TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Iris Rumm 193405IADB

# Veebikomponentide teegi arendus Smartmatic stiiliraamatu baasil

Bakalaureusetöö

Juhendaja: Karl-Erik Karu
MSc

Tallinn 2023

# 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: Iris Rumm

15.05.2023

# Annotatsioon

Käesoleva lõputöö eesmärk on luua kasutajaliidese komponentide teek, mida saab korduvkasutada ettevõtte Smartmatic-Cybernetica Centre of Excellence for Internet Voting OÜ käimasolevates ja tulevastes projektides. Komponenditeegi loomisel võetakse aluseks Smartmatic stiiliraamat.

Töö koosneb teoreetilisest ja praktilisest osast. Teoreetilises osas tutvustatakse kasutatavate tehnoloogiate ja veebi ligipääsetavuse tausta ning kaardistatakse loodavate komponentide visuaalne stiil, vajalik funktsionaalsus ja ligipääsetavuse nõuded. Samuti võrreldakse erinevaid raamistikke komponenditeegi loomiseks.

Praktilises osas arendatakse vastav komponenditeek. Komponente testitakse ühiktestimisega ning luuakse näidisprojektid Angular.IO ja Vue.js raamistikes. Valminud teek publitseeritakse NPM registris. Tulemuseks on Smartmatic stiiliraamatuga kooskõlas komponenditeek, mis vastab oodatud ligipääsetavuse nõuetele ning on taaskasutatav ettevõtte käimasolevates ja tulevastes projektides.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 6 peatükki, 23 joonist, 5 tabelit.

# Abstract

## Development of a Web Component Library based on the Smartmatic Style Guide

The goal of this thesis is to create a web component library that can be reused in current and future projects of Smartmatic-Cybernetica Centre of Excellence for Internet Voting OÜ. The Smartmatic style guide will be used as a basis for the creation of the library.

The thesis consists of a theoretical and a practical part. In the theoretical part, the background of the used technologies and web accessibility is introduced and the necessary functionality, accessibility requirements and visual style of the components is mapped. In addition, different frameworks for creating the library are compared.

In the practical part, the corresponding component library is developed. Components are tested with unit testing and example projects are made in Angular.IO and Vue.js frameworks. Finished library is published in the NPM registry. The result is a library that is compliant with the Smartmatic style guide, meets the expected accessibility requirements and is reusable in the company's current and future projects.

The thesis is in Estonian and contains 28 pages of text, 6 chapters, 23 figures, 5 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| Angular.IO | TypeScript raamistik kliendirakenduste loomiseks |
| API | *Application Programming Interface* – rakendusliides |
| ASCII | *American Standard Code for Information Interchange* – elektrooniliste tähemärkide kodeerimise standard |
| Bootstrap | JavaScript raamistik reageerimisvõimeliste veebisaitide loomiseks |
| CSS | *Cascading Style Sheets* – märgistuskeel, millega määratakse veebilehe kujundus |
| DOM | *Document Object Model* – dokumendi objektimudel |
| Github | Veebimajutusteenus tarkvaraarenduseks |
| HTML | *HyperText Markup Language* – märgendkeel veebilehtede loomiseks |
| JavaScript | Programmeerimiskeel veebilehtede skriptimiseks |
| JPEG | *Joint Photographic Experts Group* – digipiltide failivorming |
| JSX | *JavaScript XML* – Javascripti süntaksi laiendus |
| Lit | Raamistik veebikomponentide loomiseks |
| NPM | *Node Package Manager* – JavaScripti käituskeskkonna paketihaldur |
| NPM Trends | Veebisait NPM paketide allalaadimiste statistika kuvamiseks |
| Paoklahv | *„Escape“* klahv klaviatuuril |
| *Placeholder* | Atribuut, mis kirjeldab sisendvälja eeldatavat väärtust |
| PNG | *Portable Network Graphics* – digipiltide failivorming |
| Sass | *Syntactically Awesome Style Sheets* – stiililehe keel veebilehtede kujundamiseks |
| *Shadowroot* | DOM-alampuu juur, mis renderdatakse dokumendi DOM-i põhipuust eraldi |
| Sisestusklahv | *„Enter“* klahv klaviatuuril |
| Skate.js | Raamistik veebikomponentide loomiseks |
| Stencil | Raamistik veebikomponentide loomiseks |
| SVG | *Scalable Vector Craphics* – vektorpiltide failivorming |

| | |
|---|---|
| Tabeldusklahv | *„Tab“* klahv klaviatuuril |
| Tühikuklahv | *„Spacebar“* klahv klaviatuuril |
| TypeScript | JavaScripti ülemkogum, mis lisab staatilise tüübistamise |
| UTF-8 | *UCS Transformation Format 8* – tähemärkide kodeerimise standard, mis laiendab ASCII märgihulka, et kasutada 8-bitiseid koodipunkte |
| Vue.js | JavaScript raamistik kliendirakenduste loomiseks |
| W3C | *World Wide Web Consortium* – rahvusvaheline organisatsioon, mis loob veebistandardeid |
| WAI-ARIA | *Web Accessibility Initiative-Accessible Rich Internet Applications Suite* – spetsifikatsioon veebi ligipääsetavuse parandamiseks |
| WCAG | *Web Content Accessibility Guidelines* – veebisisu juurdepääsetavuse juhised |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Veebipõhised teenused on muutumas üha aktuaalsemaks. Need on klientide jaoks kiirem ja mugavam alternatiiv traditsioonilistele teenustele. Erandiks ei ole ka valimised: kui aastal 2005 jäi Eestis e-hääletajate osakaal alla 2%, siis 2019. aasta Riigikogu valimistel oli e-hääletajate osakaal juba 43,8% ja 2023. aasta Riigikogu valimistel 50,9%. Riigi Infosüsteemi Ameti andmetel peaks e-hääletajate osakaal moodustama lähiaastatel enamiku antud häältest [1] [2].

Üheks veebipõhiste e-valimisteenuste pakkujaks on Smartmatic-Cybernetica Centre of Excellence for Internet Voting OÜ (edaspidi SCCEIV), mis on tegevusalalt internetihääletamise kompetentsikeskus [3]. Ettevõte loob pidevalt uusi e-valimiste lahendusi klientidele globaalsel tasemel ning ühetaolise kasutajakogemuse (vaateelementide käitumise ja välimuse) saavutamine on oluline ettevõtte ära tundmiseks. Sellel eesmärgil kasutab ettevõte emafirma Smartmatic poolt loodud disainisüsteemi, et ühtlustada kasutajakogemust erinevate teenuste vahel.

Eelnevalt mainitud disainisüsteem on tänaseks aegunud. Selles sisalduvaid komponente on raske modifitseerida ning osad komponendid ei vasta täielikult ligipääsetavuse nõuetele. Smartmatic on lõplikult peatanud disainisüsteemi haldamise ja arendamise.

Uute arendusprojektide ja klientide erisoovide tõttu on aeg-ajalt tarvis kasutatavaid komponente kohandada, samas säilitades kasutajaliidese visuaalse ja kasutajakogemusliku sarnasuse Smartmatic disainisüsteemi komponentidega. Selleks on hulk komponente välja vahetatud arendajate endi kirjutatud kohandatud komponentidega. Kohandatud komponendid on kasutusel nii Angular kui ka Vue projektides, kirjutatud vastavates natiivsetes raamistikes ning korduvkasutatavad ainult antud projektide piires. Arendajate töö vähendamiseks peab SCCEIV vajalikuks luua ühtne taaskasutatav komponenditeek, et hoida kokku aega kohandatud komponentide loomisega iga uue projekti raames.

## 1.1 Töö eesmärk

Käesoleva lõputöö eesmärk on arendada kasutajaliidese komponentide teek, mille tulemus vastaks ettevõtte visuaalsele stiilile ning oleks korduvkasutatav käimasolevates ja tulevastes arendusprojektides. Komponenditeek annab arendajatele suuremat kontrolli komponentide välimuse ja funktsionaalsuse üle, kui seda vajalikuks peetakse.

Eesmärgi saavutamiseks on autor esitanud lahendusele järgnevad kriteeriumid:

- komponendid peavad vastama visuaalselt ja kasutajakogemuselt ettevõtte loodud stiilile;
- komponendid peavad vastama WCAG tase AA ligipääsetavuse nõuetele;
- komponendid peavad olema kliendipõhise tootearenduse korral hõlpsalt modifitseeritavad;
- komponenditeeki saab kasutada Angular.IO ja Vue.js raamistikus kirjutatud rakendustes.

## 1.2 Töö struktuur

Käesoleva töö järgnevates peatükkides tutvustatakse, mis on veebikomponentide loomisel kasutatavad tehnoloogiad, kasutajaliidese komponendid ja komponenditeek. Seejärel kaardistatakse komponentide loomisel vajalikud nõuded. Nõuded moodustuvad Smartmatic stiiliraamatu poolt paika pandud visuaalsetest nõuetest ning ligipääsetavuse standardite kriteeriumidest. Teoreetilise osa lõpus leitakse teegi loomiseks sobilik raamistik.

Töö praktilises osas luuakse nõuetele toetudes vastav komponenditeek. Esile tuuakse komponendid, mille loomise käigus ilmnes eripärasid või raskusi. Loodavatele komponentidele kirjutatakse ka ühiktestid. Valminud teek publitseeritakse NPM keskkonnas ning viimaks luuakse minimalistlikud näidisrakendused Angular.IO ja Vue.js raamistikes.

# 2 Taust

Käesolevas peatükis kirjeldatakse, mis on veebikomponendid (ja veebikomponentide all paiknevad tehnoloogiad), kasutajaliidese komponendid ning komponenditeegid. Lisaks tehakse ülevaade ligipääsetavusest internetis.

## 2.1 Veebikomponendid

Koodi korduvkasutamine on hea tava, mida tuleks teha nii palju kui võimalik, et hoida koodi puhtana ja muuta see skaleeritavaks. Veebikomponendid on üks lähenemine koodi taaskasutamiseks. Tegu on kohandatud HTML-siltidega, mis on defineeritud JavaScriptis, kapseldatud funktsionaalsusega ning neid saab kasutada mistahes veebiprojektis. Veebikomponendid põhinevad kolme tehnoloogia kombinatsioonil, mida saab koos kasutada mitmekülgse HTML-sildi loomiseks: kohandatud elemendid, varju DOM ja mallid [4].

### 2.1.1 Kohandatud elemendid

Kohandatud element on element, mille konstruktor ja prototüüp on määratletud autori, mitte kasutajaagendi poolt. Kohandatud elemendid pakuvad autoritele võimalust luua täielikult funktsionaalsed DOM-elemendid. Nende määratlemisel saavad autorid anda parserile teada, kuidas elementi õigesti konstrueerida ja kuidas selle klassi elemendid peaksid muutustele reageerima [5].

### 2.1.2 Varju DOM

Varju DOM (*Shadow DOM*) on DOM-puu, mille elemendid ja stiilid on põhilisest DOM-ist isoleeritud. See on üks kapseldamise vormidest. Kasutades varju DOM-i koodi isoleerimiseks, mis mõjutab ainult teatud elemente, saab vältida kokkupõrkeid ülejäänud koodiga. See erineb tavalisest DOM-ist selle poolest, et tema stiil ei ole mõjutatud põhilisest CSS-ist [6].

### 2.1.3 HTML mallid

HTML element <template> (mall) on mehhanism HTML-i hoidmiseks, mida ei ole kavas kohe lehe laadimisel renderdada, vaid mida võib hiljem käivitamise ajal JavaScripti abil instantseerida. Mall on justkui sisufragment, mis salvestatakse dokumendis hilisemaks

kasutamiseks. Kuigi parser töötleb <template> elemendi sisu lehe laadimise ajal, teeb ta seda ainult selleks, et tagada sisu kehtivus. Chromiumil põhinevates brauserites toetab mallielement ka *shadowroot* atribuuti, mis on osa varju DOM-ist. Mallielementi saab kasutada veebikomponendi varju DOM-i täitmiseks [7].

## 2.2 Kasutajaliidese komponent ja komponenditeek

Kasutajaliidese komponent on korduvkasutatav programmi ehitusplokk või element, mida saab kombineerida teiste komponentidega, et moodustada terviklik kasutajaliides. Komponentideks on näiteks sisendväljad, ripploendid, nupud, sildid ja palju muud. Kasutajaliidese komponendid pakuvad kasutajale interaktiivsust, kui ta liigub läbi rakenduse või veebisaidi [8].

Komponenditeek on korduvkasutatavate kasutajaliidese elementide (komponentide) kogum, näiteks nuppude kujundus, kirjatüüpide valik ning muud tüpograafilised ja visuaalsed elemendid. Teegi eesmärk on hoida kasutajaliidese komponente ühes kohas, mida saab seejärel kasutada erinevates projektides. See aitab tagada toodete ja teenuste kasutajaliidese järjepidevuse ning säästab arendamisel aega. Samuti on nende hooldus ja uue funktsionaalsuse lisamine lihtsam, kuna see kajastub igas projektis, kus komponente kasutatakse. Komponenditeek võib koosneda ühest failist või kaustast. Teegis sisalduvate elementide ja kujundusdetailide arv sõltub organisatsioonist ja projektide iseloomust.

Komponenditeegid on osa disainisüsteemidest. Teegid sisaldavad ainult kasutajaliidese elemente, mil disainisüsteemid sisaldavad ka poliitikaid ja protseduure ning dikteerivad kogu disainimise protsessi [9].

## 2.3 Veebi ligipääsetavus

Veebilehe ligipääsetavus kujutab endast võimalust kõigil inimestel veebilehe poolt pakutavaid teenuseid tarbida, olenemata sellest, millised tingimused neil on. Teisisõnu tähendab see, et puuetega inimesed saavad veebi mugavamalt kasutada ja nautida.

### 2.3.1 WCAG

Organisatsioon W3C on loonud veebisisu ligipääsetavuse suunised *Web Content Accessibility Guidelines* (edaspidi WCAG), mida peetakse ligipääsetavuse tehniliseks

standardiks. WCAG suuniste järgimine on parim ja lihtsaim viis muuta veebisait kõikidele klientidele kasutatavaks.

WCAG on jagatud kolmeks tasemeks: A, AA ja AAA. Tasemed määravad kindlaks eri olukordade ja nõuete jaoks soovitatava vastavusastme. A on madalaim ning vastab minimaalsele nõutavale standardile. AA on keskmine ning enamikul juhtudel soovitatav tase. AAA on kõrgeim võimalik hinne, kuid on keeruline teha veebilehte, mis selle taseme saavutab. Iga WCAG tase järgneb eelmisele, mis tähendab, et kui veebisait vastab AA-tasemele, siis peab see vastama ka A-tasemele [10]. Käesolevas töös loodavad komponendid peavad vastama WCAG AA-tasemele.

### 2.3.2 WAI-ARIA

Üks olulisemaid ligipääsetavuse tehnoloogiaid on W3C alamrühma *Web Accessibility Initiative* poolt kirjutatud standardite ja materjalide komplekt *Accessible Rich Internet Applications Suite* (edaspidi WAI-ARIA). WAI-ARIA on spetsifikatsioon elementidele rakendamiseks, et lisada täiendavat semantikat ja parandada ligipääsetavust seal, kus see puudub. Järgnevalt on välja toodud spetsifikatsioonis määratletud kolm peamist omadust.

- Rollid: määratlevad, mis element on või mida see teeb. Kasutatakse peamiselt struktuurielementide semantilise väärtuse parandamiseks, näiteks *role="navigation"*.

- Omadused: määratlevad elementide omadused, mida saab kasutada lisatähenduse andmiseks. Näiteks täpsustab *aria-required="true"*, et sisend peab olema täidetud.

- Olekud: eriomadused, mis määravad elementide praegused tingimused. Näiteks täpsustab *aria-disabled="true"*, et sisend on hetkel keelatud. Olekud erinevad omadustest selle poolest, et omadused ei muutu rakenduse elutsükli jooksul, kuid olekud võivad muutuda.

WAI-ARIA spetsifikatsioone tuleks kasutada ainult siis, kui see on vajalik. Võimaluse korral tuleks kasutada natiivseid HTML funktsioone, kus vastav toetus on vaikimisi olemas [11].

# 3 Analüüs

Käesolevas peatükis analüüsitakse loodavate komponentide nõudeid ning leitakse praktilise osa tegemiseks sobilik raamistik.

## 3.1 Komponentide nõuded

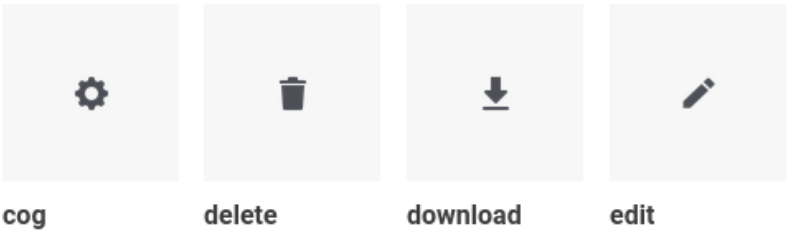
Järgnevalt kirjeldatakse, millised komponendid lõputöö raames luuakse ning millised on neile esitatud nõuded. Loodavaid komponente on kokku kaheksa, millest seitse on interaktiivsed. Kaardistatavad nõuded hõlmavad nii visuaalseid kui ka ligipääsetavusega seotud kriteeriume. Ligipääsetavusel lähtutakse (võimaluse korral) natiivsetest HTML elementidest, kus nõuded on vaikimisi sisse ehitatud.

### 3.1.1 Ikoon

Ikoon on kujutis või sümbol. Kasutatavad ikoonid pärinevad Smartmatic stiiliraamatu ikonograafia kogumist ning on .svg failitüüpi. SVG peamine eelis teiste pildifailide tüüpide (JPEG, PNG) ees on tema sõltumatus resolutsioonist. See tähendab, et neid saab skaleerida samamoodi nagu kõiki teisi elemente ning pilt näeb välja terav igal ekraanil. SVG omab ka teisi eeliseid [12]:

- Brauser käsitleb SVG-sid eraldi dokumendina ja paigutab selle lehekülje tavalise DOM-i sisse. See võimaldab CSS-il ja JavaScriptil suhelda SVG-siseste elementidega, tänu millele saab elemente animeerida ja stiliseerida.
- Tänu SVG vektorilisele olemusele on nende optimeeritud faili suurus väike võrreldes peaaegu kõigi teiste pildifailide tüüpidega.

Ikooni tüüpvärv on stiiliraamatust pärinev #3D3D3D. Kui ikoon kuulub teksti juurde, on tema värvus sama, mis teksti oma, kui ei ole täpsustatud teisiti. Joonisel 1 on kujutatud ikoonid *cog*, *delete*, *download* ja *edit*.




Joonis 1. Ikoonid *cog*, *delete*, *download* ja *edit*.

Ikoonid peavad olema arusaadavad igale veebilehe külastajale. Vältimaks ebaselgust ikooni tähenduse ümber, peab ikoonidele olema lisatud tekstisilt (ingl.k. *label*), kui see vastavas kontekstis vajalik on [13]. Ikoonid peavad olema ligipääsetavad ka ekraanilugeritele: kui ikoon on semantiline (näiteks toimib nupuna), peab välimisele (nupu)elemendile olema määratud *aria-label* atribuut ikooni tegevust või tähendust selgitava kirjeldusega. Dekoratiivsetel ikoonidel peab olema atribuut *aria-hidden* [14].

### 3.1.2 Nupp

Nupp (ingl.k. *button*) on komponent, millel klikates käivitatakse määratud tegevus. Nuppe on kolmes variatsioonis: *solid* (primaarne), *outline* (sekundaarne) ja *ghost*. *Solid* variatsioon jaguneb omakorda neljaks alamtüübiks: *primary*, *default*, *warning* ja *alert*. Teised variatsioonide alamtüübid on *primary* ja *default*. Joonisel 2 on kujutatud nupud kolmes variatsioonis alamtüübis *primary*.


Joonis 2. Nupud *solid*, *outline* ja *ghost* alamtüübis *primary*.

Nupu tekst lisatakse läbi mallielemendi ja see paigutatakse nupu keskele. Tekstile on võimalik kõrvale lisada ikoon, mis võib paikneda tekstist vasakul või paremal. Nuppude värvid pärinevad Smartmatic stiiliraamatust ning muutuvad vastavalt nupu olekule (*enable*, *hover*, *active*, *focus* ja *disabled*).

Tabelites 1 ja 2 on välja toodud nuppude tausta ja teksti värvid variatsioonide ja olekute põhjal. Kui nuppudel on ääris (*border*), on ka need tabelis määratletud.

Tabel 1. Nupu *solid* variatsiooni värvid olekutes *enable, hover, active, focus* ja *disabled*.

| | **Primary** | **Default** | **Warning** | **Alert** |
|---|---|---|---|---|
| **Enable** | Taust: #4F23C9<br>Tekst: #FFFFFF | Taust: #F5F5F5<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #FFAA00<br>Tekst: #292929 | Taust: #8E1F1F<br>Tekst: #FFFFFF |
| **Hover** | Taust: #663DD4<br>Tekst: #FFFFFF | Taust: #E0E0E0<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #FFCC66<br>Tekst: #292929 | Taust: #B12626<br>Tekst: #FFFFFF |
| **Active** | Taust: #441EAB<br>Tekst: #FFFFFF | Taust: #CCCCCC<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #FFAA00<br>Tekst: #292929 | Taust: #6D1717<br>Tekst: #FFFFFF |
| **Focus** | Taust: #4F23C9<br>Tekst: #FFFFFF | Taust: #F5F5F5<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #FFAA00<br>Tekst: #292929 | Taust: #8E1F1F<br>Tekst: #FFFFFF |
| **Disabled** | Taust: #F5F5F5<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 | Taust: #F5F5F5<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 | Taust: #F5F5F5<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 | Taust: #F5F5F5<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 |

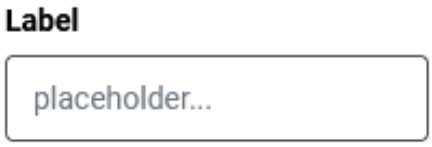
Tabel 2. Nupu *outline* ja *ghost* variatsioonide värvid olekutes *enable, hover, active, focus* ja *disabled*.

| | **Outline primary** | **Outline default** | **Ghost primary** | **Ghost default** |
|---|---|---|---|---|
| **Enable** | Taust: #FFFFFF<br>Tekst: #4F23C9<br>Ääris: #4F23C9 | Taust: #FFFFFF<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #FFFFFF<br>Tekst: #4F23C9 | Taust: #FFFFFF<br>Tekst: #3D3D3D |
| **Hover** | Taust: #EEE9FB<br>Tekst: #4F23C9<br>Ääris: #4F23C9 | Taust: #E0E0E0<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #EEE9FB<br>Tekst: #4F23C9 | Taust: #E0E0E0<br>Tekst: #3D3D3D |
| **Active** | Taust: #CBBDF2<br>Tekst: #4F23C9<br>Ääris: #4F23C9 | Taust: #CCCCCC<br>Tekst: #3D3D3D<br>Ääris: #B8B8B8 | Taust: #CBBDF2<br>Tekst: #4F23C9 | Taust: #CCCCCC<br>Tekst: #3D3D3D |
| **Focus** | Taust: #FFFFFF<br>Tekst: #4F23C9<br>Ääris: #4F23C9 | Taust: #FFFFFF<br>Tekst: #3D3D3D | Taust: #FFFFFF<br>Tekst: #4F23C9 | Taust: #FFFFFF<br>Tekst: #3D3D3D |
| **Disabled** | Taust: #FFFFFF<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 | Taust: #FFFFFF<br>Tekst: #8F8F8F<br>Ääris: #E0E0E0 | Taust: #FFFFFF<br>Tekst: #8F8F8F | Taust: #FFFFFF<br>Tekst: #8F8F8F |

Nupp on fokusseeritav. Nupp opereerib hiirega nupule klikates ning klaviatuuril nii sisestusklahvile või tühikuklahvile vajutades [15]. Kui nupp käitub lingina, peab nupuelemendil olema atribuut *role="link"*, et ekraanilugerid seda semantiliselt korrektselt tuvastaks.

### 3.1.3 Sisend

Sisendi (ingl.k. *input*) komponenti kasutatakse kasutajatelt andmete vastu võtmiseks. See, kuidas sisendelement töötab, erineb oluliselt sõltuvalt selle *type* atribuudi väärtusest [16]. Antud lõputöö raames käsitletakse sisendi komponenti vaikimisi tüübi „tekstina". Sisend koosneb sisendi elemendist ja tekstisildist. Täitmata sisendil võib olla *placeholder* tekst, kui on vastavalt täpsustatud. Joonisel 3 on kujutatud sisendkast *placeholder* tekstiga.




Joonis 3. Sisend *placeholder* tekstiga.

Sisend on fokusseeritav. Sisendile on antud atribuut *type="text"*. Komponendil peab olema tekstisilt, mis peab viitama sisendelemendi ID-le [17].

### 3.1.4 Märkeruut

Märkeruut (ingl.k. *checkbox*) on vaikimisi kujul kast, mida saab aktiveerida. Märkeruute kasutatakse ühe või mitme võimaluse valimiseks piiratud arvu valikute hulgast. Märkeruut koosneb sisendist ja tekstisildist [18]. Valimata märkeruudu korral kuvatakse tühja kasti, mille piirjoon on stiiliraamatust pärinev #4F23C9. Valitud märkeruudu kast on täidetud sama värviga kui piirjoon ning kasti keskel kuvatakse valget linnukese ikooni. Joonisel 4 on kujutatud märkeruut valimata ja valitud olekus.


Joonis 4. Märkeruut valimata ja valitud olekus.

Märkeruut on fokusseeritav. Komponent peab olema *input* atribuudiga *type=“checkbox“*, et tagada selle ligipääsetavus. Märkeruutu saab valida hiirega sisendile ja tekstisildile klikates ning klaviatuuril tühikuklahvile vajutades. Tekstisilt peab viitama sisendelemendi ID-le [18].

### 3.1.5 Raadio märkeruut

Raadio märkeruut või raadionupp (ingl.k. *radio button*) on vaikimisi kujul ring, mida saab aktiveerida. Raadio märkeruut on ligilähedane tavalisele märkeruudule, kuid seda kasutatakse ainult ühe võimaluse valimiseks piiratud arvu valikute hulgast. Valiku muutmisel deaktiveeritakse eelnevalt valitud raadionupp. Sarnaselt märkeruudule koosneb raadio märkeruut sisendist ja tekstisildist. Valimata raadio märkeruudu korral kuvatakse tühja ringi piirjoonega #3D3D3D. Valitud raadio märkeruut sisaldab väiksemat täidetud ringi ringjoone sees ning nii ring kui ka ringjoon on värvis #441EAB. Joonisel 5 on kujutatud raadio märkeruut valimata ja valitud olekus.

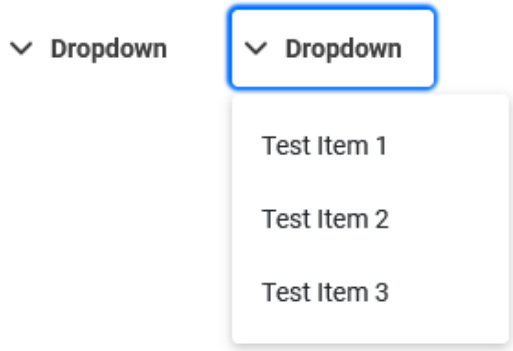

Joonis 5. Raadio märkeruut valimata ja valitud olekus.

Raadio märkeruut on fokusseeritav. Komponent peab olema *input* atribuudiga *type=“radio“*. Raadio märkeruudud kuuluvad rühma. Igal märkeruudul on *name* atribuut vastava rühma nimetusega. Tänu sellele tühistakse mis tahes uue raadionupu valimisel parajasti valitud raadionupp samas rühmas. Raadionupul on atribuut *value*, mis sisaldab raadionupu väärtust. Seda kasutatakse selleks, et tuvastada, milline raadionupp on rühmast valitud [19]. Raadio märkeruutu saab valida hiirega sisendile ja tekstisildile klikates ning klaviatuuril tühikuklahvile vajutades. Tekstisilt peab viitama sisendelemendi ID-le.

### 3.1.6 Rippmenüü

Rippmenüü on komponent, mis esitab valikute loetelu. Valikuid kasutatakse tegevuste sooritamiseks, filtreerimiseks või olemasoleva sisu sorteerimiseks. Rippmenüü on *select* komponendi stiliseeritud versioon [20]. Komponent koosneb rippmenüü avavast nupust, tesktisildist ja vaikimisi peidetud valikutest. Nupule klikates kuvatakse valikud. Nupp on

vaikimisi variatsioonis *ghost* alamtüübis *default*. Joonisel 6 on kujutatud ripmenüü avamata ja avatud olekus.




Joonis 6. Rippmenüü avamata ja avatud olekus.

Rippmenüü on fokusseeritav. Valiku saab sooritada hiirega valikule klikates ja klaviatuuril sisetusklahvile või tühikuklahvile vajutades. Klaviatuuriga saab valikute vahel liikuda „üles“ ja „alla“ nooltega.

### 3.1.7 Kontekstuaalne kast

Kontekstuaalne kast (ingl.k. *contextual box*) on kastikujuline komponent, mis edastab kasutajale teavet. Kontekstuaalsel kastil on neli variatsiooni: *success*, *warning*, *info* ja *error*. Komponent koosneb pealkirjast ja sisust. Pealkirjast vasakul paikneb variatsioonil põhinev ikoon. Sisu tekst lisatakse läbi mallielemendi. Kontekstuaalne kast on suletav, kui on vastavalt täpsustaud. Suletaval kastil on atribuut *dismissable=“true“*, tänu millele ilmub pealkirja paremasse äärde risti ikoon. Vaikimisi on *dismissable* väärtus *false*. Joonisel 7 on kujutatud kontekstuaalne kast igas variatsioonis.


Joonis 7. Kontekstuaalne kast *success*, *warning*, *info* ja *error* variatsioonis.

Kontekstuaalne kast on ümbritsetud äärisega #E0E0E0 ja varjuga #333333, läbipaistvusega 40%. Kontekstuaalse kasti pealkirja värvid sõltuvad variatsioonist. Tabelis 3 on välja toodud pealkirja tausta, teksti ja kaasneva ikooni värvid variatsiooni põhjal.

Tabel 3. Kontekstuaalse kasti pealkirja tausta, teksti ja ikooni värvid igas variatsioonis.

| | ***Success*** | ***Warning*** | ***Info*** | ***Error*** |
|---|---|---|---|---|
| **Taust** | #D9F2DD | #FFF7E5 | #F6D5D5 | #CCE2FF |
| **Tekst** | #17451F | #855800 | #4B1010 | #00285C |
| **Ikoon** | #2B823A | #AD7400 | #004BAD | #8E1F1F |

Kontekstuaalne kast ei ole fokusseeritav. Komponendipõhised ligipääsetavuse nõuded puuduvad.

### 3.1.8 Staatusmärgis

Staatusmärgis (ingl.k. *badge*, *tag*) on komponent, mida rakendatakse märgistamist vajavatel esemetel, kasutades neid kirjeldavaid märksõnu. Staatusmärgisel on viis variatsiooni: *success*, *alert*, *warning*, *neutral*, *default*. Lisaks on iga variatsiooni võimalik kasutada *subtle* versioonis. Elemendi vaikimisi variatsioon on *success*. Staatusmärgise sisu lisatakse läbi mallielemendi. Joonisel 8 on kujutatud staatusmärgis tava- ja *subtle* versioonis igas variatsioonis.


Joonis 8. Staatusmärgis *success*, *alert*, *warning*, *neutral* ja *default* variatsioonis.

Tabelis 4 on näidatud staatusmärgisel kasutatavad värvid vaikimisi ja *subtle* versioonides.

Tabel 4. Staatusmärgise variatsioonide värvid vaikimisi ja *subtle* versioonis.

| | ***Success*** | ***Alert*** | ***Warning*** | ***Neutral*** | ***Default*** |
|---|---|---|---|---|---|
| **Vaikimisi** | Taust: #D9F2DD<br>Tekst: #17451F<br>Ääris: #B3E5BB | Taust: #F6D5D5<br>Tekst: #4B1010<br>Ääris: #EDABAB | Taust: #FFF7E5<br>Tekst: #5C3D00<br>Ääris: #FFDD99 | Taust: #CCE2FF<br>Tekst: #00285C<br>Ääris: #99C5FF | Taust: #F5F5F5<br>Tekst: #3D3D3D<br>Ääris: #E0E0E0 |
| ***Subtle*** | Tekst: #17451F<br>Ring: #2B823A | Tekst: #4B1010<br>Ring: #8E1F1F | Tekst: #5C3D00<br>Ring: #AD7400 | Tekst: #00285C<br>Ring: #004BAD | Tekst: #3D3D3D<br>Ring: #666666 |

Staatusmärgis ei ole fokusseeritav. Komponendipõhised ligipääsetavuse nõuded puuduvad.

### 3.1.9 Üldine ligipääsetavus

Kõikidele komponentidele kehtivad lisaks üldised ligipääsetavuse nõuded. Nõuded pärinevad W3C organisatsiooni kodulehelt [21].

- Teksti ja kujutiste visuaalne esitus peab olema kontrastsusega vähemalt 4,5:1. Suurte tekstide ja kujutiste suhe peab olema vähemalt 3:1 (§1.4.3 *Contrast Minimum*).

- Igal klaviatuuriga juhitaval (fokusseeritaval) elemendil on fookuse indikaator nähtav (§2.4.7 *Focus Visible*). Indikaator hõlmab komponenti või selle alamkomponenti ning fokusseeritud ja fokusseerimata oleku kontrastsuse suhe on vähemalt 3:1 (§2.4.11 *Focus Appearance*). Lõputöös kasutatakse fookuse indikatoorina stiiliraamatust pärinevat kontuuri värvusega #006FFF.

## 3.2 Raamistiku valik

Käesolevas peatükis valitakse komponenditeegi loomiseks raamistik. Esialgne valik moodustub raamistikest, mis toetavad kohandatud elemente, varju DOM-i ja HTML malle [22]. Sobiva raamistiku valimiseks on paika pandud järgnevad kriteeriumid:

- Dokumentatsioon: dokumentatsioon on põhjalik ja mugavalt kasutatav, sisaldab paigaldamise ja komponentide loomise protsessi, näited ja õpetused on kättesaadavad.

- Populaarsus: raamistik on ajakohane, tuntud ja arendajate hulgas hinnatud.

- Ühilduvus ettevõttega: raamistik töötab ettevõtte praeguste ja tulevikus kasutatavate tehnoloogiatega.

Raamistiku populaarsuse määratlemisel võetakse arvesse kasutajaskonna suurust ja rahuolu. Selleks võrreldakse raamistike Github *stars* hinnangut ja sõltuvate repositooriumite arvukust, kuna läbi selle peegeldub kasutajate rahulolu vastava raamistiku tööga. Lisaks vaadatakse raamistike allalaadimiste arvu NPM registris viimaste aastate jooksul.

### 3.2.1 Lit

Lit on avatud lähtekoodiga JavaScript raamistik kiirete ja kergete veebikomponentide loomiseks. Lit'i keskmes on LitElement baasklass, mis on mugav laiendus algupärasele HTMLElemendile. Raamistik kasutab lit-html'i varju DOM-i renderdamiseks ning lisab API omaduste ja atribuutide haldamiseks. Omadusi jälgitakse vaikimisi ja elemendid uuenevad asünkroonselt, kui nende omadused muutuvad. Stiilid on vaikimisi piiritletud, mis hoiab CSS selektoreid lihtsana ja tagab, et komponendi stiil ei põrkaks kokku ümbritseva sisuga. Iga Lit komponent on standardne veebikomponent. Veebikomponente saab kasutada mis tahes HTML keskkonnas, mis tahes raamistikuga [23].

### 3.2.2 Skate.js

Skate on funktsionaalne reaktiivne abstraktsioon pakettide kogumina üle veebikomponentide standardite, mis võimaldab kirjutada väikseid, kiireid ja skaleeritavaid veebikomponente. Skate abil saab renderdada komponente, kasutades mis tahes raamistikku ning sisaldab täielikku TypeScript tuge [24].

### 3.2.3 Stencil

Stencil on kompilaator, mis genereerib veebikomponente ja ühendab endas kõige populaarsemate raamistike parimad kontseptsioonid. Stencil kasutab TypeScripti, JSX-i ja CSS-i, et luua standarditel põhinevaid veebikomponente, mida saab kasutada kvaliteetsete komponenditeekide, disainisüsteemide ja rakenduste loomiseks. Veebikomponendid töötavad koos populaarsete raamistikega. Lisaks saab genereerida raamistikule omaseid komponente. Stencil täiendavaid API-sid, asünkroonset renderdamist ja eelrenderdamist [25].

### 3.2.4 Raamistike võrdlus

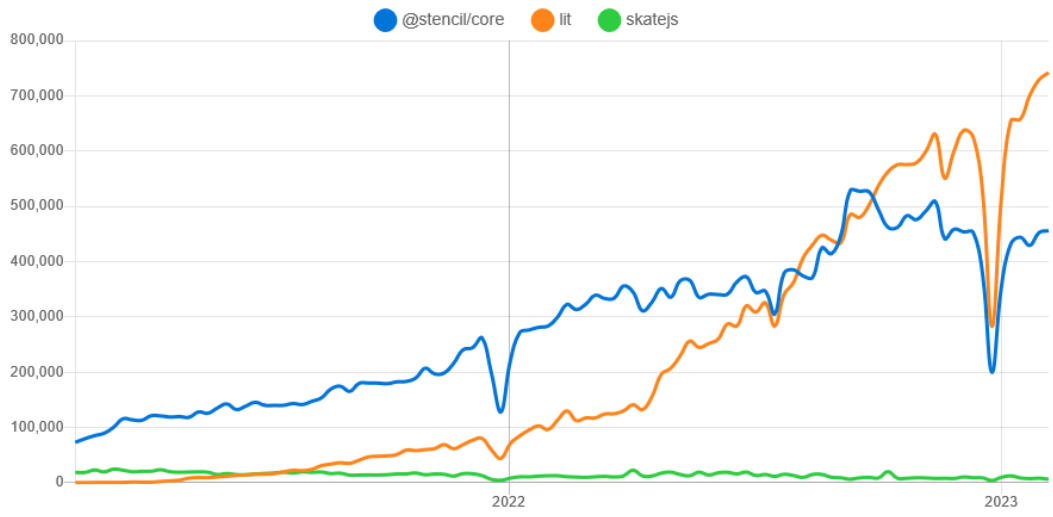
Stencil ja Lit'i dokumentatsioonid on väga põhjalikud. Mõlemad sisaldavad paigaldamise ja komponentide loomise juhendeid koos näidete ja õpetustega. Kuigi Skate dokumentatsioonis on samuti välja toodud juhendid paigaldamiseks ning komponentide loomise protsess, on näiteid ja õpetusi tunduvalt vähem. Võrreldavatest raamistikest on kõige küpsem Lit, kuid ka Stencil on järjepidevas arenduses ning saab tihedalt tarkvaravärskendusi. Skate.js arendus on peatunud.

Tabelis 5 on esile toodud raamistike *stars* arv, raamistikust sõltuvate repositooriumite arv ja viimase reliisi kuupäev seisuga 12.02.2023 [26] [27] [28]. Lit ja Stencil raamistike akumuleeritud *stars* arv on Github andmete põhjal üsnagi tasavägine ning Skate jääb eelmainitutest tunduvalt alla. Suurim erinevus väljendub raamistike sõltuvuste graafikus, mille andmetel sõltub Stencil'ist ligikaudu 60 000, Lit'ist 21 000 ning Skate.js'ist 17 repositooriumi. Viimane reliis tehti Stencil ja Lit'il käesoleval aastal ning Skate.js'il 5 aastat tagasi.

Tabel 5. Github *stars*, *used by* ja *latest release date* andmed seisuga 12.02.2023.

| | ***Stars*** | ***Used by*** | ***Latest release date*** |
|---|---|---|---|
| **Lit** | 13.7K | 21K | 18.01.2023 |
| **Skate.js** | 3.3K | 17 | 20.02.2017 |
| **Stencil** | 11.4K | 59.9K | 25.01.2023 |

Joonisel 9 on välja toodud raamistike allalaadimiste arv NPM registris NPM Trends andmetel viimase kahe aasta jooksul (andmed seisuga 16.02.2023) [29]. Joonisel on näha, et Lit ja Stencil allalaadimiste arv järgib sarnast mustrit. Stencil'i allalaadimiste arvukus on püsinud esikohal kuni oktoobrini 2022, millest alates on esikoha saavutanud Lit ning püsib seal tänaseni. Skate.js allalaadimiste arv on märkimisväärselt madalam.




Joonis 9. Võrreldavate raamistike allalaadimiste arv NPM registris viimase kahe aasta vältel [29].

Kuna tegu on veebikomponentidega, mis on kasutatavad mis tahes raamistikus, ühilduvad kõik võrreldavad raamistikud ettevõttes kasutatavate tehnoloogiatega.

Eelnevalt kogutud info põhjal arvestatakse Skate.js raamistiku valikust välja. Skate.js'i dokumentatsioon on vähene, arendus on seiskunud ning kasutajaskond on mõjuvalt väiksem teiste valikute kasutajaskondadest. Kuna Lit ja Stencil on püstitatud kriteeriumide põhjal sarnasel positsioonil, siis toetutakse lõppvaliku tegemisel ka töö autori subjektiivsele arvamusele. Töö autorile on meelepärasem Stencil'i süntaks tänu varasemale kokkupuutele Stencil raamistikuga. Samuti on Stencil'is kirjutatud Smartmatic disainisüsteem, mis võib lihtsustada komponentide välja vahetamist käimasolevates projektides. Kogu eelmainitu põhjal otsustati antud lõputöö kirjutada Stencil raamistikus.

# 4 Arenduse realisatsioon

Käesolevas peatükis räägitakse komponenditeegi arendusprotsessist. Tutvustatakse projekti initsialiseerimist ja konfigureerimist. Seejärel räägitakse komponentide loomisest ning viimasena valminud teegi publitseerimisest.

## 4.1 Projekti initsialiseerimine ja konfigureerimine

Uus Stencil projektipõhi initsialiseeriti käsureal käsuga *npm init stencil*. Käsurida kuvas erinevad valikud, mis tüüpi projekti soovitakse luua (Joonis 10).

```
? Select a starter project.

Starters marked as [community] are developed by the Stencil Community,
rather than Ionic. For more information on the Stencil Community, please see
https://github.com/stencil-community » - Use arrow-keys. Return to submit.
>   component   Collection of web components that can be used anywhere
    app         Minimal starter for building a Stencil app or website
    ionic-pwa   Ionic PWA starter with tabs layout and routes
```

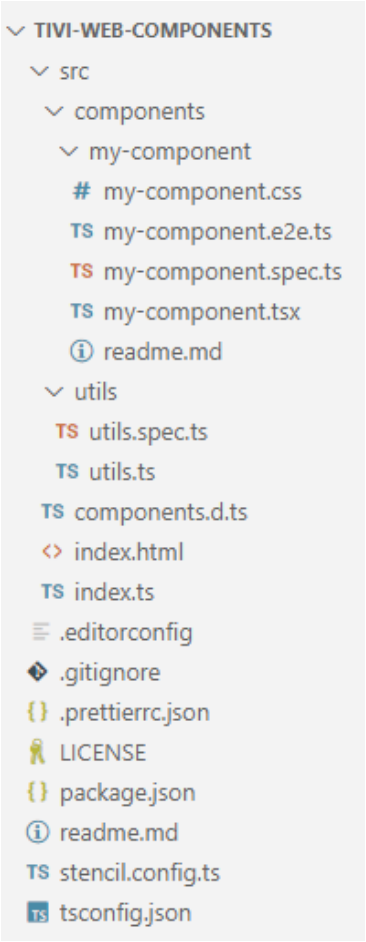
Joonis 10. Stencil projekti initsialiseerimine käsureal.

Valides võimaluse „*component*“, küsiti kasutajalt loodava projekti nime (Joonis 11). Projektile nime määramisel lähtuti kahest põhimõttest: nimi peab olema iseennast kirjeldav ning loodava paketi nimi ei tohi olla hõivatud NPM registris. Projekti nimeks valiti *tivi-web-components*.

```
https://github.com/stencil-community » component
Collection of web components that can be used anywhere
? Project name »
```

Joonis 11. Stencil projektile nime määramine.

Pärast nime kinnitamist initsialiseeris Stencil uue projekti. Projekt sisaldas komponendi näidet *my-component* ning abivahendeid *utils*, mida näidiskomponent rakendab. Joonisel 12 on esitatud loodud projekti esialgne struktuur.




Joonis 12. Stencil projekti esialgne struktuur.

Enne komponentide looma hakkamist oli vaja projekti paigaldada Sass ja Bootstrap. Samuti tuli konfigureerida globaalsete stiilide kasutamine.

### 4.1.1 Sass

Sass on CSS-i alamkogum, mida kasutatakse kliendipoolse kujunduse jaoks. See on sisuliselt CSS-i täiustatud versioon, mis pakub puhtamat ja paremini hallatavat kujundust ning olulist funktsionaalsuse parandamist. Pakutav funktsionaalsus on näiteks muutujad, operaatorid ja funktsioonid [30].

Vaikimisi ei ole Stencil'il Sass toetust. See lisati *stencil.config.ts* failis pluginate massiivi abil. Pluginasse sisestati konfiguratsioon *injectGlobalPaths* komponendikausta teekonnaga, tänu millele lisatakse Sassi import automaatselt kõikide komponentide stiilifaili.

### 4.1.2 Bootstrap

Bootstrap on avatud lähtekoodiga raamistik veebisaitide ja veebirakenduste loomiseks. See on kõige populaarsem HTML-, CSS- ja JavaScript raamistik reageerivate projektide arendamiseks. Bootstrap-iga ehitatud veebisaidid skaleeruvad automaatselt erinevate seadmete vahel [31]. Bootstrapi sõltuvus lisati käsuga *npm install bootstrap*.

### 4.1.3 Globaalsed stiilid ja stiilimuutujad

Globaalsed stiilid on stiilid, mis on saadaval kõikidele projektis olevatele komponentidele. Globaalsete stiilide rakendamiseks loodi globaalne stiililеht, mis asetati kausta nimega *scss*. Stiililehte imporditi Bootstrap ja lisati komponentide kirjatüüp. Komponentidele rakendatavaks kirjatüübiks on Roboto. Lisaks globaalsele stiilifailile loodi samasse kausta *variables.scss*, kus täpsustatati stiilimuutujad komponentide stiilifailides kasutamiseks. Stiilimuutujad aitavad koondada komponentides kasutatavad värvid (ja muud stiilid) ühte faili, tänu millele on neid lihtsam vajadusel välja vahetada ning erinevate komponentide vahel korduvkasutada. Stiilimuutujate fail imporditi globaalsesse stiilifaili.

Selleks, et globaalsed stiilid oleksid kättesaadavad kõigile projekti komponentidele, lisati *stencil.config.ts* faili seadistus *globalStyle*, mis võtab vastu stiililehe teekonna.

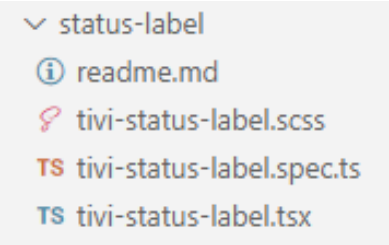
## 4.2 Komponentide loomine

Komponentide loomisel jälgiti, et komponentide sildid oleks järjepidevad. Komponentide siltide nimetamisel tuleb järgida ka etteantud reegleid [32]:

- nimi ei tohi sisaldada ühtegi suurt ASCII tähemärki;
- lubatud on ainult UTF-8 märgid;
- nimi peab sisaldama mõttekriipsu, mis aitab brauseril eristada kohandatud elemente tavalistest elementidest;
- komponendi silt ei tohi olla isesulguv.

Selleks, et komponendi kasutuse eesmärk oleks ühiselt arusaadav, määrati siltide nimeks vastava natiivse elemendi nimi (olemasolul) eesliitega *tivi*, näiteks *tivi-button*.

Iga komponent paigutati eraldi kausta, mis sisaldab nelja faili: komponendi dokumentatsioon (*readme.md*), stiilifail (*.scss*), testifail (*.spec.ts*) ja TypeScript fail JSX süntaksis (*.tsx*). Stiili-, testi- ja TypeScript failid loodi käsitsi ning dokumentatsiooni faili koostas automaatselt Stencil. Joonisel 13 on kujutatud näide staatusmärgise komponendi struktuurist.




Joonis 13. Näide: staatusmärgise komponendi struktuur.

### 4.2.1 Komponentide varjuosad

Kuna komponendid renderdatakse varju DOM-is, ei ole väliselt võimalik komponendi stiile muuta. Selleks, et komponendid oleks modifitseeritavad, kasutatakse CSS varjuosi (ingl.k. *shadow parts*). Varjuosad võimaldavad stiilida varju DOM-is asetsevaid elemente väljaspool seda varjundipuud. Selleks tuleb osa paljastada, lisades elemendile atribuudi *part* ning seejärel saab seda stiliseerida selektoriga *::part* [33]. Joonisel 14 on välja toodud näide modifitseeritud staatusmärgise CSS deklaratsioonist, kus kasutatakse varjuosa *badge*.

```
.custom-status-label::part(badge) {
  color: white;
  background-color: blue;
  border: 1px solid darkblue;
}
```

Joonis 14. Näide: modifitseeritud staatusmärgise CSS deklaratsioon varjuosaga *badge*.

### 4.2.2 Ikooni loomine

Kõik ikoonid pandi kausta *assets/svg/*. Ikoonide küsimiseks loodi abimeetod *getSvgContent(url)*, mis võtab sisendina ikooni teekonna ning tagastab sõnena vastava ikooni SVG faili sisu (Joonis 15). Juhul, kui küsitud teekonda ei leita, tagastatakse viga sisuga „*Icon not found*“. Küsitud ikoonide teekond ja seotud päring salvestatakse võti-

väärtus paaridena paisktabelisse, et päringut ei korrataks sama teekonna korduvküsimisel. Tagastatud sõne süstitakse tänu *innerHTML* atribuudile komponendi *div* elementi.

```
const requests = new Map<string, Promise<string>>();

export const getSvgContent(url: string) {
  let req = requests.get(url);

  if (!req) {
    req = fetch(url).then((rsp) => {
      if (rsp.status <= 299) {
        return rsp.text();
      }
      return Promise.reject('Icon not found');
    });
    requests.set(url, req);
  }
  return req;
};
```

Joonis 15. Meetod ikooni SVG faili sisu küsimiseks.

Selleks, et ikooni kujutis, värvus ja suurus oleks dünaamiliselt muutuv, kuulatakse *@Watch()* dekoraatoriga vastavate väljade muudatusi. Värvi muutumisel tuleb jälgida, et värvi muudetakse kõikidel failis sisalduvatel elementidel. Selle jaoks loodi meetod *colorSVGNodes(children, color)*, millele antakse sisendina SVG element ning värv (Joonis 16). Meetod käib rekursiivselt läbi kõik elemendi lapsed ning määrab neile oodatud värvi.

```
export function colorSVGNodes(children:  NodeListOf<ChildNode>, color?:
string) {
  for (let i = 0; i < children.length; i++) {
    const child = (children[i] as HTMLElement);

    if (child.nodeType === 1) {
      const structuralElements = ['defs', 'g', 'svg', 'symbol', 'use'];
      const graphicsElements = ['circle', 'ellipse', 'foreignObject',
'image', 'line', 'path', 'polygon', 'polyline', 'rect', 'text', 'textPath',
'tspan'];

      if (graphicsElements.includes(child.nodeName)) {
        if (color) {
          child.setAttribute('fill', color);
        } else {
          child.style.fill = 'currentColor'
        }

      } else if (structuralElements.includes(child.nodeName)) {
        this.colorSVGNodes(child.childNodes, color)
      }
    }
  }
}
```

Joonis 16. Meetod SVG elemendi värvimiseks.

### 4.2.3 Raadio loomine

Olukordades, mil soovitakse kuvada ainult ühte raadionuppu, on parem kasutada märkeruutu. Selle tõttu võib raadionuppe kuvada alati grupina, kuna eraldiseisvaid raadionuppe ei esine. Raadio jaoks loodi kaks komponenti: raadiogrupp ja raadionupp. Raadiogrupp võtab sisse *RadioItem* objektide massiivi ja renderdab soovitud väärtused raadionupu komponentidena üksteise alla. *RadioItem* on liides, mis koosneb kirje sildist ja väärtusest (Joonis 17).

```
export interface RadioItem {
  label: string;
  value: string;
}
```

Joonis 17. *RadioItem* liides.

Raadiogrupile antakse kaasa nimi, mis edastatakse igale raadionupu elemendile *name* atribuudina. *Name* atribuudiga jõustatakse, et uue raadionupu valimisel muudetakse eelnevalt valitud nupp „valimata“ olekusse.

### 4.2.4 Rippmenüü loomine

Rippmenüü loomisel ei kasutatud natiivset HTML <*select*> elementi stiliseerimisega seotud piirangute tõttu. Rippmenüü moodustati nupust ja sorteerimata nimekirjast <*ul*> elemendi näol. Nimekirja kuvatakse ja peidetakse CSS *display* deklaratsiooni abil: *display: block*, kui rippmenüü on avatud ja *display: none*, kui rippmenüü on suletud. Sarnaselt raadiole tehti menüü elementide sisestamiseks liides *DropdownItem* (Joonis 18). *DropdownItem* koosneb identifikaatorist, kirje sildist, ikoonist ja kirje valimisel sooritatavast funktsioonist.

```
export interface DropdownItem {
  id: string;
  label: string;
  icon: string;
  click: () => void;
}
```

Joonis 18. *DropdownItem* liides.

Komponent võtab vastu masiivi *DropdownItem* objektidega ning renderdab need <*li*> elementidena nimekirja sees.

Ekraanilugerite jaoks määrati rippmenüü nupule WAI-ARIA atribuudid *aria-haspopup* tõeväärtusega *true* ja *aria-expanded*. *Aria-expanded* väärtus muutub avatud oleku põhjal: *true*, kui rippmenüü on avatud ning *false*, kui rippmenüü on suletud.

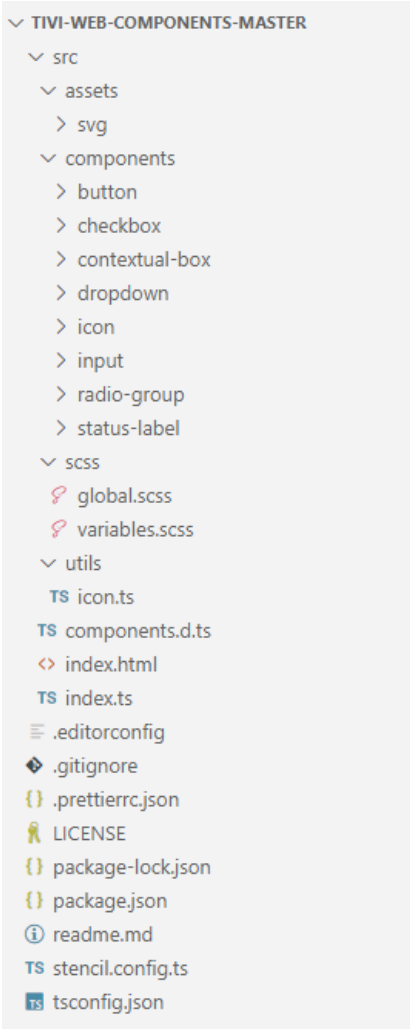
Kohandatud rippmenüül ei ole klaviatuuriga kasutamise toetust. Selleks kirjutati meetod, mis kuulab avatud oleku korral klaviatuuri sündmusi. „Üles“ või „alla“ klahvile vajutades fokusseeritakse vastavalt eelmist või järgmist kirjet. Sisestusklahvile või tühikuklahvile vajutades käivitatakse fokusseeritud kirje funktsioon ning suletakse menüü. Paoklahvile või tabeldusklahvile vajutades suletakse menüü.

## 4.3 Teegi publitseerimine

Valminud komponenditeek publitseeriti NPM registris. Kõik NPM paketid on määratletud failis *package.json*. Fail peab sisaldama vähemalt kahte välja: projekti nimi ja versioon. Versioonimisel järgiti NPM dokumentatsioonis soovitatavat inkrementeerimist: esimene publikatsioon on versioon 1.0.0 (*major*), tagurpidi ühilduvate uute funktsionaalsuste lisamisel suurendatakse keskmist numbrit (*minor*) ning tagurpidi ühilduvate vigade parandamisel suurendatakse viimast numbrit (*patch*). Keskmise numbri suurendamisel muudetakse viimane number nulliks [34]. Viimane lõputöö raames publitseeritud versioon on 1.2.3. Valminud teek on leitav registris aadressil https://www.npmjs.com/package/tivi-web-components.

## 4.4 Realisatsiooni kokkuvõte

Lõputöö raames valmisid kõik kaheksa komponenti. Joonisel 19 on toodud projekti struktuur peale komponentide loomist.




Joonis 19. Projekti struktuur peale komponentide loomist.

Realisatsiooni käigus ei ilmnenud väga suuri takistusi. Komponentide loomise protsess oli loogiline ja kõik raskused said lahendatud. Kõige keerulisem oli tööle saada ikooni komponenti. Algul oli probleem ikoonifailide kausta konfigureerimisega, et ikoonifailid ehitamisel õigesse kohta paigutataks. Lahendus saadi Stencil dokumentatsioonist, kus leidusid juhised ja näited pildifailidega tegelemiseks. Teine komponent, millega läks pikemalt aega, oli nupp. Nupul esines enim variatsioone ja mitmeid olekuid, mille puhul muutusid nupuga seotud värvid. Seetõttu tuli hoolikalt jälgida, et iga variatsiooni ja oleku värvid vastaks ettenähtud stiilile.

Lõputöö raames valminud teek sisaldab ainult komponente, mis on ettevõtte projektide raames praegu kasutusel. Komponendid on läbinud ettevõtte sisese koodi ülevaatuse. Teegi arendus jätkub seni, kuin kõik Smartmatic stiiliraamatus olevad komponendid saavad asendatud.

# 5 Teegi testimine

Käesolevas peatükis tutvustatakse komponenditeegi testimist. Komponente testitakse kahel peamisel viisil: ühiktestimine ja näidisprojektid. Ühiktestid kirjutati paralleelselt komponentide loomisega. Üheks lähtetingimuseks on, et komponenditeek peab olema kasutatav Angular ja Vue projektides. Selleks tehti mõlemas raamistikus projektid, kus kuvatakse kõik loodud komponendid ja demonstreeritakse nende funktsionaalsust.

## 5.1 Ühiktestimine

Igale komponendile kirjutati ühiktestid. Kõik testid sisaldavad kahte peamist kontrolli: ehitamine (*builds*) ja renderdamine (*renders*). Ehitamisel luuakse komponendi instants ning kontrollitakse selle tõesust. Renderdamisel kasutatakse funktsiooni *newSpecPage()*, et renderdada komponent HTML-ina ning vaadatakse, kas tagastatav väärtus vastab oodatud väärtusele. Kuna komponendid kasutavad varju DOM'i, on see tarvis oodatavas tulemuses märkida <*mock:shadow-root*> sildiga. Kui komponent kasutab malli, pannakse vastavasse kohta <*slot*> element ning selle oodatav sisu väljapoole varju DOM elementi. Joonisel 20 on kujutatud näide staatusmärgise peamistest testidest.

```
describe('tivi-status-label', () => {
  it('builds', () => {
    expect(new StatusLabelComponent()).toBeTruthy();
  });

  it('renders', async () => {
    const { root } = await newSpecPage({
      components: [StatusLabelComponent],
      html: '<tivi-status-label>Completed</tivi-status-label>',
    });
    expect(root).toEqualHtml(`
      <tivi-status-label>
        <mock:shadow-root>
          <span part='badge' class='badge badge-success'>
            <p class="status-text"><slot></slot></p>
          </span>
        </mock:shadow-root>
        Completed
      </tivi-status-label>
    `);
  });
});
```

Joonis 20. Näide: staatusmärgise ehitamise ja renderdamise testid.

Lisaks ehitamise ja renderdamise kontrollile kirjutati komponentidele omased testid. Komponentidel, millel on variatsioone, testiti vähemalt ühe variatsiooni korrektset renderdamist. Interaktiivsetel komponentidel kontrolliti kasutaja tegevuste korral oodatud sündmuste saatmist, näiteks märkeruudul klikates märkeruudu oleku muutumine.

## 5.2 Angular näidisprojekt

Komponenditeegi sõltuvus lisatakse projekti käsuga *npm install tivi-web-components*.

Kuna Angular projektides ei toetata vaikimisi kohandatud komponente, seadistati see manuaalselt. Selleks, et ikoonid oleks komponentidele kättesaadavad, määrati angular.json failis ikoonide väljundkaustaks projektisisene teekond *assets/svg*, mida kasutatakse teegis ikoonide küsimiseks. Samuti lisati teekond teegi globaalsele stiilifailile, et komponentidel oleks ligipääs teegis täpsustatud stiilimuutujatele ja kirjatüüpidele.

Komponentide näited jagati eraldiseisvates kaustades klassidesse. Kaustad genereeriti Angulari poolt käsuga *ng generate component [komponendi nimi]*, mis koostas komponendi *.html*, *.scss*, *.spec.ts* ja *.ts* failid. Loodud komponendinäited koondati baasprojektis sisalduvasse põhilisse HTML faili.

Valminud näidisprojekt on kättesaadav aadressil https://github.com/irrumm/component-test-angular. Lisa 1 all on välja toodud ekraanitõmmis Angular raamistikus loodud näidisprojektist.

## 5.3 Vue näidisprojekt

Sarnaselt Angular projektile installiti komponenditeegi sõltuvus, seadistati kohandatud komponentide toetus ja lisati teekond teegi stiilifailile. Ikoonide kättesaamiseks kasutati pluginat, millega võimalik kopeerida teegi ikoonide kausta sisu projekti *assets/svg* kausta.

Komponentide näited loodi ühisesse *components* kausta Vue failidena. Failid sisaldavad komponendi HTML-i, skripti ja stiile. Selleks, et komponendid saaksid kasutada stiliseerimisel Sass-i, installiti Sass-iga seonduvad sõltuvused ning lisati komponendifailides sisalduvatele stiiliblokkidele *lang='scss'*.

Valminud näidisprojekt on kättesaadav aadressil https://github.com/irrumm/component-test-vue. Lisa 2 all on välja toodud ekraanitõmmis Vue raamistikus loodud näidisprojektist.

## 5.4 Testimise tulemused

Ühiktestid töötavad ootuspäraselt. Komponendid läbivad ehitamise, renderdamise ja komponendipõhised testid. Läbitud testkomplekte on üheksa ning läbitud teste on kakskümmend seitse. Testid läbitakse 2.25 sekundiga. Joonisel 21 on esitletud jooksutatud ühiktestimise tulemused käsureal.

```
PASS  src/components/checkbox/tivi-checkbox.spec.ts
PASS  src/components/radio-group/tivi-radio-group.spec.ts
PASS  src/components/input/tivi-input.spec.ts
PASS  src/components/contextual-box/tivi-contextual-box.spec.ts
PASS  src/components/status-label/tivi-status-label.spec.ts
PASS  src/components/dropdown/tivi-dropdown.spec.ts
PASS  src/components/button/tivi-button.spec.ts
PASS  src/components/radio-group/radio/tivi-radio.spec.ts
PASS  src/components/icon/tivi-icon.spec.ts

Test Suites: 9 passed, 9 total
Tests:       27 passed, 27 total
Snapshots:   0 total
Time:        2.253 s
Ran all test suites.
```

Joonis 21. Ühiktestimise tulemused.

Angular ja Vue näidisprojektides kuvatud komponendid vastavad visuaalselt ja kasutajakogemuslikult Smartmatic stiiliraamatu loodud komponentidele. Variatsioonidega komponentide puhul kuvatakse kõik variatsioonid. Komponentidele, mis sisaldavad teisi elemente (näiteks rippmenüü), loodi massiiv elementide näidetega. Igast komponendist tehti lisaks ka arendaja poolt valitud värviskeemiga versioon, veendumaks, et modifitseerimisega seotud lähtetingimus on täidetud.

Näidisprojektides sisalduvad interaktiivsed komponendid töötavad ootuspäraselt Orca ja NVDA ekraanilugeritega. Komponentide fokusseerimisel teavitatakse kasutajat, milline element on fokusseeritud. Ekraanilugerid teavitavad kasutajat ka komponendi oleku muutumisel, näiteks rippmenüü avamine või sulgemine ja märkeruudu oleku muutmine.

# 6 Kokkuvõte

Käesoleva lõputöö eesmärk oli arendada kasutajaliidese komponentide teek, mis vastab ettevõtte visuaalsele stiilile ning oleks korduvkasutatav käimasolevates ja tulevastes arendusprojektides. Arendatava teegiga soovitakse välja vahetada ettevõtte emafirma poolt pakutav stiiliraamat, mis on aegunud.

Teegi loomiseks kaardistati esmalt komponentide visuaalsed ja funktsionaalsed nõuded. Lisaks koostati ülevaade elementide-põhistest ja üleüldistest ligipääsetavuse tingimustest, mida komponendid peavad järgima. Pärast kaardistamist võrreldi erinevaid raamistikke, mida teegi loomisel kasutada ning valiti neist kõige sobilikum.

Töö tulemusena valmis komponenditeek, mis sisaldas kaheksat komponenti. Teek publitseeriti NPM registris. Komponentide testimiseks kirjutati igale komponendile ühiktestid ning loodi näidisprojektid Angular.IO ja Vue.js raamistikes. Kõik ühiktestid läbiti positiivselt ja näidisprojektides esitletud komponendid olid ootuspärased nii visuaalselt kui ka kasutajakogemuslikult. Komponendid vastasid vajalikele ligipääsetavuse nõuetele ja töötasid ekraanilugeritega ettenähtud viisil. Eelmainitule toetudes võib tõdeda, et kõik lähtetingimused said täidetud.

Komponenditeegi arendamine ei ole lõppenud ning väljaspool lõputööd jätkatakse kõikide stiiliraamatu komponentide ümberkirjutamisega. Stiiliraamatu komponendid asendatakse teegiga siis, kui teegi arendus on jõudnud lõppfaasi.

# Kasutatud kirjanduse loetelu


[1] „E-hääletamine ja e-valimised,“ Riigi Infosüsteemi Amet, [Võrgumaterjal]. Available: https://www.id.ee/artikkel/e-haaletamine-ja-e-valimised/. [Kasutatud 05 February 2023].

[2] „Riigikogu valimised 2023,“ [Võrgumaterjal]. Available: https://rk2023.valimised.ee/et/detailed-voting-result/index.html. [Kasutatud 21 April 2023].

[3] „TIVI - Verifiable Voting,“ TIVI, [Võrgumaterjal]. Available: https://tivi.io/content/pdfs/Factsheet_TIVI-46baa9a2df.pdf. [Kasutatud 05 February 2023].

[4] C. Ramirez, „Web Components. What are they? Pros, Cons and more.,“ Medium, 22 March 2022. [Võrgumaterjal]. Available: https://medium.com/geekculture/web-components-what-are-they-pros-cons-and-more-77ad56711e49. [Kasutatud 25 January 2023].

[5] „HTML 4.13 Custom Elements,“ [Võrgumaterjal]. Available: https://html.spec.whatwg.org/multipage/custom-elements.html. [Kasutatud 15 January 2023].

[6] „What is Shadow DOM?,“ Mouseflow, [Võrgumaterjal]. Available: https://mouseflow.com/blog/shadow-dom-is-supported-by-mouseflow-what-is-shadow-dom-web-component/. [Kasutatud 15 January 2023].

[7] „<template>: The Content Template element,“ MDN Web Docs, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template. [Kasutatud 15 January 2023].

[8] R. Steele, „UI Components- What Are They?,“ CometChat, 20 May 2021. [Võrgumaterjal]. Available: https://www.cometchat.com/tutorials/what-are-ui-components. [Kasutatud 18 January 2023].

[9] „Design System vs Component Library: Key Differences,“ Ramotion, 24 September 2022. [Võrgumaterjal]. Available: https://www.ramotion.com/blog/design-system-vs-component-library/. [Kasutatud 15 January 2023].

[10] A. Smith, „What is WCAG and why is it important?,“ 30 August 2022. [Võrgumaterjal]. Available: https://resources.10to8.com/blog/wcag-compliance-at-10to8/. [Kasutatud 18 January 2023].

[11] „WAI-ARIA basics,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics. [Kasutatud 18 March 2023].

[12] S. Roberts, „6 reasons why you should be using SVG,“ Creative Bloq, 22 March 2021. [Võrgumaterjal]. Available: https://www.creativebloq.com/news/6-reasons-why-you-should-be-using-svg. [Kasutatud 16 February 2023].

[13] „How to Make Icons Accessible to The Widest Range of Users? 10 Best Practices,“ Medium, 3 June 2020. [Võrgumaterjal]. Available: https://medium.com/@OPTASY.com/how-to-make-icons-accessible-to-the-widest-range-of-users-10-best-practices-f1deefd4768b.

[14] C. Coyier, „Accessible SVG Icons,“ 28 December 2020. [Võrgumaterjal]. Available: https://css-tricks.com/accessible-svg-icons/. [Kasutatud 16 February 2023].

[15] P. J. Adam, „Building Accessible Buttons with ARIA: A11y Support Series,“ Deque, 28 September 2016. [Võrgumaterjal]. Available: https://www.deque.com/blog/accessible-aria-buttons/. [Kasutatud 13 February 2023].

[16] „<input>: The Input (Form Input) element,“ MDN Web Docs, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input. [Kasutatud 23 January 2023].

[17] „Inputs,“ [Võrgumaterjal]. Available: https://a11y-101.com/development/inputs. [Kasutatud 23 January 2023].

[18] „Accessible Checkbox,“ 17 September 2019. [Võrgumaterjal]. Available: https://www.a11ymatters.com/pattern/checkbox/. [Kasutatud 21 January 2023].

[19] „<input type="radio">,“ MDN Web Docs, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/radio. [Kasutatud 23 January 2023].

[20] „Dropdown,“ [Võrgumaterjal]. Available: https://carbondesignsystem.com/components/dropdown/usage/. [Kasutatud 10 February 2023].

[21] „Web Content Accessibility Guidelines (WCAG) 2.2,“ W3C, [Võrgumaterjal]. Available: https://www.w3.org/TR/WCAG22/. [Kasutatud 18 March 2023].

[22] „Why use a web component library?,“ WebComponents, [Võrgumaterjal]. Available: https://www.webcomponents.org/libraries. [Kasutatud 10 January 2023].

[23] „What is Lit?,“ Lit, [Võrgumaterjal]. Available: https://lit.dev/docs/. [Kasutatud 07 February 2023].

[24] „SkateJS - Effortless custom elements powered by modern view libraries.,“ [Võrgumaterjal]. Available: https://skatejs.netlify.app/. [Kasutatud 08 February 2023].

[25] „Stencil: A Web Components Compiler,“ [Võrgumaterjal]. Available: https://stenciljs.com/docs/introduction. [Kasutatud 08 February 2023].

[26] „Lit,“ Github, [Võrgumaterjal]. Available: https://github.com/lit/lit. [Kasutatud 12 February 2023].

[27] „Stencil,“ Github, [Võrgumaterjal]. Available: https://github.com/ionic-team/stencil. [Kasutatud 12 February 2023].

[28] „Skatejs,“ Github, [Võrgumaterjal]. Available: https://github.com/skatejs/skatejs. [Kasutatud 12 February 2023].

[29] „@stencil/core vs lit vs skatejs,“ npm trends, [Võrgumaterjal]. Available: https://npmtrends.com/@stencil/core-vs-lit-vs-skatejs. [Kasutatud 16 February 2023].

[30] „Introduction to SCSS,“ [Võrgumaterjal]. Available: https://codebots.com/docs/introduction-to-scss. [Kasutatud 17 January 2023].

[31] „Bootstrap 4 Introduction,“ GeeksforGeeks, [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/bootstrap-4-introduction/. [Kasutatud 04 February 2023].

[32] „Naming Rules for Custom Elements,“ Useful Angle, 2 February 2021. [Võrgumaterjal]. Available: https://usefulangle.com/post/363/custom-element-valid-name. [Kasutatud 02 March 2023].

[33] „CSS Shadow Parts,“ [Võrgumaterjal]. Available: https://ionicframework.com/docs/theming/css-shadow-parts. [Kasutatud 21 January 2023].

[34] „About semantic versioning,“ [Võrgumaterjal]. Available: https://docs.npmjs.com/about-semantic-versioning. [Kasutatud 15 April 2023].

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

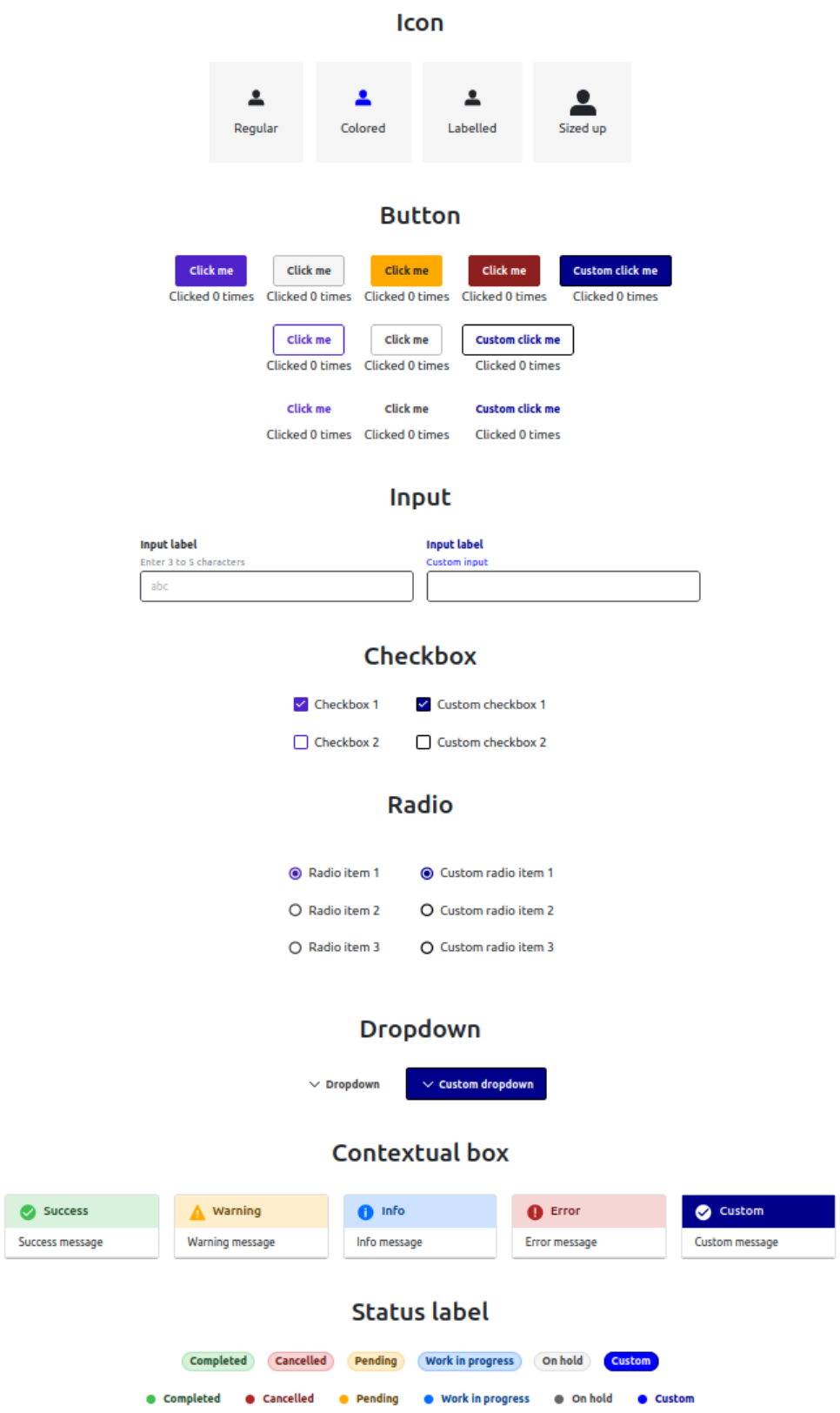
Mina, Iris Rumm

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebikomponentide teegi arendus Smartmatic stiiliraamatu baasil“, mille juhendaja on Karl-Erik Karu
   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.

15.05.2023

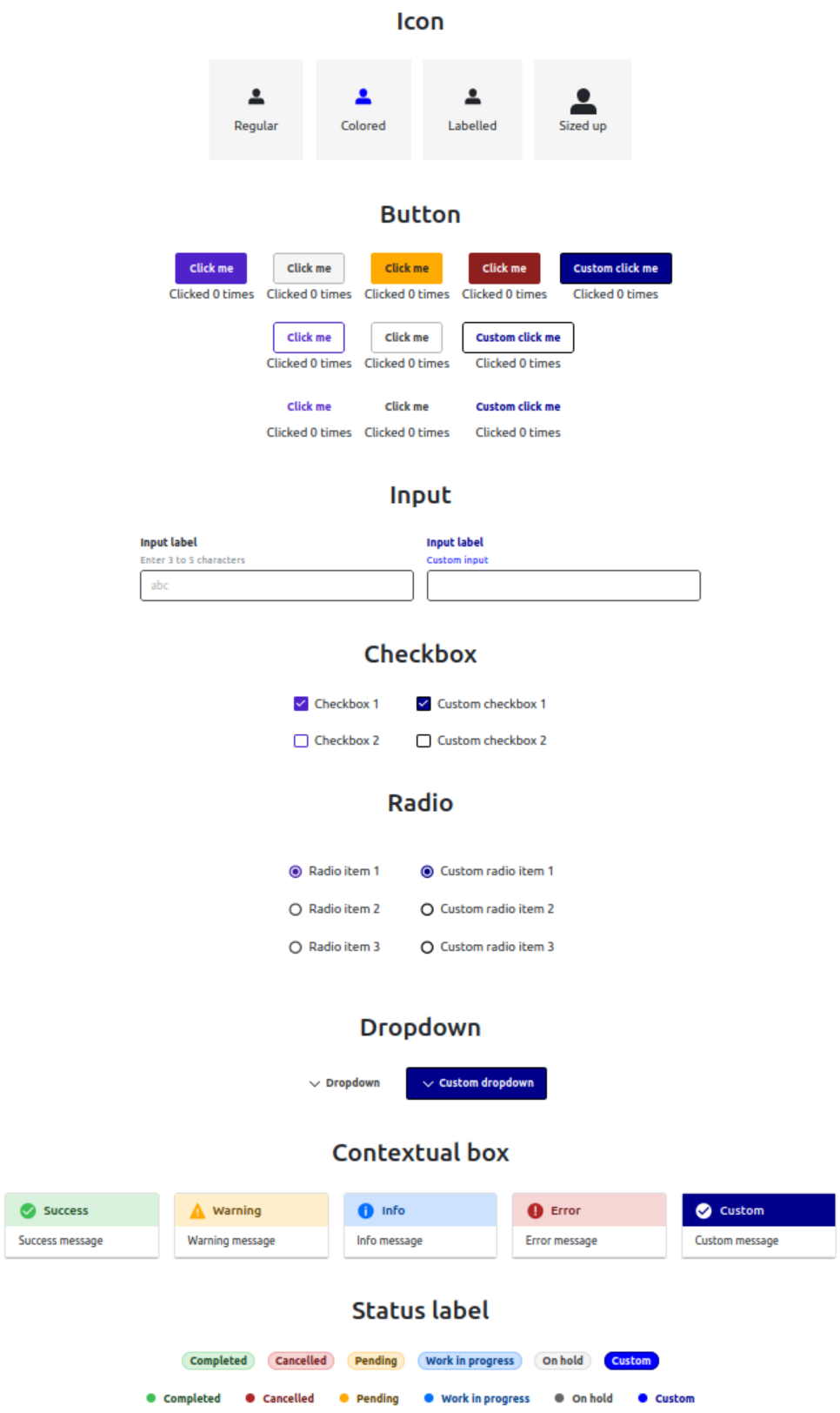
[1]Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või 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 – Näidisprojekt Angular raamistikus




Joonis 22. Näidisprojekt Angular.IO raamistikus.

# Lisa 3 – Näidisprojekt Vue raamistikus




Joonis 23. Näidisprojekt Vue.js raamistikus.