

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Tímový projekt
E-health - Monitorovanie
zdravotného stavu pacienta
Inžinierske dielo

Číslo tímu: 18

Vedúci tímu: Ing. Jaroslav Erdelyi

Členovia tímu: Bc. Milan Bohňa
Bc. Martin Hradský
Bc. Florián Chmelár
Bc. Ján Lenický
Bc. Jozef Olejník
Bc. Samuel Sagan

Ak. rok: 2020/2021

Úvod	5
Globálne ciele projektu na letný semester (Floro)	6
Celkový pohľad na systém	7
Výsledky projektu	7
Architektúra systému	7
Meracie zariadenia Arduino	8
Zariadenia na strane klienta	8
Front End	8
Backend	8
Databázový server	9
Architektúra serverovej časti	9
Celková funkcionálnosť	10
Dátový model	11
AspNetUsers	11
Alarms	12
Events	12
UserConnections	12
Monitorings	12
EMGs	12
Pressures	12
EKGs	12
Oxymeters	12
Temperatures	12
DoctorTypes	12
__EFMigrationsHistory	13
Moduly systému	14
Login	14
Analýza	15
Návrh	15
Implementácia	21
Implementácia endpointov spojených k modulu Login	23
Hlavná štruktúra modulu login:	28
Spájanie používateľov	30
Analýza	30
Návrh	30
Prípady použitia	30
Náčrty okien modulu	33
Okno na zaslanie žiadosti o spojenie	33
Okno zoznamu žiadostí	33
Implementácia	34
Frontend	34
Okno Zoznam žiadostí	34

Komponenty v okne osobného profilu používateľa	35
Komponent v okne vyhľadávania	36
API endpointy	36
UserConnection - send request	36
UserConnection - change state	37
UserConnection - list	38
Monitoring	39
Modul Monitorovanie	39
Nadviazanie spojenia	39
Analýza	39
Návrh	39
Implementácia	41
Spustenie merania	42
Analýza	42
Návrh	42
Implementácia	44
Implementácia endpointov pre monitorovanie:	48
Profil	54
Analýza	55
Návrh	55
Implementácia	56
Vyhľadávanie	64
Analýza	64
Návrh	64
Prípady použitia	64
Náčrty okien modulu	65
Okno vyhľadávania	65
Implementácia	65
Frontend	65
Okno Vyhľadávania	65
API endpointy	66
User - list patients	66
User - list doctors	68
Alarmy	68
Vytvorenie alarmu	69
Analýza	69
Zobrazenie všetkých alarmov používateľa	71
Analýza	71
Vymazanie alarmu	73
Analýza	73
Vytvor udalosť	74
Analýza	74
Príručky	76

Inštalačná príručka	76
Používateľská príručka	76
Webová aplikácia	76
Mobilná aplikácia	82
Technická dokumentácia	83
Webová aplikácia	83

Úvod

Tento dokument obsahuje dokumentáciu k inžinierskemu dielu projektu Monitorovanie zdravotného stavu pacienta (E-Health) v rámci projektu Tímový projekt.

V prvej časti opisujeme Globálne ciele na letný semester, následne prinášame celkový pohľad na náš projekt - architektúru projektu, dátový model a moduly systému, ktoré sú v ďalšej časti konkrétne rozobrané. Jedná sa o Login, Spájanie používateľov, Monitoring, Vyhľadávanie, Alarmy a Profil používateľa.

Globálne ciele projektu na letný semester (Floro)

Hlavným cieľom letného semestra bolo najmä na implementovať ucelený produkt, ktorý bude mať sfunkčnené všetky základné požiadavky product ownera. Jedná sa o mobilnú aplikáciu do ktorej bude možné prihlasovať sa a sledovať jednotlivé naplánované monitoringy pacientov. Taktiež sme chceli mať spojzdených čo najviac monitorovacích senzorov - najmä teplotu, EKG a EMG. Ďalšou časťou bola webová aplikácia, ktorá mala umožniť prihlasovanie, registráciu, vyhľadávanie iných používateľov, spájanie sa s nimi, prezeranie profilov, spúšťanie monitorovaní a prezeraní výsledkov z nich v grafoch.

Ďalším cieľom bolo zdokumentovanie celého produktu. Jednak v našom internom nástroji Confluence, ale aj prostredníctvom Inžinierskeho diela. Zdokumentovať sme chceli aj to, akým štýlom sme projekt riadili a postupnosť s akou boli vypracované jednotlivé príbehy a úlohy.

Nemenej podstatným cieľom bola výuka jednotlivých členov a skúsenosť s prácou na väčšom projekte vo väčšom tíme ľudí.

Keď hodnotíme globálne splnenie našich cieľov, tak môžeme povedať, že hlavné ciele sme dokázali splniť. Problémy sa vyskytli napríklad pri priebežnej dokumentácií, neskorom plnení niektorých príbehov, či pri doladzovaní detailov.

Celkový pohľad na systém

Výsledky projektu

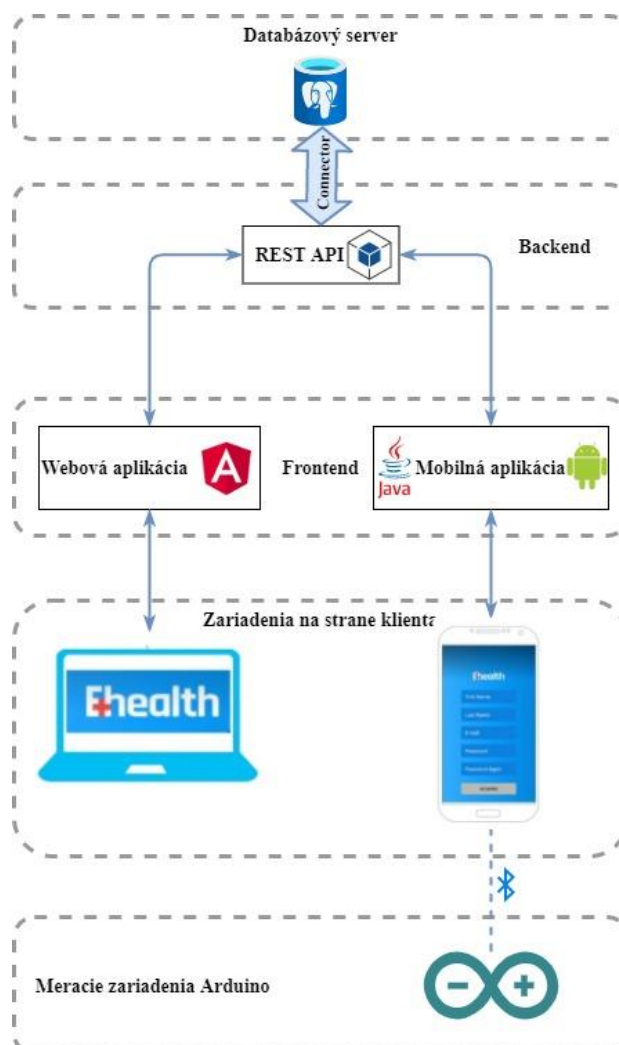
Výsledkom našej práce na projekte je systém umožňujúci realizáciu vzdialeného monitorovania pacienta. Systém pomocou používateľských aplikácií umožňuje lekárom a pacientom realizovať všetky aspekty vzdialeného monitorovania zdravotného stavu pacienta.

Jadro systému tvoria Arduino meracie prístroje, používateľské aplikácie a 3 servery:

1. Apache webový server pre používateľskú Angular aplikáciu
2. Apache webový server pre komunikáciu aplikácií s databázovým serverom pomocou REST API
3. Databázový server slúžiaci ako úložisko dát

Architektúra systému

Architektúra systému je rozdelená do piatich kategórií podľa funkcionality, akú v našom systéme daná kategória vykonáva. Na obrázku nižšie môžeme vidieť diagram tejto architektúry.



Meracie zariadenia Arduino

Tieto zariadenia slúžia na realizáciu merania monitorovaného pacienta. Pre tento účel sme si zvolili zariadenia Arduino, a to z viacerých dôvodov.

Tým prvým je, že poskytujú možnosť automatického odoslania meraných údajov rozhraním Bluetooth, vďaka čomu sa zvýši komfort pri vykonávaní samotného merania, pretože pacient nemusí manuálne zadávať údaje pri odosielaní, a tiež sa eliminuje možnosť, že pacient manuálne zapíše nesprávne hodnoty. Ďalším dôvodom je široká škála možných modulov, ktoré sa dajú k základnej doske pripojiť. Miesto viacerých prístrojov merajúcich špecifické životné funkcie tak získame jeden prístroj, ktorý dokáže zmerať takmer všetky.

Zariadenia na strane klienta

Do tejto kategórie spadajú počítače, laptopy a zariadenia s operačným systémom Android, pomocou ktorých môžu používatelia systému prostredníctvom webovej alebo mobilnej aplikácie interagovať s naším systémom.

Front End

Túto kategóriu reprezentujú dve entity. Tou prvou je webová aplikácia vyvinutá na platforme Angular, ktorej hlavnou úlohou je poskytovanie informácií lekárovi a umožnenie vykonania úloh spojených s monitorovaním pacienta. Pacient v tejto aplikácii môže kontrolovať stav a merané hodnoty svojich monitorovaní a posilať žiadosti lekárom, ktorých chce do svojich monitorovaní pridať.

Výhodami webovej aplikácie je multiplatformovosť a eliminácia problémov spojených s inštaláciou. Vďaka tomu sa výrazne zvyšuje prístupnosť aplikácie.

Druhou entitou je mobilná aplikácia vytvorená v jazyku Java. Táto aplikácia slúži na vykonanie merania a odoslania údajov pri diaľkovom monitorovaní. Lekár môže pomocou mobilnej aplikácie inicializovať konkrétne monitorovanie pacienta a pacient si môže v tejto aplikácii kontrolovať výsledky daného monitorovania. Jazyk java sme si zvolili pre jeho podporu na operačných systémoch Android, ktoré tvoria väčšinu zariadení na súčasnom trhu.

Backend

Kategóriu backend tvorí len REST API, ktorého úlohou je vybavovať žiadosti aplikácií týkajúcich sa dát uložených na databázových serveroch. Výhodou REST API je minimalizácia rizík spojených s nesprávnou manipuláciou s dátami, kontrola prístupu jednotlivých používateľov k dátam a vyššie zabezpečenie dát. V našej implementácii sme sa

rozhodli pre .NET API implementované v jazyku C#, pretože tento typ API poskytuje pokročilé možnosti manažmentu oproti jednoduchším frameworkom, akým je napríklad Flask.

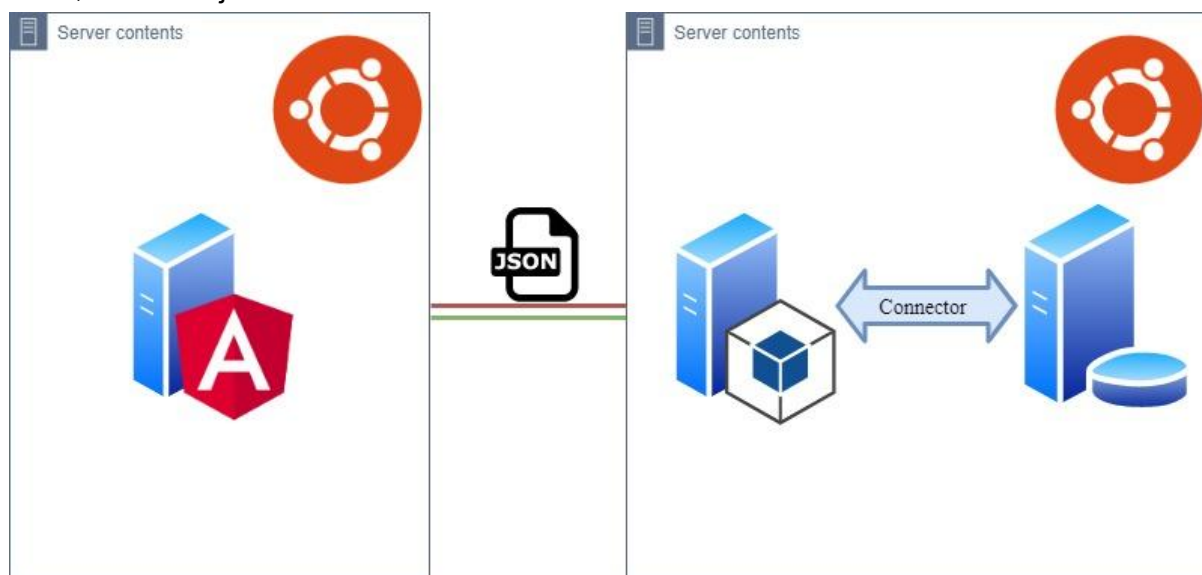
Databázový server

Túto kategóriu tvorí PostgreSQL server s relačnou databázou, ktorá zastrešuje aktérov systému a ich spoločnú interakciu. Vymedzuje vzťahy a prepojenia medzi používateľmi a uchováva základné informácie o jednotlivých používateľoch.

Okrem dát týkajúcich sa priamo používateľov nášho systému sú v tejto databáze ukladané aj hodnoty jednotlivých monitorovaní.

Architektúra serverovej časti

Serverovú časť nášho systému reprezentujú dva virtuálne stroje s operačným systémom Linux, konkrétnejšie distribúcia Ubuntu 18.04 LTS.



Prvý server slúži ako hosťujúci server webovej aplikácie. Softvérovú časť tohto servera reprezentuje Apache webový server, na ktorom je uložená Angular aplikácia.

Druhý server obsahuje taktiež Apache webový server s nasadenou .NET REST API a databázový PostgreSQL server.

Podstatou rozdelenia týchto častí na 2 rôzne stroje je redukcia konfiguračných a bezpečnostných problémov. Vzhľadom na to, že oba webové servery používajú protokol HTTP a pri nasadení bolo nutné zaistiť špecifické potreby (kestrel web služba), bolo najjednoduchším riešením tieto dve aplikácie rozdeliť.

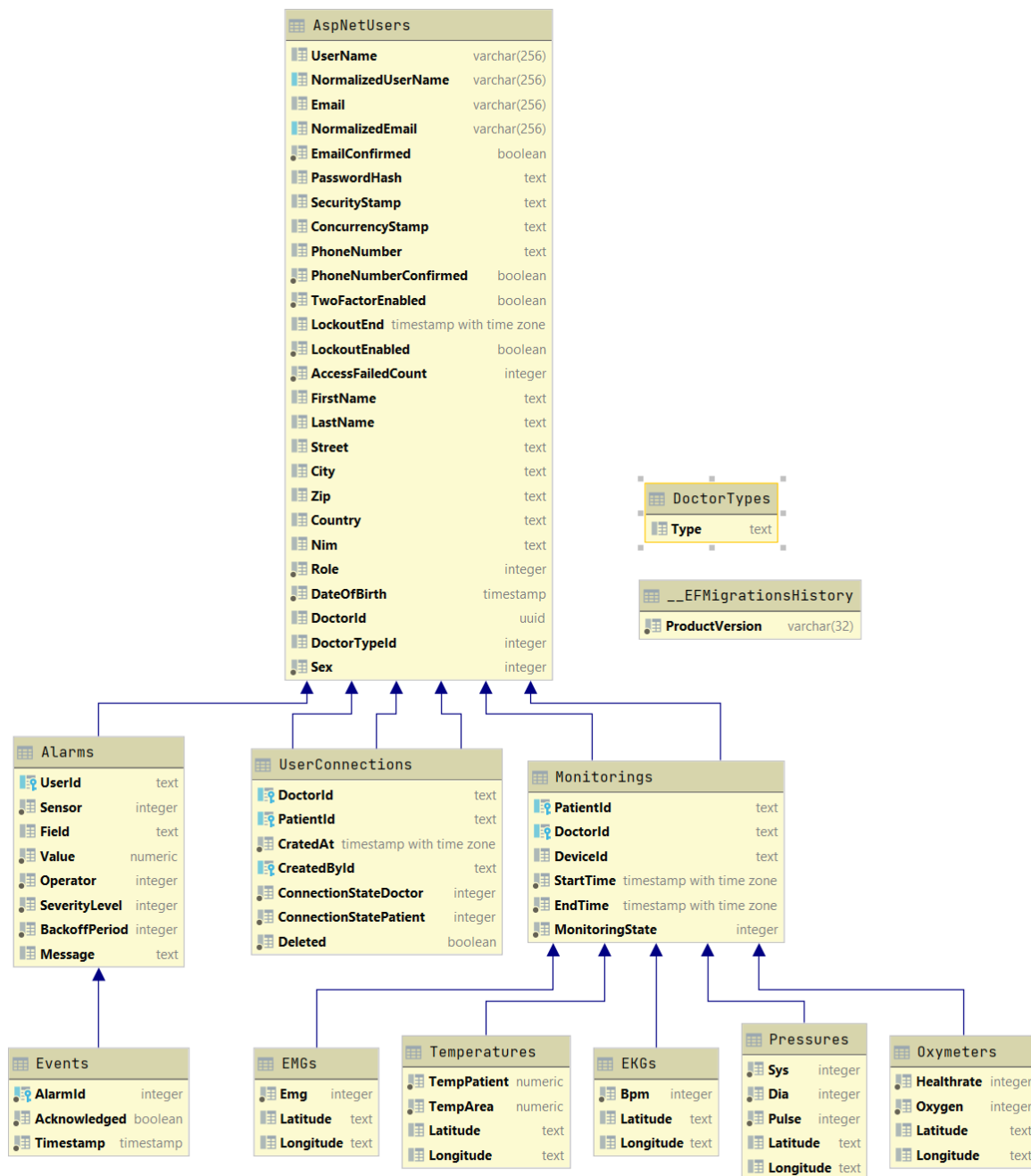
Ďalšou výhodou rozdelenia je zvýšenie bezpečnosti. Aplikácia sa nenachádza na rovnakom stroji ako databázový server, a tento server povoľuje len lokálne pripojenia, čiže sme vďaka tomuto rozdeleniu redukovali riziko kompromitácie databázového servera v dôsledku prípadných zraniteľností aplikácie.

Celková funkcionálnosť

System zabezpečuje používateľom 2 druhy aplikácií, pomocou ktorých môže vykonávať nasledujúce funkcionality:

- Zasielanie nameraných životných funkcií
- Kontrolu nameraných hodnôt lekárom
- Manažment samotných Monitorovaní
- Manažment používateľského účtu
- Vytváranie prepojení medzi pacientmi a lekármi

Dátový model



Powered by yFiles

AspNetUsers

Táto tabuľka je najhlavnejšou tabuľkou figurujúcou v našom systéme, pretože slúži na uchovávanie údajov o používateľoch nášho systému. V tejto tabuľke sú uložené osobné a kontaktné informácie ako o pacientoch, tak aj o lekároch.

Alarms

Táto tabuľka uchováva upozornenia pre lekárov v prípade, že namerané hodnoty pacienta boli v kritické.

Events

Tabuľka rozširuje tabuľku Alarms o status a dátum, kedy daný lekár

UserConnections

Tabuľka určuje prepojenia medzi pacientami a lekármi. Pred samotným začiatkom monitorovania je nutné vytvoriť spojenie s lekárom. Tu môže používateľ vybrať pacienta alebo lekára (záleží od roly používateľa), s ktorým chce v systéme vytvoriť spojenie. Po odoslaní žiadosti sa vytvorí záznam v tejto tabuľke a po prijatí je vytvorené prepojenie medzi lekárom a pacientom. Pri inicializácii nového monitorovania je nutnou podmienkou existujúci záznam v tejto tabuľke.

Monitorings

Táto tabuľka uchováva informácie o monitorovaniach. Sú v nej uložené informácie o aktéroch monitorovania, dátumoch začiatku a konca ako aj stavu monitorovania. Tabuľka je doplnená o konkrétne typy hodnôt, ktoré sú pri monitorovaní pozorované:

EMGs

Tabuľka uchováva údaje nameraných hodnôt EMG príslušného monitorovania.

Pressures

Tabuľka uchováva údaje nameraných hodnôt tlaku príslušného monitorovania.

EKGs

Tabuľka uchováva údaje nameraných hodnôt EKG príslušného monitorovania.

Oxymeters

Tabuľka uchováva údaje nameraných hodnôt okysličenia krvi príslušného monitorovania.

Temperatures

Tabuľka uchováva údaje nameraných hodnôt teplôt tela a okolitého prostredia príslušného monitorovania.

DoctorTypes

Tabuľka uchováva údaje o medicínskych zameraniach lekárov. Tabuľka je využívaná pri registrácii lekára a slúži na zachovanie konzistencie dát.

`__EFMigrationsHistory`

Táto tabuľka je potrebná pri prípadných migráciách, ktoré vykonáva .NET REST API.

Moduly systému

System sme rozdelili do nasledujúcich častí/modulov:

- Login používateľa
- Spájanie používateľov
- Monitoring pacienta
- Profil používateľa
- Vyhľadávanie používateľov
- Alarmy

Testovanie príbehov a úloh prebiehalo jednak individuálne, teda splnenie každej z nich skontroloval ešte jeden člen tímu okrem toho, ktorý ju robil. Následne boli príbehy kontrolované pri stretnutiach, kde sa vždy prezentovala daná funkčnosť. Na záver bola webová aplikácia odovzdaná na testovanie tretej strane, protokol je súčasťou príloh k inžinierskemu dielu.

Login

Modul Login pokrýva prihlásenie pacientov a lekárov do E-health systému, konkrétnejšie do webovej aplikácie celkového systému. Ďalej zabezpečuje registráciu nových účtov pre pacientov alebo lekárov a taktiež ich odhlásenie. Registrovať sa môže iba overený lekár, či samotní pacienti.

Analýza

Počas tejto fázy sme si podrobne spísali požiadavky na registráciu, prihlásenie a odhlásenie. Týmto máme na mysli to, že je potrebné pri registrácii lekára overiť, či sa jedná o pravého lekára. V tomto bode sme sa skontaktovali s doktorkou, kde sme diskutovali ako by sa tento fakt dal overiť. Riešenie spočíva v tom, že ak sa registruje lekár, v registračnom formulári je možnosť zaškrtnúť ‚Zaregistrovať sa ako lekár‘. Po zakliknutí sa zobrazí textové okno, kde lekár zadá svoje unikátne ID číslo, ktoré používa aj v komore lekárov (<https://lekom.sk/slovenska-lekarska-komora/organy-slk/register-lekarov>).

Registrácia nového účtu pacienta by mala spočívať klasicky. Pacient zadá emailovú adresu, heslo a vytvorí sa mu účet. Pacient, tak ako aj lekár, musí pre úspešné vytvorenie nového účtu súhlasiť s podmienkami používania E-health systému.

Prihlásenie do webovej, či mobilnej aplikácie je štandardné, a preto nebola nutná hlbšia analýza.

Návrh

Nakreslili mockup registrácie. Taktiež sme si vytvorili biznis procesný model a diagram aktivít pre registráciu.

Registrácia pacienta, či lekára je možná prostredníctvom registračného formulára, ktorého grafické prostredie poskytne *RegistrationComponent*. Pacient alebo lekár zadá email, heslo a následne heslo aj potvrdí. Po vyplnení registračného formuláru, ak sa heslá zhodujú, zadané údaje sa pošlú na stranu servera, kde sa overí dostupnosť emailovej adresy, a v prípade, že sa jedná o lekára overí sa platnosť ID lekára v komore lekárov SR

Priebeh registrácie

Registrácia - Používateľ je na stránke prihlasovania. Kliká na tlačidlo *zaregistrujte sa*. Ocitá sa na stránke registrácie kde vypĺňa svoje údaje.

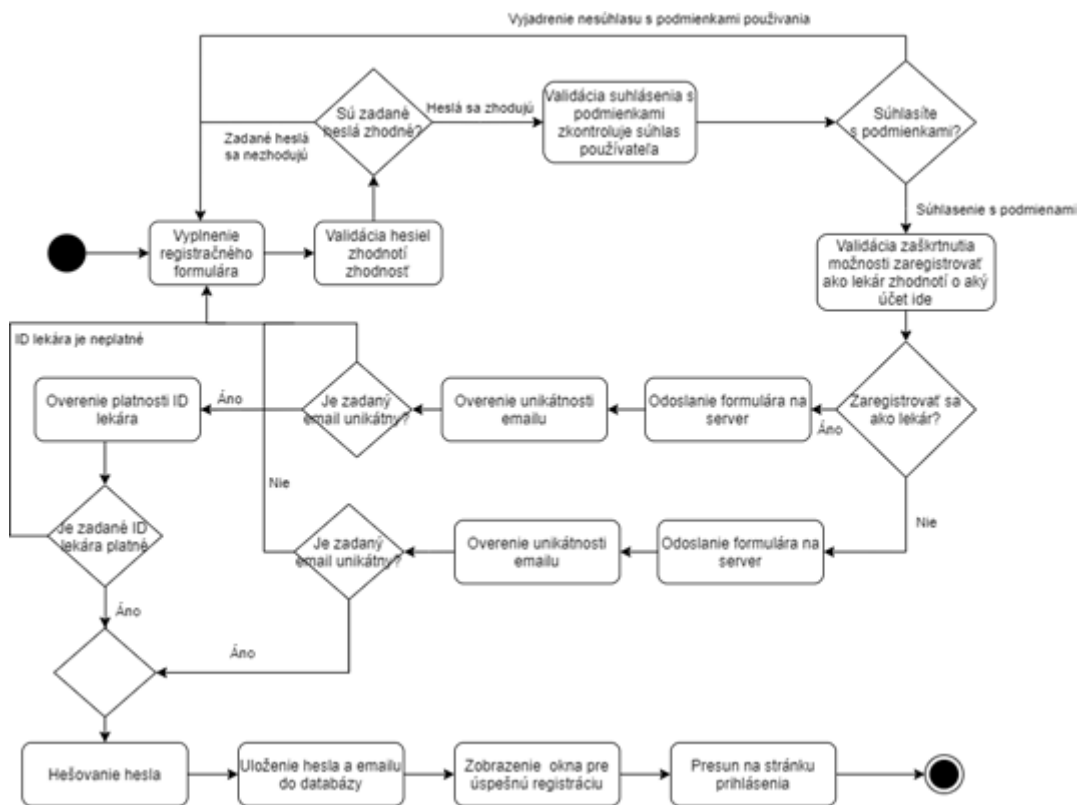
- Heslá musia byť rovnaké
- Ak je používateľ lekár, zaklikáva checkbox *Zaregistrovať sa ako lekár*. Odkrýva sa mu textové pole, kde zadáva svoje ID v komore lekárov SR. Prebehne kontrola správnosti tohto ID.
- Používateľ musí odkliknúť súhlas s podmienkami
- Po kliknutí na tlačidlo *Zaregistrovať sa* prebehne kontrola toho, či už používateľ s danou emailovou adresou existuje. Ak nie, používateľ bude vytvorený v systéme a presmerovaný na stránku *Prihlásenia*. V opačnom prípade dostane chybovú hlášku.

Registrácia

Zobrazenie podmienok používania



Use case diagram registrácie



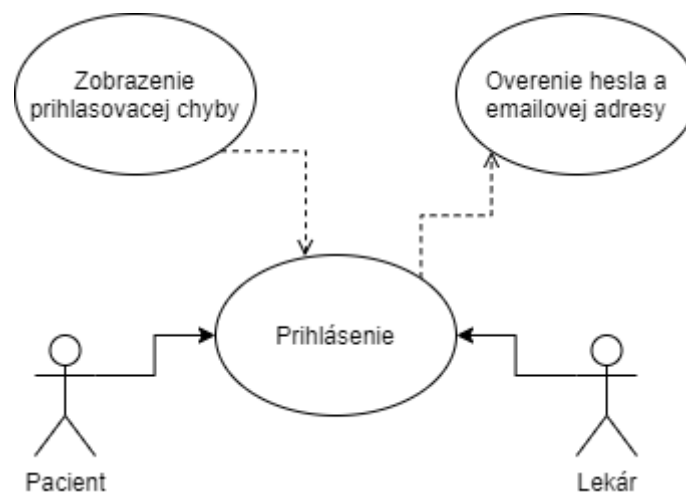
Activity diagram registrácie

Po zaslaní registračného formuláru na server je na strane servera identifikovaný zadaný email a heslo, poprípade aj ID lekára. Server skontroluje unikátnosť emailu, a keď sa jedná o lekára, taktiež sa skontroluje platnosť zadaného ID lekára. Ak je všetko korektné heslo sa zahešuje a spolu s emailom sa tieto údaje zapíšu do databázy a registrácia nového účtu prebehla úspešne.

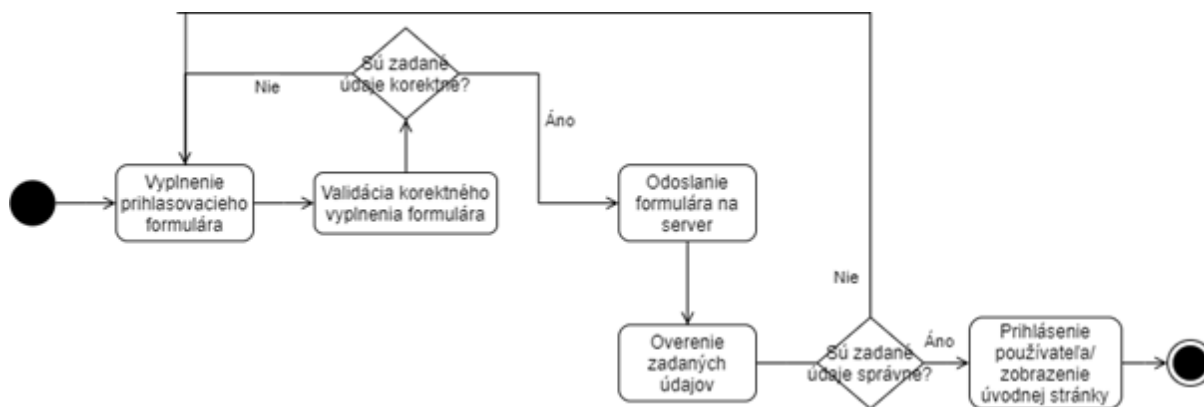
Obdobne sme si opísali aj prihlásenie používateľa. Taktiež sme si priblížili tento prípad použitia pomocou diagramu aktivít a biznis procesným diagramom.



Mockup obrazovky prihlásenia



Use case diagram prihlásenia



Activity diagram prihlásenia

Modul Login má taktiež funkciu odhlásenia používateľa. Pacient alebo lekár sa môže odhlásiť v hlavičke webovej aplikácie a to stlačením ikony (šipka) odhlásiť sa. Odhlásenie je pomerne jednoduchý prípad použitia a preto nebola nutná analýza funkcionality odhlásenia

Náčrty daných prípadov použitia, ktoré slúžili na prvotný návrh:

E-Health

Registrácia

Zaregistrovať sa ako lekár

Súhlasím s [podmienkami používania](#)

Máte už účet? [Prihlásiť sa](#)

Náčrtok registrácie

E-Health

Registrácia

Krstné meno

Podmienky

Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore
et dolore magna aliqua.

Súhlasím

Registrovat' sa

Máte už účet? [Prihlásiť sa](#)

Náčrtok podmienok používania

E-Health

Vitajte

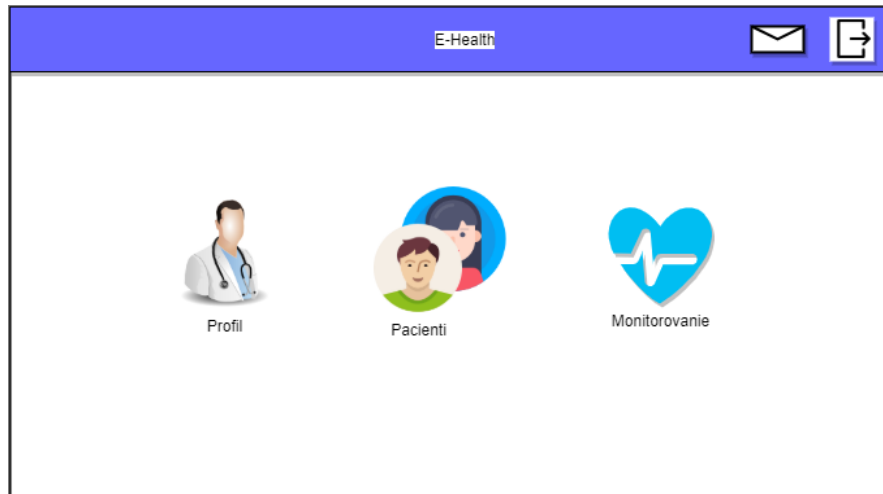
Email

Heslo 

Prihlásiť sa

Nemáte účet? [Registrujte sa](#)

Náčrtok prihlásenia sa do webovej aplikácie



Náčrtok hlavného menu/ hore v pravom rohu možnosť odhlásenia sa

Implementácia

Frontend modulu loginu tvorí viacero súborov:

registration.component.ts – obsahuje aplikačnú logiku registrácie

registration.service.ts – tento súbor obsahuje komunikáciu so serverom, a čaká sa na validáciu zadaných údajov

registration.component.html – tento HTML súbor obsahuje template pre grafické zobrazenie registrácie

registration.component.css – súbor obsahuje ako registračný formulár vyzerá

Obdobne to vyzerá aj pri prihlásení používateľa. Konvečne sme sa snažili dodržiavať princípy aby bol kód intuitívny.

Ukážky implementovaných prípadov použitia:

Vitajte

Email

Heslo

PRIHĽÁSIŤ SA

Nemáte ešte účet? [Registruovať sa](#)

Implementované prihlásenie sa do webovej aplikácie

health

Vaše telefónne číslo...

Rodné číslo

Pohlavie: Muž Žena Oštre

Lekár vypĺňa miesto ordinácie a pacient trvale bydlisko!

Ulica číslom

Mesto

PSČ

Krajina

Heslo

Zopakovať heslo

Zaregistruovať sa ako lekár

Súhlasím s podmienkami používania

REGISTROVAŤ SA

Máte už účet? [PRIHĽÁŤ SA](#)

Implementácia registrácie vo webovej aplikácii

Podmienky používania

Cras mattis consectetur purus sit amet fermentum. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

Prasent commodo cursus magna, vel scelerisque nisl consectetur et. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Aenean lacina bibendum nulla sed consectetur. Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Donec sed odio dui. Donec ullamcorper nulla non metus auctor fringilla.

Cras mattis consectetur purus sit amet fermentum. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

Prasent commodo cursus magna, vel scelerisque nisl consectetur et. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Aenean lacina bibendum nulla sed consectetur. Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Donec sed odio dui. Donec ullamcorper nulla non metus auctor fringilla.

Cras mattis consectetur purus sit amet fermentum. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

Prasent commodo cursus magna, vel scelerisque nisl consectetur et. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Aenean lacina bibendum nulla sed consectetur. Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Donec sed odio dui. Donec ullamcorper nulla non metus auctor fringilla.

Cras mattis consectetur purus sit amet fermentum. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

Prasent commodo cursus magna, vel scelerisque nisl consectetur et. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Aenean lacina bibendum nulla sed consectetur. Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Donec sed odio dui. Donec ullamcorper nulla non metus auctor fringilla.

Rozumiem

© 2013 Tm, LLC. All rights reserved.

Implementácia okna podmienky používania vowebovej aplikácii



Implementácia endpointov spojených k modulu Login

Registrácia pacienta

HTTP POST metóda na url v tvare: **`{{url}}/api/user/registerPatient`**

Poz.: V aplikáciách (pre produkčné prostredie) treba **`{{url}}`** nahradiť ip adresou, alebo doménovým menom napr.: <http://team18-20.studenti.fiit.stuba.sk> (Celá url teda vyzerá [:http://team18-20.studenti.fiit.stuba.sk/api/user/registerPatient](http://team18-20.studenti.fiit.stuba.sk/api/user/registerPatient))

V headroch, treba mať nastavene:

Content-Type : application/json

Tento request je dostupný všetkým používateľom aplikácie.

V tele správy zadávam údaje:

```
1 {
2   "email": "anna@test.com",
3   "password": "Pa$$w0rd",
4   "firstName": "Anna",
5   "lastName": "Pacientovska",
6   "street": "Zelezniciarov",
7   "city": "Trstena",
8   "zip": "69420",
9   "country": "Uganda",
10  "dateOfBirth": "1993-12-24",
11  "nim": "931224/6969",
12  "sex": "Female",
13  "phoneNumber": "+421420420420"
14 }
```

dateOfBirth - dátum narodenia v ISO formáte t.j.: YYYY-MM-DD

nim - rodné číslo

sex - pohlavie používateľa, možné hodnoty: **Female, Male ,Undefined**

ostatné parametre - self explanatory

Odpoveď :

```
1 {
2   "id": "983e41c7-d729-4dae-89d1-3ed47f555dd8",
3   "firstName": "Anna",
4   "lastName": "Pacientovska",
5   "dateOfBirth": "1993-12-24T00:00:00",
6   "nim": "931224/6969",
7   "sex": "Female",
8   "phoneNumber": "+421420420420",
9   "email": "anna@test.com",
10  "street": "Zelezniciarov",
11  "city": "Trstena",
12  "zip": "69420",
13  "country": "Uganda"
14 }
```

Príklad neuspesneho request, kvoli zlemu nim:

Request:

```
1 {
2   "email": "anna@test.com",
3   "password": "Pa$$w0rd",
4   "firstName": "Anna",
5   "lastName": "Pacientovska",
6   "street": "Zelezniciarov",
7   "city": "Trstena",
8   "zip": "69420",
9   "country": "Uganda",
10  "dateOfBirth": "1993-12-24",
11  "nim": "xxxx/6969",
12  "sex": "Female",
13  "phoneNumber": "+421420420420"
14 }
```

Registrácia doktora

HTTP POST metóda na url v tvare: `{{url}}/api/user/registerDoctor`

Poz.: V aplikáciách (pre produkčné prostredie) treba {{url}} nahradiť ip adresou, alebo doménovým menom napr.: `http://team18-20.studenti.fiit.stuba.sk` (Celá url teda vyzerá :`http://team18-20.studenti.fiit.stuba.sk/api/user/registerDoctor`)

V headroch, treba mat nastavene:

Content-Type : `application/json`

Tento request je dostupný všetkým používateľom (aj neprihláseným) .

V tele správy zadávam údaje:

```
1 {
2   "email": "zuzanaLekar@test.com",
3   "password": "Pa$$w0rd",
4   "firstName": "Zuzana",
5   "lastName": "Urologicka",
6   "street": "Janotikova",
7   "city": "Buzerec",
8   "zip": "69420",
9   "country": "Buzerec",
10  "dateOfBirth": "1956-12-12",
11  "nim": "931224/6969",
12  "sex": "Female",
13  "phoneNumber": "+421420420420",
14  "doctorId": "fa6bd776-7a35-452c-8cfa-ee26e0308026",
15  "doctorTypeId": 3
16 }
```

dateOfBirth - dátum narodenia v ISO formáte t.j.: YYYY-MM-DD

sex - pohlavie používateľa, možné hodnoty: Female, Male ,Undefined

doctorId - unikátne číslo doktora → v tejto verzii stačí zadať platný Guid

doctorTypeId - id špecializácie doktora (dá sa zistiť pomocou endpointu popísaného v :

CodeList - get doctor types

ostatné parametre - self explanatory

Odpoveď :

```
1 {
2   "id": "50e7013d-f996-4d49-b8d3-bcb280b38c56",
3   "firstName": "Zuzana",
4   "lastName": "Urologicka",
5   "dateOfBirth": "1956-12-12T00:00:00",
6   "nim": "931224/6969",
7   "sex": "Female",
8   "doctorTypeId": 3,
9   "phoneNumber": "+421420420420",
10  "email": "zuzanaLekar@test.com",
11  "street": "Janotikova",
12  "city": "Buzerec",
13  "zip": "69420",
14  "country": "Buzerec"
15 }
```

Prihlásenie:

HTTP POST metóda na url v tvare: {{url}}/api/user/login

V headroch, treba mat nastavene:

Content-Type : application/json

V tele správy:

```
1 {
2   "email": "lekarZNiznej@test.com",
3   "password": "Pa$$w0rd"
4 }
```

Odpoveď ak je login uspesny:

```
1 {
2   "token": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1IjoiMTc0MTA1MS1iOTc5LTRkNC"
3 }
4
```

```
1 {
2   "email": "lekarZNiznej@test.com",
3   "password": "Pa$$w0rd"
4 }
```

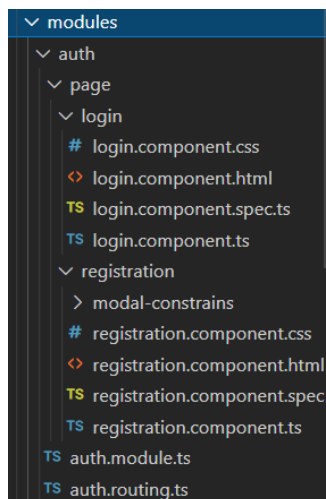
Ak je login neuspesny z dovodu zleho mena alebo hesla vrati sa http code :401 Unauthorized

Príklad neuspesneho request, kvoli prazdnym parametrom v Jsone:

```
1 {
2   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3   "title": "One or more validation errors occurred.",
4   "status": 400,
5   "traceId": "|809b5324-4ed278ac445db56f.",
6   "errors": {
7     "Email": [
8       "Pole 'Email' nesmie byť prázdne."
9     ],
10    "Password": [
11      "Pole 'Password' nesmie byť prázdne."
12    ]
13  }
14 }
```

Hlavná štruktúra modulu login:

Na obrázku nižšie máme možnosť vidieť štruktúru login modulu. Pozostáva štandardne ako aj iné angular projekty. Každý modul má service, ktorý komunikuje so serverom, a teda modul login používa konkrétne, **auth.service.ts**, ktorý umožňuje sa registrovať a prihlasovať do webovej aplikácie.



Ukážka auth.service.ts metód:

```
registerDoctor(formInput) {
  const body: RegisterDoctor = {
    email: formInput.email,
    password: formInput.password,
    firstName: formInput.firstName,
    lastName: formInput.lastName,
    street: formInput.street,
    city: formInput.city,
    zip: formInput.zip,
    country: formInput.country,
    nim: formInput.nim,
    dateOfBirth: formInput.dateOfBirth,
    sex: formInput.sex,
    phoneNumber: formInput.phoneNumber,
    DoctorId: formInput.doctorId,
    DoctorTypeId: formInput.doctorTypeId,
    role: "Doctor"
  }
  var reqHeader = new HttpHeaders({'Content-Type': 'application/json'});
  return this.http.post(environment.apiUrl + environment.apiEndpointRegisterDoctor, body, this.options);
}
```

```
login(formInput) {
  var email = formInput.account.email;
  var password = formInput.account.password;

  return this.http.post<User>(environment.apiUrl + environment.apiEndpointLogin, { email, password }, this.options)
    .pipe(map(user => {
      // store user details and jwt token in local storage to keep user logged in between page refreshes

      user.role = jwt_decode(user.token)['role'];

      localStorage.setItem('currentUser', JSON.stringify(user));
      this.userSubject.next(user);
      return user;
    }));
}
```


Spájanie používateľov

Tento modul rieši problematiku prepojenia lekárov a pacientov.

Analýza

Po diskusiách s product ownerom sme zistili, že v našom systéme budú vystupovať dva typy používateľov - pacient a lekár. Objektom monitorovania budú pacienti. Pacienti budú mať takisto v systéme svoj profil a históriu svojich monitorovaní. Lekári budú mať takisto v systéme svoj profil. Nato, aby tieto dáta boli viditeľné a interagovateľné aj pre lekára, musí byť medzi lekárom a daným pacientom vytvorené spojenie. Toto spojenie môže iniciovať spájaný pacient, spájaný lekár. Po iniciovaní spojenia ešte musí byť spojenie prijaté oboma spájanými používateľmi, inak nie je platné. Spojenie takisto možno odmietnuť alebo vymazať.

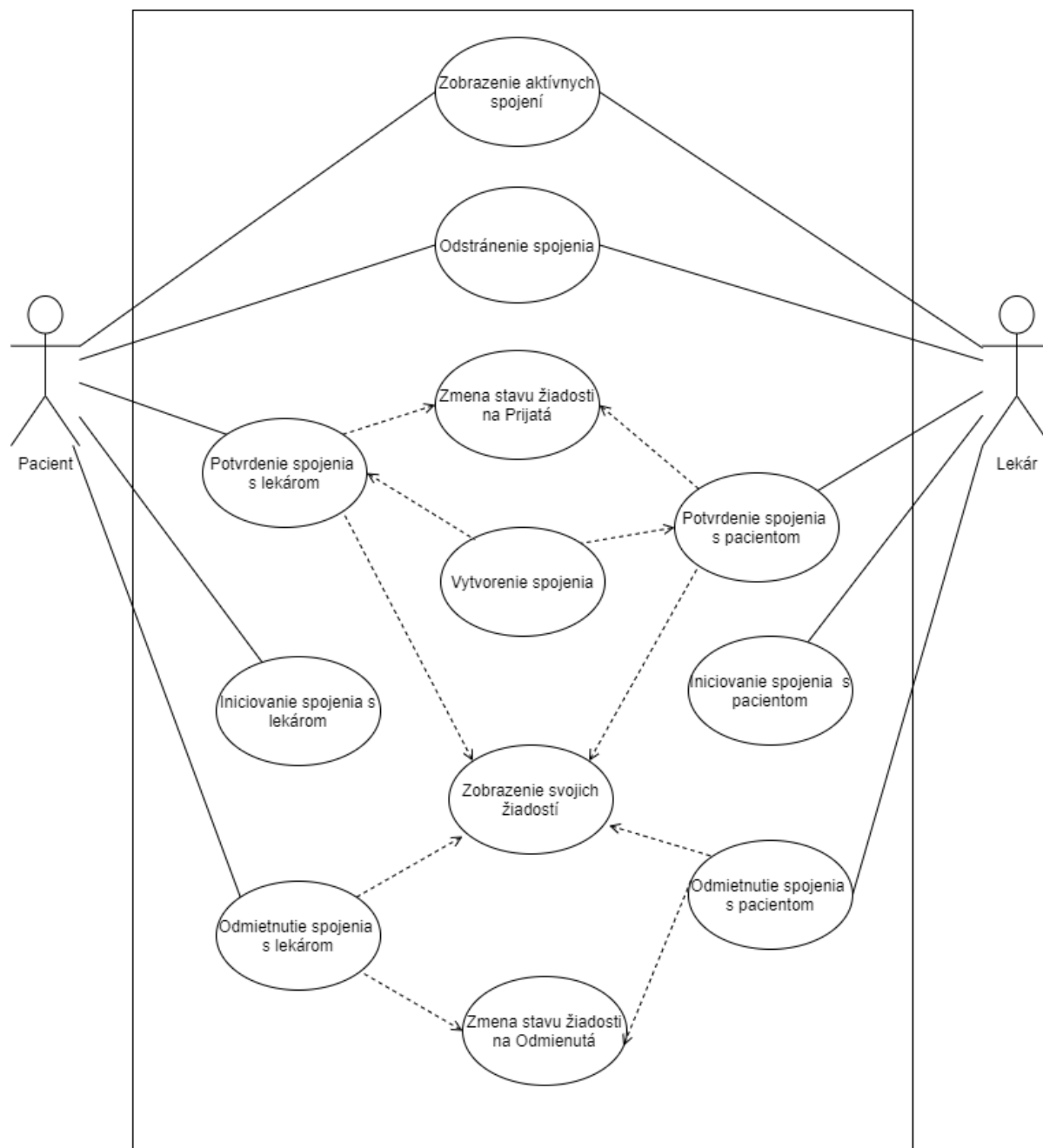
Návrh

Prípady použitia

Všetky identifikované prípady použitia (viď obr. nižšie) budú prebiehať v rámci webovej aplikácie. Nasleduje opis konkrétnych prípadov použitia:

- **Zobrazenie svojich žiadostí** - Používateľ si chce zobrazit' svoje žiadosti. Kliká na sekciu Moje žiadosti.
- **Zaslanie žiadosti lekárovi** - Pacient je prihlásený. Kliká na sekciu Lekári, kde rozklikáva profil lekára, s ktorým chce byť spojený. Na jeho nájdenie môže použiť filtre. Následne v profile kliká v profile na tlačidlo Pridať lekára.
- **Potvrdenie žiadosti od lekára** - Pacient kliká na sekciu Moje žiadosti. Nachádza žiadosť od daného lekára a kliká na tlačidlo Prijat'
 - Stav danej žiadosti sa mení na Prijatá a spojenie je vytvorené
- **Odmietnutie žiadosti od lekára** - Pacient kliká na sekciu Moje žiadosti. Nachádza žiadosť od daného lekára a kliká na tlačidlo Odmietnuť.
 - Stav danej žiadosti sa mení na Odmietnutá, žiadosť zo zoznamu mizne a spojenie sa nevytvára.
- **Zaslanie žiadosti pacientovi** - Lekár je prihlásený. Kliká na sekciu Pacienti, kde rozklikáva profil pacienta, s ktorým chce byť spojený. Na jeho nájdenie môže použiť filtre. Následne v profile kliká v profile na tlačidlo Pridať pacienta.

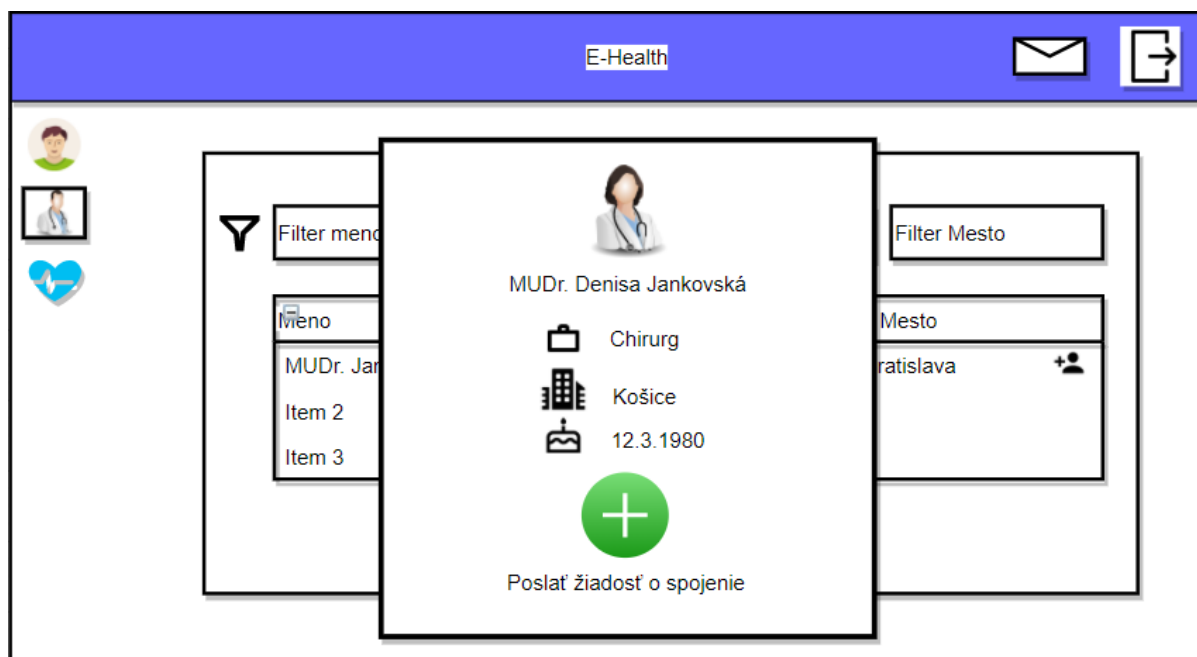
- **Potvrdenie žiadosti od pacienta** - Lekár kliká na sekciu Moje žiadosti. Nachádza žiadosť od daného pacienta a kliká na tlačidlo Prijat'
 - Stav danej žiadosti sa mení na Prijatá a spojenie je vytvorené
- **Odmietnutie žiadosti od pacienta** - Pacient kliká na sekciu Moje žiadosti. Nachádza žiadosť od daného pacienta a kliká na tlačidlo Odmietnuť.
 - Stav danej žiadosti sa mení na Odmietnutá, žiadosť zo zoznamu mizne a spojenie sa nevytvára.
- **Odstránenie vlastnej žiadosti** - Používateľ rozklikne svoje žiadosti a kliká na tlačidlo "X". Žiadosť zmizne.
- **Odstránenie spojenia** - Používateľ rozklikáva sekciu Lekári/Pacienti. Zaklikáva si filter "Zobraziť len lekárov/pacientov s ktorými som spojený" a nachádza požadovaného používateľa. Rozklikáva jeho profil a kliká na tlačidlo zrušiť spojenie.
- **Zobrazenie aktívnych spojení** - Používateľ rozklikáva sekciu Lekári/Pacienti. Zaklikáva si filter "Zobraziť len lekárov/pacientov s ktorými som spojený".



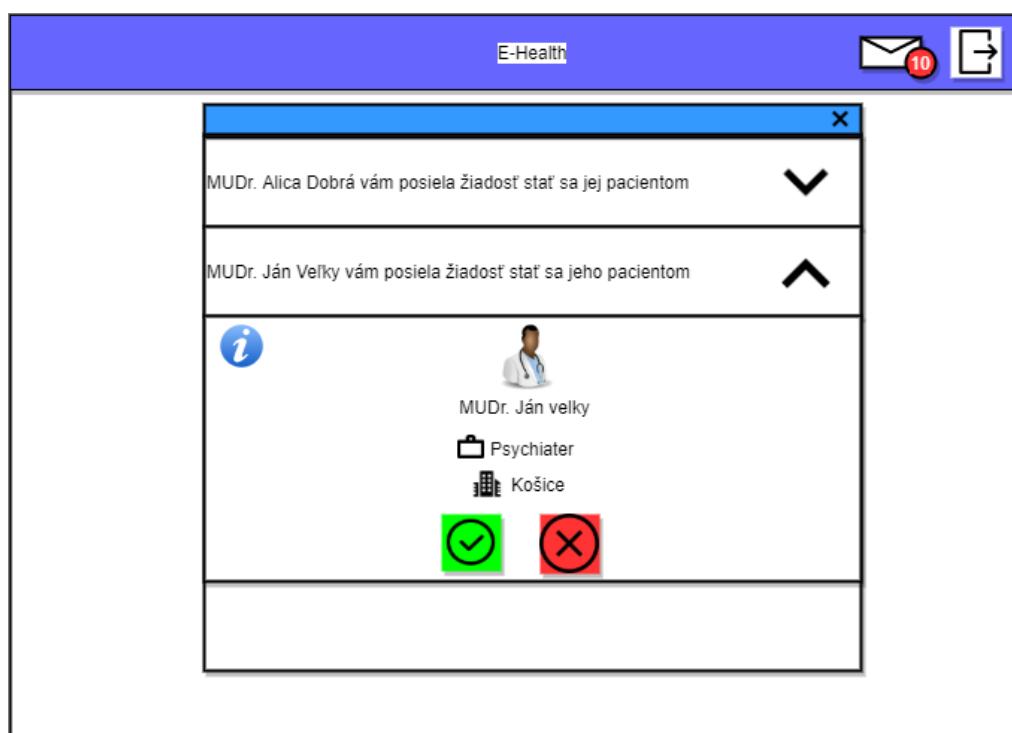
Náčrty okien modulu

Súčasťou návrhu boli aj návrhy okien tohoto modulu. Tieto náčrty (a popisy k nim) sa do istej miery využili, avšak pri tvorbe reálneho dizajnu sme sa odklonili jednak od vzhľadu a čiastočne aj funkcionality.

Okno na zaslanie žiadosti o spojenie



Okno zoznamu žiadostí



Implementácia

Frontend

Okno Zoznam žiadostí

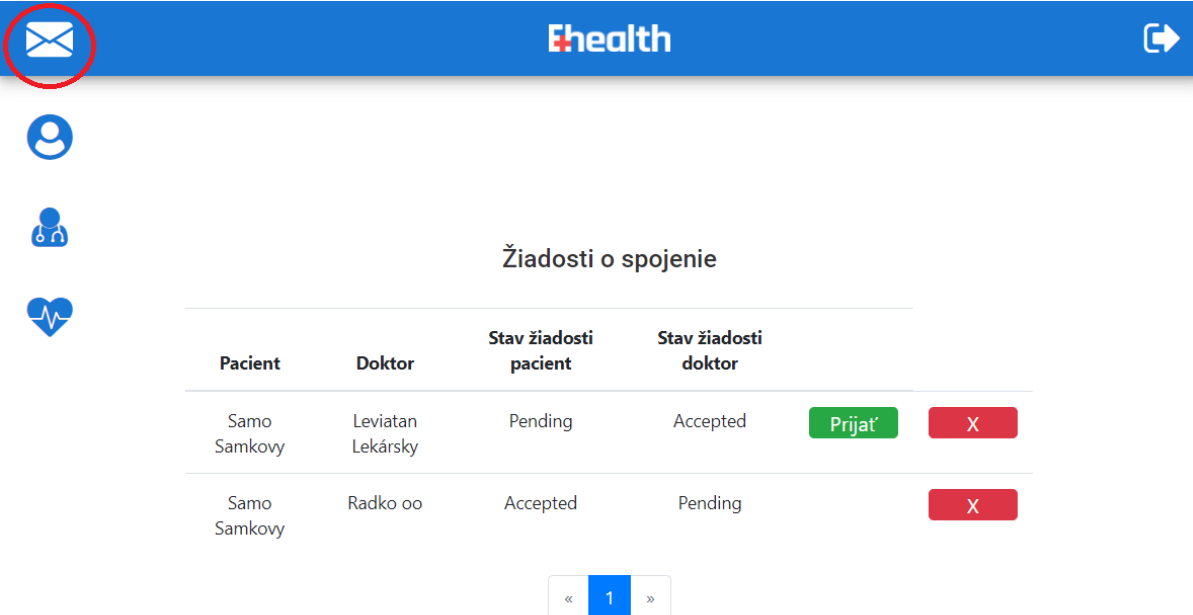
connection-requests.component.ts – obsahuje aplikačnú logiku Zoznamu žiadostí

connection-requests.service.ts – tento súbor obsahuje komunikáciu so serverom, a čaká sa na validáciu zadaných údajov

connection-requests.component.html – tento HTML súbor obsahuje template pre grafické zobrazenie Zoznamu žiadostí

connection-requests.component.css – súbor obsahuje štýly pre Zoznam žiadostí

Na obrázku nižšie môžeme vidieť okno aplikácie pre Zoznam žiadostí. V ľavom hornom rohu je v červenom krúžku ikona obálky, na ktorú je potrebné kliknúť v prípade potreby dostať sa na zoznam žiadostí.



The screenshot shows the Fhealth application interface. At the top is a blue header with the 'Fhealth' logo and a right-pointing arrow. On the left side, there is a vertical navigation menu with three icons: an envelope icon (circled in red), a person icon, and a person with a plus sign icon. Below the header, the main content area is titled 'Žiadosti o spojenie'. It contains a table with two rows of data. The first row shows a patient named 'Samo Samkovy' and a doctor named 'Leviatan Lekársky'. The patient's request status is 'Pending' and the doctor's is 'Accepted'. There are two buttons: a green 'Prijat' button and a red 'X' button. The second row shows the same patient 'Samo Samkovy' and a doctor named 'Radko oo'. The patient's request status is 'Accepted' and the doctor's is 'Pending'. There is a red 'X' button. At the bottom of the table, there is a pagination control showing '« 1 »'.

Pacient	Doktor	Stav žiadosti pacient	Stav žiadosti doktor		
Samo Samkovy	Leviatan Lekársky	Pending	Accepted	Prijat	X
Samo Samkovy	Radko oo	Accepted	Pending		X

« 1 »

Komponenty v okne osobného profilu používateľa

V rámci profilu používateľa je možné zaslať žiadosť o spojenie, vidieť informáciu o tom, či bola žiadosť odoslaná a v akom je stave a taktiež je možné zrušiť existujúce spojenie.

Zrušenie spojenia

Osobný profil

Meno: Leviatan	Miesto pracoviska	
Priezvisko: Lekársky	Ulica: Ilieš	
Dátum narodenia: dd.mm.rrrr	Mesto: Krivá	Telefónne číslo: +421915619797
Rodné číslo: Vaše rodné číslo...	PSČ: 02755	Email: levolek@test.com
Pohlavie: <input checked="" type="radio"/> Muž <input type="radio"/> Žena <input type="radio"/> Iné	Krajina: Slovensko	
Špecializácia: Všeobecný lekár		

Zrušiť spojenie

Poslať žiadosť

Osobný profil Poslať žiadosť

Meno: Mino	Miesto pracoviska
Priezvisko: Lekar	Ulica: Druzstevna2
Dátum narodenia: dd.mm.rrrr	Mesto: Nizna2
Pohlavie: <input checked="" type="radio"/> Muž <input type="radio"/> Žena <input type="radio"/> Iné	PSČ: 69420
Špecializácia: Chirurg	Krajina: Uganda

Čaká sa na potvrdenie

Osobný profil Čaká sa na potvrdenie

<p>Meno: Ondrej</p> <p>Priezvisko: Chirurgicky</p> <p>Dátum narodenia: dd.mm.rrrr</p> <p>Pohlavie: <input checked="" type="radio"/> Muž <input type="radio"/> Žena <input type="radio"/> Iné</p> <p>Špecializácia: Chirurg</p>	<p>Miesto pracoviska</p> <p>Ulica: Janotikova</p> <p>Mesto: Bratislava</p> <p>PSČ: 69420</p> <p>Krajina: Buzerec</p>
---	---

Komponent v okne vyhľadávania

V rámci vyhľadávania je prítomný checkbox *Zobraziť pacientov/lekárov s ktorými som spojený*, ktorý odfiltruje používateľov, s ktorými používateľ spojený nie je.

List pacientov

Zobraziť len pacientov s ktorými som spojený

Meno	Priezvisko	Dátum narodenia	Mesto	
Samo	Samkovy	01.01.1	Nizna	» Detail

« 1 »

API endpointy

UserConnection - send request

Poslanie žiadosti o spojenie s používateľom.

HTTP POST metóda na url v tvare: `{{url}}/api/UserConnection/sendRequest`

Popis:

- Ak posielať žiadosť pod svojim menom (nezáleží či som doktor alebo pacient), tak je automaticky z mojej strany aj prijatá resp. UserConnectionState na mojej strane = Accepted.
 - Zároveň platí, že UserConnectionState druhej strany spojenia je = Pending.

- Ak posielať žiadosť na vytvorenie spojenia medzi iným lekárom a pacientom, tak `UserConnectionState` je na oboch stranách = Pending .

UserConnection - change state

Zmena stavu spojenia medzi lekárom a pacientom podľa id

HTTP PUT metóda na url v tvare: `{{url}}/api/profiles/personal/{{guid UserConnection}}`

Popis:

- Mením stav `UserConnection`, v ktorom figurujem ako pacient alebo lekár
- Môžem vždy meniť len stav, ktorý prislúcha mne t.j. ak som pacient, môžem meniť len svoj stav
- Ostatní usri nemôžu meniť stav `UserConnection`, v ktorom nefigurujú

UserConnection - list

Zoznam spojení v ktorých figurujem

HTTP GET metóda na url v tvare: {{url}}/api/UserConnection

Popis:

- Request slúži na získanie všetkých spojení v ktorých figurujem, pričom:
 - môžem byť pacient, ktorý je v spojení
 - môžem byť lekár, ktorý je v spojení
 - alebo môžem byť lekár, ktorý vytvoril spojenie, ale v ňom nie je

Monitoring

Modul Monitorovanie

Modul monitorovanie zastrešuje úkony potrebné na získanie zdravotných údajov pacienta. Prvotným krokom pre získanie týchto dát je potrebné nadviazať spojenie s meračom, identifikovať o aký druh merača ide a spustenie samotného merania. V procese návrhu sme pre účely získavania dát zamýšľali použiť dosku Arduino MKR 1010 WiFi. Toto zariadenie je podľa dokumentácie k produktu schopné pracovať i s technológiou Bluetooth LowEnergy. Napriek našej snahe sa nám však nepodarilo sfunkčniť komunikáciu touto technológiou. Neúspech pripisujeme chybe hardvéru. Spojenie s doskou sa nám podarilo nadviazať ale ďalšia komunikácia bola príliš nestabilná pre potreby nášho projektu. Keďže tento hardvér je primárne navrhnutý na prácu cez wifi, rozhodli sme sa zariadenie využiť ako webový server. Z mobilného zariadenia sme následne prostredníctvom GET requestov zamýšľali získavať dáta. Tu sme však narazili na ďalší problém. Spracovanie requestu zariadením trvalo viac ako 10 sekúnd a kým prišla odpoveď na mobilné zariadenie, vypršal čas pre request na zariadení a opäť sa nám nepodarilo získať potrebné dáta. Rozhodli sme sa teda pre komunikáciu využiť dosku Arduino UNO s použitím BTshield-u od spoločnosti Elecrow. Tento hardvér bol členom tímu známy, čo urýchlilo priebeh následných prác. Behom implementácie sme však museli čeliť ďalším problémom s BTshieldom. Tento shield sa dostal do chybového stavu, z ktorého sa nám nepodarilo dostať ani po preskúmaní všetkých dohľadateľných zdrojov. Problémom bola veľmi strohá dokumentácia k produktu. Keďže je tento modul kľúčovou časťou nášho projektu, zaobstarali sme po vlastnej osi štandardný modul HC-05. S týmto modulom sme už nenarazili na žiadne komplikácie a mohli sme sa naplno venovať implementácií potrebnej funkcionality. Tieto problémy nás však dostali do časovej tiesne, napriek skutočnosti, že sme s implementáciou začali v dostatočnom predstihu.

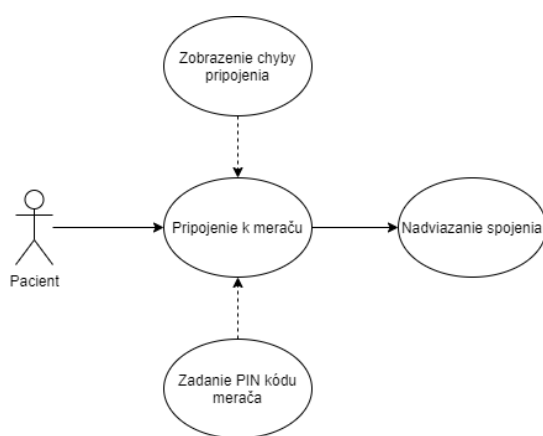
Nadviazanie spojenia

Analýza

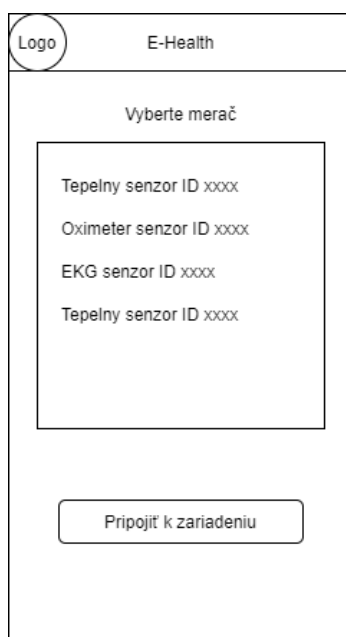
Používateľ vyberie zariadenie, ku ktorému sa chce pripojiť. Zadá PIN daného zariadenia. Ak je kód nesprávny alebo sa zariadenie nenachádza v dostatočnej blízkosti, používateľ dostane chybovú hlášku. V opačnom prípade pripojenie prebehne úspešne.

Návrh

Našu mobilnú aplikáciu sme sa rozhodli cieľiť na platformu Android. Implementácia prebiehala v jazyku Java v prostredí AndroidStudio. Keďže sme sa rozhodli pre použitie technológie Bluetooth, samotné nadviazanie spojenia prebieha štandardným spôsobom, typickým pre tento druh komunikácie. Komunikácia prebieha v režime Master/Slave, pričom mobilné zariadenie pracuje v režime Master. Keďže budeme používať viacero typov meračov, je nutné aby ihneď po nadviazaní spojenia odoslal merač (Arduino) riadiacu správu, vďaka ktorej mobilné zariadenie detekuje o aký merač ide. Na základe tejto správy budú vykonané ďalšie akcie. Pre lepšiu ilustráciu sme vytvorili náčrt okna výberu zariadenia a biznis procesný model.



Biznis procesný model



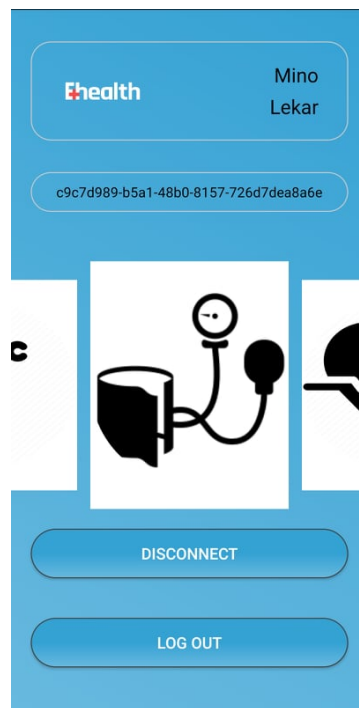
Mockup

Implementácia

Nadviazanie spojenia prebieha oproti zimného semestra viac automatizovane. Na serveri je ku konkrétnemu monitoringu priradená MAC adresa určeného merača. V prípade nášho projektu sa jednalo len o jednu adresu, no napriek tomu sme pre jednoduchú rozšíriteľnosť implementovali dynamické priradovanie adres zariadení. Po prihlásení, sa používateľovi zobrazí zoznam aktívnych monitoringov. Po kliknutí na vybraný monitoring je zahájený proces pripájania. Užívateľ zadá heslo zariadenia. Tento krok je nutné vykonať len v prípade prvotného pripojenia k meraču. Pri opakovanom pripojení je heslo zapamätané a používateľ pokračuje pramo do aktivity na výber senzoru. Po nadviazaní spojenia je zo zariadenia Arduino odoslaná správa, ktorá hovorí aký typ merača je momentálne pripojený. Používateľ môže následne pristupovať len k meračom, ktoré sú momentálne zapojené. V tejto správe sa zároveň odošle aj momentálne nastavenie konfiguračných parametrov. Tieto údaje sú následne zobrazené v aktivite konkrétneho senzoru. Lekár môže tieto parametre editovať.



Obrazovka výberu monitoringu



Obrazovka na výber senzoru

Spustenie merania

Analýza

Po nadviazaní spojenia s meracím zariadením môžeme pristúpiť k získavaniu dát. Pri tom však musíme vziať do úvahy povahu meraní. Merania by sa dali rozdeliť do dvoch skupín: jednorázové merania a kontinuálne merania. Pri jednorázových meraniach (teplota, okysličenie krvi) je výsledkom merania jedna diskrétna hodnota. Meranie končí dorúčením výsledku merania (jediná správa). Pri kontinuálnych meraniach (EKG, EMG) je pre dosiahnutie výpovednej hodnoty merania potrebné získavať spojité dáta po dobu dlhšieho časového intervalu. Výstupom je teda set hodnôt v čase, ktoré sú používateľovi zobrazené vo forme grafu vo formáte hodnota/čas. Nepreržitý prenos údajov by znamenal príliš veľké energetické zaťaženie senzorov. Z tohto dôvodu musí prenos údajov prebiehať po dávkach, napríklad každú minútu.

Návrh

Pre normalizáciu komunikácie sme sa rozhodli, že veškeré dáta budú prenášané vo formáte JSON. Spustenie merania prebehne odoslaním riadiacej správy. Tento krok je pre oba typy merania rovnaký. V prípade jednorázového merania je priebeh jednoduchší. Po prijatí správy senzorom je okamžite vykonané meranie, dáta sú transformované do formátu JSON a následne odoslané do mobilnej aplikácie. Po odoslaní dát senzor očakáva odpoveď o úspešnom prijatí. Ak táto správa nepríde do 10 sekúnd, alebo je prijatá správa hovoriaca o neúspešnom prijatí dát, odošle senzor údaje znova. V opačnom prípade je pamäťový blok obsahujúci JSON s dátami z pamäte senzoru uvoľnený. Namerané hodnoty sú následne vypísané do mobilnej aplikácie. Po vypísaní dát má používateľ možnosť dáta odoslať na server. Tento scenár je však už súčasťou iného modulu. V prípade kontinuálnych meraní je komunikácia o niečo zložitejšia. Keďže zariadenia Arduino nepodporujú multithreading je potrebné prispôbiť komunikačný mechanizmus i samotný priebeh merania. Rozhodli sme sa, že odoslanie dát prebehne každú minútu. Z tohto rozhodnutia vyplýva nasledovné. Po prijatí správy, začne senzor proces merania. Meranie prebieha po dobu jednej minúty. Po uplynutí tohto času sú namerané dáta zabalené do formátu JSON a odoslané do mobilnej aplikácie. Následne prebehne kontrola buffru komunikačného kanálu zariadenia Arduino. Ak buffer neobsahuje správu ukončujúcu meranie, senzor opakuje tento postup a zahajuje nové meranie. Optimalizácia nastavenia intervalu meraní pre maximalizovanie efektivity komunikácie bude predmetom práce ďalšieho semestra.

Zoznam senzorov:

- **Tlak + Tep**
 - omron intelli rs7
- **Teplota**
 - MXL90614ESF-BA
- **EKG**
 - <https://rlx.sk/sk/biometric-medical-e-health-sensor-eeg-ekg/5422-bl18-ecg-maven-im160729001-ecg-monitoring-module-based-on-nrf51822-ble-bl1860-uartble.html>
- **Oximeter**
 - <https://rlx.sk/sk/biometric-medical-e-health-sensor-eeg-ekg/7367-sparkfun-pulse-oximeter-and-heart-rate-sensor-max30101-max32664-qwiic-sf-sen-15219.html>
- **EMG**
 - <https://rlx.sk/sk/biometric-medical-e-health-sensor-eeg-ekg/5557-myoware-muscle-sensor-sparkfun-sen-13723.html>

Formát správ:

Riadiace správy:

```
{  
  "id":          "987642",      -int    //identifikátor zariadenia  
  "config":      "true",        -bool  
  "measurement": "false",      -bool  
}
```

Začiatok merania:

```
    config: true          measurement: true
```

Koniec merania:

```
    config: false        measurement: true
```

Opätovné vyžiadanie údajov:

```
    config: true          measurement: false
```

Ukončenie spojenia:

```
    config: false        measurement: false
```

Implementácia

Nadviazanie spojenia v princípe prebieha podľa návrhu z predchádzajúcej kapitoly. Jednorázové merania sú pomerne priamočiare. Zmenou oproti návrhu je, že pre spustenie monitoringu sa zo zariadenia odošle správa s flagmi config: false a measurement: true. Po prijatí správy zariadenie spustí proces merania, ktoré prebieha cyklicky podľa nastavených parametrov. Tieto hodnoty sú priemerované a odoslané na mobilné zariadenie. Pri kontinuálnych meraniach sme sa kvôli problémom s hardvérom museli mierne odkloniť od návrhu. Spustenie merania prebieha rovnako ako v prípade jednorázových meraní. V tomto prípade však merania musia prebiehať kontinuálne s kadenciou niekoľkých hodnôt za sekundu. Pre získanie dát s minimálnou výpovednou hodnotou bolo v prípade EMG merača získavať aspoň 10 hodnôt za sekundu. Pôvodne sme plánovali dať používateľovi možnosť zmeniť tieto konfiguračné hodnoty. Pri testovaní sme však narazili na problém s obmedzenou pamäťou Arduina. Pri odosielaní viac ako 10 hodnôt sa pri behu programu prepisoval odosielaný JSON, v dôsledku čoho nebolo možné spracovať túto správu na mobilnom zariadení. Pri zmenšení obnosu prenášaných dát sa nám tento problém podarilo eliminovať. Prenos pri kontinuálnych meraniach preto prebieha raz za sekundu, čo sa prejavuje zvýšením energetických nárokov senzoru. Program je však pripravený na fungovanie podľa návrhu a pri zmene hardvéru je jeho úprava na normálne fungovanie triviálna.

Začiatok merania:

config: false **measurement:** true

Koniec merania:

Kontinuálne **config:** false **measurement:** false **connection:** true

Vyžiadanie konfigurácie

config: true **measurement:** false

Ukončenie spojenia:

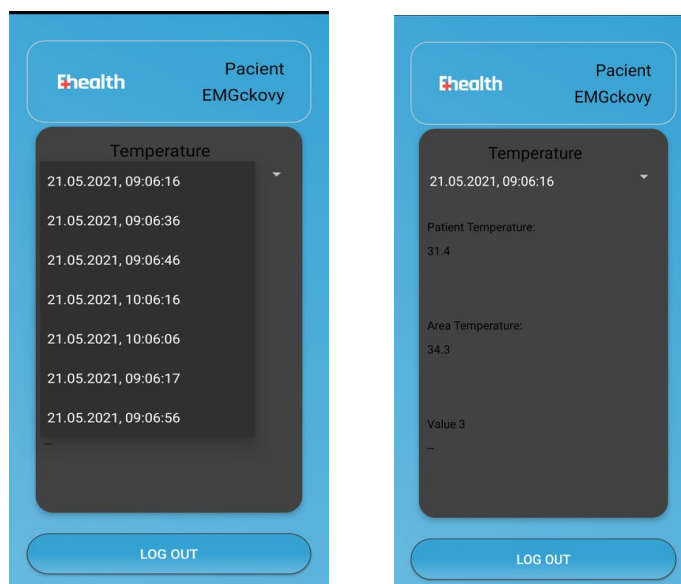
Jednorázové **config:** false **measurement:** false

Kontinuálne **config:** false **measurement:** false **connection:** false

Po prihlásení Lekára do Mobilnej aplikácie, lekár vidí monitorovania, ktoré spustil cez webovú aplikáciu a ktoré sú v statuse Initialized a Active. Tu sa mu zobrazí meno Pacienta,

meno príslušného Arduino zariadenia a dátum začiatku monitorovania. Monitorovania, ktoré ešte neboli zahájené, t.j. dátum začiatku pochádza z budúcnosti, sú zobrazené, no nie je možné na dané monitorovania kliknúť. Po kliknutí na zariadenie sa mobilný telefón pripojí na Arduino a po úspešnom spárovaní sa lekár dostáva na Hlavné menu, v ktorom môže vidieť dostupné merače.

Pri prihlásení Pacienta do mobilnej aplikácie má pacient možnosť prezerať si namerané hodnoty.



Zobrazenie nameraných hodnôt pacientom v mobilnej aplikácii



Výber merača lekárom

Konfigurácia merača

Zobrazenie výsledkov

Webová aplikácia:

Ako už sme skôr spomínali v module Profil, spúšťanie monitorovanie je realizované vo webovej aplikácii v profile daného pacienta, pre ktorého chceme spustiť monitorovanie.

Nižšie je implementovaná obrazovka vo webovej aplikácii:

Miesto bydliska

Monitorovanie pacienta

Spustenie monitorovania pre pacienta Pacient EMGckovy na nasledujúcich 24 hodín. Vyberte jedno z dostupných Arduino zariadení, z ktorého bude meranie realizované. Po spustení pokračujte na mobilnom telefóne.

Arduino:

Je potrebné vybrať jedno zo zariadení

Dátum začiatku:

dd.mm.rrrr 13:00

Dátum skončenia monitorovania:

dd.mm.rrrr 13:00

Spustiť monitorovanie **Ukončiť**

Pacient má možnosť vidieť jeho merania, alebo doktor môže vidieť merania, ktoré on inicializoval. Ukážka nižšie.

Pacient/lekár má možnosť vidieť monitorovania v tabuľke a následne sa prekliknúť na detail tohto daného monitorovania.

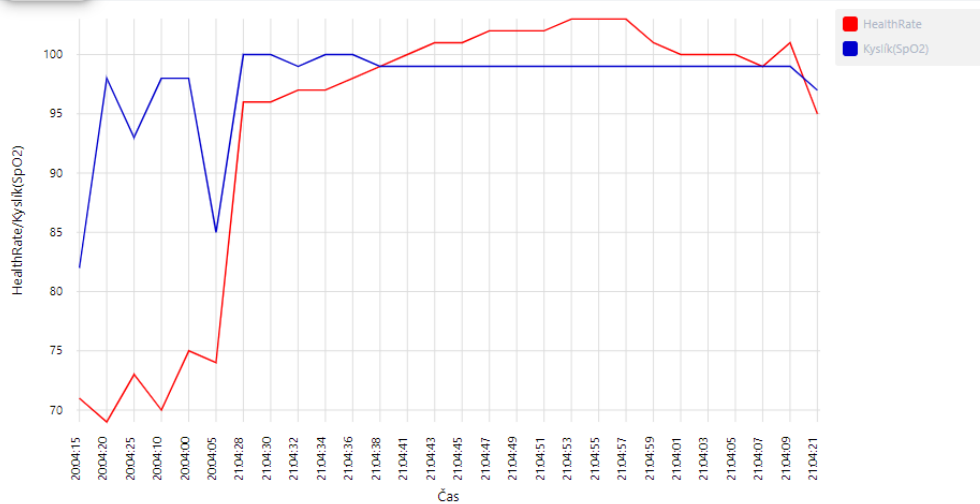
List monitorovaní

Meno	Priezvisko	Začiatok monitorovania	Koniec monitorovania	Stav monitorovania	
Pacient	EMGckovy	2021-04-16 19:00:00	2021-04-17 19:00:00	Finished	» Detail
Pacient	EMGckovy	2021-05-21 17:50:00	2021-05-21 23:50:00	Active	» Detail
Pacient	EMGckovy	2021-05-07 10:40:00	2021-05-07 10:41:00	Finished	» Detail
Pacient	EMGckovy	2021-05-06 13:00:00	2021-05-06 14:00:00	Finished	» Detail
Pacient	EMGckovy	2021-05-03 15:00:00	2021-05-03 16:00:00	Finished	» Detail
Pacient	EMGckovy	2021-05-02 08:42:13	2021-05-03 08:42:13	Finished	» Detail
Pacient	EMGckovy	2021-04-26 17:00:00	2021-04-30 20:00:00	Finished	» Detail
Samo	Samkovyo	2020-04-02 17:00:00	2020-04-02 20:00:00	Finished	» Detail

« 1 »

Meno pacienta: Pacient EMGckovy Dátum začiatku: 2021-04-26 17:00:00 Dátum ukončenia: 2021-04-30 20:00:00 Stav: Finished

Oxymeter Teplota Tlak EKG EMG



Pacient aj lekár vidí dané monitorovanie (detail). Obdobne je spravené aj teplota, tlak atď.

Implementácia vo webovej aplikácii:

Pre získanie monitorovaní slúži servis s názvom `searching.service.ts`. Tento súbor obsahuje metódy pre získanie príslušných hodnôt.

Príklad implementácie získania hodnôt pre teplotu:

```
getTemperature(monitoringID) {
  return this.http.get(environment.apiUrl + environment.apiEndpointOneMonitoring + '/' + monitoringID + '/temperature/series', this.options)
    .pipe(
      map(data => {
        return data;
      })
    );
}
```

Komponent pre zobrazenie monitorovania sa nachádza v menu adresári v priečinku `monitoring`. Komponent `monitoring` slúži pre zobrazenie všetkých monitorovaní, a `monitoring.detail.component.ts` slúži pre zobrazenie jedného merania.

Implementácia endpointov pre monitorovanie:

Spustenie monitorovania

Vytvorenie monitoring session

HTTP POST metóda na url v tvare: `{{url}}/api/monitorings/initialize`

Popis

môže posilať iba prihlásený Doctor

medzi Doctorom a Pacientom musí byť vytvorený `UserConnection` (`UserConnection` - send request)

stav spojenia na oboch stranách musí byť `Accepted` (`UserConnection` - change state)

Po vytvorení monitoring je stav = "Initialized"

Na jednom Pacientovi v jednom čase môže byť len jedno monitoring session

Na jednom Zariadení v jednom čase môže byť len jedno monitoring session

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer {{token}}

V tele správy zadávam parametre vytváraného **Monitoringu** viac nižšie:

Príklady requestov:

```
1 {
2   "PatientId":"b37fbda6-cd4a-4243-ab9a-f06c53dbe75d",
3   "DeviceId":"18-72-3C-45-C5-4D",
4   "StartTime":"2020-04-02 17:00:00",
5   "EndTime":"2020-04-02 20:00:00"
6 }
```

- patientId
 - id pacienta s ktorým sa chcem spojiť (možem získať napr. pomocou User - list patients)
 - musí byť validný Guid
 - pacient sa musí nachádzať v systéme
- DeviceId
 - MAC adresa Arduino zariadenia (momentálne nie je jasné ako ju získavame ????)
 - Musí byť validná MAC adresa t.j: šesť dvojíc hexadecimálnych číslíc, oddelených ':' alebo '-'
- StartTime
 - začiatkový čas monitorovania
- EndTime
 - koncový čas monitorovania

Odpoveď:

200 OK

prázdne telo

Request ak už na danom zariadení v danom čase prebieha monitoring

```
1 {
2   "PatientId":"16919a15-4af3-4606-92e5-e3fe9cb08011",
3   "DeviceId":"18-72-3C-45-C5-4D",
4   "StartTime":"2020-04-02 17:00:00",
5   "EndTime":"2020-04-02 20:00:00"
6 }
```

Odpoved':

400 Bad Request

```
1 {
2   "errors": {
3     "DeviceId": "Device '18-72-3C-45-C5-4D' already in use in time interval 2. 4. 2020 17:00:00 - 2020-04-02 20:00:00"
4   }
5 }
```

Request ak už na danom pacientovi v danom čase prebieha monitoring

```
1 {
2   "PatientId":"16919a15-4af3-4606-92e5-e3fe9cb08011",
3   "DeviceId":"18-72-3C-45-C5-4A",
4   "StartTime":"2020-04-02 17:00:00",
5   "EndTime":"2020-04-02 20:00:00"
6 }
```

Odpoved':

400 Bad Request

```
1 {
2   "errors": {
3     "PatientId": "Monitoring already initiated or active for patient '16919a15-4af3-4606-92e5-e3fe9cb08011'"
4   }
5 }
```

Získanie mojich monitorovaní

Zoznam monitoringov

HTTP GET metóda na url v tvare: **{{url}}/api/monitorings**

Popis

- Ak som **Doctor**
 - request vráti **všetky monitoringy**, ktoré som **vytvoril** a sú v stave **Initialized** alebo **Active**
- Ak som **Patient**
 - request vráti **všetky monitoringy**, v ktorých **figurujem** a sú v stave **Initialized** alebo **Active**

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer {{token}}

Parametre volania

Request nemá žiadne parametre (do budúcnosti budeme možno meniť)

Možné filtre:

Request nemá žiadne parametre (do budúcnosti budeme možno meniť)

Príklady requestov:

Request ak som doktor

Odpoveď:

200 OK

```

1  [
2    {
3      "patient": {
4        "id": "b6914507-bbed-430f-8604-4e444bd1b0d7",
5        "firstName": "Petko1",
6        "lastName": "Ferkovsky1"
7      },
8      "deviceId": "18-72-3C-45-C5-4D",
9      "startTime": "2020-04-02T15:00:00",
10     "endTime": "2020-04-02T16:59:00",
11     "monitoringId": "47b2f265-2eb2-446c-8bec-8c04fde3371e",
12     "monitoringState": "Initialized"
13   },
14   {
15     "patient": {
16       "id": "b37fbda6-cd4a-4243-ab9a-f06c53dbe75d",
17       "firstName": "Maria",
18       "lastName": "Petkovska"
19     },
20     "deviceId": "18-72-3C-45-C5-4D",
21     "startTime": "2020-04-02T17:00:00",
22     "endTime": "2020-04-02T19:00:00",
23     "monitoringId": "fdb4117d-8126-412b-bfeb-bc02fa3c39c4",
24     "monitoringState": "Active"
25   },
26   {
27     "patient": {
28       "id": "16919a15-4af3-4606-92e5-e3fe9cb08011",
29       "firstName": "Janko",
30       "lastName": "Hrasko"
31     },
32     "deviceId": "18-72-3C-45-C5-4A",
33     "startTime": "2020-04-02T17:00:00",
34     "endTime": "2020-04-02T20:00:00",
35     "monitoringId": "7c81fae7-3645-44a9-9172-a0d5b8ae476b",
36     "monitoringState": "Initialized"
37   }
38 ]

```

patient- objekt pacienta

id - id pacienta

firstName - meno pacienta

lastName - priezvisko pacienta

deviceId - MAC adresa Arduino zariadenia

StartTime - začiatkový čas monitorovania

EndTime - koncový čas monitorovania

monitoringId - id monitorovania (monitoring session)

monitoringState - stav daného monitoring

Získanie daných hodnôt pre jednotlivé meranie

Pre získanie daných hodnôt bol vytvorený samostatný endpoint. Pre ukážku si ukážeme ako získať teplotu pacienta, a obdobne sú spravené aj iné merače.

Zoznam nameraných hodnôt na senzore teploty

HTTP GET metóda na url v tvare: `{{url}}/api/monitorings/{{guid MonitoringId}}/temperature`

napr. : `{{url}}/api/monitorings/d7738ce2-c9b1-42e9-bc8c-05072610a33a/temperature`

Popis

- Keď zavolám request **bez filtrov**, vráti **všetky hodnoty**
- Hodnoty sú usporiadané **zostupne** podľa času, t.j. najnovšia hodnota je prvá

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer `{{token}}`

Parametre volania

Jednotlivé položky, podľa ktorých filtrujem sa zadávajú ako query parametre.

Filtre sa v podmienke AND-ujú.

Možné filtre:

- limit - max. počet výsledkov, ktoré api vráti (na webe veľkosť stránky)
- offset - o koľko prvkov sa mám posunúť (na webe číslo stránky * veľkosť stránky)
- startTime - začiatkový čas od, ktorého chcem hodnoty
- endTime - posledný čas do, ktorého chcem hodnoty
- maxTempPatient - self explanatory
- minTempPatient - self explanatory

- maxTempArea - self explanatory
- minTempArea - self explanatory

Príklady requestov:

Url: `{{url}}/api/monitorings/d7738ce2-c9b1-42e9-bc8c-05072610a33a/temperature`

Odpoved':

200 OK

```
1 {
2   "data": [
3     {
4       "tempPatient": 36.9,
5       "tempArea": 23.0,
6       "timestamp": "2020-04-25T19:56:06"
7     },
8     {
9       "tempPatient": 36.7,
10      "tempArea": 25.5,
11      "timestamp": "2020-04-25T19:56:05"
12     },
13     {
14       "tempPatient": 36.9,
15       "tempArea": 23.0,
16       "timestamp": "2020-04-24T19:56:06"
17     },
18     {
19       "tempPatient": 36.7,
20       "tempArea": 24.5,
21       "timestamp": "2020-04-24T19:56:05"
22     },
23     {
24       "tempPatient": 36.9,
25       "tempArea": 23.0,
26       "timestamp": "2020-04-17T19:56:01"
27     },
28     {
29       "tempPatient": 36.7,
30       "tempArea": 24.5,
31       "timestamp": "2020-04-17T19:56:00"
32     }
33   ],
34   "count": 6
35 }
36
```

- data- pole nameraných hodnôt na senzore v čase
 - TempPatient - teplota pacienta, dátový typ decimal
 - TempArea - teplota okolia, dátový typ decimal
 - Timestamp - časová pečiatka, kedy bolo meranie vykonané. ISO formát
- count - celkový počet výsledkov

Profil

Tento modul obsahuje profil používateľa. Profil zahŕňa osobné údaje, ako aj možnosť editovania profilu, či zmeny hesla používateľa. V profile taktiež je možné spustiť monitorovanie.

Analýza

Analýza pre tento modul bola základná. Keďže pri registrácii máme všetky potrebné údaje, tento modul slúži na zobrazenie týchto údajov vo webovej aplikácii, aby pri spájaní doktorov s pacientmi, pacienti či lekári mali možnosť vidieť o koho sa jedná. V tejto fáze sme si zanalyzovali aké údaje potrebujeme mať na strane pacienta a taktiež na strane lekára.

Premýšľali sme ako spravíme jednotný formát pre obe strany, a podarilo sa nám navrhnuť riešenie kde lekár má viac funkcionalít v module profil. Napríklad ak ide o prezeranie pacientovho profilu tak lekár má možnosť odoslať žiadosť o spojenie, a následne aj spúšťať dané monitorovanie pacientovi.

Taktiež je nutná editácia profilu, čo bude navrhnuté štandardným spôsobom.

V analýze sme si ešte stanovili čo vidí pacient/lekár na základe spojenia. Ak lekár nieje spojený s pacientom tak nemá možnosť vidieť pacientovo rodné číslo, či iné hlbšie údaje. Viac je to opísane v implementácii, kde podľa response zo servera máme možnosť vidieť kedy nám vráti aké informácie.

Návrh

Spravili sme si základné náčrty daného modulu, ktorý sa samozrejme bude iteratívne meniť.

Tento modul si bude vyžadovať komunikáciu so serverom pre získanie, editovanie profilu a taktiež zmeny hesla.

E-Health

Osobný profil / Zdravotný profil

Meno:

Priezvisko:

Titul pred menom:

Titul za menom:

Dátum narodenia:

Rodná číslo:

Pohlavie: Čižna Muž

Rodný stav:

Povolanie:

Ulica:

Mesto:

PSČ:

Krajina:

Telefónne číslo:

Email:

Aktualizovať údaje

Zmeniť heslo

Náčrtok profilu z pohľadu používateľa

E-Health

Osobný profil / Zdravotný profil

Meno:

Priezvisko:

Titul pred menom:

Titul za menom:

Dátum narodenia:

Rodná číslo:

Pohlavie: Čižna Muž

Rodný stav:

Povolanie:

Ulica:

Mesto:

PSČ:

Krajina:

Telefónne číslo:

Email:

Aktualizovať údaje

Zmeniť heslo

Náčrtok profilu z pohľadu keď lekár vyhľadáva pacienta

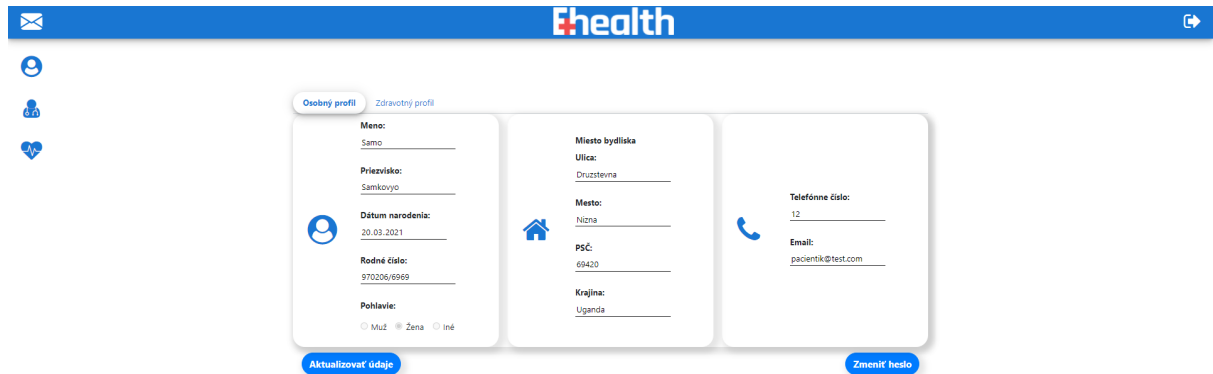
Implementácia

Implementácia modulu Profil je riešená ako sme si už skôr spomínali v analýze a návrhu modulu. Nižšie sú ukážky implementácie modulu ako aj opísane endpointy, ktoré slúžia na komunikáciu frontendu s backendom.

Hlavné časti implementácie modulu:

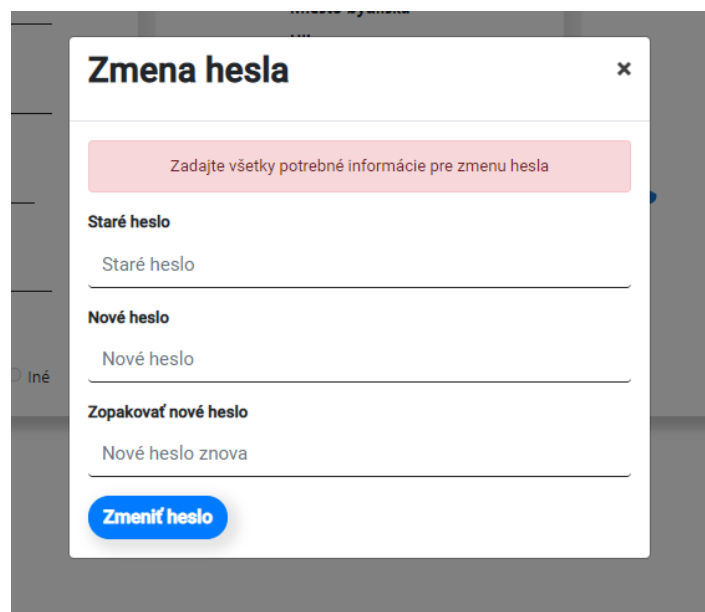
Pre komunikáciu so serverom slúži kľúčový service a tým je profile.service.ts. Tento komponent zahŕňa všetky potrebné metódy pre komunikáciu s backendom.

Vizuálna stránka modulu je implementovaná štandardným spôsobom. A teda názov .ts súboru je personal-profile.component.ts. Každý modul má vlastný routing, a je implementovaný štandardným spôsobom a intuitívne. Hlbší opis nie je nutný, na koľko názvy tried, funkcií sú tak napísané aby bolo jasné na čo slúžia.



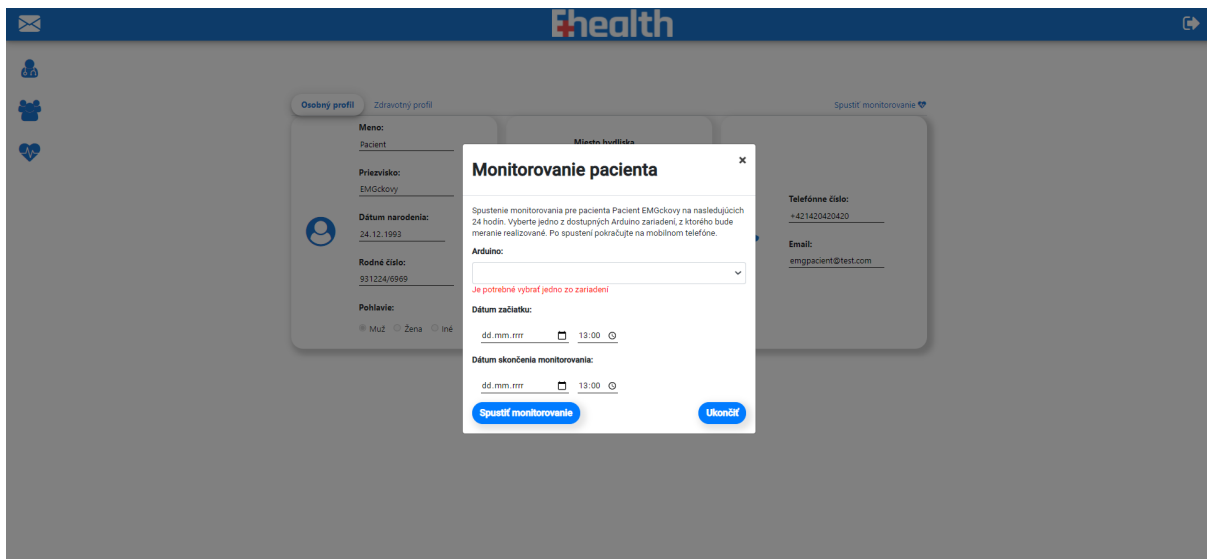
The screenshot shows a web application interface for a patient profile. At the top, there is a blue header with the 'health' logo and a home icon. Below the header, there are three navigation icons: a person, a person with a plus sign, and a heart. The main content area is a white card with a light blue border, divided into three columns. The first column is titled 'Osobný profil' and contains fields for 'Meno' (Name) with the value 'Samo', 'Priezvisko' (Surname) with 'Samkovyo', 'Dátum narodenia' (Date of birth) with '20.03.2021', 'Rodné číslo' (ID number) with '970206/9999', and 'Pohlavie' (Gender) with radio buttons for 'Muž', 'Žena', and 'Iné'. The second column is titled 'Miesto bydliska' (Residence) and contains fields for 'Ulica' (Street) with 'Družstevna', 'Mesto' (City) with 'Nizna', 'PSČ' (Postal code) with '69420', and 'Krajina' (Country) with 'Uganda'. The third column is titled 'Telefónne číslo' (Phone number) with the value '12' and 'Email' with 'pacientik@test.com'. At the bottom of the card, there are two blue buttons: 'Aktualizovať údaje' (Update data) and 'Zmeniť heslo' (Change password).

Implementácia profilu vo webovej aplikácii

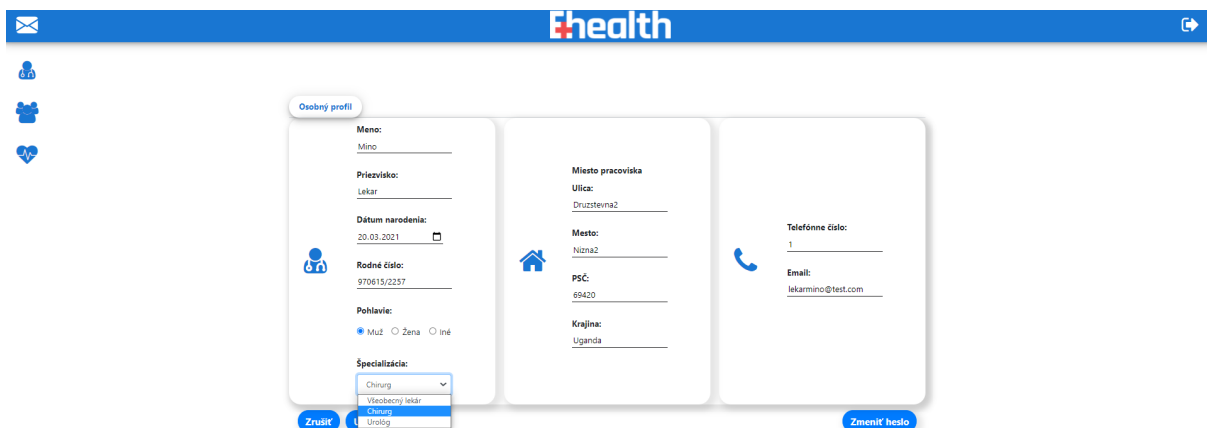


The screenshot shows a modal window titled 'Zmena hesla' (Change password) with a close button (X) in the top right corner. Below the title, there is a pink instruction box that says 'Zadajte všetky potrebné informácie pre zmenu hesla' (Enter all necessary information for password change). The form contains three input fields: 'Staré heslo' (Old password) with the value 'Staré heslo', 'Nové heslo' (New password) with the value 'Nové heslo', and 'Zopakovať nové heslo' (Repeat new password) with the value 'Nové heslo znova'. At the bottom of the form, there is a blue button labeled 'Zmeniť heslo' (Change password).

Implementácia zmeny hesla vo webovej aplikácii



Implementované spustenie monitorovania v profile pacienta



Implementácia profilu z pohľadu doktora, (lekár aj doktor) si vedia editovať svoje osobné údaje.

Implementácia endpointov spojenými s profilom:

Získať profil:

Získanie profilu používateľa podľa id

HTTP GET metóda na url v tvare: `{{url}}/api/profiles/personal/{{guid usra}}`

napr.: `http://localhost:5000/api/profiles/personal/b6914507-bbed-430f-8604-4e444bd1b0d7`

!!!! Správanie requestu je závislé na tom kto vykonáva request.

- pacient nevidí ostatných pacientov

- ak nie je vytvorené spojenie medzi **pacientom** a **lekárom**, lekár vidí len **niektoré údaje o pacientovi**
- ak je vytvorené medzi pacientom a lekárom **spojenie** a stav spojenia je **Accepted**, tak usri vidia svoje celé profily so všetkými údajmi.

User musí byť prihlásený:

V headroch, treba mať nastavené:

Authorization : Bearer {{token}}

Odpoveď ak som používateľ s rolou Doctor a spojenie nie je vytvorené:

```

1  {
2    "firstName": "Petko",
3    "lastName": "Ferkovsky",
4    "street": "Druzstevna 852",
5    "city": "Nizna",
6    "zip": "69420",
7    "country": "Uganda",
8    "dateOfBirth": "1990-02-07T00:00:00",
9    "sex": "Female",
10   "role": "Patient",
11   "connectionState": null
12  }
```

sex - pohlavie používateľa, možné hodnoty: **Female, Male ,Undefined**

role- rola používateľa

connectionState = null → spojenie zatiaľ nebolo ani iniciované

Odpoveď ak som používateľ s rolou Doctor a spojenie je vytvorené :

```
1 {
2   "firstName": "Petko",
3   "lastName": "Ferkovsky",
4   "street": "Druzstevna 852",
5   "city": "Nizna",
6   "zip": "69420",
7   "country": "Uganda",
8   "dateOfBirth": "1990-02-07T00:00:00",
9   "sex": "Undefined",
10  "nim": "970206/6009",
11  "role": "Patient",
12  "phoneNumber": "+421420420420",
13  "email": "pussyslayer69@test.com",
14  "connectionState": "Accepted"
15 }
```

"connectionState": "Accepted" → spojenie je prijaté

Odpoveď ak som používateľ s rolou Patient a snažím sa vyhládať iného pacienta:

403 Forbidden

A obdobne to je aj pri ďalších variantoch získania profilu. Je to spravené všeobecne aby pre jeden endpoint sme vedeli dostať aj svoj profil ale aj profil iného pacienta, či doktora, a teda snažíme sa o to aby v kóde sa nevyskytovali redundantné časti kódu.

Editovanie profilu:

HTTP PUT metóda na url v tvare: *{{url}}/api/profiles/personal*

Poz.: V aplikáciách (pre produkčné prostredie) treba *{{url}}* nahradiť ip adresou, alebo doménovým menom napr.: *http://team18-20.studenti.fiit.stuba.sk* (Celá url teda vyzerá :*http://team18-20.studenti.fiit.stuba.sk/api/user/registerDoctor*)

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer *{{token}}*

- Request zmení údaje profilu usra podľa zadaných properties v Jsone.
- Údaje o tom, ktorému usrovi mením údaje sú v tokene pomocou, ktorého sa user autorizuje.
- Request je jednotný pre doktora aj pacietna.
- v jsone zdávam len tie údaje, ktoré chcem zmeniť, ostatné by mali byť null
- Ak mením údaje špecifické pre doktora, ako doctorId alebo doctorTypeld , tak ich stačí len zadať do jsonu, netreba nič iné meniť
- v princípe je možné meniť všetky parametre, ktoré používateľ zadal pri registrácii

Zmena Hesla:

Zmena hesla

HTTP POST metóda na url v tvare: `{{url}}/api/user/changePassword`

User musí byť prihlásený

V headroch, treba mať nastavené:

Content-Type : application/json

Prirodzene treba byť prihlásený a posielat' v headroch:

Authorization : Bearer `{{token}}`

V tele správy:

```
1 {
2   "OldPassword": "Pa$$w0rd0000",
3   "NewPassword": "Pa$$w0rd"
4 }
```

Odpoveď ak je zmena hesla úspešná:

200 OK - bez tela správy

Odpoveď ak je zmena hesla neúspešná, kvôli zlému starému heslu:

```
1 {
2   "errors": [
3     {
4       "Code": "PasswordMismatch",
5       "Description": "Incorrect password."
6     }
7   ]
8 }
```

Implementácia profilu vo webovej aplikácii:

Vo webovej aplikácii sa o profil stará skôr spomínaný `profile.service.ts`.

Tento súbor obsahuje metódy, ktoré komunikujú so serverom, a podľa prihláseného používateľa sa nájde a vráti dotýčny profil.

Ukážka metód implementovaných pre prípad použitia profil:

```

getProfile(id): Observable<User> {
  //tu sa vrati zo servera profil na zaklade id
  return this.http.get<User>(environment.apiUrl + environment.apiEndpointGetProfile + id, this.options)
    .pipe(
      map(data => {
        return data;
      })
    );
}

editProfile(bodyRequest) {
  return this.http.put(environment.apiUrl + environment.apiEndpointEditProfile, bodyRequest, this.options)
    .pipe(
      map(data => {
        return data;
      })
    );
}
}

```

Profil je riešený všeobecne, to znamená, že komponent je dynamický, a na základe roly, a zadaného ID, sa zobrazí používateľovi príslušné dáta.

Vyhľadávanie

Tento modul rieši problematiku prehliadania a vyhľadávania používateľov.

Analýza

Používatelia v našom systéme musia byť schopní medzi sebou interagovať. Začiatkom takejto interakcie je vyhľadanie používateľa.

Výsledkom analýzy je, že pacienti budú môcť vyhľadávať iba lekárov a lekári iba pacientov.

Pri výslednom zozname lekárov budú pre každého zobrazené nasledovné stĺpce: Meno, Priezvisko, Špecializácia, Mesto + preklik na detail. Pri zozname pacientov to budú: Meno, Priezvisko, Dátum narodenia, Mesto + preklik na detail.

Dôležitou súčasťou sú filtre, ktoré budú pre každý stĺpec. Taktiež si bude možné odfiltrovať používateľov, s ktorými je daný používateľ spojený.

Návrh

Prípady použitia

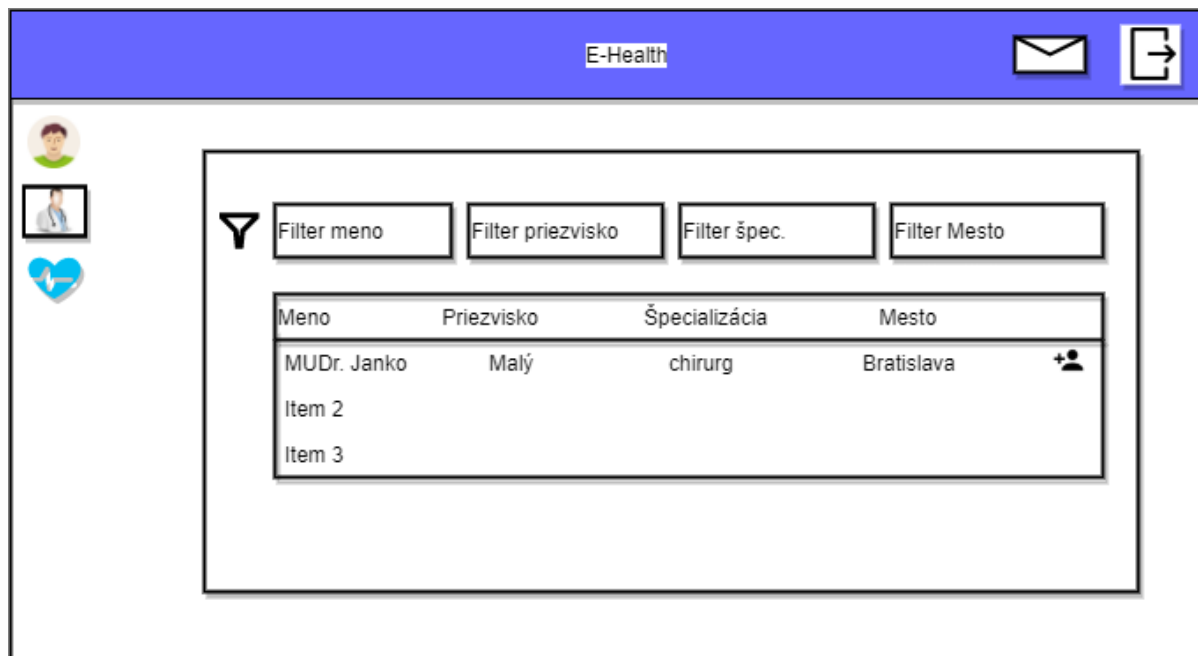
Všetky identifikované prípady použitia (viď obr. nižšie) budú prebiehať v rámci webovej aplikácie. Nasleduje opis konkrétnych prípadov použitia:

- **Vyhľadanie lekára** - Pacient kliká na tlačidlo Lekári. V zozname sa objavia všetci lekári a pacient nájde toho, o ktorého má záujem.
- **Vyhľadanie pacienta** - Lekár kliká na tlačidlo Pacienti. V zozname sa objavia všetci pacienti a lekár nájde toho, o ktorého má záujem.
- **Filtrovanie na základe stĺpcov** - Používateľ vyplní jednotlivé filtre. Dynamicky sa mení zoznam používateľ podľa filtrov.
- **Vyfiltrovanie aktívnych spojení** - Používateľ kliká na checkbox Zobrazit' len pacientov s ktorými som spojený. V zozname sú len používatelia s ktorými je spojený.
- **Preklik na detail profilu** - Používateľ kliká na detail v riadku používateľa, ktorého profil si chce otvoriť. Otvorí sa mu okno jeho osobného profilu.

Náčrty okien modulu

Súčasťou návrhu boli aj návrhy okien tohoto modulu.

Okno vyhľadávania



Implementácia

Frontend

Okno Vyhľadávania

searching.component.ts – obsahuje aplikačnú logiku Vyhľadávania

searching.service.ts – tento súbor obsahuje komunikáciu so serverom, a čaká sa na validáciu zadaných údajov

searching.component.html – tento HTML súbor obsahuje template pre grafické zobrazenie Vyhľadávania

searching.component.css – súbor obsahuje štýly pre Vyhľadávanie

Na obrázku nižšie môžeme vidieť okno aplikácie pre Vyhľadávanie z pohľadu lekára.

Meno	Priezvisko	Dátum narodenia	Mesto	
Alojz	<script>alert(1)</script>	12.12.1998	Bratislava	» Detail
Pacient	EMGckovy	24.12.1993	Bratislava	» Detail
Petko	Ferkovsky	07.02.1990	Nizna	» Detail

Na obrázku nižšie môžeme vidieť okno aplikácie pre Vyhľadávanie z pohľadu pacienta.

Meno	Priezvisko	Špecializácia	Mesto	
Ondrej	Chirurgicky	Chirurg	Bratislava	» Detail
Mino	Lekar	Chirurg	Nizna2	» Detail
Testo	Lekarsky	Všeobecný lekár	bla	» Detail

API endpointy

User - list patients

Zoznam pacientov/ podľa zadaného filtra

HTTP POST metóda na url v tvare: `{{url}}/api/user/listPatients`

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer `{{token}}`

Tento request môže vykonať jedine používateľ s rolou Doctor. (Ak ho vykonám ako Patient, tak API vráti 403 Forbidden)

V tele správy hodnoty filtra.

Ak sa nefiltruje podľa daného parametra, tak mu treba nastaviť hodnotu null, resp. nemusíme ani uvádzať daný parameter.

```
1 {
2   "limit" : null,
3   "offset" : null,
4   "dateOfBirth": "1990-02-07",
5   "firstName": null,
6   "lastName": null,
7   "city": null
8 }
```

limit - max. počet výsledkov, ktoré api vráti (na webe veľkosť stránky)

offset - o koľko usrov sa má posunúť (na webe číslo stránky * veľkosť stránky)

ostatné parametre - self explanatory

Jednotlivé parametre sa v podmienke AND-ujú.

Odpoveď :

```
1 {
2   "users": [
3     {
4       "id": "b6914507-bbed-430f-8604-4e444bd1b0d7",
5       "firstName": "Petko",
6       "lastName": "Ferkovsky",
7       "dateOfBirth": "1990-02-07T00:00:00",
8       "city": "Nizna"
9     },
10    {
11      "id": "b7484837-78bf-478e-b487-9214f5052cba",
12      "firstName": "Samo",
13      "lastName": "Samkovy",
14      "dateOfBirth": "1990-02-07T00:00:00",
15      "city": "Nizna"
16    }
17  ],
18   "count": 2
19 }
```

V odpovedi sa nachádza pole user objektov a počet všetkých nájdených usrov, ktorí spĺňajú podmienky zadané vo filtri.

User - list doctors

Zoznam pacientov/ podľa zadaného filtra

HTTP POST metóda na url v tvare: `{{url}}/api/user/ListDoctors`

V headroch, treba mať nastavene:

Content-Type : application/json

Authorization : Bearer `{{token}}`

V tele správy hodnoty filtra.

Ak sa nefiltruje podľa daného parametra, tak mu nastaviť hodnotu null, resp. nemusíme ani uvádzať daný parameter.

```
1 {
2   "limit" : null,
3   "offset" : null,
4   "DoctorTypeId": 2,
5   "firstName": null,
6   "lastName": null,
7   "city": null
8 }
```

limit - max. počet výsledkov, ktoré api vráti (na webe veľkosť stránky)

offset - o koľko usrov sa má posunúť (na webe číslo stránky * veľkosť stránky)

DoctorTypeId - id špecializácie doktora

ostatné parametre - self explanatory

Jednotlivé parametre sa v podmienke AND-ujú.

Odpoveď:

```
1 {
2   "users": [
3     {
4       "id": "07480f37-18bb-41a2-ac4d-404a1b975a71",
5       "doctorTypeId": 2,
6       "firstName": "Ondrej",
7       "lastName": "Chirurgicky",
8       "city": "Bratislava"
9     }
10  ],
11   "count": 1
12 }
```

V odpovedi sa nachádza pole user objektov a počet všetkých nájdených usrov, ktorí spĺňajú podmienky zadané vo filtri.

Alarmy

Modul Alarmy umožňuje používateľovi vykonávať základné CRUD operácie nad entitou alarm. Následne sa pri plnení nameraných údajov na jednotlivých senzoch kontroluje, či vkladaná hodnota neprekročila nastavený limit. V prípade, že daná hodnota prekročila limit, tak je vytvorená udalosť.

Prípady použitia:

1. Vytvorenie alarmu
2. Zobrazenie všetkých alarmov používateľa
3. Vymazanie alarmu
4. Vytvor udalosť

Vytvorenie alarmu

Analýza

Alarm sa definuje nad konkrétnym **senzorom** pričom sa takisto udáva, ktorú **veličinu** má na danom senzore monitorovať. Následne sa zadefinuje **hraničná hodnota a porovnávací operátor** (väčší, menší, väčší alebo rovný ...). Takisto je možné určiť parameter "**BackoffPeriod**", ktorý určuje ako často majú alarmy toho istého typu vznikajú. Napr. ak má pacient mierne zvýšenú teplotu, lekár môže nastaviť, že je o tomto fakte upozornený každú hodinu nakoľko sa nejedná o vážnu udalosť. Posledné parametre, ktoré je na alerme možné nastaviť je : **správa**, ktorá sa zobrazí v prípade, že nastane daná udalosť a **stupeň závažnosti**. V systéme sú zadefinované 4 stupne závažnosti.

Používateľ môže mať zadefinovaných **neobmedzený počet alarmov** nad ktorýmkoľvek senzom. Pri **plnení** sa kontrolujú všetky zadefinované alarmy.

Návrh

Funkcionalita na webovej aplikácii nie je v tejto verzii produktu implementovaná, tým pádom nie je vytvorený ani grafický návrh.

Implementácia(API)

HTTP POST metóda na url v tvare: **{{url}}/api/alarm**

Poz.: V aplikáciách (pre produkčné prostredie) treba **{{url}}** nahradiť ip adresou, alebo doménovým menom napr.: <http://team18-20.studenti.fiit.stuba.sk> (Celá url teda vyzerá <http://team18-20.studenti.fiit.stuba.sk/api/alarm>)

Popis

- Umožňuje vytvorenie **alarmu** nad daným **senzorum**
- Alarm vytvára používateľ **pod svojim kontom**, musí byť teda riadne **autentifikovaný**

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer {{token}}

V tele správy zadávam parametre vytváraného **Alarmu**

Príklady requestov:

```
1 {
2   "Sensor": "Pressure",
3   "Unit": "Sys",
4   "Value": 180,
5   "Operator": "ge",
6   "BackoffPeriod": 3600,
7   "Message": "You are about to pop",
8   "SeverityLevel": "High"
9 }
```

- **Sensor**
 - Senzor, nad ktorým sa definuje daný alarm
 - Možné hodnoty: Temperature, Oxymeter, EKG, EMG, Pressure
- **Unit**
 - Meraná veličina na danom senzore
 - Možné hodnoty pre jednotlivé senzory:
 - Senzor Temperature
 - TempPatient, TempArea

- Senzor Oxymeter
 - Healthrate, Oxygen
 - Senzor EKG
 - Bpm
 - Senzor EMG
 - Emg
 - Senzor Pressure
 - Sys, Dia, Pulse
- Value
 - Hraničná hodnota
 - decimal
- Operator
 - Operátor porovnania, ktorý sa má použiť s hraničnou hodnotou
 - Možné hodnoty: eq,ge,le
- BackoffPeriod
 - Čas v sekundách , ktorý určuje ako často majú alarmy toho istého typu vznikat'
 - integer
- Message
 - správa, ktorá sa zobrazí v prípade, že nastane daná udalosť
 - deciamal
- SeverityLevel
 - stupeň závažnosti
 - Možné hodnoty: Urgent,High,Normal,Minor

Odpoveď:

200 OK

prázdne telo

Zobrazenie všetkých alarmov používateľa

Analýza

Používateľovi sú zobrazené všetky jeho alarmy. Následne ich môže prezerať prípadne vymazať.

Návrh

Funkcionalita na webovej aplikácii nie je v tejto verzii produktu implementovaná, tým pádom nie je vytvorený ani grafický návrh.

Implementácia(API)

HTTP GET metóda na url v tvare: **{{url}}/api/alarm**

Poz.: V aplikáciách (pre produkčné prostredie) treba **{{url}}** nahradiť ip adresou, alebo doménovým menom napr.: <http://team18-20.studenti.fiit.stuba.sk> (Celá url teda vyzerá napr. : <http://team18-20.studenti.fiit.stuba.sk/api/UserConnection/sendRequest>)

Popis

- Request slúži na získanie všetkých alarmov, ktoré som zdefinoval
-

V headroch, treba mat nastavene:

Content-Type : application/json

Authorization : Bearer {{token}}

Parametre volania

Jednotlivé položky, podľa ktorých filtrujem sa zadávajú ako **query parametre**.

Filtre sa v podmienke AND-ujú.

Možné filtre:

- Limit - max. počet výsledkov, ktoré api vráti (na webe veľkosť stránky)
- Offset - o koľko prvkov sa mám posunúť (na webe číslo stránky * veľkosť stránky)

Príklady requestov:

Pošlem request na url: **{{url}}/api/alarm?limit=2**

Odpoveď:

200 OK

Telo správy:

```
1 {
2   "userAlarms": [
3     {
4       "id": 1,
5       "sensor": "Temperature",
6       "field": "TempPatient",
7       "value": 36.9,
8       "operator": "ge",
9       "backoffPeriod": 3600,
10      "message": "you are on fire baby"
11    },
12    {
13      "id": 3,
14      "sensor": "Oxymeter",
15      "field": "Healthrate",
16      "value": 80.0,
17      "operator": "ge",
18      "backoffPeriod": 3600,
19      "message": "Mierne zvyšeny pulz"
20    }
21  ],
22  "count": 4
23 }
```

- userAlarms- pole samotných Alarmov
- count - celkový počet výsledkov

Vymazanie alarmu

Analýza

Používateľ má možnosť vymazať zvolený alarm.

Návrh

Funkcionalita na webovej aplikácii nie je v tejto verzii produktu implementovaná, tým pádom nie je vytvorený ani grafický návrh.

Implementácia(API)

HTTP DELETE metóda na url v tvare: **{{url}}/api/alarm/{{alarmId}}**

napr. : **{{url}}/api/alarm/2**

Poz.: V aplikáciách (pre produkčné prostredie) treba **{{url}}** nahradiť ip adresou, alebo doménovým menom napr.: <http://team18-20.studenti.fiit.stuba.sk> (Celá url teda vyzerá napr. : <http://team18-20.studenti.fiit.stuba.sk/api/UserConnection/sendRequest>)

Popis

- Request slúži na zmazanie alarmu podľa id.
- Alarm môže byť vymazaný jedine používateľom, ktorý ho vytvoril

V headroch, treba mať nastavené:

Content-Type : application/json

Authorization : Bearer {{token}}

Príklady requestov:

Chcem zmazať alarm s **Id = 2**

Pošlem request na **{{url}}/api/alarm/2**

Odpoveď:

200 OK

Vytvor udalosť

Analýza

Udalosť je inštancia alarmu. Udalosť je vytváraná počas plnenia hodnôt pomocou API, v prípade že bola prekročená hraničná hodnota na alarme. Najprv sa načítajú všetky alarmy

podľa senzora, z ktorého vkladáme nameraná údaje. Následne sa skontroluje, kedy bola vytvorená posledná udalosť z načítaných alarmov. Vyberáme len také alarmy, z ktorých nebola vytvorená udalosť v čase:

súčasnosť - BackoffPeriod

Následne kontrolujeme, či sa v našich plnených dátach nenachádzajú také hodnoty, ktoré by mohli "spustiť" alarm.

Implementácia(API)

Daná funkcionálnosť je implementovaná na API pri plnení jednotlivých údajov. Pri plnení sa takisto posielajú GPS súradnice daného zariadenia.

Funkcionálnosť je implementovaná tak, aby v budúcnosti bolo možné rozšíriť systém o nové senzory. Samotné kontrolovanie a následne vytváranie udalostí je implementované v osobitnej triede mimo triedy, ktorá realizuje plnenie údajov:

```
await eventLogger.CheckForAlarms(oxymeters, monitoring.PatientId);
```

Takýto spôsob je pomerne flexibilný a umožňuje vytváranie udalostí z alarmov v princípe kdekoľvek v kóde. Tým pádom nie sme limitovaný konkrétnou implementáciou plnenia údajov.

```
public interface IEventLogger
{
    Task CheckForAlarms<T>(List<T> data, string userId);
}
```

Metóda prijíma zoznam anonymných objektov. Typ konkrétneho senzoru sa zistí až počas behu programu.

Príručky

Inštalačná príručka

Postup inštalácie mobilnej aplikácie je nasledovný:

1. Stiahnutie inštalačného súboru z nasledujúcej [adresy](#)
2. Povolenie inštalácie z neznámych zdrojov v nastaveniach Android zariadenia [1].
3. Spustenie stiahnutého inštalačného súboru

Po týchto krokoch je aplikácia nainštalovaná a pripravená na používanie.

V prípade webovej aplikácie nie je inštalácia potrebná a aplikácia je dostupná na adrese <http://167.71.69.53/#/auth/login>

Používateľská príručka

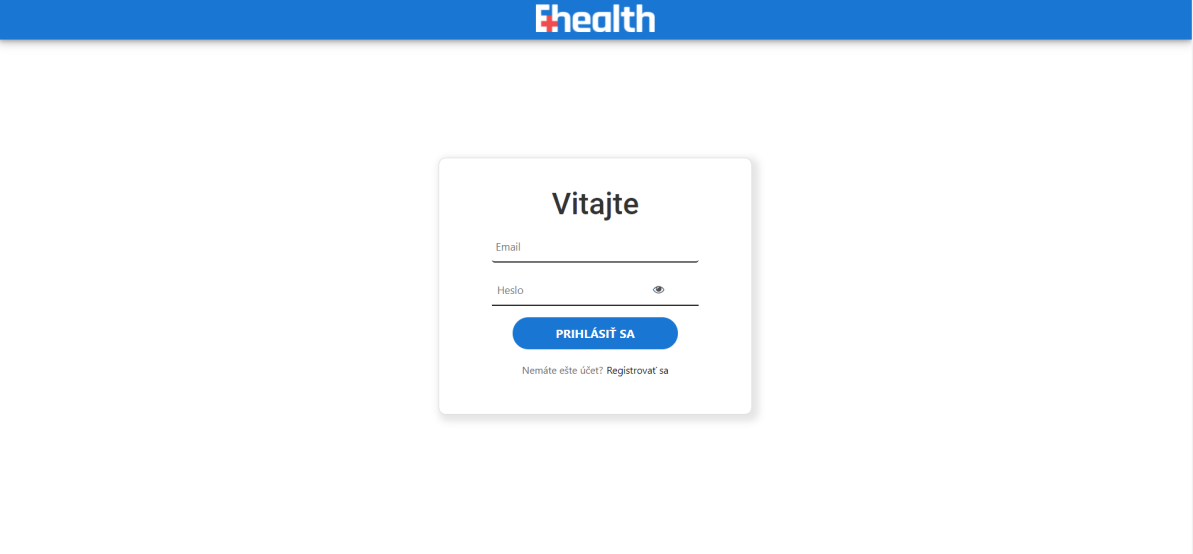
Webová aplikácia

Stručný popis aplikácie: Je určená pre lekárov a pacientov, je možné v nej vyhľadávať pacientov/lekárov, spájať sa s nimi a uskutočňovať monitorovania.

Ako sa k nej dostať: <http://167.71.69.53/>

Registrácia/login/odhlásenie

Prihlasovanie prebieha rovnako pre pacientov aj lekárov. V prihlasovacom formulári zobrazenom nižšie používateľ zadá e-mailovú adresu a heslo. Po správne zadaných prihlasovacích údajoch je používateľ presmerovaný do hlavného menu aplikácie.



Ehealth

Vitajte

Email

Heslo

PRIHLÁSIŤ SA

Nemáte ešte účet? [Registrovať sa](#)

Pri registrácii je formulár taktiež totožný, avšak v prípade lekára je tento formulár rozšírený o id lekára v komore lekárov a zameranie medicínskej praxe.

ehealth

Registrácia

Meno _____

Priezvisko _____

Email _____

Vaše telefónne číslo... _____

dd.mm.yyyy _____

Rodné číslo _____

Pohlavie: Muž Žena Olné

Lekár vyplňa miesto ordinácie a pacient trvale bydlisko!

Ulica č.domu _____

Mesto _____

ehealth

Lekár vyplňa miesto ordinácie a pacient trvale bydlisko!

Ulica č.domu _____

Mesto _____

PSČ _____

Krajina _____

Heslo _____

Zopakovať heslo _____

Zaregistrovať sa ako lekár

Súhlasím s podmienkami používania

REGISTROVAŤ SA

Máte už účet? [Prihlásiť sa](#)

Hlavné menu pozostáva z 3 hlavných častí. Prvou časťou je osobný profil, v ktorom môže používateľ aktualizovať osobné údaje, prípadne zmeniť heslo. Druhá časť sa mení v závislosti od roly používateľa. Pacientom sa zobrazuje zoznam lekárov a v prípade lekára sú zobrazení lekári. Obaja používatelia majú v tejto sekcii možnosť filtrovania záznamov a tiež môžu vytvárať spojenia s novými lekármi, prípadne pacientmi v rámci monitorovaní.

Poslednou časťou hlavného menu je zoznam monitorovaní, v ktorých je daný používateľ zapojený. Tu si môže vybrať monitorovanie ktoré chce, a pomocou grafov zobraziť namerané hodnoty sledovanej životnej funkcie.



Profil



Pacienti



Monitorovanie

Vyhľadávanie lekárov/pacientov

Pri vyhľadávaní sú formuláre totožné, mení sa len rola používateľa, ktorý je vyhľadávaný. Pacient si v tomto okne môže vyhľadať lekárov, zobrazíť ich detail, prípadne požiadať o vytvorenie spojenia pre monitorovanie.

Lekár má v tomto okne rovnaké možnosti, avšak vyhľadáva pacientov.

Zobrazenie svojho profilu/zmena hesla/zmena údajov

Profil je v prípade každého používateľa rovnaký. Používateľ môže zobrazovať a meniť svoje osobné a kontaktné informácie pomocou tohto okna. Okrem toho je v tomto okne možná aj realizácia zmeny hesla.

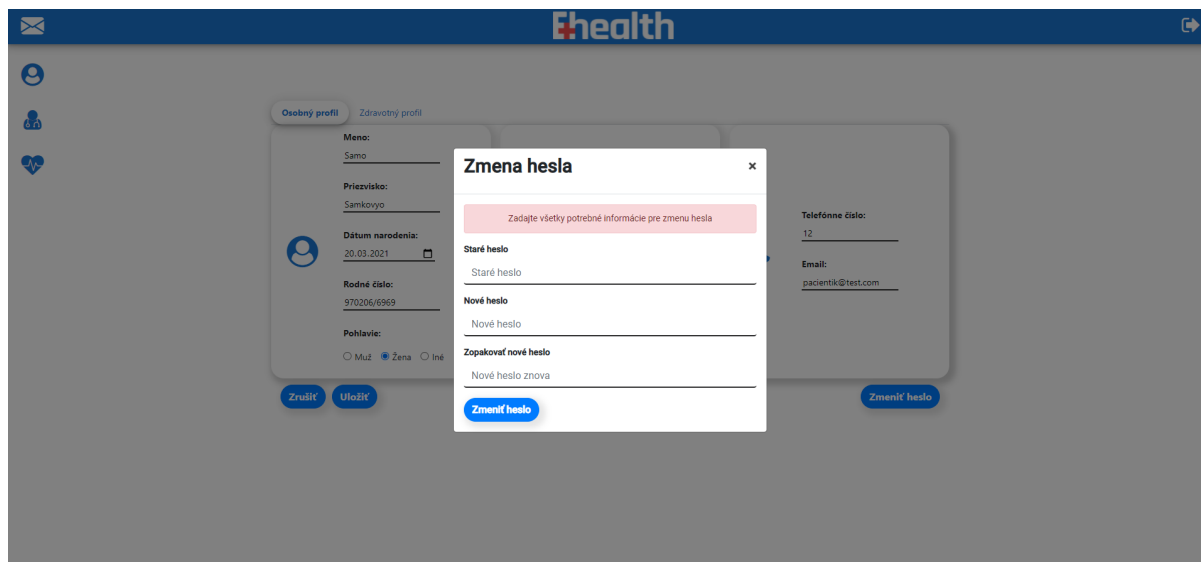
Ehealth

Osobný profil Zdravotný profil

<p>Meno: Samo _____</p> <p>Príezvisko: Samkovyjo _____</p> <p>Dátum narodenia: 20.03.2021 _____</p> <p>Rodné číslo: 970206/6969 _____</p> <p>Pohlavie: <input type="radio"/> Muž <input checked="" type="radio"/> Žena <input type="radio"/> Iné</p>	<p>Miesto bydliska Ulica: Druzstevna _____</p> <p>Mesto: Nizna2 _____</p> <p>PSČ: 69420 _____</p> <p>Krajina: Uganda _____</p>	<p>Telefónne číslo: 12 _____</p> <p>Email: pacientik@test.com _____</p>
--	--	---

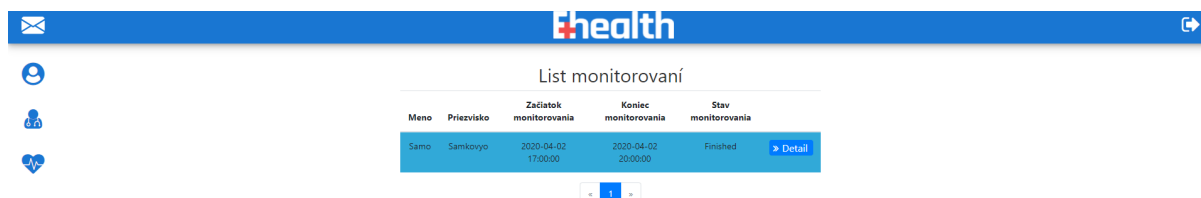
Aktualizovať údaje Zmeniť heslo

Pri zmene hesla sa zobrazí vyskakovacie okno, v ktorom používateľ zadá staré heslo a z bezpečnostných dôvodov 2 krát nové heslo. Pri úspešnom vyplnení týchto hodnôt sa heslo aktualizuje.



Zobrazenie svojich monitorovaní

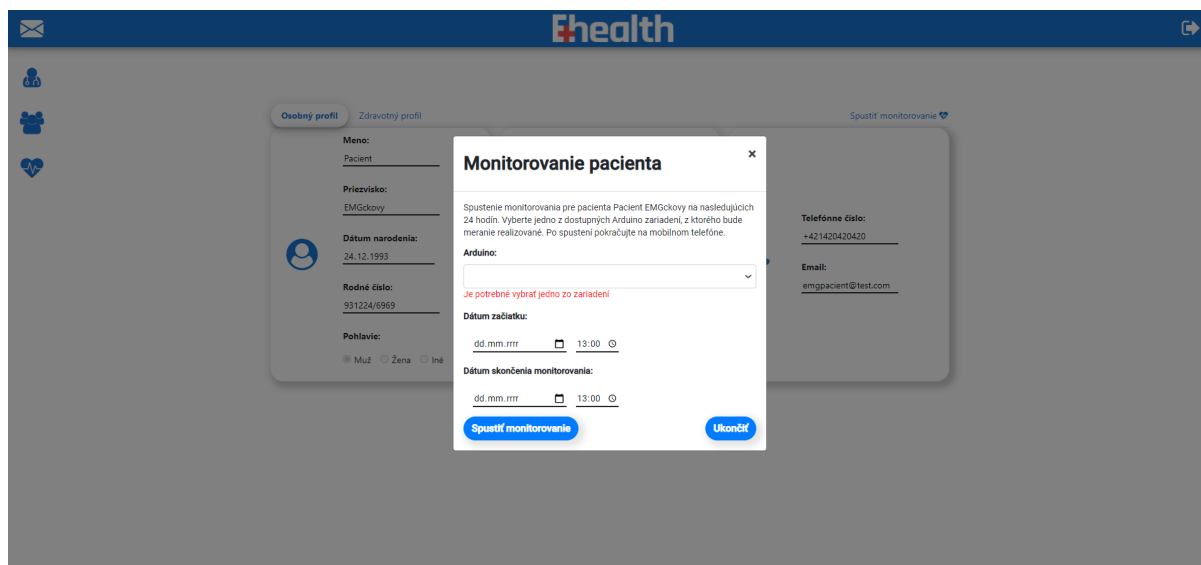
Obaja používatelia majú možnosť zobraziť si tie monitorovania, ktorých sú účastníkmi. Tu si používateľ môže rozkliknúť možnosť detailu monitorovania, ktorý sa mu zobrazí na novom okne.



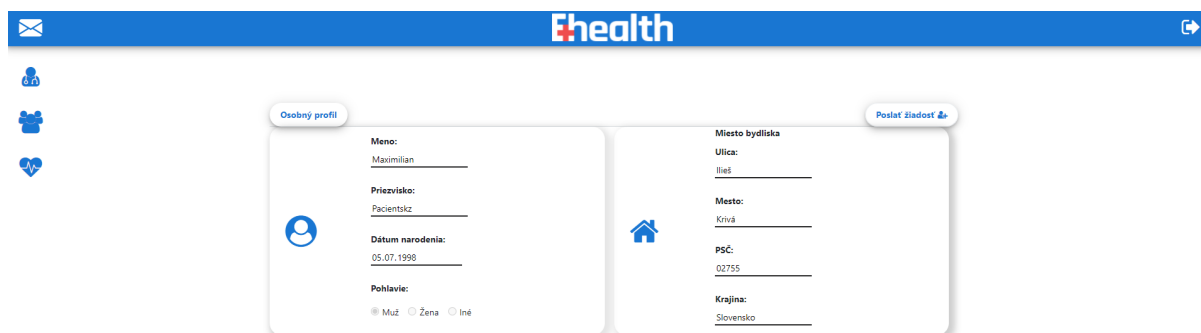
Spustiť monitorovanie

Pre spustenie monitorovania je potrebné byť prihlásený ako lekár a musí byť vytvorené spojenie s daným pacientom. Lekár v detaile pacienta klikne na tlačidlo spustiť monitorovanie, vyberie MAC adresu Arduino zariadenia a dátumy začiatku a konca

monitorovania. Následne po stlačení tlačidla spustiť monitorovanie a zadaní validných údajov sa spustí monitorovanie.



V prípade, že lekár nemá s pacientom nadviazané spojenie, môže to vykonať v detaile pacienta. Kliknutím na tlačidlo poslať žiadosť sa pacientovi odošle žiadosť o nadviazanie spojenia, a po potvrdení pacientom je možné inicializovať monitorovanie.



Zobrazenie mojich requestov/potvrdiť žiadosť

Pre zobrazenie žiadostí musí používateľ kliknúť na logo obálky v ľavom hornom rohu aplikácie. Následne sa mu zobrazí zoznam žiadostí o spojenie, ktoré mu boli odoslané. Tu pomocou dvoch tlačidiel používateľ rozhodne, či chce dané spojenie potvrdiť alebo zamietnuť.



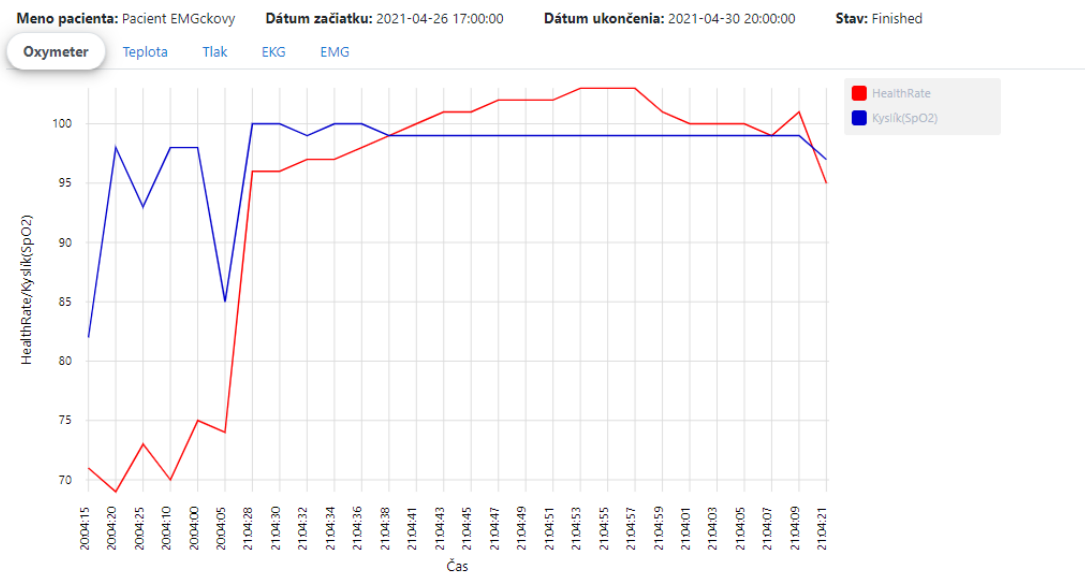
Ziadosť o spojenie

Pacient	Doktor	Stav ziadosťi pacient	Stav ziadosťi doktor
Petko Ferkovsky	Mino Lekar	Pending	Accepted

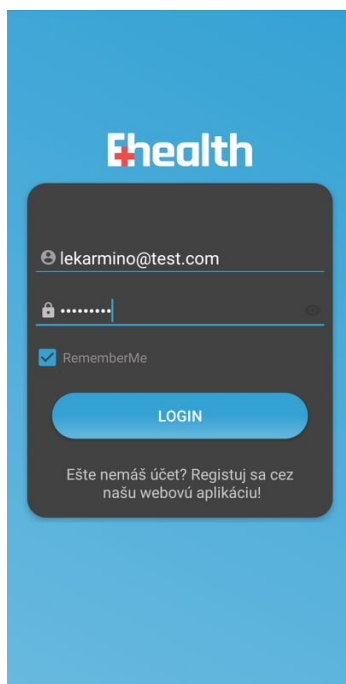
< 1 >

Monitoring detail

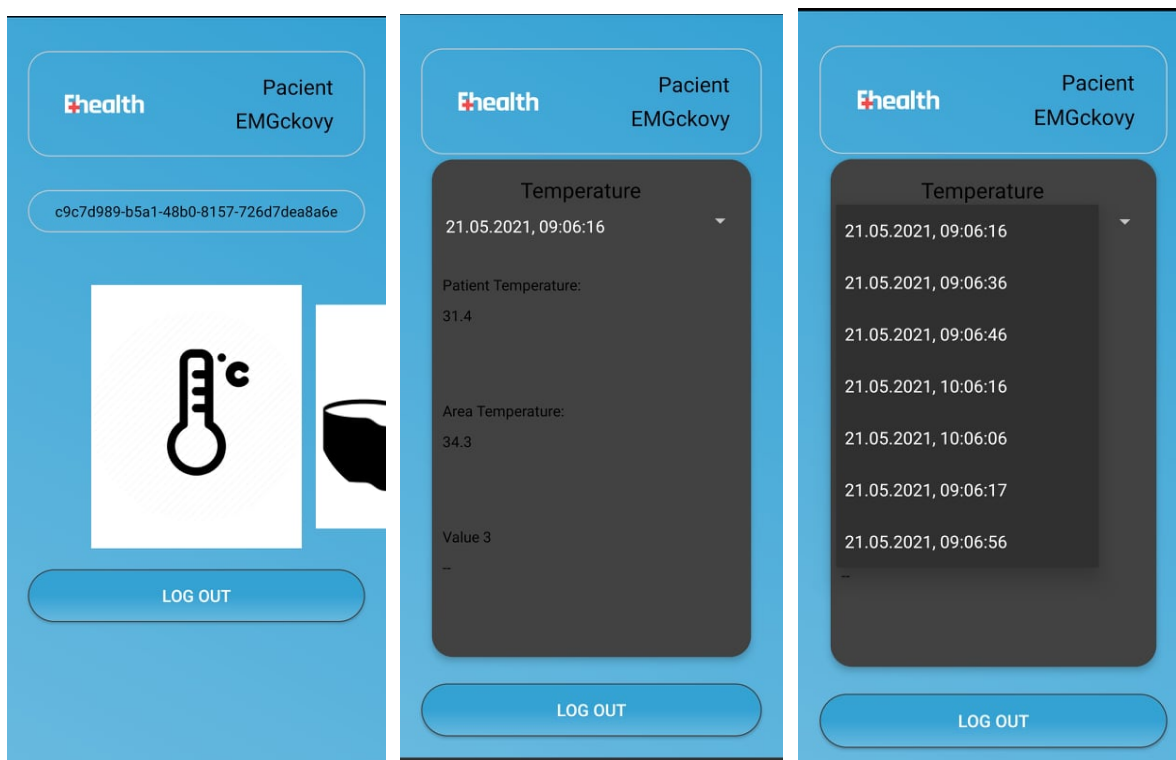
Pre zobrazenie konkrétneho monitoringu si používateľ v hlavnom menu zvolí možnosť Monitorovanie. Tu si zo zoznamu monitorovaní vyberie to, ktorého hodnoty chce prehliadať. Následne sa mu zobrazí okno, v ktorom sú graficky zobrazené jednotlivé monitorované životné funkcie. Prepínaním medzi kartami nad grafom sa menia tieto grafy podľa zvoleného parametra.



Mobilná aplikácia

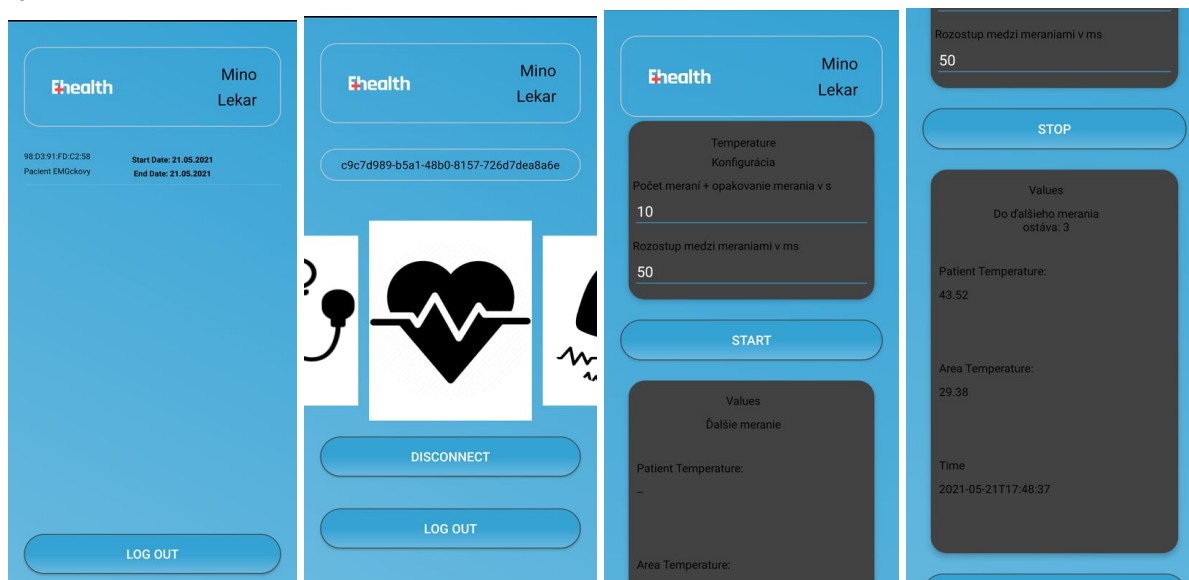


Prihlásenie je rovnaké v prípade pacienta i lekára. Je potrebné zadať mail a heslo. Pri prihlásení do role pacienta je používateľ presmerovaný do menu, kde si môže prezerať svoje namerané hodnoty. Na voľbu senzora slúži posúvateľné menu. Po zvolení želaného senzoru sú používateľovi zobrazené hodnoty. Na filtrovanie hodnôt slúži dropmenu, kde sú hodnoty rozlíšené na základe dátumu. Obrázky nižšie ilustrujú popísaný postup v poradí left ->right



Pri prihlásení do role lekára sú používateľovi zobrazené prebiehajúce monitorings (prvý obrázok zľava). Po kliknutí na prave prebiehajúce monitorovanie sa nadviaže spojenie s

meračom. Po úspešnom pripojení si lekár môže zobrazíť merač, ku ktorému je momentálne pripojený (druhý obrázok). V tejto aktivite môže lekár konfigurovať merač a spustiť monitorovanie. Skončenie merania iniciuje používateľ kliknutím na tlačidlo STOP. Ukončenie spojenia môže vykonať kliknutím na tlačidlo disconnect, ktoré sa nachádza v menu pre výber senzorov ale i v aktivite konkrétneho senzora..



Technická dokumentácia

Webová aplikácia

Webová aplikácia je riešená štandardným spôsobom. Metódy, premenné sú písané tak, že hlbšia dokumentácia nie je potrebná. Nižšie si opíšeme štruktúru webovej aplikácie, ktorá bola inšpirovaná pre dobrú škálovateľnosť pri väčších projektoch. Webová aplikácia je naprogramovaná v programovacom jazyku angular. Teda skladá sa z hlavného modulu menom app.module, a následne, každý samostatný modul ma svoj vlastný modul ktorý ho implementuje, a na grafické znázornenie (tvorbu komponentu) sa používa štandardne štvorica súborov.

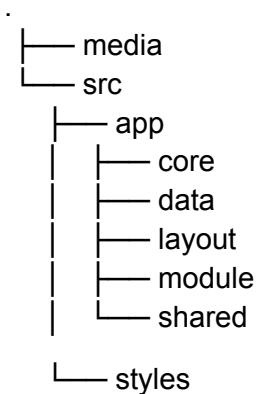
Komponent sa teda skladá zo súborov vid' obrázok nižšie (napr.):

```
# monitoring-detail.component.css
<> monitoring-detail.component.html
TS monitoring-detail.component.spec.ts
TS monitoring-detail.component.ts
```

Tento štandard je všade zachovaný, a teda je zbytočné opisovať aj ostatné komponenty.

Inšpirácia: <https://angular-folder-structure.readthedocs.io/en/latest/#> [Zdroj, ukažok]

V tomto strome vidíme základnú štruktúru našej aplikácie:



- ~/media priečinok

Priečinok /media obsahuje všetky podporujúce súbory aplikácie. Slúži na “junk” súbory, ktoré využívame.

- ~/src/app/core modul

Tento modul slúži pre uchovávanie tried, ktoré využíva app.module, ako napr. route guards, HTTP interceptors, application level services a iné.

- ~/src/app/data priečinok

-obsahuje typy (modely / entity) a služby (úložiská) údajovpoužívaných v aplikácii.

Priečinok data obsahuje ďalšie 2 priečinky a to /types a /service.

~/src/app/data

/types

/service

Adresár types obsahuje súbory definícií tried pre dátové štruktúry. Príklad dátovej štruktúry:

```

1 export class Project {
2   link: string;
3   title: string;
4   thumbnail: string;
5 }

```

Adresár služieb obsahuje služby na načítanie údajov. Príklad:

```

1 import { Injectable } from '@angular/core';
2 import { Observable } from 'rxjs';
3
4 import { Project } from '../types/project';
5 import { ApiService } from './api.service';
6
7 const routes = {
8   projects: '/projects',
9   project: (id: number) => `/projects/${id}`
10 };
11
12 @Injectable({
13   providedIn: 'root'
14 })
15 export class ProjectService {
16   constructor(
17     private apiService: ApiService) {}
18
19   getAll(): Observable<Array<Project>> {
20     return this.apiService.get(routes.projects);
21   }
22
23   getSingle(id: number): Observable<Project> {
24     return this.apiService.get(routes.project(id));
25   }
26 }

```

Pozn: Ak by sme používali viacero zdrojov na získanie dát tak dobre je to ešte obaliť do ďalších foldrov, napr:

```

1 ~/src/app/data
2   /data-source-one
3     /types
4     /service
5   /data-source-two
6     /types
7     /service
8   /data.module.ts

```

- ~/src/app/layout priečinok

Adresár layout obsahuje súčasti, ktoré fungujú ako rozloženie alebo sú súčasťou rozloženia, napríklad hlavička, navigácia, päta atď., A majú: `<router-outlet><router-outlet>`

Z konvečnosti hlavným je `app.component.html`, ktorý zavola ďalší layout a tie môžu zavolať ďalšie komponenty.

Znovupoužiteľné komponentmi ako Nav a Footer sa importujú do šablóny komponentu, stále tam kde ich potrebujeme. S týmto znižujeme redundanciu kódu. Zavoláme v html templete napr: `<app-nav></app-nav>`

Čo sa týka html súborov, tak tie sú stále pri svojom komponente. Teda html template sa nachádza pri svojom `.ts` súbore, ktorý daný template využíva.

- ~/src/app/modules priečinok

Adresár modulov obsahuje kolekciu modulov, ktoré sú navzájom nezávislé. Každý modul má svoje vlastné smerovanie, čo je smerovanie zo zdroja `loadChildren` definovaného v `AppRoutingModule`.

V tomto priečinku ma každý svoj modul vlastný priečinok kde su všetky potrebné súbory ohľadom daného modulu.

- ~/src/app/shared priečinok

Shared priečinok sa delí na `components`, `services`, `models`, podľa toho čo konkrétne zdieľame tak do takého priečinka sa hodí daný súbor.

shared.modul.ts obsahuje triedy a prostriedky, ktoré sa používajú vo viacerých dynamicky načítaných moduloch. Tu je dobré mať spravený import a export `FormsModule` alebo `ReactiveFormsModule` a pod.

- - Alebo ak budeme používať nejaké ikonky a pod. tak tu je správne miesto kde to importnúť a nie v každom module.

- ~/src/styles priečinok

Adresár sa používa na ukladanie šablón štýlov css pre aplikáciu.

- ~/src/assets/images priečinok

-tu sa nachádzajú všetky používané obrázky v aplikácii

Routing

app-routing.module.ts obsahuje hlavný routing e-health webovej aplikácie. Routing robíme pomocou volaní loadChildren(). Každý modul môže mať vlastný routing, teda v hlavnom app-routing.module sú len deti ktoré priamo nadväzujú na uvod. Následne ďalšie komponenty môžu mať svoje vlastné routing.module.ts kde je logika rovnaká.

príklad:

```

const routes: Routes = [
  {
    path: '',
    redirectTo: '/auth/login',
    pathMatch: 'full'
  },
  {
    path: 'auth',
    component: AuthLayoutComponent,
    canActivate: [IsSignedInGuardGuard],
    loadChildren: () =>
      import('@modules/auth/auth.module').then(m => m.AuthModule)
  },
  {
    path: 'menu',
    component: MenuLayoutComponent,
    canActivate: [AuthGuard],
    loadChildren: () =>
      import('@modules/menu/menu.module').then(m => m.MenuModule)
  },
  // Fallback when no prior routes is matched
  { path: '**', redirectTo: '/auth/login', pathMatch: 'full' }
];

```

Každý modul ak má deti (viacero komponentov), obsahuje vlastný routing.

Napríklad menu modul, má vlastný routing kde sa nachádza aj profil, monitorovanie a ďalšie komponenty webovej aplikácie.


```

const routes: Routes = [
  {
    path: '',
    redirectTo: '/menu/main',
    pathMatch: 'full'
  },
  {
    path: '',
    canActivate: [AuthGuard],
    children: [
      {
        path: 'main',
        component: MainMenuComponent
      },
      {
        path: 'connection-requests',
        component: ConnectionRequestsComponent
      },
      {
        path: 'profile/:id',
        component: ProfileLayoutComponent,
        loadChildren: () =>
          import('@modules/menu/page/profile/profile.module').then(m => m.ProfileModu
    ]
  }
]

```

Path Aliasing

npm install -g json

```

1 json -f tsconfig.json -I -e "this.compilerOptions.paths['@app/*'] = ['src/app/core/*']"
2 json -f tsconfig.json -I -e "this.compilerOptions.paths['@shared/*'] = ['src/app/shared/*']"
3 json -f tsconfig.json -I -e "this.compilerOptions.paths['@modules/*'] = ['src/app/modules/*']"
4 json -f tsconfig.json -I -e "this.compilerOptions.paths['@env'] = ['src/environments/enviromn

```

Vytvorili sme si aliasy ciest na zjednodušenie importov z rôznych ciest v našom programe.

Výhoda: namiesto

```
import { SharedModule } from '../..../shared/shared.module';
```

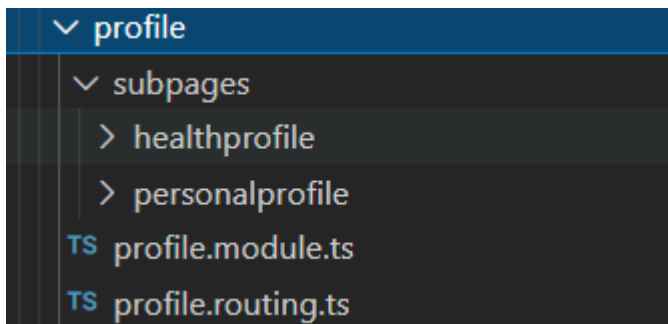
stačí napísať

```
import { SharedModule } from '@shared/shared.module';
```

Ukážka jedného modulu z čoho sa skladá a ako sú usporiadané hlavné komponenty, iné časti sú obdobne implementované.

Modul Profil

Na obrázku nižšie máme možnosť vidieť štruktúru modulu profil. Tento modul má vlastnú routing ako aj svoje stránky, ktoré su vo forme komponentov.



profile.module.ts - deklaruje/importuje vlastné dependencies v programe

Ukážka:

```
import { NgModule } from '@angular/core';
import { DisableDirective } from '@shared/directives/disable.directive';
import { SharedModule } from '@shared/shared.module';
import { FilterPipeModule } from 'ngx-filter-pipe';
import { ProfileRoutingModule } from './profile.routing';
import { HealthProfileComponent } from './subpages/healthprofile/health-profile.component';
import { ChangePasswordModalComponent } from './subpages/personalprofile/modal/change-password.component';
import { PersonalProfileComponent } from './subpages/personalprofile/personal-profile.component';
import { CommonModule } from '@angular/common';
import { StartMonitoringModalComponent } from './subpages/personalprofile/modal/start-monitoring.component';

@NgModule({
  declarations: [PersonalProfileComponent, HealthProfileComponent, ChangePasswordModalComponent, StartMonitoringModalComponent],
  imports: [ProfileRoutingModule, SharedModule, FilterPipeModule],
})
export class ProfileModule {
}
```

profile.routing.ts - ako sme skôr znázornili obsahuje vlastný routing, vďaka čomu sa dokážeme dostať na inú stránku

Ukážka:

```

import { AuthGuard } from '@app/guard/auth.guard';
import { HealthProfileComponent } from './subpages/healthprof
import { PersonalProfileComponent } from './subpages/personal
import jwt_decode from 'jwt-decode';
import { AuthService } from '@app/service/auth.service';

const routes: Routes = [
  {
    path: '..',
    redirectTo: "../personal",
    pathMatch: 'full'
  },
  {
    path: '',
    canActivate:[AuthGuard],
    children: [
      {
        path: 'personal',
        component: PersonalProfileComponent
      },
      {
        path: 'health',
        component: HealthProfileComponent
      }
    ]
  }
];

```

```

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class ProfileRoutingModule implements OnInit {

  constructor(private authenticationService: AuthService) {
    //routes[0].redirectTo = "/menu/profile/"+this.id+"/personal";
  }

  ngOnInit(): void {
    console.log("hallo");
  }
}

```