Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Monitorovanie a vyhodnocovanie fyziologických procesov človeka [StressMonitor]

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Katarína Jelemenská, PhD.
Členovia tímu: Bc. Martin Nemček, Bc. Oliver Moravčík, Bc. Miloš Pallo, Bc. Dániel Papp, Bc. Barbora Pavlíková, Bc. Martin Šrank
Školský rok: 2016/2017

OBSAH

Úvod 1

Slovník pojmov 2

1 Globálne ciele projektu 3

1.1 Ciele pre zimný semester 3

1.2 Bližší opis cieľov pre zimný semester 3

1.3 Naplnenie cieľov pre zimný semester 4

2 Celkový pohľad na systém 5

2.1 Architektúra 5

2.2 Dátový model 5

2.3 Diagram tried 6

3 Moduly systému 9

3.1 Meranie stresu 9

3.2 Prototyp aplikácie 10

ÚVOD

Tento dokument je dokumentáciou softvérového produktu vytvoreného v rámci predmetu Tímový projekt v akademickom roku 2016/2017 pre tím číslo 12 s názvom Guardians of Stress. Témou tímového projektu je Vyhodnocovanie fyziologických procesov človeka (StresMonitor). Úlohou tímového projektu je vytvoriť mobilnú aplikáciu pre analýzu stresu používateľa. Cieľom tohto dokumentu je špecifikovať vytvorený softvérový produkt.

V prvej kapitole sú definované globálne ciele projektu pre zimný semester. V druhej kapitole je špecifikácia celého produktu. V tretej kapitole sú opísané moduly produktu.

SLOVNÍK POJMOV

Pojem	Vysvetlenie

1 GLOBÁLNE CIELE PROJEKTU

Cieľom projektu je vytvoriť mobilnú aplikáciu slúžiacu pre zbieranie dát z meracieho zariadenia, vyhodnotenie týchto dát a vizualizáciu informácií získaných z týchto dát.

1.1 Ciele pre zimný semester

Globálnym cieľom projektu pre zimný semester je vytvorenie funkčného prototypu mobilnej aplikácie, ktorá bude zobrazovať používateľovi informácie o jeho strese a zbierať dodatočné informácie o používateľovej aktivite. Mobilná aplikácia má byť podporovaná serverom, ktorý bude dáta ukladať, spracovávať a získavať namerané dáta z meracieho zariadenia. Preto sme pre zimný semester naplánovali vytvorenie funkčného prototypu mobilnej aplikácie s podporou serveru. Funkčný prototyp musí splniť nasledovné ciele:

1. Zbieranie dát z meracieho zariadenia.
2. Zbieranie dát pre vytvorenie kontextu.
3. Spracovanie a ukladanie dát na serveri.
4. Zobrazenie informácií získaných analýzou dát.

1.2 Bližší opis cieľov pre zimný semester

Meracie zariadenie vyvíja externá firma. Zariadenie a ešte nie je dokončené. Pre náš projekt je dostupný jeho prototyp, ktorý odosiela namerané dáta na server jeho tvorcov. Z tohto dôvodu budeme považovať cieľ zbierania dát z meracieho zariadenia za splnený, ak budeme získavať dáta vodivosti kože používateľa z tohto serveru.

Meracie zariadenie meria vodivosť kože používateľa a zmeny v hodnote vodivosti kože indikujú stres používateľa. Pre porozumenie hodnôt stresu je potrebné ich zasadiť do kontextu, preto je cieľom projektu využiť dáta dostupné z mobilného telefónu pre vytvorenie tohto kontextu. Aplikácia preto bude okrem dát o vodivosti kože zbierať aj dáta o geografickej polohe používateľa, počasí v danej lokalite, pohybe používateľa, jeho programe v kalendári, telefónnych hovoroch a SMS správach. Tento kontext pomôže pochopiť faktory vplývajúce na stres používateľa alebo zmeny v hodnote vodivosti kože, ktoré nie sú priamou odpoveďou na stresovú situáciu ale iba prirodzenou reakciou ľudského organizmu na jeho okolie alebo činnosť.

Mobilná aplikácia bude zbierať pomerne veľké množstvo dát, ktorých ukladanie a spracovanie nie je vhodné pre mobilný telefón, preto bude aplikácia nazbierané dáta odosielať na server, ktorý ich uloží a spracuje, a taktiež bude poskytovať dáta pre zobrazenie v mobilnej aplikácii. Prenos dát medzi mobilnou aplikáciou a serverom bude realizovaný pomocou REST API.

Mobilná aplikácia zobrazí používateľovi informácie o tom, kedy pociťoval stres, kde sa pri tom nachádzal, a aké faktory mohli jeho stres ovplyvniť (počasie, program v kalendári, telefónny hovor, atď.). Cieľom projektu je zobraziť tieto informácie používateľovi v takej forme, aby ich výpovednej hodnote čo najľahšie porozumel, pri čom najväčšou výzvou bude vizualizácia

týchto informácii na mobilnom telefóne, kde sú možnosti vizualizácie obmedzené veľkosťou displeja.

1.3 Naplnenie cieľov pre zimný semester

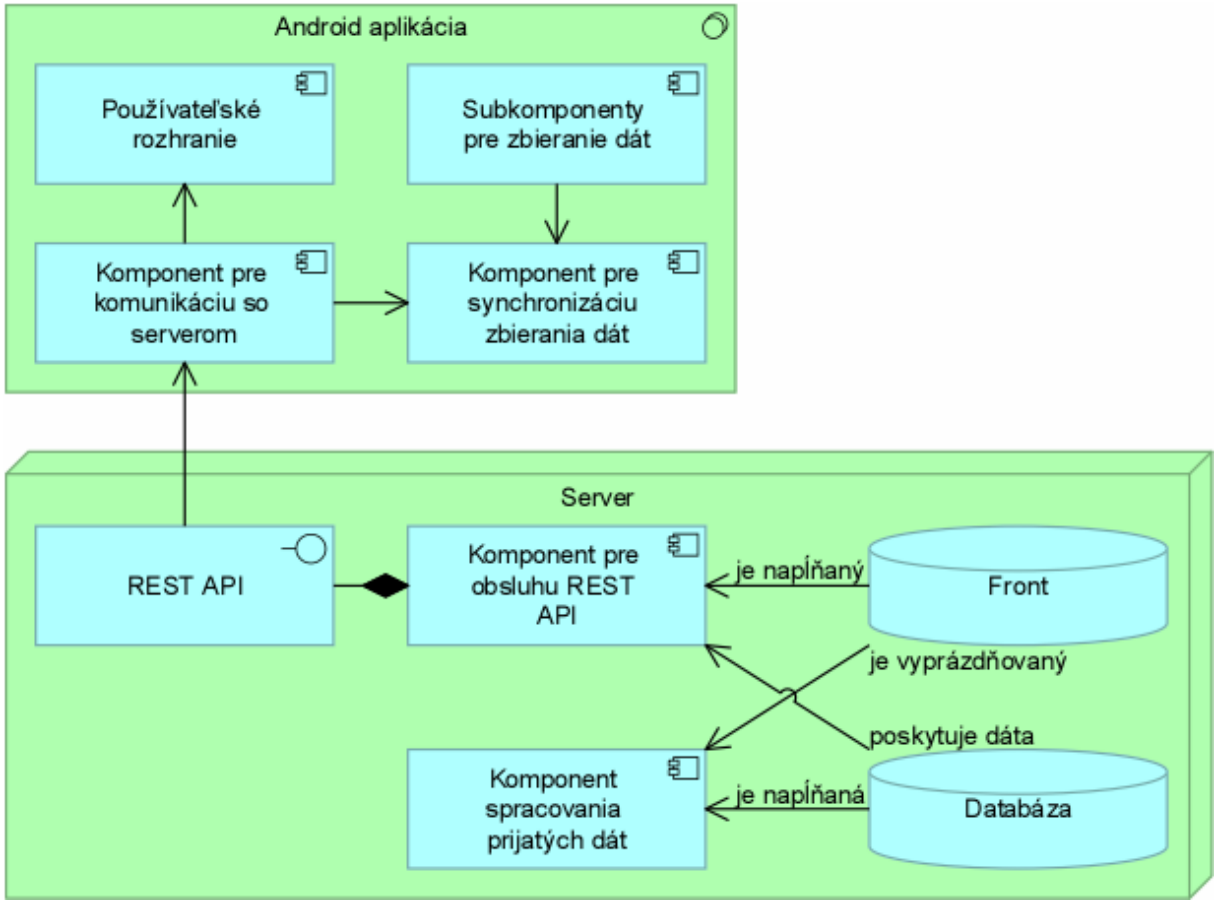
Z cieľov pre zimný semester sme zatiaľ splnili každý do len určitej miery. Zbieranie dát z meracieho zariadenia ešte nie je nasadené na serveri, ale implementácia je funkčná. Zistiť kedy mal používateľ stres ešte nevieme zistiť, pretože meracie zariadenie nám bolo sprístupnené len nedávno. Vykonali sme ale analýzu nad nazbieranými dátami a máme pripravený základný koncept pre implementáciu spracovania týchto. Aplikácia lokálne zbiera dáta o používateľovi pre vytvorenie kontextu nutného pre porozumenie jeho stresu. Odosielanie týchto dát na server ale ešte nebolo implementované. Mobilnú aplikáciu podporuje server s REST API rozhraním, ktorý dáta ukladá a spracováva, a poskytuje dáta pre vizualizáciu informácií o strese v mobilnej aplikácii. Mobilná aplikácia dané informácie vizualizuje v zrozumiteľnej forme pre používateľa.

2 CELKOVÝ POHĽAD NA SYSTÉM

2.1 Architektúra

Vytvorený produkt sa skladá z dvoch častí, mobilnej aplikácie pre Android a Python servera s REST API, Redis frontom a databázou PostgreSQL. Architektúra produktu preto predstavuje formu klient-server architektúry.

Klientská časť zbiera dáta, požiadavkou na REST API na serveri ich odošle. Server dáta prijme, spracuje a uloží. Keď klientská časť požiadavkou na REST API vyžiada dáta pre vizualizáciu, server dáta vytvorí v požadovanej forme a vráti klientskej časti v odpovedi na požiadavku.

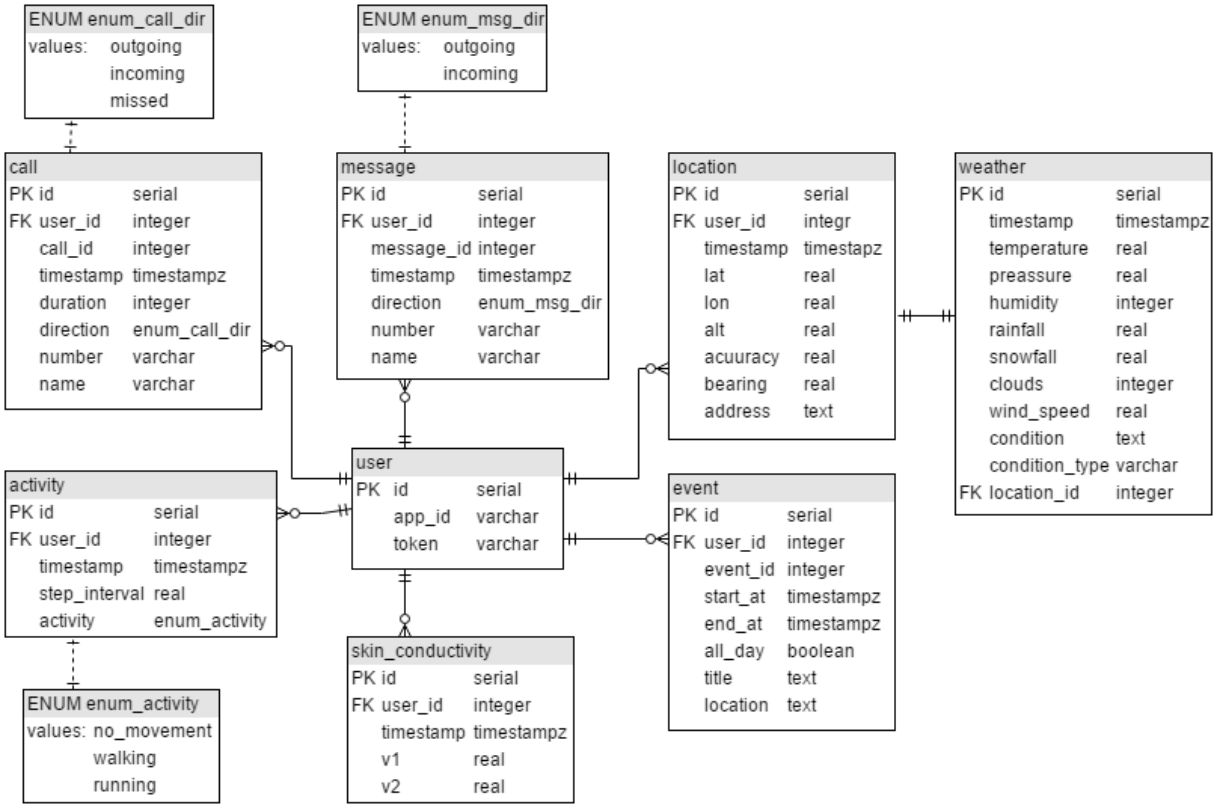


Obrázok 1: Diagram architektúry

2.2 Dátový model

Na serveri v databáze sú dáta uložené v dátovom modeli reprezentovanom v nasledujúcom diagrame. Entita *user* reprezentuje používateľa, entita *skin_conductivity* reprezentuje vodivosť

kože, entita *call* reprezentuje telefónny hovor, entita *message* SMS správu, *location* geografickú polohu, *weather* počasie v lokalite danej polohy, *event* udalosť v kalendári, *activity* fyzickú aktivitu (beh, chôdzu, bez pohybu).



Obrázok 2: Diagram dátového modelu

2.3 Diagram tried

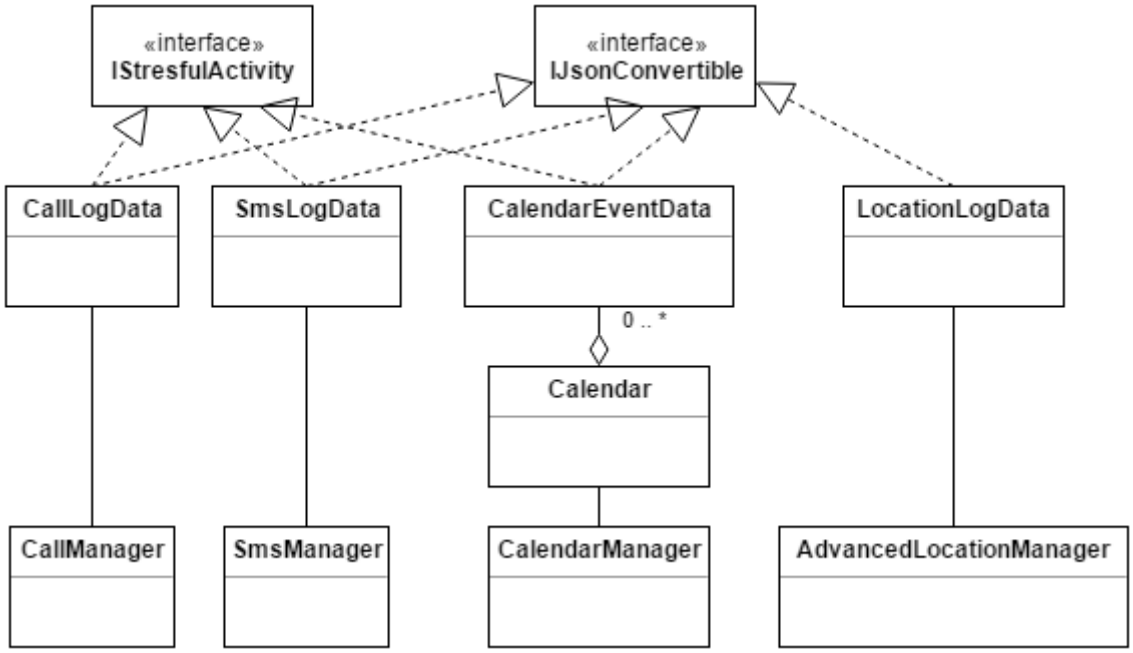
V nasledujúcom diagrame tried je znázornená doteraz implementovaná časť Android aplikácie.

Na obrázku Obrázok 3 pri pohľade z hora na dol vidíme rozhrania *IJsonConvertible* a *IStresfulActivity* a dátové objekty, ktoré ich realizujú. Dátové objekty realizujú rozhranie *IJsonConvertible* z dôvodu, že je nutné ich previesť do JSON formátu pre odosielanie dát. Tieto dátové objekty slúžia na reprezentáciu zbieraných dát v aplikácii.

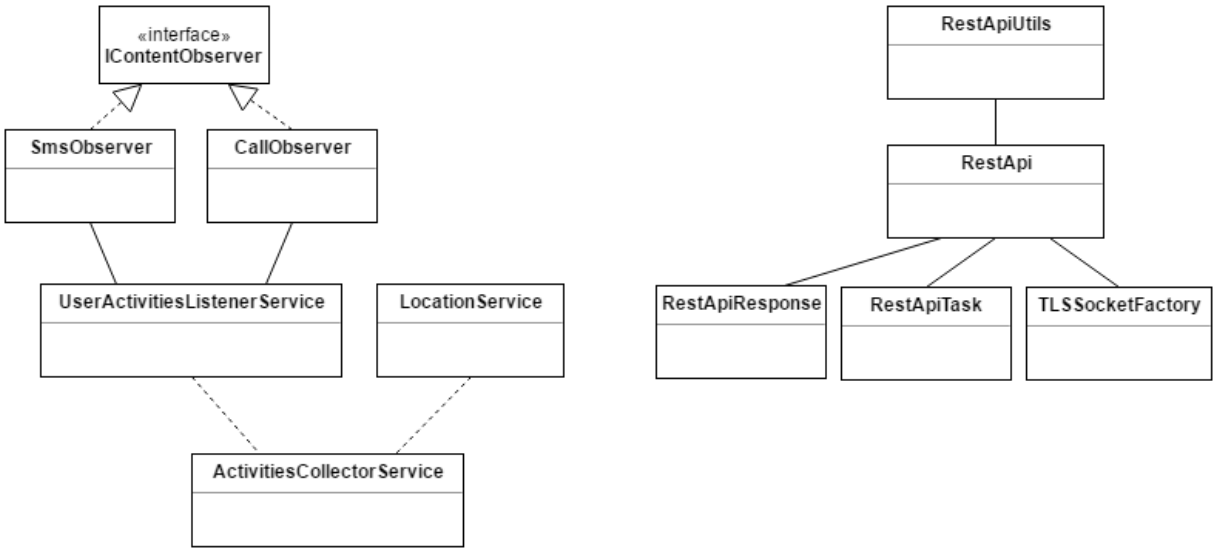
Na obrázku Obrázok 4 vpravo sú triedy, ktoré spolupracujú pri posielaní požiadaviek na REST API rozhranie, ktoré vystavuje server, a pri prijímaní odpovedí z rozhrania. Vľavo sú triedy, ktoré zabezpečujú zbieranie dát o používateľovi na pozadí.

Na obrázku Obrázok 5 sú triedy používateľského rozhrania aplikácie.

Celkový pohľad na systém

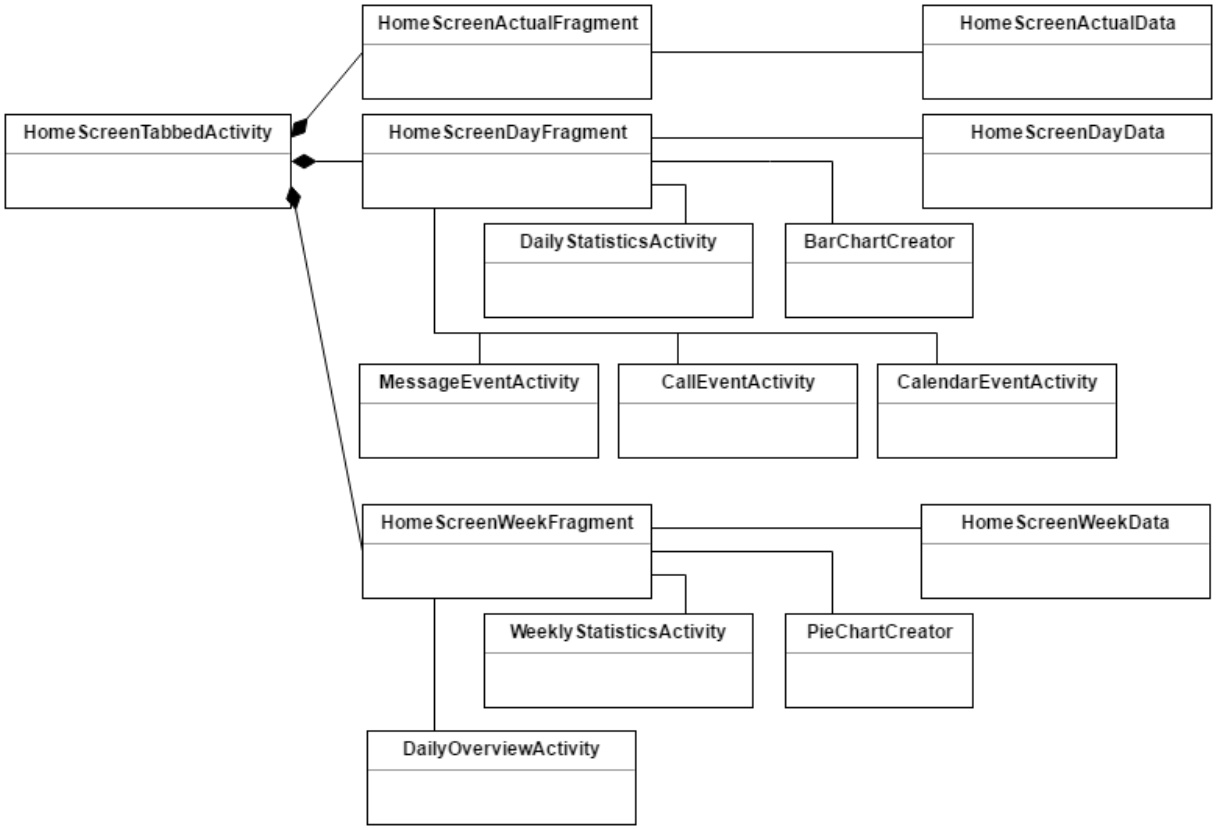


Obrázok 3: Diagram tried 1



Obrázok 4: Diagram tried 2

Celkový pohľad na systém



Obrázok 5: Diagram tried 3

3 MODULY SYSTÉMU

3.1 Meranie stresu

3.1.1 Analýza

Pre sledovanie stresu používateľa sa používa meranie vodivosti jeho kože. Sympatiková nervová sústava, ktorá reaguje na stresové podnety, ovplyvňuje odpor kože, ktorý je možné namerať. Odpor kože sa rovná elektrickému napätiu medzi dvomi elektródami na povrchu kože, predelenému prúdom prechádzajúcim cez kožu. $R = U/I$ (1) Prevrátená hodnota odporu kože je vodivosť kože. $G = = 1/R$ (2) Namerané hodnoty sú v micro Siemensoch (µS). Najvhodnejšie umiestnenie meracieho zariadenia je na druhom článku prsta nedominantnej ruky.

$$R = U/I \tag{1}$$

$$G = 1/R \tag{2}$$

Hodnoty odporu kože môžeme rozdeliť do dvoch pozadí: tonické a fázové pozadie. Hodnoty na tonickom pozadí opisujú základnú úroveň vodivosti kože. Úroveň týchto hodnôt sa kontinuálne mení v čase a je závislá od psychického stavu a autonómnej regulácie jedinca. Číselné hodnoty závisia od veľkosti elektród a umiestnenia elektród. Hodnoty na fázovom pozadí opisujú reakcie sympatikovej nervovej sústavy na stresové stimuly. Reakciu na stimul predstavujú krátke zvýšenia s prudkým stúpaním a postupným klesaním. Takéto zvýšenia sa vyskytujú aj spontánne približne jeden až tri krát za minútu. Pri stresovej situácii je počet zvýšení väčší, až do 20 zvýšení za minútu. Nie je preukázané, že by hodnota amplitúdy zvýšenia mala výpovednú hodnotu o stresovom stimule, preto jej veľkosť nie je podstatná. Vhodnou metrikou merania stresu je počet zvýšení za stanovený časový interval, preto je potrebné rozoznať všetky zvýšenia.

Faktorom ovplyvňujúcim meranie vodivosti kože je vlhkosť pokožky napríklad v dôsledku potenia. Meracie zariadenie, ktoré máme k dispozícií je ale podľa jeho tvorcov menej náchylné na zmenu hodnôt v dôsledku potenia, preto tento faktor v súčasnosti neriešime.

Výsledkom merania vodivosti kože je prúd hodnôt. Pre detekciu reakcií na stresové stimuly potrebujeme implementovať metódu, ktorá bude nezávislá od aktuálnej hodnoty, pretože aktuálne hodnoty závisia od tonickej úrovne vodivosti kože. Metóda musí rozoznať prudké zvýšenie v postupnosti hodnôt nasledované miernym klesaním. Po odborných konzultáciách máme základný koncept prístupu pre implementáciu algoritmu, ktorý zvýšenia rozozná. Je založený na premene aktuálnych hodnôt na rozdiel medzi po sebe nasledujúcimi hodnotami v postupnosti hodnôt. Vhodnou je aj úprava postupnosti hodnôt pohyblivým priemerom pre „vyhľadenie“ postupnosti.

Je pre nás vhodnejšie implementovať vlastný algoritmus než použitie knižničných funkcií, pretože môžeme algoritmus prispôsobiť na mieru charakteristike dát a taktiež prispôsobiť výstup algoritmu pre vlastné potreby.

3.1.2 Návrh

Pôvodne sme mali zbierať dáta z meracieho zariadenia pripojeného k mobilnému telefónu cez Bluetooth. Meracie zariadenie nám nebolo dostupné, preto sme implementovali vlastný generátor. Situácia sa ale zmenila a dostupné meracie zariadenie dáta odosiela na server jeho tvorcov, z ktorého dáta môžeme získať. Tento stav je dočasný, závisí od tvorcov meracieho zariadenia, preto sa musíme prispôsobiť. Do budúcnosti však tvorcovia zariadenia plánujú bezdrôtové spojenie s mobilným telefónom. To však ale nebude pravdepodobne možné počas trvania nášho tímového projektu. Z tohto dôvodu prispôsobujeme náš produkt situácií, nebudeme implementovať bezdrôtové spojenie s meracím zariadením a získavanie nameraných dát cez mobilný telefón, ale vhodnejšie je v tomto prípade získavať dáta z ich servera priamo na náš server, kde ich môžeme spracovať a uložiť.

Pre zbieranie dát zo servera, na ktorý ich odosiela meracie zariadenie môžeme využiť technológiu Web Socket, ktorá je na to veľmi vhodná pre jej jednoduchosť a rýchlosť.

Algoritmus pre detekciu reakcií pre stresové stimuly zatiaľ nie je implementovaný, pretože meracie zariadenie nám bolo sprístupnené len nedávno. V budúcom semestri návrh algoritmu spresníme a implementujeme na serveri.

3.2 Prototyp aplikácie

3.1.1 Analýza

Android aplikácia

Android aplikácie sú písané v jazyku Java, využívajú preto klasický prístup objektovo orientovaného programovania. Programovanie aplikácií pre Android je však špecifické kvôli tomu, že tento operačný systém je založený na Linuxe, a postupným vývojom Android API, ktoré je pre zjednodušenie vývoja aplikácií veľmi špecifické. Android aplikácie sa skladajú z komponentov, špecifických tried, ktoré majú svoj špeciálny účel a použitie (Activities, Services, Broadcast Receivers, atď.). Keďže tím nemal dostatočné skúsenosti s vývojom Android aplikácií, bolo nutné naštudovať základy o Android API.

REST služby na serveri

Aby Android aplikácia nebola náročná pre mobilný telefón, potrebuje delegovať spracovanie a ukladanie dát mimo mobilný telefón. Je preto potrebný server, ktorý dáta bude spracovávať a ukladať. Mobilná aplikácia tak musí využívať webové (aplikačné) služby, ktoré jej tento ser-

ver poskytne. Pre zjednodušenie komunikácie a šetrenie spotrebných dát internetového pripojenia na mobilnom telefóne je najvhodnejšou formou poskytovania webových služieb využitie technológie REST API.

3.2.2 Návrh

Android aplikácia

Android aplikácia bude zbierať dáta o používateľovej geografickej polohe, fyzickej aktivite, jeho udalostiach v kalendári, telefonátoch a SMS správach. Pre zber týchto dát využijeme vhodné komponenty Android API akými sú Services a BroadcastReceivers, ktoré zbierajú dáta na pozadí systému alebo reagujú na udalosti v systéme.

Najpodstatnejšou funkcionalitou aplikácie bude zobrazovanie informácií o používateľovom strese. Prototyp aplikácie bude zobrazovať grafy o používateľovom strese počas daného dňa, počas týždňa a pre dni v danom týždni. Pri grafoch aplikácia používateľovi zobrazí aj aké faktory mohli ovplyvniť jeho hodnoty stresu ako napríklad naplánované udalosti v kalendári, telefonáty, jeho poloha alebo počasie.

Server

Server musí vystaviť REST API rozhranie, ktoré mobilná aplikácia bude používať pre odosielanie zbieraných dát a pre vyžiadanie dát pre zobrazenie grafov a informácií o používateľovom strese.

Server musí prijaté dáta uložiť do databázy, odkiaľ mu budú dostupné pre spracovanie a zostavenie dát pre grafy zobrazené v mobilnej aplikácií. Server bude spracovávať a ukladať dáta nezávisle od prijímania, aby mohol prijímať dáta od viacerých používateľov, neblokoval komunikáciu a aby klient nemusel čakať na odpoveď. Server bude vedieť rozlíšiť komunikáciu od rôznych používateľov.

Server bude taktiež podľa geografickej polohy získanej z mobilného telefónu, zistiť adresu na danej geografickej polohe a získať dáta o počasí v danej lokalite.

V neposlednom rade server bude získavať dáta vodivosti kože používateľa namerané meracím zariadením uložené na server tvorcov meracieho zariadenia. Pre túto funkciu využije technológiu Web Socket. Dáta získa vo formáte JSON a budú obsahovať dve merané hodnoty a časovú známku. Získané dáta server pridá do Redis frontu pre ďalšie sprapocanie

Všetky prijímane dáta na serveri budú najskôr uložené do frontu, odkiaľ Komponent spracovania prijatých dát vyberie, skontroluje ich stav a spracuje.

3.2.3 Implementácia

Android aplikácia

V implementácii aplikácie získať dáta o používateľových telefonátoch, SMS správach, udalostiach v kalendári, geografickej polohe a fyzickej aktivite.

Pre získanie geografických súradníc aplikácia využíva Google API a rozhranie LocationListener z Android API, ktoré je špecifické pre Android, a ktoré zavolá udalosť v prípade, ak sa zmení poloha zariadenia.

Dáta z kalendára a dáta o telefonátoch a SMS správach aplikácia získava z vnútornej databázy systému Android, nad ktorou aplikácia vytvára dopyty, pre získanie konkrétnych dát.

Pre získanie dát o fyzickej aktivite používateľa aplikácia implementuje open source algoritmus, ktorý zo zariadenia Accelerometer, ktoré je v každom mobilnom zariadení, rozozná kroky používateľa a tieto kroky počíta. Prvou implementáciou tejto funkcionality bolo vopred naplánované využitie dedikovaného zariadenia pre rozoznávanie krokov, ale pretože toto zariadenie má len malá množina momentálne používaných mobilných telefónov, rozhodli sme sa pre implementáciu spomínaného algoritmu.

Pre zbieranie spomínaných dát aplikácia implementuje *service*, ktorý sa vykonáva na pozadí a pomocou *observerov* sleduje zmenu v zbieraných dátach o telefonátoch a SMS správach. Ak rozozná nové dáta, service vyšle *broadcast* s novými dátami. Rovnako vysiela broadcast s novými dátami aj service, ktorý sleduje zmeny v dátach o geografickej polohe. Iný service, ktorý sa vykonáva na pozadí a slúži na zbieranie dát, prijíma broadcast s novými dátami a zbiera ich. Z tohto servis-u aplikácia po ďalšej implementácii bude vyberať dáta pre ich odoslanie na server pomocou REST API.

Pre posielanie požiadaviek na server sme pri implementácii využili triedu HttpsURLConnection z Android API, využitím ktorej implementujeme synchrónnu aj asynchrónnu komunikáciu. Asynchrónnosť komunikácie je zabezpečená využitím triedy AsyncTask z Android API. Asynchrónna komunikácia podporuje aj komunikáciu s „callback“ funkcionalitou. Všetka komunikácia je šifrovaná použitím TLS protokolu.

Registrácia používateľa je implementovaná tak, že pri spustení aplikácie sa zistí, či je vo vnútornej databáze systému Android uložený registračný token pre túto aplikáciu. Ak token nie je v databáze uložený, tak aplikácia registruje nového používateľa.

Pri implementácií frontend-u, teda obrazoviek, ktoré aplikácia zobrazuje používateľovi, a cez ktoré používateľ aplikáciu ovláda, sú využité *activity* triedy, ktoré agregujú niekoľko *fragment* tried.

Pre implementáciu zobrazenia informácií o používateľovi sme použili voľne dostupnú open source knižnicu MPAndroidChart pre vykresľovanie grafov. Táto knižnica je dostupná na https://github.com/PhilJay/MPAndroidChart.

Server

Pre implementáciu servera sme zvolili jazyk Python, databázy PostgreSQL a Redis. Databáza PostgreSQL slúži na dlhodobé ukladanie dát a Redis slúži na dočasné ukladanie dát hneď po tom, ako ich server prijme cez REST API rozhranie a pridá ich do frontu. Z tohto frontu ich server odoberie, spracuje a uloží do príslušných tabuliek v PostgreSQL databáze. Server sa tak skladá z dvoch komponentov, z ktorých prvý prijíma požiadavky cez REST API, prijíma dáta a vkladá ich do frontu, druhý vyberá dáta z frontu, skontroluje dáta, spracuje ich a uloží, taktiež získava dáta o počasí a adrese na danej lokalite. Komponenty pracujú asynchrónne, aby mobilná aplikácia nemusela čakať na odpoveď zo serveru.

REST API rozhranie je implementované použitím webového rámca Flask. Pre získanie adresy na danej lokalite server využíva Google Maps API a pre sťahovanie dát o počasí v danej lokalite využíva Weather Map API.

Na serveri zatiaľ nezískavame dáta zo zariadenia. Dáta sme získavali mimo nášho produktu. Implementácia tejto funkcionality bude dokončené začiatkom ďalšieho semestra.

Nasadenie zdrojového kódu servera je plne automatizované.

3.2.4 Testovanie

Android aplikácia

Keďže aplikácia je stále prototypom a implementuje málo funkcionality, testovanie nebolo náročné a nemali sme potrebu automatizovaných testov.

Používateľské rozhranie Android aplikácie sme testovali používateľsky. Podľa dohodnutých scenárov používania aplikácie sme simulovali akcie používateľa. Odhalili sme tak pár chýb, kedy aplikácia zlyhala, alebo sa nesprávala tak, ako bolo navrhnuté. Hodnotili sme aj dizajn aplikácie a z hodnotenia vzišlo niekoľko nápadov, ako dizajn obohatiť.

Funkčnosť aplikácie sme testovali debugovaním, sledovaním obsahu objektov a premenných a hodnotením správnosti obsiahnutých hodnôt.

V súčasnosti sme v procese zavádzania automatizovaných testov použitím rámca JUnit.

Server

Funkcionalita serverovej časti aplikácie je vždy testovaná pred nasadením novej verzie na lokálnej inštancii servera sadou predpripravených scenárov. Scenáre simulujú očakávané udalosti a správanie aj chybové požiadavky na server, s ktorými sa musí server vysporiadať.