TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

ITI40LT

Martin Andreas Maarand 134858IAPB

LIFERAY PORTAALILE VEEBITEENUSE JA SELLELE ANDROIDI KLIENDI ARENDAMINE

Bakalaureusetöö

Juhendaja: Aivo Anier
Magister
Tarkvaraarhitekt

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: Martin Andreas Maarand

23.05.2016

Annotatsioon

Antud töö eesmärk on analüüsida kohalike omavalitsuste vajadusi seoses elanike teavitamisega. Selle põhjal luuakse veebiteenus, mis suudab Liferay kohalike omavalitsuste teenusportaali *Uudised ja teated* portletist uudiseid hankida. Lisaks arendatakse Androidi rakendus, millega saab eelmainitud uudiseid vaadata ja mis suudab vajadusel kasutajat teavitada uutest uudistest.

Töö tulemuseks on Liferay veebiteenus, millelt saab uudiseid erinevate parameetrite abil pärida ja Androidi rakendus, mis tarbib eelmainitud veebiteenust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 5 peatükki, 8 joonist.

Abstract

Development of a Web Service and its Android Client for Liferay Portal

The aim of this thesis is to analyze the needs of local governments in Estonia about notifying their residents. On the basis of this analysis the web service for Liferay portal is developed. The aforementioned web service will query news from the *News and Notices* (*Uudised ja teated*) portlet of Liferay service portal of local governments (KOVTP). In addition, the Android application, which can query the web service for news is developed. The Android application is capable of showing the news to the user and notify the user about the news, if needed.

The Liferay web service, which can be queried for news filtered by different parameters and the Android application, which consumes the web service is developed as a result of this thesis.

The thesis is in Estonian and contains 37 pages of text, 5 chapters, 8 figures.

Lühendite ja mõistete sõnastik

Lühendid:

KOV – kohalik omavalitsus

KOVTP – Kohalike omavalitsuste teenusportaal [1]

RSS – Rich Site Summary on uudisvoo vormingu standard

API – application programming interface ehk rakendusliides, mille kaudu saab rakendusega suhelda

CSRF – Cross-site request forgery on pahatahtlik veebilehe rünnak, kus kasutatakse ära veebilehe usaldust kasutaja suhtes

JSON – JavaScript Object Notation on andmevahetusvorming, mis põhineb JavaScript-i süntaksil

XML – Extensible Markup Language ehk laiendatav märgistuskeel on andmevahetusvorming

REST – representational state transfer on tarkvara arhitektuuri stiil

HTTP – Hypertext Transfer Protocol ehk hüperteksti edastusprotokoll on andmevahetuse protokoll

GET – HTTP meetod andmete saamiseks

LED – light-emitting diode ehk valgusdiood

CRUD – create, read, update, delete ehk põhilised andmete muutmise operatsioonid: loomine, lugemine, uuendamine ja kustutamine

URL – Uniform Resource Locator ehk internetiaadress

SDK – software development kit on hulk tarkvara arendamise tööriistu, mis võimaldab mingil kindlal platvormil arendada

APK – Android application package on failitüüp Androidi rakenduste jaoks

HTML – HyperText Markup Language on veebilehtede märgendkeel

ORM – Object-relational mapping seob objekti mudeli relatsioonilise andmebaasiga

Mõisted:

veebiportaal ehk portaal – veebilehekülg paljude alateemade ja teenustega

portlet – kasutajaliidese komponent, mida saab veebiportaali kergelt lisada ja saab toimida teistest komponentidest sõltumatult

autentimispiletike – (i.k. authentication token) juhuslik sõne, millega server tuvastab sessiooni käigus kasutajat

piletike – (i.k. token) juhuslik sõne

kasutaja – Androidi rakenduse kasutaja, KOV-i elanik

git – hajutatud versioonihaldustarkvara

Sisukord

Jooniste loetelu

1 Sissejuhatus

Kohalike omavalitsuste teenusportaal, mida pakub AS Andmevara, on veebikeskkond, mis võimaldab inimestel kohalike omavalitsustega suhelda, infot saada ja avaldusi esitada. KOVTP-s on komponent (portlet) *Uudised ja teated*, kus kuvatakse uudiseid ja teateid, mida KOV on pidanud vajalikuks elanikele kuvada. Kuna *Uudised ja teated* portletti ei ilmu väga tihti uut infot, siis ei käi ka KOV-i elanikud eriti tihti vaatamas, kas on midagi uut tekkinud. Sellest tulenevalt võib juhtuda, et KOV annab mingisuguse olulise teate, aga elanikud märkavad seda alles mitu päeva hiljem ning teatest pole mingit kasu.

Antud töö eesmärgiks on luua Androidi mobiilirakendus, mis kuvaks operatiivselt *Uudised ja teated* portletis olevat infot. Sõltuvalt uudise olulisusest tuleb kasutajat teavitada ehk kui tekib mingi uudis, mida on vaja kiiresti kõigile kuulutada, siis saab nutitelefon, kõiki võimalikke teavitusfunktsioone kasutades, kasutaja tähelepanu püüda. Juhul, kui tegemist ei ole olulise uudisega, siis võib kasutajat kas minimaalselt teavitada või üldse mitte.

Mobiilirakendusse uudiste saamiseks on vaja luua veebiteenus, mis hangiks *Uudised ja teated* portletis olevad andmed ning suudaks need mobiilirakendusele sellisel viisil edastada, et mobiilirakenduse nõuded oleksid täidetavad. Veebiteenuse loomiseks on vaja kasutada Liferay'd, sest KOVTP kasutab Liferay'd.

Lõpptulemusena valmivad veebiteenus, millelt saab uudiseid pärida, ja esialgne Androidi rakendus põhifunktsionaalsusega.

2 Taust

Projekti arenduses on tavaks, et kõigepealt tehakse taustaanalüüs, et näha, kuidas varasemalt on sarnasele probleemile lähenetud või kas keegi on juba leidnud sobiva lahenduse antud probleemile.

2.1 Varasemad lahendused

Kasutajatele uudiste kiireks edastamiseks kerkib esile peamiselt kaks meetodit:

- RSS-voog
- Eraldiseisev rakendus

RSS-voog

Eesti populaarsetest uudisteportaalidest näiteks Postimees, Õhtuleht ja Delfi omavad kõik RSS-voogu.

RSS-voo eeliseks on see, et leidub palju rakendusi, mis suudavad RSS-voogu lugeda. Seega ei ole vaja luua eraldi mobiilirakendust ja serverile RSS-i lisamiseks on mitmeid teeke. RSS-voo rakenduste rohkuse põhjuseks on see, et mõnda aega tagasi olid blogid väga populaarsed ja RSS-voo lugeja oli kasulik tööriist.

RSS-voo puuduseks on see, et selle populaarus langes, kui tekkisid uued sotsiaalvõrgustikud, näiteks Facebook või Twitter, millega on palju lihtsam kedagi jälgida ja teavitusi saada, piisab ainult „Jälgi” nupu vajutusest. Erinevalt RSS-ist, kus tuleb RSS-i voo linke lõigata ja kleepida vastavasse rakendusse. See võib tunduda kasutajale tüütu ja ebamugav ning ta loobub RSS-iga tegelemisest.

Eraldiseisev rakendus

Eesti populaarsetest uudisteportaalidest näiteks Postimees, Õhtuleht ja Delfi omavad kõik ka eraldiseisvat mobiilirakendust uudiste lugemiseks.

Eraldiseisva rakenduse eeliseks on asjaolu, et see võimaldab teavitusi kasutades kasutajale märku anda, kui on uusi uudiseid ilmunud. Lisaks saab kasutaja valida, missugustest uudistest ta on rohkem huvitatud ning tekib võimalus kuvada kasutajale personaalseid uudiseid. Kuna tänapäeval on veebilehekülje mahult suureks kasvanud, siis võib kehva võrguühendusega veebilehe laadimine võtta liiga kaua aega, mis halvendab kasutajakogemust. Eraldiseisva rakenduse puhul on selle välimus ja käitumine ainult üks kord vaja alla laadida ning edaspidi kasutatakse võrguühendust ainult uudiste saamiseks, mis vähendab kasutatavat andmemahtu.

Eraldiseisva rakenduse puuduseks on see, et kasutaja peab iga veebilehe jaoks eraldi rakenduse installima, mis võib nutitelefoni piiratud sisemälust liiga palju ruumi võtta ning nutitelefoni aeglasemaks muuta. Lisaks on puuduseks see, et on vaja arendada eraldi rakendus, mis nõuab ressurssi. Samuti võib juhtuda, et serveripoolset koodi tuleb palju muuta, et oleks võimalik uudiseid saada veebilehitsejas olevast HTML-lehest eraldi.

2.2 Kasutatud tehnoloogiad

Liferay portaal

Liferay [2] portaal on Liferay poolt arendatud vabavaraline raamistik, mis võimaldab arendada keerukaid veebilehekülgi, mis koosnevad paljudest komponentidest, milleks on portletid. Lisaks on Liferay hea sisuhaldussüsteem, millega administraator saab soovi korral veebilehele lisada uusi portlette, lisada sisu ja muuta veebilehe välimust. Liferay portaal on hästi skaleeruv ning suudab toimida ka suure hulga kasutajatega, kui portaal vastavalt konfigureerida [3] . Versiooniks on 6.1, kuna KOVTP on arendatud Liferay 6.1 abil. Lisaks on Liferay portaalile portleti arendamisel hea see, et osa tööd on arendaja eest ära tehtud. Sellest tulenevalt on alguses õppimiskurv (i.k. learning curve) võrdlemisi järsk, kuid kui aru saada, kuidas protsessid toimivad, siis on edaspidi võimalik kiiremini arendada.

Liferay Service Builder

Service Builder [4] on Liferay tehtud mudelipõhine koodigenereermise tööriist, mis võimaldab defineerida kasutaja poolt loodud mudeleid. Service Builder loob teenuse

kihi ORM-tehnoloogia abil, mis tekitab selge eraldatuse mudeli ja andmebaasi koodi vahel. Service Builder genereerib XML-faili põhjal mudeli, teenuskihi, andmebaasi tabeli ja koodi lihtsamate CRUD-operatsioonide täitmiseks. Lisaks genereerib liidese, millele saab HTTP-ühenduse kaudu päringuid teha.

Android SDK

Android [5] on kõige populaarsem nutitelefonide operatsioonisüsteem maailmas. 2015. aasta seisuga kasutavad 82.8% nutitelefonidest Androidi operatsioonisüsteemi [6] . Androidi looja on Google ja enamasti kasutatakse põlisrakenduste (i.k. native application) tegemiseks Java keelt.

Kuna Android on laialdaselt levinud, siis esineb ta paljude erineva riistvaralise võimekusega nutitelefonidel, mistõttu on mitu Androidi versiooni korraga kasutusel ja sellega tuleb arvestada. Erinevalt näiteks Apple'i iPhone'st, kus enamik kasutajaid kasutavad sama operatsioonisüsteemi.

Android SDK on hulk arendustööriistu Androidi platvormile rakenduste arendamiseks, näiteks silumistööriist (i.k. debugger), Androidi emulaator, vajalikud teegid. Iga Androidi versiooni jaoks on vastav Android SDK.

3 Analüüs

Käesolevas peatükis analüüsitakse, mida täpsemalt on vaja arendada, missugused on nõuded ja vajadused ning kuidas tekkivaid probleeme lahendada.

3.1 Kohaliku omavalitsuse vajadused

KOVTP kaudu on inimestel võimalik suhelda kohaliku omavalitsusega. Inimesed saavad esitada avaldusi ja vaadata, mis KOV-is toimub. KOV-il on portlet, kus kuvatakse uudiseid ja teateid. Kuid oluline informatsioon ei jõua inimesteni piisavalt kiiresti, sest ei käida piisavalt tihti KOV-i kodulehel.

Seega on vaja sellist lahendust, mis teavitaks KOV-i elanikke, kui uudiseid on juurde tekkinud. Antud probleemi lahendamiseks on kaks varianti:

- RSS-voog
- mobiilirakendus

RSS-voog

RSS-voo jaoks on vaja, et kasutajad omaksid rakendust, mis suudab RSS-voogu lugeda. Kuid paljud kasutajad ei pruugi teada RSS-i olemasolust ega oska seda kasutada.

Mobiilirakendus

Kuna nutitelefonidel on head võimalused kasutajale teavituste esitamiseks ja enamik nutitelefoni kasutajaid hoiab nutitelefoni enda läheduses, siis on võimalik kasutajat kiiresti teavitada. Lisaks on tellimusena arendatud rakendusega lihtsam kontrollida, kuidas kasutajat teavitada (sõltuvalt uudise olulisusest kasutada heli, värinat, LED-tulukest). Samuti võimaldab eraldi rakendus soovi korral lisada lisafunktsionaalsust, et pakkuda kasutajale rohkemat. KOV on huvitatud, et uudised ja teated jõuaksid paljude

inimesteni ning leiaksid kõlapinda. Seetõttu on vaja, et uudiseid saaks Facebookiga "Like"-da.

Kuna Android on kõige populaarsem mobiilioperatsioonisüsteem, siis on mõistlik arendada esialgu Androidi rakendus, et näha, kas leidub kasutajaid, kes oleksid sellisest rakendusest huvitatud.

3.1.1 Nõuete analüüs

Rakenduse funktsionaalsed nõuded:

- Rakendus peab olema ühendatud KOV-i kodulehega
- Rakendus peab kuvama uudised nimekirjana
- Rakenduses peab nägema uudise pilti, pealkirja ja lühikirjeldust
- Uudise vaates peab olema link, mis suunab antud uudise KOVTP lehele, kogu sisu nägemiseks
- Rakendus peab vastavalt uudise olulisusele kasutajat teavitama
- Rakendus peab sisaldama seadete vaadet, kus kasutaja saab rakendust seadistada
- Kasutaja peab saama valida teda huvitavad sildid
- Rakendus peab kuvama ainult uudiseid, millel on kasutajat huvitavad sildid
- Uudiseid peab saama Facebookiga "Like"-da, kui Facebooki rakendus on installitud
- Kasutaja peab saama vaadata ka vanu uudiseid
- Rakendus peab uudiste nimekirjas kuvama juba vaadatud uudised teistest erinevalt
- Rakendus peab sisaldama linke KOVTP teistesse rakendustesse

Rakenduse mittefunktsionaalsed nõuded:

- KOVTP *Uudised ja teated* peavad võimalikult ruttu pärast ilmumist rakendusse jõudma
- Rakendus peab töötama vähemalt Android 4.0.3-ga (API level 15), sest KOVTP-ga seotud rakendus Anna teada toetab versiooni Android 4.0.3 ja uuemaid

- Rakendus peab vigade korral kasutajat teavitama, ei tohi tüüpvigade korral „kokku joosta”

Serveripoole funktsionaalsed nõuded:

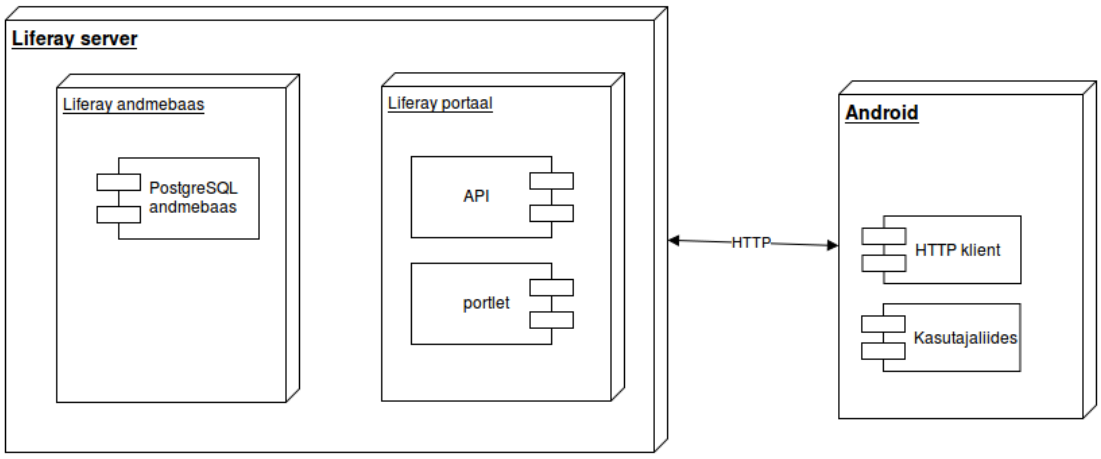
- Server peab olema tarnitud (i.k. deployed) KOVTP-sse
- Serveri meetodid peavad olema olekuta
- Kõik vajalikud parameetrid tuleb iga päringuga kliendi poolt kaasa anda
- **Päringu korral:**
  - Serveril peab olema meetod, millega saab pärida *Uudised ja teated* portletis olevaid uudiseid
  - Server peab tagastama uudised pärast mingit kuupäeva, mis on päringu parameetris
  - Server peab tagastama uudised, millel on päringu parameetrist tulevad sildid
  - Serveril peab olema meetod, millega saab pärida kõik olemasolevad sildid
  - Server peab tagastama vastused JSON kujul

Serveripoole mittefunktsionaalsed nõuded:

- Server peab olema kaitstud CSRF-i eest
- Serveri API peab olema avalik

3.2 Üldine struktuur

Antud projekt koosneb kolmest osast: eksisteeriv Liferay server, kus on andmed, mis on vaja kätte saada, API, mille kaudu saab Liferay serveriga suhelda ja Androidi rakendus, kus kuvatakse valitud andmed töödeldud kujul. Joonis 1 annab üldmulje, missugune on projekti struktuur.



Joonis 1. Tarnemudel

3.3 Siltide süsteem

Uudised ja teated on jagatud tähtsuse järgi viieks: kriitiline, väga oluline, oluline, hea teada ja väheoluline. Mida tähtsam teade on, seda rohkem antakse kasutajale märku (heli, vibreerimine, LED-tuluke).

- "kriitiline" - väga tähtis informatsioon (kasutada suhteliselt harva).
- "väga oluline" - väga oluline informatsioon, näiteks lumesahk on katki ja teid puhtaks ei lükata.
- "oluline" - oluline informatsioon, näiteks vallamaja on mingil põhjusel suletud.
- "hea teada"- kasulik informatsioon, näiteks toimub mingi üritus.
- "väheoluline" - vähetähtis informatsioon, kuid oleks ikkagi hea, et inimesed teaksid sellest.

Loomulikult võib lisada ka muid silte, kuid eelnevalt loetletud siltide kasutamine on oluline Androidi rakenduse heaks toimimiseks.

3.4 Administraatori vaade

Administraatori (KOV-i töötaja) vaadet Androidi rakenduses ei ole. Administraator suhtleb ainult Liferay portaalil oleva kasutajaliidesega. Antud projekti lisamine Liferay serverisse administraatorile väga palju lisatööd ei tekita.

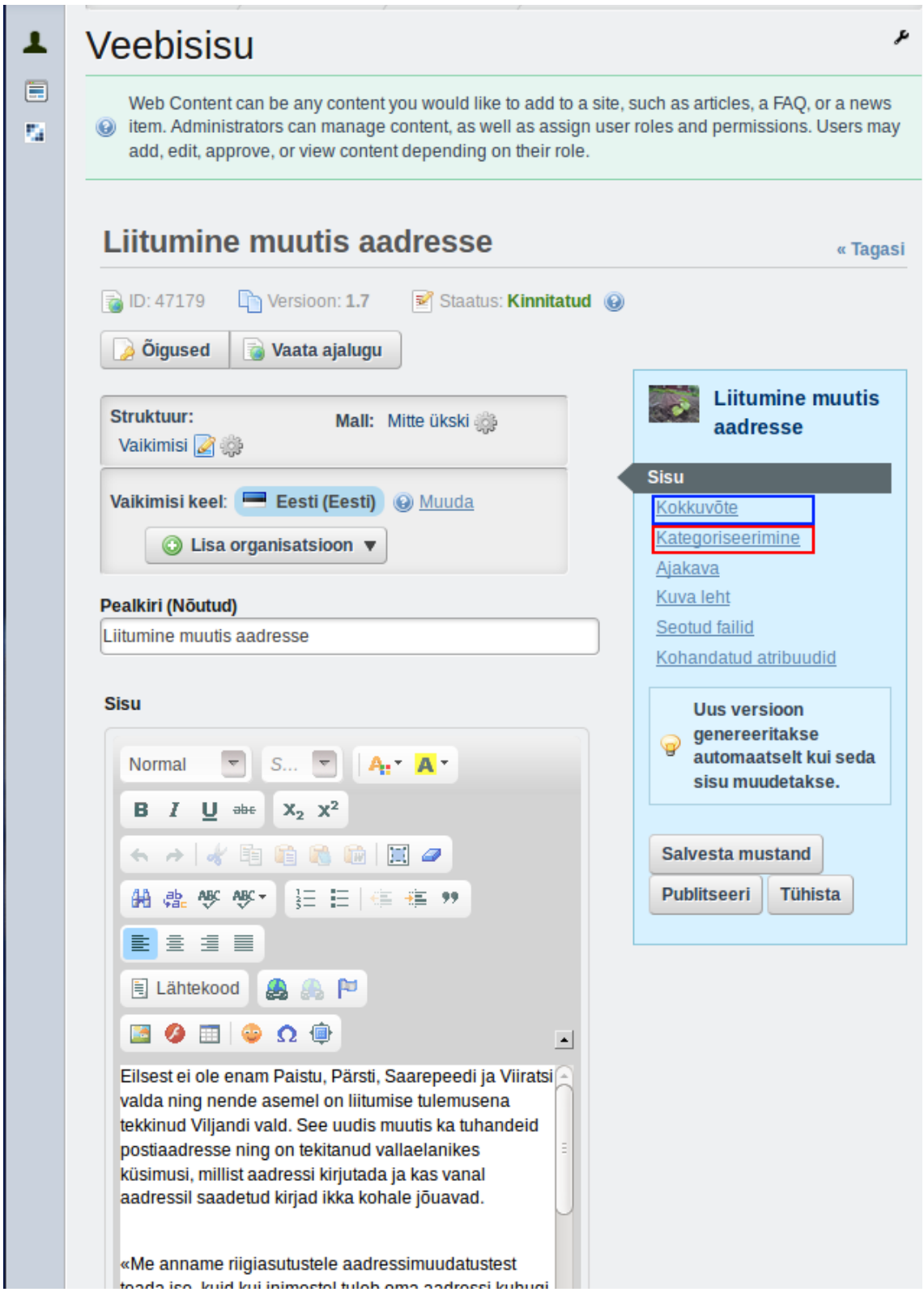
Selleks, et Androidi rakendus saaks mõistlikult töötada, peaks administraator uudised ja teated nende sisestamisel sildistama. Võimaldades seeläbi hiljem uudiseid Androidi rakenduse jaoks filtreerida. Samuti oleks väga hea, kui administraator täidaks uudiste ja teadete sisestamisel ka lühikirjelduse välja, et Androidi kasutaja saaks rohkem infot.

Administraator võib ka loobuda nende ülesannete täitmisest, kuid sellisel juhul Androidi rakenduse kasulikkus väheneb märgatavalt.

Administraatori juhend:

Liferay administraatori vaates tuleb lisada uus veebisisu. Tuleb täita ära pealkiri ja uudise sisu. Seejärel valida kokkuvõte ja kirjutada kokkuvõte. Soovi korral võib lisada ka pildi. Järgmisena tuleb valida kategoriseerimine, kus siltide alt tuleb valida silt, mis iseloomustab uudise tähtsust. Loomulikult võib lisada ka muid silte. Lisaks tuleb valida ka kategooria (i.k topic) „Uudis”, et uudis ilmuks *Uudised ja teated* portletti ning seeläbi ka Androidi rakendusse. Lõpetuseks tuleb uudis publitseerida.

Joonisel 2 on näha veebisisu lisamine. Paremal helesinise taustaga kastis on näha kahte joonistatud kasti. Sinises kastis on link nimega „Kokkuvõte”, mis avab vaate, kus saab lisada uudisele lühikokkuvõtte. Punases kastis on link nimega „Kategoriseerimine”, mis avab vaate, kus saab lisada uudisele silte ja kategooriaid.



Joonis 2. Liferay admininistraatori vaate kuvatõmmis

3.5 Kasutaja vaade

Androidi rakenduses on ainult tavakasutaja vaade, ei ole mingeid erilisi rolle. Kasutaja saab vaadata KOV-i uudiseid ja teateid. Kasutaja saab seadistada parameetreid, mille järgi tehakse API-le päringuid. Kasutaja saab muuta vanima kuupäeva parameetrit ja tänu sellele vaadata ka juba loetud uudiseid. Varasemalt loetud uudised peavad olema eristatavad lugemata uudistest, näiteks nimekirja vaates halli taustaga. Samuti saab kasutaja Facebookiga uudiseid “Like”-da. Lisaks on võimalus minna KOV-i kodulehele, vaadata KOV-i kontakte või avada rakendus „Anna teada”, mis võimaldab KOV-i probleemidest teavitada.

Soovi korral saab kasutaja panna rakenduse regulaarselt pärima, kas on uusi uudiseid. Kui on tekkinud uusi uudiseid, siis sõltuvalt kasutaja valitud siltidest vastab server nimekirjaga sobivatest uudistest. Sõltuvalt kokkulepitud siltide süsteemist otsustab Androidi rakendus nimekirjas olevate siltide põhjal, kuidas kasutajat teavitada uutest uudistest.

Kasutaja teavitamine erinevate siltide korral:

- "kriitiline" – kasutab kõiki võimalikke viise kasutajale märku andmiseks iga kord, kui serverilt tuleb vastus, nii kaua kuni kasutaja reageerib
- "väga oluline" – kasutab kõiki võimalikke viise kasutajale märku andmiseks, kuid ainult üks kord, korduvalt ei teavitata
- "oluline" – mängib teavitusheli
- "hea teada" – vibreerib
- "väheoluline" – ei saada teavitust, kuid kuvatakse, kui kasutaja siseneb rakenduse uudiste nimekirja vaatesse

3.6 API loomine

Androidi rakendus ja Liferay serveri vaheliseks suhtluseks tuleb luua API. Kuna JSON-formaat on lihtsam ja tabeli väljade väikese arvu korral pole vajadust XML-i eriliste tüübi ja formaadi valideerimisvõimaluste järele ning JSON veebiteenused on väga populaarseks muutnud, siis tundub JSON-i kasutamine mõistlikum. Samuti on JSON-i

kasutamine Androidiga mugav, kuna leidub lihtsasti kasutatavaid JSON-i serialiseerimise ja deserialiseerimise teeke.

REST-i korral on API loomine suhteliselt lihtne, kuna server ei pea olekut hoidma, sest meetodite toimimiseks vajalikud parameetrid antakse alati päringuga kaasa. Tänu sellele on serveripoole keerukus väiksem ja kergem arendada. API on kergekaaluline ja võimaldab paljudel kasutajatel korraga päringuid teha, sest pole vaja mälus hoida olekuid, mistõttu kulub vähem ressurssi ühe kasutaja kohta. Samuti ei teki probleeme, kus, mingi vea tõttu lõimede töös, tehakse päring teise kasutaja olekut kasutades. Androidi rakendus omab ainult tarbija funktsiooni, see tähendab, et ainult küsib informatsiooni, ise uut ei tekita. Seega on vaja realiseerida ainult HTTP GET päringud.

Liferayga loodud veebiteenusega suhtlemiseks kasutatakse p_auth piletikest, mis kehtib Liferay portaalis ja kaitseb CSRF-i eest (i.k. Portal authentication token for CSRF protection). Kuna API-s on ainult GET päringud ja mingit konfidentsiaalset infot nendega kaasa ei tule, siis põhimõtteliselt võiks lubada ka päringuid p_auth-i kasutamata. Kuid Liferay iseärasustest tulenevalt tuleks sellisel juhul p_auth-i kontroll välja lülitada terve portaali ulatuses ning see ei ole kindlasti soovitav tulemus.

Esialgu on vaja kahte meetodit, millelt infot pärida. Esiteks on vaja meetodit, mis tagastaks nimekirja uudiste objektidest, mis on loodud pärast kuupäeva parameetrit ja omaksid vähemalt ühte silti siltide nimekirja parameetrist. Teiseks on vaja meetodit, mis tagastaks kasutuses olevate siltide nimekirja.

4 Realiseerimine ja tulemused

Käesolevas peatükis analüüsitakse, kuidas peavad Androidi rakendus ja Liferay server suhtlema. Samuti on kirjeldatud projekti tehnilisi lahendusi.

4.1 Detailanalüüs

Alljärgnevalt kirjeldatakse, kuidas toimub suhtlus Androidi rakenduse ja Liferay serveri vahel.

4.1.1 *Uudised ja teated* andmeobjekti kirjeldus

Liferay *Uudised ja teated* portlet kasutab andmete hoiustamiseks `JournalArticle` klassi. Enamiku vajalikest andmeväljadest saab `JournalArticle` objektilt, kuid silte ei saa. Siltide saamiseks tuleb pärida `AssetEntry`-lt `AssetTag`-e. Lisaks on `JournalArticle` klassil palju välju ning kuna osal nendest pole Androidi rakenduse jaoks mingit kasu, siis on mõistlik luua uus andmeobjekt. Uus andmeobjekt sisaldab alamhulga `JournalArticle` väljadest ja lisaks siltide välja, mis saadakse `AssetEntry` ja `AssetTag`-i kaudu. Lisas 1 on Liferay `JournalArticle` näide JSON-kujul.

Loodud artikli objekt sisaldab:

- articleId: unikaalne artikli id genereeritud Liferay poolt
- content: artikli kirjeldus
- id: unikaalne artikli versiooni id genereeritud Liferay poolt
- smallImageId: unikaalne artikli pildi id genereeritud Liferay poolt
- smallImageUrl: artikli pildi asukoht
- tags: sildid, mis antud artiklile on lisatud
- title: artikli pealkiri

- type: artikli tüüp Liferay poolt
- urlTitle: artikli pealkiri URL-is

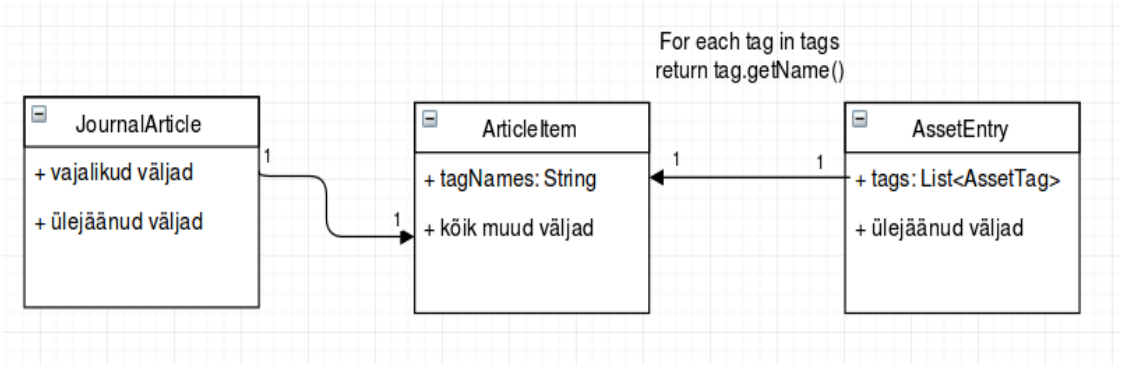
Artikli objekti tulevad väärtused:

- articleId: `JournalArticle.getArticleId()`
- content: `JournalArticle.getDescriptionCurrentValue()`
- id: `JournalArticle.getId()`
- smallImageId: `JournalArticle.getSmallImageId()`
- smallImageUrl: `JournalArticle.getSmallImageURL()`
- tags: `AssetEntry.getTags() -> List<AssetTag> AssetTag.getName()`
- title: `JournalArticle.getTitleCurrentValue()`
- type: `JournalArticle.getType()`
- urlTitle: `JournalArticle.getUrlTitle()`

Aritkli objekti näide JSON-kujul:

```
  {
    "articleId": "50095",
    "content": "11.-17. novembril ehk tänasest pühapäevani toimub juba
viiendat korda üle-eestiline Energiasäästunädal, mille raames andis
Viljandi maavanem Lembit Kruuse \nkoolitunni Olustvere põhikooli 7-9
klassi õpilastele.",
    "id": 94617,
    "smallImageId": 50105,
    "smallImageUrl": "",
    "tags": "",
    "title": "Viljandi maavanem avas Energiasäästunädala koolitunniga
Olustvere Põhikoolis",
    "type": "general",
    "urlTitle": "viljandi-maavanem-avas-energiasaastunadala-
koolitunniga-olustvere-pohikoolis"
  },
```

Joonis 3 näitab `ArticleItem` väljade päritolu.



Joonis 3. Lihtsustatud klassi diagramm

4.1.2 API kirjeldus

Esialgu on vaja luua 2 API-t. Esimene *Uudised ja teated* portletist andmete saamiseks ja teine kasutuselolevate siltide saamiseks.

API *Uudised ja teated* andmete saamiseks on vaja meetodit, mis võtaks parameetriteks:

- groupId (vajalik Liferay päringute tegemiseks)
- dateString (selleks, et pärida ainult uuemaid artikleid, vältida tühja tööd, kuupäeva formaat `HH:mm-dd-MM-yyyy`)
- tagNames (selleks, et tagastada ainult vajaliku sildiga artiklid)
- p_auth (portaali autentimispiletike CSRF kaitseks, mis lisandub API meetodile automaatselt)

ja tagastaks JSON-listi artikli objektidest. Java meetodi signatuur on järgnev:

```
getArticlesByTagsObj(long groupId, String dateString, String[] tagNames, String p_auth)
```

Päring URL-kujul: get-articles-by-tags-obj/group-id/(groupId)/date-string/(dateString)/tag-names/(tagNames)?p_auth=(p_auth token)

Näite URL: http://vald.kovtp.girf.ee/RestPlugin-portlet/api/jsonws/carrot/get-articles-by-tags-obj/group-id/10783/date-string/10:00-01-01-2010/tag-names/all?p_auth=(siia tuleb panna auth token)

Kõik vajalikud parameetrid päringu tegemiseks on meetodis olemas, seega ei pea server olekut hoidma, mis teeb serveri lihtsamaks. API siltide saamiseks on vaja meetodit, mis võtaks parameetriks:

- p_auth (portaali autentimispiletike CSRF kaitseks, mis lisandub API meetodile automaatselt)

ja tagastaks kõik sildid, et oleks teada, mis silte `getArticlesByTagsObj` tagNames parameetrisse saab kaasa anda. Java meetodi signatuur on järgnev:

```
getAllTags(String p_auth)
```

Päring URL-kujul: get-all-tags?p_auth=(p_auth token)

Näite URL: http://vald.kovtp.girf.ee/RestPlugin-portlet/api/jsonws/carrot/get-all-tags?p_auth=(siia tuleb panna auth token)

4.1.3 Androidi rakenduse analüüs

Eelnevas (API kirjelduse) peatükis on kirjas parameetrid, mida Androidi rakendus peab päringute tegemiseks teadma. Need on `long` groupId, `String` dateString, `String[]` tagNames ja `String` p_auth. Järgnevalt tuleb leida võimalused nendele parameetritele väärtuste leidmiseks või loomiseks.

Parameeter `long` groupId on Liferay sisemine andmeväli, mis lisatakse igale `JournalArticle` objektile. Hetkel on testserveris kõik *Uudised ja teated* portletis olevad objektid sama groupId-ga, kuid need võivad olla ka erinevad. Seega tekib võimalus ka groupId abil *Uudised ja teated* sisu filtreerida. Selleks, et Androidi rakenduses oleks teada, mis groupId-d on võimalikud, tuleb Liferay serverisse lisada staatiline JSON-fail, kus on kirjas groupId ja sellele vastav nimi, mis kuvatakse kasutajale.

Näiteks: http://vald.kovtp.girf.ee/RestPlugin-portlet/kov.json

```
{
10101: "Viljandi vald",
10532: "Saku vald",
```

```
10783: "Keila vald",
11034: "Viimsi vald",
60259: "Märjamaa vald",
65875: "Räpina vald"
}
```

Parameeter `String` dateString on `HH:mm-dd-MM-yyyy` formaadis kuupäev, mille loomiseks ei ole vaja Liferay portaaliga suhelda, võib lihtsalt võtta mõne suvalise ajahetke.

Parameeter `String[]` tagNames on massiiv siltidest, mille järgi filtreerida *Uudised ja teated* artikleid. Kasutuselolevate siltide saamiseks tuleb teha päring `getAllTags(String p_auth)` meetodile.

Parameeter `String` p_auth on 8 märgi pikkune tähtedest ja numbritest koosnev juhuslik sõne, mille Liferay portaal genereerib autentimiseks. Kuna API päringute tegemiseks on p_auth-i vaja, siis tuleb enne päringute tegemist p_auth Liferay portaalilt kätte saada. Selleks tuleb Liferay portletis kasutada järgnevat meetodit, millega saadakse p_auth piletike ja siis saab selle kasutajale kuvada.

```
com.liferay.portal.security.auth.AuthTokenUtil.getToken(request)
```

Näiteks: http://vald.kovtp.girf.ee/RestPlugin-portlet/view.jsp

Lisaks on tulevikus vaja võimaldada valikut erinevate KOV-ide portaalide vahel. Selleks on vaja Androidi rakenduse käivitamisel kuvada kasutajale valikut saadaolevatest KOV-idest. Seejärel tuleb Androidi HTTP kliendile anda vastava KOV-i URL ning rakenduses endas kasutada muud kasulikku infot KOV-i kohta. Seega on vaja serveris hoida staatilist JSON-faili, kus on nimekiri teenusega liitunud KOV-idest, nende Liferay portaali URL-idest ja muu vajalik info.

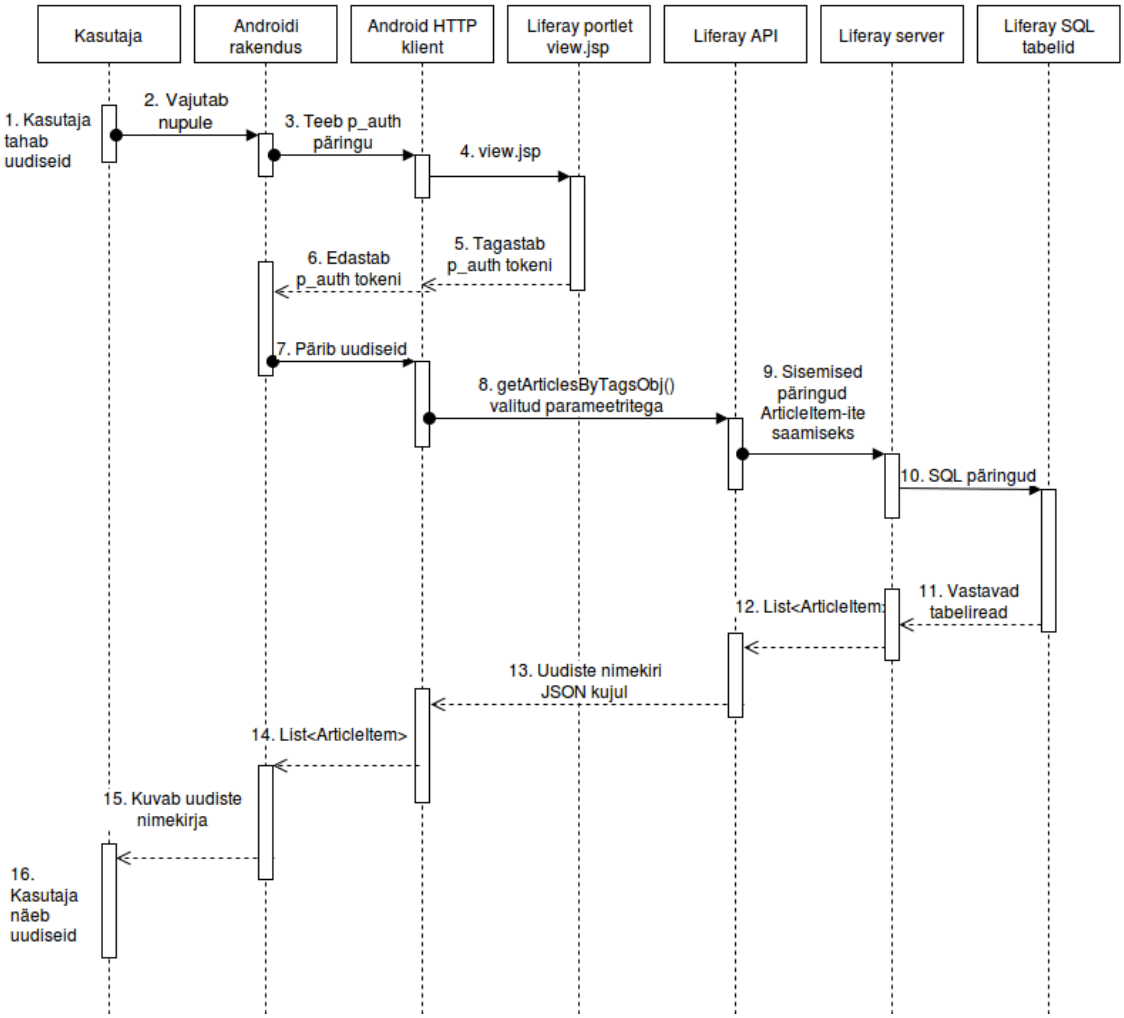
Näiteks: https://api.myjson.com/bins/2863v

```
[{
"kovName":"Saue vald",
"fbPageUrl":"https://www.facebook.com/Sauevald",
"kovtpUrl":"http://vald.kovtp.girf.ee/",
```

"groupId":10783,
"offers":["www.postimees.ee","www.neti.ee","www.eesti.ee"]
}]

Kuna hetkel on kasutusel ainult üks testserver, siis toimub KOV-i info salvestamine automaatselt ja kasutajale valikut ei pakuta.

Joonis 4 näitab põhilist kasutusjuhtu kasutajast andmebaasini.



Joonis 4. Uudiste pärimise kasutusjuhu diagramm

4.2 Liferay API arendamine

Liferay lihtsustab arendaja tööd, võimaldades genereerida mingi osa koodist, mis on suhteliselt üldine ja korduv. Samuti Liferay abstraheerib arendaja jaoks ära andmebaasi

ning suhtlus käib läbi Liferay liidese, näiteks on olemas `AssetEntryQuery` klass, millele saab väljade väärtusi seada, mille järgi Liferay loob päringu, mis tuleb

```
AssetEntryLocalServiceUtil.getEntries(AssetEntryQuery assetEntryQuery)
```

meetodi parameetrisse anda. Seejärel Liferay moodustab SQL-päringu ning tagastab vastava Java objekti. Veel on olemas `DynamicQuery`, millega saab moodustada keerulisemaid päringuid.

Liferay Service Builder-i abil saab luua veebiteenuse kasutaja poolt loodud klassi jaoks, millele genereeritakse meetodid CRUD-operatsioonide jaoks. Antud projektis ei ole vaja luua uut klassi, millele saaks CRUD-operatsioone rakendada, vaid on vaja kahe klassi põhjal moodustada uus klass. Seega saab Service Builder-ile mudeli loomiseks vajalikku XML-faili kirjutada mudeli kirjelduse, kus pole ühtegi välja. Fail `service.xml`, kui see on vastavalt konfigureeritud, lubab loodud mudelit API-s JSON-iga kasutada.

Tulenevalt sellest, et loodud mudelis ei ole ühtegi välja, siis Service Builder ei genereeri ühtegi CRUD-meetodit. Küll aga genereeritakse implementatsiooni klass, kuhu saab isetehtud meetodid panna, millest luuakse liides, millega saab API kaudu suhelda. Liferay Service Builder'i juures on huvitav detail see, et `{mudeli nimi}ServiceImpl.java` klassi lood kõigepealt meetodi implementatsiooni. Seejärel genereeritakse implementatsioonile liides, millega saab API kaudu suhelda.

API genereerimise järel saab minna veebilehekülje {Liferay portaali URL} + api/jsonws, kus on näha kõik avalikud API meetodid. Seda ainult siis, kui antud projekti portlet on Liferay portaali lisatud. Näiteks: http://vald.kovtp.girf.ee/api/jsonws?contextPath=/RestPlugin-portlet

API Siltide saamiseks

```
getAllTags(String p_auth)
```

Sisuliselt on tegemist Liferay

`com.liferay.portlet.asset.service.AssetTagLocalServiceUtil.getTagNames();` meetodi väljakutsega.

API Artiklite saamiseks

`getArticlesByTagsObj(long groupId, String dateString, String[] tagNames, String p_auth)`

Koodist arusaamise lihtsuse huvides kasutab `getArticlesByTagsObj` meetod mitmeid abimeetodeid. API väljakutsel kutsutakse välja kaks abimeetodit.

Esimene – `getArticlesAfterDateModified(String dateString, long groupId)` tagastab `JournalArticle` nimekirja, milles olevate elementide groupId on võrdne meetodi parameetri groupId-ga ja artikli viimane muutmine on toimunud pärast parameetri dateString väärtust. Nimekirjas on igast artiklist ainult viimane versioon, et mitte tagastada ühte artiklit mitu korda.

Teine – `getArticlesByTagsObj(long groupId, String[] tagNames)` tagastab juba nimekirja andmeobjektidest, mille kirjeldus on peatükis 4.1.1 *Uudised ja teated* andmeobjekti kirjeldus. See meetod leiab kõik `AssetEntry` objektid, millel on vähemalt üks tagNames massiivis olev silt ja mis kuvatakse *Uudised ja teated* portletis. Seejärel leitakse vastavale `AssetEntry` objektile kuuluvad sildid ja vastav `JournalArticle` ning ühendatakse API-s kasutatavaks andmeobjektiks.

Seejärel moodustatakse kahe abimeetodi tulemuse põhjal nimekiri API andmeobjektidest, kuhu lisatakse kõik `getArticlesByTagsObj` abimeetodiga saadud objektid, mille id leidub `getArticlesAfterDateModified` abimeetodiga saadud nimekirjas, ehk siis on muudetud pärast dateString parameetris olevat väärtust.

4.3 Androidi rakenduse arendamine

Androidi rakenduse poolt kasutatavad teegid

- com.google.code.gson:gson:2.3.1

  Gson [7] on Google'i poolt arendatud Java teek lihtsaks Java objekti serialiseerimiseks JSON-i ja JSON-i deserialiseerimiseks Java objektiks.

- com.squareup.retrofit:retrofit:1.9.0

Retrofit [8] on Square'i poolt arendatud HTTP klient Java ja Androidi jaoks.

- com.squareup.picasso:picasso:2.5.2

  Picasso [9] on Square'i poolt arendatud võimekas piltide allalaadimise ja puhvrimise (i.k. caching) teek Androidi jaoks.

- com.facebook.android:facebook-android-sdk:4.+

  Facebook-android-sdk [10] on Facebooki poolt arendatud teek, et lihtsustada Androidi ja Facebooki vahelist integreerimist.

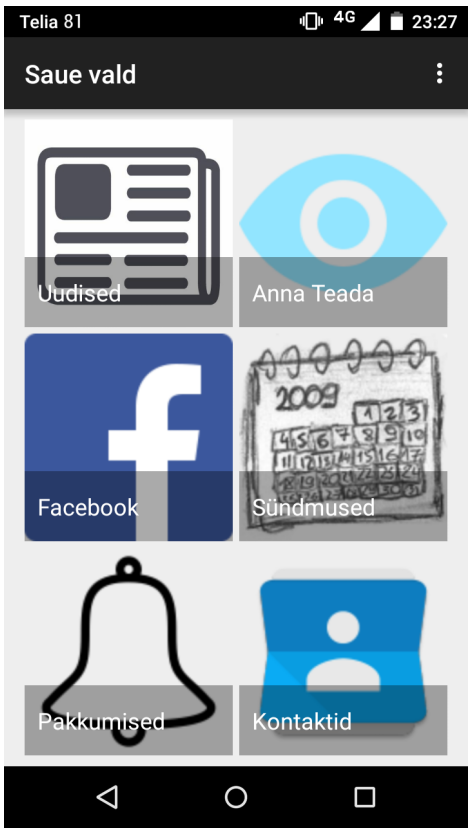
Androidi rakenduse tähtsamad vaated on esileht, artiklite nimekiri, artikli detailvaade ja seaded.

Esileht

Esilehel on ruudustikvaade võimalikest tegevustest. Hetkel on neid 6.

- *Uudised* (kõige olulisem), mis viib artiklite nimekirja vaatesse.
- *Anna teada*, mis avab Anna teada rakenduse, kui see on installitud või avab rakenduse Google Play's.
- *Facebook*, mis avab KOV-i Facebooki lehe Facebooki rakenduses.
- *Sündmused*, mis avab brauseris KOV-i kalendri Liferay portaalis.
- *Pakkumised*, kus kuvatakse KOV-i valitud lingid, mis avanevad brauseris.
- *Kontaktid*, mis avab brauseris KOV-i kontaktide lehe Liferay portaalis.

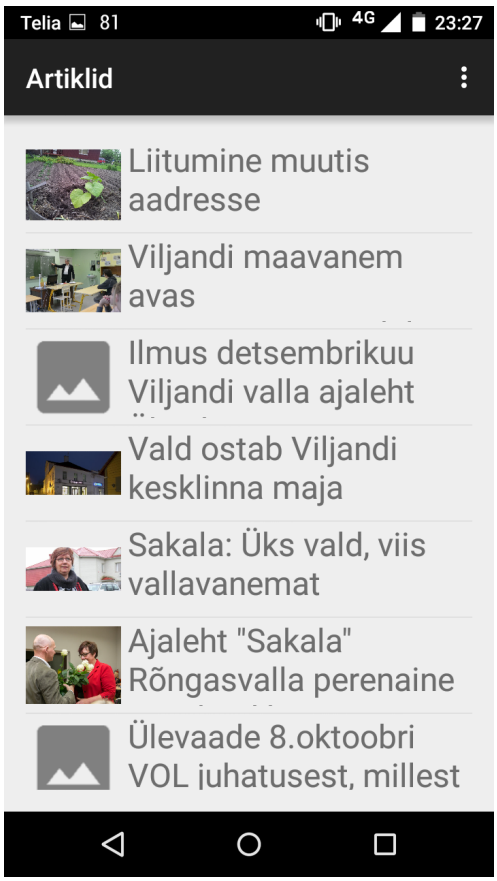
Joonisel 5 on näha Androidi rakenduse Esilehe vaate kuvatõmmis.



Joonis 5. Androidi rakenduse esilehe vaade

Artiklite nimekiri

Artiklite nimekirjas kuvatakse nimekirjavaade uudistest, mis on API kaudu Liferay portaalilt päritud. Nimekirja elemendil on näha uudise pilt (kui pilti ei ole, siis kuvatakse vaikimisi pilt) ja uudise pealkiri. Nimekirja element, mida on juba vaadatud, on teistest eristatav sellega, et taust on halli värvi. Juhul, kui ei ole ühtegi uut uudist, mida kasutajale näidata, siis kuvatakse tekst, mis ütleb, et ei ole uusi uudiseid. Nimekirja elemendil klõpsates avaneb uudise detailvaade. Joonisel 6 on näha Androidi rakenduse *Artiklite nimekiri* vaate kuvatõmmis.



Joonis 6. Androidi rakenduse nimekirja vaade

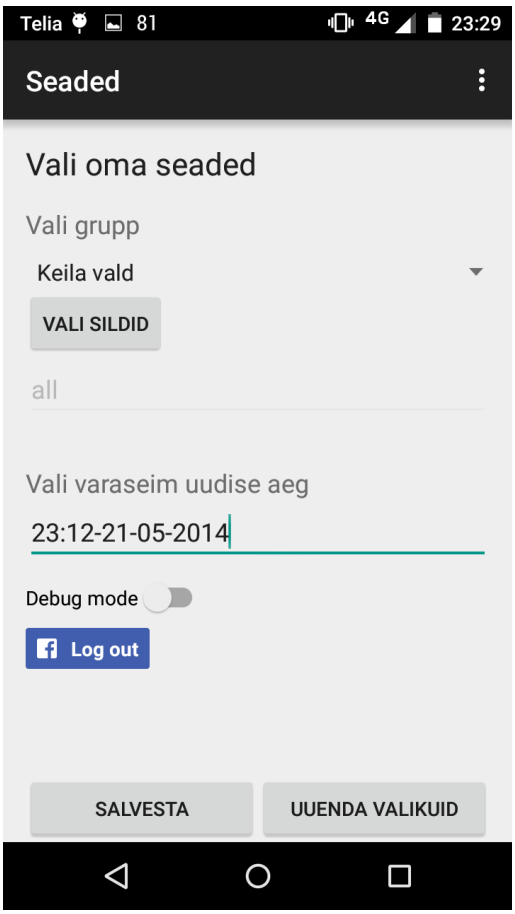
Artikli detailvaade

Artikli detailvaates kuvatakse uudise pilt, pealkiri, sildid ja lühikokkuvõte. Kui pilti või lühikokkuvõtet ei ole, siis näidatakse kasutajale vaikimisi pilti ja kirjeldust. Lisaks on olemas Facebooki „Like” nupp, millega saab antud uudist „Like”-da. Kui Facebooki rakendus ei ole installitud, siis nuppu ei kuvata. Samuti on nupp, mis avab antud uudise brauseris, kus on näha kogu artikli sisu. Joonisel 7 on näha Androidi rakenduse *Artikli detailvaate* kuvatõmmis.

Joonis 7. Androidi rakenduse detailvaade

Seaded

Seadete vaates, saab valida, millise grupi uudiseid serverilt päritakse. Samuti saab valida, missugused sildid päringu siltide parameetrisse lähevad. Lõpetuseks saab valida aja, millest uuemaid uudiseid päritakse. Seadete vaates on ka kaks nuppu. *Salvesta* nupp salvestab kasutaja valitud seaded Androidi SharedPreferences'sse ning kasutab edaspidi päringuid tehes neid parameetreid. *Uuenda valikuid* nupp teeb Liferay serverisse päringu ja uuendab siltide ning gruppide valikut. Joonisel 8 on näha Androidi rakenduse *Seaded* vaate kuvatõmmis.



Joonis 8. Androidi rakenduse seadete vaade

Gradle

Gradle on koodi ehitamise automatiseerimise (i.k. build automation) tööriist. Antud projektis on Androidi sõltuvused lisatud Gradle'i abil. Samuti on loodud Gradle'i skript, mille abil on lihtsustatud uue rakenduse versiooni loomine. Eelmainitud skript kontrollib, kas kõik muudatused on git-is olemas, seejärel käivitab testid ja kui probleeme ei teki, siis uuendab rakenduse versiooni numbrit ning laeb APK-faili kohalikku serverisse. Sealt saab võtta APK ja laadida uus rakenduse versioon Google Play'sse üles.

Androidi testimine

Androidi rakenduse testimiseks kasutatavad teegid

- junit:junit:4.12

JUnit [11] on lihtne raamistik korratavate testide kirjutamiseks.

- org.mockito:mockito-all:1.10.19

  Mockito [12] on ühiktestide jaoks võltsobjektide (i.k mock) loomise raamistik.

- org.robolectric:robolectric:3.0

  Robolectric [13] on raamistik, mis võimaldab testida Androidi kasutajaliidest emulaatorit käivitamata, kuna on loodud väiksem versioon Android SDK-st, mille meetodeid testimisel välja kutsutakse tegeliku Android SDK meetodite asemel.

Ühiktestid

Ühiktestide tegemiseks kasutatakse JUnit'it. Peamiselt testitakse meetodeid, mis ei sisalda Androidi spetsiifilisi muutujaid ja võiksid esineda ka suvalises Java rakenduses, mistõttu on neid kergem testida. Näiteks test, mis testib `MyUtils.getArticlesImportanceFromTags(List<ArticleItem> articleItems)` meetodit ja kontrollib, kas saab etteantud `List<ArticleItem>` nimekirjas olevate elementide siltidest õige kriitlisuse taseme, millest sõltub, kuidas kasutajat teavitada.

Kasutajaliidese testid

Lisaks on loodud mõned kasutajaliidese testid, kuid mitte palju, sest kasutajaliides on hetkel lihtne ja võib palju muutuda. Kasutajaliidese testimiseks on kasutusel Robolectric, mis võimaldab testida ilma, et peaks Androidi emulaatori tööle panema või Androidi telefoni kasutama. Näiteks on testitud *Esilehte*, et kas avaneb õige vaade, kui mõnele ruudustikvaate elemendile vajutada.

Lisaks on rakendust testitud käsitsi Androidi emulaatoritel Android API 15 ja API 23 jaoks ja Motorola Moto G peal (API 22) ning HTC Desire peal (API 15).

5 Kokkuvõte

Käesoleva töö eesmärgiks oli luua Androidi mobiilirakendus, mis suudaks kasutajat teavitada võimalikult kiiresti, kui Liferay KOVTP *Uudised ja teated* portletti ilmuvad uudised. Võib öelda, et eesmärk sai saavutatud – valmis lihtsa kasutajaliidesega mobiilirakendus, mis suudab nõudeid täita. Mobiilirakenduse toimiseks oli vaja luua Liferay serverisse veebiteenus, mis on samuti nõuetelevastav. Nüüd on võimalik uurida, missugune võiks kasutajaliides välja näha ja mis lisafunktsionaalsust veel vaja oleks.

Haldusreformi tõttu tekkib paljudel KOV-i elanikel vajadus varasemast rohkem KOV-iga suhelda. Valdade ühinemise tõttu kaovad mitmed vallavalitsused, mistõttu liigub paljude elanike jaoks vallavalitsus kümnete kilomeetrite kaugusele. Seega võib aeglane uudiste liikumine olla elanikule ajaliselt ja rahaliselt kulukas. Sellest tulenevalt võib öelda, et antud projekt, kui see KOV-ides kasutusele võtta, oleks elanikele kasulik.

Käesolevas töös uuriti, kuidas on võimalik KOVTP-st vajalikku informatsiooni kätte saada. Analüüsiti ja realiseeriti API, mille kaudu toimib Liferay serveri ja Androdi rakenduse vaheline suhtlus. Mõeldi välja siltide süsteem, mille järgi KOV-i töötaja saab märkida uudise tähtsust. Uudise tähtsusest sõltuvalt mobiilirakendus teavitab kasutajat erinevatel viisidel.

Isiklikus plaanis oli antud projekt väga kasulik ja arendav, sest sain kasutada tehnoloogiaid, millega varasemad kokkupuuted puudusid. Liferay'st ja selle toimimisest arusaamine võttis, Liferay keerukuse ja suuruse tõttu, mõnda aega. Kuna Liferay on vabavaraline, siis dokumentatsioon ei ole kõige parem ja probleemidele sobiva lahenduste leidmine võib olla aeganõudev tegevus. Androidile olin varem paar rakendust teinud, kuid need polnud eriti keerukad, seega õppisin ka Androidi vallas palju juurde.

Kasutatud kirjandus

[1] KOVTP “KOVTP võimalused”

[WWW] https://www.kovtp.ee/esileht (Kasutatud 21.05.2016)

[2] Liferay “What is Liferay?”

[WWW] https://dev.liferay.com/discover/portal/-/knowledge_base/6-1/what-is-liferay (Kasutatud 13.05.2016)

[3] Liferay “Liferay configuring for high availability”

[WWW] https://dev.liferay.com/discover/portal/-/knowledge_base/6-1/configuring-liferay-for-high-availability (Kasutatud 19.05.2016)

[4] Liferay “What is Service Builder?”

[WWW] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-1/what-is-service-builder (Kasutatud 1.05.2016)

[5] Android “Android”

[WWW] https://developer.android.com/index.html (Kasutatud 18.05.2016)

[6] IDC “Smartphone OS market share”

[WWW] http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Kasutatud 21.05.2016)

[7] Github “Gson”

[WWW] https://github.com/google/gson (Kasutatud 12.05.2016)

[8] Github “Retrofit”

[WWW] https://github.com/square/retrofit (Kasutatud 12.05.2016)

[9] Github “Picasso”

[WWW] https://github.com/square/picasso (Kasutatud 12.05.2016)

[10] Github “Facebook Android SDK”

[WWW] https://github.com/facebook/facebook-android-sdk (Kasutatud 12.05.2016)

[11] Github “JUnit4”

[WWW] https://github.com/junit-team/junit4/ (Kasutatud 12.05.2016)

[12] Github “Mockito”

[WWW] https://github.com/mockito/mockito (Kasutatud 12.05.2016)

[13] Github “Robolectric”
[WWW] https://github.com/robolectric/robolectric (Kasutatud 12.05.2016)

Lisa 1 – JournalArticle näide JSON kujul


```
{    "articleId": "107302",
    "classNameId": 0,
    "classPK": 0,
    "companyId": 10758,
    "content": "\n\n\n\t\n",
    "createDate": 1450435735910,
    "description": "väike kokkuvõte",
    "descriptionCurrentValue": "väike kokkuvõte",
    "displayDate": 1447412040000,
    "expirationDate": null,
    "groupId": 10783,
    "id": 108576,
    "indexable": true,
    "layoutUuid": "",
    "modifiedDate": 1450435736383,
    "resourcePrimKey": 107304,
    "reviewDate": null,
    "smallImage": false,
    "smallImageId": 0,
    "smallImageURL": "",
    "status": 0,
    "statusByUserId": 10798,
    "statusByUserName": "Test Test",
    "statusDate": 1450435736383,
    "structureId": "",
    "templateId": "",
    "title": "",
    "titleCurrentValue": "proov",
    "type": "general",
    "urlTitle": "proov",
    "userId": 10798,
    "userName": "Test Test",
    "uuid": "6e7ca73e-be8d-4352-8733-e7f626e2020c",
    "version": 1.1  }
```