VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

PRAKTIKOS ATASKAITA

Praktiką atliko: Maksim Prokofjev
Programų sistemos, bakalauras, IV kursas

Praktikos institucija: UAB „INVENTI“

Organizacijos praktikos vadovas: Projektų vadovė Kristina Šaulinskienė

Organizacijos praktikos vadovo įvertinimas: _____

Universiteto praktikos vadovas: Lekt. Gediminas Rimša

_____

Ataskaitos įteikimo data _____

Registracijos Nr. _____

Įvertinimas _____

Vilnius – 2020

TURINYS

Įvadas

Profesinė praktika buvo atliekama 2011 metais įkurtoje įmonėje UAB „INVENTI“. Ši įmonė pasirinkta dėl kelių priežasčių:

1. **Vykdomi projektai.** Įmonėje vykdomi skirtingi ir besikeičiantys projektai. Įmonės klientų verslo sritys - nuo telekomunikacijų iki finansinio sektoriaus. Dėl šios priežasties, galima įgauti ne tik programavimo žinių, bet ir įsigilinti į projekto verslo sritį. Susitarus, yra galimybė pakeisti projektą su kuriuo dirbama.
2. **Įmonės darbo aplinka.** Įmonė rūpinasi savo darbuotojų tiek fizine, tiek psichologine sveikata, kiekvieną mėnesį organizuojamos išvykos į renginius, pavyzdžiui, apsilankymas virtualios realybės arenoje, išvyka į krepšinio varžybas, pokerio, stalo žaidimų turnyrai, protmušiai. Patys darbuotojai organizuoja bėgimus, dalyvauja bėgimo varžybose. Kadangi įmonės kolektyvas nėra didelis, šios ataskaitos rašymo metu - 33 darbuotojai, yra galimybė su visais susipažinti ir pabendrauti. Taip pat kiekvienais metais įmonė vykdo *„workation“* praktiką. *„Workation“* yra tendencija, kai įmonės kolektyvas išvyksta į užsienio šalį padirbėti nuotoliniu būdu. Pavyzdžiui, 2019 metais *„workation“* vyko Maltoje.
3. **Naujų ir aktualių žinių įgijimas.** Įmonė stengiasi žingsniuoti koja kojon su naujausiomis technologijų tendencijomis, pavyzdžiui, naujausi projektai parašyti su Kotlin programavimo kalba. Taip pat įmonės darbuotojai savarankiškai organizuoja technologijų mokymosi sesijas (angl. *„tech-talk“*), kuriose vienas darbuotojas dalinasi su kolektyvu patirtimi, PĮ tendencijomis, naujovėmis. Taip pat įmonė turi biblioteką, kurioje yra virš 120 knygų, ir jų sąrašas pastoviai pildomas.
4. **Naudojamos technologijos.** Įmonės pagrindinė programavimo kalba yra Java/Kotlin, UI („vartotojo sąsajos“) programavimo darbams naudojama *„React“* biblioteka. Tai yra privalumas, kadangi turiu darbo patirties dirbant su šiomis technologijomis.
5. **Lanksčios darbo sąlygos.** Esant poreikiui, įmonė suteikia galimybę dirbti iš namų. Darbo valandos taip pat nėra fiksuotos, svarbiausia yra dalyvauti pokalbiuose, kurie yra planuojami iš anksto. Šios sąlygos leidžia puikiai suderinti darbą ir asmeninį gyvenimą.

Profesinės praktikos metu buvo prižiūrima ir tobulinama „Bitė Lietuva“ vidinė klientų aptarnavimo informacinė sistema. Sistemos UI aplikacija yra parašyta su JavaScript kalba, naudojama „React“ biblioteka. Sistemos serverinė dalis parašyta su Java programavimo kalba, naudojamas *„Spring“* programavimo karkasas. Serverinė dalis yra išskaidyta į posistemes, šis išskaidymas yra mikroservisų (angl. *„microservices“*) architektūrinis principas. Servisai tarpusavyje apsikeičia duomenimis per REST (angl. *„Representational State Transfer“*) arba SOAP (angl. *„Simple Object Access Protocol“*) sąsają. Taip pat tam tikri servisai įgyvendina CQRS (angl. *„Command Query Response Segregation“*) architektūrinį principą, kuris atskiria duomenų įrašymo ir duomenų užklausų sluoksnius.

Programavimui naudojama *„IntelliJ IDEA“* integruota programavimo aplinka, versijų kontrolės sistema - GIT. Bitės komanda sudaryta iš vieno projekto vadovo ir trijų programuotojų. Darbas vyksta pagal Agile metodologiją, kas dvi savaites planuojami sprintai. Projekto valdymui naudojamos dvi sistemos - vidinė *„Jira“*, ir kliento - *„Phabricator“*.

Praktikos laikotarpiui buvo iškelti tokie uždaviniai:

1. Įgyvendinti naują funkcionalumą mikroservisų ekosistemoje taikant CQRS architektūrinį principą;
2. Įgyvendinti vartotojo sąsajos pakeitimus naudojant *„React“* biblioteką.

Prisijungus prie Bitės komandos, jau buvo pradėtas vykdyti naujas projektas - *„Project TV“*. Projekto esmė - įgyvendinti funkcionalumą, kuris leistų „Bitė Lietuva“ darbuotojams per vidinę klientų aptarnavimo sistemą pardavinėti televizijos paslaugas. Kadangi sistema buvo skirta pardavinėti telekomunikacijų paslaugas, pakeitimai buvo reikalingi visose posistemėse.

Praktikos atlikimo eiga buvo tokia:

1. Susipažinti su Bitės komanda;
2. Pasiruošti darbo vietą, susikonfigūruoti programavimo aplinkas, prieigą prie kliento VPN ir versijavimo sistemos;
3. Susipažinti su projekto dokumentacija, kodo rašymo standartais;
4. Kartu su komanda planuotis sprintus ir vykdyti paskirtas užduotis.

Visų pirma buvo pristatyti Bitės komandos nariai, sistema, sistemos paskirtis ir klientas. Papasakota į ką kreiptis, prireikus pagalbos. Sekantis etapas buvo susikonfigūruoti aplinką, atsisiųsti ir įsirašyti reikiamus įrankius, susikonfigūruoti IDE, VPN, GIT. Toliau buvo pateikti susipažinimui *„Project TV“* projekto reikalavimai ir vartotojo istorijos (angl. *„User story“*). Toliau vyko Bitė komandos susitikimas, kurio metu buvo planuojamas dviejų savaičių sprintas, tarp programuotojų buvo paskirstytos užduotys. Ir galiausiai reikėjo vykdyti priskirtas užduotis, iki sekančio sprinto planavimo.

1. Praktikos vietos aprašymas

1.1. Įmonės apibūdinimas

UAB „INVENTI“ yra 2011 metais įkurta įmonė, kuri siūlo informacinių sistemų kūrimo ir integracijos paslaugas, taikydama inovatyvius sprendimus siekiant sukurti pridėtinę vertę savo klientams. Imonėje dirba specialistai, turintys didelę patirtį kuriant ir integruojant sudėtingas bankininkystės, veiklos optimizavimo, elektroninių paslaugų, valstybės informacines sistemas, registrus. Didžiausią patirtį turi finansų sektoriuje. Naudodami pažangią projektų vykdymo metodiką Agile, „INVENTI“ sumažina projekto riziką, dinamiškai reaguodama į pasikeitimus ir užtikrindama terminų laikymąsi. Taikydama iteracinį sistemų kūrimo metodą, svarbiausius sistemos funkcionalumus gali pateikti dalimis, pagal kliento prioritetus (1 pav.).

1 pav. UAB „INVENTI“ įmonės klientai (paimta iš įmonės puslapio).

1.2. Įmonės organizacinė struktūra

Šiuo metu įmonėje dirba 33 darbuotojai. Kiekvienas projektas turi savo komandą, ir komandų dydis priklauso nuo projekto apimties, prie vieno projekto dirba nuo 1 iki 10 žmonių, be projekto vadovo. Projektų vadovai vienu metu gali vadovauti keliems projektams. Mano komandoje be projekto vadovo dirbo 4 žmonės. Įmonėje didžiausia dalis kolektyvo yra programuotojai, jiems vadovauja keli projektų vadovai, taip pat yra devOps inžinierius, žmogiškųjų išteklių specialistė, administratorė ir direktorius.

1.3. Darbo sąlygos

Įmonės ofisas randasi Lvovo g. 105A, Vilnius, moderniame verslo centre „Park Town“, 3 aukšte. Darbuotojams suteikiamos parkavimo vietos, kurios randasi prie Fanų stadiono. Pesčiomis ofisą galima pasiekti per 4 minutes. „Park Town“ objektas sudarytas iš dviejų pastatų,

viename iš jų yra „Lunch up“ restoranas, į kurį ofisų darbuotojai ateina pietauti. Taip pat yra sporto salė „RE. FORMATAS“. Įmonė aprūpina visomis darbo priemonėmis, nešiojamu kompiuteriu, dviem monitoriais ir kitais įmonės identiteto atributais, pavyzdžiui, kuprinė, marškiniai, užrašų knygelė. Taip pat naujas darbuotojas gauna magnetinę kortelę, skirta patekti į ofisą. Ofisas yra atviros erdvės, vienoje erdvėje dirba 10-12 žmonių, visi komandos nariai sėdi vienoje erdvėje. Ofise yra kelios erdvės pokalbiams, susitikimams. Kiekviena erdvė, arba konferencijų salė turi unikalų pavadinimą, pagal tam tikrą atributą, pavyzdžiui, erdvė, kurioje sienos yra iš stiklo, vadinama akvariumu. Darbuotojų poilsiui įrengtas poilsio kambarys, taip pat įrengta virtuvėlė, kurioje dažnai vyksta įvairūs renginiai, pavyzdžiui, švenčiami gimtadieniai, projektų užbaigimas, „tech-talk'ai“.

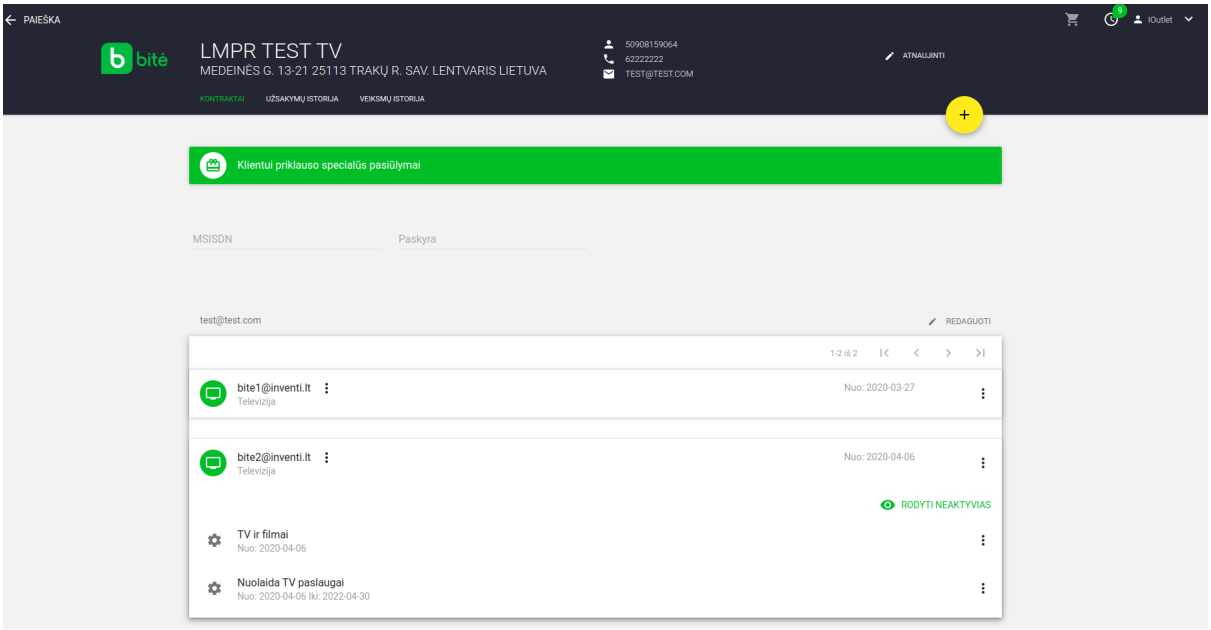
Darbo laikas nėra fiksuotas, tačiau standartiškai, darbuotojai ateina į darbą nuo 7 iki 10 ryto. Išdirbtos darbo valandos fiksuojamas „Clockify“, kur projektų vadovai veda komandos išdirbtų valandų statistiką, pagal kurią atitinkamai išrašo klientams sąskaitas. Pirma diena atėjęs į darbą, darbuotojas darbo vietoje randa lapelį, kuriame trumpai supažindinama su kolektyvu, nurodoma, kas kur sėdi, į ką kreiptis, kilus įvairiems klausimams. Taip pat gauna prieigas prie „Gmail“ el. pašto ir vidinių sistemų, tokiu kaip „Jira“, „Confluence“. Bendravimas tarp įmonės darbuotojų vyksta per „Slack“ sistemą.

Šio darbo rašymo metu šalyje yra paskelbtas karantinas, dėl koronaviruso COVID-19 grėsmės. Praktikos atlikimas persikėlė į namų erdvę, tačiau tai mažai įtakojo darbo procesą - pokalbiai vykdomi per „Slack“, „Google Hangouts“ arba „Microsoft Teams“ sistemas, penktadieniais per šias sistemas vyksta protmušiai, darbuotojai organizuoja kolektyvinius bėgimus, tačiau atsižvelgiant į karantino sąlygas - darbuotojai bėga po vieną, o po bėgimo pasidalina nuotrauka ir nubėgtu atstumu. Visa tai padeda išlaikyti įmonėje draugiškumo, vieningumo ir jaukumo jausmus.

2. Praktikos veiklos aprašymas

2.1. Projektas

Kaip buvo minėta ankščiau, profesinės praktikos metu visi darbai buvo vykdomi su „Bitė Lietuva“ vidinė klientų aptarnavimo informacinė sistema. „Bitė Lietuva“ yra įmonės nuolatinis klientas, 2018 metais „INVENTI“ sukūrė šią sistemą [Bit18], ir integravo su egzistuojančiomis „Bitė Lietuva“ sistemomis. Sistema įmonės viduje turi kodinį pavadinimą „medus“ (2 pav.). Klientui „Bitė Lietuva“ atsirado poreikis atlikti televizijos paslaugų pardavimą per šią sistemą, tam kad galutinis vartotojas galėtų naudotis „Go3 | BITĖ“ televizijos paslaugomis [Bit20]. „INVENTI“ pasirašė sutartį atlikti šį projektą dviem etapais. Taip pat prie projekto prisijungė ir kiti vendoriai - įmonė „Baltic Amadeus“.



2 pav. Vidinės klientų aptarnavimo sistemos vartotojo sąsaja.

2.2. Darbo procesas

Prieš pradedant projektą, komanda atlieka kliento pateiktų vartotojų istorijų analizę, išskaido jas į užduotis, kiekvieną užduotį įvertina, ir sudeda į projekto neatliktų užduočių sąrašą. Darbas vykdavo pagal Agile metodologiją, kas dvi savaites planuojami sprintai. Sprinto planavimo metu, visa komanda susirenka ir padaro atliktų užduočių apžvalgą „Jira“ užduočių projekto sistemoje, apžvelgiama, kokios užduotys buvo atliktos praeitame sprinte, o kurios keliauja į naują. Atlikus peržiūrą, praeitas sprintas uždaromas, atidaromas naujas. Neužbaigtos užduotys perkeliamos į naują sprintą, ir tada atliekama neatliktų užduočių sąrašo analizė. Iš neatliktų užduočių sąrašo, užduotys perkeliamos į naują sprintą pagal svarbumo prioritetą, tol, kol nebus viršytas dviejų savaičių darbo valandų limitas. Įmonėje taikoma praktika, kad programuotojas programavimo darbams vidutiniškai skiria 6 valandas, 2 valandos lieka susitikimams, pokalbiams su klientu, arba kitai, su programavimu nesusijusiai veiklai. Pirmadieniais vykdavo susitikimas, kurio metu buvo uždaromas praeitas sprintas ir buvo planuojamas naujas.

Kiekviena diena ryte vyksta „stand-up“ susitikimai, dažniausiai tai konferencinis pokalbis su klientu ir kitais vendoriais, bet kartais tekdavo vykti pas klientą į ofisą, kur šie susitikimai vykdavo gyvai. Kiekvienas „stand-up“ dalyvis trumpai nupasakoja, ką praeitą dieną nuveikė, su kokiomis problemomis susidūrė ir kaip jas sprendė. Šie susitikimai dažniausiai užtrunka 15 minučių, po jų seka komandos vidiniai komandos pasitarimai.

Programavimo darbai prasideda nuo „Jira“ užduoties statuso pakeitimo, kurį pradėjus darbą reikia pakeisti į „in progress“. Programavimo darbams sukuriama nauja GIT šaka, kurios pavadinimas - užduoties numeris. Atlikus programavimo darbus, kodo versijavimo sistemoje sukuriamas „merge request“ į vykdomo projekto šaką, kurio paskirtis - kodo peržiūra. Kodo peržiūras atlieka kiti komandos programuotojai, peržiūrėjus kodą, savo pastabas surašo „Gitlab“ versijavimo sistemoje. Jeigu pastabų nėra - duodamas leidimas atlikti šakų suvienijimą. Tada galima pakeisti užduoties statusą į *„done“*. Kai visos užduotys, priklausančios tam tikrai vartotojo istorijai yra užbaigtos, prašomas kliento leidimas diegti PĮ pakeitimus į testinę *„test“* aplinką, kurioje kliento testuotojai galėtų ištestuoti naują funkcionalumą.

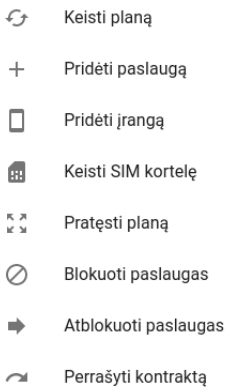
Jeigu funkcionalumas turi klaidų - testuotojai užregistruoja jas „Phabricator“ sistemoje. Iš šios sistemos, jos perkeliamos į „Jira“ sistemą, atliekami klaidų taisymo darbai, pataisymai sukeliami į testinę aplinką, kol galiausiai įgyvendintas funkcionalumas atitinka tam, koks buvo specifikuotas reikalavimuose. Kai yra įgyvendintas tam tikras funkcionalumo paketas, planuojamas išleidimas (angl. *„release“*) į produkcinę aplinką. Kai visas naujas funkcionalumas pilnai ištestuotas, ir regresinis testavimas neatranda klaidų - naujo funkcionalumo paketas diegiamas į *„staging“* aplinką, kuri yra pagal konfigūraciją - identiška produkcinei. Atlikus paskutinius testavimo darbus, funkcionalumo paketas diegiamas į produkcinę *„prod“* aplinką. Šie diegimai atliekami vidutiniškai kartą per mėnesį.

Programavimo darbai vykdomi lokaliai, norint savarankiškai ištestuoti tam tikrą funkcionalumą, kai reikalinga visa infrastruktūra, pakeitimai diegiami į *„develop“* aplinką.

2.3. Projekte naudojamos technologijos

Kaip jau buvo minėta ankščiau, sistemos UI dalis yra parašyta su JavaScript kalba, naudojama *„React“* biblioteka. Vartotojo sąsajai naudojama „Material UI“ biblioteka. UI aplikacija su serverine dalimi bendrauja naudojant tarpinį serverį - *„Nginx“*. Sistemos serverinė dalis parašyta su Java programavimo kalba, naudojamas *„Spring“* programavimo karkasas. Serverinė dalis yra įgyvendina mikroservisų architektūrinį principą. Servisai tarpusavyje apsikeičia duomenimis per REST arba SOAP sąsają. Du servisai, daugiausiai bendraujantys su UI, įgyvendina naudojant CQRS architektūrinį principą, kuris atskiria duomenų įrašymo ir duomenų užklausų sluoksnius. Vienas servisas atsakingas už užsakymų krepšelio valdymą, sistemoje įgyvendinta didelė aibė skirtingų procesų (3 pav.), kurie vykdomi per užsakymų krepšelį. Kitas servisas atsakingas krepšelio apdorojimą, iš gautų duomenų sugeneruojami sutarties duomenys, kurie nukeliauja į vidinę apskaitos sistemą. Žinutėms (angl. *„events“*) tarp servisų naudojamas žinučių brokeris *„Apache Kafka“*. Kadangi servisas, bendraujantis su apskaitos sistema naudoja SOAP protokolą, buvo sukūrtas servisas, kurio paskirtis - gaunamas REST užklausas, naudojant *„Apache Camel“* tech-

nologiją, paversti į SOAP užklausas, ir perduoti dedikuotam servisui. Taip pat yra ir kiti servisai, turintys po vieną paskirtį, pavyzdžiui, vienas servisas atsakingas už „PDF“ sutarčių generavimą, kitas – už el. laiškų siuntimą ir t.t. Kadangi *„Bitė Lietuva“* priklauso *„Bitė“* grupei, kurioje taip pat vykdo veiklą įmonė „Bitė Latvija“, tam tikri servisai veikia tik vienos šalies aplinkose.



3 pav. Galimi kliento sutarties procesai.

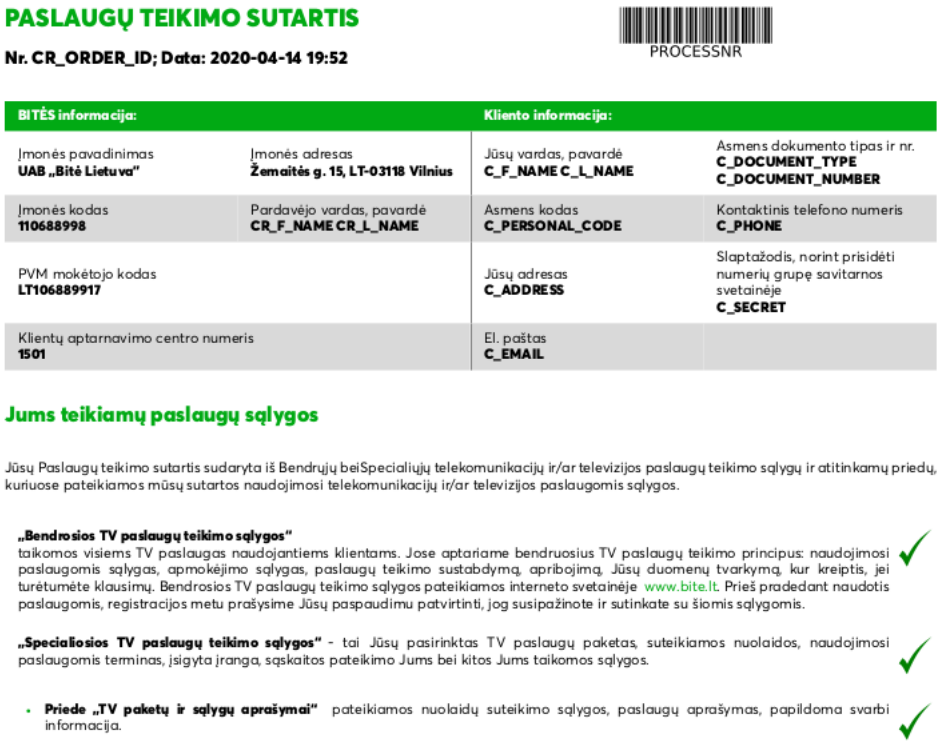
Vidinė apskaitos sistema naudoja „Oracle“ duomenų bazę, į kurią duomenys perduodami naudojant saugomas (angl.*„Stored“*) procedūras. Šias procedūras kviečia dedikuotas servisas.

Mikroservisų architektūros valdymui naudojama konteinerių technologija „Docker“, kuri leidžia programų kūrėjams ir sistemų administratoriams lengvai ir greitai talpinti aplikacijas konteineriuose, kurie izoliuoti nuo operacinės sistemos. Konteinerizuotos aplikacijos diegimui naudojama „Kubernetes“ konteinerių orkestravimo technologija. Ši technologija leidžia koreguoti resursus pagal poreikį, lengvai atnaujinti PĮ versijas, apdoroti žurnaliavimą ir t.t.

3. Programavimo darbai

3.1. Užduotis: Pritaikyti PDF šablonus televizijos paslaugų sutartims

Šios užduoties aprašymas apima kelias užduotis, kadangi sistemoje yra didelis kiekis skirtingų PDF šablonų, pavyzdžiui, nauja sutartis, sutarties nutraukimas, paslaugų pajungimas, paslaugų nutraukimas ir t.t. Klientas pateikė PDF failus, kaip turi atrodyti nauji televizijos sutarčių šablonai (4 pav.). Taip pat pakeitimai turi būti pritaikyti ne tik lietuviškoms sutartims, bet ir latviškoms.



PASLAUGŲ TEIKIMO SUTARTIS

Nr. CR_ORDER_ID; Data: 2020-04-14 19:52

PROCESSNR

BITĖS informacija:		Kliento informacija:	
Įmonės pavadinimas **UAB „Bitė Lietuva"**	Įmonės adresas **Žemaitės g. 15, LT-03118 Vilnius**	Jūsų vardas, pavardė **C_F_NAME C_L_NAME**	Asmens dokumento tipas ir nr. **C_DOCUMENT_TYPE** **C_DOCUMENT_NUMBER**
Įmonės kodas **110688998**	Pardavėjo vardas, pavardė **CR_F_NAME CR_L_NAME**	Asmens kodas **C_PERSONAL_CODE**	Kontaktinis telefono numeris **C_PHONE**
PVM mokėtojo kodas **LT106889917**		Jūsų adresas **C_ADDRESS**	Slaptažodis, norint prisidėti numerių grupę savitarnos svetainėje **C_SECRET**
Klientų aptarnavimo centro numeris **1501**		El. paštas **C_EMAIL**	

Jums teikiamų paslaugų sąlygos

Jūsų Paslaugų teikimo sutartis sudaryta iš Bendrųjų beiSpecialiųjų telekomunikacijų ir/ar televizijos paslaugų teikimo sąlygų ir atitinkamų priedų, kuriuose pateikiamos mūsų sutartos naudojimosi telekomunikacijų ir/ar televizijos paslaugomis sąlygos.

„Bendrosios TV paslaugų teikimo sąlygos" taikomos visiems TV paslaugas naudojantiems klientams. Jose aptariame bendruosius TV paslaugų teikimo principus: naudojimosi paslaugomis sąlygas, apmokėjimo sąlygas, paslaugų teikimo sustabdymą, apribojimą, Jūsų duomenų tvarkymą, kur kreiptis, jei turėtumėte klausimų. Bendrosios TV paslaugų teikimo sąlygos pateikiamos interneto svetainėje www.bite.lt. Prieš pradedant naudotis paslaugomis, registracijos metu prašysime Jūsų paspaudimu patvirtinti, jog susipažinote ir sutinkate su šiomis sąlygomis.

„Specialiosios TV paslaugų teikimo sąlygos" - tai Jūsų pasirinktas TV paslaugų paketas, suteikiamos nuolaidos, naudojimosi paslaugomis terminas, įsigyta įranga, sąskaitos pateikimo Jums bei kitos Jums taikomos sąlygos.

- **Priede „TV paketų ir sąlygų aprašymai"** pateikiamos nuolaidų suteikimo sąlygos, paslaugų aprašymas, papildoma svarbi informacija.

4 pav. Naujas televizijos paslaugų PDF šablonas.

Problema. Visi sistemos procesai ir esybės remiasi į mobilųjį numerį (toliau „MSISDN"). Televizijos paslaugos (toliau „TV") telefono numerio neturi, todėl buvo nuspręsta, kad pagrindinis TV paslaugų unikalus identifikatorius bus elektroninis paštas (2 pav.).

Analizė. Telekomunikacijų paslaugų (toliau „GSM") PDF šablonai yra labai panašūs į TV paslaugų šablonus, tam tikrose vietose vietoj „mobilusis numeris" pasikeisdavo į „elektroninis paštas". Tam tikrose vietose pasikeisdavo statinis tekstas. Tačiau tam tikri šablonai buvo visiškai skirtingi. Kaip ir buvo minėta, mikroservisų architektūroje kiekvienas servisas atsakingas už tam tikrą funkcionalumą, šiuo atveju, yra du servisai - vienas atsakingas už lietuviškų sutarčių generavimą, kitas - už latviškų. PDF dokumentų generavimui naudojama „Apache FOP" „XML" pagrįsta biblioteka. „XML" faile yra aprašomos dokumento formatavimo, stiliaus taisyklės, biblioteka nuskaito iš Java kalbos objekto duomenis, ir juos atvaizduoja nurodytose „XML" failo vietose. Taip pat biblioteka palaiko sąlyginius sakinius, ciklus duomenų generavimui.

Sprendimas. Pirminis sprendimas buvo prie PDF dokumento generavimo užklausos ob-

jekto pridėti papildomą el. pašto lauką, tačiau šis sprendimas yra gali sukelti klaidų, todėl jo buvo atsisakyta. Kadangi didžioji dalis sistemos (ne tik PDF dokumentų generavimo) funkcionalumo remiasi į MSISDN lauką, nuspręsta šį lauka pervadinti į „accessPointAddress“, ir jame leisti laikyti elektroninį paštą. Šis sprendimas nėra pats geriausias, tačiau greičiausias. Šis sprendimas įtakojo PDF dokumentų generavimą, ir tam buvo įvesta žyma *„contractType“*, kurios reikšmės gali būti arba *„TV“*, arba *„GSM“*.

Įgyvendinimas. Pagal *„contractType“* žymą, atitinkamai buvo generuojama arba TV, arba GSM paslaugų sutartys (5 pav.). Taip pat generuojamiems PDF dokumentams, buvo parašyti parametrizuoti „JUnit“ bibliotekos testai (6 pav.), kurie generuoja skirtingus PDF dokumentus su skirtingais parametrais (7 pav.), bei tikrina, ar šie duomenys atvaizduojami dokumente. Tikrinimui naudojami „Hamcrest“ bibliotekos tikrinimo metodai (angl. *„matchers“*). Tačiau surašius testavimo atvejus, atsirado kita problema - kliento parašas. UI aplikacija, naudojama „Bitė“ salone, yra prijungta prie planšetinio kompiuterio, kuris, gaudamas žinutę iš aplikacijos, atvaizduoja kliento sutartį, bei leidžia naudojant elektroninį pieštuką, dokumentą pasirašyti.

Parašas buvo įklijuojamas į PDF dokumentą. Problema buvo tame, kad PDF dokumentuose, parašas atsidurdavo ne jam skirtoje vietoje. Parametrizuoti testai leido anksti atrasti šios problemos egzistavimą, ir atlikus analizę paaiškėjo, kad parašas įklijuojamas pagal šablonui nurodytas koordinates. Galiausiai problema buvo išspręsta šablonus pakoregavus taip, kad parašo vieta visada liktų toje pačioje vietoje, nepriklausomai nuo duomenų.

```
<xsl:choose>
    <xsl:when test="//contractType = 'TV'">
        <xsl:call-template name="contractTerminationTV"/>
    </xsl:when>
    <xsl:otherwise>
        <xsl:call-template name="contractTerminationGSM"/>
    </xsl:otherwise>
</xsl:choose>
```

5 pav. *„contractType“* žymos tikrinimas.

```
@Test
public void template() throws IOException {
    long startTime = currentTimeMillis();
    long startHeap = currentUsedHeap();

    String documentId = documentsService.generatePdfAndStore(request, processNo, template, props);

    long duration = currentTimeMillis() - startTime;
    long usedHeap = currentUsedHeap() - startHeap;

    assertThat(logListener.errors(), is(empty()));
    assertThat(documentId, is(processNo));
    byte[] pdfBytes = documentsService.retrieveContract(documentId).orElse( other: null);
    assertThat(pdfBytes, is(notNullValue()));

    String text = extractPdfText(pdfBytes);
    assertThat(text, matcher);

    if (signature) {
        documentsService.appendSignatureImage(documentId, getSignature(), DocumentType.valueOf(template));
    }

    System.out.println("=====================================================================");
    System.out.println("PDF generation took " + duration + " ms and used " + (usedHeap / 1024) + " KB of memory");
}
```

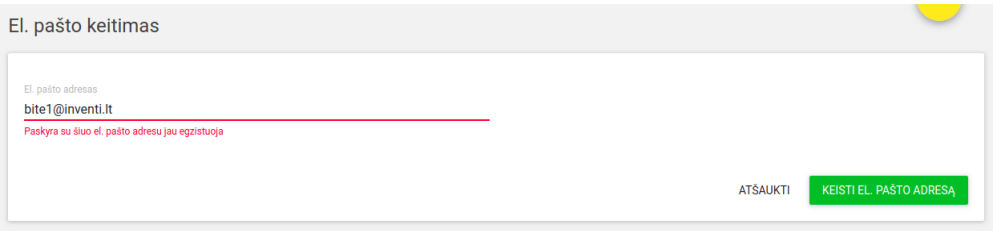
6 pav. Parametrizuoto testo šablonas.

```
@Parameterized.Parameters(name = "should generate PDF {1} for template {2}")
public static Collection<Object[]> data() {
    return Arrays.asList(
        createData(contractRequest(GSM), template: "CommonRules", orderIdMatcher()),
        createData(contractRequest(GSM), template: "FiberTeliaAppeal", allOf(customerShortMatcher(), customerAddressMatcher(), customerPhoneMatcher())),
        createData(contractRequest(GSM), processNo: "GeneralInfoGSM", template: "GeneralInfo", allOf(generalInfoMatcher(), gsmNumberDetailsMatcher())),
        createData(contractRequest(TV), processNo: "GeneralInfoTV", template: "GeneralInfo", allOf(generalInfoMatcher(), tvNumberDetailsMatcher())),
        createData(contractRequestMixed(), processNo: "GeneralInfoMixed", template: "GeneralInfo", allOf(generalInfoMatcher(), tvNumberDetailsMatcher(), gsmNumberDetailsMatcher())),
        createData(numberDetailsRequest(), template: "ChangeSimInstructions", allOf()),
        createData(numberDetailsRequest(), template: "GDPR", gdprParams(), allOf(customerGDPRMatcher(), numberDetailsSellerMatcher(), gdprPhoneMatcher()), signature: false),
        createData(numberDetailsRequestWithMnp(), template: "ContractArchive", allOf(customerMatcher(), mnpMatcher())),
        createData(numberDetailsRequestWithMnp(), processNo: "MNP", template: "MNPContract", allOf(customerMatcher(), mnpMatcher())),
        createData(numberDetailsRequestWithInsurance(), template: "HomeInsurance", allOf(customerNameMatcher(), homeInsuranceMatcher())),
        createData(numberDetailsRequestWithInsurance(), template: "TravelInsurance", travelInsuranceMatcher()),
        createData(numberDetailsRequestWithReconnect(), template: "ReconnectContract", allOf(customerMatcher(), numberDetailsSellerMatcher(), reconnectMatcher()), signature: true),
        createData(numberDetailsRequestWithEquipment(INSURANCE_FULL), processNo: "FullInsurance", template: "Insurance", allOf(customerNameMatcher(), shortEquipmentMatcher(), insuranceMatcher())),
        createData(numberDetailsRequestWithEquipment(INSURANCE_SCREEN), processNo: "ScreenInsurance", template: "Insurance", allOf(customerNameMatcher(), shortEquipmentMatcher(), insuranceMatcher())),
        createData(numberDetailsRequestWithTransfer(), template: "ContractTransfer", allOf(customerMatcher(), targetCustomerMatcher(), ndPhoneNumberMatcher()), signature: true),
        createData(eSignatureRequest(), template: "ElectronicSignatureTermsAndConditions", eSignatureMatcher()),
        createData(sodraConsentRequest(), template: "SodraConsent", customerShortMatcher()),
        createData(creditInvoiceRequest(), template: "CreditInvoice", creditInvoiceMatcher()),
        createData(equipmentInvoiceRequest(), template: "EquipmentInvoice", allOf(equipmentInvoiceMatcher(), equipmentMatcher()), signature: true),
        createData(contractTerminationRequest(GSM), processNo: "ContractTerminationGSM", template: "ContractTermination", allOf(customerMatcher(), contractTerminationMatcher(), contractTerminationMigrationMatcher()),
        createData(contractTerminationRequest(TV), processNo: "ContractTerminationTV", template: "ContractTermination", allOf(customerMatcher(), contractTerminationMatcher()), signature: true),
        createData(serviceTerminationRequest(GSM), processNo: "ServiceTerminationGSM", template: "ServiceTermination", allOf(customerMatcher(), serviceTerminationMatcher(), serviceNameMatcher()), signature: true),
        createData(serviceTerminationRequest(TV), processNo: "ServiceTerminationTV", template: "ServiceTermination", allOf(customerMatcher(), serviceTerminationMatcher()), signature: true),
        createData(contractRequestChangeService(GSM), processNo: "BundleAppealGSM", template: "BundleAppeal", allOf(customerMatcher(), ndPhoneNumberMatcher(), sellerMatcher()), signature: true),
        createData(contractRequestChangeService(TV), processNo: "BundleAppealTV", template: "BundleAppeal", allOf(customerMatcher(), bundleMatcher(), sellerMatcher()), signature: true),
        createData(suspendServicesAppealRequest(), template: "SuspendServices", allOf(customerMatcher(), suspendServicesMatcher()), signature: true),
        createData(equipmentTerminationAppealRequest(), template: "EquipmentTermination", allOf(customerMatcher(), equipmentTerminationMatcher()), signature: true)
    );
}
```

7 pav. Parametrizuoto testo duomenų sukūrimas.

3.2. Užduotis: El. pašto adreso keitimas.

Problema. TV sutarties identifikatorius - el. pašto adresas, gali būti keičiamas, tačiau jis turi išlikti unikalus. Pakeitimas turi būti atvaizduojamas vidinėje apskaitos sistemoje, taip pat UI aplikacijos pakeitimų istorijoje. Keičiant el. pašto adresą, turi būti atliekamas asinchroninis tikrinimas, ar nurodytas naujas el. pašto adresas yra validus (8 pav.).



8 pav. El. pašto keitimo procesas.

Analizė. Atliekant analizę pastebėta, kad procesas beveik sutampa su „MSISDN“ keitimo procesu, išskyrus validavimo žingsnį, kadangi keičiant „MSISDN“, vartotojas atlieka paiešką tarp neužimtų „MSISDN“, o el. pašto adresu atveju - adresas įvedamas rankiniu būdu. Pakeitimas turi būti atliktas keliuose servisuose, taip pat ir UI aplikacijoje. Taip pat pastebėta, kad el. pašto adreso validavimo API yra įgyvendintas, ir naudojamas naujos TV paslaugų sutarties sudarymo metu.

Sprendimas. Nuspręsta, kad procesas bus įgyvendintas analogiškai, kaip įgyvendintas „MSISDN“ keitimo procesas, išskyrus validavimo žingsnį. Tikrinimui bus naudojamas egzistuojantis el. pašto adreso validavimo API.

Įgyvendinimas. Saugomas procedūras įgyvendina „Bitė Lietuva“ programuotojai, reikėjo įgyvendinti šios procedūros kvietimą servise, kuris sąveikauja su apskaitos sistemos duomenų baze. Buvo įgyvendintas SOAP protokolo API, kurio kvietimui buvo atnaujintas kitas servisas, atsakingas už REST užklausų vertimą į SOAP. Šio serviso API kvietimas įgyvendintas kitame servise, kuris glaudžiai bendrauja su UI aplikacija. Šiame servise taip pat įgyvendintas el. pašto validavimas, tam kad vartotojas negalėtų apeiti šį žingsnį, nenaudodamas aplikacijos UI. El. pašto

adreso validavimo žingsnis buvo įgyvendintas tiek sinchroniškai, tiek asinchroniškai (9 pav.). Sudėtingumas iškilo atlikus el. pašto validavimo API. API įgyvendintas naudojant „Apache Camel“ technologiją, su kurią patirties neturėjau. Naujos paslaugų sutarties sudarymo metu, el. pašto validavimo API reikalauja kitų duomenų iš užsakymų krepšelio, kurių el. pašto keitimo proceso metu, nėra. Taip pat servisas, API kvietimo metu, iškviečia 5 skirtingų servisų API, tam kad užtikrinti el. pašto unikalumą. Jeigu bent vienas API grąžina klaidą, vartotojui grąžinamas klaidos kodas, o UI aplikacijoje pagal klaidos kodą parenkamas vertimas.

```
form: {
  form: FORM_NAME,
  validate: ({ tvEmailAddress }, props) => {
    if (!tvEmailAddress) {
      return {
        tvEmailAddress: props.t('form.changeEmail.validation.emailRequired')
      };
    } else if (!isEmailValid(tvEmailAddress)) {
      return { tvEmailAddress: props.t('form.changeEmail.validation.email') };
    }
  },
  asyncValidate: ({ tvEmailAddress }, dispatch, props) => {
    const { validateChangeEmail: validate, t } = props;

    return validate(tvEmailAddress).then(({ payload }) => {
      if (payload.status !== 'OK') {
        // eslint-disable-next-line no-throw-literal
        throw {
          tvEmailAddress: t(`form.changeEmail.error.${payload.errorCodes[0]}`)
        };
      }
    });
  }
```

9 pav. „React“ komponento el. pašto keitimo validavimo kodas.

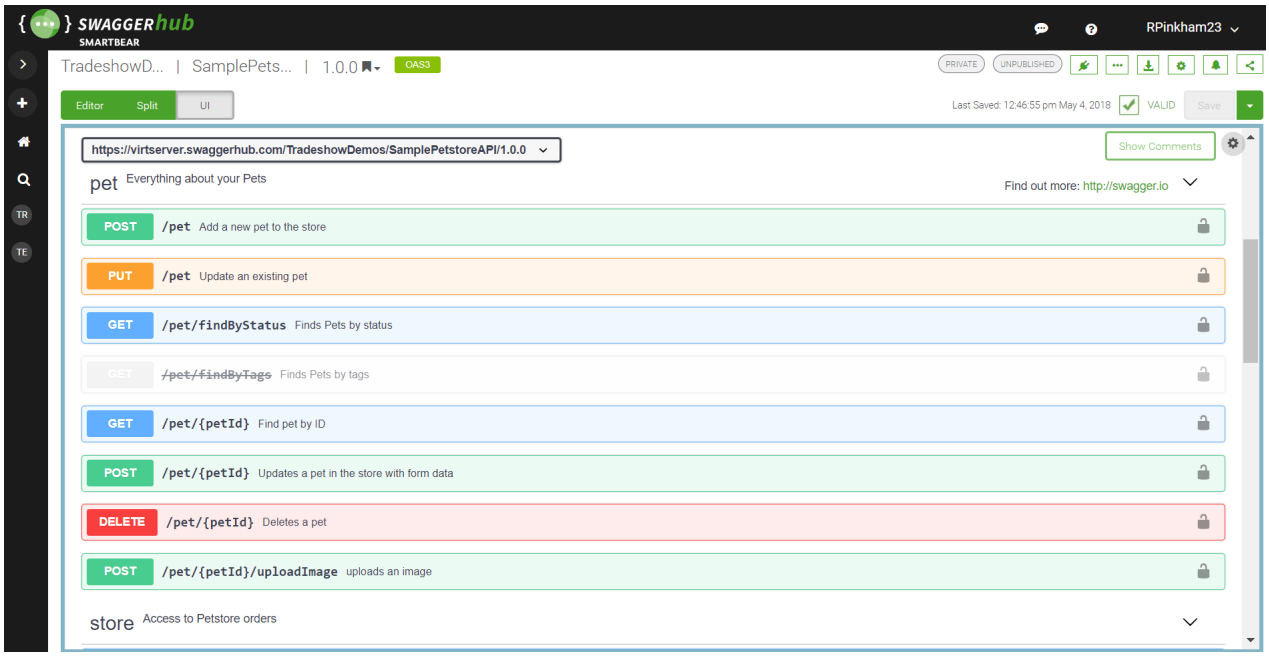
El. pašto keitimo metu, tam tikros validacijos nėra būtinos, todėl nuspręsta validavimo API perduoti el. pašto validavimo proceso tipą, pagal kurį servisas nusprendžia, kuriuos kvietimus būtina atlikti.

3.3. Užduotis: Krepšelio REST API dokumentacija.

Problema. Kaip buvo minėta ankščiau, dauguma sistemos procesų vykdomi per krepšelį, nurodant proceso pavadinimą. Sistemoje galima pridėti skirtingas paslaugas į krepšelį, pašalinti iš krepšelio, tačiau svarbiausias žingsnis - krepšelio tvirtinimas (angl. *„checkout“*). Šis žingsnis reiškia, kad krepšelio duomenys perduodami kitam servisui sutarties generavimui. Tačiau pardavimus planuojama vykdyti ne tik per „medus“ sistemą, „Bitė Lietuva“ programuotojų komanda vykdo savitarnos kūrimo projektą „Go3 | BITĖ“, leidžianti klientui savarankiškai užsisakyti paslaugas. Šiam funkcionalumui įgyvendinti buvo sukurtas krepšelio tvirtinimo REST API, kuriam perduodant duomenis, paslaugos galėtų būti aktyvuotos be UI aplikacijos pagalbos. Kadangi REST API įgyvendino „INVENTI“ programuotojai, šios užduoties tikslas - suteikti informaciją apie API, padėti „Bitė Lietuva“ sukurti integraciją.

Analizė. Analizę sudarė informacijos apie REST API laukus rinkimas. Pagrindiniai informacijos šaltiniai - komandos nariai, turintys patirties dirbant su šia sistema, taip pat „Bitė Lietuva“ darbuotojai - sistemos analitikai.

Sprendimas. Nuspręsta REST API dokumentaciją sudaryti naudojant *„Swagger 2“* biblioteką. Ši biblioteka leidžia nesudėtingai, anotacijų pagalba, dokumentuoti sistemos REST API. Biblioteka yra lengvai konfigūruojama, taip pat yra galimybė sukonfigūruoti *„Swagger UI“* (10 pav.[1]) sąsają, kuri leidžia atlikti REST API operacijas, ir yra pasiekiama per nuorodą. Taip pat ši biblioteka leidžia REST API duomenis pasiekti JSON formatu, dėl šios priežasties, yra galimybė importuoti API dokumentaciją ir naudotis kitais įrankiais, pavyzdžiui, *„Postman“*.



10 pav. „Swagger UI“ sąsaja.

Įgyvendinimas. Apibrėžtiems REST API užklausos laukams buvo sudėtos anotacijos, kurioje nurodoma, už ką laukas atsakingas, taip pat pateikiami pavyzdžiai (11 pav.). Tačiau šio API problema tame, kad didžioji dalis laukų yra neprivalomi. Dėl šios priežasties, šie laukai nėra apibrėžti kode, o sudedami į *„Map“* duomenų struktūrą, iš kurios vėliau ištraukiami pagal raktą. Tačiau *„Swagger 2“* leidžia apibrėžti savo duomenų tipą, ir jį priskirti bet kuriam laukui. Atlikus detalesnę kodo analizę, pavyko surinkti visas įmanomas šios duomenų struktūros reikšmes, kurios vėliau buvo dokumentuojamos. *„Bitė Lietuva“* programuotojų komandai pora kartų kilo problemos su šiuo REST API, tačiau pateikus užklausos pavyzdį, klaidos buvo greitai identifikuotos ir pašalintos.

[1]Paimta iš oficialaus *„swagger.io“* puslapio.

```java
@ApiModelProperty(value = "Customer type ID", example = "4", position = 1)
public Long customerTypeId;

@ApiModelProperty(value = "Customer first name", example = "Vardenis", position = 2)
public String firstName;

@ApiModelProperty(value = "Customer last name", example = "Pavardenis", position = 3)
public String lastName;

@ApiModelProperty(value = "Customer personal code", example = "48908319269", position = 4)
public String personalCode;
```

11 pav. *„Swagger 2“* dokumentavimo pavyzdys.

4. Rezultatai

4.1. Išvados ir apibendrinimai

Praktika įmonėje UAB *„INVENTI“* suteikė galimybę išbandyti savo programavimo įgūdžius vystant telekomunikacijų verslo sistemą, kurioje naudojamos didelė aibė skirtingų technologijų, tai leido įgyti daug naudingos patirties. Teko išmokti dirbti pagal Agile metodologiją, atlikti darbų vertinimus, planuoti sprintus, graudžiai bendrauti su klientu, išsiaiškinti jo poreikius sistemai, teikti pasiūlymus.

Teko susipažinti su nedidelės IT įmonės kasdienybe, įgauti nuolatinio tobulinimosi ir mokymosi mentalitetą. Darbas komandoje patobulino komunikavimo įgūdžius, diskusijos su komandos nariais leido pasirinkti geriausius ir optimaliausius sprendimus, taip pateisinant kliento lūkesčius.

4.2. Privalumai ir trūkumai

4.2.1. Privalumai

- Vienas svarbiausių aspektų praktikos metu, buvo universitete įgytų žinių taikymas. Trečiame kurse dalyko „Interneto technologijos“ įsisavintos žinios leido greičiau atlikti UI darbus su *„React“* biblioteka. Dalyko „Programų sistemų kūrimas“ sistemos kūrimo metu, buvo naudojamas Java kalbos *„Spring“* karkasas, tai leido pritaikyti šias žinias praktikoje ir kokybiškai atlikti programavimo darbus.
- Didžioji dalis programų sistemų dalykų orientuoti į komandinį darbą - tai atsispindėjo praktikos atlikimo metu, įsilieti į komandinį darbą buvo pakankamai lengva, tačiau komandos atsakomybė žymiai didesne - komandos narių klaidos gali įtakoti kliento pasitenkinimą, o didelė klaida gali kainuoti įmonei prarastą klientą, taip paveikiant įmonės pajamas.
- Įgyta daug patirties iš labiau patyrusių kolegų, iš vidinių mokymų, naujų technologijų nagrinėjimo.

4.2.2. Trūkumai

Praktikoje didelių trūkumų nepastebėta. Organizaciniai klausimai buvo sprendžiami sklandžiai, įmonės profesionalumo lygis aukštas. Tačiau galėčiau išskirti šiuos dalykus:

- Išdirbtų darbo valandų pildymas. Išdirbtą laiką reikėjo fiksuoti ne tik *„Clockify“* sistemoje, bet ir *„Jira“*. Kadangi vartotojų istorijos ir klaidos buvo registruojamos kliento projektų valdymo sistemoje *„Phabricator“*, iš kurios buvo perkeliamos į „Jira“, užtrukdavo susigaudyti, prie kurios užduoties ir kiek laiko reikia užpildyti.
- Kliento pateikiamos vartotojo istorijos ne visada atspindėjo visą vaizdą, nebuvo iki galo aišku, kokį funkcionalumą reikia įgyvendinti, todėl analizės etapas užtrukdavo ilgiau negu buvo planuojamas.

4.2.3. Pasiūlymai

Kadangi didelių trūkumų nepastebėta, tačiau galimi tam tikri procesų patobulinimai. Darbo įrankių integravimas, pavyzdžiui, užduotys, sukurtos kliento „Phabricator“ sistemoje, tam tikro įrankio dėka galėtų būti automatizuotai perkeltos į įmonės sistemą - „Jira“. Tai liečia ir darbo valandų pildymą. Kadangi „Clockify“ pildomas laikas gali būti nesusijęs su projektu, pavyzdžiui, vidiniai mokymai, renginiai ir t.t., galima būtų „Clockify“ sistemą integruoti su „Jira“, taip pildant laiką „Clockify“, jeigu laiko pildymo esybėje yra „Jira“ užduoties numeris - automatiškai tos užduoties laikas pildomas ir „Jira“ sistemoje.

Šaltiniai

[Bit18] „Bitė“. „bitė“ į išmanų klientų aptarnavimą investavo beveik 200 tūkst. eurų, 2018. URL: https://www.bite.lt/apie/ziniasklaidai/bite-i-ismanu-klientu-aptarnavima-investavo-beveik-200-tukst-euru.

[Bit20] „Bitė“. „bitė“ pristato televizijos paslaugą, dėl lietuvoje paskelbto karantino 3 mėn. leis naudotis nemokamai, 2020. URL: https://www.bite.lt/apie/ziniasklaidai/bite-pristato-televizijos-paslauga.