LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

Parametrizējams lietotāju apmierinātības analīzes rīks

BAKALAURA DARBS

Autors: **Kristaps Fabiāns Geikins**

Studenta apliecības Nr.: kg13036

Darba vadītājs: asoc. profesore, Dr. dat. Zane Bičevska

Rīga 2017

ANOTĀCIJA

Mūsdienās ir daudz un dažādi uzņēmumi, kas pieņem un izstrādā dažādu tipu pasūtījumus (piemēram, programmatūras izstrāde, grafiskā dizaina izveide) no klientiem, un šo pasūtījumu pārvaldību veic izmantojot digitālus risinājumus, kuros arī klienti var sekot līdzi padarītajam darbam, reģistrēt jaunus darbus, komunicēt ar pasūtītājiem un darbu izpildītājiem kā arī kaut kādā veidā novērtēt padarīto, lai parādītu savu apmierinātību.

Lieliem uzņēmumiem, kuriem ir vairāki desmiti vai pat simti ar vienlaicīgiem pasūtījumiem un projektiem, ir grūti izsekot visu savu klientu apmierinātībai, tāpēc ir nepieciešams risinājums, kas šo problēmu risinātu ātri, ērti un efektīvi.

Pētījuma rezultātā tika izanalizētas visas iesaistītās problēmas un noprojektēts risinājums, kas izmanto projektu pārvaldības rīku funkcionalitāti, lai šīs problēmas risinātu pēc iespējas vairāk automatizēti, kā arī tika izstrādāts strādājošs prototips noprojektētajam rīkam.

Atslēgvārdi: klientu apmierinātība, analīze, automatizācija, pasūtījumi, problēmu izsekošana, biznesa sistēmas, projektu pārvaldība, rīks

ABSTRACT

PARAMETERIZABLE TOOL FOR MEASURING USER SATISFACTION

Nowadays there are all kinds of companies which take orders from clients (for example, software development, graphical design creation) and manage these orders using digital solutions, in which the clients can also track the work being done, register new work that needs to be done, communicate with the people doing the work and in some way, rate the work has been done to show how satisfied they are.

For big companies which have tens or even hundreds of simultaneous orders and projects it is difficult to track the satisfaction of all their clients so a solution is needed which would solve this problem quickly, comfortably and effectively.

The result of this research paper is an analysis of the problems involved as well as a finished design of a solution, which solves the aforementioned problems using the functionality of project management tools as automatically as possible and a working prototype for the designed tool.

Keywords: customer satisfaction, analysis, automatization, orders, issue tracking, business systems, project management, tool

SATURS

DEFINĪCIJAS UN APZĪMĒJUMI

CSS (Cascading Style Sheets) – Stila lapu valoda, ko lieto, lai aprakstītu izskatu iezīmēšanas valodā veidotiem dokumentiem. Populārākais lietojums ir HTML un XHTML valodās veidotu tīmekļa lappušu izskata aprakstīšanai.

HTML (Hypertext Markup Language) – Iezīmēšanas valoda, kas ir izstrādāta tīmekļa lappušu un citas pārlūkprogrammā attēlojamas informācijas glabāšanai.

Tīmeklis (vispasaules jeb globālais tīmeklis) – Savienotu hipertekstu dokumentu sistēma, kurai piekļūst izmantojot Internetu.

Tīmekļa lietotne – Klienta-servera tipa lietojumprogrammatūra, kuras klients (lietotāja saskarne) izpildās tīmekļa pārlūkā.

Projektu pārvaldības rīks – Programmatūra, kas atvieglo projektu pārvaldību.

Darba biļete – Darba vienums jeb uzdevums projektu pārvaldības rīkā, kas raksturo kādu vienu darbu, kas ir jāizdara.

Versiju pārvaldība – Funkcionalitāte, kas ļauj izsekot un pārvaldīt dažādās versijās kādam digitālam vienumam (piemēram, datnēm).

Zināšanu bāze – Kolekcija ar rakstiem un informāciju par dažādām tēmām.

Ziņu dēlis (forums) – Tiešsaistes diskusiju vieta, kurā cilvēki var sarunāties viens ar otru sūtītu ziņu veidā.

Ganta diagramma – Joslu diagramma, kas ir paredzēta projekta grafiku ilustrēšanai un parāda katras projekta darbības vai uzdevuma sākuma un beigu laikus.

AGILE (Spējā izstrāde) – Pieeja programmatūras izstrādei, kura satur organizatorisku un metodoloģisku pasākumu kopumu ar mērķi nodrošināt ātru (laika ziņā), elastīgu (lietotāju prasību ziņā) un efektīvu (izstrādes ziņā) programmatūras projektu izstrādi un ieviešanu.

Sprints – Spējajā izstrādē definēts laika posms, kurš cikliski atkārtojas, un kurš sadala visu programmatūras izstrādi.

Lietotāju stāsts – Spējajā izstrādē izmantota pieeja gala lietotāju prasību aprakstīšanai vienkāršā un konkrētā veidā.

GIT – Versiju pārvaldības rīks, kas tipiski ir izmantots programmatūras izstrādei.

API (Lietojumprogrammas saskarne) – Iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums (bibliotēkas, servisi), kuru iespējams izmantot ārējiem programmatūras produktiem.

IIS – Elastīgs, drošs un ērti pārvaldāms tīmekļa serveris, kuru ir izstrādājis Microsoft, un kas strādā uz Windows operētājsistēmas.

UNIX - Kopa ar operētājsistēmām, kuras ir atvasinātas no oriģinālās AT&T Unix sistēmas, kuru sāka izstrādāt 1970. gados.

Server-puse – Operācijas, kuras izpilda serveris klienta-servera tipa programmatūrā datortīklos.

Klient-puse – Operācijas, kuras izpilda klients (parasti interneta pārlūks) klienta-servera tipa programmatūrā datortīklos.

MVC – Programmatūras izstrādes pieeja, kurā lietotne tiek sadalīta trīs, savstarpēji savienotās daļās (modeļos, skatos, kontrolieros), lai labāk atdalītu programmas iekšējo informācijas reprezentācija no tā kā informācija tiek attēlota lietotājam.

Vienību testēšana – Programmatūras testēšanas metode, ar kuru tiek testētas atsevišķas pirmkoda vienības, ar mērķi pārbaudīt vai tās strādā.

Atkarību injicēšana – Pieeja programminženierijā, kad viens objekts sagādā atkarības citam objektam, nevis objekts tās inicializē sev pats.

Servisu konteineris – Īpašs objekts, kas pārvalda visas servisu (atkarību) instances un tās sagādā tiem objektiem, kuri tās pieprasa.

Modelis – MVC pieejā, modelis apraksta objektu, kurā glabājas dati, kurus paņem kontrolieris un parāda skatā.

Skatumodelis – Atšķirībā no parastajiem modeļiem, kas parasti tiešā veidā raksturo kādu datubāzes objektu (tabulu), skatumodelis ir modelis, kas apraksta kaut kādus datus, kurus attēlo skatā, bet kas neapraksta kādu konkrētu datubāzes objektu.

Entītija – Entītija ir objekts datubāzē, kas apraksta kaut kādu lietu.

Serviss – Serviss ir jebkurš objekts, kuru pārvalda servisu konteineris. Tie ir neatkarīgi objekti, kas nav sasaistīti ar kādu konkrētu programmatūras daļu, kuros ir definēta funkcionalitāte, kuru var izmantot jebkurā vietā sistēmā, kur ir pieejams servisu konteineris.

Statiska funkcija – Klases funkcija, kuru ir iespējams izpildīt, bez nepieciešamas klases objekta inicializācijas.

Instances funkcija – Klases funkcija, kuru ir iespējams izpildīt, tikai ja tā ir inicializēta kā objekts.

Nosaukumvieta – Nosaukumvieta ir simbolu virkne, kuru izmanto lai organizētu dažādus objektus (piemēram, klases), tā lai uz tiem varētu atsaukties pēc nosaukuma.

UNIX laika zīmols – Laika pieraksta sistēma, kurā to uzrāda kā sekunžu skaitu kopš 1970. gada 1. janvāra pusnakts pēc GMT+0 laika zonas.

Regulārā izteiksme – Simbolu virkne, kas apraksta meklējamo izteiksmi. To izmanto programmēšanā, lai meklētu simbolu virknes.

Kešošana – Nesen izmantotu datu pagaidu saglabāšana, lai pie tiem varētu ātrāk piekļūt.

IEVADS

Kopš datoru izgudrošanas informācijas tehnoloģiju pasaule ir kļuvusi par milzīgu, miljardu vērtu industriju, kas vairāk vai mazāk ir saistīta ar visiem mūsu dzīves aspektiem. Datorizēti risinājumi ieņem nozīmīgu lomu gan mūsu personīgajās un sociālajās dzīvēs, atvieglojot tās vai nu padarot komunikāciju vieglāku, vai padarot mūsu ikdienas dzīvi ērtāku, daļu tās rutīnas nododot datoriem, gan arī biznesa pasaulē, kurā tiek investētas milzīgas naudas summas, lai ar datoru palīdzību padarītu dažādus procesus ātrākus un efektīvākus, kā arī lai veidotu inovatīvus risinājumus, kuri nebūtu iespējami bez datoru palīdzības, un ar kuriem uzņēmumu ienākumi tiek palielināti daudzkārtīgi.

Tā kā mūsdienās informācijas tehnoloģiju industrija ir uzaugusi tik milzīga un ir iesaistītas tik lielas naudas summas, ir it īpaši svarīgi biznesa pasaulē klientiem pārraudzīt kā notiek viņu vēlmju izpilde dažādos digitālajos risinājumos, tā pat kā jebkurā citā industrijā. Vai nu tā ir programmatūras izstrāde, vai varbūt bezlīguma grafiskā dizaina veidošana – ir daudz un dažādas digitālās projektu pārvaldības sistēmas, kurās klienti kaut ko pasūta, un ir izpildītāji šim pasūtījumam, kuri ir atbildīgi par darba izpildi. Ir daudzi uzņēmumi, kuri fokusējas uz kaut kāda veida pasūtījumu pieņemšanu un izpildi, un kuru visa peļņa ienāk no šādiem pasūtījumiem. Viena, diezgan liela problēma, kas rodas, kad šāda tipa uzņēmums kļūst pietiekami liels un pieņem pietiekami daudz pasūtījumus, ir ka paliek grūti izsekot šiem visiem pasūtījumiem un tieši konkrēti vienam no svarīgākajiem faktoriem – klientu apmierinātībai.

Tātad ir acīmredzama nepieciešamība pēc kāda risinājuma, kas šī nozīmīgā faktora mērīšanu padarītu vieglāku un ātrāku. Šī darba ietvaros šī problēma tika pētīta un tika meklēts labs, pietiekami vispārīgs risinājums, kuru varētu izmantot biznesa pasaulē, un tiks pētīts tas, kādam šim risinājumam būtu jābūt un kādas problēmas tam būtu jārisina.

Pirmajā nodaļā tiek apskatīts, kas ir projektu pārvaldība, un kā tajā iesaistītos procesus mēdz atvieglot projektu pārvaldības rīki, kā arī tas kādi šie rīki ir tirgū populārākie.

Otrajā nodaļā tuvāk tiek izpētīts kas ir klientu apmierinātība, kā to mēdz mērīt gan digitālajā, gan īstajā pasaulē, kā arī to, kādi ir populārākie risinājumi tās mērīšanai projektu pārvaldības rīkos.

Trešajā nodaļā ir aprakstīts darba laikā izstrādātais rīks ‘CustoMood’, kas ļauj automatizētā, netiešā veidā izmērīt klientu (vai jebkādu citu cilvēku) apmierinātību projektu pārvaldības (un arī cita veida) rīkos.

1. PROJEKTU PĀRVALDĪBA

1.1. Kas ir projektu pārvaldība?

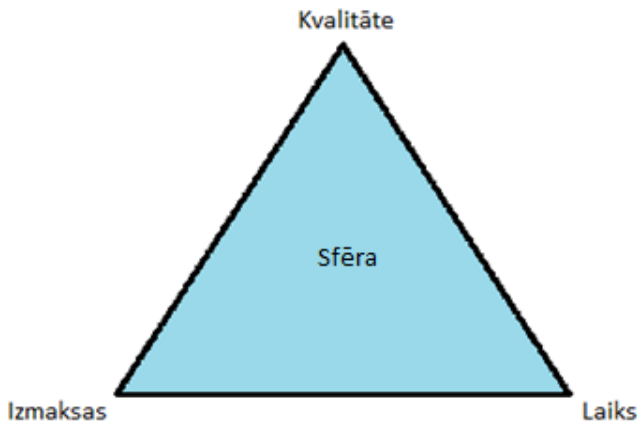
Projektu var definēt kā jebkādu procesu vai pasākumu kopumu, kuram ir noteikts sākums un beigas, kas ir unikāls tādā ziņā, ka nav kāda rutīnas darbība, un kuram ir kaut kāds mērķis [1].

Projektu pārvaldība ir koncepts, par kuru būtu jāzina jebkura projekta vadītājam, jo bez tā ir grūti nodrošināt veiksmīgu projekta izpildi. Projektu pārvaldību var definēt kā sekojošo piecu procesu izpildi:

1. Projekta inicializācija
2. Plānošana
3. Izpilde
4. Uzraudzība un kontrole
5. Noslēgšana

Veiksmīgi pārvaldīt projektu nav vienkārši, tāpēc ir speciāli izstrādāti kursi, kuros tas tiek mācīts, un ir cilvēki, kuru viss ikdienas darbs sastāv no projektu pārvaldīšanas un visu tajā iesaistīto projektu izpildes.

Projektu pārvaldībā ir definēts tāds koncepts kā "Projektu pārvaldības trijstūris", kas attēlo trīs galvenos ierobežojumus projektu pārvaldībā – kvalitāte, laiks, izmaksas [2].



1.1. att. **Projektu pārvaldības trijstūris**

Trijstūrī attēlots tas, ka jebkāda veida projekts ir atkarīgs no iepriekšminētajiem trim faktoriem, un tas, ka tie ir savstarpēji saistīti, piemēram, mēģinot samazināt izmaksas, pavisam noteikti tiks palielināts laiks un iespējams samazināta kvalitāte.

Lai gan projekti var būt dažādi, šī darba ietvaros vairāk tiek fokusēts uz IT projektiem, kā piemēram, programmatūras izstrāde.

1.2. Projektu pārvaldības rīku funkcionalitāte

Ir skaidrs kāpēc projekta pārvaldība ir nepieciešama, tāpēc ir arī nepieciešamība pēc produktiem un risinājumiem, kas projektu pārvaldību varētu atvieglot. Mūsdienu digitālajā pasaulē ir jau radīti daudz un dažādi produkti, kas tiecas projektu pārvaldību padarīt vienkāršāku un efektīvāku, kā piemēram JIRA un Trello, kuri ir aprakstīti nodaļā 1.3., bet kopumā to funkcionalitāti var reducēt uz vienu kopu, kuru pēc autora domām var veidot šādi [4] [5]:

- Sadarbība
- Komunikācija
- Plānošana un laika pārvaldība
- Informācijas un darba organizācija
- Izmaksu kontrole
- Analīze
- Projektā iesaistīto cilvēku pārvaldība

1.2.1. Sadarbība

Visi projektu pārvaldības rīki ņem vērā nepieciešamību sadarboties starp projektā iesaistītajiem cilvēkiem – gan projekta pasūtītājiem, gan projekta izpildītājiem. Šajos rīkos ir izstrādāta funkcionalitāte, kas ļauj lietotājiem ērti dalīties ar informāciju, datnēm, ļauj šo informāciju kaut kādā veidā apstrādāt un izmainīt, un tad nodot tālāk. Grūti atrast kādu projektu pārvaldības rīku, kurā nav datņu pārvaldība, iespējams pat ar versiju pārvaldību, kas

ļauj sistēmā glabāt datnes, kā arī kaut kāda veida čata un ziņu sistēma, kas ļauj lietotājiem vienam otram nodot datnes un citu informāciju.

Rīki bieži vien ļauj darba biļetēs norādīt atbildīgos cilvēkus, cilvēkus, kuriem jāziņo par projekta izmaiņām, cilvēkus, kas būs tieši tie, kas darbu izpildīs u.tml., kas ļauj šiem cilvēkiem savstarpēji vieglāk sadarboties.

Daudziem rīkiem ir kaut kāda veida ziņu dēlis vai rakstu sistēma, kurā cilvēki var rakstīt rakstus, kuri varētu palīdzēt pārējiem cilvēkiem. Tos bieži izmanto, lai vienviet varētu atrast visu svarīgāko informāciju par projektu.

Tātad projektu pārvaldības rīki ļauj sadarboties visiem projektā iesaistītajiem cilvēkiem, bez nepieciešamības izmantot kādus ārējos rīkus.

1.2.2. Komunikācija

Tā kā projektos mēdz būt daudz cilvēki un viņiem visiem vajag ērtu veidu kā komunicēt vienam ar otru, tad vēl viena svarīga sastāvdaļa no veiksmīga projekta pārvaldības rīka ir komunikācijas atvieglošana. Lai cilvēkiem nebūtu katram jāatceras visas iesaistītās personas un viņu telefona numuri, projekta pārvaldības rīkos parasti ir iebūvētas lietotāju sistēmas, kurās lietotāji var apskatīt citus lietotājus un informāciju par viņiem, kā arī ir implementēta kaut kāda veida čata un ziņu sistēma, kas ļauj cilvēkiem nekavējoties sazināties ar pārējiem projekta dalībniekiem tajā pašā rīkā, kurā notiek visa pārējā projekta pārvaldība.

Ja rīks atbalsta projekta plānošanu kaut kāda veida darba biļetēs, tad parasti arī ir implementēta komentēšana, kas ļauj projekta dalībniekiem runāt par konkrētiem darbiem iekšā šajās darba biļetēs.

1.2.3. Darba plānošana un laika pārvaldība

Lai sasniegtu projektā nostādīto mērķi, iekļaujoties nostādītajos termiņos, projektu ir nepieciešams saplānot, un var teikt, ka projekta plānošana ir viena no svarīgākajām projektu pārvaldības aktivitātēm. Lai šo procesu atvieglotu, projektu pārvaldības rīkos parasti ir implementēta funkcionalitāte, kas ļauj projektu sadalīt vairākās mazākās daļās – darba biļetēs,

kurām var norādīt termiņus, prioritātes, atbildīgās personas, kā arī pozīciju laikā relatīvi pret pārējām darba biļetēm.

Šāda veida darba sadalījums padara projektu daudz uzskatāmāku un izsekojamāku, jo katrā mirklī ir iespējams redzēt cik tālu projekts ir ticis, cik liels darbs ir izdarīts un cik vēl ir jādara, kā arī to vai projekts iekļausies termiņos.

1.2.4. Informācijas un darba organizācija

Ja rīks atļauj projekta dalībniekiem komunicēt un sadarboties vienam ar otru, ir svarīgi lai šo procesu rezultāts nepazūd un ir viegli atrodams un pieejams.

Daudzi izstrādātie projektu pārvaldības rīki satur zināšanu bāzes tipa funkcionalitāti, kas ļauj vienā centralizētā vietā katram projektam likt visu svarīgo informāciju.

Projektiem var būt galvenā lapa, kas kalpo kā "ievads" projektam, un kurā var būt aprakstīti projektā iesaistītie dalībnieki, viņu kontaktinformācija, termiņi un projekta apraksts.

Līdzīgi ir svarīgi arī organizēt projektā saturošo darbu, it īpaši, ja tajā ir iesaistīti daudz cilvēki. Parasti rīki atļauj projekta darbu sadalīt pa laika periodiem, pa atbildīgajām personām, prioritātēm u.tml.

1.2.5. Izmaksu kontrole

Kā jau tika minēts iepriekš, parasti projektu pārvaldības rīki ļauj kontrolēt darbiem atvēlēto laiku un termiņus, kas attiecīgi arī ļauj kontrolēt projekta izmaksas, bet daži rīki arī ļauj tiešā veidā sekot iztērētajam laikam ar hronometru, kuru ieslēdz un izslēdz strādājot pie kāda konkrēta darba, kā arī ir rīki, kuros tiešā veidā ir integrēta rēķinu izsniegšanas sistēma, kas atvieglo grāmatvedības darbu ar klientiem [3].

1.2.6. Analīze

Parasti organizācijas, kuras īsteno projektus, īsteno vairākus, nevis tikai vienu, tāpēc ir vērts gan projekta laikā, gan projektu noslēdzot, izanalizēt to, noteikt to cik veiksmīgs tas bija,

pārskatot dažādas metrikas. Projektu pārvaldības rīki šo analīzi padara vienkāršāku, veicot automātisku statistikas uzkrāšanu un grafiku ģenerēšanu.

Rīki ļauj ģenerēt dažāda veida grafikus, kurus var izmantot lai saprastu to, kā rit projekts, piemēram, Ganta diagrammas, iztērēto laiku sadalījumu pa darba uzdevumiem, veiksmīgi izpildīto darba biļešu sadalījumu pa projekta dalībniekiem u.tml. Parasti rīkiem nāk līdzi populārākie grafiku veidi, kā arī tie ļauj lietotājiem veidot jaunus grafiku veidus.

1.2.7. Projektā iesaistīto cilvēku pārvaldība

Lai gan mēdz arī būt projekti, kuros ir pilnībā horizontāls vadības stils, parasti pastāv kaut kāda vertikāla hierarhija, kurā virs katras projekta dalībnieku grupas ir vadītāji, tāpēc projektu pārvaldības rīki to ņem vērā un īsteno lietotāju sistēmu, kurā lietotājus var sadalīt grupās ar dažādām atļaujām.

Piemēram, atļaujas veidot jaunus darba uzdevumus vai veidot un apskatīt grafikus var dot tikai lietotājiem, kuri ir grupā "Vadītājs", kamēr parastie darbinieki drīkst tikai redzēt savus darba uzdevumus un pārvietot tos pa darbplūsmu.

1.3. Populārākie projektu pārvaldības rīki

Iepriekšējajā nodaļā tika aprakstīta funkcionalitāte, kas parasti tiek iekļauta visos projektu pārvaldības rīkos, bet šādi rīki ir daudz un dažādi, un katram ir kaut kādi aspekti, kas tiem liek atšķirties no pārējiem. Rīki mēdz atšķirties gan pēc piedāvāto funkciju klāsta, gan pēc cenas, gan arī pēc tā, cik intuitīva un ērti izmantojama ir to grafiskā saskarne. Tā kā arī projekti mēdz būt dažādi, tad nebūs vienmēr viens un tas pats rīks vislabākais visiem gadījumiem, un ir svarīgi izvērtēt iespējamos variantus atkarībā no konkrētām vajadzībām – lielam programmatūras izstrādes uzņēmumam pavisam noteikti vajadzības būs savādākas nekā mazam marketinga uzņēmumam.

Zemāk tiek apskatīti daži no populārākajiem projektu pārvaldības rīkiem.

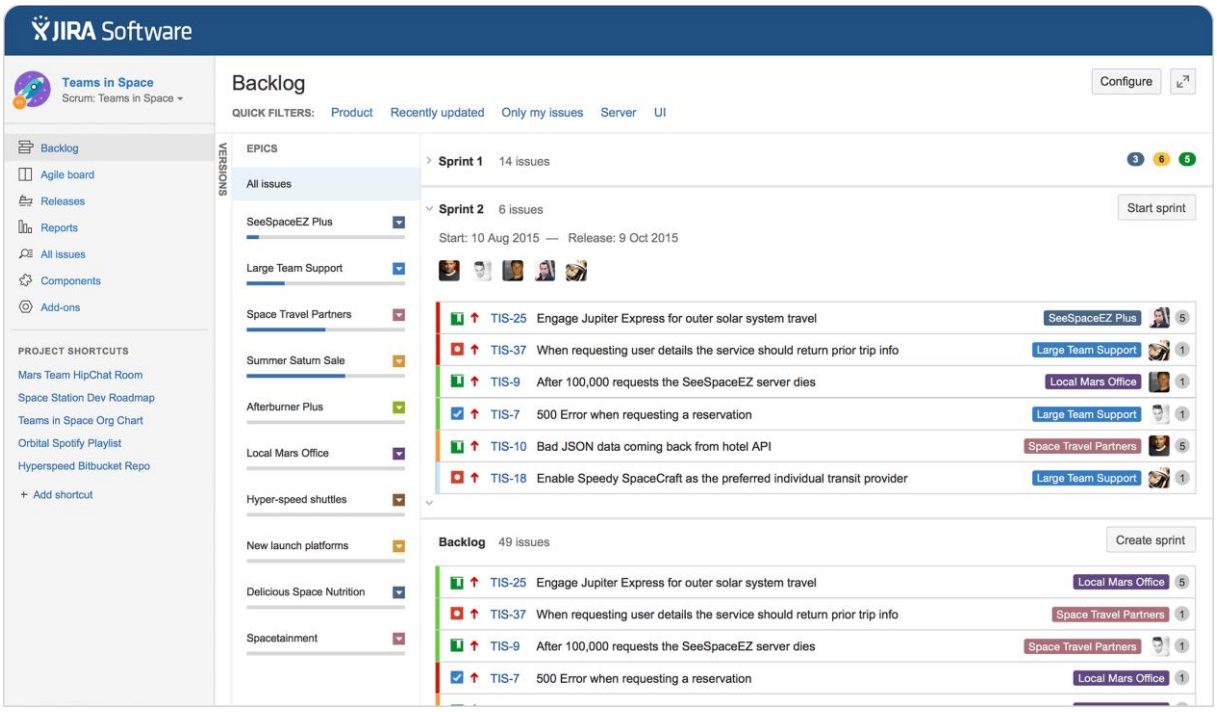
1.3.1. JIRA

Pavisam noteikti viens no populārākajiem rīkiem, kas ir tēmēts tieši uz programmatūras izstrādi ir Atlassian rīks JIRA [4]. Lai gan šo rīku var izmantot jebkāda veida projektiem, tas tika veidots galvenokārt priekš AGILE programmatūras izstrādes uzņēmumiem – tajā ir iestrādāta funkcionalitāte lietotāju stāstiem, darba uzdevumus var novērtēt pēc stāstu punktiem un sadalīt sprintos u.tml. Šis rīks ļoti labi atvieglo strādāšanu pēc AGILE metodoloģijas gan tad, ja pie tās strikti pieturās, gan tad, ja tā ir tikai izmantota par pamatu.

Tā kā rīks ir domāts tieši programmatūras izstrādei, tad attiecīgi tajā arī parādās funkcionalitāte, kas ņem to vērā. Piemēram, izstrādāto darbu var organizēt versijās, var izmantot dažādas darbplūsmas pēc kurām darbs tiek organizēts, tajā var integrēt dažādus populārus izstrādātāju rīkus (piemēram, var saintegrēt ar GIT repozitorijiem).

JIRA it īpaši labi strādā ar to pašu izstrādātāju izstrādātajiem produktiem Confluence un Bitbucket, kas ir Wiki tipa sadarbības sistēma un GIT versiju kontroles pārvaldības sistēma attiecīgi.

JIRA ir maksas rīks, ko var instalēt vai nu lokāli, vai arī izmantot kā Atlassian mākoņa maksas pakalpojumu [5].



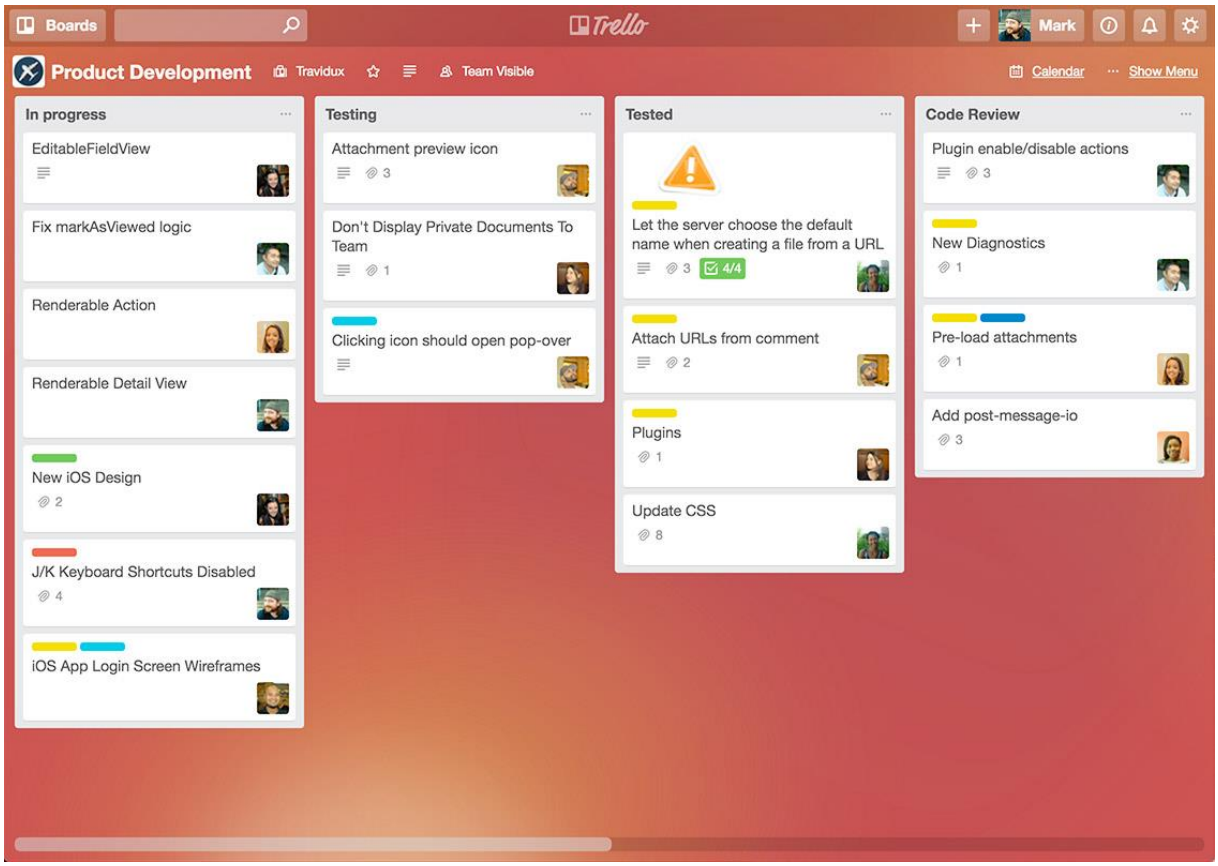
1.2. att. **JIRA projektu pārvaldības sistēma [4].**

1.3.2. Trello

Trello ir projektu pārvaldības rīks, kuru var izmantot gan pa brīvu, gan pa maksu, kas izceļas no pārējiem ar ļoti ērtu un intuitīvu vizuālo projekta reprezentāciju [6]. Atšķirībā no JIRA, Trello nav orientēts uz jebkāda konkrēta tipa projektiem un labi der jebkāda tipa projektu pārvaldībai. Arī Trello ļauj pielāgot darbplūsmu konkrētajām projekta vajadzībām un ļauj darbu organizēt dažādos veidos, to sadalot tādās fāzēs, kādas projektam ir nepieciešamas.

Līdzīgi kā JIRA, arī šajā rīkā pārsvarā darba organizācija notiek uz vizuālā darba dēļa, kurā darba biļetes var intuitīvi pārvietot starp fāzēm un prioritāšu līmeņiem.

Trello atbalsta dažādu lietotņu integrāciju un ir pieejams kā mākoņa serviss.



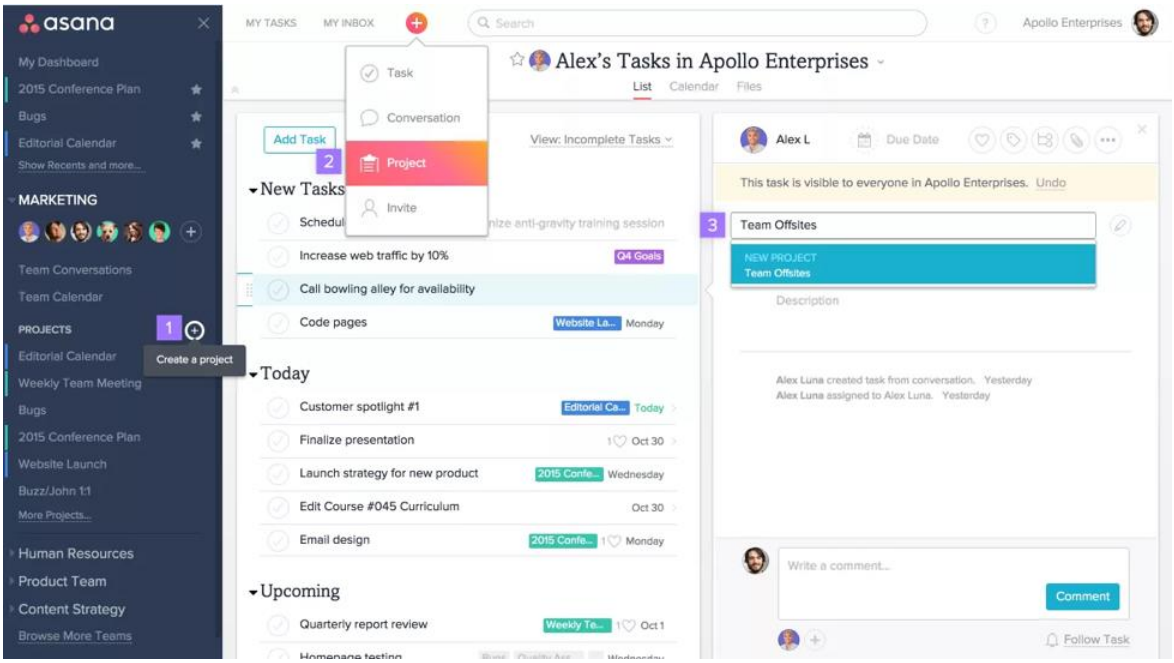
1.3. att. **Trello projektu pārvaldības rīks [6].**

1.3.3. Asana

Asana ir projektu pārvaldības rīks, kuru izstrādāja divi kādreizējie Facebook vadītāji, un kurš līdzīgi kā Trello ir domāts ne tikai programmatūras izstrādei, bet arī jebkāda cita veida projektiem [7] [8].

Asana atšķiras no konkurentu produktiem ar to, ka tas ir vienkāršāks un abstraktāks, kaut gan ar to arī var ļoti veiksmīgi pārvaldīt tik sarežģītus projektus kā AGILE programmatūras izstrādes projektus. Asana ir ļoti intuitīvs – projektus var sastādīt un pārskatīt līdzīgi kā iepriekšminētajos rīkos, uzdevumus sastādot pa fāzēm un prioritātēm uz darba dēļa, bet katram lietotājam rezultātā arī ir sastādīts saraksts ar uzdevumiem, kas ir jāizpilda, un tos izpildot tos vienkārši ieķeksē, bez vajadzības sekot līdzi visam darba laukam. Ja ir jāizvēlas rīks, kas būtu visērtākais visiem, arī lietotājiem, kuri pie tādiem nav pieraduši, tad Asana ir ļoti labs variants.

Asana mazas komandas var izmantot par brīvu, bet lai varētu izmantot visu funkcionalitāti ir jāmaksā [9].



1.4. att. **Asana projektu pārvaldības rīks [8].**

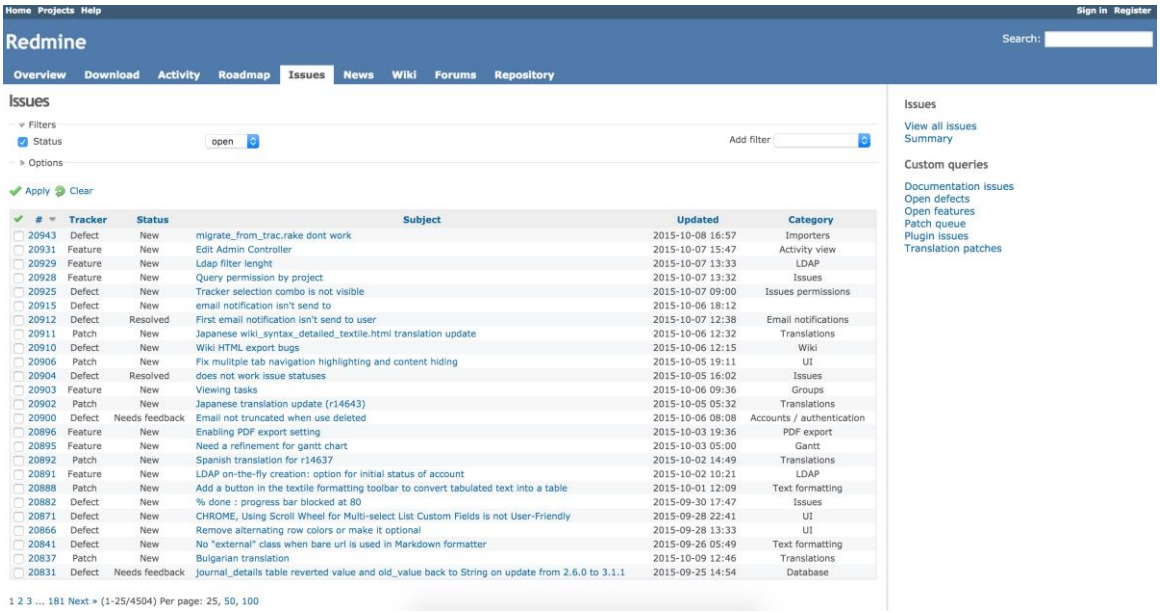
1.3.4. Redmine

Redmine ir rīks, kurš atšķirībā no iepriekš minētajiem ir vienīgais atvērtā pirmkoda rīks un tas vienīgais ir pilnībā par brīvu [10].

Šis risinājums piedāvā funkcionalitāti, kas ir ļoti līdzīga tai, ko piedāvā arī citi rīki tirgū, bet tā galvenā atšķirība ir, ka visa šī funkcionalitāte ir par velti. Piemēram, lai JIRA dabūtu arī Wiki tipa zināšanu bāzes par to būtu papildus jāmaksā, bet Redmine šāda funkcionalitāte jau ir iekļauta un ir par velti. Kopumā var sagaidīt visu to pašu ko no pārējiem rīkiem – problēmu izsekošanas sistēmu, grafikus, datņu pārvaldību, lietotāju sistēmu, laika izsekošanu utt.

Galvenais mīnuss šim rīkam ir tas, ka tā kā tas ir atvērtā pirmkoda un par brīvu, tad tai nav apmaksātas grafisko dizaineru un programmētāju komandas kas to uzturētu, un tas nozīmē, ka grafiskā saskarne ir relatīvi vienkārša un nav tik intuitīva, kā arī funkcionalitāte nav tik nopulēta.

Šis rīks der, ja visi projekta dalībnieki prot labi apieties ar datoru un tas cik lietotne ir intuitīva nav tik svarīgi.



1.5. att. **Redmine projektu pārvaldības rīks [10].**

2. KLIENTU APMIERINĀTĪBAS MĒRĪŠANA

Iepriekšējās nodaļās tika īsi aprakstīts, kas ir projektu pārvaldība, kā to var atvieglot ar dažādiem rīkiem, kā arī tika minēti daži populārākie rīki, kurus mēdz izmantot uzņēmumi, kuri pieņem pasūtījumus no klientiem.

Galvenais šī darba mērķis ir izpētīt labāko veidu kā, izmantojot projekta pārvaldības rīku funkcionalitāti un to kā klienti mēdz šos rīkus izmantot, nomērīt un klientu apmierinātību.

Klientu apmierinātības mērīšana pavisam noteikti nav viendimensionāla problēma, jo to ir jāpēta no dažādiem skatu punktiem un jārod atbildes uz tādiem jautājumiem kā "kas ir klientu apmierinātība?" un "kā to var automatizēti nomērīt?".

2.1. Kas ir klientu apmierinātība?

Klientu apmierinātību var definēt dažādi, bet viens no veidiem kā to definēt ir – apmierinātība ar kāda uzņēmuma produktiem un/vai servisiem, vēlme tos izmantot atkal un ieteikt citiem [11] [14].

2.1. Kā nomērīt klientu apmierinātību?

To kā nomērīt klientu apmierinātību ir jau bijis daudz pētīts ne tikai IT sfērā, bet jebkurā industrijā, kurā ir jādarbojas ar klientiem. Tā ir diezgan svarīga marketinga daļa, ar kuru nodarbojas daudz cilvēku, un kura ir aprakstīta un izpētīta daudz un dažādos veidos, kurus katrs apraksta savādāk. Tā kā tas ir ļoti subjektīvi, tad nav viena konkrēta veida kā nomērīt klientu apmierinātību, bet to var fundamentāli pārbaudot saņemot atbildes no klienta par sekojošajiem faktoriem [11] [12] [13]:

- Kopējā apmierinātība - Šis faktors apraksta klienta kopējo apmierinātību ar produktu vai servisu, kuru viņš ir izmantojis.
  Jautājuma piemērs: Kopumā, cik apmierināts jūs esat ar mūsu restorānu?

- Lojalitātes mērījums – Šis apraksta iespējamību, ka klientam zīmols (uzņēmums) patīk, viņš atgriezīsies atkal un ieteiks zīmolu citiem
  Jautājuma piemērs: Vai jūs ieteiktu mūsu restorānu ģimenei un draugiem?
- Emocionālā un kognitīvā apmierinātība – Šis apraksta klienta emocionālo saikni ar produktu, to kādas emocijas tas izraisa, kā arī klienta spriedumu par to cik objektīvi noderīgs bija produkts
  Jautājuma piemērs: Cik apmierināts jūs esat ar mūsu ēdiena garšu? Cik svarīga jums ir ēdiena garša izvēloties mūsu restorānu?
- Atkārtotas pirkšanas nodoma mērījums – Šis apraksta iespējamību, ka klients produktu vai servisu nopirks atkārtoti.
  Jautājuma piemērs: Vai jūs plānojat atgriezties uz mūsu restorānu nākamo 30 dienu laikā?

Uzzināt informāciju par šiem četriem faktoriem var uzdodot klientam specifiskus jautājumus, kuriem piemēri tika minēti augstāk, un izvērtējot atbildes var iegūt labu sapratni par to cik klients ir apmierināts ar produktu vai servisu.

2.1.1. Kā apmierinātību automatizēti nomērīt netieši?

Sarežģītāk ir izmērīt klienta apmierinātību ar pasūtītāju rīkos, kuros šādi jautājumi netiek uzdoti, un kuros apmierinātību jāmēģina izmērīt no tā kā klients izmanto rīku. Galvenie veidi kā to ir iespējams darīt ir sekojoši:

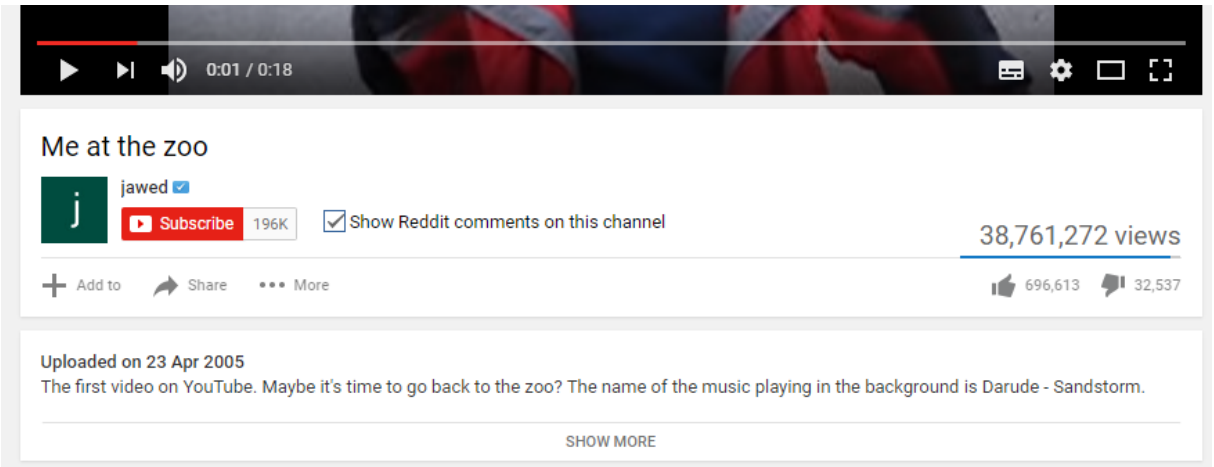
- Analizējot klienta darbības rīkā (pogu spiešana, staigāšana pa lapām/logiem)
- Analizējot to cik bieži, ilgi un kad klients izmanto rīku
- Analizējot klienta rakstīto tekstu

Visefektīvākais veids kā to darīt pavisam noteikti ir analizējot klienta rakstīto tekstu, jo pārējie veidi var tikai aptuveni un iespējams kļūdaini pateikt kaut ko par klienta apmierinātību. Pārējos veidus var izmantot kopā ar teksta analīzi, lai iegūtu labākos rezultātus.

2.1.2. Kā apmierinātību automatizēti nomērīt tieši?

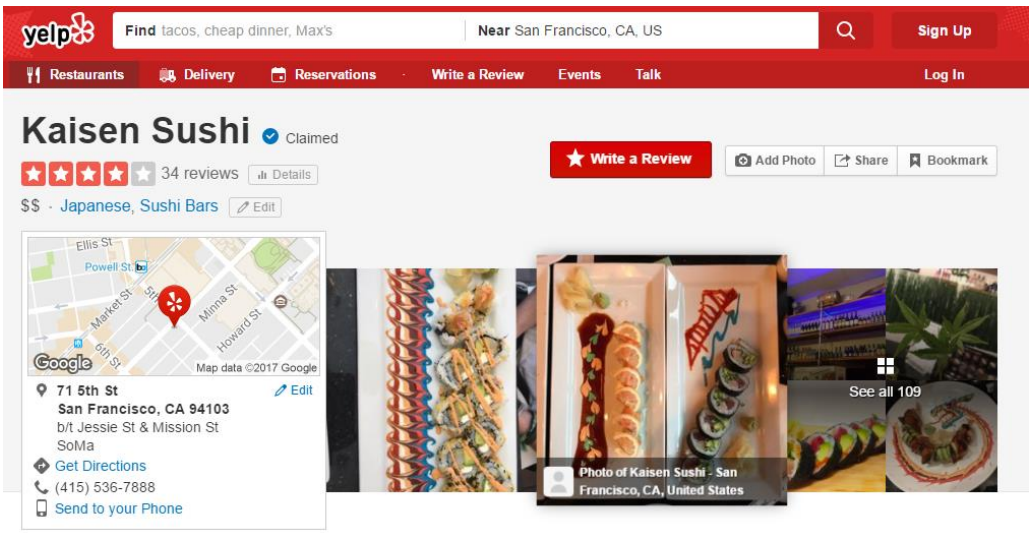
Bez iepriekšminētajiem četriem faktoriem, pēc kuriem var veidot jautājumus un tos automatizēti uzdot rīkos, ir arī citi veidi kā apmierinātību mēdz mērīt. Tos mēs redzam ikdienā visur – no rakstiem sociālajos tīklos, līdz video ierakstiem tādās vietnēs kā YouTube, līdz lietotnēm mūsu telefonu lietotņu veikalos. Zemāk aprakstīti populārākie veidi kā to mēdz darīt:

- **Īkšķis uz augšu / īkšķis uz leju (un citas sistēmas, kur ir viena negatīva un viena pozitīva vērtība)** – Viena no visbiežāk sastopamākajām sistēmām; diezgan primitīvs, bet precīzs veids kā noteikt apmierinātību vienkāršā līmenī. Šādu sistēmu var novērot vietnē YouTube, piemēram.



2.1. att. **"Īkšķis uz augšu / īkšķis uz leju" novērtēšanas sistēma lapā YouTube [15].**

- **Vērtējums 5 zvaigznēs (un citas sistēmas, kur ir vērtējums no 1 līdz 5)** – Šī sistēma jau dod precīzāku novērtējumu par to cik apmierināts ir lietotājs, jo sistēma jau ļauj lietotājam novērtēt produktu/servisu 5 ballēs. Lai gan skaitlis 5 liekas patvaļīgi izvēlēts un tik pat labi varētu likt 10 vai 13 vai jebkāda cita skaita balles, cilvēki ir pieņēmuši 5 balles kā optimālāko baļļu skaitu, jo pie lielāka baļļu skaita lietotājiem paliek grūtāk izvēlēties un starpība starp katru balli paliek arvien mazāka. Šādu sistēmu var novērot vietnē Yelp, piemēram.



2.2. att. **Piecu zvaigžņu vērtēšanas sistēma lapā Yelp [16].**

- **Radīto emociju novērtējums** – Šī sistēma tiek bieži vien izmantota tieši biznesa pasaulē iekš apmierinātības novērtējuma anketās, kur klientiem tiek piedāvātas vairākas emocijas, no kurām viņam ir jāizvēlas tā, kuru viņš jūt visvairāk. Šādu sistēmu arī izmanto sociālais tīkls Facebook, atļaujot cilvēkiem novērtēt rakstus atkarībā no tā kā viņi jūtās.

2.3. att. **Emociju novērtējuma sistēma lapā Facebook [17].**

2.2. Klientu apmierinātības novērtēšanas funkcionalitāte projektu pārvaldības rīkos

Tā kā šī darba mērķis ir izpētīt veidu kā novērtēt klientu apmierinātību tieši projektu pārvaldības sistēmās, kuras klienti mēdz izmantot, tad ir jāizpēta vai ir kādi esošie risinājumi, un ja ir, tad cik labi tie strādā un cik noderīgi tie ir.

Risinājumi tiks novērtēti pēc sekojošajiem kritērijiem:

- Cik precīzi klientu apmierinātība tiek novērtēta?
- Cik automatizēta ir šī funkcionalitāte?
- Cik vispārīgs ir apmierinātības novērtēšanas rīks (vai var izmantot pie dažādām sistēmām)?
- Kādas ir izmaksas šim rīkam?

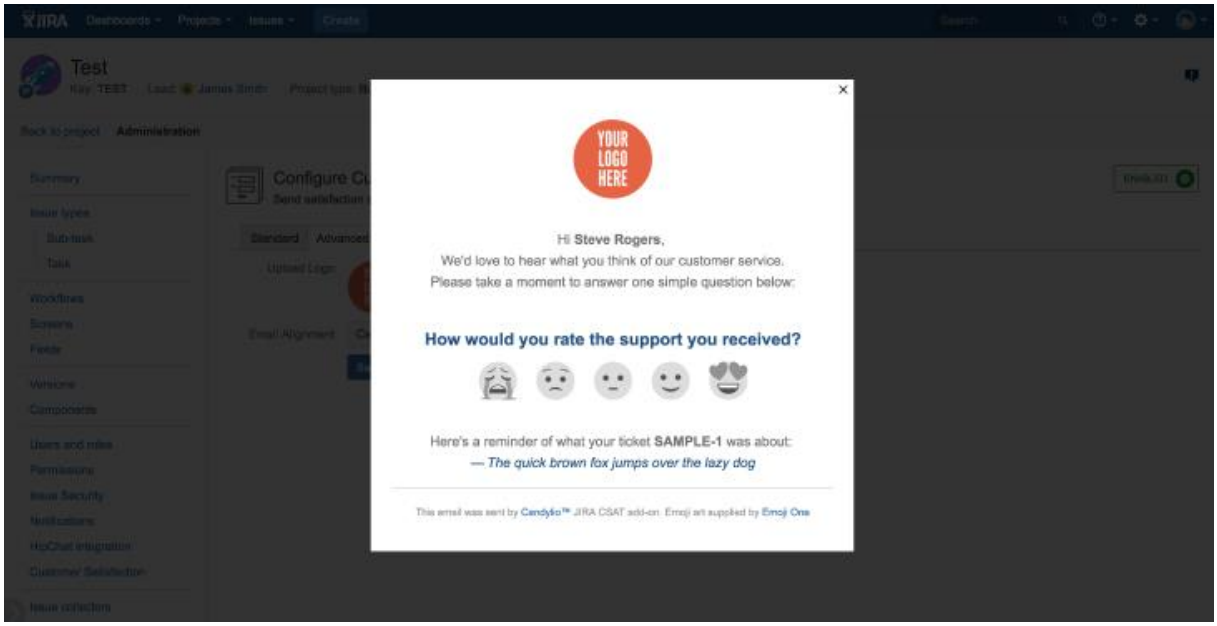
2.2.1. Customer Satisfaction Survey for JIRA

JIRA sistēmas papildinājumu tirgū tiek piedāvāts papildinājums "Customer Satisfaction Survey for JIRA" [18]. Šis papildinājums ļauj klientiem novērtēt konkrētas darba biļetes, un rezultātā šie dati tiek apkopoti tā, ka pasūtītājs var ērti apskatīt statistiku un grafikus par to, kā rit darbs.

Šis rīks atļauj klientu apmierinātību novērtēt ļoti precīzi, jo ļauj padarīto darbu novērtēt gan ar apmierinātības skalu, gan ar tekstuāliem komentāriem, kā rezultātā ir ļoti viegli pateikt cik apmierināts ir klients ar padarīto darbu. To kādai jābūt aptaujai un kam tajā jāatrodas var mainīt administrators.

Mīnuss rīkam gan ir tas, ka funkcionalitāte nav vispār automatizēta – klientam ir īpaši jānovērtē darbs un jāaizpilda anketa, lai viņa apmierinātība tiktu novērtēta. Pasūtītāja pusē gan visas anketas tiek apkopotas un dati tiek parādīti uzskatāmi grafiskā veidā.

Šī rīka cena ir atkarīga no JIRA sistēmas lietotāju skaitu un tā svārstās no 10 USD līdz pat 5000 USD, ja sistēmā ir pāri pa 10,000 lietotājiem. Rīks nav vispārīgs un strādā tikai uz JIRA projektu pārvaldības sistēmas instalācijām.



2.4. att. **"Customer Satisfaction Survey for JIRA" rīks [18].**

2.2.2. Redmine Helpdesk

Helpdesk ir papildinājums projektu pārvaldības rīkam Redmine, kas to pārvērš par atbalsta tipa sistēmu, kurā var veidot darba biļetes kā problēmas, kurām ir piesaistīti klienti, kuri var novērtēt padarīto darbu [19].

Klientam tiek dota iespēja parādīt savu apmierinātību pie katra sava komentāra, kur viņš var to norādīt ar emocijām "bēdīgs", "neitrāls" vai "priecīgs". Šie visi dati tiek apkopoti un tos var apskatīt grafikos.

Arī šajā rīkā novērtēšana nav automatizēta, jo to klientam jādara manuāli liekot novērtējumus pie katra komentāra darba biļetē, kas nozīmē, ka bieži vien šī funkcionalitāte netiks izmantota.

Šis papildinājums tika konkrēti veidots Redmine projektu pārvaldības rīkam, to var nopirkt sākot no 299 USD, un tā cena ir atkarīga no lietotāju skaita.

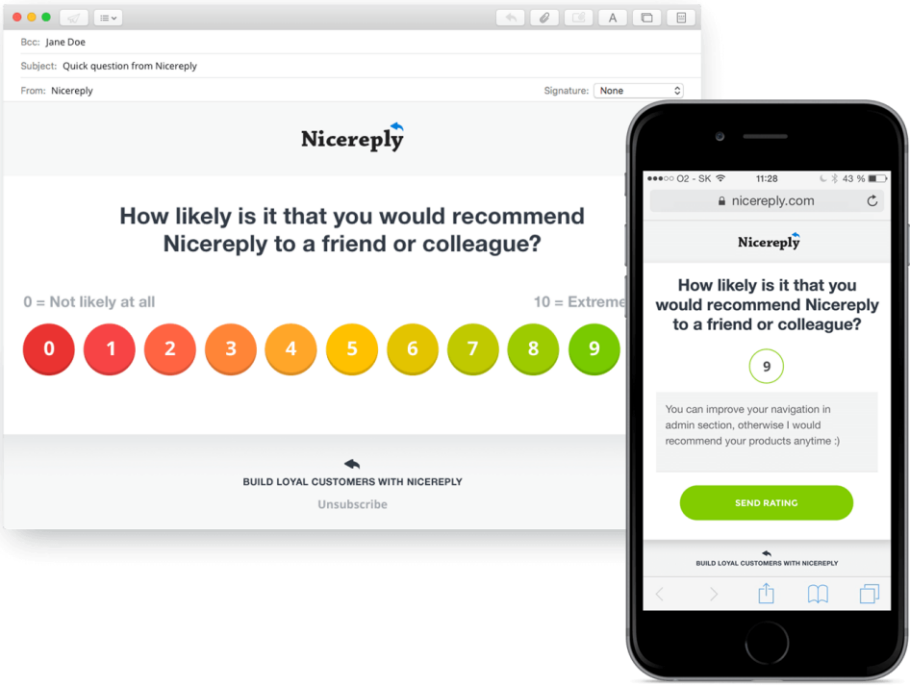
2.2.3. Nicereply

Nicereply ir klientu apmierinātības novērtēšanas rīks, kas ir izstrādāts tā, lai tas strādātu ar dažādām sistēmām. Tā centrālā funkcionalitāte ir klientu apmierinātības anketu izsūtīšana un pārvaldība dažādām projektu pārvaldības sistēmām [20].

Arī šajā rīkā funkcionalitāte nav automatizēta, jo rīks paļaujas uz anketām, kuras lietotājiem ir jāaizpilda, kurās tiek norādīta apmierinātība teksta veidā kā arī ar "emociju" novērtējumu. Visu anketu dati tiek apkopoti vienviet un par tiem var apskatīt statistiku un dati ir arī attēloti grafikos.

Šis rīks ir īpaši atšķirīgs no citiem ar to, ka tas ir vispārīgs un sasaistās ar dažādām sistēmām. Lai gan oficiālas integrācijas ar iepriekšējās nodaļās minētajām sistēmām nav, ir API, kas ļauj šādas integrācijas izveidot.

Nicereply maksā sākot no 89 USD, un tā cena ir atkarīga no sistēmas lietotāju skaita.



2.5. *att.* **Nicereply klientu apmierinātības rīka izsūtītais e-pasts [20].**

2.2.4. CustomerThermometer

CustomerThermometer ir rīks, kas līdzīgi kā Nicereply ir veidots, lai veidotu, pārvaldītu un izsūtītu klientu apmierinātības anketas vai nu saintegrējot to kopā ar kādu projektu pārvaldības (vai cita veida) sistēmu, vai arī bez [21].

Šis rīks klienta apmierinātības novērtēšanu neautomatizē, bet dod iespēju klientiem to manuāli pievienot vai nu kaut kur sistēmā, ar kuru tas saintegrējas, vai arī anketās, kuras klientiem var būt izsūtītas uz e-pastu vai arī atvērtas izmantojot personalizētās saites.

Šim rīkam ir iespējamas integrācijas ar dažādām sistēmām, ieskaitot projektu pārvaldības rīkiem, kaut gan to skaits ir mazāks nekā rīkam Nicereply. CustomerThermometer arī piedāvā API, kas atļauj veidot integrācijas ar jebkuru sistēmu.

Rīks maksā sākot no 29 USD un tā cena ir atkarīga no lietotāju skaita.

2.2.5. Secinājumi par rīkiem

Meklējot klientu apmierinātības mērīšanas rīkus, kurus varētu izmantot kopā ar darba biļešu bāzētajām projektu pārvaldības sistēmām, tika atrasti daži vispārīgi rīki un daži rīki, kuri der tikai konkrētām sistēmām. Redzams ir tas, ka tikai retajai sistēmai ir papildinājumi, kas šādu funkcionalitāti sniedz – no tām, kuras tika aprakstītas nodaļā 1.3., klientu apmierinātības mērīšanas papildinājumus varēja atrast tikai divām.

Lai gan eksistē arī vispārīgi rīki, kurus var teorētiski saintegrēt ar jebkuru sistēmu, tiem gatavi risinājumi iepriekšminētajām sistēmām nebija, un nāktos integrāciju izstrādāt pašam.

Visiem atrastajiem rīkiem galvenā problēma bija tā, ka tie nebija automatizēti un visi paļāvās uz manuālu anketēšanu un apmierinātības novērtēšanu, kas darbā pētīto problēmu nerisina.

Tātad secināms ir tas, ka vispārīgs, automatizēts klientu apmierinātības mērīšanas rīks darba biļešu bāzētajām projektu pārvaldības sistēmām neeksistē.

3. PARAMETRIZĒJAMS KLIENTU APMIERINĀTĪBAS MĒRĪŠANAS RĪKS - CUSTOMOOD

Iepriekšējās nodaļās tika izpētīti un izanalizēti esošie tirgus risinājumi klientu apmierinātības mērīšanai, bet rezultātā tika izsecināts, ka nav neviena rīka, kas atbilstu šajā darbā definētajām prasībām.

Daļa no šī darba ir projektējums rīkam, kas vislabāk risinātu problēmu – kā automatizēti un pēc iespējas vispārīgi mērīt klientu apmierinātību projektu pārvaldības rīkos?

3.1. Galvenās problēmas un vērā ņemamie faktori

3.1.1. Automatizācija

Kā iepriekš tika minēts, ir svarīgi, lai risinājums ļautu klientu apmierinātību mērīt automatizēti. Ir tirgū pieejami vairāki rīki, ar kuriem var mērīt klientu apmierinātību, bet tie visi paļaujas uz klienta iesaistīšanos mērīšanas procesā – anketu izpildi vai vērtējuma došanu jebkādā citā tiešā veidā.

Rīkam "CustoMood" būtu jādarbojas savādāk, uz cita principa pamata, nevis liekot lietotājiem tiešā veidā veikt vērtējumu, bet analizējot darbības, ko lietotājs jau sistēmā veic, un no tām izsecinot klienta apmierinātību, līdzīgi kā tas tika aprakstīts nodaļā 2.1.1.

3.1.2. Abstrakcija / vispārināšana

Lai gan rīks ir domāts konkrēti projektu pārvaldības sistēmām, tam būtu jāstrādā ar pēc iespējas vairāk, kā arī būtu jābūt iespējai to lietot ar pavisam cita tipa sistēmām, pat tādām, kurām nav nekāda sakara ar projektu pārvaldību. Šo rīku būtu jāvar saintegrēt ar jebkāda tipa sistēmu, kuru izmanto kaut kāda cilvēku grupa ("klienti"), kuru apmierinātību vajag izmērīt.

Tātad nepietiek ar to, ka rīks strādā tikai ar sistēmu JIRA, vai sistēmu Trello. Tam jābūt taisītam tā, lai to varētu saintegrēt kopā ar jebkādu sistēmu, kura atbilst iepriekšminētajiem nosacījumiem.

Svarīgi minēt, ka tas nenozīmē, ka rīkam automātiski ir jāstrādā ar visām iepriekšminētajām sistēmām, bet gan to, ka rīkam jābūt papildināmam un sakonfigurējam, bez papildus nepieciešamas rīka izstrādātāja (manas) iejaukšanās. Jābūt iespējamībai šīs integrācijas izveidot kādai trešajai pusei, piemēram, tai izstrādājot papildinājumu, kas sasaistās ar šo rīku. Šajā gadījumā rīkam jābūt gatavai un funkcionālai papildinājumu saskarnei.

3.1.3. Rezultātu precizitāte

Tā kā rīks veiks netiešu apmierinātības analīzi, tad ir nepieciešams pievērst uzmanību rezultātu precizitātei. Ja vērtēšana notiktu tieši – klients savu vērtējumu tiešā veidā norādītu pats, piemēram, ieliekot vērtējumu baļļu skalā, tad precizitātei īpaša uzmanība nebūtu jāpievērš, bet kā tika minēts iepriekš, šim rīkam apmierinātība jāveic netieši, kas nozīmē, ka no darbībām, kas kvantificējami tiešā veidā neko par apmierinātību nepasaka, ir jāiegūst apmierinātības vērtējums. Tas kāda tipa darbības varētu mērīt ir aprakstīts nodaļā 2.1.1., bet tas kā tās analizēt pēc iespējas precīzāk ir jāizpēta tuvāk.

3.1.4. Rezultātu attēlojums

Rīkam ir jābūt uztaisītam tā, ka tas rezultātus pasniedz lietotājam ērtā un uzskatāmā veidā. Nepietiek ar to, ka dati tiek kaut kā ievākti un saglabāti datubāzes tabulās – tiem ir jābūt organizētiem tā, lai šī rīka izmantošana būtu lietotājam intuitīva un neprasītu zināšanas vairāk par to kā apieties ar standarta lietojumprogrammatūru.

Informācijai jābūt labi organizētai un pēc iespējas koncentrētākai tā, lai lietotājs var ātri, bez nepieciešamības apskatīt lielus apjomus ar datiem, novērtēt situāciju jebkurā konkrētajā brīdī par to kāda ir klienta apmierinātība.

Lai šo mērķi sasniegtu, ir ieteicams izmantot vizuālus elementus, piemēram, grafikus, kurus var viegli konfigurēt, lai redzētu nepieciešamo informāciju.

3.1.5. Rīka darbības prasības

Izstrādājot rīku ir svarīgi ņemt vērā to, kā šis rīks parasti tiks izmantots, uz kādām sistēmām tas tiks instalēts un kā to radīt tā, lai tā prasības būtu pieņemamas sistēmu administratoriem.

Jāņem vērā tas, ka sistēmas ar kurām šis rīks būs sasaistīts gandrīz vienmēr būs tīmekļa lietotnes. Vērojot programmatūras izstrādes ekosistēmu šobrīd ir redzams, ka, tā kā tīmekļa programmatūrai ir ļoti daudzi plusi, kas nav darbvirsmas lietotnēm (lietotājam tās nav jāinstalē, tās resursus patērē tikai uz servera, ir vieglāk izstrādāt, jo izstrāde notiek vienādi visām operētājsistēmām), pārsvarā visa jaunā programmatūra ir tīmekļa bāzēta un reti vairs programmatūru izstrādā darbvirsmām. Visas iepriekšējās nodaļās minētās sistēmas (JIRA, Trello un pārējās) ir tīmekļa lietotnes, kuras atrodas uz kāda servera, un kuras izmanto lietotāji, savienojoties ar šo serveri, nevis, piemēram, darbvirsmas lietotnes.

Zinot to, ir jāņem vērā tas kāda ir pašreizējā tīmekļa lietotņu ekosistēma un to kādas tajā ir tendences, piemēram, tas kāda tipa serveri tiek izmantoti visbiežāk. Netcraft 2017. gada aprīļa apskate norāda uz to, ka tikai ~10% no aktīvajām tīmekļa vietnēm ir uzstādītas uz Microsoft IIS tipa serveriem, ~45% tīmekļa vietnes ir uzstādītas uz Apache serveriem un ~19% uz nginx serveriem [22]. No šiem populārākajiem serveru tipiem, IIS ir vienīgais, kas nestrādā uz Unix bāzētajām operētājsistēmām[1] un pārējie strādā vairāk vai mazāk uz visām, kas izskaidro to kāpēc IIS serveri tiek izmantoti tik maz. Tātad, lai rīku "CustoMood" varētu izmantot pēc iespējas vairāk sistēmu administratori, rīku vajag veidot kādā no valodām, kas strādā gan uz Unix bāzēto operētājsistēmu serveriem, gan Windows serveriem, kur automātiski atkrīt .NET valodas.

Tā kā rīkam arī jābūt funkcionālai papildinājumu saskarnei, tad arī ir jāņem vērā populārākās tīmekļa lietotņu izstrādes valodas, tā lai būtu pēc iespējas vairāk izstrādātāju, kas varētu izstrādāt nepieciešamos integrāciju papildinājumus. Pēc TIOBE indeksa datiem 2017. gada maijā, vispopulārākās tīmekļa lietotņu izstrādes server-puses valodas, kuras strādā gan uz Unix bāzētajām operētājsistēmām, gan Windows operētājsistēmām, ir Java, Python, JavaScript, PHP, Ruby (kārtojot popularitātē no lielākās uz mazāko) [24].

[1] Microsoft relatīvi nesen ir izlaidis .NET ietvara versiju ".NET Core", kas strādā arī uz Unix operētājsistēmu serveriem, bet tā kā šī versija vēl ir ļoti jauna un daudzas bibliotēkas vēl to neatbalsta, tad tā šajā darbā netika ņemta vērā [23].

3.2. Lietotnes funkcionalitāte

3.2.1. Par lietotni ‘CustoMood’

Šī darba ietvaros tika izstrādāts rīks, kas risina iepriekš apskatītās problēmas – ‘CustoMood’. Šis rīks ir tīmekļa lietotne, kura mēra lietotāju apmierinātību, analizējot jebkādas datu kopas, kādas spēj atgriezt ar lietotnes adapteru palīdzību.

Adapteru sistēma nodrošina vieglu veidu kā sistēmā reģistrēt saskarnes starp pašu ‘CustoMood’ lietotni un jebkādu datu kopu, piemēram, kādas projektu pārvaldības sistēmas klientu komentāru datubāzi.

Lietotnes administrators var ieinstalēt (vai izstrādāt) sev nepieciešamos adapterus, piereģistrēt projektus, kas šos adapterus izmanto, iestatīt projektiem adapteros definētos iestatījumus atbilstoši vajadzībām un tad parastie lietotāji var jebkurā brīdī atlasīt datus par apmierinātību, kuri tiek automātiski analizēti un par kuriem informācija tiek parādīta uzskatāmi un intuitīvi grafika veidā.

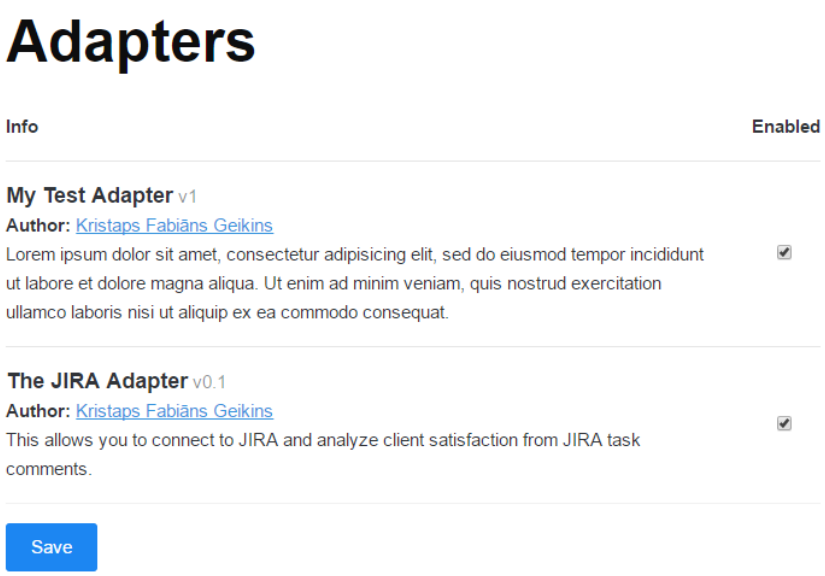
3.2.2. Adapteru sistēma

Kā tika minēts iepriekšējā nodaļā, adapteru sistēma ļauj lietotnē reģistrēt vairākas, jebkāda tipa saskarnes starp pašu lietotni un kādu datu kopu, piemēram, datubāzi ar klientu atsauksmēm. Lai gan lietotne ir domāta pārsvarā sasaistei projektu pārvaldības sistēmām, piemērām sistēmām JIRA un Trello, šī adapteru funkcionalitāte ir tik abstrakta, ka ļauj sasaistīties ar jebkuru sistēmu, ja no tās kaut kā ar PHP skriptu var nolasīt datus. Vienīgais, kam ir jābūt, lai sistēma varētu no kādas datu kopas iegūt datus, ir jābūt izstrādātam adapterim.

Tas kā adapteru sistēma ir izstrādāta ļauj adapterus viegli izstrādāt pilnībā neaiztiekot pašas ‘CustoMood’ lietotnes pirmkodu, tik izveidojot vienu PHP klasi un ievietojot to atbilstošajā mapē lietotnē.

Nav arī nepieciešama nekāda sarežģīta instalācija, jo adaptera klasi ieliekot pareizajā mapē, lietotne to uzreiz atpazīst un integrē sistēmā un tos uzreiz var lietot, veidojot projektus.

Lai demonstrētu adapteru funkcionalitāti, lietotnei tika izstrādāti divi gatavi adapteri – testa adapteris un adapteris JIRA sistēmai, un ja tie nav izņemti no lietotnes instalācijas, tad tos abus var redzēt administrācijas panelī adapteru sadaļā.



3.1. att. **Bakalaura darba laikā izstrādātie adapteri.**

Kā ir redzams attēlā 3.1., šos adapterus var arī izslēgt, kas tos neatinstalēs, bet tos nevarēs izmantot jaunos projektos un esošie projekti, kas tos izmanto, būs atslēgti.

3.2.2.1. Kā tā strādā?

Kā tika minēts iepriekš Adapteris ir PHP klase, kurai var nākt līdzi vairākas papildus klases, ja ir nepieciešams (piemēram, ja ir daudz koda, kuru vajag sadalīt mazākās daļās, lai kods būtu pārskatāmāks un labāk organizēts), kurā ir nodefinētas vairākas statiskas un dažas instances funkcijas.

Statiskās funkcijas ir adaptera metadatu iegūšanai, kuras atļauj to darīt pašu klasi neinicializējot, kas savukārt strādā ātrāk un neizmanto papildus sistēmas resursus, jo ir vairākas vietas sistēmā, kur ir nepieciešams tikai iegūt metadatus, bet pašu adapteri neizmantot. Ja adapteris ir uztaisīts ļoti liels ar daudzām operācijām un objektiem, tad tā instances izveidošana katrā vietā varētu sistēmai prasīt daudz resursus padarot to lēnāku.

Tā kā adapteru metadati (nosaukums, apraksts, informācija par autoru utt.) glabājas pašās adapteru klasēs nevis datubāzē, tāpēc izmainot tās, jaunā informācija uzreiz būs redzama arī lietotnē.

```php
/**
 * Unique ID of this adapter
 * @return string
 */
public static function getId(): string
{
    return "kristaps_geikins_mytestadapter";
}

/**
 * Display name that will be used in CM (can be null, in which case the ID will be used)
 * @return string
 */
public static function getDisplayName(): string
{
    return "My Test Adapter";
}
```

3.2. att. **Piemērs dažām no adaptera statiskajām metadatu funkcijām.**

Pārējās funkcijas ir tās, kurām ir nepieciešama adaptera klases inicializācija, un šīs funkcijas atbildīgas par adaptera ielādi un par pašu datu atlasi.

Visiem adapteriem ir obligāti jāimplementē 'CustoMood' lietotnē definēta saskarne 'BaseAdapterInterface', kas nosaka to kādām funkcijām adapterim ir jābūt. Pēc šīs raksturiezīmes lietotne arī atlasa no visām klasēm tās, kuras ir adapteri un kuras nav. Tiek atlasītas visas klases, kas implementē 'BaseAdapterInterface' un atrodas pareizajā mapē (jeb precīzāk nosaukumvietā), un tās lietotne atpazīst kā adapterus.

3.2.2.2. Saskarne 'BaseAdapterInterface'

Kā tika minēts iepriekš, visi adapteri implementē saskarni 'BaseAdapterInterface', kurā ir definētas funkcijas, kuras adapterim ir jāimplementē. Šī saskarne ir atrodama 'CustoMoodAppBundle' saišķī nosaukumvietā CustoMood\Bundle\AppBundle\BaseAdapter, un tajā definētās funkcijas ir aprakstītas tabulā 3.1.

3.1. tabula

‘BaseAdapterInterface’ saskarnes funkcijas, kuras jāimplementē.

Funkcija	Tips	Apraksts
getId(): string	Statiska	Šai funkcijai ir jāatgriež unikāls identifikators adapterim simbolu virknes formātā. Šim identifikatoram jābūt unikālam globāli, tāpēc ir ieteicams to likt tādu, lai nebūtu konflikti ar citiem adapteriem.
getDisplayName(): string	Statiska	Šai funkcijai ir jāatgriež adaptera nosaukums, kas tiks rādīts lietotājam sistēmā. Tas var būt tukšs, kā arī tam nav jābūt unikālam.
getAuthor(): string	Statiska	Šai funkcijai ir jāatgriež autora nosaukums (piemēram, organizācijas nosaukums, cilvēka vārds un uzvārds).
getDescription(): string	Statiska	Šai funkcijai ir jāatgriež īss apraksts adapterim, kas tiks parādīts lietotnē adapteru sarakstā. Tas var būt tukšs, bet vēlams to uzstādīt.
getWebsite(): string	Statiska	Šai funkcijai ir jāatgriež autora mājaslapa. Ja tāda nav, tad tā var būt tukša.
getVersion(): float	Statiska	Šai funkcijai ir jāatgriež adaptera versiju kā decimālskaitli.
getSettingsSchema(): array	Statiska	Ja adapterim ir nepieciešamas kādas vērtības, ko vajag iegūt no lietotāja, tad šī funkcija var atgriezt kolekciju ar iestatījumiem, kuru vērtības adapteris saņems, kad tas tiks izsaukts.

Funkcija	Tips	Apraksts
		Katrai iestatījuma definīcijai kolekcijā jābūt masīvam ar sekojošiem indeksiem, kuros ir attiecīgās vērtības: • key – Unikāls iestatījuma nosaukums • value – Noklusētā vērtība • display_name – Nosaukums iestatījumam tāds, kādu rādīs lietotājam • type – Lauka tips (0 = simbolu virkne, 1 = vesels skaitlis, 2 = loģiskā jā/nē vērtība) • required – Atbilde uz jautājumu vai lietotājam šis lauks jāizpilda obligāti? • order – Secība attiecībā pret pārējiem šī adaptera iestatījumiem
load($projectId,$settingValues)	Instances	Šī funkcija tiek izsaukta pirms datu analīzes un tajā tiek padots projekta identifikators un vērtības iepriekšējā funkcijā definētajiem iestatījumiem.
getMood($dateFrom, $dateTo): array	Instances	Šī ir galvenā adaptera funkcija, kura tiek izsaukta pēc 'load' funkcijas izsaukuma, un kurā jānotiek datu savākšanai no ārējās datu kopas. Tiek padoti divi parametri, pēc kuriem datiem jābūt atlasītiem – sākuma un beigu datums, starp kuriem atlasītajiem datiem jābūt veidotiem.

Funkcija	Tips	Apraksts
		Funkcijai atpakaļ ir jāatgriež masīvs ar datiem, kur katrs vienums ir masīvs ar sekojošajiem indeksiem, kuros jābūt attiecīgajām vērtībām: • date – UNIX laika zīmogs ierakstam • text – Simbolu virkne, kurā ir klienta teiktais

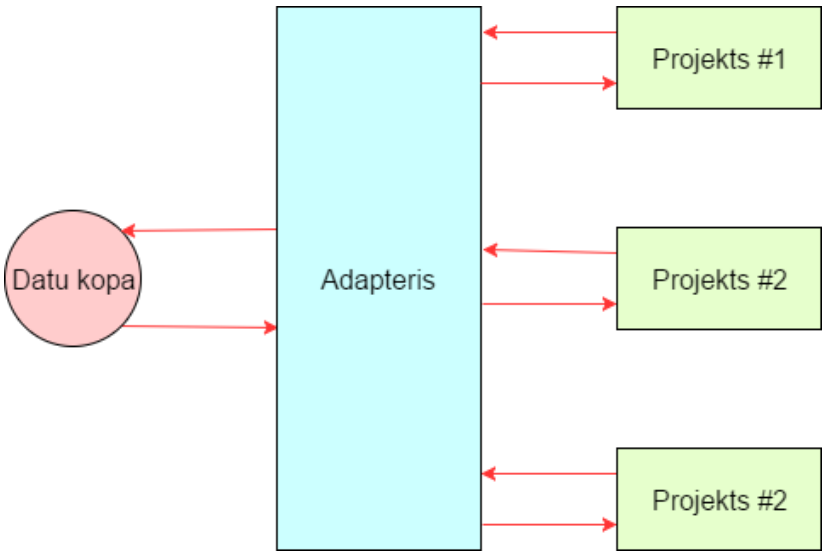
Saskarnes pilnais pirmkods ir atrodams 2. pielikumā.

3.2.3. Projekti

Iepriekšminētie adapteri ļauj lietotnei ‘CustoMood’ savākt datus no dažādām sistēmām, bet lai varētu šos datus analizēt vispirms ir jāizveido projekts, kurā tiek norādīti adapterim nepieciešamie parametri un tātad pateikts tas, kas tieši no attiecīgās sistēmas ir jāpaņem.

Izveidojot projektu, tam tiek izveidota saistība ar izvēlēto adapteri, un lietotājam ir jāaizpilda vērtības adapterī definētajiem iestatījumiem, kuri tālāk tiek saglabāti datubāzē. Var būt iespējams, ka adapterim nekādi iestatījumi nav, kas nozīmē, ka izveidojot projektu arī nekas papildus nav jāaizpilda.

Kad projekts ir izveidots un iestatījumi ir aizpildīti, tad katrs lietotājs var atvērt projekta lapu, kurā var norādīt parametrus apmierinātības analīzei un atpakaļ tiks atgriezti dati par apmierinātību.



3.3. att. **Sasaiste starp adapteri un projektiem.**

3.2.4. Lietotāju pārvaldība

Lietotnē ir izstrādāta reģistrācijas sistēma, kas ļauj reģistrēties jauniem lietotājiem, norādot lietotājvārdu, e-pastu un paroli. Visi šie jaunie lietotāji ir grupā "Lietotājs", kas viņiem ļauj tikai rediģēt savus datus un apskatīt apmierinātību izveidotajiem projektiem.

Īpašajiem administrācijas lietotājiem ir pieeja administrācijas panelim, kurā var tikt iespējoti un atspējoti adapteri, izveidoti un rediģēti projekti, kā arī mainīti globālie sistēmas iestatījumi. Piemēram, viens no iestatījumiem ir reģistrācijas iespējošana un atspējošana.

3.2.5. Apmierinātības analīze

3.2.5.1. Analīzes un mērīšanas process

Cilvēku apmierinātības mērīšana un analīze ir galvenais rīka mērķis, tāpēc ir svarīgi arī aprakstīt kā tas konkrēti notiek. Lai mērīšana varētu notikt automātiski, ir jāizmanto kāda no netiešajām apmierinātības mērīšanas metodēm, un no tām visprecīzākā bija teksta analīze, tāpēc arī tā tiek izmantota šajā rīkā.

Kad adapteris atgriež kolekciju ar visiem komentāriem (vai atsaucēm, vai jebkāda cita veida simbolu virknēm), tad katrs no tiem tiek analizēts, izmantojot divas lietotnē definētas vārdnīcas – vārdnīca ar pozitīvām frāzēm un vārdnīca ar negatīvām frāzēm.

Šajās vārdnīcās ir angļu valodā definētas frāzes un katrai ir piesaistīts skaitlisks svars – pozitīvām frāzēm pozitīvs, negatīvām frāzēm negatīvs. Frāzi analizējot tajā tiek atrastas visas vārdnīcu pozitīvās frāzes, negatīvās frāzes, tiek sasummēti visi svari un iegūts apmierinātības mērījums. Šādi tiek apstrādātas visas no adaptera atgrieztās simbolu virknes, un kad visi apmierinātības mērījumi ir iegūti, tad notiek datu grupēšana un agregācija.

Piemēram, ja ir frāze "Sveiki, man ļoti patīk padarītais, viss ir izdarīts labi, bet dažas problēmas vēl ir", un ir sekojošas vārdnīcas:

- "ļoti patīk" ar svaru +30
- "labi" ar svaru +10

- "slikti" ar svaru -30
- "problēmas" ar svaru -15

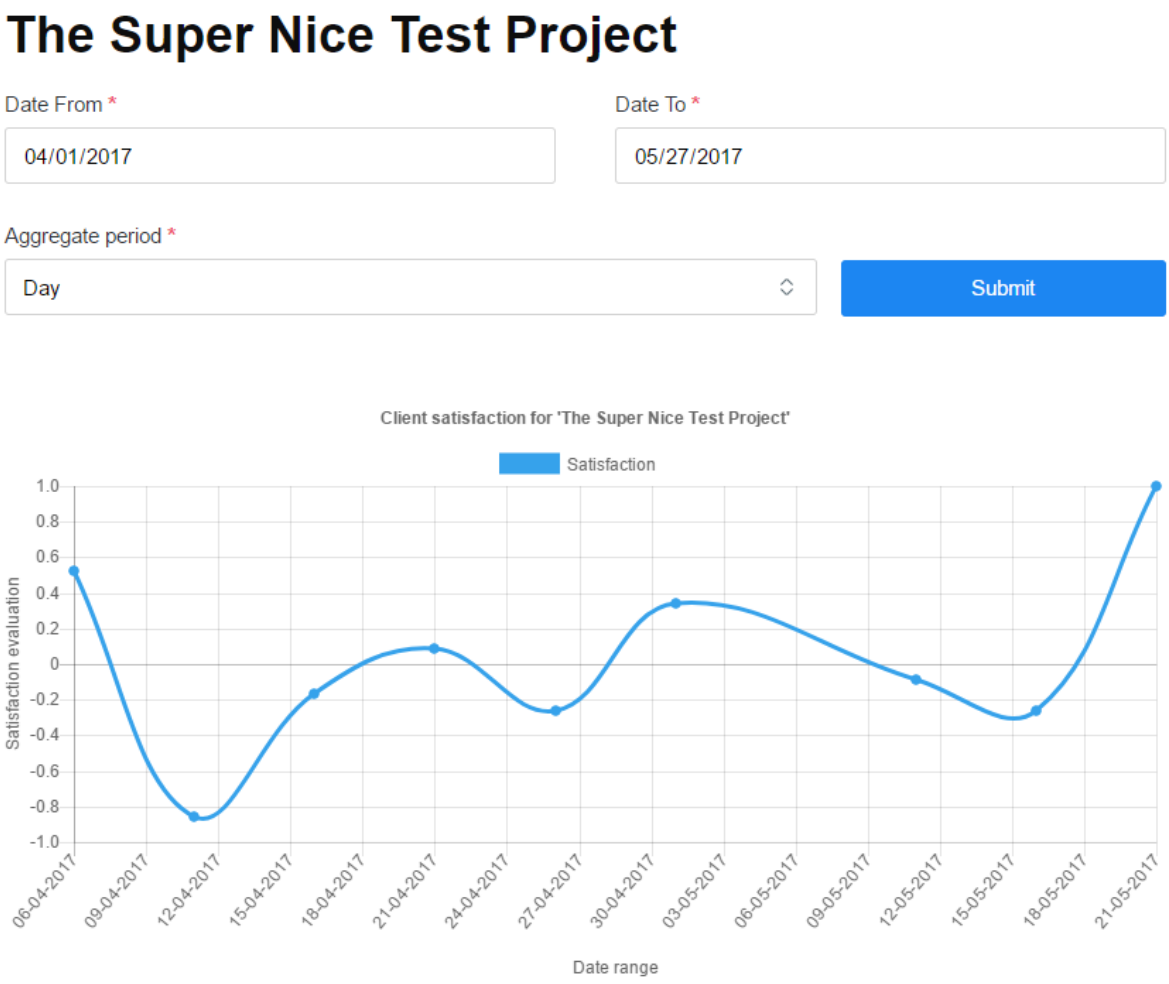
Tādā gadījumā rezultējošais apmierinātības novērtējums būs +10 (jeb 30 + 10 – 30).

Vārdnīcās mēdz būt frāzes, kas ir daļa no citām frāzēm, piemēram var būt divas frāzes – "labi" un "ļoti labi". Lai svars tiktu ieskaitīts tikai par vienu no tiem, prioritāte ir garākām frāzēm.

Viens no parametriem, ko var norādīt veicot apmierinātības mērīšanu ir agregācijas periods, kas nosaka to kādos periodos grupēt datus grafikā, kurš tiek attēlots lietotāja saskarnē procesa beigās. Mazākais iespējamais agregācijas periods ir "1 diena", kas nozīmē, ka visas frāzes ar datumu vienā un tajā pašā dienā tiek grupētas kopā un grafikā vērtības ir sadalītas pa dienām. Periods "1 nedēļa", piemēram, datus grupēs pa nedēļām.

Kad visas frāzes ir izmērītas un nogrupētas, tad tiek veikta vērtību vienkāršošana, kas skaitliskās vērtības novienkāršo tā, lai tās visas būtu intervālā [-1.0; 1.0], tā lai rezultāti būtu uzskatāmāki un nebūtu jāvēro katru reizi savādākā skalā.

Lietotāja saskarnes pusē viss, kas lietotājam ir jāizdara, ir jānorāda analīzes sākuma datums, beigu datums, agregācijas periods un tad parametrus var iesniegt, un pēc analīzes procesa beigām tiks attēlots grafiks ar rezultātiem.



3.4. att. **Piemērs klientu apmierinātības analīzes rezultātu skatam.**

3.2.5.2. Ātrdarbība

Ņemot vērā, ka no adaptera varētu nākt milzīgi apjomi ar datiem, projektējot šo rīku un it īpaši klienta apmierinātības mērīšanas funkcionalitāti nācās arī ņemt vērā ātrdarbību un datora resursu izmantošanu.

Kad runa iet par teksta analīzi, viena no lietām, kas uzreiz varētu uzreiz ienākt prātā analīzes veikšanai ir regulārās izteiksmes, bet problēma ar tām ir, ka teksta analīze ar regulārajām izteiksmēm pie lieliem apjomiem paliek ļoti lēna, tāpēc nācās domāt par citiem veidiem kā tekstā meklēt pozitīvās un negatīvās fāzes. PHP funkcija str_replace izrādījās laba alternatīva, jo tā atrod un aizvieto daļu teksta kādā teksta virknē relatīvi ātri, neizmantojot daudz resursus.

Rezultātu kešošana arī ir funkcionalitāte, kas ļoti noderētu šim rīkam, bet laika trūkuma dēļ tā prototipā netika īstenota.

3.2.6. Plānotā funkcionalitāte nākotnei

Šī darba laikā izstrādātais rīks tika izstrādāts kā prototips, lai parādītu kā tas strādātu, bet nākotnē ir plānots to papildināt un uzlabot. Dažas no lietām, ko ir plānots uzlabot ir:

- Ātrdarbības uzlabojumi – Izstrādātais rīks var potenciāli tikt izmantots ar milzīgiem datu apjomiem, kas nozīmē, ka vajag īpašu uzmanību pievērst tam kā lietotne izmanto servera resursus. Būtu nepieciešams izstrādāt kaut kādu kešatmiņas sistēmu, kas analīzes rezultātus saglabātu tā, lai nākamreiz tos nebūtu jāanalizē no jauna. Rezultātu agregāciju derētu pārcelt uz klient pusi, tā lai mainot agregācijas periodu nebūtu katru reizi no jauna jāveic analīze.
- Lielākas frāžu vārdnīcas – Negatīvo un pozitīvo frāžu vārdnīcas vajadzētu papildināt ar jaunām frāzēm, kā arī dot iespēju administratoriem reģistrēt jaunas frāzes.
- Lokalizācija – Noderīga varētu būt iespēja definēt vārdnīcas dažādās valodās, kas ļautu izmantot dažādiem projektiem dažādas vārdnīcas, atkarībā no projektos izmantotās valodas.

3.3. Projektējums

Ņemot vērā iepriekšējās nodaļās aprakstīto un to kādas ir galvenās problēmas, kuras būtu jārisina, šajā darbā izstrādātajam prototipam iepriekš aprakstītajam rīkam bija jābūt projektētam un izstrādātam konkrētā veidā. Tas tiek aprakstīts tālāk.

3.3.1. Izstrādes un darbības vide, un tehnoloģijas

Nodaļā 3.1.5. tika minēti faktori, kas jāņem vērā izstrādājot lietotni, tā lai tā būtu pieejama un izmantojama pēc iespējas vairāk cilvēkiem un būtu uzstādāma uz pēc iespējas vairāk serveriem.

PHP ir viena no piecām populārākajām server-puses programmēšanas valodām un no šīm piecām populārākajām tā ir tā, ar kuru šī darba autors ir saskāries visvairāk ,un ar kuru ir vislielākā pieredze, izstrādājot tīmekļa lietotnes, tādēļ šī valoda ir pamatā izstrādātajam rīkam.

Rīks tika izstrādāts uz Windows vides, un palaists uz nginx servera, bet šo lietotni bez papildus konfigurācijas var palaist arī uz UNIX tipa operētājsistēmām, uz kurām arī ir pieejams nginx serveris [25] [26]. PHP tīmekļa lietotnes var arī iedarbināt uz cita tipa serveriem, kā piemēram Apache, bet tad varētu būt nepieciešama papildus konfigurācija [27].

Tā kā mūsdienu tīmekļa lietotnes ir kļuvušas daudz sarežģītākas nekā tās bija pirms, piemēram, 10 gadiem, tad ir ieteicams tās labāk taisīt, izmantojot kādus ietvarus, kas atvieglo izstrādi un sniedz tādu funkcionalitāti, kas ir parasti nepieciešama visām lietotnēm, lai tā nebūtu jāizstrādā atkal. Tāpat arī šis rīks tika izstrādāts, izmantojot Symfony tīmekļa lietotņu ietvaru, kas ir noderīgs jebkāda veida PHP MVC tipa tīmekļa lietotņu izstrādei [28]. Symfony piedāvā tādas lietas kā resursu pārvaldību, ceļu reģistrēšanu, atkarību injicēšanu, servisu izstrādi, lietotāja saskarnes izstrādi, objektorientētu datubāzes saskarni u.c.

Izstrādes atvieglošanai projektā tika izmantots PHP atkarību pārvaldītājs Composer, kas nodrošina ērtu un organizētu atkarību pārvaldību, trešo pušu bibliotēku un papildinājumu pievienošanu, instalēšanu un izmantošanu [33].

Klient-puses tehnoloģijas lietotnei sastāv no ierastajām trīs – HTML, CSS un JavaScript. Lietotāja saskarnes struktūra tika izstrādāta, izmantojot HTML valodu kopā ar Symfony piedāvāto Twig saskarnes veidošanas dzini, kas atvieglo HTML koda izveidi un ģenerēšanu [29]. Saskarnes stila izveidei tika izmantots SASS, kas ir CSS papildinājums, kas atvieglo CSS stila lapu izstrādi ļaujot tās organizēt labāk un rakstīt stila deklarācijas īsāk un pārskatāmāk [30]. Saskarnes skriptiem tika izmantota valoda JavaScript, kopā ar jQuery bibliotēku, kas atvieglo darbu ar JavaScript un sniedz daudz funkcijas, kuras valodai JavaScript pēc noklusējuma nenāk līdzi [31].

Strādājot ar klient-puses tehnoloģijām, kad projektā mēdz atrasties daudz skriptu, šablonu un stila lapu, ir vērts izmantot rīkus, kas veic šo resursu sasaistīšanu, apvienošanu un minimizāciju, un tā tas arī bija šajā projektā, tāpēc arī tika izmantots JavaScript uzdevumu

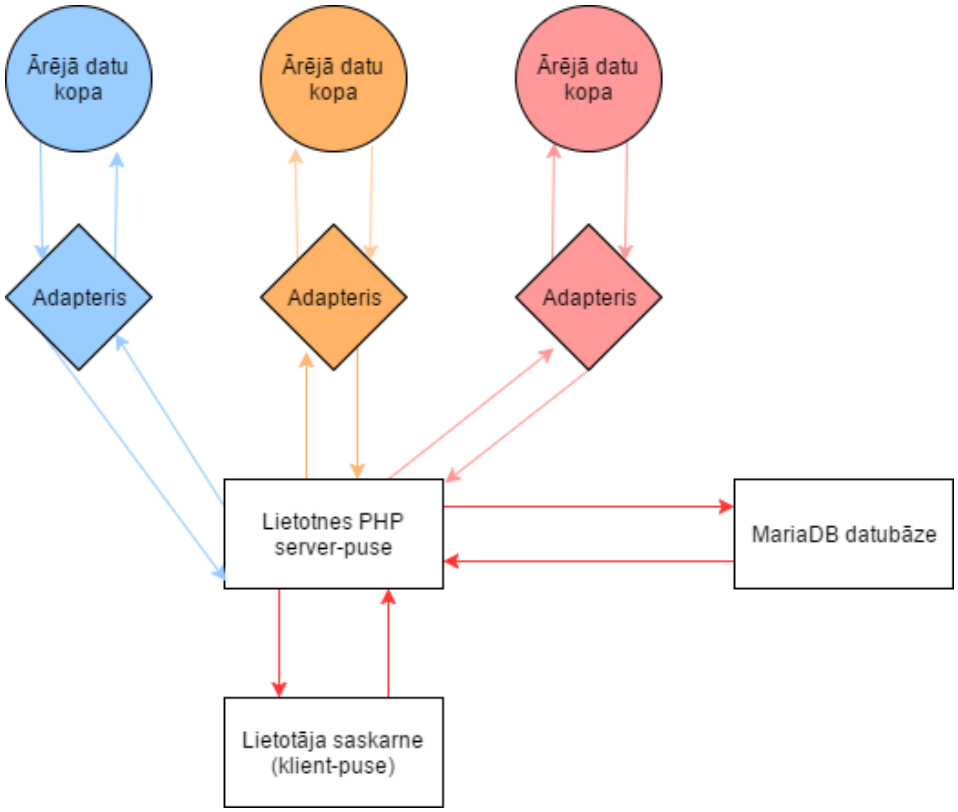
izpildītājs Gulp, kas automatizēja iepriekšminētās darbības kā arī automātiski reāllaikā kompilēja SASS stila lapas [32].

Datu glabāšanai tika izmantota MariaDB datubāžu pārvaldības sistēma, kas ir drošāks un ātrāks aizvietojums industrijas līderim MySQL [34] [35].

3.3.2. Vispārīgais projektējums

Rīks "CustoMood" ir PHP MVC tipa tīmekļa lietotne, kas visu loģiku pilda uz servera, un darba rezultātus atgriež klientam HTML lietotāja saskarnes veidā. Dati glabājas MariaDB datubāzē, ar kuru lietotne veic savienojumu, kad ir nepieciešams iegūt vai saglabāt kādus datus.

Lietotnē ir arī integrējami trešās puses papildinājumi – "adapteri", kas strādā kā saskarne starp lietotni "CustoMood" un jebkādu trešās puses datu glabātuvi. Šos adapterus var ieinstalēt ieliekot tos specifiskā mapē lietotnes instalācijas atrašanās vietā, un tie tiek automātiski atpazīti un integrēti sistēmā. Adapteri var būt vairāki.



3.5. att. **Augsta līmeņa projektējums lietotnei.**

3.3.3. Detalizēts projektējums

Lietotne ir sadalīta vairākās komponentēs atbilstoši tam kāda ir labā prakse, izstrādājot tīmekļa lietotnes izmantojot ietvaru Symfony. Symfony lietotnēm parasti ir sekojoša struktūra [36]:

- **app/** - Mape, kurā atrodas galvenās lietotnes konfigurācijas datnes, globālie tulkojumi un lietotāja saskarnes
- **bin/** - Mape ar izpildāmajām palīgprogrammām. Tajā arī atrodas Symfony konsole, ar kuru var ģenerēt jaunas lietotnes daļas, atjaunināt datubāzes shēmu, iztīrīt kešatmiņu un veikt citas darbības.
- **src/** - Galvenā mape, kurā tiek likts viss kods, ko izstrādātājs rada pats.
- **tests/** - Mape, kurā tiek likti automātisko vienību testu definīcijas.
- **var/** - Mape pagaidu failiem (kešatmiņai, atkļūdošanas informācijai u.tml.)
- **vendor/** - Mape, kurā glabājas Composer rīka pārvaldītās trešās puses bibliotēkas un papildinājumi.
- **web/** - Lietotnes saknes mape, kurā tiek glabātas statiskās datnes un PHP skripti, kuri tiešā veidā ir pieejami interneta pārlūkā. Te glabājas lietotnes ieejas punkts 'app.php', kurā iesākas pašas PHP lietotnes darbība.

Šī pati struktūra tika ievērota arī lietotnei 'CustoMood', bet ņemot vērā lietotnes prasības un to, ka lietotni plānots izmantot vairākiem cilvēkiem, kuriem pieeja pie lietotnes pirmkoda nav vajadzīga un ir maksimums nepieciešamība pievienot papildus adapterus, pašas lietotnes pirmkods ir izcelts kā atsevišķs Symfony saišķis un Composer pakotne.

Ietvars Symfony ļauj izstrādātājam kodu grupēt saišķos, kas katrs skaitās kā diskrēta, neatkarīga pakotne, kuru, ja vēlas, var arī reģistrēt kā Composer pakotni. Šāda sistēma ļauj izstrādātājiem viegli izstrādāt Symfony ietvara papildinājumus un bibliotēkas – izstrādātāji savus saišķus var publicēt, un citi ar Composer palīdzību tos var izmantot savās Symfony lietotnēs [37].

Tā kā viss lietotnes galvenais kods ir izcelts atsevišķā Symfony saišķī, tad tas nozīmē, ka galvenajā lietotnes komponentē, kuras struktūra tika aprakstīta iepriekš, neatrodas nekāds

pirmkods, tajā atrodas tikai galvenie lietotnes konfigurācijas faili (mapē “app/”) kā arī īpašā mape, kurā var likt lietotnes adapterus.

3.3.3.1. Saišķa “CustoMoodAppBundle” struktūra

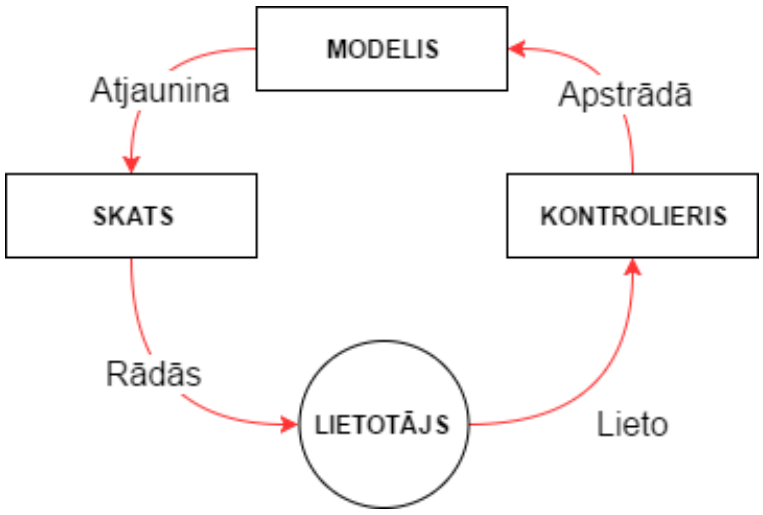
Līdzīgi kā pašai Symfony lietotnei ir standarta struktūra, kas tika aprakstīta iepriekš, arī Symfony saišķiem ir standarta struktūra. Izstrādātais kods tiek organizēts mapēs atkarībā no tā kādu funkcionalitāti tas sniedz, un “CustoMood” lietotnes galvenā saišķa “CustoMoodAppBundle” struktūra ir sekojoša:

- BaseAdapter – Šajā mapē glabājas adapteru izstrādei nepieciešamās PHP klases, kā piemēram pamata saskarne ‘BaseAdapterInterface’, kuru ir obligāti jāimplementē visiem adapteriem.
- Controller – Šajā mapē glabājas visi MVC kontrolieri.
- DataFixtures – Šajā mapē glabājas datu migrāciju un armatūru klases, kas ir atbildīgas par datubāzes piepildīšanu ar nepieciešamajiem datiem.
- DBAL – Šajā mapē ir definēti datubāzes pieejas līmeņa papildinājumi, kā piemēram, jauni lauku tipi. Šī projekta ietvaros tika izstrādāts jauns iteratora lauks, kas apraksta iestatījumu lauku tipu.
- DependencyInjection – Šajā mapē atrodas PHP klases, kas ļauj saišķim injicēt savus servisus ārējās lietotnes servisu konteinerī.
- Entity – Šajā mapē glabājas visas MVC modeļu definīcijas, kas apraksta to kāda ir modeļu struktūra un kā tiem jāglabājas datubāzē.
- Form – Šajā mapē glabājas visas HTML formu definīcijas.
- Helper – Šajā mapē glabājas dažādas statiskas palīgklases.
- Model – Šajā mapē atrodas dažādi modeļi un skatu-modeļi, kas nav saistīti ar datubāzi.
- Repository – Šajā mapē ir definētas repozitoriju klases entītijām. Tās atbild par papildus funkcionalitātes pievienošanu operācijām ar datubāzes entītijām (modeļiem), kā piemēram, kāda specifiska veida datu atlases, datu kārtošanas u.tml.

- Resources – Šajā mapē glabājas saišķa konfigurācijas datnes, statiskie resursi (bildes, stila lapas, JavaScript skripti utt.) kā arī Twig skatu definīcijas.
- Service – Šeit glabājas servisu klases. Servisi ir klases, kuras ir globāli pieejamas jebkurā vietā sistēmā, kur ir pieejams servisu konteineris. Šo klašu atkarības automātiski pārvalda servisu konteineris, un to instances nav nepieciešams manuāli izveidot.
- Twig – Šajā mapē atrodas Twig dziņa papildinājumu klases, kurās var definēt papildus Twig filtrus un funkcijas, kuras var tālāk izmantot Twig skatos.

3.3.3.2. Komponenšu projektējums

Tā kā lietotni augstā abstrakcijas līmenī var nosaukt par MVC tipa lietotni, tad to var sadalīt MVC komponentēs, kura katra sastāv no kontroliera un ar to saistītajiem modeļiem un skatiem. MVC sistēmā kontrolieris ir komponentes daļa, kurā notiek visa loģika un algoritmu darbība, modeļi ir datu struktūras, kurās tiek glabāti dati, un ir definēts kā šiem datiem jāizskatās un kā tiem jāglabājas datubāzē un skati ir lietotnes saskarnes, kurās kontrolieros apstrādātā informācija tiek izvadīta.



3.6. att. **MVC paradigma.**

Zemāk tabulā 3.2. ir aprakstīti lietotnes kontrolieri un funkcionalitāte ko tie nodrošina lietotājam.

3.2. tabula

Lietotnes kontrolieri.

Kontrolieris	Apraksts
AdminController	Atbild par administrācijas paneļa funkcionalitāti. Administrācijas panelis ir pieejams lietotājiem grupā "Administrators" un tajā var apskatīt pieejamos adapterus, taisīt/rediģēt projektus un rediģēt vispārīgos sistēmas iestatījumus.
MainController	Atbild par galveno sistēmas paneli, no kura var tikt pie visu projektu apmierinātības grafiku lapām, un apskatīt tās.
ProjectController	Atbild par projektu saraksta apskati, jaunu projektu izveidi, projektu rediģēšanu kā arī projektu adapteru iestatījumu pārvaldību.
UserController	Atbild par lietotāju pārvaldību. Ja ir ieslēgta reģistrācija, tad lietotāji ar šī kontroliera palīdzību var reģistrēties, ieiet sistēmā, rediģēt savus datus (paroli, e-pastu). Lietotājs, kas nav autentificējies sistēmā nevar lietotni izmantot, un var redzēt tikai pieteikšanās lapu.

Tabulā 3.3. ir aprakstītas lietotnes entītijas un tātad arī datubāzes struktūra, kas ir nepieciešama lietotnes darbībai.

3.3. tabula

Lietotnes entītijas.

Entītija	Apraksts
AdapterMetadata	Lai gan informācija par adapteriem tiek nolasīta reāllaikā no adaptera klases, tad kad tie tiek izmantoti, par tiem papildus informāciju arī var glabāt datubāzē, ja tas ir nepieciešams. Piemēram, adapterus ir iespējams atspējot,

Entītija	Apraksts
	kā rezultātā datubāzes AdapterMetadata tabulā pie konkrētā adaptera tiek atzīmēts, ka tas ir atspējots.
Project	Apraksta informāciju par projektu, ieskaitot to kādu adapteri tas izmanto, kādam mērķim tas veidots, kāds ir tā nosaukums u.tml.
ProjectSetting	Tā kā adapteriem var būt savi iestatījumi, to vērtības katram projektam tiek glabātas datubāzē. Tā kā iepriekš nav iespējams pateikt kādi šie iestatījumi būs un kādas būs to vērtības, šajā entītijā glabājas arī iestatījuma tips (piemēram, teksts vai skaitlis), tas vai tā aizpildīšana ir obligāta un kāda ir tā pašreizējā vērtība.
Role	Apraksta lietotāja lomu jeb grupu. Sistēmā ir divas lomas: lietotājs un administrators.
Setting	Strādā pēc līdzīga principa kā entītija 'ProjectSetting', bet ir domāta vispārīgajiem lietotnes iestatījumiem, kuri tiek izveidoti un ievietoti datubāzē automātiski ar datu migrācijām.
User	Apraksta lietotāju – viņa lietotājvārdu, e-pastu, paroli, lomas u.tml.

Tā kā lietotnē ir arī dažāda tipa funkcionalitāte, kuru ir nepieciešams izmantot no dažādām, savstarpēji nesaistītām vietām lietotnē, tad tika izveidoti arī vairāki servisi, kuri tiek izmantoti gan kontrolieros, gan skatos, un tie ir aprakstīti tabulā 3.4:

3.4. tabula

Servisu apraksts.

Serviss	Apraksts
AdapterService	Ar šī servisa palīdzību var iegūt informāciju par visiem iespējotajiem un atspējotajiem sistēmā ieinstalētajiem

Serviss	Apraksts
	adapteriem, iegūt to metadatus un tos inicializēt un izmantot.
AnalysisService	Šis serviss ir atbildīgs par pašu apmierinātības analīzes mērīšanu. Tam tiek padoti parametri un lietotāju tekstuālās ziņas, un tad tas analizē tās un atgriež apmierinātības novērtējumu.
SettingsService	Šis serviss nodrošina pieeju pie sistēmas globālajiem iestatījumiem no jebkuras vietas sistēmā. Sistēmas globālie iestatījumi glabājas datubāzē 'Setting' tipa entītijās.

3.4. Uzstādīšana

Izstrādājot lietotni 'CustoMood', īpaša uzmanība tika pievērsta tam cik ērti to būs uzstādīt un izmantot sistēmu administratoriem. Kā tika minēts iepriekš lietotne ir izveidota tā, lai tajā glabātos tikai galvenie konfigurācijas faili, un viss rakstītais kods būtu atsevišķā Symfony saišķī, kuru sistēmas administratoram nevajadzētu aiztikt, un kurš būtu pieinstalēts automātiski ar Composer rīku.

3.4.1. Sistēmas prasības

Lai lietotni varētu izmantot, tai ir sekojošas sistēmas prasības [38]:

- Jebkāda tipa serveris, kas atbalsta PHP tīmekļa lietotnes (piemēram, nginx vai Apache)
- PHP 7+ ar ieslēgtiem JSON, ctype papildinājumiem un uzstādītu 'date.timezone' PHP konfigurācijas datnē php.ini
- MariaDB 5+ vai MySQL 3+ datubāžu pārvaldības sistēma
- Composer rīks (vēlams pēc iespējas jaunāka versija, bet jebkura der)

- GIT rīks (vēlams pēc iespējas jaunāka versija, bet jebkura der)

3.4.2. Instalācijas pamācība

Pirms sekojošo instrukciju izpildīšanas ir pieņemts, ka sistēmas prasības ir ievērotas un visi nepieciešamie rīki ir ieinstalēti.

- Ar GIT tiek lejupielādēts repozitorijs no https://github.com/fabis94/CustoMood ‘master’ zara
- Izmantojot sistēmas termināli, atrodoties rīka saknes mapē tiek izpildīta komanda ‘composer install’, un kad termināls to prasa, jāievada parametri datubāzes savienojumam
- Terminālī jāizpilda komanda ‘php bin/console doctrine:schema:update –force’
- Terminālī jāizpilda komanda ‘php bin/console doctrine:fixtures:load’

Pēc šo komandu izpildes lietotne ir veiksmīgi ieinstalēta ar noklusēto administratora lietotāju, kura lietotājvārds ir ‘admin’ un noklusētā parole ir ‘password123’, kuru ir vēlams nomainīt uz kādu citu.

Lai rīks būtu pieejams publikai, tam ir jābūt uzstādītam uz servera, norādot lietotnes ‘web/’ mapi kā tīmekļa lietotnes sakni. Konfigurācija serverim ir tāda pati kā jebkādai citai Symfony lietotnei un populārākajiem serveriem šo konfigurāciju var atrast internetā [39].

3.5. Iepriekš apskatīto problēmu risinājumu novērtējums

Nodaļā 3.1. tika apskatītas galvenās problēmas, kuras lietotnei būtu jārisina, un šajā nodaļā tiek izvērtēts tas, cik labi vai slikti tas bija izdarīts.

3.5.1. Automatizācija

Viens no rīka mērķiem bija klientu apmierinātību mērīt automatizēti – bez nepieciešamības pēc jebkādas iejaukšanās no pašiem klientiem. Šis rīks šo mērķi sasniedz apmierinātību mērot netieši – analizējot komentārus, atsauksmes vai jebkāda cita veida tekstu, ko klients ir sistēmā rakstījis

3.5.2. Abstrakcija / vispārināšana

Rīks ir izstrādāts tā, lai tas varētu apmierinātību mērīt datus ņemot gandrīz no jebkādas sistēmas. Tā kā rīkā ir iebūvēta adapteru sistēma, tad lai to sasaistītu ar kādu sistēmu, viss kas ir jāizdara ir jāizstrādā adapteris (saskarne) starp šo sistēmu un 'CustoMood' rīku, vai arī, ja šāds adapteris jau ir izstrādāts, var izmantot gatavu risinājumu. Šāda funkcionalitāte ļauj klientu apmierinātību mērīt jebkādā sistēmā, kurā lietotāji var glabāt kādu informāciju.

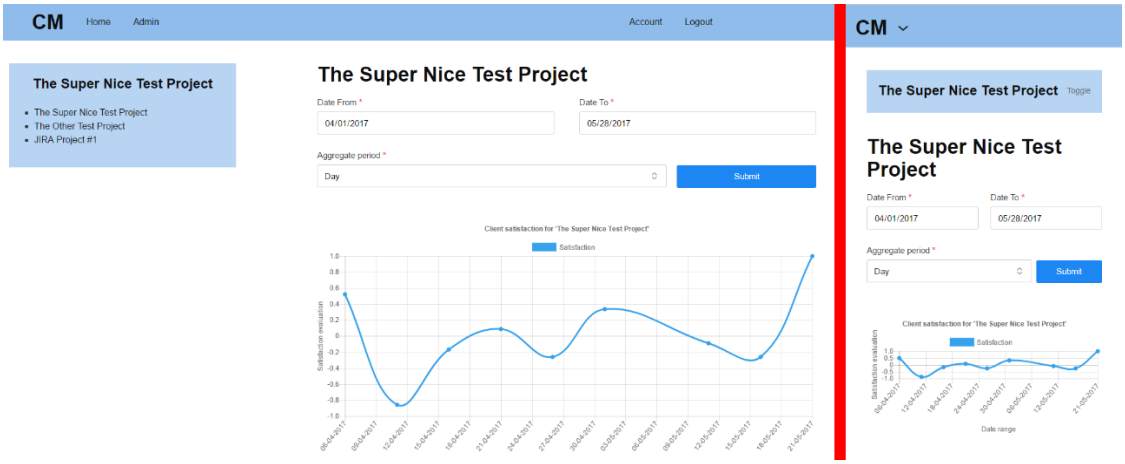
3.5.3. Rezultātu precizitāte

Lai rezultāti būtu pēc iespējas precīzāki, bet apmierinātības mērīšana vēl aiz vien būtu automatizēta, tika izvēlēts visprecīzākais netiešais mērīšanas veids – teksta analīze. Lai gan šādi mērot apmierinātību ir iespējams arī būt kļūdainiem rezultātiem, palielinot vārdnīcu izmērus arī attiecīgi palielinās rezultātu precizitāte, un pie pietiekami lielām pozitīvo un negatīvo frāžu vārdnīcām rezultāti ir precīzi.

3.5.4. Rezultātu attēlojums

Liels uzsvars izstrādājot rīku 'CustoMood' arī tika likts uz to, lai lietotājiem būtu ērti šo lietotni izmantot. Tāpēc izstrādājot rīka lietotāja saskarni, tika izmantoti responsīvās saskarnes projektējuma principi, kas to padara lietojamu gan uz datoriem, gan arī mazajiem telefonu

ekrāniem [40]. Tā kā tika izstrādāts tikai prototips, tad pats rīka stils arī ir prototipa stadijā, bet strukturāli lietotne ir ļoti ērti lietojama un ir saskarne ir izkārtota atbilstoši ekrāna izmēram.



3.7. att. **Lietotne datora ekrānā (pa kreisi) un lietotne mobilā telefona ekrānā (pa labi).**

Kā ir redzams attēlā 3.7., uz mazajiem ekrāniem lietotne ir tik pat lietojama kā uz lielajiem ekrāniem. Šādi lietotne strādā visās lapās, ne tikai grafika skatā, tāpēc to var pilnībā izmantot ar mobilajām ierīcēm, bez nekādas nepieciešamības pēc datora.

Citas lapas no saskarnes var redzēt 1. pielikumā.

3.5.5. Rīka darbības prasības

Lai rīku varētu izmantot pēc iespējas vairāk cilvēki, tas tika izstrādāts ar tādām sistēmas prasībām, kuras ir ļoti izpildīt. Rīku var uzstādīt uz jebkādas operētājsistēmas datora, un tas var strādāt uz dažādiem serveriem (piemēram, nginx un Apache).

Rīka server-puse tika rakstīta vienā no populārākajām programmēšanas valodām pasaulē – PHP, kas nozīmē, ka atrast izstrādātājus, kas mācētu izstrādāt adapterus sistēmai nevajadzētu būt grūti.

REZULTĀTI

Šajā darbā tika veiksmīgi izpētīts, kas ir projektu pārvaldība un kā ar to saistītos procesus mēdz atvieglot projektu pārvaldības rīki, kā arī tika izpētīti un atrasti labākie tirgus piedāvātie risinājumi. Lai gan šādu rīku ir daudz, tika atrasti tie, kurus izmanto un par kuriem var dzirdēt visbiežāk.

Tika izskatīts arī klientu apmierinātības jēdziens un apskatīts tas, ko tas parasti mēdz nozīmēt. Konkrētāk tika apskatīts tieši tas kā to mēdz definēt komerciālajā pasaulē un digitālajā pasaulē un tika aprakstīti visbiežāk sastopamākie veidi kā apmierinātību vispārīgi novērtē dažādi digitālie risinājumi. Tuvāk tika apskatīti tirgū populārākie klientu apmierinātības novērtēšanas rīki tieši konkrēti populārākajiem projektu pārvaldības rīkiem, to plusi un mīnusi, un tas kā tos varētu uzlabot.

Pēc tam, kad bija aprakstīti visi pieejamie risinājumi klientu apmierinātības mērīšanai projektu pārvaldības rīkos, tika izpētītas visas lielākās problēmas, kuras šiem rīkiem ir, un tika izstrādāts prototips rīkam, kas šīs problēmas risina. Rezultāts ir rīks ‘CustoMood’, kurš automatizētā, netiešā veidā ļauj veikt apmierinātības mērīšanu gandrīz jebkādai digitālai sistēmai.

SECINĀJUMI UN NOBEIGUMS

Šī kursa darba mērķis bija izpētīt tirgus situāciju projektu pārvaldības un klientu apmierinātības mērīšanas rīkiem, apskatīt veidus kā šīs sistēmas tiek savienotas, izskatīt kādas ir lielākās problēmas, kuras šiem risinājumiem ir, un izstrādāt konkrētu rīka prototipu, kas šīs problēmas risinātu un ļautu automatizētā, netiešā un abstraktā veidā izmērīt klientu apmierinātību.

Apskatot populārākos projektu pārvaldības rīkus, ir secināms, ka visbiežāk tiek izmantoti rīki JIRA, Trello, Asana un Redmine, bet nevienam no tiem nav iebūvēta klientu apmierinātības mērīšanas funkcionalitāte.

Apskatot populārākos risinājumus klientu apmierinātības novērtēšanai digitālajā pasaulē, tika atrasti vairāki rīki, daži no kuriem bija savietojami ar iepriekš minētajiem projektu pārvaldības rīkiem, bet lielākā daļa no kuriem nebija, kā arī visi no šiem rīkiem mērīja apmierinātību tiešā veidā un tā, ka klientam sava apmierinātība bija jānorāda manuāli.

Tika izsecināts, ka tirgū neeksistē vispārīgs, automatizēts rīks, kas mēra apmierinātību, bez nepieciešamas iejaukšanās no klienta, liekot viņam manuāli norādīt savu apmierinātību kaut kādā skalā, un tātad šādu rīku bija nepieciešams izstrādāt.

Rezultātā tika izstrādāts rīks ‘CustoMood’, kas tika noprojektēts un izstrādāts, ņemot vērā visus secinājumus par tirgū eksistējošajiem risinājumiem. Lai gan darba laikā tika izstrādāts prototips, kuram vēl būtu nepieciešami daži uzlabojumi, kuri tika darbā aprakstīti, kā, piemēram, ātrdarbības uzlabošana pie lieliem datu apjomiem, apmierinātības analīzes precizitātes uzlabošana, palielinot pozitīvo un negatīvo frāžu vārdnīcas, šis rīks jau šajā stadijā var ļoti labi novērtēt apmierinātību automatizēti no jebkāda teksta, kā arī tas ir ļoti abstrakts, un ir izstrādāts tā, lai trešo pušu izstrādātāji varētu viegli to savietot ar jebkādām sistēmām – gan projektu pārvaldības, gan citām.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. *What is Project Management?* [tiešsaiste] PMI [atsauce 15.04.2017.]
   Pieejams internetā: https://www.pmi.org/about/learn-about-pmi/what-is-project-management
2. *What is a Project Management Triangle?* [tiešsaiste] Project Management Software, 2012. [atsauce 15.04.2017]
   Pieejams internetā: http://www.projectmanagesoft.com/faq/what-is-a-project-management-triangle
3. *Paymo – Online Project Management App* [tiešsaiste] Paymo [atsauce 20.04.2017]
   Pieejams internetā: https://www.paymoapp.com/
4. *JIRA Software – Issue & Project Tracking for Software Teams* [tiešsaiste] Atlassian [atsauce 20.04.2017]
   Pieejams internetā: https://www.atlassian.com/software/jira
5. *JIRA Software – Pricing* [tiešsaiste] Atlassian [atsauce 20.04.2017]
   Pieejams internetā: https://www.atlassian.com/software/jira/pricing?tab=cloud
6. *Trello* [tiešsaiste] Trello [atsauce 23.04.2017]
   Pieejams internetā: https://trello.com/
7. *Jira vs Asana vs Trello – Comparing The Project Management Softwares* [tiešsaiste] Bridget Rogers, 05.12.2015. [atsauce 23.04.2017]
   Pieejams internetā: http://gazettereview.com/2015/12/jira-vs-asana-vs-trello-best-option/
8. *Asana* [tiešsaiste] Asana [atsauce 23.04.2017]
   Pieejams internetā: https://asana.com/
9. *Asana Premium and Enterprise plans* [tiešsaiste] Asana [atsauce 23.04.2017]
   Pieejams internetā: https://asana.com/pricing
10. *Redmine* [tiešsaiste] Redmine [atsauce 23.04.2017]
    Pieejams internetā: http://www.redmine.org/
11. *Customer Satisfaction Metrics: 6 Metrics You Need To Be Tracking* [tiešsaiste] Ross Beard, 02.09.2013. [atsauce 05.05.2017]
    Pieejams internetā: http://blog.clientheartbeat.com/customer-satisfaction-metrics-6-metrics-you-need-to-be-tracking/

12. *How to Measure Customer Satisfaction: Do You Overlook these 4 Key Customer Satisfaction Measurements?* [tiešsaiste] Scott Smith, 03.12.2012. [atsauce 05.05.2017] Pieejams internetā: https://www.qualtrics.com/blog/customer-satisfaction-measurement/
13. *How to Measure Customer Satisfaction* [tiešsaiste] Tom Smith, 11.07.2014. [atsauce 05.05.2017] Pieejams internetā: http://www.insightsfromanalytics.com/blog/bid/391487/how-to-measure-customer-satisfaction
14. *What is customer satisfaction?* [tiešsaiste] Business Dictionary [atsauce 05.05.2017] Pieejams internetā: http://www.businessdictionary.com/definition/customer-satisfaction.html
15. *Me at the zoo | YouTube* [tiešsaiste] Jawed Karim, 23.04.2005. [atsauce 10.05.2017] Pieejams internetā: https://www.youtube.com/watch?v=jNQXAC9IVRw
16. *Kaisen Sushi | Yelp* [tiešsaiste] Kaisen Sushi [atsauce 10.05.2017] Pieejams internetā: https://www.yelp.com/biz/kaisen-sushi-san-francisco
17. *Facebook* [tiešsaiste] Facebook [atsauce 10.05.2017] Pieejams internetā: https://www.facebook.com/
18. *Customer Satisfaction Survey for JIRA* [tiešsaiste] Candylio, 23.02.2017. [atsauce 10.05.2017] Pieejams internetā: https://marketplace.atlassian.com/plugins/com.candylio.jira.plugins.jira-csat/server/overview
19. *Redmine Helpdesk Plugin for full-featured Customer Service from RedmineUP* [tiešsaiste] Redmineup [atsauce 10.05.2017] Pieejams internetā: https://www.redmineup.com/pages/plugins/helpdesk#features
20. *Nicereply | Customer Satisfaction Survey, NPS & CES* [tiešsaiste] Nicereply [atsauce 10.05.2017] Pieejams internetā: https://www.nicereply.com/
21. *Customer Thermometer* [tiešsaiste] Customer Thermometer [atsauce 10.05.2017] Pieejams internetā: https://www.customerthermometer.com/
22. *April 2017 Web Server Survey | Netcraft* [tiešsaiste] Netcraft, 21.04.2017. [atsauce 15.05.2017] Pieejams internetā: https://news.netcraft.com/archives/2017/04/21/april-2017-web-server-survey.html
23. *Announcing .NET Core 1.0* [tiešsaiste] Rich Lander, 27.06.2016. [atsauce 15.05.2017]

Pieejams internetā: https://blogs.msdn.microsoft.com/dotnet/2016/06/27/announcing-net-core-1-0/

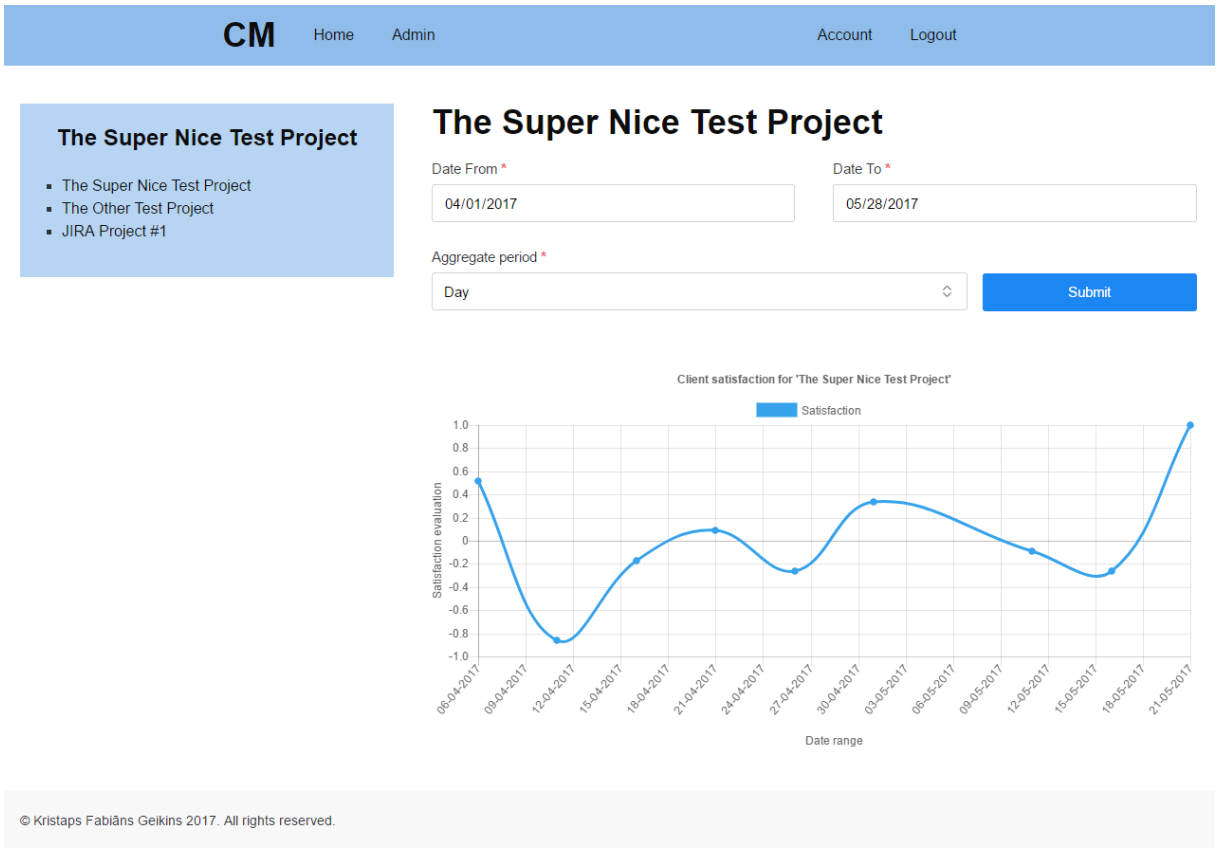
24. *TIOBE Index* [tiešsaiste] TIOBE [atsauce 15.05.2017]
    Pieejams internetā: https://www.tiobe.com/tiobe-index/
25. *Microsoft Windows* [tiešsaiste] Microsoft [atsauce 24.05.2017]
    Pieejams internetā: https://www.microsoft.com/en-us/windows/
26. *nginx* [tiešsaiste] nginx [atsauce 24.05.2017]
    Pieejams internetā: http://nginx.org/
27. *The Apache HTTP Server Project* [tiešsaiste] Apache [atsauce 24.05.2017]
    Pieejams internetā: https://httpd.apache.org/
28. *Symfony, High Performance PHP Framework for Web Development* [tiešsaiste] SensioLabs [atsauce 24.05.2017]
    Pieejams internetā: https://symfony.com/
29. *Twig | The flexible, fast, and secure template engine for PHP* [tiešsaiste] SensioLabs [atsauce 24.05.2017]
    Pieejams internetā: https://twig.sensiolabs.org/doc/2.x/
30. *SASS: Syntactically Awesome Style Sheets* [tiešsaiste] Sass [atsauce 24.05.2017]
    Pieejams internetā: http://sass-lang.com/
31. *jQuery* [tiešsaiste] jQuery [atsauce 24.05.2017]
    Pieejams internetā: http://jquery.com/
32. *gulp.js* [tiešsaiste] gulp.js [atsauce 24.05.2017]
    Pieejams internetā: http://gulpjs.com/
33. *Composer – Dependency Manager for PHP* [tiešsaiste] Nils Adermann, Jordi Boggiano [atsauce 24.05.2017]
    Pieejams internetā: https://getcomposer.org/
34. *MariaDB | Enterprise Open Source Database & Data Warehouse* [tiešsaiste] MariaDB [atsauce 24.05.2017]
    Pieejams internetā: https://mariadb.com/
35. *MySQL* [tiešsaiste] MySQL [atsauce 24.05.2017]
    Pieejams internetā: https://www.mysql.com/
36. *The Architecture | Symfony* [tiešsaiste] SensioLabs [atsauce 26.05.2017]
    Pieejams internetā: http://symfony.com/doc/current/quick_tour/the_architecture.html
37. *The Bundle System | Symfony* [tiešsaiste] SensioLabs [atsauce 26.05.2017]
    Pieejams internetā: http://symfony.com/doc/current/bundles.html

38. *Requirements for Running Symfony | Symfony* [tiešsaiste] SensioLabs [atsauce 26.05.2017]
Pieejams internetā: http://symfony.com/doc/current/reference/requirements.html
39. *Configuring a Web Server | Symfony* [tiešsaiste] SensioLabs [atsauce 26.05.2017]
Pieejams internetā: http://symfony.com/doc/current/setup/web_server_configuration.html
40. *Responsive Web Design Basics* [tiešsaiste] Pete LePage, 12.05.2017. [atsauce 26.05.2017]
Pieejams internetā: https://developers.google.com/web/fundamentals/design-and-ui/responsive/

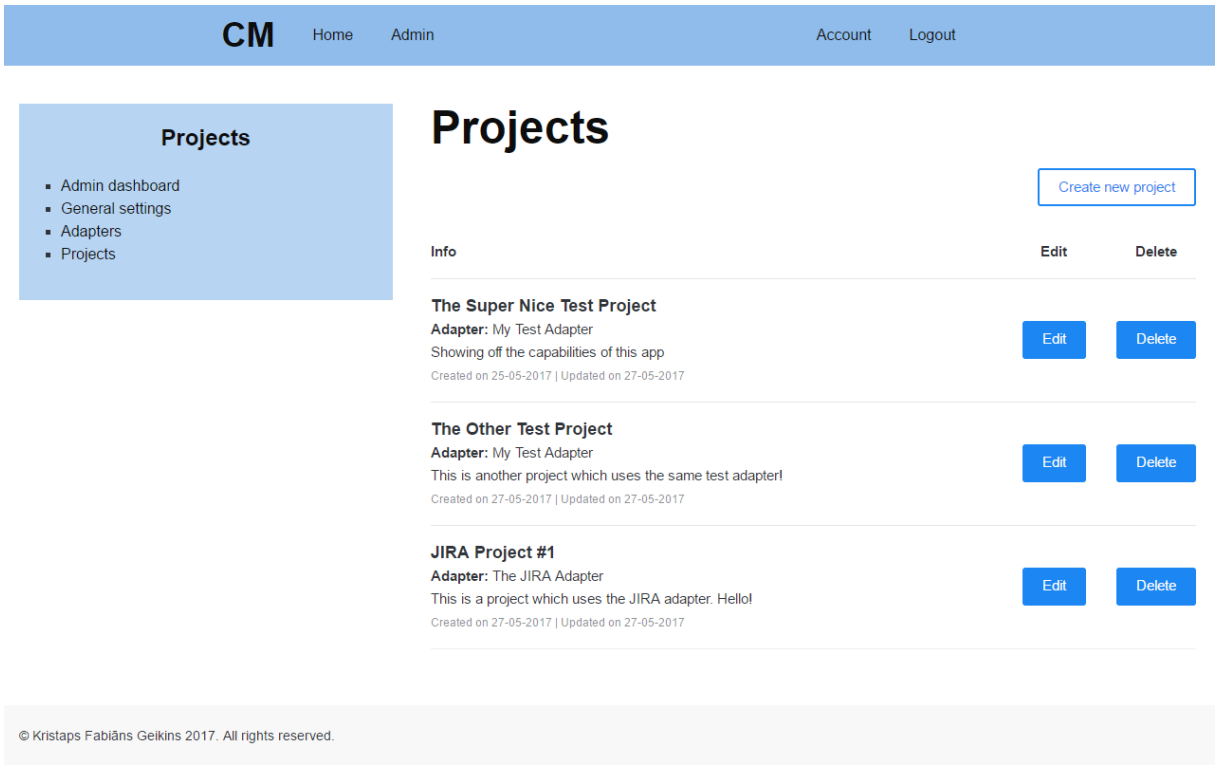
PIELIKUMI

1. Pielikums. Lietotāja saskarnes ekrānuzņēmumi

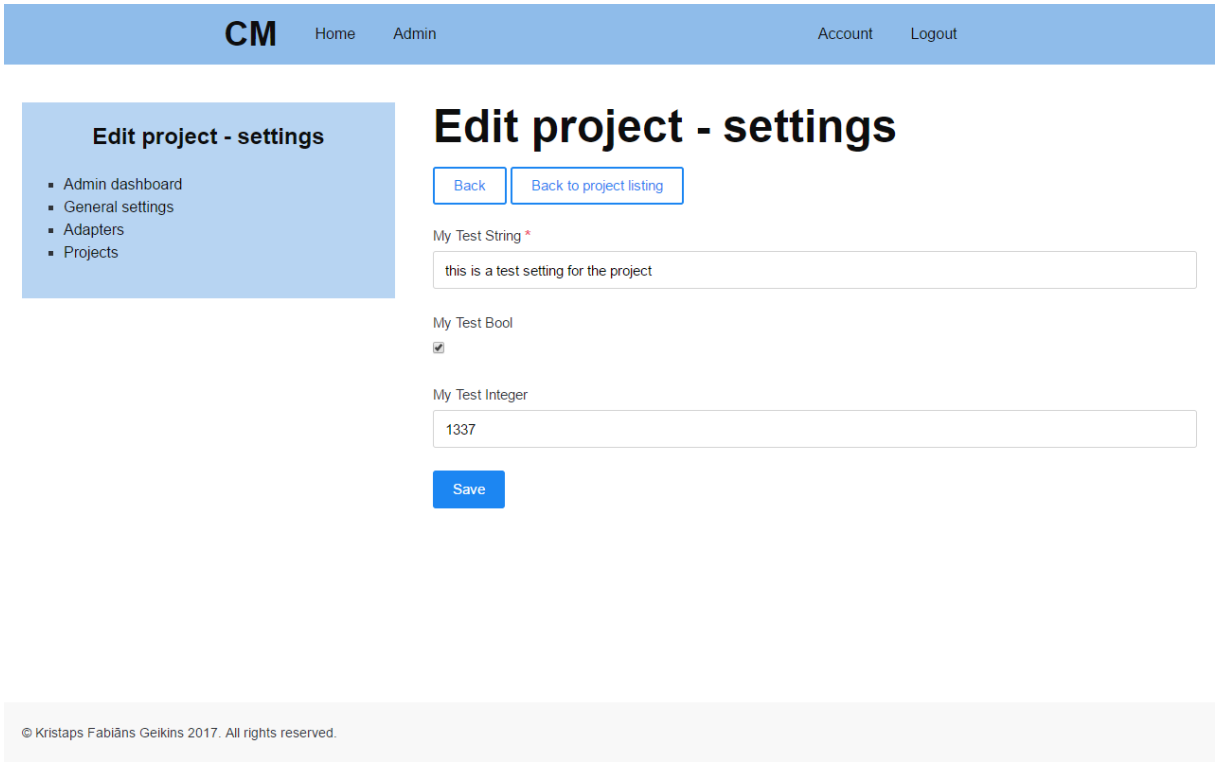
1.1. Apmierinātības rezultātu apskates lapa



1.2. Projektu saraksta lapa



1.3. Projekta adaptera iestatījumu lapa



2. Pielikums. ‘BaseAdapterInterface’ saskarnes pirmkods.

```php
<?php

namespace CustoMood\Bundle\AppBundle\BaseAdapter;

interface BaseAdapterInterface
{
    /**
     * Unique ID of this adapter
     * @return string
     */
    public static function getId(): string;

    /**
     * Display name that will be used in CM (can be null, in which case the ID will be used)
     * @return string
     */
    public static function getDisplayName(): string;

    /**
     * Author
     * @return string
     */
    public static function getAuthor(): string;

    /**
     * Description (can be null)
     * @return string
     */
    public static function getDescription(): string;

    /**
     * Author's website (can be null)
     * @return string
     */
    public static function getWebsite(): string;

    /**
     * Version
     * @return float
     */
    public static function getVersion(): float;

    /**
     * If needed, define any settings that your adapter needs here
     * A single array entry should be structured like this:
     * [
     *      'key' => 'my_first_setting', // Key/Name of the setting. This should be unique within your settings
     *      'value' => '0', // Default value
     *      'display_name' => 'My First Setting',
     *      'type' => 2, // 0 = string, 1 = integer, 2 = boolean
     *      'required' => false, // True, if user has to fill this out


     *      'order' => 1, // Order of this setting within your other settings
     * ]
     * @return array
     */
    public static function getSettingsSchema(): array;

    /**
     * This will get called on load and will be filled with the projectId and also values to the settings
     * defined in getSettingsSchema()
     * @param $projectId string
     * @param $settingValues array
     * @return mixed
     */
    public function load($projectId, $settingValues);

    /**
     * This should return an array of client comments between the two dates. Each array item should
     * follow this format:
     * [
     *      'date' => an integer UNIX timestamp of the comment,
     *      'text' => the actual comment
     * ]
     * @param \DateTime $dateFrom Date from which to get data
     * @param \DateTime $dateTo Date till which to get data
     * @return mixed
     */
    public function getMood($dateFrom, $dateTo): array;
}
```

3. Pielikums. Pirmkoda fragmenti.

3.1. Projektu adapteru iestatījumu rediģēšanas darbība kontrolierī

```
/**
 * @Route("/settings/edit/{pid}", name="admin_project_settings_edit", requirements={"page": "\d+"})
 */
public function settingsEdit(Request $request, $pid)
{
    if ($pid == null)
        throw new NotFoundHttpException();

    $repo = $this->getProjectRepo();
    /** @var Project $project */
    $project = $repo->find($pid);

    if ($project == null)
        throw new NotFoundHttpException();

    // Get settings
    /** @var AdapterService $adapterService */
    $adapterService = $this->get('customood.adapter');
    $adapter = $adapterService->getAdapter($project->getAdapterId());
    if ($adapter == null) {
        $this->addFlash('error', "Can't find project's adapter. Is it installed properly?");
        return $this->redirectToRoute('admin_project_all');

    }
    $settings = ProjectSettingMapper::createProjectSettings($adapter::getSettingsSchema());
    $savedSettings = $project->getSettings();

    // Create form
    $form = $this->createFormBuilder();
    /** @var ProjectSetting $setting */
    foreach ($settings as $setting) {
        switch ($setting->getType()) {
            case SettingType::BOOLEAN:
                $fieldType = CheckboxType::class;
                break;
            case SettingType::NUMBER:
                $fieldType = IntegerType::class;
                break;
            case SettingType::STRING:
            default:
                $fieldType = TextType::class;
                break;
        }

        // Find saved setting value, if any
        $settingValue = null;
        foreach ($savedSettings as $savedSetting) {
            if ($savedSetting->getName() == $setting->getName()) {
                $settingValue = $savedSetting->getFormattedValue();
            }
        }


        $form->add($setting->getName(), $fieldType, [
            'label' => $setting->getDisplayName() ?: $setting->getName(),
            'required' => $setting->getRequired(),
            'data' => $settingValue ?: $setting->getFormattedValue()
        ]);
    }

    $form
        ->add('save', SubmitType::class, array('label' => 'Save'));
    $form = $form->getForm();

    // Handle request, if posted
    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        $results = $form->getData();

        $em = $this->getDoctrine()->getManager();
        $settingsRepo = $em->getRepository(ProjectSetting::class);
```

```
            $form->add($setting->getName(), $fieldType, [
                'label' => $setting->getDisplayName() ?: $setting->getName(),
                'required' => $setting->getRequired(),
                'data' => $settingValue ?: $setting->getFormattedValue()
            ]);
        }

        $form
            ->add('save', SubmitType::class, array('label' => 'Save'));
        $form = $form->getForm();

        // Handle request, if posted
        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
            $results = $form->getData();

            $em = $this->getDoctrine()->getManager();
            $settingsRepo = $em->getRepository(ProjectSetting::class);

            // Now save settings if any
            if (count($results) > 0) {
                foreach ($settings as $oldSetting) {
                    $setting = $settingsRepo->findOneBy([
                        'name' => $oldSetting->getName(),
                        'project' => $project->getId()
                    ]);
                    if ($setting == null) {
                        $setting = $oldSetting;
                        $setting->setProject($project);
                        $em->persist($setting);
                    }

                    if ($setting != null) {
                        $result = $results[$setting->getName()];
                        $setting->setValue($result);
                    }
                }

                $em->flush();

                // Show success message
                $this->addFlash('success', 'Project setting successfully updated!');
            }
        }

        return $this->render('CustoMoodAppBundle::Admin/Project/settings_new.html.twig', [
            'form' => $form->createView(),
            'title' => 'Edit project - settings',
            'project' => $project
        ]);
    }
```

3.2. Instalēto adapteru klašu meklēšanas metode

```php
/**
 * Get/Re-get all added third-party adapters
 * @return array
 */
protected function findValidAdapters()
{
    $adapterLocation = $this->getAdapterLocation();
    $finder = new Finder();
    $finder->files()->in($adapterLocation);
    $result = [];

    foreach ($finder as $file) {
        if ($file->getExtension() != 'php')
            continue;

        $fileName = $file->getBasename('.php');
        $className = $this->getAdapterNamespace() . '\\' . $fileName;

        $potentialAdapter = new $className(); // For some reason without instantiating, class_exists() sometimes fails
        if (!class_exists($className, false))
            continue;

        if (!is_subclass_of($className, BaseAdapterInterface::class))
            continue;

        // Class looks valid
        $result[] = $className;
    }

    return $result;
}
```

3.3. Noklusēto globālo iestatījumu uzstādīšanas datu migrācija

```php
<?php

namespace CustoMood\Bundle\AppBundle\DataFixtures\ORM;

use CustoMood\Bundle\AppBundle\DBAL\Types\SettingType;
use CustoMood\Bundle\AppBundle\Entity\Setting;
use Doctrine\Common\DataFixtures\FixtureInterface;
use Doctrine\Common\Persistence\ObjectManager;

class LoadSettingData implements FixtureInterface
{

    /**
     * Available settings
     */
    const APP_SETTINGS = [
        'registration_open' => [
            'type' => SettingType::BOOLEAN ,
            'displayName' => 'Registration open',
            'value' => 'false',
            'order' => 1
        ],
        'welcome_message' => [
            'type' => SettingType::STRING,
            'displayName' => 'User welcome message',
            'value' => null,
            'order' => 3
        ],
        'maximum_user_amount' => [
            'type' => SettingType::NUMBER,
            'displayName' => 'Max allowed user registrations',
            'value' => 100,
            'required' => true,
            'order' => 2
        ]
    ];

    /**
     * Load data fixtures with the passed EntityManager
     *
     * @param ObjectManager $manager
     */


    public function load(ObjectManager $manager)
    {
        foreach (self::APP_SETTINGS as $key => $settingData) {
            $setting = new Setting();

            $setting
                ->setName($key)
                ->setValue($settingData['value'])
                ->setType($settingData['type'])
                ->setDisplayName($settingData['displayName'])
                ->setSettingOrder($settingData['order']);

            if (array_key_exists('required', $settingData)) {
                $setting->setRequired($settingData['required']);
            }

            $manager->persist($setting);
        }

        $manager->flush();
    }
}
```

3.4. Projektu izveides formas definīcija

```php
<?php

namespace CustoMood\Bundle\AppBundle\Form;

use CustoMood\Bundle\AppBundle\{
    Entity\Project, Service\AdapterService
};
use Symfony\Component\Form\{
    AbstractType, Extension\Core\Type\ChoiceType, Extension\Core\Type\TextareaType,
    Extension\Core\Type\TextType, FormBuilderInterface
};
use Symfony\Component\OptionsResolver\OptionsResolver;

class ProjectType extends AbstractType
{
    /**
     * @var AdapterService
     */
    private $adapterService;

    /**
     * ProjectType constructor.
     * @param AdapterService $adapterService
     */
    public function __construct(AdapterService $adapterService)
    {
        $this->adapterService = $adapterService;
    }

    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $adapters = $this->adapterService->getAdapters();
        $builder
            ->add('name', TextType::class)
            ->add('description', TextareaType::class)
            ->add('adapterId', ChoiceType::class, [
                'choices' => $this->createChoiceArray($adapters),
                'expanded' => false,
                'multiple' => false,
                'label' => 'Adapter'
            ]);
    }

    /**
     * {@inheritdoc}
     */
    public function configureOptions(OptionsResolver $resolver)
    {


        $resolver->setDefaults(array(
            'data_class' => Project::class,
        ));
    }

    /**
     * Create array of valid adapter choices for select field
     */
    protected function createChoiceArray($adapters)
    {
        $choices = [];
        foreach ($adapters as $adapter) {
            $label = $adapter::getDisplayName() ?: $adapter::getId();
            $value = $adapter::getId();

            $choices[$label] = $value;
        }

        return $choices;
    }
}
```

Bakalaura darbs „Parametrizējams lietotāju apmierinātības analīzes rīks” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Kristaps Fabiāns Geikins ___.05.2017.

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītāja: asoc. profesore, Dr. dat. Zane Bičevska _____ ___.05.2017.

Recenzents: Dr.habil.dat., prof. Juris Borzovs

Darbs iesniegts Datorikas fakultātē ___.05.2017.

Dekāna pilnvarotā persona:

vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

___.06.2017. prot. Nr. ___.

Komisijas sekretārs(e):