TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kaisa Lindström 175931IDDR

VEEBILEHE JUURDEPÄÄSETAVUSE PARANDAMINE TPILETI RAKENDUSE NÄITEL

Diplomitöö

Juhendaja: Kaido Kikkas
Tehnikateaduse doktor

Tallinn 2020

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: Kaisa Lindström

27.04.2020

Annotatsioon

Käesoleva diplomitöö eesmärk oli parandada Tpileti rakenduse juurdepääsetavust ekraanilugejat kasutavatele nägemispuudega inimestele. 2019. aasta märtsis valminud Tpileti rakenduses ei olnud võimalik ekraanilugejaga piletit osta. Töö sisuks oli juurdepääsetavust parandavate tehniliste lahenduste väljatöötamine ja elluviimine.

Töö eesmärgi saavutamiseks kaardistati rakenduse juurdepääsetavuse probleemid, määratleti paranduste maht, konkreetsed eesmärgid ja arendati vajalikud lahendused. Töö tulemusena esitatakse juurdepääsetavust parandavad lahendused, mille loomisel järgiti WCAG (*Web Content Accessibility Guidelines*) nõudeid niivõrd, kui projekti piiratud maht seda võimaldas. Väljatöötatud lahenduste veebirakendusse lisamine võimaldab nüüd ekraanilugejaga pileti ostmise teekonda läbida. Rakenduse juurdepääsetavuse ja kasutatavuse täiendavaks hindamiseks on vajalik kasutajate tagasiside või veebirakenduse testimine koos nendega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 6 peatükki, 20 joonist.

Abstract

Improving Web Accessibility Based on the Example of Tpilet Ticket Service Application

The aim of the following thesis was to improve accessibility of Tpilet ticket service application. Tpilet application was released in March 2019, but due to accessibility issues an effective interaction with the website was not possible for screen reader users. The following thesis proposes and demonstrates technical solutions to improve web accessibility for screen reader users.

Web accessibility enables people with disabilities to use the web. In Estonia, people with disabilities make up about 12% of the population and due to population ageing this number will probably grow. The EU directives state that by 2018 all public sector websites need to meet certain accessibility requirements. Starting from 2025 this will also apply to important products and services provided by the private sector. The impact of enhancing web accessibility is even broader because accesibility improves usability for everyone.

The goal of the thesis was achieved by mapping accessibility problems, defining the range of needed improvements, and developing required solutions, which follow WCAG (Web Content Accessibility Guidelines) criteria. The results of the thesis are technical solutions, which upgraded web accessibility of Tpilet application. The evaluation of the results confirmed that applied solutions improved accessibility for screen reader users, making it possible to buy a ticket in the application. For the final evaluation of accessibility and usability of the application, it is recommended to conduct testing with users.

The thesis is in Estonian, and contains 30 pages of text, 6 chapters, 20 figures.

Lühendite ja mõistete sõnastik

ARIA	*Accessible Rich Internet Applications*, HTML-i täiendus, mis laiendab juurdepääsetavuse tagamise võimalusi
CSS	*Cascading Style Sheets*, kaskaadlaadistik. Veebisisu visuaalset esitlust kirjeldav keel
DOM	*Document Object Model*, dokumendi objektimudel. HTML dokumendi esitlus, mida saab JavaScripti (vmt skriptimiskeele) abil muuta
Ekraanilugeja	Riist- või tarkvara, mis teisendab digitaalse sisu kõneks või muuks juurdepääsetavaks infoks
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel
JavaScript	Programmeerimiskeel, mida kasutatakse peamiselt interaktiivsete veebirakenduste loomiseks
Punktkiri	Sõrmedega loetav reljeefsetel punktikombinatsioonidel põhinev kiri, mida kasutavad nägemispuudega inimesed
React	Kasutajaliideste loomiseks mõeldud JavaScripti teek
Ref	Kasutatakse Reactis DOM või React elementidele ligipääsemiseks
Teek	Funktsioonide kogum mõne komponendi loomiseks või tegevuse lihtsustamiseks. Üldjuhul luuakse teegid avalikuks kasutamiseks, mis võimaldab teiste arendajate poolt loodud koodi korduvkasutada
UML	*Unified Model Language*, süsteemi toimise loogikat kirjeldav visuaalne esitluskeel
WCAG	*Web Content Accesibility Guidelines*, veebisisu juurdepääsusuunised
WebAIM	*Web Accessibility In Mind*, juurdepääsetavuse teadmiste levitamiseks loodud organisatsioon
W3C	*World Wide Web Consortium*, veebikonsortsium

Sisukord

1 Sissejuhatus....9
2 Juurdepääsetavus....10
2.1 Teema olulisus....10
2.2 Veeb ja juurdepääsetavus....11
2.3 Euroopa Liidu õigusaktid....12
2.4 Olukord Eestis....12
3 Juurdepääsetavuse lähtekohad....15
3.1 Veebisisu juurdepääsusuunised....15
3.2 Ekraanilugeja tugitehnoloogia näitena....15
3.3 HTML ja ARIA....17
3.4 Juurdepääsetavuse järgimine arendusprotsessis....18
4 Tpileti rakendus....21
4.1 Ülevaade rakendusest....21
4.2 Juurdepääsetavuse probleemid....23
4.3 Paranduste maht....24
4.4 Protsessi kirjeldus....24
5 Tpileti rakenduse parandused....26
5.1 Metoodika....26
5.2 Klaviatuur....27
5.3 Fookuse muutmine....29
5.4 Mittetekstiline sisu....30
5.5 Vea tuvastamine....31
5.6 Navigeerimine....32
5.7 Järeldused....34
5.8 Võimalikud edasiarendused....35
6 Kokkuvõte....37
Kasutatud kirjandus....39
Lisa 1 – Ekraanipildid Tpileti rakenduse põhivaadetest....43

Lisa 2 – Pileti valiku komponendi koodinäide.....47
Lisa 3 – Kuupäeva valiku komponendi koodinäide.....48
Lisa 4 – Istekoha valiku komponendi koodinäide.....49

Jooniste loetelu

Joonis 1. UML - Kasutaja teekond pileti ostmisel.....22
Joonis 2. Nähtav fookus.....29
Joonis 3. Fookuse muutmise komponent.....30
Joonis 4. Fookuse muutmise komponendi kasutamine.....30
Joonis 5. Bussifirma nimetus ja logo.....31
Joonis 6. Ekraanilugejale teate edastamise komponent.....32
Joonis 7. Ekraanilugejale teate edastamiseks vajaliku komponendi kasutamine.....32
Joonis 8. "Liigu sisuni nupp“.....33
Joonis 9. "Liigu sisuni nupu" komponent.....33
Joonis 10. "Liigu sisuni nupu" komponendi kasutamine.....34
Joonis 11. Esilehekülg.....43
Joonis 12. Reiside loetelu.....43
Joonis 13. Pileti valiku paneel.....44
Joonis 14. Pikenev pileti valiku paneel.....44
Joonis 15. Istekoha valik pileti valiku paneelis.....45
Joonis 16. Ostukorvi paneel.....45
Joonis 17. Maksemeetodi valik ostukorvi paneelis.....46
Joonis 18. Pileti valiku komponent.....47
Joonis 19. Kuupäeva valiku komponent.....48
Joonis 20. Istekoha valiku komponent.....49

1 Sissejuhatus

Käesoleva diplomitöö eesmärgiks on parandada Tpileti veebirakenduse juurdepääsetavust. Tpilet tegeleb Eesti-siseste ja rahvusvaheliste bussipiletite müügiga pakkudes ühiskonnale olulist teenust. Tpileti tarkvara arendamise lähteülesanne nägi ette vastavuse osadele juurdepääsetavuse nõuetele, kuid erinevatel põhjustel ei arvestatud neid veebirakenduse arendamisel.

Tpiletil ei ole erafirmana kohustust enne 2025. aastat juurdepääsetavuse nõudeid järgida, kuid reaalsuses jätab nõuetele mittevastamine erivajadustega inimesed ilma teenuse kasutamise võimalusest. Tpileti 2019. aasta märtsis valminud uus veebirakendus ei võimaldanud ekraanilugejat kasutavatel nägemispuudega inimestel rakendust kasutada s.t nad ei saanud osta veebist bussipileteid.

Tpilet otsustas 2019. aasta suvel tellida arendustööd veebirakenduse juurdepääsetavuse parandamiseks. Esmaseks sihiks seati ekraanilugeja abil veebilehe kasutamisvõimaluse loomine. Tpileti veebirakenduse juurdepääsetavuse parandamine – konkreetsete lahenduste väljatöötamine ja tehniline elluviimine, milles käesoleva töö autor tarkvaraarendajana osales, on alljärgneva töö peamiseks sisuks. Lõputöös on esitatud Tpileti rakenduse parandamiseks väljatöötatud lahendused, mis näitavad, kuidas laiendada juurdepääsetavust HTML-i, ARIA ja JavaScripti õige kasutamise abil.

Käesolev diplomitöö koosneb neljast peatükist. Esimeses peatükis antakse ülevaade juurdepääsetavuse olulisusest, valdkonda reguleerivatest normidest ja veebilehtede juurdepääsetavusest Eestis. Teises peatükis tutvustatakse juurdepääsetavuse suuniseid, ekraanilugeja toimispõhimõtteid, semantilise HTML-i olulisust ja juurdepääsetavuse tagamist arendusprotsessis. Kolmandas peatükis kirjeldatakse Tpileti rakendust, kaardistatakse juurdepääsetavuse probleemid ja esitletakse lahenduste leidmise protsessi. Neljandas peatükis tuuakse välja rakenduse parandamiseks leitud lahendused, järeldused ja võimalikud edasiarendused.

2 Juurdepääsetavus

Peatükk annab ülevaate juurdepääsetavuse olulisusest, valdkonda reguleerivatest normidest ja veebilehtede juurdepääsetavusest Eestis.

2.1 Teema olulisus

Juurdepääsetavuse peamine eesmärk on võimaldada puuetega inimeste kaasamist infokeskkonda ja ühiskondlikku ellu. Eestis moodustab puuetega inimeste osakaal ligi 12% ja on tõenäoline, et rahvastiku vananemisel see protsent kasvab [1] . On prognoositud, et võttes arvesse vananemist on 2020. aastal ligikaudu 120 miljonil inimesel Euroopa Liidus mitu puuet ja/või kerge puue [2] .

Mõiste „puue“ on ajas muutunud ega väljenda enam üksnes inimese tervislikku seisundit. Eesti seadusandluse järgi on puue “… inimese anatoomilise, füsioloogilise või psüühilise struktuuri või funktsiooni kaotus või kõrvalekalle, mis koostoimes erinevate suhtumuslike ja keskkondlike takistustega tõkestab ühiskonnaelus osalemist teistega võrdsetel alustel” [3] .

Puudega inimeste ühiskonnaelus osalemise piirangute vähendamisel on suurt rolli mänginud tehnoloogia. Algselt vaid puudega inimeste vajadustest lähtunud tehnilised lahendused on mõjutanud ning muutnud ühiskonda laiemalt. Näiteks sai esimese töötava trükimasina leiutamine 19. sajandi alguses tõuke Pellegrino Turri soovist pidada kirjavahetust oma pimeda sõbrannaga [4] . Telefoni leiutaja ja teadlase, Alexander G. Belli huvi kõnetehnika ning kõne- ja kuuldeaparaatide vastu mõjutas asjaolu, et tema abikaasa ja ema olid kurdid [5] .

Aina rohkem on puudega inimeste tegutsemisvõimalusi avardanud arvuti, mida on võimalik juhtida hääle, silmade või mõne muu kehaosa abil. Üheks tuntumaks näiteks

oli inglise füüsik ja kirjanik Stephen W. Hawking, kes sai arvutit juhtida oma põse abil. Veebilehtede ja -rakenduste juurdepääsetavuse parandamine annab erivajadustega inimestele ühiskonnaelus osalemiseks vajalikud võimalused palju laiemalt [6] .

Juurdepääsetavuse nõuete rakendamisest on kasu ka erivajadusteta inimestele. Kui erivajadusega kasutaja jaoks tähendab juurdepääsetavuse nõuete rakendamine teenuse kasutamisvõimaluse tagamist, siis erivajaduseta kasutaja jaoks võib see suurendada kasutatavust (inglise keeles *usability*). Piisav kontrastsus aitab eredas valguses ekraani paremini lugeda, tänu subtiitritele on mürarikkas bussis võimalik loengut kuulamise asemel lugeda, autojuhil on häälkäskluste abil nutitelefoni kasutamine palju ohutum. Eelnimetatud, ja teised vahendid, võivad olla ainukeseks võimaluseks puuetega inimestele, kuid avardavad ka kõigi teiste kasutajate võimalusi veebirakendustega suhtlemiseks [7] .

Kasutatavus on laiem valdkond, mille sisse kuulub ka juurdepääsetavus, kuid tihti ei arvestata kasutatavuse parandamisel puuetega inimeste vajadusi. Veebirakenduste lahendused peaksid arvestama mõlema teguriga. Juurdepääsetavuse ja kasutatavuse kombinatsiooni nimetatakse kasutatavaks juurdepääsetavuseks (inglise keeles *usable accessibility*) või juurdepääsetavaks kasutajakogemuseks (inglise keeles *accessible user experience*) [7] .

2.2 Veeb ja juurdepääsetavus

Veebiga on lahutamatult seotud kõikehõlmavuse ja juurdepääsetavuse põhimõtted. Tim Berners-Lee, veebi looja, sõnastas selle väga elegantselt juba 1997. aastal: „*The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.“* [8] .

Kuid veebisisu juurdepääsetavus ei sünni iseenesest, kui sellega ei ole teadlikult arvestatud. See nõuab mitmekülgset ja eesmärgipõhist tegevust. Selleks on vaja nii poliitilist tahet kui ka kokkulepitud standardeid. Esimese teedrajava saavutuseni jõuti 1999. aastal, kui Veebikonsortsium (*World Wide Web Consortium,* edaspidi W3C*)*

avaldas veebisisu juurdepääsuunised (*Web Content Accessibility Guidelines,* edaspidi WCAG ja WCAG suunised) [9] , mis on kogum nõuetest, millele peab veebilehekülg vastama, et tagada lehe juurdepääsetavus. W3C juurdepääsusuuniste sisu on üle võetud ka õiguslikult siduvatesse standarditesse.

2.3 Euroopa Liidu õigusaktid

2016. aastal jõustunud Euroopa Liidu direktiiv nägi ette, et alates 2018. aastast peavad avaliku sektori veebilehed vastama kindlatele juurdepääsetavuse nõuetele [10] . Vastavalt 2019. aastal vastuvõetud direktiivile laieneb see kohustus alates 2025. aastast ka era- ja avaliku sektori ettevõtetele, mis pakuvad ühiskonnale olulisi teenuseid, näiteks pangad ja transpordifirmad [11] .

Alates 2018. aastast on Euroopa Liidus juurdepääsetavuse tagamiseks kasutusel harmoniseeritud Euroopa standard EN 301 549 V2.1.2 (2018-08) [12] . Antud standard puudutab laiahaardeliselt info- ja kommunikatsioonitehnoloogiat, näiteks pangaautomaate, printereid, mobiiltelefone, tarkvara, elektroonilisi dokumente. Veebisisu osas viitab standard WCAG 2.1 versiooni nõuetele. Erinevalt W3C veebisisu juurdepääsusuunistest on Euroopa standard, vastavalt kehtestatud ulatusele, järgimiseks kohustuslik.

2.4 Olukord Eestis

Eestis on juurdepääsetavuse valdkonnaga tegeletud alates 1990. aastate algusest. Käesoleva lõputöö juhendaja Kaido Kikkas kaitses puuetega inimesi ja infotehnoloogiat käsitleva magistritöö 1994. aastal (ilmunud monograafiana 1995. aastal) [13] ja doktoritöö 1999. aastal [14] . Kaido Kikkas oli ka üks Tallinna Tehnikaülikooli rehabilitatsioonitehnoloogia labori asutajatest, mis tegutses aastatel 1995-2002 [15] . Veebilehtede juurdepääsetavusega on viimastel aastatel tegeletud ka magistri- ja bakalaureusetööde raames. Triinu Tamberg on oma magistritöös analüüsinud juurdepääsetavusnõuete arvestamist arendusprotsessis [16] , Mari-Ell Mets esitles oma

bakalaureusetöös suunised juurdepääsetavuse tagamiseks puutetundlikel mobiilseadmetel [17]. Kärolin Kivisikk on oma töös uurinud veebikeskkondade vastavust WCAG 2.0 nõuetele [18] ja Oliver Ainsalu on käsitlenud veebilehtede käideldavuse ja kasutatavuse hindamist [19].

Eestis peavad alates 2012. aastast avaliku sektori veebilehed vastama juurdepääsetavuse nõuetele [20]. Kuid kui ühiskonnas ei ole kaasava ühiskonna põhimõtted levinud ei pruugi seadusest piisata. Seda võib järeldada 2015. aasta uuringust „Avaliku sektori veebilehtede vastavus WCAG 2.0 nõuetele“ [21], milles leiti, et enamik avaliku sektori lehti ei vastanud vajalikele nõuetele. Nii nagu linnades ei piisa, et ligipääsetavad on vaid üksikud hooned, nii on ka veebis oluline lehtede üldine juurdepääsetavuse tase. Tervikuna on veebilehtede juurdepääsetavust Eestis siiski vähe uuritud, mistõttu on keeruline hinnata avaliku sektori ja oluliste teenuste veebilehtede tänast juurdepääsetavuse taset. Loodetavasti toovad suurema tähelepanu kaasa Euroopa Liidu juurdepääsetavusega seotud direktiivid.

Põhjuseid, miks senini ei ole veebi juurdepääsetavust laialdaselt rakendatud, saab välja tuua peamiselt oletuste tasemel. WCAG standard on mahukas ja tehniline dokument. Mari-Ell Mets on oma bakalaureusetöös leidnud, et tarkvaraarendajad peavad WCAG dokumendist vajaliku info üles otsimist keeruliseks [17]. Samas on viimasel ajal loodud palju uut õppematerjali, mis lihtsustab juurdepääsetavuse nõuetest arusaamist.

Teadlikkus kaasava ühiskonna olulisusest on Eestis kasvanud, kuid üsna visalt. Nii on ka veebi juurdepääsetavuse puhul tõenäoline, et üheks põhiliseks probleemiks on teadmatus ja ühiskondlike väärtuste eiramine. Puudub teadlikkus, et veebilehtede arendamisel tuleb arvestada juurdepääsetavuse nõuetega. Teadmatus juurdepääsetavusega mittearvestamise tagajärgedest võib väljenduda ka piiratud aja- ja rahaliste ressurside tõttu tehtavates valikutes, kui juurdepääsetavusega seonduv jäetakse tagaplaanile.

Juurdepääsetavuse nõuete rakendamisest ja nende järgimisest veebirakenduste arendamise käigus võidavad lõppkokkuvõttes kõik. Juurdepääsu võimaldamine

suurendab puuetega inimeste võimalusi ühiskonnaelus aktiivselt osaleda. Ülejäänud kasutajate jaoks tähendab nõuete järgimine suuremat kasutusmugavust. Ettevõtete jaoks tähendab see potentsiaalselt suuremat kasutajate arvu (Euroopa tasemel on jutt vähemalt kümnetest miljonitest inimestest) ja positiivset mõju nende kuvandile.

3 Juurdepääsetavuse lähtekohad

Käesolevas peatükis antakse ülevaade veebisisu juurdepääsusuunistest, ekraanilugeja toimimisest, semantilise HTML-i olulisusest ja juurdepääsetavuse arvestamisest arendusprotsessis.

3.1 Veebisisu juurdepääsusuunised

WCAG suunised lähtuvad inimeste vajadustest, kellel võivad esineda häireid kuulmises, motoorikas, kõnes, nägemises või kognitiivsetes funktsioonides. Veebirakenduste arendamisega seotud nõuded on eelkõige seotud veebisisu, kujunduse või koodiga [22] . Kujundusega seotud nõuded puudutavad näiteks minimaalset teksti suurust või värvi kontrasti. Koodi nõuded on enamjaolt seotud semantilise HTML-i või ARIA (*Accessible Rich Interface Applications*) kasutamisega. Tugitehnoloogia toimimine, millele paljud erivajadustega inimesed arvuti ja veebi kasutamisel toetuvad, sõltub eelkõige HTML-i semantikast [23] .

WCAG dokument püüab leida tasakaalu juurdepääsetavuse ja kasutusmugavuse vahel esitades nõuete täitmiseks kohustuslikud ja soovituslikud tehnikad. Kohustuslikud tingimused on suunatud juurdepääsetavuse tagamisele, soovituslikud tehnikad kasutusmugavuse parandamisele. Nõuded jagunevad kolme tasemesse – A (madalaim), AA ja AAA (kõrgeim). Üldjuhul peetakse veebilehte juurdepääsetavaks, kui see vastab AA taseme nõuetele. Viimane WCAG versioon 2.1 avaldati 2018. aastal [22] .

3.2 Ekraanilugeja tugitehnoloogia näitena

Tugitehnoloogia on tarkvara või riistvara, mis aitab erivajadustega inimestel kasutada arvutit või veebi [24] . Tugitehnoloogia otstarve sõltub konkreetsest puudest, mistõttu

tugitehnoloogia võib olla väga erisugune. Alates kohandatud klaviatuurist või käetoest kuni silmade jälgimise tehnoloogia ja häältuvastusvahendini välja [13]. Nägemispuudega inimestele on abiks ekraanilugeja, mis tõlgib digitaalse sisu heli ja/või punktkirja vormi. Näiteks veebisisus leiduva „nupp“ elemendi korral teatab ekraanilugeja kasutajale, et asub nupu peal, nupu nimetus on X ja nuppu saab aktiveerida vajutades järgmisi klahve [25] .

Ekraanilugejat saab kasutada nii arvutis kui ka nutitelefonis. Erinevaid ekraanilugejaid on palju. Näiteks macOS operatsioonisüsteemiga seadmetel (nii telefonis kui ka arvutis) on sisse-ehitatud ekraanilugeja VoiceOver, Windows operatsioonisüsteemil saab kasutamiseks alla laadida JAWS-i (tasuline) või NVDA (vabavara). Linux operatsioonisüsteemile on loodud Orca-nimeline ekraanilugeja ja Android nutitelefonis on sisse-ehitatud TalkBack. Laua- või sülearvutis saab ekraanilugejale anda käsklusi tavalise klaviatuuri abil, nutitelefonis puudutusega ja näpuga ekraanil paremale või vasakule liikudes (inglise keeles *swipe*).

WebAIM (*Web Accessibility In Mind*) 2019. aasta uuringu kohaselt on 1224 vastanu seas kõige populaarsemad ekraanilugejad NVDA (41%), JAWS (40%) ja VoiceOver (13%) [26] . Samal aastal viidi Eestis läbi 29 vastanuga küsitlus, kellest 68% eelistas ekraanilugejat JAWS, 65% VoiceOverit ja 13% NVDA-d [27] . Tihti kasutatakse ekraanilugejat koos kindla brauseriga. WebAIM-i uuringu kohaselt on kõige populaarsemad kombinatsioonid järgmised: JAWS ja Chrome (21%), NVDA ja Mozilla FireFox (20%), NVDA ja Google Chrome (18%), JAWS ja Internet Explorer (12%) ning VoiceOver ja Safari (9%) [26] . Eestis tehtud uuring ekraanilugeja ja brauserite eelistusi ei kajasta, aga tuuakse välja, et kõige populaarsemad brauserid on Internet Explorer (68%), Google Chrome (65%), Safari (32%) ja Mozilla FireFox (23%) [27] . Erinevad ekraanilugejate ja brauseri kombinatsioonid võivad toimida erinevalt, mistõttu on tarkvaraarenduse seisukohalt oluline teada eelistatud variante, et saaks keskenduda kõige olulisemate kombinatsioonide toimimisele ja testimisele.

3.3 HTML ja ARIA

Kõige uuem HTML versioon on HTML5, mis sisaldab muuhulgas palju uusi semantilisi elemente. HTML5-ga lisandus mitmeid elemente, mis aitavad senisest paremini sisu struktureerida tänu sellele, et kirjeldavad ka objekti enda omadusi. Näiteks navigatsioonimenüü (<*nav*>), artikkel (<*article*>), sektsioon (<*section*>) jpt [28]. HTML-i spetsifikatsioon kirjeldab, millistest elementidest võib veebilehtede struktuur koosneda, mis on nende elementide tähendus ja omadused. Spetsifikatsioon määrab millisesse kategooriasse element kuulub, millises olukorras võib elementi kasutada, kas ja millised lapselemendid (inglise keeles *child elements)* võivad elemendil olla, kas ja kuidas saab elementi aktiveerida, mis on elemendi parameetrid (inglise keeles *attribute*), sealhulgas ARIA parameetrid [29] .

HTML-i suhteline lihtsus on võimaldanud erineva tasemega veebiarendajatel veebilehti luua, kuid selle tagajärjel kasutatakse HTML-i elemente tihtipeale valesti. Seda olukorda on omakorda soodustanud brauserid, mis on kohandatud töötlema vigast HTML-i [30] . Olukorda muudab keerulisemaks see, et HTML-i DOM-i on võimalik JavaScriptiga muuta. Üks levinud viga on üldelemendi <*div*> kasutamine nupu <*button*> asemel. Üldelemendile lisatakse JavaScriptiga nupule sarnane aktiveerimise funktsionaalsus <*div onClick="..."*>. Kuid selline lahendus toimib üksnes hiirega vajutamise korral ega ole klaviatuuriga kasutatav. See tähendab, et üksnes klaviatuuri kasutavatele inimestele, näiteks ekraanilugeja kasutajatele, jääb element juurdepääsmatuks. JavaScriptiga on võimalik ka klaviatuuri kasutamise funktsionaalsus kaasa anda, kuid tihti seda ei tehta. Interaktiivsetele HTML elementidele (nupp <*button*>, hüperlink <*a*> jt) on juba sisse-ehitatud klaviatuuri juurdepääsetavus. Näiteks lahenduse <*button onClick="..."*> puhul aktiveeritakse *onClick* funktsioon nii hiireklõpsatuse kui ka klaviatuuril tühik või sisestusklahvi vajutamise peale.

Semantiline HTML loob vundamendi juurdepääsetavuse tagamisel, kuid kõiki veebiarenduse ja juurdepääsetavuse tagamisega seotud väljakutseid üksnes HTML-ga lahendada ei saa. Sel põhjusel arendas W3C täienduseks HTML-le ARIA (*Accessible Rich Interface Applications*). ARIA on täiendavate parameetrite kogum, mida saab lisada HTML elementidele. ARIA viimane versioon 1.1 avaldati 2017. aastal. Tänu

ARIA kasutamisele on tugitehnoloogiale võimalik anda lisateavet HTML-i elementide kohta (nt *role="menu"* annab kasutajale märku, et tegu on menüüga), omaduse (nt *aria-required="true"* märgib, et väli on kohustuslik) ja oleku (nt *aria-expanded="true"* viitab, et element on avatud) kohta [31] . Nii nagu HTML-i puhul on ka ARIA puhul oluline selle õige kasutamine. ARIA-t tuleb ja tohib kasutada vaid juhul, kui HTML-ist ei piisa [32] . Üldjuhul tuleb veebiarendajal ARIA kasutamisel korral elemendi korrektse funktsioneerimise nimel rohkem vaeva näha ning ARIA toimimine eritehnoloogiates võib olla ebaühtlane, mistõttu tuleb seda hoolikalt testida. HTML-i kasutamine peaks olema alati esimene valik, sest elementidele sisse-ehitatud funktsionaalsus on ettearvatav ja usaldusväärne [33] .

Tänu semantilisele HTML-le ja ARIA-le on võimalik ka väga keerulised ning dünaamilised kasutajaliidesed juurdepääsetavaks teha. HTML-i semantika korrektne kasutamine omab tegelikult veel laiemat mõju. Ka otsingumootorid ja hääljuhitavad seadmed sõltuvad HTML-i semantikast. Mida semantilisem on HTML, seda lihtsam on vajalikku infot masintöödelda ja esitada.

3.4 Juurdepääsetavuse järgimine arendusprotsessis

Juurdepääsetavusega tuleb tegeleda kogu projekti kestel ehk nii analüüsi-, disaini-, arendus- kui ka haldamisfaasis. Tänu järjepidevusele võib avastada kriitilisi probleeme või neid ära hoida. Tagantjärele parandamine on märksa kallim ja keerulisem. Juurdepääsetavuse tagamine alates projekti algusfaasist võimaldab luua lahendusi, mis on paremini kasutatavad nii puuetega inimestele kui ka laiemale kasutajaskonnale [34] .

Projekti alustamisel tuleb sõnastada eesmärgiks olev juurdepääsetavuse tase, valida või töötada välja selle saavutamiseks vajalik metoodika. Samuti tuleb kokku leppida kriteeriumid, mille alusel hinnata eesmärkide saavutamist.

Juurdepääsetavuse planeerimisel soovitatakse kasutada persoona spektrit (inglise keeles *persona spectrum*), mis arvestab üheaegselt püsiva, lühiajalise ja olukorrast (nt haigus või trauma) tingitud puudega kasutajaid. Näiteks on püsiv puue vaid ühe käe

funktsioneerimine, lühiajaline puue võib olla käeluumurd ja olukorrast tingitud puue sülelapse käes hoidmine. Lühiajalise või olukorrast tingitud puudega inimeste hulk on palju suurem kui püsiva puudega inimeste arv. Seega on juurdepääsetavuse ja kasutusmugavuse suurendamine kasulik palju laiemale kasutajaskonnale, mitte ainult püsiva puudega inimestele [35] .

Juurdepääsetavuse kontrollimiseks on välja arendatud suur hulk erinevaid tööriistu, mida saab kasutada nii disaini- kui ka arendusfaasis [36] . Tööriistu on lihtne kasutada ja selleks pole vaja eelteadmisi. Juurdepääsetavuse tööriistad aitavad kiiresti vigu avastada. Automaattestid kontrollivad HTML-i ja ARIA korrektsust, teksti loetavust jpm. Üldiselt annavad tööriistad probleemi markeerimise kõrval ka soovitusi parandamiseks. Ühed kõige populaarsemad tööriistad on WAVE (*Web Accessibility Evaluation Tool*) ja aXe.

Arendusfaasis on olemasolevad tööriistad isegi olulisemad kui disainifaasis. Lisaks tööriistadele on kaasaegsete kliendirakenduste arendamiseks loodud mitmeid teeke, näiteks *eslint-plugin-jsx-a11y*, *react-a11y* ja *pa11y*. Erinevalt tööriistadest otsivad teegid vigu kogu koodibaasist ühekorraga. Ka on välja arendatud teegid, näiteks *axe-core*, mida saab integreerida kasutusel oleva testimisraamistikuga. Integreeritavad teegid võimaldavad kasutaja sisendist sõltuvate lahenduste juurdepääsetavust kontrollida. Näiteks olukorrad, kus testitakse vormi saatmist või mõnele toimingule tagasiside andmist. Integreeritud automaattestid aitavad ära hoida olukordi, kus funktsioneeriv ja juurdepääsetav arendus muutub lisaarenduste käigus juurdepääsmatuks.

Automaattestimine, kuigi kiire ja ressursse kokkuhoidev, ei ole piisav juurdepääsetavuse taseme hindamiseks, sest kõiki vigu ei saa avastada automaattestimise abil. Sel põhjusel on oluline lisaks rakendada manuaalset ja kasutajatega testimist. Manuaalse testimise eesmärk on kontrollida, kas digitaalne sisu on arusaadav ja asjakohane kasutajale. Näiteks kui on pilt, mille tekstiline alternatiiv tugitehnoloogia kasutajale on kirjeldus, siis on vaja inimese hinnangut, kas pildi kirjeldus on piisav iseloomustamaks pildil kujutatut. Manuaalne testimine on ressursimahukas, kuid vältimatu vahend juurdepääsetavuse testimisel.

Üks oluline manuaalse testimise viisidest on *screening*, mis tähendab, et (tugitehnoloogiatega) testimisel piiratakse või muudetakse vähemalt ühte füüsilist või aistingulist võimekust. Näiteks ekraanilugejaga testimisel lülitatakse kuvar välja. Tuleb arvestada, et tugitehnoloogia vähese kasutamisoskuse tõttu võib *screening*-meetodil põhinev testimine anda ebatäpse tulemuse. Samas aitab *screening*-testimine hinnata, kas valdav osa lahendusest on tugitehnoloogiaga kasutatav. Kui lahenduse funktsionaalsus on olulisel määral juurdepääsetav, siis on järgmine loogiline samm kasutajatega testimine, mille eesmärk on parandada kasutusmugavust [34]. Kasutatava juurdepääsetavuse tagamiseks on kõige tulemuslikum kõigi kolme meetodi – automaattestimise, manuaalse testimise ja kasutajatega testimise – kombineerimine.

Tarkvara uuenduste puhul on oluline järgida lähteülesandes seatud eesmärke tervikuna ning mitte piirduda vaid ühe või teatud probleemide lahendamisega. Selleks tuleb nii uuenduste loomise käigus kui ka töö lõpus kasutada juurdepääsetavuse automaatteste ja võimalusel ka teste kasutajatega. Ebaõnnestunud haldamise näiteks on 2018. aastal valminud DigiDoc4 tarkvara, mis ei toetanud ekraanilugejaid seitsme esimese kuu vältel alates tarkvara väljalaskmisest [37]. Taolisi probleeme aitaks ära hoida projekti integreeritud automaattestid, mis ei lubaks vigast koodi esitada.

W3C soovitab juurdepääsetavuse tagamisel kaasata protsessi ka puuetega inimesed, kellega koos lahendust välja töötatakse ja/või testitakse. W3C hoiatab, et kui juurdepääsetavusele lähenetakse vaid tehnilise külje alt ja kasutuskogemusele tähelepanu ei pöörata, siis tõenäoliselt juurdepääsetavust ei tagata. Suuniste järgimine ja kasutajate kaasamine aitab paremini saavutada peamist eesmärki: võimaldada puuetega inimestel kasutada veebi ilma takistusteta [7].

4 Tpileti rakendus

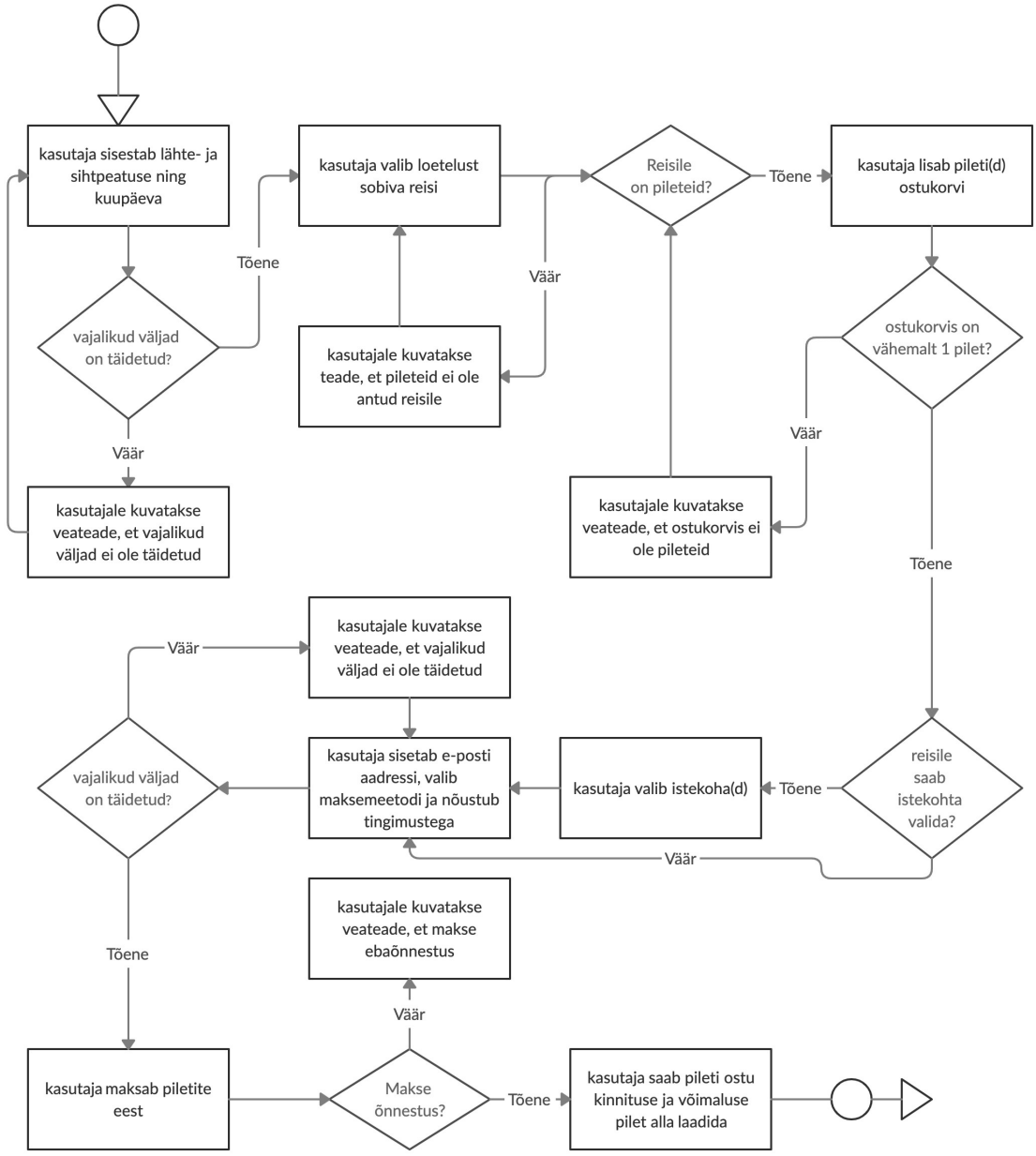
Käesolevas peatükis tutvustatakse Tpileti veebirakendust, kaardistatakse Tpileti juurdepääsetavuse probleemid, määratletakse parandamise maht ja kirjeldatakse lahenduste leidmise ning testimise protsessi.

4.1 Ülevaade rakendusest

Tpilet tegeleb Eesti-siseste ja rahvusvaheliste bussiliinide piletite müügiga. Kodulehel on võimalus osta või tagastada piletit, saada teavet sõiduplaanide kohta, sõlmida kliendileping ja jälgida Tpileti postitatud uudiseid. Rakendust on võimalik kasutada veebibrauserist nii arvutis kui ka mobiilis.

Tpileti kõige olulisem funktsionaalsus on seotud pileti ostmisega. Kasutaja teekond pileti ostmisel on välja toodud UML (*Unified Model Language*) skeemina joonisel 1. Samuti aitavad rakendust paremini mõista lisas olevad kuvatõmmised (Lisa 1 – Ekraanipildid Tpileti rakenduse põhivaadetest).

Tpileti veebirakendus järgib teenuspõhise arhitektuuri mudelit, kus klient ja server on teineteisest eraldatud ning suhtlevad omavahel kokkulepitud rakendusliidese ehk API (*Application Programming Interface*) abil. Serveri- ja andmebaasi pool tegeleb andmete hoidmise ja töötlemisega ning kliendiks nimetatakse liidest, mille abil saab rakendust kasutada. Juurdepääsetavus sõltub seega vaid kliendirakendusest. Tpileti kliendipoolne rakendus on üheleherakendus, mis tähendab, et veebisisu laetakse dünaamiliselt kliendi poolel. Tänu sellele on päringuid serverile vähem ja kasutaja perspektiivist tähendab see rakenduse kiiremat reageerimist erinevatele toimingutele. Klientrakendus on ehitatud kasutades JavaScriptil põhinevat teeki ReactJS ja andmetega tegelemiseks teeki nimega Redux. Veebirakendus toetab järgmisi brausereid: Google Chrome, FireFox, Safari, Edge ja Internet Explorer (versiooni 11).



Joonis 1. UML - Kasutaja teekond pileti ostmisel

4.2 Juurdepääsetavuse probleemid

Tpileti veebirakenduse juurdepääsetavuse probleemid toodi esile ühe kasutaja poolt esitatud kaebuses. Olukorrast parema ülevaate saamiseks korraldas Tpilet kaebuse esitaja ja projekti tarkvaraarendaja vahel kokkusaamise, kus üheskoos kaardistati pileti ostu teekonnal esinevad probleemid. Rakenduse hindamisel kasutati kasutaja eelistatud

tehnoloogiat – ekraanilugejat JAWS ja Internet Explorer brauserit. Järgnevalt on välja toodud nimekiri peamistest probleemidest lehekülgede kaupa.

Esileheküljel:

- kasutajal ei ole võimalik kuupäeva valida;
- kasutajal ei ole võimalik reiside otsimise nupule vajutada;
- kasutajat ei teavitata veateadete puhul;
- kasutaja viiakse automaatselt uuele lehele, kui kõik väljad on täidetud.

Reiside loetelu lehel:

- kasutajat ei teavitata uue lehekülje laadimisest;
- kasutajal ei ole võimalik soovitud reisile piletit valida;
- kasutajat ei teavitata pileti valiku paneeli laadimisest.

Pileti valiku paneelis:

- kasutajal ei ole võimalik piletit ostukorvi lisada;
- kasutajal ei ole võimalik istekohta valida;
- kasutajal ei ole võimalik ostukorvi viiva nupule vajutada.

Ostukorvi paneelis:

- kasutajat ei teavitata uue sisu laadimisest paneeli;
- kasutajat ei teavitata veateadete puhul;
- kasutajal ei ole võimalik maksemeetodit valida;
- kasutajal ei ole võimalik nõustuda pileti ostutingimustega.

Muud tähelepanekud:

- kasutajat ei teavitata ostukorvi aegumise korral.

4.3 Paranduste maht

Juurdepääsetavuse parandamisel seati eesmärgiks lahendada kasutaja välja toodud probleemid ja luua võimalus ekraanilugejaga pileti ostmiseks. Juurdepääsetavus on palju laiem teema, kuid rahaliste võimaluste piirangute tõttu otsustati luua töötav lahendus ekraanilugeja kasutajatele.

4.4 Protsessi kirjeldus

Projekti meeskonnas ei olnud tarkvaraarendajat, kes oleks varasemalt juurdepääsetavuse temaatikaga tuttav. Juurdepääsetavuse parandamisele keskendus töö autor, kellel tuli end kurssi viia juurdepääsetavuse temaatikaga ja seejärel välja töötada lahendusvõimalused, nende teostamiseks vajalik metoodika ja programmeerida juurdepääsetavust tagavad koodimuudatused. Paar juurdepääsetavuse paranduse lahendust teostas projekti juhtivarendaja, ülejäänud lahenduste eest vastutas autor.

Juurdepääsetavuse teemat ja praktikaid tutvustavad spetsiaalselt tarkvaraarendajatele suunatud materjalid, näiteks Google Developers [38] ja Mozilla Developer Network [39] poolt loodu. Juurdepääsetavust tuvustab ka Google'i loodud veebikursus Udacity platvormil [40] ja Google'i arendaja Rob Dodsoni avaldatud videomaterjalid, kus muuhulgas õpetatakse ka erinevate ekraanilugejate kasutamist [41] .

Peamistest juurdepääsetavuse põhimõtetest ja nende tagamise viisidest arusaamine on vajalik alus WCAG suuniste sisuliseks mõistmiseks. WebAIM on loonud WCAG suuniste põhjal lihtsas ja arusaadavas keeles kokkuvõtte [42] , mis on suureks abiks mahukate ja tehniliste WCAG dokumentide mõistmisel. Lisainfot on võimalik leida W3C poolt avaldatud abimaterjalis, mis selgitab detailselt ja arusaadavalt iga nõude tausta, vajalikkust, näiteid ja võimalikke lahendusi [43] . Läbitöötatud materjalide põhjal koostas käesoleva töö autor ettevõte sisemiseks kasutamiseks mõeldud juurdepääsetavuse tagamise juhised tarkvaraarendajatele.

Juurdepääsetavuse parandamisel saab olla edukas vaid juhul, kui osatakse mõista, miks kasutajale antud lahendust oleks vaja ja kuidas need lahendused töötavad. Seega on vajalik juurdepääsetavuse arendamisel tugitehnoloogia abil loodud lahendusi praktiliselt testida. Autori töökoha sülearvuti operatsioonisüsteem on macOS, mistõttu alustati esimesi teste ekraanilugejaga VoiceOver.

Arvestades, et automaattestid aitavad kiiresti vigu leida, prooviti rakenduse koodibaasi integreerida juurdepääsetavuse automaattestide teeki *eslint-plugin-jsx-a11y,* kuid sellest

loobuti põhjusel, et kõikide vigade parandamine oleks olnud väga ajamahukas ja oleks jäänud väljaspoole kokkulepitud paranduste mahust.

Kaardistatud juurdepääsetavuse probleemide põhjal loodi arenduspiletid. Arendusprotsessi oluline osa on koodi ülevaatused ja testimine. Iga arenduspileti jaoks loodud lahendus pidi enne testimist saama mõne teise arendaja heakskiidu. Üldjuhul on lahenduse funktsionaalsuse kontrollimiseks ette nähtud eraldi testija, kuid juurdepääsetavuse parandamisel vastutas töö autor nii lahenduste loomise kui ka lahenduste testimise eest. Lahenduste testimine toimus paralleelselt lahenduste loomisega. Valminud lahendused esitleti tellijale (Tpilet) ja seejärel integreeriti Tpileti toodangukeskkonda.

5 Tpileti rakenduse parandused

Peatükk annab ülevaate juurdepääsetavuse parandamiseks tehtud lahendustest, järeldustest ja võimalikest edasiarendustest.

5.1 Metoodika

Enamike Tpileti juurdepääsetavuse parandamiseks välja töötatud lahenduste aluseks on WCAG 2.1 suunised. Esmalt kaardistati juurdepääsetavuse probleemid WCAG nõuete kontekstis ning teostati lahendused lähtudes WCAG nõude täitmise kriteeriumist. Antud metoodikaga on seotud alampeatükid 5.2 Klaviatuur ja 5.5 Vea tuvastamine.

Osad juurdepääsetavuse probleemid olid tingitud üheleherakenduse dünaamilisest sisust ja nende puhul ei saanud tuua otsest seost WCAG nõude vastu eksimisega. Dünaamilise sisuga seotud probleemide lahendused (alampeatükk 5.3 Fookuse muutmine) lähtuvad Google'i arendaja Rob Dodsoni soovitustest [44] .

Tpileti vigade kaardistamisel ei toodud eraldi välja kasutusmugavuse tagamiseks vajalikke muudatusi. Kasutusmugavuse parandamise lahenduste osas lähtuti WebAIM-i soovitustest, eesmärgiga lihtsustada navigeerimist ekraanilugeja kasutaja jaoks [45] . WebAIM-i soovitused lähtuvad WCAG nõuetest. Antud meetodikaga on seotud alampeatükk 5.6 Navigeerimine. Tpileti rakenduses esitatakse reisi kohta informatsiooni ka piltide ja ikoonide kujul, mille sisule ei olnud ekraanilugejal juurdepääsu. See ei ole piletiostmiseks hädavajalik funktsionaalsus, kuid on oluline teave kasutajale. Probleemi lahendus järgib WCAG nõuet ja on esitatud alampeatükis 5.4 Mittetekstiline sisu.

Tpileti rakenduse juurdepääsetavuse parandamisel oli autor nii tarkvaraarendaja kui ka testija rollis. Testimisel lähtus autor *screening*-metoodikast [34] . Arenduse jooksul testis autor lahendusi järgmiste ekraanilugeja ja brauseri kombinatsioonidega:

VoiceOver ja Chrome, VoiceOver ja Safari, JAWS (beeta) ja Internet Explorer, JAWS ja Chrome ning NVDA ja FireFox. Valitud ekraanilugeja ja brauseri kombinatsioonid lähtuvad Eestis läbiviidud [26] ja WebAIM-i uuringu tulemustest [27]. Projekti juhtivarendaja vastutas tüüpnuppude ja fookuse muutmise komponendi lahenduse eest ning ülejäänud lahenduste eest vastutas käesoleva töö autor.

5.2 Klaviatuur

WCAG tase A kriteerium 2.1.1 Klaviatuur näeb ette veebilehe kõikide funktsioonide kasutatavust klaviatuuriliidese abil [46] . Kriteeriumil on erandid, kuid need ei ole antud rakenduse kontekstis olulised. Klaviatuuriga seotud lahendused parandasid järgmisi probleeme:

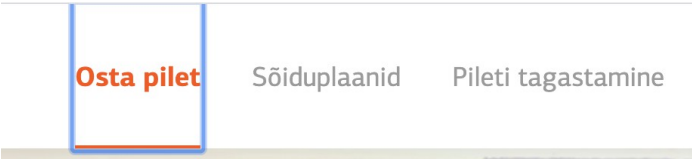
- kasutajal ei ole võimalik kuupäeva valida;
- kasutajal ei ole võimalik reiside otsimise nupule vajutada;
- kasutajal ei ole võimalik soovitud reisile piletit valida;
- kasutajal ei ole võimalik piletit lisada ostukorvi;
- kasutajal ei ole võimalik istekohta valida;
- kasutajal ei ole võimalik ostukorvi lisamise nupule vajutada;
- kasutajal ei ole võimalik maksemeetodit valida;
- kasutajal ei ole võimalik nõustuda pileti ostutingimustega.

Probleemid olid põhjustatud HTML elementide valest kasutamisest. Rakenduses oli enamik nuppe ja kasutajaliidese komponente loodud olemasoleva HTML elemendi asemel üldelemendiga <*div*>. Selliste vigade parandamine oli suhteliselt lihtne, sest lahendus seisnes <*div*> elemendi asendamises õige elemendiga (nt <*button*>, <*input field="checkbox"*>, <*input field="radio"*>). Kuid selliste vigade parandamine oli arendustundides ajamahukas, sest sama probleem esines üle rakenduse korduvalt, nii jagatud kui ka eraldiseisvate komponentide puhul. Paranduste koodinäited pileti valiku ja istekoha valiku komponendis on toodud lisades (vastavalt Lisa 2 – Pileti valiku komponendi koodinäide ja Lisa 4 – istekoha valiku komponendi koodinäide).

Tpileti veebilehel oli kuupäeva võimalik valida vaid hiirt kasutades. Kuupäeva valiku komponendi jaoks oli kasutusel teek *react-datepicker* (kalendri komponendi teek) versiooniga 1.0.4. Selle teegi kõige viimaste versioonide kasutamine oleks laiendanud komponendi juurdepääsetavuse võimalusi. Paraku ei olnud see võimalik, sest uuenduste käigus oleks tulnud välja vahetada üks teine teek, millest sõltuvad ülejäänud rakenduse lahendused ning see oleks nõudnud juba väga mahukat koodi ümberkirjutamist. Kalendri komponendi teegid põhjustavad tihti juurdepääsetavuse probleeme. Töö autor otsustas kasutada juurdepääsetavuse eksperdi Adrian Rosselli pakutud alternatiivset varianti [47]. Teek uuendati versioonile 1.8.0, mis tõi võimaluse hüpikaknas klaviatuuriga liikuda, kuid ekraanilugeja kasutaja ei saanud kuupäevade osas vajalikku infot. Sel põhjusel peideti hüpikaken ekraanilugeja eest ära ja kuupäeva valiku vormiväljaga seoti hääljuhised, mis teavitavad kasutajat, millises formaadis tuleb kuupäev sisestada. Juhiste tekst on järgmine: „Sisesta reisi väljumiskuupäev kujul kuupäev.kuu.aasta, näiteks 21.02.2020“. Ideaalis oleks kalendri hüpikaken juurdepääsetav kõikidele kasutajatele, mistõttu saab antud lahenduse sobilikkust hinnata üksnes ekraanilugeja kasutajate tagasiside põhjal.

Testimisel selgus, et ekraanilugejad käituvad kirjavahemärkide esitamisel erinevalt. JAWS ja NVDA ütlevad välja „kuupäev punkt kuu punkt..“, kuid VoiceOver seda ei tee. Kuna ekraanilugejate kasutamine sõltub ka kasutusoskusest võib kogenud VoiceOveri kasutajatel sellise olukorra lahendamiseks olla mingi oma viis. Sel põhjusel otsustati kasutajate tagasiside laekumiseni lahendust enam mitte muuta.

Klaviatuuri kasutamise võimalusi saaks suuremale kasutajaskonnale laiemalt kättesaadavaks teha, kui parandada rakendust lisaks vastavalt WCAG tase A kriteeriumile 2.4.7 Nähtav fookus, mis ütleb, et fokusseeritavatel elementidel peab olema visuaalselt eristuv disain, mida on näha, kui kasutaja elemendi peale liigub [48] . Brauseritele on sisse-ehitatud fookuse kuvamine (nähtav fookus), kui tegemist on standardsete fokusseeritavate elementidega nagu on näha joonisel 2. Tpileti rakenduses on osadel elementidel fookus nähtav, kuid paljudel elementidel on see disaini otsusena CSS-ga peidetud. Kui klaviatuuri fookus oleks nähtav järjepidevalt, siis saaksid kõik kasutajad üksnes klaviatuuri abil pileti ostmise teekonna läbida.



Joonis 2. Nähtav fookus

5.3 Fookuse muutmine

Veebisisu fookuse muutmine parandas järgmisi probleeme:

- kasutajat ei teavitata uue lehekülje laadimisest;
- kasutajat ei teavitata pileti valiku paneeli laadimisest;
- kasutajat ei teavitata uue sisu laadimisest paneeli.

Tpileti rakenduses on pileti ostmise toimingu erinevad etapid jagatud paneelidesse nagu lisades toodud kuvatõmmistelt näha võib (Lisa 1 – Ekraanipildid Tpileti rakenduse põhivaadetest). Kolmas paneel avaneb pileti valiku võimalusega alles siis, kui soovitud reis on valitud. Kui kasutaja on pileti valinud, asendatakse paneeli sisu asendatakse ostukorvi sisuga. Taoline dünaamiline sisu muutumine on üheleherakenduste puhul tavaline.

Ekraanilugeja kasutaja jaoks on olukord eksitav, sest kasutajale ei anta märku uue sisu laadimisest. Samavõrd on oluline, et uue sisu teavituse järel saaks kasutaja kohe selle sisu kuulata, mis tähendab, et lehekülje ehk ühtlasi ka ekraanilugeja fookus tuleb viia uue, laaditud sisu algusesse. Kui fookust ei muudeta, siis tekib olukord, kus näiteks reiside loetelus sobiva reisi valikul kasutajat teavitatakse kolmanda ehk pileti valiku paneeli tekkimisest, kuid selleks, et kasutaja paneeli jõuaks, tuleks navigeerida läbi kõikide elementide, mis reiside loetelus veel on. Kasutajale tähendaks see liikumist läbi suure hulga ebavajaliku info ja elementide.

Sel põhjusel on dünaamilise sisu tekkimisel tarvis teavitada kasutajat ja lisaks viia fookus kohe uue sisu juurde. ReactJS-s saab fookust muuta *ref*-ide abil ja ReactJS-i loonud meeskond on avaldanud ka näite kuidas seda teha [49] . Avaldatud näite põhjal teostatud lahendus on esitatud joonisel 3.

```
export class FocusTrap extends React.PureComponent {
    ref = React.createRef()

    componentDidMount() {
        this.ref.current.focus()
    }

    render() {
        return (
            <span tabIndex={-1} ref={this.ref}>
                {this.props.children}
            </span>
        )
    }
}
```

Joonis 3. Fookuse muutmise komponent

Selleks, et ekraanilugeja saaks vihje uue sisu kohta, peab *FocusTrap* komponent olema pakitud ümber sisu pealkirja. Joonis 4 esitab näite komponendi kasutamisest Tpileti rakenduse koodis.

```
<FocusTrap>
     <h2 className={styles.title}>
          <T>web.basket.basketPanel.title</T>
     </h2>
</FocusTrap>
```

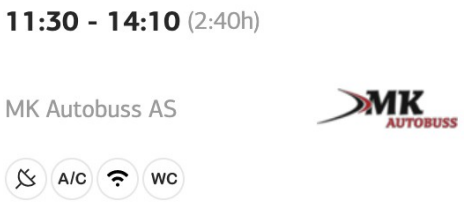
Joonis 4. Fookuse muutmise komponendi kasutamine

5.4 Mittetekstiline sisu

WCAG tase A kriteerium 1.1.1 Mittetekstiline sisu nõuab, et „kogu kasutajale esitatud mittetekstilisel sisul on võrdväärne tekstiline alternatiiv“ [50]. See tähendab, et kõikidele ikoonidele ja piltidele peab olema lisatud kirjeldav tekst. Tpileti rakenduse peamiseks veaks oli piltide (<*img*>) puhul seletava teksti ehk *alt* parameetri puudumine. Kui pildi puhul jäetakse ära *alt* parameeter, siis ekraanilugeja loeb ette pildi lähteallika, mis mitte kuidagi ei kirjelda pildil kujutatut ja on ekraanilugeja kasutajale mittevajalik info. Tpileti rakenduses kasutatakse bussis pakutavatest teenustest teavitamiseks ikoone (wifi, wc jne.). Ikoonidele tuli lisada kirjeldav parameeter *aria-*

label, mis lubab ekraanilugejal ikooni sisu kasutajale edastada siis, kui ikoon fokusseeritakse.

Kuid WCAG kriteerium kehtib vaid nende piltide ja ikoonide osas, mis edastavad olulist sisu. Kui pildid või ikoonid on ainuüksi kirjeldavad või kordavad tekstis olemasolevat infot, siis on lahendus ekraanilugeja kasutatavuse seisukohalt hoopis pildi või ikooni peitmine ekraanilugeja jaoks. Vastasel juhul hakkab ebaoluline või korduv sisu koormama kasutajat. Kui pildile lisada tühi *alt* parameeter (<*img alt=“*“>), siis ekraanilugeja ignoreerib elementi. Näiteks joonisel 5 kujutatud olukorras, kus on kõrvuti bussifirma nimetus ja samanimeline logo, otsustati logo pildile anda kaasa tühi *alt* parameeter.



Joonis 5. Bussifirma nimetus ja logo

Illustratiivsete ikoonide ja piltide puhul soovitatakse kasutada HTML-i asemel CSS-i [51] . Seda põhimõtet oli Tpileti rakenduse koodis juba järgitud.

5.5 Vea tuvastamine

WCAG tase AA kriteerium 3.3.1 Vea tuvastamine näeb ette, et „kui automaatselt tuvastatakse sisestusviga, tuleb leida vea asukoht (vigane element) ja esitada kasutajale vea kirjeldus tekstina“ [52] . Vea tuvastamise lahendused parandasid järgmisi probleeme:

- kasutajat ei teavitata veateadete puhul esileheküljel;
- kasutajat ei teavitata veateadete puhul ostukorvi paneelis.

Tpileti rakenduses ei saanud ekraanilugeja veateadete kohta infot, sest veateated sisestati DOM-i dünaamiliselt JavaScripti abil. Selle puuduse kõrvaldamiseks tuli veateatele lisada ARIA parameeter *role="alert"*, mis annab ekraanilugejale märku veateate olemasolust.

Tpileti rakenduses on mitmes kohas veateated antud edasi vaid värviga, mis tegelikult eksib WCAG 2.1 tase A kriteeriumi 1.4.1 vastu, mis sätestab, et „värvi ei või kasutada kui ainsat visuaalset teabe edastamise, tegevusele viitamise, vastama ajendamise või visuaalse elemendi eristamise vahendit“ [53]. Laiema juurdepääsetavuse tagamiseks oleks tarvis veateadete edastamise formaati muuta kõikide kasutajate jaoks. Antud olukorras loodi üksnes ekraanilugeja kasutajatele tekstilised veateated, mis peideti CSS-i abil visuaalselt ära. Joonisel 6 on välja toodud koodinäide komponendist ja joonisel 7 on näide komponendi kasutamisest.

```
export const ScreenReaderAlert = ({ text }) => (
    <div className="sr-only" role="alert">
        <T>{text}</T>
    </div>
)
```

Joonis 6. Ekraanilugejale teate edastamise komponent

```
<fieldset className={styles.paymentMethods}>
        {paymentMethodErrorVisible &&
            <ScreenReaderAlert text={'web.basket.payment.error'} />}
        ...
        ...
</fieldset>
```

Joonis 7. Ekraanilugejale teate edastamiseks vajaliku komponendi kasutamine

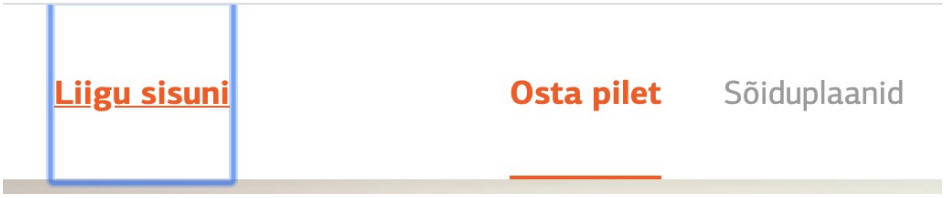
5.6 Navigeerimine

Ekraanilugeja kasutajad kasutavad lehel navigeerimisel mitmeid otseteid [25], et kiiremini jõuda vajaliku infoni. Selleks kasutatakse võimalust „hüpata“ pealkirjade, vormielementide vmt vahel.

Tase A kriteerium 1.3.1 Teave ja seosed nõuab, et „teave, struktuur ja seosed, mida annab edasi sisu esitus on kas tarkvaraliselt kindlaks tehtud või olemas tekstina“ [54]. Ekraanilugeja kasutajale on oluline saada võimalikult palju infot lehe struktuuri kohta.

Peamised Tpileti rakenduses tehtud muudatused olid järgmised. Esiteks, põhisisu pakkimine *<main role="main">* elemendi sisse. Parameeter *role="main"* on antud elemendile kaasa Internet Explorer brauseri tõttu. Teiseks, pealkirjade puhul asendati * * semantiliste pealkirja elementidega (*<h1>*...*<h6>*), mis on vastavalt HTML spetsifikatsioonile mõeldud sektsioonide pealkirjastamiseks.

Tase A kriteerium 2.4.1 Sisuplokkide vahelejätmine näeb ette, et „olemas on mehhanism, mille abil saab jätta vahele veebilehtedel korduvaid sisuplokke“ [55]. Antud kriteeriumi täitmiseks loodi „liigu sisuni nupp“, mis on näidatud joonisel 8. Menüüribal olev esimene element ehk Tpileti logo muutub fookuse saamisel tekstiks „Liigu sisuni“ ja teksti vajutamine viib kasutaja lehe peamise sisuni.



Joonis 8. "Liigu sisuni nupp“

Koodinäide „liigu sisuni nupu“ kohta on esitatud joonisel 9.

```
export const SkipNavLink = ({ href, className }) => (
    <a href={href} className={className}>
        <T>web.title.skipNav</T>
    </a>
)
```

Joonis 9. "Liigu sisuni nupu" komponent

Koodinäide „liigu sisuni nupu“ kasutamise kohta on esitatud joonisel 10.

```
export const DesktopHeader = () => (
    <header className={styles.container}>
       <SkipNavLink href="#main-content" className={styles.skipNav} />
        ...
        ...
    </header>
```

Joonis 10. "Liigu sisuni nupu" komponendi kasutamine

5.7 Järeldused

Käesolevas töös esitatud Tpileti arendused parandavad rakenduse juurdepääsetavust ekraanilugeja kasutaja jaoks pileti ostu teekonna kõikides etappides. Rakenduse parandatud funktsionaalsus muutis piletiostu toimingu ekraanilugeja ja klaviatuuri kasutamisel võimalikuks. Ekraanilugeja kasutajat teavitatakse veateadetest ja dünaamilise sisu vahetusest pileti ostmise paneelides. Lisaks on võimalik ekraanilugeja kasutajal saada infot bussis pakutavatest teenustest. Tänu HTML pealkirjadele ja struktuuri elementidele on rakenduses lihtsam ekraanilugejaga navigeerida.

Enamik Tpileti juurdepääsetavuse probleeme oli põhjustatud HTML-i valest kasutamisest. Arendusprojektis väljatöötatud lahendused ei olnud tehniliselt ülemäära keerukad, kuid eeldasid semantilise HTML-i ja ARIA põhimõtete teadmist. Üheleherakenduse tehnoloogia kasutamisel tuleb arvestada, et dünaamiline sisu võib tekitada juurdepääsetavuse probleeme, kuid selle lahendamine on fookuse muutmise abil võimalik. Juurdepääsetavuse laiemaks tagamiseks peaks vastavaid nõudeid ja praktikaid rohkem tutvustama õppeprogrammides.

Juurdepääsetavuse paranduste lähteülesande koostamisel ei olnud keegi meeskonnast juurdepääsetavuse temaatikaga tuttav. Kuna ekraanilugeja-brauseri kombinatsioonide käitumises on erisusi, siis oleks tulnud juurdepääsetavuse probleemide kaardistamisel kaasata rohkem kasutajaid, kes kasutavad erinevaid ekraanilugeja-brauseri kombinatsioone, et saada selgem sisend vajalike muudatuste kohta.

Juurdepääsetavuse tagamisel on projekti haldamine ehk erinõuete täitmise järgimine kogu rakenduse kontekstis sama oluline kui parandamiseks tehtud arendustööd. Tpileti projektis ei saanud haldamine piisavalt tähelepanu. Pärast töötavate lahenduste esitamist tellijale (Tpilet) ilmnes mõni aeg hiljem, et pileti valiku lisatud funktsionaalsus ei toimi päris õigesti. Kui juurdepääsetavusega seotud lahendusi koostati ühes keskkonnas (niiöelda arendusharus), siis paralleelselt arendati muid funktsionaalsusi teistes harudes. Erinevate lahenduste koodide kokkupanemisel tekkisid ühildumisprobleemid, mida ei lahendatud õigesti ja seetõttu jäi osa juurdepääsetavuse lahendustest integreerimata. Probleemi avastamisel lisati rakendusse algselt välja jäänud koodimuudatused.

Loodud lahenduste testimine *screening*-meetodiga kinnitas, et ekraanilugejaga on võimalik piletit osta, kuid piletiostu toiming ei ole ekraanilugeja kasutamisel mugav. Samas tuleb arvestada, et autori läbiviidud testimise tulemus ei ole piisav rakenduse juurdepääsetavuse lõplikuks hindamiseks, sest see ei ole samaväärne tagasisidega inimestelt, kes kasutavad ekraanilugejat igapäevaselt. Rakenduse juurdepääsetavuse ja kasutatavuse täpsemaks hindamiseks tuleks läbi viia kasutajatega testimine.

Lahenduste loomise protsess kinnitas, et juurdepääsetavuse lahendused, näiteks klaviatuuri kasutamise võimaluste laiendamine, parandavad rakenduse kasutatavust kõikide inimeste jaoks. Samuti oleks juurdepääsetavuse nõuetega arvestamine Tpileti projekti algfaasist alates olnud ajaliselt ja rahaliselt tulusam.

5.8 Võimalikud edasiarendused

Järgnevalt antakse ülevaade võimalikest järgmistest sammudest juurdepääsetavuse laiendamiseks.

Käesolevas töös esitatud Tpileti arendused on märkimisväärselt parandanud rakenduse juurdepääsetavust, mistõttu saaks kasutajatega testimisel keskenduda kasutusmugavusele. Tpileti rakendus võiks võimaldada mugavat ja juurdepääsetavat pileti ostmise kogemust. Tpileti rakenduse juurdepääsetavuse ja kasutatavuse täiendavaks hindamiseks oleks mõistlik läbi viia testimine ekraanilugeja kasutajatega.

Testimisel võiks kaasata erineva arvuti- ja ekraanilugeja kasutusoskustega inimesi, kelle eelistused ekraanilugeja-brauseri kombinatsioonide osas on erinevad.

Kasutatavuse tagamiseks ja edasiarendamises tuleb kasutajatega testimise vajadus ette näha nii arendusfaaasi jooksul kui ka lõpus. Testimine peaks olema fikseeritud nii lähteülesandes kui projektikirjelduses. Samas peab eesmärk olema ka täpselt määratletud ning keskenduma eelkõige enamlevinud ekraanilugejatele ja veebilehitsejatele ning nende kombinatsioonidele.

Tpileti juurdepääsetavuse parandamisel seati esmaseks sihiks ekraanilugeja abil veebilehe kasutamisvõimaluse loomine. Edasine samm võiks olla juurdepääsetavuse laiendamine teistele sihtrühmadele, kes võivad praegu olla välja jäetud rakenduse kasutamise võimalusest. Suurema kasutajaskonna saavutamiseks oleks hea viia rakendus vastavusse WCAG 2.1 AA taseme nõuetega. Samuti võimaldaks suuniste järgimine tagada paremat kasutuskogemust kõikidele kasutajatele.

Juurdepääsetavuse tagamisel mängib olulist rolli haldamine. Hetkel puuduvad arenduskeskkonnas automaattestid, mis kontrolliksid eri elementide vastavust juurdepääsetavuse nõuetele. Seega võivad tekkida olukorrad, kus lisaarenduste käigus juurdepääsetavus kannatab. Tpileti kontekstis oleks soovitatav katta kõige olulisemad ehk pileti ostmiseks vajalikud komponendid juurdepääsetavust kontrollivate automaattestidega.

6 Kokkuvõte

Veebilehtede juurdepääsetavuse tagamine on oluline, et hõlbustada puuetega inimestel teostada igapäevatoiminguid ja võimaldada paremini osaleda ühiskondlikus elus. Juurdepääsetavuse nõuete rakendamine omab palju laiemat mõju, sest parandab lehe üleüldist kasutusmugavust kõikide kasutajate jaoks. Eestis moodustab puuetega inimeste osakaal ligikaudu 12% kogu elanikkonnast ja tõenäoliselt rahvastiku vananemisega see protsent kasvab.

Juurdepääsetavuse tagamiseks vastu võetud Euroopa Liidu direktiivid näevad ette, et alates 2018. aastast peavad avaliku sektori asutused vastama WCAG juurdepääsusuunistele ja alates 2025. aastast laieneb see kohustus ka olulisi teenuseid pakkuvatele ettevõtetele. Seega suureneb lähiaastatel oluliselt juurdepääsetavuse arvestamise vajadus arendusprojektides. Teenused ja rakendused, mida praegu luuakse, peaksid juba arvestama juurdepääsetavuse nõuetega, sest tagantjärele on juurdepääsetavuse parandamine kallim ja keerulisem.

Käesoleva diplomitöö eesmärgiks oli välja töötada lahendused Tpileti veebirakenduse juurdepääsetavuse parandamiseks. Tpiletil ei ole erafirmana veel kohustust juurdepääsetavuse nõudeid järgida, kuid Tpileti veebirakendus ei võimaldanud ekraanilugejat kasutavatel nägemispuudega inimestel veebist bussipileteid osta. Eesmärgi saavutamiseks kaardistati Tpileti veebirakenduse juurdepääsetavuse probleemid, piiritleti parandamise maht ja loodi vajalikud lahendused.

Käesoleva töö tulemusena valmisid Tpileti rakenduse juurdepääsetavust parandavad lahendused. Tpileti rakenduse paranduste loomisel lähtuti WCAG juurdepääsusuuniste nõuetest. WCAG juurdepääsusuunisete täitmine on eelkõige seotud semantilise HTML-i ja ARIA korrektse kasutamisega. HTML-i kasutamine juurdepääsetavuse tagamiseks esimese eelistusena vähendab vajadust tegeleda eraldi erinevate ekraanilugejate ja veebilehitsejate ühilduvusprobleemidega. ARIA kasutamine rakenduste väljatöötamisel

ning uute versioonide kasutusevõtul nõuab täiendavat testimist. Loodud lahendusi testiti *screening*-meetodil erinevates ekraanilugeja-brauseri kombinatsioonides. Testimise tulemusena oli ekraanilugejaga pileti ostmine võimalik, kuid piletiostu toiming ei olnud ekraanilugejat kasutades mugav. Rakenduse juurdepääsetavuse ja kasutatavuse täiendavaks hindamiseks on soovituslik viia järgmisena läbi testimine kasutajatega.

Kui esmaparandustena nähti ette üksnes osadele WCAG nõuetele vastamise, siis edaspidi oleks mõistlik viia kogu rakendus WCAG 2.1 AA taseme nõuetega vastavusse. Juurdepääsusuuniste järgimine suurendaks rakenduse kasutajaskonda ning üldist kasutusmugavust. Juurdepääsetav ja kasutatav rakendus on lõppkokkuvõttes kasulik nii kasutajale kui ka Tpileti ettevõttele.

Töös esitatud konkreetsed lahendused ja projekti elluviimisest saadud kogemused saavad olla eeskujuks teistele arendajatele. Sellega on lõputööl lisaks Tpileti rakenduse juurdepääsetavuse parandamisele loodetavasti panus üldise teadlikkuse tõstmisele ja annab julgustust juurdepääsetavuse tagamiseks tulevastes arendusprojektides.

Kasutatud kirjandus

[1] Eesti puuetega inimestega koda. [WWW] https://epikoda.ee/spetsialistile/statistika (21 03.2020)

[2] Euroopa Parlamendi ja nõukogu 2.12. 2015 seletuskiri COM(2015) 615 liikmesriikide õigus- ja haldusnormide ühtlustamise kohta seoses toodete ja teenuste ligipääsetavusnõuetega. [WWW] https://eur-lex.europa.eu/legal-content/ET/TXT/HTML/?uri=CELEX:52015PC0615&from=ET (11.01.2020)

[3] Puuetega inimeste sotsiaaltoetuste seadus.(27.01.1999). Riigi Teataja I, 1999, 16, 273. [WWW] https://www.riigiteataja.ee/akt/13114771 (14.02.2020)

[4] Harry Stephen Keeler Society. [WWW] https://site.xavier.edu/polt/typewriters/tw-history.html (12.02.2020)

[5] Encyclopaedia Britannica. [WWW] https://www.britannica.com/biography/Alexander-Graham-Bell (12.11.2019)

[6] Introduction to Accessibility. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/fundamentals/accessibility-intro/ (16.07.2019)

[7] Accessibility, Usability, Inclusion. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/ (17.08.2019)

[8] World Wide Web Consortium Launches International Program Office for Web Accessibility Initiative. World Wide Web Consortium. [WWW] https://www.w3.org/Press/IPO-announce (19.09.2019)

[9] Web Content Accessibility Guidelines 1.0. World Wide Web Consortium. [WWW] https://www.w3.org/TR/WAI-WEBCONTENT/ (18.08.2019

[10] Euroopa Parlamendi ja nõukogu 26.10.2016 direktiiv 2016/2102/EL avaliku sektori asutuste veebisaitide ja mobiilirakenduste juurdepääsetavusest. [WWW] https: https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:32016L2102&from=ET (18.07.2019)

[11] Euroopa Parlamendi ja nõukogu 17.04 2019 direktiiv 2019/882/EL toodete ja teenuste ligipääsetavusnõuete kohta. [WWW] https: https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:32019L0882&from=et (18.10.2019)

[12] Accessibility requirements for ICT products and services. EN 301 549 V2.1.2 (2018-08). Harmonised European Standard [WWW] https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf (12.09.2019)

[13] Kikkas, K. Puuetega inimesed ja infotehnoloogia - Mis? Kuidas? Miks? : monograafia. Tallinna Tehnikaülikooli Kirjastus, 1995.

[14] Kikkas, K. Using the internet in rehabilitation of people with mobility impairments - case studies and views from Estonia. (Doktoritöö, Tallinna Tehnikaülikool). TTU Press, 1999.

[15] Kikkas, K. Ühe hullu idee lugu. [WWW] http://kakupesa.net/kakk/rtkroonika/TTY.php (14.03.2020)

[16] Tamberg, T. Ettepanekud arendusprotsessis ligipääsetavuse nõuete arvestamiseks SEB Eesti näitel. Magistritöö. Tallinna Tehnikaülikool, 2018.

[17] Mets, M.-E. Suunised tagamaks veebi sisu juurdepääsetavust puutetundlikel mobiilseadmetel. Bakalaureusetöö. Tallinna Tehnikaülikool, 2017.

[18] Kivisikk, K. Veebikeskkondade WCAG 2.0 nõuetele vastavuse ning kasutatavuse hindamine nägemispuudega inimeste vaatenurgast. Bakalaureusetöö. Tallinna Tehnikaülikool, 2017.

[19] Ainsalu, O. Veebilehtede käideldavuse ja kasutatavuse hindamine erivajadustega inimestele avaliku sektori veebilehtede näitel . Magistritöö. Tallinna Tehnikaülikool, 2016.

[20] Veebide koosvõime raamistik. Versioon 1.0. Majandus- ja kommunikatsiooniministeerium, 2012. [WWW] https://www.mkm.ee/sites/default/files/veebide_raamistik.pdf (16.12.2019)

[21] Avaliku sektori veebilehtede vastavus WCAG 2.0 nõuetele 2015. aastal: Uuringu aruanne. Majandus- ja Kommunikatsiooniministeerium, 2015. [WWW] https://www.mkm.ee/sites/default/files/wcag_aruanne_2015.pdf (15.12.2019)

[22] Web Content Accessibility Guidelines (WCAG) 2.1. World Wide Web Consortium. [WWW] https://www.w3.org/TR/WCAG21/ (17.08.2019)

[23] Richards, M. (2019). Semantics To Screen Reader Users. [WWW] https://alistapart.com/article/semantics-to-screen-readers/ (19.09.2019)

[24] Tools And Techniques in How People with Disabilities Use the Web. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/people-use-web/tools-techniques/ (14.08.2019)

[25] What is a screen reader? Axess Lab. [WWW] https://axesslab.com/what-is-a-screen-reader/ (11.08.2019)

[26] Screen Reader User Survey #8 Results [WWW] https://webaim.org/projects/screenreadersurvey8/ (12.10.2019)

[27] Nuum, T. ja Mets, M-E. Uuring: Kuidas nägemispuudega inimesed veebilehti kasutavad? (2019) [WWW] https://blog.twn.ee/et/n%C3%A4gemispuudega_inimeste_eelistuste_uuring (27.00.2019)

[28] Using HTML sections and outlines. Mozilla Developer Network. [WWW] https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML_sections_and_outlines (15.08.2019)

[29] Element definitions. World Wide Web Consortium. [WWW] https://www.w3.org/TR/html52/dom.html#element-definitions (16.08.2019)

[30] Eisenberg, J. D., (2001) „Forgiving" Browsers Considered Harmful. [WWW] https://alistapart.com/article/forgiving/ (17.08.2019)

[31] Accessible Rich Internet Applications (WAI-ARIA) 1.1. World Wide Web Consortium. [WWW] https://www.w3.org/TR/wai-aria/ (22.09.2019)

[32] Using ARIA. W3C Working Draft 27 September 2018. World Wide Web Consortium. [WWW] https://www.w3.org/TR/using-aria/ (22.09.2019)

[33] ARIA. Mozilla Developer Network. [WWW] https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA (15.09.2019)

[34] Henry, S. L. (2007). Just Ask: Integrating Accessibility Throughout Design. [*Online*] http://www.uiaccess.com/JustAsk/index.html (16.12.2019)

[35] Holmes, K. Mismatch: How Inclusion Shapes Design. MIT Press, 2018.

[36] Web Accessibility Evaluation Tools List. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/ER/tools/ (18.08.2019)

[37] Madise, Ü. Ligipääs e-teenustele. https://www.oiguskantsler.ee/sites/default/files/field_document2/Ligip%C3%A4%C3%A4s%20e-teenustele.pdf. 2019. (18.10.219)

[38] Accessiblity. Web Fundamentals. Google Developers. [WWW] https://developers.google.com/web/fundamentals/accessibility (26.08.2019)

[39] Accessibility. Web Techonology for Developers. Mozilla Developer Network. [WWW] https://developer.mozilla.org/en-US/docs/Web/Accessibility (29.08.2019)

[40] Web Accessibility by Google. [WWW] https://www.udacity.com/course/web-accessibility--ud891 (19.10.2019)

[41] A11ycasts with Rob Dodson. [WWW] https://www.youtube.com/watch?v=HtTyRajRuyY&list=PLNYkxOF6rcICWx0C9LVWWVqvHlYJyqw7g / (16.10.2019)

[42] WebAIM's WCAG 2 Checklist. WebAIM [WWW] https://webaim.org/standards/wcag/checklist (26.09.2019)

[43] Understanding WCAG 2.1. Updated. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/WCAG21/Understanding/ (12.08.2019)

[44] A11ycasts with Rob Dodson. [WWW] https://www.youtube.com/watch?v=srLRSQg6Jgg (19.09.2019)

[45] Designing for Screen Reader Compatibility. WebAIM. [WWW] https://webaim.org/techniques/screenreader/#how (11.10.2019)

[46] Guideline 2.1 Keyboard Accessible. World Wide Web Consortium. [WWW] https://www.w3.org/TR/WCAG21/#keyboard (21.09.2019)

[47] Roselli, A. (2019) Maybe You Don’t Need a Date Picker. [WWW] https://adrianroselli.com/2019/07/maybe-you-dont-need-a-date-picker.html (16.09.2019)

[48] Success Criterion 2.4.7 Focus Visible. World Wide Web Consortium. [WWW] https://www.w3.org/TR/WCAG21/#focus-visible (14.08.2019)

[49] Refs and the DOM. ReactJS Documentation. [WWW] https://reactjs.org/docs/refs-and-the-dom.html (10.10.2010)

[50] Suunis 1.1 Tekstilised alternatiivid: tagada tekstiline alternatiiv mittetekstilisele sisule, et seda saaks kasutajate vajadustest lähtuvalt muuta, nt suureks kirjafondiks, punktkirjaks,

kõneks, sümboliteks või lihtsustatud keeleks. World Wide Web Consortium. [WWW] https://www.w3.org/Translations/WCAG20-et/WCAG20-et-20131210/#text-equiv (13.10.2019)

[51] Decorative Images. Web Accessibility Initiative. [WWW] https://www.w3.org/WAI/tutorials/images/decorative/ (14.03.2020)

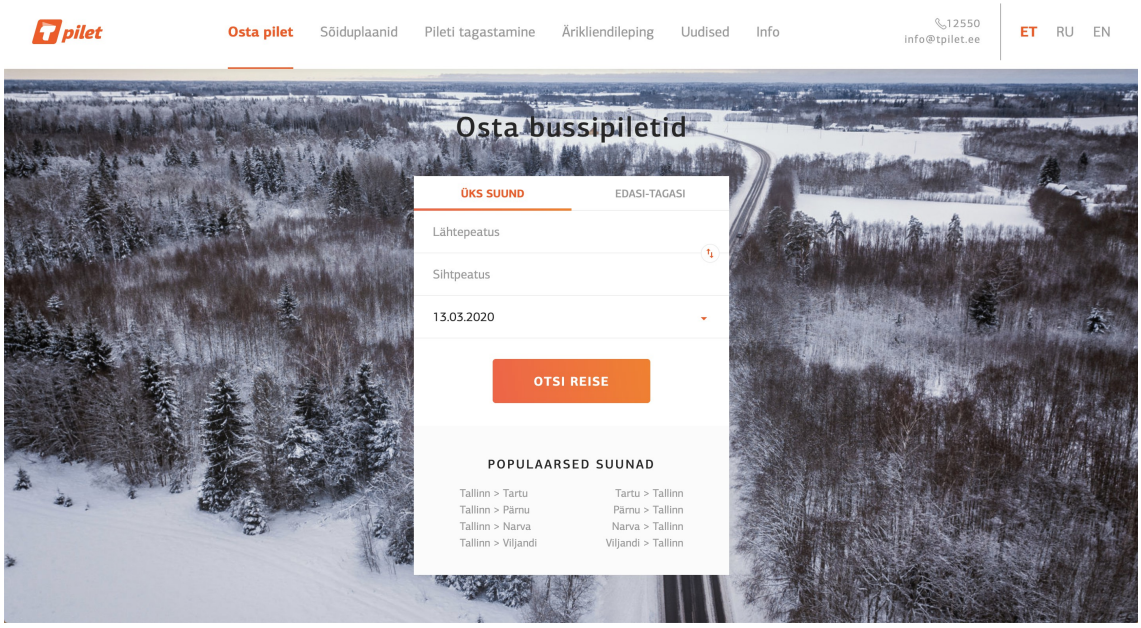
[52] Suunis 3.3 Sisestusabi: aidata kasutajatel vigu vältida ja parandada. World Wide Web Consortium. [WWW] https://www.w3.org/Translations/WCAG20-et/WCAG20-et-20131210/#minimize-error (11.10.2019)

[53] Suunis 1.4 Eristatavus: teha sisu nägemine ja kuulmine kasutajatele lihtsamaks muuhulgas esiplaani eraldamisega taustast.World Wide Web Consortium. [WWW] https://www.w3.org/Translations/WCAG20-et/WCAG20-et-20131210/#visual-audio-contrast (14.10.2019)

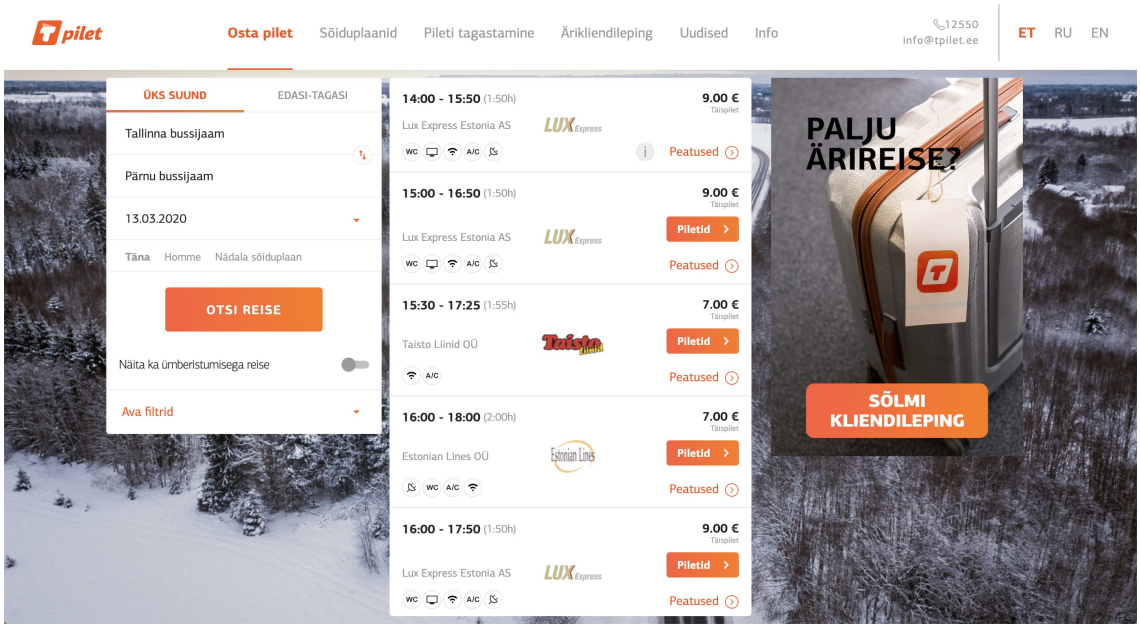
[54] Suunis 1.3 Kohandatavus: luua sisu, mida saab teavet või struktuursust kaotamata eri viisidel (nt lihtsama küljenduse abil) esitada. World Wide Web Consortium. [WWW] https://www.w3.org/Translations/WCAG20-et/WCAG20-et-20131210/#content-structure-separation (11.09.2019)

[55] Suunis 2.4 Navigeeritavus: tagada kasutajatele võimalused, mis aitavad navigeerida, sisu leida ja oma asukohta määrata. World Wide Web Consortium. [WWW] https://www.w3.org/Translations/WCAG20-et/WCAG20-et-20131210/#navigation-mechanisms (14.09.2019)

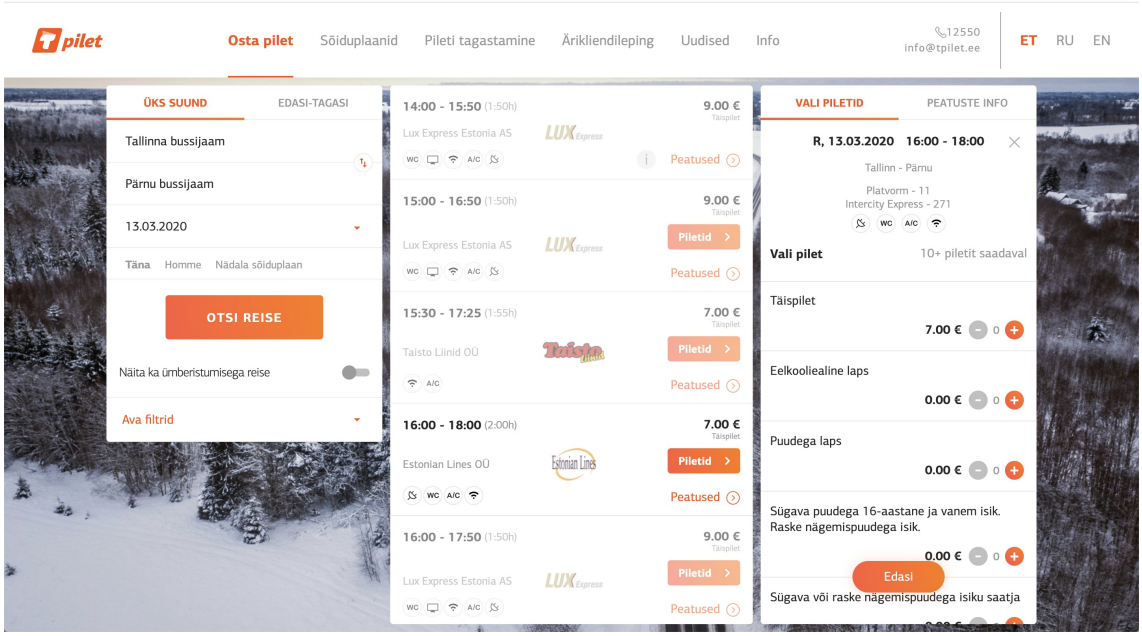
Lisa 1 – Ekraanipildid Tpileti rakenduse põhivaadetest



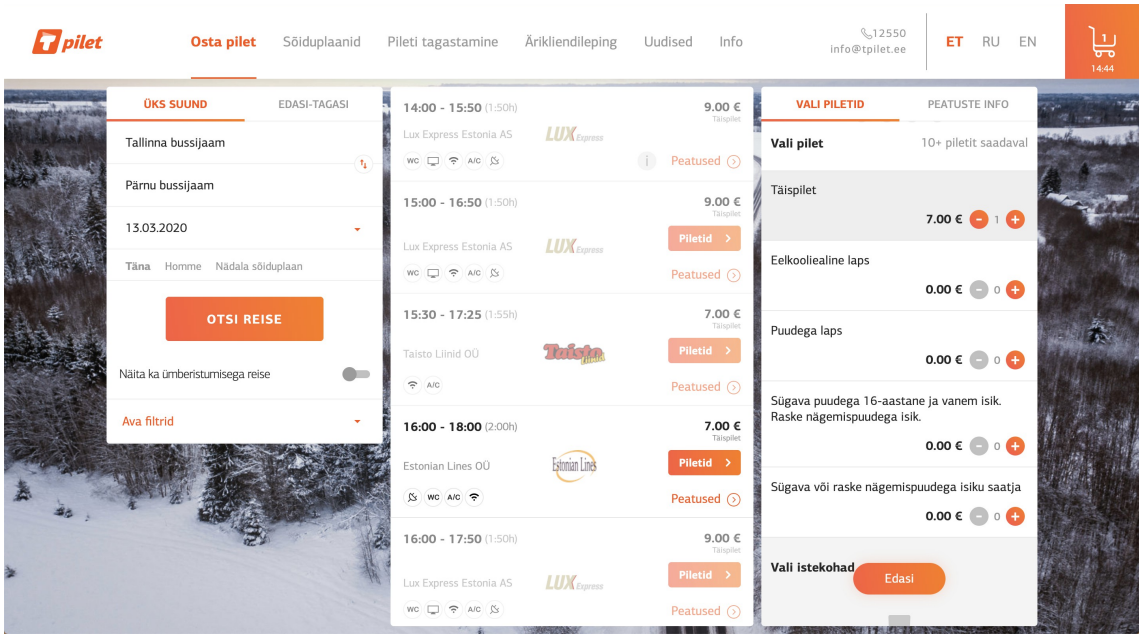
Joonis 11. Esilehekülg



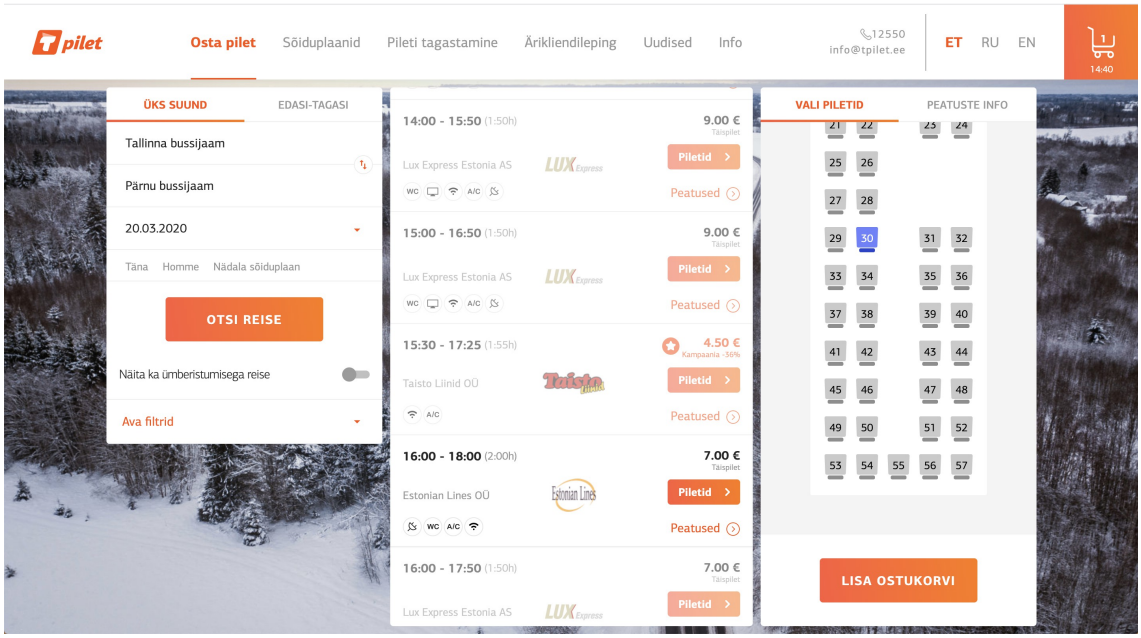
Joonis 12. Reiside loetelu



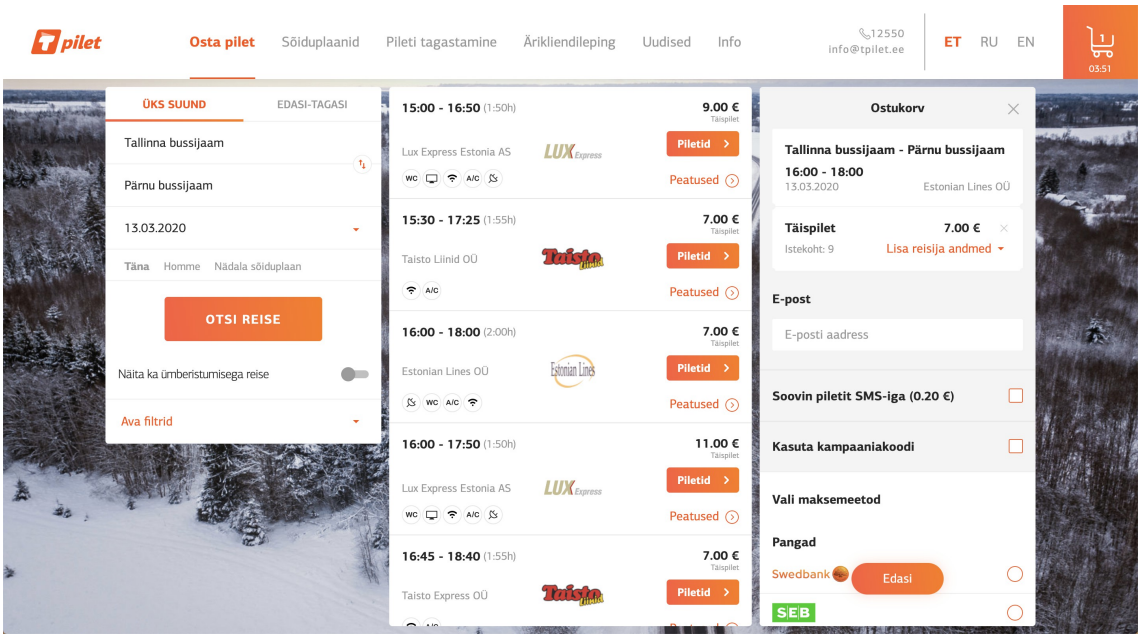
Joonis 13. Pileti valiku paneel



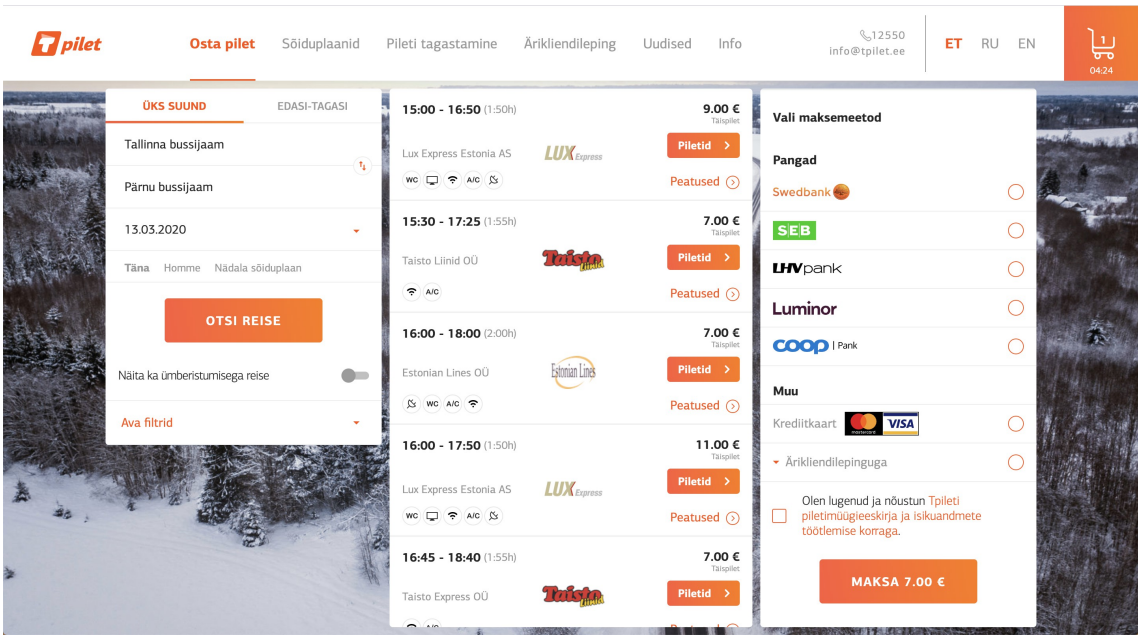
Joonis 14. Pikenev pileti valiku paneel



Joonis 15. Istekoha valik pileti valiku paneelis



Joonis 16. Ostukorvi paneel



Joonis 17. Maksemeetodi valik ostukorvi paneelis

Lisa 2 – Pileti valiku komponendi koodinäide

```
const $NumberPicker = (...) => {

const decrementDisabled = value === 0

return (
    <div className={...}>
      <button onClick={decrement}
          className={...}
          aria-label={translate(removeDesc)}
      />
      <span className={...}></span>
      <button
          onClick={increment}
          className={...}
          aria-label={translate(addDesc)}
      />
      </div>
)
}
```

Joonis 18. Pileti valiku komponent

Lisa 3 – Kuupäeva valiku komponendi koodinäide

```
const SingleDateSelect = (...) => (
    <div className={...}>
        <ScreenReaderLabel text="web.searchForm.dateLabel" />
        <DatePicker
            dateFormat={dateFormatDots}
            onFocus={handleDepartureFocus}
            onBlur={handleBlur}
            popperContainer={(...) =>
                  <div aria-hidden>
                        {children}
                  </div>}
            selected={...}
            placeholderText={...}
            onChange={...}
            minDate={moment()}
            locale={activeLanguage}
            popperPlacement="bottom-start"
            fixedHeight
        />
    </div>
)
```

Joonis 19. Kuupäeva valiku komponent

Lisa 4 – Istekoha valiku komponendi koodinäide

```
const $BusSeat = (...) => (
    <div className={...}>
        <button
            id={...}
            className={...}
            onClick={handleClick}
            aria-label={translate(getAriaLabel(...))
            .replace('<NUMBER>', seatNumber )}
            aria-hidden={
              'nonSeat || isDriver ? true : false
            }
            tabIndex={nonSeat || isDriver ? -1 : 0}>
           ...
        </button>
    </div>
)
```

Joonis 20. Istekoha valiku komponent