Tallinna Reaalkool

Pranglimisrakenduse loomine välearenduse põhimõtetel

Uurimistöö

Hermann Käbi

140.a reaal-programmeerimise õppesuund

Juhendaja: õp Hanna Britt Soots

Tallinn 2024

Sisukord

Sissejuhatus

Peastarvutamine on matemaatikatunnis õpetatavatest oskustest üks eluliselt kasulikem. See on oskus, mida vajab ja kasutab pea igaüks. Põhikooli matemaatika õppekavasse on peastarvutamine oskuse omandamine ka sisse kirjutatud. Paraku muutub matemaatika III kooliastmest alates liiga kompleksseks, et seda oskust harjutada. Õpilastel, kes soovivad oma peastarvutusoskust hoida ja parandada, tuleb seega leida selleks mingi muu võimalus kui matemaatikatund.

Üheks enimsoovitatud võimaluseks on pakutud pranglimist. Pranglimine on võisteldes aja peale peastarvutamine (ÕS 2018 *s.v.* pranglima). Pranglimine on olnud populaarne koolides üle Eesti, seda eeskätt tänu Miksikese keskkonnale. Miksike ongi olnud pranglimise algatajaks ja eestvedajaks. Kahjuks sulges Miksike 2022. aastal oma veebiserverid, peale mida on pranglimise populaarsus nii tunnisiseselt kui -väliselt langenud. Alternatiivne Nutispordi keskkond on aga Tallinna Reaalkooli matemaatikaõpetajate sõnul puudulik. Tekkis soov luua Tallinna Reaalkoolile oma pranglimisrakendus, mille saaks teha täpselt soovidele vastava.

Käesoleva uurimistöö eesmärgiks on selline veebirakendus luua ja arendusprotsessi käigus vastata uurimisküsimusele: kuidas kasutada välearenduse põhimõtteid reaalses elus pranglimisrakenduse arenduse näitel? Uurimistöö koosneb välearenduse tsükli kirjeldusest, pranglimisrakenduse arendamiseks kasutatud tehnoloogiate kirjeldusest ja arendusprotsessi kirjeldusest. Peamiseks allikaks välearenduse kohta on James Shore'i ja Shane Warden'i teos „The Art of Agile Development“ („Välearenduse kunst“). Erinevate tehnoloogiate kirjeldamisel on aluseks võetud vastavad dokumentatsioonid.

1. Välearendus

1.1. Välearenduse kirjeldus

Välearendus (*agile development*) on tarkvaraarendus, kus kasutatakse agiilseid metoodikaid (Wikipedia 2023l; Andmekaitse ja infoturbe leksikon 2023 *s.v.* agile development). Eestikeelne termin „välearendus“ on välja pakutud Andmekaitse ja infoturbe leksikon (AKIT), mis ühtlasi nimetab „agiilset arendustsüklit“ naiivseks poolkeelseks laenuks (Andmekaitse ja infoturbe leksikon 2023 *s.v.* agile development). Sel põhjusel kasutatakse ka selles uurimistöös võimalusel sõna „välearendus“. Kuna välearendus sarnaneb pigem filosoofilise mõtte kui algoritmiga, on sellele täpset definitsiooni võimatu anda. See tähendab ka, et ühte kindlat välearenduse mudelit pole olemas. Selleks, et praktiseerida välearendust, tuleb kasutada agiilsed meetodid vastavalt konkreetsele projektile. (Shore, Warden 2008)

Välearenduse tavad on sõnastatud *Manifesto for Agile Software Development*'s („Agiilse tarkvaraarenduse manifest“). Manifestis kirjeldatakse välearenduse nelja põhimõtet:

1. Hindame enam inimesi ja nende suhtlemist kui protsesse ja arendusvahendeid.

2. Hindame enam töötavat tarkvara kui täiuslikku dokumentatsiooni.

3. Hindame enam koostööd kliendiga, kui läbirääkimisi lepingute üle.

4. Hindame enam muudatustega hakkamasaamist kui algse plaani järgimist.

See ei tähenda aga, et välearenduses parempoolseid tegureid ei hinnata, vaid vasakpoolseid hinnatakse rohkem. (Beck *et al.* 2023; Wikipedia 2023l)

Oluline on mõista, et välearendus ei ole iseenesest konkreetne arendusmeetod, vaid pigem katustermin. Selleks, et välearenduse kasutamine oleks lihtsam, on kasutusele võetud erinevad välearenduse raamistikud (*framework*). Kõige populaarsem välearenduse raamistik on Scrum, mida kasutab 87% arendajatest (State of Agile 2022). Scrum hõlmab endas projekti jagamist väikesteks osadeks ehk sprintideks, mille pikkus jääb tavaliselt nädala ja

kuu vahele. Korraga keskendub arendustiim vaid ühele sprindile. (PremierAgile 2022) Väga sarnane raamistik on Extreme Programming (XP), mis jagab projekti samuti sprintideks, kusjuures pärast igat sprinti üritatakse meetodit vastavalt hetkeseisule võimalikult palju optimeerida. Samas, erinevalt Scrumist rõhutab XP pideva kliendiga suhtlemise vajadust. Erinevalt Scrumist, mida kasutatakse projektijuhtimises üldiselt, on XP, nagu nimigi viitab, kasutusel vaid tarkvara arendamisel. (Raeburn 2022) Kolmas populaarne raamistik on Kanban (56%), mille võlu seisneb lihtsuses. Selle meetodiga luuakse arendusseisust visuaalne tahvel, kus on näha nii tulevikus alustatavad, praegu töösolevad kui ka juba tehtud arendused. Lisaks mainitutele on välearenduse põhimõtetele ülesehitatud raamistikke veelgi. (PremierAgile 2022; State of Agile 2022)

Tänapäeval on välearendus tarkvaraarenduse oluline osa. Välearenduse meetodeid kasutavad pea kõik suured tehnoloogiafirmad, muuhulgas Microsoft, IBM ja Apple. (Flynn 2022) 2023. aasta seisuga kasutab välearenduse meetodeid 86% professionaalsetest tarkvaraarendajatest ning 71% tarkvara arendavatest firmadest (Outworks Solutions Private Ltd. 2023). Seega tuleb arendajale välearenduse vähemalt põhiliste meetodite tundmine ja kogemus kasuks nii teiste arendajatega sujuvama koostöö loomiseks, aga ka aina konkurentsitihedamal tööturul. Kuna välearendust kasutab aina rohkem tarkvarafirmasid, on see oluline mitte ainult arendajatele, vaid ka näiteks projektijuhtidele. Samuti kasutatakse välearenduse põhimõtteid ka projektides, kus tarkvara ei arendata, seega tuleb see oskus kasuks ka väljaspool IT-sektorit. (PremierAgile 2023)

Vaatamata väga suurte firmade mõjule ja laialdasele populaarsusele arendajate seas, on oluline märkida, et empiirilised uurimused välearenduse efektiivsuse kohta on jõudnud erinevate järeldusteni. Dybå ja Dingsøyr toovad 36 välearenduse uuringut analüüsides, et nende uurimuste tulemuste väärtus on kaheldav. Valdavad veakohad on nii uurimuslikud, näiteks potentsiaalsete kalduvuste (*bias*) eiramine, aga vigu on tehtud ka andmete kogumise faasis. Nimelt leiti, et vaid vähesed uurimused olid tehtud firmades, mis olid välearendust kasutanud üle aasta. Valdav osa (76%) uurimustest uuris vaid XP kasutamist, seega ei saa nende põhjal eeldada samasugust mõju, kui kasutada mõnda muud välearenduse raamistikku, näiteks Scrumi. Järelikult ei ole nende uurimuste tulemused olukorras, kus valdav osa arendajaid kasutab teisi raamistikke, enam relevantsed. Dybå ja Dingsøyr rõhutavad just Scrumi efektiivsuse ja eeliste põhjalikuma uurimise olulisust. Seega tuleb

enne välearenduse kasutuselevõtmist kaaluda selle võimalikke eelised konkreetses projektis, mitte toetuda vaid suurfirmade eeskujule. (Dybå, Dingsøyr 2008)

1.2. Välearenduse ajalugu

Tsüklilised arendusmeetodid olid kasutusel juba 1950. aastate keskpaigast (Larman, Basili 2003). Samas oli selliste meetodite kasutamine harv. Pigem eelistati koskmudelil põhinevaid arendusmeetodeid. (Sacolick 2022) Koskmudel on nime saanud kosest läbi voolava vee tõttu - vesi liigub ühesuunaliselt edasi ja tagasi enam ei pöördu. Koskmudeli järgi peab enne uue etapi alustamist kõik eelnevad täielikult valmis olema. Näiteks ei hakata rakendust programmeerima enne, kui kõik nõuded ja kogu disainiprotsess läbitud on. (Wikipedia 2022) Koskmudeli eeskujuks oli Fordi konveieriliin, mis oli 20. sajandi alguses teinud revolutsiooni autotööstuses. Koskmudeli eesmärk oli kindlustada projekti vastavus kliendi algsetele soovidele. Koskmudeli üheks suuremaks puuduseks oli arenduse ajaline pikkus ja jäikus. (Sacolick 2022)

Veebirakenduste esiletõusuga hakkasid koskmudeli puudused aina enam silma paistma. Kui varasemalt oli rakenduste loomiseks erinevaid tehnoloogiaid ja uuendusi võrdlemisi vähe, siis veebi pideva ja kiire arenemise tõttu pidid ka arendajad aina paindlikumaks ja kiiremaks muutuma. Efektiivsuse suurendamiseks ei olnud kogu dokumentatsiooni ja disaini valmis tegemine enne arenduse algust enam võimalik. Aina suurenevate arendustiimide tõttu tekkis vajadus koostööd suurendavate meetodite järele. (*ibid.*)

2001. aastal kogunes Utah's Snowbirdi kuurordis grupp kogenud arendajaid, kes olid mõistnud, et arendavad tarkvara traditsioonilisest koskmudelist erinevalt. Need 17 arendajat, teiste seas XP looja Kent Beck ja Scrumi looja Jeff Sutherland, panid oma kogemustele tuginedes kirja välearenduse põhimõtted (*Manifesto for Agile Software Development*). Välearendus, mis rõhutas koostöö ja pideva muutumise olulisust ja kritiseeris jäika juhtimisstiili, oli sündinud. (Sacolick 2022; Wikipedia 2023a)

Sellest ajast on välearendus arendusmaailmas aina levinud. 2005. aastal avaldasid Cockburn ja Highsmith dokumendi, milles avaldati välearendusliku projektijuhtimise põhimõtted (Highsmith, Cockburn 2005). 2009. aastal avaldatud *Software Craftmanship Manifesto* täiendas Beck *et al.* kirjutatud manifesti, lisades juhendi välearenduse kasutamiseks suurtes

ja professionaalsetes arendustiimides (Martin 2009). 2011. aastal lõi Agile Alliance avatud lähtekoodiga sõnaraamatu *Agile Glossary* välearenduses kasutatavate terminite ja tavade selgitamiseks (Agile Alliance 2023; Wikipedia 2023a).

Välearenduse põhimõtteid hakkasid suurfirmad üsna kiiresti kasutama. Juba 2007. aastal oli Google mitmes projektis välearenduse põhimõtted kasutusele võtnud (Striebeck 2007). Apple on üle viieteistkümne aasta kasutanud Scrumi (Denning 2012). Forbes'i 2015. aastal avaldatud artikli kohaselt on ka Microsoft välearenduse põhimõtted kasutusele võtnud (Denning 2015). Iga-aastase välearenduse seisu analüüsiva raporti *State of Agile* 2022. aasta väljalaske andmetel kasutab välearendust mingil määral 80% vastanud firmadest (State of Agile 2022).

1.3. Välearenduse eelised

1.4. Välearenduse põhimõtete kasutamine pranglimisrakenduse loomisel

2. Pranglimisrakenduse loomisel kasutatud tehnoloogiad

2.1. Veebiarenduse liigitus

Üks enimkasutatavaid viise veebiarenduse liigitamiseks on liigitus *front-end*'iks ja *back-end*'iks (GeeksForGeeks 2023). Selline jaotus on kasulik, sest paljud veebiarendajad keskenduvad oma oskustest ühele nendest.

*Front-end*i all mõeldakse seda osa veebilehest, mida kasutaja näeb ja millega ta interakteerub. Selleks on veebilehe graafiline kasutajaliides (GUI). *Front-end*i alla kuuluvad näiteks küljendus, fondid, värvid, nupud, pildid, videod, navigatsioonimenüüd. *Front-end* koosneb põhiliselt kolmest komponendist: HTML, CSS ja JavaScript. Nendele lisanduvad arendajakogemuse (DX) ja koodi haldamise ja loetavuse parandamiseks erinevad raamistikud.(*ibid.*)

Back-end on veebilehe osa, mida kasutaja otseselt ei näe. See aga ei tähenda, et tegu oleks kuidagi vähem olulisema osaga veebiarendusest. *Back-end*i alla kuuluvad näiteks *routing* (vastavalt URL-ile sobiva sisu leidmine), andmete kogumine ja kasutamine, serverite haldamine jne. Erinevalt *front-end*ist pole *back-end*is välja kujunenud standardkeeli, see valitakse vastavalt vajadustele. Kõige populaarsem *back-end* keel on PHP, aga kasutatakse ka C++ ja JavaScripti. *Back-end*is kasutatakse väga tihti erinevaid raamistikke, et lihtsustada integratsioone andmebaasidega, autentimist ja *front-end* raamistike kasutust. (*ibid.*)

2.2. *Front-end* tehnoloogiad

2.2.1. *Hypertext Markup Language* (HTML)

Hypertext Markup Language (HTML) on *front-end* veebiarenduse kõige põhilisem tehnoloogia. HTML annab veebilehel olevale sisule tähenduse ja struktuuri. (MDN Web Docs 2023c) Oluline on siinjuures mõista, et HTML ei ole programmeerimiskeel, vaid märkekeel. See tähendab, et kasutades HTML süntaksit saab muuta sisu väljanägemist, aga mitte anda arvutile käsklusi mingite ülesannete täitmiseks. Näiteks saab HTML'i kasutades

muuta tekst rasvaseks (bold), aga mitte teksti ennast.

HTML dokument koosneb erinevatest elementidest, mis enda sisse jääva sisu teatud moodi vormistab. Näiteks vormistab element *p* enda sisu tavalise tekstina, *h1* aga suure pealkirjana. HTML süntaksi lihtsus peitub asjaolus, et elemente saab ka üksteise sisse panna. Nii saab näiteks ühe osa lõigust kaldkirja panna. Elemendile saab lisada erinevaid atribuute (*attribute*), näiteks hüperlingi (*a*) loomisel lisatakse attribuut *href*, millega määratakse URL, kuhu link suunab. Igale elemendile saab lisada klassi (*class*), mis saab olla mitmel elemendil, ja ID, mis peab igal elemendil olema unikaalne. (MDN Web Docs 2023c)

Kuna HTML mängis suurt rolli kogu veebi loomisel, on sellel pikk ajalugu. Esimese versiooni lõi CERNis töötav füüsik Tim Berners-Lee juba 1991. aastal, kuid see jäi avaldamata. See versioon, nimega HTML 1.0, avaldati 1993. aastal ja pani aluse veebile endale. Sellest ajast peale on HTML pikkamisi arenenud. Tänapäeval kasutatakse 2014. aastast pärit HTML5 versiooni. See versioon lisas muuhulgas platvormipõhise toe video- ja audioformaatidele ja palju semantilisi elemente. (University of Boston 2020b) Kui HTML 1.0 oli vaid 18 elementi, siis HTML5 sisaldab endas lausa 137 elementi. (MDN Web Docs 2023d)

Kuna HTMLi võib vaadelda veebilehe selgrooga, on lehtede tegemine ilma selleta võimatu. Uurimistöö käigus arendatud pranglimisrakenduse kasutajaliides on Reacti-põhine, mis kasutab HTML-sarnast keelt JSX, mis ühendab endas JavaScripti ja HTMLi ja millest räägitakse täpsemalt osas 2.2.4.

2.2.2. *Cascading Style Sheets* (CSS)

Cascading Style Sheets (CSS) on laadilehekeel, mis määrab, kuidas HTML koodi ekraanil kuvatakse. Erinevalt HTMList ei lisa CSS veebilehele sisu, vaid kujundab seda. Väikeste eranditega võime öelda, et HTML vastutab veebilehe tekstilise sisu ja CSS selle esitamise eest. (MDN Web Docs 2023b)

CSS kasutab reeglitepõhist süntaksit. Reegel (*rule*) algab selektori (*selector*) määramisega. Selektor on struktuur, millega määratakse valitavate HTML-elementide hulk. Näiteks saab selektoriga valida kõik pealkirjad (*h1*) dokumendis, aga samuti klasse ja IDsid. Kasutades sellised lihtsaid selektoreid saab kombineerida ka tunduvalt komplekssemaid. Näiteks saab valida iga teise lõigu, millel on klass „tekst“ ja mis asuvad *div* elemendi

sees. Selektorile järgneb loend deklaratsioone (*declaration*), mis muudavad kindla omaduse (*property*) väärtust (*value*). Näiteks omadusega *font-family* saab muuta elemendi fonti ja *background-color* muudab taustavärvi. (W3Schools 2023b)

Lisaks elementide laadi muutmisele pakub CSS ka võimekat kastimudelit (*box model*) (W3Schools 2023a). CSS kohtleb kõiki elemente ristkülikutena, mis võimaldab kuvada neid ekraanile võimalikult väikese jõudlusega. Tänu kastimudelile saab elementide paigutust väga lihtsalt ja paindlikult muuta. (Wikipedia 2023c) CSSi teine suur eelis on meediapäringud (*media query*), mis lisab võimaluse muuta elementide laadi vastavalt brauseriakna suurusele (*viewport*). Seega saab CSSiga teha lehti, mis on ka mobiilisõbralikud (*responsive*). (MDN Web Docs 2023a)

Kuna esialgu oli veeb mõeldud teadlastele sisu jagamiseks, ei olnud veebilehe laad oluline. Veebi populariseerumine tõi aga kaasa vajaduse visuaalselt ilusate lehtede loomiseks. Juba 1994. aastal pakkus Håkon Wium Lie välja idee CSSist ja 1996. aastal avaldatigi CSS 1, millega sai muuta väga põhilisi omadusi, nagu fonti ja värve. Kaks aastat hiljem avaldati CSS 2 ja alustati tööd CSS 3-ga. Selle versiooniga tehti põhimõtteline uuendus: kui varem oli avaldatud versioon tervikuna, siis CSS 3 avaldati moodulitena. (University of Boston 2020a) Esimene suur moodul avaldati 2012. aastal, peale mida on keelt senimaani pidevalt uuendatud. (Wikipedia 2023c)

Kuna CSS on niivõrd palju muutunud ja arenenud, on see muutunud üsna keeruliseks ja koodi poolest pikaks. Probleemi lahendamiseks on loodud erinevaid CSS raamistikke, mille eesmärgiks on muuta põhiliste kasutajaliidese komponentide loomine võimalikult lihtsaks. Sellel lähenemisel on mitmeid eeliseid. Esiteks muudab see veebilehe disaini lihtsaks ja kiireks. Samuti on komponendid disaininud tiim professionaalseid disainereid, mistõttu sobivad komponendid (näiteks nupud, lingid jne) omavahel visuaalselt hästi kokku. Raamistike kasutamisel on ka negatiivseid pooli. Kuna komponendid on juba valmis tehtud, tähendab see, et iga leht, mis ühe raamistikuga tehtud on, näeb võrdlemisi sarnane välja. Kuna raamistikud on disainitud võimalikult erinevatele lehtedele sobivaks, võivad nendega tehtud lehed tavalised ja isegi igavad välja näha. Märksa tehnilisem probleem, eriti vanemate raamistikega, on liigne kood. Kuna raamistikud koosnevad väga paljudest komponentidest erinevateks olukordadeks, juhtub praktikas tihti, et veebileht neid kõiki ei kasutada. Seega jääb veebilehele palju koodi, mida see tegelikult ei kasuta,

mis võib lehe aeglasemaks muuta. (Ayebola 2023; Bose 2023)

Pranglimisrakenduse arendamisel kasutasin CSSi ilma ühegi raamistikuta. Tegin sellise otsuse, võttes arvesse eelmises lõigus toodud puudusi. Minu jaoks on oluline rakenduse visuaalne erilisus ja ise komponentide loomine on mulle oluline veebiarenduse osa. Seega lõin pranglimisrakenduse jaoks sisuliselt eraldi enda raamistiku, sest iga lehe puhul on ühtlane disainikeel oluline. Kuna CSS on mulle varasemast kogemusest tuttav, ei olnud lehtede kujundamine CSSiga minu jaoks väga keeruline. Samas üritasin arenduse käigus kirjutada võimalikult modulaarset ja muudetavat CSSi. Üks CSSi funktsioon, mida õppisin selle projekti käigus, oli muutujate kasutamine CSSis. Kui paljudel komponentidel on näiteks sama taustavärv, ei ole mõtet igale komponendile eraldi ühte värvikoodi kopeerima, sest kui seda on hiljem muuta vaja, tuleb selleks palju tööd teha. Siis on mõistlikum salvestada värv muutujasse ja kasutada komponentides muutujat. Muutujatesse saab loomulikult salvestada igat tüüpi väärtuseid. Muutujate kasutamine CSSis seondub ka välearendusega, sest pideva tagasiside saamine muudab tõenäoliseks asjaolu, et osa disainist tuleb ümber teha, mis on muutujaid kasutades palju kiirem.

2.2.3. JavaScript

JavaScript on kolmas veebiarenduse põhitehnoloogia. Tegu on dünaamiliselt kirjutatud interpreteeritud kõrgetasemelise programmeerimiskeelega. (MDN Web Docs 2023e) Kuigi brauseris (*client-side*) saab veebileht jooksutada erinevaid programmeerimiskeeli, kasutavad veebiarenduse uuringufirma W3Techs hinnangul JavaScripti brauseris (*client-side*) 98,7% veebilehtedest (W3Techs 2023). Populaarse arenduskeskkonna Stack Overflow iga-aastase küsitluse tulemusena oli 2022. aastal JavaScript juba kümnendat aastat järjest kõige sagedamini kasutatud programmeerimis- ja märkekeel (Stack Overflow 2022). Seetõttu on veebilehtede arendamisel JavaScripti oskus sisuliselt kohustuslik.

JavaScripti kasutusaladeks pole aga ainult koodi jooksutamine brauseris. Just suure populaarsuse tõttu kasutatakse JavaScripti tihti ka *back-end* arenduseks (MDN Web Docs 2023e). Stack Overflow 2022. aasta küsitlusest nähtub, et nii professionaalsete arendajate kui kõigi vastajate hulgas on populaarseim veebiraamistik (*framework*) JavaScripti-põhine Node.js (Stack Overflow 2022).

Üheks näiteks, kus JavaScripti kasutatakse väljaspool brauserit, on Node.js. Node.js on

JavaScriptil põhinev populaarne platvormist sõltumatu *back-end* raamistik (Node.js 2023). Node.js muutis JavaScripti kasutamise populaarseks väljaspool brauserit (Wikipedia 2023g). Node.js'i populaarsust omistatakse eeskätt lihtsale ja mõistetavale süntaksile, kõrgel kohal on ka raamistiku kasutatud JavaScript, mis on arendajatele juba tuttav ja seega muudab arenduse kiiremaks (BrainHub 2023).

Kolmest põhitehnoloogiast on JavaScriptil kõige keerulisem ajalugu. Veebi esimestel aastatel said veebilehed olla ainult staatilised, sest programmeerimiskeelt brauseris kasutada ei saanud. Selle probleemi lahendamiseks ühines NetScape'i (esimene populaarne veebibrauser) meeskonnaga Brendan Eich. Eichi ülesandeks sai luua uus Java-laadse süntaksiga programmeerimiskeel veebilehtede jaoks. (Wikipedia 2023g; Wikipedia 2023b; W3Schools 2023c)

Veebibrausereid tegid juba tol ajal erinevad firmad, mis kasutasid pisut erinevaid programmeerimiskeeli. Näiteks lõi Microsoft 1996. aastal Interneti Exploreri jaoks JavaScripti alusel enda keele - JScript. See põhjustas raskusi arendajatele, kes ei suutnud teha lehti mõlema brauseri jaoks. Probleemi lahendamiseks esitas Netscape 1996. aasta novembris JavaScripti standardiorganisatsioonile Ecma International (Wikipedia 2023d). Sellest arenes 1997. aasta juunis välja ECMAScript, standardiseeritud programmeerimiskeel, mis loodigi tagamaks koodi töötamise brauseriüleselt. (Wikipedia 2023g; W3Schools 2023c; Wikipedia 2023e)

Vahepeal saavutas Microsft Internet Exploreriga jõupositsiooni. 2000. aastate alguseks kasutas Internet Explorerit 95% internetikasutajatest. Kuigi alguses läks Microsoft ECMAScripti standardiga kaasa, lõpetasid nad hiljem koostöö Ecmaga. Kuna sisuliselt oli JScriptist Internet Exploreri populaarsuse tõttu saanud uus standardkeel, jättis see ECMAScripti arengu seisma. Internet Exploreri populaarsuse aegadel kestis veebiarenduses programmeerimiskeele vaatepunktist stagnatsioon. Seda muutis 2004. aastal ilmunud uus brauser, Firefox, mille lõi Netscape'st välja arenenud uus firma Mozilla (Wikipedia 2023i). Järgmisel, 2005. aastal ühines Mozilla Ecma standardiga. Tänu Firefoxi populaarsusele tõusis taas ECMAScripti olulisus, ent Microsoft oli siiski liiga suur, et oleks võinud seda standardi arendamisel ignoreerida. (Wikipedia 2023g)

2008. aastal tõi Google turule Chrome'i brauseri, mis oli teistest oluliselt kiirem. Kiirus

saavutati tänu *just-in-time* koodikompileerimisele (JIT), mis kompileerib koodi programmi töö ajal vahetult enne, kui seda vaja läheb, mitte kogu koodi enne programmi käivitamist (Wikipedia 2023h). Selle uuenduse mõju oli niivõrd suur, et 2008. aastal kohtusid firmad olukorra arutamiseks ja lepiti kokku ECMAScripti kasutamises. Selle kokkuleppe tulemus, ECMAScript 5, avalikustati 2009. aasta detsembris. Sellest ajast on ECMAScript pidevalt uuenenud ja JavaScript kui ECMAScripti spetsifikatsioonil põhinev programmeerimiskeel on kindlalt veebiarenduse standard. (Wikipedia 2023g; Wikipedia 2023e)

TODO: PRANGLIMISRAKENDUSES KASUTAMINE

2.2.4. React

React on vabavaraline avatud lähtekoodiga JavaScripti teek (Wikipedia 2023j). React on *front-end* raamistik, mida kasutatakse, et luua komponendipõhiseid kasutajaliideseid. See tähendab, et Reactiga tehtud kasutajaliidesed koosnevad väikestest, modulaarsetest osadest ehk komponentidest (ingl k *component*), mida omavahel kombineerides saab luua keerulisi lehekülgi. Kuigi React oli algselt loodud vaid veebilehekülgede loomiseks, saab sellega tänapäeval luua ka mobiiliäppe (raamistikuga *React Native*) või *full-stack* äppe (raamistikuga *Next.js*). (React 2023)

Kuigi veebiarenduse põhitehnoloogiaid on ainult kolm (HTML, CSS, JavaScript), ei tehta tänapäeval sisuliselt ühtegi suuremat veebilehte vaid nendega. Kuna veebilehed on ajas vaid keerukamaks muutunud, oleks tänapäeval ainult põhitehnoloogiate kasutamine aja- ja ressursikulukas väga pika koodi tõttu. Sellist koodi on ka raskem hallata, testida ja parandada. Seetõttu kasutataksegi veebiarenduses hulgaliselt teeke. Teegid on teiste programmeerijate loodud koodikogud, mida saab programmis vajaduspõhiselt kasutada (Wikipedia 2023k). Heaks näiteks on kasutaja autentimine: kuna turvaline autentimine on keeruline ja nõuab palju teadmisi ka näiteks krüptograafia valdkonnas, kasutatakse üldiselt autentimisteeke, mis teevad arendaja eest suure töö ära. Niimoodi ei pea iga arendaja looma enda koodibaasi (*code base*). Kuna enamik programmeerijaid ei ole krüptograafiaeksperid, teeksid nad enda autentimissüsteemi luues arvatavasti ka vigu, mistõttu võib teekide kasutamine olla ka turvalisem. Seega on teekide eelisteks nii turvalisus, lihtsam haldus kui ka pidev uuendamise võimalus.

Kasutajaliidese loomine on veebiarenduses väga olulisel kohal, seetõttu on selleks ülesandeks palju erinevaid teeke, millest kõige populaarsem on React (Stack Overflow 2022). Reacti populaarsul põhineb mitmel omadusel. Üks olulisematest on fakt, et React on JavaScripti raamistik. See tähendab, et enamik arendajatest ei pea Reacti kasutamiseks uut programmeerimiskeelt õppima, vaid saavad kasutada juba õpitud keelt. Veebiarenduses, kus olulisel kohal on nii arendusaeg kui ka koodi hooldatavus, on tuntud programmeerimiskeele, eriti JavaScripti kasutamine tihti raamistiku valiku puhul eelduseks. Veel on oluline Reacti modulaarne arhitektuur, mis võimaldab koodi korduvalt kasutada. Väga palju annab juurde ka Reacti suur kogukond, mis aitab teeki õppida, toetab üksteist ja korraldab erinevaid üritusi ja seminare, mis muudab teegi ajakohaseks ja töökindlaks. (Soomro 2023)

React on raamistikuna suhteliselt uus ja lühikese ajalooga. Reacti lõi 2011. aastal Meta tarkvaraarendaja Jordan Walke. Esimest korda kasutati seda samal aastal Facebooki veebilehel, hiljem ka Instagramis. Reacti lähtekood avati 2013. aasta maikuus. Kaks aastat hiljem avaldati ka Reacti mobiiliraamistiku React Native lähtekood. Sellest ajast saati on React olnud pidevas arenduses, näiteks uuendati 2017. aastal teegi renderdamisalgoritmi. Tänaseks on Reactist saanud ülipopulaarne teek, mida kasutavad lisaks millionitele teistele lehtedele näiteks Facebook, Instagram, Netflix ja ChatGPT. (BuiltWith 2023; Wikipedia 2023j)

Pranglimisrakenduse *front-end* kood on kirjutatud Reactis. React paistab teiste omasuguste seast silma just väga suure populaarsuse ja koodi mugava haldamise ning taaskasutamisega. Pranglimisrakenduse jaoks Reacti õppimine nõudis küll pisut aega, ent polnud ülemäära raske. Tänu JavaScripti tundmisele juba enne arenduse algust oli ka Reacti sellevõrra kergem omandada. Lisaks on selle projekti juures väga oluline rakenduse kiirus, nii veebilehe laadimisel kui lehesiseselt (näiteks uue tehte näitamisel kasutajale). React kasutab virtuaalset DOMi (*Document Object Model*, veebilehe struktuuripuu), mis võimaldab rakendusel iga muudatust testida ja seeläbi ka riske hinnata. Siiski on Reactil kui *front-end* raamistikul ka omad puudused. Kuna tegemist on võrdlemisi mahuka raamistikuga on ka failisuurused selle võrra suuremad.

2.2.5. Figma

Figma on veebirakendus kasutajaliideste disainimiseks. Figma on loodud disainerite reaalajas koostöö edendamiseks. See on võrdlemisi uus tööriist, mis on tarkvara disainimisprotsessi täielikult muutnud. Erinevalt teistest uurimistöös esitatud tehnoloogiatest, ei nõua Figma programmeerimisoskust, vaid on puhtalt disainimisplatvorm. (Wikipedia 2023f)

Idee veebipõhisest disainiplatvormist tuli Evan Wallace'le ja Dylan Field'le juba 2011. aastal. Nende eesmärkideks oli muuta disain kättesaadavamaks rohkematele inimestele ja võimaldada mitmetel disaineritel töötada korraga ühe projekti kallal ilma versioonihalduseta. (Figma 2024a) 2015. aasta detsembris lasti välja kutsepõhine Figma beetaversioon, järgmise aasta septembris esimene Figma avalik väljalase (Wikipedia 2023f). Esialgne tagasiside ei olnud positiivne. Figmat kirjeldati kui huvitavat, ent ebapraktilist ideed või isegi õudusunenägu. Reaalajaline koostöö arvati olevat disainerivaenulik. (Figma 2024a) Siiski leidis enamik disaineritest, et Figma võimaldab efektiivsemat ja kiiremat tööd, võimalust töötada väljaspool kontorit ja koos teistega. Seetõttu sai Figma kiiresti väga populaarseks. Tänapäeval kasutavad Figma võimalusi paljud suured tehnoloogiafirmad, teiste hulgas Dropbox, GitHub ja Airbnb. (*ibid.*)

Figma on rakendusena pidevalt arenemas. 2017. aastal avaldati Figma mobiiliversioon, millega on võimalik Figma prototüüpe mobiilsetel seadmetel katsetada (Google Play 2023). Lisaks saab Figmat kasutada ja eraldi rakendusena Windowsil, MacOSil ja Linuxil. 2021. aastal avaldati FigJam, digitaalne tahvel. FigJami kasutatakse tarkvara planeerimiseks, muuhulgas ka välearenduse põhimõtetel. (Figma 2024b)

Pranglimisrakenduse kasutajaliidese disain töötati välja Figmas. Valisin selle ülesande jaoks justnimelt Figma tänu selle lihtsusele, suurele populaarsusele ja võimalusele tööd lihtsasti jagada. Pranglimisrakenduse kasutajaliidese disainimise kogemuse pealt vastab Figma suuresti nendele põhjendustele. Leidsin, et enda ideede teisendamine Figmasse oli intuitiivne ja mugav. Enne Figma kasutamist olin murelik, kas suudan selle kasutamise piisaval määral õigeks ajaks ära õppida, ent alustamine on väga lihtne, sest pole vaja midagi alla laadida. Figma kasutamine oli mugav ka välearenduse kontekstist vaadatuna, sest kõik ettepanekud, mida kasutajaliidese osas tehti, sai väga kiiresti visualiseerida, ilma programmeerimata.

2.3. *Back-end* tehnoloogiad

2.3.1. *PHP: Hypertext Preprocessor*

Vastik keel

2.3.2. Laravel

2.3.3. MySQL

Kokkuvõte

Kasutatud materjalid

Agile Alliance. (2023) Agile Glossary. Loetud: https://www.agilealliance.org/agile101/agile-glossary/, 02.01.2024.

Andmekaitse ja infoturbe leksikon. (2023) Agile development. Loetud: https://akit.cyber.ee/term/2148-agile-development, 19.10.2023.

Ayebola, J. (2023) CSS Frameworks vs Custom CSS – What's the Difference? Loetud: https://www.freecodecamp.org/news/css-frameworks-vs-custom-css/, 03.01.2024.

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., C. Martin, B. M. andRobert, Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2023) Agiilse tarkvaraarenduse manifest. Loetud: http://agilemanifesto.org/iso/et/manifesto.html, 19.10.2023.

Bose, S. (2023) Top 5 CSS Frameworks for Developers and Designers. Loetud: https://www.browserstack.com/guide/top-css-frameworks, 03.01.2024.

BrainHub. (2023) Why Node JS? 7 Reasons Famous Companies Choose It. Loetud: https://brainhub.eu/library/why-node-js, 22.11.2023.

BuiltWith. (2023) React Usage Statistics. Loetud: https://trends.builtwith.com/javascript/React, 03.12.2023.

Denning, S. (2012) Is Apple Truly 'Agile'? Loetud: https://www.forbes.com/sites/stevedenning/2012/02/03/is-apple-truly-agile/, 02.01.2024.

Denning, S. (2015) Surprise: Microsoft Is Agile. Loetud: https://www.forbes.com/sites/stevedenning/2015/10/27/surprise-microsoft-is-agile/, 02.01.2024.

Dybå, T., Dingsøyr, T. (2008) *Empirical studies of agile software development: A systematic review.* Loetud: https://www.researchgate.net/publication/222827396_Empirical_studies_of_agile_software_development_A_systematic_review, 01.01.2024.

Figma. (2024) About Figma, the collaborative interface design tool. Loetud: https://www.figma.com/about/, 03.01.2024.

Figma. (2024) The Online Collaborative Whiteboard for Teams. Loetud: https://www.figma.com/figjam/, 03.01.2024.

Flynn, J. (2022) 16 Amazing Agile Statistics [2023]: What Companies Use Agile Methodology. Loetud: https://www.zippia.com/advice/agile-statistics/, 01.01.2024.

GeeksForGeeks. (2023) Frontend vs Backend. GeeksForGeeks. Loetud: https://www.geeksforgeeks.org/frontend-vs-backend/, 10.09.2023.

Google Play. (2023) Figma – prototype mirror share. Loetud: https://play.google.com/store/apps/details?id=com.figma.mirror, 03.01.2024.

Highsmith, J., Cockburn, A. (2005) Declaration of Interdependence. Loetud: https://web.archive.org/web/20180127094805/http://www.pmdoi.org/, 02.01.2024.

Larman, C., Basili, V. (2003) Iterative and incremental developments. a brief history. Loetud: https://ieeexplore.ieee.org/document/1204375, 02.01.2024.

Martin, R. C. (2009) Manifesto for Software Craftsmanship. Loetud: https://manifesto.softwarecraftsmanship.org/#/en, 02.01.2024.

MDN Web Docs. (2023) At-rules. Loetud: https://developer.mozilla.org/en-US/docs/Web/CSS/At-rule, 25.09.2023.

MDN Web Docs. (2023) CSS: Cascading Style Sheets. Loetud: https://developer.mozilla.org/en-US/docs/Web/CSS, 25.09.2023.

MDN Web Docs. (2023) HTML basics. Loetud: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics, 21.09.2023.

MDN Web Docs. (2023) HTML elements reference. Loetud: https://developer.mozilla.org/en-US/docs/Web/HTML/Element, 21.09.2023.

MDN Web Docs. (2023) JavaScript. Loetud: https://developer.mozilla.org/en-US/docs/Web/JavaScript, 26.09.2023.

Node.js. (2023) About Node.js® | Node.js. Loetud: https://nodejs.org/en/about, 22.11.2023.

Outworks Solutions Private Ltd. (2023) Agile Software Development: Real-World Impact and Tools for Agile Project Management. Loetud: https://www.linkedin.com/pulse/agile-software-development-real-world-k1iac, 01.01.2024.

PremierAgile. (2023) The Future of Agile Jobs: Trends and Predictions. Loetud: https://premieragile.com/future-agile-jobs-trends-predictions/, 01.01.2024.

PremierAgile. (2022) Top 7 Agile Frameworks. Loetud: https://premieragile.com/types-of-agile-frameworks/, 01.01.2024.

Raeburn, A. (2022) Extreme programming (XP) gets results, but is it right for you? Loetud: https://asana.com/resources/extreme-programming-xp, 01.01.2024.

React. (2023) React. Loetud: https://react.dev/, 22.11.2023.

Sacolick, I. (2022) A brief history of the agile methodology. Loetud: https://www.infoworld.com/article/3655646/a-brief-history-of-the-agile-methodology.html, 02.01.2024.

Shore, J., Warden, S. (2008) The Art of Agile Development. Loetud: https://drive.google.com/file/d/115H0oaGIxdBkXJ-uigiT5Tv2wToxMvGD/view?usp=drive_link, 19.09.2023.

Soomro, H. A. (2023) Why React is so Popular? Loetud: https://www.linkedin.com/pulse/why-react-so-popular-hamza-ali-soomro, 22.11.2023.

Stack Overflow. (2022) Stack Overflow Developer Survey 2022. Loetud: https://survey.stackoverflow.co/2022/, 22.11.2023.

State of Agile. (2022) 16th State of Agile Report. Loetud: https://digital.ai/resource-center/analyst-reports/state-of-agile-report/, 01.01.2024.

Striebeck, M. (2007) Agile Adoption at Google. Loetud: https://www.agilesociety.co.kr/news_file/google_agile.pdf, 02.01.2024.

University of Boston. (2020) History of CSS. Loetud: https://www.bu.edu/lernet/artemis/years/2020/projects/FinalPresentations/HTML/historyofcss.html, 25.09.2023.

University of Boston. (2020) History of HTML. Loetud: https://www.bu.edu/lernet/artemis/years/2020/projects/FinalPresentations/HTML/historyofhtml.html, 21.09.2023.

W3Schools. (2023) CSS Box Model. Loetud: https://www.w3schools.com/css/css_boxmodel.asp, 22.10.2023.

W3Schools. (2023) CSS Syntax. Loetud: https://www.w3schools.com/css/css_syntax.asp, 25.09.2023.

W3Schools. (2023) JavaScript History. Loetud: https://www.w3schools.com/js/js_history.asp, 22.11.2023.

W3Techs. (2023) Usage statistics of JavaScript as client-side programming language on websites. Loetud: https://w3techs.com/technologies/details/cp-javascript, 26.09.2023.

Wikipedia. (2023) Agile software development. Loetud: https://en.wikipedia.org/wiki/Agile_software_development, 01.01.2024.

Wikipedia. (2023) Brendan Eich. Loetud: https://en.wikipedia.org/wiki/Brendan_Eich, 22.11.2023.

Wikipedia. (2023) CSS. Loetud: https://en.wikipedia.org/wiki/CSS, 25.09.2023.

Wikipedia. (2023) Ecma International. Loetud: https://en.wikipedia.org/wiki/Ecma_International, 03.12.2023.

Wikipedia. (2023) ECMAScript. Loetud: https://en.wikipedia.org/wiki/ECMAScript, 03.12.2023.

Wikipedia. (2023) Figma. Loetud: https://en.wikipedia.org/wiki/Figma, 03.01.2024.

Wikipedia. (2023) JavaScript. Loetud: https://en.wikipedia.org/wiki/JavaScript, 26.09.2023.

Wikipedia. (2023) Just-in-time compilation. Loetud: https://en.wikipedia.org/wiki/Just-in-time_compilation, 03.12.2023.

Wikipedia. (2022) Koskmudel. Loetud: https://et.wikipedia.org/wiki/Koskmudel, 02.01.2024.

Wikipedia. (2023) Mozilla. Loetud: https://en.wikipedia.org/wiki/Mozilla, 03.12.2023.

Wikipedia. (2023) React (software). Loetud: https://en.wikipedia.org/wiki/React_(software), 03.12.2023.

Wikipedia. (2023) Teek. Loetud: https://et.wikipedia.org/wiki/Teek, 03.12.2023.

Wikipedia. (2023) Välearendus. Loetud: https://et.wikipedia.org/wiki/V%C3%A4learendus, 19.10.2023.

ÕS. (2018) Pranglima. Loetud: https://www.eki.ee/dict/qs/index.cgi?Q=pranglima&F=M, 19.11.2023.

Resümee

Abstract

This will contain the English translation of the „Resümee“ section.

Kinnitusleht

Uurimistöö autorina kinnitan, et

- koostasin uurimistöö iseseisvalt ning kõigile töös kasutatud teiste autorite töödele ja andmeallikatele on viidatud;
- uurimistöö vastab Tallinna Reaalkooli uurimistöö juhendile;
- olen teadlik, et uurimistööd ei edastata teistele tulu teenimise eesmärgil ega jagata teadlikult plagieerimiseks.

. . . . . .

kuupäev / nimi / allkiri

Juhendajana kinnitan, et uurimistöö vastab Tallinna Reaalkooli uurimistöö juhendile ja lubatakse kaitsmisele.

Juhendaja

. . . . . .

kuupäev / nimi / allkiri