|  |  |
| --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| **NÁVRH A VYTVORENIE APLIKÁCIE PRE POMOC ZDRAVOTNÍKOM PRI PRAVIDELNEJ LEKÁRSKEJ VIZITE**  **Bakalárska práca** | |
|  | |
| **2022** | **Lýdia Tarkaničová** |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| **NÁVRH A VYTVORENIE APLIKÁCIE PRE POMOC ZDRAVOTNÍKOM PRI PRAVIDELNEJ LEKÁRSKEJ VIZITE**  **Bakalárska práca** | |
|  | |
| Študijný program: | Hospodárska informatika |
| Študijný odbor: | Informatika |
| Školiace pracovisko: | Katedra kybernetiky a umelej inteligencie (KKUI) |
| Školiteľ: | Ing. Martin Sarnovský, PhD. |
| Konzultant: | Ing. Michal Kolárik |
|  | |
| **2022 Košice** | **Lýdia Tarkaničová** |

**Abstrakt v SJ**

Abstrakt v slovenčine (referát) je povinnou súčasťou každej práce. Je výstižnou krátkou charakteristikou obsahu dokumentu. Abstrakt býva informatívny a zachováva tematické a štýlové vlastnosti práce. Nevyjadruje hodnotiace stanovisko autora. Obsahuje údaje o cieľoch práce, metódach, výsledkoch a záveroch. Text abstraktu sa píše ako jeden odstavec. Abstrakt neobsahuje odkazy na samotný text práce. Mal by mať rozsah asi 250 slov, nemal by presiahnuť jednu stranu. Pri štylizácii sa používajú celé vety, slovesá v činnom rode a tretej osobe. Používa sa odborná terminológia, menej zvyčajné termíny, skratky a symboly sa pri prvom výskyte v texte definujú.

**Kľúčové slová v SJ**

Kľúčové slovo1, kľúčové slovo2, kľúčové slovo3, kľúčové slovo4,...

Kľúčové slová sú slová, ktoré úzko definujú tému práce a pomocou nich je možné jednoduchšie vyhľadať prácu podľa jej odborného významu.

**Abstract**

Text abstraktu v svetovom jazyku je potrebný pre integráciu do medzinárodných informačných systémov (napr. The Network Digital Library of Theses and Dissertations). Ak nie je možné jazykovú verziu umiestniť na jednej strane so slovenským abstraktom, je potrebné umiestniť ju na samostatnú stranu (cudzojazyčný abstrakt nemožno deliť a uvádzať na dvoch stranách).

**Keywords**

Key word 1, Key word 2, Key word 3, Key word 4,...

**Zadanie práce**

* Túto stranu nahraďte naskenovaným zadávacím listom.
* Odporúčame skenovať na 200-300 DPI, Odtiene sivej.
* v jednej vytlačenej ZP musí byť vložený originál zadávacieho listu !
* Neskenovaný obrázok roztiahnite na celu stranu.

**Čestné vyhlásenie**

Vyhlasujem, že som celú prácu vypracoval/a samostatne s použitím uvedenej odbornej literatúry.

Košice, 15. mája 2022 ..........................................

vlastnoručný podpis

**Poďakovanie**

Na tomto mieste môže byť vyjadrenie poďakovania napr. vedúcemu práce resp. konzultantom za pripomienky a odbornú pomoc pri vypracovaní práce. Nie je zvykom ďakovať za rutinnú kontrolu, menšiu spoluprácu alebo všeobecné rady. Vyjadrenie poďakovania v prípade využitia inej práce sa uskutočňuje formou citácie na konci hlavného textu práce a odkazy na citáciu sa musia uviesť aj na zodpovedajúcich miestach v texte.

**Obsah**

[Zoznam obrázkov 9](#_Toc103456637)

[Zoznam tabuliek 10](#_Toc103456638)

[Zoznam symbolov a skratiek 11](#_Toc103456639)

[Úvod 12](#_Toc103456640)

[1. Formulácia úlohy a cieľ práce 13](#_Toc103456641)

[2. Vývoj aplikácií pre mobilné zariadenia 14](#_Toc103456642)

[2.1. Mobilné operačné systémy 14](#_Toc103456643)

[2.1.1. Android 14](#_Toc103456644)

[2.1.2. iOS 15](#_Toc103456645)

[2.2. Mobilné aplikácie a technológie 15](#_Toc103456646)

[2.2.1. Natívny prístup 15](#_Toc103456647)

[2.2.2. Webový prístup 16](#_Toc103456648)

[2.2.3. Hybridný prístup 17](#_Toc103456649)

[2.2.4. Multiplatformový prístup 17](#_Toc103456650)

[2.3. React Native 18](#_Toc103456651)

[2.4. Xamarin 19](#_Toc103456652)

[2.5. Flutter 19](#_Toc103456653)

[3. Lekárska vizita a využitie mobilných aplikácií v jej procese 21](#_Toc103456654)

[3.1. Vizita 21](#_Toc103456655)

[3.1.1. Charakteristika vizity. 21](#_Toc103456656)

[3.1.2. Formy vizít 21](#_Toc103456657)

[3.1.3. Príprava vizity 21](#_Toc103456658)

[3.1.4. Vedenie zdravotnej dokumentácie 22](#_Toc103456659)

[3.2. Existujúce aplikácie pre vizitu 22](#_Toc103456660)

[3.2.1. Mobilná aplikácia Denník krvného tlaku 23](#_Toc103456661)

[3.2.2. Mobilná aplikácia Medical records 24](#_Toc103456662)

[3.2.3. Mobilná aplikácia Medical Records Clinic app 24](#_Toc103456663)

[3.2.4. Zhodnotenie kvalít dostupných aplikácií 25](#_Toc103456664)

[4. Návrh aplikácie eVizita 26](#_Toc103456665)

[4.1. Analýza používateľských požiadaviek 26](#_Toc103456666)

[4.2. Persóny 28](#_Toc103456667)

[4.3. Plánované funkcie 29](#_Toc103456668)

[4.3.1. Zdravotná sestra 29](#_Toc103456669)

[4.3.2. Lekár 29](#_Toc103456670)

[4.3.3. Vedenie nemocnice 30](#_Toc103456671)

[4.4. Scenáre použitia 30](#_Toc103456672)

[4.5. UML diagram prípadov použitia 33](#_Toc103456673)

[4.6. Architektúra systému 34](#_Toc103456674)

[4.6.1. Firebase 34](#_Toc103456675)

[4.7. Návrh používateľského rozhrania 36](#_Toc103456676)

[5. Implementácia aplikácie eVizita 38](#_Toc103456677)

[5.1. Prototyp 38](#_Toc103456678)

[5.2. Finálna verzia 38](#_Toc103456679)

[6. Testovanie 39](#_Toc103456680)

[6.1. Priebeh testovania 39](#_Toc103456681)

[6.2. Výsledky testovania 39](#_Toc103456682)

[Záver 40](#_Toc103456683)

[Zoznam použitej literatúry 41](#_Toc103456684)

[Prílohy 43](#_Toc103456685)

Zoznam obrázkov

[Obr. 1 Podiel mobilných OS na celosvetovom trhu 2018-2022 [9] 14](#_Toc103083517)

[Obr. 2 Persóna Lekár 26](#_Toc103083518)

[Obr. 3 Persóna Zdravotná sestra 26](#_Toc103083519)

[Obr. 4 Persóna Vedenie nemocnice 27](#_Toc103083520)

[Obr. 5 UML diagram prípadov použitia 31](#_Toc103083521)

[Obr. 6 UML diagram nasadenia 32](#_Toc103083522)

[Obr. 7 Obrázok grafického CD média 40](#_Toc103083523)

Zoznam tabuliek

[Tab. 1 Používateľské požiadavky lekárov 25](#_Toc103083524)

[Tab. 2 Používateľské požiadavky zdravotných sestier 25](#_Toc103083525)

[Tab. 3 Používateľské požiadavky vedenia nemocnice 25](#_Toc103083526)

[Tab. 4 Scenár použitia - Pridať meranie pacientovi 29](#_Toc103083527)

[Tab. 5 Scenár použitia - Zaznamenať vizitu 29](#_Toc103083528)

[Tab. 6 Scenár použitia - Vyhľadať a pridať lekára na oddelenie 30](#_Toc103083529)

Zoznam symbolov a skratiek

OS Operating System

iOS iPhone Operating System

SDK Software Development Kit

API Application Programming Interface

DOM Document Object Model

UI User Interface

NoSQL Not only Structured Query Language

HTML HyperText Markup Language

CSS Cascading Style Sheets

PHP Hypertext Preprocessor

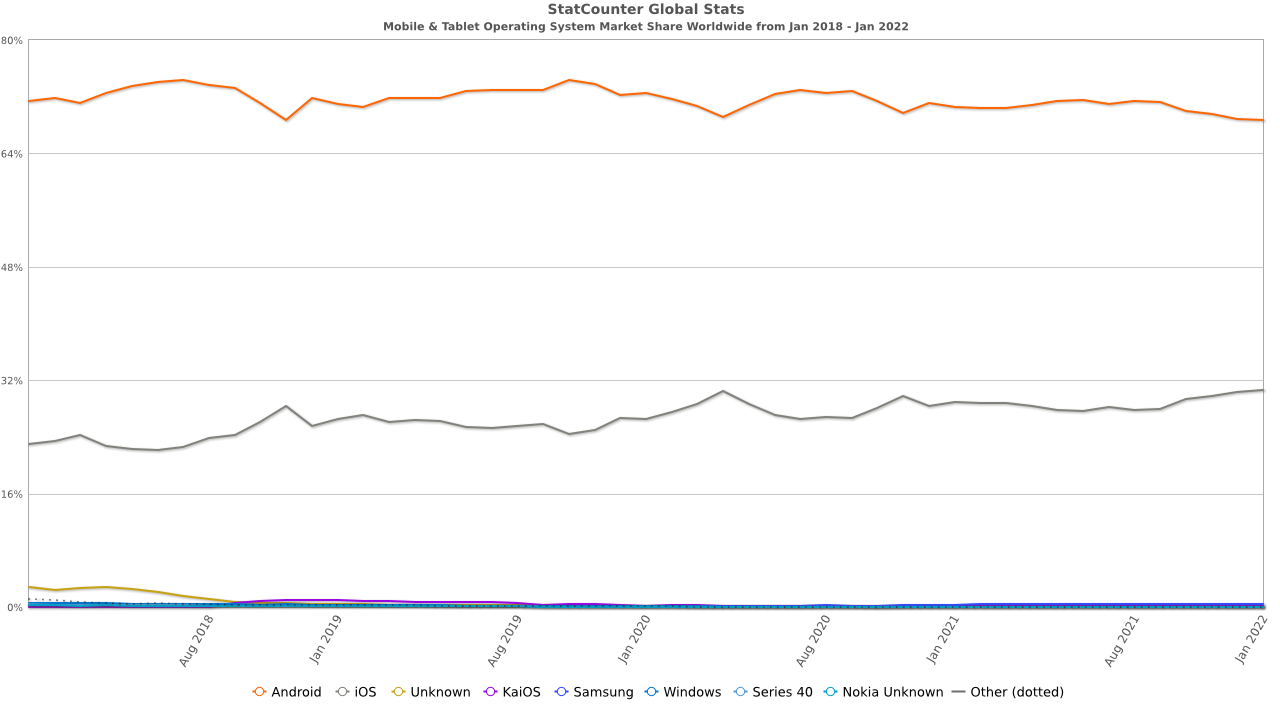
GPS Global Positioning System

CSV Comma-separated values

Úvod

1. Formulácia úlohy a cieľ práce
2. Vývoj aplikácií pre mobilné zariadenia
   1. Mobilné operačné systémy

Operačný systém (OS) je softvérové rozhranie, ktoré je zodpovedné za správu a prevádzku hardvérových jednotiek a pomáha používateľovi tieto jednotky používať. Najznámejšie mobilné OS na trhu sú Android, iOS, Windows Phone OS a Symbian [1]. V súčasnosti aj na Slovensku teda prevládajú dve najväčšie mobilné platformy a to Android a iOS.



Obr. Podiel mobilných OS na celosvetovom trhu 2018-2022 [2]

* + 1. Android

Operačný systém Android je základný stavebný prvok mobilného telefónu vyvinutý primárne pre smartfóny a tablety spoločnosťou Android Inc., ktorý bol v roku 2005 zakúpený spoločnosťou Google. Momentálne ide o najrozširenejší operačný systém pre mobilné zariadenia, ktorý je založený na jadre Linux 2.6. Android možno najlepšie opísať ako kompletný balík riešení, ktorý zahŕňa operačný systém, komponenty a aplikácie [3].

Výhodou vývoja Android aplikácií je určite flexibilita, pretože vo všeobecnosti Android predstavuje oveľa menej obmedzujúce prostredie oproti iným OS. Taktiež Google ponúka rozsiahlu základňu materálov pre rôzne úrovne vývojárov. V súčasnosti spoločnosť Google už nie je taká zhovievavá pri publikácií aplikácie. Proces kontroly môže trvať aj týždeň, no pokiaľ nie sú porušené obsahové pravdilá spoločnosti, aplikácia dostane schválenie. Medzi nevýhody môžeme zaradiť náročné zabezpečenie kvality, pretože veľká časť používateľov sa drží starších verzií OS, čo si vyžaduje zložitejšie testovanie, následkom čoho sa zvyšujú náklady pri vývoji [4].

* + 1. iOS

Mobilný operačný systém iOS je operačný systém pre iPhone, iPad a ďalšie mobilné zariadenia Apple. Apple iOS, založený na Mac OS operačnom systéme, na ktorom beží rad stolných a prenosných počítačov Mac od spoločnosti Apple, je navrhnutý pre jednoduché a bezproblémové prepojenie medzi radom produktov Apple. Prvá verzia systému iOS bola vydaná v júni 2007, keď na trhu debutoval iPhone. iOS je operačný systém odvodený od Unixu, ktorý poháňa všetky mobilné zariadenia Apple. Názov iOS nebol oficiálne použitý pre softvér až do roku 2008, keď spoločnosť Apple vydala súpravu na vývoj softvéru pre iPhone (SDK), ktorá umožňuje všetkým tvorcom aplikácií vytvárať aplikácie pre túto platformu [5].

Vývoj mobilných aplikácií pre iOS je veľmi žiadaný, pretože tieto aplikácie vždy fungujú mimoriadne dobre. Táto platforma je rýchla, spoľahlivá a ľahko použiteľná. Taktiež Apple poskytuje vývojárom podrobné pokyny na vytvorenie používateľského rozhrania pre ich aplikácie. Veľkou nevýhodou pre vývojárov natívnych aplikácií určite sú prísne požiadavky na vývojové prostredie. Pre vývoj je potrebné použiť softvér, ako napríklad Xcode, ktorý beží iba na Macu. Problém môže nastať aj pri vydávaní aplikácie. Aplikácia môže byť odmietnutá aj keď neporušuje žiadne pravidlá, stačí ak nie je dostatočne relevantná alebo užitočná [4].

* 1. Mobilné aplikácie a technológie

V súčasnosti hýbu mobilné aplikácie svetom. Aj naše najmenšie aspekty života sú nimi ovplyvnené. V dnešnom svete existujú mobilné aplikácie prakticky na čokoľvek. Pri takej veľkej konkurencii sa pri vývoji kladie oveľa väčší dôraz na kvalitu, precíznosť, funkčnosť a ďalšie iné parametre, ktoré presviedčajú potencionálnych používateľov k ich použitiu. Je preto veľmi dôležité byť čo najviac informovaný o trendoch vo vývoji mobilných aplikácií. Vytváranie mobilných aplikácií je možné vykonať použitím viacerých metód. Potreby a ciele každého projektu sú iné, a preto sa podľa toho vždy mení aj prístup. Na výber je jedna z týchto možností: natívny, hybridný, webový a multiplatformový prístup.

* + 1. Natívny prístup

Natívne aplikácie sa vyznačujú tým, že sú vytvorené špeciálne pre konkrétny operačný systém, ako sú Android či iOS, s využitím programovacích jazykov špecifických pre platformu. Pri procese vývoja takýchto aplikácií alebo softvéru pre Android sa využíva Java alebo Kotlin, v prípade iOS ide o Objective-C alebo Swift. Vývoj natívnych mobilných aplikácií je ideálny, ak chceme poskytnúť čo najlepší používateľský zážitok z hľadiska vzhľadu a celkového dojmu z aplikácie. Pri vývoji získavajú vývojári prístup ku kompletnej sade možností a funkcií zariadenia, keďže tvorba takýchto aplikácií umožňuje využívať primarné hardvérové prvky smartfónov, sko sú senzory priblíženia, fotoaparát, mirofón a GPS. Medzi výhody natívnych aplikácií určite patrí vysoká rýchlosť, keďže neprichádzajú so zložitým kódom ako hybridné a miltiplatformové aplikácie, kde sa kód musí prispôsobovať viacerým platformám. V tomto prípade sa väčšina prvkov aplikácie zobrazuje pomerne rýchlo, pretože sa vopred načítajú. Jednou z významných výhod je, že fungujú bezchybne aj bez pripojenia k internetu. Zaisťuje to používateľom väčšie pohodlie a prístup ku všetkým funkciám aj v režime lietadlo či offline prostredí. Správa dvoch rôznych kódových báz je určite náročnejšia v porovnaní s jednou kódovou bázou, a tak môžeme povedať, že natívne aplikácie majú minimálny výskyt chýb. Optimalizácia pri vývoji na konkrétnu platformu zaisťuje vysoký výkon. Okrem toho sú tieto aplikácie zostavené pomocou základných programovacích jazykov a rozhraní API, vďaka čomu sú eketívnejšie a intuitívnejšie. Medzi nevýhody vývoja natívnych aplikácií určite môžeme zaradiť vysoké náklady na údržbu, ktoré môžu byť dokonca vyššie ako samotný vývoj, ktorý je už aj tak sám o sebe dosť nákladný, keďže je potrebné nájsť vývojárov, ktorí dokážu vyvíjať natívne aplikácie naraz na viacerých platformách, čo vyžaduje veľa času, úsilia, zdrojov, ale aj už spomínaných nákladov. Tvorbu natívnej aplikácie môžeme zvažovať, ak máme dostatok času a financií, alebo potrebujeme kódovať len pre jednu platformu. Vysoká rýchlosť a výkon sa určite zíjdu pri tvorbe hier a animácií. Rozhodujúcim by mohla byť aj potreba pridať funkcie špecifické pre zariadenie, ako GPS či fotoaparát [6].

* + 1. Webový prístup

Vývoj webových aplikácií je vytváranie aplikačných programov, ktoré bežia na webovom serveri a interagujú s prehliadačom. Webové aplikácie sú responzívne verzie webových stránok, ktoré prispôsobujú svoje používateľské rozhranie akémukoľvek zariadeniu. Taktiež môžu fungovať na každom mobilnom zariadení alebo operačnom systéme, pretože sa dodávajú pomocou mobilného prehliadača, vďaka čomu sú náklady na vývoj podstatne nižšie. Prevažne sú navrhnuté prostredníctvom jazykov HTML, CSS, JavaScript pre frontend a pre backend je to napríklad PHP alebo Ruby. Správaním pripomínajú natívne aplikácie, no nejde o samostatné aplikácie, ktoré by bolo nutné stiahnúť a nainštalovať do zariadenia. Pri možnosti inštalovať webovú aplikáciu do zariadenia si najčastejšie aplikácia len uloží URL adresu do záložiek webovej stránky. Avšak môže ísť o progresívnu webovú aplikáciu, ktorú je možné pridať ako ikonu do zariadenia. Nevýhodou všetkých webových aplikácií je určite nutnosť internetového pripojenia. Aj keď majú režim offline, tak na zálohovnie údajov v zariadení, zobrazenie nových údajov alebo na obnovenie obrazovky bude zariadenie stále potrebovať internetové pripojenie. Takisto sú úplne závislé od prehliadača používaného v zariadení. V porovnaní s mobilnými aplikáciami, webové aplikácie ponúkajú najrýchlejší čas uvedenia na trh, keďže vývoj trvá oveľa kratšie a môže osloviť používateľov iOS aj Androidu súčasne [7].

* + 1. Hybridný prístup

Vývoj hybridných aplikácií je kombináciou natívnych a webových riešení, kde vývojári vkladajú kód napísaný v jazykoch pre vývoj webu ako HTML, CSS a JavaScript do natívnej aplikácie pomocou doplnkov ako Apache Cordova či Ionic's Capacitor. Po stiahnutí aplikácie a lokálnej inštalácií sa dokáže pripojiť k všetkým funkciám, ktoré mobilná platforma poskytuje. Pri vývoji hybridných aplikácií sa kód píše iba raz. Takýto kód je možné použiť na viacerých platformách, čo zabezpečuje rýchly vývoj a uvedenie aplikácie na trh v pomerne krátkom čase a s minimálnymi nákladmi. Keďže sú tieto aplikácie založené na webových technológiách, sú jednoduchšie na udržiavanie a údržbu. Keďže hybridné aplikácie majú nasadený jeden kód a existujú funkcie, ktoré sú použiteľné len pre konkrétny operačný systém alebo nefungujú dokonale na iných systémoch, môže dôjsť k nezrovnalostiam, kedy je potrebné hľadať vhodnejšiu alternatívu. S tým je úzko spojená potreba testovať aplikáciu na rôznych zariadeniach kvôli kompatibilite na platformách Android a iOS, a teda niečo, čo sme považovali za výhodu, je aj nevýhodou. Medzi nevýhodu oproti natívnym aplikáciám môžeme považovať aj nutnosť internetového pripojenia. Len zriedka hybridné aplikácie podporujú offline verziu. Hybridné aplikácie integrujú výhody webových a natívnych aplikácií, a tak prinášajú to najlepšie pre používateľa na platformách Android a iOS. Vývoj hybridnej aplikácie je na zváženie, ak je na tvorbu nedostatok času a finančných prostriedkov a chceme využívať natívne funkcie zariadenia. Určite ide o jasnú voľbu pri tvorbe webovej a mobilnej aplikácie súčasne [6].

* + 1. Multiplatformový prístup

Najprv je dôležité podotknúť, že hybridné a multiplatformové aplikácie nie sú to isté. Multiplatformový prístup umožňuje vývojárom napísať kód raz a použiť ho na rôzne platformy. Je to najlepší prístup pre lacné vlastné aplikácie s bezpečnými, stabilnými, ľahko udržiavateľnými a opakovateľnými funkciami. Existuje mnoho rámcov na vývoj takýchto aplikácií, ktoré pomáhajú dosiahnúť pocit a používateľský zážitok podobný natívnej aplikácií. Najčastejšie využívanými sú React Native, Xamarin a Flutter. Medzi výhody multiplatformových aplikácií určite patrí opätovná použiteľnosť kódu, takže nie je potrebné písať osobitne kód pre Android a iOS. S tým súvisí aj výhoda rýchleho a bezproblémového vývoja, či nízkych nákladov na vývoj takejto aplikácie. Keďže multiplatformové aplikácie majú len jednu kódovú základňu, testovanie, nasadzovanie či aktualizácie aplikácie sú oveľa jednoduchšie a ponúkajú vyššiu presnosť a kvalitu oproti natívnemu prístupu, kde je takáto údržba zložitejšia. Medzi nevýhody môžeme v niektorých prípadoch zaradiť výkon aplikácie. Ak by sme porovnali dve aplikácie, jednu natívnu a jednu multiplatformovú s rovnakými funkciami, tak je pravdepodobné, že natívna bude rýchlejšia. Multiplatformové aplikácie taktiež nie sú schopné naplno využiť natívne funkcie. Je to zapríčinené najmä rôznymi rozloženiami obrazovky a rôznymi platformami. Tento spôsob vývoja sa najviac oplatí ak potrebujeme zacieliť na používateľov aplikácií pre Android aj iOS a máme obmedzené zdroje, čas aj rozpočet. Aplikácia, ktorú chceme vytvárať by nemala byť príliš zložitá a nepotrebuje funkcie, ktoré sa medzi rozdielnymi platformami líšia [6].

* 1. React Native

React je knižnica JavaScriptu na vytváranie rýchlych a interaktívnych používateľských rozhraní. Prevádzkuje ho Meta (predtým Facebook) a komunita jednotlivých vývojárov a spoločností. Aj keď je React skôr knižnica než jazyk, je momentálne najpoužívanejší pri vývoji webových aplikácií. React Native je rámec založený na Reacte, ktorý sa zameriava na vykresľovanie mobilných aplikácií na všetkých zariadeniach. Ako aj sami tvorcovia Reactu hovoria: „Naučte sa raz, píšte kdekoľvek“, čo znamená, že je možné vytvárať zaujímavé projekty na nových platformách bez toho, aby sme museli investovať čas a energiu do učenia sa úplne nového jazyka. Myšlienka písania mobilných aplikácií v JavaScripte pôsobí trochu zvláštne, ale tím vývojárov spoločnosti Facebook prišiel na riešenie, ktoré obohatilo celú komunitu programátorov. Facebook vydal prvú verziu React Native v marci 2015, a v pribehu pár rokov sa stal dôležitým hráčom v natívnom vývoji aplikácii mobilných zariadení. Keďže ide o voľnedostupný mobilný rámec, svoje výhody získava aj od vývojárskej komunity [8][9].

React Native sa snaží uľahčiť a skrátiť vývojárovi prácu, v čom spočíva aj jeho popularita. Klasický vývoj aplikácií zvyčajne vyžaduje zručnosti a znalosti niekoľkých vývojárov na zostavenie, posúdenie a testovanie softvéru. Táto knižnica umožňuje zvládanie viacerých častí vývoja menšiemu vývojárskemu tímu a minimalizuje čas na realizáciu nápadu aplikácie. S týmito zdrojmi je vývoj aplikácie nákladovo efektívny. React Native umožňuje jednému vývojárskemu tímu vytvoriť mobilnú aplikáciu, ktorá funguje na všetkých platformách. Nie je nutné písať a udržiavať rôzne kódy oddelene. Ponúka možnosť zdieľať a znova používať kód. Navyše pomáha programátorom vytvárať natívne aplikácie pre obe najpoužívanejšie platformy s bežnými znalosťami webových technológií, ako je JavaScript. Inými slovami, aplikácia je rýchla, plynulá a flexibilná a je ekvivalentná akejkoľvek natívnej aplikácii vytvorenej pomocou tradičných technológií. Aj tieto prednosti mu pomáhajú k zisku veľkej časti trhu. Výhody React Native spočívajú aj v jeho základnej vlastnosti použitia virtuálneho vykresľovania objektového modelu dokumentu (DOM). Virtuálny objekt dokumentu je odľahčená verzia skutočnej DOM štruktúry a funguje ako vrstva medzi popisom vývojára, ako by veci mali vyzerať a prácou vykonanou na ich vykreslení. Na vykreslenie interaktívnych používateľských rozhraní v prehliadači musia vývojári upraviť objektový model dokumentu. Tento krok si vyžaduje nadmerné zapisovanie DOM, a tým výrazne vplýva na výkon zariadenia, na ktorom vykresľovanie prebieha. React vypočítava potrebné zmeny pomocou virtuálnej DOM v pamäti, a tým znižuje počet priamych zmien v reálnej DOM štruktúre v prehliadači. Virtuálna DOM má určité výkonnostné výhody, ale jeho skutočný potenciál spočíva v sile jeho abstrakcie. Umiestnenie čistej abstrakcie medzi kód vývojára a skutočné vykresľovanie otvára veľa zaujímavých možností. V roku 2015 na konferencii React, tím Facebook prišiel s možnosťou vykresľovať aplikáciu inde ako v objektovom modely dokumentu, a predstavil tak knižnicu React Native. Pri React Native, natívne vykresľovanie znamená, že React vykresľuje pomocou natívnych rozhraní API na vytváranie používateľských rozhraní. To všetko je možné vďaka rozhraniu knižnice, ktoré zabezpečuje vykresľovanie UI elementov v danom hostiteľskom zariadení. Komponenty Reactu odovzdávajú kód v podobe značkovacieho jazyka funkcii vykresľovania, ktorá dáva pokyn Reactu, ako ich vykresliť. S Reactom pre web sa tento kód odovzdáva priamo do DOM prehliadača. Pre React Native je tento kód preložený tak, aby vyhovoval hostiteľskej platforme, takže napríklad element „View“ sa môže stať „UIView“ špecifický pre iOS. Prístup založený na komponentoch vývojárov implicitne nabáda k napísaniu čistého, modulárneho kódu, ktorý sa dá škálovať tak, aby vyhovoval zložitým aplikáciám. Riziko, ktoré so sebou používanie React Native prináša súvisí s úrovňou jeho vyspelosti. Ešte stále môžeme React Native považovať za pomerne mladú technológiu, a preto môže táto knižnica obsahovať svoj podiel chýb a neoptimalizovaných implementácií. Toto riziko do istej miery kompenzuje jej komunita [10][11].

* 1. Xamarin

Xamarin je platforma, ktorá umožňuje vytvárať a odosielať aplikácie pre Android a iOS pomocou .NET. Vývojárska platforma .NET je zložená z nástrojov, programovacích jazykov a knižníc na vytváranie rôznych typov aplikácií. Xamarin rozširuje túto vývojársku platformu o nástroje a knižnice špeciálne na vytváranie aplikácií Android, iOS, a Windows. Bol vytvorený v roku 2011 vývojármi Mono, no od roku 2016 ho vlastní Microsoft. Xamarin predstavuje abstraktnú vrstvu, ktorá riadi komunikáciu zdieľaného kódu s kódom základnej platformy. Je tiež súčasťou prosperujúceho mobilného ekosystému obsahujúceho všetko od cloudových zdrojov špecifických pre mobilné zariadenia od spoločnosti Microsoft a nástrojov DevOps. Ide o voľnedostupný rámec na vytváranie multiplatformových aplikácií, ktoré vyzerajú ako natívne na akomkoľvek zariadení. Takisto je súčasťou obrovskej vývojárskej komunity s otvoreným zdrojovým kódom. Je to spôsob, ako používať rovnaký jazyk a technológiu v systémoch iOS a Android, čo umožňuje opätovné použitie veľkého množstva kódu a knižníc tretích strán na dvoch veľmi odlišných mobilných platformách [12].

Xamarin.Forms je samostatný produkt určený na vytváranie prototypov alebo mobilných aplikácií. Ide o súpravu nástrojov medziplatformových tried používateľského rozhrania postavenú na základnejších triedach používateľského rozhrania špecifických pre platformu a to Xamarin.Android a Xamarin.iOS. Jednou z najväčších výhod Xamarin.Forms je, že dáva možnosť vyvíjať natívne mobilné aplikácie pre niekoľko platforiem súčasne. Tento nástroj tiež zlepšuje rýchlosť vývoja aplikácií, čo ocenia najmä biznisovo orientované projekty. Medzi plusy môžeme zaradiť aj opätovné využitie kódu až na 98% tým, že abstrahuje funkcie špecifické pre zariadenie, ako sú mapy alebo fotoaparát. Takáto extra abstrakcia však môže znížiť výkon. Ak teda chceme dosiahnúť čo najvyšší výkon pre vlastné používateľské rozhranie, je možné využívať Xamarin.iOS a Xamarin.Android samostatne. Vytváranie mobilných aplikácií s ich využitím síce ponúka natívny vzhľad a prostredie, no stále je nutné napísať vrstvu kódu špecifickú pre platformu pomocou natívnych rozhraní API každej platformy. V takomto prípade je pri tvorbe užitočné mať aspoň základné znalosti natívnych technológií ako napríklad Java a Kotlin pre Android a Objective-C alebo Swift pre iOS. Alternatívou môže byť knižnica Xamarin.Essentials, ktorá poskytuje viacero multiplatformových API, pretože je možné ju použiť na prístup k natívnym funkciám. To sa však nevzťahuje na Xamarin.Forms [13].

V závislosti od typu a zložitosti sú aplikácie Xamarin zvyčajne väčšie ako natívne, preto potrebujú dodatočnú optimalizáciu, ako napríklad prepojenie, aby sa odstránil nepoužitý kód zo zahrnutých knižníc, a tým sa zachovala priemerná veľkosť súboru aplikácie. Xamarin je bezplatná platforma, no len pre malé firmy. Komerčný vývoj aplikácie v tejto platforme si vyžaduje licenciu a zakúpenie Microsoft Visual Studio IDE. Na druhú stranu Microsoft a Google, dve z najväčších technologických firiem, poskytujú vynikajúcu dokumentáciu na prácu s ich nástrojmi [14].

* 1. Flutter

Flutter je mobilný rámec, ktorý slúži na vývoj softvéru používateľského rozhrania. Je podporovaný a vytvorený spoločnosťou Google na vytváranie moderných, natívnych a reaktívnych aplikácií pre iOS a Android. Google tiež pracuje na doplnení možnosti tvorby web aplikácií. Ide o voľnedostupný projekt nachádzajúci sa na GitHube, takže svojimi príspevkami ho doplňuje a vylepšuje aj vývojárska komunita. Prvýkrát bol Flutter prezentovaný na Dart vývojárskom samite v roku 2015, vtedy však ešte pod názvom Sky. V tom čase bol prezentovaný ako Dart v mobile. Dart je programovací jazyk taktiež od spoločnosti Google, ktorý sa prevažne využíva na tvorbu mobilných a webových aplikácií. Používateľské rozhranie a obchodná logika Flutteru boli napísané v Dart. Na samite bola odprezentovaná ukážka aplikácie, ktorá bola schopná konzistentne vykresľovať rýchlosťou 60 snímkov za sekundu, pri výkonnejších zariadeniach aj 120, a tak splnila cieľ systému a to byť rýchla a plynulejšia. Pod názvom Flutter bol tento rámec prvýkrat vydaný so svojou stabilnou verziou v roku 2018 [15].

Flutter používa Dart na vytvorenie používateľského rozhrnia, čím zaniká potreba využívať ďalšie samostatné jazyky ako napríklad značkovací jazyk. Aplikácie sú zostavené z jednej kódovej základne a sú kompilované do natívneho kódu. Do používateľského rozhrania je možné pridať plynulé animácie a rozpoznávanie gest. Flutter využíva Skia, jeho vlastné renderovacie jadro na vykresľovanie, vďaka ktorému môže byť spustený prakticky na akejkoľvek platforme. Svojou deklaratívnou povahou vytvára používateľské rozhrania tak, aby odrážali skutočný stav aplikácie. Ak sa stav alebo údaje zmenia, prekreslí sa používateľské rozhranie a vytvorí sa nová inštancia widgetu. Widgety sú stavebnými kameňmi aplikácie Flutter a každý widget je nemennou časťou používateľského rozhrania. Sú to konfiguračné časti používateľského rozhrania a spoločne vytvárajú akýsi strom widgetov, kde strom je sám o sebe považovaný za widget. Flutter používa widgety ako konfiguráciu na zostaveie každého prvku na obrazovke. Za strom teda môžeme považovať aplikáciu, ktorá je len kombináciou viacerých jednoduchších widgetov. Keďže je Flutter doposiaľ relatívne nový, ešte nie je vytvorené veľké množstvo knižníc pre vlastnú potrebu. Rovnako aj niektoré pluginy sú stále v procese vývoja, avšak kompenzáciou môže byť možnosť nájdenia a implementovania funkcionality, ktorú potrebujeme. Dart, v porovnaní s JavaScriptom, využíva relatívne malá komunita, takže existujú len obmedzené zdroje, pomocou ktorých je možné naučiť sa tento jazyk [15][16].

1. Lekárska vizita a využitie mobilných aplikácií v jej procese
   1. Vizita
      1. Charakteristika vizity.

Dôležitou súčasťou ošetrovateľskej starostlivosti o pacienta je vizita. Vizita je pravidelná kontrola hospitalizovaných pacientov, účelom ktorej je dopodrobna zistiť aktuálne informácie o stave pacienta, kontrolujú sa výsledky predchádzajúcich vyšetrení, určujú sa nasledujúce postupy pri liečení pacienta, rozhoduje sa o preložení, či prepustení pacienta. Ide o pravidelnú návštevu, ktorá prebieha v určitých časových intervaloch. Vizity sa zúčastňuje lekár, zdravotná sestra alebo iní pracovnci zdravotníckeho tímu. Môže byť realizovaná rôznymi spôsobmi. Najčastejšie prebieha priamo v izbe pacienta pri lôžku chorého, no v praxi sa taktiež môžeme stretnúť aj s realizáciou vizity vo vyšetrovni či ambulancii praktického lekára, vždy to ale zavisí od samotnej organizácie a preferencií daného oddelenia. Z pohľadu pacienta ide o najvážnejšiu udalosť v jeho celodennom programe, ak sa práve nejedná o jeho operačný deň. Počas priebehu vizity sú pacienti informovaní o svojich výsledkoch, nadchádzajúcich vyšetreniach či o prognóze do budúcna [17].

* + 1. Formy vizít

Vizity môžeme rozdeliť do dvoch základných skupín a to lekárske a sesterské vizity. Sesterské vizity sa líšia od lekárskej hlavne svojím zameraním a účelom. Lekársku vizitu poznáme individuálnu, skupinovú, malú a veľkú. Pri sesterskej môžeme hovoriť o individuálnej, malej a veľkej. Malá lekárska vizita je zvyčajne dvakrát denne. Obvykle vykonáva túto vizitu službukonajúci lekár, ktorý sa informuje o anamnéze a aktuálnom stave. Veľká lekárska vizita, alebo inak primárska, sa vykonáva dva až trikrát týždenne. Vizitu vedie primár oddelenia a zúčastňujú sa jej všetci členovia zdravotníckeho tímu. Cieľom je vzájomná informovanosť všetkých členov týmu o stave a liečebných postupoch pacienta. Ak lekár doplní túto vizitu o individuálnu konzultáciu v ambulancii, tak hovoríme o individuálnej vizite. Skupinová vizita je zvláštna forma vizity, ktorá prebieha v liečebných alebo psychiatrických oddeleniach. Sesterské vizity sú určené na zistenie aktuálnych potrieb pacientov. Sestra v rámci individuálnej vizity dáva inštrukcie pacientovi alebo ošetruje a prevezuje rany [17].

* + 1. Príprava vizity

Sestra má pripraviť vizitu tak, aby prebiehala v príjemnom a pokojnom prostredí. Je potrebné skontrolovať a upraviť izbu a upozorniť pacientov na nadchádzajúcu vizitu, aby sa zdržiavali vo svojich izbách. Povinnosťou sestier, ktoré sprevádzajú lekára pri vizite, je vedieť o pacientovi čo najviac informácií. Tieto informácie zisťujú ešte pred vizitou a taktiež ešte pred ňou ich čo najpodrobnejšie a najpresnejšie podávajú lekárovi. Ide napríklad o aktuálny stav pacienta, tlak krvi, pulz, príjem a výdaj tekutín. Podľa anamnézy sa zisťujú aj iné informácie ako napríklad saturácia krvi, teplota či glykémia krvi. Nakoniec je ešte potrebné pripraviť zdravotné dokumentácie a základné pomôcky pre vizitu [17].

* + 1. Vedenie zdravotnej dokumentácie

Súčasťou poskytovania zdravotnej starostlivosti je vedenie zdravotnej dokumentácie. Sestra vedie zdravotnú dokumentáciu hospitalizovaného pacienta. Zdravotná dokumentácia je súhrn osobných údajov osoby, ktorej je poskytovaná zdravotná starostlivosť. Ide teda o údaje ako meno, priezvisko, rodné číslo, dátum narodenia, adresa, telefónne číslo, aktuálny zdravotný stav pacienta a zdravotné údaje, ktoré boli zistené od pacienta či samotným poskytovaním zdravotnej starostlivosti, vrátane podaných liekov a vykonaných lekárskych vyšetrení a výsledkoch z týchto vyšetrení. Táto dokumentácia sa vedie v písomnej alebo v elektronickej forme [18]. Väčšina zdravotníckych zariadení v  súčasnosti využíva elektronickú formu stále viac a snažia sa o redukciu vedenia písomnej formy zdravotnej dokumentácie. Elektronická forma je vytváraná sestrou, ktorá vkladá údaje o pacientovi do databázy v počítači. Dôležité je, aby novovytvorený digitálny záznam bol zhodný s predchádzajúcim písomným [19].

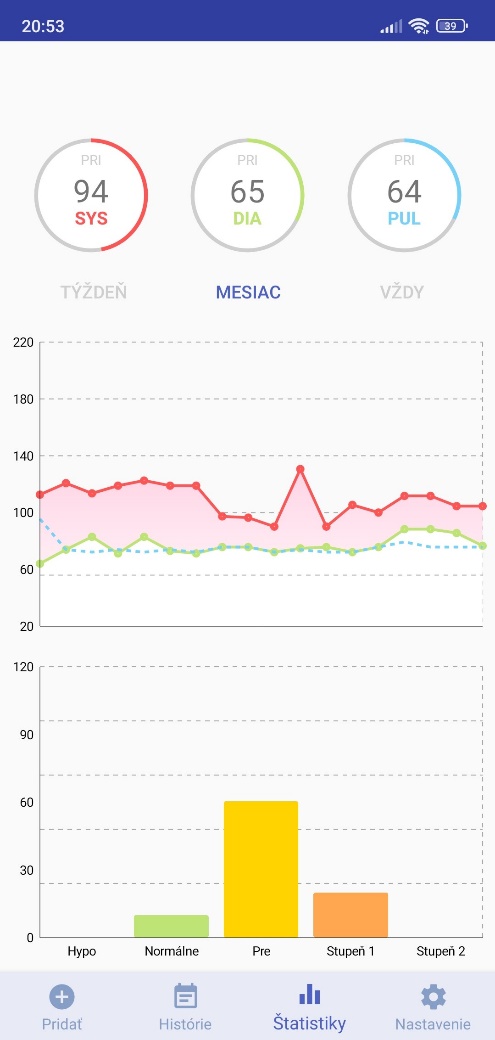
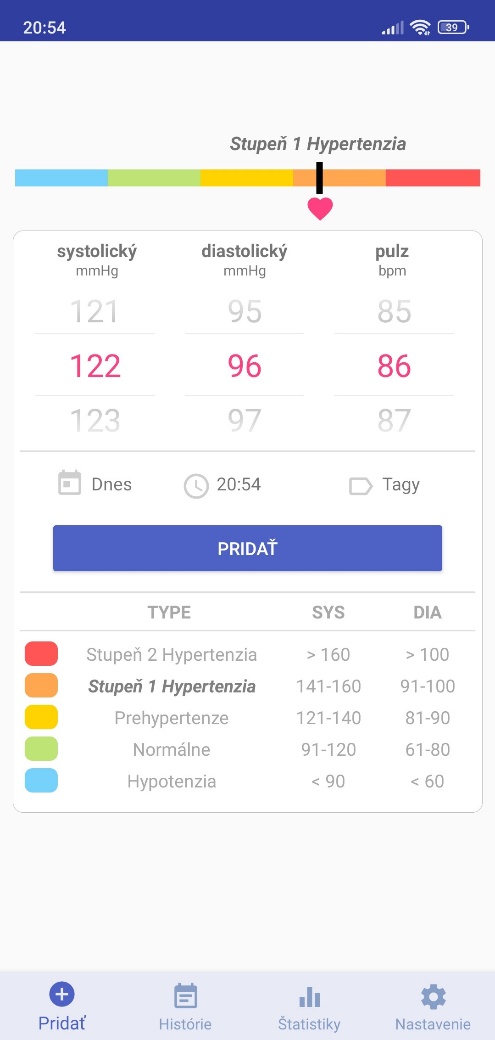
* 1. Existujúce aplikácie pre vizitu

V súčasnosti Google ponúka veľa rôznych aplikácií pre systémovú platformu Android, ktoré sú lekárskeho charakteru. Medzi nimi nájdeme množstvo takých, ktoré slúžia na zaznamenávanie meraní. Väčšina z nich je zameraná na jedno konkrétne meranie, ako napríklad len na meranie krvného tlaku, teploty či glukózy v krvi. Takéto aplikácie vám pomôžu okrem zaznamenávania hodnôt do denníka sledovovať históriu týchto meraní, zobrazovať grafy a diagramy. Lepšie z nich majú viacero zaujímavých funkcií ako exportovanie údajov ako CSV súbor a upozornenie pomocou notifikácie na vynechané meranie. Niektoré z týchto aplikácií po zaznamenaní nameranej hodnoty, vyhodnotia zadanú hodnotu a upozornia používateľa na prípadné odchýlky od normálu. Potom tu je skupina aplikácií, ktoré sú vytvorené priamo výrobcami meracích prístrojov, ktoré sľubujú ľahkú komunikáciu aplikácie s ich zariadením, no okrem toho však majú rovnaké funkcie ako vyššie spomenuté aplikácie. Funkcionálne však nie vždy spĺňajú to čo sľubujú, čo môžeme vidieť aj na veľkom množstve negatívnych hodnotení a recenzií. Aplikáciu, v ktorej by bolo možné zaznamenávať viacero rôznych meraní, pre väčšie množstvo ľudí sme našli len jednu. Zväčša aplikácie podporovali zápis meraní jednej osobe, nanajvyš dvom, preto by takéto aplikácie nemohli byť použité na pomoc pri vizite. Posledné negatívum, ktoré sme pri prieskume spozorovali bolo, že len málo z nich bolo v slovenskom jazyku. Nemôžem očakávať, že všetci ovadajú anglický jazyk, alebo že by ho chceli využívať na dennej báze.

Náš prieskum existujúcich aplikácii sme vykonávali v internetovom obchode Google Play, ktorý slúži ako oficiálna aplikačná databáza pre operačný systém Android. Ako ukážku uvádzame aplikácie, ktoré nás pri vyhľadávaní najviac zaujali.

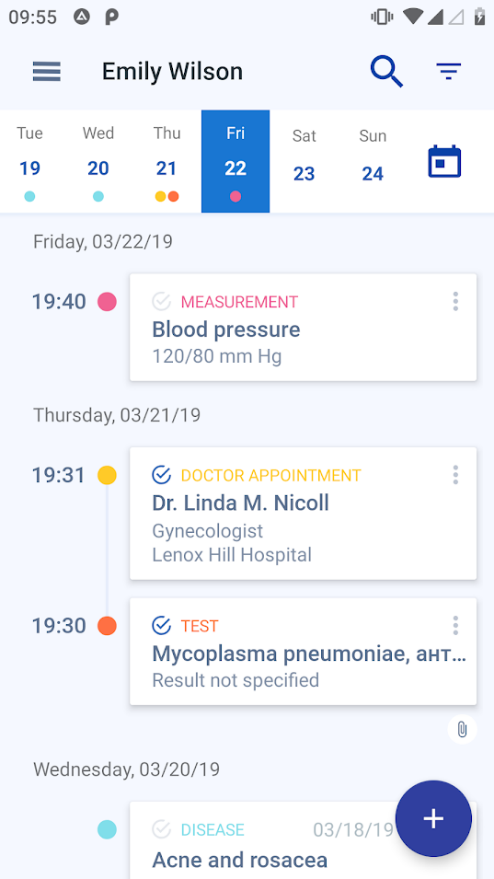
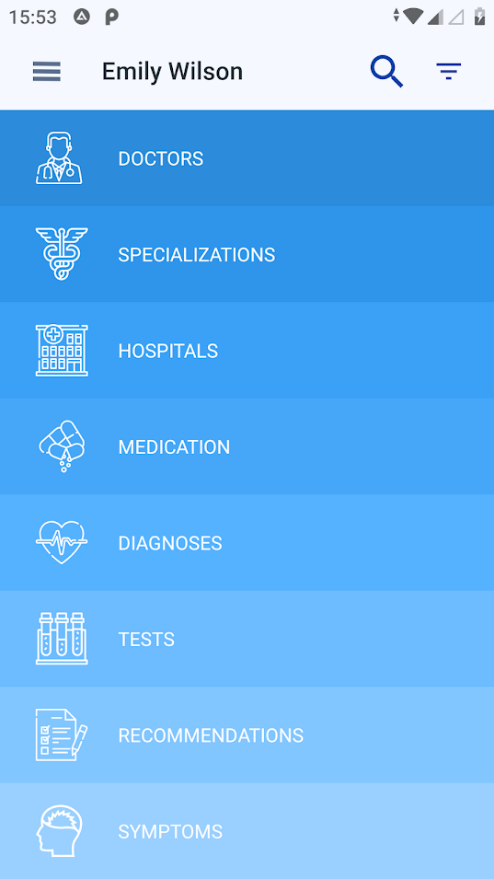
* + 1. Mobilná aplikácia Denník krvného tlaku

Mobilná aplikácia Denník krvného tlaku [20] je aplikácia, ktorej funkcionality najmenej spĺňali používateľské požiadavky. Typ tejto aplikácie sa najčastejšie vyskytoval pri vyhľadávaní medicínskych výrazov či podobných kľúčových slov. Ide o jednoduchú a efektívnu aplikáciu pre zaznamenávanie jedného typu merania pre jednu osobu. V tomto prípade ide o krvný tlak. Ako už bolo spomenuté, takýchto aplikácií bolo mnoho, či už v slovenskom alebo anglickom jazyku. Túto sme na ukážku vybrali práve preto, že bola preložená do Slovenského jazyka. Aplikácia už vopred upozorňuje, že krvný tlak nemeria, len je možné do nej zapísať namerané hodnoty. Obsahuje štatistiky s históriou meraní aj vo forme grafu. V tomto prípade nie je možné prepojenie prístroja s aplikáciou. I keď aplikácia sľubuje upozornenie na hypertenziu či iné odklony od normálu, neberie do úvahy hodnoty pulzu, a teda aj ak by bol zadaný vysoký krvný pulz, aplikácia to nevyhodnotí ako riziko.



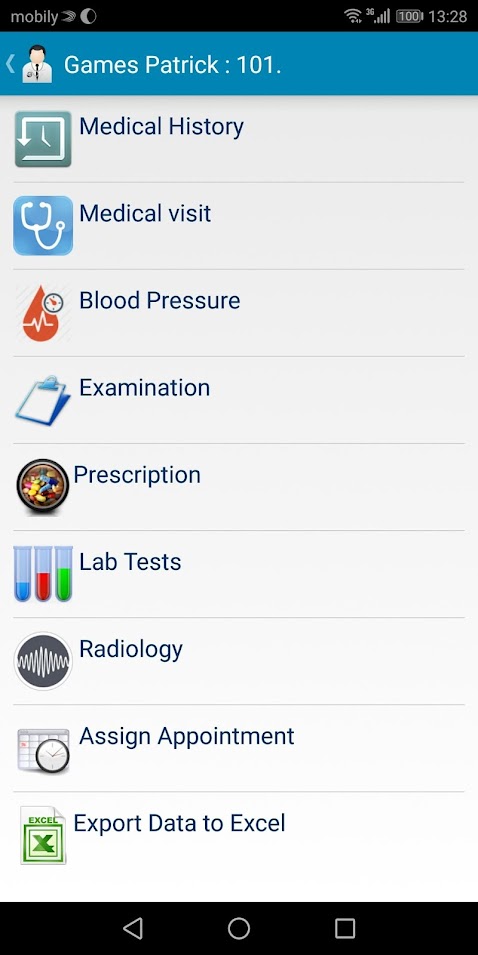
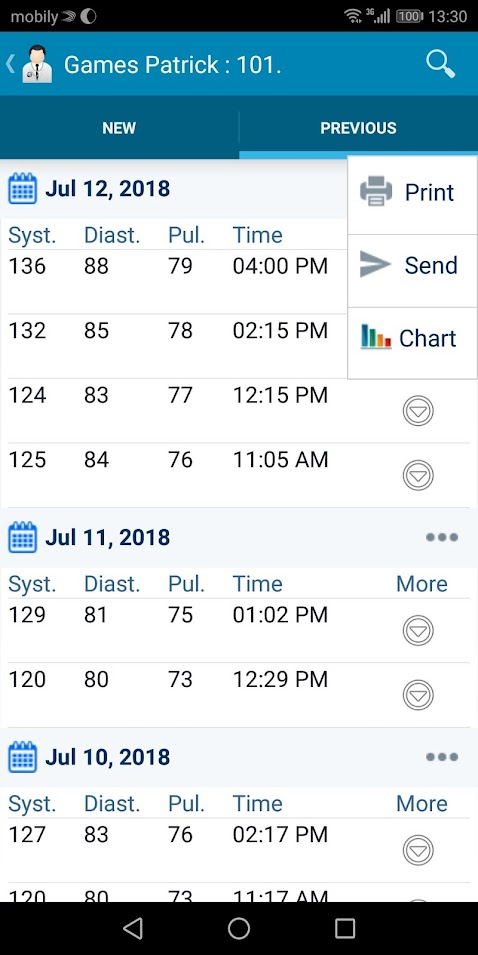
* + 1. Mobilná aplikácia Medical records

Medical records [21] je mobilná aplikácia, ktorej funkcionality sa viac priblížili považovanému využitiu. Medzi výhody oproti predchádzajúcej aplikácií patrí určite zaznamenávanie rozličných meraní, historiu návštev lekárov, výsledky testov alebo diagnózy chorôb. Celkovo vieme viesť osobné informácie o našom zdraví na jednom mieste. V prípade, ak má používateľ pravidelne užívať lieky, aplikácia mu môže túto povinnosť pripomínať. Taktiež je možné sledovať kalendár udalostí, ktorý je veľmi pekne dizajnovo spracovaný. Čo sa týka využitia, nebolo by ju možné použiť na zaznamenávanie meraní pacientov, a teda ani rozdeliť skupiny ľudí na oddelenia, preto nespĺňa naše zadanie. Je možné okrem svojho profilu, viesť profil aj rodinnému príslušníkovi, čo môže napríklad matka s dieťaťom považovať za užitočné. Za nevýhodu môžeme považovať aj to, že nepodporuje slovenský jazyk.

* + 1. Mobilná aplikácia Medical Records Clinic app

Mobilná aplikácia Medical Records Clinik app [22] je aplikácia, ktorá najviac naplnila používateľské požiadavky. Je v nej možné pridávať viacero pacientov, vytvoriť im profil s detailnými informáciami, vykonávať zápis pre rôzne merania, či vytvárať lekárov a následne ich priradiť pri tvorbe záznamu. Funkciu pre sledovanie histórie záznamov a vidieť ich aj vo forme grafu berieme už ako samozrejmosť. V aplikácii je možné zapísať aj o akú nemocnicu ide, no z nášho pohľadu to nemalo veľký význam, pretože nebolo možné tejto nemocnici detailne priradiť oddelenia či pacientov. Ak sme napríklad vytvorili aj dve nemocnice, pri tvorbe pacienta nebolo možné zapísať, ktorej nemocnici by sme ho chceli priradiť. Taktiež nebolo možné delenie účtu kvôli právomociam, a teda z tohto pohľadu pôsobila aplikácia trochu chaoticky. Podľa nášho názoru by nemala mať zdravotná sestra možnosť prepísať informácie o nemocnici. Taktiež sme nenašli spôsob, ako by bolo možné vidieť pacientov na oddelení na dvoch zariadeniach pri viacerých účtoch. Za jej slabú stránku, i keď nejde o jej funkčnosť, považujeme taktiež dizajn. Posledná prekážka, ktorú by sme chceli spomenúť je, že aplikácia je v anglickom jazyku.



* + 1. Zhodnotenie kvalít dostupných aplikácií

Z aplikácií, ktoré Google Play ponúka sme nenašli žiadnu, ktorá by spĺňala všetky používateľské požiadavky. Pre nás najdôležitejšie však bolo nájsť takú, v ktorej by sa dalo zaznamenávať viacero meraní pre viacero ľudí. Túto požiadavku najlepšie spĺňala Medical Records Clinik app, no je potrebné podotknúť, že bola v anglickom jazyku a mala najhoršiu prehľadnosť a dizajn. V tejto aplikácií ale vidíme najväčší potenciál pre jej využitie v medicínskom zariadení.

1. Návrh aplikácie eVizita
   1. Analýza používateľských požiadaviek

Pred samotnou tvorbou aplikácie je potrebné identifikovať používateľske požiadavky pre aplikáciu. Snažili sme sa určiť, čo má aplikácia presne spĺňať, pre koho bude určená a aké by mala mať funkcie. Pre zistenie týchto požiadaviek sme využili dotazník, ktorý sme primárne zamerali na zdravotné sestry, ktoré by nám mohli bližšie objasniť priebeh a fungovanie vizity. Pýtali sme sa na základné informácie a to aké údaje by mali byť zaznamenané v profile pacienta alebo ako pravidelne sa uskutočňuje vizita a aké merania sa vykonávajú. Z odpovedí sme identifikovali tieto požiadavky:

Pacienta na oddelenie prijíma zdravotná sestra. V niektorých prípadoch túto úlohu zastáva lekár, keď sa chce oboznámiť so stavom pacienta. Pri príjmaní mu vytvára kartu, do ktorej je mu počas celej hospitalizácie zaznamenávaný zdravotný stav. Na začiatku sa taktiež zaznamenávajú základné údaje o pacientovi, jeho meno, vek, diagnóza, zdravotná poisťovňa, alergie, momentálne užívané lieky, kontaktná osoba a pod. V rámci aplikácie bude mať možnosť vytvárať pacienta zdravotná sestra. Okrem toho by mala mať možnosť ho editovať a v prípade prepustenia pacienta ho aj ostrániť.

Zdravotné sestry pripravujú pacientov na vizitu. Pred samotnou vizitou zisťujú aktuálny stav pacienta, zisťuje sa príjem a výdaj tekutín a vykonajú sa pravidelné merania. Tieto merania sa u každého pacienta môžu líšiť. Najčastejšie však ide o meranie krvného tlaku a kontroluje sa pulz pacienta. U diabetikov je kontrolovaná glykémia krvi a v určitých prípadoch sa kontroluje teplota či saturácia krvi. Preto všetky tieto merania by mali byť obsiahnuté v aplikácií.

Počas samotného priebehu vizity sestra už len asistuje lekárovi, prípadne mu poskytuje dôležité informácie o zdravotnom stave pacienta, ktoré si pred vizitou všimla. Lekár kontroluje namerané hodnoty, ktoré mu sestra pripravila, kontroluje výsledky z vyšetrení a on sám kontroluje aktuálny stav pacienta. Následne stanovuje nasledujúci postup pri liečbe pacienta a rozhoduje o jeho prípadnom prepustení. V dôsledku toho by rola lekára v aplikácií mala byť schopná zobrazovať všetky namerané hodnoty u pacienta, vidieť jeho profil a zaznamenávať vizitu, teda zapísať aktuálny stav pacienta, nasledujúce vyšetrenia či dávkovanie liekov. Ako už bolo spomenuté, elektronická forma záznamov je vytváraná sestrou, preto by aj ona mala mať možnosť spätne zapísať vizitu, ak ju tým lekár poverí.

Poslednou rolou by malo byť vedenie nemocnice. Táto kontrolná rola by mala spravovať samotnú nemocnicu v aplikácií. Mala by mať možnosť pridávať a odstraňovať oddelenia, priradiť týmto oddeleniam registrovaných lekárov a zdravotné sestry a sledovať obsadennosť oddelení.

Všetky používateľské požiadavky sú ďalej zhrnuté pomocou stanovenia priorít MoSCoW metódy.

|  |  |
| --- | --- |
| Lekári | |
| * Rýchly a prehľadný systém * Možnosť zapísať údaje ku každej vizite * Vidieť základné osobné aj zdravotné údaje o pacientovi | M |
| * Možnosť napísať poznámku k pacientovi * Zobraziť zoznam nameraných hodnôt u pacienta | S |
| * Vidieť grafické znázornenie nameraných hodnôt | C |
| * Odporúčania na vyšetrenia na iných oddeleniach | W |

Tab. Používateľské požiadavky lekárov

|  |  |
| --- | --- |
| Zdravotné sestry | |
| * Rýchly a prehľadný systém * Prístup k pacientom iba na ich oddelení * Možnosť vytvoriť nový profil hospitalizovaného pacienta * Zapísať údaje z meracých prístrojov do aplikácie | M |
| * Vidieť základné údaje o pacientovi * Zobraziť zoznam nameraných hodnôt u pacienta * Aplikácia v slovenskom jazyku | S |
| * Možnosť napísať poznámku k pacientovi * Pripomenutie nezadaných meraní | C |

Tab. Používateľské požiadavky zdravotných sestier

|  |  |
| --- | --- |
| Vedenie nemocnice | |
| * Vidieť obsadenosť oddelenia * Dostatočné zabezpečenie údajov o pacientoch | M |
| * Lekari priradení k oddeleniam * Zdravotné sestry priradené k oddeleniam | S |

Tab. Používateľské požiadavky vedenia nemocnice

• (M) – Must have: dôležité požiadavky, ktoré sú základom fungovania výsledného informačného systému.

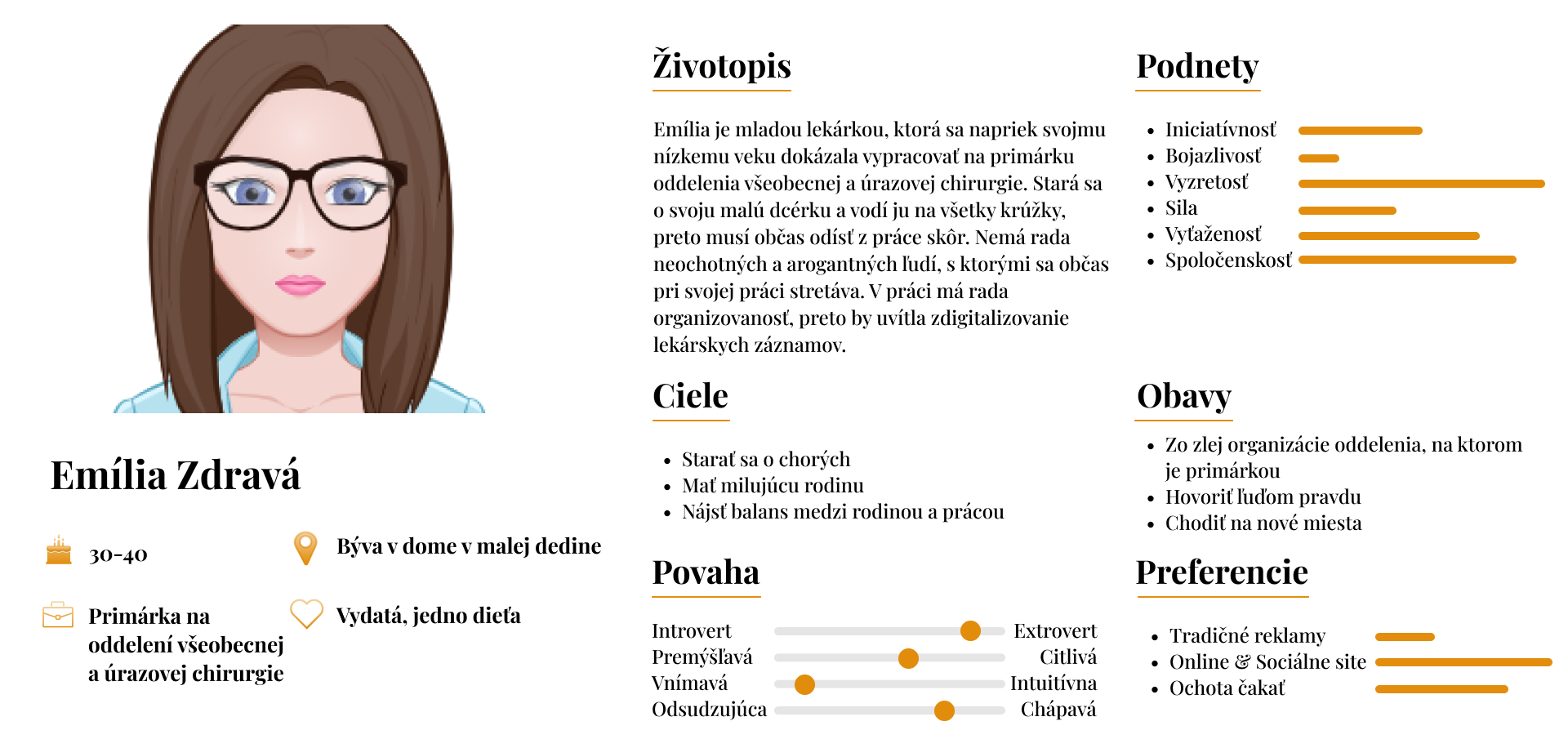
• (S) – Should have: dôležité požiadavky, ale nie nevyhnutné pre fungovanie systému, je možné ich zanedbať.

• (C) – Could have: požiadavky, ktoré sú zaujímavé pre používateľov ale nemajú požadovanú dôležitosť. Špecifikujú a implementujú sa len v takom prípade, keď je voľný čas a dostupné potrebné zdroje.

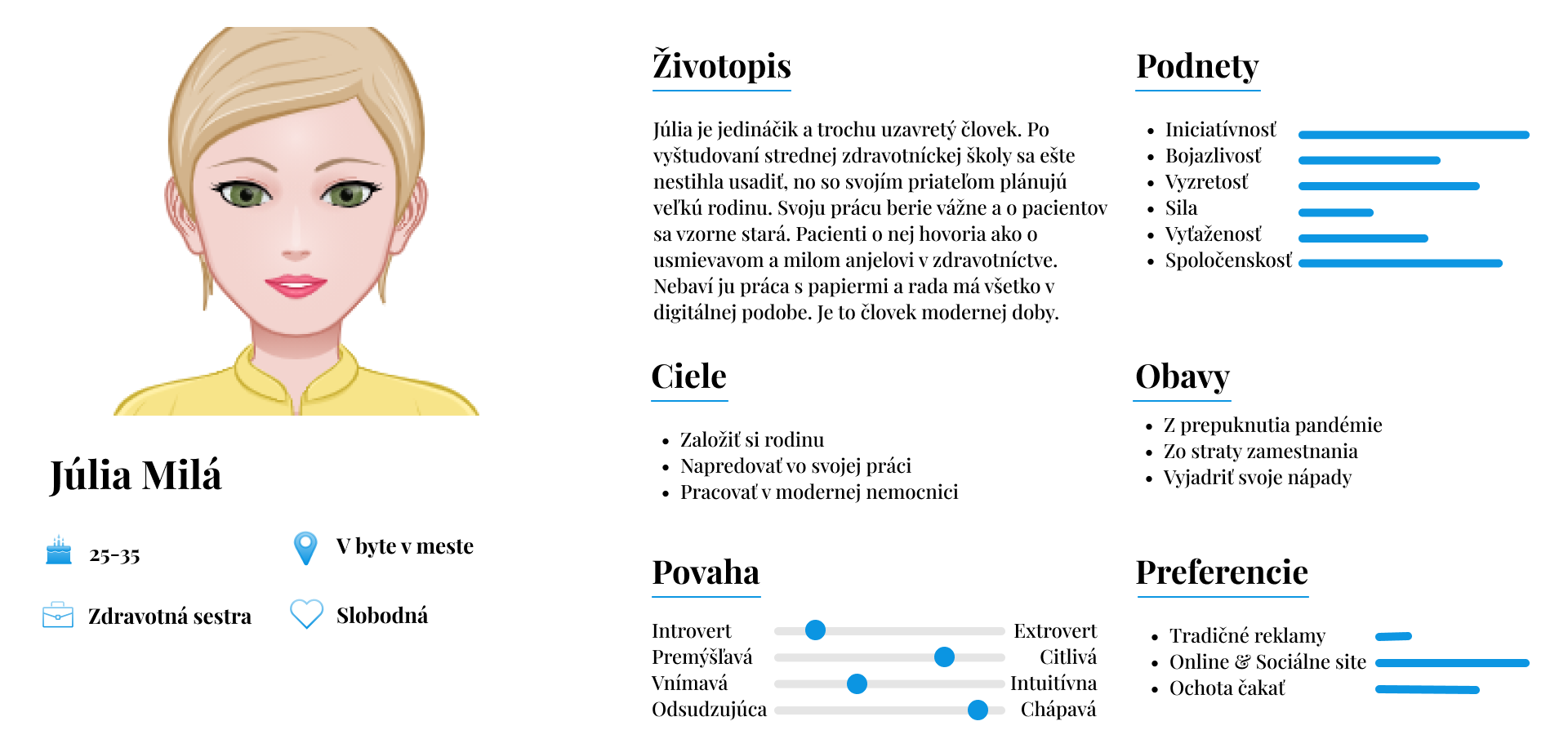
• (W) – Won't have: požiadavky, pri ktorých sa zákazník a dodávateľ dohodnú, že nebudú súčasťou pripravovanej verzie, ale môžu byť implementované niekedy v budúcnosti.

* 1. Persóny

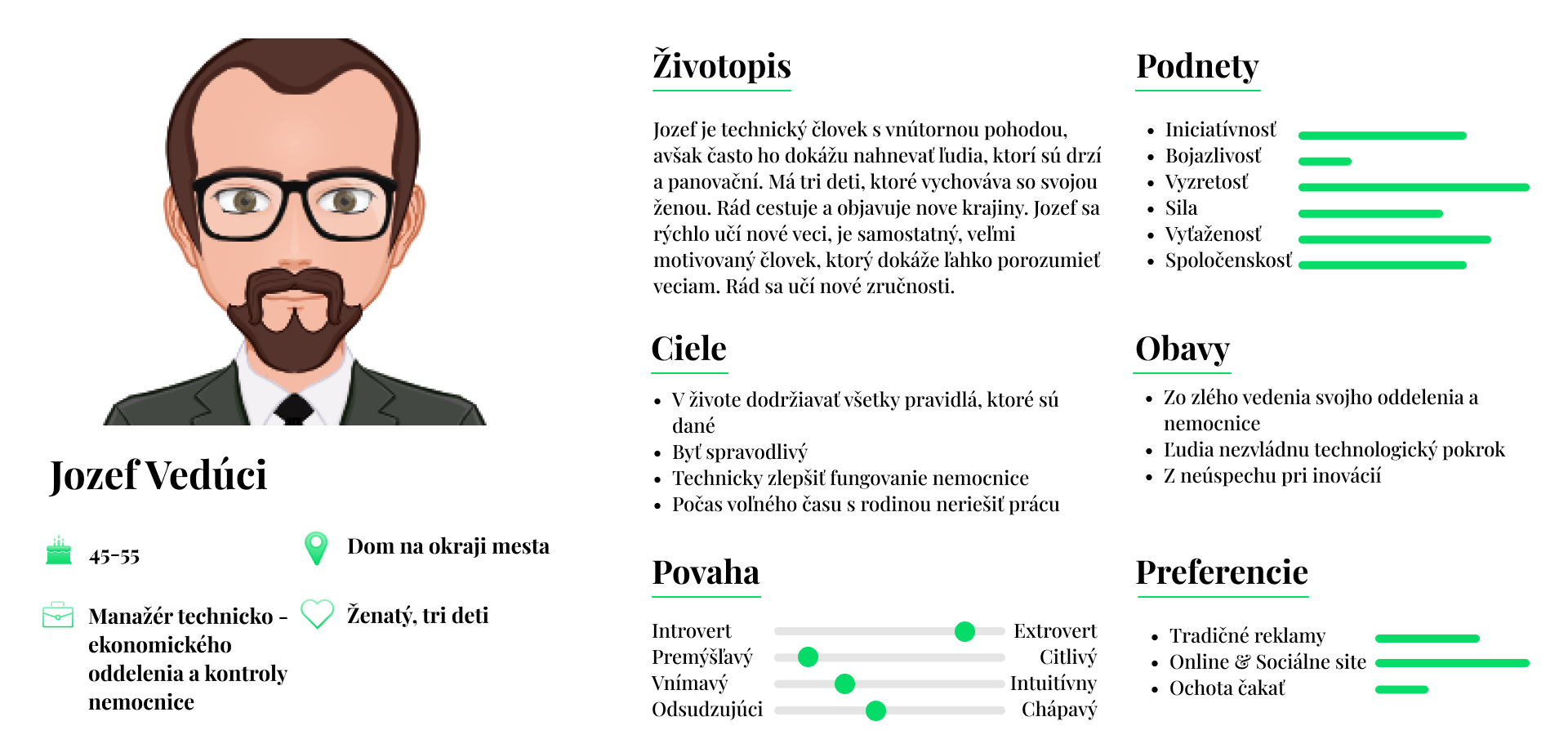
Pred tvorením samotného systému je potrebné vytvoriť si persóny, ktoré budú reprezentovať skutočných používateľov. Každá persóna reprezentuje špecifický typ používateľov. Sú vytvárané na základe kvalitatívneho a určitého kantitatívneho používateľského prieskumu realizovaného pomocou rozhovorov, prieskumov, dotazníkov a pod. Dobre navrhnuté osobnosti pomáhajú formovať plán produktu a zosúladiť všetky zainteresované strany [23].



Obr. Persóna Lekár



Obr. Persóna Zdravotná sestra



Obr. Persóna Vedenie nemocnice

* 1. Plánované funkcie
     1. Zdravotná sestra

1. Zobraziť pacientov na oddelení
2. Vytvoriť profil pacienta
3. Zobraziť profil pacienta
4. Upraviť profil pacienta
5. Odstrániť profil pacienta
6. Pridať meranie pacientovi
7. Upraviť pridané meranie
8. Zobraziť všetky namerané hodnoty
9. Pridať poznámku k pacientovi
10. Zobraziť a editovať svoj profil
    * 1. Lekár
11. Zobraziť pacientov na oddelení
12. Zobraziť profil pacienta
13. Upraviť profil pacienta
14. Zobraziť výsledky meraní pacienta
15. Zaznamenať vizitu
16. Upraviť zaznamenanú vizitu
17. Pridať poznámku k pacientovi
18. Zobraziť a editovať svoj profil
    * 1. Vedenie nemocnice
19. Zobraziť oddelenia v nemocnici
20. Vytvoriť oddelenie nemocnice
21. Upraviť oddelenie
22. Odstrániť oddelenie
23. Zobraziť lekárov na oddelení
24. Vyhľadať a pridať lekára na oddelenie
25. Zobraziť zdravotné sestry na oddelení
26. Vyhľadať a pridať zdravotnú sestru na oddelenie
27. Zobraziť pacientov na oddelení
28. Zobraziť údaje o pacientovi
29. Zobraziť a editovať svoj profil
    1. Scenáre použitia

Na základe používateľských požiadaviek a persón sme navrhli možné scenáre použitia. Používateľský scenár popisuje vstupné a výstupné podmienky, a ako budú používatelia v systéme vykonávať jednotlivé kroky. Vybrané scenáre použitia sú prezentované nižšie formou tabuliek: Pridať meranie pacientovi (Tab. 4), Zaznamenať vizitu (Tab. 5) a Vyhľadať a pridať lekára na oddelenie (Tab. 6).

|  |  |
| --- | --- |
| Use Case ID: | 6 |
| Názov: | **Pridať meranie pacientovi** |
| Aktor: | Zdravotná sestra |
| Popis: | Zdravotná sestra môže pridať meranie pacientovi na jej oddelení. |
| Preconditions: | * Zdravotná sestra je prihlásená do systému * Zdravotná sestra je priradená k oddeleniu * Existuje pacient na jej oddelení |
| Postconditions: | * Meranie sa uloží do databázy s informáciou o tom kedy a ktorou zdravotnou sestrou bolo meranie vykonané * V zozname predchádzajúcich meraní sa pacientovi zaznamená vykonané meranie |
| Štandardný tok: | 1. **Pridať meranie pacientovi** 2. Zdravotná sestra si zvolí pacienta, ktorému chce pridať meranie. 3. Systém zobrazí profil daného pacienta. 4. Zdravotná sestra klikne na tlačidlo – Pridať meranie. 5. Systém zobrazí ponuku s možnými meraniami. 6. Zdravotná sestra si zvolí meranie, ktoré chce vykonať. 7. Systém zobrazí možnosť zadať namerané hodnoty. 8. Zdravotná sestra zadá nameranú hodnotu a klikne na tlačidlo – Uložiť. 9. Systém zapíše do databázy nameranú hodnotu, dátum a čas a zobrazí zoznam všetkých nameraných hodnôt daného merania u pacienta. |
| Alternatívny tok: |  |
| Výnimky: |  |
| Poznámky: | 1. Nie je obmedzený počet denných záznamov meraní. 2. Nie je možná editácia dátumu a času. Tieto údaje sú automaticky zadané podľa aktuálneho dátumu a času. 3. Ku každému meraniu je možné pridať poznámku. |

Tab. Scenár použitia - Pridať meranie pacientovi

|  |  |
| --- | --- |
| Use Case ID: | 15 |
| Názov: | **Zaznamenať vizitu** |
| Aktor: | Lekár |
| Popis: | Lekár môže pacientovi na jeho oddelení zaznamenať vizitu, teda zapísať aktuálny stav pacienta do databázy. |
| Preconditions: | * Lekár je prihlásený do systému * Lekár je priradený k oddeleniu * Existuje pacient na jeho oddelení |
| Postconditions: | * Záznamy sa uložia do databázy s informáciou o tom kedy a ktorým lekárom bola vizita vykonaná * Nová vizita sa zapíše do zoznamu vykonaných lekárskych záznamov. |
| Štandardný tok: | **15.0 Zaznamenať vizitu**   1. Lekár si zvolí pacienta, ktorému chce zaznamenať vizitu. 2. Systém zobrazí profil daného pacienta. 3. Lekár klikne na tlačidlo – Zaznamenať vizitu. 4. Systém zobrazí ponuku s možnosťami zapísať aktuálny stav pacienta, lieky, nadchádzajúce vyšetrenia alebo poznámku. 5. Lekár vyplní ním zvolené polia a klikne na tlačidlo – Uložiť. 6. Systém zapíše do databázy zapísané polia, dátum a čas a zobrazí zoznam všetkých vykonaných vizít u pacienta. |
| Alternatívny tok: |  |
| Výnimky: |  |
| Poznámky: | 1. Nie je obmedzený počet denných záznamov vizity. 2. Nie je možná editácia dátumu a času. Tieto údaje sú automaticky zadané podľa aktuálneho dátumu a času. |

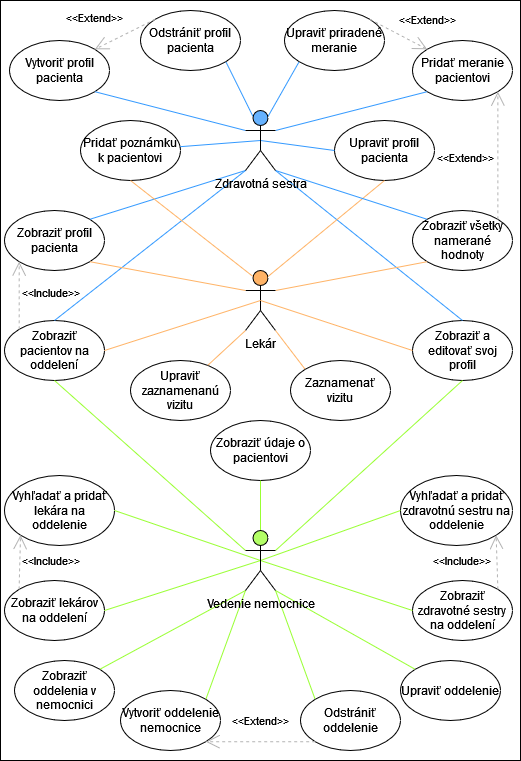
Tab. Scenár použitia - Zaznamenať vizitu

|  |  |
| --- | --- |
| Use Case ID: | 24 |
| Názov: | **Vyhľadať a pridať lekára na oddelenie** |
| Aktor: | Vedenie nemocnice |
| Popis: | Lekár po zaregistrovaní sa do systému nie je hneď priradení k oddeleniu. Vedenie nemocnice vie v zozname nepriradených lekárov vyhľadať lekára a priradiť ho v svojej nemocnici ku konkrétnemu oddeleniu. |
| Preconditions: | * Vedenie nemocnice je prihlásené do systému * Vedenie nemocnice má vytvorené oddelenie |
| Postconditions: | * Lekár sa priradí v databáze k nemocnici na konkrétne oddelenie * Vedeniu nemocnice sa v systéme aktualizuje zoznam lekárov * Lekár sa priradí k oddeleniu a uvidí pacientov na svojom oddelení |
| Štandardný tok: | **24.0 Vyhľadať a pridať lekára na oddelenie**   1. Vedenie nemocnice si zvolí oddelenie svojej nemocnice. 2. Systém zobrazí možnosti výberu medzi lekármi, zdravotnými sestrami a pacientmi. 3. Vedenie nemocnice si zvolí možnosť – Lekári. 4. Systém zobrazí zoznam lekárov priradených k oddeleniu. 5. Vedenie nemocnice klikne na tlačidlo “+” znázorňujúce možnosť pridať lekára. 6. Systém zobrazí možnosť vyhľadať lekára spolu so zoznamom nepriradených lekárov. 7. Vedenie nemocnice zadá do vyhľadávania meno hľadaného lekára. 8. Systém zobrazí všetkých lekárov, ktorých meno sa zhoduje s vyhľadávaním. 9. Vedenie nemocnice klikne na meno lekára, ktorého chce pridať na oddelenie. Svoj výber potvrdí kliknutím na tlačidlo – Pridať lekára. 10. Systém zapíše lekára na oddelennie v databáze a zobrazí aktualizovaný zoznam lekárov oddelenia. |
| Alternatívny tok: |  |
| Výnimky: | **24.0.E.1Pre hľadaný výraz neexistujú v databáze lekári** (v kroku 7)   1. Systém nezobrazí žiadnych lekárov. 2. Vedenie nemocnice sa rozhodne pokračovať v hľadaní.    1. Upraví hľadaný výraz.    2. Späť na krok 8. 3. Vedenie nemocnice sa rozhodne nepokračovať v hľadaní.    1. Návrat na predchádzajúcu stránku.    2. Systém uzavrie prípad použitia. |
| Poznámky: | 1. Pri vyhľadávaní sa zobrazujú všetci lekári, ktorí ešte nie sú priradení k žiadnej nemocnici ani oddeleniu. 2. Vedenie nemocnice môže kedykoľvek priradeného lekára odstrániť. |

Tab. Scenár použitia - Vyhľadať a pridať lekára na oddelenie

* 1. UML diagram prípadov použitia

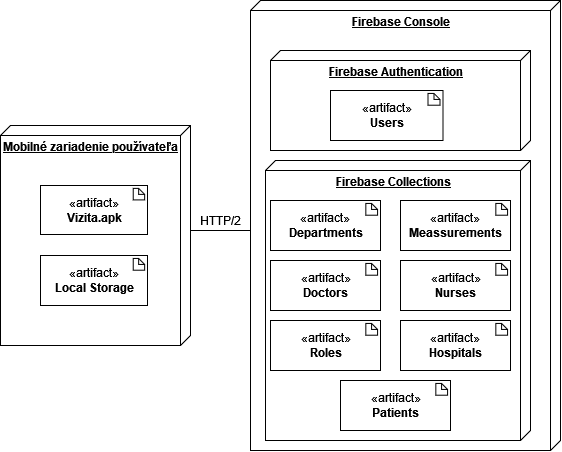
Nasledujúci diagram prípadov použitia (Obr. 5) popisuje funkcionalitu systému, teda čo by mal vedieť urobiť. Diagram sa skladá z prípadov použitia, aktérov a vzťahov medzi nimi. Prípady použitia popisujú spôsoby použitia systému z pohľadu koncového používateľa. Aktéry sú možný používatelia aplikácie, v tomto prípade ide o zdravotnú sestru, lekára a vedenie nemocnice.



Obr. UML diagram prípadov použitia

* 1. Architektúra systému

Pri tvorbe mobilných aplikácií sa častokrát používa implementácia vo forme plnohodnotnej architektúry klient-server. Zvyčajne existuje frontendová aplikácia, backendová aplikácia s databázou a rozhraním API, ktoré umožňuje komunikáciu medzi nimi. Vytvárať takéto komplikované systémy vlastnými silami však môže byť zdĺhavé. Spravidla si to vyžaduje ovládanie viacerých robustných programovacích jazykov. V našom prípade, keďže sme mali na tvorbu len obmedzený čas, a technológie nám neboli až tak známe, tak sme hľadali jednoduchšie riešenia, na ktorých by sme mohli aplikáciu postaviť. Hľadali sme riešenie, pri ktorom bude použité čo najmenšie množstvo programovacích jazykov. Keďže sme už nejaké skúsenosti s JavaScriptom mali, rozhodli sme sa ho využiť. Pre frontedovú aplikáciu riešenia sme použili knižnicu React Native, a ako server sme použili Firebase. Firebase ponúka jednoduché riešenia využitia databáz, autentifikáciu a autorizáciu. Toto cloudové riešenie je jednoduchšou formou backedovej aplikácie [24].



Obr. UML diagram nasadenia

* + 1. Firebase

Nástroj Firebase bol kúpený v roku 2014 spoločnosťou Google a stal bežnou náhradou backendových aplikácií. Vývojárom poskytuje množstvo nástrojov a služieb, ktoré im pomáhajú vyvíjať kvalitné aplikácie, rozširovať používateľskú základňu a zarábať. Vyvíjať backendové riešenia priamo pre platformy Android a iOS môže byť náročné, najmä ak ide o spravovanie viacerých samostatných kódových báz. Zložité úlohy ako je správa databáz, overovanie identity a iné, výrazne zvyšujú čas a úsilie, ktoré je potrebné investovať do tvorby aplikácie. Firebase je platforma na vývoj aplikácie, ktorá je postavená na infraštruktúre Google. Jej cieľom je poskytnúť nástroje a infraštruktúru, ktoré sú potrebné na vytváranie aplikácií, rast úspešného podnikania a zárobky zo svojej práce. Nie je to náhrada za už existujúce rozhrania API, ide len o vylepšenie, ktoré poskytuje bežné služby, ktoré by inak mohli byť potrebné. Sú to služby ako napríklad backend databázy, bezpečné overovanie, posielanie správ a ďalšie. To umožňuje zamerať sa na dôležitejšie aspekty aplikácie. Je dôležité taktiež poznamenať, že každý si môže vybrať, ktorú z ponúkaných služieb chce vo svojej aplikácií použiť. Nie je nutné používať všetky z nich. Mnohé ponúkané služby sú dostupné zadarmo. Pre tie ostatné existuje bezplatná verzia, ktorá funguje na testovanie s primeranými limitmi pre menšie aplikácie [25].

Firebase má 8 služieb, ktoré sú navrhnuté tak, aby pomohli zlepšiť vývoj aplikácií. V našej aplikácií sme využili dve z nich.

* + - 1. Firebase overovanie

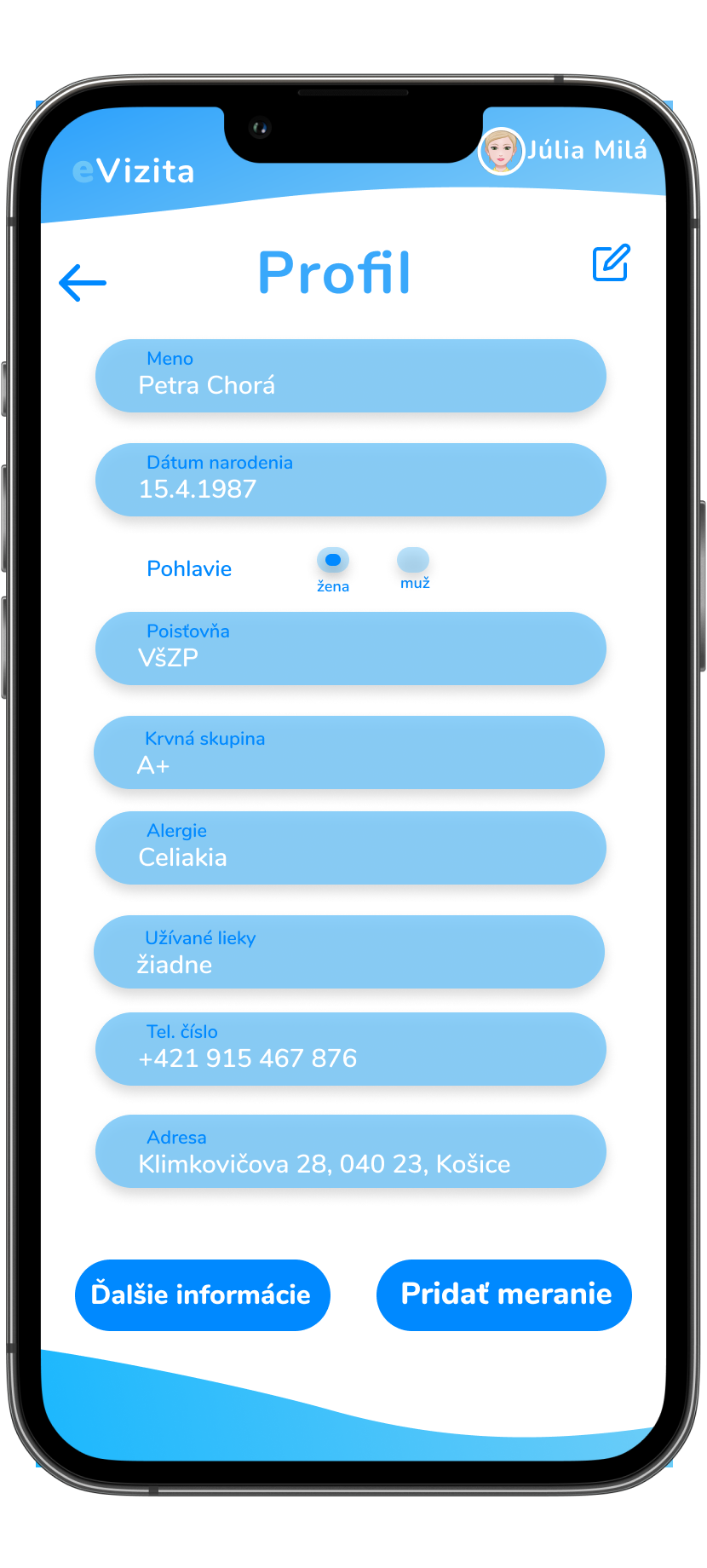
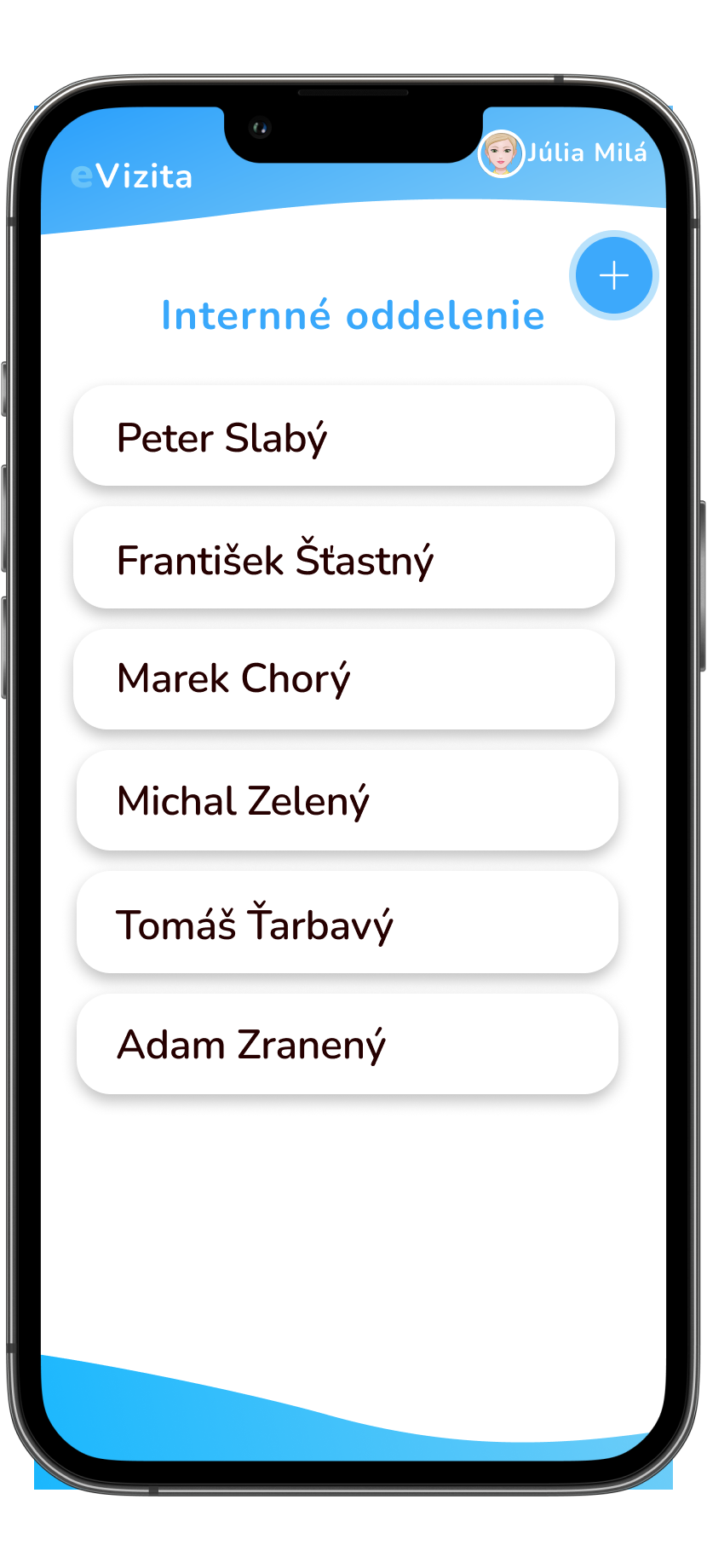
Pri aplikácií, ktorá potrebuje poznať identitu používateľa, aby mu mohla poskytnúť špecifické údaje, je nevyhnutná forma bezpečného prihlásenia. Vybudovanie a údržba prihlasovacej infraštruktúry je náročná a nákladná záležitosť. Odmietnutie používania aplikácie môže byť spojené práve s nutnosťou udelenia poverení a osobných údajov, ktoré môžu byť pre používateľa akousi pomyselnou stopkou. Popularita aplikácií, ktoré umožňujú používateľom prihlásiť sa pomocou známych overení za pomoci aplikácií ako Google, Facebook či Twitter stále narastá. S ohľadom na všetky pozitívne a negatívne aspekty bolo vytvorené Firebase overovanie, ktoré poskytuje jednoduché rozhranie API. To umožňuje rozšíriť si svoje vlastné overovanie, využívať overovanie emailom a heslom, či prihlásenie od federatívnych poskytovateľov. Firebase overovanie je prepojené s inými Firebase službami, ako je napríklad databáza v reálnom čase a Cloud Firestore, takže je možné kontrolovať, kto má prístup k akým dátam [25].

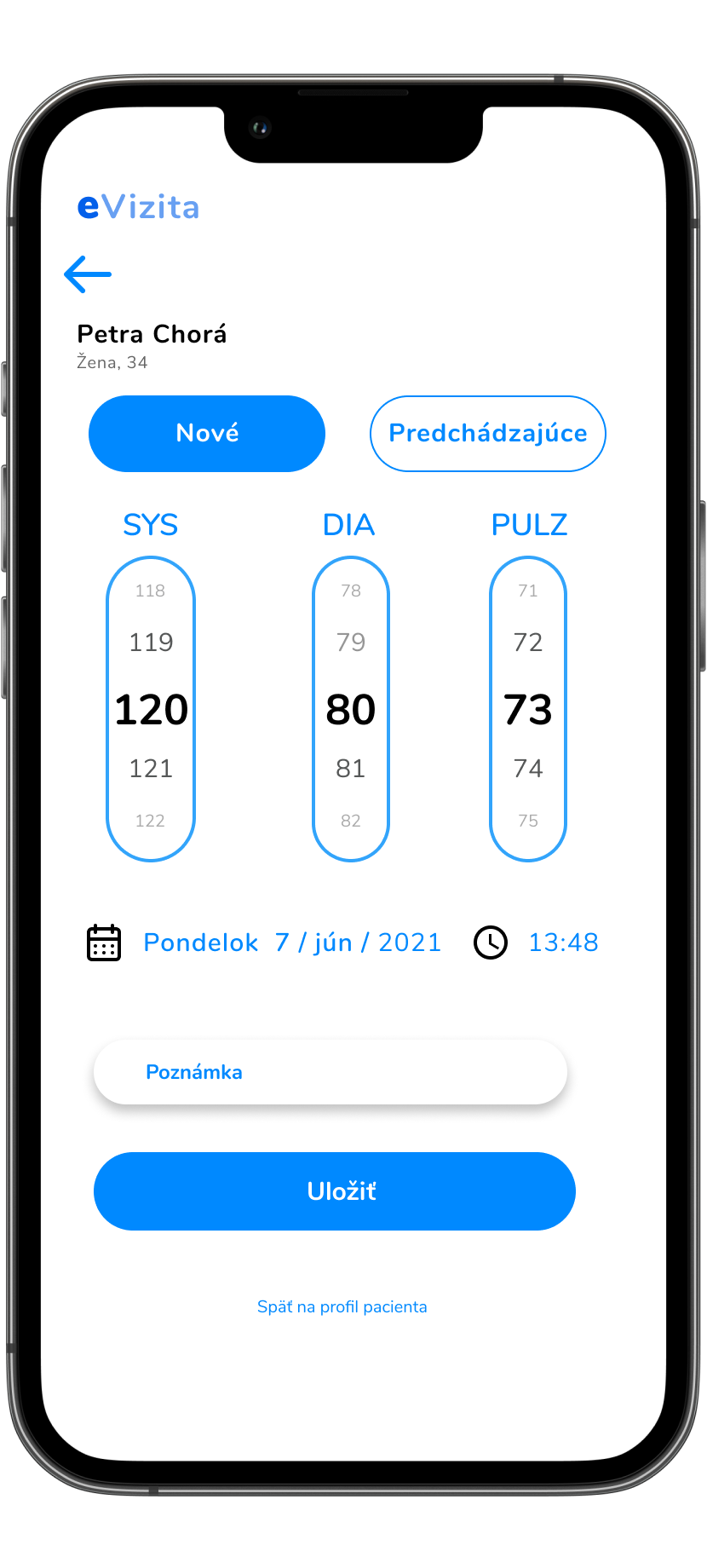
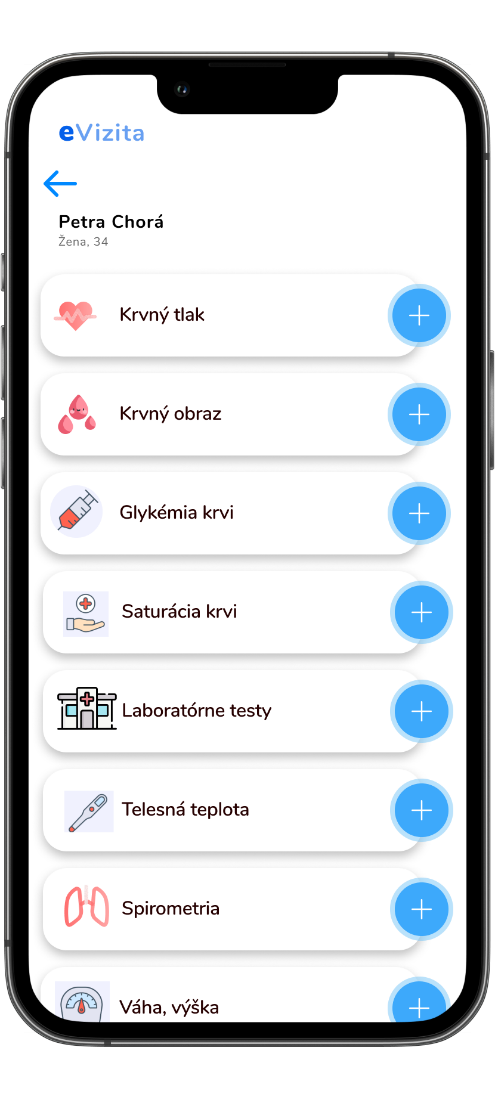
* + - 1. Cloud Firestore

Cloud Firestore je najnovšia verzia databázy Firebase v reálnom čase. Je kategorizovaný ako databázový program NoSQL , ktorý umožňuje ukladať, synchronizovať a vyhľadávať údaje. Poskytuje synchronizáciu medzi viacerými zariadeniami pričom automaticky dostávajú aktualizácie s najnovšími údajmi. Pomocou lokálnej vyrovnávacej pamäte zabezpečuje dostupnosť aj bez internetového pripojenia. Ide o databázu riadenú udalosťami, ktorá funguje veľmi odlišne od tradičných databáz. Veľkou výhodou tejto databázy je, že neexistuje žiaden kód na strane servera, všetko kódovanie sa vykonáva na strane klienta [25].

* 1. Návrh používateľského rozhrania

Pri tvorbe návrhu používateľského rozhrania sme vychádzali z už predtým spísaných používateľských požiadaviek. Vďaka nim sme vedeli pre koho je systém určený, čo od neho očakávame a aké má mať vlastnosti a funkcie. Snažili sme sa docieliť to, aby všetky požiadavky boli zahrnuté a zobrazené už pri tvorbe dizajnu. Pri procese navrhovania grafického rozhrania sme pracovali s grafickým editorom Figma. Táto aplikácia poskytuje prostredie pre tvorbu grafických dizajnerských prác, webstránok, prototypovania a navrhovania rozhraní mobilných aplikácií. Súčasťou Figma editora je aj priestor Figma komunita, kde môžu ľudia, tímy alebo organizácie publikovať svoje nápady a súbory. Tento pristor nám poskytol inšpiráciu pred tvorbou nášho mockupu, pretože sme si mohli pozrieť nápady a tvorbu iných dizajnerov, ktorých aplikácie boli podobné tej našej. Celkovo sme sa snažili, aby používateľské rozhranie bolo jednoduché a efektívne. Keďže sme si pripravili návrh dizajnu, zvýšili sme tým efektivitu pri samotnom programovaní a štýlovaní aplikácie. Vytvorili sme si návrh rozloženia aplikácie, základné komponenty a paletu farieb, aby sme docielili konzistentnosť celej aplikácie.





1. Implementácia aplikácie eVizita
   1. Prototyp
   2. Finálna verzia
2. Testovanie
   1. Priebeh testovania
   2. Výsledky testovania

Záver

Zoznam použitej literatúry

1. HAMED, Trafa - DARA, Rozita: Intrusion Detection in Contemporary Environments. Guelph: University of Guelph, 2017. ISBN 978-012-803-929-8.
2. StatCounter Global Stats: Mobile & Tablet Operating System Market Share Worldwide | Statcounter Global Stats. [online]. 2022 [cit. 3.5.2022]. Dostupné na internete: <https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-201801-202201>
3. JAISWAL, Manishaben: Android the mobile operating system and architecture. In: International Journal of Creative Research Thoughts. Roč. 6, č. 1 (2018), s. 514-515 ISSN 2320-2882.
4. ANWAR, Rami: Android vs iOS development: Whitch platform should I develop for first? [online]. 2020 [cit. 5.5.2022]. Dostupné na internete: <https://easternpeak.com/blog/android-vs-ios-development-which-platform-first/#so-which-platform-to-choose-first>
5. KNEZOVIĆ, Martin: Okienko do minulosti: História iOS. [online]. 2019 [cit. 5.5.2022]. Dostupné na internete: <https://www.macblog.sk/2019/okienko-do-minulosti-historia-ios/>
6. SINGH, Satinder: Native vs Hybrid vs Cross Platform – What to Choose in 2022? [online]. 2021 [cit. 2.5.2022]. Dostupné na internete: <https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/#what-is-a-native-app>
7. VALDELLON, Lionel: What Are the Different Types of Mobile Apps? And How Do You Choose? [online]. 2020 [cit. 13.5.2022]. Dostupné na internete: <https://clevertap.com/blog/types-of-mobile-apps/>
8. DESHPANDE, Chinmayee: The Best Guide to Know What Is React. [online]. 2021 [cit. 6.5.2022]. Dostupné na internete: <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs>
9. DESHPANDE, Chinmayee: The Ultimate React Native Tutorial. [online]. 2022 [cit. 6.5.2022]. Dostupné na internete: <https://www.simplilearn.com/react-native-tutorial-article?referrer=search&tag=React%20native>
10. PAUL, Akshat - NALWAYA, Abhishek: React Native for Mobile Development. Haryana: Apress, 2019. ISBN 978-1-4842-4454-8.
11. EISENMAN, Bonnie: Learning React Native: Building Native Mobile Apps with JavaScript. O'Reilly Media Inc., 2016. ISBN 978-1-491-92900-1.
12. BENNETT, Jim: Xamarin in Action. Manning Publications Co., 2018. ISBN 978-1617294389
13. BRITCH, David: What is Xamarin? [online]. 2021 [cit. 15.5.2022]. Dostupné na internete: <https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>
14. KOPACHOVETS, Oleg: Xamarin vs Flutter: What to Choose for Your Preject? [online]. 2022 [cit. 15.5.2022]. Dostupné na internete: <https://procoders.tech/blog/flutter-vs-xamarin/>
15. NAPOLI, Marco L.: Beginning Flutter. Indianapolis: John Wiley & Sons, Inc., 2020. ISBN 978-1-119-55082-2
16. FREEBORN, Charles: Pros and cons of Flutter app development. [online]. 2021 [cit. 14.5.2022]. Dostupné na internete: <https://blog.logrocket.com/pros-and-cons-of-flutter-app-development/>
17. TIRPÁKOVÁ, Libuša - SOVÁRIOVÁ SOÓSOVÁ, Mária: Ošetrovateľské techniky. Košice: Univerzita Pavla Jozefa Šafárika v Košiciach, 2016. ISBN 978-8152-441-7.
18. Zákon č. 576/2004 Z.z. o zdravotnej starostlivosti, službách súvisiacich s poskytovaním zdravotnej starostlivosti a o zmene a doplnení niektorých zákonov
19. MASARYK, Vladimír - LESŇÁKOVÁ, Anna: Vedenie zdravotnej dokumentácie v súlade s legislatívou. [online]. 2018 [cit. 3.4.2022]. Dostupné na internete: [MASARIK, V. - LESŇÁKOVÁ, A. - VEDENIE ZDRAVOTNEJ DOKUMENTÁCIE V SÚLADE S LEGISLATÍVOU.pdf (vsdanubius.sk)](https://revue.vsdanubius.sk/sites/default/files/MASARIK%2C%20V.%20-%20LES%C5%87%C3%81KOV%C3%81%2C%20A.%20%20-%20VEDENIE%20ZDRAVOTNEJ%20DOKUMENT%C3%81CIE%20V%20S%C3%9ALADE%20S%20LEGISLAT%C3%8DVOU.pdf?msclkid=34cfe4f1b39011ec9464d941b2621892)
20. Health & Fitness AI Lab: Denník krvného tlaku. [online] in Google Play [citované 10.4.2022] dostupné na internete: <https://play.google.com/store/apps/details?id=com.bluefish.bloodpressure&gl=SK>
21. YEROKHIN Vladimir: Medical records. [online] in Google Play [citované 10.4.2022] dostupné na internete: <https://play.google.com/store/apps/details?id=vladimir.yerokhin.medicalrecord>
22. MedClin: Medical Records Clinic app. [online] in Google Play [citované 10.4.2022] dostupné na internete: <https://play.google.com/store/apps/details?id=com.cliniconline>
23. PRADHAN, Sourabh: How to Create Effective User Personas. [online]. 2020 [cit. 9.5.2022]. Dostupné na internete: <https://medium.com/agileinsider/how-to-create-effective-user-personas-9e383d166cf>
24. WIERUCH, Robin: The Road to React with Firebase. Self-published, 2019. ISBN 978-1795010894.
25. MORONEY, Laurence: The Definitive Guide to Firebase. Seattle: Apress 2017. ISBN 978-1-4842-2943-9.

Prílohy

Príloha A: CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe. CD je spravidla grafické s logom univerzity a fakulty. Pozri . Tieto CD robia v Univerzitnej knižnici TUKE.

Príloha B: Používateľská príručka

Príloha C: Systémová príručka

Táto časť diplomovej práce je povinná a obsahuje zoznam všetkých príloh vrátané elektronických nosičov. Názvy príloh v zozname musia byt’ zhodné s názvami uvedenými na príslušných prílohách. Tlačené prílohy majú na prvej strane identifikačné údaje – informácie zhodné s titulnou stranou diplomovej práce doplnené o názov príslušnej prílohy (Systémová príručka, Používateľská príručka). Identifikačné údaje sú aj na priložených diskoch alebo disketách. Ak je médií viac, sú označené aj číselne v tvare I/N, kde I je poradové číslo a N je celkový počet daných médií.

Každá príloha začína na novej strane a je označená samostatným písmenom (Príloha A, Príloha B, ...). Číslovanie strán príloh nadväzuje na číslovanie strán v hlavnom texte.



Obr. Obrázok grafického CD média