Slovenská technická univerzita

Fakulta informatiky a informačných technológii

Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt

BREYSLET 2.0

Inžinierske dielo

Názov tímu: Watchmen (tím č. 6)
Web: http://team06-17.studenti.fiit.stuba.sk
Kontakt: watchmenteam2017@gmail.com
Akademický rok: 2017/2018

Vedúci projektu: Ing. Vladimír Kunštár

Členovia tímu: Bc. Michal Oláh,
Bc. Joachim Svítek,
Bc. Dominik Kačmár,
Bc. Martin Škodáček,
Bc. Lam Tuan Anh

Obsah

1 Úvod

Monitorovanie zdravia osôb je spravidla veľmi náročná a namáha činnosť. Hlavným problémom je časová náročnosť a nedostatok ľudských kapacít. Prejavuje sa to okrem iného aj v nemocniciach, kde nie je nezvyčajné, že sa jedna zdravotná sestra musí starať o desiatky pacientov, čo nie je jednoduchá úloha. Tento problém sa však nemusí prejavovať len v oblasti zdravotníctva ako takého, niekedy je taktiež potrebné dohliadať na nám blízke osoby. Nakoľko nie vždy je možné vykonávať takúto službu v dostatočnej kvalite a časovej dostupnosti ako by sme sami chceli, je na mieste, aby sme sa zamysleli, ako by nám v tomto smere mohla pomôcť novodobá technológia? Je možné ľudom zjednodušiť sledovanie zdravia inej osoby a aspoň z malej časti im pomôcť? Napríklad technológia Sigfox ponúka nový rozmer v technológiách, ktorý by sme mohli pri riešení tohoto problému využiť preto sme sa rozhodli preskúmať túto oblasť a navrhnúť riešenie, ktoré by mohlo priniesť pozitívne výsledky.

Naším riešením je náramok Breyslet. Tento náramok má špeciálne nami navrhnutý hardvér, ktorý umožní plnohodnotné využitie siete Sigfox a spolu so sadou ďalších senzorov umožní vzdialené monitorovanie zdravotného stavu osoby. S náramkom spárovaná mobilná aplikácia spravuje všetky dáta z náramku a čo je dôležité, umožňuje inej osobe na diaľku kontrolovať zdravotný stav osoby nosiacej náramok Breyslet. Náramok bude pomocou siete Sigfox odosielať namerané hodnoty ako napríklad teplotu, pulz a okysličenie krvi, a pomocou ďalších senzorov bude detekovať nezvyčajnú aktivitu ako napríklad pád. V prípade akejkoľvek nezvyčajnej aktivity bude náramok schopný notifikovať poverenú osobu, ktorej bude odoslané zhrnutie dát spolu s aktuálnou polohou monitorovanej osoby. Okrem vyššie spomenutých senzorov, bude náramok obsahovať aj moduly sledujúce akceleráciu, gravitačné zrýchlenie a GPS súradnice náramku. Kombinácia práve týchto senzorov nám pomôže zistiť o nositeľovi náramku čo najviac informácií o jeho stave.

2 Globálne ciele pre LS

Globálne ciele projektu na letný semester nadväzovali na stanovený plán na úlohy zo zimného semestra. Na základe retrospektívy zo ZS sme vykonali potrebné návrhové a implementačné úpravy. Pre úlohy spojené s firmvérom, hardvérom, serverovou časťou a aplikáciami sme na základe vykonaných analýzy a úprav pokračovali v implementácií.

2.1 Globálne ciele pre ZS

V tomto projekte vychádzame z riešenia predchádzajúceho tímového projektu Breyslet. Majiteľom produktu je jeden z vtedajších riešiteľov. V našom projekte chceme rozšíriť a zlepšiť ich riešenie. Konkrétne ciele týkajúce sa priamo projektu sú zhrnuté v texte nižšie. Okrem nich sa sústredíme na pravidelné stretnutia, správne prístupy k agilnému vývoju, pravidelné dokumentovanie a verziovanie systému. V rámci zimného semestra sme si s majiteľom produktu vytýčili nasledovné ciele rozdelené do daných oblastí.

Úlohy spojené celkovo s vývojom systému:

- Oboznámenie sa s pôvodným projektom.
- Identifikovanie prípadov použitia.
- Identifikácia funkcionálnych a nefunkcionálnych požiadaviek.
- Identifikácia architektúry systému.
- Návrh dátového modelu.

Úlohy spojené s hardvérom a firmvérom:

- Analýza dostupných hardvérových modulov potrebných k zostaveniu náramku.
- Analýza konkrétnych vybraných modulov pre tento projekt.
- Vytvorenie schémy zapojenia modulov.
- Zapojenie všetkých modulov do funkčného zväčšeného prototypu.
- Naprogramovanie komunikácie medzi modulmi.

Úlohy spojené so serverovou časťou:

- Analýza dostupných technológií pre vytvorenie servera, prostredníctvom ktorého bude náramok komunikovať s aplikáciami.
- Inštalácia servera a implementácia základných funkcií servera pre komunikáciu aplikácií s náramkom.
- Vytvorenie databázy podľa dátového modelu a jej prepojenie so serverom.

Úlohy spojené s aplikáciami:

- Analýza dostupných technológií pre tvorbu responzívnej webovej aplikácie a grafov, ktoré sa budú používať na vizualizáciu údajov nazbieraných z náramku.
- Implementácia základnej funkcionality webovej aplikácia a grafov pre vizualizáciu.
- Analýza a návrh mobilných aplikácií pre Android a iOS.
- Implementácia základnej funkcionality aplikácií.

Ďalšie úlohy sa špecifikujú časom na základe výstupov z predošlých úloh. Podľa splnenia daných plánov pre zimný semester, chceme v nasledujúcom semestri po dôkladnom otestovaní zmenšiť prototyp systému na plošný spoj. V pláne máme aj vytvorenie dizajnu a puzdra pre náramok. Taktiež chceme dokončiť implementáciu servera, všetkých aplikácií a ich vzájomnú komunikáciu.

3 Celkový pohľad na systém

Ako sme na začiatku spomenuli, cieľom tohto projektu je vytvoriť malé nositeľné zariadenie, ktorého úlohou je sledovanie fyziologických procesov v ľudskom tele. Na základe nameraných údajov toto zariadenie vyhodnocuje zdravotný stav nositeľa, ktorý môže byť na diaľku sledovaný prostredníctvom aplikácií. Z tejto základnej špecifikácie a požiadaviek od majiteľa produktu môžeme odvodiť požiadavky na vyvíjaný systém.

3.1 Funkcionálne požiadavky

- sledovanie pulzu, okysličenia krvi, teploty, cukru v krvi a polohy nositeľa náramku,
- detekcia pádu nositeľa náramku,
- privolanie pomoci po stlačení záchranného tlačidla alebo pri nezvyčajnom stave fyziologických procesov,
- vzdialené sledovanie nameraných hodnôt prostredníctvom webovej alebo mobilnej aplikácie

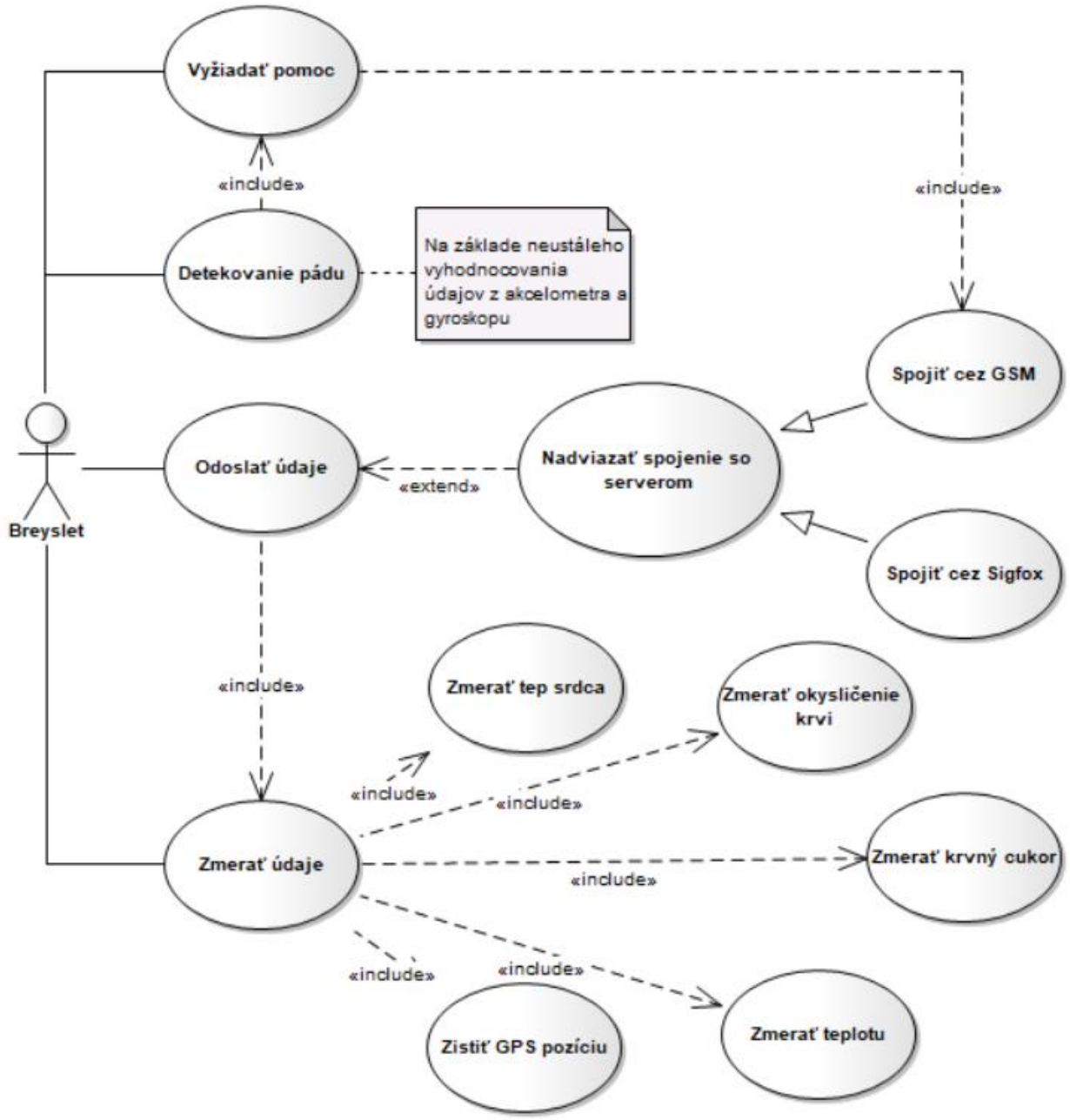
3.2 Nefunkcionálne požiadavky

- spoľahlivosť
- výdrž batérie
- odolnosť
- bezpečnosť
- presnosť
- jednoduché použitie
- malé rozmery a hmotnosť

3.3 Prípady použitia

V tejto časti predstavíme identifikované prípady použitia celého systému.

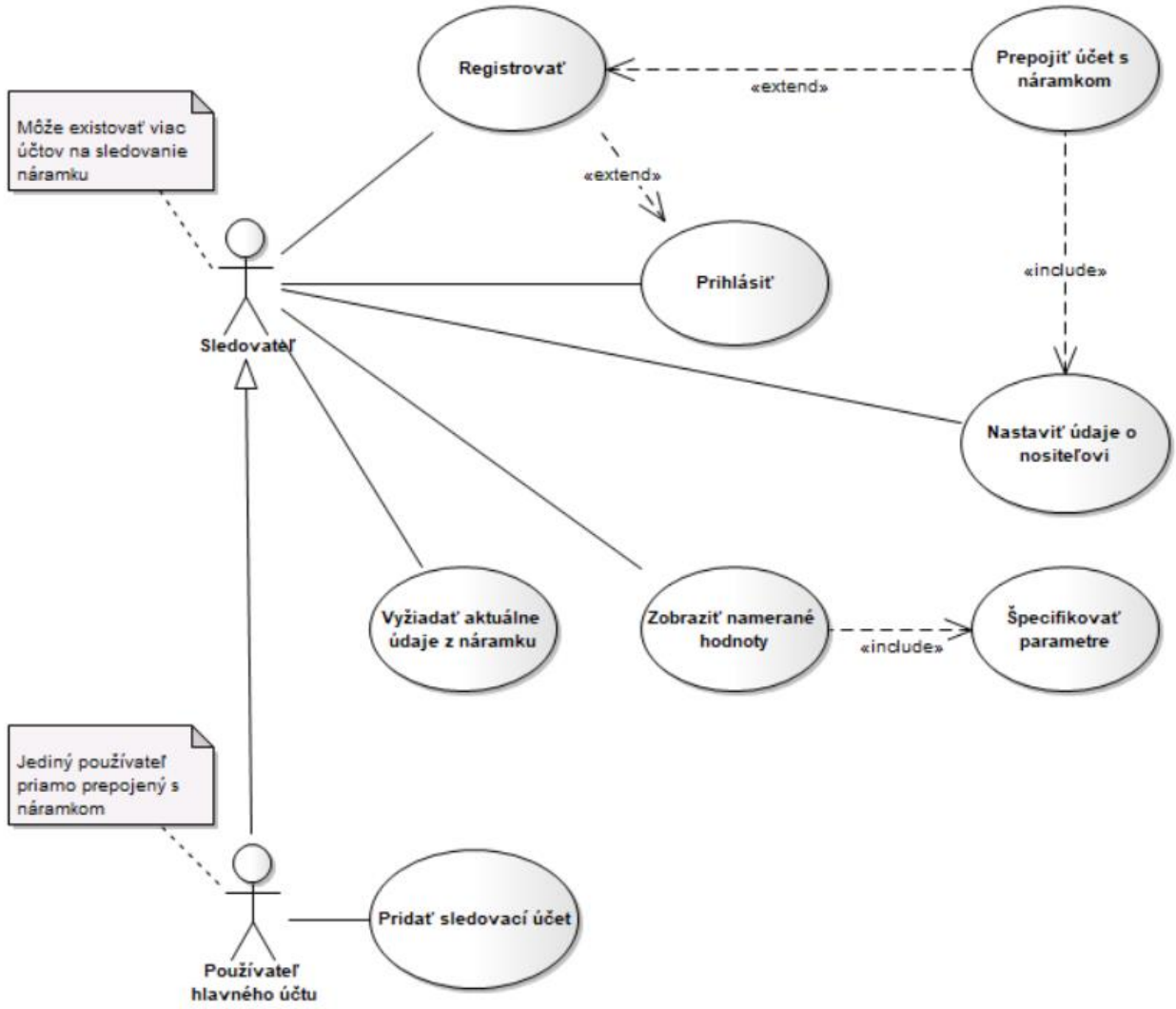
Diagram na obrázku (Obr.1) približuje funkcionalitu vykonávanú samotným náramkom. Približuje proces merania, spracovania a zdieľania získaných údajov.



Obr. 1 - Diagram prípadov použitia - náramok Breylset

Obrázok (Obr. 2) obsahuje diagram prípadov použitia pre možných používateľov systému. Rozhodli sme sa pre vytvorenie dvoch úrovní používateľských účtov. Náramok je prepojený s jedným hlavným účtom (používateľ), ale môže byť sledovaný aj z iných používateľských účtov (sledovateľ). Tak isto ako aj účet sledovateľa, aj hlavný účet môže sledovať údaje z náramku. Avšak len hlavný účet má právo vytvoriť iné účty sledovateľov, ktoré môžu podľa požiadaviek zaniknúť. Hlavný účet nezanikne, pokiaľ nezanikne samotný náramok.

Ako z diagramu vyplýva, hlavnou úlohou týchto účtov je sledovanie údajov z náramku prostredníctvom mobilnej alebo webovej aplikácie.

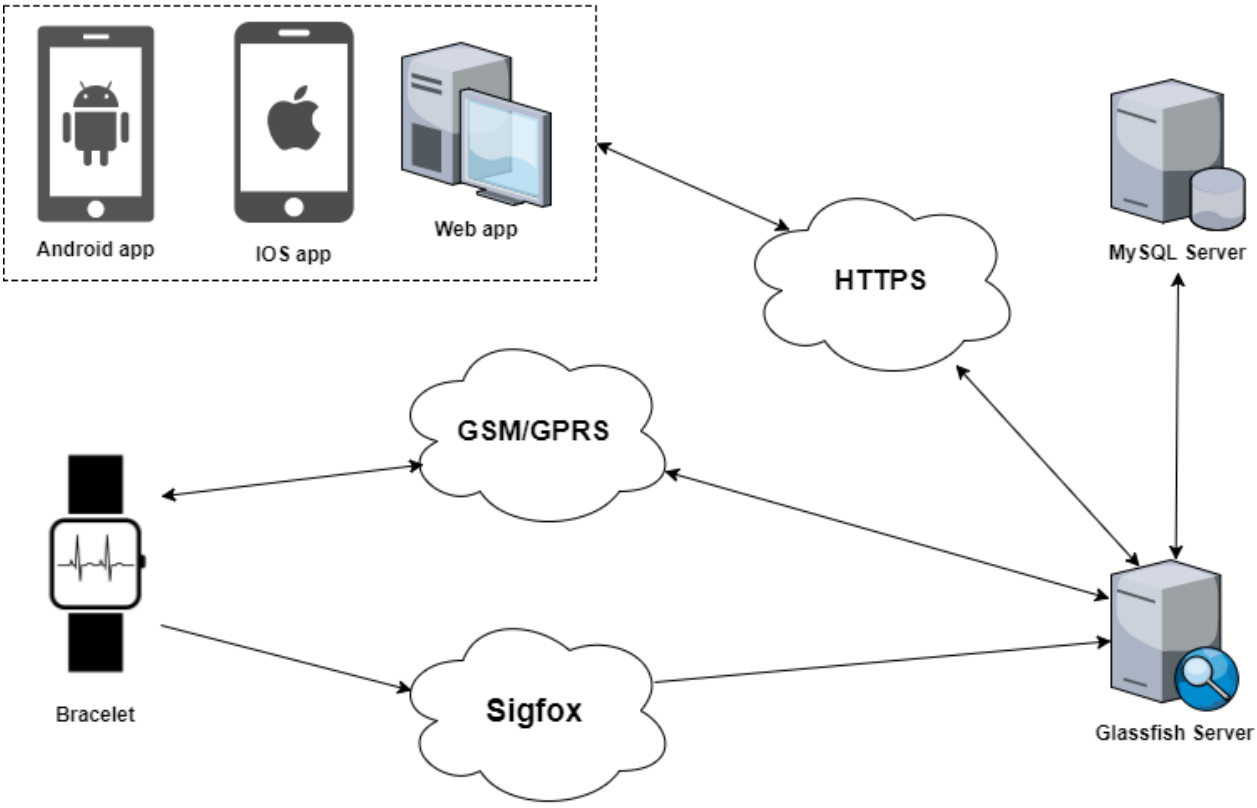


Obr. 2 - Diagram prípadov použitia - používateľ/sledovateľ

3.4 Architektúra

Obrázok (Obr. 3) znázorňuje architektúru celého systému z pohľadu komunikácie a prepojenia jednotlivých častí systému medzi sebou. Centrum celého systému tvorí server, ktorý riadi komunikáciu a tok údajov medzi náramkom, databázou a používateľskými aplikáciami. Aplikácie komunikujú so serverom prostredníctvom REST API. Náramok dokáže, z dôvodu vysokej spoľahlivosti, komunikovať so serverom dvomi spôsobmi a to cez siete Sigfox a GSM. Detailnejšie informácie o komunikácii prostredníctvom týchto sietí sú popísané nižšie v dokumente (kapitola 4).

.



Obr. 3 - Architektúra systému

3.5 Dátový model

Najdôležitejšími entitami, okolo ktorých je všetko v modeli nadviazané, sú entity *Account*, *Breyslet* a *Wearer* (Wearer_info).

*Accoun*t (používateľ) - má svoje osobné údaje, ktorými sa autentifikuje pri prihlasovaní do aplikácií. Používateľ môže mať k svojmu účtu priradených viacero náramkov a tak monitorovať viacero osôb (väzobná tabuľka "breyslet_accounts"), ktoré tento náramok majú tzv. sú nosičmi náramku (Wearer). K používateľovi zbierame informácie o jeho prihláseniach (úspešných/neúspešných).

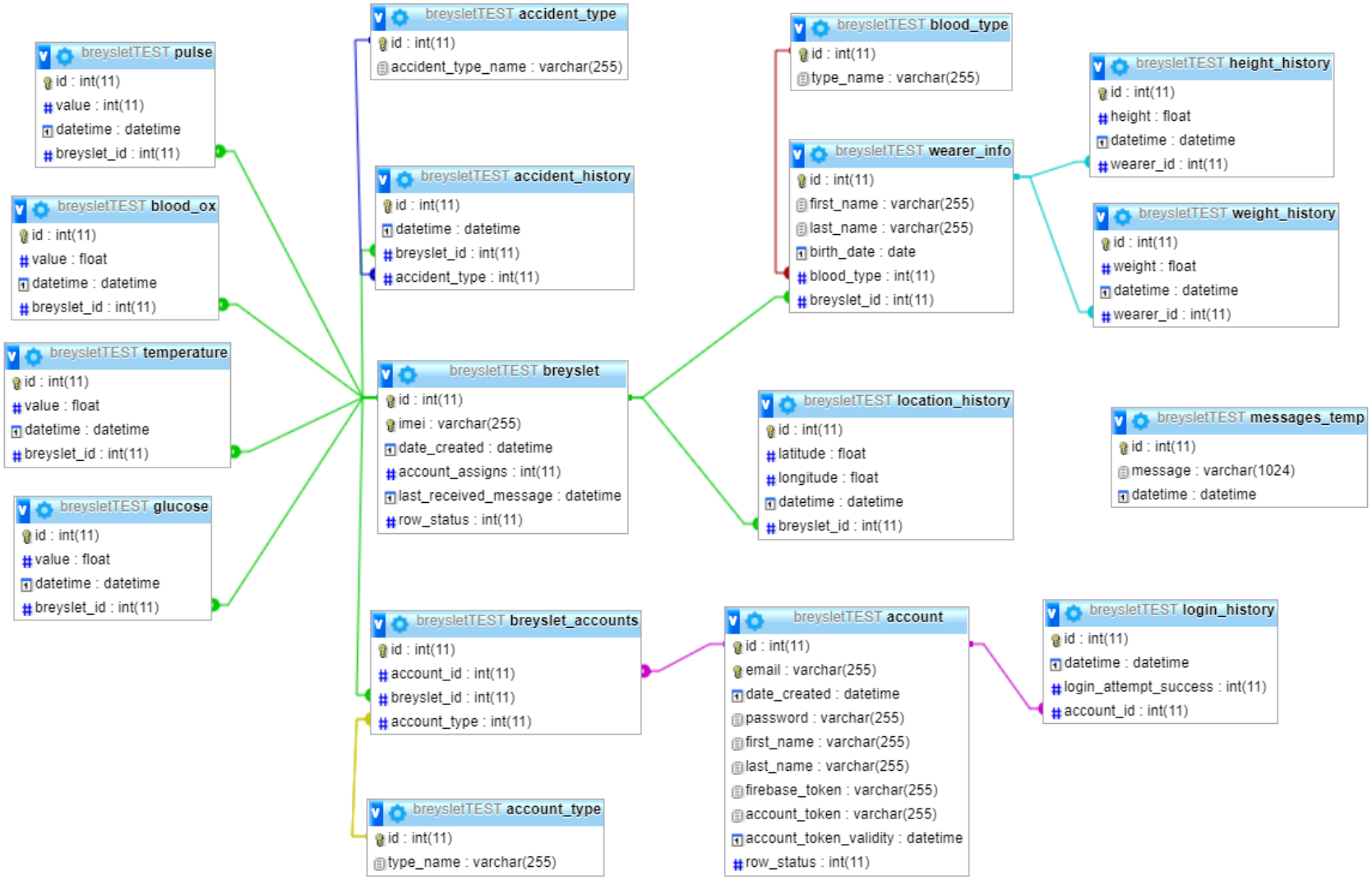
Wearer (nositeľ) - je človek, ktorý je náramkom monitorovaný. O ňom evidujeme užšie informácie ako o samotnom používateľovi, ktoré sú potrebné pre presnejšie vyhodnocovanie zdravotného stavu nositeľa. Evidujeme aj históriu údajov o nosičovi o údajoch, ktoré majú predpoklad sa v budúcnosti vyvíjať, teda napríklad váhu a výšku.

K nositeľovi sú ďalej naviazané všetky informácie, ktoré sa z náramku zbierajú. Návrh modelu je koncipovaný tak, že umožňuje pridávať ďalšie tabuľky, ktoré charakterizujú nové zaznamenávané dáta. Takže ak by hardvér v budúcnosti umožňoval zaznamenávať nové udalosti o nosičovi, bolo by možné to jednoducho zakomponovať do modelu.

K „breyslet"-u (náramku) - sú zaznamenávame do tabuľky (accident_history) všetky udalosti kedy sa niečo, spojené so zhoršením zdravotného stavu, udeje nositeľovi náramku (napr. nositeľ padne na zem). Tieto udalosti je dobré mať oddelené, nakoľko práve týmto udalostiam chceme vďaka monitorovaniu predísť (napríklad ako odpadnutie nositeľa). V location_history evidujeme nameranú históriu polohy nositeľa náramku. Podľa polohy potom vieme, v prípade neočakávanej udalosti, identifikovať kde sa nositeľ nachádza a pomôcť mu. Ku každému náramku sa v databáze evidujú jeho časti (zariadenia). Ku každému zariadeniu prislúcha typ a jeho mac-adresa.

V tabuľke "message_temp" ukladáme všetky správy, ktoré sa pošlú na serverovú časť ešte pred rozpársovaním správy. V prípade prijatia chybnej správy, ktorú by sme nevedeli rozpársovať, si budeme vedieť správu pozrieť a problém identifikovať.

Každá tabuľka v databázovom modeli obsahuje tzv. „row status", ktorý značí to či je daný riadok záznamu aktuálny (1) alebo neaktuálny (0). Teda záznamy v databáze nemažeme, ale len k nemu pridelíme hodnotu pre aktuálnosť. Pri potrebe obnoviť niektorý riadok, napríklad pri typoch zariadenia, stačí opäť nastaviť príslušnú hodnotu na 1.



Obr. 4 - Fyzický dátový model

4 Moduly

4.1 Hardvér

Táto časť je venovaná analýze komponentov a technológií, ktoré sa používajú v takýchto zariadeniach a taktiež tých, ktoré plánujeme použiť pre náš vnorený systém. Niektoré informácie o moduloch sme prebrali z predošlého projektu Breyslet 1.0 so súhlasom vedúceho tímu.

4.1.1 Sigfox modul WSSFM20R1

Sigfox je mobilná sieť, ktorá pracuje v nelicencovanom pásme 868 MHz a predovšetkým je optimalizovaná nie na rýchlosť a objem prenesených dát, ako napríklad siete LTE, ale na minimálnu spotrebu energie.

Sigfox modul umožňuje nášmu Breysletu komunikovať so serverom, posielať a prijímať dáta cez Sigfox sieť. Modul WSSFM20R1 obsahuje aj wifi, bluetooth, GPS, akcelerometer moduly. Komponenty integrované v module WSSFM20R1 sú[1]:

- Sigfox Configuration 2 RC1
- WIFI (2.4GHz) Supports 802.11 b/g/n.
- BLE Support version BT4.2.
- GPS Supports GPS and GLONASS.
- Accelerometer : ±2g/±4g±/8g

4.1.2 GSM modul QUECTEL M66

GSM modul umožňuje náramku posielať SMS správu v mobilnej sieti a v prípade, že bol aktivovaný pohotovostný mód, tak posiela dáta pomocou technológie GPRS. Táto technológia (často nazývaná aj 2,5G) umožnuje beztrôdotové pripojenie k internetu a ďalšie dátové komunikácie v GSM sieti. Modul M66 obsahuje aj bluetooth modul.

Tab. 1 - Parametre modulu QUECTEL M66:

Parameter	**Hodnota**
Napájanie	3,3 ~ 4,6 V
GSM	850/900/1800/1900 MHz
Rozhranie	UART
Rozmer	15.8 x 17.7 x 2.3 mm

Režimy	IDLE, SLEEP, DATA

4.1.3 Procesor ATXMEGA128A1U-AU

Jadrom systému je procesor. Zvolili sme procesor ATXmega 128 A4 postavený na základe RISC architektúry a obsahuje 138 inštrukcií, 32x8 bitových registrov priamo napojených na aritmeticko-logickú jednotku. Vlastnosti procesora sú popísané v tabuľke 2.

Tab. 2 - Parametre procesora ATXMEGA128A1U

Parameter	**Hodnota**
Veľkosť flash pamäte	128kB
Veľkosť RAM pamäte	8kB
Rozsah analógovo-digitálneho prevodníka	12 bit
Počet I/O	78
Napájanie	1.6V - 3.6V
Rozhranie	SPI, TWI, USART
Veľkosť EEPROM pamäte	2kB

4.1.4 Senzor na meranie teploty MAX30205

Teplotný senzor MAX30205 ponúka presné meranie teploty a alarm, prerušenie a následne vypnutie pri prehriatí. Tento senzor konvertuje nameranú teplotu do digitálnej formy sigma-delta analógovo-digitálnym prevodníkom(ADC). Zariadenie poskytuje presnosť, ktorá vyhovuje klinickej termometrickej špecifikácií ASTM E1112.

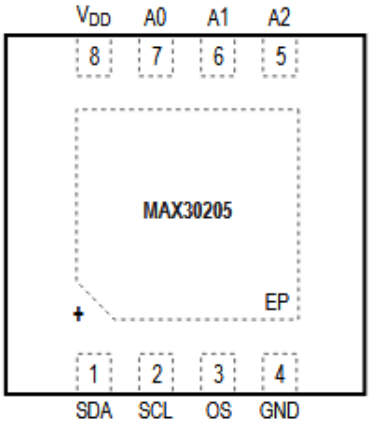
Komunikácia je zriadená cez I2C kompatibilné, 2-drôtové sériové rozhranie. I2C rozhranie využíva štandardné príkazy:

- Write byte
- Read byte
- Send byte
- Receive byte

Tabuľka (Tab.3) udáva hodnoty paramterov a napájania modulu MAX30205

Tab. 3 - Hodnoty parametrov a napájania modulu MAX30205

Parameter	Symbol	Podmienky	Min	Typ	Max	jednotky
Chybovosť termometra	T_{EER}	0°C – +15°C	-0.5		+0.5	°C
		+15°C – +35.8°C	-0.3		+0.3	
		+35.8°C – +37°C	-0.2		+0.2	
		+37°C – +39°C	-0.1		+0.1	
		+39°C – +41°C	-0.2		+0.2	
		+41°C – +45°C	-0.3		+0.3	
		+45°C – +50°C	-0.5		+0.5	
Opakovateľnosť ADC	T_{repeat}	1 Sigma		0.009		°C
Napájanie	V_{DD}		2.7	3.0	3.3	V
Rozsah dát teploty				16		Bits
Čas konverzie				44	50	ms
Odber prúdu	I_{DD}	Neaktívne I^2C, T_A = 0°C – +50°C		600	925	µA
		Shutdown mód, Neaktívne I^2C, T_A = 0°C – +50°C		1.65	3.5	



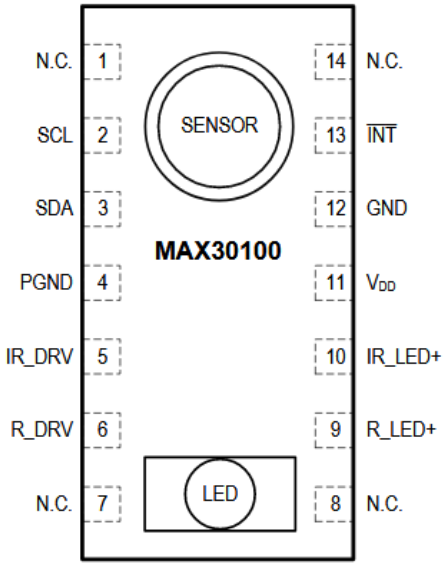
Obr. 5 - Rozloženie pinov modulu MAX30205

4.1.5 Pulzný oximeter a senzor na meranie tepu MAX30100

Modul MAX30100 je integrovaný pulzný oximeter a senzor na meranie tepu. Obsahuje 2 LED svetlá, fotodetektor, optimalizovanú optiku a spracováva nízkošumové analógové signály na detekovanie signálov pulznej oximetrie a srdcového tepu.

Tab. 4 - Hodnoty napájania modulu MAX30100

Parameter	Symbol	Podmienky	Min	Typ	Max	jednotky
Napájanie	V_{DD}		1.7	1.8	2.0	V
LED napájanie (R_LED+/IR_LED+ – PGND	V_{LED+}		3.1	3.3	5.0	V
Odber prúdu	I_{DD}	Meranie tepu + pulzný oximeter aktívny		600	1200	µA
		Meranie tepu		600	1200	
Odber prúdu v shutdown móde	I_{SHDN}	T_A = +25°C, mód = 0x80		0.7	10	µA



Obr. 6 - rozloženie pinov modulu MAX30100

4.1.6 Akcelerometer a gyroskop

Pre šetrenie miesta na doske sme vybrali modul, ktorý obsahuje akcelerometer aj gyroskopJedným z takýchto modulov je MPU-6000. Tento modul bol navrhnutý pre použitie v mobilných telefónoch, tabletoch, a v nositeľných zariadeniach. Využitie akcelerometru nájdeme v troch oblastiach:

- Meranie rýchlosti a polohy
- Snímanie 2 alebo 3 rozmerného pohybu
- Vibračný a nárazový senzor

Gyroskop nám poslúži pre navigáciu v priestore a meranie uhlovej rýchlosti. Podľa ich činností sa delia na:

- Rotačný gyroskop (využíva zákon zachovania momentu hybnosti)
- Gyroskop s vibračnou konštrukciou (meranie prebieha pomocou kapacitných snímačov)

- Optický gyroskop

Špecifikácia modulu MPU6000

- Počet osí: 3+3
- Rozsah merania gyroskopu: 250/500/1000/2000 °/s
- Rozsah merania akcelerometra: 2/4/8/16 g
- Rozhranie: SPI, I2C
- Napájanie: 2.375 ~ 3.46V
- Odber prúdu: 3.9mA

4.1.7 Displej

Na zobrazenie času, a rôznych iných informácií, ktoré zozbierame z modulov a spracujeme, použijeme 0.91 palcový OLED displej. Tento displej obsahuje I2C rozhranie, 14 pinové napájanie, čierne pozadie s vysoko kontrastnými bielymi pixelmi.

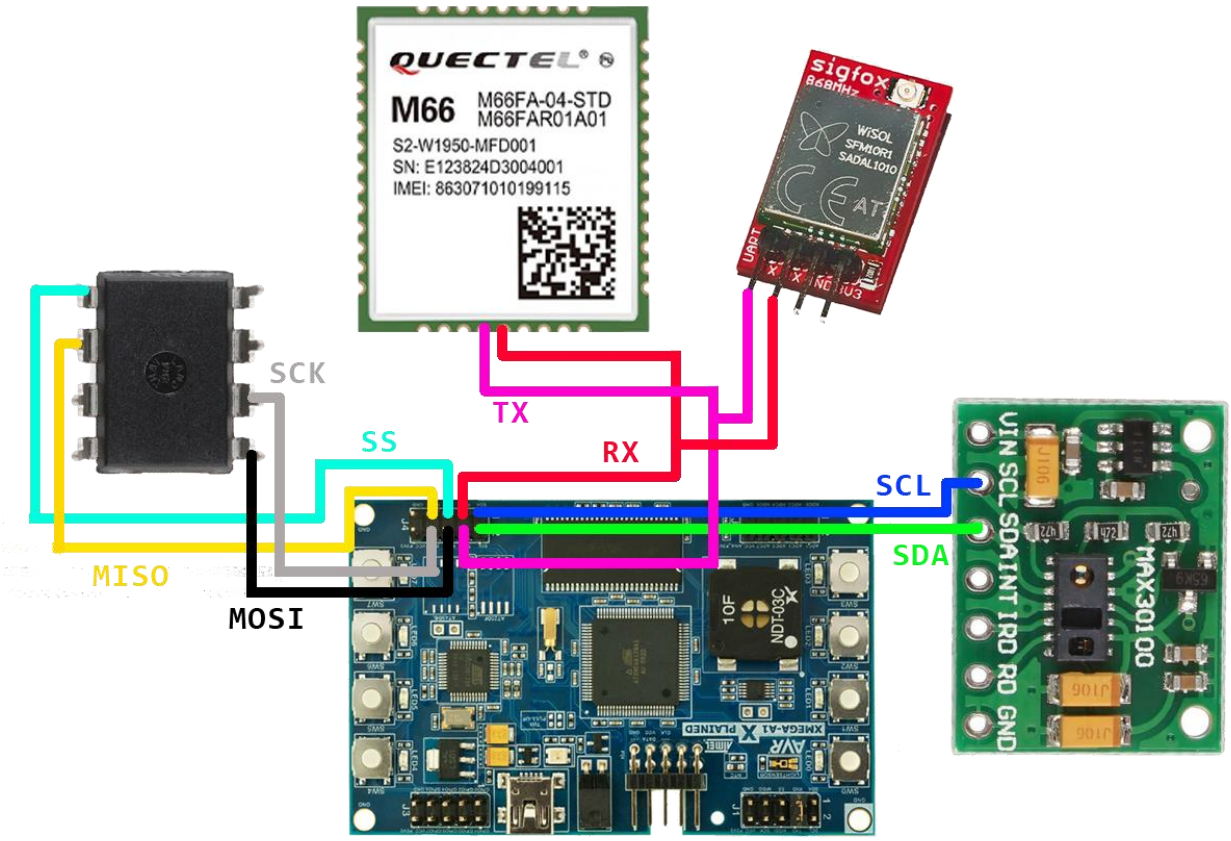
Špecifikácia displeja:

- Rozlíšenie: 128x32
- Typ skla: OLED
- Pozadie: čierne, text biely
- Napájanie: 2.8 – 3.3V
- Napájanie potrebné na rozsvietenie displeja: 7.25V
- Odber prúdu: 6mA – 8mA

4.1.8 Napájanie

Nakoľko naše zariadenie chceme uchovať v nositeľnej veľkosti, použitie batérie sa javí ako jedno z riešení pre vyriešenie problému so zdrojom. Jednou z ďalších riešení získavania energieje použitie fotovoltaických článkov, ktoré konvertujú slnečnú energiu na elektrickú energiu. Pri zapnutí všetkých modulov je odber prúdu ~500mA, čo eliminuje možnosť použitia malých gombíkových batérií. Z uvedených spôsobov získavania energie sme si vybrali použitie batérie, nakoľko fotovoltaické články potrebujú slnečnú energiu na dobíjanie a nemôžeme predpokladať, že potenciálny pacient, ktorý by nosil náramok, bude schopný výjsť na slnko. Od batérie požadujeme kapacitu aspoň 1Ah s napätím 3.7V. Jednou z možných gombíkových batérií, ktoré spĺňajú tieto parametre je batéria TL5134/P, ktorá ale nie je nabíjateľná. Pre naše účely sme zatial nabíjatelnú batériu, ktorá by spĺňala potrebné parametre sme zatiaľ nenašli.

4.1.9 Schéma prototypu



Obr. 7. Schéma prototypu

Na obrázku (Obr. 7) je zobrazená schéma prototypu hardvéru. Na schéme sme zorbazili 3 typy prepojenia medzi MPU(atxmega128a1) a vybranými modulmy(M66, MAX30100, SFM0R1, AT25M02):

- UART
- SPI
- I^2C

Na schéme zatiaľ nie sú zobrazené všetky moduly, ktoré budú použité v zariadení.

4.2 Firmvér

Analýza

Hlavnou úlohou firmvéru je oživenie a prepojenie všetkých používaných modulov. Komunikáciu a všetky akcie bude riadiť procesor ATXMEGA128A1 programom nahraným v jeho pamäti. V tomto projekte používame niekoľko druhov komunikácie medzi procesorom jednotlivými modulmi:

- Universal synchronous and asynchronous receiver-transmitter (USART) je typ sériového rozhrania, ktoré umožňuje, v závislosti od spôsobu naprogramovania, synchrónnu alebo asynchrónnu komunikáciu. Používa piny TX a RX, kde pin TX slúži na vysielanie (transmit) a RX slúži na prijímanie (recieve) bitov.
- Inter-Integrated Circuit (I^2C) je master-slave sériová zbernica, ktorá umožňuje pripojenie viacerých podriadených (slave) zariadení k hlavnému (master) zariadení. Tiež umožňuje pripojenie jedného alebo viacerých podriadených zariadení k viacerým hlavným zariadeniam. Používa piny SDA a SCL, kde pin SDA (Serial Data) slúži na posielanie dát a pin SCL (Serial Clock) slúži na synchronizáciu komunikácie.
- Serial Peripheral Interface (SPI) je zbernica poskytuje sériovú komunikáciu. Obdobne ako I^2C umožňuje komunikáciu medzi hlavným a podriadeným zariadením. Používa piny MOSI, MISO, SCLK a SS. Pin MOSI prenáša výstupné dáta z hlavného zariadenia, zatiaľ čo MISO prenáša výstupné dáta z podriadeného zariadenia. Pin SCLK slúži na synchronizáciu a pin SS slúži na identifikáciu podriadeného zariadenia pri použití viacerých podriadených zariadení.

Návrh

Použitie viacerých senzorov a GPS súčasne s technológiou Sigfox, ktorá limituje počet odosielaných správ na 140 denne, si vyžadovalo navrhnutie presného protokolu obsahu týchto správ. Každá správa môže obsahovať maximálne 12 bytov. Preto sme sa rozhodli odosielať aktuálne namerané údaje každých 15 minút, t.j. 96 správ denne. Zvyšné správy nechávame v rezerve pre núdzové správy. V každej správe sa nachádzajú tieto údaje:

- čas vytvorenia správy,
- stav batérie,
- pulz,
- teplota,
- okysličenie krvi,
- výška cukru v krvi,

- GPS súradnice.

Implementácia

S implementáciou firmvéru sme čakali, kým budú vybrané a dostupné aspoň základné komponenty náramku ako procesor, GSM modul, GPS, modul a niektoré senzory. Dovtedy sme sa oboznamovali so súčiastkami a spôsobmi ako medzi sebou komunikujú. Predovšetkým sme sa sústredili na procesor ATXMEGA128A1 a GSM modul M66. Úspešne sme naprogramovali procesor základnými programami a úspešne sme odoslali HTTPS požiadavku prostredníctvom GSM modulu na náš server, ktorú sme spracovali.

Keďže sa nám dlho nepodarilo rozbehať I2C na procesore ATXMEGA128A1, šesť týždňov do konca projektu sme sa rozhodli pracovať na vývojovej doske Arduino a teda na procesore ATmega328. Na tomto procesore sme nemali problém ani s UART komunikáciou, ani s I2C komunikáciou a úspešne sme zapojili dostupné moduly. Podarilo sa nám namerať tep a odoslať ho cez GSM a Sigfox moduly na server, kde boli následne spracované.

4.3 Back-end

Back-end prepája všetky časti nášho systému. Je kombináciou databázy a aplikačnej logiky napísaného v programovacom jazyku na strane servera. Používateľ (klient) posiela požiadavky odoslané na server, kde serverové API túto požiadavku spracuje, v prípade potreby vytiahne dáta z databázy, upraví ich do požadovanej formy a pošle späť, kde sú interpretované zmenou používateľského rozhrania. Podobne ako na klientskej strane webových a mobilných aplikácií, aj serverová strana webových aplikácií môže byť vyvíjaná množstvom rôznych jazykov a framework-ov.

Na strane servera je možné pracovať s PHP, Node.js alebo Java EE.

Java EE (Java Enterprise Edition) je platforma, ktorá rozširuje Java SE a definuje štandard pre vývoj viacvrstvových enterprise aplikácií, ktoré sú dobre škálovateľné. Java EE poskytuje knižnice pre prístup k databázam (JDBC, JPA), vzdialené volania metód (RMI), webové služby, spracovanie XML a definuje štandardné rozhrania API napríklad pre enterprise JavaBeans, servlety, portlety a JavaServer Pages (JSP). JSP poskytujú metódy pre aktivity, ako je zistenie toho, čo používateľ napísal do textového poľa v online formulári alebo ukladanie súborov cookie do prehliadača používateľa.

PHP (Hypertext Preprocessor) je skriptovací programovací jazyk určený predovšetkým pre tvorbu dynamických stránok a webových aplikácií. PHP súbor môže obsahovať okrem samotného PHP aj HTML, CSS a JavaScript. Je platformovo nezávislý, schopný bežať na

veľkom množstve operačných systémov. V prípade zmeny operačného systému sú skripty prenositeľné bez akýchkoľvek zmien. Taktiež je kompatibilný takmer so všetkými súčasnými servermi. Jeho dôležitou funkciou je podpora širokého spektra databáz, s ktorými pracuje prostredníctvom triedy PHP Data Objects (PDO) , ktorá zabezpečuje napríklad transakcie a vykonáva dopyty.

Java EE v porovnaní s PHP je lepšie škálovateľná aj keď na druhej strane pomalšia. Na základe analýzy, z hľadiska robustnosti a spoľahlivosti a predošlých skúseností, sme vybrali Java EE. V kombinácií s Java EE využijeme nástroj Maven Apache, ktorý sa používa predovšetkým v projektoch Java.

Servlet[1] je trieda jazyku Java, ktorá sa používa na rozšírenie schopností serverov, na ktorých sa nachádzajú prístupové aplikácie na modeli požiadavka-odpoveď (angl. request-response). Aj keď servlety môžu reagovať na akýkoľvek typ požiadavky, sú bežne používané na rozšírenie aplikácií hosťovaných webovými servermi. Pre takéto aplikácie, technológia Java Servlet definuje triedy servletu špecifické pre HTTP (HTTPS).

Balíky javax.servlet a javax.servlet.http poskytujú rozhrania a triedy pre písanie servletov. Všetky servlety musia implementovať rozhranie Servlet, ktoré definuje metódy životného cyklu.. Trieda HttpServlet poskytuje metódy, ako napríklad doGet a doPost, na spracovanie špecifických služieb HTTP.

Maven rieši dva aspekty pri vyvíjaní softvéru: po prvé opisuje, ako je softvér naprogramovaný a po druhé opisuje jeho závislosti. Maven dynamicky sťahuje Java knižnice a moduly z jedného alebo viacerých repozitárov, ako napríklad Maven 2 Central Repository a uloží ich do lokálnej vyrovnávacej pamäte.

4.3.1 Server a databáza

Serverová časť je zložená z webového a aplikačného servera GlassFish[2] vo verzii 4.0, ktorý je dobre dokumentovaný, široko využívaný a je plne podporovaný firmou Oracle. Aplikačná logika na serveri je programovaná v jazyku Java. Pre ľahký prístup, manažovanie, ukladanie a prácu s dátami sme vybrali relačnú databázu MySQL[3] vo verzii 5.7.20 na operačnom systéme Ubuntu. MySQL databáza je dobre dokumentovaná, je open-source a jednoducho implementovateľná v kombinácii spolu s Java.

[1] https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html
[2] https://javaee.github.io/glassfish/
[3] https://www.mysql.com/

Server obsluhuje požiadavky doGet, o ktoré sa stará Java Servlet. Server odpovedá na všetky požiadavky (HTTP request) z aplikácií príslušným „HTTP response“ , pričom ako odpoveď sa posiela JSON s príslušnými dátami.

Komunikácia vie prebiehať prostredníctvom protokolu HTTP a HTTPS. HTTPS[4] stránky obvykle používajú jeden z dvoch zabezpečených protokolov na šifrovanie komunikácie - SSL (Secure Sockets Layer) alebo TLS (Transport Layer Security). Protokoly TLS a SSL používajú takzvané "asymetrické" infraštruktúry verejného kľúča (PKI). Asymetrický systém používa dve kľúče na šifrovanie komunikácie, "verejný" kľúč a "súkromný" kľúč.

Z dôvodu zachovania maximálnej bezpečnosti pri komunikácií klient-server využívame len šifrovaný protokol HTTPS, ktorý obsahuje aj overený certifikát.

Na serveri sa nachádza API (z angl. application progamming interface), ktoré obsahuje zbierku funkcií a tried, ktoré určujú, akým spôsobom sa majú funkcie volať. Používateľské API je v štádiu rozpracovania, preto zatiaľ nie je implementovaná všetka logika.

Používateľské API

- *Registrácia* - metóda slúži na registráciu používateľa do systému. Parametrami sú: meno, priezvisko, platná emailová adresa (slúži ako prihlasovacie meno) a heslo (prihlasovacie heslo).
- *Prihlásenie* - metóda slúži na prihlásenie používateľa do systému. Parametrami metódy sú platná e-mailová adresa používateľa a heslo. Pričom pre prihlásenie je potreba byť už zaregistrovaný v systéme.
- *Priradenie nového používateľa k Breysletu* - metóda slúži na priradenie náramku Breyslet k novo-zaregistrovanému používateľovi buď ako nositeľa alebo pozorovateľa. Parametre sú zatiaľ jedinečné identifikačné meno náramku, podľa ktorého je rozpoznaný konkrétny náramok.
- *Zobrazenie nameraných hodnôt (sumárny prehľad)* – metóda slúži na vrátenie posledných nameraných hodnôt náramkom. Hodnoty v kartičkách na úvodnej obrazovke predstavujú súhrn najaktuálnejších nameraných hodnôt (sumárny prehľad a konkrétne pre pulz, okysličenie krvi a teplotu). Tieto hodnoty sa zobrazia napríklad v mobilných aplikáciách v podobe „kariet“, kde budú vyhodnotené tieto udalosti. Parametrami metódy sú identifikácia používateľa a náramok, pre ktorý chceme zobraziť hodnoty.
- *Zobrazenie histórie nameraných hodnôt* - slúži na zobrazenie histórie nameraných hodnôt buď pre pulz, teplotu alebo okysličenie krvi. Po rozkliknutí kartičky v sumárnom prehľade je k dispozícii výber záujmové obdobia pre ktoré chceme dáta zobraziť a taktiež interval v ktorom majú byť dáta z obdobia zobrazované.

[4] https://www.instantssl.com/ssl-certificate-products/https.html

- *Prijatie správy s nameranými hodnotami* - slúži na prijatie správ zo zariadenia Breyslet, ktoré boli odoslané pomocou GSM alebo SigFoxu. V prijatej správe rozpoznávame okrem samotnej časti s dátami aj typ správy (panická, normálna) a typ zariadenia, ktoré túto správu odoslalo (SigFox, GSM).
- *Panická správa* - v prípade prijatia panickej správy sa okamžite zo servera posiela na všetky zariadenia priradené k danému breysletu upozornenie pomocou FireBase Cloud Messaging.
- *Vygenerovanie meraných hodnôt (testovacie účely)*– Vygenerované dáta korešpondujú s jedným odmeraním všetkých hodnôt a slúžia hlavne pre účel testovania a simulovania rôznych možných nameraných stavov.

Na pozadí servera beží časovač, ktorý poskytuje aplikačný kontajner EJB, ktorý v pravidelných intervaloch (aktuálny interval je 15 minút) kontroluje čas poslednej prijatej správy od všetkých zariadení breyslet. V prípade, že server neobdrží správu viac ako 15 minút (interval v ktorom pravidelne odosiela breyslet namerané údaje pomocou zariadenia SigFox) od niektorého zo zariadení, okamžite vyšle upozornenie cez Firebase Cloud messaging na všetky napojené zariadenia k tomuto náramku.

Testovanie

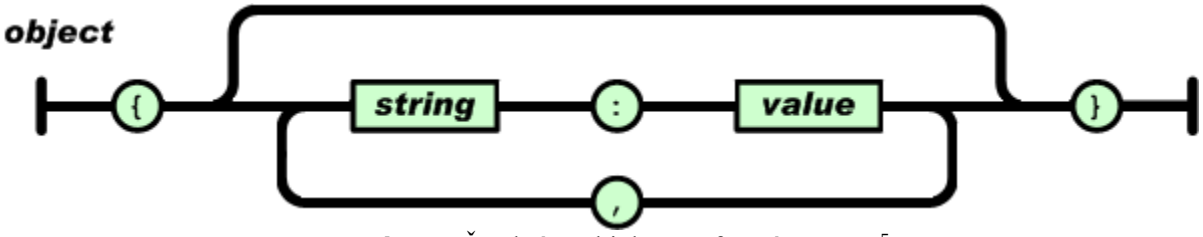
Po každej novo implementovanej funkcionalite vykonávame Unit testy, aby sme overili správnosť implementovania nových funkcionalít. Tým skontrolujeme či náhodou zavedenie nových metód neovplyvnilo správanie sa doteraz už implementovaných častí. Na unit testy využívame JUnit.

4.3.2 Komunikácia

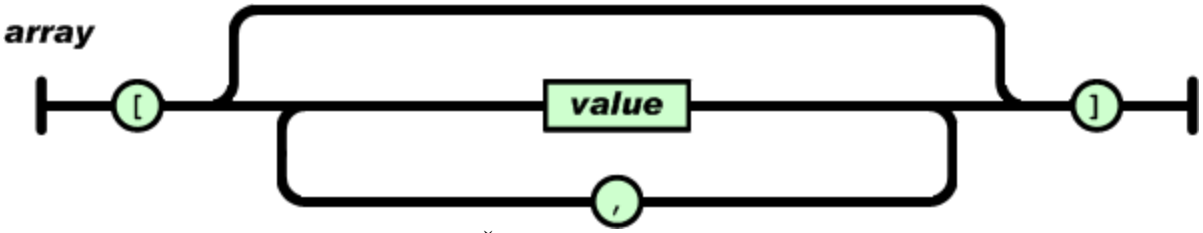
JSON (JavaScript Object Notation) je odľahčený formát, ktorý je jednoducho čitateľný a vytvoriteľný. Vzhľadom k tomu, že formát JSON je len text, môže byť ľahko odoslaný na server a zo servera a tak byť použitý ako dátový formát v ľubovoľnom programovacom jazyku. Je založený na podmnožine jazyka JavaScript (na spôsobe na akom sú vytvárané objekty v jazyku JavaScript)

JSON je postavený na dvoch štruktúrach:

- Zbierka dvojíc názov / hodnota, znázornené na obrázku 8. V rôznych jazykoch je to realizované ako objekt, záznam, štruktúra, slovník, hash, zoznam kľúčov alebo asociačné pole.
- Usporiadaný zoznam hodnôt, znázornené na obrázku 9. Vo väčšine jazykov sa to realizuje ako pole, vektor, zoznam alebo postupnosť.



Obr. 8 - Štruktúra objektu vo formáte JSON[5]



Obr. 9 - Štruktúra poľa vo formáte JSON[6]

Na základe spomínaných výhod sme vybrali a využívame práve JSON na výmenu dát medzi serverom a klientmi.

4.4 Aplikácia pre Android a iOS

Analýza

Medzi najrozšírenejšie operačné systémy pre mobilné telefóny a tablety patrí Android a iOS. Android je rozšírenejší operačný systém ako iOS, ale to najmä preto, lebo je používaný viacerými výrobcami mobilných telefónov a tabletov ako sú napríklad Samsung, Sony, LG, Huawei, Xiaomi a veľa ďalších. Operačný systém iOS používa iba jeden výrobca mobilných telefónov a tabletov a to Apple.

iOS

K programovaniu na iOS potrebujeme špeciálny hardvér a to počítač od firmy Apple, do ktorého si nainštalujeme vývojové prostredie XCode. XCode podporuje viac programovacích jazykov, v ktorých budeme písať našu aplikáciu ako napríklad Objective – C alebo Applom vyvinutý veľmi

[5] http://json.org/object.gif - obrázok prevzatý ku dňu 2.11.2017
[6] http://json.org/array.gif - obrázok prevzatý ku dňu 2.11.2017

mladý programovací jazyk Swift. Swift je založený na Objective-C, ale je menej náchylný na chyby a jednoduchší na pochopenie. Väčšina nových vývojárov si vyberá práve Swift. V prípade vývojových prostredí nemáme na výber a musíme použiť práve vyššie spomenutý XCode priamo od firmy Apple

Android

Naopak k programovaniu na operačný systém Android nepotrebujeme žiaden špeciálny hardvér, a môžeme programovať takmer na všetkom, čo má operačný systém Windows, Linux alebo MacOS, v ktorom budeme mať nainštalované prostredie pre vývoj Andorid aplikácií či už je to Android Studio, Eclipse alebo iné. Android aplikácie môžeme vyvíjať vo viacerých jazykov ako sú napríklad Java, Kotlin alebo C++. Keďže najlepšie poznáme jazyk Java, rozhodli sme sa práve pre tento jazyk.

Návrh

Počas návrhu funkcionality sme identifikovali niekoľko prípadov použitia, týkajúcich sa používania ako mobilných aplikácií, tak aj webovej aplikácie.

1. Registrácia

1. Používateľ zadá svoju emailovú adresu, meno, priezvisko, dátum narodenia a svoje heslo.
2. Používateľ vyberie či je nositeľom náramku alebo len dozerajúca osoba
3. Používateľ stlačí tlačidlo registrovať.
4. Systém skontroluje či existuje účet zaregistrovaný pod zadaný email. V prípade ak áno aplikácia vyzve používateľa aby zadal inú emailovú adresu a use case prejde na krok 3.
5. Spustí sa *Use case 4: Zaslanie overovacieho emailu.*
6. Po úspešnom zadaní overovacieho kódu registrácia dokončená a aplikácia používateľa prihlási.

2. Prihlásenie do aplikácie

Predpoklady: Používateľovi bol zaslaný overovací kód

1. Používateľ zadá meno a heslo.
2. V prípade nezhody bude používateľ vyzvaný znovu zadať heslo.
3. Po 5 neúspešných pokusoch sa spustí *Use case 3: nesprávne zadané heslo 5x.*
4. Po úspešnom zadaní hesla systém skontroluje či bol účet overený.
5. V prípade že účet nebol overený spustí sa *Use case 4: Zaslanie overovacieho emailu.*
6. Aplikácia používateľa prihlási.

3. Nesprávne zadané heslo 5x

1. Používateľovi sa zablokuje možnosť zadávať heslo pre účet ktorému zadal heslo nesprávne 5x na 5 minút (ak sa nesprávne pokusy opakujú časový limit sa predĺži na 10 min, 30 min, 60 min, 24 hodín).
2. V prípade že používateľ stlačí tlačidlo na zresetovanie hesla systém odošle emailovú notifikáciu na účet, ku ktorému sa používateľ snažil prihlásiť, s upozornením na neúspešné pokusy o prihlásenie a unikátnym URL na zmenu hesla.
3. Use case sa ukončí.

4. Zaslanie overovacieho emailu

1. Systém skontroluje či je požadovaný účet (identifikovaný unikátnou emailovou adresou) už overený.
2. V prípade že účet už overený bol use case sa ukončí.
3. Systém odošle email s overovacím kódom a aplikácia vyzve používateľa na zadanie tohoto kódu.
4. V prípade že používateľovi overovací kód neprišiel môže stlačiť tlačidlo "Znovu zaslať overovací kód".
5. V prípade nesprávne zadaného overovacieho kódu aplikácia vyzve používateľa o znovu zadanie overovacieho kódu.
6. Po úspešnom zadaní overovacieho kódu sa use case ukončí.

5. Nastavenia

1. Používateľ chce zmeniť údaje, ktoré zadal pri registrácií.
2. Aplikácia ponúkne používateľa prechod do nastavení.
3. Používateľ prejde do nastavení a upraví si údaje a potvrdí zmeny.
4. Aplikácia uloží zmeny na server a informuje používateľa o úspešnosti.
5. Use case sa ukončí.

6. Spárovanie s náramkom

1. Používateľ vyberie možnosť "Spárovať aplikáciu s náramkom".
2. Aplikácia vyzve používateľa o vykonanie akcie na náramku.
3. Používateľ zadá unikátny kód z náramku.
4. Aplikácia odošle požiadavku na server ktorý požiadavku spracuje a spáruje aplikáciu s náramkom.
5. Use case sa ukončí.

7. Zapnutie aktívneho sledovania (životných funkcií)

1. Používateľ chce zmerať životné funkcie chorého.
2. Používateľ v aplikácií zvolí možnosť aktívneho sledovania životných funkcií.
3. Aplikácia sa spojí so serverom. (spustí sa *Use case 10: Nadviazanie spojenia so serverom*).
4. Server nadviaže spojenie s Breysletom a vyzve ho aby zapol GSM modul a začal merať všetky životné funkcie.
5. Breyslet zapne GSM modul a spustí sa *Use case Aktívne odosielanie údajov z Breysletu.* (tento use case je use case pre firmware).
6. Aplikácia aktívne prijíma a zobrazuje live údaje z Breysletu.
7. Po uplynutí časového intervalu server vyzve Breyslet aby ukončil *Use case Aktívne odosielanie údajov z Breysletu.*
8. Aplikácia oznámi ukončenie aktívneho merania.
9. Use case sa ukončí.

8. Zobrazenie histórie nameraných životných funkcií

1. Používateľ si chce zobraziť namerané hodnoty.
2. Používateľ v aplikácii zvolí možnosť zobrazenia nameraných hodnôt.
3. Aplikácia mu zobrazí namerané hodnoty spolu s dátumom a časom merania.
4. Use case sa ukončí.

9. Notifikovanie používateľa aplikácie o nehode (napr. pád)

1. Aplikácia prijme od serveru urgentnú notifikáciu o nehode alebo nezvyčajnej aktivite.
2. Spustí sa Use case 7: Zapnutie aktívneho sledovania (životných funkcií) od kroku 3.
3. Aplikácia používateľovi ponúkne kontaktovať núdzovú linku 112 prípadne osobu ktorú používateľ zvolil ako nultú pomoc (napríklad sused nositeľa náramku ktorý ho môže rýchlo skontrolovať).
4. Use case sa ukončí.

10. Nadviazanie spojenia so serverom

1. Aplikácia sa pokúsi nadviazať spojenie so serverom prostredníctvom internetového pripojenia.
2. Aplikácia sa úspešne spojila so serverom.
3. Use case sa ukončí.

11. Update dát zo servera (refresh)

1. Používateľ sa pripojí na internet a spustí aplikáciu.
2. Spustí sa *Use case Nadviazanie spojenia so serverom*.
3. Aplikácia požiada server o aktuálne dáta.
4. Server pošle dáta do aplikácie.
5. Aplikácia zaktualizuje údaje a zobrazí ich používateľovi.
6. Use case sa ukončí.

12. Identifikácia chyby v spojení

1. V prípade že sa aplikácii nepodarí získať údaje z Breysletu za posledných 15 minút, aplikácia skontroluje, či je dostupné pripojenie na internet.
2. V prípade že pripojenie na internet dostupné nie je Use case sa ukončí s identifikovaným problémom "Žiadne pripojenie na internet".
3. Aplikácia skontroluje spojenie so serverom.
4. V prípade že sa spojenie so serverom nepodarí nadviazať Use case sa ukončí s identifikovaným problémom "Nepodarilo sa nadviazať spojenie so serverom".
5. Aplikácia vyzve server aby skontroloval spojenie s náramkom.
6. V prípade že server odpovie že sa spojenie s náramkom nepodarilo Use case sa ukončí s identifikovaným problémom "Nepodarilo sa nadviazať spojenie s Breysletom".
7. Use case sa ukončí.

13. Aktívne zobrazovanie live údajov z Breysletu (náramok - server - aplikácia)

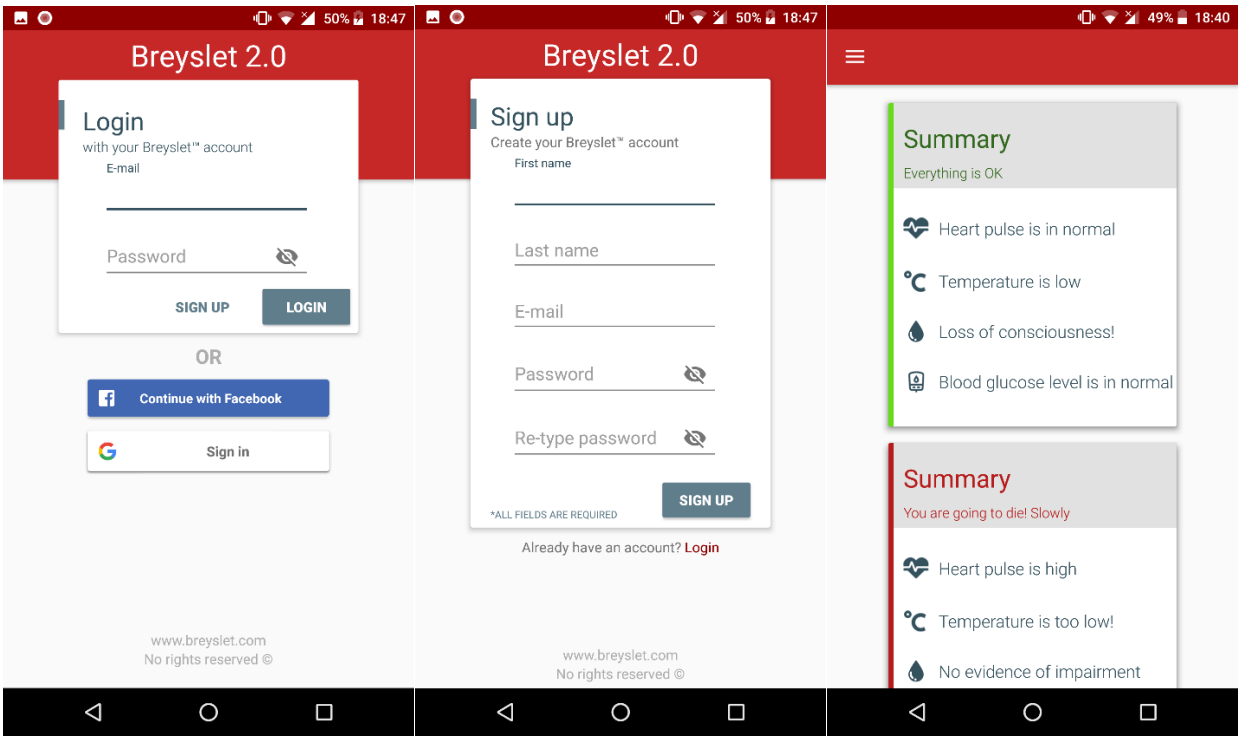
1. Aplikácia prijme dáta zo serveru a graficky ich zobrazí.
2. V prípade straty pripojenia sa spustí Use case 12: Identifikácia chyby v spojení (náramok - server - aplikácia).
3. Aplikácia skontroluje či server neoznámil ukončenie aktívneho odosielania live údajov.
4. V prípade že server ukončil aktívne odosielanie Use case sa ukončí a aplikácia zobrazí oznámenie o uplynutí doby aktívneho sledovania.
5. Use case pokračuje na krok 1.

Implementácia

Android aplikácia je implementovaná v jazyku JAVA. Používateľské rozhrania je napísané v značkovacom jazyku XML. Vývojové prostredie, v ktorom bola implementovaná aplikácia pre Android je Android Studio.

Aplikácia pre iOS je implementovaná v jazyku Swift. V prostredí XCode, kde bola vyvíjaná aplikácia pre iOS sa používateľské prostredie iba vyklikáva. Nie je na to potrebný žiadny jazyk. Keďže k vyvíjaniu aplikácie na iOS je potrebný špeciálny hardvér, zvolili sme Macbook Pro.

Mobilné aplikácie komunikujú so serverom pomocou HTTP resp. HTTPS requestov. Aplikácia odošle serveru žiadosť o informácie o konkrétnom náramku. Server výzvu spracuje z databázy vyberie všetky vyžiadané hodnoty a odošle ich vo forme JSON súboru naspäť aplikácii. Aplikácia zobrazí získané údaje. Na obrázku (Obr. 10) sú zobrazené snímky obrazovky z aplikácie Android.



Obr. 10 Snímky obrazovky z Android aplikácie. Zľava: prihlásenie, registrácia, zobrazenie získaných údajov.

Testovanie

Korektnosť implementácie bude prebiehať unit testovaním ale aj používateľským testovaním, do ktorého zapojíme nie len členov tímu ale aj používateľov, ktorí neboli pri vývoji Breysletu. Od toho očakávame užitočné návrhy na zlepšenie, či už používateľského rozhrania alebo funkcionality.

Firebase Cloud Messaging

Je medzi platformová služba, ktorá slúži na zjednodušené posielanie správ medzi serverom a klientskym zariadením, čo v našom prípade predstavuje Android a iOS aplikácia. Hlavným

dôvodom pre využitie tejto služby je fakt, že sa jedná o odporúčanú službu pre rýchle a spoľahlivé odosielanie notifikácii zo servera na Android zariadenia. Okrem iného nám táto služba poskytuje možnosť nielen odosielania notifikácii ale akýchkoľvek správ zo servera na klientske zariadenie alebo naopak.

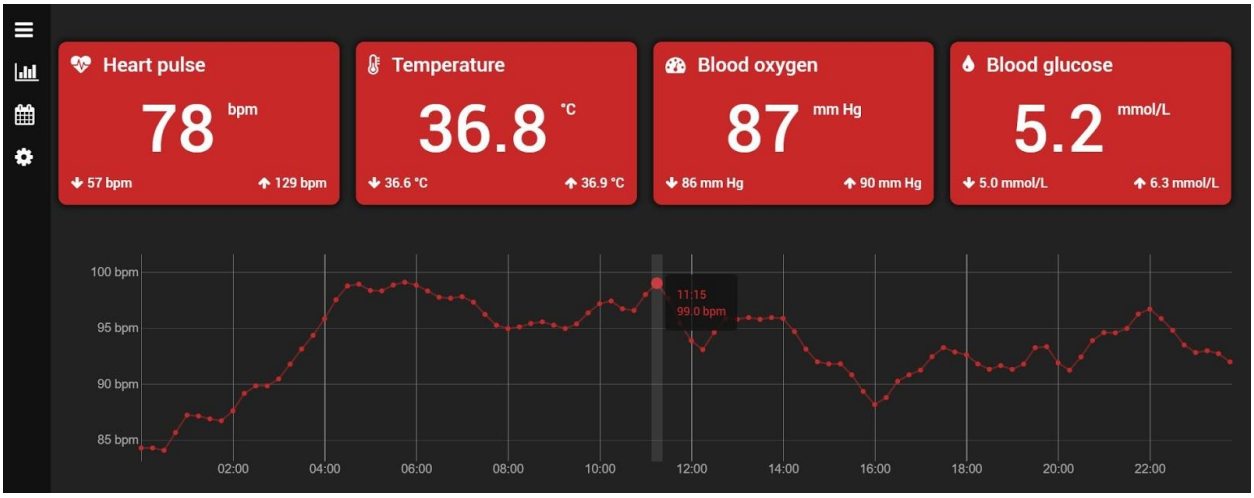
4.6 Front-end

Analýza

V analýze tejto časti sme sa rozhodovali, ktoré technológie použijeme pri tvorbe webovej aplikácie. Tiež sme analyzovali metódy, ktorými by sme efektívne prepojili aplikáciu so serverom. Dôležitou súčasťou aplikácie je vizualizácia dát do grafov, preto sme analyzovali aj dostupné knižnice zaoberajúce sa touto problematikou. Snažili sme sa vybrať technológie tak, aby čo najlepšie vyhovovali vyššie špecifikovaným požiadavkám.

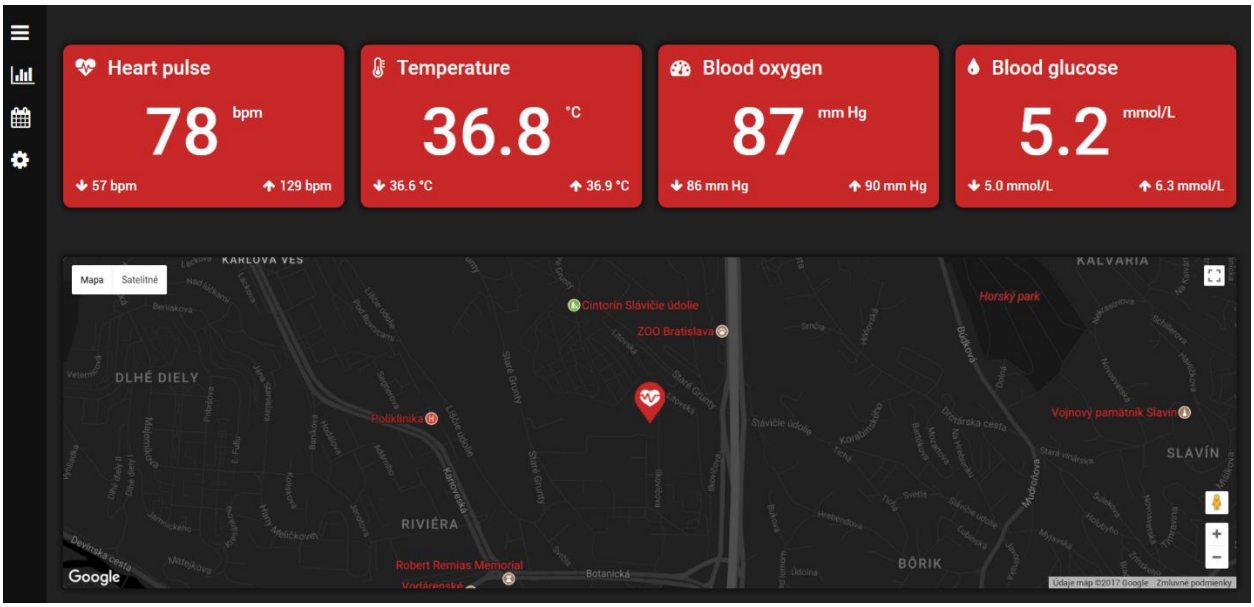
Návrh

Po analýze sme sa rozhodli, že webová aplikácia bude vytvorená prostredníctvom frameworku Bootstrap v. 3, čím zabezpečíme respoznzivitu základných prvkov aplikácie. Grafy, ktorými budú vizualizované údaje z náramku, implementujeme knižnicou D3.js. Táto knižnica uľahčuje vytváranie a manipuláciu s SVG elementami. Responzivita týchto grafov musí byť dodatočne zabezpečená prostredníctvom JavaSriptu. Obrázok (Obr. 11) približuje základný návrh dashboardu webovej aplikácie.



Obr. 11 - Dashboard webovej aplikácie

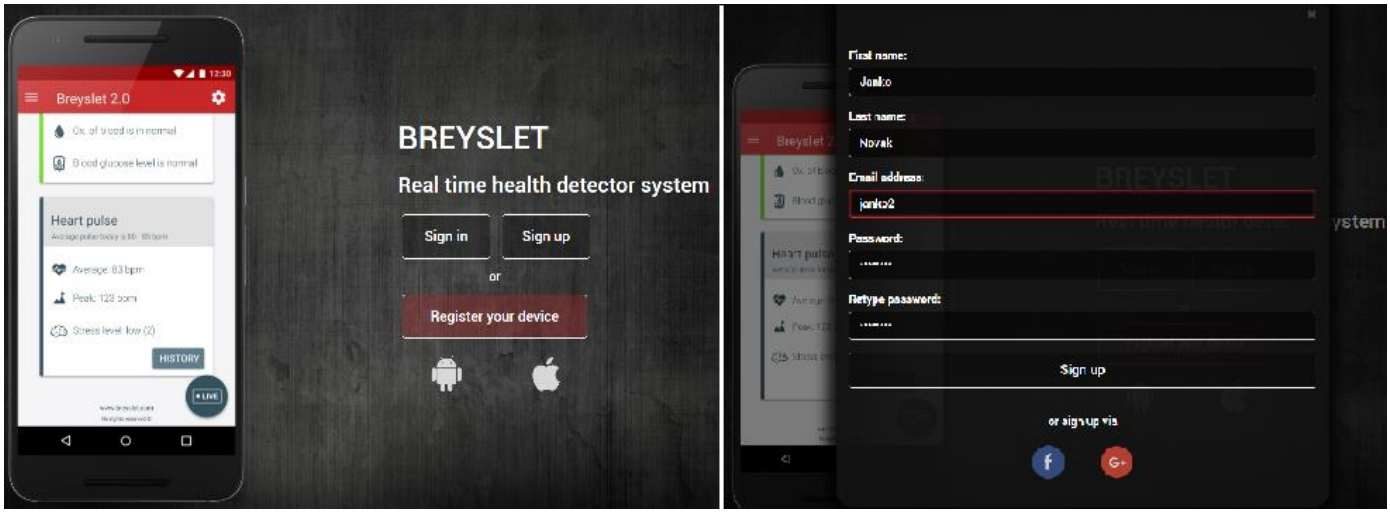
Po prihlásení sa používateľovi zobrazí dashboard so štyrmi blokmi reprezentujúcimi dané fyziologické procesy. Po kliknutí na príslušný blok sa vytvorí graf s dátami nameranými za daný deň. Na dashboarde je zobrazená aj Google mapa s aktuálnou pozíciou náramku (Obr. 12).



Obr. 12 - Dashboard s Google mapou

Ďalej aplikácia používateľovi poskytuje vybrať a zobraziť údaje o nositeľovi náramku z minulosti.

Aplikácia samozrejme poskytuje aj základné funkcie ako prihlásenie, registrácia a nastavenia. Vzhľad propagačnej, registračnej a prihlasovacej stránky systému je zobrazený na obrázku (Obr. 13).



Obr. 13 - Prihlasovacia, registračná a propagačná stránka

Implementácia

Dôležitou časťou webovej aplikácie je komunikácia so serverom. Toto je zabezpečené HTTP resp. HTTPS požiadavkami smerovanými na počúvajúce HTTPServlety servera postavenom na Jave EE. Požiadavky sú vykonávané prostredníctvom CORS[7] (Cross-Origin Resource Sharing) mechanizmu a JavaScript objektu XMLHttpRequest[8]. V požiadavkách a odpovediach sa odosielajú JSON objekty s dátami.

Testovanie

Správnosť návrhu implementácie sa bude testovať hlavne pomocou používateľského testovania, do ktorého zapojíme členov tímu, ale aj neskúsených používateľov, ktorí nám môžu priniesť užitočné návrhy na zlepšenie používateľského rozhrania.

4.6.1 Breyslet simulátor

Pri vývoji hardvéru a softvéru sme potrebovali, aj na účely testovania, dáta, s ktorými by sme vedeli reálne v aplikáciách pracovať a predstavovali by nateraz fiktívne namerané hodnoty. Pre tento účel sme sa rozhodli implementovať tzv. simulátor breysletu, ktorý by predstavoval webovú aplikáciu, cez ktorú by sme si vedeli zasimulovať proces odoslania dát na server tak, ako by to bolo realizované cez náramok breyslet.

Návrh

Breyslet simulátor sme navrhli tak, aby dokázal odoslať dáta vo formáte správy, ktorý sme si definovali a ktorá je identická so správou odoslanou pomocou zariadenia SigFox, ktorá je súčasťou náramku breyslet.

Implementácia

Breylest simulátor je implementovaný ako webová aplikácia obsahujúca formulár, do ktorého je možné zadať presne tie isté hodnoty, aké je schopný namerať a odoslať náramok. Vo webovej aplikácii je implementovaný rovnaký algoritmus parsovania dát na 12 bytový frame, ktorý odosiela modul sigfox. Následne je tento frame poslaný na rovnaký servlet ako odosiela modul sigfox, prostredníctvom XMLHttpRequest.

[7] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
[8] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

4.7 Dizajn

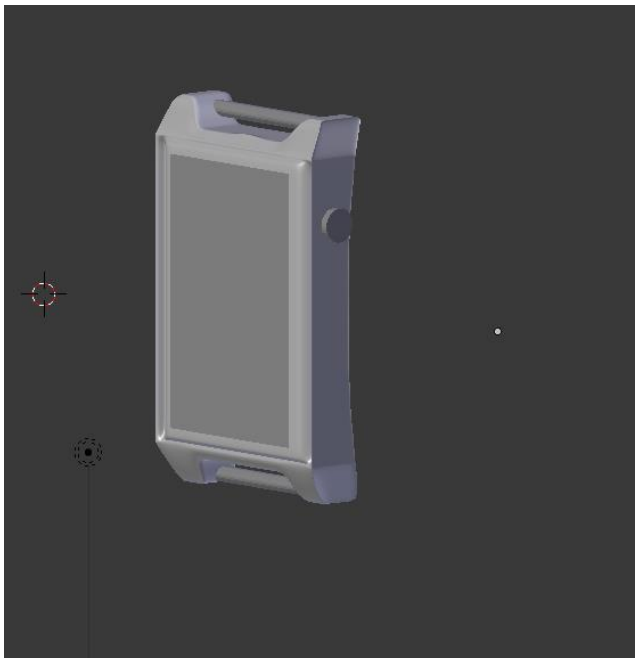
Analýza

Aby Breyslet spĺňal definované požiadavky, jeho dizajn musí spĺňať tieto kritériá:

- musí pevne držať na ruke používateľa,
- musí byť čo najmenší, aby neprekážal pri pohybe používateľa,
- musí pokryť všetky potrebne moduly,
- musí byť nárazu a vode-odolný.

Návrh

Po analýze sme sa rozhodli vytvoriť návrh nástrojom Blender, ktorý podporuje 3D dizajn. Na obrázku (Obr .14) je prvý návrh puzdra s veľkosťou 3,5cm x 5 cm. Má jedno núdzové tlačidlo na ľavom okraji a display pokrývajúci čo najväčšiu plochu. Z analýzy vyplýva, že použitým materiálom na puzdro bude gumy. Na ďalšom obrázku (Obra. 15) je farebný dizajn s displejom.



Obr. 14 – 3D návrh náramku

Obr. 15 – Realistický návrh