Tallinna Ülikool

Digitehnoloogiate instituut

Informaatika õppekava

Iseõppimist toetava rakenduse loomine Angulari ja Spring Booti abil

Bakalaureusetöö

Autor: Konstantin Tenman

Juhendaja: Romil Rõbtšenkov

Autor: ..... „ ..... „2018

Juhendaja: ..... „ ..... „2018

Instituudi direktor: ..... „ ..... „2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Konstantin Tenman (sünnikuupäev: 10.06.1987)

1. Annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose " Iseõppimist toetava rakenduse loomine Angulari ja Spring Booti abil", mille juhendaja on Romil Rõbtšenkov, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.
2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Sissejuhatus

Antud bakalaureusetöö eesmärgiks on luua iseõppimise veebirakendus Flashcards. Autorit motiveeris antud teemal kirjutama asjaolu, et igal aastal on Tallinna Ülikooli Digitehnoloogiate instituudi „Teoreetilise informaatika“ aine eksami esimese katsega sooritanute osakaal väike olnud. Flashcardsi rakendus teeb teoreetilise informaatika kursuse eksamiks valmistumise huvitavaks ja lihtsaks, pakkudes välja mõistekaartide metoodika.

Flashcards on edasiarendus John Washami analoogsest koodist, mis on kättesaadav GitHubist[1], John Washami repositooriumist “Coding Interview University”[2]. John Washam suutis teha kõikehõlmava loetelu arvutiteaduse erinevatest aspektidest nagu näiteks andmestruktuurid, algoritmid, programmeerimismustrid jne. Informatsiooni arvutiteadusest oli nii palju, et seda kõike oli korraga raske meeles pidada, seepärast otsustas Washam teha “Computer Science Flash Cards”[3] veebirakenduse, mis kasutab järgnevaid tehnoloogiad: Python 3[4], Flask[5], SQLite[6] ja Bootstrap 3[7].

Töö autoril tekkis idee kasutada antud veebirakendust “Teoreetilise informaatika” kursuse teooria õppimise tarbeks. Kuna Washami veebirakendus on avatud lähtekoodiga, otsustas töö autor selle aluseks võtta ja täiendada koodi enda loodud repositooriumis kohandatud veebirakenduse jaoks. Käesolevas töös annab autor ülevaate, mida rakendus teeb, kellele see mõeldud on ning milleks seda kasutada.

Bakalaureusetöö koosneb kolmest osast. Esimeses peatükis tutvustatakse erinevaid iseõppimise võimalusi ning antakse ülevaade olemasolevatest rakendustest. Samuti tutvustatakse loodava veebirakenduse funktsionaalseid nõudeid. Teises peatükis kirjeldab autor rakenduse loomise protsessi, tutvustab Flashcardsi loomise etappe, ideed ja arenduse käigus tekkinud tähelepanekuid. Peatükk hõlmab ka loodava rakenduse tutvustust. Viimases peatükis antakse ülevaade rakenduse kasutajakogemuse testimisest ja pakutakse erinevaid edasiarendamise ideid.

[1] https://github.com
[2] https://github.com/jwasham/coding-interview-university
[3] https://github.com/jwasham/computer-science-flash-cards
[4] https://www.python.org
[5] http://flask.pocoo.org
[6] https://www.sqlite.org
[7] http://getbootstrap.com/docs/3.3

Antud bakalaureusetöö raames valminud rakendus asub aadressil www.ati.ee ning selle lähtekood on saadaval aadressil https://github.com/ktenman/flashcards5.

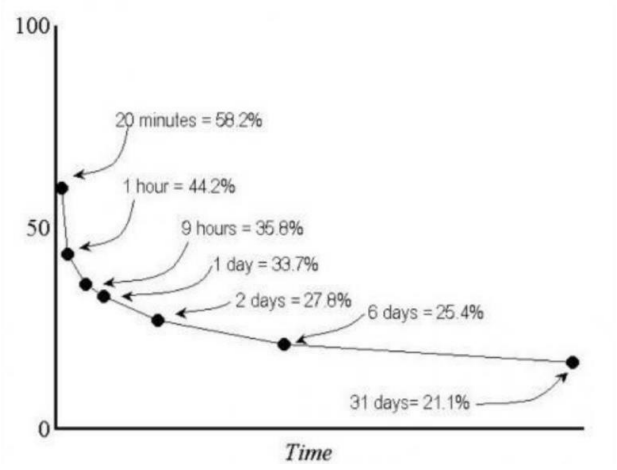
1 Iseõppimise võimalused

Järgnevas peatükis tutvustatakse erinevaid õpimeetodeid, võrreldakse olemasolevaid rakendusi ja sellest tulenevalt tuuakse välja nõuded veebirakenduse loomiseks, mille kaudu saab tõhusamalt õppida eelkõige Tallinna Ülikooli Digitehnoloogiate instituudi „Teoreetilise informaatika“ aine teooriat.

1.1 Õpimeetodid

Tänapäeval toimuvate muutuste kiirus nõuab kiiret ja pidevat uute teadmiste omandamist. Wozniaki (1999) arvates sõltub meelde jätmise kiirus eelkõige sellest, kuidas on suudetud materjali omandada. Hästi struktureeritud materjali võib õppida kordades kiiremini. Oluline on õpitust aru saada ning mitte lihtsalt pähe tuupida (Ullman & Lovelett, 2018).

Ebbinghaus (1885) näitas oma eksperimendis, et inimesed unustavad õpitud informatsiooni väga kiiresti (vt Joonis 1). 20 minutit peale õppimist mäletasid inimesed vähem kui 60% õpitud informatsioonist. Üks tund peale õppimist oli keskmiselt inimestel meeles ainult 44,2%. Ühe päeva pärast oli ununenud juba ligi 2/3 õpitust ja kuue päeva pärast 3/4.



Joonis 1. Ebbinghausi klassikaline unustamiskõver (Ebbinghaus, 1885)

1.1.1 Sõnaseletus- ehk mõiste- ehk tähenduskaardid

Tähenduskaarti defineeritakse, kui kartongi, mis koosneb sõnast, lausest või lihtsat pildist (Baleghizadeh & Ashoori, 2011). Tähenduskaardi loomisel tuleb õpitava aine kontekstist välja võtta kõige olulisem teave ja kirja panna kaardile nii, et ühel pool on küsimus ja

teisel pool vastus. Dizon ja Tang (2017) leidsid, et nii paber- kui digitaalkujul tehtud tähenduskaardid on sama efektiivsed, kuid üliõpilased eelistavad pigem digitaalset varianti. Tõenäoliselt on bussis olles või poes järjekorras seistes kõige mugavam memoreerida tähenduskaarte just mobiiltelefonist.

Robert Harris (2014) on toonud välja tähenduskaartide kolm põhilist hüve:

- Kerge kasutatavus – lihtne on kaasas kanda paberkujul 100 kaarti, aga veel lihtsam on kasutada neid veebis ning pääseda neile ligi igal ajal nutitelefoni kaudu.
- Kohene tagasiside – kaarti keerates saab kiirelt tagasisidet, kas vastus oli õige või vale.
- Õppimise tükeldamine osadeks – õppimine on lihtsam, kui see on tükeldatud väikesteks osadeks.

Paljud teadlased arvavad, et sõnaseletuskaartide strateegiat kasutades saab lihtsasti ja efektiivselt õpetada järgnevaid oskusi: tähestikku (Young, Hecimovic, & Salzberg, 1983), tähtsamaid kuupäevi ajaloost (Maheady & Sainato, 1985), uut sõnavara (Heron, Heward, Cooke, & Hill, 1983), definitsioone (Olenick & Pear, 1980), teoreeme vms. Uute sõnade õppimisel on üliõpilaste seas levinuim õppemeetod tähenduskaardid (Oxford & Crookall, 1990).

Mõistekaartidega õppimisel ei teki ka niinimetatud "loendi efekt". Nakata (2008) arvates põhjustab "loendi efekti" nähtust sõna asetus loetelus, mille tulemusel õppurid suudavad lihtsamini meelde jätta loetelu sees olevaid sõnu. Juhul, kui sõna asub väljaspool konteksti, võib tekkida märkimisväärseid raskused. Lisaks võib tekkida olukordi, kus konspekti alguses olevatele mõistetele pööratakse rohkem tähelepanu näiteks ajanappuse tõttu.

1.1.2 Intervallidega kordamine

Intervallidega kordamine (ingl *spaced repetition,* edaspidi SR) on alternatiivne meetod traditsioonilisele õppimisele ja on tõestatud, et nii suudavad inimesed teadmisi kauem meeles pidada ning õppimiseks kulub vähem aega (Kang, kuupäev puudub). Korduvat kordamist on ka lihtne rakendada, näiteks eksisteerib nutirakendusi, mis võimaldavad kasutajatel digitaalseid tähenduskaarte näpuga lohistades hinnata oma tulemusi vastuste

meeldejätmisel ning vastavalt algoritmile kuvatakse kaart uuesti teatud aja tagant (deWinstanley & Bjork, 2002).

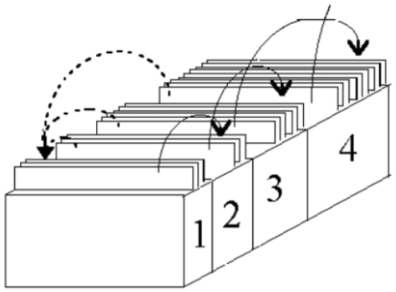
Donovan ja Radosevich (1999) uurisid üle viiekümne 20. sajandi teadustöö, mis testisid SR meetodi efektiivsust, ja näitasid, et selle meetodi kasutajad parandasid oma õppetulemusi vähemalt pool standardhälvet. SR efektiivsusel on 3 põhilist põhjust (Kwantlen Polytechnic University, kuupäev puudub):

- materjali kordamine teatud ajaperioodi tagant on efektiivsem kui materjali lugemine mitu korda samal õppeperioodil;
- kordamine vähendab informatsiooni ununemist, kui materjali ei ole kaua aega kasutatud;
- intervallide tagant kordamine aitab unustatud asju uuesti meelde tuletada, kui need esimeste kordamiste järel ununevad.

Kornell (2009) leidis, et intervallide tagant kordamine omab meeldejätmisele paremat mõju (90% eksperimendis osalejate seast) kui lihtsalt pähe tuupimine. Vaatamata statistikale aga jäi 72% osalejatest arvamusele, et tuupimine on efektiivsem kui SR – see on ka ilmselt põhjuseks, miks enamus tudengeid ei kasuta intervallidega kordamist.

1.1.3 Leitneri süsteem

Sebastian Leitner (1972) esitas 1970ndatel aastatel omanimelise süsteemi, mille järgi on võimalik tähenduskaarte efektiivsemalt kasutada. Selle meetodi rakendamisel sorteeritakse kaardid rühmadesse sõltuvalt sellest, kui hästi õppija igat kaarti teab. Meetodi tõhusus seisneb selles, et halvemini omandatavat materjali korratakse tihedamalt ning õpitut korratakse regulaarselt üle järjest suurenevate vahedega.



Joonis 2. Skemaatiline Leitneri süsteemi representatsioon. Valesti vastatud kaardi teekond on tähistatud punktiirjoonega (Kollegium des Clavius-Gymnasiums, 2016).

Oletame, et õppija soovib materjali korrata vähemalt neli korda. siis tal tuleb panna neli tühja ritta nii, et alguses asetsevad kõik kaardid esimeses karbis (vt Joonis 2). Õppija üritab sõnaseletuskaardile kirjutatud lahendust meenutada. Õige vastuse korral tõstab õppija kaardi järgmisse 2. karpi. Valesti vastatud kaart liigutatakse alati esimesse karpi (vt Joonis 2). Kui karp on täis, korratakse üle 2. karbis olev materjal ja need kaardid, mis meelde jäid, tõstetakse järgmisesse, 3. karpi. Nii kuni kõik kaardid on viimases karbis.

1.2 Analoogsed rakendused

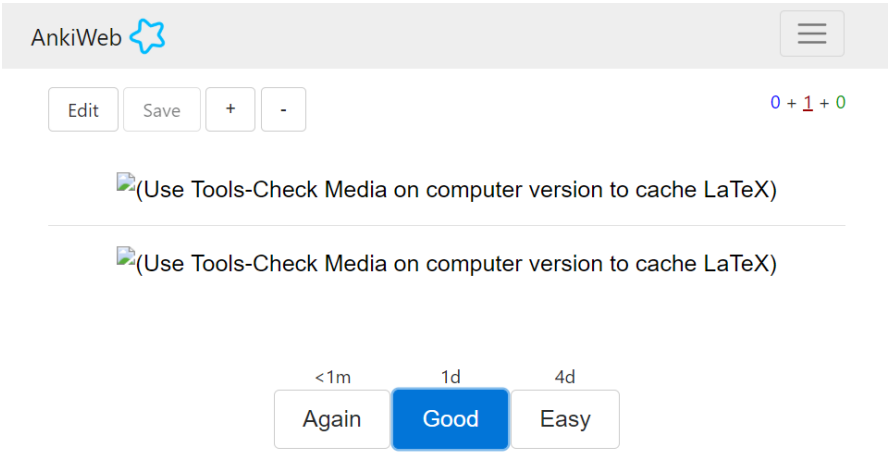
Selles peatükis antakse ülevaade olemasolevatest mõistekaartide põhimõttel töötavatest rakendustest või platvormidest, kus on võimalik sisestada erinevaid teoreetilise informaatika aine valemeid kasutades LaTeXit. LaTeX on Leslie Lamporti poolt kirjutatud makrokäskude pakett, mis on mõeldud eelkõige matemaatiliste valemite trükkimiseks ja kujundamiseks (Lamport, 1994).

1.2.1 Anki

Anki[8] on avatud lähtekoodiga programm, mis on loodud efektiivseks mõistekaartide õppimiseks (Anki, kuupäev puudub). Anki töötab kõigil enamlevinud platvormidel – Windows, MacOS, Linux, Android, iOS ning samuti on see saadaval veebis (Anki, kuupäev puudub).

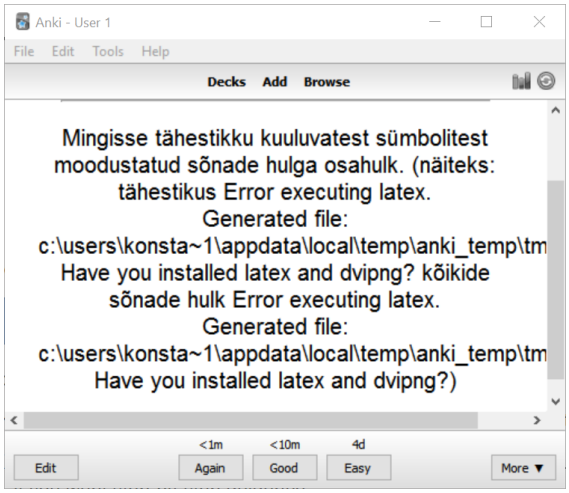
Anki Windowsi versioon on oma välimuse poolest aegunud ning koheselt tekib tunne, et rakendust ei ole uuendatud juba pikemat aega. Veebiversioon Ankist näeb välja palju kaasaegsem ja kasutajasõbralikum, aga seal ei ole kahjuks võimalik lisada LaTeXi käsklustega tähenduskaarte (vt Ekraanitõmmis 1).

[8] https://apps.ankiweb.net



Ekraanitõmmis 1. Veebiversioon Ankist

Anki Windowsi versiooni paigaldamisel probleeme ei tekkinud ja tegemist oli lihtsa protsessiga, aga probleeme tekitas see, et peale esimest käivitust polnud võimalik sisestada LaTeXi käsklusi. Probleemi lahendamiseks on vaja lisada veel täiendavalt LaTeX ja dvipng[9] paketid. Kõik vajalikud paketid saab paigaldada MiKTeXi[10] tarkvara abil.

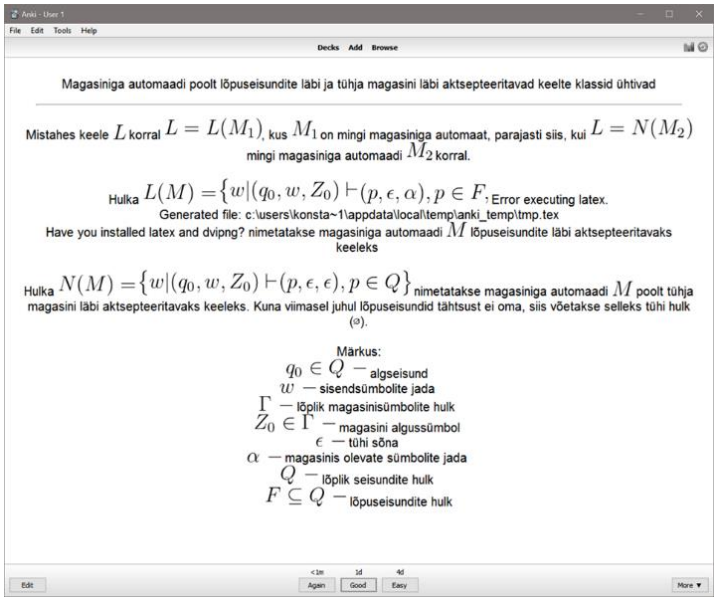


Ekraanitõmmis 2. Anki esimene käivitus Windowsi operatisionisüsteemiga arvutis

Peale MiKTeXi paigaldamist hakkas Anki kuvama ka LaTeXi käsklusi (vt Ekraanitõmmis 3). Lisaks LaTeXi käskudele on võimalik lisada tähenduskaartidele heliklippe, pilte ja videosid (Anki, kuupäev puudub). AnkiWebi kasutamiseks veebibrauseris on vaja luua kasutaja. Kasutades sama kasutajat erinevatel seadmetel, saab

[9] Tarkvarapakett, mis oskab LaTeX kujul sisestatud valemid muuta piltideks
[10] https://miktex.org

kaardid hoida alati sünkroonis, mis teeb Anki kasutamise väga mugavaks (Anki, kuupäev puudub).



Ekraanitõmmis 3. Anki ja LaTeX

Ankis on kasutusel intervallidega kordamine, iga vastust saab hinnata selle järgi, et kui hästi vastust teati ning vastavalt hinnangule kuvatakse tähenduskaart uuesti teatud aja pärast. AnkiWebis olev intervallskaala jaguneb kolmeks: alla minuti (raske), alla kümne minuti (keskmise raskusega) ja neli päeva (lihtne) (vt Ekraanitõmmis 1). Juhul, kui tahta kõik tähenduskaardid uuesti üle korrata, siis peab ootama vähemalt neli päeva, sest AnkiWeb ei võimalda taastada algseisu. Anki suureks plussiks on ka see, et võimalik on paigaldada palju lisasid (ingl *add-on*), mida on Ankile saadaval üle 400, nagu näiteks AwesomeTTS[11], millega tuleb kaasa sõnahäälduse tugi (Anki, kuupäev puudub).

Head omadused:

- Anki Windowsi versioon toetab LaTeXit;
- AnkiWeb hoiab tähenduskaarte serveris.

Puudused:

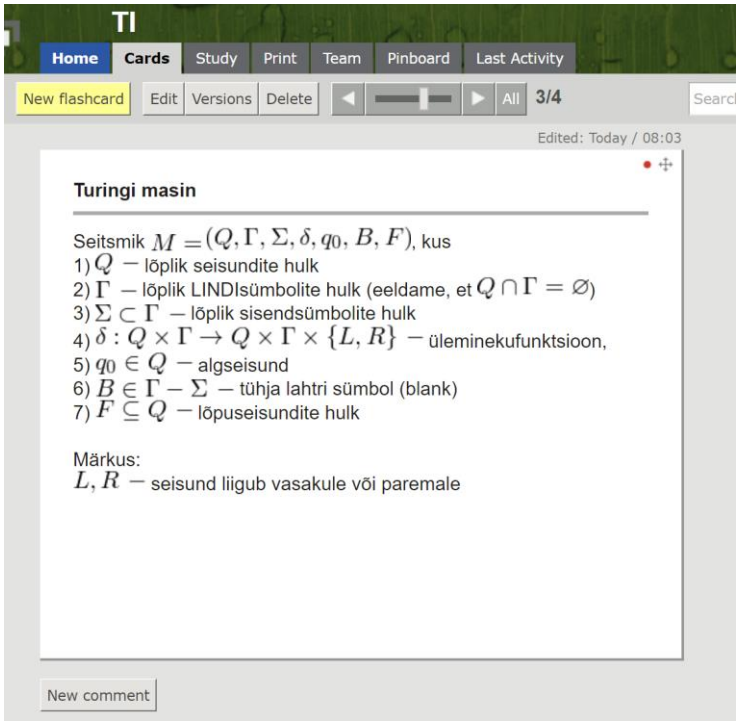
- Anki Windowsi versiooni talletab tähenduskaarte kõvakettal;
- telefonis saab kasutada ainult AnkiWebi versiooni;

[11] https://ankiweb.net/shared/info/301952613

- AnkiWeb ei toeta LaTeXit;
- AnkiMobile iOSi versioon maksab $42.99 (küll aga on Androidi versioon on tasuta);
- vanamoodne välimus.

1.2.2 CoboCards

CoboCards[12] on kasutatav veebis ning samuti on olemas rakendused nii iOSile kui ka Androidile (CoboCards, kuupäev puudub). Katsetamiseks valiti veebiversiooni, mis sarnaselt Anki Windowsi versiooniga näeb vanamoodne välja. Mobiilis kasutamiseks tuleb alla tõmmata rakendus, sest veebiversioon ei ole kohanduv väikestele ekraanidele. Lisaks sellele ei toetanud iOSi versioon üle ühe aasta vanust iPhone X mobiiltelefoni (11.11.2018). Anki veebiversioonist on aga CoboCards parem selle poolest, et viimasega saab kohe kirjutada LaTeXi käsklusi (vt Ekraanitõmmis 4).



Ekraanitõmmis 4. CoboCards

CoboCardsi kõik funktsionaalsused ei ole tasuta saadaval. Selleks et saada kõik funktsionaalsused tuleb osta Pro versioon, mis sõltuvalt paketi liitumise pikkusest maksab

[12] https://www.cobocards.com

3,59-5,99$ kuus. Pro versioonis on näiteks saadaval Leitneri süsteem, piiramatu arv tähenduskaarte, õppimise ajal muutmine, mõistekaartide importimine jpm.

Head omadused:

- LaTeXi tugi;
- eelnevalt salvestatud kaarte saab sirvida nii telefonis kui ka arvutis.

Puudused:

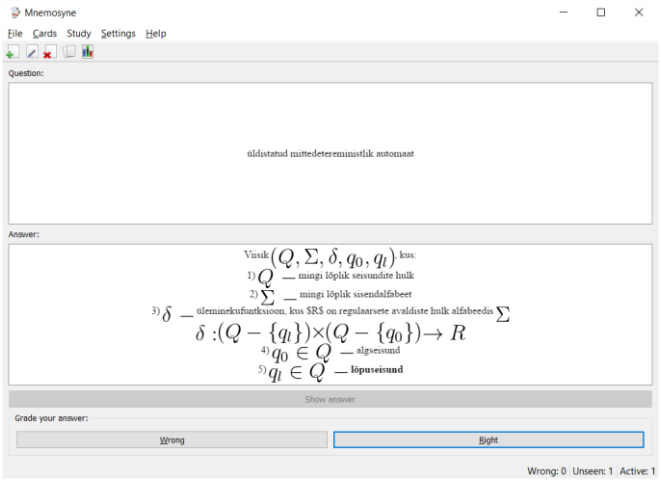
- tasuta kasutamisel on funktsionaalsus piiratud;
- vanamoodne välimus.

1.2.3 Mnemosyne

Mnemosyne on saadaval neljal erineval platvormil (The Mnemosyne Project, kuupäev puudub): Windows; Linux; Mac OS X; Android.

Kahjuks aga puudub sellel iOS versioon ja samuti ei ole saadaval veebiversiooni. Sarnaselt Anki Windowsi versiooniga on ka Mnemosyne Windowsi versioonil vaja LaTeXi ja dvipng pakette, et LaTeXi käsklused töötaksid (The Mnemosyne Project, kuupäev puudub).

Mnemosyne suureks puuduseks on tema kehv tekstiredaktor, milles saab kirjutada ainult tavalist teksti ja kõik muudatused on vaja teha HTML märgendeid kasutades, mis teeb tavakasutajale teksti redigeerimise keeruliseks (vt Ekraanitõmmis 5).



Ekraanitõmmis 5. Mnemosyne'i Windowsi versioon

Mnemosyne'l on kasutusel samuti intervalliga kordamine, milles hinnanguid oma teadmistele saab anda nullist viieni ning vastavalt hinnangule kuvatakse kaart teatud intervalli tagant.

Head omadused:

- LaTeXi tugi;
- intervallidega kordamine.

Puudused:

- kehv tekstiredaktor;
- iOSi tugi puudub;
- tähenduskaardid salvestatakse kohalikku seadmesse.

1.3 Nõuded enda rakenduse loomiseks

Eelpool väljatoodud rakenduste katsetamisel ja analüüsimisel on kõige olulisem erinevate seadmete ja platvormide tugi. Suurimaks puuduseks rakenduste puhul oli eelvaate puudumine. See tähendab, et on raske ennustada, kuidas tegelikult sisestatud LaTeXi valem hakkab välja nägema. CoboCards oli ainuke rakendus, mis suutis kuvada samu tähenduskaarte erinevatel seadmetel, kuid samal ajal oli CoboCards kasutajaliides vanamoodne.

Kõike eelnevat arvesse võttes peab loodav veebirakendus olema võimalikult lihtne. Rakenduses on kaks vaadet: esimeses toimub tähenduskaartide abil õppimine ja teises andmebaasi redigeerimine: kirjete muutmine, kustutamine ja uute kaartide lisamine. Õppimise protsess toimub nii, et kasutajale kuvatakse juhuslikult valitud üks "Teoreetilise informaatika" kursuse eksami mõiste ning arenduse lihtsustamiseks otsustas autor loobuda intervallmeetodist.

Rakendusele seatud nõuded on järgnevad:

- rakendus peab toetama LaTeXi makrokäskude paketti;
- kasutajapõhised tähenduskaardid on sünkroniseeritud kõikidel seadmetel. Mitme seadme tugi on oluline sellepärast, et arvutis on mugavam sisestada LaTeXi valemeid, kuid telefon on alati igal pool kaasas ning seega õppimisel käepärasem;

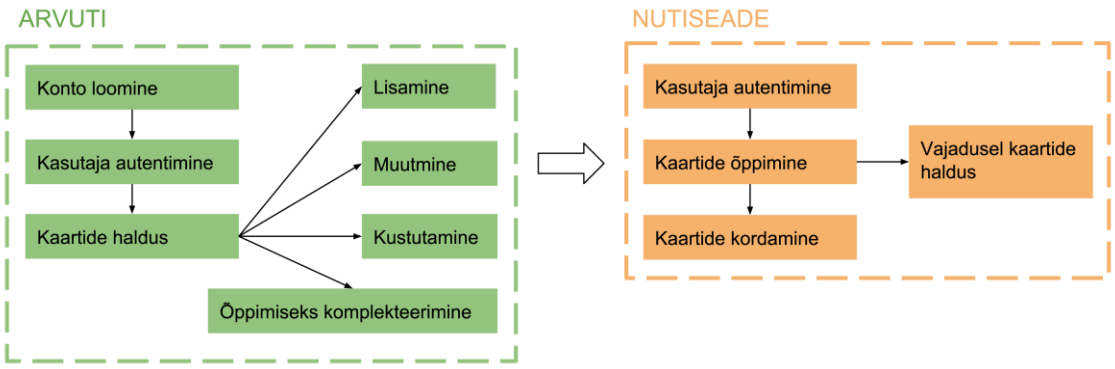
- kui kaart on selgeks õpitud, siis peab olema märgistamise võimalus, et seda enam ei kuvataks;
- kaarte kuvatakse juhuslikus järjekorras, kuid sama kaarti ei kuvata järjest;
- rakendust saab kasutada mitu kasutajat paralleelselt ning arenduse lihtsustamiseks toimub sisselogimine kasutades Google'i kontot;
- rakenduses on mitmekeelsuse tugi.

2 Arhitektuur ja arendus

Kõige mõistlikum ja kiirem on luua piisavalt universaalse koodiga rakendus, mida saab kasutada ilma suuri muudatusi tegemata iga operatsioonisüsteemi jaoks. Selle saavutamiseks tuleb teha veebirakendus, mis on mugandatud võimalikult paljudele ekraanidele, näiteks lauaarvuti monitorile, tahvelarvutile või nutitelefonile. Järgnevalt on toodud lühikirjeldused valitud tehnoloogiatest ja arendusprotsessist.

Töö autor võttis oma veebirakenduse loomisel aluseks Washami veebirakenduse avatud lähtekoodi. proovida kasutada "Teoreetilise informaatika" kursuse teooria õppimise tarbeks. Koodi täiendati enda loodud repositooriumis isikliku veebirakenduse jaoks. Esialgu keskenduti rakenduse funktsionaalsuse loomisele ning kasutajaliidese tõlgiti osaliselt. Järgnevalt tutvustatakse tehnilisi eesmärke, kirjeldatakse arendusprotsessi ning antakse ülevaade tekkinud probleemidest ja muudatustest nende lahendamiseks.

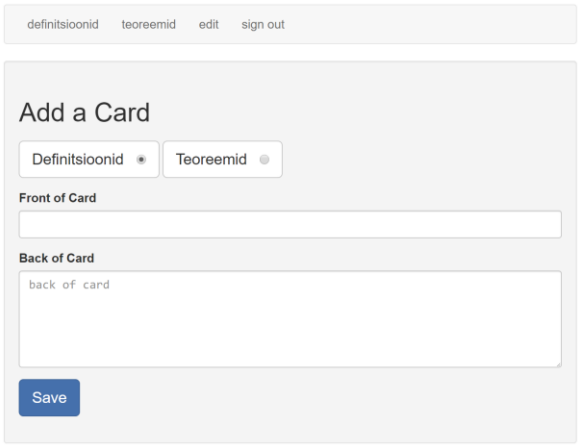
2.1 Rakenduse funktsionaalsus



Joonis 3. Soovituslikud tegevused erinevatel seadmetel

Rakenduse kasutamisel on mõistlikum keerulisemad tegevused nagu näiteks kaardi lisamine teha süle- või lauaarvutis. Õppematerjali omandamist tasub teha nutitelefonis (vt Joonis 3).

Mõistekaartide rakenduse tööpõhimõte seisneb selles, et alguses sisestab kasutaja kõik definitsioonid ükshaaval *edit*-vaates. Iga mõistekaardi puhul on vaja kirjutada pealkiri esiküljele ja selle definitsioon tagaküljele (vt Ekraanitõmmis 6).



Ekraanitõmmis 6. Definitsiooni mõistekaardi loomine

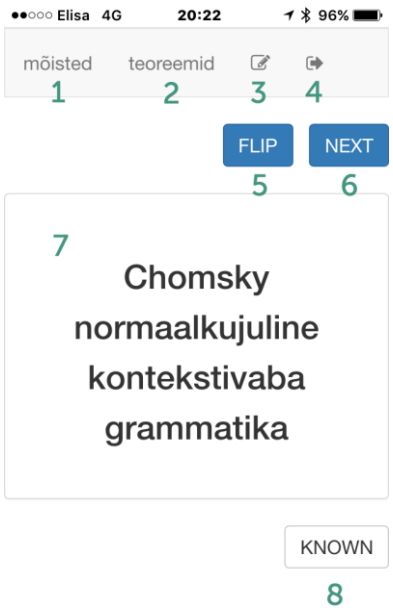
Kõiki sisestatud definitsioone ja teoreeme on võimalik hallata *edit*-vaates (Lisa 1). Mõistekaardi muutmiseks tuleb vajutada pliiatsit kujutavale nupule (Lisa 1, nr 8).

Kokku on viis erinevat filtreerimise võimalust (Lisa 1, nr 7):

- kuvada kõiki kaarte (*all*);
- ainult definitsioone;
- ainult teoreeme;
- selgeks õpitud kaarte (*known*);
- õppimiseks mõeldud kaarte (*unknown*).

Kaartide lisamine toimub samuti *edit*-vaates (Lisa 1, nr 5 ja 6) ja kaartide õppimisprotsessi alustamiseks tuleb vajutada vastavalt soovile kas “definitsioonid” või “teoreemid” lingile (Lisa 1, nr 1 või 2). Kasutajale kuvatakse üks talle veel õppimata mõistekaart, mille juhtimiseks on mõeldud kolm erinevat nuppu (vt Ekraanitõmmis 7):

- *flip* – näitab mõistekaardi tagakülge (nr 5);
- *next* – võtab juhuslikus järjekorras ühe kaardi kogumikust (nr 6);
- *known* – märgistab kaardi selgeks (nr 8).



Ekraanitõmmis 7. Õppimise vaade

2.2 Arendusetapid

Esimene eesmärk oli teha veebirakendus kergesti ning mugavalt kättesaadavaks nii arvutist kui ka nutitelefonist sõltumata kasutaja asukohast. Rakenduse majutamiseks sobis soodne **virtuaalne privaatserver** (ingl *Virtual Private Server*, VPS), sest see annab rohkelt võimalusi operatsioonisüsteemi haldamiseks ja erinevate programmide paigaldamiseks. Toetudes varasemale kogemusele ja heale hinnapoliitikale, osutus valituks DigitalOceani[13] kõige soodsam VPS viie dollari eest kuus. 2018. aastal avastas töö autor veelgi odavama pilvelahenduste platvormi Vultr[14], kust saab tellida VPSi kõigest kahe ja poole dollari eest. Mõistekaartide rakendus töötab edukalt serveris, mille tehnilised andmed on 512 megabaiti operatiivset mälu, 20 gigabaiti pooljuhtketast ja operatsioonisüsteemiks näiteks Debian[15] 9 x64.

Lihtsustamaks rakenduse paigaldamist kasutatakse **Dockerit**[16]. Docker on tagaplaanil jooksev käsurea programm ja kaugteenuste kogumik, mis kasutab logistilist

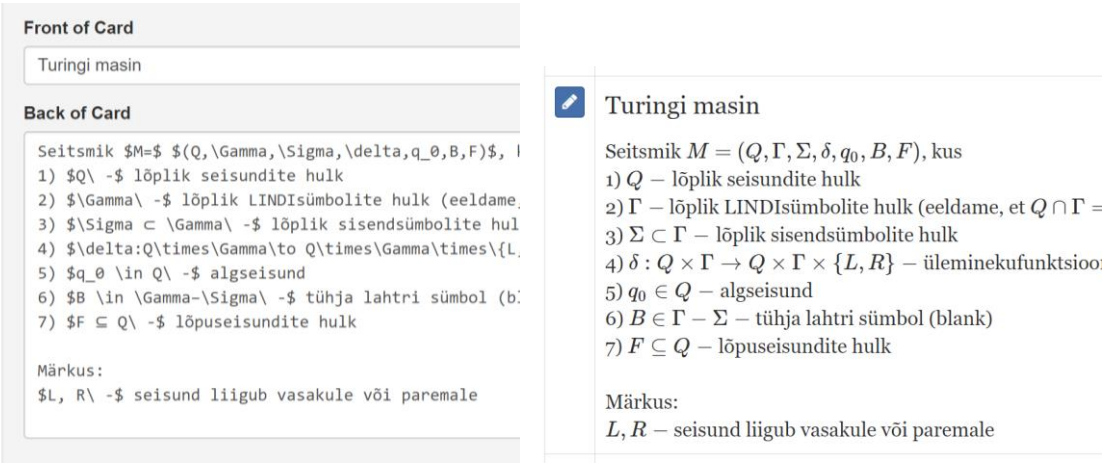
[13] https://www.digitalocean.com
[14] https://www.vultr.com
[15] https://www.debian.org
[16] https://www.docker.com

lähenemisviisi tavapäraste tarkvaraprobleemide lahendamisel ja lihtsustab tarkvara paigaldamise, käitamise, avaldamise ja eemaldamise protsessi (Nickoloff, 2016).

Projekti seadistamise alguses tekitab Docker tarkvarale juurde lisakeerukuse, kuid seeläbi saavutab eelise tagades, et kohalik arenduskeskkond on identne mistahes võimaliku keskkonnaga (näiteks Ubuntu[17] serveris töötav Docker), kuhu soovitakse rakendust paigaldada (Cook, 2017).

Mõistekaartide sisestamise käigus tuli leida hea lahendus matemaatiliste valemite sisestamiseks korrektses formaadis. Erinevaid võimalusi uurides, leidis töö autor, et Javascripti[18] teek **MathJax**[19] konverteerib LaTeXiga tehtud valemeid täpselt sellisteks nagu "Teoreetilise informaatika" kursuse materjalides.

Kuna valemite kuvamine nõuab konverteerimist, siis on oluline andmete sisestamisel eristada valemeid tavatekstist. Selle saavutamiseks tuli välja valida erisümbol, mis märgistab, kust valem algab ja kus lõpeb. Antud programmi jaoks piisavalt universaalseks tähemärgiks valiti dollarimärk (**$**) (vt Ekraanitõmmis 8).



Ekraanitõmmis 8. LaTeXi kasutamine mõistekaardi muutmisel ja eelvaade

Esialgu oli mõte veebirakendust juhtida kas arvutihiirega või näpuga, kuid arvuti taga istudes ei olnud see mugav. Korduvate tegevuste kiirendamiseks ja lihtsustamiseks loodi

[17] https://www.ubuntu.com/server
[18] https://www.javascript.com
[19] https://www.mathjax.org

kiirklahvid erinevate tegevuste jaoks. Kiirklahvide defineerimiseks võttis töö autor kasutusel **jQuery**[20] teegi.

Kui kõik tähenduskaardid on selgeks märgitud, siis võib õppijal tekkida soov uuesti kogu teooria üle korrata. Et vältida olukorda, kus kasutaja peab ükshaaval igat kaarti muutma "mitte selgeks", oli vaja teha lisaarendus uue nupu näol, millele vajutamine muudaks iga sisestatud andmebaasi kirje (mõistekaardi) "mitte selgeks" (Lisa 2, nr 1).

2.3 Rakenduse täiendatud versioon

Veebirakenduse esimeses versioonis oli palju puudusi. Esiteks polnud võimalik mitmel kasutajal korraga õppida. Samuti polnud mitte mingisugust registreerumise võimalust. Ka GitHubis olev repositoorium ei motiveerinud üliõpilasi piisavalt, et endale raha eest näiteks DigitalOceanisse rakendus paigaldada.

Lisaks oli suureks puuduseks see, et lehel navigeerides pidi iga kord täielikult lehe uuesti laadima, mistõttu veebileht oli aeglane. Samas saab rakenduse täielikult ümber kirjutada nii, et kogu programmeerimisloogika tehakse veebilehitsejas kasutades JavaScripti – sellist rakendust nimetatakse üheleherakenduseks ehk SPA (ingl *Single Page Application*, SPA). Kui SPA rakendus laetakse brauserisse, siis tulemuseks on üks HTML-i fail koos viidetega erinevatele JavaScripti failidele, mis suudavad omakorda serveerida veebirakenduse teisi lehti. Selline lähenemisviis suudab genereerida sadu lehti andmetest, mida saadakse veebi API-liidese[21] kaudu (Fox, 2018).

Täiendatud versiooni raames arendati juurde järgmised funktsionaalsused:

- Google'i kaudu registreerumine ja autentimine;
- mitmekeelsuse tugi;
- võimalus kaarte vaadata, sorteerida, muuta, lisada ja kustutada;
- "Teoreetilise informaatika" kursuse eksami sõnaseletuskaartide sisestamine CSV-failist;
- kasutajaliidese parandused;
- erinevatele ekraanisuurustele kohanduvad vaated;

[20] http://jquery.com
[21] Reeglistik, mille alusel rakendusprogramm kasutab teise rakendusprogrammi teenuseid (Vallaste, kuupäev puudub).

- optimisatsioon, et veebilehe uuesti laadimine ei toimuks iga tegevuse peale, vaid vajalikud andmed laetakse serverist ja kuvatakse;
- Angulari moodul LaTeXi visualiseerimiseks;
- võimalus näha eelvaadet enne salvestamist;
- dünaamiline päringine andmebaasisit, mis muutis veebilehe kiiremaks, sest andmebaasist võetakse viimased 20 kaarti ning kerimisriba alla kerides päritakse dünaamiliselt serverist veel 20 kaarti.

2.4 Kasutatud tehnoloogiad

Bakalaureusetöö mahu piirangute tõttu antakse ülevaade ainult kõige olulisematest tehnoloogiatest. Lähtudes eelnevalt mainitud asjaoludest uuris autor hetkel populaarseid JavaScripti teeke, mis aitaksid lihtsama vaevaga luua SPA rakendust. Erinevatele allikatele tuginedes on populaarseimateks järgnevad veebiraamistikud: React[22], Angular[23] ja Vue.js[24]. Järgnevalt selgitab autor, miks ta otsustas edasi liikuda Angulariga.

Angulari kasutamine eesrakenduse platvormina on tulevikku vaatav suund, sest selle arendamisel lööb kaasa Google kui ka suur hulk arendajaid. Google on üks peamine eraettevõte, mis teeb mitmeid Interneti tulevikku mõjutavaid arhitektuurilisi otsuseid. Autoril oli varsem kogemus Angulari esimese versiooniga (AngularJS) ning kuna viimane versioon on ülesehituselt väga erinev, siis uue versiooni õppimiseks sobib taolise rakenduse tegemine väga hästi.

Lihtsama veebirakenduse loomisel tasub kasutada monoliitset arhitektuurimustrit, mis võimaldab järgmist (Sasidharan & Nellaiyapen, 2018):

- toetab erinevaid kliente: nii arvuti- kui ka mobiilibrausereid, ja ka arvuti- ja mobiilirakendusi (ingl *native applications*);
- API-liidestust võivad kasutada kolmandad osapooled;
- võib integreeruda teiste rakendustega, näiteks RESTi[25] või sõnumite ootejärjekorra kaudu (ingl *message queue*);

[22] https://reactjs.org
[23] https://angular.io
[24] https://vuejs.org
[25] REST on hüperteksti edastamiseks kasutatav klient-server protokoll.

- võib hallata HTTP[26] päringuid, täita äriloogikat, suhelda andmebaasiga ning vahetada andmeid teiste süsteemidega;
- lisades arvutusvõimsust saab skaleerida vertikaalselt või koormuse tasakaalustamisega (ingl *load balancing*) horisontaalselt.

2.4.1 JHipster

Uurides tänapäeva tarkvara arendusprotsesse tuleb ilmsiks tõsiasi, et ligi 70% arendajate ajast kulub projekti ülesseadmisele ning esialgse struktuuri loomiseks (Sasidharan & Nellaiyapen, 2018). Selle lihtsustamiseks on mitmeid erinevaid viise ning antud töö autor otsustas kasutada JHipster toodet.

JHipster[27] on 2013. aastal välja tulnud tasuta avatud lähtekoodiga rakenduse generaator tänapäevaste veebirakenduste ja mikroteenuste kiireks arendamiseks, kasutades Angular ja Spring raamistikku. JHipster pakub tööriistu, et genereerida projekti Java pinuga (ingl *stack*) serveripoolel (kasutades Spring Booti) ja kliendipoolel Angulari ning 2018. aastal lisandus ka Reacti tugi. Lisaks aitab Bootstrap 4 muuta veebirakendusi dünaamiliselt kohanduvateks kõikidele ekraanidele (Raible, 2018).

Tänapäeval on JHipster on muutunud populaarseks ka ettevõtete maailmas, keskendudes tugevalt arendaja produktiivsusele, tööriistadega varustamisele ja kvaliteedile. Ametliku statistika järgi oli 2018. aasta alguses genereeritud igakuiselt vähemalt 5000 rakendust ning JHipsteri generaator on paigaldatud umbes üks miljon korda (Sasidharan & Nellaiyapen, 2018).

JHipsteri eesmärk on pakkuda arendajatele platvormi, mille abil põhiline tähelepanu on pööratud äriloogika teostamise peale. Programmeerija ei pea raiskama aega uuesti jalgratta leiutamise peale ning muretsema, kuidas erinevad tehnoloogiad seadistada ja omavahel koos tööle panna.

2.4.2 Angular

Kui Angulari eelmise põlvkonna raamistik AngularJS[28] oli ehitatud kui mudel-vaade-kontroller (ingl *Model-View-Controller*, MVC), siis alates Angulari teisest versioonist on

[26] HTTP on hüperteksti edastusprotokoll, mida kasutatakse andmete edastamisel veebis.
[27] https://www.jhipster.tech
[28] https://angularjs.org

arhitektuur muutnud komponendipõhiliseks. Suurimaks eeliseks on see, et mitu veebirakenduse arendajat saavad paralleelselt luua uut funktsionaalsust nii, et tekib koodis võimalikult vähe konflikte. Komponendipõhise arhitektuuri head omadused:

- **Kapseldamine** (ingl *encapsulation*). Sarnased komponendid on kapseldatud ja neid on lihtne taaskasutada kogu veebirakenduse ulatuses. Lisaks tõstab kapseldamine koodi loetavust ning arendusprojektiga alles liitunud programmeerijad muutuvad produktiivseks üsna kiiresti.
- **Lihtsasti testitav**. Komponendid on väiksemad eraldiseisvad üksused ja neid on lihtsam testida.
- **Parem hooldatavus**. Komponente, mis on üksteisest lahutatud on lihtsamad vahetada.

Angular on kirjutatud **TypeScripti**[29] programmeerimiskeelega, mis on põhimõtteliselt pealisehits JavaScripti jaoks. TypeScript kompileerub täielikult JavaScriptiks ning aitab vältida üldlevinuid vigu, mis on tingitud muutujate tüüpidest. Google'i Angulari arendusmeeskond soovitab Angulari veebirakendusi kirjutada kasutades TypeScripti (Google, 2016). Kuigi väiksemate projektide puhul saab ka ilma hakkama, siis äriprojektides peab kood olema võimalikult hästi loetav ja kvaliteetne. Angulari meeskonna juhtivarendaja Victor Savkini (2016) arvates võimaldab TypeScript parimat koodi autokompileerimist, navigatsiooni ja redigeerimist (ingl *refactoring*). Selliste tööriistade puudumisel muutub suurema koodibaasi täiustamine riskantseks ja kulukaks.

2.4.3 Spring Boot

Spring boot on tänapäeval laialdaselt kasutatav lahendus serveripoolsete Java rakenduste loomisel (Sasidharan & Nellaiyapen, 2018). Spring Booti raamistik oli valitud sellepärast, et see pakub mitmeid eeliseid:

- toetab MVC arhitektuurimustrit;
- andmebaasi ühenduse seadistamiseks on võimalik lisada vajalik Springi moodul;
- sõltuvuste injektsioon (ingl *dependency injection*) on lihtsasti konfigureeritav ja seepärast arendamine mugav (Suryotrisongko, Jayanto, & Tjahyanto, 2017);

[29] https://www.typescriptlang.org

- toetab aspekt-orienteeritud programmeerimist (ingl *aspect-oriented programming*, AOP). AOP on programmeerimise paradigma, mille idee seisneb selles, et tarkvara programm jagatakse mooduliteks vastavalt funktsionaalsusele (Kumar, Kumar, & Iyyappan, 2016)
- kasutab sisseehitatud Java servleti (ingl *servlet*) konteinerit, nagu näiteks Tomcat[30], Jetty[31] või Undertow[32] (Sasidharan & Nellaiyapen, 2018). Antud töö raames kasutatakse Undertow-d, mis Andy Beck (2017) järgi on et kõige kiirem konteiner;
- erinevate profiilidega on võimalik lihtsasti muuta rakenduse konfiguratsiooni ning rakenduse käivitamisel vajalik profiil ette anda;
- pakub integratsioonitestide jaoks utiliite.

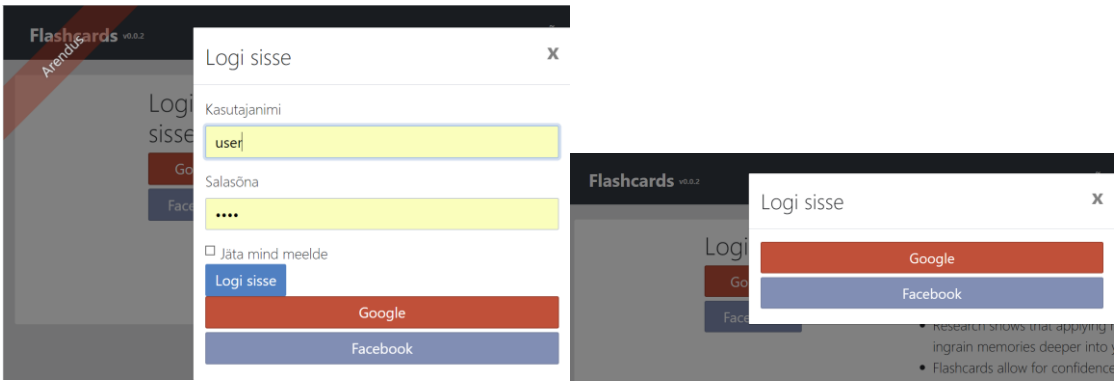
Spring Booti puuduseks võib lugeda seda, et mida suuremaks muutub rakendus seda aeglasemaks muutub tema käivitamine.

2.5 Arenduse olulisemad pidepunktid

Antud töö raames oli kaks arenduskeskkonda: kohalik (dev.ati.ee) ja toodang (ati.ee). Kohalikus keskkonnas olev IP aadress oli suunatud dev.ati.ee domeeni vastu. Vajalikud seadistused tehti Windowsi operatsioonisüsteemi *hosts*-failis. See muudatus oli vajalik selleks, et ka kohalikus arvutis töötaks Google'i kaudu autentimine.

Erinevad keskkonnad võimaldavad lihtsustada arendust nii, et saab rakendusse sisse logida ilma Google'i teenuseid kasutamata (vt Ekraanitõmmis 9). Näiteks, kui autentimine töötab tekstipõhise SMSi kaudu peab iga sõnumi pealt maksma teenusepakkujale, aga kasutajaga sisselogides saab raha kokku hoida.

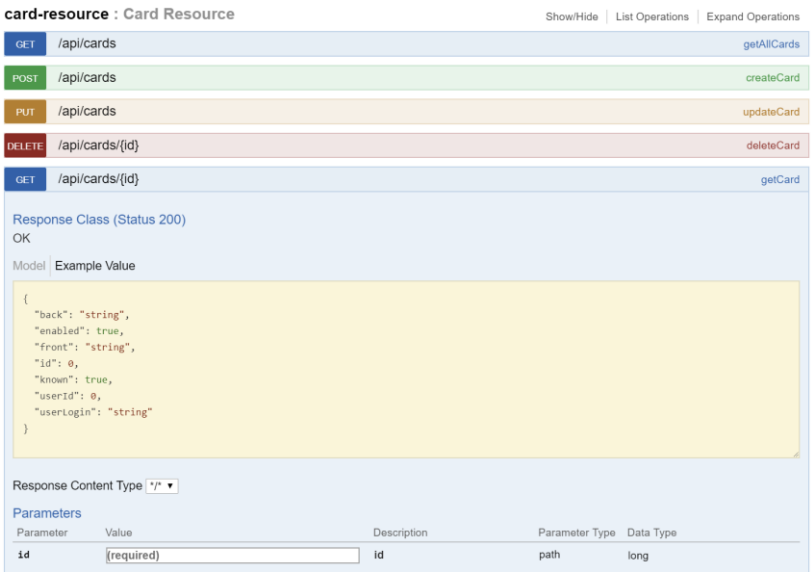
[30] http://tomcat.apache.org
[31] https://www.eclipse.org/jetty
[32] http://undertow.io



Ekraanitõmmis 9. Vasakul on kohalik keskkond ja paremal on toodangu keskkond

Antud töö raames valminud rakendus kasutab kohalikus keskkonnas mälupõhist **H2**[33] andmebaasi. See asjaolu kiirendab arendust seetõttu, et igal programmi käivitamise korral saab olla kindel, et andmebaasiga ühendus on loodud ning andmed salvestatakse. Toodangu keskkonnas kasutatakse **PostgreSQLi**[34] andmebaasi, sest andmeid on vaja salvestada kõvakettale.

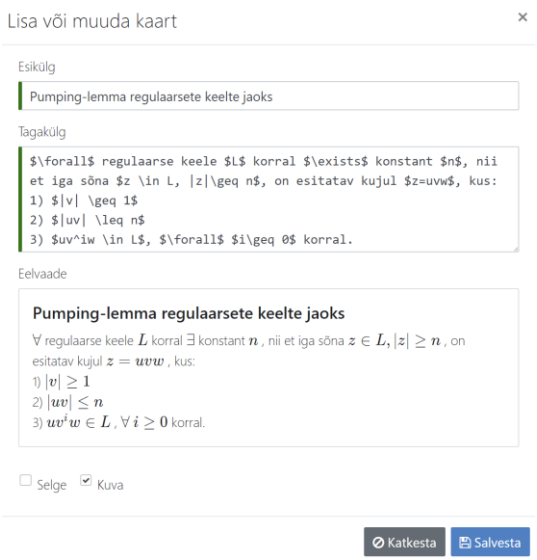
Serveripoolse arenduse lihtsustamiseks kasutati **Swaggeri** tööriista. Swaggeri veebipõhine kasutajaliides on utiliit, mis võimaldab lihtsustada REST-põhise teenuse arendamist, dokumenteerimist ja testimist. Swaggeriga saab katsetada kõiki sisse- ja väljaminevaid päringuid vastavalt rakenduse ärireeglitele (vt Ekraanitõmmis 10).



Ekraanitõmmis 10. Swaggeri utiliidi vaade

[33] http://www.h2database.com
[34] https://www.postgresql.org

Suurimaks komistuskiviks osutus LaTeXi kuvamine, sest arendusprojekti tegemise hetkel polnud saadaval teeki, mis toetaks kõige uuemat Angulari versiooni. Küll aga leidis käesoleva töö autor avatud lähtekoodiga ng-KaTeX[35]i mooduli, mille autor otsustas aluseks võtta ja täiendada (vt Ekraanitõmmis 11).



Ekraanitõmmis 11. Tähenduskaardi muutmise vaade, kus kuvatakse LaTeXi kasutamist

Kasutajaliidese parandamise huvides loobuti kerimisriba (ingl *scrollbar*) kasutamisest tähenduskaardi tagakülje tekstiväljal ning arendati Angulari direktiiv, mis suudab automaatselt muuta tekstivälja suurust vastavalt sisule.

Tähenduskaartide sisetamisel võib tekkida lohakusvigu, mille vältimiseks võeti kasutusele validatsioonid, mis kontrollivad, et kõik vajalikud väljad oleksid täidetud nii kliendi poolel kui ka andmebaasi tasandil.

Kuna esialgses versioonis kasutati SQLite andmebaasi, siis oli vaja leiutada viis, kuidas kõik tähenduskaardid üle viia H2 või PostgreSQL andmebaasi. Selle ülesande lahendamisel migreeriti kõik andmed CSV-faili **DB Browser for SQLite**[36] utiliidi abil. Ati.ee veebikeskkonnas kasutatakse kasutaja loomise hetkel **SimpleFlatMapperi**[37] teeki, mille abil salvestatakse kasutaja andmebaasi ning tema konto alla luuakse kõik CSV-failis olevad sõnaseletuskaardid.

[35] https://github.com/garciparedes/ng-katex
[36] https://github.com/sqlitebrowser/sqlitebrowser
[37] https://simpleflatmapper.org

Veel võib lisada, et andmete üleviimine ühest andmebaasist teise oli üsna keeruline protsess. Probleeme põhjustas vale tekstikodeering ja seetõttu pidi kogu teksti uuesti redigeerima.

3 Rakenduse testimine

Autor testis rakendust, et leida potentsiaalseid probleeme ja saada ideid edasiarenduseks. Veebirakenduse testimine viidi läbi arvuti ja mobiiltelefoni brauserites, et testida rakendust erinevatel seadmetel. Nutitelefoniks oli kasutada Apple iPhone Xs ja arvutiks 2017. aasta Apple Macbook Pro 15".

Ati.ee kasutajakogemuse testimiseks rakendati valjusti mõtlemise (ingl *think-aloud protocol*) meetodit üliõpilastega, kes õpivad või on õppinud "Teoreetilise informaatika" ainekursusel ega ole autori veebiprogrammiga varem kokku puutunud. Testsessiooni käigus paluti kasutajatel kasutada rakendust, et läbida konkreetseid ülesandeid. Nõnda saab kõige paremini tajuda kasutajate mõtlemist, sest meetod keskendub tööülesannete täitmisele, mitte tunnetele (Lin, Chen, Chen, & Yuan, 2018).

Iga testimise sessiooni läbiviimisel osales kaks inimest: testija ja käesoleva töö autor, kes lindistas kogu protsessi nii helis kui ka pildis, et hiljem dokumenteerimise käigus mitte ükski detail ei läheks kaduma.

Kasutades nelja teststsenaariumi, lasi autor kasutajatel testida ati.ee rakenduse põhifunktsionaalsusi ja nendega seotud tegevusi alguses sülearvutis ja seejärel nutitelefonis (vt Lisa 3):

- Esimeses teststsenaariumis oli kasutaja sisseelamise (ingl *onboarding*) jälgimine, kus paluti konto loomiseks sisse logida kasutades Google'i kontot.
- Teises teststsenaariumis oli põhifookuses kaartide kategooria vaade, kus lasti testijal muuta mõnda mõistekaarti valemit. Eesmärgiks oli mõista, kas kasutaja mõistab, kuidas antud rakenduses saab valemite loomiseks tarvitada LaTeXi makrokäskude paketti.
- Kolmandas teststsenaariumis paluti lisada uus sõnaseletuskaart ja see järgnevalt koheselt kustutada.
- Neljandas teststsenaariumis lasti proovida õppimise funktsionaalsust.

Nelja stsenaariumi testimiseks kulus umbes 55 minutit. Tulemuste ja ettepanekute põhjal võib järeldada, et kasutajaliides on loogiliselt üles ehitatud, küll aga vajab juurde funktsionaalsusi nagu näiteks uue kasutaja loomine ilma sotsiaalvõrgustiku keskkonda kasutamata. Antud tulemusi arvestab autor ati.ee rakenduse viimistlemisel.

Testimise käigus ilmnenud veebirakenduse kitsaskohad ja soovitused on välja toodud antud töö lisades (vt Lisa 3).

Olulisemad soovitused võib sõnastada järgmiselt:

- lahti seletada, kuidas rakendust kasutada;
- tuua rohkem esile tähenduskaardi tagakülg, et see eristuks paremini esiküljest;
- nutitelefonis on navigeerimisnupud ebavajalikud ning nende asemel soovitati kasutada näpuga lohistamist või viipamist;
- muuta nutitelefoni vaade võimalikult lihtsaks ja kõrvaldada visuaalsed ebakõlad;
- andmete muutmistel tehtud muudatused peavad igal juhul tagataustal salvestuma iga natukese aja tagant;
- lisada kaartide otsimise ja filtreerimise funktsionaalsus;
- lisada rohkem rohkem informatiivset sisu kaartide kustutamise hüpikaknasse;
- võimaldada nn klassikaline kasutaja loomise viis, kus registreerimine ja autentimine toimub e-posti kaudu;
- puuduvad tõlkeid.

Mõtted edasiarenduseks:

- kasutaja registreerimine ilma sotsiaalvõrgustikku kasutamata;
- Facebooki sisselogimise aktiveerimine;
- Leitneri süsteemi algoritmi väljamõtlemine ja integreerimine;
- proovida koormuse tasakaalustamist, sest nii saab teha õmblusteta (ingl *seamless*) toodangu tarneid.

Kokkuvõte

Käesolev bakalaureusetöö lähtus probleemist, et teoreetilise informaatika kursuse eksami sooritusprotsent on olnud viimastel aastatel madal ning uusi teadmisi tänapäeva keskkonnas on vaja omandada kiiresti ja efektiivselt.

Eesmärgi saavutamiseks määratleti kõigepealt arendatava projekti struktuur – mida ja millises järjekorras teha. Töös püstitatud probleemist paremaks arusaamiseks tehti laialdaselt levinud iseõppimisvõimaluste analüüs. Uuriti olemasolevaid rakendusi, et selgitada välja, milliseid on head omadused ja mida vältida.

Rakenduse disainimise ja arendamise etapis selgitati välja, millistele nõuetele peab loodava rakenduse funktsionaalsus vastama. Arenduse faasis programmeeriti ning kirjeldati arenduse käigus tekkinud põhilisi probleeme ja nende lahendusi.

Loodud rakendust testiti kolme inimesega. Tulemuste ja ettepanekute põhjal võib järeldada, et rakenduse ülesehitus on intuitiivne ja vastab püstitatud eesmärkidele, kuid vajab juurde funktsionaalsusi nagu näiteks kasutusjuhend. Tagasiside saamine on tähtis, sest ainult seeläbi saab tuvastada rakenduse kitsaskohad, mida tulevikus saab parandada.

Käesoleva töö raames valmis Flashcards veebirakendus, mis on tasuta kättesaadav kõikidele huvilistele aadressil www.ati.ee. Antud keskkonnas saab valmistuda “Teoreetilise informaatika“ kursuse eksamiks, kuid ka sisestada uut õppimismaterjali ja muuta olemasolevat. Loodud rakendust saab mugavalt kasutada nii sülearvutis kui ka nutitelefonis.

Bakalaureusetöö autorile andis tehtud eeltöö ja rakenduse arendamine arvestataval määral uusi teadmisi kohanduva disainiga veebirakenduse loomisel ning suurendades huvi moodsate tehnoloogiate vastu, nagu näiteks Angular, Spring Boot, JHipster, Swagger, Bootstrap jpm.

Kasutatud kirjandus

Kollegium des Clavius-Gymnasiums. (2016). Loetud aadressil https://perma.cc/4DVS-KSU4

Anki. (kuupäev puudub). *Powerful, intelligent flash cards.* Loetud aadressil https://apps.ankiweb.net/

Baleghizadeh, S., & Ashoori, A. (2011). The Impact of Two Instructional Techniques on. *MEXTESOL*.

Ball, K. (2018. 19. juuni). *Github Stars !== Usage: React is still blowing Vue and Angular Away*. Loetud aadressil https://zendev.com/2018/06/19/react-usage-beating-vue-angular.html

Beck, A. (2017, 26. jaanuar). *Tomcat vs. Jetty vs. Undertow: Comparison of Spring Boot Embedded Servlet Containers*. Loetud aadressil https://examples.javacodegeeks.com/enterprise-java/spring/tomcat-vs-jetty-vs-undertow-comparison-of-spring-boot-embedded-servlet-containers/

CoboCards. (kuupäev puudub). *The best online flashcard software.* Allikas: CoboCards: https://www.cobocards.com

Cook, J. (2017). *Docker for Data Science.* Santa Monica: Apress.

deWinstanley, P. A., & Bjork, R. A. (2002). Successful Lecturing: Presenting Information in Ways That Engage Effective Processing. *New Directions for Teaching and Learning*.

Dizon, G., & Tang, D. (2017). Comparing the efficacy of digital flashcards versus paper flashcards to improve receptive and productive L2 vocabulary. *The EUROCALL Review*.

Donovan, J., & Radosevich, D. J. (1999). A metaanalytic review of the distribution of practice effect. *Journal of Applied Psychology*.

Ebbinghaus, H. (1885). *Memory, a Contribution to Experimental Psychology.*

Fox, R. (2018). Information economy. *Digital Library Perspectives*.

Google. (2016, 14. september). *Angular - What is Angular?* Loetud aadressil https://angular.io/docs

Harris, R. (2014, 27. veebruar). *Learning Strategy 10: The Leitner Flash Card System*. Loetud aadressil https://www.virtualsalt.com/learn10.html

Heron, T., Heward, W., Cooke, N., & Hill, D. (1983). Evaluation of a classwide peer tutoring system: First graders teach each other sight words. *Education and Treatment of Children*.

Houten, R. V., & Rolider, A. (1989). An analysis of several variables influencing the efficacy of flash card instruction. *Journal of Applied*.

Kang, S. (kuupäev puudub). *Ask the Scientist: Which is the Best Way to Study? How Often? Does Cramming Work?* Allikas: Temporal Dynamics of Learning Center: https://perma.cc/4VK8-XHFK

Kornell, N. (2009). Optimising Learning Using Flashcards: Spacing Is More Effective Than Cramming. *Applied Cognitive Psychology*.

Kumar, A., Kumar, A., & Iyyappan, M. (2016). Applying Separation of Concern for Developing Softwares Using Aspect Oriented Programming Concepts. *Procedia Computer Science*.

Kwantlen Polytechnic University. (kuupäev puudub). *Spaced Repetition: Remembering What You Learn.* Loetud aadressil https://perma.cc/6JR6-7LRH

Lamport, L. (1994). *LATEX: A Document Preparation System.* Massachusetts: Pearson Education.

Leitner , S. (1972). *So lernt man lernen. Angewandte Lernpsychologie - ein Weg zum Erfolg.* Freiburg: Herder.

Lin, T., Chen, W., Chen, L., & Yuan, P. (2018). Automated comprehensive evaluation approach for user interface satisfaction based on concurrent think-aloud method. *Universal Access in the Information Society*.

Maheady, L., & Sainato, D. M. (1985). The effects of peer tutoring upon the social status and social interaction patterns of high and low status elementary school students. *Education and Treatment of Children*.

Nakata , T. (2008). English vocabulary learning with word lists, word cards and computers: implications from cognitive psychology research for optimal spaced learning. *ReCall*.

Nickoloff, J. (2016). *Docker In Action.* Shelter Island: Manning Publications.

Olenick, D. L., & Pear, J. J. (1980). Differential reinforcement of correct responses to probes and prompts in picture-name training with severely retarded children. *Journal of Applied Behavior Analysis*.

Oxford, R., & Crookall, D. (1990). Vocabulary Learning: A Critical. *TESL Canada Journal*.

Raible, M. (2018). *The JHipster Mini-Book.* C4Media.

Sasidharan, D. K., & Nellaiyapen, K. S. (2018). *Full Stack Development with JHipster.* Birmingham: Packt Publishing.

Savkin, V. (2016, 22. juuli). *Angular: Why TypeScript?* Allikas: Angular: https://vsavkin.com/writing-angular-2-in-typescript-1fa77c78d8e8

Suryotrisongko, H., Jayanto, D. P., & Tjahyanto, A. (6. november 2017. a.). Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot. *Procedia Computer Science*.

Teninbaum, G. H. (2017). Spaced Repetition: A Method For Learning More Law In Less Time. *Journal of High Technology Law*.

The Mnemosyne Project. (kuupäev puudub). *Documentation*. Loetud aadressil https://mnemosyne-proj.org/help/index.php

The Mnemosyne Project. (kuupäev puudub). *Download*. Loetud aadressil https://mnemosyne-proj.org/download-mnemosyne.php

Ullman, M. T., & Lovelett, J. T. (2018). Implications of the declarative/procedural model for improving second language learning: The role of memory enhancement techniques. *Second Language Research*.

Vallaste, H. (kuupäev puudub). *Uued sõnad.* Allikas: e-teatmik: http://vallaste.ee

Wozniak, P. (1999). *Effective learning: Twenty rules of formulating knowledge*. Loetud aadressil https://www.supermemo.com/en/articles/20rules

Young, C. C., Hecimovic, A., & Salzberg, C. L. (1983). Tutor–tutee behavior of disadvantaged kindergarten children during peer teaching. *Education and Treatment of Children*.

Summary

The Development of a Self-study Web Application Using Angular and Spring Boot

Bachelor's Thesis

The goal of this Bachelor's thesis was to research different techniques for memorizing information and to develop a web application *Flashcards* on the JHipster platform. The purpose of this application is to help students study more efficiently for their exam in Theoretical Computer Science.

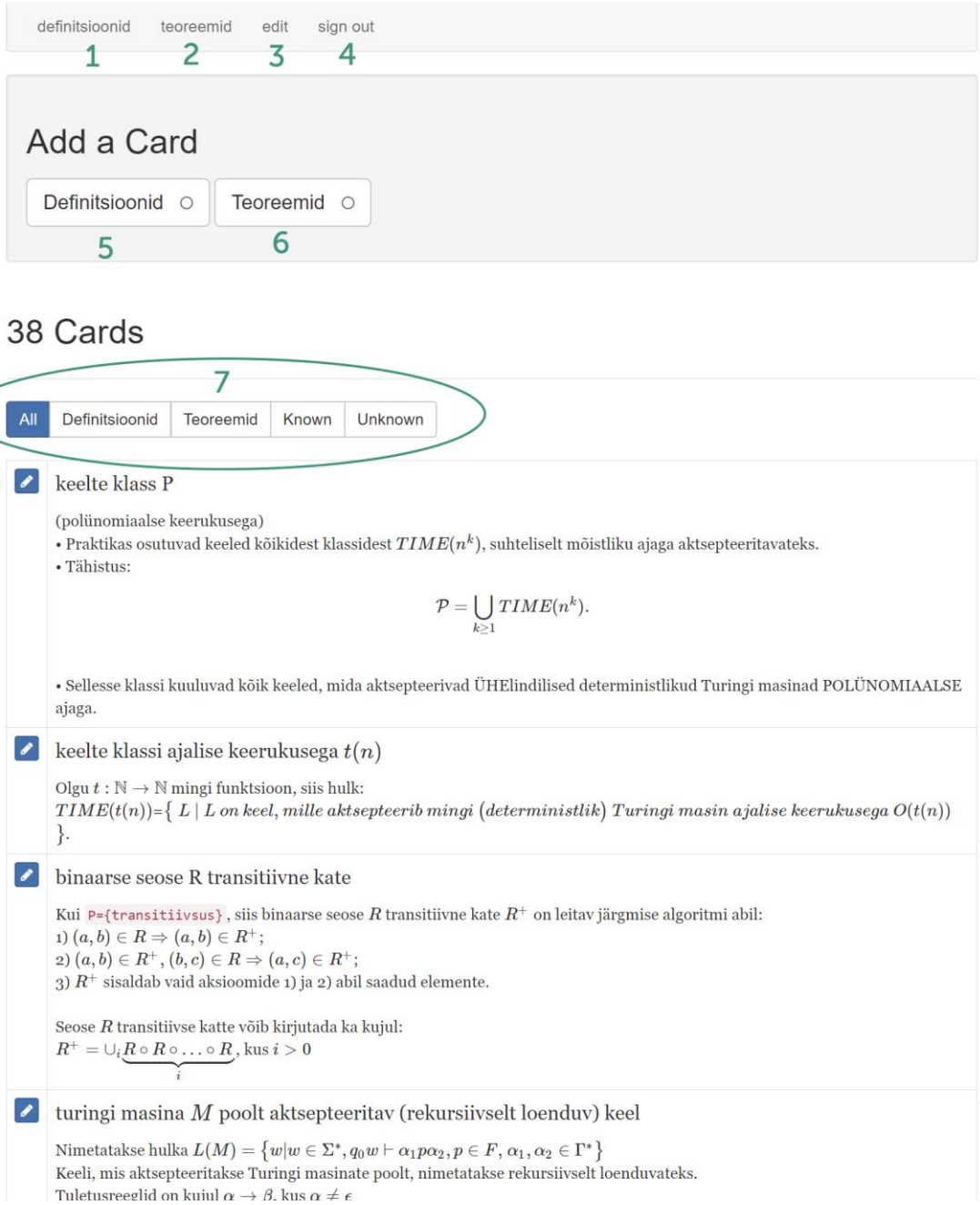
To reach this goal, the following questions were researched and answered:

- what are the self-study techniques;
- which technologies to use to create an efficient self-study web application;
- the user experience with the developed web application.

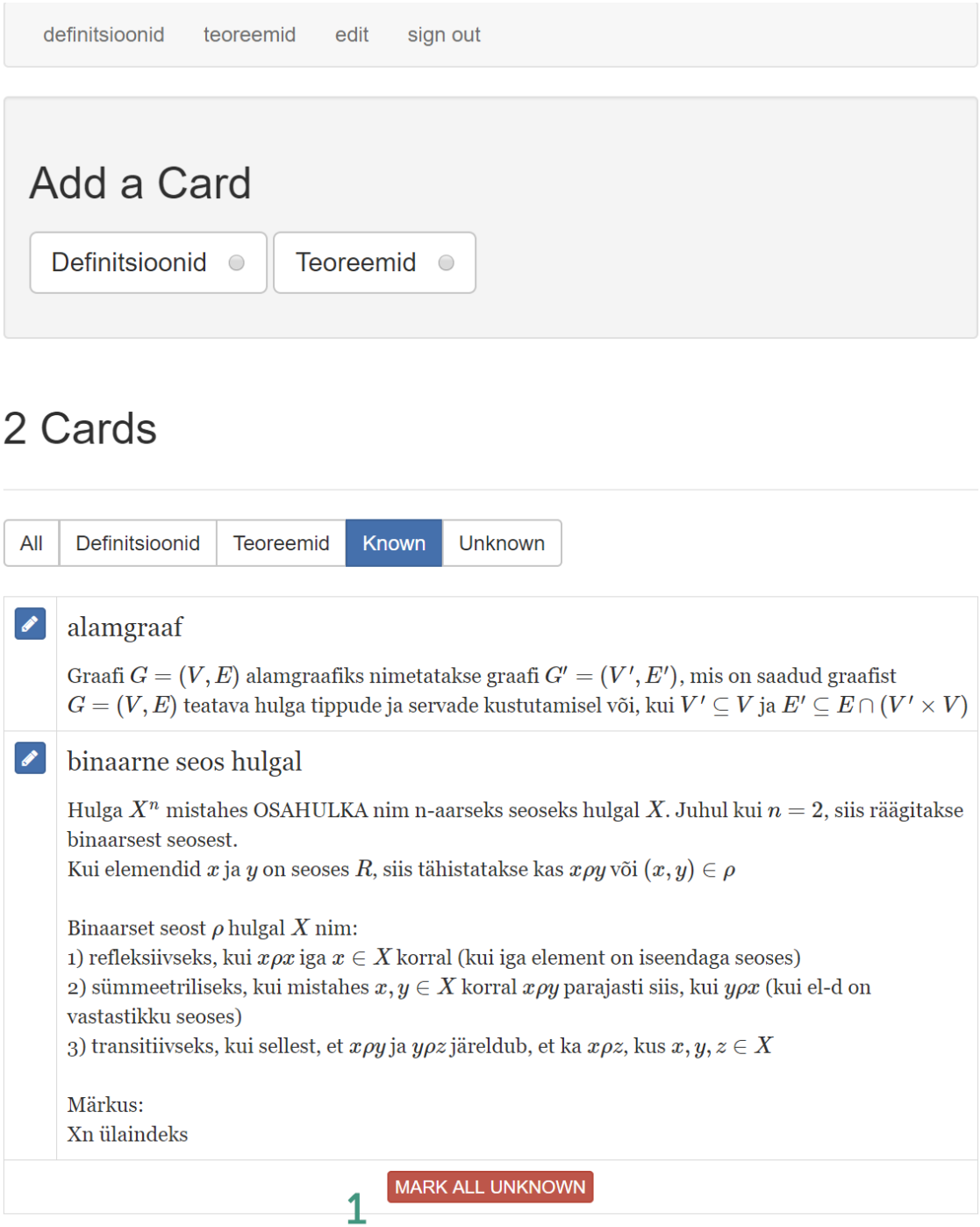
The first chapter of this thesis describes the background of the problem and the techniques of efficient learning. The second chapter describes the design of the Flashcards application, giving an overview of the application's architecture and development phases. This chapter also gives an overview of some of the problems encountered during the development. The final chapter describes the testing process of the created web application. Based on the outcome of the testing, the prospect for future development of Flashcards is introduced.

LISAD

Lisa 1. Rakenduse kaartide vaade

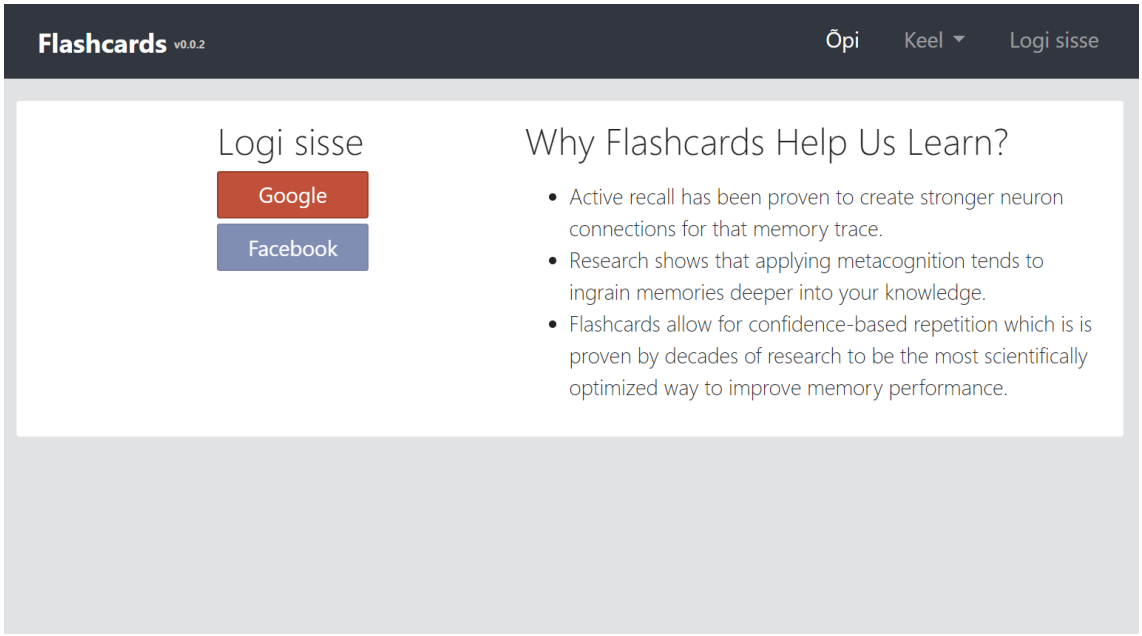


Lisa 2. Rakenduse täiendatud kaartide vaade



Lisa 3. Teststsenaariumite tulemused

Esimene teststsenaarium sülearvutis: „Konto loomine Google'i kaudu" (vt Ekraanitõmmis 12).



Ekraanitõmmis 12. Maandumisleht

Testija nr 1 täheldas, et keskkonda sisenemiseks pakuti võimalust ennast tuvastada ainult Google'i konto kaudu. Pealehel oli ka Facebooki nupp, millele vajutamisel aga ei juhtunud midagi (funktsionaalsus puudub). Sisselogimisel suunatakse kasutaja ümber Google'i sisselogimise lehele ja pärast edukat autentimist tagasi Flashcards veebirakendusse. Kogu protsess oli kiire ja mugav. Pärast sisselogimist kuvatakse lehel üleval paremal menüüs Google'i konto pilt. Sellel klikkides avaneb menüü väljalogimise nupuga.

Kuna testija peab oluliseks isikuandmete kaitset ja privaatsust, siis tema oleks soovinud, et autentimiseks oleks võimalik valida ka klassikaline viis – sisselogimine meiliaadressi kaudu. Nii on võimalik logida sisse ilma, et kasutaja andmed satuksid kolmanda osapoole kasutusse.

Testija kiitis keskkonna kasutajakogemust ja disaini. Sisselogimise nupp on ilmekalt välja toodud kohe pealehel, kasutaja ei pea seda menüüst otsima minema. Kirjatüüp on piisavalt suur ja teksti on kerge lugeda. Samuti on hästi ja selgelt pealehel välja toodud selgitav informatsioon veebilehe eesmärgi kohta.

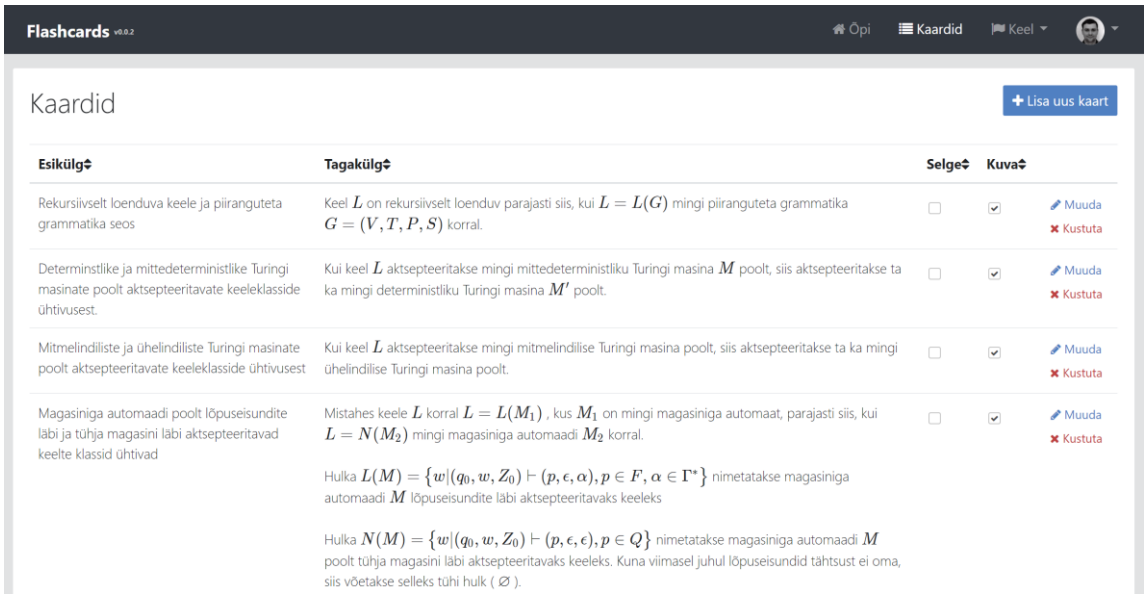
Lehe paremal üleval servas on menüüriba. Menüü on minimalistlik, lihtne ja arusaadav. Kasutajale kuvatakse kolm nuppu – “Õpi”, “Kaardid”, “Keel” ja konto sätted. See menüü, kus parajasti viibitakse, on rasvases kirjas ja teise värviga.

Menüüs pakutud keelte valikus on eesti ja inglise keel. Vaatamata võimalusele keelt valida on tõlked mitmes kohas puudu, ja keelevahetus ei muutnud näiteks pealehel olevat selgitavat teksti.

Testija nr 2 arvas, et peale ati.ee keskkonda sisenemist ja sellega tutvumist on tegu niinimetatud Flashcardsi õpikeskkonnaga. Ati.ee pealehel on lühitutvustus keskkonna kasulikkusest. Testija arvas, et maandumislehel peab olema ka märge sellest, mis on rakenduse eesmärk ning mis probleemis tahetakse sellega lahendada. Lisaks võib olla on hea välja tuua kõik tegevused, mida veebilehel saab teha. Testija märkas, et Google kaudu autentimine toimis veatult.

Testija nr 3 jaoks tekitas segadusi, miks Google’iga registreerimise ja autentimisel toimus sama protsess kaks korda. Teistkordsel proovimisel sisselogimine töötas nii nagu peab. Lisaks testija märkas, et keskkonda on võimalik siseneda ka veebilehe päises oleva “Logi sisse” nupu alt.

Teine teststsenaarium sülearvutis: “Kaartide kategooria vaatega tutvumine ja mõistekaardi valemi muutmine kasutades LaTeXit” (vt Ekraanitõmmis 13).



Ekraanitõmmis 13. Kaartide vaade

Testija nr 1 arvas, et kaartide kategooria vaade on sisuliselt üks suur HTML tabel, kus on kokku neli veergu.

Tabeli ülesehitus on kompaktne ja lihtne. Kasutajal on esimesel pealevaatamisel kohe võimalik aru saada mis andmetega on tegemist ja kuidas on süsteemis õppekaardid registreeritud.

Leheküljel alla liikudes laetakse optimeerimise eesmärgil kaardid dünaamiliselt. Võrdluseks, et sarnaselt töötab näiteks ka Facebooki uudistevoog. Lehel alla sirvides laetakse sisse uus info ja leht töötab kiiremini, kuna sisu laetakse järk-järgult.

Lähemal vaatamisel määratakse veerus esikülg kaardi esipoole tekst ja veerus tagakülg kaardi tagumise poole tekst. Tagakülje veerus sisestatud andmed kuvatakse veebilehes tabelis täies ulatuses otse välja ja ei vaja täiendavat klikkimist. Lisaks sellele on tabelis ka “Selge” ja “Kuva” nupud, mille funktsionaalsus jääb hetkel selgusetuks. Iga rea lõpus on ka muutmise ja kustutamise nupp.

Muutmise nuppu vajutades avaneb hüpikaken, kus on võimalik muuta kaardi andmeid. Kaardi tagakülg kirjeldatakse tavateksti kujul. Lisaks sellele kuvatakse kasutajale kaardi eelvaade, kus on näha lõpptulemust. Testija arvates on tegemist väga kasuliku omadusega.

Nagu autor on maininud, siis matemaatilisi valemeid on võimalik sisestada TeX süntaksiga. Juhendid süntaksi kasutamiseks tuli otsida internetist, aga kuna TeX on teadustöödes laialdaselt kasutatud märgistuskeel, siis näiteid ja juhendeid on väga palju. Valemite sisestamine võtab algaja jaoks väga kaua aega ning vajab kinnistumist.

Testija leidis huvitava tähelepaneku, et TeX valemite sisestamine algab ja lõpeb alati dollari märgiga. Lisaks testija arvas, et see on veebirakenduse jaoks vajalik, sest nii saab eristada tavateksti matemaatilistest valemitest.

Testija kiitis sisestavate väljade validatsiooni. Kuna kaardi esikülg ja tagakülg on kohustuslikud andmed, siis ilma neid sisestamata kaarti salvestada pole võimalik. Kuid häirivaks asjaoluks osutus see, et hüpikaknast mööda klikkides läheb see kinni ja kõik tehtud muudatused lähevad kaduma.

Testija tõi välja asjaolu, et rakendusel puudub kasutusjuhend. Sellest tingituna parema kasutajakogemuse eesmärgil võiks kategooria vaate avanedes kasutajale kuvada selgitavat informatsiooni. Näiteks võiks avaneda hüpikaken, kus on kirjeldatud mis on antud lehe eesmärk ja kuidas käib kaartide lisamine ja muutmine.

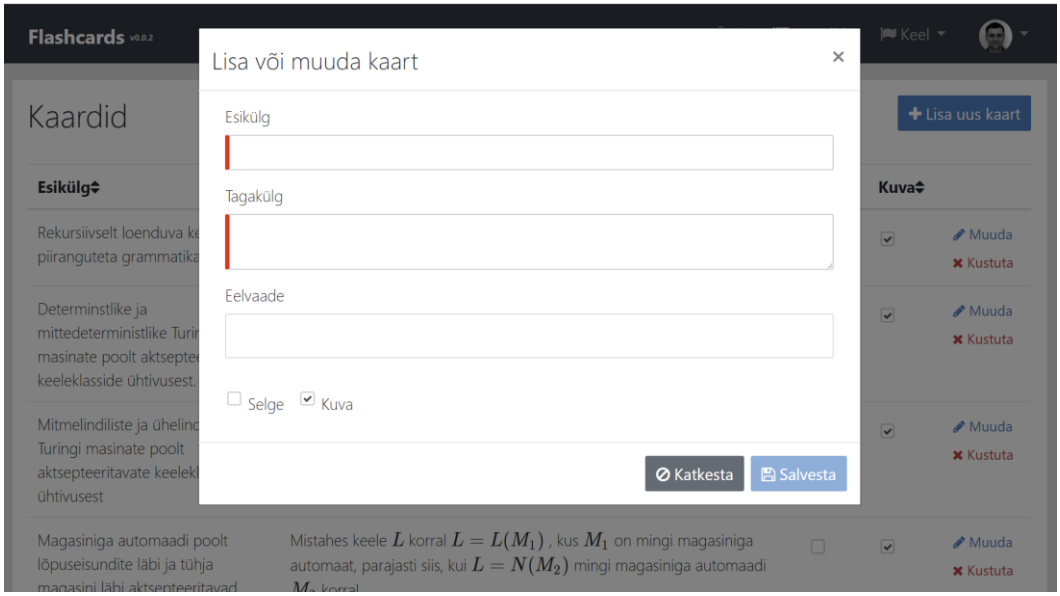
Lisaks sellele oleks kasutaja jaoks väga kasulik kui rakenduses oleks TeX valemite kirjutamise juhend või vähemalt viide veebilehele, kus selle informatsiooni kiiresti kätte saaks.

Testija nr 2 täheldas, et kaartide vaadet sirvides on võimalik intuitiivselt aru saada, kuidas keskkond toimib. Kaartide lisamise, kustutamise ja muutmise funktsioonid on selgelt kuvatud ja toimivad ilusti.

Samas testija rõhutas, et valemite muutmine on keeruline ülesanne inimestele, kes ei ole tuttav LaTeXi keelega. Seepärast peab olema vastav kasutusjuhend.

Testija nr 3 jäi kaartide vaatega rahule, kuid soovis rohkem filtreerimise võimalusi näha ning erinevaid tähenduskaarte jagada erinevatesse kategooriatesse. Testija oli varem LaTeXiga kokku puutunud ning kaartide muutmisel probleeme ei täheldanud.

Kolmas teststsenaarium sülearvutis: “Uue sõnaseletuskaardi lisamine, ostimine ja kustutamine“ (vt Ekraanitõmmis 14).



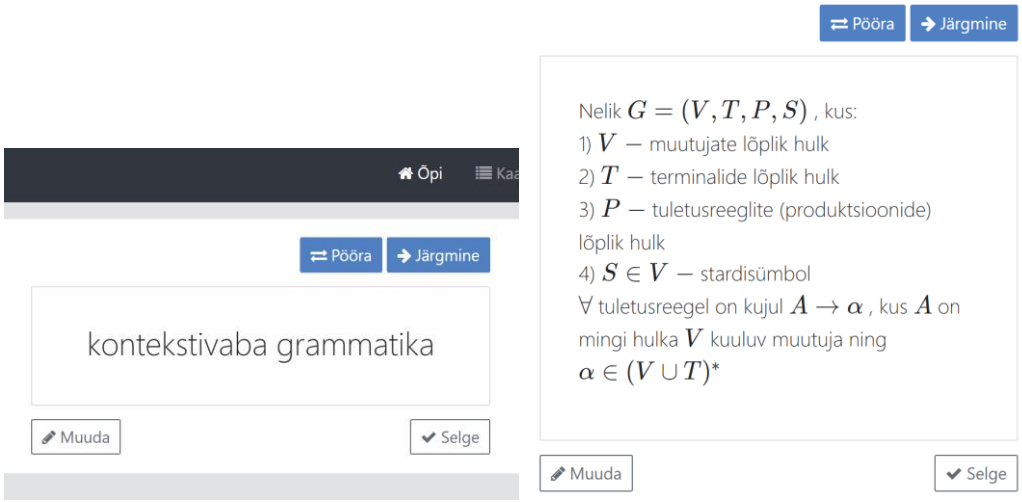
Ekraanitõmmis 14. Kaartide lisamise vaade

Testija nr 1 leidis, et kohe lehekülje päise paremas servas kaardi lisamise nupu. Uue kaardi lisamisel kuvati kasutajale infomullike, et kaart on loodud. Uus kaart lisati tabelis esimeseks ja on kergesti leitav. Testija kiitis, et kustutamisel küsiti kasutajalt täiendavalt üle, et kas oled ikka otsuses kindel. See välistab võimaluse kaarte kogemata kustutada.

Testija nr 2 arvas, et kaartide lisamise ja kustutamise funktsioonid on kergesti leitavad ja toimivad veatult. Lisatud kaart kuvatakse nimekirjas esimesena. Kaarte saab ka ümber sorteerida tähestikulises järjekorras esikülje alusel. Testija ei leidnud kaartide sisupõhist otsingu funktsionaalsust ning leidis, et see oleks kasulik lisa keskkonnale.

Testija nr 3 meeldis kaartide sorteerimise funktsionaalsus. Ka kolmandale testijale meeldis, et vastloodud kaart kuvatakse esimesena. Testija soovitas kustutamisel kuvada kaardi esikülge mitte mingit suvalist numbrit.

Neljas teststsenaarium sülearvutis: “Õppimise funktsionaalsuse katsetamine“ (vt Ekraanitõmmis 15).



Ekraanitõmmis 15. Õppimise vaade. Kaardi esi- ja tagakülg

Testija nr 1 mõistis õppimise vaadet, kus kuvatakse kasutajale nimekirjast järgmine kaart ja navigeerimise nupud. Testija märkas, et kasutajal on võimalik kaarti pöörata ja näha sisu, muuta kaardi andmeid, võtta nimekirjast järgmine kaart ja määrata käesolev kaart õpituks. Kui kaart on märgitud õpituks, siis seda enam kasutajale ei pakuta.

Kasutaja saab kõik toimingud õppimise vaates teha hiirega vajutades navigeerimisnuppudele. Lisaks sellele selgus eksperimenteerimise käigus, et võimalik on

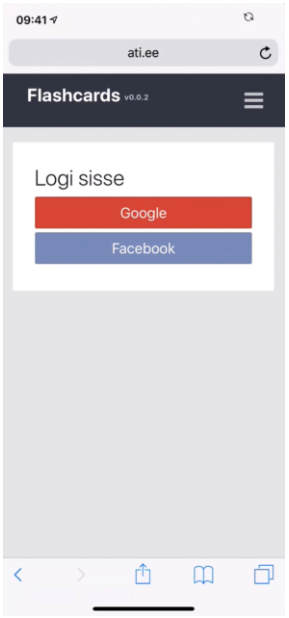
kasutada ka klaviatuuri kiirklahve. Noolte klahvidega saab kaarti pöörata ja võtta nimekirjast järgmine kaart. Järgmise kaardi saab ette ka tühiku klahviga.

Testija arvab, et õppimise protsess ise oli lihtne ja intuitiivne. Kuid puudusena täheldas, et rakendusel puudub kasutusjuhend. Sellest tingituna parema kasutajakogemuse eesmärgil peab kategooria vaate avanedes kasutajale kuvada selgitavat infromatsiooni. Näiteks võiks avaneda hüpikaken, kus on kirjeldatud, mis on antud lehe eesmärk ja kuidas käib õppimine. Lisaks sellele võiksid olla üles loetletud võimalikud klaviatuuri kiirklahvide kombinatsioonid, mida saab lehel kasutada.

Ka **testija nr 2** arvas, et õppimise protsess on lihtne ja intuitiivne. Kuvatakse kaardi esikülg, ning kaarti saab oma teadmiste kontrolliks pöörata, ning vajadusel sisu redigeerida "Muuda" nupust. Kaartide sirvimise ja pööramise nupud on selgelt ja mugavalt kuvatud. Kaarte saab sirvida ainult ühes suunas. Võiks olla ka võimalus eelmisele kaardile tagasi pöörduda. Testija tõstis esile keskkonna funktsionaalsust: kõik vajalik on olemas ja pole liigseid lisavõimalusi.

Testija nr 3 pakkus, et tagakülg võib olla eristuv esiküljest, sest "Pööra" nuppu vajutades ei ole võimalik mõista, kas kuvatakse esi- või tagakülge. Lisaks eelnevale keskkonnas peab olema tagasi liikumise võimalus.

Esimene teststsenaarium nutitelefonis: „Google'i kaudu autentimine" (vt Ekraanitõmmis 16).



Ekraanitõmmis 16. Maandumisleht nutitelefonis

Testija nr 1 hindas kõrgelt rakenduse optimeeritust nutitelefoni peal kasutamiseks. Testija arvas, et on kasutatud kohalduva veebidisaini parimaid praktikaid. Nutitelefoniga rakenduse lehele minnes tunduvad tekstid ja lehe elemendid suuremad. Rakendus on skaleeritud kasutaja jaoks vastavalt seadme ekraanile. Menüü on kompaktne ja kuvatud ühe vertikaalse tulbana. Rakendusse saab siseneda Google'i kontoga. Nutiseadmega rakendusse sisenemise kogemus on sarnane sülearvuti kogemusega.

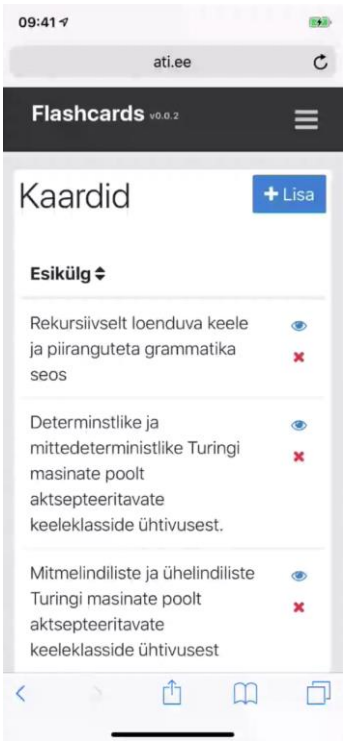
Testija pidas üheks puuduseks seda, et väiksemal ekraanil puudub esilehel selgitav tekst ning menüüs puuduvad menüüvalikute ikoonid

Testija nr 2 ei täheldanud probleeme sisselogimisega. Testija oli juba keskkonnaga veidi tuttav. Seega telefonist ati.ee kasutamine on samuti intuitiivne, ning kõik funktsioonid lihtsasti leitavad. Testija leidis, et selle keskkonna kasutaja eelistaks lisada ja muuta kaarte arvutis, ning kasutada telefonis eelkõige õppimisfunktsiooni.

Testija arvas, et esmakordne kasutamine ja keskkonnaga tutvumine telefonist võib olla veidi keerukam kui arvutist, sest kõik nupud ja ka kaartide sisu ei ole ekraanil koheselt kuvatud. Kaartide vaade telefonist näitab ainult esikülje tekste, ning seega vajab rohkem eksperimenteerimist, et sisuliselt aru saada, kuidas kaarte lisada, muuta ja sorteerida.

Testija nr 3. Nutitelefoni kaudu Google'i konto abil sisenemine läks kiiresti ja mugavalt. Testija valis konto ja oligi sisselogitud. Samamoodi nagu eespool mainitud – autentimine oli kerge ja ei tekitanud ühtegi probleemi.

Teine teststsenaarium nutitelefonis: „Kaartide kategooria vaatega tutvumine ja mõistekaardi valemi muutmine kasutades LaTeXit" (vt Ekraanitõmmis 17).



Ekraanitõmmis 17. Kaartide vaade nutitelefonis

Testija nr 1 arvas, et kaartide kategooria vaade on optimeeritud nutitelefoni peal kuvamiseks. Sülearvutis kuvatud lai HTML tabel on telefonis kuvatud kompaktselt ühe vertikaalse tulbana. Ruumi kokkuhoidmise eesmärgil on välja kuvatud ainult kaardi esikülje andmed ja iga rea jaoks muutmise ja kustutamise nupud. Nupud ise samuti optimeerimise eesmärgil muutunud. Redigeerimise nupud kuvatakse kasutajale ikoonidena ja "Lisa uus kaart" asemel on nupu tekst lühendatud: "Lisa".

Rea peale klikkides avaneb kaardi andmete eelvaade. Vaatel on olemas muutmise nupp, millel vajutades avaneb hüpikaken kaardi andmete muutmiseks koos TeX eelvaate aknaga. Erinevusi sülearvuti vaatega siin pole.

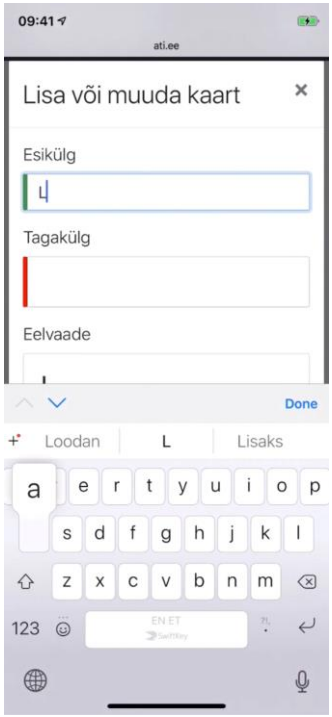
Säilinud on ka funktsionaalsus, et leheküljel alla liikudes laetakse optimeerimise eesmärgil kaardid dünaamiliselt. Kaardi muutmisel või lisamisel kuvatakse kasutajale ka nutitelefonis infomullike.

Puudusena testija tõi välja mõned visuaalsed ebakõlad. Kuna tabelis on kuvatud ainult üks veerg, siis pole vajadust kasutajale välja kuvada ka veeru nime “Esikülg”.

Testija nr 2 leidis, et matemaatiliste valemite sisestamine TeX süntaksiga on väikesel nutitelefoni ekraanil veidi kohmakas ja ebamugav. Kui TeX süntaks pole just kõrvale välja prinditud, siis tuleb ikkagi valemite sisestamiseks kõrvale võtta teine seade või sülearvuti. Samas on see ikkagi oluline omadus, et nutitelefonis on võimalik kaardil teha kiire teksti parandus või täiendus.

Testija nr 3 mõistis muutmise vajalikkust, kuid tõi välja selle, et telefonis mingit kaarti muuta ei ole kohe kindlasti nii mugav kui seda oleks teha arvutis. Testija, et telonis andmeid muutmine on tülikam, sest vead võivad kergesti tekkida.

Kolmas teststsenaarium nutitelefonis: „Uue sõnaseletuskaardi lisamine, ostimine ja kustutamine“ (vt Ekraanitõmmis 18).



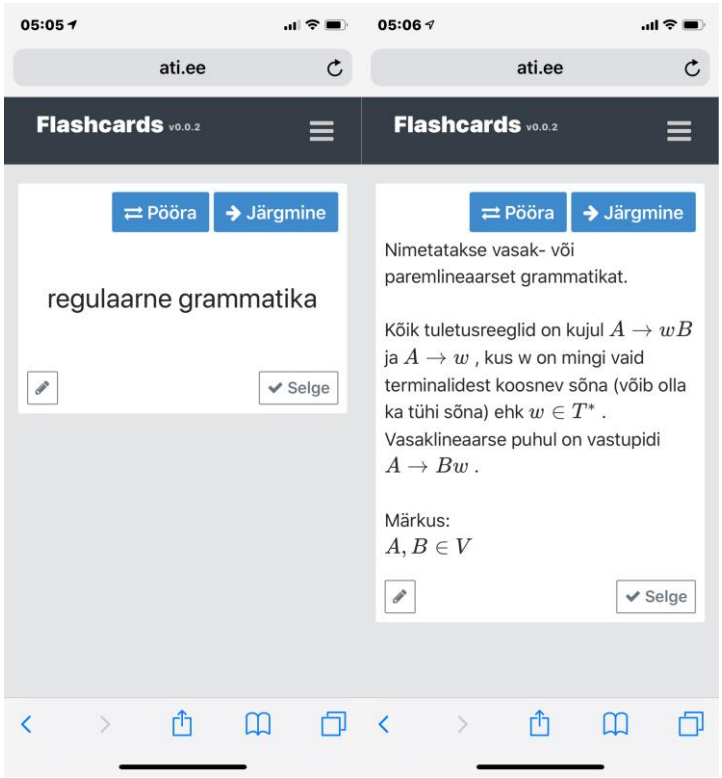
Ekraanitõmmis 18. Tähenduskaardi loomine nutitelefonis

Testija nr 1. Nutitelefoniga kaartide lisamise kogemus on sarnane sülearvuti kogemusega. Lehe tekstid ja komponendid on selgelt välja toodud ja nutitelefoni ekraanil loetavad.

Testija nr 2. Uue kaardi lisamine ja kustutamine oli sujuv. Uus kaart tekkis kõige üles ja kustutamine oli samuti lihtne.

Testija nr 3 oli juba eespool mainitud asjaolu, et telefonis kaartide muutmist või lisamist teha ei eelistaks. Seega testija jaoks oli ka uue kaardi lisamine kohutavalt ebamugav. Testija oli täiesti kindel, et tema eelistaks andmete sisestamist teha arvutis.

Neljas teststsenaarium nutitelefonis: „Õppimise funktsionaalsuse katsetamine“ (vt Ekraanitõmmis 19).



Ekraanitõmmis 19. Õppimise vaade nutitelefonis. Esikülg on vasakul ja tagakülg on paremal

Testija nr 1 arvas, et suurima eelise annab nutitelefoni kasutamine just õppimise vaates. Kasutaja saab suunata kogu oma tähelepanu kaardile.

Õppimise vaates kuvatakse kasutajale nimekirjast järgmine kaart ja navigeerimisnupud. Kaarti on võimalik pöörata esi- või tagaküljel klikkides, kuid ka vajutades nuppu “Pööra”.

Nupule “Selge” vajutades märgib kasutaja kaardi õpituks ja seda uuesti kasutajale ei kuvata.

Puudustena tõi testija välja, et navigeerimisnupud tunduvad olevat nutitelefoni ekraanil üleliigsed. Nutitelefoni parimaks omaduseks on võimalus navigeerimiseks kasutada näpuga lohistamist ja viipamist. Nuppude asemel võiks olla võimalus kaarte vahetada viipamisega paremalt vasakule ja kaardi tagakülje lugemiseks selle peale vajutada.

Hetkel saab kasutaja kaarti pöörata kahte moodi – pööramisnupule või kaardi peale vajutades. See tundub üleliigne ja piisaks täiesti ka ühest viisist.

Kaart võiks olla välja kuvatud veelgi ilmekamalt. Näiteks oleks mõistlik kaart kuvada täisekraanil ning kaardi peal klikkides ilmub kaardi tagakülg. Kaartidelt õppimise juures on kõige olulisem sisu, kõik muu peaks jääma kasutaja jaoks varjatuks ja tahaplaanile.

Testija nr 2 leidis, et õpikeskkond nutitelefonis on funktsionaalne. Testijale meeldis, et kaartide vaade telefonis on minimalistlik. Tema arvates peab õppimise vaates olema võimalus kuvada ainult kaardi sisu täisekraanil, ning kaart võiks olla pööratav näiteks kaardile vajutamisega.

Testija nr 3 arvas, et telefonis õppimine on samuti lihtne ja mugav. Telefoni vaade on hea ja loob õppimise jaoks head võimalused. Probleeme ei esinenud ja testija ei osanud täiendusi välja pakkuda.