Bakalaureusetöö ülesandepüstitus

Autor: Steven Juks (164662IAPB)
Juhendaja: Martin Verrev
Töö teema: "Kassapõhisel raamatupidamisarvestusel põhinev laoseisu haldamise veebirakendus"

Taust

Olemasolevad eestikeelsed laotarkvarad on suunatud pigem ettevõtetele, kellel on näiteks laoseis vaja ühendada e-poega, pidada palgaarvestust, ühendada tarkvara pangaga jne. Väikeettevõtjatel ei pruugi aga olla ressurssi selliste tarkvarade soetamiseks ja võib puududa ka vajadus kõigi nende funktsionaalsuste järele, mis muudavad rakenduse liiga keeruliseks. Näiteks kodumasinate remonditeenust pakkuvale FIE-le piisaks vaid laoseisu, müügiarvete, ostuarvete ning kulude ja tulude haldamisest, et võimalikult hõlpsalt laoseisust ning raamatupidamisest ülevaade saada.

Eesmärk

Töö eesmärgiks on luua eestikeelne veebirakendus, mis võimaldab kaupa lattu lisada või ostuarvete põhjal sisse osta, müügiarveid koostada ning kaupa laost välja müüa. Ostu- ning müügiarvete põhjal moodustab rakendus ka kulude ja tulude ülevaate. Rakenduse sihtgrupiks on peamiselt väikeettevõtjad, kes vajavad vaid laohalduse baasfunktsionaalsust.

Alternatiivsed lahendused

Olemasolevad rakendused, millest projekti realiseerimisel eeskuju saab võtta:

- Zoho[2] ja TradeGecko[3] - tasulised ingliskeelsed veebirakendused laoseisu ja raamatupidamise haldamiseks
- Wave[4] - tasuta ingliskeelne veebirakendus raamatupidamise haldamiseks
- Intellisoft Profit[5] - tasuline eestikeelne Windowsi rakendus laoseisu ja raamatupidamise haldamiseks

Olemasolevate rakenduste analüüsi põhjal sain aru, et olemasolevad rakendused on kas liiga keerulised - s.t. lisaks vajaminevale funktsionaalsusele on pakendatud kaasa mooduleid, mida vaja ei ole; sellised, mis võimaldavad hallata kas ainult ladu või ainult kassat - s.t. peaks tegelema nende liidestamisega ja lisaks ei võta ükski Eestist väljaspool arendatud tarkvara arvesse Eesti õigusakte.

Metoodika

Rakenduse realiseerimiseks ilmutan kõigepealt nõuded. Selleks konsulteerin rakenduse potentsiaalsete kasutajatega ja teen põhjaliku võrdluse teiste samalaadsete rakendustega. Seejärel loon Figmas[6] rakenduse disaini prototüübi. Valideerin prototüübi võimalikel kasutajatel ning võttes arvesse nendepoolset tagasisidet programmeerin Reactis[7] selle põhjal kasutajaliidese. Kasutajaliideses realiseerin andmeobjektid staatiliste JSON mudelitena. Kasutan *front-end-first* lähenemist[8], s.t. pärast kasutajaliidese valmimist realiseerin rakenduse serveripoole kasutades PostgreSQL[9] andmebaasisüsteemi ja Node.js-i[10] Express[11] raamistikuga. Tehnoloogiate valik tuleneb sellest, et olen eelnevalt antud tehnoloogiaid kasutanud ning oskan neid kõige paremini rakendada. Kui rakendus on valmis, siis hindan rakenduse nõuetele vastavust.

Viited

1. Accrual Accounting [WWW] https://www.entrepreneur.com/encyclopedia/accrual-accounting (12.02.2019)
2. Zoho [WWW] https://www.zoho.com/books/ (12.02.2019)
3. TradeGecko [WWW] https://www.tradegecko.com/ (12.02.2019)
4. Wave Financial [WWW] https://www.waveapps.com/ (12.02.2019)
5. Profit [WWW] http://intellisoft.ee/profit/ (12.02.2019)
6. Figma - the collaborative interface design tool [WWW] https://www.figma.com/ (12.02.2019)
7. React [WWW] https://reactjs.org/ (27.01.2019)

8. Front-end first software development approach [WWW] https://medium.com/swlh/front-end-first-software-development-approach-1aa8b7f35447 (27.02.2019)
9. PostgreSQL [WWW] https://www.postgresql.org/ (27.01.2019)
10. Node.js [WWW] https://nodejs.org/en/ (27.01.2019)
11. Express - Node.js web application framework [WWW] https://expressjs.com/ (27.01.2019)