TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Danel Uukado 179495IADB

Kasutajaliidese stiilijuhiste näidislahenduse loomine tarkvaraarendusettevõttele

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2022

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: Danel Uukado

02.01.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärk oli luua veebipõhine näidislahendus kasutajaliidese stiilijuhistest, mida oleks tarkvaraarendusettevõttel Kodality OÜ hea eeskujuks võtta. Lahenduse puhul tuli tagada, et see oleks selge struktuuriga ning et see sisaldaks enamlevinud kasutajaliidese elemente. Kuna ettevõte keskendub peamiselt tervishoiu valdkonnale, oli ka loodava näidislahenduse seisukohalt oluline, et selle visuaalne disain sobiks tervishoiu rakendustele ja infosüsteemidele. Lisaks oli tähtis pöörata tähelepanu kasutusmugavusele ja funktsionaalsustele, mis on kasulikud tarkvaraarendajatele.

Lõputöös tehakse ülevaade lahendusele esitatud nõuetest, kasutatud tehnoloogiatest, arendusprotsessist ning testimisest. Eesmärgini jõudmiseks töötati läbi teemakohast kirjandust, mille abil selgitati välja parimad praktikad ja soovitused, mida oli stiilijuhendi loomisel asjakohane järgida. Samuti sisaldab käesolev töö ülevaadet võimalikest edasiarendustest.

Ettevõtte soovist lähtudes realiseeriti stiilijuhend kasutades Angular raamistikku. Arendusprotsessi lõpptulemuseks on töötav veebipõhine rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 6 peatükki, 6 joonist, 1 tabelit.

Abstract

Creating a Sample Solution of User Interface Style Guides for a Software Development Company

The aim of this bachelor's thesis was to create a web-based sample solution of user interface style guides that could serve as an example at a software development company Kodality OÜ. It was important to ensure that the solution had a clear structure and that it contained the most common user interface elements. As the company focuses primarily on the healthcare sector, it was also vital that the sample solution had a visual design that is suitable for healthcare applications and information systems. In addition, it was necessary to pay attention to ease of use and functionalities that are useful for software developers.

The thesis provides an overview of the solution by covering the requirements, used technologies, development process and testing. To fulfill the objective of the work, relevant sources were used for researching best practices and recommendations to follow when creating the style guide. The paper also contains some ideas for further developments.

Due to the company's wishes, the style guide was implemented by using the Angular framework. The end result of the development process is a working web-based application.

The thesis is in Estonian and contains 25 pages of text, 6 chapters, 6 figures, 1 table.

Lühendite ja mõistete sõnastik

CD	*Continuous Delivery*, pidev tarne
CI	*Continuous Integration*, pidev integratsioon
CSS	*Cascading Style Sheets*, veebilehtede kujundamiseks kasutatav keel
Disainikeel	Disainireeglite ja -põhimõtete kogum [12]
DOM	*Document Object Model*, dokumendi objektimudel
HTML	*HyperText Markup Language*, veebilehtede märgendamise keel
HTTP	*HyperText Transfer Protocol*, hüperteksti edastusprotokoll
JavaScript	Programmeerimiskeel
JSON	*JavaScript Object Notation*, JavaScriptil põhinev andmevahetusvorming
JSX	*JavaScript XML*, JavaScripti süntaksi laiendus
LESS	*Leaner Style Sheets*, CSS-i laiendav keel
RWD	*Responsive Web Design*, reageeriv veebidisain
SPA	*Single Page Application*, üheleherakendus
TypeScript	JavaScripti täiendav programmeerimiskeel
UI	*User Interface*, kasutajaliides
URL	*Uniform Resource Locator*, veebiaadress
WCAG	*Web Content Accessibility Guidelines*, veebi sisu juurdepääsetavussuunised

Sisukord

1 Sissejuhatus 10
2 Ülevaade probleemist 12
2.1 Vajadus näidislahenduse järele 12
2.2 Loodava stiiliraamatu tüüp 13
3 Loodava näidisrakenduse analüüs 14
3.1 Kasutajaliideste disain tervishoius 14
3.2 Kasutajaliidese elementide valik ja grupeerimine 15
3.3 Funktsionaalsete nõuete määramine 17
3.3.1 Koodilõikude kuvamine 17
3.3.2 JSON väljundi kuvamine 17
3.3.3 Olekute kuvamine 18
3.3.4 Otsingukasti kuvamine 18
3.4 Mittefunktsionaalsete nõuete määramine 18
3.4.1 Rakenduse tehnoloogia 18
3.4.2 Kasutajaliidese disain 19
3.4.3 Rakenduse kasutatavus 20
3.5 Versioonihaldus 20
4 Teostus 22
4.1 Rakenduse arhitektuur 22
4.2 Kasutajaliidese elementide loomine 23
4.2.1 Nupu loomine 24
4.2.2 Märkeruudu loomine 24
4.3 Vaadete loomine 25
4.3.1 Vormielementide vaade 25
4.3.2 Muud vaated 28
4.4 Testimine 29
4.4.1 Funktsionaalsuste testimine 29
4.4.2 Juurdepääsetavuse testimine 30
5 Hinnang tulemusele 32

5.1 Saavutatud kasutusmugavus 32
5.2 Võimalikud edasiarendused 33
6 Kokkuvõte 34
Kasutatud kirjandus 35
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 37
Lisa 2 – Rakenduse komponent: nupp 38
Lisa 3 – Rakenduse komponent: märkeruut 39
Lisa 4 – Veebirakenduse vaated 41

Jooniste loetelu

Joonis 1. Nupp: (a) aktiivses olekus, (b) mitteaktiivses olekus, (c) laadimise olekus.... 24
Joonis 2. Märkeruut: (a) aktiivses ja märgitud olekus; (b) aktiivses ja märkimata olekus; (c) mitteaktiivses ja märgitud olekus; (d) mitteaktiivses ja märkimata olekus. 25
Joonis 3. Kuvatõmmis vormielementide vaate ülaosast: vaikimisi seadistustega..... 26
Joonis 4. Kuvatõmmis vormielementide vaate keskosast: „Show JSON“ seadistusega. 27
Joonis 5. Kuvatõmmis vormielementide vaate ülaosast: „Show Disabled“ seadistusega. 27
Joonis 6. Kuvatõmmis vormielementide vaate ülaosast: kuvades „Selector“ saki sisu. 28

Tabelite loetelu

Tabel 1. Loodava stiiliraamatu üldine struktuur..... 16

1 Sissejuhatus

Tarkvaraarenduse puhul on tänapäeval üheks oluliseks aspektiks UI (*User Interface*) ehk kasutajaliidese disain, millele on tihti seatud kõrged nõuded nii värvilahenduse kui ka kujunduse osas. Samuti on tähtis ühesuguse stiili tagamine kogu tarkvara ulatuses, mis eeldab arendajatelt häid erialaseid oskusi ning ülevaate omamist kasutusel olevatest UI elementidest. Mahukamate projektide puhul võib aga paratamatult ette tulla olukordi, kus mõnest elemendist on rakenduses tekkinud mitu erineva kujundusega varianti. Tõhus viis selle vältimiseks oleks välja töötada kasutajaliidese stiilijuhised (ingl *User Interface Guidelines*) ehk luua stiiliraamat, millega määratakse kindlaks kõik kasutatavad elemendid ning nende disain.

Kasutajaliidese stiilijuhiste olemasolu tähtsust ja kasulikkust mõistab ka tarkvaraarendusettevõte Kodality OÜ, mis keskendub peamiselt veebipõhistele IT-lahendustele tervishoiu valdkonnas. Stiiliraamatu, mis oleks selge struktuuriga ja mis täidaks täielikult oma eesmärki, loomine võib aga osutuda üsna ajakulukaks ülesandeks. Sellest tulenevalt on Kodality OÜ huvitatud näidislahendusest, millele tuginemine aitaks ettevõttel tulevastes projektides stiilijuhendi loomisel aega kokku hoida.

Käesoleva lõputöö peamine eesmärk on luua veebipõhine näidislahendus kasutajasõbralikust stiiliraamatust, mis sisaldaks enamlevinud UI elemente ning milles sisalduv disain oleks sobilik meditsiinirakendustele. Lisaks peab Kodality OÜ oluliseks pöörata tähelepanu arendajasõbralikkusele, et pakkuda välja ideid, mille rakendamine tulevastes projektides aitaks ettevõtte tarkvaraarendajatel oma tööd kiiremaks ja tõhusamaks muuta. Ettevõtte huvi arvestades tuleb autoril seatud eesmärgini jõudmisel kasutada Angular raamistikku. Lähtekoodi kirjutamiseks kasutab autor integreeritud arenduskeskkonda IntelliJ IDEA ning koodi haldamiseks versioonihalduskeskkonda GitLab.

Käesolev töö koosneb neljast osast. Esimeses osas selgitatakse teema valikut ning tehakse ülevaade probleemist. Teises osas tuuakse välja kasutatavad tehnoloogiad ja esitatakse nõuded, millele loodav lahendus peab vastama. Lisaks töötatakse läbi teemakohast

kirjandust, et välja selgitada head tavad, mida järgida veebipõhise kasutajaliidese ning stiiliraamatu loomisel. Töö kolmandas osas kirjeldatakse lahenduse teostust. Neljandas antakse hinnang saavutatud tulemusele ning tuuakse välja võimalikud edasiarendused.

2 Ülevaade probleemist

Kasutajaliidese stiilijuhiste loomine on tänapäeval tarkvaraarenduses levinud praktika. Tegu on süstematiseeritud kogumiga kõikidest kasutajaliidese elementidest, millest konkreetne toode koosneb [1]. Stiilijuhendi põhiülesandeks on aidata tagada ühtne disain kogu toote ulatuses [2]. Isegi kui stiiliraamatu loomine ja pidev täiendamine tähendab ettevõttele lisakulusid, on selle olemasolu suuremate projektide puhul kriitilise tähtsusega ning toob pikas perspektiivis pigem kasu kui kahju.

2.1 Vajadus näidislahenduse järele

Ettevõttes Kodality OÜ on arendatud seni mitmeid rakendusi, kus ühtse UI disaini saavutamiseks on tuginetud stiiliraamatutele. Stiilijuhend on eelkõige mõeldud selle loomise ja täiendamise eest vastutavatele arendajatele ning ettevõtte kliendile – lõppkasutajatele pole see kättesaadav.

Tutvudes mõne ettevõttes loodud stiiliraamatuga lähemalt, võis aga täheldada mõningaid nõrku külgi. Näiteks olid töö autori hinnangul UI elemendid kohati ebaselgelt grupeeritud. Samuti võis märgata rakenduses kasutusel olevate elementide puudumist stiilijuhendist.

Olles töötanud arendajana ettevõttes Kodality OÜ, puutus ka töö autor kokku projektiga, mis eeldas stiiliraamatule toetumist ning selle täiendamist. Arenduse käigus tundis aga autor puudust näidisest, millele oleks hea tugineda ja mis aitaks tööd lihtsustada. Sellest tulenevalt tekkis ka suurem huvi käesoleva töö teema vastu ning kasvas välja idee universaalse näidislahenduse loomisest.

Ettevõtte poolsest huvist tulenevalt peaks loodava lahenduse disain olema sobilik suurematele tervishoiu valdkonda kuuluvatele rakendustele ja infosüsteemidele. Näidis võiks välja pakkuda ka uut moodi lähenemist, mis puudutab kasutajaliidese stiilijuhiseid ning meditsiinirakenduste disaini. Loodavale lahendusele oleks ettevõtte arendajatel hea

tugineda eelkõige projektide puhul, kus kliendi poolt pole ette antud spetsiifilisi nõudmisi UI visuaalsele disainile, mis erineks näidislahenduses toodust.

2.2 Loodava stiiliraamatu tüüp

Kasutajaliideste arendamise jaoks loodud stiiliraamatud täidavad kõik samasugust eesmärki, kuid sellegipoolest jagunevad need üldjuhul järgmiselt:

- Staatilised stiilijuhised (ingl *Static Style Guides*) – tootearenduse algusfaasis loodav staatiline dokument. Kiiresti areneva ja muutuva toote puhul eeldab enamasti suurt panust, et dokument oleks alati ajakohane ning kooskõlas toote hetkeseisuga [3].
- Dünaamilised ehk „elavad“ stiilijuhised (ingl *Dynamic/Living Style Guides*) – HTML-põhised (*HyperText Markup Language*) juhised, mis on otseselt seotud arendatava toote lähtekoodiga. Muudatused koodis kajastuvad koheselt ka stiiliraamatus ning tänu sellele puudub vajadus eraldiseisva staatilise dokumendi järele, mille uuendamine eeldaks lisa ajakulu [3]. Lisaks sellele, et dünaamiline stiiliraamat on lihtsamini hallatav, on selles sisalduvad kasutajaliidese elemendid nii välimuse kui ka interaktiivsuse poolest identsed tootes esinevatega.

Käesoleva töö käigus luuakse näidislahendus, kus rakendatakse dünaamiliste stiilijuhiste põhimõtet.

3 Loodava näidisrakenduse analüüs

Käesolevas peatükis tehakse vajalik eeltöö veebipõhise näidisrakenduse loomiseks. Peatükk sisaldab analüüsi, nõuete määramist ning ülevaadet tehnoloogiatest.

3.1 Kasutajaliideste disain tervishoius

Kuna tervishoid on väga keeruline ja kompleksne valdkond, kajastub keerukus ka tänapäeva IT-lahendustes. Suuremad infosüsteemid sisaldavad tihti vaateid ja vorme, kus kuvatakse korraga väga palju erinevat informatsiooni ning milles orienteerumine ei pruugi olla lihtne. Telemeditsiini eksperdi Eric Greenmani sõnul on kasutajasõbralike ja intuitiivsete IT-lahenduste loomine, mis samal ajal hõlmaksid endas tervishoiuandmetele omast keerukust, suur väljakutse [4]. Oluliseks märksõnaks, millele tuleks kasutajaliideste puhul palju tähelepanu pöörata, on lihtsus [4].

Tervishoiu kasutajaliideste loomisel on mõistlik järgida üldlevinud tavasid ja soovitusi, mida rakendatakse ka teistes valdkondades, kuna nende tõhusus ja otstarbekus on varasemalt leidnud juba palju tõestust. Näiteks on tähtis meeles pidada, et sobiva kirjatüübi ja -suuruse valik mängib kasutajasõbralikkuse tagamisel suurt rolli. Samuti on mõistlik võtta aluseks veebi kasutatavuse eksperdi Jakob Nielseni 10 heuristikut ehk kasutatavuse printsiipi. Kuigi tegu on põhimõtetega, mis pärinevad aastast 1994, on need asjakohased ka tänapäeval [5]. Näiteks on oluline kaheksas printsiip, mille kohaselt peaks disain olema esteetiline ja minimalistlik [5]. Visuaalsed elemendid peaksid toetama kasutajate peamisi soove ja eesmärke – vältida tuleks kõike, mis lisab üleliigset keerukust [5].

Samas on võimalik välja tuua ka mõningaid soovitusi, mis on rohkem suunatud tervishoiu IT-lahendustele. Näiteks tuleks meditsiinirakenduste puhul erksate värvide asemel eelistada pigem külmi või pastelseid toone, mis omavad rahustavat mõju [6]. Samuti on oluline mitte pöörata liiga palju tähelepanu erinevatele UI trendidele, kuna väljatöötatud ja pikalt kasutuses olnud lahendused täidavad sageli hästi oma eesmärki ning sobivad paremini tervishoiule omase keerukusega [7].

Käesoleva peatükiga selgitas töö kirjutaja välja üldised soovitused ning põhimõtted, mida on otstarbekas järgida tarkvara kasutajaliideste arendamisel tervishoiu valdkonnale. Peatükis toodut peab autor oluliseks rakendada ka loodava stiilijuhendi näidislahenduse puhul.

3.2 Kasutajaliidese elementide valik ja grupeerimine

Veebipõhise stiiliraamatu loomisel võis üheks oluliseks nõudeks pidada selget struktuuri. Selle tagamine eeldas UI elementide loogilist grupeerimist lihtsasti eristatavatesse kogumitesse, võttes arvesse nende funktsionaalsusi ja visuaalseid omadusi.

Kasutajaliidese elementide rohkust ja eripärasid arvestades on võimalik neid väga erinevalt grupeerida. Üheks võimaluseks oleks jaotada need nelja suurde kategooriasse: sisendi juhtelemendid (ingl *Input Controls*), navigeerimiskomponendid (ingl *Navigational Components*), teabekomponendid (ingl *Informational Components*) ja ümbriselemendid (ingl *Containers*) [8]. Kirjeldatud jaotusele oli töö autori hinnangul mõistlik toetuda, kuid oluline oli ka nende täpsem grupeerimine.

Kasutajaliidese elementide grupeerimisel oli autori hinnangul oluline tugineda ka Geštalt printsiipidele. Tegu on 20. sajandi alguses esitatud põhimõtetega visuaalsest tajust, mille kohaselt näeb inimene eraldatud servade, joonte ja alade asemel terviklikke vorme, kujundeid ja objekte [9]. Üheks olulisimaks neist on läheduse printsiip (ingl *Principle of Proximity*), mis ütleb, et objektide lähestikku paiknevus aitab neid näha grupeerituna [9].

Eriti oli aga kasulik toetuda sarnasuse printsiibile (ingl *Principle of Similarity*), mille kohaselt sarnase välimusega objektid tunduvad grupeeritud [9]. Seega tuli kasutajaliidese elementide grupeerimisel pöörata tähelepanu ka nende värvile, kujule ja suurusele.

Võttes arvesse Geštalt printsiipe, pani töö autor paika loodava stiiliraamatu üldise struktuuri, jaotades enamlevinud UI elemendid sobivatesse gruppidesse (Tabel 1). Lisaks eespool kirjeldatud kategooriatele, aitas elementide valimisele kaasa kasutajakogemuse spetsialisti Page Laubheimeri poolt kirja pandud 25 UI elementi, mida kasutajaliidese stiilijuhendid tavaliselt sisaldavad [2].

Tabel 1. Loodava stiiliraamatu üldine struktuur.

Grupp	**Element/alajaotus**	**Märkused**
Tüpograafia		Tuua välja kirjalaadid (tavakiri, kaldkiri, paks kiri) ja kasutusvaldkonnad (pealkirjad, sisutekstid).
Värvid	Põhivärvid	
	Tugivärvid	
Nupud	Nupud (ingl *buttons*)	
	Lingid (ingl *links*)	
Vormid	Tekstiväli (ingl *text field*)	
	Teksti ala (ingl *text area*)	
	Otsinguväli (ingl *search field*)	Kuvada testandmeid.
	Rippmenüü (ingl *drop-down list*)	Kuvada testandmeid.
	Arvu sisestus (ingl *numeric input*)	
	Kuupäeva valija (ingl *date picker*)	
	Kellaaja valija (ingl *time picker*)	
	Märkeruut (ingl *checkbox*)	
	Raadionupp (ingl *radio button*)	
Tabelid	Töödeldav (ingl *editable*) tabel	Lisada paginatsioon ja sorteerimine.
	Mittetöödeldav (ingl *non-editable*) tabel	Lisada paginatsioon ja sorteerimine.
Ümbriselemendid	Akordion (ingl *accordion*)	
	Sakid (ingl *tabs*)	
Teavitused	Hüpikteated (ingl *toasts*)	
	Laadimise indikaator (ingl *loading indicator*)	
	Vihjemullid (ingl *tooltips*)	
Ikoonid		Igale ikoonile lisada juurde selle kasutust võimaldav identifikaator.
Muu	Sildid (ingl *tags*)	
	Kahe-olekuline lüliti (ingl *toggle switch*)	

Lisaks UI elementide valimisele ja grupeerimisele selgitas autor toodud tabeliga välja, millistest vaadetest peaks loodav veebipõhine lahendus koosnema. Võttes arvesse elementide arvu, moodustatud gruppide eristatavust ning võimalikke edasiarendusi tulevikus, otsustas töö kirjutaja, et iga grupi jaoks tuleks kasutajaliideses teha eraldi vaade. Kuigi Geštalt printsiipidele tuginemine võimaldas arusaadava struktuuri loomist ka ühelehelises stiiliraamatus, oleks veebileht väga pikaks osutunud ning selles orienteerumise ebamugavaks teinud.

3.3 Funktsionaalsete nõuete määramine

Käesolevas alampeatükis määratakse loodava näidisrakenduse funktsionaalsed nõuded. Samuti selgitatakse lühidalt nende vajalikkust.

3.3.1 Koodilõikude kuvamine

Kuigi stiilijuhend, mis sisaldab selgesti struktureeritult kõiki kasutajaliidese elemente, täidab oma peamist eesmärki, on see tarkvaraarendaja jaoks ebapiisav. Selleks, et stiiliraamatus toodut oleks arenduse käigus mugav kasutusse võtta, oli autori hinnangul oluline ka koodilõikude kuvamine. Samuti oli parema kasutatavuse tagamiseks asjakohane näidata nuppu, mille abil oleks vastavat koodi võimalik kiirelt lõikelauale kopeerida.

Kuna koodilõikude lisamine muudaks stiiliraamatu kasutajaliideses orienteerumise raskemaks, on tähtis neid vaikimisi kasutaja eest varjata. Kasutajakogemuse spetsialisti Page Laubheimeri kohaselt tagatakse koodi peitmine tihti akordioni nimelise elemendi abil [2]. Tegu on olemuselt vertikaalselt laotud üksuste kogumiga, kus üksuse peal vajutamine võimaldab kuvada ja peita sellega seotud sisu [8]. Laubheimerile tuginedes pidas loodava lahenduse puhul akordioni kasutamist mõistlikuks ka töö autor.

3.3.2 JSON väljundi kuvamine

Lisaks koodilõikude kuvamisele oli üheks oluliseks nõudeks JSON (*JavaScript Object Notation*) kujul väljundi näitamine kasutajapoolset sisendit nõudvate elementide puhul. Tegu on funktsionaalsusega, mis aitab elementide korrektset töötamist kontrollida ning hõlbustada arendustegevust.

3.3.3 Olekute kuvamine

Kuna teatud kasutajaliidese elementide puhul saab eristada aktiivset (ingl *enabled*) ja mitteaktiivset (ingl *disabled*) olekut, siis pidas töö autor oluliseks ka nende välja toomist. Selleks, et stiiliraamat ei muutuks liiga keerukaks, oli autori hinnangul mõistlik võimaldada vastavas olekus elementide kuvamine märkeruudu abil.

3.3.4 Otsingukasti kuvamine

Selleks, et stiiliraamatust oleks kiirelt võimalik konkreetset UI elementi leida, pidas töö autor oluliseks ka otsingukasti kuvamist. Tegu oli olulise nõudega, kuna aitab vältida liigset ajakulu stiiliraamatus orienteerumiseks.

3.4 Mittefunktsionaalsete nõuete määramine

Käesolevas alampeatükis määratakse kasutatavad tehnoloogiad ning tehakse ülevaade võimalikest alternatiividest. Samuti esitatakse mõningad mittefunktsionaalsed nõuded, mis puudutavad rakenduse kasutatavust.

3.4.1 Rakenduse tehnoloogia

Veebirakenduste kiiremaks ja mugavamaks arenduseks on loodud erinevaid JavaScripti (programmeerimiskeel) raamistikke, millest kõigil on omad eelised ja puudused. Kuigi saadaval olevaid raamistikke on palju, on mõningad nende seast arendajate hulgas laiemalt levinud kui teised. Populaarsed valikud kaasaegsete kasutajaliideste arendamisel on järgmised [10]:

- Angular – Google'i poolt arendatud TypeScripti (JavaScripti täiendav programmeerimiskeel) põhine raamistik, mis omab palju sisseehitatud funktsionaalsusi. Lisaks DOM'i (*Document Object Model*) ehk dokumendi objektimudeli muutmisele on raamistiku poolt tagatud näiteks HTTP (*HyperText Transfer Protocol*) päringute sooritamine, kasutaja sisendi valideerimine ja testimine. Tegu on laiaulatusliku terviklahendusega kasutajaliideste arendamiseks, kus peamine rõhk seisneb korduvkasutatavate UI komponentide loomises [10].
- React – Facebooki poolt arendatud teek kasutajaliideste loomiseks, mis võimaldab peamiselt vaid DOM'i muutmist. Kuigi React on Angulariga võrreldes väga

minimalistlik, on puuduvad funktsionaalsused võimalik tagada Reacti kogukonna poolt loodud vahendite abil. Samuti on Reacti üheks oluliseks tunnuseks JSX'i (*JavaScript XML*) kasutus. Tegu on JavaScripti süntaksi laiendusega, mis võimaldab ühendada HTML'i ja JavaScripti loogika [10].

- Vue – raamistik, mille loojaks on Evan You ning mille arendus toimub kogukonnapõhiselt. Sisseehitatud funktsionaalsuste poolest jääb Vue Angularile alla, kuid Reactiga võrreldes on funktsionaalsusi rohkem. Sarnaselt Angularile, on Vue puhul tagatud koodi (s.o JavaScript) ja malli (s.o HTML) eraldatus. Sarnaselt Angularile ja Reactile, toimub ka Vue puhul kasutajaliideste ehitamine korduvkasutatavaid komponente kombineerides [10].

Ettevõttes Kodality OÜ on muutunud tavaks Angulari kasutamine. Eelistus kujunes välja ajal, mil SPA (*Single Page Application*) ehk üheleherakenduste loomisele suunatud raamistikke oli oluliselt vähem ning üheks põhjuseks, mis soodustas Angulari valimist, oli TypeScripti kasutamise võimaldamine. Raamistiku kasuks tuli otsustada ka käesoleva töö autoril. Rakenduse lähtekoodi kirjutamiseks valis autor integreeritud arenduskeskkonna IntelliJ IDEA, kuna omas sellega palju kogemusi.

3.4.2 Kasutajaliidese disain

Tuginedes peatükile 3.4.1 võib väita, et Angular raamistiku poolt on tagatud erinevad vahendid, mis muudavad veebipõhiste rakenduste arendamise kiireks ja mugavaks. Sellele vaatamata tuleb Angular raamistiku kasutusele võtmisel teha üldjuhul veel üks oluline otsus. Kuna kasutajasõbralike ja atraktiivsete UI komponentide loomine on mahukas protsess ning eeldab suurel hulgal CSS (*Cascading Style Sheets*) koodi, on mõistlik võtta aluseks mõni valmislahendus, mida vastavalt vajadustele kohandada. Mõningad levinud vahendid (teegid valmiskomponentidega) Angularile on järgmised:

- Angular Material – komponentide teek, mis on loodud Angulari tiimi poolt ning mis rakendab Google'i disainikeelt ehk disainireeglite ja -põhimõtete kogumit nimega Material Design [11], [12]. Saada olevaid valmiskomponente on 36 [13]. Lisaks kuulub Angular Material'i juurde teek Angular CDK (*Component Dev Kit*), kus on 18 täiendavat tööriista erinevate UI elementide loomiseks [14].
- NG Bootstrap – teek, mis on loodud tuginedes Bootstrap 4 raamistikule. Teegil puuduvad kolmandate osapoolte JavaScripti sõltuvused [15]. Saada olevaid komponente on 18 [16].

- PrimeNG – Angulari komponentide teek, mis sisaldab üle 90 UI komponendi. Teegis on rakendust leidnud nii Bootstrap'i kui ka Material Design'i stiilireeglid [17].
- NG-ZORRO – komponentide teek, mis põhineb Ant Design'i nimelisel disainikeelel. Saada olevaid komponente on üle 60 [18].

Töö kirjutaja välistas NG-ZORRO kasutuselevõtu, kuna tegu on teegiga, millega Kodality OÜ on juba tihedalt kokku puutunud ning mille pakutavad võimalused on ettevõttele hästi teada. Kuigi NG Bootstrap sisaldas enamlevinud UI elemente, võis selle puhul täheldada komponentide vähesust võrreldes teiste populaarsete teekidega. Vaatamata sellele, et Angular Material on loodud Angulari tiimi poolt, otsustas autor tänu komponentide rohkusele PrimeNG kasuks. Samuti soosis PrimeNG valimist selle elementide visuaalne disain, mis töö autori hinnangul vastasid paremini peatükis 3.1 toodud soovitustele ning olid sobilikumad kasutamaks ka suuremate tervishoiu infosüsteemide puhul.

3.4.3 Rakenduse kasutatavus

Lisaks rakenduse tehnoloogiale ja kasutajaliidese disainile on oluline välja tuua veel mõningad mittefunktsionaalsed nõuded, mis puudutavad rakenduse kasutatavust. Arvestades, et töö raames loodav lahendus on veebipõhine, peab see vastama järgmistele nõuetele:

- Rakendus peab olema kasutatav enamlevinud veebilehitsejatega (Google Chrome, Mozilla Firefox, Microsoft Edge).
- Kasutajaliidese arendamisel tuleb arvestada erinevate ekraani resolutsioonidega. Järgida tuleb RWD (*Responsive Web Design*) ehk reageeriva veebidisaini põhimõtet, mille kohaselt veebilehe kujundus sõltub kasutatava seadme ekraani suurusest [19].

3.5 Versioonihaldus

Loodava näidislahenduse puhul on oluline kasutusele võtta koodihalduskeskkond, kuna tegu on tänapäeval väga levinud praktikaga tarkvaraarenduses ning aitab omada head ülevaadet koodis tehtud muudatustest. Samuti võimaldavad versioonihalduskeskkonnad

koodi teistega mugavalt jagada. Populaarsed keskkonnad, mille hulgast valida, on järgmised:

- GitHub – keskkond, millel on üle 65 miljoni kasutaja ning üle 200 miljoni koodihoidla (ingl *repository*), mis teeb sellest kõige suurema kasutajaskonnaga keskkonna. GitHub võimaldab piiramatult tasuta luua nii avalikke kui ka privaatseid koodihoidlaid [20].
- GitLab – keskkond, millel on üle 30 miljoni registreeritud kasutaja. GitLab on tuntud oma võimeka CI/CD (*Continuous Integration* / *Continuous Delivery*) tööriista poolest [21]. Nii nagu GitHubi puhul, saab ka GitLabi keskkonnas tasuta genereerida avalikke ja privaatseid koodihoidlaid [22].
- Bitbucket – keskkond, mida kasutab üle 10 miljoni arendaja. Tasuta versioon võimaldab piiramatult luua privaatseid koodihoidlaid, kuid seda vaid kuni 5-liikmelistele tiimidele. Bitbucket pakub liidestust ka projektihaldustarkvaraga Jira [23].

Töö autor otsustas GitLabi kasuks, kuna see on kasutusel ka OÜs Kodality. Ettevõte hindab kõrgelt selle pakutavat CI/CD lahendust.

4 Teostus

Käesolevas peatükis tehakse ülevaade rakenduse arhitektuurist ning tehnilisest teostusest. Suur rõhk on kasutajaliidese elementide ja vaadete loomisel. Lisaks antakse ülevaade ka näidislahenduse testimisest, kirjeldades nii funktsionaalsuste kui ka juurdepääsetavuse testimist. Peatükk koosneb neljast alampeatükist.

4.1 Rakenduse arhitektuur

Käesoleva töö raames valmib rakendus, mille loomiseks kasutatakse Angular raamistikku. Rakendus on veebipõhine ning sõltub välisest teegist nimega PrimeNG, mis pakub arendajatele erinevaid UI komponente. Kuna tegu on lahendusega, mis ei eelda andmete töötlemist, puudub vajadus ka eraldiseisva andmebaasiga suhtleva serverrakenduse vastu. Teostusel kasutatakse Angular 12.2.7 ning PrimeNG 12.2.0 versioone.

Angular rakendusi iseloomustab modulaarne arhitektuur ning olulisel kohal on koodi korduvkasutatavuse võimaldamine. Peamisteks ehitusplokkideks on Angulari komponendid (*Components*), mis on jaotatud erinevatesse moodulitesse (*NgModules*). Moodulid sisaldavad enamasti sarnaseid või omavahel tihedalt seotud komponente. Iga Angular rakendus sisaldab ühte juurmoodulit, mille nimeks on tavapäraselt *AppModule* ning mille kaudu alustab rakendus oma tööd [24].

Lähtudes loodava lahenduse olemusest ning sellele esitatud nõuetest, otsustas autor rakenduse jagada üheksaks mooduliks:

- *AppModule* – rakenduse juurmoodul.
- *AppRoutingModule, HomeRoutingModule, UigRoutingModule* – moodulid, kus registreeritakse kõik rakenduse URL-id ning komponendid, mida konkreetse veebiaadressi ehk URL-i korral kasutajale kuvada.
- *CoreModule* – moodul, mis sisaldab stiilijuhendis esitletavaid kasutajaliidese elemente. Reaalse arendusprojekti puhul oleks tegu mooduliga, kus paikneksid kõik projektis kasutust leidvad elemendid.

- *HomeModule* – moodul, mis vastutab rakenduse avavaate eest.
- *MenuModule* – moodul, mis sisaldab rakenduse menüüd, mille abil kasutaja saab liikuda erinevate vaadete vahel.
- *PageModule* – kasutajaliidese vaadete üldise struktuuri ja väljanägemise eest vastutav moodul. Samuti paikneb selles HTML dokumendi pealkirjastamisega seotud loogika.
- *UigModule* – moodul, mis sisaldab stiilijuhendi vaateid.

Kirjeldatud jaotus aitab autori hinnangul rakenduses tagada selge struktuuri ning muudab koodis orienteerumise lihtsaks. Mooduliteks jagamisel ning nende nimetamisel tugines töö kirjutaja OÜs Kodality kehtivatele tavadele.

Angulari komponent koosneb klassist, mis sisaldab rakenduse andmeid ja loogikat ning on seotud HTML-malliga, mida kasutatakse vaate kuvamiseks [24]. Komponendi juurde kuulub enamasti ka CSS kood, millega määratakse vaate visuaalne disain. Raamistik lubab tavalise CSS-i asemel ka erinevate eeltöötlejate (ingl *preprocessor*) kasutamist, mis laiendavad CSS-i võimalusi ning lisavad täiendavat funktsionaalsust. Töö autor otsustas LESS (*Leaner Style Sheets*) nimelise eeltöötleja kasuks, kuna see on leidnud kasutust ka OÜs Kodality. Näiteks laseb LESS luua muutujaid, tänu millele on kood lihtsamini hallatav [25].

4.2 Kasutajaliidese elementide loomine

Kuna töö autor oli otsustanud kasutusele võtta valmiskomponente pakkuva teegi PrimeNG, oli kasutajaliidese elementide loomiseks kaks lähenemisviisi:

- Esitleda stiiliraamatu vaadetes PrimeNG komponente.
- Esitleda stiiliraamatu vaadetes enda loodud eraldi komponente, mis tuginevad PrimeNG komponentidele.

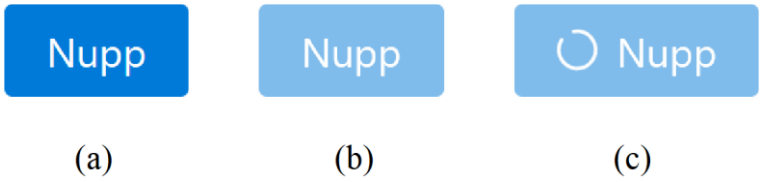
Autor pidas mõistlikumaks viimast lähenemisviisi, kuna see tagab komponentide mugavama kohandatavuse. Näiteks olukorras, kus komponent on rakenduses kasutusel juba mitmes kohas ning saabub vajadus selle täiustamiseks, piisaks koodis muudatuste tegemisest vaid ühes kohas. Samuti puudub oht erinevuste tekkeks. Kuigi käesoleva töö raames loodava lahenduse puhul ei olnud mugava kohandatavuse võimaldamine kriitilise

tähtsusega, pidas autor oluliseks lähtuda arhitektuurilistest põhimõtetest, mille rakendamine oleks asjakohane ka reaalsetes arendusprojektides.

Olles loonud mooduli nimega *CoreModule*, asus autor UI elementide arendamise juurde. Kokku valmis töö raames 17 komponenti, millest kahe loomist kirjeldab kirjutaja lähemalt.

4.2.1 Nupu loomine

Korduvkasutatava nuppu kuvava komponendi loomiseks genereeris käesoleva töö autor kaks omavahel seotud faili, milleks olid *button.component.ts* ja *button.component.html* (Lisa 2). Selleks, et valmiv komponent oleks paindlik, lisas töö kirjutaja TypeScripti ehk *.ts* laiendiga failis olevasse *ButtonComponent* klassi mitu *@Input()* dekoraatoriga tähistatud muutujat, mis võimaldavad komponenti kasutaval ülemkomponendil muuta selle erinevaid omadusi. Autor tagas, et nende abil oleks võimalik määrata nupu tüüpi, olekut ning sellel kuvatavat teksti. Näiteks võis nupp omada aktiivset, mitteaktiivset või laadimise olekut (Joonis 1).



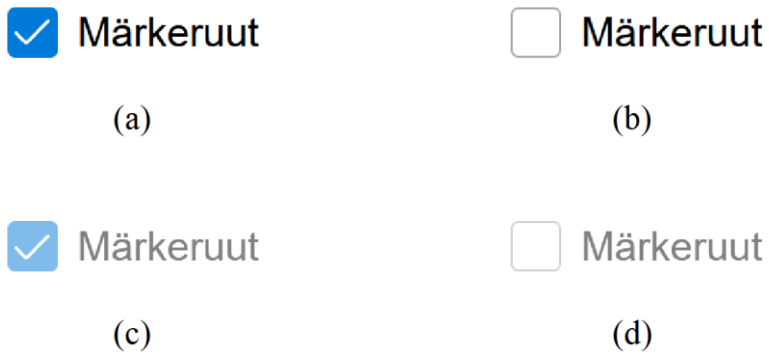
Joonis 1. Nupp: (a) aktiivses olekus, (b) mitteaktiivses olekus, (c) laadimise olekus.

Komponendi mallis ehk *.html* laiendiga failis kasutas autor PrimeNG nupu komponenti. Selleks, et eespool kirjeldatud muutujate väärtused kajastuksid ka PrimeNG nupul, leiab aset *property binding* ehk andmete sidumine, mille abil toimub vastavate väärtuste edastamine PrimeNG komponenti [26]. Samuti lisas töö kirjutaja *ButtonComponent* klassi *@Output()* dekoraatoriga tähistatud muutuja, mille tüübiks oli *EventEmitter* ning millega oli võimalik ülemkomponenti nupuvajutusest teavitada. Selle rakendamiseks lõi autor meetodi *onButtonClick()*, mis kutsutakse välja PrimeNG nupu peal vajutamise korral ning kus emiteeritakse vastav sündmus ülemkomponendile.

4.2.2 Märkeruudu loomine

Märkeruudu jaoks lõi töö kirjutaja failid *checkbox.component.ts* ja *checkbox.component.html* (Lisa 3). Sarnaselt nupule lisas autor märkeruudu *CheckboxComponent* klassi *@Input()* dekoraatoritega tähistatud muutujaid. Komponendi

puhul sai määrata märkeruudu juures kasutajale kuvatavat teksti ning elemendi olekut (Joonis 2).



Joonis 2. Märkeruut: (a) aktiivses ja märgitud olekus; (b) aktiivses ja märkimata olekus; (c) mitteaktiivses ja märgitud olekus; (d) mitteaktiivses ja märkimata olekus.

Erinevalt nupust, kus oluline oli vaid vajutusest teada andmine, pidi märkeruut omama muudetavat väärtust. Kuna märkeruut on element, mida tihti kasutatakse vormides, siis pidas töö autor tähtsaks komponendi kasutatavust ka Angulari vormides. Selleks tuli tagada, et *CheckboxComponent* klass realiseeriks *ControlValueAccessor* liidest *@angular/forms* paketist.

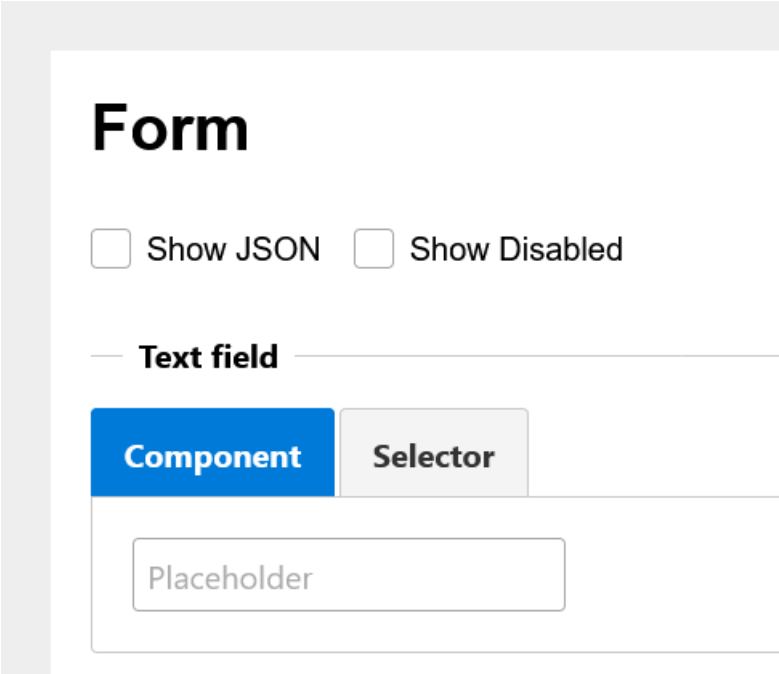
4.3 Vaadete loomine

Stiiliraamatu arendamisel lähtus töö kirjutaja põhimõttest, et *CoreModule*'is olevad komponendid on mõeldud vaid stiiliraamatus esitlemiseks, mitte vaadete ehitamiseks. Näiteks olukorras, kus stiilijuhendi vaade eeldas seadete muutmise võimaldamiseks märkeruudu olemasolu, tuli selle lisamiseks kasutada vastavat PrimeNG komponenti. Töö autor valis kirjeldatud lähenemise, kuna reaalsetes arendusprojektides ei pruugi vaja minna kõiki *CoreModule*'is toodud UI elemente ning pidas oluliseks järgida põhimõtet, et stiiliraamat ei peaks esitlema komponente, mis on loodud vaid stiiliraamatu vaadete koostamiseks.

4.3.1 Vormielementide vaade

Vormielementide vaate puhul tagas käesoleva töö autor, et selle ülaosas paikneksid vaate pealkiri ja märkeruudud, mille abil oleks võimalik seadistada kõiki kuvatavaid UI elemente ning nende juurde kuuluvat lisainfot (Joonis 3). Näiteks oli märkeruudu, mida kirjeldab tekst „Show JSON", loomise eesmärgiks JSON kujul väljundi näitamise sisse- ja väljalülitamise võimaldamine. Selleks, et vaatesse jõudval kasutajal, kelle jaoks

JSON väljundi nägemine pole esmatähtis, oleks koheselt vaates lihtsam orienteeruda, määras autor selle vaikimisi olekuks *false* ehk „märkimata“. Lisaks lõi töö kirjutaja märkeruudu „Show Disabled“, mille abil sai elemente kuvada kas aktiivses või mitteaktiivses olekus ning mille vaikimisi väärtuseks sai samuti „märkimata“. Kuigi kirjeldatud seadistuste puhul oli alternatiiviks tuua need iga elemendi juures eraldi välja, oli autori hinnangul mõistlikum nende rakendatavus kogu vaate ulatuses, et vältida kasutajaliideses liigset keerukust.

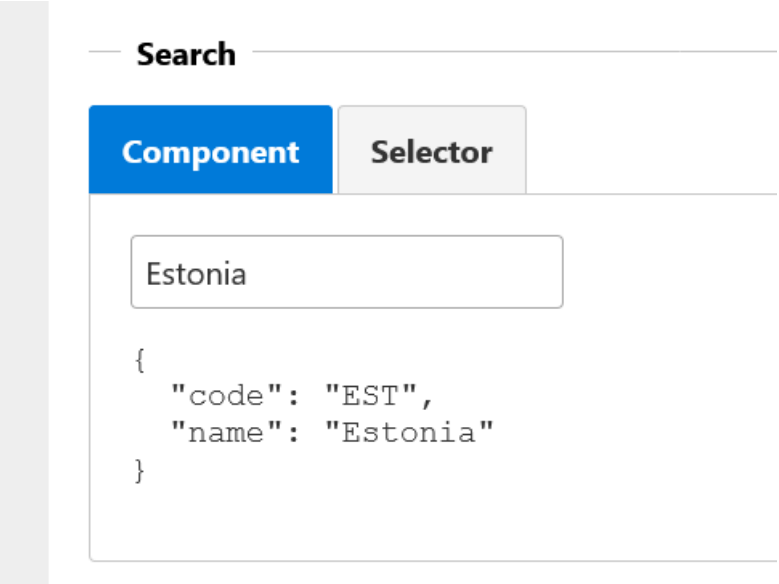


Joonis 3. Kuvatõmmis vormielementide vaate ülaosast: vaikimisi seadistustega.

Loodava veebipõhise stiiliraamatu puhul oli oluline näidata erinevat lisainfot UI elementide kohta ning kasutusmugavuse seisukohalt oli tähtis kogu info välja tuua konkreetse komponendi juures. Algselt plaanis töö autor iga elemendi kuvamiseks ja liigse info vaikimisi peitmiseks rakendada kasutajaliideses akordioni, kuid olles lähemalt tutvunud PrimeNG pakutavate valmiskomponentidega ning arvestanud stiiliraamatu võimalike edasiarendustega tulevikus, otsustas autor akordioni asemel sakkide kasuks. Töö kirjutaja loobus akordionite kasutuselevõtust, kuna nende puhul oleks veebileht pikemaks osutunud ning nõudnud kasutajatelt rohkem kerimist, et näha kõiki vaates sisalduvaid UI elemente.

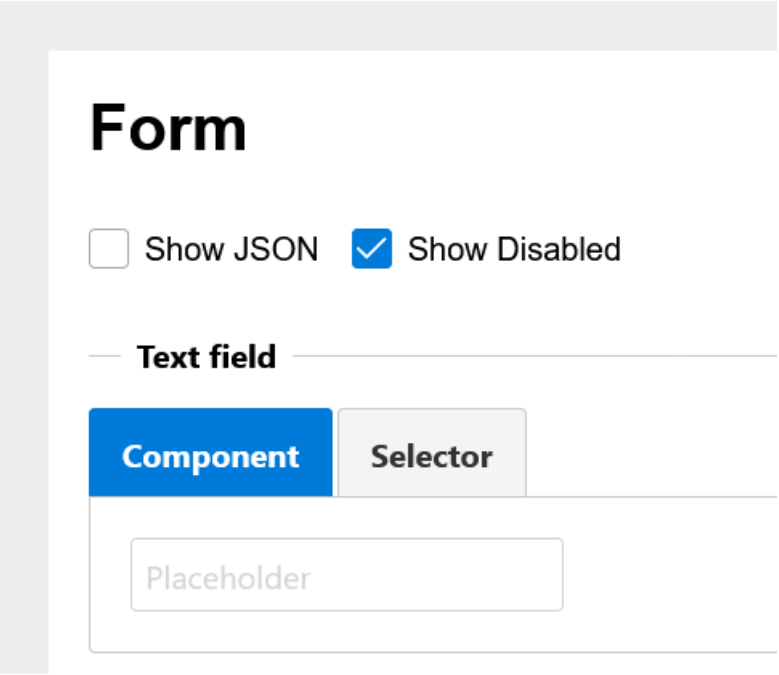
Sakkide kuvamiseks lõi töö autor korduvkasutatava komponendi, mida sai rakendada ka teistes vaadetes. Valminud komponent võimaldas valida „Component“ ja „Selector“ saki vahel. „Component“ saki sisus otsustas autor kuvada UI elementi ning selle all vastava

elemendi JSON kujul väljundit juhul, kui on määratud eespool kirjeldatud seadistus „Show JSON“ (Joonis 4). JSON-i näitamiseks rakendas töö kirjutaja Angulari sisseehitatud *JsonPipe pipe*'i, mis tagab etteantud väärtuse kuvamise nõutud kujul [27]. Selleks, et vaatesse jõudes oleks kasutajale kõik UI elemendid koheselt nähtaval, määras autor „Component“ saki vaikimisi valitud sakiks.



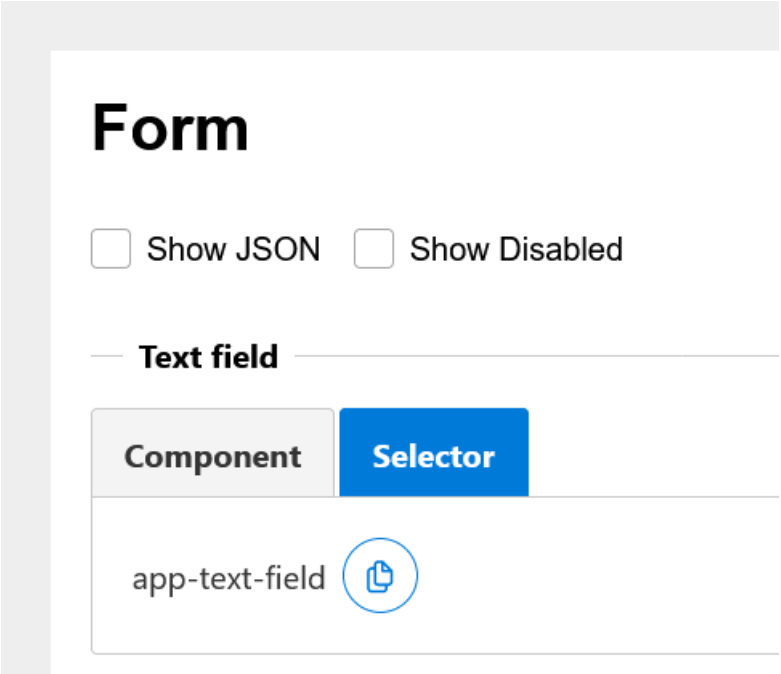
Joonis 4. Kuvatõmmis vormielementide vaate keskosast: „Show JSON“ seadistusega.

„Show Disabled“ seadistuse korral kuvatakse „Component“ sakis elementi mitteaktiivses olekus (Joonis 5).



Joonis 5. Kuvatõmmis vormielementide vaate ülaosast: „Show Disabled“ seadistusega.

„Selector“ saki puhul oli eesmärgiks kuvada komponendi selektorit, mis võimaldab HTML mallides vastavat komponenti kasutusse võtta (Joonis 6). Näiteks tuleb komponendi, mille selektoriks on *app-text-field*, kasutamiseks malli lisada märgendid *<app-text-field></app-text-field>*. Lisaks tagas autor, et saki sisus oleks ka ikoon, mille peal vajutades saaks selektori kiirelt lõikelauale kopeerida.



Joonis 6. Kuvatõmmis vormielementide vaate ülaosast: kuvades „Selector“ saki sisu.

Kuigi ka komponendi selektori kuvamise oleks saanud märkeruuduga seadistatavaks teha, oli töö autori eesmärgiks luua sakkide abil lahendus, mis võimaldab edasiarenduste korral mugavalt elementide juurde täiendavat infot lisada. Uute sakkide loomisel on tagatud info vaikimisi peitmine ning välistatud on kasutajaliidese liigselt keerukaks ja detailseks muutumine.

4.3.2 Muud vaated

Stiiliraamatu vaadete arendamisel pööras töö autor rõhku sellele, et need oleks oma ülesehituselt võimalikult sarnased (Lisa 4). Näiteks paiknesid sarnaselt vormielementide vaatele ka nuppude vaate ülaosas märkeruudud, millega sai teha teatud seadistusi. Lisaks „Show Disabled“ seadistusele oli võimalik märkida valik „Show Loading“, mille abil sai kõiki nuppe näha laadimise olekus. „Show Loading“ seadistuse võimaldas töö kirjutaja ka tabelite vaates.

Sarnasust aitas kasutajaliideses tagada ka vormielementide vaate arenduse käigus loodud sakkide komponent, mis leidis veel rakendust vaadetes nagu „Button“, „Table“, „Notification“ ja „Miscellaneous“. Teistsugust lähenemist eeldas näiteks „Notification“ ehk teavituste vaates toodud hüpikteated, mille nägemine oli võimaldatud nuppude abil. Tüpograafia vaates leidis aga kasutust PrimeNG komponent nagu *Fieldset*, mis lubab päisel vajutades peita ja kuvada sellega seotud sisu.

4.4 Testimine

Käesolevas alampeatükis kirjeldatakse loodud näidislahenduse testimist. Ülevaade sisaldab nii funktsionaalsuste kui ka juurdepääsetavuse testimist.

4.4.1 Funktsionaalsuste testimine

Kvaliteetse tarkvara loomisel mängib olulist rolli selle testimine. Kuigi käesoleva töö raames arendatud rakendus ei hõlmanud suurel hulgal keerukat funktsionaalsust, oli ka selle testimine siiski vajalik. Olgugi et töö kirjutaja testis funktsionaalsusi juba nende arendamise käigus, oli kvaliteedis veendumiseks tähtis ka hilisem kontroll. Arvestades, et rakendus pidi korrektselt töötama enamlevinud veebilehitsejatega, valis autor selle testimiseks brauserid nagu Google Chrome (versioon 96.0.4664.45), Mozilla Firefox (versioon 94.0.2) ja Microsoft Edge (versioon 96.0.1054.34). Testimisel oli oluline veenduda, et soovitud tegevusi oleks võimalik sooritada ning et veebilehitsejate konsoolides ei ilmneks vigu. Kõigi kolme brauseriga testis töö autor järgmisi funktsionaalsusi:

- Avavaates olevad lingid peavad avanema uuel vahelehel.
- Menüü abil peab rakenduses saama navigeerida.
- Valikud „Show JSON“, „Show Disabled“ ja „Show Loading“ peavad olema rakendatavad.
- Komponentide selektoreid peab saama lõikelauale kopeerida.
- Hüpikteated peavad vaates ilmuma ülesse paremasse nurka ning neid peab saama sulgeda.

Kirjeldatud funktsionaalsused töötasid kõikides brauserites korrektselt.

4.4.2 Juurdepääsetavuse testimine

Lisaks manuaalsele funktsionaalsuste testimisele pidas autor oluliseks kontrollida lahenduse vastavust juurdepääsetavuse nõuetele. Kuigi töö alguses seatud mittefunktsionaalsete nõuete hulka see ei kuulunud, tuleb ligipääsetavusele pöörata rohkem tähelepanu edasiarenduste käigus, et rakenduse kasutajaliides oleks loodud viisil, mis arvestaks veebisisu kättesaadavusega ka erivajadustega inimestele.

Kasutajaliideste juurdepääsetavuse kontrollimiseks on välja töötatud standard nimega WCAG (*Web Content Accessibility Guidelines*). Standard koosneb neljast juurdepääsetavuse põhimõttest, mille alla kuuluvad neid täpsustavad suunised. Suunised omakorda jagunevad edukriteeriumiteks, mille puhul on võimalik eristada kolme taset: A (madalaim), AA ja AAA (kõrgeim). Selleks, et saavutada kasutajaliideses soovitud ligipääsetavuse tase, tuleb täita kõik vastava taseme ning sellest madalamate tasemete edukriteeriumid [28].

Selleks, et hinnata loodud näidislahenduse vaadete juurdepääsetavust, oli mõistlik kasutada mõnda tasuta kättesaadavat validaatorit, mis kontrolliks automaatselt HTML koodi vastavust WCAG standardile. Autor testis nõuetele vastavust töö lõppfaasis ning kasutas selleks veebipõhist validaatorit nimega AChecker. Kuigi testimise hetkel oli WCAG standardi värskeimaks versiooniks 2.1, võimaldas AChecker kontrollida versiooni 2.0 nõuete täitmist. Testitavaks tasemeks määras töö kirjutaja AA.

Esmase kontrolli abil tuvastas autor mittevastavusi kolmele edukriteeriumile, milleks olid 1.3.1, 1.4.4 ja 3.3.2. Peale mõningate paranduste tegemist jäid veel alles edukriteeriumid 1.3.1 ja 3.3.2, mis eeldasid täpsustavate siltide (ingl *label*) kuvamist vormielementide juures. Valminud rakenduse eripäradest tulenevalt ei leidnud autor aga esinenud probleemidele kiiret ja asjakohast lahendust. Sellele vaatamata tuleb ilmnenud vead edasiarenduste käigus parandada, et kõik automaatselt kontrollitavad nõuded saaksid korrektselt täidetud.

Ehkki automaatsed testimisvahendid aitavad leida puudusi HTML koodis, on juurdepääsetavuse tagamiseks oluline ka manuaalne kontroll [29]. Näiteks osutus autori hinnangul probleemseks kohaks erinevas olekus nuppude visuaalne sarnasus. Kuigi nuppe on võimalik eristada nende värvi alusel, võib sellest aga jääda väheks. Tuginedes edukriteeriumile 1.4.1, mille kohaselt ei tohiks värv olla ainus vahend elementide

eristamiseks, on lahenduse edasiarenduste käigus mõistlik rakendada veel mõnda moodust, mis aitaks visuaalselt nuppude olekutel paremini vahet teha [28]. Probleemi leevendaks näiteks see, kui kuvada mitteaktiivse nupu taustal triibulist mustrit.

5 Hinnang tulemusele

Saavutatud tulemusega võib jääda rahule, kuna töö raames valmis näidislahendus selge struktuuriga stiiliraamatust. Arenduse käigus said täidetud kõik põhilised funktsionaalsed nõuded, milles aitas veenduda loodud rakenduse testimine. Töö skoopi kuulunud funktsionaalne nõue, mille kohaselt pidi UI elementide kiiremaks leidmiseks kuvama kasutajale otsingukasti, jäi küll täitmata, kuid lahenduse edasi arendamisel on see oluliseks prioriteediks. Samuti vajab veel mõningast tähelepanu reageeriva disaini põhimõtete rakendamine.

Töö autor selgitas näidislahenduse loomisega välja, milline võiks üks stiiliraamat välja näha ning kasutas selle arendamisel UI komponentide teeki PrimeNG, millega ettevõttel varasem kogemus puudus. PrimeNG kasutuselevõtt oli igati sobilik, kuna selle komponentide disain ning värvilahendus oli kohane meditsiinirakendustele.

5.1 Saavutatud kasutusmugavus

Saavutatud kasutusmugavust võib hinnata kõrgeks, kuna UI elemendid on arusaadavalt jaotatud erinevatesse vaadetesse ning neid esitletakse asjakohasel viisil. Elemente kuvatakse kasutajaliideses koos täiendava lisainfoga, mis on lihtsasti leitav ning mis on kasulik tarkvaraarendajatele. Samuti muudab lahenduse kasutajasõbralikuks selle lihtne visuaalne disain.

Hinnates koos ettevõttega saavutatud kasutusmugavust, pakkusid mõtteainet märkeruudud, mida kuvatakse nuppude ja vormielementide vaadete ülaosas ning millega saab kogu vaate ulatuses teha seadistusi. Kuna reaalsetes arendusprojektides võib vaates oluliselt rohkem UI komponente olla, võib selline lahendus aga kasutajatelt palju kerimist nõuda. Näiteks olukorras, kus stiiliraamatu kasutaja on vaate alaosas leidnud mõne elemendi ning tahab mõnda seadistust muuta, tuleb tal selleks vaate ülaossa kerida ning peale seda soovitud element uuesti üles otsida. Kuigi loodud näidislahenduse puhul ei olnud see kriitiline, on suurtes arendusprojektides mõistlik kaaluda kirjeldatud märkeruutude kuvamist eraldi iga komponendi juures.

5.2 Võimalikud edasiarendused

Kuna rakenduse *CoreModule*'isse jäid loomata mõned töö skoopi kuulunud UI komponendid, tuleks esmalt keskenduda nende lisamisele. Stiiliraamatus jäid esitlemata elemendid nagu akordion, sakid ja töödeldav tabel. Samuti vajab veel tähelepanu värvipaleti vaade.

Rakendusel on mitmeid erinevaid edasiarendamise võimalusi. Näiteks oleks üheks oluliseks täienduseks vormielementide sisendi valideerimine. Kuna reaalsetes projektides on vajalik tagada ebasobivast sisendist kasutajale teavitamine, peaks seda ka stiilijuhendi näidislahendus kajastama. Lisaks saab edasi arendada koodilõikude kuvamist. Kuigi töö autor tagas, et stiiliraamatus näidatakse UI komponentide selektoreid, mis võimaldavad arendajatel komponente mugavalt kasutusele võtta, oleks asjakohane välja tuua ka terviklikke koodilõike, mida on kasutatud elementide kuvamiseks stiiliraamatus. Samuti tuleb edasiarenduste käigus pöörata tähelepanu juurdepääsetavusele, et lahendus vastaks WCAG 2.1 tasemele AAA.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua veebipõhine näidislahendus selgesti struktureeritud ja kasutajasõbralikust stiiliraamatust, millele tuginemine võimaldab tarkvaraarendusettevõttel Kodality OÜ tulevastes projektides aega kokku hoida. Lahenduseni jõudmiseks selgitati esmalt välja, milliseid kasutajaliidese elemente peaks loodav stiiliraamat sisaldama. Seejärel määrati kindlaks kõik funktsionaalsed ja mittefunktsionaalsed nõuded. Näidislahenduse loomisel pöörati palju tähelepanu kasutusmugavusele ning selleks, et saavutada parim võimalik tulemus, töötati läbi teemakohast kirjandust.

Seatud eesmärki võib pidada täidetuks, kuna töö käigus valmis veebipõhine lahendus, mida on OÜl Kodality hea eeskujuks võtta ning millele toetumine aitab kaasa kõrgetasemeliste kasutajaliideste loomisele. Vaatamata sellele, et rakendus omab mitmeid edasiarendamise võimalusi, on selles tagatud kõik kriitilise tähtsusega funktsionaalsused.

Kasutatud kirjandus

[1] J. Gothelf, *Lean UX: Applying Lean Principles to Improve User Experience*. Beijing: O'Reilly Media, 2013.

[2] P. Laubheimer, *Front-End Style-Guides: Definition, Requirements, Component Checklist*, Nielsen Norman Group, 2016. [Online]. Loetud aadressil: https://www.nngroup.com/articles/front-end-style-guides Kasutatud: 04.09.2021.

[3] A. Williams, *Your Product Needs a Style Guide*, The Path Forward. [Online]. Loetud aadressil: https://thepathforward.io/your-product-needs-style-guide Kasutatud: 10.09.2021.

[4] E. Greenman, *Human-centric simplicity in healthcare IT design*, iTel, 2018. [Online]. Loetud aadressil: https://www.itelcompanies.com/?p=1877 Kasutatud: 22.09.2021.

[5] J. Nielsen, *10 Usability Heuristics for User Interface Design*, Nielsen Norman Group, 2020. [Online]. Loetud aadressil: https://www.nngroup.com/articles/ten-usability-heuristics Kasutatud: 22.09.2021.

[6] D. Bila, *User Interface Design Best Practices for Medical and Healthcare Apps*, UGEM Design, 2020. [Online]. Loetud aadressil: https://ugem.design/blog/ui-design-tips-for-healthcare-apps Kasutatud: 22.09.2021.

[7] E. Stein and B. Gulten, *Designing for Generations – A Look at UX in Healthcare*, UX Booth, 2019. [Online]. Loetud aadressil: https://www.uxbooth.com/articles/designing-for-generations-ux-in-healthcare Kasutatud: 22.09.2021.

[8] *User Interface Elements*, Usability.gov. [Online]. Loetud aadressil: https://www.usability.gov/how-to-and-tools/methods/user-interface-elements.html Kasutatud: 04.09.2021.

[9] J. Johnson, *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Amsterdam: Morgan Kaufmann, 2010.

[10] M. Schwarzmüller, *Angular vs React vs Vue – My Thoughts*, Academind, 2020. [Online]. Loetud aadressil: https://academind.com/tutorials/angular-vs-react-vs-vue-my-thoughts Kasutatud: 05.09.2021.

[11] *Angular Material*, Angular Material. [Online]. Loetud aadressil: https://material.angular.io Kasutatud: 26.09.2021.

[12] *What Actually Constitutes Design Language?*, UXPin. [Online]. Loetud aadressil: https://www.uxpin.com/studio/blog/design-language Kasutatud: 21.12.2021.

[13] *Components*, Angular Material. [Online]. Loetud aadressil: https://material.angular.io/components/categories Kasutatud: 26.09.2021.

[14] *CDK*, Angular Material. [Online]. Loetud aadressil: https://material.angular.io/cdk/categories Kasutatud: 26.09.2021.

[15] *NG Bootstrap*, NG Bootstrap. [Online]. Loetud aadressil: https://ng-bootstrap.github.io/#/home Kasutatud: 26.09.2021.

[16] *NG Bootstrap Components*, NG Bootstrap. [Online]. Loetud aadressil: https://ng-bootstrap.github.io/#/components Kasutatud: 26.09.2021.

[17] *PrimeNG*, PrimeNG. [Online]. Loetud aadressil: https://www.primefaces.org/primeng/showcase Kasutatud: 26.09.2021.

[18] *Ant Design of Angular*, NG-ZORRO. [Online]. Loetud aadressil: https://ng.ant.design/docs/introduce/en Kasutatud: 26.09.2021.

[19] A. Schade, *Responsive Web Design (RWD) and User Experience*, Nielsen Norman Group, 2014. [Online]. Loetud aadressil: https://www.nngroup.com/articles/responsive-web-design-definition Kasutatud: 09.10.2021.

[20] *GitHub: Where the world builds software*, GitHub. [Online]. Loetud aadressil: https://github.com Kasutatud: 23.09.2021.

[21] *Is it any good?*, GitLab. [Online]. Loetud aadressil: https://about.gitlab.com/is-it-any-good Kasutatud: 23.09.2021.

[22] *GitLab Pricing*, GitLab. [Online]. Loetud aadressil: https://about.gitlab.com/pricing Kasutatud: 23.09.2021.

[23] *Bitbucket*, Bitbucket. [Online]. Loetud aadressil: https://bitbucket.org Kasutatud: 23.09.2021.

[24] *Introduction to Angular concepts*, Angular. [Online]. Loetud aadressil: https://angular.io/guide/architecture Kasutatud: 02.10.2021.

[25] *Less CSS*, Less CSS. [Online]. Loetud aadressil: https://lesscss.org Kasutatud: 02.10.2021.

[26] *Property binding*, Angular. [Online]. Loetud aadressil: https://angular.io/guide/property-binding Kasutatud: 14.11.2021.

[27] *JsonPipe*, Angular. [Online]. Loetud aadressil: https://angular.io/api/common/JsonPipe Kasutatud: 21.11.2021.

[28] *Web Content Accessibility Guidelines (WCAG) 2.1*, World Wide Web Consortium, 2018. [Online]. Loetud aadressil: https://www.w3.org/TR/WCAG21 Kasutatud: 16.12.2021.

[29] *Testimine*, Majandus- ja Kommunikatsiooniministeerium, 2015. [Online]. Loetud aadressil: https://www.mkm.ee/et/eesmargid-tegevused/wcag-20-rakendusjuhised/testimine Kasutatud: 16.12.2021.

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

Mina, Danel Uukado

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kasutajaliidese stiilijuhiste näidislahenduse loomine tarkvaraarendusettevõttele“, mille juhendaja on Meelis Antoi
   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.

02.01.2022

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 – Rakenduse komponent: nupp

Fail button.component.ts:

```
import {Component, EventEmitter, Input, Output} from '@angular/core';

@Component({
  selector: 'app-button',
  templateUrl: './button.component.html'
})
export class ButtonComponent {
  @Input() public label: string;
  @Input() public type: 'primary' | 'secondary' | 'danger' | 'outlined' |
    'link' = 'primary';
  @Input() public disabled: boolean = false;
  @Input() public loading: boolean = false;
  @Output() public buttonClick: EventEmitter<void> =
    new EventEmitter<void>();

  public onButtonClick(): void {
    this.buttonClick.emit();
  }
}
```

Fail button.component.html:

```
<p-button
  [label]="label"
  [styleClass]="type === 'primary' ? '' : 'p-button-' + type"
  [loading]="loading"
  [disabled]="disabled"
  (onClick)="onButtonClick()"
>
</p-button>
```

Lisa 3 – Rakenduse komponent: märkeruut

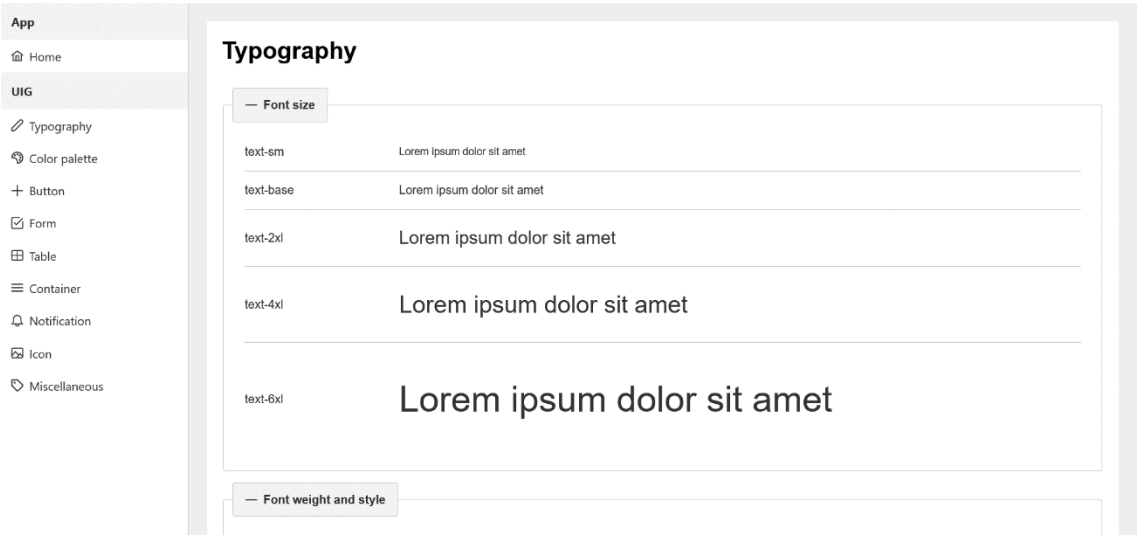
Fail checkbox.component.ts:

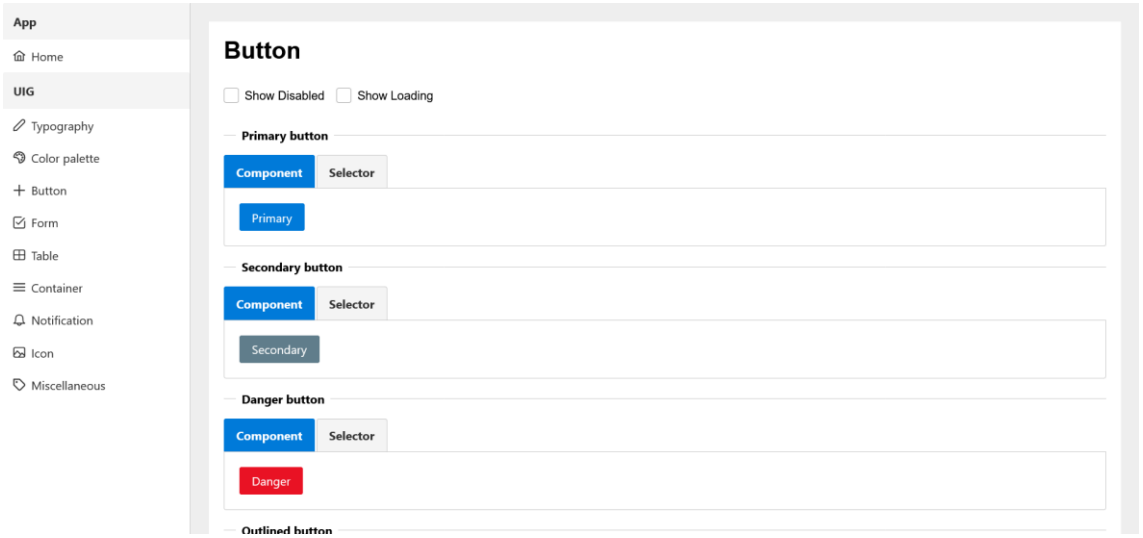
```
import {Component, forwardRef, Input} from '@angular/core';
import {ControlValueAccessor, NG_VALUE_ACCESSOR} from "@angular/forms";

@Component({
  selector: 'app-checkbox',
  templateUrl: './checkbox.component.html',
  providers: [
    {
      provide: NG_VALUE_ACCESSOR,
      useExisting: forwardRef(() => CheckboxComponent),
      multi: true
    }
  ]
})
export class CheckboxComponent implements ControlValueAccessor {
  @Input() public disabled: boolean = false;
  @Input() public label: string = 'Checkbox';

  public value: boolean = false;
  private onChange: (value: boolean) => void = (_) => {};

  public registerOnChange(fn: (value: boolean) => void): void {
    this.onChange = fn;
  }

  public registerOnTouched(fn: any): void {
  }

  public writeValue(value: boolean): void {
    this.value = value;
  }

  public fireOnChange(): void {
    this.onChange(this.value);
  }

}
```

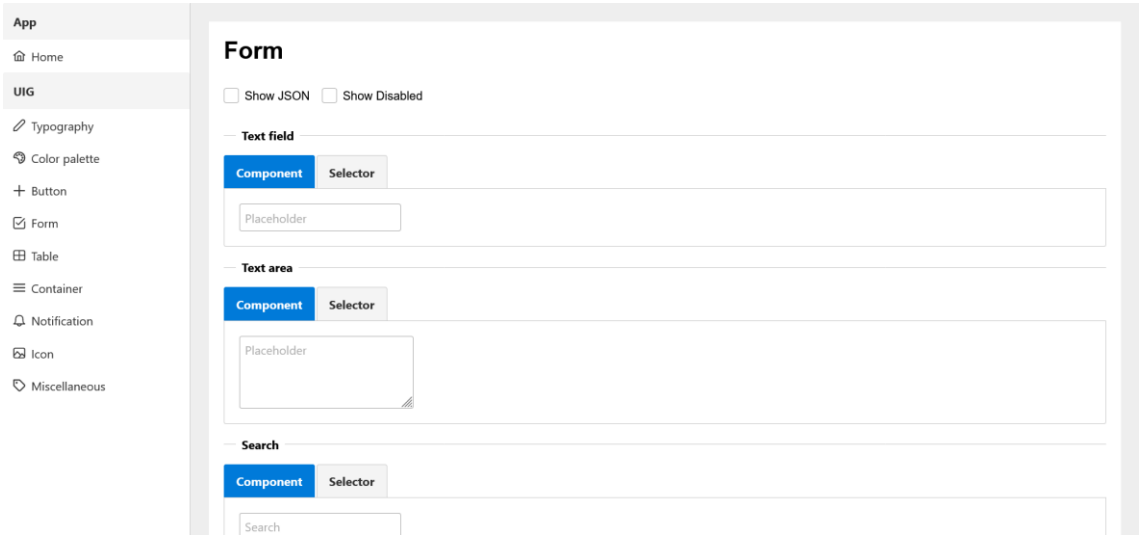
Fail checkbox.component.html:

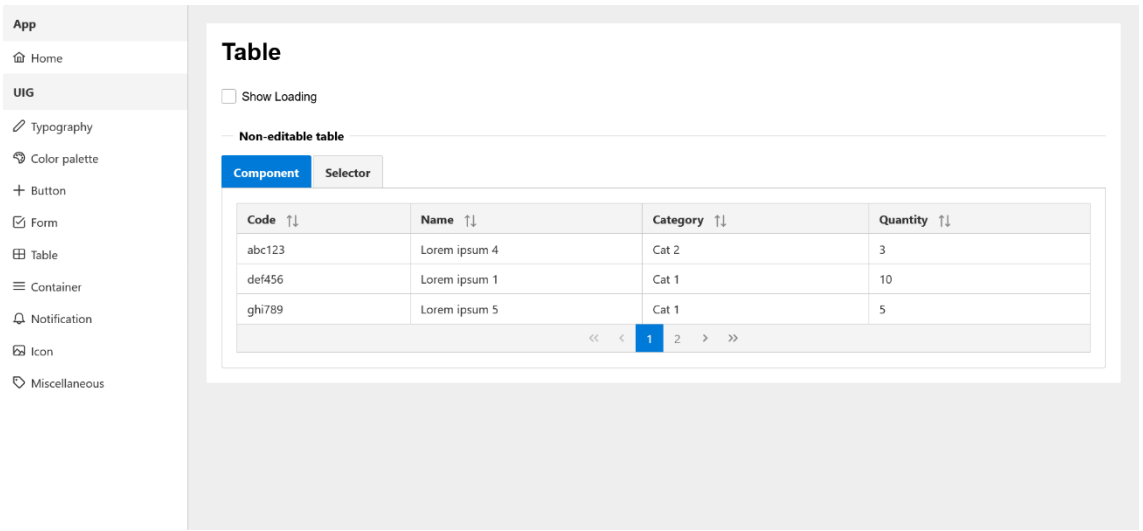
```
<p-checkbox
  [(ngModel)]="value"
  (ngModelChange)="fireOnChange()"
  [binary]="true"
  [label]="label"
  [disabled]="disabled"
>
</p-checkbox>
```

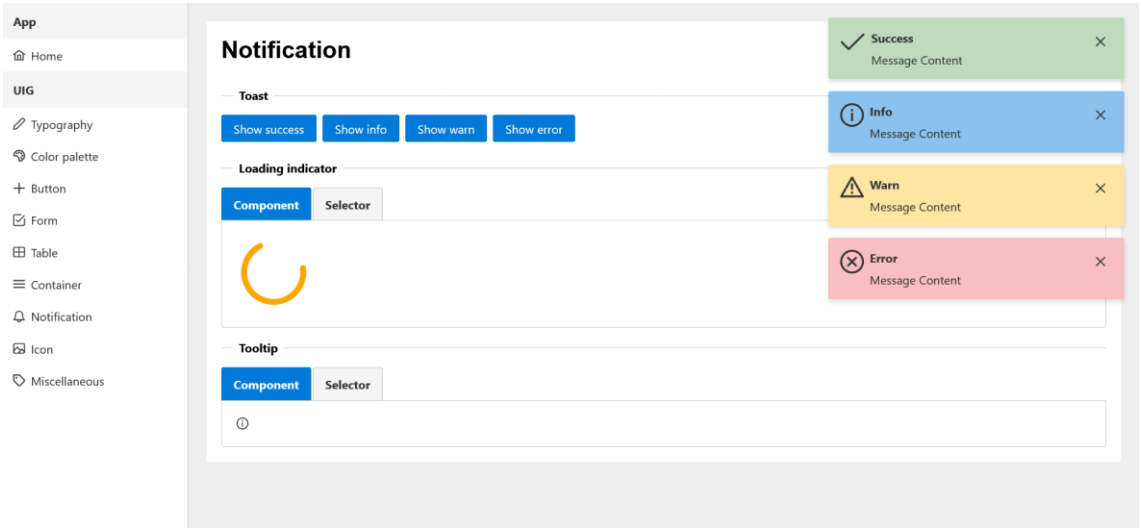
Lisa 4 – Veebirakenduse vaated

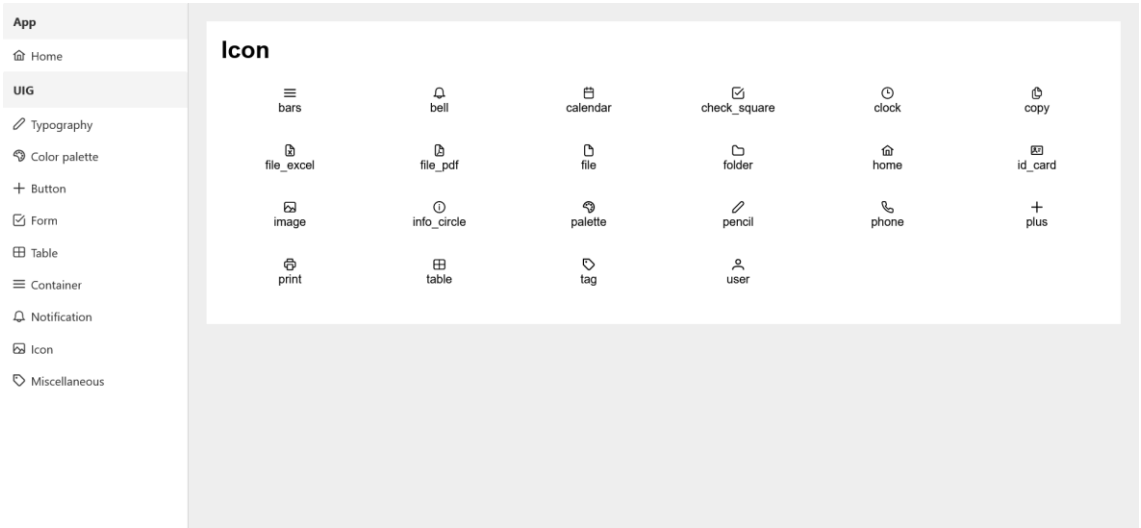


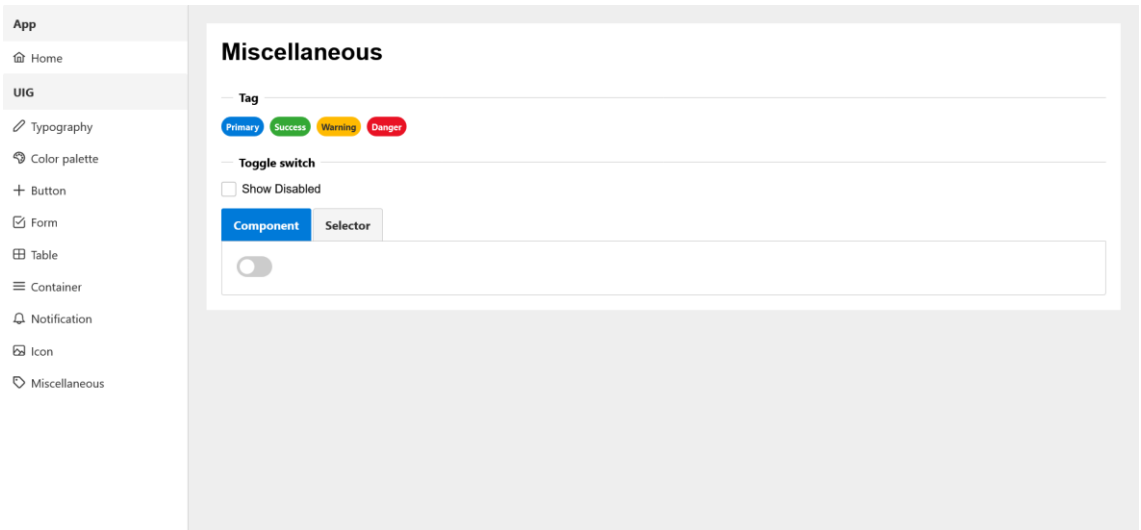

App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Button
Show Disabled
Show Loading
Primary button
Component
Selector
Primary
Secondary button
Component
Selector
Secondary
Danger button
Component
Selector
Danger
Outlined button


App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Form
Show JSON
Show Disabled
Text field
Component
Selector
Placeholder
Text area
Component
Selector
Placeholder
Search
Component
Selector
Search


App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Table
Show Loading
Non-editable table
Component
Selector
Code
Name
Category
Quantity
abc123
Lorem ipsum 4
Cat 2
3
def456
Lorem ipsum 1
Cat 1
10
ghi789
Lorem ipsum 5
Cat 1
5
1
2


App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Notification
Toast
Show success
Show info
Show warn
Show error
Loading indicator
Component
Selector
Tooltip
Component
Selector
Success
Message Content
Info
Message Content
Warn
Message Content
Error
Message Content


App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Icon
bars
bell
calendar
check_square
clock
copy
file_excel
file_pdf
file
folder
home
id_card
image
info_circle
palette
pencil
phone
plus
print
table
tag
user


App
Home
UIG
Typography
Color palette
Button
Form
Table
Container
Notification
Icon
Miscellaneous
Miscellaneous
Tag
Primary
Success
Warning
Danger
Toggle switch
Show Disabled
Component
Selector