TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Markus Mänd 175900IDDR

EUROOPA LIIDU GDPR REGULATSIOONI RAKENDAMINE TELIA EESTI DIGIKANALITE NÄITEL

Bakalaureusetöö

Juhendaja: Jaanus Pöial
PhD
Kaasjuhendaja: Taavi Kivimaa
BSc

Tallinn 2021

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: Markus Mänd

07.01.2021

Annotatsioon

Käesolev bakalaureusetöö tutvustab Euroopa Liidu isikuandmete kaitse üldmäärust GDPR-regulatsiooni ning selle rakendamist Telia Eesti digikanalites.

Töö eesmärk on arendada Euroopa Liidu GDPR'i regulatsiooni rahuldav küpsiste halduslahendus. Ühtlasi annab see ülevaate sellest, miks oli vaja Telia Eesti digikanalites välja töötada uus küpsiste halduslahendus.

Antud lõputöö on jaotatud kaheks suuremaks osaks. Kõigepealt analüüsitakse Euroopa Liidu GDPR'i regulatsiooni ning antakse ülevaade, miks ei olnud Telia Eesti digikanalites varem kasutatav küpsiste halduslahendus sellega kooskõlas. Teises osas analüüsitakse arenduse funktsionaalsusi ning tehakse valik, kas arendada kohaldatud haldusrakendus või kasutada selleks mõnda olemasolevat teenust. Ühtlasi selgitatakse küpsiste halduslahenduse arenduskäiku ning seda, kuidas *modal*'i hallatakse.

Töö võib olla kasulik inimesele, kes soovib teada, kuidas Euroopa Liidu GDPR'i regulatsiooniga kooskõlas olevat küpsise haldussüsteemi arendada või tunneb huvi, miks Telia Eesti digikanalites tuli uus küpsiseid haldav süsteem luua.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, kahte peatükki, 19 joonist, ühte tabelit.

Abstract

Implementation of the European Union GDPR Regulation on the Example of Telia Estonia Digital Channels

This thesis introduces the European Union GDPR regulation and how it is being implemented on the example of Telia Estonia digital channels.

The main goal of this thesis is to implement a cookie management solution which satisfies the European Union GDPR regulation. Also, it gives an overview of why Telia Estonia digital channels needed a new cookie management solution.

This thesis has been divided into two parts. In the first part, the European Union GDPR regulation is analyzed and there is an overview of why the cookie management solution that was being used in Telia Estonia digital channels was not in accordance with it. In the second part of the thesis, the technical requirements are being analyzed and there is a choice made whether to develop a custom solution or to use an existing service. Furthermore, there is an explanation of the development process and how the newly developed modal would be maintained.

This thesis might be useful to people who would like to know how to develop a cookie management system that is in accordance with the European Union GDPR regulation or are interested in why Telia Estonia digital channels had to develop a new cookie management system.

The thesis is in Estonian and contains 22 pages of text, two chapters, 19 figures, one table.

Lühendite ja mõistete sõnastik

Cookie	Küpsis
CSS	Arenduskeel, mille abil saab deklareerida, kuidas HTML elemendid peavad veebilehel näidatud olema [1]
css-loader	Webpack'i laiendus, mis rakenduse ehitamisel tõlgendab CSS stiilifailides olevaid @import (stiilifailide importimiseks kasutatav silt) ja url() nagu import/require(). [2]
Custom	Kohaldatud
DOM	Programmeerimise liides (*interface*), mis on loodud HTML'ile. Rakendused saavad selle abil muuta dokumendi struktuuri, stiili ning sisu. [3]
Event listener	Javascript'is kasutuses olev objekt, mis kuulab tegevuse (n. nupu vajutus) järel saadetud sündmusi (*event*) [4]
Frontend	Kasutajaliides, kõik see mida rakenduse kasutaja näeb [5]
GDPR	Isikuandmete kaitse üldmäärus
HTML	Märgistuskeel, mille abil kirjeldatakse veebilehe struktuur [6]
IntelliJ	Arenduskeskkond
Interface	Kasutatakse Typescript'i andmetüüpide nimetamiseks. Ühtlasi võib *interface* defineerida, kui lepingut, mille abil saab märkida üles piirid, millest kirjutatud kood kinni peab pidama. Muutes *interface* mõjutab see koheselt kirjutatud koodi. [7]
Javascript	Objektorienteeritud arenduskeel, mida kastutakse põhiliselt veebirakenduste arenduses ning mis töötab veebilehe kasutaja seadmes. [8]
JSX	React'is kasutatav Javascript'i süntaksi laiend [9]
Loader	Webpack kasutab neid failide eellaadimiseks [10]

MiniCssExtractPlugin	Webpack'i laiend, mis loob iga Javascript'i faili kohta vastava CSS'i stiilifaili. [11]
Modal	Ehitatud HTML, CSS ning Javascript'iga ja positsioneerib lehel üle kõigi teiste elementide [12]
npm	Npm on maailma kõige suurem tarkvara register, mis aitab mugavalt projekti importida erinevaid teeke, mida saab kasutada üle terve rakenduse. [13]
package.json	Projekti juurkaustas paiknev fail, mis sisaldab projektiga seotud infot ning mille abil oskab npm sõltuvusi (*dependencies*) hallata. [14]
React	React on Javascript'i raamistik, millega on võimlaik ehitada lihtsaid vaateid, mis andmete muutumisel uuendavad ainult vajalike komponente, kasutades selleks *state*. [15]
Rollup	Javascript'i moodulite ehitaja, mis muudab Javascript'is ehitatud rakenduse väiksemad kooditükid üheks failiks. [16]
State	React komponendi sisse ehitatud objekt [17]
Story point	Ülesannete mahu ja keerukuse hindamiseks kasutatav mõõtühik, kus number võrdub arenduse valmimiseks kuluvate päevadega. [18]
style-loader	Webpack'i laiendus, mis rakenduse ehitamisel sisestab CSS'i, HTML'i, DOM'i. [19]
Typescript	Typescript on programmeerimiskeel, mis kujutab endast tüübikindlat Javascript'i koodi, kus on võimalik objekti tüüpe määrata ja defineerida. [20]
Webpack	Selle abil optimeeritakse koodi ning ühendatakse mitmed koodifailid üheks kokku, vähendades sellega rakenduse mahtu ning valmistades seda ette veebibrauseris kasutamiseks. [21]
XML	Märgistuskeel, mida kasutatakse andmete kirjeldamiseks. [22]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö teemaks on Euroopa Liidu isikuandmete kaitse üldmääruse GDPR-regulatsiooni rakendamine Telia Eesti digikanalites. Telia Eesti soovis olemasoleva halduslahenduse ümber töötada, et see muutunud seadusandlusega kooskõlla viia.

Euroopa Liidu Kohus jõudis 1. oktoobril 2019 Planet 49 kohtuasjas otsusele, mille tulemusel sätestati, et andmekaitseseaduse alusel rakenduse kasutajalt küpsise nõusoleku võtmine eeltäidetud märkeruudu läbi, ei ole kooskõlas GDPR regulatsiooniga. Ühtlasi toodi välja, et nõusoleku vorm peab olema eristatav ning selle sisu võimalikult arusaadav ning selge. [23]

Sarnaselt Telia Eestile oli Planet 49 veebipõhine rakendus ning mõlemale kehtisid samad GDPR regulatsioonid. Sel hetkel oli Telia Eesti digikanalites kasutusel küpsiste teavitus, mida kuvati lehe päise all ribana. Veebilehe esmasel külastamisel laeti veebibrauserisse küpsised olenemata kasutaja nõusolekust ja küpsise teavitus ei olnud selgesti eristatav. Seega tuli olemasolev lahendus seadusega kooskõlla viia.

Töö eesmärgiks oli välja töötada küpsiseid haldav lahendus, mis rahuldab Euroopa Liidu GDPR regulatsiooni. Ühtlasi pidi olema võimalikult kerge rakendust hallata ja tuleviku seadusmuudatuste puhul sinna ka uut infot lisada.

Lõputöö käigus uuritakse esmalt Euroopa Kohtu otsuses kajastatud regulatsioone ning tuuakse välja, millised on põhilised artiklid, mida peab arendatav lahendus täitma. Seejärel valiti välja lahendus, mis võtaks võimalikult vähe aega ning arendusressurssi. Peale seda kirjeldatakse tarkvaralise lahenduse realiseerimiseks kasutatavaid tehnoloogiaid ning täpsustatakse arenduse käiku. Lõpetuseks selgitatakse, kuidas toimib rakenduse haldamine.

2 Euroopa Liidu GDPR regulatsioon

Enne Planet 49 kohtuasja otsust puudus selge arusaam GDPR'i regulatsiooni osas, kuidas kujutada andmekasutusega nõustumist seadusega kooskõlastatult. Peale otsust said ettevõtted ülevaate täpsetest regulatsioonidest ning nende alampunktidest, mille jälgimine on kohustuslik. Sel hetkel oli selge, et enamus internetis baseeruvaid rakendusi neid ei jälginud. Järgnevalt tuuakse välja mõned tähtsamad punktid regulatsioonidest ning selgitatakse, kuidas need veebirakenduse kontekstis rakenduvad.

2.1 Euroopa Parlamendi ja Nõukogu direktiiv 2002/58/EÜ

Direktiivis käsitletakse isikuandmete töötlemist ja eraelu puutumatuse kaitset elektroonilise side sektoris. Selle raames kohustatakse liikmesriike tagama füüsiliste isikute õigused ja vabadused, võimaldades sellega isikuandmete vaba liikumise Euroopa Liidus. [24]

2.1.1 Artikkel 5 punkt 3

Punktis sätestatakse, et elektrooniliste sidevõrkude kasutamine teabe salvestamiseks ja juurdepääsuks eeltellija või kasutaja lõppseadmesse salvestatud teabele on lubatud ainult juhul, kui kasutajale on esitatud selge ja arusaadav teave andmete töötlemise eesmärgi kohta ning tal on võimalus keelduda sellest. [24]

2.1.2 Järeldus

Veebirakenduse kontekstis sätestab eelolev lõik, et küpsiste nõusoleku küsimine on legitiimne juhul, kui vastavas *modal*'is on selgelt välja toodud, milliste küpsistega kasutaja nõustub ning mis on nende eesmärk.

2.2 Määrus (EL) 2016/679

Määrus seab õigusnormid, mis reguleerivad füüsilise isiku isikuandmete vaba liikumist ja kaitset nende töötlemisel. Ühtlasi tunnistab see direktiivi 95/46/EÜ kehtetuks. [25]

2.2.1 Artikkel 4 punkt 11

Punktis sätestatakse mõiste „nõusolek“, mis tähendab vabatahtlikku, konkreetset, teadlikku ning ühemõtelist tahteavaldust, millega kasutaja nõustub kas avalduse vormi või selget nõusolekut väljendava tegevusega tema kohta käivate isikuandmete töötlemisega. [25]

2.2.2 Artikkel 6 punkt 1 a

Punktis sätestatakse, et isikuandmete töötlemine on seaduslik ainult juhul, kui kasutaja on andnud nõusoleku töödelda neid ühel või mitmel kindlal eesmärgil. [25]

2.2.3 Järeldus

Veebirakenduse kontekstis sätestatakse, et küpsiseid ei tohi kasutaja veebibrauserisse läbi eeltäidetud nõusoleku vormi või teavituse salvestada, vaid veebikülastaja peab olema selleks andnud selge loa.

2.3 Kokkuvõte regulatsioonist

Eelmainitud regulatsiooni, määruse ning nende artiklite põhjal anti ülevaate sellest, mida Telia Eesti digikanalitesse arendatav küpsiste haldamis lahendus pidi järgima. Kindlasti peab küpsiste salvestamine olema läbipaistavalt lahti selgitatud ning nõusoleku saamiseks tuleb need kuvada eraldi *modal*’i, mis eristuks ülejäänud veebilehest. Ühtlasi peab kasutaja otsus olema ühemõtteline ning ta peab saama otsustada, kas tahetakse küpsiste salvestamisega jätkata või mitte.

3 Metoodika

Peatükis kirjeldatakse, arenduse realiseerimiseks valitud teenusepakkujat. Ühtlasi selgitatakse milliseid arendusmeetodeid ja tööriistu kasutati. Lõpuks kirjeldatakse rakenduse malli arenduskäiku ja küpsiste *modal*'i haldamist.

3.1 Analüüs

Analüüsi eesmärgiks oli otsustada, kas arendatakse ise kohaldatud küpsiste haldussüsteem või liidestatakse mõni olemasolev teenus Telia Eesti digikanalitega.

3.1.1 Funktsionaalsed nõuded

Analüüsi esimeses järgus märgiti üles funktsionaalsused, mida peaks arendatav küpsiste haldussüsteem täitma.

Nendeks kujunesid:

- Skript/teek, mis küpsise nõusolekute valikut kuvab ja haldab. Sisuliselt see pool, mida näeb rakenduse kasutaja, kui ta veebilehel navigeerib.
- Haldusliides, kus on võimalik hallata küpsiste nimekirja, näha statistikat ning võimalusel tühistada küpsiste nõusolekuid. See on põhiliselt suunatud sisuhaldusele, kes võiks ilma suurema vaevata saada redigeerida küpsistega seonduvat infot.

Edaspidises analüüsis tehtud rahalised, kui ka ajalised hinnangud järgivad neid funktsionaalsusi.

3.1.2 Olemasolev teenus

Teenus, mida analüüsi käigus uuriti oli Cookiebot. Põhilised funktsioonid, mida olemasolev teenus pakkus olid:

- Automaatne küpsiste jälgimine – Cookiebot käib üks kord kuus üle kõik lehel olevad küpsised ning genereerib vastavalt leitule küpsiste deklaratsiooni. [26]

- Küpsiste kontroll ning Javascript'i tugi - Cookiebot pakub erinevaid omadusi, meetodeid, sündmusi ja funktsioone, mille abil saab *frontend*'i rakenduses aktiveerida skripte vastavalt kasutaja nõusoleku staatusele. [26]

- Globaalne küpsiste hoidla – Cookiebot viib veebilehel leitud küpsised kokku globaalses hoidlas oleva infoga ning seejärel küsib sealt neile selgitused. Peale seda salvestatakse info lokaalsesse küpsiste hoidlasse ja neid saab haldusrakenduses tõlkida ning muuta. Seda infot saab ühtlasi küpsise nõusoleku vormil välja näidata. [26]

- Küpsiste nõusoleku *modal* – Cookiebot võimaldab disainida ja kuvada küpsiste nõusolekuid *modal*'ina. [26]

3.1.3 Võimalike lahenduste võrdlus

Esmalt sai kaalutud ikkagi kohaldatud lahenduse arendamist, sest see tundus piisavalt väikse ajakuluga olevat ning saadav paindlikus oli seda väärt. Pärast arenduskulude kokku arvutamist otsustati, aga olemasolevaid teenuseid uurida ning nende maksumust võrrelda kohaldatud lahendusega.

Esmane kaalutlus oli arenduse maksumus ning ajakulu, mida kajastatakse Tabelis 1. Kuna tegemist on hinnangutega, siis on maksumused ligikaudsed, kuid nende suurusjärgud on õiged.

Tabel 1. Kohaldatud lahenduse ning olemasoleva teenuse võrdlus

	Kohaldatud lahendus	Cookiebot
Ajakulu	Ligikaudselt oleks arendusprotsess aega võtnud 300+ tundi.	Ligikaudselt oleks arendusprotsess võtnud aega alla 100 tunni.
Arendamise kulu	Arvutuses võeti arendaja tunnihinnaks konservatiivne 40 € ning	Olemasoleva teenuse üles seadistamine oleks maksnud 4000 €.

	sellest järeldus, et arenduse kogumaksumuseks oleks kujunenud 12 000 €.	
Lisakulud	Telias on mitu erinevat domeeni, kus oleks tulnud vastavat lahendust eraldi hooldada ning hallata ka pärast seda, kui esmane arendus on tehtud.	Olemasoleva teenusega oleks pidanud tasuma igakuist tellimustasu, mis oli suurusjärgus 250 € kuus. Haldus ning hooldustöödest poleks lisakulusid tekkinud.

Telia Eesti on telekom ettevõte ning oma enda kohaldatud küpsiste haldussüsteemi arendamine pole nende põhitegevusetega seotud. Seega võiks teise kaalutlusena võimalusel arendusressurssi suunata tegevusvaldkonnaga seotud toodete arendamisele.

Kolmanda punktina võib välja tuua, et Telia Eesti arendussektoris puudus kogemus küpsiste haldusega. Kuna tööga oli võrdlemisi kiire, siis oleks enda küpsiste haldussüsteemi arendamine olnud seetõttu ka ajakulukam.

Viimase punktina võib välja tuua Cookiebot'i haldus *modal*'i, mis on väga kergesti mõistetav ning võimaldas ka mitte arendajatel (sisuhalduritel) hallata, kuidas *modal* välja võiks näha ning milliseid küpsise nõusolekuid seal näidata võiks

Võttes arvesse arenduse maksumuse ning analüüsiosas väljatoodu, otsustati arenduse realiseerimiseks liidestada Telia Eesti digikanalitega olemasolev küpsiste haldusteenus (Cookiebot).

3.2 Arenduses kasutatud keeled, raamistikud ja metoodikad

Järgnevalt on toodud välja arenduses kasutatud keeled ja raamistikud, mida on teisteski Telia Eesti digikanalite projektides juurutatud ning arendatud rakenduses kasutatud.

3.2.1 React

React on Javascript'i raamistik, millega on võimlaik ehitada lihtsaid vaateid, mis andmete muutumisel uuendavad ainult vajalike komponente, kasutades selleks *state*. Deklaratiivsed vaated muudavad kirjutatud koodi arusaadavamaks ning võimaldavad kergemat koodi silumisprotsessi. [15]

Rakenduse arendamisel sai React'i kasuks otsustatud, kuna Telia Eesti digikanalites on teisedki projektid realiseeritud seda raamistikku kasutades. Ühtlasi võimaldab see kõigil Telia Eesti digikanalite arendajatel ilma õppimiskõverata seda projekti arendada.

3.2.2 Typescript

Typescript on programmeerimiskeel, mis kujutab endast Javascript'i koodi, kus on võimalik objekti tüüpe määrata ja defineerida. Kuna Javascript'il puudub võimalus tüübivigade leidmiseks enne rakenduse käivitamist, siis siin kohal aitabki Typescript koodi valideerida ning potentsiaalseid vigu ära hoida. [20]

Arenduses sai Typescript'i kasutatud selle pärast, et koodi kirjutamisel hoitaks ühtlast stiili. Ühtlasi võimaldab Typescript kasutada kõige uuemaid Javascript'i funktsioone, luues samal ajal võimaluse rakendusel toimida vanemates brauserites. See on eriti tähtis Telia Eesti digikanalite juures, kuna lehekülje kasutajate seas on inimesi, kes kasutavad vanemaid veebibrausereid.

3.2.3 Babel ja Webpack

Babel on sisuliselt kompilaator, mis võimaldab uuema põlvkonna Javascripti ümber töödelda vanemale versioonile ning seeläbi tagab rakenduse toimimise erinevates vanaemates brauserites (nt Internet Explorer 9). [27]

Kasutades Webpack'i kaardistatakse Babel'i poolt ümber töödeldud koodi moodulid ning genereeritakse pundardeks (*bundles*). Sisuliselt optimeeritakse koodi ning ühendatakse mitmed koodifailid üheks kokku, vähendades sellega rakenduse mahtu ning valmistades seda ette toodanguks. [21]

Webpack'i kasutati arendamisel, sest tegemist on puhtalt *frontend* rakendusega ning kompileerides koodi väiksemateks pundardeks (*bundles*) on seda parem sisestada Cookiebot'i haldusrakendusse.

3.2.4 Scrum

Scrum on raamistik, mis aitab arendusmeeskondadel paremini koostööd teha. Scrum'i raames jaotatakse arendus lühikesteks sprintideks. Iga sprindi alguses toimub planeerimine, mille raames annab meeskond ülesannetele hinnangud. Sprindi lõpus tehakse vastavalt sprindi tulemustele järeldused ning võetakse seda arvesse järgmise sprindi planeerimisel. Selline lähenemine tarkvara arendusele võimaldab meeskondadel pidevalt areneda. [28]

Rakenduse arendamisel kasutati Scrum'i raamistiku, sest see on teiste projektide ning arenduste abil testitud raamistik ning Telia Eesti digikanalite puhul toimib see hetkel kõige paremini.

3.2.5 Npm

Npm on maailma kõige suurem tarkvara register, mis aitab mugavalt projekti importida erinevaid raamistike, mida saab kasutada üle terve rakenduse. [13]

Rakenduse sõltuvuste üles märkimiseks tuleb need lisada *package.json* faili, mis peab asuma projekti juurkaustas. Sõltuvused alla laadimiseks tuleb jooksutada käsureal *npm install*.

Arenduses otsustati kasutada *npm*, sest see vähendab segadust koodist. Täpsemalt saab *npm'i* abil alla laetud sõltuvusi kasutada üle terve projekti importides seda otse moodulist ning seetõttu kaob ära vajadus sisestada *script*'i silte (*tag*) HTML koodi.

3.2.6 Rollup

Rollup on Javascript'i moodulite ehitaja, mis muudab Javascript'is ehitatud rakenduse väiksemad kooditükid üheks failiks. [16]

Üheskoos Typescript'i laiendusega võimaldab Rollup ka Typescript'i kooditükke töödelda. Kuna arenduses kasutatakse Typescript'i, siis on see põhjus, miks otsustati arenduse käigus seda sõltuvust kasutada.

3.3 Arendus

Arenduse eesmärgiks oli luua rakendus, mis Webpack'i abil minimeerib ReactJS, CSS ning HTML koodi kujule, mida saaks Cookiebot'i haldusrakendusse sisestada. Cookiebot

seejärel ehitab sisestatud koodi põhjal *modal*'i ning haldab, milliseid küpsiseid kasutaja veebibrauserisse salvestatakse.

3.3.1 Arenduskäik

Arenduse realiseerimisel jälgiti Scrum raamistiku, sprindi pikkuseks kujunes kaks nädalat. Iga sprindi alguses ning lõpus toimus koosolek, mille raames vaadati üle kahe nädala jooksul arendatud ülesanded ning plaaniti uue sprindi sisu. Ühtlasi anti meeskonnaga koos ülesannetele ajahinnangud, kasutades selleks *story point*'e. Võrreldes tavaliste ajahinnangutega, kus hinnatakse ülesandeid tundides on *story point*'id mitmekülgsemad ning võimaldavad arendajal tööd teha vabamalt.

Arendus algas Cookiebot'i malli loomisega IntelliJ's. Selleks kopeeriti olemasolevast Telia *frontend* projektist üldine kaustade struktuur ning *npm* jaoks vajalik *package.json* fail. Seejärel hakkas põhilise äriloogika arendamine, milleks oli Cookiebot'i jaoks vajalike skriptide, HTML ning CSS välja töötamine.

Skriptide arendamiseks loodi juurkausta Typescript'i failid, kuhu paigutati *modal*'il olevate nuppude ja rippmenüüde loogika. Selleks, et *modal*'il olevate tegevuste abil käivituks õige skript loodi *event listener*'id, mis reageerisid HTML elemendi muutumisele või vajutusele (Joonis 1). Ühtlasi deklareeriti, kuidas *modal* peab käituma, kui kasutaja ilma varasema küpsise nõusolekuta lehele navigeerib.

```
allowAllButton.addEventListener( type: 'click', listener: function (event : MouseEvent ) {
  event.preventDefault();
  Cookiebot.dialog.submitConsent();
});

allowSelectionButton.addEventListener( type: 'click', listener: function (event : MouseEvent ) {
  event.preventDefault();
  teliaCookiebotHandleCustomizedCookieSubmit();
});

selectViewButton.addEventListener( type: 'click', listener: function (event : MouseEvent ) {
  event.preventDefault();
  switchToSelectView();
});

collapseButton.addEventListener( type: 'click', listener: function (event : MouseEvent ) {
  event.preventDefault()
  if (collapseOpen) {
    return teliaCookiebotHideCollapse();
  }
  teliaCookiebotShowCollapse();
});
```

Joonis 1. *Event listener*'i lisamine Javascript'is

Samaaegselt tegeleti ka HTML malli loomisega. Kasutades näitena protoüüpi, arendati kolm React'i komponenti ning üks üldine mall (*template*), kus neid kolme kasutati. Komponentides deklareeriti JSX'is elementide paigutus algselt ilma stiilideta. Typescript'is olevate *event listener*'ide registreerimiseks oli vaja lisada nuppudele ning rippmenüüdele ID'd, mille abil sai dokumendist leida elemendi infot (Joonis 2).

```
collapseButton = document.getElementById( elementId: 'collapseBtn');
const selectViewButton = document.getElementById( elementId: 'showDetails');
const allowAllButton = document.getElementById( elementId: 'allowAll');
const allowSelectionButton = document.getElementById( elementId: 'allowSelection');
const checkboxes = [document.getElementById( elementId: "statistics"), document.getElementById( elementId: "marketing")];
const languages = [document.getElementById( elementId: 'firstLangChoice'), document.getElementById( elementId: "secondLangChoice")];
```

Joonis 2. HTML'i elementide info leidmine id järgi

Hiljem, kui funktsioonid ning JSX paigutus oli valmis, hakati tegelema CSS'i kirjutamisega. Selle käigus loodi igale Telia Eesti digikanalite domeenile eraldi stiilifail. Eesmärgi saavutamiseks loodi Javascript'i failid, kuhu imporditi olemasolevast komponentide kogumikust vajalikud stiilid (Joonis 3).

```
import './entry.base'

import '@brandful-front/collapse/lib/telia.css'
import '@brandful-front/modal/lib/telia.css'
import '@brandful-front/reset/lib/telia.css'
import '@brandful-front/grid/lib/telia.css'
import '@brandful-front/icon/lib/telia.css'
import '@brandful-front/button/lib/telia.css'
import '@brandful-front/anchor/lib/telia.css'
import '@brandful-front/typography/lib/telia.css'
import '@brandful-front/list/lib/telia.css'
import '@brandful-front/check/lib/telia.css'
import '@brandful-front/text/lib/telia.css'
import '@brandful-front/collapse-toggle/lib/telia.css'
import '@brandful-front/heading/lib/telia.css'
import '@brandful-front/separator/lib/telia.css'
```

Joonis 3. Stiiliklasside importimine komponentide kogumikust

Mõningate mallil üldist kasutust leidnud elementide jaoks oli vajalik kirjutada kohaldatud stiilid. Selleks loodi klassikaline CSS fail, kuhu deklareeriti klassid ning nende parameetrid (Joonis 4).

```css
#CybotCookiebotDialog.is-visible {
 position: fixed;
 width: 100%;
 height: 100%;
 overflow: auto;
 overflow-y: scroll;
 z-index: 1000;
}

#cbDialog {
 background: none !important;
}

.collapse {
 display: block;
}

.modal {
 padding: 0;
}

.modal:not(.is-entered) {
    display: none;
}

.modal__footer--sticky {
 transform: translate3d(0,0,0);
}
```

Joonis 4. Kohaldatud CSS faili sisu

Peale seda, kui üldine äriloogika oli arendatud, hakati tegelema Webpack'i ehitus konfiguratsioonide kirjutamisega. Selleks sai loodud kaks konfiguratsiooni faili, kuhu deklareeriti vajalikud skriptid, mis hakkasid JSX ning CSS'i minimeerima.

JSX'i konfiguratsiooni kirjutamisel tuli kasutusse Babel. Sisuliselt kasutati seal olevat abistavat laadijat (*loader*), mis otsis üle projekti JSX ja Javascript'i laiendiga faile ning töötles need ümber minimeeritud HTML'iks (Joonis 5).

```
{
  test: /\.(js|jsx)$/,
  exclude: /node_modules/,
  loader: "babel-loader",
  options: { presets: ['@babel/preset-react'] }
},
```

Joonis 5. Koodilõik JSX'i töötlevast Webpack'i konfiguratsioonist

CSS'i konfiguratsiooni skripti loomisel tuli jälgida, et minimiseerimise käigus tekkiks iga domeeni kohta eraldi CSS fail ning Javascript laiendiga failides olevad impordid saaksid

CSS'iks ümber töödeldud. Selleks võeti kasutusele *css-loader*, *style-loader*, *MiniCssExtractPlugin* laiendused, mis töötlesid kõiki CSS laiendiga faile. (Joonis 6).

```
{
  test: /\.css$/,
  use: [
    "style-loader",
    MiniCssExtractPlugin.loader,
    "css-loader",
  ],
},
```

Joonis 6. Koodilõik CSS'i töötlevast Webpack'i konfiguratsioonist

Ühtlasi kõikide Javascript'i laiendiga failide sees olevate stiili importide minimeerimiseks deklareeriti konfiguratsiooni failis nende asukohad. Seejärel kasutades *MiniCssExtractPlugin*'i töödeldi nad ümber eraldi CSS'i stiilifailideks (Joonis 7).

```
entry: {
  telia: './src/css/entry.telia.js',
  diil: './src/css/entry.diil.js',
  free: './src/css/entry.free.js',
  greenit: './src/css/entry.greenit.js',
  mtasku: './src/css/entry.mtasku.js',
  neti: './src/css/entry.neti.js',
  simpel: './src/css/entry.simpel.js',
  super: './src/css/entry.super.js',
  tv: './src/css/entry.tv.js',
  vunk: './src/css/entry.vunk.js'
},
resolve: {
  extensions: ['.js'],
  modules: [path.join(__dirname, "node_modules")]
},
output: {
  path: path.resolve(__dirname, './build'),
  filename: 'js/[name].js',
  publicPath: '/',
  pathinfo: ifNotProd()
},
plugins: removeEmpty([
  new MiniCssExtractPlugin({filename: 'css/[name].css'}),
  ifProd(
    new PurgecssPlugin( options: {
    whitelist: [COOKIEBOT_DIALOG_ID.replace( searchValue: "#",  replaceValue: '')],
    paths: [path.join(__dirname, 'build', 'index.html'), ...glob.sync( pattern: `${path.join(__dirname, 'src')}/**/*`,  options: { nodir: true })],
  })),
  new WebpackOnBuildPlugin( callback: function () {
    // fs.rmdirSync(path, {recursive: true}) does not work on linux
    deleteFolderRecursive(path.resolve(__dirname, './build/js'));
  })
]),
```

Joonis 7. Koodilõik Javascript'i failides olevate CSS importe töötlevast Webpack'i konfiguratsioonist

Seejärel oli vaja luua veel Rollup konfiguratsioon, mille abil töödeldakse Typescript'i koodi. See oli võrdlemisi kergem, kui Webpack'i konfiguratsiooni seadistamine. Sisuliselt tuli deklareerida sisendfaili asukoht ning sihtkoht kuhu see peale töötlemist paigutatakse (Joonis 8).

```
export default {
  input: './src/js/TeliaCookieBot.ts',
  output: {
    file: './build/TeliaCookieBot.js',
    format: 'iife',
    extend: true,
    name: 'window',
  },
  plugins: [typescript()],
  onwarn: function(warning, warn) {
    // suppress eval warnings
    if (warning.code === 'EVAL') return
    warn(warning)
  }
};
```

Joonis 8. Koodilõik Typescript'i töötlevast Rollup'i konfiguratsioonist

Lõpuks kui minimeerimiseks vajalike skriptide konfiguratsiooni failid olid loodud, tuli nende jaoks deklareerida käsud *package.json*'is. Selleks lisati *scripts* objekti Webpack'i ning Rollup'i käivitamise käsud. Lisaks sellele loodi *npm build* käsk, mis eelpool deklareeritud käske järjest välja kutsub (Joonis 9). Selle kõige tulemusena paigutati minimeeritud koodifailid rakenduse *build* kausta.

```
"build": "npm run build:html && npm run build:css && npm run build:js",
"build:css": "webpack -p --env.prod --config webpack.css.config.js --mode production",
"build:html": "webpack -p --env.prod --config webpack.html.config.js --mode production",
"build:js": "rollup -c rollup.js.config.js",
```

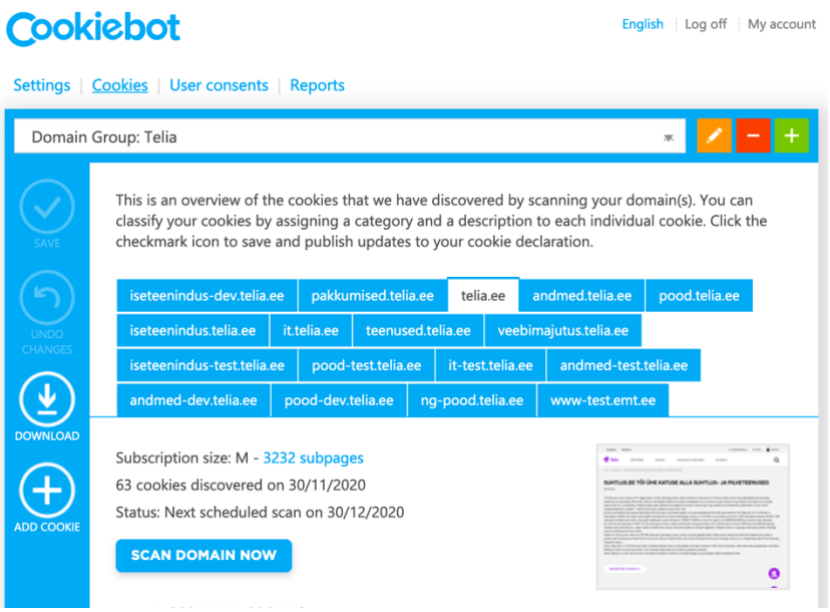
Joonis 9. Koodilõik rakendust ehitavatest *npm* käskudest

Webpack'i skriptide ehitamine oli arenduse kõige keerulisem osa, sest see kõik oli töö autori jaoks täiesti uus. Lugedes dokumentatsiooni ning samal ajal skripte kirjutades õnnestus aga vajalikud ehituskäsud valmis teha.

Ühtlasi täideti arenduse eesmärk – loodi mall rakendus, mille ehitus skriptide tulemusel tekkinud minimeeritud CSS, Javascript ning HTML koodi saab sisestada Cookiebot'i haldusrakendusse. Haldusest ning kohtadest kuhu kood sisestada tuleb kirjutatakse täpsemalt järgmises peatükis.

3.4 Infohaldus

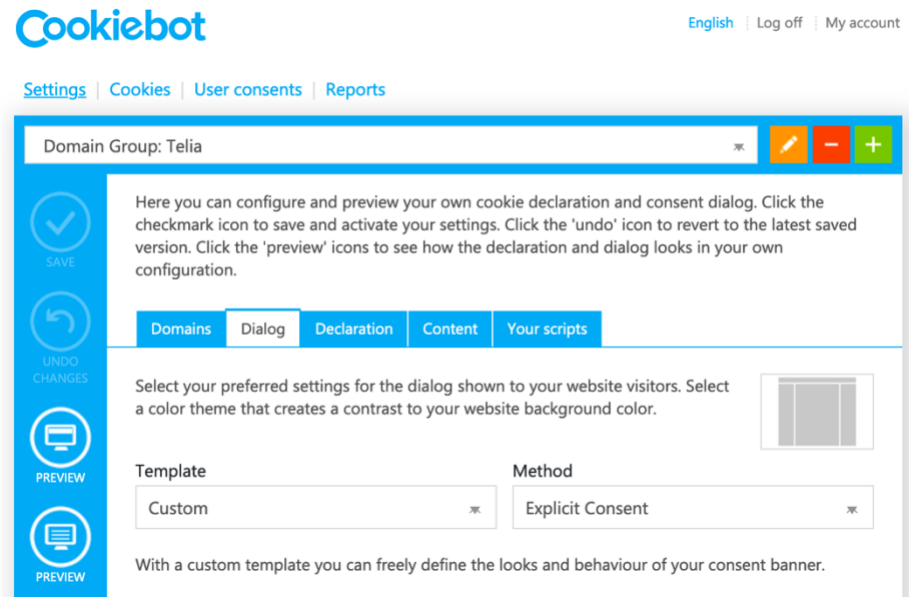
Küpsistega seotud info haldamine on koondatud Cookiebot'i pakutavale veebilehele, mis on nähtav Joonisel 10. Seal on võimalik uurida statistikat ning lisada ja eemaldada küpsiste nõusolekuid.



Joonis 10. Cookiebot'i haldusrakendus

3.4.1 *Modal*'i seadistus

Navigeerides seadete vahelehele on võimalik seadistada küpsiste *modal* ning seal olevat sisu. Joonisel 11 on välja toodud vastava lehe sisu koos seal olevate vahelehtedega.



Joonis 11. Cookiebot'i haldusrakenduse seadete vaheleht

„Dialog" vaheaknas on võimalik seadistada küpsiste *modal'i* HTML paigutust, stiili ning lõpuks sisestada ka Javascript'i funktsioone.

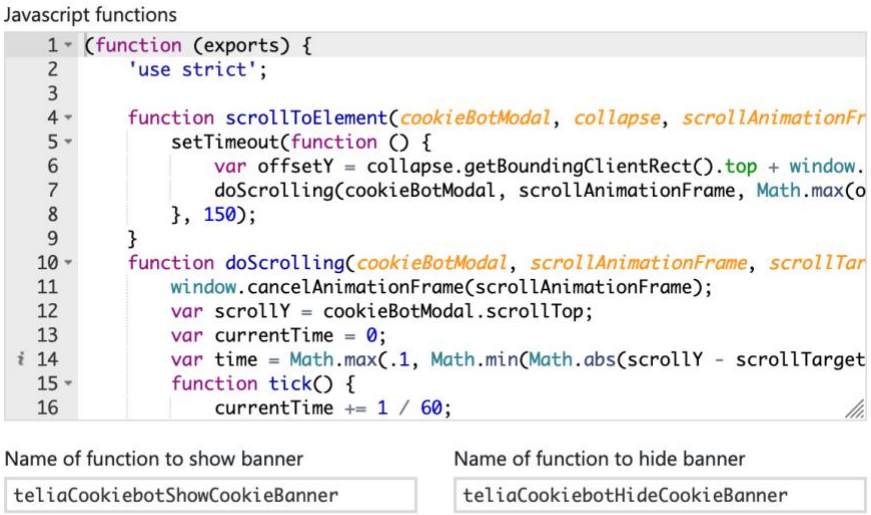
Rakenduse ehitamise käigus genereeritud index.html sisu sisestatakse Joonisel 12 näidatud kasti. Selle abil ehitab haldusrakendus *modal*'i sisu.

Joonis 12. Cookiebot'i HTML'i sisestamise aken

Ühtlasi on eraldi kast (Joonis 13) mõeldud ka minimiseeritud CSS jaoks, kuhu sisestatakse ehituse käigus loodud stiilifaili sisu, mille abil annab rakendus küpsise *modal*'ile stiili. Kuna Telia Eesti digikanalites on mitu domeeni, siis on oluline jälgida, et iga domeeni jaoks saaks sellele vastav CSS sisestatud.

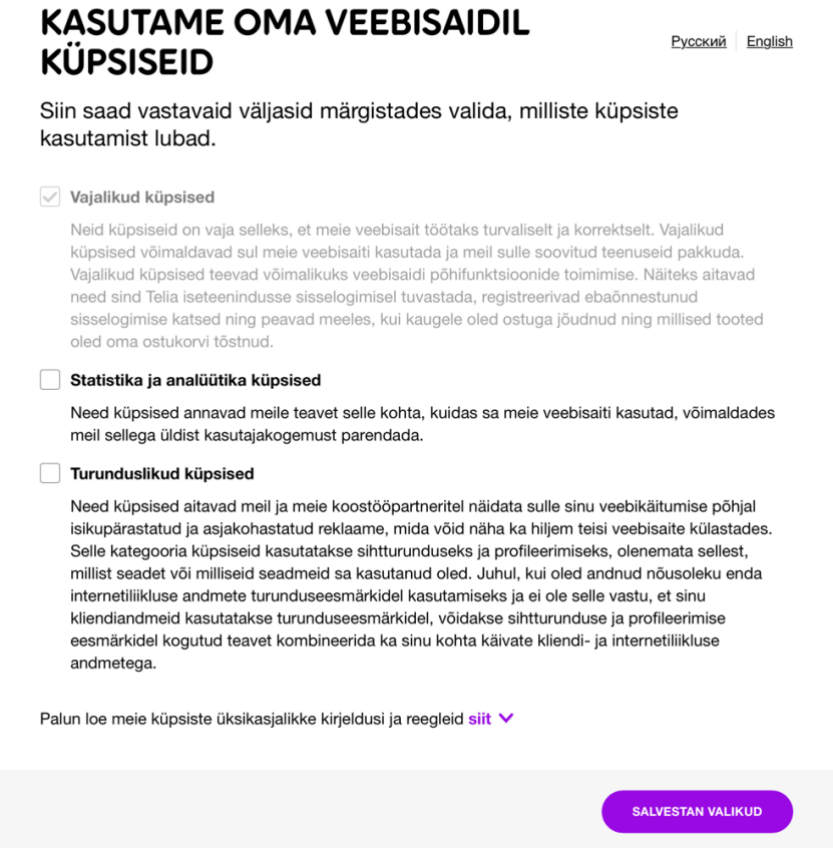
Joonis 13. Cookiebot'i stiilide sisestamise aken

Rollup'i abil töödeldud Javascript'i kood sisestatakse Joonisel 14 kuvatud „Javascript functions" aknasse. See deklareerib funktsioonid, mida küpsiste *modal'is* olevad tegevused kasutavad.



Joonis 14. Cookiebot'i Javascript'i funktsioonide sisestamis aken

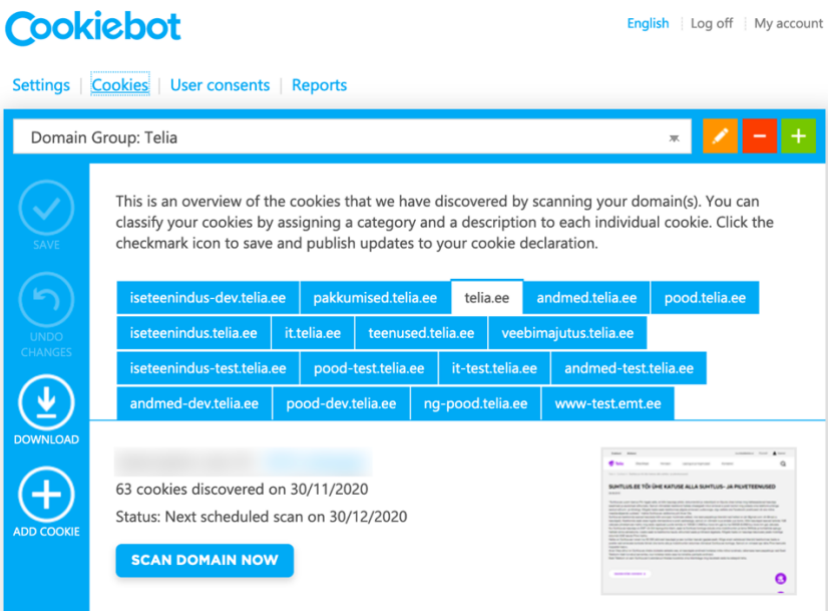
Lõpuks seadistamise tulemusel näidatakse Telia Eesti Digikanalite domeenidel Joonisel 15 välja toodud *modal'i*. Sellist vormi näeb veebilehe kasutaja, kui ta navigeerib www.telia.ee lehele.



Joonis 15. www.telia.ee küpsise *modal*.

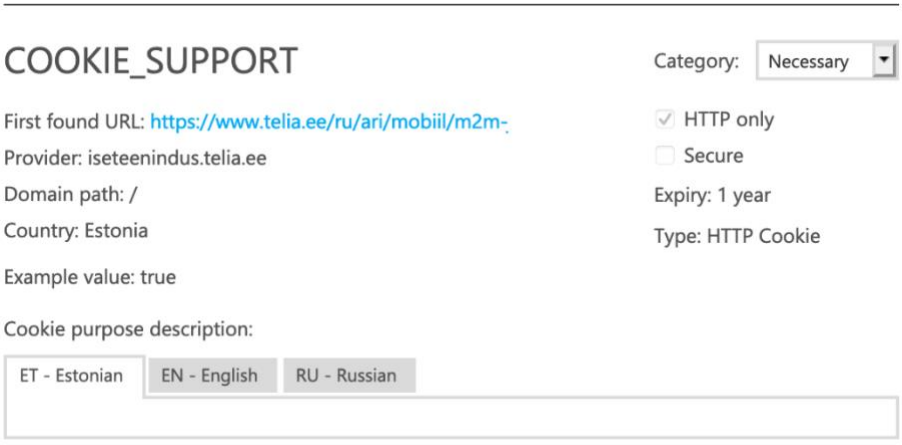
3.4.2 Küpsiste haldamine

Cookiebot võimaldab haldusrakenduses kergesti küpsiseid hallata ning juurde lisada. Selleks tuleb navigeerida „Cookies“ alamlahele, mida on kuvatud Joonisel 16.



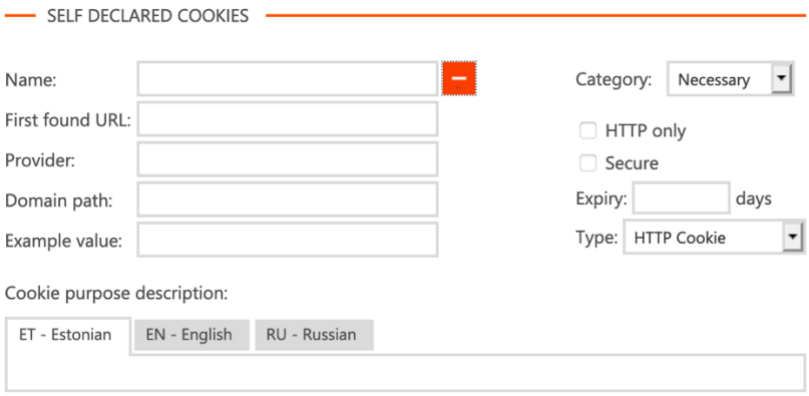
Joonis 16. Küpsiste haldus alamleht

Üldiselt skaneerib Cookiebot lehte ning selle käigus leitud uued küpsised kuvatakse haldusrakendusse. Seejärel seadistatakse tagaplaanil neile kategooria ning näidatakse milliselt lehelt vastav küpsis leiti. Seda kuidas infot välja näidatakse ning haldusrakenduses kuvatakse on välja toodud Joonisel 17.



Joonis 17. Näide Cookiebot'i skaneerimise funktsiooni tulemusel leitud küpsisest Telia Eesti digikanalites

Ühtlasi on võimalik küpsiseid ka manuaalselt lisada. Selleks tuleb Joonisel 16 kuvatud vahelehe vasakul ribal vajutada „ADD COOKIE" nuppu. See avab uue küpsise lisamise vormi (Joonis 18), kus on võimalik manuaalselt täita info, mida muidu Cookiebot ise automaatselt teeb.



Joonis 18. Cookiebot'i haldusrakenduses uue küpsise lisamine

3.4.3 Küpsiste statistika

Lisaks küpsiste lisamisele ja *modal*'i seadistamisele on võimalik Cookiebot'i haldusrakenduses vaadata kasutajate nõusolekute statistikat. Seda infot saab näha „User consents" vahelehel, mis on välja toodud Joonisel 19.

Joonis 19. Cookiebot'i küpsiste statistika vaheleht

Lehel on näidatud kui palju veebilehe külastajaid on nõustunud või keeldunud küpsistega. Ühtlasi on „Renew now“ nupu abil võimalik kõikide veebilehe kasutajate küpsiste nõusolekuid tühistada. See on kasulik, kui peaks lisanduma mõni uus küpsis, mille nõusolekut peaks kasutajalt uuesti küsima.

Kokkuvõtlikult võib öelda, et haldusrakendus täidab arenduse funktsionaalseid nõudeid, olles kergesti kasutatav ning võimaldades nii arendajatel, kui ka sisuhalduritel küpsise *modal'i* hallata.

4 Kokkuvõte

Bakalaureusetöö eesmärgiks oli arendada Euroopa Liidu GDPR-regulatsiooni rahuldav küpsiste halduslahendus. Lisaks annab töö ülevaate, miks oli vaja Telia Eesti digikanalites välja töötada uus küpsiste halduslahendus.

Töö esimeses osas analüüsiti Euroopa Liidu GDPR-regulatsiooni ning seda, miks ei olnud Telia Eesti digikanalites varem kasutuses olnud küpsiste halduslahendus sellega kooskõlas. Teises osas analüüsiti arenduse funktsionaalsusi ning tehti valik, kas arendada kohaldatud haldusrakendus või kasutada selleks mõnda olemasolevat teenust. Ühtlasi selgitati küpsiste halduslahenduse arenduskäiku ning seda, kuidas *modal'i* hallatakse.

Telia Eesti digikanalitesse tuli luua uus küpsiseid haldav lahendus, mis rahuldaks Planet 49 kohtuasja tulemusel selginenud GDPR-regulatsiooni. Probleemi lahendamiseks loodi mall, mille töödeldud kood sisestati olemasoleva küpsiste haldusteenuse haldusrakendusse. Selle tulemusel valmis Telia Eesti digikanalitesse küpsiseid haldav *modal*, mis võimaldas kasutajat detailselt teavitada lehel olevate küpsiste olemasolust ning andis veebikülastajale otsustusõiguse konkreetsete küpsiste allalaadimiseks (Lisa 2).

Bakalaureusetöö käigus töötas autor välja kergesti kasutatava ning GDPR-regulatsiooni rahuldava küpsiste halduslahenduse, mis on heaks näiteks teistele veebipõhistele rakendustele, millel puudub seadusega kooskõlas olev lahendus. Ühtlasi omandas autor põhjalikud teadmised Euroopa Liidu GDPR-regulatsiooni tähtsamatest punktidest. Lisaks õppis autor, kuidas välja töötada andmekaitse seadusi rahuldavat küpsiste halduslahendust. Autori arvates oli üks töö kõige keerulisem osa *Webpack'i* konfiguratsiooni kirjutamine, sest tal puudusid selles osas varasemad teadmised.

Kasutatud kirjandus

[1] „MDN Web Docs,“ November 2020. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Kasutatud 27 Detsember 2020].

[2] „webpack,“ Detsember 2020. [Võrgumaterjal]. Available: https://webpack.js.org/loaders/style-loader/. [Kasutatud 29 Detsember 2020].

[3] „MDN Web Docs,“ Detsember 2020. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. [Kasutatud 02 Jaanuar 2021].

[4] „MDN Web Docs,“ Detsember 2020. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/API/EventListener. [Kasutatud 29 Detsember 2020].

[5] „Concepta,“ Veebruar 2019. [Võrgumaterjal]. Available: https://www.conceptatech.com/blog/difference-front-end-back-end-development. [Kasutatud 27 Detsember 2020].

[6] „MDN Web Docs,“ Veebruar 2020. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Kasutatud 27 Detsember 2020].

[7] „Typescript,“ Detsember 2020. [Võrgumaterjal]. Available: https://www.typescriptlang.org/docs/handbook/interfaces.html. [Kasutatud 27 Detsember 2020].

[8] „MDN Web Docs,“ Detsember 2020. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Kasutatud 27 Detsember 2020].

[9] „ReactJS,“ Detsember 2020. [Võrgumaterjal]. Available: https://reactjs.org/docs/introducing-jsx.html. [Kasutatud 29 Detsember 2020].

[10] „webpack,“ 2020. [Võrgumaterjal]. Available: https://webpack.js.org/loaders/. [Kasutatud 29 Detsember 2020].

[11] „webpack,“ Detsember 2020. [Võrgumaterjal]. Available: https://webpack.js.org/plugins/mini-css-extract-plugin/. [Kasutatud 29 Detsember 2020].

[12] „GitHub,“ Detsember 2020. [Võrgumaterjal]. Available: https://github.com/twbs/bootstrap/blob/main/site/content/docs/5.0/components/modal.md. [Kasutatud 27 Detsember 2020].

[13] „npmjs,“ November 2020. [Võrgumaterjal]. Available: https://docs.npmjs.com/about-npm. [Kasutatud 27 November 2020].

[14] „nodejs,“ August 2011. [Võrgumaterjal]. Available: https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/. [Kasutatud 29 Detsember 2020].

[15] „React,“ November 2020. [Võrgumaterjal]. Available: https://reactjs.org/. [Kasutatud 21 November 2020].

[16] „Rollup.js,“ April 2020. [Võrgumaterjal]. Available: https://rollupjs.org/guide/en/. [Kasutatud 28 Detsember 2020].

[17] „GitHub,“ Veebruar 2019. [Võrgumaterjal]. Available: https://github.com/reactjs/reactjs.org/blob/master/content/docs/faq-state.md. [Kasutatud 27 Detsember 2020].

[18] „Atlassian,“ 2020. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/project-management/estimation. [Kasutatud 27 Detsember 2020].

[19] „webpack,“ Oktoober 2020. [Võrgumaterjal]. Available: https://webpack.js.org/loaders/style-loader/. [Kasutatud 29 Detsember 2020].

[20] „TypeScript,“ 2020. [Võrgumaterjal]. Available: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html. [Kasutatud 22 November 2020].

[21] „webpack,“ November 2020. [Võrgumaterjal]. Available: https://webpack.js.org/. [Kasutatud 23 November 2020].

[22] „WhatIs,“ Detsember 2014. [Võrgumaterjal]. Available: https://whatis.techtarget.com/definition/XML-Extensible-Markup-Language. [Kasutatud 27 Detsember 2020].

[23] K. Wiedemann, „The ECJ’s Decision in ‘‘Planet49’’ (Case C-673/17): A Cookie Monster or Much Ado About Nothing?,“ 23 Märts 2020. [Võrgumaterjal]. Available: https://link.springer.com/content/pdf/10.1007/s40319-020-00927-w.pdf. [Kasutatud 18 November 2020].

[24] E. Ü. TEATAJA, „EUR-lex,“ 31 Juuli 2002. [Võrgumaterjal]. Available: https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:32002L0058&from=ET. [Kasutatud 21 November 2020].

[25] E. Ü. TEATAJA, „EUR-Lex,“ 04 Mai 2016. [Võrgumaterjal]. Available: https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:32016R0679&from=et. [Kasutatud 21 November 2020].

[26] „Cookiebot,“ Cookiebot, 2020. [Võrgumaterjal]. Available: https://www.cookiebot.com/en/functions/. [Kasutatud 2 Detsember 2020].

[27] „BabelJS,“ November 2020. [Võrgumaterjal]. Available: https://babeljs.io/docs/en/. [Kasutatud 23 November 2020].

[28] C. Drumond, „Atlassian,“ 2020. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/scrum. [Kasutatud 24 November 2020].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

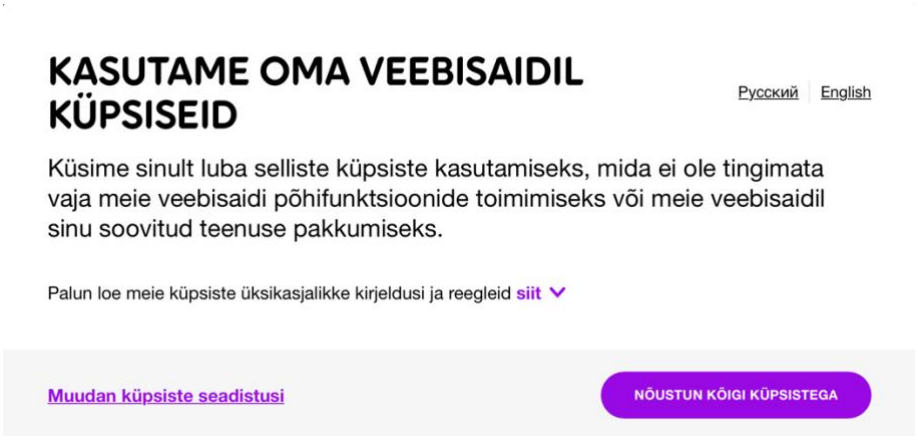
Mina, Markus Mänd

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Euroopa liidu GDPR regulatsiooni rakendamine Telia Eesti digikanalite näitel“, mille juhendaja on Jaanus Pöial
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

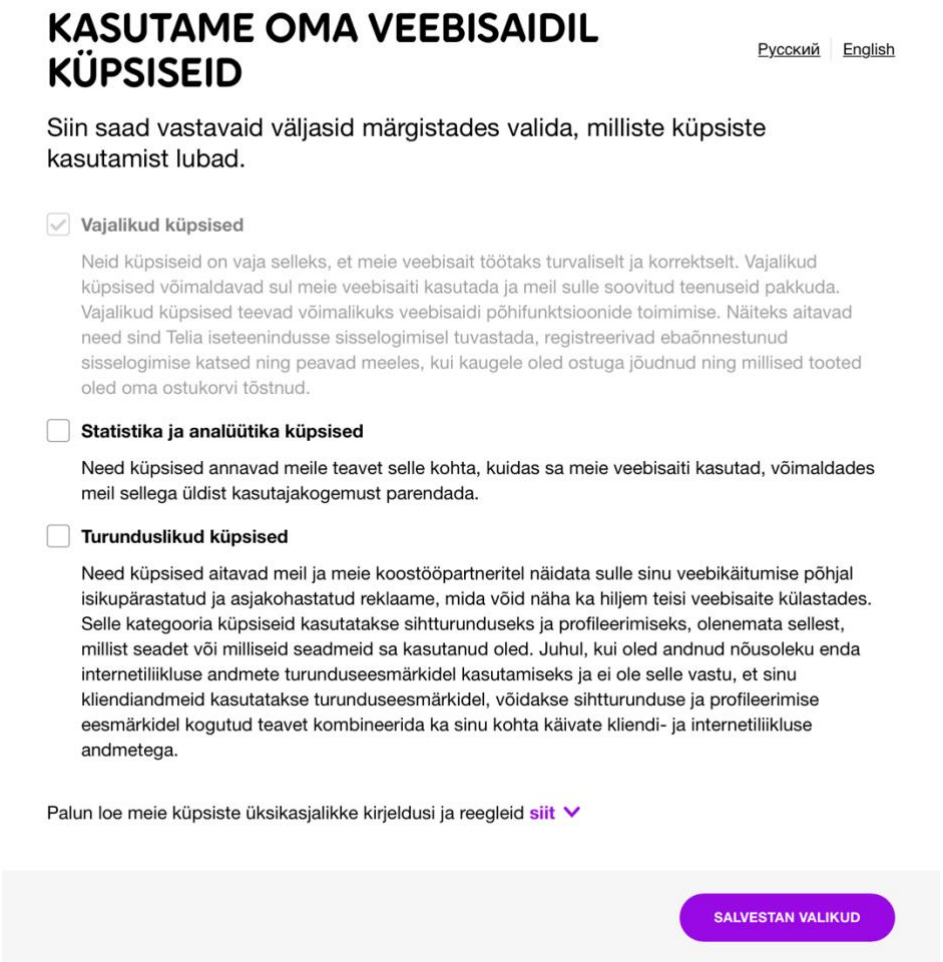
07.01.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Arendatud küpsise *modal*



Joonis 20 Arendatud küpsise haldus *modal*



Joonis 21 Arendatud küpsise *modal*'i küpsiste seadistus