TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

ITI40LT
Karl-Martin Miidu 131858

HÄIREKESKUSE HÄIREEDASTUSPLATVORMI VEEBI- NING MOBIILIRAKENDUS

Bakalaureusetöö

Juhendaja: Roger Kerse
MSc
Lektor

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Karl-Martin Miidu

08.05.2016

Annotatsioon

Antud lõputöö raames on loodud Häirekeskuse tarbeks häireedastussüsteem, mis koosneb veebirakendusest ja Android rakendusest. Töös annab autor ülevaate veebirakenduse ja Android rakenduse loomise vajalikest aspektidest. Loodud Android rakendus on häire teavitusvahend, millega on võimalik edastada reaalajas videot, heli, seadme GPS asukohta ning eelnevalt rakenduses sisestatud kasutaja andmeid. Androidi rakendus sisaldab kasutajapõhiseid kontosid, kusjuures rakendus eeldab, et kasutaja on enne rakenduse funktsionaalset kasutamist ennast registreerinud, mida on võimalik mobiilirakenduses teha. Loodud veebirakendus on häire vastuvõtmise ning häirete haldamise vahend, kus kasutajal on võimalik mobiilirakendusest edastavale häirele vastata, salvestada ning hilisemalt hallata. Veebirakenduse kasutamine eeldab kasutajakonto olemasolu. Vaatamata sellele vajaksid nii veebirakendus kui ka Android rakendus paremate tulemuste saavutamiseks mõningaid täiendusi, et parandada nende funktsionaalsust ja kasutusmugavust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 8 peatükki, 12 joonist.

Abstract

Development of a web application and Android application for the Estonian Emergency Center signaling platform

Web and mobile applications have become inseparable tools in our everyday lives. Constant improvement in application development have made applications more effective and innovative than ever before thus making mobile phones next to computers essential tools in our everyday lives. However mobile applications are still limited in functionality when it comes to real time solutions. To overcome that obstacle, it is necessary to create a symbiosis between the web application in the web server and the application inside the smart phone.

The outcome of this thesis is a emergency signaling system for the Emergency Center, which consists of a web application and an Android application. The thesis will give an overview of the necessary aspects which should be followed while creating web applications and Android applications. The web application has created a workspace for receiving emergency signals including live video stream, sound and user data with device GPS coordinates. The Android application which was created is a tool for sending emergnecy signals. It allows users to send live video and audio from device with user data. The mobile application requires an account which can be created from the app by anyone. However, both the web application and Android application could need some extra developments for improving their functionality and ease of use in order to achieve better results.

The thesis includes a description and structure of the web application and Android application, thier architectural overview and descriptions of use cases.

The thesis is in Estonian and contains 38 pages of text, 8 chapters, 12 figures.

Lühendite ja mõistete sõnastik

APK	*Android application package,* pakettfaili formaat, mida kasutab Android operatsioonisüsteem programmide levitamiseks ja paigaldamiseks
Android	Google poolt arendatav operatsioonisüsteem, mis on rajatud Linux kernelile ning mõeldud eeskätt puuteekraanidega seadmetele
DoS	Denial-of-Service Attack, teenusetõkestamise rünne, arvutisüsteemi või võrgu vastu suunatud rünnak, mis koormab võrku suure tarbetu liiklusega
GPS	*Global positioning system* – satelliitnavigatsioonisüsteem
HTML	*Hypertext Markup Language* – Hüperteksti märgistuskeel
HTTP	*Hypertext Transfer Protocol* Hüperteksti edastusprotokoll, protokoll andmete edastamiseks arvutivõrkudes
HTTPS	*Hypertext Transfer Protocol Secure* Täiendatud turvalisusega HTTP
iOS	Apple poolt arendatav operatsioonisüsteem mobiilsetele seadmetele
JavaScript	Objektorienteeritud programmeerimiskeel
Linux	UNIX-il baseeruv operatsioonisüsteem
PostgreSQL	Populaarne avatud lähtekoodiga relatsiooniline andmebaasisüsteem
REST	*Representational State Transfer,* tarkvara arhitektuuri stiil, mis koosneb juhenditest ja reeglitest, et koostada skaleeritav veebiteenus
SDK	*Software development kit,* arendustarkvara, mis võimaldab arendajatel programme luua kindlale platvormile
SSL	*Secure Socket Layer* - Turvasoklite kiht
SQL	Relatsioonilise andmebaasisüsteemi päring
SQLLite	Relatsiooniline andmebaasi haldamise süsteem
URL	*Uniform Resource Locator,* internetiaadress, mis vastab igale dokumendile või ressursile internetis

WebRTC	HTML-il ning JavaScript-il põhinev reaalajas andmeedastusplatvorm
XML	*Extensible Markup Language*, W3C poolt väljatöödatud üldotstarbeline märgistuskeel, mille eesmärgiks on info jagamine erinevate infosüsteemide vahel

Sisukord

Jooniste loetelu

1 Sissejuhatus

Veebi- ning mobiilirakendused on tänapäeva ühiskonnas muutunud lahutamatuks osaks nii töö- kui ka eraelus. Rakenduslikud lahendused pakuvad üha enam innovaatilisemaid ning efektiivsemaid lahendusi igapäevaste ülesannete lahendamisel. Tänu sellele on lisaks arvutile muutunud ka telefon hädavalikuks tööriistaks, mis hõlbustab paljusid olmetegevusi. Siiski on mobiilirakenduste võimekus piiratud, kui tegemist on veebipõhise reaalajas tegutseva lahendusega. Kvaliteetse ning kasutajasõbraliku lahenduse loomiseks on tarvis luua kooslus veebiserveris olevast veebirakendusest ning telefonis asuvast rakendusest.

Käesoleva bakalaureusetöö eesmärgiks oli luua Häirekeskusele prototüüp häireedastussüsteemist, mis koosneks veebirakendusest Häirekeskuse töötajale ning mobiilirakendusest tavakasutajale. Prototüübi eesmärk on võimaldada Häirekeskusel parendada ning välja töötada efektiivsemat häireedastussüsteemi. Loodud mobiilirakenduse abil on võimalik häire edastajatel saata reaalajas videot, heli, GPS asukoha koordinaate ning kasutaja andmeid. Loodud veebirakendus realiseerib Häirekeskuse töötaja töökohta ja võimaldab vastu võtta mobiilirakendusest edastatavat häiret koos eelmainitud infoga.

Bakalaureusetöö teises ja kolmandas peatükis annab autor ülevaate veebirakenduste arendamise aspektidest ning tutvustatakse Android rakenduste arhitektuuri. Häirekeskuse jaoks loodud veebirakenduse arhitektuuri ja kasutusjuhendit kirjeldatakse neljandas peatükis ning Android rakenduse ülesehitust, struktuuri ja kasutusjuhtusid viiendas peatükis. Töö kuuendas peatükis selgitab autor käesoleva töö käigus valminud rakenduste arenduskäiku ning vahendeid. Seitsmes peatükk analüüsib loodud veebirakendust ja Android rakendust lähtudes esimesest kasutamise tagasisidest ning kirjeldab süsteemi nõrku kohti.

2 Veebirakendused

Veebirakendus on klient-server mudelil põhinev tarkvararakendus, mis kliendi poolel töötab enamasti veebibrauseris [1]. Veebirakendused on tänapäeva interneti tarbijatel igapäevases kasutuses. Lihtsa ning mugava loomuse poolest kolib üha enam teenuseid just veebirakendustesse, mistõttu on veebirakendused üheks kiiremini arenevaks valdkonnaks IT sektoris. Järgnevalt tutvustatakse lähemalt veebirakenduste eesmärke ja vajalikkust, struktuuri ja turvalisust.

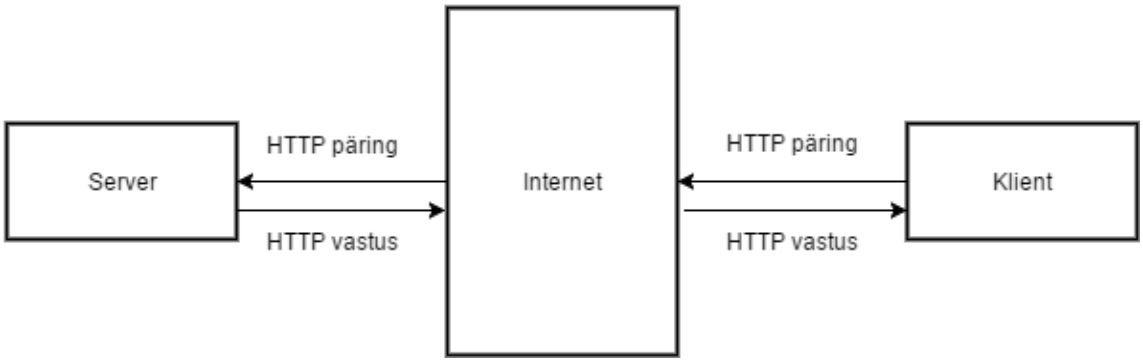
2.1 Veebirakenduste eesmärk ning vajalikkus

Veebirakenduste eesmärk on pakkuda kasutajatele mugavaid ning efektiivseid lahendusi veebitehnoloogia abil [2].

Veebirakendused võimaldavad pakkuda teenust korraga suurele hulgale tarbijatele ilma märkimisväärsete lisakuludeta. Selle tulemusena hoiavad nii kliendid kui ka pakkujad kokku aega ja raha, lihtsustades ning automatiseerides mitmeid tegevusi, mis muidu märkimisväärselt rohkem ressurssi nõuaksid. Efektiivsed ning innovaatilised lahendused on ühiskonna arengus kriitilise tähtsusega, mistõttu on muutunud veebirakendused hädavajalikuks komponendiks igas valdkonnas.

2.2 Veebirakenduste struktuur

Klassikaline veebirakenduse ülesehitus põhineb klient-server mudelil (Joonis 1). Serveris valmistatakse ette dokumendid kasutajale andmete kuvamiseks ning andmete töötlemiseks. Tänu sellele võimaldavad veebirakendused kasutada erinevaid funktsioone ja dünaamilist sisu, rakendades HTTP (Hypertext Transfer Protocol - hüperteksti edastusprotokoll) või turvalist HTTPS (Hypertext Transfer Protocol Secured - hüperteksti turvaline edastusprotokoll) protokolle. Nende abil serveeritakse kliendiprogrammile, üldjuhul brauserile, veebirakenduse sisu, mille kaudu vastavaid teenuseid pakutakse.



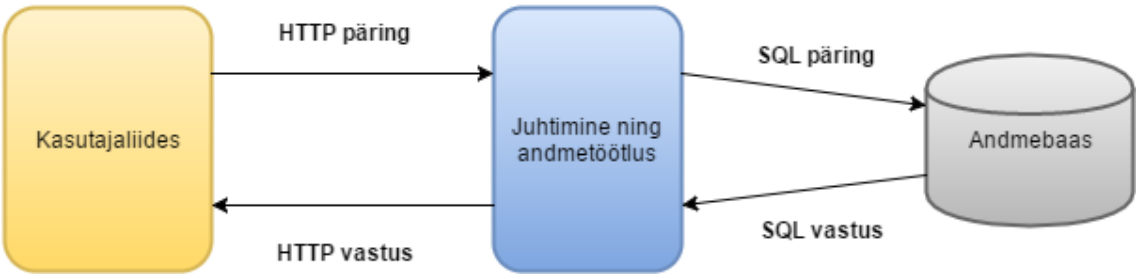
Joonis 1. Klient-server mudel.

Tänapäeva veebirakendusi arendatakse enamasti *framework*-ide ehk raamistike baasil [3]. Raamistikud on valmislahendused, mis võimaldavad arendajal luua veebirakendusi märgatavalt kiiremini, kui ilma raamistikuta. Eeskätt sisaldavad raamistikud valmiskomponente, mis on valitud tehnoloogia puhul hädavajalikud ja mida saab kasutada sageli korduvateks ülesanneteks (nt turvakomponentide jaoks). Ühe veebirakenduse jaoks kasutatakse sageli korraga mitut raamistikku, mis toetavad erinevaid rakenduse elemente, näiteks kasutajaliidese disain, pöördumiste haldamine (Joonis 2), andmeobjektide haldus ja mitut komponenti (nt autentimine, seansihaldus). Tihtipeale eksisteerib sama valdkonna jaoks mitmeid raamistikke, mis võivad olla erineva suunitlusega. Seetõttu tuleb juba planeerimisetapis võtta arvesse nii raamistike väljavalimise kui ka tarkvara arhitektuuriga seonduvaid aspekte.

```
16
17  @RestController
18  @RequestMapping(value = "/", produces = MediaType.APPLICATION_JSON_VALUE)
19  public class ApplicationRestController {
20
```

Joonis 2. Spring raamistikus pöördumise halduse deklaratsioon.

Tavapärase veebirakenduse võib jagada kolmeks võrdse tähtsusega põhikomponendiks: kasutajaliides, andmetöötlus ning juhtimine ja andmebaas (Joonis 3).



Joonis 3. Veebirakenduste komponentide struktuur.

Kasutajaliidese ülesanne on kasutajale visualiseerida rakendusega suhtlemiseks vajalikke elemente. Kuna kasutajaliides on ainus osa rakendusest, mis tavakasutajale ilmneb, on kasutajaliidese ülesehitus kriitilise tähtsusega rakenduse valmistamisel. Intuitiivne ning loogiline kasutajaliides juhendab ning aitab rakenduses vajalikke toiminguid sooritada hõlpsalt ning tulemuslikult, mis jätab kasutajale kvaliteetse mulje kogu rakendusest [4]. Seevastu komplitseeritud ning ebaintuitiivne kasutajaliides põhjustab tihtipeale kasutajale segadust, mille tulemusel võib kasutaja soovitud tegevused jääda rakenduses realiseerimata ning tekitada kasutajas rahulolematust rakendusest.

Andmetöötluse ning juhtimise ülesanne on võtta vastu kasutaja sisend, teha vastavaid andmeoperatsioone, siduda tulem andmebaasiga ning edastada kasutajale vastus. Lisaks käib kogu rakenduse navigatsioon ning muu juhtimine selles komponendis.

Andmebaasi ülesanne on säilitada nii rakenduse andmeid, näiteks konfiguratsioonid kui ka kasutaja sisestatud ning vastavalt töödeldud andmeid. Andmebaasitüüpe on mitmeid, hetkel on populaarsemaks andmebaasisüsteemi tüübiks relatsiooniline andmebaas (Lisa 1).

2.3 Veebirakenduste turvalisus

Veebirakenduste kiire arengu tõttu liigub veebirakenduste kaudu üha enam infot, mistõttu on aina rohkem hakatud tähelepanu pöörama rakenduste turvalisusele [5]. Turvalisuse all mõistetakse ligipääsu nii rakenduse enda infole kui ka rakenduse kasutatavatele andmetele, eeskätt andmebaasile, ükskõik mis kujul või vormis, mis ei ole mõeldud kõrvalistele osapooltele. Veebirakendused võivad sisaldada konfidentsiaalseid andmeid, mistõttu on nende ligipääsetavuse piiramine ning andmekaitse sätestatud ka seadustega ning tihtipeale ka lepingutega.

Turvalisuse tagamiseks kasutatakse rakenduses mitmeid erinevaid komponente ning tehnoloogiaid. Tarkvaraettevõtte Microsofti[1] MSDN (Microsoft Developer Network) väljastatud juhendis „Improving Web Services Security: Scenarios and Implementation Guidance for WCF“ [6] on välja toodud turvalisuse põhialused:

- **Auditeerimine** ehk logimine. Antud turvakomponendi ülesanne on monitoorida ning logida kõik rakenduses toimuvad tähtsamad protsessid. Logi andmete põhjal on võimalik nii ennetada kui ka hilisemalt tuvastada rünnakuid ning nende detaile, mis võimaldab rakenduses vastavad korrektuurid sisse viia.

- **Autentimine** ehk kasutaja tuvastamine. Antud komponendis tuvastatakse kasutaja ning määratakse talle unikaalne identifikaator. Autentimine on üks populaarsemaid turvamehhanisme, mis võib olla realiseeritud klassikaliselt kasutajanime ning parooli kasutamisega või näiteks Mobiil-ID tehnoloogia abil.

- **Autoriseerimine** ehk kasutajate tegevuste piiramine. Antud komponendis tehakse kindlaks, millised operatsioonid ning tegevused on kasutajale lubatud. Tihtipeale kasutatakse rakendustes volitusi, mis võimaldavad hallata ligipääsu veebirakenduse tegevustele (Joonis 4). Tegevusteks võib olla näiteks andmebaasi kirjete lisamine. Paljudes raamistikes on ka vastavad komponendid eelnevalt realiseeritud.

```
@PreAuthorize("hasAuthority('ADMIN')")
@RequestMapping(value = "/archiveList", method = RequestMethod.GET)
public String archiveList(@RequestParam(value = "type", required = false)
```

Joonis 4. Springi raamistikus ressursi piiramine PreAuthorize annotatsiooniga.

- **Krüpteerimine.** Andmete krüpteerimine kujule nii, et andmete konfidentsiaalsus on tagatud ning ligipääs ainult selleks ettenähtud kasutajatele. Krüpteerimise lahendusi on mitmeid, üheks populaarsemaks tehnoloogiaks on SSL protokolli kasutamine kliendi ning rakenduse serveri vahel. SSL protokolli kasutamise eelduseks on SSL-sertifikaadi olemasolu.

[1] https://www.microsoft.com

- **Valideerimine.** Rakenduse sisendite kontroll ning filtreering välistamaks ebakorrektse või pahatahtliku sisendi jõudmist süsteemi. Põhjalik valideerimine eeldab sisendite kontrolli mitmes rakenduse kihis minimeerides võimaliku ohu. Sisendite valideerimine on vajalik näiteks SQL süstimisrünnakute kaitseks.

- **Sessioonihaldus.** Antud komponendi ülesandeks on kasutaja üldine haldus rakenduse külastamise kestel.

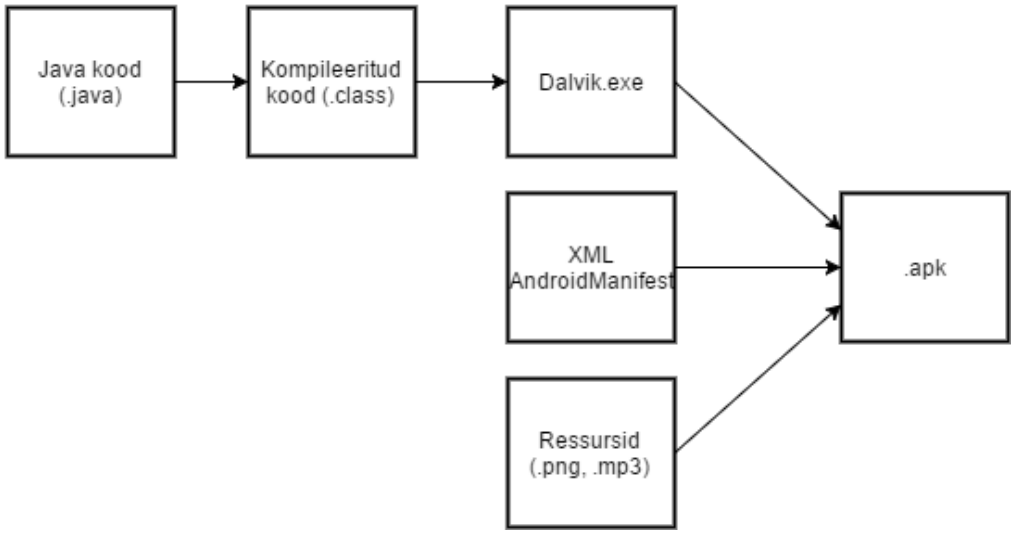
- **Tõrkehaldus.** Süsteemis potentsiaalsete veaolukordade ning teavete haldus ning ennetamine. Tõrkehaldust kasutatakse näiteks DoS rünnakute puhul, kus rakenduse ülekoormuse vältimiseks blokeeritakse rünnakut initsialiseeriv rakenduse klient.

3 Mobiilirakenduse arendamine Android platvormil

Android on Google Inc.[1] poolt loodud operatsioonisüsteem puuteekraaniga seadmetele, eeskätt nutitelefonidele ning tahvelarvutitele. Nutitelefonide turul on Android operatsioonisüsteem suurima osakaaluga maailmas. 2015. aastal teises kvartalis müüdud nutitelefonidest oli 82.20% Android operatsioonisüsteemiga, millest võib järeldada, et Android jääb juhtivaks nutitelefonide operatsioonisüsteemiks ka lähiajal (Lisa 2).

3.1 Android rakenduse struktuur

Androidile loodavad rakendused on loodud Java programmeerimiskeeles. Androidi rakenduste arendamiseks on vajalik Androidi SDK (Software Development Kit - arendustarkvara). Arendustarkvara on vajalik APK (Android application package - pakettfaili formaat) genereerimiseks, mis koosneb rakenduse andmetest (Joonis 5). APK on käivitatav Androidi operatsioonisüsteemis ning võimaldab paigaldada rakenduse seadmesse [7].



Joonis 5. Androidi APK struktuur.

[1] https://www.google.com/about/company/

Iga Androidi seadmel paigaldatud rakendus on omavahel eraldatud *sandbox*-is ehk turvakeskkonnas:

- Androidi operatsioonisüsteem põhineb mitme kasutajakontoga Linuxi süsteemil, millel iga rakendus esindab ühte kasutajakontot.

- Vaikimisi seab operatsioonisüsteem igale rakendusele unikaalse ID, mille väärtust teab ainult süsteem ise. Unikaalse ID abil seadistatakse rakenduse kõik failid nii, et ainult see kasutajakonto, mida konkreetne rakendus esindab, pääseb neile ligi.

- Igal protsessil on oma virtuaalne masin, mis teistest eraldatuna rakenduse koodi käivitab.

- Vaikimisi jooksevad kõik rakendused eraldi Linux protsessides. Android käivitab protsessi kui mõni rakenduse komponent käivitatakse ning sulgeb protsessi, kui rakendus enam ressursse ei vaja või siis, kui on vaja mälu juurde teiste rakenduste tegevuseks.

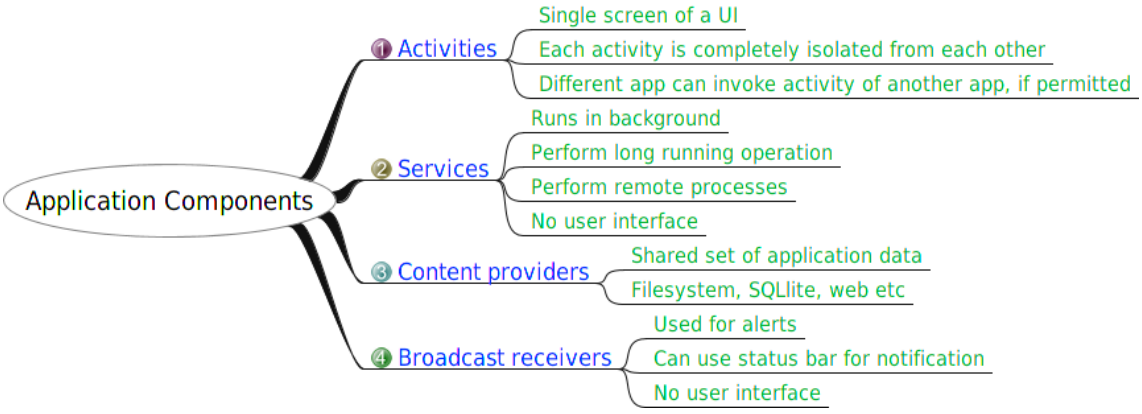
3.2 Android rakenduse komponendid

Android rakenduse komponendid on olulised rakenduse elemendid. Igal komponendil on erinev ligipääsupunkt, mille kaudu käib operatsioonisüsteemiga suhtlus. Samuti on igal komponendil erinev eesmärk ning elutsükkel [8]. Kokku eksisteerib neli erinevat tüüpi komponente (Joonis 6):

- ***Activities*** – *Activity* ehk tegevus iseloomustab ühte vaadet kasutajaliidesega. Näiteks emaili rakendusel võib olla üks tegevus emailide nimekirja jaoks ja teine ühe konkreetse emaili lugemiseks. Vaatamata sellele, et kõik tegevused töötavad koos, on igaüks neist iseseisev üksus. See võimaldab liidestada erinevaid rakendusi, lubades näiteks teisel rakendusel ühte konkreetset tegevust käivitada. Nii võib eeltoodud näite puhul veebilehitseja rakendus avada näiteks emaili rakenduse emaili saatmise tegevuse. Iga *activity* on alamklass klassile `Activity`.
- ***Services*** – *Service* ehk teenus on komponent, mis töötab rakenduse taustal ja viib läbi pikaajalisi ülesandeid või töötab kaugtöödega. Teenusel ei ole kasutajaliidest. Näiteks võib teenus andmeid laadida alla samal ajal kui kasutaja on teises rakenduses, ilma et kasutaja rakenduse tegevust peaks häirima. Iga *service* on

alamklass klassile `Service`.

- ***Content providers*** – *Content provider* ehk sisuhaldur haldab jagatud osa rakenduse andmetest. Android rakenduses on andmeid võimalik salvestada failides, SQLLite andmebaasis, veebis või mingis muus andmete hoiustamiseks mõeldud asukohas, kus rakendusel on ligipääs. Sisuhalduri abil on võimalik ka teistel rakendustel nendele andmetele ligipääs anda – selleks on rakendusel vaja vastavat õigust. Näiteks võib rakendus küsida andmeid kasutaja kontaktiraamatus konkreetse isiku kohta. Iga sisuhaldur on alamklass klassile `ContentProvider`.
- ***Broadcast receivers*** – *Broadcast receiver* ehk rakendusteülene vastuvõtja on komponent, mis reageerib ülesüsteemsetele teavitustele. Mitmeid ülesüsteemseid teavitusi saadab operatsioonisüsteem ise, näiteks aku tühjenemise teavitus. Ka rakendused ise saavad ülesüsteemseid teavitusi saata. Üheks väljundiks on näiteks teiste rakenduste teavitamine äsja allalaetud tarkvara kasutamise võimalusest. Teine võimalus on saata teavitusi teavituspaneelile, mille kaudu on võimalik ka kasutajat teavitada. Rakendusteülene vastuvõtja on alamklass klassile `BroadcastReceiver`.



Joonis 6. Android rakenduse komponendid.[10]

4 Häireedastusplatvormi veebirakendus

4.1 Veebirakenduse tutvustus

Antud töö käigus autori poolt loodud veebirakendus oli Häirekeskuse katsetatava häireedastusplatvormi prototüüplahenduse üks osa. Rakendus realiseerib Häirekeskuse häire vastuvõtja töökohta. Kuna loodava platvormi prototüübi eesmärk oli luua võimalus edastada reaalajas videot, heli ning kasutaja andmeid, siis hetkel eksisteerivast häireedastussüsteemist realiseeriti vaid osa funktsionaalsusest. Veebirakendus võimaldab sooritada järgmised tegevusi:

- Kasutaja autentimine
- Viimase 24 tunni häirete ülevaate kuvamine graafikuna eraldi Häirekeskuse üksuste lõikes kui ka üleüldiselt kõigi häirete üleselt
- Häire vastuvõtmine. Sissetulevat signaali on võimalik vastu võtta ning kuvada videot koos edastatud andmetega. Lisaks videole ning helile edastatakse ka kasutaja andmed ning GPS koordinaadid. GPS koordinaatide põhjal kuvatakse kasutajale kaart signaali asukohast ning asukoha aadress
- Häire salvestamine
- Salvestatud häirete nimekirja kuvamine
- Salvestatud häirete vaatamine
- Salvestatud häirete muutmine
- Kasutaja andmete muutmine
- Väljalogimine

4.2 Veebirakenduses kasutatud tehnoloogiad

Rakenduse loomiseks kasutati Java platvormil põhinevat Spring raamistikku. Spring on laialdaselt levinud ning äärmiselt mitmekülgse funktsionaalsusega raamistik, mille üheks suunitluseks on veebirakendused. Lisaks veebirakendustele võimaldab Spring hõlpsalt

ehitada ka veebiteenuseid, mis oli ehitatava platvormi üheks võtmekomponendiks eeskätt mobiilirakendusega suhtlemiseks. Lisaks omab Springi raamistiku arhitektuur väga head skaleeritavust ning sisaldab mitmeid valmiskomponente, mis antud veebirakenduse valmistamiseks tarvilikud olid, näiteks autentimismoodul ning seansihaldus. Java programmeerimiskeele ning Spring raamistiku kasutamine eeldab Java olemasolu, mis on antud tehnoloogia puhul negatiivseks küljeks. Samuti võib negatiivseks lugeda rakenduse kompileerimiseks minevat ajakulu, mis aeglustas arendusprotsessi. Vaatamata sellele oli Springi raamistiku valik antud rakenduse tarbeks sobilik.

Kuna veebirakenduse andmed on kasutajate vahel jagatud, kasutati andmete salvestamiseks tsentraalset relatsioonilist PostgreSQL andmebaasisüsteemi. Spring raamistikus andmeobjektide haldamiseks andmebaasiga kasutati Hibernate raamistikku. Hibernate on ORM (*object-relational mapping* – objekt relatsiooniline kaardistus) raamistik Java programmeerimiskeeles, mis võimaldab kerge vaevaga sisestada, lugeda ning muuta andmebaasikirjeid andmeobjektide kontekstis. Rakenduse kasutajaliidese veebilehtede ehituseks kasutati Thymeleaf raamistikku. Lisaks neile on rakenduse kasutajaliideses kasutusel jQuery raamistik koos AJAX (*Asynchronous JavaScript And XML* – asünkroonne JavaScript ja XML) tehnoloogiaga. Kujunduse raamistikuks otsustus sobivaks Bootstrap 3. Lisaks kasutati häire asukoha kuvamiseks veebirakenduses Google Maps integratsiooni.

4.3 Veebirakenduse arhitektuur

Loodud veebirakendus kasutab AJAX tehnoloogiat kasutaja ning rakenduse vahelise andmevahetuse tarbeks. Spring raamistikus on loodud vastavad ressursid, et AJAX-i abil tagastada andmeid. Vastavalt ressursile edastatakse vajalikud parameetrid POST argumentidega.

Vaikimisi eeldavad kõik rakenduse pöördumisressursid, et kasutaja on ennast süsteemis autentinud, välja arvatud */login* ehk autentimispöördumine ning */register* ehk registreerimispöördumine (Joonis 7). Kuna nii veebirakenduse klient kui ka mobiilirakendus kasutab andmevahetuseks samu veebirakenduse ressursse, on rakenduses kasutuses Spring Security kasutajarollide funktsionaalsus, mis võimaldab vastavalt kasutajarollile piirata ligipääsu valitud ressurssidele. Lisaks on kasutusel Spring

raamistiku seansihaldus, mis võimaldab kasutajal veebiteenust kasutada nii, et seansi piires toimub autentimine vaid esimesel päringul /login ressursile, peale mida on kasutaja autentitud ning edasistes andmevahetustes kasutajat eraldi autentida ei ole vaja.

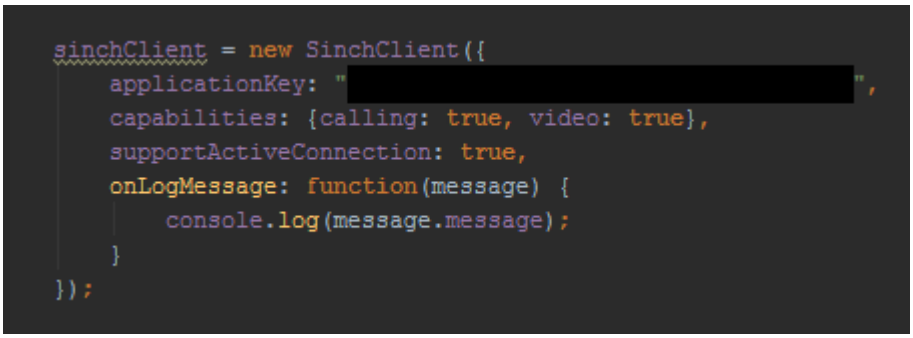
```
@Override
protected void configure(HttpSecurity http) throws Exception {

    http
        .authorizeRequests()
            .antMatchers("/register","/css/**", "/js/**", "/fonts/**", "/images/**").permitAll()
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .permitAll()
            .and()
        .logout()
            .permitAll();
```

Joonis 7. Spring Security seadistamine.

Veebirakenduse üheks põhikomponendiks on funktsionaalsus, mis võimaldab Android rakendusest edastatavat signaali vastu võtta ning kasutajale kuvada. Selleks on veebirakenduses kasutuses tarkvaraarendusettevõtte Sinch[1] poolt arendatav teek Sinch.js. Sinch.js abil on võimalik veebirakenduse kasutaja brauserisse edastada Android rakendusest edastatavat videot ning heli. Lisaks sellele saab signaalis edastada ka tekstandmeid, näiteks mobiilirakenduse kasutaja andmeid ning seadme GPS koordinaate. Lahenduse toimimiseks peab ka Android rakendus kasutama Sinch-i poolt välja töötatud teeki.

Sinch platvormil signaalide vastuvõtmiseks tuleb kliendi veebibrauseris initsialiseerida Sinch klient, mis põhineb `SinchClient` klassil. Kuna Sinch.js teek võimaldab kasutada võrdlemisi suurt funktsionaalsust, on Sinch kliendi initsialiseerimisel tarvilik edastada vajalikud parameetrid kliendi funktsionaalsuse täpsustamiseks (Joonis 8). Käesoleva töö käigus loodud veebirakenduse kliendil on esmatähtsateks võimalusteks video ning heli vastuvõtmine. Sinch platvorm võimaldab ka veebibrauseri kliendil edastada video- ja helisignaali, kuid antud rakenduse raames seda funktsionaalsust ei realiseeritud.

[1] https://www.sinch.com/



Joonis 8. Sinch kliendi initsialiseerimine JavaScriptis.

Loodud veebirakenduses käivitatakse Sinch klient peale õnnestunud kasutaja autentimist. Platvormi kasutamine rakenduses eeldab eelnevalt rakenduse registreerimist Sinch-i kodulehel https://www.sinch.com/

5 Häireedastusplatvormi Android rakendus

5.1 Android rakenduse tutvustus ja eesmärk

Käesoleva töö raames valminud häireedastusplatvormi Android rakenduse prototüübi eesmärk on pakkuda võimalust edastada häiresignaali koos reaalajas video, heli ning kasutaja andmetega, et parendada Häirekeskuse senist häire vastuvõtmise protsessi ning võimaldada kasutada informatsiooni, mis seni polnud võimalik. Rakendus on mõeldud kasutamiseks Eestis viibivatele inimestele. Rakendus on loodud inglisekeelsena.

Android rakendus sisaldab järgmisi funktsioone:

- Kasutaja registreerimine
- Kasutaja autentimine
- Kasutaja andmete muutmine
- Häiresignaali edastamine koos reaalajas video, heli ning kasutaja andmete koos seadme GPS koordinaatidega
- Väljalogimine

Rakendus on valmistatud töötama versioonil Android 5.0-5.1 „Lollipop“ ning 6.0 „Marshmallow“.

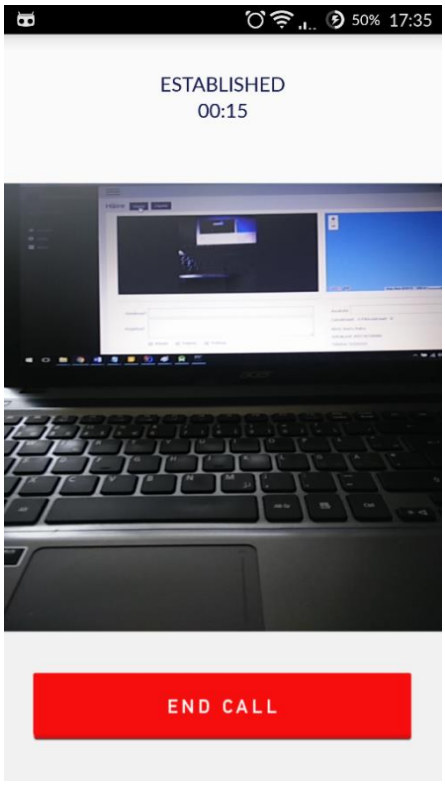
5.2 Android rakenduse kasutusjuhud

Loodud rakenduse peamine eesmärk on luua võimalus kasutajale häireedastusel kasutada reaalajas video, heli ning kasutaja andmeid.

Hädajuhtumist teatamiseks tuleb eelnevalt süsteemis end registreerida. Registreerimiseks tuleb avada rakenduse avaleht ning vajutada nuppu „Register“. Seejärel suunatakse kasutaja registreerimisvormi vaatesse, kus kasutaja sisestab nime, isikukoodi,

telefoninumbri ning kasutajanime ja salasõna. Andmete salvestamisel kuvatakse vastav teade ning suunatakse kasutaja tagasi avalehele. Registreerinud kasutajad saavad rakenduse avalehe kaudu ennast süsteemile tuvastada sisestades kasutajanime ning salasõna.

Süsteemile tuvastatud kasutajale avaneb põhivaade. Hädajuhtumi korral tuleb kasutajal vajutada „Häire“ nuppu, mille peale avaneb kasutajale häireedastusvaade, kus alustatakse ühenduse loomist platvormi veebikeskkonnaga. Ühenduse loomise ajal annab rakendus kasutajale tagasisidet helisignaali abil. Kui ühendus on loodud, käivitatakse seadme tagakaamera ning alustatakse video ning heli reaalajas edastamist. Ühenduse loomise hetkel edastatakse ka seadme GPS koordinaadid ja kasutaja andmed. Õnnestunud ühenduse loomisel kuvatakse edastatavat videopilti ka häireedastusvaates (Joonis 9).

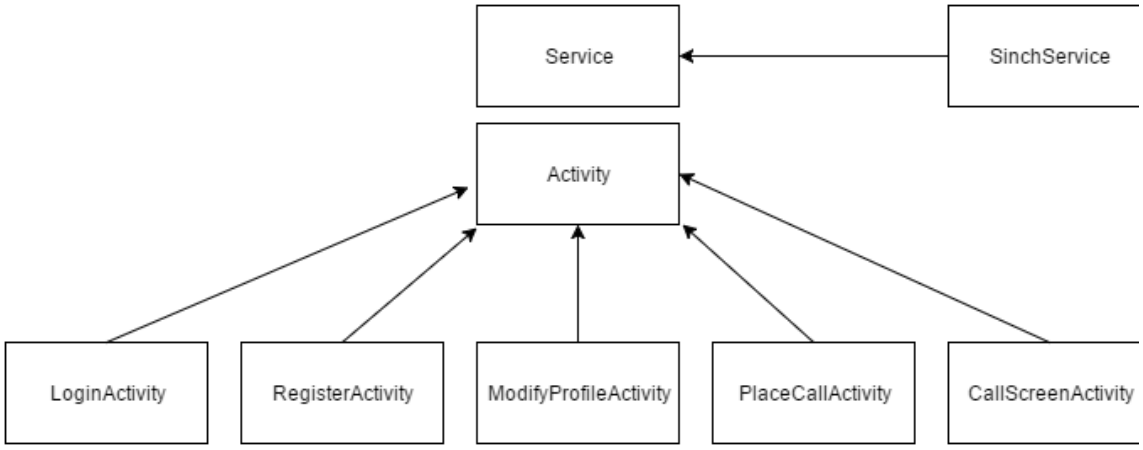


Joonis 9. Android rakenduse häireedastusvaade.

5.3 Android rakenduse struktuur

Käesolevas töös loodud mobiilirakenduse siht Android versioon on 5.1 ning 6.0. Rakendus on arendatud ning testitud nii 5.1 kui ka 6.0 versioonil, kuid ei sisalda komponente, mis ei tööta versioonil 4.2 või uuemal. Rakendus sisaldab endas viit *Activity*-it ning ühte *Service*-t ehk teenust (Joonis 10). Lisaks on rakenduses eraldi loodud

klass `EmergencyRESTClient`, mis sisaldab veebirakendusega suhtlemiseks vajalikke meetodeid ning parameetreid ja mille kasutamine eeldab internetiühenduse olemasolu.



Joonis 10. Android rakenduse struktuur.

`LoginActivity` on rakenduse esmane *activity*, mis avaneb rakenduse käivitamisel. `LoginActivity` sisaldab kasutaja sisselogimisvaadet, kus kasutajal on võimalik ennast autentida või konto puudumisel liikuda registreerimise vaatesse. Registreerimine toimub `RegisterActivity`-s, kus kasutaja sisestab konto loomiseks vajalikud andmed, mille põhjal luuakse kasutajale uus konto. Õnnestunud konto loomisel suunatakse kasutaja tagasi `LoginActivity`-sse. Autentimine toimub samuti kasutades `EmergencyRESTClient`-it ning teostatakse asünkroonselt, kasutades klassi, mis laiendab `AsyncTask` klassi. Nii ei hangu kogu *activity* autentimise ajaks ära ning on endiselt interaktiivne. Lisaks initsialiseeritakse sisselogimisel ka `SinchClient` klassi objekt, mille abil on hilisemalt võimalik häiret edastada. Nii sisselogimisel kui ka muudel veebirakendusega andmevahetuses olevate tegevuste ajal kasutatakse Android süsteemi `ProcessDialog`-i, mis annab süsteemi hetkeolekust kasutajale tagasisidet. Sisselogimisel avaneb kasutajale `PlaceCallActivity`, mille kaudu võib alustada häire edastamist või profiili muutmist. Profiili muutmisel avaneb `ModifyProfileActivity`, mis sarnaselt registreerimistegevusele sisaldab kasutaja andmevälju ning võimaldab kasutaja andmeid muuta. Häire edastamisega tegeleb `CallScreenActivity`. Enne antud tegevuse avamist küsitakse seadme GPS koordinaadid ning initsialiseeritakse videoühendus koos GPS koordinaatidega ning kasutaja andmetega. Kogu häireedastussignaali edastusega tegeleb `SinchService`. Seejärel avaneb `CallScreenActivity`, mis esmalt kuni signaali vastuvõtmiseni või lõpetamiseni

imiteerib kõne kutsumist helifaili abil. Kui signaalile ei vastata või keeldutakse, suunatakse kasutaja tagasi eelmisesse tegevusse. Signaali vastuvõtmise korral kuvatakse kasutajale veebirakendusest edastatavat videopilti. Kaamera käivitamiseks ning video initsialiseerimiseks kasutatakse `SinchService` `VideoController`-it. Häire lõpetamisel suunatakse kasutaja tagasi `PlaceCallActivity`-sse.

6 Häireedastusplatvormi prototüübi arendus

6.1 Prototüübi arenduskäik

Platvormi arendus toimus järk-järgult täiendades paralleelselt nii veebirakendust kui ka Android rakendust. Platvormi põhiarendus kestis 4 nädalat.

Autor alustas platvormi arendust antud töö ühest olulisemast funktsionaalsusest ehk video edastusest Android rakendusest veebikeskkonda. Põhjaliku uurimise ning katsetuste tulemusel otsustas käesoleva töö autor kasutada WebRTC tehnoloogial põhinevat Sinch platvormi. Prototüübi arendust alustas platvormi looja veebirakenduse loomisest. Esmalt seadistas töö autor Spring raamistiku ning lõi esmase häire vastuvõtmisvaate Sinch.js teegi abil. Seejärel alustati Android rakenduse arendusega, kus kasutati Sinchi poolt arendatud SDK-d. Seejärel tegi autor vajalikud konfiguratsioonid ning suutis tulemuslikult luua video ja heli ühenduse mobiilirakenduse ning veebirakenduse vahel. Peale rakendustevahelise videoedastuse väljaarendamist alustas autor andmebaasi väljatöötamist. Autori arvamusel osutus sobivaks andmebaasisüsteemiks PostgreSQL. Loodud andmebaasiskeemile alustas platvormi looja veebirakenduse muu funktsionaalsuse arendamist eeskätt häire salvestamist. Lisaks häire salvestamisele alustas autor ka Spring Security autentimismooduli rakendamist nii veebikeskkonnas kui ka Android rakendusest kasutatavates moodulites.

Järgmisena arendas autor veebirakenduses salvestatud häirete halduse mooduli ning kasutaja andmete halduse. Seejärel täiendas autor ka Android rakenduses kasutaja andmete muutmise funktsionaalsust ning implementeeris kasutajate registreerimise.

Järgnevalt täiendas autor häireedastuse funktsionaalsust, kus Android rakenduse poolel lisas häireedastusele seadme GPS koordinaadid ning kasutaja andmed. Veebirakenduse poolel lisas autor Google Maps integratsiooni, mille abil häireedastussignaali andmetest sai häire edastaja asukoha kaardil kuvada. Lisaks asukohale pandi kuvama ka häire edastaja andmed.

Andmebaasi struktuuri täiendas autor jooksvalt erinevate funktsionaalsuste arendamisel.

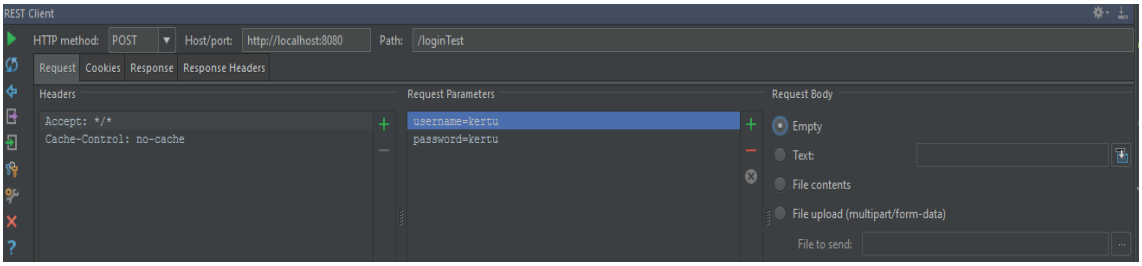
Viimasena viimistles autor veebi- ning Android rakenduse kasutajaliidest.

6.2 Prototüübi arendamisel kasutatud tööriistad

6.2.1 IntelliJ IDEA[1]

Intellij IDEA on tarkvaraettevõtte JetBrains[2]-i poolt loodud tarkvaraarendusprogramm, mis suunatud põhiliselt Java programmeerimiskeeles kodeerimiseks. Esimene Intellij IDEA versioon avalikustati 2001. aasta jaanuaris ning oli üks esimesi tarkvaraarendusprogramme, mis sisaldas arenenud koodi refaktoreerimis- ning navigeerimisvõimalusi. Intellij IDEA põhjal on loodud mitmeid teisi arenduskeskkondi muude programmeerimiskeelte suunitlusega, näiteks PhpStorm[3]. 2014. aastal avaldas Google Android Studio[4] 1.0 versiooni, mis baseerus samuti Intellij IDEA-l.

Tänapäeva tarkvaraarenduseks tarvilik arenduskeskkond sisaldab enamasti mitut tööriista. Intellij oluliseks eeliseks on mitmed erinevad lahendused, mis automatiseerivad nende vahendite kasutamist, võimaldades programmeerijal hoida aega kokku. Näiteks on võimalik loodavad veebiteenust otse arenduskeskkonnas testida sisseehitatud veebiteenuse kliendi abil (Joonis 11).



Joonis 11. Intellij IDEA sisseehitatud REST klient.

[1] https://www.jetbrains.com/idea/

[2] https://www.jetbrains.com/

[3] https://www.jetbrains.com/phpstorm/

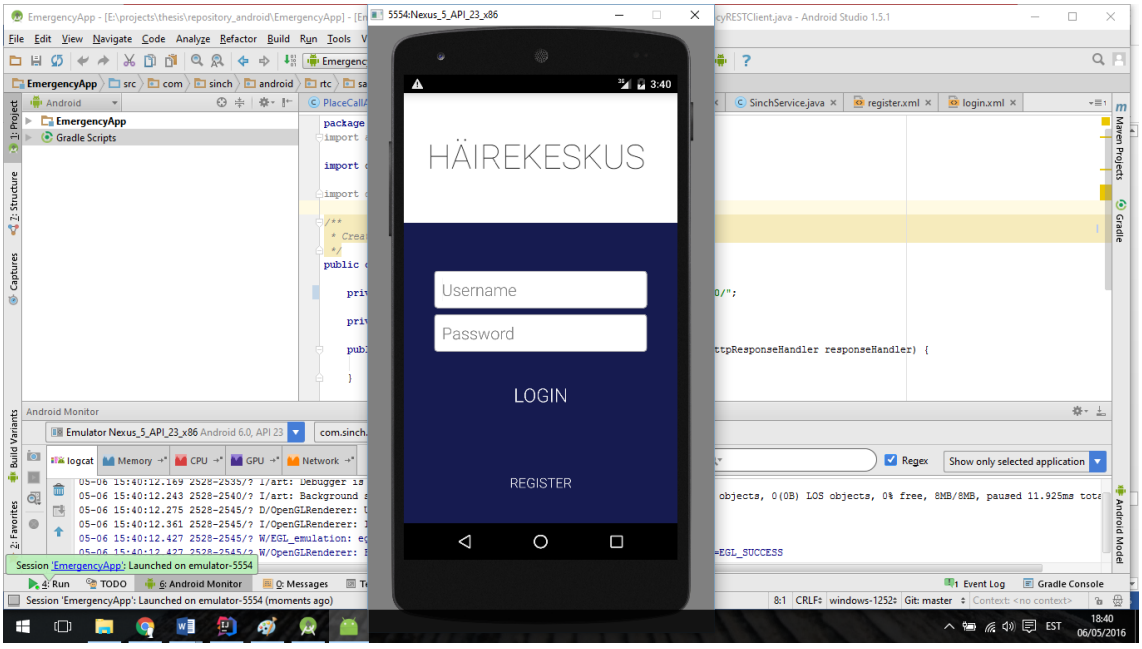
[4] http://developer.android.com/sdk/index.html

Lisaks on loodud mitmeid koodikirjutamist abistavaid lahendusi, näiteks automaatne fraaside lõpetamine, klasside- ja meetoditevaheline navigeerimine ning koodi refaktoreerimisvahendid.

Käesolevas töös kasutati IntelliJ IDEA 15.0.4 versiooni platvormi veebirakenduse loomiseks.

6.2.2 Android Studio

Android Studio on 2013. aastal Google I/O-l[1] välja kuulutatud arendustööriist. Android Studio on alates 2015. aastast soovitatud tööriist Android rakenduste loomiseks ning on konkreetselt Android jaoks loodud ja kohandatud. Kasutajaliides sarnaneb kõikide teiste JetBrainsi toodetega, nagu näiteks Intellij IDEA. Android Studio omab mitmeid võimalusi sarnaselt Intellij IDEA-le, kuid oluliselt on parendatud Android koodi refaktoreerimist ja automaatset koodi lõpetamist. Lisaks omab Android Studio mitmeid Androidi arendust hõlbustavaid tööriistu, näiteks XML vaadete kuvamine ning muutmine lohista-ja-paiguta põhimõttel. Lisaks on sisse ehitatud ka Androidi nutitelefoni emulaator, mis võimaldab arendatavat programmi virtuaalmasinas käivitada (Joonis 12).



Joonis 12. Android Studio seadme emulaator.

[1] https://en.wikipedia.org/wiki/Google_I/O

Käesolevas töös kasutati Android Studio 1.5.1 versiooni platvormi Android rakenduse loomiseks.

6.2.3 Git[1]

Git on 2005. aastal loodud versioonihaldustarkvara, mis loodi algselt Linus Torvaldsi poolt Linuxi tuuma arendamiseks. Tänaseks on Git kasvanud üheks populaarsemaks versioonihaldustarkvaraks tarkvaraprojektide seas [9]. Erinevalt enamus klient-server versioonihaldustarkvaradest põhineb Git hajutatuse põhimõttel, kus iga versioonihaldusest osavõttev instants omab täielikku ajalugu ning versiooni jälgimisvõimalusi, olenemata võrguühenduse ligipääsust või kesksest serverist.

Käesolevas töös kasutati Git 2.6.0 versiooni nii veebirakenduse kui ka mobiilirakenduse arendamisel.

[1] https://git-scm.com/

7 Häireedastusplatvormi analüüs

7.1 Platvormi rakendamine ja tagasiside kasutajatelt

Platvormi kasutuselevõtmiseks on mobiilirakenduses loodud kasutajate registreerimise võimalus, et kasutajad looksid endale uue kasutajakonto, millega oleks võimalik rakendust kasutada.

Häireedastusplatvorm ise koosneb kolmest eraldiseisvast komponendist – mobiilirakendus, veebirakendus ning andmebaas. Andmebaasi saab lihtsalt ja kiirelt luua kasutades arenduskeskkonna koopiat – selleks võib kasutada näiteks andmebaasi koopiafaili, mis sisaldab nii andmebaasi struktuuri kui ka andmeid. Lisaks on ka defineeritud vastava andmebaasi kasutaja õigused. Antud töös kasutatakse andmebaasi ning veebirakenduse vaheliseks ühenduseks ühte kasutajakontot.

Veebirakenduse käivitamiseks serveris on eelnevalt vaja paigaldada Java 8. Veebirakenduse ja andmebaasi vahel on ühenduse loomiseks tarvis veebirakenduses andmebaasi rekvisiidid seadistada. Seadistus tuleb teha `AuthenticationProviderConfig` klassis. Peale seda on vajalik rakendus uuesti kompileerida.

Mobiilirakendus kasutab andmeoperatsioonideks veebirakenduse teenuseid, mistõttu eeldab mobiilirakendus interneti olemasolu. Lisaks on mobiilirakenduses vajalik seadistada veebirakenduse serveri aadress `EmergencyRESTClient` klassis. Kuna loodud rakendus on momendil ainult Häirekeskusele suunatud, siis ei ole võimalik lisada rakendust Google Play rakenduste poodi. Selle asemel tuleb rakenduse paigaldamiseks ettenähtud .apk fail manuaalselt telefoni lisada ning seejärel käivitada.

Rakenduse funktsionaalsust ja kasutajamugavust testis Häirekeskuse arendusosakonna planeerimistalituse ekspert Madle Puusepp. Puusepp tõi välja, et mobiilirakenduse kasutamise puhul on suureks abiks video ning GPS koordinaatide edastus, mis aitaks abi

väljasaatmist kiirendada. Puusepp lisab, et rakenduse positiivseks küljeks on kiirus – teate edastamiseks piisab ühest nupuvajutusest. Samas mainib Puusepp, et rakenduse kasutamise puhul esineb mitmeid nüansse: „Hädaabiteate edastamine peab olema võimalikult kiire. Eestis on nii kiirabi, pääste kui ka politsei kutsumiseks vaja meeles pidada vaid ühte numbrit - 112. Kriitilises situatsioonis ei pruugi abivajajal muud alternatiivid meelde tulla, mistõttu võib taoline rakendus pigem olla efektiivsem vähem aegkriitiliste sündmuste kirjeldamisel. Samuti seab rakenduse kasutamisele teatava piirangu nutiseadmeid kasutava sihtgrupi suurus ja rakenduse allalaadinud inimeste arv.“ Puusepp toob välja ka järgmise: „Rakenduse kõige suurem väärtus võib peituda vähem aegkriitiliste või pikemaaegsete sündmuste lahendamises. Näiteks aidates määrata ära kulupõlengu suurust ja suunda, selgitada välja veeuputuste mahtu jne. Lisaks võib videokõne olla abiks meedikule, kes konsulteerib enne kiirabi kohale jõudmist vigastatud inimeste sündmuskohal abistamisel.“ Viimaseks mainib Puusepp, et käesoleva lõputöö käigus valminud platvorm on Häirekeskusele ja üldiselt sisejulgeoleku valdkonnast lähtuvalt väga tänuväärne.

7.2 Platvormi analüüs ning puudused

Käesoleva töö käigus loodud häireedastuse platvormi üks nõudmistest oli kasutajatepõhine tegevus. Selle realiseerimiseks oli vaja luua nii veebirakenduses kui ka mobiilirakenduses sisselogimismoodul. Selleks kasutati Spring Security moodulit, mis sisaldab mitmeid kasutajate halduseks vajalikke vahendeid. Rakenduse kontekstis realiseeriti kasutajate sisselogimine üheselt, eristades mobiilirakenduse kasutajaid veebirakenduste kasutajatest rolli põhjal. Mobiilirakenduse turvalisuse kontekstis eraldab see mobiilirakenduse ja kasutajate autentimise, mis vähendab turvariske, kuna mobiilirakendus ise ei oma andmeid kasutajate kohta. Kontroll ning kasutaja andmete pärimine toimub eraldiseisvas serveris. Lisaks võimaldab see loodud autentimismoodulit kasutada ka potentsiaalsetes tulevastes rakendustes, mis sisaldavad endas loodud Häirekeskuse prototüübi platvormi kasutajakontosid.

Häirekeskuse platvormi veebirakendus on loodud realiseerima Häirekeskuse töötaja töökohta. Lisaks võimaldab rakendus suhelda Android rakendusest vajalikke andmebaasioperatsioone teostada. Häirekeskuse töötaja töökoha realiseerimisel oli üheks põhifunktsionaalsuseks mobiilirakendusest edastava reaalajas video, heli ning kasutaja

andmete vastuvõtmine. Lisaks võimaldab rakendus häire andmeid salvestada. Hetkel kuulub salvestamisele häire edastanud kasutaja seos, häire asukoht, pealkiri, kirjeldus ning valitud brigaadide tüübid. Paraku ei ole hetkel salvestamisel realiseeritud video ning heli salvestamist, mis tähendab, et arhiivis olevate häiretel videot ning heli taasesitada ei ole võimalik. Veebirakenduse arhiivis olevate häirete puhul ei ole loodud otsingut, kuid nõutud prototüübi lahenduses piisab ka nimekirjapõhisest arhiivi otsimisest.

Android rakendus Häirekeskuse platvormi üks osa, mis realiseerib tavakasutaja töökohta. Rakenduse peamine eesmärk on võimaldada kasutajal edastada reaalajas video, heli, kasutaja andmeid ning seadme GPS asukohta. Tänu Androidi operatsioonisüsteemi paindlikkusele GPS asukohapärimise kontekstis ning Sinch SDK teegile on see võimalik. Lisafunktsioonina on võimalik kasutaja andmeid muuta. Rakenduse üks puudustest on see, et GPS asukoha määramine võib olevalt asukohast olla ebatäpne. Lisaks võib puuduseks lugeda mõistliku internetiühenduse olemasolu, mida reaalajas videoedastus nõuab. Internetiühenduse kvaliteedi langemisel langeb ka edastatava video ning heli kvaliteet, mis võib kokkuvõttes häireedastuses infot vähem edasi anda kui traditsiooniline telefonikõne. Lisaks nõuab kvaliteetne videoedastus Android seadme kaamera keskkonnas valgust, et video kvaliteet oleks kasutatav. See tähendab, et pimedas võib video edastus kehva kvaliteedi tõttu puudulikuks jääda.

8 Kokkuvõte

Käesoleva tööga loodi Häirekeskusele reaalajas video, heli, seadme asukohaandmete ning kasutaja andmete häireedastussüsteem ning sellega täidab püstitatud ülesannet, milleks oli luua Häirekeskusele häireedastuse efektiivsemaks muutmise uute tehnoloogiate katsetamiseks prototüüp. Töö esimeses pooles anti ülevaade nii veebirakenduste kui ka Android rakenduste loomisest, millele lähtuvalt sai loodud häireedastus platvormi veebirakendus ja mobiilirakendus.

Loodud veebirakendus realiseerib Häirekeskuse töötaja töökohta ning võimaldab vastu võtta mobiilirakendusest sissetulevaid häired, neid salvestada ning hallata. Android rakendus realiseerib tavakasutaja töökohta, mille kasutamiseks on vajalik kasutajakonto, mida on võimalik rakenduses luua. Peale reaalaajas video ning muu info edastusega, on võimalik rakenduses ka kasutaja andmeid muuta. Siiski oleksid vajalikud nii veebirakenduses kui ka mobiilirakenduses mõningased täiendused, et parandada mõlema komponendi funktsionaalsust ja kasutusmugavust. Veebirakenduse puhul tuleks täielikuma lahenduse loomiseks lisada ka Häirekeskuse äriloogika häireedastuse küsimustiku osas. Paraku nõuab selle realiseerimine suurt töömahtu, mistõttu käesolevas töös antud funktsionaalsust ei realiseeritud. Mobiilirakenduse poolelt peaks lahendust laiendama ka iOS operatsioonisüsteemiga seadmetele.

Häireedastusplatvorm, mis koosneb veebirakendusest ja Android rakendusest täidab neile seatud eesmärke ning parendab oluliselt Häirekeskuse häireedastuse uute lahenduste väljatöötamist.

Kasutatud kirjandus

[1] What is a web application (or a „webapp“)? [WWW] http://www.jguru.com/faq/view.jsp?EID=129328 Alex Chaffee (23.04.2016)

[2] Web applications [WWW] http://webtrends.about.com/od/webapplications/a/web_application.htm Daniel Nation (23.04.2016)

[3] What is a framework? [WWW] https://jeffknupp.com/blog/2014/03/03/what-is-a-web-framework Jeff Knupp (25.04.2016)

[4] In Defence of Eye Candy [WWW] http://alistapart.com/article/indefenseofeyecandy Stephen P. Anderson (27.04.2016)

[5] Web Application Security Statistics [WWW] http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics Sergey Gordeychik (30.04.2016)

[6] Improving Web Services Security: Scenarios and Implementation Guidance for WCF [WWW] https://msdn.microsoft.com/en-us/library/ff650794.aspx / J.D. Meier, C. Farre, J. Taylor, P. Bansode, S. Gregersen, M. Sundararajan, R. Boucher (01.05.2016)

[7] Application Fundamentals [WWW] https://developer.android.com/guide/components/fundamentals.html (05.05.2016)

[8] Processes and Application Life Cycle [WWW] http://developer.android.com/guide/topics/processes/process-lifecycle.html (05.05.2016)

[9] Version Control Software in 2014: What are your options? [WWW] http://www.sitepoint.com/version-control-software-2014-what-options Shaumik Daityari (07.05.2016)

[10] What is Android Application Components And How We Use It? [WWW] http://mithileshjoshi.blogspot.com.ee/2015/06/what-is-android-application-components.html Mithilesh Joshi (15.05.2016)

Lisa 1 – Kasutuses olevate andmebaasisüsteemide populaarsus mais 2016

Allikas: http://db-engines.com/en/ranking (08.05.2016)

Koht	Andmebaasisüsteem	Andmebaasi tüüp	Skoor
1	Oracle	Relatsiooniline	1462.02
2	MySQL	Relatsiooniline	1371.83
3	Microsoft SQL Server	Relatsiooniline	1142.82
4	MongoDB	Dokumendipõhine	320.22
5	PostgreSQL	Relatsiooniline	307.61
6	DB2	Relatsiooniline	185.96
8	Microsoft Access	Relatsiooniline	131.58
9	Redis	Võti-väärtus objekt	108.24
10	SQLite	Relatsiooniline	107.26

Lisa 2 – Nutitelefonide müük operatsioonisüsteemide lõikes 2015. ja 2014. aastal

Allikas: http://www.gartner.com/

Operating System	2Q15 (Units)	2Q15 Market Share (%)	2Q14 (Units)	2Q14 Market Share (%)
Android	271,010	82.2	243,484	83.8
iOS	48,086	14.6	35,345	12.2
Windows	8,198	2.5	8,095	2.8
BlackBerry	1,153	0.3	2,044	0.7
Others	1,229.00	0.4	1,416.80	0.5
Total	**329,676.40**	**100**	**290,384.40**	**100**