Tartu Ülikool
Arvutiteaduse Instituut
Informaatika õppekava

Sander Ruusmaa

Kasutajaliides Maa-ameti aadressiotsingu komponendi koodigeneraatorile

Bakalaureusetöö (9 EAP)

Juhendaja: Vambola Leping, MSc

Tartu 2022

Kasutajaliides Maa-ameti aadressiotsingu komponendi koodigeneraatorile

Lühikokkuvõte:

Bakalaureusetöös uuritakse, kirjeldatakse ja uuendatakse Maa-ameti aadressiandmete infosüsteemi (ADS) ja selle ökosüsteemi kuuluvat aadressiotsingu komponenti (In-ADS) koos spetsiaalse koodigeneraatoriga. Mainitud süsteemist ning komponentidest luuakse lühike ülevaade. Lisaks, luuakse ülevaade arendusel kasutatud tehnoloogiatest. Töö põhieesmärk on uuendada olemasolevat aadressiotsingu komponendi koodigeneraatorit, millega on võimalik arendajal mugavalt luua In-ADS komponenti koodi kirjutamata. Töös antakse lühiülevaade olemasolevast koodigeneraatorist ning hiljem ka uuendatud rakendusest. Samuti kirjeldatakse uue kasutajaliidese eeliseid vana ees.

User interface development for the Estonian Land Board's address search component's code generator

Abstract:

This thesis describes the knowledge of a brief research of the Estonian Land Board's Address Data System (ADS) and its ecosystem, in which the address search component (In-ADS) and its specific code generator resides. A brief overview is given about the aforementioned system and its components. Additionally, an overview is given about the used technologies during development. The main goal of this thesis is to update the already existing address search component's code generator, which a developer can use to conveniently create an In-ADS component without writing a single line of code. Furthermore, this thesis gives an overview of the already existing code generator and thereafter of the updated code generator. The advantages of the updated user interface will be described with regards to the old one.

Keywords: React.js, Semantic UI, In-ADS, ADS, SonarQube, SonarLint

CERCS: P175 Informaatika

Sisukord

Sissejuhatus

Maa-amet on tegelenud sellega, et Eesti riigi jaoks loodud aadressiandmete infosüsteemiga liidestamine oleks võimalik erinevatel viisidel. Selleks on Maa-amet loonud mitmeid võimalusi, kuid üks kiiremini rakendatavaid ning majanduslikult mõistlike viise selleks on integreeritava aadressiotsingu (In-ADS) komponendi kasutamine. Aadressiotsingu komponendi kasutamise teeb mugavamaks spetsiaalse koodigeneraatori rakendamine, kus on arendajal võimalik läbi kasutajaliidese genereerida koodi In-ADS komponendi loomiseks. Koodigeneraatori kasutamine teeb eeldatava genereeritud komponendi tulemuse selgemaks ning kiirendab arendaja jaoks komponendi loomist.

Lõputöö eesmärk on uuendada olemasolevat 2014. aastal loodud koodigeneraatorit, viies see üle uuele kaasaegsele React.js kasutajaliidese raamistikule ning uuendada kasutajaliidese disaini, mis vastab Maa-ameti uute rakenduste disainikontseptsioonile. Lisaks toimub paralleelselt koodigeneraatori uuendamisega ka aadressiotsingu komponendi uuendus- ning täiendustööd, mille tulemusena lisatakse koodigeneraatorisse uusi valikuid, mis võimaldavad mugavalt kasutusele võtta uued In-ADS komponendi funktsionaalsused.

Kirjaliku töö struktuur on järgnev. Töö 1. osas luuakse põgus ülevaade Maa-ameti aadressiandmete infosüsteemist ning selle ümber ehitatud ökosüsteemist, mis aitab lugejal mõista loodava koodigeneraatori tähtsust selles ökosüsteemis. Töö 2. osa koosneb olemasoleva koodigeneraatori ülevaatest, mis loodi aastal 2014. Selles osas tuuakse ka välja koodigeneraatori rakenduse puudusi, mis on tekkinud nii möödunud aja kui ka Maa-ameti aadressiotsingu komponendi uuenduste soovide tõttu. Töö 3. osas tutvustatakse praktilises osas kasutatavaid tarkvaraarenduse tööriistu/raamistike ning kirjeldatakse React.js kasutajaliidese raamistikule loodud uut koodigeneraatorit ning arendustöö tulemusi. Lisaks on selles osas välja toodud ka mõned uued funktsionaalsused, mis lisandusid aadressiotsingu komponendi arendustöö käigus.

1. Aadressiandmete infosüsteem ja integreeritav aadressiotsing

Aadressiotsingu komponendi koodigeneraatori loomiseks on vaja aru saada loodava komponendi tööst. Koodigeneraatori töö on genereerida aadressiotsingu (In-ADS) moodul, mis omakorda kasutab Maa-ameti aadressiandmete infosüsteemi. Nende tehnoloogiate selgitamiseks on välja toodud vastavad lühikirjeldused neist.

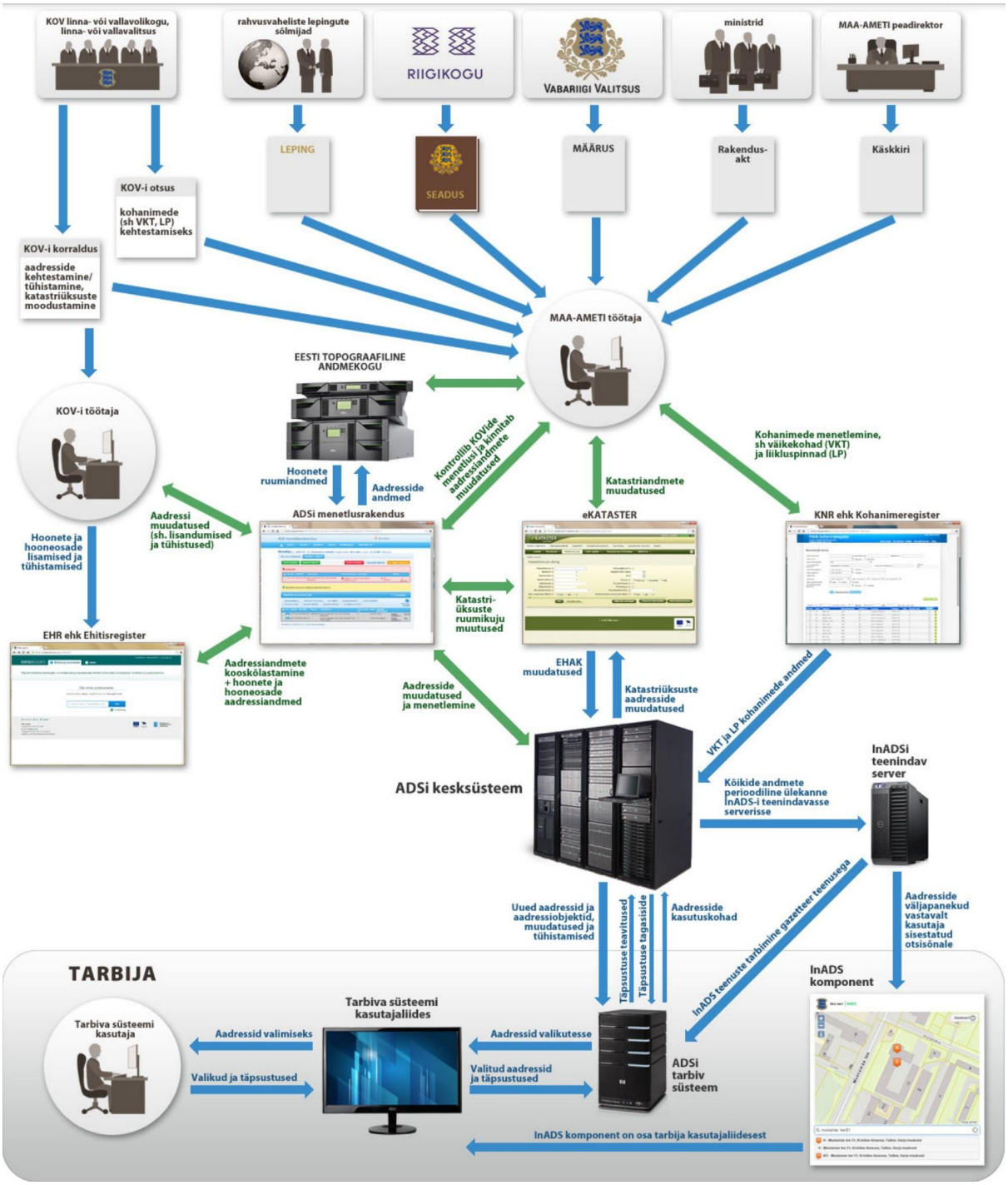
1.1 Aadressiandmete infosüsteem (ADS)

Järgnev informatsioon on pärit Maa-ameti "ADS-ga liidestamise juhend" dokumendist [1], kus kirjeldatakse Aadressiandmete infosüsteemi.

Alates 2007. aastast on Maa-amet tegelenud sellega, et Eestis oleks ühtne standardiseeritud Eesti koha-aadresside ja aadressiandmete muutumise dünaamika. Tänaseks on Aadressiandmete süsteemi infosüsteem (ADS-i infosüsteem ehk veelgi lühemalt ADS) seadusandluses sätestatud ning töötav lahendus. Riigiasutused ning riigi poolt hallatavad infosüsteemid on kohustatud kasutama samu standardeid, nagu ADS-is. Selle eesmärk on muuta aadressi mõiste kõikidele ühtselt mõistetavaks, teha aadressiandmete töötlust analoogseks ning mugavdada ja odavdada aadressiga seotud süsteemide omavahelist liidestamist.

Maa-ameti dokument toob ka välja aadressidega seotud tegevused ADS-is. Kirjeldatakse, kuidas süsteem võimaldab omavalitsustel mugavalt süsteemi lisada või süsteemist eemaldada aadresse ning lisada või eemaldada ühelt aadressilt aadressiobjekte. Ühele aadressile võib vastata mitu aadressiobjekti, mis tähendab, et samal aadressil võib asuda nii elumaja, kõrvalhoone, kuur või mõni muu objekt. Samuti on võimalus muuta aadressiobjekte kehtetuks ning ka taastada kehtivust.

ADS-i ökosüsteemi kirjeldab ka Maa-ameti dokumendis [1] välja toodud joonis, millel on näha ka selle töö autori koodigeneraatori tulemuse ehk In-Ads komponendi ülesannet ökosüsteemis.



Joonis 1. ADS ökosüsteemi kirjeldav jooins (võetud Maa-ameti dokumendist [1])

1.2 Integreeritav aadressiotsing (In-ADS)

Järgnevas alampeatükis põhinev info on pärit Maa-ameti Geoportaalist [2], mis kirjeldab In-ADS komponenti.

Maa-amet on loonud integreeritava aadressiotsingu mooduli, mis kasutab aadressiandmete süsteemi (ADS) andmeid, et teha kõige täpsemaid aadressipäringuid. In-ADS teenusega on

võimalik kätte saada täpsed ning korrektsed Eesti aadressid ja neid vajadusel kasutada oma infosüsteemis. Eelkõige on In-ADS loodud asutustele, kes soovivad oma veebirakenduste kasutajatelt saada sisendit aadressi näol, et seda sisendit vajadusepõhiselt töödelda. Lisaks tekstipõhisele aadressiotsingule on võimalik kasutada kaardipõhist otsingut, kuna In-ADS sisaldab ka kaardikomponenti. Eelnevalt välja toodud kaardikomponendiga on lisavõimalus staatiliselt kuvada asukohta ilma otsingu funktsionaalsuseta. Kirjeldatud teenus on praegu kasutusel mitmes veebikeskkonnas (nt www.eesti.ee ja www.politsei.ee).

ADS-iga liidestumiseks on mitmeid viise, kuid üheks lihtsamaks ja kiiremaks viisiks, nagu ka Maa-ameti "ADS-ga liidetumise juhend" [1] välja toob, on In-ADS komponendi kasutusele võtmine. Mainitud dokumendis kirjeldatakse seda kui kergliidestumist. See tähendab, et ei ole vaja luua uut täisbaasilist liidestust, kus andmeid hoitakse ning uuendatakse lokaalses andmebaasis, et neid hiljem edastada tarbivale kasutajaliidesele. See tähendaks, et aadressiandmete infosüsteemis olevad andmed oleks vaja dubleerida ning sellega kaasnevalt kulutada tohutult andmepinda. Lisaks on vaja dubleeritud andmeid pidevalt ka uuendada. Selle tulemusena on täisbaasiline liidestamine väga kulukas ning ei ole sobilik lahendus lihtsamate projektide jaoks, mis kasutaksid ADS-i. Kergliidestumisel pärib In-ADS komponent hoopis andmeid läbi In-ADS-i teenindava serveri (vt joonis 1), mis teeb sellist sorti liidestumise kergekaaluliseks, kuna kogu aadressiandmete infosüsteemi andmeid ei ole vaja talletada lokaalselt, et neid kätte saada.

Järgnevalt kirjeldatakse seni kasutuses olnud ehk aastal 2014 loodud aadressiotsingu komponenti.

2. Olemasolev koodigeneraator (2014. a)

Aastal 2014 arendati paralleelselt In-ADS komponendiga ka selle komponendi koodigeneraator koos kasutajaliidesega. Koodigeneraatori olemasolu loob eelduse, et kui arendustöödes on vajalik luua liidestus ADS süsteemiga, siis saab ära kasutada valmisloodud In-ADS komponenti, mille loomise keerukust vähendab koodigeneraator. Selle asemel, et lugeda In-ADS arendusjuhendit [3], et paremini mõista komponendi initsialiseerimisel erinevate parameetrite ülesandeid, on arendajatele antud võimalus ära kasutada koodigeneraatori mugavusi. Generaatori kasutajaliides annab selgema ülevaate seadistuste võimalustest, kategoriseerides ära erinevad seadistused ning piiritledes tehtavaid valikuid. Tulemusena saab arendaja olla kindel, et koodi genereerides on tulemuseks töötav In-ADS komponent. Luues käsitsi sama komponenti võivad tekkida igasugused vead ning küsimused komponendi seadistuse parameetrite suhtes.

2.1 Vana koodigeneraatori kasutajaliidese tehnoloogia

“Vana” koodigeneraatori kasutajaliides kasutas klassikalist struktuuri veebiaplikatsiooni arendamisel, kus aplikatsioon koosneb kolmest osast:

- HTML dokument, mis kirjeldab aplikatsiooni struktuuri
- CSS fail, mis kirjeldab HTML elementide stiili (asukoht, värv, font, suurus jne)
- JavaScript fail, mis loob HTML elementide taha mingi funktsionaalsuse, et oleks võimalik mingit tegevust teha või ülesannet täita.

Selleks, et kasutajaliidese arengut teha lihtsamaks, oli võetud kasutusele tolleaegne populaarne JavaScript teek, nimega jQuery.

2.1.1 jQuery

Olemasoleva ehk nüüdseks vana koodigeneraatori kasutajaliidese funktsionaalne osa oli arendatud jQuery JavaScript teegiga. JQuery on JavaScript programmeerimiskeele jaoks 2006. aastal loodud teek, milles leidub mitmeid funktsioone/meetodeid, millega on võimalik manipuleerida DOM-i, HTML dokumendi stiliseerimise faili (CSS), luua JavaScripti animatsioone, saata ja vastu võtta andmeid serverilt jne [4]. Eelnevalt kirjeldatud tegevused on kõik tehtavad ka standardses JavaScript keeles, kuid teegi rakendamine teeb nende tegevuste implementeerimise lihtsamaks. Mõned tegevused/ülesanded nõuavad mitme

koodirea kirjutamist tavalises JavaScriptis, kuid jQuery teegi kasutamisel saab ühe koodireaga välja kutsuda jQuery meetodi, mis just neid tegevusi/ülesandeid täidab.

JQuery populaarsus on endiselt meeletu. Aasta 2022 seisuga on jQuery kasutusel 73% veebilehtedel, mis kuuluvad külastatavuse poolest top 10 miljoni veebilehtede hulka [5]. See tuleneb osaliselt sellest, et lihtsakoeliste veebilehtede loomisel ongi mugav kasutada jQuery teeki, mis teeb JavaScript funktsionaalsuse loomise lihtsamaks. Küll aga suurem osa jätkuvast populaarsusest tuleneb sellest, et need miljonid veebiaplikatsioonid, mis loodi jQuery teegiga aastast 2006, vajasid ja vajavad ka edaspidist tuge, et olla töökorras. Sellest tulenevalt ei saa jQuery populaarsus ka lähiajal drastiliselt langeda.

2.2 Puudused olemasolevas koodigeneraatori kasutajaliideses

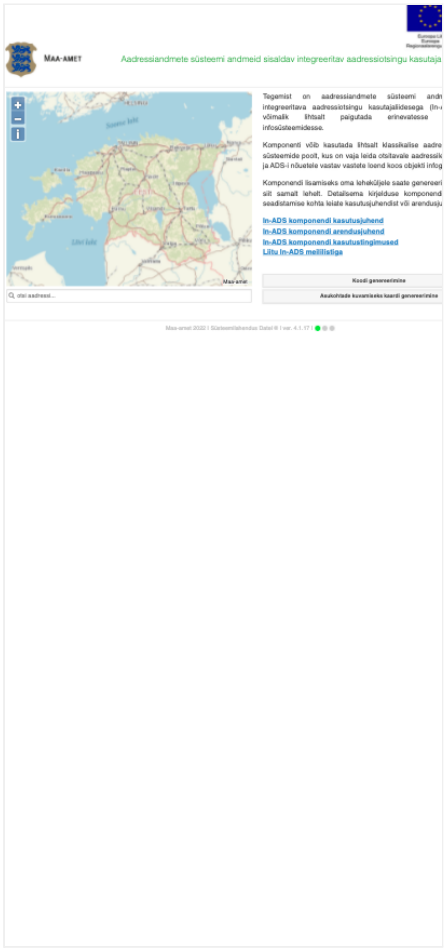
Arvestades algse In-ADS koodigeneraatori loomisaega (2014), siis on arusaadav, et peaaegu 8 aastaga võib aeguda kasutajaliidese disain ning Maa-ametil tekkida soov saada juurde lisafunktsionaalsusi In-ADS komponenti (sh ka koodigeneraatorisse). Maa-amet on vastu võtnud otsuse, millega laseb kõik uuendatavad või uued rakendused üle viia ühtsele disainikontseptsioonile. Selleks paremini mõistmiseks, et milline disain välja näeb, on autor lisanud töö lõppu lisamaterjalide alla lisa 1, milles on kujutatud ühe Maa-ameti uuendatava rakenduse prototüüpi. Mainitud rakendus on geokodeerimise teenust pakkuv veebirakendus, mille arendustööd on 2022. aasta I kvartali seisuga käimas.

2.2.1 Kujundus

Avades 2014. aasta In-ADS koodigeneraatori veebirakenduse näeb kasutaja esimese asjana rakendust tutvustavat esilehte. Siinkohal saab võrrelda üldist disaini vana koodigeneraatori ja uuendatava geokodeerimise rakenduse vahel. On selge, et valgel taustal, peaaegu must-valge disain ei lähe kokku uue disainikontseptsiooniga.

Välja on toodud mõned vaated 2014. aasta koodigeneraatori rakendusest:

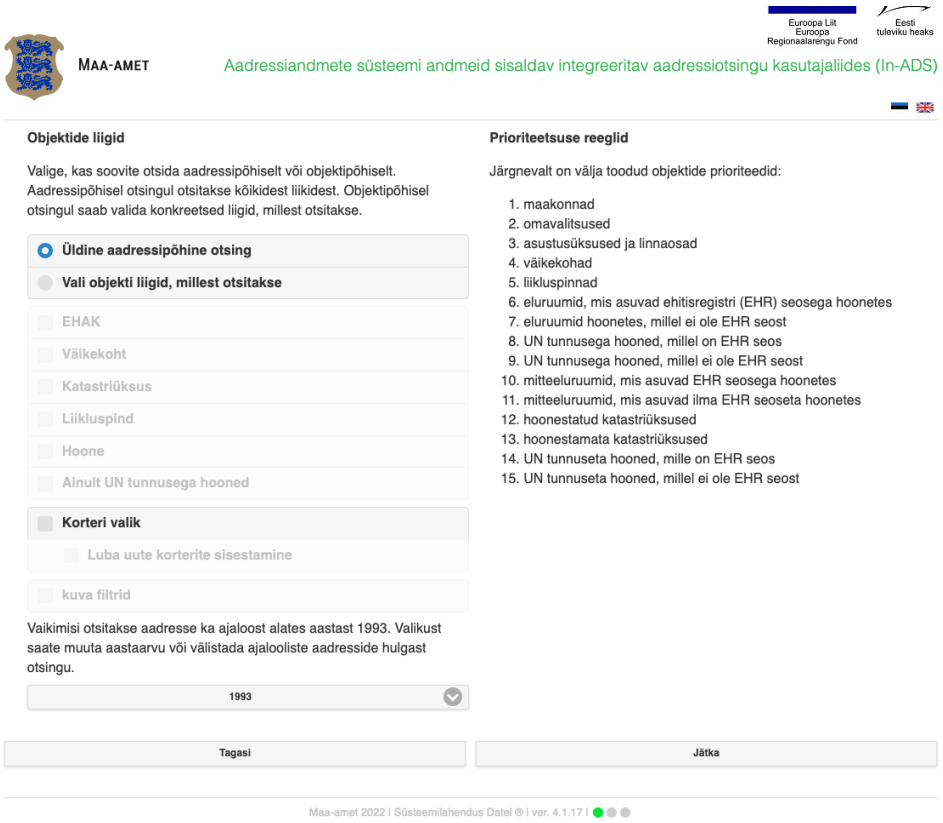
Joonis 2. Esilehe vaade arvutis



Joonis 3. Esilehe vaade mobiilis

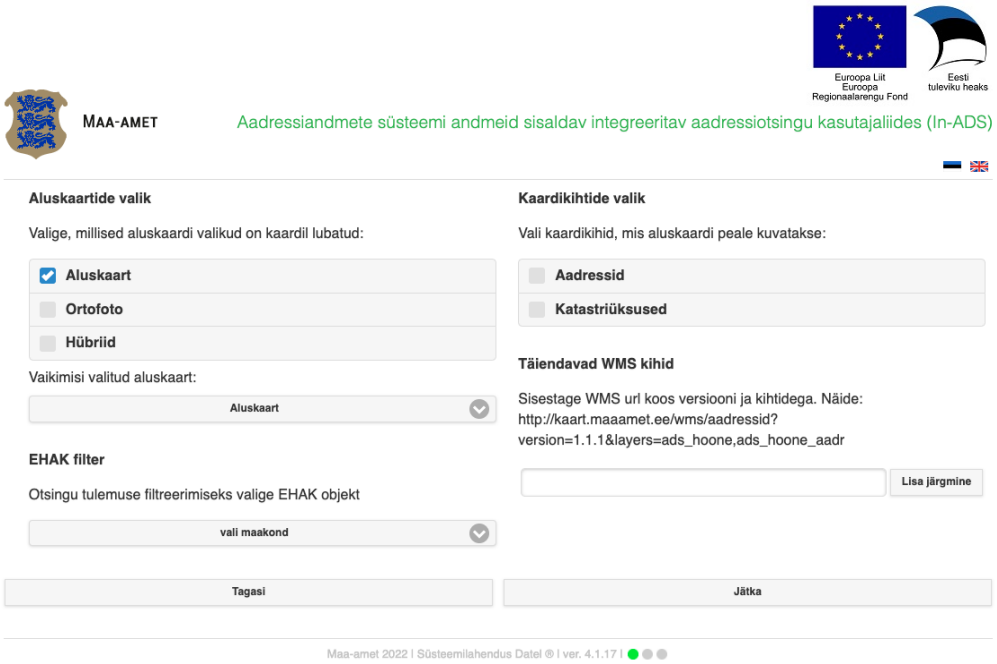
Tähtis on teada, et koodigeneraatori kasutamine ei ole mõeldud toimuma mobiilseadmes, kuna arendustöid tehakse reeglina arvutis. Sellegipoolest leidis Maa-amet, et uuendatud koodigeneraatori esileht peab olema mobiilsõbralik ning kasutajaliides võiks takistada mobiilseadmes edasi liikumist teistesse vaadetesse. Kuna vana koodigeneraatori kasutajaliides ei pidanud olema mobiilsõbralik, siis järgnevad kuvatõmmised on tehtud vaid arvutis, kus kasutaja läbib In-ADS komponendi jaoks koodi genereerimise teekonna vaikeväärtustega (st kasutaja ei ole teinud ühtegi muudatust märkeruutudes ja muudes valikutes).

Joonis 4. Komponendi valiku vaade

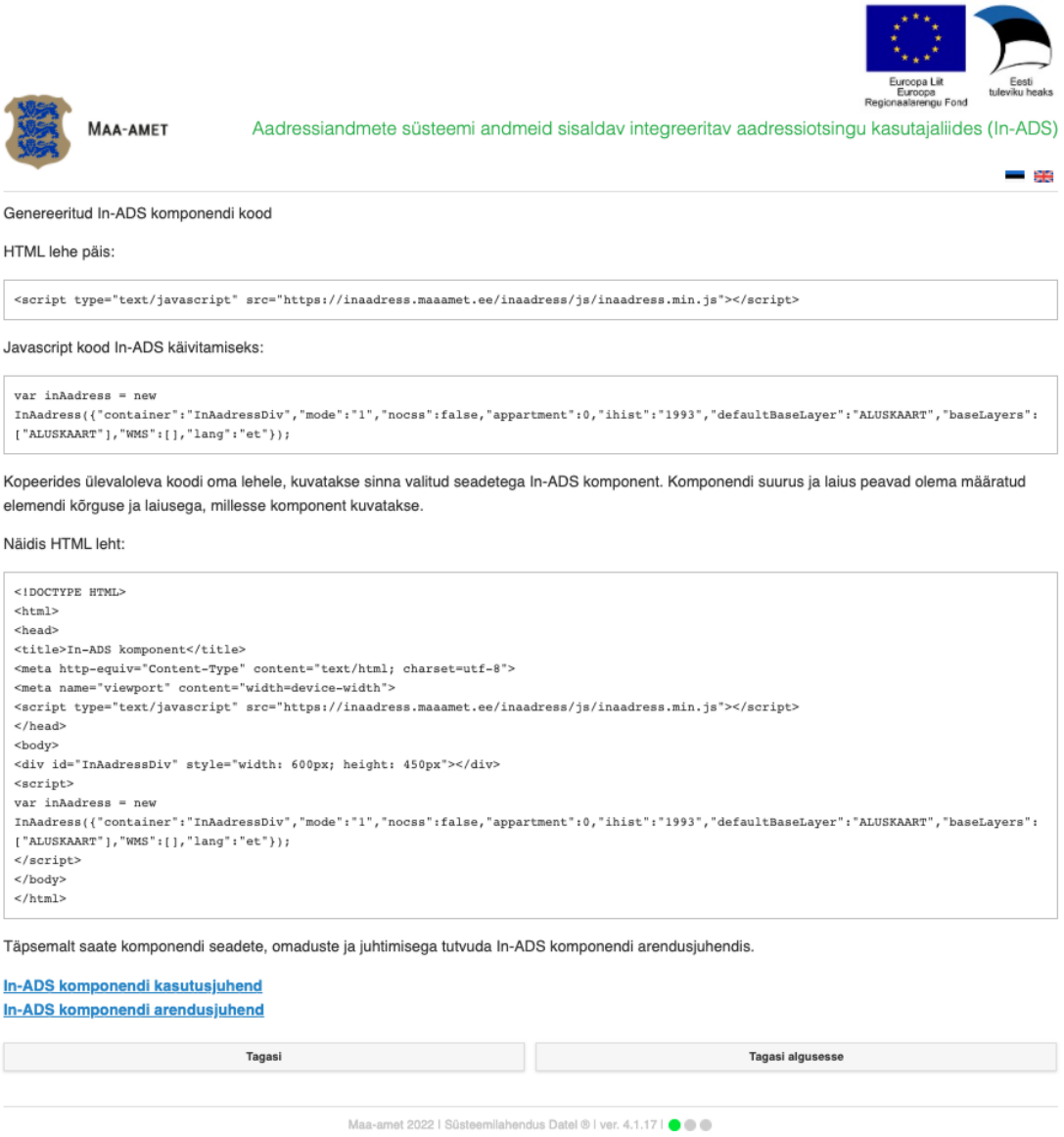


Joonis 5. Otsingutüübi seadistuse vaade

5. joonisel on näha, kuidas Euroopa Liidu Regionaalarengu Fondiga seotud logod jäävad osaliselt peitu isegi, kui arvutis kuvada veebirakendust.



Joonis 6. Kaardi kohandamise vaade



Joonis 7. Genereeritud koodi vaade

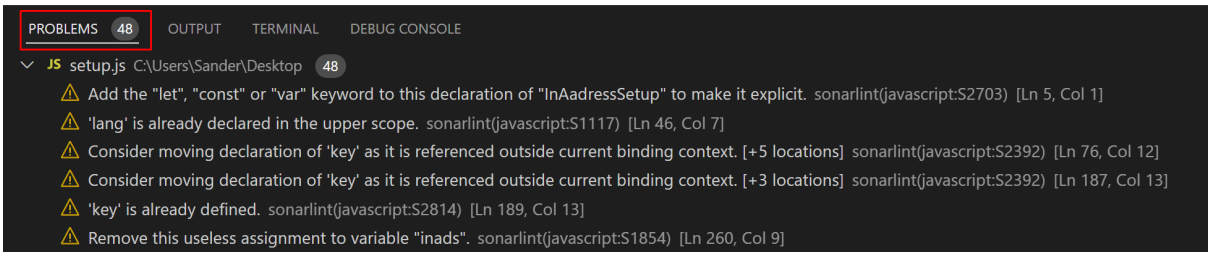
Jõudes viimasesse vaatesse saab kasutaja kopeerida koodi tekstikastidest ning kasutada genereeritud koodi oma enda veebirakenduses. Koodi kopeerimiseks peab kasutaja käsitsi kopeeritava teksti esile tõstma ning andma käsu “kopeeri”. Selline tegevus võib kasutajate seas tekitada probleeme ning vigu kopeerimisel. Võib juhtuda, et mõni tähemärk jääb algusest või lõpust esile tõstmata, mis võib tekitada süntaktilisi vigu programmeerimiskeskkondades. Lisaks puuduvad kopeeritavas HTML lehe näidiskoodis ka taanded, mis teeb koodi kleepimisel loetavuse keerulisemaks.

2.2.2 Lähtekood

2014. aasta koodigeneraatori projekt on hoiustatud suletud koodihoidlas, millele ligipääs on piiratud. Küll aga on töö autorile antud vastavad õigused koodigeneraatori lähtekoodi uurimiseks. Vana koodigeneraatori kogu loogika ning funktsionaalsus peitus ühes *setup.js* nimelises failis, mille kogupikkus on 1338 koodirida. Töö autor rakendas *setup.js* failile SonarLinti staatilise programmi analüüsi, et uurida potentsiaalseid probleeme koodiloetavuses ning loogikas.

2.2.3 Lähtekoodi staatiline programmi analüüs SonarLintiga

Kogu peamist funktsionaalsust sisaldav fail *setup.js* laeti alla koodihoidlast ning avati Visual Studio Code IDE programmiga. Lisaks oli alla laetud SonarLint pistikprogramm programmeerimiskeskkonda, mis hakkab igat avatud faili analüüsima, et tuvastada potentsiaalseid veakohti koodis. Avades *setup.js* faili tuvastas SonarLint pistikprogramm 48 probleemi.



Joonis 8. SonarLint pistikprogrammi probleemituvastuse vaade.

Üks sagedasemaid probleeme, mida SonarLint tuvastas, oli ebavajalik indekseeritud for-tsükli kasutamine, mida tuvastati 9 korral. Selle asemel on soovitatud kasutada for-of tsükli süntaksit, mis samuti läbib itereeritavate andmetüüpide (nt järjend, hulk, massiv) väärtused, kuid loetavamalt. Töö autor toob välja ka SonarLinti poolt selgitava näidislahenduse probleemile.

Noncompliant Code Example

```
const arr = [4, 3, 2, 1];

for (let i = 0; i < arr.length; i++) {  // Noncompliant
  console.log(arr[i]);
}
```

Compliant Solution

```
const arr = [4, 3, 2, 1];

for (let value of arr) {
  console.log(value);
}
```

Joonis 9. SonarLint probleemi kirjelduses paiknev näidislahendus probleemile.

Tähtis on mõista, et SonarLinti poolt välja toodud probleemsed kohad mõjutavad tihtipeale koodi loetavust ning arusaamist just inimese vaatenurgast. See tähendab, et koodi töökindlus ega funktsionaalsus ei ole otseselt mõjutatud kuvatud probleemidest, mida SonarLint välja toob.

Kõige sagedasemini esile kerkinud probleem oli lähemalt uurides hoopis valepositiivne. SonarLint tuvastas ühe pideva objekti instantsi loomist, mis on justkui “mõttetu”.

```
396             if (mapMode == 1) {
397                 sp("#inaadressSetup").html("");
398                 new InAadress({
399                     hideLogo: 1,
400                     container: "inaadressSetup",
401                     mode: 1,
402                     appartment: 1,
403                     searchLayers: ["EHAK", "TANAV", "VAIKEKOHT", "EHITISHOONE", "KATASTRIYKSUS"],
404                     lang: currentLang
405                 });
406             }
407             else if (mapMode == 2) {
408                 sp("#inaadressSetup").html("");
409                 new InAadress({
410                     hideLogo: 1,
411                     container: "inaadressSetup",
412                     mode: 2,
413                     appartment: 1,
414                     searchLayers: ["EHAK", "TANAV", "VAIKEKOHT", "EHITISHOONE", "KATASTRIYKSUS"],
415                     lang: currentLang
416                 });
417             }
418             else if (mapMode == 3) {
419                 sp("#inaadressSetup").html("");
420                 new InAadress({
421                     hideLogo: 1,
422                     container: "inaadressSetup",
423                     mode: 3,
424                     appartment: 1,
425                     lang: currentLang
426                 });
427             }
```

Joonis 10. Kuvatõmmis setup.js koodist, kus luuakse uut InAadress objekti.

Omades teadmisi sellest, kuidas In-ADS komponent töötab veebibrauseris, siis on võimalik aru saada, et tegu on valeposiitivse probleemituvastusega. Joonisel 10 on tegemist täiesti korrektse In-ADS komponendi kasutamisega, kus veebibrauseri aknas uuendatakse In-ADS komponendi seadistusi, et illustreerida tehtavate valikute tulemust koodi genereerimisel.

Ülejäänud probleemsetest kohtades võib välja tuua mõned olulisemad neist, mis mõjutavad koodi puhtust ning loetavust:

- välja kommenteeritud koodi esinemine failis (6 korral)
- kasutamata funktsiooni parameetrite deklareerimine JavaScript funktsioonis (4 korral)
- tühi ning kasutamata funktsioon failis (1 korral)
- muutuja uuesti deklareerimine, kui ta juba eksisteerib kuskil ülemises/välises skoopis (6 korral)

Selliseid probleeme koodis aitavad vältida tänapäeval staatilise programmi analüüsi rakendused, mis juhivad arendaja tähelepanu nendele probleemidele ning annab võimaluse neid parandada. Tasub arvesse võtta, et 2014. aastal ei olnud staatilise programmi analüüsi pistikprogrammid laialdaselt levinud, seega keskmine arendaja sai jääda lootma vaid enda teravale silmale, mis tuvastab neid loogika ja loetavuse probleeme koodis. Veelgi raskemaks teeb inimese poolt probleemide tuvastuse see, kui koodiridade arv failis ületab teatud piiri, kus arendaja enam ei osuta tähelepanu valmis kirjutatud ning töötavatele koodiosadele. Pidevalt sadade koodiridade vahel kerimine vähendab üle keritud koodile tähelepanu andmist. Selle tulemuseks võib valmida arendustöö, mis on täiesti töökorras, töökindel ning täidab ka oma eesmärki, kuid samas mitte veatu. Siiski on leitud, et ei ole võimalik täie kindlusega öelda, et suurem koodiridade arv failis, mis tõstab faili koodi keerukust, mõjutaks arendaja koodist arusaamist negatiivselt [6].

3. Uus koodigeneraator (2022. a)

Seoses In-ADS komponendi uuenemisega, mis on Maa-ameti poolt tellitud töö, on vaja uuendada vastavalt ka In-ADS komponendi koodigeneraatorit. In-ADS komponendile lisanduvad uued funktsionaalsused, mida on võimalik sisse-välja lülitada komponendi loomisel. Need uued valikud on vaja peegeldada koodigeneraatorisse, et generaatori kasutajad oleksid teadlikud uuendustest. Uuendada on vaja lisaks uute funktsionaalsuste toetamisele ka kasutajaliidese kujundust, mis sobiks Maa-ameti uue disainikontseptsiooniga.

Uus koodigeneraator koos kasutajaliidesega peab olema loodud React.js raamistikul, nagu ka teised uued Maa-ameti rakendused. Ühtsele kasutajaliidese raamistikule loodud rakendused teeb nende haldamise ning edasiarenduse mugavamaks. See tähendab, et esmalt on vaja vana koodigeneraator üle viia uuele raamistikule uue disainiga, säilitades algupärase funktsionaalsuse. Seejärel tuleb oodata ära In-ADS komponendi uuendused, mille järel saab teada, kuidas uusi funktsioone aktiveerida aadressiotsingu komponendil. Olles teada saanud nende parameetrite nimed ning oodatavad väärtused, mis aktiveerivad uusi funktsioone, on võimalik need lisada koodigeneraatorisse, et teised arendajad saaksid uusi funktsionaalsusi mugavalt kasutusele võtta oma In-ADS komponendis läbi koodi genereerimise kasutajaliidese.

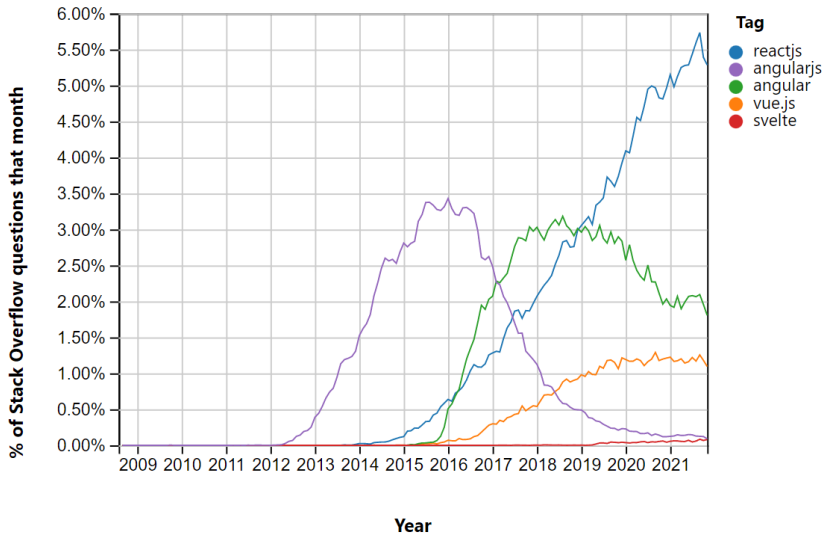
3.1 Uue koodigeneraatori kasutajaliidese tehnoloogia

Kasutajaliidese loomiseks on mitmeid tehnoloogiad, kuid selleks tööks kasutatakse React.js raamistikku koostöös Semantic UI raamistikuga. Lisaks rakendatakse projektile koodi kvaliteeti kontrollivaid tehnoloogiaid, et projekti lähtekood oleks lihtsamini hallatav. Mainitud tehnoloogiatest antakse lühiülevaade.

3.1.1 React.js

Kasutajaliideste arendamiseks on olemas mitu JavaScript raamistikku - Vue.js[1], Angular[2], Ember.js[3] jne. Mitmed küsitlused näitavad seda, et viimase kahe aasta populaarseim JavaScript raamistikuks on arendajate seas olnud aga React.js[4] [7]. Mainitud küsitlustes tuleb ka välja, et React.js populaarsus on endiselt kasvutrendis.

[1] https://vuejs.org/
[2] https://angular.io/
[3] https://emberjs.com/
[4] https://reactjs.org/



Graafik 1: Stack Overflow trendid vastavalt tehnoloogia märgistele [8]

React.js, mis on loodud Facebooki poolt, on tehtud töövahendiks, millega saab arendada kasutajaliideseid [9]. Komponendipõhine arendus teeb töö kiiremaks ning lõpptulemuse efektiivsemaks [10]. Raamistiku loojad kirjeldavad ka seda, et iga komponent omab enda loogikat, olekut ning kuvatavat infot ja on taaskasutatav. Tuues mõne illustreeriva näite võib veebilehel olla komponentideks päis, jalus, seadete vaade, otsingulahter, kontaktinformatsiooni vaade jne.

Sarnaselt teistele JavaScript raamistikele on React.js avatud lähtekoodiga [11]. Iga arendaja saab panustada React.js arengusse tarkvara vigade tuvastamisega ning sellest teada andmisega või React GitHubi koodihoidlasse tõmbetaotluse (ingl *pull request*) saatmisega koos asjalike koodimuudatuste ettepanekutega.

Üks märkimisväärsem omadus React raamistikul on virtuaalne dokumendi objektimudel. Empiirilises uurimuses [12] on kirjeldatud dokumendi objektimudelit (DOM) kui dünaamilist puu andmestruktuuri, mis kirjeldab veebiaplikatsioonis esinevaid elemente, nende kujundust ning nende omavahelisi suhteid. Uurimus toob välja ka selle, et DOM-iga programeerimine on arendaja jaoks mugav vahend veebiaplikatsioonis muudatuste kuvamiseks, kuna see ei vaja, et aplikatsiooni uuesti alla laetakse. Seevastu leiti uurimuses, et DOM-is tehtud muudatused tekitavad tihtipeale vigu, mille põhjuste ülesotsimine osutub sageli raskeks ning ajakulukaks. Lisaks, ei ole vigade mõju aplikatsioonile alati selge. Uuring [12] leidis ka, et

pärast 8 tuntuma aplikatsiooni JavaScripti tarkvaravigade analüüsimist tuleb välja, et 65% nendest vigadest tulenevad DOM-i kasutamisest.

S. Aggarwal on oma töös [13] kirjeldanud DOM-i olemust ja tööd ning järgnev lõik, mis kirjeldab DOM-i, põhineb Aggarrwali tööl. DOM-i otsese manipuleerimise vältimiseks saab võtta kasutusele React JavaScript raamistiku, mis abstraheerib DOM-i ära ning võtab kasutusele virutaalse DOM-i. Pärast elementide muutmist virtuaalses DOM-is rakendatakse algoritmi, mis võrdleb virtuaalset DOM-i ning brauseri DOM-i ja viib muudatused brauseri DOM-i sisse. Tänu virtuaalse DOM-i kasutuselevõtule toimuvad andmestruktuuri muutused mälus, mitte brauseris, ning see teeb muudatuste rakendumise efektiivsemaks. Sellise kindla raamistiku rakendamine vähendab inimfaktorist tulenevaid arendusvigu DOM-i kasutamisel, sest kirjeldatud raamistikus toimetavad DOM-iga kindlad välja töötatud algoritmid.

3.1.2 Semantic UI

Veebilehti kujundatakse tavapäraselt CSS (*Cascading Style Sheets*) failiga, millega on võimalik valida veebilehel olev HTML element ning seda kujundada, andes sellele näiteks värv, pöörata pahupidi ning suurendada 1.5 korda [14]. CSS failiga kujundamine on suur käsitöö ja alatihti võivad need failid koosneda sadadest kui mitte tuhandetest koodiridadest.

Semantic UI on kasutajaliidese raamistik, mis teeb veebilehe disainimise mugavamaks [15]. Mainitud raamistikul on üle 50 kasutajaliidese elemendi ning kõik elemendid on võimalik hõlpsasti muuta ekraanisuuruse muutumisele reageerivaks [15]. Selgitades eelnevat väidet on võimalik nii lauaarvuti, kui ka nutitelefoni ekraanil kuvada veebilehte ilma informatsiooni kaduta või mõne muu visuaalse veata. Semantic UI on teinud koostööd paljude populaarsete JavaScript raamistikega (sh. React.js) ning võimaldab rakendada Semantic UI kasutajaliidese raamistikku koostöös JavaScripti raamistiku veebiaplikatsiooni loogikaga [15].

Arendajal on võimalik mugavalt kasutada valmisloodud kasutajaliidese elemente, nagu näiteks nupp, sisendkast, nimekiri, tabel, hüpikaken, märkeruut ning lisada neile vajadusel kohandatud funktsionaalsust [16]. Suur eelis Semantic UI kasutamisel on selle raamistiku loojate poolt tehtud dokumentatsioon, mis sisaldab endas iga valmisloodud elemendi kohta kirjeldust, mitmeid näiteid ning sisendandmeteks antavate parameetrite nimekirja koos täpsete kirjelduste ja selgitustega.

Selle töö autori praktilises osas kasutati kasutajaliidese arendusel Semantic UI raamistikus olevaid elemente, et mugavdada ja kiirendada arendust ning luua töökindlam kasutajaliides.

3.1.3 Staatilise programmi analüüs

Tarkvara testimine on tähtis protsess tarkvaraarenduses, mis likvideerib erinevaid vigu loodud töös. Testimine on reeglina käsitöö, mis nõuab käsitsi testide kirjutamist, kuid selleks, et juba eos vältida tarkvara loomises vigu, on võimalus rakendada tarkvara projektile staatilise programmi analüüsi, mis suudab automaatselt tuvastada vigu kirjutatud koodis ning määrata ka vigade tõsisust [17].

Uuringus [18] analüüsiti kolme suurt tarkvara projekti, mis koosnesid kokku ligikaudu 1000000 C++ programmeerimiskeele koodi reast. Uuring näitas, et tänu staatilise programmi analüüsile oleks olnud võimalik vähendada arenduskulusid ühe projekti korral 17% ning teise projekti korral 23%. Analüsaator oleks genereerinud turvaraporti, kus on esile toodud potentsiaalsed veakohad koodis ning nendega oleks saadud kohe tegeleda vältides neid vigu testimise faasis, kus arendaja peab tagasi vaatama oma tehtud tööle ning teostama seal parandusi, mis tõstab tootmiskulusid.

Tarkvaraarendaja poolt kirjutatud koodi kvaliteedi valideerimiseks on nüüdseks loodud mitmeid tööriistu, mis aitavad tõsta koodi töökindlust, jätkusuutlikkust ning loetavust läbi staatilise programmi analüüsi. Selles töös kasutati selleks SonarLinti ja SonarQube'i.

3.1.4 SonarLint

Üheks tööriistaks, mida arendaja saab kasutada staatilise programmi analüüsiks, on SonarLint, mis analüüsib arendaja integreeritud programmeerimiskeskkonnas (IDE) kirjutatud koodi [19]. SonarLint tootja SonarSource toob välja suure eelise selle kasutamisel - koodianalüüs ning raportite genereerimine toimub reaalajas, mis teeb koodi kirjutamise efektiivsemaks ning filingraansemaks [20].

Töö autor kasutas praktilises osas tarkvaraprojekti loomisel samuti SonarLint tehnoloogiat oma programmeerimiskeskkonnas (IDE) Visual Studio Code, et analüüsida JavaScriptis kirjutatud koodi kvaliteeti ning vähendada arenduses tekkivaid tüüpvigu, mis mõjutavad tarkvara projekti töökindlust.

3.1.5 SonarQube

Tööriist, mida kasutatakse tarkvara ettevõtetes staatilise programmi analüüsiks, on SonarQube, mis analüüsib arendaja kirjutatud koodi serveris, kus loodud tarkvara koodi hoiustatakse [21]. Serveris toimuv analüüs võimaldab näha analüüsist tulenevat raportit mitmel inimesel, kes võivad olla tarkvarapojektiga seotud. SonarLint töötab lokaalselt arendaja programmeerimiskeskkonnas, mis avaldab raporteid ainult arendajale. Soovitatav on võtta kasutusele SonarQube, kui koodianalüüsi tulemit soovib näha rohkem kui üks inimene.

Töö autori tarkvaraprojekti analüüsiti ka serveris SonarQube'i poolt, mis võimaldas töökaaslastel ning projekti vastutavatel isikutel kontrollida kirjutatud koodi kvaliteeti.

3.2 Uuendatud kujundus

Uue koodigeneraatori kasutajaliidese disaini jaoks eraldi dokumenti ei loodud, mis kirjeldaks just selle rakenduse kujundust. Antud olid juhised, mis käskisid võtta tugevat inspiratsiooni teistest Maa-ameti arendusprojektide kujundusest, nagu Aadressiandmete infosüsteemi avalik teenus[5], Aadressiandmete geokodeerimise teenus[6] ning Maa-ameti geoportaali rakenduste disainikontseptsioon (vt lisa 2). Analüüsides mainitud projekte ning dokumente oli võimalik kindlaks määrata peamised tunnusjooned uue koodigeneraatori disainis.

3.2.1 Värvid, font ja muud kasutajaliidese elemendid

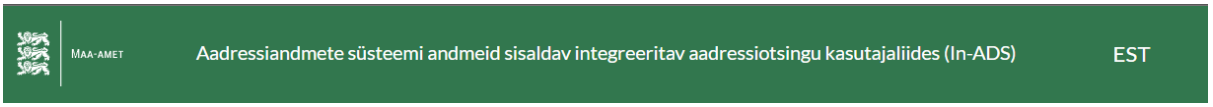
Põhilised värvitoonid ja font ehk kirjatüüp sai võetud Geoportaali disainikontseptsiooni dokumendist (vt lisa 2). Veel lisandus värvipaletti värv, mida kasutatakse nii Aadressiandmete infosüsteemi avalikus teenuses kui ka Aadressiandmete geokodeerimise teenuse rakenduses taustavärvina (hekskoodiga #fcf9ee).

Muude interaktiivsete elementide kujundus on samuti jäljendatud eelnevalt mainitud kahe teenuse kasutajaliideste elementide pealt. Selliste elementide hulka kuuluvad märkeruut, raadio nupp, nupp, sisend, rippmenüü jms.

Kasutajaliidese päis ja jalus on uutes Maa-ameti rakendustes läbivalt sama rohelise (hekskoodiga #32774e) taustaga ning valget värvi tekstiga. Päise vasakusse serva jääb

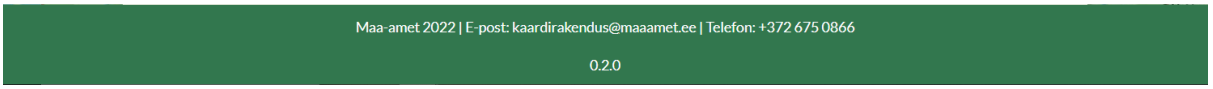
[5] https://xgis.maaamet.ee/adsavalik/
[6] https://xd.adobe.com/view/67632147-f703-4094-a9c7-4de1e2200c6c-3230/grid (prototüüp)

Maa-ameti logo ning paremasse serva kasutajaliidese keele valiku rippmenüü. Päise keskel paikneb rakenduse nimetus.



Joonis 11. Koodigeneraatori kasutajaliidese päis.

Jaluse sisu võib kohati erineda erinevate rakenduste vahel, kuid üldine disain on samasugune - roheline riba, millel on peal valge tekst. Tähtis on see, et jaluses oleks Maa-ameti kontaktinfo probleemide või info pärimise korral ühenduse võtmiseks.



Joonis 12. Koodigeneraatori kasutajaliidese jalus.

Kasutajaliidese keha, kus toimub peamine interaktsioon kasutaja ning kasutajaliidese vahel, on lai beežika taustavärviga riba (hekskoodiga #fcf9ee), kus paikneb kasutajaliidese sisuline osa. Juhul, kui brauseriakna laius ületab teatud suurust, tekib laia beežika riba taha taustaks vaade Eesti kaardist, mis on kaetud kergelt roheka filmiga. Sarnane kujundus on kasutusel ka tulevasel geokodeerimise teenuse rakendusel (vt lisa 1).

Autor on lisanud kirjaliku osa lõppu vaated uue koodigeneraatori koodi genereerimise teekonna vaikeväärtustega (vt lisa 3). Teekond, mida läbitakse on analoogne eelnevalt 2. peatükis läbitule ning loob lugejale võrdlusmomendi uue ja vana koodigeneraatori kujunduse vahel.

3.2.2 Mobiilseadme tugi

Vanas koodigeneraatoris (2014) puudus mobiilseadme tugi. Põhjus selle puuduseks oli selge - arendustööd toimuvad arvutites, seega lisatöö tegemine selleks, et mobiilsed seadmed saaksid mugavalt koodigeneraatorit kasutada, oli põhjendamata. Uus koodigeneraatori kasutajaliides kasutab React.js spetsiifilist Semantic UI raamistikku, millega saab mugavalt struktureerida HTML dokumenti ning sinna luua funktsionaalseid elemente koos elementaarse kujundusega. Kasutajaliidese beežikale laiale ribale sai struktureeritud kogu kasutajaliidese sisu Semantic UI Grid elemendiga[7] ning sellele alluvate alamelementitega. Hea omadus Semantic UI

[7] https://react.semantic-ui.com/collections/grid/

elementide juures on see, et nad oskavad säilitada enda ilu ja kuju ka siis, kui brauseriakna laius või kõrgus muutub. Teisiti öeldes omavad Semantic UI elemendid dünaamilise veebidisaini omadusi. See võimaldas Grid elementidele ühe parameetri lisamisega muuta kogu kasutajaliidese mobiilisõbralikuks.

Töö praktilises osas oli nõutud, et ainult koodigeneraatori avaleht oleks mobiilisõbralik, sest seal on võimalik kasutajal In-ADS komponenti kasutada ning aru saada tema tööst mingil määral. Sellegipoolest otsustas töö autor muuta terve rakenduse ulatuses kasutajaliides mobiilisõbralikuks, kuna selleks vajaminev töö ja vaev oli väike, tänu korralikule kasutajaliidese struktureerimisele, kasutades Semantic UI raamistikku.

Töö kirjaliku osa lõpus lisade all on kuvatõmmised (lisa 4) sellest, kuidas mobiilses seadmes (iPhone 11) läbitakse In-ADS komponendi koodi genereerimise teekond kasutajaliideses vaikeväärtustega.

3.3 Uued funktsionaalsused In-ADS komponendis

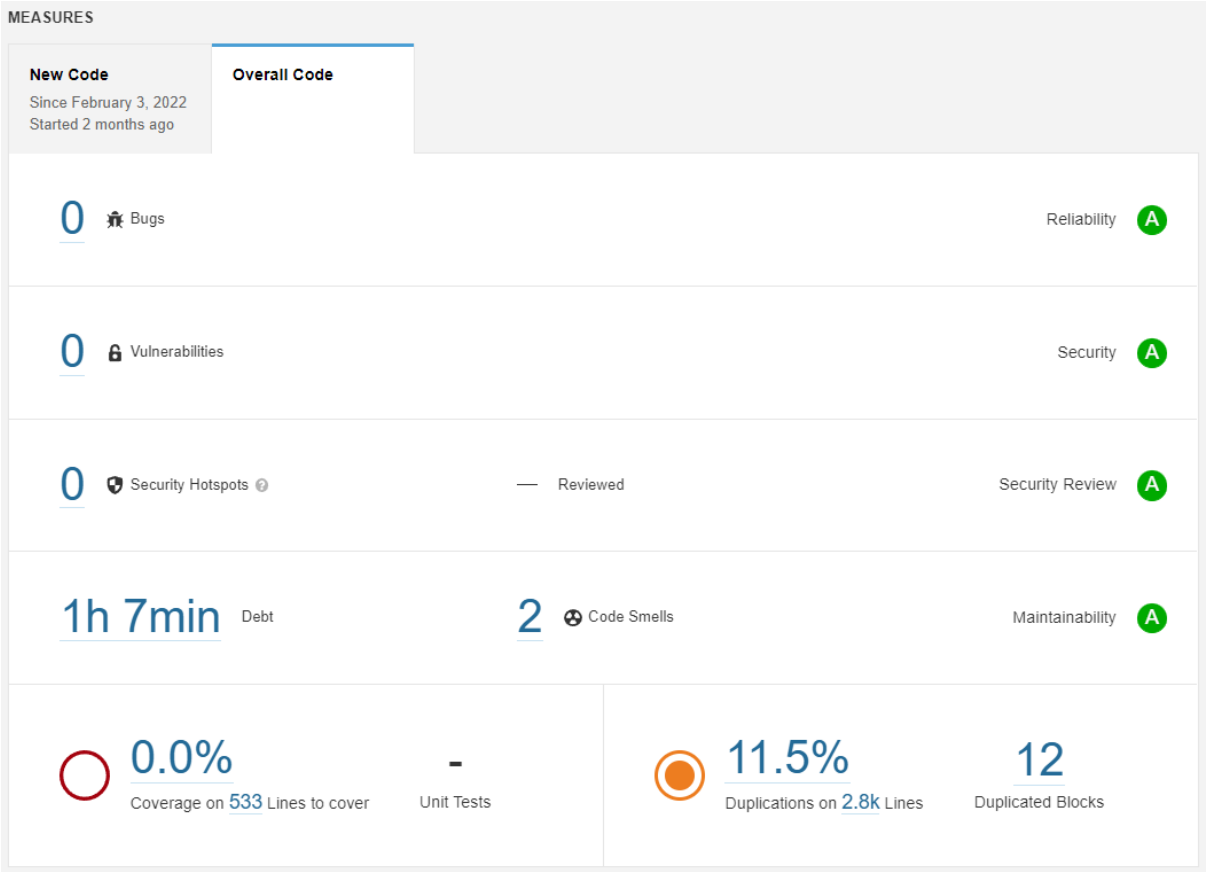
In-ADS komponendi uuenemise järel lisandusid mõned uued funktsionaalsused ning nende sisse lülitamise võimalus peab olemas olema ka koodigeneraatoris. Mõned uued funktsionaalsused In-ADS komponendis on järgnevad, mida oli vaja ka esindada koodigeneraatoris:

- võimalus keelata/lubada klaviatuuri tugi otsingu komponendis
- võimalus otsida asumeid, kvartaleid, väikesaari ja osavaldu
- võimalus seadistada, kas peale otsingutulemuse väljavalimist otsingutulemuste nimekiri suletakse või mitte
- võimalus kuvada keelevalik või mitte
- võimalus aktiveerida otsingulahter programselt komponendi kuvamisel või mitte, et kasutaja ei peaks hiirega käsitsi otsingulahtrit aktiveerima

3.4 Staatilise programmi analüüsi rakendamine uuele koodigeneraatorile

Kogu arendustöö vältel kasutati SonarLint staatilise programmi analüüsi pistikprogrammi Visual Studio Code programmeerimise keskkonnas. See võimaldas saada reaalajas tagasisidet koodis tuvastatud probleemsete kohtade korral. Enamik probleemseid kohti, mis SonarLint rakendus oli välja toonud, said ka lahendused, säilitades või tõstes kirjutatud koodi kvaliteeti.

Detailsem programmi analüüs toimus arenduskeskkonna serverisse üles seadistatud SonarQube tööriistaga. Arenduse lõppjärgus on võimalik välja tuua analüüsiraport, kus on ülevaade rakenduse probleemsetest kohtadest. Lisaks annab SonarQube hinnangud koodi töökindluse, turvalisuse ning edasise arenduse lihtsuse kohta.



Joonis 14. Uue koodigeneraatori projekti SonarQube analüüsi raport 17.04.2022.

Kaks tuvastatud probleemset kohta (*code smells*) on üle vaadatud autori poolt ning leitud, et koodi ümberkirjutamine niiviisi, et staatilise programmi analüsaator neid ei tuvastaks, vähendaks koodi loetavust ning raskendaks arusaadavust. Samuti on ka koodikorduste juhtumid (*duplications*) üle vaadatud töö autori poolt ning leitud, et arvestades rakenduse suurust, siis tuvastatud koodi kordused ei vähenda koodi loetavust.

3.5 Võimalikud edasiarendused ja eeldused selleks

Juhul kui Maa-amet soovib In-ADS komponenti tulevikus edasi arendada ning luua veelgi uusi funktsionaalsusi, siis on vajalik täiendada ka selle komponendi koodigeneraatorit, et loodud uuendusi oleks arendajatel võimalik mugavalt kasutusele võtta. Uus koodigeneraatori rakendus on loodud selliselt, et uute tõeväärtuse tüübiga parameetrite juurde lisamine koodi

genereerimisel, mis aktiveerivad uusi funktsioone komponendis, oleks võimalikult lihtne. See tähendab, et arendaja, kellele määratakse koodigeneraatori täiendamine, saab kiiresti ja vähese vaevaga aktiveerida lihtsamad uued funktsionaalsused komponendis. Seda enam teeb tööd lihtsamaks see, et uus kasutajaliides on loodud React.js raamistikule, mis ühtlustab kasutajaliideste loogikat ning teeb erinevate projektide vahel vahetamise arendaja jaoks lihtsamaks, kui ta on React.js raamistikuga varem tööd teinud. Samuti on uus kasutajaliides läbinud SonarLint ja SonarQube staatilse programmi analüüsid ning saanud head hinnangud pärast analüüsimist, mis võiks anda eelduse, et kood, millega arendaja kokku puutub koodigeneraatori kasutajaliidese jätkuarendusel on piisavalt loetav ja kiiremini arusaadav.

Kokkuvõte

Käesoleva töö raames loodi Maa-ametile React.js kasutajaliidese raamistikul koodigeneraator, mis lihtsustab Maa-ameti aadressiotsingu komponendi (In-ADS) kasutamist arendustes. Uus koodigeneraator vastab uuele ühtsele Maa-ameti kasutajaliideste kujundusele. Paralleelselt koodigeneraatori kasutajaliidese loomisega uuendati ka aadressiotsingu komponenti, mille käigus lisandusid uued funktsionaalsused. Läbi uuendatud koodigeneraatori on võimalik neid funktsioone mugavalt kasutusele võtta, kuna lisandunud funktsionaalsused on esindatud ka uuenenud koodigeneraatoris.

Lisaks praktilisele tööle oli vajalik uurida Maa-ameti aadressiandmete infosüsteemi (ADS), et paremini mõista loodava tarkvaraprojekti olemust. Samuti oli tarvis uurida praktilises osas kasutatavaid tehnoloogiaid, mille tulemusena tekkisid paremad eeldused sujuvamaks töövooks ning kvaliteetsemale töötulemusele.

Viidatud kirjandus

[1] AS Datel. ADS-iga liidestumise juhend. *Maa-amet*. 2013. pp 5-10. https://geoportaal.maaamet.ee/docs/aadress/ADS-ga_liidestumise_juhend.pdf (08.03.2022).

[2] Maa-ameti Geoportaal. https://geoportaal.maaamet.ee/est/Teenused/Integreeritav-aadressiotsing-In-ADS-p504.html (07.12.2021).

[3] Maa-ameti dokument. https://inaadress.maaamet.ee/inaadress/pdf/et/in_aadress_developer_manual.pdf (28.03.2022)

[4] W3Schools. *jQuery Introduction*. https://www.w3schools.com/jquery/jquery_intro.asp (28.03.2022)

[5] W3Techs. *Usage statistics of JavaScript libraries for websites.* https://w3techs.com/technologies/overview/javascript_library (28.03.2022)

[6] Oram, Andy, Wilson G., 2010. Making software: What really works, and why we believe it. *O'Reilly Media, Inc.,* pp 139-140. https://books.google.ee/books?id=DxuGi5h2-HEC&lpg=PA125&ots=0XutklX7lS&dq=more%20lines%20of%20code&lr&pg=PA125#v=onepage&q&f=false (16.04.2022)

[7] Front-end frameworks popularity (React, Vue, Angular and Svelte). *GitHub*. https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190 (07.12.2021).

[8] StackOverflow. https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Csvelte%2Cangularjs (07.12.21).

[9] Rascia T. React Tutorial: An Overview and Walkthrough. *TaniaRascia*. 20.08.18 https://www.taniarascia.com/getting-started-with-react/ (07.12.2021).

[10] React. https://reactjs.org/ (07.12.2021).

[11] React. https://reactjs.org/docs/how-to-contribute.html (07.12.2021).

[12] Ocariza, F., Bajaj, K., Pattabiraman, K., Mesbah, A., 2013, An Empirical Study of Client-Side JavaScript Bugs, *ACM / IEEE International Symposium on Empirical Software Engineering and Measuremen*t, pp. 55–56, http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf (09.12.21).

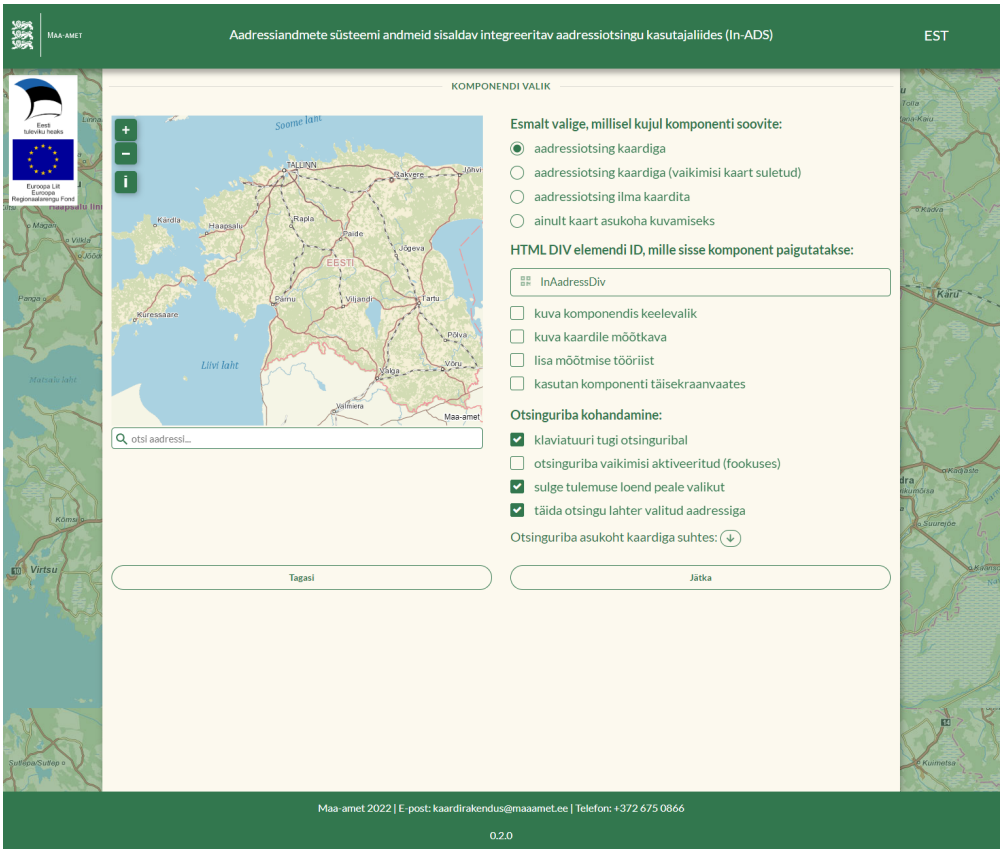
[13] Aggarwal, S., Modern Web-Development Using ReactJS, *International Journal of Recent Research Aspects*, vol. 5, no. 1, Mar. 2018, pp. 133–134,

https://scholar.google.com/citations?view_op=view_citation&hl=en&user=V-0DsakAAAAJ&citation_for_view=V-0DsakAAAAJ:u-x6o8ySG0sC (08.03.2022).
[14] W3Schools. *CSS Syntax.* https://www.w3schools.com/css/css_syntax.asp (07.12.2021).
[15] Semantic UI. *GitHub.* 21.10.18. https://github.com/Semantic-Org/Semantic-UI (07.12.2021).
[16] Semantic UI React. https://react.semantic-ui.com/ (07.12.2021).
[17] Bardas, A.G., 2010. Static code analysis. *Journal of Information Systems & Operations Management*, *4*(2), pp.99-107. http://www.rebe.rau.ro/RePEc/rau/jisomg/WI10/JISOM-WI10-A10.pdf (14.03.2022)
[18] Baca D. , Carlsson B., Lundberg L., 2008. Evaluating the cost reduction of static code analysis for software security. *Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security*, pp 79-88 https://www.diva-portal.org/smash/get/diva2:836174/FULLTEXT01.pdf (14.03.2022)
[19] Eclipse, 2019. SonarLint - Fix Issues Before They Exist. *Eclipse Newsletter*. https://www.eclipse.org/community/eclipse_newsletter/2019/march/sonarlint.php (14.03.2022)
[20] SonarLint toote kirjeldus https://www.sonarlint.org/?utm_source=ecl_news&utm_content=product (14.03.2022)
[21] Shah. V, 2020. SONARLINT VS SONARQUBE. *TatvaSoft*. https://www.tatvasoft.com/blog/introduction-to-sonarqube-sonarlint/#:~:text=Difference%20between%20SonarLint%20and%20SonarQube&text=SonarLint%20supports%20only%20in%20the,as%20you%20type%20your%20code. (14.03.2022)

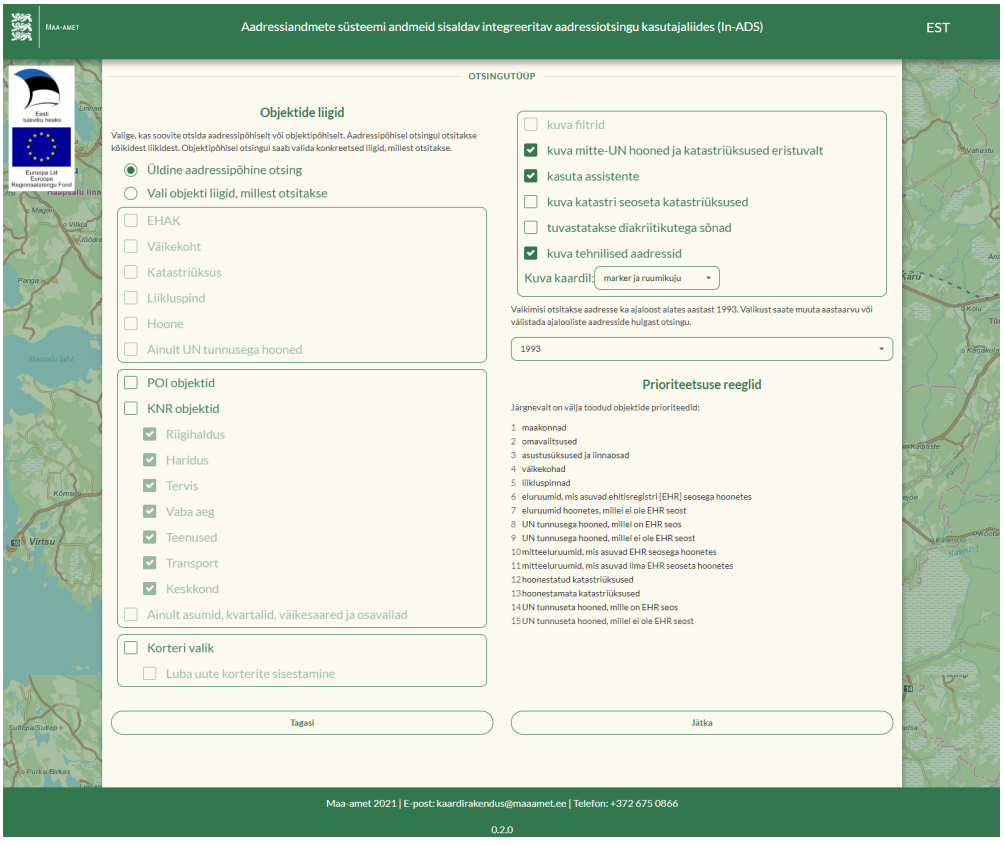
Lisad

1. Geokodeerimise rakenduse prototüüp: https://xd.adobe.com/view/67632147-f703-4094-a9c7-4de1e2200c6c-3230/grid (29.03.2022)
2. Maa-amet_geoportaal_disainikontseptsioon_versioon_1.0.pdf
3. Koodi genereerimise teekond vaikeväärtustega arvutis.

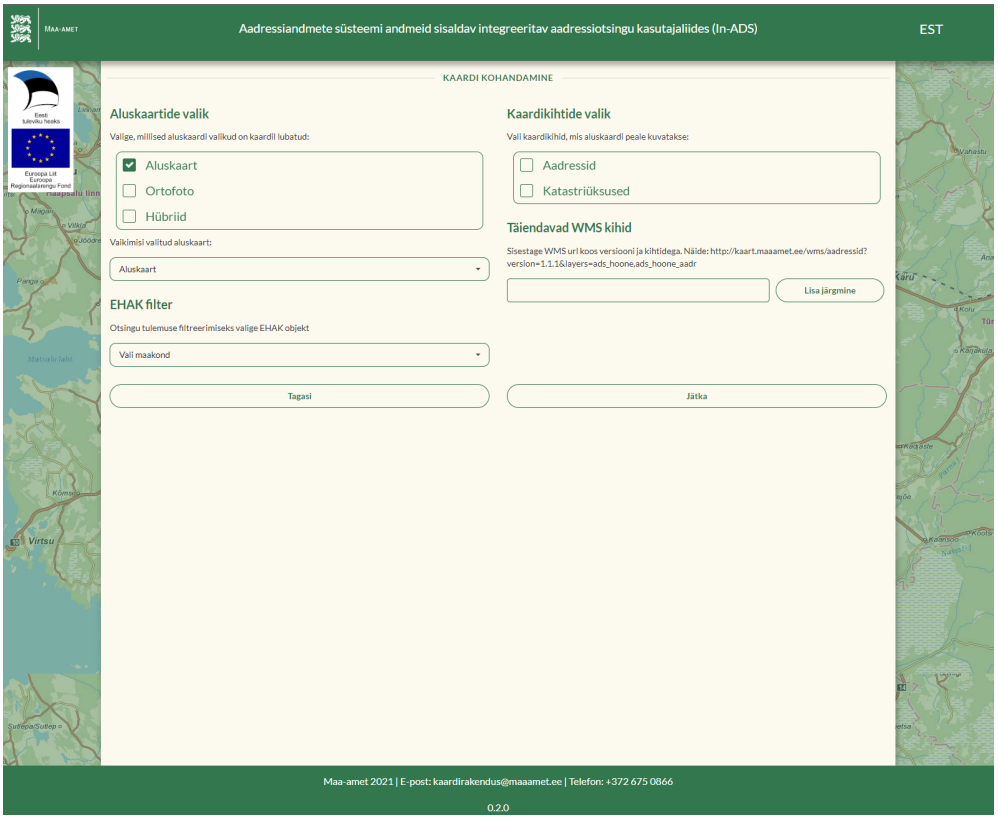
Lisajoonis 3.1. Uue koodigeneraatori esileht



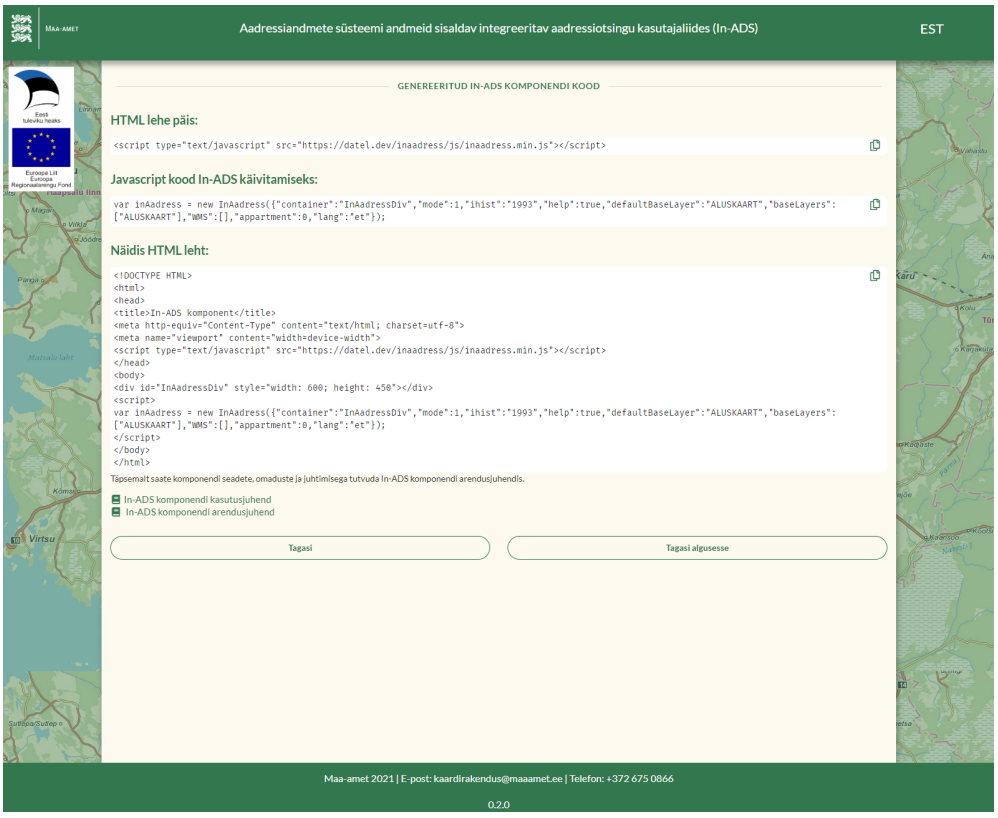
Lisajoonis 3.2. Uue koodigeneraatori komponendi valiku vaade



Lisajoonis 3.3. Uue koodigeneraatori otsingutüübi vaade



Lisajoonis 3.4. Uue koodigeneraatori kaardi kohandamise vaade



Lisajoonis 3.5. Uue koodigeneraatori geneeritud koodi tulemuse vaade

4. Koodi genereerimise teekond vaikeväärtustega mobiilis (iPhone 11)

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Sander Ruusmaa,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Kasutajaliides Maa-ameti aadressiotsingu komponendi koodigeneraatorile,
   mille juhendaja on **Vambola Leping**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates ***09.05.2022*** kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Sander Ruusmaa
09.05.2022