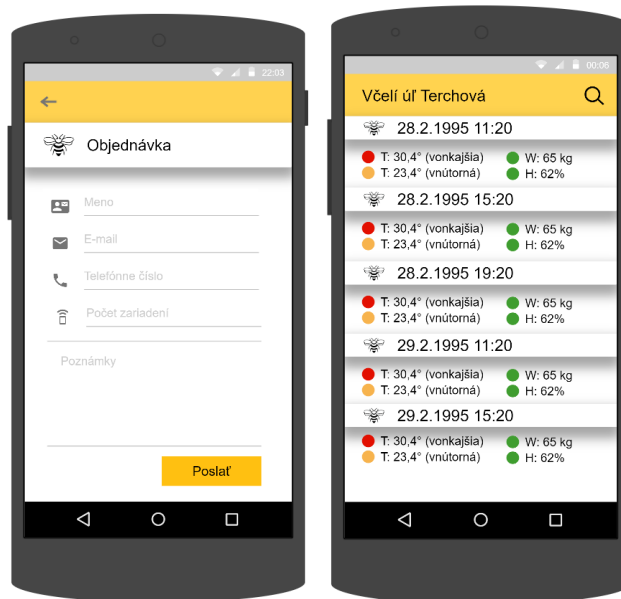
Samotná registrácia používateľa sa vykonáva, keď sa nový používateľ potrebuje zaregistrovať. Príslušné tlačidlo, ktorým sa dostaneme ku registračnému formuláru sa nachádza na úvodnej prihlasovacej obrazovke. Medzi nevyhnutné body formuláru sme zaradili meno používateľa, email a heslo. Pri nesprávne zadaných údajov sa predpokladá, že bude používateľ oboznámený s touto skutočnosťou a pokiaľ správne neupraví obsah daného poľa nebude zaregistrovaný.

Časť odhlásenie sa aktuálne nachádza v ľavom bočnom paneli, ktorý bol implementovaný v predchádzajúcom šprinte. Po kliknutí na odhlásenie bude používateľ ešte vyzvaný na potvrdenie odhlásenia, čím predídeme nechcenému odhláseniu používateľa.

## 5.1.4 Objednávanie včelieho úľa a zobrazenie detailu úľa

Ku objednávke úľa sa používateľ dostane cez ľavý bočný panel. Jednotlivé polia obsahujú meno používateľa, email, telefónne číslo, počet zariadení, ktoré si chce objednať a prípadné poznámky.

Posledným z mockupov štvrtého sprintu bolo navrhnutie zobrazenia detailu úľa. Po kliknutí na jeden konkrétny úľ sa mu zobrazí nová obrazovka s meraniami prislúchajúcimi tomuto úľu. Pôjde o dátum merania, hodnotu vonkajšej a vnútornej teploty, ďalej váhy a vlhkosti.



Obrázok 24 Objednávanie včelieho úľa a zobrazenie detailu úľa

## 5.2 Implementácia

Aplikácia bola implementovaná pre operačný systém Android a implementácia bola realizovaná pomocou vývojového prostredia Android Studio 3.0 od firmy JetBrains. Android aplikácia má za úlohu uľahčiť včelárovi sledovanie aktuálneho stavu úľov. Používateľ sa do aplikácie prihlási pomocou používateľského mena, čiže e-mailovej adresy a hesla. Backend aplikácie zabezpečí zahashovanie hesla a následné odoslanie na REST API serveru včelička. Server overí používateľa v databáze, a ak používateľ zadal správne prihlasovacie údaje prihlási ho do aplikácie. Používateľ si následne môže v aplikácii pozrieť hodnoty namerané senzormi na jeho úľoch. Dáta zo senzorov sa získavajú zo serveru včelička cez jeho REST API.

### 5.2.1 Prihlasovacia stránka

Prihlasovacia stránka obsahuje textové polia na zadanie prihlasovacích údajov e-mailu a hesla. Po ich zadaní a potvrdení stlačením tlačidla Prihlásiť aplikácia najprv overí zadanie e-mailu a hesla. Ak je zadaný platný e-mail a heslo tak overí pripojenie k internetu. Ak je pripojenie v poriadku tak spustí metódu prihlásenia. Táto metóda zostaví http požiadavku typu POST, vytvorí telo tejto požiadavky zo

zadaných parametrov a odošle požiadavku na server. Server skontroluje používateľa v databáze a aj správnosť zadaného hesla a vráti buď kladnú alebo zápornú odpoveď. Ak vráti odpoveď, že všetko je v poriadku tak aplikácia zobrazí základnú stránku úľov, v opačnom prípade zobrazí chybové okienko s hláškou, že používateľ zadal chybné prihlasovacie údaje. Po prihlásení používateľa sa tento používateľ uloží do lokálnej SQLite databázy. Do databázy sa uložia údaje, ktoré vráti server v odpovedi a to jeho osobné údaje a aj vygenerovaný token, pomocou ktorého bude používateľ vykonávať všetky ďalšie akcie. Rovnako sa nastaví aj premenná, ktorá sa použije pri nasledujúcom spustení aplikácie na priame prihlásenie používateľa. Týmto umožníme používateľovi opätovné spúšťanie aplikácie a nebude musieť stále zadávať prihlasovacie údaje.

### **5.2.2 Registračný formulár**

Prihlasovacia stránka obsahuje aj tlačidlo na registráciu nového používateľa. Po kliknutí na toto tlačidlo sa zobrazí registračný formulár. Po vyplnení údajov o používateľovi a stlačení tlačidla “Vytvoriť účet”, registračná metóda skontroluje správnosť zadaných údajov. Ide hlavne o overenie zadania mena a priezviska do jedného poľa s medzerou, platnosti zadaného e-mailu, či obsahuje @ a bodku a nakoniec dĺžku hesla, ktorá musí byť minimálne 8 znakov. Rovnako aj overuje, či sa zhodujú zadané heslá. Ak sú všetky údaje v poriadku, tak metóda ďalej vytvorí http požiadavku typu POST s príslušným telom a odošle tieto dáta na server. Ten overí, či sa používateľ s daným e-mailom už nenachádza v databáze a vráti odpoveď. V prípade negatívnej odpovede aplikácia zobrazí používateľovi chybovú hlášku, že používateľ s daným e-mailom už existuje a umožní mu zmeniť zadané údaje. Ak registrácia prebehne úspešne, tak aplikácia oznámi používateľovi, že registrácia bola úspešná a že sa môže prihlásiť. Aplikácia vráti používateľa automaticky na základnú prihlasovaciu stránku.

### **5.2.3 Objednávka zariadenia**

Bočný navigačný panel hlavnej obrazovky obsahuje aj možnosť objednávky zariadenia. Po kliknutí na túto položku sa používateľovi zobrazí formulár na vytvorenie objednávky. Tento formulár obsahuje informácie o používateľovi ako jeho meno a priezvisko, e-mail, telefónne číslo, počet zariadení a poznámky. Všetky tieto polia sú pred odoslaním dát na server skontrolované, či už správnosť zadania mena a priezviska s medzerou, zadanie platného e-mailu a overenie, či používateľ do polí telefónneho čísla a počtu zariadení zadal len čísla a nie iné znaky. Ak sú údaje správne, tak metóda vytvorenia objednávky vytvorí http požiadavku typu POST s telom, ktoré obsahuje príslušné údaje. Telo požiadavky obsahuje aj ID používateľa a jeho token, ktorý mu server vygeneroval počas prihlásenia. Pomocou tohto tokenu a ID sa overí, či používateľ je naozaj ten za koho sa vydáva a dáta sa uložia do databázy na serveri. Objednávka je takto odoslaná na spracovanie. Poverený správca systému musí túto objednávku overiť a schváliť a následne je možné pokračovať ďalšími akciami na uspokojenie potrieb zákazníka.

## 5.2.4 O projekte

Z bočného navigačného panelu je možné zobrazíť aj aktivitu O projekte. Táto aktivita zobrazuje mená všetkých členov tímu a aj hlavný popis nášho projektu ako aj poďakovanie za používanie nášho produktu.

## 5.2.5 Dôležité triedy

### Trieda: `MainActivity.java` (Android aktivita)

Slúži pre zobrazenie základného prehľadu o úľoch (názov úľa a posledné namerané hodnoty). Taktiež obsahuje aj bočný panel s menu (položky menu: o projekte, profil, notifikácie a odhlásenie ).

#### Dôležité metódy:

- **`onCreate()`** - Slúži pre inicializáciu android aktivity. Inicializuje všetky grafické prvky aplikácie, spustí načítanie zoznamu úľov (metóda `loadHiveNames`) a načítanie posledných nameraných hodnôt pre každý úľ (metóda `loadHiveBaseInfoServerReq`).
- **`loadHiveBaseInfoServerReq()`** - Načíta zo najnovšie hodnoty zo senzorov pre každý úľ (vnútorná/vonkajšia teplota, vlhkosť, stav batérie, hmotnosť úľa a stav akcelerometra). Dáta sa načítavajú cez REST API serveru včelička. Na server sa v žiadosti odosiela názov úľa a prihlasovací token. Prijaté dáta sú vo formáte JSON.
- **`loadHiveNames()`** - Načíta zoznam úľov pre prihláseného používateľa. Dáta sa načítavajú cez REST API serveru včelička. Na server sa v žiadosti odosiela identifikačné číslo používateľa a jeho prihlasovací token. Prijaté dáta sú vo formáte JSON.
- **`hiveClicked()`** - Po kliknutí používateľa na vybraný úľ, otvorí android aktivitu „HiveDetailsActivity“ obsahujúcu detaily pre vybraný úľ (história meraní, grafy, štatistiky atď.). Do vytvorenej aktivity posíela: názov úľa, identifikačné číslo úľa a prihlasovací token používateľa.
- **`onNavigationItemSelected()`** - Po kliknutí na jednu z položiek menu v bočnom menu otvorí novú android aktivitu (profil, o projekte, notifikácie a odhlásenie).
- **`createTestData()`** - Vytvorí falošné dáta slúžiace pre testovanie aplikácie.

### Trieda: `HiveDetailsActivity.java` (Android aktivita)

Slúži pre zobrazenie detailov pre vybraný úľ (história meraní, grafy, štatistiky atď..., aktuálne implementované len zobrazenie histórie meraní). Taktiež obsahuje aj bočný panel s menu (položky menu: o projekte, profil, notifikácie a odhlásenie ).

### **Dôležité metódy:**

- ***onCreate()*** - Slúži pre inicializáciu android aktivity. Inicializuje všetky grafické prvky aplikácie, spustí načítanie histórie nameraných hodnôt (metóda *loadHiveBaseInfoServerReq*).
- ***loadHiveBaseInfoServerReq()*** - Načíta históriu meraní obsahujúcu jednotlivé namerané hodnoty (vnútorná/vonkajšia teplota, vlhkosť, stav batérie, hmotnosť úľa, stav akcelerometra a čas merania). Dáta sa načítavajú cez REST API serveru včelička. Na server sa v žiadosti odosiela názov úľa a prihlasovací token. Prijaté dáta sú vo formáte JSON.
- ***createTestData()*** - Vytvorí falošné dáta slúžiace pre testovanie aplikácie.
- ***onNavigationItemSelected()*** - Po kliknutí na jednu z položiek menu v bočnom menu otvorí novú android aktivitu (profil, o projekte, notifikácie a odhlásenie).

### **Trieda: HiveBaseInfo**

Dátová trieda reprezentujúca informácie o včel'om úle (názov, identifikačné číslo) a nameraných hodnotách z posledného merania (vnútorná/vonkajšia teplota, vlhkosť, stav batérie, hmotnosť úľa, čas merania a stav akcelerometra).

### **Trieda: AdapterHive**

Trieda slúži pre načítanie dát z dátovej triedy (HiveBaseInfo) do grafického prvku *hive\_row* pre zobrazenie základného prehľadu o úľoch (android aktivita „MainActivity“).

### **Trieda: AdapterHiveDetails**

Trieda slúži pre načítanie dát z dátovej triedy (HiveBaseInfo) do grafického prvku *hive\_row* pre zobrazenie detailov pre vybraný úľ (android aktivita „HiveDetailsActivity.java“).

## **5.3 Testovanie**

Táto kapitola opisuje jednotlivé testy funkčných modulov systému. V každom teste je podrobne popísaná akcia a porovnaná reakcia testera s očakávanou reakciou.

### 5.3.1 Testovanie „Main Activity“

Tabuľka 20 Testovanie "MainActivity"

<b>ID:</b>	9	<b>Názov:</b>	Testovanie „MainActivity“ (Android aktivita)		
<b>Úroveň splnenia testu:</b>	Musí		<b>Tester:</b>	Jozef Vaľko	
<b>Rozhranie:</b>	Systém/používateľ				
<b>Účel:</b>	Overenie správnej funkčnosti „MainActivity“ (Android aktivita )				
<b>Vstupné podmienky:</b>	Funkčné pripojenie na internet. Používateľ je prihlásený do aplikácie				
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>		
1	Spustenie aktivity po prihlásení používateľa.	Zobrazí sa prehľad úľov, a najnovších hodnôt, ktoré boli na nich namerané.	Zobrazí sa prehľad úľov, a najnovších hodnôt, ktoré boli na nich namerané.		
2	Potiahnutie prstom z ľavého kraja doprava	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.		
3	Kliknutie na ikonu menu (ľavý horný roh)	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.		
4	Kliknutie na ikonu možnosti (pravý horný roh)	Otvorenie panelu možností. Panel možností obsahuje možnosť „Edituj“	Otvorenie panelu možností. Panel možností obsahuje možnosť „Edituj“		
5	Kliknutie na vybraný úľ	Otvorí sa aktivita zobrazujúca detailne informácie o vybranom úli	Otvorí sa aktivita zobrazujúca detailne informácie o vybranom úli		

### 5.3.2 Testovanie „Login Activity“

Tabuľka 21 Testovanie "LoginActivity"

<b>ID:</b>	10	<b>Názov:</b>	Testovanie „LoginActivity“ (Android aktivita)	
<b>Úroveň splnenia testu:</b>	Musí		<b>Tester:</b>	Jakub Pullmann
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie správnej funkčnosti „LoginActivity“ (Android aktivita )			
<b>Vstupné podmienky:</b>	Funkčné pripojenie na internet.			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Spustenie aplikácie	Zobrazenie aktivity „LoginActivity“	Zobrazenie aktivity „LoginActivity“	
2	Zadanie správnych prihlasovacích údajov (email, heslo). Stlačenie tlačidla prihlásiť	Zobrazenie aktivity „MainActivity“	Zobrazenie aktivity „MainActivity“	
3	Zadanie nesprávnych prihlasovacích údajov (email, heslo). Stlačenie tlačidla prihlásiť	Zobrazenie okna s chybovou hláškou: „Chyba! Nesprávne prihlasovacie údaje“	Zobrazenie okna s chybovou hláškou: „Chyba! Nesprávne prihlasovacie údaje“	
4	Stlačenie tlačidla prihlásiť bez zadania prihlasovacích údajov	Zobrazenie okna s chybovou hláškou: „Chyba! Nesprávne prihlasovacie údaje“	Zobrazenie okna s chybovou hláškou: „Chyba! Nesprávne prihlasovacie údaje“	



### 5.3.3 Testovanie „HiveDetails“

Tabuľka 22 Testovanie „HiveDetails“ (Android aktivita)

<b>ID:</b>	11	<b>Názov:</b>	Testovanie „HiveDetails“ (Android aktivita)	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Barbora Čelesová	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie správnej funkčnosti „HiveDetails“ (Android aktivita )			
<b>Vstupné podmienky:</b>	Funkčné pripojenie na internet. Používateľ je prihlásený do aplikácie			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Kliknutie na vybraný úl v Prehľade úl'ov (MainActivity)	Otvorí sa aktivita zobrazujúca históriu meraní pre vybraný úl (čas a dátum merania, vnútorná/vonkajšia teplota, hmotnosť úľa, vlhkosť a stav batérie)	Otvorí sa aktivita zobrazujúca históriu meraní pre vybraný úl (čas a dátum merania, vnútorná/vonkajšia teplota, hmotnosť úľa, vlhkosť a stav batérie)	
2	Potiahnutie prstom z ľavého kraja doprava	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	
3	Kliknutie na ikonu menu (ľavý horný roh)	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	Zobrazenie bočného panelu (Menu). Bočný panel obsahuje položky: „Profil“, „Notifikácie“, „Odhlásenie“, Názov tímu, Mail tímu a odkaz na webovú stránku.	
4	Kliknutie na ikonu možnosti (pravý horný roh)	Otvorenie panelu možností. Panel možností obsahuje možnosť „Edituj“	Otvorenie panelu možností. Panel možností obsahuje možnosť „Edituj“	

### 5.3.4 Testovanie registrácie používateľa v Android aplikácii

Tabuľka 23 Testovanie registrácie používateľa v Android aplikácii

<b>ID:</b>	12	<b>Názov:</b>	Testovanie registrácie používateľa v Android aplikácii	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Barbora Čelesová	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie funkčnosti registrácie nového používateľa			
<b>Vstupné podmienky:</b>	Funkčné pripojenie na internet.			
<b>Výstupné podmienky:</b>	Registrovaný používateľ			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Pokus o registráciu s prázdnyimi poliami	Zvýraznenie polí s nezadanými údajmi a hláškou „Povinné pole“	Zvýraznenie polí s nezadanými údajmi a hláškou „Povinné pole“	
2	Registrácia s chybnými vyplnenými poliami	Zvýraznenie polí s chybnými údajmi a zobrazenie príslušných hlášok	Zvýraznenie polí s chybnými údajmi a zobrazenie príslušných hlášok	
3	Registrácia so správnymi zadanými údajmi	Zobrazenie oznámenia o úspešnej registrácii a návrat používateľa na prihlasovaciu stránku	Zobrazenie oznámenia o úspešnej registrácii a návrat používateľa na prihlasovaciu stránku	

### 5.3.5 Testovanie vytvorenia objednávky používateľom

Tabuľka 24 Testovanie vytvorenia objednávky používateľom

<b>ID:</b>	13	<b>Názov:</b>	Testovanie vytvorenia objednávky používateľom	
<b>Úroveň splnenia testu:</b>	Musí	<b>Tester:</b>	Barbora Čelesová	
<b>Rozhranie:</b>	Systém/používateľ			
<b>Účel:</b>	Overenie funkčnosti vytvorenia novej objednávky			
<b>Vstupné podmienky:</b>	Funkčné pripojenie na internet a prihlásený používateľ			
<b>Výstupné podmienky:</b>	Vytvorená objednávka v databáze			
<b>Krok</b>	<b>Akcia</b>	<b>Očakávaná reakcia</b>	<b>Skutočná reakcia</b>	
1	Pokus o vytvorenie objednávky s prázdnyimi poliami	Zvýraznenie polí s nezadanými údajmi a hláškou „Povinné pole“	Zvýraznenie polí s nezadanými údajmi a hláškou „Povinné pole“	
2	Vytvorenie objednávky s chybnými údajmi	Zvýraznenie polí s chybnými údajmi a zobrazenie príslušných hlášok	Zvýraznenie polí s chybnými údajmi a zobrazenie príslušných hlášok	
3	Vytvorenie objednávky so správne zadanými údajmi	Zobrazenie oznámenia o úspešne vytvorenej objednávke a návrat používateľa na hlavnú stránku	Zobrazenie oznámenia o úspešne vytvorenej objednávke a návrat používateľa na hlavnú stránku	

## 6 Bibliografia

- [1] M. Curtis, „Arduino Beehive monitor,“ 15 08 2014. [Online]. Available: <https://hackaday.io/project/2453-arduino-beehive-monitor>. [Cit. 24 10 2017].
- [2] Bee Smart Technologies, „Bee smart Technologies,“ [Online]. Available: <https://beesmarttechnologies.com/beebot/>. [Cit. 24 10 2017].
- [3] ProBee, „ProBee,“ [Online]. Available: <http://www.probee.cz/Default>. [Cit. 24 10 2017].
- [4] Arduino, [Online]. Available: <https://www.arduino.cc/en/Guide/Introduction>. [Cit. 19 10 2017].