TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Liisa Heinla 194147IADB

Veebirakenduse loomine testiaruandluse koostamiseks Videobeti näitel

Bakalaureusetöö

Juhendaja: Maili Markvardt
MSc

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Liisa Heinla

16.04.2022

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua veebirakendus firmale nimega Videobet, mis kuulub Playtech PLC-le. See rakendus võimaldab anda ülevaate käimasolevatest toote kliendile üleandmisega seonduvatest protsessidest. Samuti võimaldab veebirakendus arvutada välja keskmise toote üleandmispaketi testimise ja arendamise aja, näidata testimise edukust ning kas toote üleandmise protsess jõudis õigeks ajaks valmis.

Lõputöö käigus analüüsiti erinevaid tehnoloogiaid ja valiti nende seast välja arenduseks sobilikud vahendid. Esirakenduse puhul kasutati Vue.js raamistikku, tagarakenduse puhul Springi ja Javat ning andmebaasina oli kasutuses MySQL andmebaas. Loodi firma poolt antud nõuetele vastav rakenduse MVP. Seda rakendust hakkavad kasutama testimise juht, üleandmistestidega tegelevad testijad, arendajate juht ning firma projektijuhid. Rakendusega muudeti protsess läbipaistvamaks, et juhid saaksid langetada adekvaatseid otsuseid ning seeläbi kiirendada ja muuta üleandmised ajaliselt efektiivsemaks. Rakendus täidab kõik firma poolt seatud funktsionaalsed ja mittefunktsionaalsed nõuded ning on abiks üleandmisi puudutavate valikute tegemisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 43 leheküljel, 7 peatükki, 18 joonist, 4 tabelit.

Abstract

Web application creation for the reporting of testing: case of Videobet

The purpose of the current thesis is to create a web application for a company called Videobet, which is owned by Playtech PLC. This application makes it possible to give an overview of ongoing product deliveries to clients. In addition, the web application calculates the average time for how long the delivery package is developed and tested, shows how successful the testing was and if the delivery process was finished on time.

Within the thesis, the author analyzed different technologies and chose the most useful ones to develop the web application with. For the front-end part of the application, Vue.js was used. For the back-end, the author used Spring and Java. MySQL was used for creating the database. An MVP was created, which matched the requirements given by the company. This application will be used by the testing manager, testers dealing with delivery testing, developers' team lead and project managers. With the creation of the web application, the delivery process is made more transparent, so that the Videobet team could make adequate decisions. This will speed up the delivery process and makes the deliveries as well-timed as possible. The application fulfills every requirement given by the company.

The thesis is in Estonian and contains 43 pages of text, 7 chapters, 18 figures, 4 tables.

Lühendite ja mõistete sõnastik

AOP	*Aspect Oriented Programming*, aspektorienteeritud programmeerimine
API	*Application Programming Interface*, rakendusliides. Võimaldab kahel rakendusel omavahel suhelda
Batch	Kogum SQL päringuid
CPU	*Central Processing Unit*, arvuti põhiline protsessor
CRUD	*Create, Read, Update, Delete,* neli andmebaasiga suhtlemiseks vajalikku meetodit
CSRF	*Cross-site request forgery*, rünnakuliik, mis sunnib kasutajat veebirakendusele kahjulikke päringuid tegema
CSS	*Cascading Style Sheets,* keel veebilehe disaini kujundamiseks
Data Access Layer	Andmete ligipääsukiht, mis lubab rakenduses andmetele kergemini ligi pääseda
Dependency Injection	Sõltuvuste pealesüstimine, sõltuvus lisatakse manuaalselt
DOM	*Document Object Model*, programmeerimisliides, mis lubab veebilehte muuta
ERD	*Entity Relationship Diagram*, olemisuhtediagramm
FURPS	*Functionality, Usability, Reliability, Performance, Supportability,* nõuete haldamise mudel
HTML	*HyperText Markup Language*, veebilehtede loomiseks vajalik märgendkeel
HTTP(S)	*Hypertext Transfer Protocol (Secure)*, veebis andmete saatmist võimaldav protokoll
JIT kompileerija	*Java-In-Time* kompileerija, optimeerib rakenduse jõudlust
JPA	*Java Persistence API*, kogum klasse ja meetodeid, mis lihtsustavad rakendusel andmebaasiga suhtlemist

JSP	*Java Server Pages*, võimaldab kirjutada Java veebirakendustele lehti
JSON	*JavaScript Object Notation*, skriptimiskeel veebiarenduseks ja andmete kompaktseks transportimiseks
JVM	*Java Virtual Machine*, Java virtuaalmasin
MVP	*Minimum Viable Product*, toode, mis on piisavalt valmis, et seda klientidele varakult tutvustada
.NET	Microsofti poolt loodud raamistik
PDF	*Portable Document Format*, Adobe poolt loodud faililiik
PHP	*Hypertext Preprocessor*, skriptimiskeel veebilehe protsesside haldamiseks
Pointer	Muutuja, mis hoiab endas mäluaadressi
REST	*Representational State Transfer*, arhitektuuristiil, mis kasutab HTTP päringuid andmete edastamiseks
SPA	*Single Page Application*, üheleherakendus, dünaamiliselt sisu muutev leht
SQL	*Structured Query Language*, keel andmebaasiga suhtlemiseks
SQL süstimine	Rünnakuliik, kus rakenduse väljadele sisestatakse SQL laused, mis teevad andmebaasile kahju
URL	*Uniform Resource Locator*, veebilehe aadress
XSS	*Cross-Site Scripting*, rünnakuliik, kus veebirakendus pannakse kahjulikke skripte kasutaja peal käivitama

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Firmas Videobet omab suurt tähtsust toote üleandmise testimine. See on viimane samm enne, kui lõplik toode kliendile üle antakse. Käesolev lõputöö on loodud vastavalt Videobeti töötajate vajadusele omada ülevaadet üleandmistestidest. Sellest nõudlusest lähtuvalt loob töö autor enda firma töötajate jaoks veebirakenduse MVP, mis kajastaks vajalikke andmeid parema ülevaate omamiseks ning võimalike probleemide ja tähtaegade ületamise vältimiseks.

Töö teises peatükis tutvustatakse firma ja testimise tausta, samuti ka millist rolli omab üleandmistestimise protsess Videobeti testimises üldisemalt. Kolmandas osas käsitletakse probleeme, mis hetkel firmas üleandmistestidega seonduvalt on ning kuidas veebirakendus need lahendaks. Neljandas osas kogutakse kokku nõuded, mida rakendus peaks täitma. Selle põhjal määratakse lõputöö skoop. Vastavalt nendele nõuetele tehakse analüüs tehnoloogiatest, millised oleksid parimad variandid veebirakenduse arendamiseks ning lahatakse nende olemust ja sobivust. Analüüsi põhjal tehakse valik. Valmib ka prototüüp.

Töö viiendas peatükis kirjeldatakse, kuidas valmis veebirakendus ning tutvustatakse lähemalt andmebaasi ning esi- ja tagarakendust.

Kuuendas osas analüüsitakse arenduse tulemusi. Esitatakse lühiülevaade, kuidas rakendust testiti. Samuti küsitakse rakendusega tulevikus kokku puutuvatelt isikutelt tagasisidet ning tuuakse välja võimalikud funktsionaalsused, mida edaspidi käesolevale projektile juurde lisada.

2 Praeguse süsteemi kirjeldus

Selles peatükis kirjeldatakse Videobeti ning selle toimimist. Samuti käsitletakse Videobeti testimise liike ning Jira ja TestRaili rolli üleandmistestimises.

2.1 Ettevõte

Videobet OÜ on Playtech PLC-le kuuluv ettevõte, mis tegeleb kasiinode mängumasinatele tarkvara ja mängude arendamisega. Videobet on jaotatud allosadeks ehk tiimideks, kelle vastutusala hõlmab olulisi valdkondi arendusprotsessis. Firmas on iga kliendi kohta eraldi projekt ning enamasti on käimas mitu projekti korraga. Videobeti kliente leidub Suurbritanniast, Iirimaalt, Soomest, USA-st, Taanist, Nigeeriast ning paljudest teistest riikidest. Bakalaureusetöö autor on antud firmaga seotud, töötades seal tarkvara testijana 12-liikmelises tiimis, mida juhib testijuht.

2.2 Testimise struktuur

Testimine on viimane samm, mis tehakse enne, kui toode kliendile üle antakse. Arendusprotsessi keskel või lõpus antakse toode testijatele ning arendus kestab senikaua, kuni toode on testijate poolt plaaniliselt läbi vaadatud ning nende avastatud vead ära parandatud. Keskmiselt testitakse ühte projekti 1-4 kuud, sõltuvalt selle mahust ning prioriteediastmest. Vastavalt projektis elluviidavatele muudatustele oleneb, kui süvitsi testimisega minnakse. Näiteks võib tuua raha käsitlemise – kõik sellega seonduv tuleb põhjalikult läbi katsetada ja proovida, vastasel juhul võivad väiksemadki vead põhjustada kliendile majanduslikku kahju. Uuendusi, mis pole nii kriitilise tähtsusega ja on pigem väikesed, võib pinnapealselt testida.

Testijate vastutusala on testimise planeerimine ja elluviimine. Nad tegelevad igapäevaselt mitmete erinevate testimisliikidega, nagu näiteks jõudlustestimine, mängude testimine, platvormi testimine ja üleandmistestimine.

2.2.1 Üleandmistestimine

Delivery ehk toote üleandmine on protsess, kus kliendile antakse üle valmis toode, mis vastab kokkulepitud nõuetele ning sisaldab endas tarkvaralisi ja funktsionaalseid uuendusi.

Üleandmistestimine ehk *delivery* testimine on Videobetis kasutatav testimisliik, mis läbitakse iga toote testimisfaasi lõpus ehk vahetult enne üleandmist ennast. Tehes tihedalt koostööd projektijuhtide ja arendajatega, antakse üleandmistestimise käigus ülevaade toote seisust enne, kui see edastatakse kliendile. Arendajad loovad toote koopia paketi, mis peaks olema valmis kliendile üleandmiseks ning edastavad selle üleandmise testimisega tegelevale testijale.

Üleandmistestimise peamine eesmärk on garanteerida toote terviklikkust ning kliendi rahulolu iga kord, kui toode üle antakse. Samuti hõlmab see erinevate kliendipoolsete nõuete muutumise jälgimist, et tagada testimise kõrge kvaliteet piiratud aja jooksul, mil testimist läbi peab viima. Üleandmistestidega tegelevad peamiselt vanemtestijad, sest tegemist on äärmiselt ajakriitilise protsessiga – üleandmistestid peavad alati olema kindlaks kuupäevaks valmis. Vajalik on põhjalik toote tundmine ning erinevate ajaga omandatud oskuste ja tööriistade valdamine.

Selle testimisliigi protsess on mitmeastmeline. Kõigepealt valmistab üleandmise eest vastutav arendaja ette baaskeskkonna ja toote, mida testima hakatakse. Arendaja ei loo neid nullist, vaid koondab kokku vastava koodi ja toodangukeskkonnas kasutatavad konfiguratsioonid. Sealhulgas lisatakse tootesse kõik kliendile omased funktsionaalsused, mängud ja uuendused. Kui baaskeskkond ja toode on valmis, antakse info paketi valmimisest testimisjuhile edasi, kes määrab üleandmisele testija. Testija töötab vastavalt iga kliendi jaoks kohandatud kontrollnimekirjale ning püüab baaskeskkonda ja toodet kliendi poolt kasutatavatele masinatele paigaldada. Kui toode ei toimi, saadetakse see arendajale koos infoga, mis valesti läks. Arendaja parandab vead ära ning annab toote testijale tagasi. See protsess toimub senikaua, kuni vigu enam ei leita ning toode on kontrollnimekirja edukalt läbinud või kuni tähtaeg kätte jõuab.

2.2.2 Jira roll üleandmistestimises

Jira on tööriist, mis esialgu disainiti programmivigade ja probleemide kaardistamiseks, kuid mis arenes edasi palju laiahaardelisemaks, võimaldades tegeleda agiilse arendusega

ning programmide testimise haldamisega [1]. Videobetis kasutatakse Jirat nii arenduses kui ka testimises programmivigadest teatamiseks, üleandmistestide pakettide edastamiseks testijatele ning üleandmise informatsiooni vahendamiseks ja talletamiseks projektijuhtide, arendajate ja testijuhi tarbeks. Mainitud informatsiooni hulka kuulub kuupäevade arvestus, programmi baaskeskkonna kirjeldus ja versioonid, paketis sisalduv sisu (mängud, mängude bännerid) ning muu sarnane üleandmistestide jaoks vajalik teave. Jira kommentaaridesse märgitakse testija poolt testimisprotsessi kirjeldus – mis on seni hästi ning mis on halvasti läinud. Jirale pääsevad kõik Videobeti töötajad ligi ning neil pole piiranguid, mida nad üleandmistestide suhtes vaadata ja teha võivad.

2.2.3 TestRaili roll üleandmistestimises

TestRail on veebipõhine testide haldamise keskkond. See võimaldab hallata tervet testimise protsessi ning teste läbi viia [2]. TestRaili kasutakse Videobetis rohkem tiimisiseselt, sest individuaalsete testide tulemusi vajavad pigem testijad ja testijuht kui projektijuhid ja kliendid. Loeb kokkuvõtlik tulemus, mille TestRail iga projekti testimissessioonide kohta teeb – mitu testi läbi kukuti ning kui palju neist õnnestus.

Üleandmistestimine viiakse osaliselt läbi TestRailis, kuid peamine fookus on ikkagi Jiral. TestRail on ülevaatliku rolliga. Seal koostatakse vastavalt kliendile testimissessioonid ning viiakse neid järjest läbi. TestRaili kaudu on testijal võimalik teha üleandmisest kokkuvõte.

3 Lähteülesanne

Käesolevas peatükis kirjeldatakse probleemi ning sellest lähtuvalt tuuakse välja eesmärk, mida lõputöö autor bakalaureusetöö raames saavutada plaanib.

3.1 Probleemi kirjeldus

Lõputööga lahendatakse mitu Videobeti testimise protsessis esinevat probleemi korraga. Käesoleva lõputöö teema pakkus välja Videobeti testimisjuht pärast seda, kui töö autor temaga konsulteeris. Üheks probleemiks oli puudulik ülevaade üleandmistestidest. Need testid on äärmiselt ajakriitilised ning peavad alati kindlaks kuupäevaks valmis olema. Hetkel ei ole võimalik omada üldpilti, kui palju aega kulub keskmiselt ühe üleandmise ettevalmistamisele ja testimisele. Testimise edenemine sõltub peamiselt sellest, kui kiiresti parandatakse leitud vead, et testija saaks oma tööd jätkata. Informatsioon, kui palju kordi „põrgatati“ toodet arendaja ja testija vahet, annab edasi olulist teavet – kui vigane on toode, mis ette valmistati. Mida suurem arv „põrkeid“, seda enam probleeme tootes esines ning seda vähem aega jääb testijal testimiseks. Keeruline on testida toodet piiratud ajamahuga. Paratamatult peab kannatama testimise kvaliteet, sest tähtaegade osas mööndusi sageli ei tehta.

Teiseks probleemiks oli andmete killustatus. Üleandmistestide andmeid hoitakse kahes kohas: Jiras ja TestRailis. Kahte kohta koondunud andmete puhul on puuduseks, et neid peab kahest erinevast kohast otsima. Enamasti toimub korraga mitu üleandmist, mistõttu suureneb otsitavate andmete maht veelgi. Kui juhtivatel isikutel pole ülevaadet, mis seisus üleandmistestid on, suureneb risk, et testimine venib pikemaks kui esialgu plaanitud oli. Testimine ei saa ettemääratud tähtajaks valmis ning toote kliendini jõudmine viibib. See omakorda õõnestab kliendi usaldust firma vastu ja Videobet võib potentsiaalselt kliente ka kaotada.

Kolmandaks probleemiks oli puuduv viis, kuidas oleks võimalik üleandmistestide andmeid ja tulemusi kliendile edastada. Samuti on see informatsioon oluline ka firma kõrgematele juhtidele ning projektijuhtidele, kes peavad olukorda pidevalt jälgima.

Äsjamainitud probleemid on olnud Videobetis pikalt ning testijatel ja testijuhil pole seni olnud aega ega ka oskusi, et neid lahendada. Probleemidele tuginedes oli võimalik autoril määrata eesmärk, mida lõputöö raames täita.

3.2 Eesmärk ja metoodika

Käesoleva lõputöö eesmärgiks on luua firmasisese veebirakenduse MVP, mille kaudu on testijuhil ja ka testijatel ning projektijuhtidel võimalik saada ülevaade üleandmistestide seisust – kes on hetkel üleandmise eest vastutaja, mis ajaks peab testimine valmis saama, kui kaua aega keskmiselt testimisele kulub, tootes esinevad vead jpm. Veebirakendus on valmistatud spetsiaalselt firma vajadustest lähtuvalt. Selle käigus loodab lõputöö autor saavutada üleandmistestidele kuluva keskmise aja vähenemise ning suurema läbipaistvuse, et tagada jätkusuutlik koostöö klientidega.

4 Veebirakenduse analüüs

Antud peatükis kirjeldatakse rakenduse nõudeid ning analüüsitakse arenduse tehnoloogiaid. Selle põhjal tehakse otsus, milliseid tehnoloogiaid hakatakse veebirakenduse arenduseks kasutama. Kirjeldatakse ka prototüüpi.

4.1 Rakenduse nõuded

Enne rakenduse arenduse algust on kõigepealt vaja välja selgitada nõuded, mida rakendus täitma hakkab. Konsulteerides Videobeti testijuhiga, oli lõputöö autoril võimalik kaardistada rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Autor pakkus ka enda tagasisidet ning nõuded said muudetud ja parandatud kõige optimaalsemateks. Nõustuti, et parim lahendus oleks luua MVP ning siis pärast rakenduse kasutuselevõttu vastavalt vajadusele funktsionaalsusi muuta ja/või juurde lisada.

4.1.1 Funktsionaalsed nõuded

Järgnevalt kirjeldatud nõuded on MVP jaoks vajalikud ning veebirakenduse loomisel on autor neist lähtunud. Funktsionaalseteks nõueteks on:

1) Võimalus sisse logida, kasutades töötaja unikaalset kasutajanime ja parooli, millega pääseb ligi ka Jirale ja TestRailile.

2) Võimalus rakendusest välja logida.

3) Rakendus peab kasutama Jirast ja TestRailist pärinevaid andmeid.

4) Rakendusel peab olema võimalus otsida toote üleandmisi nime järgi.

5) Rakendus peab kuvama staatust, toote üleandmise nime, tähtaega ja kokku arvutama toote üleandmise peale kulunud aja.

6) Võimalus vaadata iga toote üleandmise detaile.

7) Toote üleandmise detailides peavad olema märgitud kommentaarid, hetkel toote üleandmisega tegelev isik, tulemused TestRailist ning arendusele ja testimisele kulunud aeg.

8) Võimalus saada ülevaade mingi kindla ajaperioodi üleandmistest (tähtaeg langeb valitavate kuupäevade vahele).

9) Võimalus näha, kui paljud toote üleandmised said õigeks ajaks valmis ning kui paljud jäid hiljaks.

10) Rakendus peab välja arvutama, kui palju aega on kulunud kindla toote üleandmise protsessile. See jaguneb omakorda selleks, kui palju kulus aega testimisele ning kui palju aega kulus paketi ettevalmistamisele ja vigade parandamisele.

11) Võimalus näha ajajoont, milles on märgitud olulisemad kuupäevad ning mis näitab toote üleandmise progressi.

12) Võimalus laadida alla summaarne PDF-formaadis aruanne mingi kindla ajaperioodi üleandmistest.
13) Võimalus laadida detailvaatest alla ühe üleandmise PDF-formaadis aruanne.

4.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsete nõuete kaardistamisel kasutati enamasti FURPS (*Functionality, Usability, Reliability, Performance, Supportability*) põhimõtteid, mis kombineeriti firma privaatsuspoliitikaga.

1) Võimalus tulevikus rakendust laiendada ning funktsionaalsust juurde ehitada.

2) Lähtekood on kättesaadav ainult firma töötajatele.

3) Sobivus erinevate brauseritega nagu näiteks Google Chrome, Mozilla Firefox ja Microsoft Edge. Mobiilivaate tugi ei ole lõputöö skoobis nõutud.

4) Rakendus on kättesaadav ainult firma enda töötajatele. Sellele pääseb ligi vaid firma sisevõrgus olles ning töötaja isikliku kasutajaga, mida kasutatakse nii Jirasse kui ka TestRaili sisselogimiseks.

4.2 Rakenduses kasutatava tehnoloogia valik

Tehnoloogia valik on vast kõige olulisem tegur, mis määrab ära kui efektiivselt rakendus luuakse ning kui hästi see toimima hakkab.

4.2.1 Tagarakenduse tehnoloogia valik

Tagarakenduse arendus hõlmab endas serveripoolset arendust, millele tavakasutajatel ligipääsu pole. Selle kaudu saab teha andmetöötlust ja andmebaasipäringuid, mistõttu kutsutakse tagarakendust ka programmi *Data Access Layer*'iks ehk kohaks, kust pääseb andmetele ligi. [3]

Tagarakendus koosneb tavaliselt raamistikust ning programmeerimiskeelest, mis teineteist täiustavad ning millel on erinevaid plusse ja miinuseid. Oluline tegur tagarakenduse tehnoloogia valikul on lõputöö autori enda kogemus nende raamistike ja programmeerimiskeeltega. Järgnevalt on välja toodud 3 raamistiku ja keele kombinatsiooni, mida hakatakse võrdlema ja analüüsima. Raamistike ja keelte loetelu valik on tehtud vastavalt autori programmeerimiskeelte pädevusele.

Python ja Django

Python on objektorienteeritud programmeerimiskeel, millel on ka funktsionaalse programmeerimiskeele omadusi. Sellel on efektiivsed kõrgtasemelised andmestruktuurid, mis koos kergelt õpitava süntaksiga moodustavad hea keele, mida kasutada skriptimiseks ning kiireks rakenduse arenduseks erinevatel platvormidel [4].

Python on paindlik ning võimaldab kiiret arendust tänu sellele, et see on dünaamiliselt tüübitud. See tähendab seda, et Python võimaldab kirjutada muutujaid ilma muutuja tüüpi määramata, mistõttu arendaja ei pea tüüpide määramisele aega kulutama. Selle õppimine on kerge ning kogukond, kes aktiivselt Pythonit täiustab ning ka teisi Python arendajaid koodiprobleemidega aitab, on väga suur ja aktiivne. Samuti on Python interpreteeritud keel, mis tähendab, et kood pannakse tööle rea haaval, et avastada vead niipea kui võimalik. [5]

Pythoni negatiivseteks külgedeks on aga selle aeglus. Just see, et Python on interpreteeritud keel, tähendab, et koodi käimapanemine võtab rohkem aega kui muude mitte-interpreteeritud keelte puhul [6]. Samuti on Pythonil suurenenud mälu tarbimine peamiselt muutuja tüüpide puudumise tõttu [7]. Mobiilarenduseks Pythonit üldjuhul ei kasutata, sest nii IOS kui ka Android ei toeta seda mobiilarenduse keelena [8].

Django on Pythoni veebiarenduse raamistik. Django tugev eelis on, et see on kiire – see võimaldab arendajatel kiiresti kavanditest päris tulemusteni liikuda. Django on

skaleeritav ning võimeline järsult kasvava nõudlusega toime tulema. Sellega seoses on Django puhul võimalik igale tarkvarakihile riistvara juurde lisada [9]. Arendajad saavad funktsionaalsust lisada ilma, et nad peaksid palju koodi kirjutama, sest Djangoga tulevad kaasa valmis paketid, mis ise funktsionaalsuse rakendusse loovad. Tugev REST raamistiku tugi on samuti väga suur eelis, mis võimaldab API-sid luua ja testida [10].

Django miinuseks on asjaolu, et Django pole mõistlik väiksemate projektide jaoks, sest hoolimata pakettide olemasolust eeldab see mujal palju koodikirjutamist ning on mõeldud pigem laienevatele programmidele, mida on tulevikus tõenäoliselt vaja skaleerida [10]. Pärast teiste programmeerimiskeelte kasutamist on raske Pythonit koos Djangoga kasutama hakata. Erinev on süntaks, mis muudab õppimise aja pikemaks.

ASP.NET ja C#

C# on objektorienteeritud ja komponentorienteeritud programmeerimiskeel. See on väga sarnane C, C++, Java ja JavaScript keeltega. C# abil on võimalik luua turvalisi ja robustseid rakendusi, mis toimivad .NET toel [11].

Sarnaselt Pythoniga on ka C# objektorienteeritud, mis jagab rakenduse väiksemateks tükkideks. Neid tükke on omakorda kergem ehitada, kombineerida ja hallata. Süntaks on inimkeelega väga sarnane, mistõttu on seda kergem õppida. C# on ka tüübikindel keel, mis tähendab, et muutujate väärtused pärast nende deklareerimist muutuda ei saa. C#-l on väga põhjalik dokumentatsioon, mis hõlmab endas interaktiivseid õpetusi, videoid ja probleemide lahendusi. C# on kompileeritud keel. [12]

C# puudustest võib välja tuua, et just tänu sellele, et C# on kompileeritud keel, tuleb seda iga kord uuesti kompileerida, kui muudetakse isegi väikeseid detaile. See võtab palju aega ning võib ajapikku tüütuks muutuda. Rakendus pole paindlik, sest sõltub suuresti .NET raamistikust. Samal põhjusel pole võimalik rakendust serveris jooksutada, kui sellel pole Windowsi keskkonda seadistatud. [12]

ASP.NET ehk *Active Server Pages.NET* on raamistik, mis võimaldab luua veebilehti ja veebitehnoloogiaid. ASP.NET-il on võimekad tööriistad, mille kaudu saab arendaja kiiresti ja efektiivselt koodi kirjutada. Raamistiku paindlikkus ja objektorienteeritud funktsionaalsused on selle peamised boonused. Selleks, et rakendust turvalisemaks teha, peab arendaja rohkem vaeva nägema kui näiteks Pythoni ja Django puhul. [13]

ASP.NET miinuseks võib lugeda, et raamistiku ülalpidamine on kulukas ja ressursse-kulutav. Samuti vajab see paremaid või rohkemaid veebiservereid kui teised keeled. [13]

Java ja Spring

Java on objektorienteeritud programmeerimiskeel. Java programm on võimeline toimima igal platvormil, olgu selleks Windows, Linux või macOS. Kui arvutil on JVM (*Java Virtual Machine*) olemas, siis on võimalik ka Linuxi platvormil kompileeritud koodi näiteks Windowsil jooksutada [21].

Java boonuseks on selle sõltumatus platvormist. Võrreldes teiste keeltega on Java lihtne. Samuti on see ülimalt töökindel ja robustne – rõhku pannakse vigade varajasele avastamisele. Suur jõudlus on tagatud sellega, et Java kasutab JIT kompileerijat, mis kompileerib ainult parasjagu kasutatavaid meetodeid. Turvariskide vähesus on boonus, mis on tagatud *pointer*'ite puudumisega [21].

Java puudusteks võib lugeda mälukasutuse ning sellest tuleneva aegluse võrreldes näiteks C ja C# keelega. Java kood võib olla väga kompleksne, sest kasutab pikki ja keerulisi lauseid ning teeb koodi vähem loetavaks. Java „prügikorjaja" ehk *garbage collector* võimaldab eemaldada kasutuseta objektid, kuid tarbib selle puhul liiga palju CPU võimsust. See võib tekitada probleeme jõudluse ja kiirusega [22].

Spring on Java kõige populaarsem teek. See on kergekaaluline nii suuruselt kui ka läbipaistvuselt. Springi teeki saab kasutada igasuguste Java rakenduste arendamiseks, mis on üks selle populaarsuse põhjusi. Springi positiivne külg on modulaarne korraldatus – pakette ja klasse on väga palju, kuid kasutada saab ainult neid, mis vaja ning ülejäänut saab ignoreerida. *Dependency Injection* funktsionaalsus võimaldab klasse iseseisvamaks muuta ning neid tulevikus taaskasutada [23]. AOP (*Aspect Oriented Programming*) lisab võimaluse, et arendajal saavad olla erinevad kompileerimisüksused või eraldi klassilaadur [24].

Spring on tegelikult üpris kompleksne ning pakub vahel liiga paljusid erinevaid võimalusi, mis võivad algajad arendajad segadusse ajada. Lisaks pole selle dokumentatsioon nii põhjalik – näiteks pole seal kirjas infot XSS või CSRF rünnakute vältimise kohta [24].

Kokkuvõte

Tabelis 1 on välja toodud kokkuvõte analüüsitud tagarakenduse tehnoloogiate positiivsetest ja negatiivsetest külgedest.

Tabel 1. Tagarakenduse tehnoloogiate võrdluse kokkuvõte

Tehnoloogia	Positiivne	Negatiivne
Python	- Paindlik - Võimalik kirjutada muutujaid ilma muutuja tüüpi määramata - Kerge õppida ja suur kogukond - Varajane vigade avastamine	- Aeglane - Koodi käivitamine võtab rohkem aega - Suurenenud mälutarbimine - Autoril vähe kogemusi Pythoniga
Django	- Kiire - Kergesti skaleeritav - Kaasas valmis paketid - Tugev REST raamistiku tugi	- Pole mõistlik väiksemate projektide jaoks - Keeruline kasutuselevõtt kui on kogemusi teiste keeltega - Autoril puudub kogemus
C#	- Inimkeelega sarnane süntaks - Rakenduse jaotumine väiksemateks tükkideks - Tüübikindel - Põhjalik dokumentatsioon	- Pidev vajadus koodi uuesti kompileerida - Rakenduse sõltuvus .NET raamistikust. - Sõltuvus Windowsi keskkonnast - Autori pigem vähene kogemus.
ASP.NET	- Võimekad tööriistad - Paindlikkus	- Kulukas ülalpidamine - Vajab paremaid või rohkemaid veebiservereid - Autoril puudub kogemus
Java	- Sõltumatus platvormist	- Mälukasutusest tulenev aeglus

	- Töökindel ja robustne - Vigade varajane avastamine - Puuduvad *pointer'*id - Lihtne keel - Autori suurem kogemus	- Kompleksne kood - „Prügikorjaja“ tarbib palju CPU võimsust
Spring	- Kergekaaluline - Mitmekülgne - Modulaarne korraldatus - *Dependency Injection* - AOP võimalus - Autori suurem kogemus	- Kompleksne - Kohati segadusttekitav - Puudub põhjalik dokumentatsioon

Vaadates Pythonit ja Djangot kui võimalikke veebirakenduse tagarakenduse tehnoloogiaid, räägib nende vastu rohkem, kui poolt. Pythoni ja Django kasutamise poolt on asjaolu, et neil on laiahaardeline tugi ning informatsiooni nende tehnoloogiate kohta on külluses. On olemas ka REST API arendamise võimalus. Kui mainida puudusi, siis Django ei ole sobilik väiksemate projektide jaoks nagu on käesolev veebirakendus. Veebirakendus pole nii mahukas ja tõenäoliselt vajab tulevikus vähe skaleerimist. Django kasutamine ning õppimine pärast autori pikemat kogemust Javaga on ajakulukas ja aeglustab arendusprotsessi. Samuti pole Python Videobetis laialt kasutusel, mis teeb keeruliseks programmi firmasse integreerimise Videobeti Java arendajate toel.

Lõputöö autori kokkupuude C# keelega on pigem madal ning ASP.NET kogemus puudub. Tugev sõltuvus .NET raamistikust muudab programmi üpris rigiidseks. C#-ga tuleb erinevate platvormide jaoks kasutada erinevaid käitusaegu ja kohandada koodi vastavatele süsteeminõuetele [12]. Nende asjaolude tõttu otsustas lõputöö autor C# keelt ja ASP.NET raamistikku mitte kasutada.

Java ja Springi poolt räägib asjaolu, et need ei sõltu platvormist ning Javal on suur jõudlus. Olles üks enamlevinumaid programmeerimiskeeli, on Java jaoks olemas palju

õpetlikku sisu, samuti ka Springi jaoks. Negatiivne külg on Springi laiahaardelisus, mille mõistmine võib olla ajakulukas ning tekitada segadust. Lõputöö autor otsustas ikkagi Java ja Springi kasuks, sest tal on varasemalt pikem kogemus Java arendusega ning ka Spring pole tema jaoks võõras. Samuti kasutatakse Videobetis palju Javat, mistõttu on tulevaste edasiarenduste puhul võimalus firma arendajatelt tuge saada ning ka koodi integreerimine on kergem.

4.2.2 Esirakenduse tehnoloogia valik

Esirakendus on rakenduse osa, millega kasutaja suhelda saab. Seda nimetatakse ka rakenduse kliendipoolseks osaks. Esirakenduse vastupidavuse ja jõudluse tagamine on olulised asjad, mida autor peab silmas pidama, kui esirakendust arendama hakkab. [25]

Lihtsustatult on esirakendus kombinatsioon kahest elemendist: graafilisest disainist ja kasutajaliidesest. Mõlemad luuakse eraldi, kuid enamus tehnilisest tööst läheb kasutajaliidesesse kasutades tehnoloogiaid nagu HTML, CSS ja JavaScript. [26]

Töö autoril on kokkupuude olnud kolme erineva JavaScript raamistikuga, mille hulgast valitakse üks lõplik, et veebirakenduse esirakendust arendada.

Vue.js

Vue.js on progressiivne JavaScripti raamistik kasutajaliideste ehitamiseks. See ehitatakse HTML-i, CSS-i ja JavaScripti peale ning pakub deklaratiivset ja komponendipõhist programmeerimismudelit, mis aitab tõhusalt kasutajaliideseid arendada. Vue.js eelised on selle väike maht (allalaadimise algsuurus 18KB) ja virtuaalne DOM renderdus, millest tulenevalt on sellel parem jõudlus. Samuti on võimalik HTML-i, CSS-i ja JavaScripti kood paigutada ühte faili, mis võimaldab komponentide taaskasutust ja koodi paremat loetavust. Vue.js on kergesti õpitav. [27]

Vue.js miinusteks on keelebarjäär – nimelt on Vue.js Hiinas ülimalt populaarne, mistõttu leidub veebis palju hiinakeelset materjali. Lõputöö autor Hiina keelt ei räägi, mistõttu on raskendatud materjali otsimine ja omandamine. Vue.js'il puuduvad ka paljud pluginad, mis teiste esirakenduse teekide ja raamistike, näiteks React.js, puhul olemas on. Vue.js on suhteliselt uus raamistik. [27]

React.js

React.js on üks populaarsemaid JavaScripti teeke. React on teek, millel on raamistiku omadusi, kuid mis ise raamistik ei ole, nagu ekslikult arvatakse. React.js on sarnane Vue.js'iga: sellel on võimalik komponente taaskasutada ning sellel on virtuaalne DOM renderdus. Suur boonus on tugev Reacti kogukond, mis võimaldab probleemide korral kiirelt lahendust leida. Samuti ei mõjuta Reacti puhul *child*-komponendi muutmine *parent*-komponenti, mis paljude teiste raamistike puhul on tugevaks miinuseks. [28]

Reacti puuduseks võib nimetada dokumentatsiooni vähesuse, sest teek on pidevas muutuses. Pole mõistlik teegile täit dokumentatsiooni kirjutada, kui detailid kogu aeg muutuvad. Selle tõttu võib ka arenduse kiirus aeglustuda, sest arendaja peab protsesse uuesti omal käel õppima. [28]

Aurelia.js

Aurelia.js on JavaScripti raamistik, mis ei käitu nagu raamistik ning mis on mõeldud ühelehekülgelisi rakendusi looma (SPA – *Single Page Application*). Aurelia.js loomisel on kasutatud modernseid tööriistu nagu Node.js [29]. Aurelia on kergesti õpitav ja arusaadav tänu oma „konventsioon üle konfiguratsiooni" filosoofiale – tuleb lihtsalt järgida kindlaksmääratud mustreid, mille kaudu saab vähendada rakenduse ehitamiseks vajaminevat koodi [30].

Aurelia puhul on suurim puudus selle arenduskogukonna vähesus. Kui on vaja abi, siis seda on tõenäoliselt palju raskem leida kui teiste võrreldavate raamistike ja teekide puhul. Ka erinevaid mooduleid, mis näiteks toetavad teisi raamistikke (React, Angular jms) aga mitte Aureliat, on väga palju [31].

Kokkuvõte

Tabelis 2 on välja toodud kokkuvõte analüüsitud esirakenduse tehnoloogiate positiivsetest ja negatiivsetest külgedest.

Tabel 2. Esirakenduse tehnoloogiate võrdluse kokkuvõte

Tehnoloogia	Positiivne	Negatiivne
Vue.js	- Väike maht - Virtuaalne DOM renderdus - Komponentide taaskasutus - Suurem autori kogemus	- Vähe materjali ja dokumentatsiooni - Pluginate puudus
React.js	- Komponentide taaskasutus - Virtuaalne DOM renderdus - Suur Reacti kogukond - *Parent-* ja *child-* komponendi omavaheline sõltumatus	- Dokumentatsiooni vähesus - Vajalik protsesside uuesti õppimine - Vähene autori kogemus
Aurelia.js	- Kergesti õpitav - Kindlaksmääratud mustrid	- Arenduskogukonna vähesus - Puuduvad moodulid - Vähene autori kogemus

Põhiliseks lõputöö autori määravaks otsuse langetamise kriteeriumiks oli kogemus ning info ja dokumentatsiooni saadavust. React.js'i, Vue.js'i ja Aurelia puhul pole ühelgi neist väga head dokumentatsiooni saadavust. Valik põhines seetõttu ainult kogemusel – kuna autoril on suurim kogemus Vue.js'ga, otsustas ta seda esirakenduse loomiseks kasutada.

4.2.3 Andmebaasi valik

Andmebaas on koht rakenduses, kuhu kogutakse kõik oluline rakendusega seotud info. Hetkel eksisteerib turul üle 300 erineva andmebaasi juhtimissüsteemi. Andmebaasid jaotuvad enamjaolt kaheks: relatsioonilised ja mitte-relatsioonilised andmebaasid. Relatsioonilistel andmebaasidel on tabelite omavahelised suhted juba ette määratud ning need kasutavad päringute tegemiseks SQL-i (*Structured Query Language*). Mitte-

relatsioonilised andmebaasid on relatsioonilistele andmebaasidele alternatiiviks. Need pakuvad relatsiooniliste andmebaasidega võrreldes rohkem paindlikkust ja suuremat skaleeritavust. Autori kogemus piirdub vaid relatsiooniliste andmebaasidega, mistõttu võrreldakse just neid. Otsus langetatakse vastavalt rakenduse vajadusele ning autori kogemusele. Võrreldakse kolme populaarset relatsioonilist andmebaasi: PostgreSQL-i, MySQL-i ja SQLite'i. [32]

PostgreSQL

PostgreSQL on andmebaasisüsteem, mis toetab nii relatsioonilisi kui ka mitte-relatsioonilisi andmebaasipäringuid. See võimaldab teha komplekssseid analüütilisi päringuid. Skaleerimise võimalused teevad PostgreSQL andmebaasist hea tööriista uurimus- ja teaduslike projektide jaoks. Puudustest märkimisväärne on suur ressursitarbivus – PostgreSQL võib nõuda kuni 10MB ühe ühenduse pealt. [34]

MySQL

MySQL on üks populaarsemaid andmebaasisüsteeme. Skaleeritavus ja turvalisus on selle andmebaasi ühed võtmeomadused. Seda on kerge installida ning selle kogukond on palju suurem, kui võrrelda näiteks PostgreSQL-i ja SQLite'ga. MySQL võimaldab ühele andmebaasile ligi pääseda mitmel erineval kasutajal, mis pole näiteks SQLite'i puhul võimalik. Mass INSERT laused võivad aga vähendada jõudlust. Samaaegne lugemis-kirjutussagedus on samuti madal [34].

SQLite

SQLite on serverita andmebaas, mis on võimeline otse rakendusega integreeruma selle asemel, et andmebaasisüsteem kuskile installida ning selle kaudu rakendusega ühendus luua. SQLite'i on väga kerge seadistada ning konfigureerida. Kuid kuna andmebaas on serverita, pole sellele teistest masinatest ligipääsu, samuti ei sobi see kasutamiseks suuremate rakenduste puhul. [35]

Kokkuvõte

Tabelis 3 on välja toodud kokkuvõte analüüsitud andmebaasi tehnoloogiate positiivsetest ja negatiivsetest külgedest.

Tabel 3. Andmebaasi tehnoloogiate võrdluse kokkuvõte

Tehnoloogia	Positiivne	Negatiivne
PostgreSQL	- Komplekssed analüütilised päringud - Võimalus skaleerida	- Suur ressursitarbivus - Autori vähene kogemus
MySQL	- Turvalisus - Suur kogukond - Mitmete kasutajate ligipääs andmebaasile - Autori suurem kogemus	- Mass INSERT laused võtavad ressursse - Madal lugemis- ja kirjutussagedus
SQLite	- Integreerub otse rakendusega - Kerge seadistada ja konfigureerida - Autori suurem kogemus	- Serverita andmebaas – puudub ligipääs teistest masinatest

SQLite andmebaasi puhul võib tekitada probleeme serverita ühendus, sest andmebaasile on vaja ligi pääseda ka teistest masinatest. PostgreSQL on igati võimas andmebaas, kuid lõputöö käigus loodava väikese veebirakenduse puhul pole mõistlik nii suure jõudlusega andmebaasi kasutada, sest see sobib pigem teadustööde jaoks. Samuti on oluline kiirus ning potentsiaalne 10MB ühe ühenduse pealt ei garanteeri, et rakendus kiiresti toimiks. MySQL puhul on boonuseks suur kogukond ja selle turvalisus, samuti asjaolu, et kasutajaid võib olla mitu, kes andmebaasile ligi pääsevad. Selle tõttu otsustas autor kasutada MySQL andmebaasi.

4.3 Disain

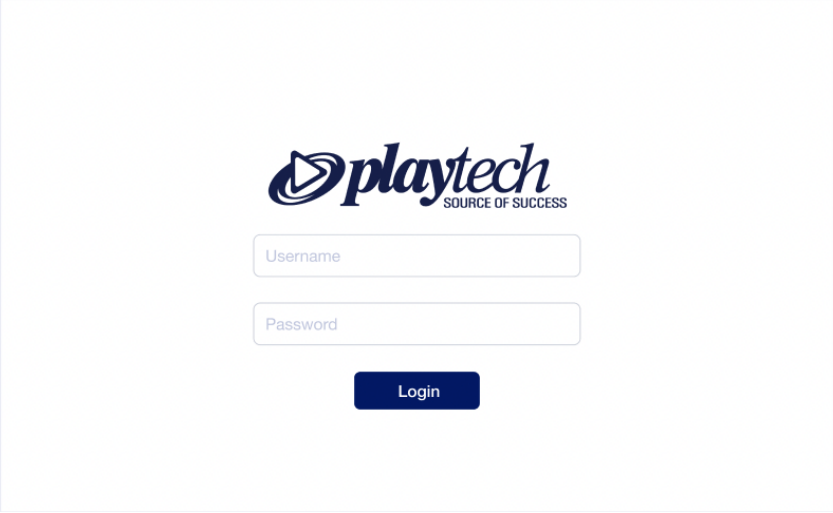
Veebirakenduse osa on selle väljanägemine ja disain. Veebirakenduse üks eesmärke oli parandada informatsiooni ülevaadet testimisaruandluses, mis tähendab, et oluline on pöörata tähelepanu kasutajakogemusele ning rakenduse visuaalsele poolele. Andmed peavad olema selgelt ja arusaadavalt esitatud.

Selleks, et paremini planeerida toote tulevast väljanägemist ja funktsionaalsust, on kasulik luua rakendusele prototüüp. Prototüüp aitab kaardistada nõudeid ning saada ka tagasisidet esialgse veebirakenduse disaini kohta. Autor on veebirakenduse prototüübi loomise käigus pärinud regulaarselt tagasisidet testijuhilt, kes eeldatavalt kõige rohkem seda rakendust kasutama hakkab.

Prototüüp loodi rakendusega nimega Figma. See on tasuta tööriist, mis võimaldab kasutajatel disainida ja prototüüpida kasutajaliideseid, veebilehti ja rakendusi. Kasutamise teeb mugavamaks erinevate väliste komponentide ja teekide olemasolu, mis võimaldavad kasutada juba valmis komponente (nt kalendrid, ikoonid, ajajooned jpm). See säästab oluliselt aega ning aitab paremini toodet visualiseerida.

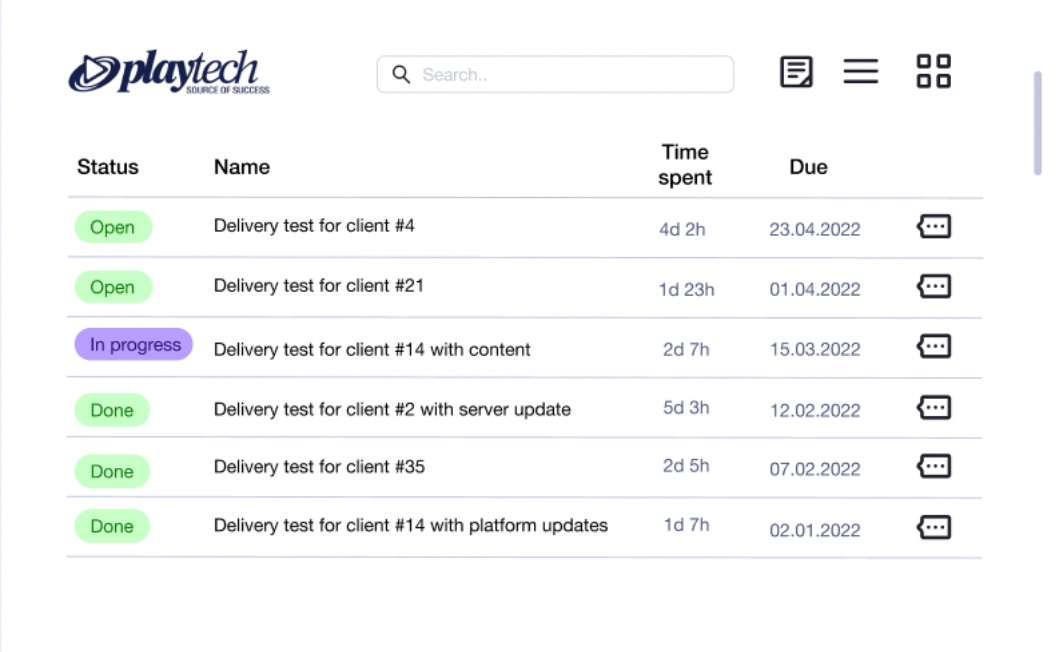
Järgnevalt on välja toodud prototüübi erinevad vaated, mis kasutajale kuvatakse.

Joonisel 1 on kuvatud sisselogimise vaade, kus kasutaja peab sisestama enda kasutajanime ja parooli, et veebirakendusse sisse pääseda.



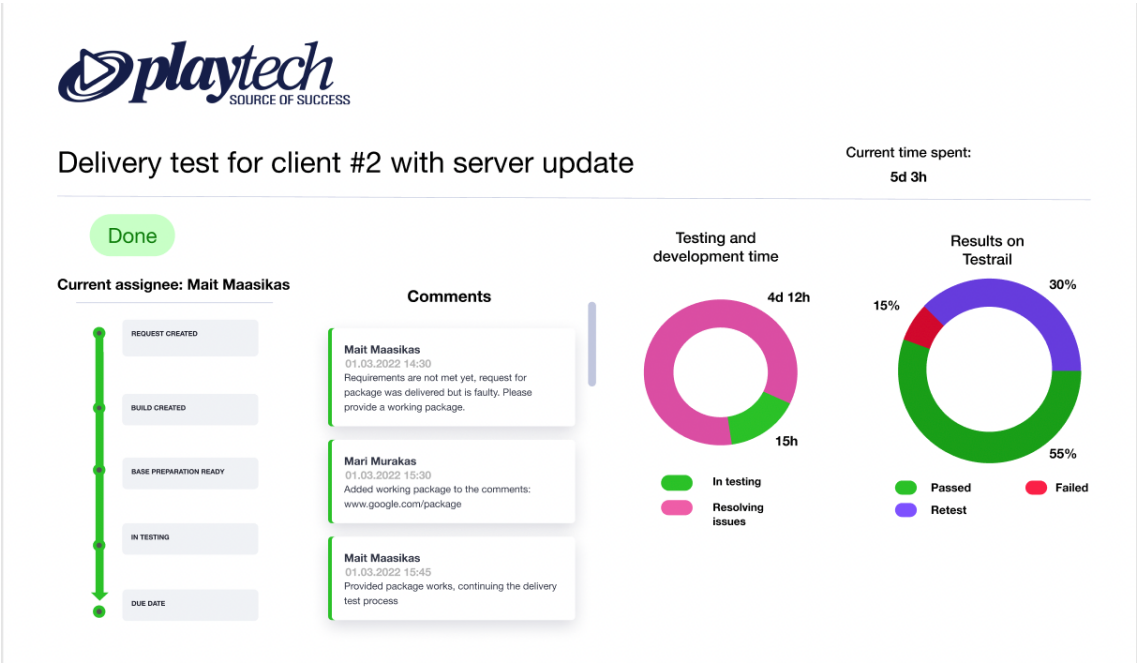
Joonis 1. Sisselogimise vaade

Joonisel 2 on näha avalehe vaadet. Seal on kirjeldatud kõik toote üleandmise instantsid ning sellega seotud andmed nagu staatus, toote üleandmise nimi ja sellega tegelemise aeg, tähtaeg ning võimalus toote üleandmist eraldi detailvaates vaadata.



Joonis 2. Avalehe vaade

Joonisel 3 on näha ühe toote üleandmise detailvaadet. Seal on kuvatud kommentaarid, ajajoon koos verstapostidega, TestRaili tulemused ning testimise ja arenduse aeg.



Joonis 3. Ühe toote üleandmise detailvaade

Muud vaated nagu summaarse kindla ajaperioodi üleandmiste aruande vaade ning selle päringu tegemise vaade ehk kuupäevalise otsingu vaade on leitavad **Lisa 2** juurest.

5 Rakenduse arendusprotsess

Selles peatükis kirjeldatakse rakenduse arendusprotsessi läbi andmebaasi ning esi- ja tagarakenduse arenduse. Tuuakse välja olulisemad veebirakenduse komponendid ja funktsionaalsused ning kuidas need toimivad.

5.1 Andmebaas

Andmebaasi loomisel juhinduti eelkõige asjaolust, et andmed peavad olema esitatud kompaktselt ning sisaldama olulisi näitajaid, mida kasutatakse statistiliste arvutuste tegemiseks. Kuna Jirast ja TestRaili API-st polnud mõistlik salvestada infot, mida rakenduse MVP kasutama ei hakka, siis tehti ainult vajalike andmete jaoks andmebaas. Kulukas on andmeid pidevalt API-delt pärida – see mõnevõrra aeglustab rakenduse tööd ning on ressursimahukas. Loodi ERD (*Entity Relationship Diagram*), mis illustreeriks andmebaasimudelit. Andmebaasiks valiti vastavalt analüüsile MySQL.

Loodud tabelid on järgmised:

- Issue – tabel, mis iseloomustab ühte *issue*'t Jiras. Iga jäädvustust Jiras nimetatakse Videobetis *issue*'ks, kaasa arvatud toote üleandmist. Seda tabelit võib nimetada ka „põhitabeliks", sest kõik ülejäänud tabelid (v.a Users), sõltuvad Issue tabelist.

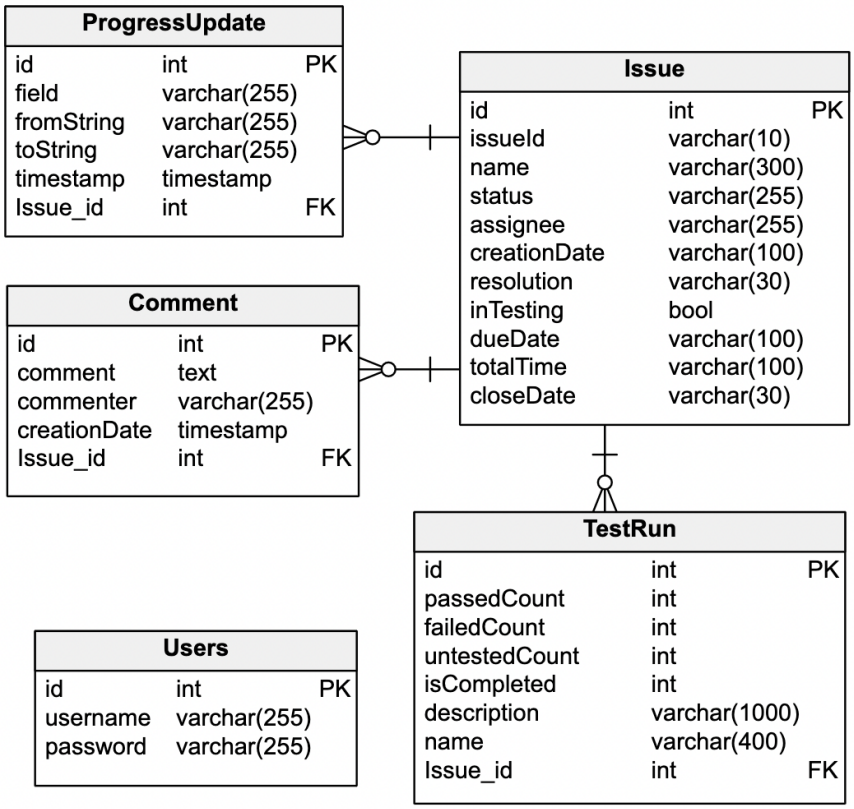
- TestRun – tabel, mis kirjeldab ühte TestRuni ehk testimissessiooni, mis TestRailis loodud. Selle tabeli kaudu peetakse arvestust, kuidas kindla toote üleandmise testimine läks.

- Comment – tabel, milles sisalduvad Jira *issue* kommentaarid. Seal leidub olulist infot, nagu näiteks millised probleemid esinesid, kuidas need parandati jms.

- ProgressUpdate – tabel, mis iseloomustab iga *issue* juures tehtud staatuse ja *issue*-ga tegeleva isiku muudatusi.

- Users – kasutajate tabel, mida kasutatakse kasutajate autentimise eesmärgil ning mis pole ühendatud ühegi teise tabeliga.

Mudel on nähtav jooniselt 4.



Joonis 4. Andmebaasi mudel

Kuna andmeid ei ole võimalik rakendusse manuaalselt sisestada vaid need pärinevad Jirast ja TestRaili API-lt, on vajalik andmebaasi tabelite regulaarne uuendamine eelmainitud TestRaili API ja Jira teegi põhjal. Tabelid Issue, ProgressUpdate, Comment ja TestRun peavad kuvama värskeimaid andmeid. Sellest kirjutatakse täpsemalt peatükis **5.2.7 Andmete sisestamine ja uuendamine**.

5.2 Tagarakenduse arendus

Tagarakendus arendati Springi abil ning kasutati Java programmeerimiskeelt. Tagarakenduse eesmärk on suhelda andmebaasiga ning sealt saadud andmeid töödelda. Selle arendus toimus pärast andmebaasi loomist ning kasutusel on REST struktuur. Rakendus loodi Intellij IDEA arenduskeskkonnas.

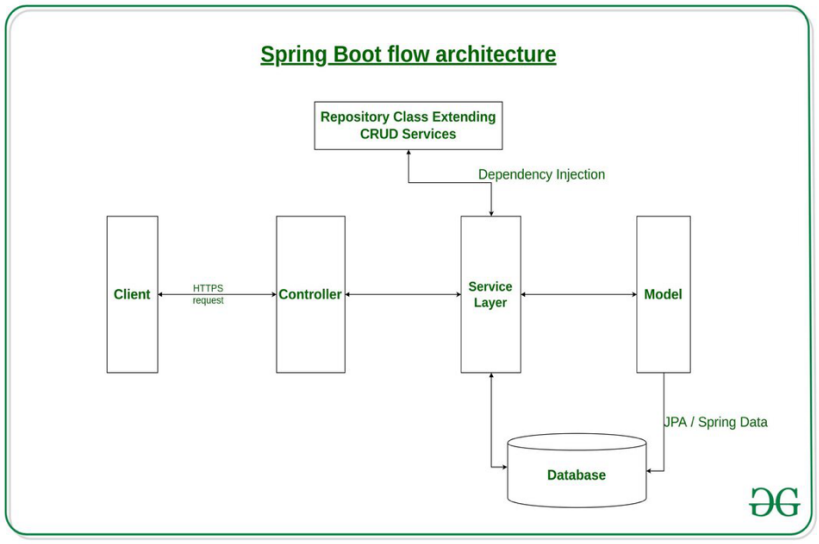
5.2.1 Tagarakenduse struktuur

Tagarakendus kasutab Spring Data JPA-d, mis hõlbustab JPA-põhiste hoidlate rakendamist. JPA-d (*Java Persistence API*) üleüldiselt kasutatakse andmete säilitamiseks Java objektide ja relatsiooniliste andmebaaside vahel [17].

Tagarakendus kasutab ka Spring Booti, mida kasutatakse REST API kontrollerite arenduses. Kontrolleritest lähemalt peatükis **5.2.5 REST API kontrollerid**.

Spring Booti tagarakendus toimib, nagu paljudki teised tagarakenduse struktuurid, üleüldise ringlusena. Struktuur koosneb mitmetest kihtidest, mis moodustavad ühtse terviku.

Joonisel 5 on välja toodud Spring Booti arhitektuur, mis koosneb kliendist, kontrollerist, teenuskihist ja mudelist.



Joonis 5. Spring Booti toimimise arhitektuur [15]

Klient (*Client*) suhtleb HTTPS päringu kaudu rakendusega (olgu nendeks PUT, GET, POST või DELETE päringud). Need päringud edastatakse kontrollerile (*Controller*), mis ühendub teenuskihiga (*Service Layer*). Teenuskiht hoiab endas rakenduse äriloogikat, juhib transaktsioone ja koordineerib enda teenuste rakendamise vastuseid [16]. Teenuskiht teostab kontrollerilt saadud info põhjal äriloogikaga määratud tegevused. Seejärel andmed kaardistatakse JPA kaudu mudeliks (*Model*). JSP (*Java Server Pages*) leht tagastatakse vastuseks kontrolleri poolt [15].

Spring Booti rakendus koosneb kokku neljast kihist: esitluskiht, ärikiht, andmete hoiustamise kiht ja andmebaasi kiht. Igal kihil on kindel roll, mida täita [15].

Andmebaasi kiht

Andmebaasi kiht, nagu nimigi viitab, on rakenduse andmebaasi osa. Seal on kõik rakenduse jaoks vajalikud andmebaasid. Käesoleva lõputöö raames on selleks andmebaasiks ainult üks MySQL-i andmebaas. Samuti tehakse seal CRUD (akronüüm *Create, Read, Update, Delete*) toiminguid [15].

Andmete hoiustamise kiht

Andmete hoiustamise kihis on kõik vajalik loogika, et võimaldada andmete andmebaasi hoiustamist. See vastutab äriobjektide muundamise eest andmebaasi ridadeks ning vastupidi [15]. Selle kihi alla kuuluvad peamiselt *repository*'d ehk hoidlad. Nende eesmärk on vähendada korduvat koodi. Käesolevas veebirakenduses on iga hoidla laiendatud JpaRepository liidesega. JpaRepository's sisaldub API nii CRUD toimingute tegemiseks kui ka sorteerimiseks.

Klassid, mis esindavad andmebaasi tabeleid, märgitakse rakenduses @Entity annotatsiooniga ning iga olemi instants esindab ühte rida selles andmebaasis. Antud veebirakenduses on kasutuses vastavalt viis klassi: User, Comment, TestRun, Issue ja ProgressUpdate.

Ärikiht

Ärikiht on koht, milles sisaldub programmi äriloogika. Äriloogika kiht antud veebirakenduse puhul on vajalik, sest päritavate andmetega on vaja teha arvutusi. Arvutused on omakorda vajalikud, et rakendus saaks kuvada testimise ja arenduse ajanäitajaid. Ärikiht on rakendatud vaid Issue puhul, sest teistel olemitel pole tarvis äriloogikat kasutada ehk midagi välja arvutada.

Esitluskiht

Esitluskiht on rakenduse pealmine osa ning koosneb esirakendusest. See käsitleb HTTP päringuid ning viib läbi autentimist. Samuti vastutab see JSON väljade Java objektideks teisendamise eest [15].

5.2.2 TestRaili API

Tabelisse TestRun lisati andmed TestRaili API kaudu. TestRaili API võimaldab andmeid TestRailist mugavalt pärida. Selleks, et TestRaili API-t kasutama hakata, tuleb läbida seadistusetapp, mis hõlmab projekti kausta API lisafailide paigaldamist [18].

TestRailist päringu tegemiseks peab olemas olema TestRaili URL, kuhu päringud suunatakse, ning kasutajanimi ja parool. Autor kasutas TestRaili API sisseehitatud meetodit „sendGet“, mis tahab sisendiks saada päringu aadressi, milleks antud veebirakenduse puhul oli „get_plans“. See pärib TestRaili toote üleandmise testimisplaane. Sinna juurde võib lisada lisaparameetreid, nagu näiteks plaani id, mis tagastab kõik plaanis sisalduvad TestRaili testimissessioonid.

Joonisel 6 on välja toodud näide TestRaili API päringust.

```
APIClient client = new APIClient(“http:///testrail/”);
client.setUser(“..”);
client.setPassword(“..”);
JSONObject c = (JSONObject) client.sendGet(“get_plans/1”);
System.out.println(c.get(“title”));
```

Joonis 6. TestRaili API päring [18]

Päring tagastab JSONObjecti või JSONArray. Objekti väljadele pääseb ligi „get“ meetodiga. Lõputöös kasutati ObjectMapperit, et JSONObjectist TestRuni objekti instants luua.

5.2.3 Jira REST Java Client teek

Jirast andmete saamiseks kasutas lõputöö autor Jira REST Java Client teeki. See võimaldab REST API kaudu suhelda iga Jira serveriga ning on välja arendatud Atlassiani poolt (Jira keskkonna looja). Teegiga tulevad kaasa erinevad domeeni objekti mudelid, mis tähistavad päringuga päritavaid andmeid nagu *issue*’d, kasutajad, staatused jpm. [19]

Selleks, et suhelda Jira teegiga, tuleb kõigepealt refereerida com.atlassian.jira.rest.client.JiraRestClient objekti, mis lubab seejärel hankida täpsemad viited *issue’*dele, projektidele, *issue* metaandmetele jne [20].

Kui see samm on tehtud, on võimalik Jira serveriga ühendust saada. Nagu TestRaili puhulgi, tuleb ka Jirale anda URL, kasutajanimi ja parool. Teegiga kaasa tulnud meetodi

„getIssue“ tagastatav väärtus on Issue objekt. „getIssue“ parameetriks tuleb anda *issue* id, mida soovitakse pärida.

Issue objektil on sisseehitatud meetodid, mida kasutati andmete saamiseks andmebaasi sisestamise eesmärgil. Selleks, et saada staatuse uuenduste kohta infot, mida sisestada ProgressUpdate tabelisse, oli tarvis lisada igasse „getIssue“ päringusse kaasa *changelog* parameeter. Sellega antakse päritava *issue*'ga kaasa kõikide muudatuste ajalugu.

Järgnevalt on välja toodud meetodid, mis kutsuti välja kindlate andmete saamise eesmärgil.

- Issue.getChangeLog() – tagastatakse kõik muudatused, mis ühe *issue* puhul tehtud on (muudatused staatuses, kuupäevas, pealkirjas jms). Muudatused on ChangeLogGroup objekti tüüpi – selleks, et saada detailsemat infot, on tarvis omakorda teine meetod välja kutsuda – getItems(). Info salvestatakse ProgressUpdate tabelisse andmebaasis.
    - Item.getField() – tagastab välja nime, mida on muudetud.
    - Item.getFromString() – tagastab esialgse välja väärtuse String kujul.
    - Item.getToString() – tagastab muudetud välja väärtuse String kujul.
    - ChangeLogGroup.getCreated() – tagastab muutuse tegemise aja.
- Issue.getComments() – tagastab kõik *issue* kommentaarid. Salvestatakse Comment tabelisse andmebaasis.
    - Comment.getBody() – tagastab kommentaari teksti.
    - Comment.getAuthor().getDisplayName() – tagastab kommenteerija täisnime.
    - Comment.getCreationDate() – tagastab kommentaari kuupäeva.
- Issue puhul päriti järgmiseid andmeid:
    - Issue.getStatus().getName() – tagastab *issue* staatuse.

- Issue.getAssignee().getDisplayName() – tagastab hetkel *issue* kallal töötava inimese täisnime.
- Issue.getCreationDate() – tagastab *issue* loomiskuupäeva.
- Issue.getResolution().getName() – tagastab *issue* tulemuse.
- Issue.getDueDate() – tagastab *issue* tähtaja.

5.2.4 Ajalised arvutused

Põhilised arvutused, mis veebirakenduses tehakse, on seotud ajaga. Testimise ja arendamise aeg on veebirakenduse tulevastele kasutajatele olulised näitajad. Veebirakendus arvutab need välja, kasutades ProgressUpdate tabelist pärinevaid andmeid. ProgressUpdate tabelis on tulp nimega „field“, mis märgib ära, millist Jira *issue* lahtrit on muudetud. Tabelisse on koondatud kahe lahtri andmed – „Assignee“ ja „Status“. Aja arvutamiseks vaadeldakse iga üleandmise puhul tabeli ridu, millel „field“ tulba väärtus on „Status“.

Vastavalt sellele, kui kaua on *issue* olnud kindlas staatuses, saab välja arvutada testimis- ja arendusaja. ProgressUpdate real on näidatud lähtestaatus ja tulemstaatus ning ajatempel, millal see muutus tehti. Seega on tarvis võrrelda iga käesolevat muudatust järgneva sissekande muudatuse ajaga ning arvutada vahepealne aeg. Näiteks *issue* staatus muudeti 02.09.2021 kell 15:00 „Open“ pealt „In Progress“ peale. Järgmine muudatus, mis tehti, oli 02.09.2021 kell 17:00 „In Progress“ pealt „Done“ peale. Muudatuste pealt on võimalik välja lugeda, et *issue* oli „In Progress“ staatuses kaks tundi, sest kell 15:00 pandi staatus „In Progress“ peale ning 17:00 muudeti see ära. Selliste arvutuskäikudega oli võimalik välja arvutada testimise ja arendamise aeg, kui testimisaja puhul jälgida staatust „In Testing “ ja arendusaja puhul jälgida staatust „In Progress“.

5.2.5 REST API kontrollerid

REST API ehk *Representational State Transfer* API on rakenduse programmi liides, mis jälgib REST-i arhitektuurilist stiili. Kui tehakse REST API kaudu päring, siis päringu tegijale või lõpp-punktile edastatakse ressursi oleku representatsioon. See representatsioon edastatakse käesoleva lõputöö puhul JSON-ina, teised variandid oleksid olnud HTML-i, PHP või tavatekstina. [36]

REST API kontrolleri klassid veebirakenduses on märgitud @RestController annotatsiooniga, mis on @Controller (ütleb Spring Bootile, et antud klass on kontroller) ja @ResponseBody (ütleb, et kontrolleri meetodid tagastavad vastuse REST API jaoks) kooslus [34] . Samuti on iga kontrolleri klassi kohal ka annotatsioon @RequestMapping, mis määrab aadressi, millelt kontroller oma sisendi saab.

Spring Booti kontrolleris võib olla mitu GET, POST jms meetodit. Tähtis on see, et need oleksid erinevatele aadressidele suunatud ning ei kattuks. Meetodid on märgitud @GetMapping või @PostMapping annotatsiooniga. Annotatsiooni järel on kuvatud aadressi lisa, kust pärides see meetod tööle läheb.

Järgnevalt on välja toodud aadressid, kuhu saab veebirakenduse REST API-s pöörduda ning mis sealt tagastatakse. API-d algavad aadressiga */api/*. Kokku on loodud neli kontrollerit: IssueController, SummaryController, CommentController ja AuthController. Kontrollerite aadressid on nähtavad tabelist 4.

Tabel 4. Kontrollerite aadressid

Kontroller	Päringu tüüp	Päringu aadress	Tegevus
IssueController	GET	„/api/dashboard/“	Kuvab kõik üleandmised.
IssueController	GET	„/api/dashboard/{id}“	Kuvab etteantud id põhjal ühe üleandmise.
IssueController	GET	„/api/dashboard/date“	Tagastab üleandmised, mis langevad kindlasse ajavahemikku.
IssueController	GET	„/api/dashboard/update“	Algatab andmebaasi uuendamise Jira teegi ja TestRaili API kaudu.
IssueController	GET	„/api/dashboard/{id}/data“	Tagastab IssueData tüüpi objekti ühe üleandmise kohta. Selles objektis on välja toodud teenuskihis

			toimunud arvutuste ja andmebaasipäringute tulemused.
SummaryController	GET	„/api/dashboard/summary/data“	Tagastab SummaryData tüüpi objekti mitmete kindla ajavahemiku üleandmiste kohta. Selles objektis on välja toodud teenuskihis toimunud arvutuste ja andmebaasipäringute keskmised tulemused nende üleandmiste kohta.
CommentController	GET	„/api/comments/{id}“	Tagastab kõik ühe Issue kommentaarid.
AuthController	POST	„/api/auth/signin“	Sisendiks võetakse LoginRequest objekt, mis koosneb kasutajanimest ja paroolist ning vastuseks seatakse brauserisse juurdepääsuloaga küpsis.
AuthController	POST	„/api/auth/signout“	Logib välja, küpsis eemaldatakse.

5.2.6 Kasutaja autentimine

Veebirakendusse on võimalik sisse logida. Sisselogimise lehte näevad kõik töötajad. Videobetis pääseb firmasisestele rakendustele ligi vaid firma sisevõrgus olles ning kasutades igale töötajale määratud isiklikku kasutajanime ja parooli. Seetõttu puudub rakendusel registreerumise võimalus.

Turvalisus on suurem selle võrra, et kõrvalised inimesed ei pääse firma sisevõrku sisse, mistõttu väheneb ka kurja tegemise võimalus, kuigi see pole välistatud. Sisselogimisel on kasutatud JWT (*JSON Web Token*) tehnoloogiat ning Spring Security't.

Töötajate andmed (kasutajanimed, paroolid) ja firma autentimistehnoloogiad pole autorile ligipääsetavad, seepärast pole võimalik firma enda andmebaase või tehnoloogiaid veebirakenduses kasutajate autentimiseks kasutada. Potentsiaalne võimalus oleks võimaldada autentimist Jira kaudu, kuid see eeldab veebirakenduse hindamist ja pikka bürokraatilist protsessi Jiraga integreerimiseks. Alles seejärel on võimalik firmapoolsetes serverites Jira kaudu sisselogimise võimaldamine. See protsess lõputöö ajaskoopi ei mahu. Jira ja TestRaili andmetele ligipääsemiseks on aga vajalik Jira konto kasutajanimi ja parool. MVP loomiseks mõeldi välja lühiajaline autentimise lahendus.

Kasutajate andmeid hoitakse tabelis Users. Päringut Jira teegile tehes tagastatakse vastavalt päritud andmed või vigaste andmete puhul (vale kasutajanimi ja/või parool) veasõnum. Esmakordse sisselogimiskatse puhul kontrollitakse Users tabelist, kas selline kasutaja on seal juba olemas. Kui ei, siis saadetakse testpäring Jira teegile. Kui päring on edukas ehk selline kasutaja Jiras eksisteerib, siis lisatakse see kasutaja Users tabelisse. Kui mitte, tagastatakse veasõnum. Edaspidiste sisselogimiskatsetega toimub autentimine Users tabeli kaudu.

Kui autentimine oli edukas, luuakse JWT sõne, mis lisatakse küpsisesse. JWT on kompaktne viis turvaliselt informatsiooni osapoolte vahel JSON objektina saata. Kui kasutaja on sisse logitud, siis pannakse JWT iga päringuga kaasa, et kasutaja pääseks ressurssidele ligi. [37]

5.2.7 Andmete sisestamine ja uuendamine

Veebirakenduse efektiivsus sõltub andmete ajakohasusest, sest üleandmistega seonduv info on pidevas uuenemises ning enamasti tegeletakse nendega iga päev. Tähtaja lähenedes muutub üleandmise infohulk suuremaks, sest lisatakse kommentaare, muudetakse üleandmise sisu ning testitakse üleandmise paketti. Nagu ka varasemalt mainitud, on oluline, et veebirakenduse andmed oleksid kõige uuemad ning ajakohasemad. Kuna üleandmistega seonduv info on sisestatud andmebaasi ning neid ei küsita otse Jirast ja TestRailist, on neid tarvis uuendada.

Andmete sisestamiseks ja uuendamiseks kasutati PreparedStatementi klassi, mis lubab kirjutada erinevaid parameetreid sisendiks võtvaid SQL päringuid. PreparedStatementi kasutamise eelised on suurem kiirus päringute teostamisel, dünaamiline andmete sisestus (ei pea pidevalt päringut ennast muutma, vaid hoopis parameetreid) ning see kaitseb SQL süstimise rünnakute eest. [41]

Selleks, et andmete sisestamine toimuks efektiivsemalt, kasutab veebirakendus *batching*'ut. *Batching* võimaldab andmebaasipäringuid kokku koondada ning seeläbi vähendada pidevat üksiku andmebaasiühenduse loomist iga päringu tarbeks. *Batch*'ides andmete sisestamine suurendab andmete sisestamise jõudlust ning on kiirem, kui iga sisestatava andmebaasirea kohta eraldi päringu tegemine. [42]

Põhilise päringuna kasutatakse INSERT lauseid. Andmete sisestamisel ja uuendamisel on vajalik, et ei sisestataks juba olemasolevate ridade koopiaid. Tuleb kontrollida, kas sisestatav üleandmine on juba andmebaasis olemas: kui jah, siis tuleb andmeid uuendada. Kui üleandmist andmebaasis pole, tuleb see üleandmine andmebaasi sisestada. Sellise päringu tegemiseks on olemas INSERT päringu laiendus INSERT...ON DUPLICATE KEY UPDATE. Kõigepealt proovib päring uut rida andmebaasi sisestada. Iga kord, kui päring leiab duplikaadist ainulaadse võtme või välja, siis INSERT asemel tehakse UPDATE päring ehk andmed uuendatakse. Järgnevalt on joonisel 7 välja toodud ühe päringu näide, mis tehti andmete sisestamiseks Comment tabelisse.

```
String queryComment = "insert into COMMENT(jiraId, comment,
commenter, creationDate) values (?, ?, ?, ?) ON DUPLICATE
KEY UPDATE jiraId=values(jiraId), comment=values(comment),
commenter=values(commenter),
creationDate=values(creationDate)";
```

Joonis 7. Comment tabeli INSERT...ON DUPLICATE KEY UPDATE päringu näide

Testijuhiga arutati, et andmeid tuleks vähemalt kord päevas uuendada, võimalusel öösel, kui keegi rakendust ei kasuta. Kuna andmeid päritakse API-lt ning neile pääseb ligi vaid kasutajanime ja parooliga, ei ole võimalik garanteerida automaatseks uuenduseks vajalikud tingimused. Lepiti kokku, et lisatakse nupp, mida vajutades uuendatakse pea kõiki andmebaasi tabelite andmeid ning mida saab iga rakendust kasutav töötaja vajutada.

5.3 Esirakenduse arendus

Esirakendus on rakenduse osa, millega kasutajad suhelda saavad. Veebirakenduse arenduse käigus valmis MVP, mille esirakenduse arendamiseks kasutati Vue.js raamistikku. Selleks, et kasutajale infot kuvada, oli vaja kasutada HTML-i ning andmete ilustamiseks CSS-i. Arendus toimus pärast tagarakenduse esialgse versiooni valmimist. Tänu esirakenduse arendusele oli võimalik täiendada ka tagarakendust. Arenduskeskkonnaks valiti varasema kogemuse tõttu Visual Studio Code.

5.3.1 Esirakenduse komponendid

Vue.js rakendus on tüüpiliselt struktureeritud taaskasutatavateks komponentideks. Iga komponent peab olema registreeritud, et seda oleks võimalik kasutada. Komponent, mida kasutatakse rakenduse „stardipunktina“, renderdatakse siis, kui see DOM-i külge pannakse [39]. Vue.js rakenduses tuleb kõigepealt luua uus Vue objekt, kuhu saab kaasa anda erinevaid rakenduse poolt kasutatavaid teeke. Seejärel paigaldatakse see komponent rakenduse külge [40].

Esirakendus koosneb mitmest osast, mis on jaotatud funktsionaalsuse ja kasutatavuse põhjal kaustadesse.

Assets – selles kaustas hoitakse kõiki rakendusega seonduvaid pilte, näiteks Playtechi logo.

Components – sisaldab erinevaid UI komponente, mida saab rakenduses taaskasutada, nagu näiteks üleandmiste tabel.

Models – selles kaustas hoitakse kasutatavate objektide klasse, nt User.

Services – teenuskihis paiknevad kõik tagarakenduse API lõpp-punktidele päringuid tegevad meetodid. Päringute tegemiseks kasutatakse Axiost (vt **5.3.1.1 Axios**).

Views – veebirakenduse vaadete kiht. Sinna on koondatud kõik veebirakenduse „leheküljed“. Nende hulka kuuluvad Dashboard, Summary, Issue, Calendar ja Login leheküljed.

5.3.1.1 Axios

Axios on HTTP kliendi teek, mis lihtsustab API päringute tegemist. Vue'l puudub korralik sisseehitatud HTTP teek, mistõttu soovitatakse kasutada Axios teeki, et API-delt informatsiooni saada [38]. Selleks, et Axiost kasutada, tuleb see kõigepealt installida ning seejärel enda projektis vajalikesse failidesse importida. Joonisel 8 on välja toodud näide Axiose kasutamisest käesoleva veebirakenduse testversioonis *dashboard/* aadressilt kõiki Issue'sid pärides. Axiost kasutatakse ainult teenuskihis põhiliselt „get" tüüpi päringute tegemiseks, sest veebirakendusel puudub nõue kasutaja poolt sisestatud andmeid salvestada.

```
import axios from „axios";
const API_URL = http://localhost:8080/api/

class IssueService {
    getAll() {
      return axios.get(API_URL + „dashboard/", {headers:
authHeader()})
    }
}
```

Joonis 8. Axios päringu näide

Iga Axiose päringuga antakse kaasa ka päis, mis sisaldab endas kasutaja juurdepääsumärgendit (kujul „Bearer " + JWT). Ilma selleta poleks võimalik päringuid kaitstud API-dele teha ning tagastatakse veateade, et kasutajal pole õigust andmetele ligi pääseda.

5.3.1.2 *Router*

Router võimaldab kasutajal rakenduse vaadete vahel navigeerida ning sealjuures tagada kiire toimimine (pole tarvis iga vaate puhul kõike nullist laadida) [44]. Kokku on veebirakendusel viis vaadet: Dashboard, Issue, Summary, Calendar ja Login. Autor on välja toonud näite joonisel 9, kuidas *router* luuakse ning sinna veebiaadress lisatakse.

```
export const router = new Router({
    mode: 'history',
    routes: [
      {
         path: '/login',
         name: 'Login',
         component: Login
     ]
})
```

Joonis 9. *Router*'i loomine ja sisselogimise aadressi määramine

Router'ile antakse kaasa komponent, käesoleval juhul sisselogimise lehe komponent, ning see kuvatakse siis, kui mööda *router*'it on */login* aadressile liigutud kas nupuvajutuse või otse lingi kaudu. Kõik eelmainitud viis vaadet on *router*'isse sarnaselt loodud.

5.3.2 Kasutatud teegid

Kliendipoolne rakendus kasutab oma töös erinevaid lisateeke, mis hõlbustavad andmete ja komponentide kuvamist ilma, et Vue.js lehed sisaldaksid suures mahus CSS koodi. Tänu nendele muutub kood loetavamaks. Samuti võimaldavad need juurde lisada funktsionaalsust.

Vuetify

Vuetify on Vue.js UI teek, mis on loonud erinevad valmistehtud komponendid laiapõhiseks kasutamiseks. Veebirakenduses kasutati tabelivaate loomisel Vuetify „v-data-table" komponenti, et loodav töölaua üleandmiste tabel oleks kergesti skaleeritav ning mahutaks andmed lahtritesse ära.

Vee-validate

Vee-validate on Vue.js teek, mis võimaldab kergesti kasutajatelt küsitavaid andmeid valideerida. Seda teeki kasutab sisselogimise leht, et valideerida kasutaja poolt sisestatud kasutajanime ja parooli.

Apexcharts

Apexcharts on graafiliste komponentide kogu, mida on võimalik andmete visualiseerimise eesmärgil rakendada. Veebirakendus kasutas seda sektordiagrammide

tegemiseks, millega illustreeriti testimis- ja arendusaega, TestRaili sessioonide tulemusi ja üleandmiste tähtajale vastavust.

jsPDF

jsPDF on lahendus, mille kaudu on võimalik luua PDF-formaadis dokumente ning neid brauserist alla laadida [43]. Veebirakendus kasutas jsPDF-i, et raporteid luua ja neid salvestada.

5.3.3 Ajajoon

Ajajoon on mõeldud üleandmiste staatuste paremaks illustreerimiseks. Ajajoone loomiseks on vajalik ajalised andmed andmebaasi talletada. Põhilised ajanäitajad pärinevad Issue tabelist, kus on märgitud creationDate, dueDate ja closeDate ehk *issue* loomishetk, üleandmise tähtaeg ja üleandmise tegelik valmissaamise aeg. Need andmed kantakse ajajoonele koos kuupäevadega. Vastavalt üleandmise staatustele märgitakse roheliselt (tehtud)/siniselt (aktiivne)/hallilt (tegemist ootav) iga verstapost, mis on vajalik üleandmise edukaks läbiviimiseks. Kokku on ajajoonel kuus verstaposti. Järgnevalt on välja toodud kasutatavad verstapostid.

Request Created - Jirasse luuakse üleandmise informatsiooniga *issue*. Selle andmed pärinevad Issue tabeli creationDate tulbast.

Package Creation - arendaja valmistab paketti. Sõltuv staatusest „In Progress“.

In Testing - paketti testitakse testija poolt. Sõltuv staatusest „In Testing“.

Waiting for Review - vastavalt testija kommentaaridele ootab pakett arendajapoolset kontrolli. Sõltuv staatusest „Unverified“ või „Test Completed“.

Closed – pakett on valmis kliendile minema ja *issue* suletakse. Selle andmed pärinevad Issue tabeli closeDate tulbast.

Due Date – paketi valmimise/väljasaatmise tähtaeg. Selle andmed pärinevad Issue tabeli dueDate tulbast.

Package Creation, In Testing ja Waiting for Review on kõik dünaamilised näitajad. Kuna need võivad aja jooksul muutuda ehk *issue* võib staatust korduvalt vahetada (nt paketis esines viga ja see saadeti arendajale tagasi), pole nende puhul kuvatud kuupäevi. Joonis,

milline näeb ajajoon esirakenduses välja, on näidatud peatükis **5.3.5 Esirakenduse vaated**.

5.3.4 Raporti loomine ja allalaadimine

Rakenduse üks funktsionaalseid nõudeid on pakkuda võimalust üleandmistestimise üksikust ja summaarsest vaatest raportit alla laadida. Mõistlik oli lahendust otsida kolmanda osapoole poolt loodud võimalustest, sest manuaalselt PDF-formaadis raportit luua on ajamahukas ja keeruline. Seetõttu otsustas autor raporti loomise funktsionaalsuse võimaldamiseks kasutada jsPDF lahendust, mis laseb rakendusel PDF dokumenti kujundada, andmetega täita ning alla laadida.

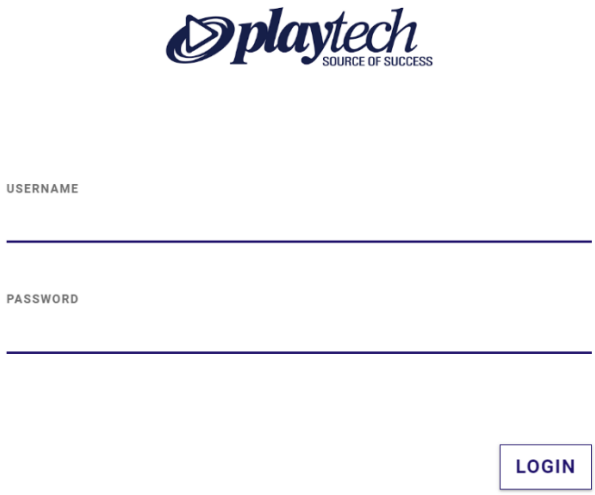
Ühe üleandmise raportil on kuvatud üks tabel ning kaks sektordiagrammi. Tabelis on märgitud kaks põhilist kuupäeva: *issue* loomishetk ja tähtaeg. Samuti on sinna märgitud töötaja nimi, kes hetkel üleandmisega tegeleb, üleandmise staatus, testmis-, arendus- ja üldine aeg ning TestRaili testide tulemused. Sektordiagrammid illustreerivad TestRaili testimise tulemusi ning testimis- ja arendusaja suhet.

Summaarse üleandmiste raportil pole kuvatud individuaalsete üleandmiste statistikat, vaid sellest on võetud keskmised tulemused. Nii on kuvatud keskmine testimis-, arendus- ja üleüldine aeg ning keskmised testide tulemused TestRailis. Samuti lisandub info, paljud ajavahemikku langevad üleandmised jõudsid õigeks ajaks valmis, paljud jäid hiljaks ning kui paljud on veel lõpetamata.

Kõik raportisse lisanduvad andmed on olemas vastavates vaadetes, kust on võimalik raportit genereerida. Raportit on võimalik genereerida, vajutades selleks ettenähtud nuppu. Raportite näited on välja toodud **Lisa 3** ja **Lisa 4** all.

5.3.5 Esirakenduse vaated

Kõikidele kasutajatele, kes pole ennast rakendusse sisse loginud, kuvatakse sisselogimise leht. Sinna on võimalik sisestada enda kasutajanimi ja parool. Joonisel 10 on see kuvatud.

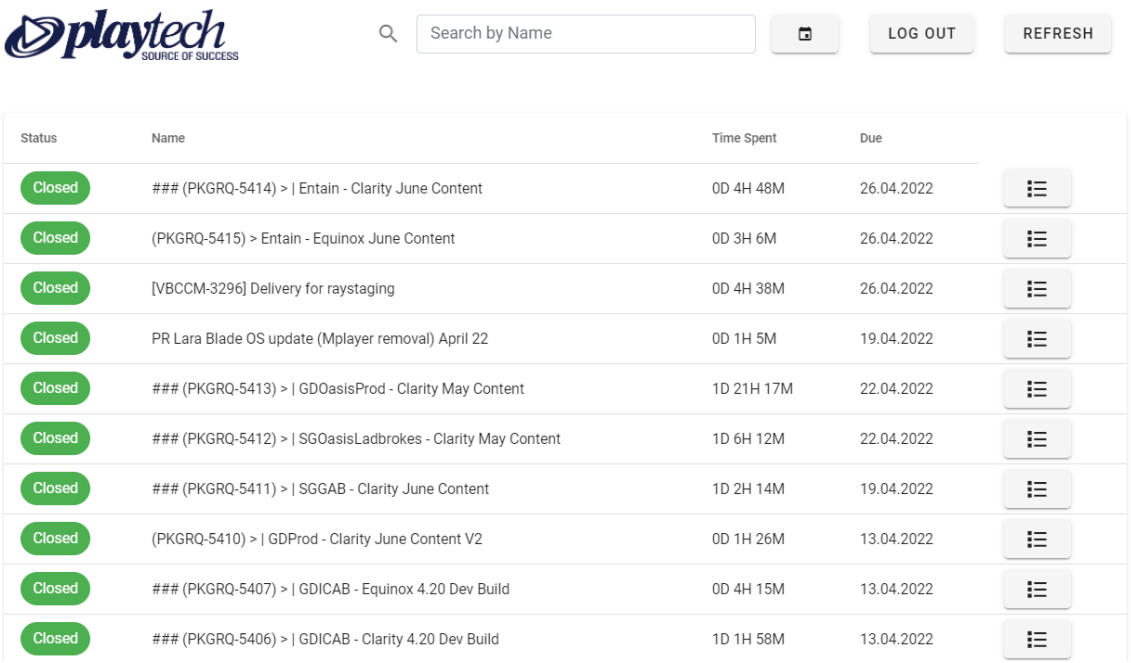


Joonis 10. Sisselogimise leht

Kui kasutaja sisselogimine ebaõnnestub, siis näidatakse kasutajale veateadet. Kui see õnnestub, siis suunatakse kasutaja edasi töölauale, mida illustreerib joonis 11.

Töölaua vaate ülemises osas on navigatsiooniriba. Vasakul navigatsiooniribas paikneb Playtechi logo, mis täidab ühtlasi ka töölaua vaate linki ehk sealt saab erinevatelt lehekülgedelt liikuda tagasi töölauale. Nime järgi on võimalik üleandmisi otsida. Nupp kalendriga võimaldab liikuda kalendri vaatesse, kust saab valida summaarse vaate jaoks välja kuupäevad (Joonis 13). Kalendri nupust paremal on väljalogimise nupp, mida vajutades eemaldatakse JWT ja autentimise info ning kasutaja suunatakse tagasi sisselogimise vaatesse. Navigatsiooniriba viimaseks parempoolseimaks komponendiks on värskendamise nupp, mis algatab andmebaasi uuendamise.

Iga toote üleandmine on kuvatud tabelis. Tabelis on andmed esitatud ridadena. Igas reas on nähtav üleandmise staatus, nimi, selle üleandmise peale kulunud aeg ja tähtaeg. Iga rea kõige parempoolsem komponent on nupp, mis viib selle üleandmise detailvaatesse. Andmeid on töölaua vaates võimalik sorteerida nime, kulutatud aja ja tähtaja põhjal kas kasvavas, kahanevas või normaalses ehk kronoloogilises järjekorras.



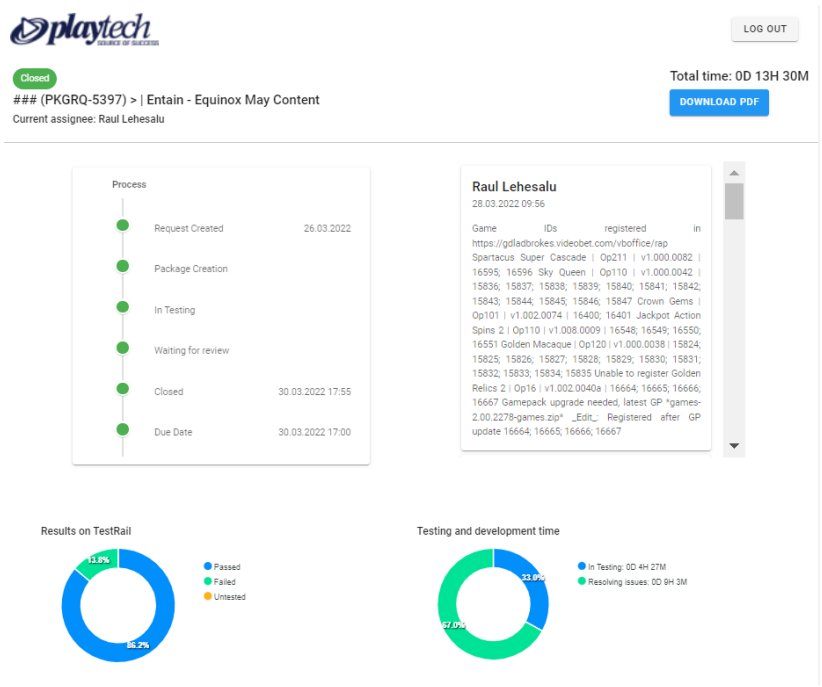
Joonis 11. Töölaua vaade

Joonisel 12 on kujutatud ühe toote üleandmise detailvaadet. Joonist on kohandatud, et kõik elemendid mahuksid korraga pildile. Vasakul ülal on märgitud üleandmise nimi, staatus ja inimene, kelle käes üleandmine hetkel on. Paremal ülal on märgitud kogu protsessile kulunud aeg, mis arvestatakse sellest hetkest, mil arendaja hakkab paketti ette valmistama ning lõppeb kõige viimase staatuse muudatusega, mis tehtud on. Samuti on seal nupp PDF-formaadis raporti allalaadimiseks.

Lehe võib jagada neljaks ruuduks, milles on eraldi informatsioon. Üleval vasakus ruudus on märgitud progressi kuus verstaposti: aeg, millal üleandmise taotlus loodi, paketi loomise verstapost, testimise verstapost, ülevaatuse verstapost, üleandmise valmimisaeg ja lõpptähtaeg. Vastavalt hetkelisele üleandmise staatusele, vahelduvad „Package Creation“, „In Testing“ ja „Waiting for Review“ värvid. Kui toode on läbinud edukalt mingi etapi, siis värvitakse pall roheliseks. Kui mitte (näiteks on testija avastanud vea ning saatnud paki tagasi arendajale), siis jääb pall halliks.

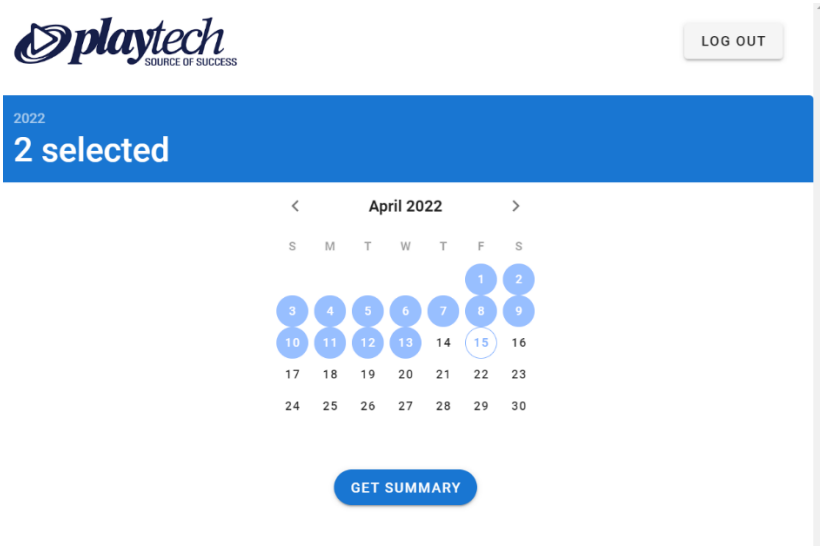
Üleval paremal on kommentaaride komponent, kus on kuvatud kommenteerija nimi, kommentaari aeg ja kommentaar ise. All vasakul näidatakse TestRaili testsessioonide tulemusi ehk kui paljud testidest õnnestusid ning kui paljud läbi kukkusid või kui paljud neist veel testimata on. All paremal on statistika, kui palju aega kulus testimisele ning kui palju aega kulus vigade parandamisele. Need ajad arvestatakse staatuste muudatuste järgi.

Kui näiteks testija saadab vea ilmnemisel paketi tagasi, siis muudetakse *issue* staatus „In testing“ pealt „In Progress“ peale ning arenduse aega hakatakse arvestama.



Joonis 12. Ühe üleandmise detailvaade

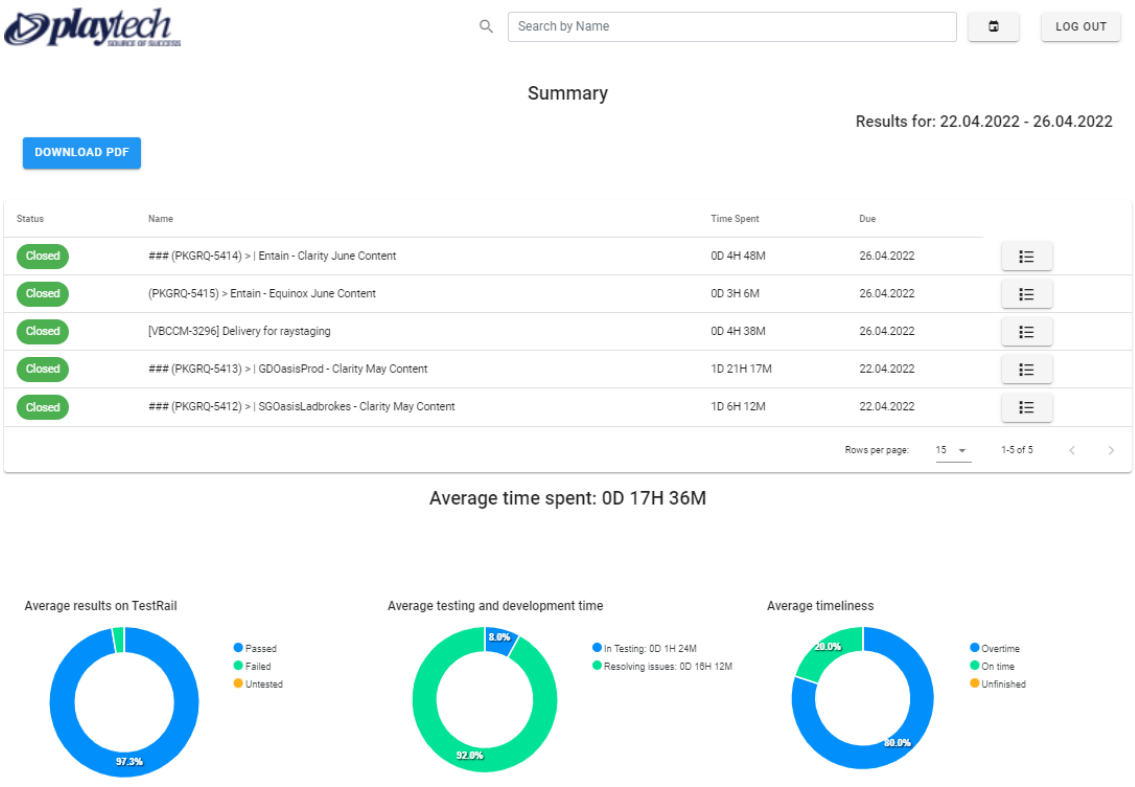
Joonisel 13 on näidatud kalendrivaade, kust saab valida ühe või kaks kuupäeva, mille vahemikku langevad üleandmised kuvatakse summaarsel lehel.



Joonis 13. Kuupäevade selekteerimise vaade

Joonisel 14 on kuvatud summaarne üleandmiste lehekülg, milleni jõuab pärast seda, kui ollakse välja valinud otsitavad kuupäevad. Navigatsiooniribas on jätkuvalt peamenüüsse viiv nupp ning võimalus otsida üleandmisi. Sellel leheküljel saab otsingumootori abil otsida nime järgi antud kuupäevade vahemikku kuuluvaid üleandmisi. Samuti on seal väljalogimise nupp ja PDF-formaadis raporti allalaadimise nupp.

Vaate allosas on kirjas keskmine aeg, mis üleandmistele kulus. See arvestatakse kõikide ajavahemikku kuuluvate üleandmistele kulunud aja põhjal. Lehel on samuti kolm sektordiagrammi. Vasakult esimene on keskmised testitulemused TestRailis, vasakult teine keskmine arenduse ja testimise aeg ning vasakult kolmas näitab üleandmiste tähtajast kinnipidamist ehk kui paljud üleandmised jõudsid õigeks ajaks valmis ning kui paljud mitte.



Joonis 14. Summaarne üleandmiste vaade

6 Tulemused

Käesoleva lõputöö käigus valmis veebirakenduse MVP, millega on võimalik saada ülevaade käimasolevatest ja juba toimunud üleandmistest. Rakendus täidab kõik firma poolt seatud funktsionaalsed ja mittefunktsionaalsed nõuded. Kuna rakendus kasutusel veel pole ning selle firmasse integreerimine võtab aega, on tulevikus eeldatavalt võimalik rakenduse kasulikkust mõõta läbi testimisele ja arendusele kulunud aja. Loomulikult ei ole aeg kõige kindlam tegur, millele toetuda, sest üleandmised võivad varieeruda ühest-kahest uuest integreeritavast mängust suure platvormi- või riistvarauuenduseni. Keskmist üleandmisele kulunud aega saab näiteks arvestada ühe kuu kaupa ning näha tolle kuu keskmist tulemust nii soorituslikus kui ka ajalises mõttes. Seda saab võrrelda nende kuudega, mil olid umbkaudselt samaväärse sisuga üleandmised. Nähes keskmist üleandmiste ajakulu ja tulemusi, on võimalik testijuhil langetada otsus, kas vahetada välja testija või hoopis anda testijale keegi abiks, kas eskaleerida mingi kindel üleandmine ning viia arendajate juhi tähelepanu sellele, et paketti valmistatakse liiga aeglaselt või kas mingi kindla kliendi üleandmised jäävad alati hiljaks ning peab uurima, mis selle põhjus on. Need kõik on otsused, mida veebirakenduse abiga saadakse teha, et üleandmiste loomise ja testimise protsess kulgeks kiiresti, efektiivselt ja sujuvalt.

6.1 Testimine

Veendumaks, et rakendus toimis ning ei esineks olulisi vigu, oli vaja rakendust testida. Eraldi automaatteste rakendusele ei kirjutatud, kuid rakendust kontrolliti manuaalselt. Põhiline testimise rõhk läks andmete ja arvutuste õigsusele. Manuaalselt on võimalik ajalised näitajad välja arvutada ning neid seejärel meetodite poolt välja arvutatud andmetega võrrelda. Ei piirdutud ühe üleandmise arvutustega, vaid mitmete üleandmiste omadega, et tagada arvutuste järjepidev õigsus. Samuti võeti rakenduse testimisel arvesse erinevaid stsenaariume, kus midagi võib halvasti minna. Näiteks testiti läbi stsenaariumid, kus testijad või arendajad polnud Jira staatuseid kokkulepitud normide kohaselt muutnud ning osa andmeid läks kaduma. Selle tulemusena kuvati rakenduses tühjasid HTML elemente ning veateateid. Tänu testimisele sai see olukord avastatud ja lahendatud.

6.2 Tagasiside

Esimest korda koguti tagasisidet, kui demonstreeriti rakenduse prototüüpi testijuhile, firma protsessijuhile, ühele projektijuhile ja vanemtestijale, kes tegeleb igapäevaselt üleandmistestidega. Korraldati koosolek ning autor näitas ja kirjeldas loodud prototüübi vaateid ning mis infot need edasi annavad. Igal osalejal oli võimalik küsida küsimusi ning teha ettepanekuid, mis info võiks veel sellest veebirakendusest välja loetav olla. Peamiseks elemendiks, mis võiks veebirakendusel veel olemas olla, oli pakutud arvnäitaja, mis väljendaks, mitu korda üleandmine kliendi poolt tagasi lükati. See tähendab seda, et kliendil tekkisid tootega probleemid ning need saadeti lahendamiseks firmasse tagasi. Oldi prototüübiga rahul ning leiti, et sellest saaks palju kasu kõigile eelmainitud koosolekul osalejatele ning ka muudele üleandmistega hõlmatud töötajatele.

Valmis MVP-d tutvustas autor testijuhile. Näidates talle rakendust ning tutvustades selle toimimispõhimõtteid, andis testijuht selle põhjal väärtuslikku tagasisidet, mida saaks juurde lisada ja muuta. Ta tõi välja, et kalendrivaates võiks olla mitu kalendrit kõrvuti, et oleks parem vaade kuudele. Samuti tegi ta ettepaneku lisada veel üks diagramm, mis väljendaks „põrkeid“ ehk kui palju kordi tootes vigu esines ja pakett tagasi arendajale saadeti. Kuna osa kommentaare on üsna pikad, siis pakkus testijuht ka välja, et kommentaarid võiksid olla laiendatavad, et ei peaks tervet pikka kommentaari kogu aeg nägema. Üldiselt oli ta rakendusega väga rahul ning arvas, et see on kasulik lisand, mis aitaks temal oluliselt üleandmistestimisest ülevaadet parandada.

6.3 Võimalikud edasiarendused

Autori plaaniks on veebirakenduse arendamisega jätkata ka edaspidi pärast lõputöö valmimist. MVP-d edasi arendades on veebirakendust võimalik esirakenduse poolelt täiendada, et rakenduse välimus vastaks rohkem prototüübile. Suurema tähtsuse omandas andmete korrektne ja selge kuvamine, mitte komponentide värvilahendus või nende detailne paiknemine. Samuti on võimalik parandada ajaarvestamist, et arendus- ja testiaja sisse ei arvestataks öötunde, vaid seda aega, mis arendajal ja testijal on märgitud igapäevaseks tööajaks. Tulevikus on ka plaan veebirakendus firma sisevõrku integreerida, et see kõikidele töötajatele kättesaadav oleks. Samuti rakendada uuendatud sisselogimist, mis on seotud Jiraga.

Pärast vestlust testijuhiga selgus, et käesolevale veebirakendusele soovitakse luua veel lisaks ühte veebirakendust. Tahetakse arendada teinegi töölaud, mis hõlbustaks projektide üleüldist testimisprotsessi. See koondaks kokku kõik testsessioonid ning toodete puhul avastatud vead, andes testijuhile ja testijale parema ülevaate iga projekti progressist.

7 Kokkuvõte

Bakalaureusetöö eesmärgiks oli Videobeti tarbeks luua veebirakendus. Selle eesmärk on üleandmistestide ülevaate parandamine, mis soosiks adekvaatsemate otsuste langetamist vastavalt esitatud andmetele ning seeläbi võimalike probleemide vältimist ja üleandmisele kuluva aja lühendamist.

Töö käigus analüüsiti erinevaid tehnoloogiaid, millega oleks kõige efektiivsemalt võimalik veebirakenduse MVP luua. Tagarakenduse tehnoloogiaks valiti Spring ja Java, esirakenduse raamistikuks Vue.js ning andmebaasiks MySQL. Luues rakendusest prototüübi, sai efektiivsemalt andmebaasi ning esi- ja tagarakenduse struktuuri planeerida. Nõudeid arvestades loodi vastav rakendus valitud vahenditega.

Veebirakendus pole lõputöö esitamise hetkeks rakenduse kasusaajatele kasutamiseks kättesaadav, sest rakenduse integreerimine firma sisevõrku on aeganõudev protsess ning turvalisuse huvides pole soovitatav rakendust sisevõrgust eemal üles panna. Kasusaajad on aga rakenduse prototüübi ja MVP-ga tutvunud ning andnud hinnangu, et see on kasulik lisand üleandmisprotsessi potentsiaalseks parandamiseks ja kiirendamiseks. Lõputöö autor loodab jätkata veebirakenduse arendamisega, muutes selle disaini prototüübile vastavamaks ning lisades soovituste kohaselt juurde erinevaid funktsionaalsusi ja diagramme. Samuti loodab ta veebirakenduse firma sisevõrku integreerida.

Kasutatud kirjandus

[1] “What is Jira used for?”, Atlassian [Võrgumaterjal]. Saadaval: https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for [13.03.2022]

[2] „Introduction to TestRail“, Gurock [Võrgumaterjal]. Saadaval: https://www.gurock.com/testrail/docs/user-guide/getting-started/walkthrough [22.03.2022]

[3] L. Fitzgibbons „front end and back end“ [Võrgumaterjal] Saadaval: https://whatis.techtarget.com/definition/front-end [24.03.2022]

[4] „The Python Tutorial“, Python [Võrgumaterjal] Saadaval: https://docs.python.org/3/tutorial/appetite.html [24.03.2022]

[5] K.Basel „Python Pros and Cons“ [Võrgumaterjal] Saadaval: https://www.netguru.com/blog/python-pros-and-cons [24.03.2022]

[6] „Python Advantages and Disadvantages – Step in the right direction“, TechVidvan [Võrgumaterjal] Saadaval: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/ [24.03.2022]

[7] „Disadvantages of Python“, GeeksForGeeks [Võrgumaterjal] Saadaval: https://www.geeksforgeeks.org/disadvantages-of-python/ [24.03.2022]

[8] M.Wasilewski „Mobile development“ [Võrgumaterjal] Saadaval: https://www.netguru.com/blog/python-vs-java-comparison [24.03.2022]

[9] „FAQ: General“, Django [Võrguteavik] Saadaval: https://docs.djangoproject.com/en/4.0/faq/general/#does-django-scale [24.03.2022]

[10] S. Bhatt „Pros and Cons of Django Framework for App development“ [Võrgumaterjal] Saadaval: https://dzone.com/articles/pros-and-cons-of-django-framework-for-app-developm [24.03.2022]

[11] „A tour of the C# language“, Microsoft [Võrgumaterjal] Saadaval: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [24.03.2022]

[12] „The Good and the Bad of C# Programming“, Altexoft [Võrgumaterjal] Saadaval: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/ [24.03.2022]

[13] „Pros and Cons of Using C# as Your Backend Programming Language“, Agilities [Võrgumaterjal] Saadaval: https://www.agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html [24.03.2022]

[14] „ASP.NET vs PHP“, Orient [Võrgumaterjal] Saadaval: https://www.orientsoftware.com/technologies/microsoft-net/aspnet-vs-php/ [24.03.2022]

[15] „Spring Boot - Architecture“, GeeksForGeeks [Võrgumaterjal] Saadaval: https://www.geeksforgeeks.org/spring-boot-architecture/ [25.03.2022]

[16] R.Stafford „Service Layer“ [Võrgumaterjal] Saadaval: https://martinfowler.com/eaaCatalog/serviceLayer.html [25.03.2022]

[17] „Java – JPA vs Hibernate“, GeeksForGeeks [Võrgumaterjal] Saadaval: https://www.geeksforgeeks.org/java-jpa-vs-hibernate/ [25.03.2022]

[18] „TestRail API manual“, Gurock [Võrgumaterjal] Saadaval: https://www.gurock.com/testrail/docs/api/getting-started/binding-java/ [25.03.2022]

[19] „JIRA REST Java Client Library“, Atlassian [Võrgumaterjal] Saadaval: https://ecosystem.atlassian.net/wiki/spaces/JRJC/overview?homepageId=27164679 [25.03.2022]

[20] W. Seliga, A. Mierzwicki „Tutorial“ [Võrgumaterjal] Saadaval: https://ecosystem.atlassian.net/wiki/spaces/JRJC/pages/27164680/Tutorial [26.03.2022]

[21] „Introduction to Java“, GeeksForGeeks [Võrgumaterjal] Saadaval: https://www.geeksforgeeks.org/introduction-to-java/ [26.03.2022]

[22] „The Good and the Bad of Java Programming“, Altexsoft [Võrgumaterjal] Saadaval: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/ [26.03.2022]

[23] „Spring Framework - Overview“, Tutorialspoint [Võrgumaterjal] Saadaval: https://www.tutorialspoint.com/spring/spring_overview.htm [26.03.2022]

[24] T. Watson „Java Spring framework – pros, cons, common mistakes“ [Võrgumaterjal] Saadaval: https://skywell.software/blog/java-spring-framework-pros-cons-mistakes/ [26.03.2022]

[25] „Frontend vs Backend“, GeeksForGeeks [Võrgumaterjal] Saadaval: https://www.geeksforgeeks.org/frontend-vs-backend/ [26.03.2022]

[26] „Front End (In a Website)“, Airfocus [Võrgumaterjal] Saadaval: https://airfocus.com/glossary/what-is-a-front-end/ [26.03.2022]

[27] „The Good and the Bad of Vue.js Framework Programming“, Altexsoft [Võrgumaterjal] Saadaval: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/ [26.03.2022]

[28] W. Baranowski „React JS: Advantages and Disadvantages“ [Võrgumaterjal] Saadaval: https://massivepixel.io/blog/react-advantages-disadvantages/ [26.03.2022]

[29] J. Duff, J. Brown „Aurelia: An Introduction“ [Võrgumaterjal] Saadaval: https://www.codemag.com/article/1607091/Aurelia-An-Introduction [26.03.2022]

[30] K.Kataria „Aurelia vs Angular(2+): Why I prefer Aurelia over Angular“ [Võrgumaterjal] Saadaval: https://medium.com/@krishankataria2015/aurelia-vs-angular-2-why-i-prefer-aurelia-over-angular-a9c96a82c13d [26.03.2022]

[31] „Aurelia VS Angular JS Developers: What’s Good For Your Business“, Artjoker Blog [Võrgumaterjal] Saadaval: https://artjoker.net/tpost/yvuscd3v6f-aurelia-vs-angularjs-developers-whats-go [26.03.2022]

[32] L. Harkushko „Vital Things to Consider When Choosing a Database for Your App“ [Võrgumaterjal] Saadaval: https://yalantis.com/blog/how-to-choose-a-database/ [26.03.2022]

[33] „The Spring @Controller and @RestController Annotations“, Baeldung [Võrgumaterjal] Saadaval: https://www.baeldung.com/spring-controller-vs-restcontroller [12.05.2022]

[34] A.Yigal „Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases“ [Võrgumaterjal] Saadaval: https://logz.io/blog/relational-database-comparison/ [27.03.2022]

[35] „SQLite vs MySQL – Comparing 2 Popular Databases“, Keycdn [Võrgumaterjal] Saadaval: https://www.keycdn.com/support/sqlite-vs-mysql [27.03.2022]

[36] „What is a REST API?“, Red Hat [Võrgumaterjal] Saadaval: https://www.redhat.com/en/topics/api/what-is-a-rest-api [27.03.2022]

[37] „Introduction to JSON Web Tokens“, JWT [Võrgumaterjal] Saadaval: https://jwt.io/introduction [27.03.2022]

[38] „Vue.js Axios“, Educba [Võrgumaterjal] Saadaval: https://www.educba.com/vue-dot-js-axios/?source=leftnav [10.04.2022]

[39] J.Ejonavi „Vue.js Tutorial: build a functional SPA from scratch“ [Võrgumaterjal] Saadaval: https://www.educative.io/blog/vue-js-tutorial [10.04.2022]

[40] N.West „Getting Started with VueJS“ [Võrgumaterjal] Saadaval: https://medium.com/js-dojo/getting-started-with-vuejs-for-web-and-native-285dc64f0f0d [15.04.2022]

[41] J. Paul „Why use PreparedStatement in Java JDBC – Example Tutorial“ [Võrgumaterjal] Saadaval https://javarevisited.blogspot.com/2012/03/why-use-preparedstatement-in-java-jdbc.html#axzz7RRaQkJHQ [25.04.2022]

[42] A. Leonard „4. Batching – Spring Boot Persistence Best Practices: Optimize Java Persistence Performance in Spring Boot Applications“ [Võrgumaterjal] Saadaval: https://goois.net/4-batching-spring-boot-persistence-best-practices-optimize-java-persistence-performance-in-spring-boot-applications.html [25.04.2022]

[43] „jsPDF“, Parallax [Võrgumaterjal] Saadaval: https://parall.ax/products/jspdf [27.04.2022]

[44] S. Haque „How to Use Vue Router: A Complete Tutorial“ [Võrgumaterjal] https://vueschool.io/articles/vuejs-tutorials/how-to-use-vue-router-a-complete-tutorial/ [14.05.2022]

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

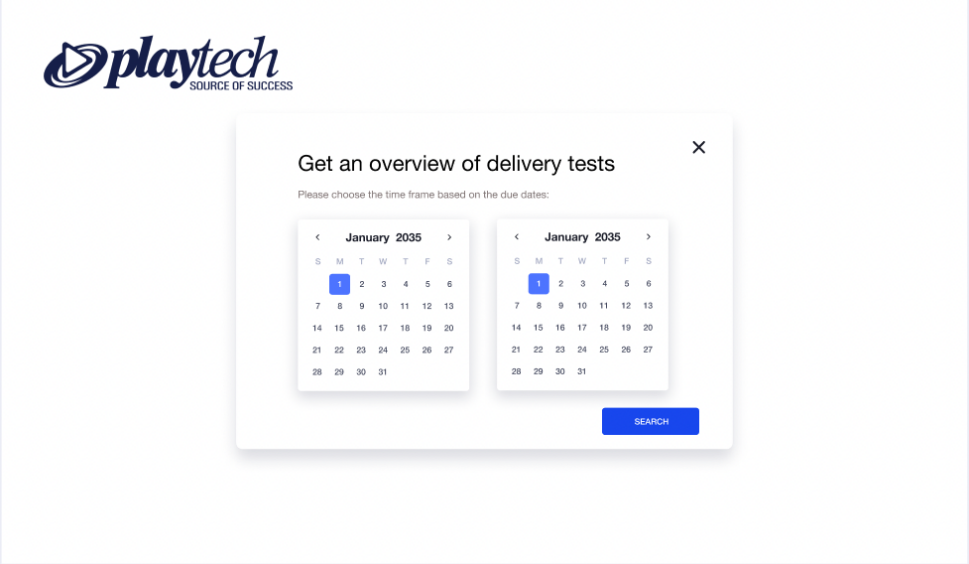
Mina, Liisa Heinla

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakenduse loomine testiaruandluse koostamiseks Videobeti näitel“, mille juhendaja on Maili Markvardt
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

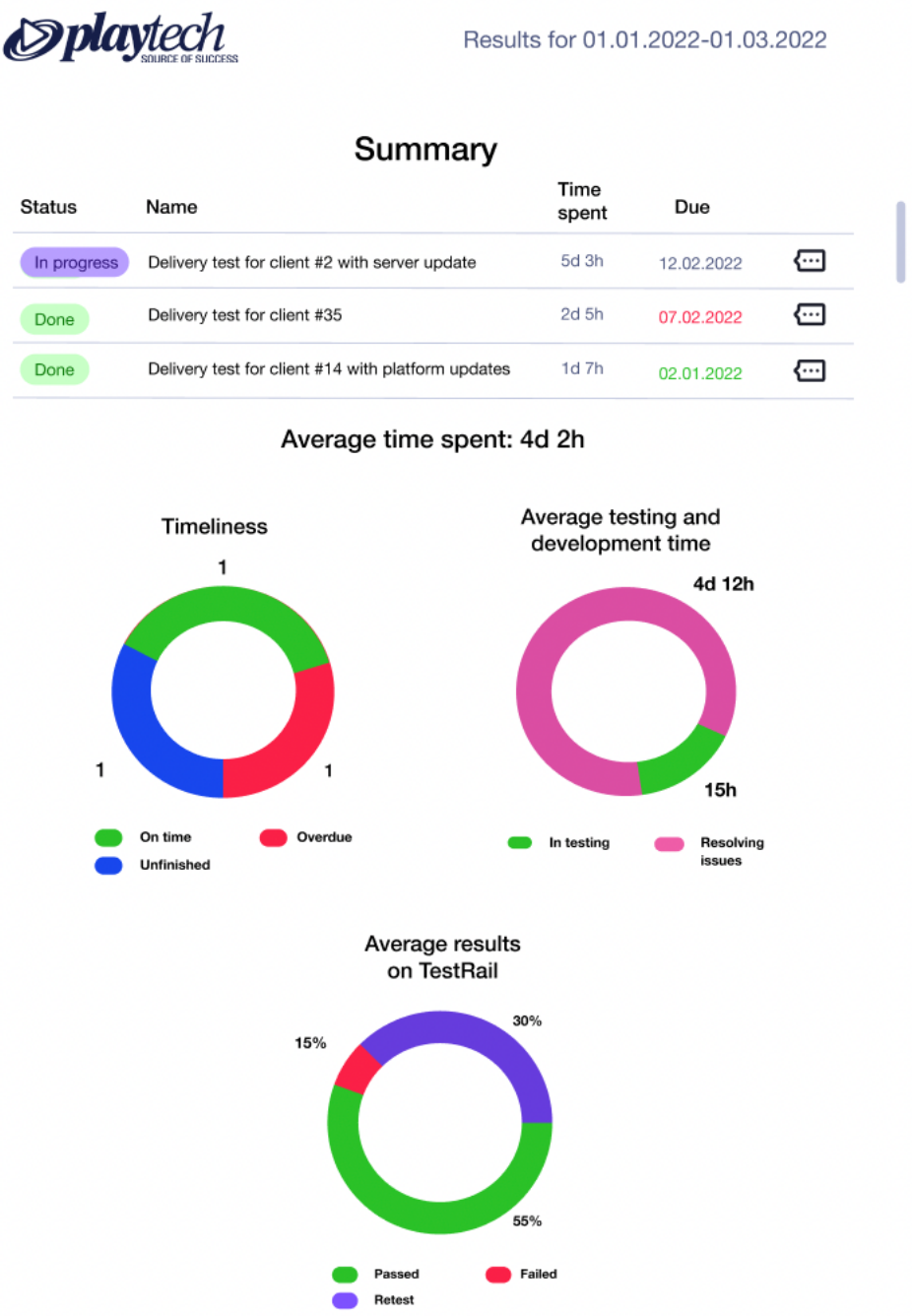
16.04.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Prototüübi muud vaated



Joonis 15. Kuupäevalise otsingu vaade

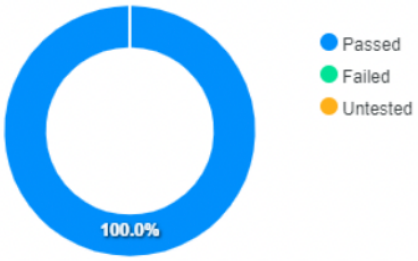


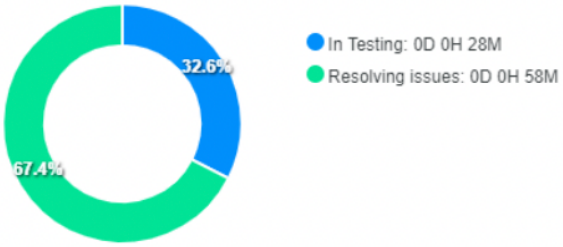
Joonis 16. Üleandmistestide summaarne vaade

Lisa 3 – Ühe üleandmise raport

(PKGRQ-5410) > | GDProd - Clarity June Content V2

Creation date	14.04.2022
Current status	Closed
Current assignee	Sachini Weragoda
Due date	13.04.2022
Untested	0
Passed	17
Failed	0
Total time	0D 1H 26M
Testing time	0D 0H 28M
Development time	0D 0H 58M





Joonis 17. Ühe üleandmise raport

Lisa 4 – Summaarne üleandmiste raport

Results for: 01.04.2022 - 13.04.2022

Average time	0D 14H 42M
Average testing time	0D 3H 33M
Average development time	0D 11H 9M
Total deliveries	11
On time deliveries	2
Overdue deliveries	9
Unfinished deliveries	0
Average passed tests	207
Average failed tests	3
Average untested tests	0

Status	Name	Time Spent	Due
Closed	(PKGRQ-5410) > | GDProd - Clarity June Content V2	0D 1H 26M	13.04.2022
Closed	### (PKGRQ-5407) > | GDICAB - Equinox 4.20 Dev Build	0D 4H 15M	13.04.2022
Closed	### (PKGRQ-5406) > | GDICAB - Clarity 4.20 Dev Build	1D 1H 58M	13.04.2022
Closed	PKGRQ-5408 > Tornado - T8 4.20 Dev Build	0D 7H 32M	11.04.2022
Closed	### (PKGRQ-5410) > | GDProd - Clarity June Content	0D 21H 3M	13.04.2022
Closed	### (PKGRQ-5409) > | SGGAB - Equinox June Content	0D 5H 38M	13.04.2022
Closed	[VBCCM-3286] Delivery against RAYPROD - Fix for VSH-2822	0D 9H 24M	12.04.2022
Closed	### (PKGRQ-5403) > | SGGAB - T8 Pot Transfer Patch (4.20 + 4.25)	0D 16H 17M	06.04.2022
Closed	### (PKGRQ-5399) > | GDOasisProd - Clarity Operators Certificate	2D 17H 54M	06.04.2022
Closed	### (PKGRQ-5400) > | GDOasisProd - Cashier Operators Certificate	0D 2H 38M	04.04.2022
Closed	### (PKGRQ-5398) > | SGGAB (Mecca) - Clarity & T8 Printer Logo Fix	0D 1H 47M	04.04.2022

Joonis 18. Summaarne üleandmiste vaade