TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Võrgutarkvara õppetool

Androidi pangarakenduse loomine Monese näitel. Android bank application - Monese

Bakalaureusetöö

Üliõpilane: Kaspar Peterson
Üliõpilaskood: 121033IAPB
Juhendaja: Roger Kerse

Tallinn
2015

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.

_____ _____

(*kuupäev*) (*allkiri*)

Annotatsioon

Töö eesmärgiks on rääkida Androidi pangarakendusest Monese, mille abil on võimalik luua endale pangakonto ükskõik kus ja ükskõik millal, kasutades vaid isikuttõendavat dokumenti ning Androidi operatsioonisüsteemiga seadet. Pärast konto loomist on võimalik teha erinevaid pangatehinguid.

Töös käsitleb autor seadme sõrmejälje tegemist. Autor uurib, kuidas kasutaja klaviatuuri sisendit jälgida ning kuidas turvaliselt parooli sisestada. Lisaks sellele kirjeldab autor antud rakenduse jaoks väljatöötatud disaini lahendusi.

Tulemuseks on rakendus, mille abil saab kasutaja teha endale pangakonto ning pangakonto eduka tegemise korral ka makseid sooritada. Klient saab kasutada selget ja arusaadavat rakendust koos ilusa kasutajaliidese ning selgust lisavate animatsioonidega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 6 peatükki, 23 joonist, 0 tabelit.

Abstract

The aim on this work is to intruduce Android bank application called Monese. In Monese user is able to make a bank account in minutes, using only Android device and identity document. After account creation user is able to make transactions.

Author describes which system atributes can be used to make device fingerprint. Author researches possibly keyloggers and desicrbes how to make on. After that author offers a solution to enter pin more securely. In addition author talks about several design solutions.

The main result of this work is a beautiful, easy to use and functional Android application.

The thesis is in Estonian and contains 42 pages of text, 6 chapters, 23 figures, 0 tables.

Lühendite ja mõistete sõnastik

App Manifest **App Manifest**
Androidi rakenduse juurkataloogis olev fail, mis esitab hädavajaliku informatsiooni rakenduse kohta. [7]

ArrayAdapter **ArrayAdapter**
Klass Androidis, mis kuvab ListView elemente ekraanile. [6]

ESN **ESN**
Unikaalne number identifitseerimaks koodjaotusega mobiiltelefone.[56]

FrameLayout **FrameLayout**
Vaate grupp, mis hoiab teisi vaateid.[20]

IMEI **IMEI**
Unikaalne number identifitseerimaks globaalse mobiilisidesüsteemi-i või lairiba-koodjaotusega mobiiltelefone ning samuti ka satelliittelefone. [56]

IMSI **IMSI**
Unikaalne identifikaator, mis on seotud kõigi globaalse

	mobiilsidesüsteemi ja universaalse mobiilsidesüsteemi võrgu mobiiltelefoni kasutajatele.[55]
LayoutInflater	**LayoutInflater** Klass, mis täidab vaate XMLis defineeritud vaatega.[27]
ListView	**ListView** Vaate grupp, mis kuvab listi keritavatest elementidest.[30]
LinearLayout	**LinearLayout** Vaate grupp, mis hoiab teisi vaateid.[28]
MEID	**MEID** Globaalselt unikaalne number identifitseerimaks koodjaotusega mobiilijaama füüsilist osa. [56]
RelativeLayout	**RelativeLayout** Vaate grupp, mis hoiab teisi vaateid.[44]
ShapeDrawable	**ShapeDrawable** Objekt, mis aitab joonistada primitiivseid kujundeid. [48]
Spinner	**Spinner** Androidi rippmenüü.[49]
TextView	**TextView** Androidi tekstiväli.[54]
TypedArray	**TypedArray** Väärtuste massiivi konteiner.[55]
ViewClass	**ViewClass** Eraldi vaate klass, mis koondab funktsionaalsuse lihtsasti kasutatavasse liidesesse.[14]
WifiManager	**WifiManager** Klass, mille kaudu saab androidis küsida kõike, mis puudutab WiFi ühendust.[61]
XML	**XML** Laiendatab märgistuskeel.[56]

Jooniste nimikiri

Sisukord

1. Sissejuhatus

1.1 Taust ja probleem

Inglismaal on välismaalastel pangakonto tegemine raske ning aeganõudev. Selleks peab inimene vähemalt korra füüsiliselt pangakontorisse kohale minema ning see võtab mitmeid nädalaid aega, seetõttu oleks vajalik pakkuda inimestele võimalust teha endale pangakonto kiiremini ja lihtsamini, ilma ise kohale minemata.

See töö ei ole vajalik mitte Inglismaa kodanikele, vaid välismaalastele, kes soovivad tulla Inglismaale tööle ja kellel on seetõttu vajalik omada Inglismaal pangakontot. Hiljem laieneb antud töö tulemusena valmiv rakendus ka teistesse riikidesse.

Androidi rakenduse osa valmib Tallinnas firma Mooncascade'i kontoris. Serveri-, andmebaasi-, disaini- ja muu arendus toimub nii Tallinnas kui ka Inglismaal väljaspool Mooncascade firmat.

Androidi rakenduse arendust alustati 2015. aasta jaanuari keskpaigas ning selle töö kirjutamise hetkel ei ole see projekt veel valmis.

1.2 Ülesande püstitus

Eesmärgiks:

- Teha kasutajale ilus, mugav ja funktsionaalne pangarakendus.
- Teha kasutajale mugav, kiire ning turvaline registreerimine ning sisselogimine.

1.3 Metoodika

Ülesandes püstitatud eesmärkide saavutamiseks arendab autor rakendust koos tiimiga. Visuaalse kujunduse koostöös autoriga valmistab antud rakendusele disainer. Seejärel töö autor kasutab disaineri poolt valmistatud ressursse ning valmistab selle põhjal reaalse rakenduse. Mugava ja funktsionaalse kasutajaliidese loogika valmistab kasutajakogemuse

disainer ning antud töö autor tegeleb selle loogika teostamisega. Selleks, et kasutaja saaks rakendust kasutada, on vaja arendada ka serveri poolt, mida arendatakse väljaspool töö autori firmat.

1.4 Ülevaade tööst

Töös tutvustab autor lühidalt, milleks on antud rakendus vajalik. Lisaks kirjutab autor, mis telefoni andmeid võiks kasutada seadme sõrmejälje moodustamiseks. Lühidalt kirjeldab autor, kuidas on võimalik Androidi operatsioonisüsteemis kasutaja iga klaviatuuri sisendit jälgida. Suurema osa tööst kirjeldab autor antud panga rakenduse jaoks välja töötatud disainielemente.

2. Pangakonto tegemine Inglismaal välismaalasena

Inglismaal on välismaalasel pangakonto tegemine pikk ning tülikas protsess.[19] Peamiseks mureks pangakonto tegemisel on aadressi kinnitamine. Nimelt on elukoha kinnitamiseks vaja pangale ette näidata isiku elukoha kommunaalkulud, teise panga kinnitus elukoha kohta või kohaliku omavalitsuse poolt kehtestatud maksude tasumise arve isiku poolt.[32, 36, 39] Kõige suuremaks probleemiks elukoha kinnitamisega on see, et isik peab selle jaoks eelnevalt Inglismaal elama. Välismaalasena Inglismaale minnes tähendab see aga seda, et kõigepealt peaks isik elama vähemalt ühe kuu Inglismaal, et saada kommunaalkulude arveid. Pärast seda ei pruugi isik aga kohe siiski pangakontot saada, kuna pank võib avalduse tagasi lükata või küsida lisa dokumente. Olukorras, kus inimene läheb näiteks ainult kolmeks kuuks Inglismaale elama ja töötama, on pangakonto saamiseks ligi kuu ootamist kahtlemata liiga pikk aeg. Lihtsam on pangakontot teha isikutel, kes lähevad Inglismaale õppima, mitte tööle. Nimelt nendel, kes Inglismaale õppima lähevad, on võimalik saada kooli poolt spetsiaalne dokument, mis kinnitab isiku elukoha ja õpilase staatuse.[25]

Antud rakenduse eesmärgiks on pakkuda inimestele võimalust teha endale pangakonto vaid loetud minutitega ning pärast edukat konto loomist sooritada koheselt ka makseid. Olukorras, kus inimene läheb vaid mõneks nädalaks või kuuks Inglismaale elama, on selline võimalus äärmiselt mugav, kuna siis ta ei pea hakkama aega kulutama pangakonto loomisele ning saab kasutada seda aega uude elukohta sisse elamiseks.

Monese pangarakenduse eeliseks on esiteks see, et kasutaja saab endale pangakonto luua ideaal olukorras vaid minutitega ning pärast pangakonto loomist sooritada koheselt ka makseid. Olukorras, kus inimene läheb ilma suure etteplaneerimiseta Inglismaale ning soovib kiiresti pangakontot luua, on valmiv Monese kõige kiirem ning lihtsam valik. Lisaks sellele on autori arvates Monese rakendusel ka ilusam ning mugavam kasutajaliides, mis on äärmiselt tähtis just noorte seas. Nimelt on noored valmis vahetama panka vaid telefonirakenduse põhjal.[37]

3. Seadme sõrmejälg

Turvalisuse ning mugavuse kompromissina on telefon seotud kindla kasutajaga. Sellise lähenemisega tekib sarnane seos nagu on hetkel tavalise pangakaardiga. Kasutaja saab pangakaardi ning neljakohalise numbrikombinatsiooniga ligi ainult enda kontole. Androidi rakenduse puhul töötab telefon sarnaselt pangakaardiga. Nimelt saab juba varem autenditud telefoniga ligi oma pangakontole teades vaid viiekohalist numbrikombinatsiooni. Selle teostamiseks on vaja luua seadme sõrmejälg.

Seadme sõrmejälg töötab kui kasutajanimi. Kasutajal on küll kasutajanimi olemas, kuid seda ütleb seadmele server. Nimelt rakenduse käivitamisel küsib seade serverilt, kas sellise sõrmejäljega seade on varem sisse loginud. Kirjutamise hetkel pole veel päris täpselt selge kuidas toimub esmane autentimine. Antud hetkel saab seade autenditud kasutaja registreerimisel.

Sellise lähenemise korral ei pea kasutaja sisestama enda kasutajanime ning saab kontole kiiremini ligi. Levinud lahendus on kasutaja sisse logimiseks kasutada kasutajanime ning parooli, ilma spetsiifilist seadet arvestamata. Sellisel juhul peaks olema aga kasutaja parool keerulisem kuna sisse saab logida igalt seadmelt. Antud rakenduse korral saab sisse logida vaid varem autenditud seadmega.

Telefoni sidumiseks kindla kasutajaga on telefonist vaja võtta seadme spetsiifilised andmed, millest moodustatakse seadme sõrmejälg. Selleks kasutab autor nii seadmele täiesti unikaalseid kui ka teiste seadmetega kattuvaid atribuute. Kattuvad atribuudid lisavad suurema keerukuse ning ajakulu seadme sõrmejälje järgitegemisele. Mõned seadme spetsiifilised atribuudid võivad ajas muutuda, kuid serveris on vastav loogika, mis sellega arvestab. Seadme sõrmejälje muutumisel nii suures hulgas, et server seda enam turvalisuse huvides kindla seadmega kokku ei vii, peab kasutaja tegema läbi uue autentimise. Sõrmejälg ei muutu ajas tihti ning kasutaja peab lisaautentimist tegema vaid erandkorras. Enamasti saab kasutaja oma pangakontole ligi vaid viiekohalise numbrikombinatsiooni sisestamisel.

3.1 Seadme sõrmejälje komponendid

Kõigepealt kirjeldab autor unikaalseid seadme atribuute. Võib juhtuda, et mõnel seadmel ei saa mõnda neist atribuutidest kätte, kuid seni autori poolt testitud seadmetega on olnud vaid üks probleem. Nimelt on juhtunud olukord, kus WiFi oli väljalülitatud ning seade ei tagastanud sellepärast MAC-aadressi. Antud rakenduse kasutamiseks peab olema WiFi sisselülitatud ning seega see antud rakenduse puhul probleeme ei tekita.

Autor on testinud järgnevatel seadmetel:

- LG G3
- LG Nexus 5
- LG L40
- LG G2 mini
- LG Optimus L3 E400
- Samsung Galaxy S3
- Motorola Moto G
- Huawei Ascend G620S
- Sony Xperia E1
- HTC One mini

3.1.1 MAC-aadress

MAC-aadress ehk meediumipöörduse juhtimise aadress on unikaalne identifikaator, mis on kirjutatud võrgukaardile, et suhelda võrgu füüsilisel tasandil.[56] Antud kontekstis on tähtis vaid see, et MAC-aadress on unikaalne ning seda on võimalik seadmelt küsida. Selle kasutamine ainukese sõrmejälje moodustajana ei ole soovitatav, sest kui WiFi pole seadmes sisse lülitatud, ei pruugi telefon MAC-aadressi tagastada. Lisaks on MAC-aadressi kätte saamiseks vaja kasutajalt küsida vastavat luba. Järgnev luba tuleb lisada App Manifesti. [7, 33] Androidi projekti juurkataloogis peab olema fail AndroidManifest.xml, kuhu tuleb

kirjutada kõik õigused, mida rakendus soovib kasutada. Juhul kui kasutaja ei märgi ära mingit õigust, kuid üritab seda pärast rakenduse töötamise ajal teha, siis rakendus lakkab töötamast. Selleks, et rakendus saaks seadmelt MAC-aadressi lugeda, peab rakenduselt küsima luba WiFi oleku lugemiseks.

Enamik Androidi rakendusi kasutavad mingil määral WiFi-t ning seetõttu pole selle õiguse küsimine kasutaja jaoks ebameeldiv ega kahtlusi tekitav. MAC-aadressi lugemiseks kasutab autor WifiManageri, mille käest saab küsida MAC-aadressi.

3.1.2 Riistvara seerianumber

Riistvara seerianumber on Androidi seadmetel toetatud alates versioonist 2.3. [10, 22, 23] Seadmete protsent, mis kasutab vanemat operatsioonisüsteemi kui 2.3 on 0.4%.[16] Antud töö raames valmiv rakendus on toetatud alates versioonist Android 4.0.3 ja seega võib arvestada, et kõigil rakenduse kasutajatel on seadmed, kus on riistvara seerianumber toetatud. Seadmetel, mis ei ole võimelised helistama, kehtib nõue, et nad peavad tagastama unikaalse seadme identifikaatori. Selle kohta, kui paljudel seadmetel seerianumber midagi tagastab, puudub dokumentatsioon. Riistvara seerianumber ei ole seega üksinda kasutatult soovitatav, kuid selle olemasolul tõstab märgatavalt kasutaja turvalisust. Seerianumbri positiivse küljena võib välja tuua, et selle lugemiseks ei pea kasutajalt õigusi küsima ning koodis saab seda lihtsalt lugeda.

3.1.3 Android ID

Android ID on 64-bitine number, mis on suvaliselt genereeritud seadme esmasel käivitamisel ning jääb muutumatuks seadme eluaja jooksul. Selle väärtus võib muutuda vaid tehase seadete taastamisel. Android ID on toetatud kõigil seadmetel, mille operatsioonisüsteem on Android 1.5 või kõrgem. Lisaks, kui Androidi seadet kasutavad mitu kasutajat, siis on neil kasutajatel erinev Android ID ning nad tunduvad ühele ja samale rakendusele kui kaks erinevat seadet.[2, 47] Olukorras, kus kaks kasutajat kasutavad sama telefoni ning neil on erinevad kasutajad, siis ei saa esimene enda telefoni kasutajaga teise pangakontole ligi. Android ID ei ole unikaalne, kuid see on siiski väga kasulik seadme sõrmejälje moodustamiseks, kuna selle väärtus on kättesaadav igal seadmel. Android ID pole aga 100% usaldatav seadmetel, kus jookseb Android 2.2 või vanem operatsioonisüsteem, kuid kuna antud töö raames valmiv rakendus töötab alates versioonist Android 4.0.3, siis pole see probleemiks. [22, 23]

3.1.4 Unikaalne number (IMEI, MEID, ESN, IMSI)

Androidi seadmed, millel on telefoni tehnoloogia sisse ehitatud, peavad tagastama sõltuvalt kasutatavast võrgu tehnoloogiast kas IMEI, MEID, ESN või IMSI. [22, 23] Need kõik on unikaalsed ja seega sobivad väga hästi telefoni sõrmejälje moodustamiseks. Neid kõike saab küsida TelephonyManageri käest. [53]

3.1.5 Telefoni number

Androidi operatsioonisüsteemiga telefonis on võimalik lugeda SIM-kaardilt telefoni numbrit. Selle kättesaadavus sõltub aga SIM-kaardi tootjast ning autor ei ole selle kohta dokumentatsiooni leidnud. Autor on testinud seda Emt, Elisa ja Tele2 SIM-kaartidega ning seni on numbri kätte saanud ainult Tele2 võrgus olevatelt SIM-kaartidelt. SIM-kaardilt numbri lugemiseks peab kasutajalt küsima luba kirjutuskaitstud telefoni oleku lugemiseks.

SIM-kaardilt numbri lugemiseks tuleb pöörduda TelephonyManageri poole.[53]Selle poole ei saa otse pöörduda vaid tuleb süsteemilt küsida viidet. Viite olemasolul saab juba otse telefoni numbrit küsida. Juhul kui telefoni number pole kättesaadav, tagastatakse tühi sõne.

3.2 Mitte unikaalsed atribuudid

Järgnevalt kirjeldab autor teisi seadme spetsiifilisi atribuute, mida saab kasutada seadme sõrmejälje moodustamisel.[10, 11] Järgnevad atribuudid ei ole küll unikaalsed, kuid lisavad seadme sõrmejäljele olulist keerukust ning on lihtsasti seadme poolt kättesaadavad. Lisaks ei pea nende atribuutide küsimiseks luba küsima. Kahjuks puudub nende kohta täpne dokumentatsioon, kuid tähtis on see, et need tagastavad erinevate telefonide ja tootjate seas erinevaid väärtusi ning on piisavalt stabiilsed seadme eluaja jooksul, et neid võiks kasutada seadme sõrmejälje moodustamisel koos vastava loogika teostamisega serveri poole peal. Autor soovitab kasutada järgnevaid atribuute:

- Operatsioonisüsteemi versioon.
- Kasutaja poolt nähtav operatsioonisüsteemi versioon.
- Tööstusliku disaini nimi.
- Lõppkasutaja poolt nähtav toote nimi.

- Üldine toote nimi.
- Riistvara nimi.
- Androidi versiooni ID.
- Seadme riistvara nimi.
- Seadme või riistvara tootja nimi.
- Atribuut nimega HOST, selle kohta puudub igasugune dokumentatsioon. Seni autori poolt testitud seadmetel on see alati väärtuse tagastanud ning see väärtus on olnud testitud seadmete hulgas unikaalne.

3.3 Kaheastmeline autentimine

Kaheastmeline autentimine tähendab seda, et autentimine koosneb kahest sammust, kuid autor leiab, et selle kasutamisega saaks turvalisust tõsta. Esimene autentimis samm on tavapärase kasutajanime ja parooli sisestamine ning teise sammuna peab kasutaja sisestama oma mobiilile tuleva SMS-i koodi. Kaheastmeline autentimine ei ole otseselt seadme sõrmejälje osa kuid aitab veenduda, et kontole üritab ligi saada selle õige omanik. Olukorras, kus kasutaja kasutab rakendust telefonis, on kaheastmeline autentimine mugav kuna ei pea hakkama otsima teist seadet. Lisaks ei välista kaheastmeline autentimine Androidi tahvelarvutiga rakenduse kasutamist. Nimelt tuleb kinnitus kood teisele seadmele. See tähendab, et rakenduse kasutamiseks on vaja kahte seadet.

Antud pangarakenduse raames ei kasutata kaheastmelist autentimist sisselogimiseks. Iga kord SMS-i ootamine sisselogimisel mõjuks kasutuskogemusele negatiivselt. Mõistlik oleks teha kaheastmelist autentimist teatud aja tagant. See tähendab, et saata SMS-i kinnitus kood näiteks päeva esimesel sisselogimisel või tund pärast viimase kasutuse möödumist. Antud pangarakenduse korral peab parooli uuesti sisestama kui rakendus on olnud 20 sekundit tagataustal. 20 sekundi möödudes ei saa kindlasti kasutada kaheastmelist autentimist. Nimelt on tüüpiline kasutusjuht see kui kasutaja hakkab sooritama makset ning tahab näiteks seadme teisest rakendusest vaadata makse saaja pangakonto numbrit.

3.4 Seadme sõrmejälje moodustamine

Olukorras, kus soovitakse kasutada ainult ühte atribuuti, siis sobib selleks kõige paremini Android ID. See on toetatud kõigil seadmetel, kus on Androidi operatsioonisüsteem 1.5 või kõrgem. Google Play pood töötab alates versioonist 2.2 ning seega võib arvestada, et ühelgi seadmel sellega probleeme ei ole. Ühe atribuudi kasutamine seadme sõrmejälje moodustamisel teeb selle lahtimurdmise kordades lihtsamaks võrreldes seadme sõrmejäljega, kus on kasutusel rohkem kui üks atribuut. Seega soovitab autor rakendustes, kus turvalisus on tähtis, kasutada võimalikult palju erinevaid atribuute, et moodustada nendest vabaltvalitud loogika abil turvalisem seadme sõrmejälg kui ainult üks atribuut seda võimaldab.

Kahjuks on tõenäosus, et sõrmejälg muutub tunduvalt suurem kui kasutada kõiki eelnimetatud atribuute. Serveri poolel peab sellega arvestama. Ühel aja hetkel on ühe panga kontoga seotud ainult üks sõrmejälg, seega kui sõrmejälg muutub nii palju, et server seda enam ei tunnista, siis tuleb teha uus autentimine. Kirjutamise hetkel pole see funktsionaalsus veel valmis, kuid ideeliselt tähendab see kas lisa paroolide, turvaküsimuste sisestamist või seadme kaameraga isikut tõendavalt dokumendilt kasutaja andmete lugemist. Üldjoontes on siiski sõrmejälg ajas piisavalt muutumatu ja kasutaja saab ikkagi ainult ühte viiekohalist numbrikombinatsiooni sisestades oma kontole ligi.

4. Parooli sisestamine

Androidi seadmetel saab salaja teiste kasutajate klaviatuuri sisendit jälgida, sealhulgas ka paroole. Autor avastas uurimuse tulemusena ainult ühe mooduse, milleks on ehitada Androidi klaviatuur. Google on selleks teinud ka oma juhendi. [15] Selle meetodi korral peab klaviatuurinuhi kasutaja aga endale ise seadmesse installima. Tehes kasutajale meeldiva välimuse või funktsionaalsusega klaviatuuri, võib kasutaja hoolimata hoiatusest, et selle abil võib keegi kasutaja klaviatuuri jälgida, selle endale seadmesse installida. Lisaks võib pahatahtlik inimene klaviatuurinuhi ise seadmesse installida kui tal on seadmele füüsiline ligipääs.

4.1 Klahvinuhi tegemine

Kasutaja iga klahvi vajutuse jälgimiseks peab kõigepealt tegema teenuse, mis võimaldaks jälgida kasutaja sisendit ning selle informatsiooni siis kas otse või mingi loogika alusel sisendit teisendades kasutatavasse rakendusse saadab. Kasutaja klaviatuuri sisend peab ekraanile jõudma täpselt samamoodi nagu Androidi tavalisel klaviatuuril, muus olukorras kasutaja seda klaviatuuri ei kasutaks.[46] Selle jaoks tuleb App Manifesti lisada järgmised read, kus Teenus on autori enda poolt loodud klass, mis kasutaja sisendit jälgib:

```
<service android:name="Teenus"
android:permission="android.permission.BIND_INPUT_METHOD">
        <intent-filter>
                <action android:name="android.view.InputMethod" />
        </intent-filter>
        <meta-data android:name="android.view.im" android:resource="@xml/method" />
</service>
```

Joonis 1: Teenuse registreerimine App Manifestis

Eel toodud näites küsib teenus endale luba, et teenus saaks süsteemiga ühendust võtta. IntentFilteriga märgitakse ära, et kasutaja sisend jõuaks eelnevalt kirjeldatud Teenusesse.[24] Intent on sõnumi objekt, millega saadetakse Androidis edasi mingit kavatsust. Kavatsus võib olla rakenduse siseselt teise ekraani vaate avamine, soov saata emaili või näiteks avada telefoni numbri valimine. IntentFilter on filter, mille abil Androidi operatsioonisüsteemis nii-öelda teatatakse süsteemile, et see rakendus soovib klaviatuuri teenust pakkuda.

Järgmiseks peab tegema vastava klassi, millele App Manifestis viidatakse. See klass peab laiendama InputMethodService klassi. See klass pakub juba standarse implementatsiooni klaviatuuri sisendi teostamiseks.

Teenuse sees tuleks üle kirjutada meetod onKey(int primaryCode, int[] keyCodes) (vt joonis 2), et Teenuse seest saaks iga kasutaja sisendi kätte. Selles meetodis määrab Teenus ennast iga klaviatuuri klahvi vajutuse pealtkuulajaks. See tähendab, et kui kasutaja klaviatuuril mingit nuppu vajutab, edastatakse see nuppu vajutus Teenuse klassi.

```
@Override
public View onCreateInputView() {
        mInputView = (LatinKeyboardView) getLayoutInflater().inflate(
                R.layout.input, null);
        mInputView.setOnKeyboardActionListener(this);
        return mInputView;
}

@Override
public void onKey(int primaryCode, int[] keyCodes) {
        // Iga klaviatuuri klahvi vajutus jõuab siia meetodisse
}
```

Joonis 2: Klaviatuuri sisendi jälgimine

Viimases meetodis onKey(int primaryCode, int[] keyCodes) (vt joonis 2) on Teenusel kasutaja sisend teada ning siin võib selle kirjutada näiteks kuskile logi faili või saata selle kuhugi serverisse. Autor märkis siin ära peamised funktsioonid, kus tegevus toimub. Ülejäänud detailid on ära märgitud Google dokumentatsioonis. [15] Selle demonstreerimise eesmärgiks oli näidata, et kasutaja sisendit on lihtne jälgida. Siin kohal on veel tähtis märkida, et kui kasutaja on sellise klaviatuuri rakenduse endale installinud, siis see klaviatuur avaneb igal pool kus seade süsteemi klaviatuuri kasutab. See tähendab, et kui kasutaja saadab näiteks smsi, kirjutab emaili, sisestab veebi aadressi, logib brauseris kuhugi sisse, sisestab rakenduse sees parooli, või kirjutab midagi muud kasutades süsteemi klaviatuuri, siis kõik need sisendid lähevad läbi eeltoodud näite klaviatuuri.

4.2 Alternatiivne parooli sisestamine

Autor ei leidnud ühtegi teist moodust, kuidas võiks kasutaja sisendit jälgida kui ainult eelkirjeldatud moodusega. Autor arvab seega, et seadet eemalt jälgida, ilma seadmele füüsilise ligipääsuta, on väga raske. Arvestades fakti, et kasutaja peab klaviatuurinuhi endale ise seadmesse installima, on sellise pealt kuulamise tõenäosus väike. Lisaks on näiteks Eesti pangarakendustes nagu Swedbank ja SEB peale kasutaja parooli vaja veel sisestada ka kood paroolikaardilt või kasutada sisse logimiseks Mobiil-ID-d ning seega ainult kasutaja parooli teadmisest ei piisa.[9]

Rakenduses, kus turvalisus on tähtis, soovitab autor kasutada rakendusesisest klaviatuuri, et kasutajate salajane informatsioon, näiteks paroolid oleksid kaitstud. Niiviisi jääb kasutaja sisend rakenduse sisse, mitte ei lähe läbi teiste rakenduste, mis klaviatuuri loogikat teostavad. Rakendusesisese klaviatuuri all mõtleb autor lihtsalt numbrite ja tähtede nuppudest koosnevat maatriksit.

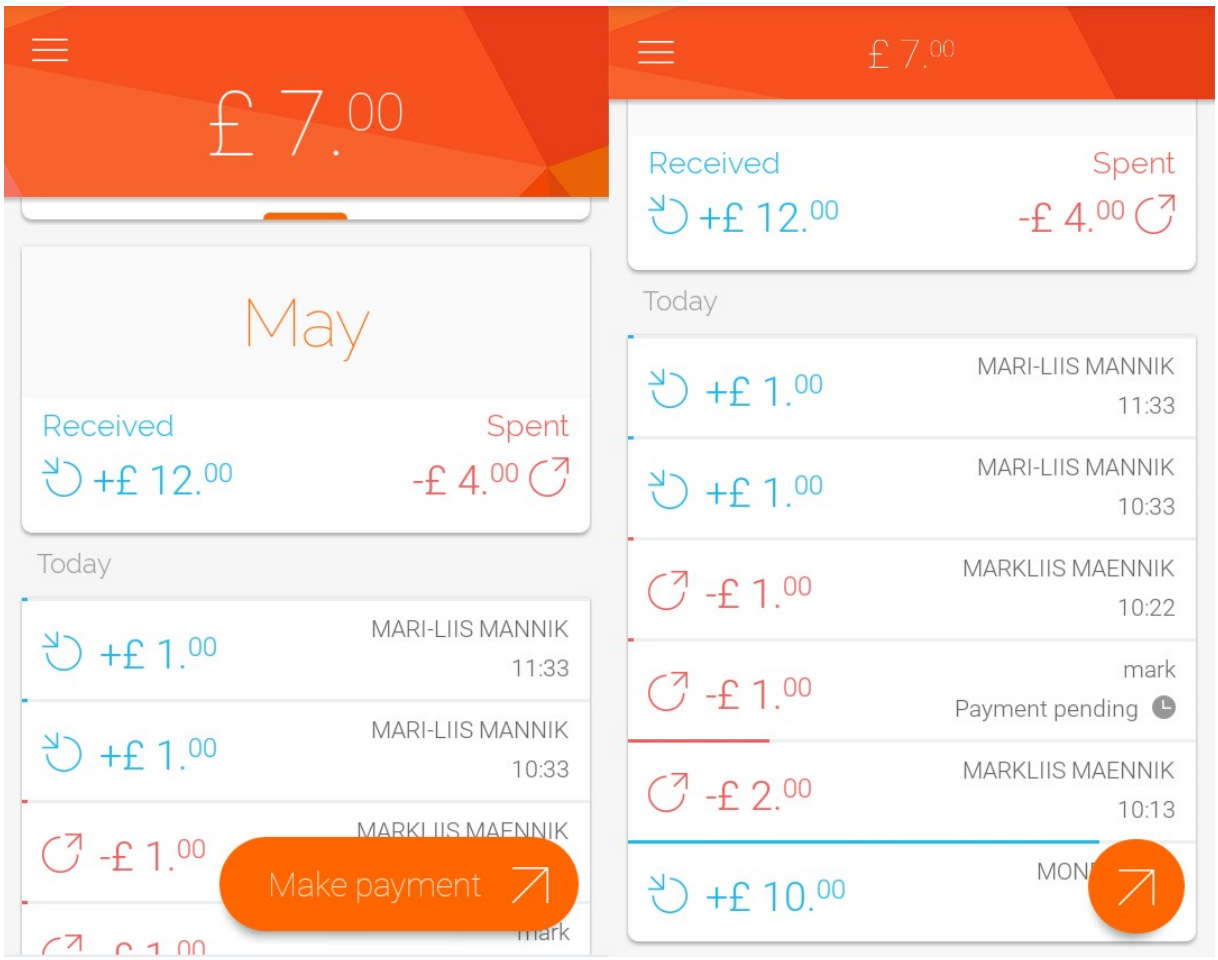
5. Mugava kasutajaliidese tegemine

Androidi rakenduse puhul on äärmiselt oluline lihtne ja mugav kasutajaliides. Võib öelda, et rakenduse kiirus ja mugavus määravad ära rakenduse edukuse. Nimelt on 37% Swedbanki uuringule vastanutest nõus vahetama panka ainult rakenduse tõttu.[37] Monese pangakonto loomine võtab vaid loetud minutid ning kasutaja saab selle rakenduse lihtsa vaevaga järgi proovida. Juhul kui rakendus meeldib kasutajale rohkem kui tema praegune pangarakendus on suur tõenäosus, et kasutaja vahetab panka.

Tähtis on ka animatsioonide taha peita serveriga suhtlemise päringud. See tähendab, et kasutaja ei pea tühja ekraani vaatama vaid serveriga suhtlemise ajal saab jälgida ekraanil toimuvaid animatsioone. Kasutajal jääb nii mulje, et ta ei peagi ühenduse järgi ootama. Lisaks teeb kasutuskogemuse paremaks riigi, telefoninumbri, telefoninumbri suunakoodi ning teiste sarnaste väljade eeltäitmine ning automaatne teksti lõpetus. Igasugune eeltäitmine on kriitiline just mobiilsetel seadmetel kus teksti sisestus väikesel ekraanil on komplitseeritud. Mugava kasutajaliidese loomisel peab arvestama ka asjaoluga, et väikese ekraaniga mobiilsetel seadmetel on informatsiooni kuvamine kohati keerukas, seega peab rakenduse sisu olema võimalikult minimalistlik. Järgnevalt kirjeldab autor konkreetse rakenduse raames väljatöötatud disaini ja funktsionaalsuse lahendusi.

5.1 Hõljuv tegevusnupp

Ruumi funktsionaalseks kasutamiseks on Androidis soovitatav kasutada hõljuvat tegevusnuppu. [12] Google kirjeldab küll enda disaini juhendis, milline see nupp peaks välja nägema ning kuidas käituma, kuid funktsionaalsuse ja välimuse peab arendaja ise tegema. Antud rakenduses kasutatakse seda nupu tehingute nimekirja peal. Tehingute nimekirja alla kerimisel kaob tekst sujuva animatsiooniga ära ning vabastab ruumi, et kasutajal oleks rohkem ruumi tehingute vaatamiseks. Algne tekst on vajalik, et kasutaja saaks aru, mida see nupp peaks tegema. Antud rakenduse raames on seda kasutatud niimoodi:



Joonis 3: Hõljuva tegevusnupu näide

5.1.1 Vari

Antud nupule on vaja lisada vari, et see tunduks tausta peal hõljuvat. Alates Androidi versioonist 5.0 Lollipop on võimalus XML disaini failis kirjeldada varju ning selle suurust. XML-is saab seda teha lihtsalt vaid ühe reaga: [17]

```
<View
        android:layout_width="2dp"
        android:layout_height="2dp"
        android:elevation="2dp"/>
```

Joonis 4: Varju lisamine Android 5.0 Lollipopis

Android 5.0 või uuemat operatsioonisüsteemi kasutavad kirjutamise hetkel vaid 5.4% inimestest ning seega ei saa seda hetkel laiemale kasutajaskonnale mõeldud rakenduses kasutada.[16] Varju tegemiseks on seega vaid üks võimalus, kasutada 9-patch'i. [13] 9-patch on olemuselt veniv rasterpilt. 9-patchi puhul saab määrata millised pildi osad on venivad.[18] 9-patchi saab panna samamoodi vaatele taustaks nagu ka teisi ressursse. Näide, kus Androidi projekti res/drawable kausta on salvestatud fail vari.9.png:

```
<View
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/vari" />
```

Joonis 5: Varju lisamine 9-patchiga

5.1.2 Visuaalse poole teostamine

Selleks, et hõljuv nupp oleks mingi teise vaate peal tuleb kasutada RelativeLayout'i või FrameLayout-i. [20, 44] Nende layout-ide puhul kehtib loogika, et kõige viimasena lisatud vaade on kõige peal.

Ümarate nurkade saamiseks tuleb kasutada ShapeDrawable, mis tuleks vaate taustaks lisada. [48] Selle faili salvestab autor antud näite puhul projekti kausta res/drawable nimega nurgad.xml. Näide selle faili sisust:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="@color/orange"/>
   <corners
      android:bottomRightRadius="@dimen/raadius"
      android:bottomLeftRadius="@dimen/raadius"
      android:topLeftRadius="@dimen/raadius"
      android:topRightRadius="@dimen/raadius"/>
</shape>
```

Joonis 6: Vaatele ümarate nurkade lisamine

ShapeDrawable raadius peab olema kaks korda väiksem kui vaate kõrgus ning laius, kuhu see tagataustaks lisatakse, kui tahta perfektset ringi. Vaatele ümarate nurkade lisamiseks tuleb see ShapeDrawable vaatele taustaks lisada.

Antud olukorras on vaja lisada vaatele ümarad nurgad koos varjuga. Selleks on kaks võimalust. Üheks neist on lisada vaatele juba eelnevalt värvitud 9-patch rasterpilt taustaks. Teiseks ning autori arvates paindlikumaks lahenduseks on lisada vari ning ümarad nurgad koos taustavärviga eraldi. Selleks tuleks lisada üks vaade teise sisse niimoodi, et ühe vaate taustaks oleks varjuga 9-patch ning teise vaate taustaks oleks ümarate nurkadega ShapeDrawable.

5.1.3 Funktsionaalsuse teostamine

Antud pangarakenduse raames muudab hõljuv tegevusnupp tagumise tehingute nimekirja liikumisel kuju. Tehingute nimekirja alla poole kerimisel muutub nupul olev tekst nähtamatuks. Nimekirja tagasi üles kerimisel muutub tekst jälle nähtavaks. Tehingute nimekirja kasutajale kuvamiseks on kasutatud ListView'd.[30] ListView kerimise registreerimiseks võib määrata ListView'le kuulaja, mis kutsutakse välja iga kord kui ListViews toimub kerimine. Näide:

```
listView.setOnScrollListener(new AbsListView.OnScrollListener() {
    @Override
    public void onScrollStateChanged(AbsListView absListView, int i) { }

    @Override
    public void onScroll(AbsListView absListView, int i, int i1, int i2) { }
});
```

Joonis 7: ListView kerimise registreerimine

Meetodi onScroll(AbsListView absListView, int i, int i1, int i2) (vt joonis 7) seest antakse hõljuvale tegevusnupule teada, millal peab nupp kuju muutma. Hõljuval tegevusnupul on kaks erinevat animatsiooni. Üks animatsioon muudab sujuvalt nupul oleva teksti suurust väiksemaks kuni see muutub nähtamatuks ning teine animatsioon muudab nähtamatu teksti nähtavaks ning suurendab teksti suurust kuni see on algse suurusega.

Vaate suuruse muutuse animatsiooni tegemiseks on autori arust mõistlik kasutada klassi Animation.[4] Animation klassi on sisse ehitatud antud animatsiooni jaoks vajalikud funktsioonid. Antud olukorras tuleb sättida animatsiooni pikkust ning vaate lõplikku suurust. Vaate kahandamisel on lõplikuks suuruseks 0 ning vaate suurendamisel on vaate lõplikuks suuruseks vaate täis suurus ehk vaate suurus kui vaatel on täissuuruses tekst täielikult nähtav.

Eeltoodud animatsiooni jaoks oleks mõistlik teha eraldi klass, mis selle animatsiooni loogika ära peidaks, nii, et sellele klassile antakse ette vaade, mille suurust muudetakse, lõplik suurus ning animatsiooni pikkuse. Tekstivaate suuruse suurendamiseks kirjutab autor üle Animation klassi funktsiooni applyTransformation(float interpolatedTime, Transformation t). Näide:

```
@Override
   protected void applyTransformation(float interpolatedTime, Transformation t) {
      int newWidth = initialWidth + (int) ((finalWidth - initialWidth) * interpolatedTime);

      view.getLayoutParams().width = newWidth;
      view.requestLayout();

      this.setAnimationListener(new AnimationListener() {
         @Override
         public void onAnimationEnd(Animation animation) {
            if (finalWidth == 0) view.setVisibility(View.GONE);
         }
      });
   }
```

Joonis 8: Tekstivaate suuruse animatsioon

Meetodis onAnimationEnd(Animation animation) vaadatakse kas, vaate lõplik laius on 0.[5] Kui väärtus on 0 siis kaotatakse vaade ekraanilt täielikult. Ainult nii on võimalik saada perfektne ring, sest kui vaade jääks ekraanile laiusega 0, oleks vaatel siiski väike vooderdus ning hõljuv tegevusnupp jääks lapik.

Loogika UI klassist välja tõstmiseks, oleks mõistlik teha selleks eraldi ViewClass. [14]Antud olukorras on vaate juurvaade LinearLayout ning seega on hõljuva nuppu juurklassiks LinearLayout. Selleks, et see vaade võtaks õige kujunduse, mille on autor XML-is juba ära kirjeldanud, peab konstruktoris LayoutInflateri abil toimima järgmiselt:

```
LayoutInflater.from(getContext()).inflate(R.layout.view, this, true);
```

Joonis 9: XML-ist vaate täitmine

Eelnevalt kirjeldatud vaate kasutamisel kuskil teises vaates tuleb viidata sellele klassile täispika nimega koos paketiga kus see asub. Näide:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout     xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <com.example.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
</RelativeLayout>
```

Joonis 10: CustomView kasutamine

5.2 Vihjega rippmenüü

Rippmenüüle ei saa Androidis anda vihjet. Selle teostamiseks kaalus autor kahte erinevat lahendust. Esimene neist oli plaan kirjutada üle Spinnerile lisatud ArrayAdapteri meetod getCount(). [6, 49, 50] Selle lahenduse puhul näeks Spinneri ArrayAdapteri kood välja selline:

```
public class SpinnerAdapter extends ArrayAdapter {
        @Override
        public int getCount() {
                return super.getCount() - 1;
        }
}
```

Joonis 11: Vihjega SpinnerAdapteri näide 1

Näide rippmenüü kasutamisest, kus sisu on List<String> tüüpi loend esemetest, mida rippmenüüs kuvatakse ning sisu viimane element on vihje:

```
SpinnerAdapter adapter = new SpinnerAdapter(this);
adapter.addAll(sisu);
spinner.setAdapter(adapter);
spinner.setSelection(sisu.size()-1);
```

Joonis 12: Vihjega SpinnerAdapateri näide 2

Antud lahendus ei ole käesoleva rakenduse jaoks siiski ideaalne. Viga on selles, et Androidi orientatsiooni muutuse korral või rakenduse tagataustale paneku korral saab esimene ese rippmenüüst valituks. See on põhjustatud ArrayAdaptri sisemise meetodi ülekirjutamisega.

Autori arvates on teiseks ning parimaks lahenduseks panna Spinneri vaatele kaks teksti vaadet nii, et alati on ainult üks neist nähtav. Seega kui kasutaja on juba midagi valinud, siis on nähtav valitud element Spinneri sisust ning kui kasutaja pole veel valikut teinud, siis on

nähtav vihje teksti vaade. Selle lahenduse jaoks tegi autor eraldi klassi, mis laiendab Spinneri klassi.

Spinneri ArrayAdapter on seotud konkreetse Spinneriga ning seega oleks mõistlik teha ArrayAdapter selle sama Spinneri alla privaatse klassina. Kasutaja valiku teada saamiseks peab üle kirjutama Spinneri meetodi setSelection(int position, boolean animate), mis kutsutakse välja igakord kui Spinneris midagi valitakse.[49] Autor leidis, et kahendmuutujaga oleks lihtne meelde jätta, kas valik on juba tehtud või peab veel vihjet näitama. See kahendmuutuja on alguses vale ning kui kutsutakse välja funktsioon setSelection(int position, boolean animate) siis märgitakse see kahendmuutuja tõeseks. Selleks, et näidata kasutajale kas vihjet või tehtud valikut peab ArrayAdapteri meetodi getView(int position, View converView, ViewGroup parent) üle kirjutama ning selles meetodis otsustama, kas kasutajale näidata vihjet või valitud elementi.

Selleks, et lisada vihje mugavalt XML-i failis, ilma Java koodita peab tegema lisatööd. Kõigepealt tuleb defineerida atribuudid, mida saab vaatele lisada. Selle jaoks tuleb projekti res/values kausta lisada attrs.xml fail. Näide:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <declare-styleable name="SpinnerWithHint">
      <attr name="hint" format="string"/>
   </declare-styleable>
</resources>
```

Joonis 13: CustomView atribuudi nime defineerimine

Koodis defineeritud uued atribuudid kuuluvad aga teise XML nimeruumi ja seega peab vihje lisamisel XML-is toimima järgmiselt: [14]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:custom="http://schemas.android.com/apk/res/com.example.customviews">

        <com.example.SpinnerWithHint
                custom:hint="Vihje" />

</LinearLayout>
```

Joonis 14: Vihje lisamine XML-is

Kõik atribuudid, mis on kirjeldatud XML-is edastatakse vaate konstruktorisse. Tehtud vaate klassis on ettenähtud XML-is edastatud informatsiooni kätte saamiseks konstruktoris käituda järgmiselt [8, 14]:

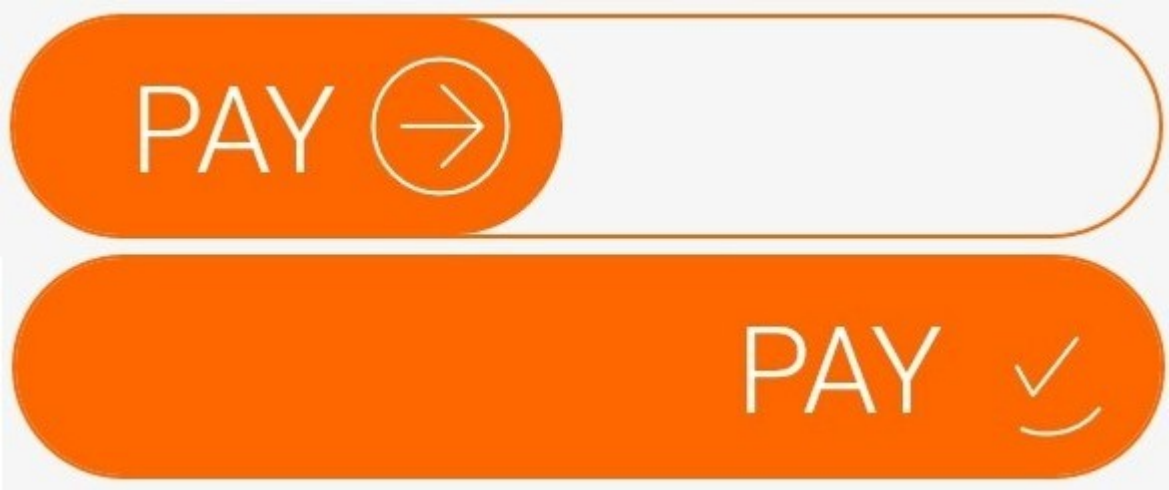
```
public SpinnerWithHint(Context context, AttributeSet attrs) {
  super(context, attrs);

  TypedArray a = context.getTheme().obtainStyledAttributes(attrs,
R.styleable.SpinnerWithHint, 0, 0);

  try {
     hint = a.getString(R.styleable.SpinnerWithHint_hint, false);
  } finally {
     a.recycle();
  }
}
```

Joonis 15: Vihje sõne kättesaamine Javas

Antud näites küsitakse meetodist obtainStyledAttributes() TypedArray-d. TypedArray tagastab massiivi väärtusi, kust on võimalik kindlate XML-is kirjeldatud väärtuste lihtsaks kättesaamiseks. TypedArray objektid on jagatud ressurss ning pärast nende kasutamist peab kindlasti ressursid vabastama.

5.3 Libistades aktiveeruv nupp

Kirjutamise hetkel ei olnud autor suuteline leidma ühtegi taolist valmis lahendust ja seega tegi autor sellise komponendi ise. Nupu kolm erinevat olekut on algolek, poole peal ning lõppolek, mis on näidatud alloleval pildil:



Joonis 16: Libistades aktiveeruv nupp

Tavalise nupu puhul toimub tegevus nupu peale vajutamise korral, kuid libistades aktiveeruva nupu puhul tuleb kasutajal nupp paremasse äärde libistada ning näpp õhku tõsta. Seda nuppu kasutatakse antud rakenduses panga makse alustamiseks. See nupp vähendab tõenäosust, et kasutaja kogemata makse sooritab. Selle saavutamiseks peab jälgima kasutaja näpu liikumist ekraanil ning selle järgi arvutama nupu libiseva osa laiuse. Lisaks muutub nool pärast kasutaja näpu õhku tõstmist, kui nupp on täielikult paremas ääres animatsiooniga linnukeseks, nii et ümber linnukese jääb ring keerlema.

5.3.1 Nupu libistamine

Esimese asjana on vaja kätte saada kasutaja näpu asukoht ekraanil. Androidis on seda lihtne teha, kuna see on saavutatav juba sisse ehitatud funktsionaalsusega. Selle jaoks peab vaatele, mille peal kasutaja näpu asukohta teada on vaja, panema kuulaja, mis kutsutakse välja iga kord kui kasutaja näppu antud vaate sees liigutab. Näide demonstreerimaks, kuidas saab kätte näpu asukoha x-koordinaadi [35, 60]:

```
view.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent motionEvent) {
                int x = (int) motionEvent.getX();
                return true;
        }
});
```

Joonis 17: Kasutaja näpu asukoha teada saamine

Järgmiseks on vaja näpu asukoha järgi arvutada nupu suurust. Sellejaoks on vaja teada meil teksti laiust, teksti veerist, noole ikooni laiust, noole ikooni veerist ning vasakut veerist.

Joonis 18: Libistades aktiveeruva nupu selgitav joonis

Androidi koodis peab dünaamiliselt vaate laiuse muutmiseks teadma täpseid väärtuseid pikslites. XML-is on heaks tavaks laiused ja veerised märkida aga tihedusest sõltumatute pikslitega [51]. Tihedusest sõltumatu piksel on abstraktne ühik, mis tähendab seda, et seadme peal arvutatakse välja mitu pikslit peaks mingi vaate kuvamiseks kasutama. Dünaamiliselt veerise saamiseks koodis kasutab autor järgmist moodust: [58]

```
int getLeftMargin(View view) {
        ViewGroup.MarginLayoutParams lp = (ViewGroup.MarginLayoutParams)
view.getLayoutParams();
        return lp.leftMargin;
}
```

Joonis 19: Vaatelt veerise küsimine

Eelnevas näites on demonstreeritud vasaku veerise saamist, kuid parema, alumise ja ülemise veerise saamine on analoogne. Siin kohal tasub märkida, et kui küsida veerist enne vaate ekraanile lisamist tagastatakse väärtus 0. Sama kehtib ka vaate laiuse küsimise kohta. Liikuva osa suuruse arvutamiseks peab kasutaja näpu asukoha x-koordinaadist maha lahutama vasaku veerise, kirja laiuse, kirja veerise, poole ikooni laiuse ning poole ikooni veerise väärtuse. Poole ikooni laiuse ja poole ikooni veerise väärtuse peab maha arvutama sellepärast, et siis viitab nupu liikumist juhtiv näpp paremas servas oleva ikooni peale.

Iga kord kui kutsutakse välja eeltoodud meetod onTouch(View view, MotionEvent motionEvent) peame arvutama nupu laiust. Nupu laiuse määrab ära liikuva osa suurus. Liikuva osana otsustas autor kasutada TextView'd. [54] TextView-l on olemas meetod setWidth(int), mis puudub aga teistel vaadetel ja seega saab selle väärtust muuta lihtsamalt kui teiste vaadete puhul. Seni pole TextView kasutamine tekitanud ühtegi probleemi ja seega võib arvestada, et TextView laiuse muutmine on antud probleemi lahendamisel aktsepteeritav. Eeltoodud meetodi onTouch(View view, MotionEvent motionEvent) sisu võiks välja näha seega selline:

```
@Override
public boolean onTouch(View view, MotionEvent motionEvent) {
        int newWidth = (int) motionEvent.getX() - leftMargin - textTotalWidth -
        (imageTotalWidth/2);
        slideView.setWidth(newWidth);

        return true;
}
```

Joonis 20: Libistades aktiveeruva nupu suuruse muutmine

Meetodis märgitud suurused leftMargin, textTotalWidth ja imageTotalWidth on juba eelnevalt välja arvutatud. Lisaks sisaldavad textTotalWidth ja imageTotalWidth vaadete laiusi koos nende veeristega.

Antud lahendusel on siiski üks probleem. Nimelt jookseb nupp üle parema ääre. Selle mitte juhtumiseks peab arvutama liikuva osa maksimaalse laiuse ning iga kord liikuva osa suuruse muutes seda kontrollima. Liikuva osa maksimaalse laiuse arvutamisel peab arvutama nupu täispikast suurusest teksti suuruse koos veerisega, pildi suuruse koos veerisega ning vasaku veerise. Igakord kui vaatele uut suurust hakatakse panema, peab kontrollima, kas see on väiksem kui maksimaalne suurus. Juhul kui see on suurem kui maksimaalne lubatud suurus, siis peab andma vaatele maksimaalse laiuse. Maksimaalse laiuse sättimata jätmisel võib tekkida olukord, kus kasutaja liigutab oma näppu kiiremini kui seade reageerida jõuab ja seega ei pruugi nupp täielikult värviga täituda. Sellises olukorras ei õnnestu kasutajal ka makset teha.

Lisaks eeltoodule peab nupul noole pilt muutuma sujuva animatsiooniga linnukeseks, kui kasutaja on nupu lõpuni tirinud ning näpu õhku tõstnud. Juhul, kui kasutaja tõstab näpu õhku enne nupu lõppu jõudmist, siis peab nupp sujuvalt algolekusse tagasi libisema. Tagasi libistamise animatsioon on vaate suuruse muutmine animatsiooniga, millest on räägitud

punktis 5.2.3. Näpu õhku tõstmise märkamine on tehtav lihtsalt. Meetodisse onTouch(View view, MotionEvent motionEvent) edastaval MotionEvent'ilt saab küsida, mis liiki tegevus juhtus. [35] Näide, eristamaks näpu õhku tõstmist mingist muust tegevusest:

```
@Override
public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_UP) {
                // Kasutaja tõstis näpu ekraanilt õhku
        } else {
                // Kasutaja näpp on endiselt ekraanil
        }
}
```

Joonis 21: Kasutaja näpu ekraanilt tõstmise registreerimine

5.3.2 Nupu animatsiooni realiseerimine

Nupu noole linnukeseks muutumise animatsiooni idee seisneb selles, et noole suurus muutub sujuvalt nii x- kui y-teljel väiksemaks, kuni vaade on täiesti nähtamatu. Pärast noole kadumist ilmub sujuva animatsiooniga nii x-kui y-teljel suurenedes linnuke.

Animatsiooni tegemiseks kasutab autor klassi Animation. [4] Animatsiooni saab lihtsalt XML-is defineerida ning seda pärast Java koodis välja kutsuda. [57] Antud olukorras on vaja muuta ainult vaate skaalat, seega peab XML-is defineerima ainult skaala animatsiooni. [45] Selle animatsiooni jaoks on vaja lisaks skaala muutumisele määrata ka animatsiooni aeg. Näide:

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:duration="150"
    android:fromXScale="1"
    android:fromYScale="1"
    android:toXScale="0"
    android:toYScale="0"
    android:pivotX="50%"
    android:pivotY="50%" />
```

Joonis 22: Vaate skaala muutmise animatsioon

Antud näite puhul on näha, et skaalalt 1 minnakse skaalale 0. Vaade muutub seega 100%-liselt suuruselt 0%-ni ehk täissuurusest nähtamatuks. Lisaks sellel on autor märkinud veel ära interpolaatori. [26] Interpolaator on Androidis mugav vahend, mis jagab animatsiooni aja algpunktist kuni lõpppunktini sammudeks. Selle abil on programmeerijal lihtne animatsioonilt lineaarsus ära kaotada. Niiviisi ei tundu animatsioon kasutajale robotlik vaid näib loomulikuna. Lisaks sellele on autor märkinud ära ka punkti mille suhtes skaala muutus toimub. Nimelt pivotX ja pivotY määravad ära vastavad punktid x- ning y-teljel, mille suhtes animatsioon toimub. Kasutades % märki saab lihtsalt määrata punkti suhteliselt vaate suurusele. Näide antud animatsiooni välja kutsumisest kui eeltoodud animatsiooni XML fail on salvestatud nimega scale_out.xml kausta res/anim:

```
Animation animation = AnimationUtils.loadAnimation(getContext(), R.anim.scale_out);
view.startAnimation(animation);
```

Joonis 23: XML-is defineeritud animatsiooni kasutamine

5.3.3 Nupu aktiveerimise registreerimine

Nupu nii öelda kliki ehk paremasse äärde libistamise kättesaamiseks soovitab autor teha liidese. Eelnevalt punktis 5.2.3 näidatud ViewClassi tegemise näitel oleks mõistlik teha eraldi klass ka sellele nupule. Igakord kui nupp jõuab paremasse äärde ning kasutaja näpu üles tõstab kutsutakse selle liidese meetodit välja. Autori arvates võiks jätta Androidi tavalise onClick(View v) meetodi selle vaate puhul üle kirjutamata kuna antud olukorras on need siiski natukene erinevad asjad ning tulevikus arenduse käigus võib onClick(View v) meetodit millekski muuks vaja minna. Punktis 5.3.1 on autor kirjeldanud, kuidas teada saada, millal kasutaja näpu õhku tõstab. Selleks, et teha kindlaks kas nupp on täielikult paremasse äärde jõudnud, tuleb lisada üks kui lause.

5.4 Aadressi automaatne lõpetamine

Aadressi kõikide komponentide sisestamine on aeganõudev tegevus, eriti mobiilsetes seadmetes. Kasutaja aja säästmiseks, kasutab autor selle jaoks Google asukohtade automaat lõpetamist. Niiviisi piisab kasutajal enda elukoha tänava nime ning maja numbri sisestamisest ning talle pakutakse juba ka linna, maakonda, riiki ning postiindeksit. Seejärel on kasutajal eeltäidetud kõik vajalikud väljad. Erandkorras peab kasutaja tegema vaid väikeseid muudatusi.

Google on teinud aadressi automaatseks lõpetamiseks valmis komponendi, mille saab lihtsa vaevaga enda Androidi projekti lisada. Autor ei hakka sellest pikemalt rääkima kuna Google poolt on selle teostamiseks õpetus olemas. [21] Kasutaja valitud aadressiga saab kaasa ka selle asukoha unikaalse ID. [40, 41] Selle ID järgi saab teha lisa päringu mille abil eeltäita kõik vajalikud väljad nagu linn, riik, postiindeks jne. Autor soovitab parema kasutaja kogemuse nimel kaaluda selliseid võimalusi nagu aadressi automaatne lõpetamine.

6. Kokkuvõte

Töö põhieesmärgiks oli luua Androidi pangarakendus ilusa, mugava ning funktsionaalse kasutajaliidesega. Lisaks sellele oli eesmärgiks võimaldada kasutajal mugavalt ja turvaliselt registreerida ja sisse logida.

Antud töö tulemusena jõudis autor kirjeldada erinevaid seadme spetsiifilisi atribuute, mille abil seadme sõrmejälge luua. Veel kirjeldas autor lühidalt ühte võimaliku kasutaja klaviatuuri jälgimise moodust. Autor jõudis välja töötada mitu lahendust kasutajale ilusa ja mugava disaini tegemiseks. Antud töös ei jõudnud autor aga rääkida pangakonto registreerimisest ega sisselogimisest, kuid uuris võimalikke klaviatuuri jälgimise meetodeid ning tegi selle raames ka ise ühe rakenduse.

Autor jõudis järeldusele, et klahvinuhi tegemine on lihtne, kuid selle kasutaja seadmesse saamine on keeruline. Lisaks jõudis autor järeldusele, et Androidi dokumentatsiooni põhjalikumalt lugedes on paljude animatsioonide tegemine lihtne. Erinevate seadmete peal testides jõudis autor järeldusele, et eri ekraani suurustega seadmetele on raske kujundust teha.

Autor tõi siin välja töö osad, mis on juba läbinud testimise faasi ning seega, ei vaja autori arust antud töös kirjeldatu edasiarendust. Antud töö käigus mainiti vaid väga väike osa kogu pangarakendusest ning edasiarendust vajab rakendus kui tervik.

1 Kas eesmärk saavutati?

Eesmärgi võib lugeda osaliselt saavutatuks. Antud töö raames ei saanud küll kogu rakendus valmis kuid antud lõputöö raames kirjeldatud on kogu rakendusest siiski märkimisväärne osa. Kõik selle töö raames kirjeldatu on valmis ning testitud.

2 Põhitulemuste loetelu

- Autor tegi seadmesõrmejälje, millest ta küll siin täpsemalt ei räägi, kuid viitab paljudele atribuutidele, mida selle jaoks võiks kasutada.
- Autor tegi parooli sisestamiseks rakenduse sisese klaviatuuri, et vältida võimalikku jälgimist töös kirjeldatud klahvinuhi meetodiga.

- Autor tegi hõljuva tegevusnupu, mis töötab korrektselt seni kõigil testitud seadmetel.
- Autor tegi vihjega rippmenüü, mis töötab korrektselt seni kõigil testitud seadmetel.
- Autor tegi libistades aktiveeruva nupu, mis töötab korrektselt seni kõigil testitud seadmetel.

3 Kas eesmärgid saavutati?

Autori eesmärk pakkuda kasutajale ilusat, mugavat ning funktsionaalset kasutajaliidest sai sooritatud osaliselt. Nimelt jõudis autor siin töös kirjeldada vaid üksikuid komponente, mis on tähtsad rakenduse hea üldmulje jätmiseks. Antud töös kirjeldatud kasutajaliidese komponendid on valmis ja testitud ning võib laiemasse kasutusse minna. Turvaliselt parooli sisestamine sai samuti autori arvates saavutatud, kuna Androidi poolt on turvalisuse tagamiseks juba palju ära tehtud. Seadme sõrmejälg sai samuti tehtud. Antud töös ei rääkinud autor aga turvalisest registreerimisest ning sisselogimisest kuna need pole täielikult autori poolt tehtud ning autor ei soovi nende laiemale avalikkusele tutvustamist.

1. Summary

The main purpose of this bachelor thesis is to make a secure, beautiful, functional and easy to use Android bank application. Additionally, the goal is to make the registration process fast, so it would not take days, but minutes. Furthermore, the author's aim is to make logging in as secure as possible. To do that the author decided to use a device fingerprint.

In this bachelor thesis, author describes what attributes can be used to make an Android device fingerprint. Author also researches about possible Android keyloggers and builds one himself. For the most part of this thesis, author describes different solutions to building good looking and functional design elements. These design elements are: a floating action button, a spinner with a hint and slide-to-open button.

As a result of this bachelor thesis, the author successfully implemented an Android keylogger and came to a conclusion that building a keylogger is easy, but getting it installed on a device is complicated. Author also described multiple attributes that can be used to make a device fingerprint. In addition to that, author described solutions to building different design elements.

Author did not talk about making an application as a whole, but described some of the parts of the application that were made by the author. The things mentioned in this thesis are tested and ready for public use. Author talks only about the things that author found interesting and did not talk about things that were not fully implemented by himself or were too confidential to be published in this thesis.

2. Kasutatud materjalid

1. AccelerateInterpolator | Android Developers [WWW]
http://developer.android.com/reference/android/view/animation/AccelerateInterpolator.html
(06.05.2015)

2. Android 4.2 APIs | Android Developers [WWW]
http://developer.android.com/about/versions/android-4.2.html (05.04.2015)

3. android.view.animation | Android Developers [WWW]
http://developer.android.com/reference/android/view/animation/package-summary.html
(04.05.2015)

4. Animation | Android Developers [WWW]
http://developer.android.com/reference/android/view/animation/Animation.html (05.05.2015)

5. Animation.AnimationListener | Android Developers [WWW]
http://developer.android.com/reference/android/view/animation/Animation.AnimationListener
.html (04.05.2015)

6. ArrayAdapter | Android Developers [WWW]
http://developer.android.com/reference/android/widget/ArrayAdapter.html (05.05.2015)

7. App Manifest [WWW] http://developer.android.com/guide/topics/manifest/manifest-
intro.html (07.05.2015)

8. AttributeSet | Android Developers [WWW]
http://developer.android.com/reference/android/util/AttributeSet.html (05.05.2015)

9. Avaleht > Mobiil-ID > ID.ee [WWW] http://id.ee/index.php?id=36809 (07.05.2015)

10. Build | Android Developers [WWW]
http://developer.android.com/reference/android/os/Build.html (05.04.2015)

11. Build.VERSION | Android Developers [WWW]
http://developer.android.com/reference/android/os/Build.VERSION.html (05.04.2015)

12. Buttons: Floating Action Button [WWW]
http://www.google.com/design/spec/components/buttons-floating-action-button.html#
(04.05.2015)

13. Canvas and Drawables [WWW] http://developer.android.com/guide/topics/graphics/2d-
graphics.html#nine-patch (04.05.2015)

14. Creating a View Class | Android Developers [WWW]
http://developer.android.com/training/custom-views/create-view.html (05.05.2015)

15. Creating an Input Method | Android Developers [WWW]
http://developer.android.com/guide/topics/text/creating-input-method.html (04.05.2015)

16. Dashboards | Android Developers [WWW]
https://developer.android.com/about/dashboards/index.html (01.05.2015)

17. Defining Shadows and Clipping Views [WWW]
https://developer.android.com/training/material/shadows-clipping.html (04.05.2015)

18. Draw 9-patch [WWW] http://developer.android.com/tools/help/draw9patch.html
(04.05.2015)

19. Foreigners, stay away! | Lauri Antalainen [WWW]

https://antalainen.wordpress.com/2011/09/19/foreigner-stay-away/ (13.05.2015)

20. FrameLayout | Android Developers [WWW]
http://developer.android.com/reference/android/widget/FrameLayout.html (04.05.2015)

21. Getting Started – Google Places API for Android – Google Developers [WWW]

https://developers.google.com/places/android/start (07.05.2015)

22. How to retrive the Device Unique ID from android device [WWW]
http://developer.samsung.com/technical-doc/view.do?v=T000000103 (05.04.2015)

23. Identifying App Installations [WWW] http://android-
developers.blogspot.se/2011/03/identifying-app-installations.html (05.04.2015)

24. Intents and Intent Filters [WWW] http://developer.android.com/guide/components/intents-

filters.html (15.05.2015)

25. International | Best Student Account Provider 2013 – Barclays [WWW]

http://www.barclays.co.uk/P1242620130417 (13.05.2015)

26. Interpolator [WWW]
http://developer.android.com/reference/android/view/animation/Interpolator.html
(06.05.2015)

27. LayoutInflator [WWW]

http://developer.android.com/reference/android/view/LayoutInflater.html (06.05.2015)

28. Layouts | Android Developers [WWW]
http://developer.android.com/guide/topics/ui/declaring-layout.html (04.05.2015)

29. List View | Android Developers [WWW]
http://developer.android.com/guide/topics/ui/layout/listview.html (04.05.2015)

30. ListView | Android Developers [WWW]
http://developer.android.com/reference/android/widget/ListView.html (04.05.2015)

31. Lloyds Bank – Banking With Us – Moving To The UK – New To UK [WWW]
http://www.lloydsbank.com/banking-with-us/joining-lloyds/new-to-the-uk.asp (01.04.2015)

32. Lloyds Bank – Legal – Proof Of Identity – What Forms of ID Can I Use? [WWW]
http://www.lloydsbank.com/legal/proof-of-identity.asp (13.05.2015)

33. Manifest.permission | Android Developers [WWW]
http://developer.android.com/reference/android/Manifest.permission.html (07.05.2015)

34. More Resource Types [WWW] http://developer.android.com/guide/topics/resources/more-resources.html (06.05.2015)

35. MotionEvent [WWW]
http://developer.android.com/reference/android/view/MotionEvent.html (06.05.2015)

36. Non UK Resident Bank Accounts | International Banking | Barclays
https://wealth.barclays.com/en_gb/home/international-banking/who-we-help/resident-non-doms/non-resident-bank-account.html (01.04.2015)

37. Noored on moobiliäppi tõttu valmis panka vahetama – Forte [WWW]
http://forte.delfi.ee/news/tarkvara/noored-on-mobiiliapi-tottu-valmis-panka-vahetama?id=71461397 (15.05.2015)

38. Opening a Bank Account Online – Apply for a Bank Account | HSBC Bank UK [WWW]
http://www.hsbc.co.uk/1/2/current-accounts/bank-account/open-bank-account (14.05.2015)

39. Opening a NatWest current account – What do you need? | NatWest current accounts [WWW] http://personal.natwest.com/personal/current-accounts/what-do-you-need-to-open-a-current-account.html (13.05.2015)

40. Place Autocomplete – Google Places API for Android – Google Developers [WWW]
https://developers.google.com/places/android/autocomplete (07.05.2015)

41. Place IDs – Google Places API – Google Developers [WWW]
https://developers.google.com/places/place-id (07.05.2015)

42. Providing Resources | Android Developers [WWW]
http://developer.android.com/guide/topics/resources/providing-resources.html (05.05.2015)

43. Resources.Theme | Android Developers [WWW]

http://developer.android.com/reference/android/content/res/Resources.Theme.html

(13.05.2015)

44. RelativeLayout | Android Developers [WWW]
http://developer.android.com/reference/android/widget/RelativeLayout.html (04.05.2015)

45. ScaleAnimation | Android Developers [WWW]
http://developer.android.com/reference/android/view/animation/ScaleAnimation.html
(06.05.2015)

46. Services | Android Developers [WWW]

http://developer.android.com/guide/components/services.html (07.05.2015)

47. Settings.Secure [WWW]
http://developer.android.com/reference/android/provider/Settings.Secure.html (05.04.2015)

48. ShapeDrawable | Android Developers [WWW]
http://developer.android.com/reference/android/graphics/drawable/ShapeDrawable.html
(04.05.2015)

49. Spinner | Android Developers [WWW]
http://developer.android.com/reference/android/widget/Spinner.html (05.05.2015)

50. Spinners | Android Developers [WWW]
http://developer.android.com/guide/topics/ui/controls/spinner.html (05.05.2015)

51. Supporting Multiple Screens [WWW]
http://developer.android.com/guide/practices/screens_support.html (06.05.2015)

52. System | Android Developers [WWW]
http://developer.android.com/reference/java/lang/System.html (05.04.2015)

53. TelephonyManager [WWW]
http://developer.android.com/reference/android/telephony/TelephonyManager.html
(05.04.2015)

54. TextView | Android Developers [WWW]
http://developer.android.com/reference/android/widget/TextView.html (06.05.2015)

55. TypedArray | Android Developers [WWW]

http://developer.android.com/reference/android/content/res/TypedArray.html (13.05.2015)

56. Vallaste, H. (2000-2015). e-Teatmik: IT ja sidetehnika seletav sõnaraamat. [WWW]
http://www.vallaste.ee (05.04.2015)

57. View Animation [WWW] http://developer.android.com/guide/topics/graphics/view-animation.html (06.05.2015)

58. ViewGroup.MarginLayoutParams | Android Developers [WWW]

http://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html

(07.05.2015)

59. View.OnClickListener [WWW]

http://developer.android.com/reference/android/view/View.OnClickListener.html (06.05.2015

60. View.OnTouchListener [WWW]
http://developer.android.com/reference/android/view/View.OnTouchListener.html
(06.05.2015)

61. WifiManager | Android Developers [WWW]

http://developer.android.com/reference/android/net/wifi/WifiManager.html (13.05.2015)