LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

PROJEKTU PĀRVALDĪBAS SISTĒMA "SPECOMATIC"

KVALIFIKĀCIJAS DARBS

Autors: Artūrs Kurzemnieks

Studenta apl. nr.: ak13153

Darba vadītājs: M.dat. Artūrs Lavrenovs

Rīga 2015

Anotācija

Šis dokuments apraksta kvalifikācijas darba ietvaros izstrādāto projektu pārvaldības sistēmu “Specomatic”. Sistēmas izveide veikta digitālās aģentūras vajadzībām, lai atvieglotu projektu vadītāju darbu, sastādot projektu tāmes un veicot to uzskaiti. Galvenais tās pienesums projektu pārvaldībā ir laika ietaupījums un identisku darbību veikšanas samazinājums, vienkāršojot un standartizējot projektu tāmēšanas procesu un automātiski veidojot gala dokumentus.

Sistēma realizēta kā tīmekļa vietne, tās izstrādei izmantota PHP programmēšanas valoda ar Laravel 4 ietvaru sistēmas servera puses izstrādei.

Abstract

This document describes the project management system “Specomatic” that was developed as part of the qualification work. The development of the system was conducted based on the needs of a digital agency, to facilitate the jobs of its project managers drawing up project price estimates and keeping their records. The main contributions are time savings and the reduction of identical actions, by simplifying and standardizing the process of project estimation, and automatic generation of the final documents.

The system is carried out as website, using the PHP programming language with Laravel 4 framework for the development of its server side software.

Satura rādītājs

Ievads

Kvalifikācijas darbā aprakstītā projektu pārvaldības sistēma “Specomatic” ir tīmekļa vidē realizēta projektu pārvaldības un tāmēšanas sistēma, pēc būtības paredzēta ierobežotam lietotāju lokam, ne plašai un publiskai lietošanai. Sistēma apstrādā un manipulē ar sensitīvu informāciju, paredzētu projektu vadītājiem un par projektu organizāciju un finansiālo pusi atbildīgajām personām ar attiecīgu piekļuvi, tāpēc sistēma, lai arī realizēta tīmekļa vidē, tādējādi atvieglojot piekļuvi un saderību ar dažādām platformām, nav paredzēta publiskai lietošanai un tās sniegtajai funkcionalitātei var piekļūt tikai reģistrēti lietotāji. Reģistrācija sistēmā iespējama tikai ar speciālu administratoru uzaicinājumu.

Sistēmas darbība balstās uz apkalpojamās aģentūras projektu līdzīgo raksturu. Lai arī katram projektam piemīt sava specifika un īpašas prasības, pamatfunkcionalitāte bieži vien atkārtojas, tādeļ tā tiek sadalīta un sistēmā definēta daudzos funkcionālo prasību blokos, kas sistēmā ir savstarpēji saistīti noteiktā hierarhijā. Attiecīgi, hierarhijas augstākajā līmenī atrodas vispārīgākās iespējamās projektu prasības, kā tīmekļa vietnes, mobilās lietotnes vai sociālo tīklu kampaņas nepieciešamība. Šīs projektu funkcijas savukārt iedalās aizvien sīkāk katrā nākošajā hierarhijas līmenī (piemēram, tīmekļa vietnei – satura vadības sistēmas ieviešana, interneta veikala funkcionalitāte). Veidojot jaunu projektu, šāda koka veida struktūra ļauj ātri loģiskā secībā atzīmēt nepieciešamo projekta funkcionalitāti. Tākā katrai funkcijai ir iepriekšdefinēti iesaistītie speciālisti, vidējais darba stundu patēriņš un katra speciālista darba izmaksas, sistēma uzreiz var uzģenerēt projekta pamattāmi Excel izklājlapu formātā, kā arī teksta dokumentus, kas apraksta visas projektā veicamās darbu pozīcijas. Projekta dokumenti pēc tam pieejami lejupielādei un pielāgošanai specifiskām projekta vajadzībām.

Projekta mērķis un izstrādes pamatojums ir vajadzība pēc vienkāršotāka un strukturizētāka projektu izstrādes dzīvescikla, samazinot patērēto laiku projektu tāmju un to skaidrojumu sastādīšanā, vienlaikus atbrīvojoties no vienveidīgākām un daudzkārt atkārtotām darbībām šajā procesā.

Sistēmas izstrāde veikta izmantojot atvērtā pirmkoda risinājumus. Servera puses programmatūra veidota, izmantojot PHP programmēšanas valodu ar uz *MVC* arhitektūras balstīto Laravel 4 ietvaru, papildinātu ar PHPOffice bibliotēkām dokumentu apstrādei. Sistēmu apkalpo MySQL datubāžu vadības sistēma. Lietotāju puses programmatūra realizēta

ar standarta tīmekļa risinājumiem – HTML5 iezīmēšanas valodu, CSS stila valodu un Javascript programmēšanas valodu, papildinātu ar jQuery bibliotēku.

Šis dokuments satur projekta programmatūras prasību specifikāciju, izveidotā programmatūras projekta aprakstu, sistēmas testēšanas dokumentācijas daļu, kā arī programmatūras pirmkoda fragmentus tā pielikumā.

Definīcijas, akronīmi un saīsinājumi

- **MVC** – no angļu val. *Model, View Controller,* programmatūras arhitektūras šablons.
- **CRUD** – no angļu val. *Create, Read, Update, Delete*. Apzīmē četras datu apstrādes un glabāšanas modeļa pamatfunkcijas – ierakstu izveidošanu, nolasīšanu, labošanu un dzēšanu.
- **ORM –** no angļu val. *Object-relational mapping*. Programmatūras izstrādes abstrakcijas līmenis objektorientētās programmēšanas objektu savietošanai ar atbilstošajiem datubāzes datiem.
- **JSON –** *JavaScript Object Notation*. Atvērta standarta cilvēkiem lasāms formāts datu pārsūtīšanai, kas izmanto atslēgu-vērtību pārus.
- **HTTP –** *Hypertext Transfer Protocol.* Lietotņu līmeņa protokols uz kā balstās globālā tīmekļa datu komunikācija.
- **CSRF –** *Cross-Site Request Forgery*. Uzbrukuma veids, izmantojot pieprasījumus no lietotāja puses, lietotājam pašam to neapzinoties un uzticoties kādam ļaundabīgam resursam.
- **XSS –** *Cross-Site Scripting.* Uzbrukuma veids, izmantojot ļaundabīgu skriptu injekciju tīmekļa vietnēs.
- **AJAX –** no *Asynchronous JavaScript and XML*. Tehnoloģija, kas izmanto asinhronus fona pieprasījumus datu apmaiņai no lietotāja puses ar serveri, neizmantojot pilnu lapas pārlādi.

1. Programmatūras prasību specifikācija

1.1 Ievads

1.1.1 Nolūks

Programmatūras prasību specifikācijas nolūks ir uzstādīt un dokumentēt projektu pārvaldības sistēmas "Specomatic" (turpmāk tekstā – *Specomatic)* funkcionālās un nefunkcionālās prasības.

Sistēmas funkcionālās prasības nav veidotas viengabalainā plānošanas fāzē. Tās mainītas un pievienotas pakāpeniski izstrādes laikā, vērtējot sistēmas moduļu un saskarnes lietojamības ērtumu un elastīgumu no projektu vadības viedokļa, rodot jaunus risinājumus un definējot jaunas prasības, kas pievienotas sākotnējām, līdztekus veidojot sistēmu.

Šis dokuments atspoguļo prasību kopsavilkuma formu pamatizstrādes beigu periodā.

Dokuments paredzēts sistēmas izstrādātājiem programmatūras projektējuma izveidei un pasūtītājiem gala produkta atbilstības izvērtēšanai.

1.1.2 Darbības sfēra

Sistēma *Specomatic* ir paredzēta uzņēmuma – digitālās aģentūras – iekšējām vajadzībām. Tai jānodrošina paātrinātu un vienkāršotu projektu tāmēšanas un pārvaldes procesu, modulāri strukturizējot to funkcionālās prasības un automātiski veidojot uzņēmuma standartiem atbilstošus dokumentus.

1.1.3 Saistība ar citiem dokumentiem

Dokuments veidots, vadoties pēc standarta LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis" [1] galvenajām vadlīnijām.

1.1.4 Pārskats

Dokuments strukturēts 3 daļās – ievadā, vispārējā aprakstā un konkrētajās prasībās, kas ietver detalizētāk aprakstītas funkcionālās prasības, drošības un lietotāja saskarnes prasības.

1.2 Vispārējs apraksts

1.2.1 Produkta perspektīva

Specomatic ir patstāvīga un no citiem produktiem neatkarīga sistēma, kas kalpo pasūtītāja iekšējām vajadzībām, taču izvietota kā publiski pieejama tīmekļa vietne, nodrošinot piekļuvi arī ārpus iekšējā tīkla, vienlaikus arī gādājot par datu aizsardzību un pieeju datiem un sistēmas funkcionalitātei tikai autorizētiem lietotājiem – par projektu pārvaldību atbildīgajām personām un sistēmas administratoriem. Jaunu lietotāju izveide un autorizēšana ir pilnā administratoru pārraudzībā. Pašrocīga reģistrācija un piekļuves ieguve nav paredzēta un nav pieļaujama.

Sistēmas darbības laiks un pieejamība tiek uzskatīta par ārēju faktoru, kas jānodrošina pasūtītāja tīmekļa pakalpojumu sniedzējiem un serveru uzturētājiem.

1.2.2 Produkta funkcijas

Sistēmai jānodrošina

Projektu funkcijas

Sistēmai ir jānodrošina pamata CRUD operāciju veikšana projektiem, kā arī jāparedz iespēja neaktīvos projektus arhivēt, nezaudējot informāciju par tiem un uzturot arhivētos projektus pieejamus.

Projektiem un tiem piederošajām tāmēm jābūt savstarpēji saistītiem, taču tehniski atdalītiem modeļiem, abstrahējot konkrētāku informāciju par projekta prasībām, un nodrošinot iespēju nākotnē, sistēmu attīstot, izstrādāt projekta tāmju versiju vēstures pārvaldības moduli, ja šāda vajadzība radīsies.

Tāmju funkcijas

Tāmju CRUD operācijas ir sasaistītas ar projektu modeli, kurš kalpo kā informācijas ietvars. Dzēšot projektu, dzēšas tam piesaistītās tāmes. Veidojot projektu, tiek izveidota un tam piesaistīte arī tāme.

Tāmes veidošanas procesā pēc darba ievadā aprakstītā principa tiek izvēlētas nepieciešamās prasības. Jānodrošina iespēja pieregulēt iepriekšdefinētās prasības projekta vajadzībām jau tā izveides laikā, beigās arī tāmēs generēšana Excel izklājlapu formātā, kā arī apraksta

ģenerēšana DOC un PDF dokumentu formātos.

Excel izklājlapas tāmei jābūt atbilstošai uzņēmuma standarta formatējumam, vēlams izmanto tāmes šablonu, kura sastāvā ir fiksēta satura daļas. Sistēmai ir jābūt spējīgai aizpildīt projekta specifiskās dokumenta sadaļas.

Jānodrošina īssaišu generēšana katrai tāmei, kas dod iespēju sistēmā nereģistrētiem lietotājiem apskatīt tāmi un lejupielādēt tās dokumentus, taču liedz pieeju pārējai sistēmas funkcionalitātei un nesaistītiem datiem.

Klientu funkcijas

Jānodrošina klientu modelis ar pamata CRUD operācijām un piesaisti projektiem, iespēju atšķirot kāda atsevišķa klienta projektus, kā arī izveidot jaunu projektu kā klientu norādot jau datubāzē esošu.

Projekta prasību funkcijas

Jānodrošina projekta prasību CRUD operācijas, sasaiste ar katru prasību veidojošajiem darba uzdevumiem, no kuriem atkarīga prasības darbietilpība un iesaistītie speciālisti. Labojot vai dzēšot projekta prasību entītijas, jānodrošina, lai esošie projekti, kur šīs prasības izmantoti, nemainītu savas īpašības un netiktu zaudēta datu konsistence, t.i., nepieciešamības gadījumā jānodrošina vecāku projekta prasību kopiju saglabāšana sistēmā.

Darba uzdevumu funkcijas

Jānodrošina darba uzdevumu CRUD operācijas. Tiem ir tieša sasaiste ar projekta prasību entītijām, jebkura izmaiņa kādā no darba uzdevumiem vai jauna darba uzdevuma pievienošana kādai projekta prasībai tiek traktēta kā izmaiņa, kas prasa nodrošināt projekta prasību entītijas kopijas saglabāšanu. Darba uzdevumu rediģēšanā maināms tikai stundu skaits. Atšķirīgāka uzdevuma datu komplekta veidošanai paredzēta jauna darba uzdevuma izveide.

1.2.3 Lietotāja raksturiezīmes

Sistēmas lietotājiem nepieciešamas pamatprasmes darbā ar datoru un interneta pārlūkprogrammu. Padziļinātas prasmes nav nepieciešamas. Sistēmas aktīvie lietotāji ir atsevišķi uzņēmuma darbinieki, kuri personīgi tiek instruēti par sistēmas lietošanu un tās darbības pamatprincipiem.

Kā vienīgos iespējamos ārējos sistēmas lietotājus jāparedz iesaistītos klientus, kam var tikt izsūtītas īssaites uz projekta specifikāciju. Šī pieeja dod iespēju tikai apskatīt aktuālo informāciju un lejupielādēt dokumentus, pārējām sistēmas daļām pieeja dota netiek un attiecīgi nav vajadzīgas speciālas zināšanas par sistēmu.

1.2.4 Vispārējie ierobežojumi

Sistēmai jābūt realizētai un pieejamai kā tīmekļa vietnei, bez specifiskām ārējām atkarībām, nodrošinot pēc iespējas elastīgāku pieeju, taču atļaujot tikai autorizētos lietotājus.

Sistēmas izstrāde jāveic ar brīvi pieejamiem, vēlams – atvērtā pirmkoda – risinājumiem.

1.2.5 Pieņēmumi un atkarības

Sistēma uzņēmuma aktīvai lietošanai paredzētā formā tiks izvietota uz uzņēmuma vietējā servera, kur par tās pieejamību atbildēs tīkla administrators, kontrolējot arī tās pieejamību ārējā un iekšējā tīklā. Servera konfigurācija vajadzības gadījumā var tikt pielāgota.

Sistēmas darbībai uz servera pieejami PHP moduļi, MySQL datubāžu vadības sistēma.

1.3 Konkrētās prasības

1.3.1 Funkcionālās prasības

N.B. Ja nav specifiski norādīti konkrēti lielumi, laukiem projektējumā paredz datubāzes noklusētās lauku garuma vērtības.

Stingra datu validācija tiek veikta tikai atslēgas punktos, pamatojoties uz šauru un zināmu lietotāju loku.

1.3.1.1 Reģistrācija

Mērķis
Reģistrē jaunu sistēmas lietotāju. Pieejama tikai caur uzaicinājuma epastu, kuru izsūta sistēmas lietotājs ar administratora tiesībām. Papildina attiecīgajam lietotājam izveidoto lietotāja profilu un padara to lietojamu.
Ievade
• Lietotājvārds (teksta lauks, obligāts) • Parole (paroles teksta lauks, obligāts, vismaz 3 simboli) Parole tiek glabāta jauktā veidā • Atkārtota parole (paroles teksta lauks, obligāts) • Reģistrācijas žetons (netieša ievade, to satur reģistrācijas saite, pēc tā atrod un identificē jauno lietotāju sistēmā)
Apstrāde
Sistēma pārbauda reģistrācijas žetona esamību un atbilstību lietotājam datubāzē. No lietotāja ievades sistēma pārbauda lietotājvārda unikalitāti, lietotājvārda un paroles esamību, paroles atbilstību minimālajām prasībām, kā arī tās sakritību ar atkārtotās paroles lauku, un veic tās sajaukšanu, pirms noglabā datubāzē. Datu atbilstības gadījumā, jaunais lietotājs tiek saglabāts datubāzē, atzīmēts kā sekmīgi reģistrēts.
Izvade
Reģistrācijas žetona neatbilstības gadījumā sistēma atgriež lapas neesamības paziņojumu. Izdošanās gadījumā lietotājs automātiski pierakstīts sistēmā un pāradresēts uz sistēmas sākumskatu. Neizdošanās gadījumā notiek pāradresācija atpakaļ uz reģistrācijas formu, norādot nederīgos laukus.

1.3.1.2 Lietotāja uzaicināšana

Mērķis
Pieejama ar administratora tiesībām. Nosūta uzaicinājumu jauna lietotāja izveidei uz administratora norādītu epastu. Faktiski izveido jaunā lietotāja profilu.
Ievade
• Vārds (teksta lauks, obligāts) • Uzvārds (teksta lauks, obligāts) • Epasts (teksta lauks, obligāts) • Lietotāja loma (izvēles lauks, obligāts, norāda lietotāja tiesības sistēmā, iespējamas pilnas administratora tiesības vai ierobežota lietotāja – novērotāja – tiesības)
Apstrāde
Sistēma datubāzē izveido jaunu lietotāju, pārbauda epasta atbilstību un saglabā to šim lietotājam. Lietotājam tiek saglabāta arī loma, atzīmēts, ka tas vēl nav reģistrējies. Tiek uzģenerēts reģistrācijas žetons un reģistrācijas saite, kas satur šo žetonu.
Izvade
Sistēma izsūta uzaicinājumu ar reģistrācijas saiti uz norādīto epastu.

1.3.1.3 Lietotāja pierakstīšanās

Mērķis
Pieraksta lietotāju sistēmā, lai sāktu darbu
Ievade
• Lietotājvārds (teksta lauks, obligāts) • Parole (paroles teksta lauks, obligāts)
Apstrāde
Sistēma pārbauda lietotājvārda un paroles esamību, cenšas pierakstīt sistēmā atbilstošam lietotājam.
Izvade
Veiksmīgas pierakstīšanās gadījumā lietotājs tiek pāradresēts uz sistēmas sākumskatu. Neizdošanās gadījumā notiek pāradresācija atpakaļ uz pierakstīšanās formu.

1.3.1.4 Lietotāja izrakstīšanās

Mērķis
Izraksta sistēmas lietotāju, beidz darbu.
Ievade
Bez ievadlaukiem, lietotājs piespiež izrakstīšanās pogu grafiskajā saskarnē.
Apstrāde
Sistēma izraksta lietotāju.
Izvade
Pāradresē uz pierakstīšanās formas skatu.

1.3.1.5 Jauna projekta un tāmes izveide

Mērķis
Izveido jaunu projektu sistēmā, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Kategorija (izvēles lauks, iespējams izsaukums jaunas kategorijas izveidei, obligāts) • Klients (izvēles lauks ar esošo klientu sarakstu, iespējams izsaukums jauna klienta pievienošanai sistēmā, obligāts) • Projekta numurs (teksta lauks, obligāts, taču pēc noklusējuma aizpildīts ar sistēmas uzģenerētu projekta numuru, kas sastāv no mēneša identifikatora un kārtas numura, vajadzības gadījumā maināms) • Projekta nosaukums (teksta lauks, obligāts) • Projekta prasības (izvēles rūtiņu koks, obligāta vismaz viena izvēle, iespējama prasību pieregulēšana projekta izveides laikā)
Apstrāde
Pārbauda obligāto lauku esamību. Tukša vai aizņemta projekta numura gadījumā aizvieto ar automātiski ģenerēto, pārbauda projekta nosaukuma unikalitāti. Pēc atzīmētajām prasībām tiek izveidota jauna tāme un uzģenerēti projekta dokumenti, kā arī tāmes īssaite. Atbilstības gadījumā dati tiek saglabāti datubāzē, kā arī atzīmēts lietotājs, kurš projektu izveidoja.
Izvade
Veiksmīgas projekta un tāmes izveides gadījumā pāradresē uz tāmes skatu. Neizdošanās gadījumā atgriež projekta izveides skatā, norādot kļūdainos laukus.

1.3.1.6 Projekta rediģēšana

Mērķis
Labo sistēmā esošu projektu un tam piesaistīto tāmi, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Identiska *1.3.1.5 Jauna projekta un tāmes izveide* ievadei, sākotnēji lauki automātiski aizpildīti ar esošajiem projekta datiem.
Apstrāde
Sistēma līdzīgi projekta izveides funkcijai veic datu pārbaudi un saglabā projekta lauku izmaiņas. Esošās projekta tāmes relācijas ar projektu prasībām tiek dzēstas un to vietā izveidotas jaunas, atbilstoši ievaddatiem. Gala rezultātā sistēma pārģenerē projekta tāmes dokumentus, kā arī projekta ierakstam datubāzē atzīmē lietotāju, kurš to pēdējais rediģēja.
Izvade
Veiksmīgas projekta un tāmes izveides gadījumā pāradresē uz tāmes skatu. Neizdošanās gadījumā atgriež projekta rediģēšanas skatā, norādot kļūdainos laukus.

1.3.1.7 Projekta dzēšana

Mērķis
Neatgriezeniski dzēš projektu un tam piesaistīto tāmi, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē, kā arī atkārtots lietotāja apstiprinājums par projekta dzēšanu.
Apstrāde
Sistēma atrod un dzēš visas relācijas starp projektu prasībām un funkcijas tāmi, tālāk dzēšot arī pašu tāmi un projektu.
Izvade
Pāradresē uz sistēmas sākumskatu.

1.3.1.8 Projekta arhivācija un dearhivācija.

Mērķis
Nomaina projekta aktivitātes statusu, kas attiecīgi to izņem vai ievieto atpakaļ aktīvo projektu sarakstā. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Bez ievadlaukiem, grafiskās saskarnes poga.
Apstrāde
Sistēma atrod projektu un pārbauda tā statusu, attiecīgi nosakot pretēju vērtību.
Izvade
Sistēma pāradresē lietotāju uz iepriekšējo skatu – arhīvu vai aktīvo projektu skatu –, atkarībā no kurienes veikts izsaukums.

1.3.1.9 Projektu kategorijas izveide

Mērķis
Izveido jaunu projektu kategoriju projekta izveides laikā, no kurienes arī tiek izsaukta, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Kategorijas nosaukums (teksta lauks, obligāts)
Apstrāde
Sistēma pārbauda, vai ievadītais nosaukums jau neeksistē datubāzē. Ja tas ir unikāls, tiek izveidota un saglabāta jauna kategorija.
Izvade
Veiksmīgas izveides gadījumā – atdots jaunās kategorijas nosaukums projekta izveides formai. Neveiksmes gadījumā – kļūdas paziņojums par jau eksistējošu kategoriju.

1.3.1.10 Klienta izveide

Mērķis
Pievieno jaunu klientu projekta izveides laikā, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Klienta vārds/nosaukums (teksta lauks, obligāts) • Klienta kontaktpersona (teksta lauks, obligāts) • Klienta epasts (teksta lauks) • Kienta telefons (skaitļa lauks, obligāts)
Apstrāde
Sistēma pārbauda vajadzīgo lauku esamību un saglabā datubāzē jauno klientu.
Izvade
Klienta izveides gadījumā atgriež klienta vārdu projekta izveides formai.

1.3.1.11 Klienta rediģēšana

Mērķis
Rediģē sistēmā esoša klienta informāciju, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Klienta vārds/nosaukums (teksta lauks, obligāts) • Klienta kontaktpersona (teksta lauks, obligāts) • Klienta epasts (teksta lauks) • Kienta telefons (skaitļa lauks, obligāts)
Apstrāde
Sistēma pārbauda vajadzīgo lauku esamību un pārraksta klienta datus datubāzē.
Izvade
Atgriež klienta esošo datu stāvokli.

1.3.1.12 **Klienta dzēšana**

Mērķis
Neatgriezeniski dzēš klientu, kā arī tam piederošos projektus un tāmes, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē, kā arī atkārtots lietotāja apstiprinājums par klienta un tā projektu dzēšanu.
Apstrāde
Sistēma atrod un izsauc visu klientam piederošo projektu dzēšanu, pēctam dzēš no datubāzes arī klienta datus.
Izvade
Pāradresē uz klientu saraksta skatu.

1.3.1.13 **Klienta projektu atlase**

Mērķis
Atlasa un atgriež lietotājam visus kādam konkrētam klientam piederošos projektus, pieejama visiem reģistrētiem lietotājiem.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē.
Apstrāde
Sistēma meklē un atgriež projektus ar atbilstošu norādīto klientu.
Izvade
Atgriež projektu skatu ar atlasītajiem projektiem.

1.3.1.14 Projektu prasības izveide

Mērķis
Pievieno sistēmā jaunu iespējamo projektu prasību ar tai atbilstošajiem darba uzdevumiem, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Nosaukums (teksta lauks, obligāts) • Apraksts (teksta lauks) • Dizaina skati (skaitļa lauks) • Norādījumi dizaineriem (teksta lauks) • Norādījumi tekstu autoriem (teksta lauks) • Darba uzdevumi (identifikācijas numuru masīvs, netieši ievadīts, ievadei skatīt apakšfunkciju *1.3.1.15 Darba uzdevumu izveide*, obligāts vismaz viens definēts darba uzdevums) Noklusēti padod arī prasību hierarhijas vecāka identifikācijas numuru, ja tāds ir (t.i., jaunā projektu prasība nav augstākajā līmenī), nodrošina izveides saskarne, atkarībā kādā pozīcijā jaunā prasība tiek pievienota.
Apstrāde
Sistēma pārbauda obligāto lauku esamību, dizaina skatu lauka skaitlisko vērtību. Atbilstības gadījumā, projektu prasības parametri tiek saglabāti datubāzē. Tiek atrasti atbilstošie darba uzdevumi, kas jau iepriekš izveidoti datubāzē (skatīt *1.3.1.15 Darba uzdevumu izveide*) un tiem kā ietverošā projektu prasība atzīmēta jaunizveidotā.
Izvade
Veiksmīgas izveides gadījumā pāradresē uz projektu prasību skatu. Kļūdas gadījumā atpakaļ uz projektu prasības izveides skatu, norādot kļūdainos ievades laukus.

1.3.1.15 Darba uzdevumu izveide

Mērķis
Izveido jaunu darba uzdevumu, ko pievienot jaunizveidotajai projekta prasībai, pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Nosaukums (teksta lauks, obligāts) • Stundas (vidējais stundu patēriņš uz šo uzdevumu, skaitļa lauks, obligāts) • Speciālists (norāda iesaistīto speciālistu, izvēles lauks, obligāts, iespēja arī pievienot jaunu, iepriekš nedefinētu speciālistu)
Apstrāde
Sistēma pārbauda obligāto lauku esamību un datu atbilstību, izveido un saglabā datubāzē jaunu darba uzdevumu, uz esošo brīdi nepiesaistītu nevienai projektu prasībai.
Izvade
Darba uzdevuma izveides gadījumā atgriež un nodod projektu prasības izveides formai jaunā darba uzdevuma identifikācijas numuru. Kļūdas gadījumā atgriež kļūdas ziņojumu par nederīgajiem ievadlaukiem.

1.3.1.16 Speciālista pozīcijas izveide

Mērķis
Sistēmas datubāzē ir definētas uzņēmuma standarta speciālistu pozīcijas. Gadījumā, ja kāds projekts vai specifiska prasība rada nepieciešamību pēc papildus speciālista piesaistes, kurš nesakrīt ar kādu no iepriekšdefinētajiem, ir iespējams pievienot jaunu. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Pozīcijas nosaukums (teksta lauks, obligāts) • Stundas likme (naudiskā nozīmē, skaitļa lauks, obligāts)
Apstrāde
Sistēma pārbauda obligāto lauku esamību un datu atbilstību, izveido un saglabā datubāzē jauno speciālista pozīciju.
Izvade
Atgriež jaunizveidotās speciālista pozīcijas datus darba uzdevuma izveides formai projektu prasības izveides skatā. Kļūdas gadījumā atgriež kļūdas ziņojumu par nederīgajiem ievadlaukiem.

1.3.1.17 Projektu prasības rediģēšana

Mērķis
Rediģē kādu no definētajām pamata projektu prasībām. Svarīgi saglabāt datu konsistenci, dublējot vecākās projektu prasību versijas, kas izmantotas kādā no projektiem. Šī funkcija rediģē projektu prasības pamatformu (skatīt *1.3.1.18 Projektu prasības regulēšana* par projektam specifisku izmaiņu veikšanu prasībām projekta izveides laikā). Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Ievaddati sakrīt ar *1.3.1.14 Projektu prasības izveide* ievaddatiem. Lauki pēc noklusējuma aizpildīti ar esošajiem projektu prasības datiem.
Apstrāde
Sistēma pārbauda ievades lauku datu atbilstību. Pirms datubāzē tiek saglabātas izmaiņas, sistēma pārbauda, vai konkrētā projektu prasība šajā stāvokli netiek izmantota kādā no esošajām projektu tāmēm. Tādā gadījumā šī projektu prasības versija tiek dublēta un saglabāta kā vecāka prasības kopija, kas vairs nav aktīvā lietošanā, taču tiek izmantota iepriekšējos projektos, lai tie nemainītu savas īpašības. Kad vajadzīgie dati ir saglabāti, vai, ja šāda nepieciešamība nav bijusi, no funkcijas ievaddatiem tiek pārrakstīta konkrētās projektu prasības informācija un saglabāta datubāzē.
Izvade
Veiksmīgas izveides gadījumā pāradresē uz projektu prasību skatu. Kļūdas gadījumā atpakaļ uz projektu prasības rediģēšanas skatu, norādot kļūdainos ievades laukus.

1.3.1.18 Projektu prasības regulēšana

Mērķis
Veic specifiskas izmaiņas un izveido jaunu konkrētam projekta paredzētu prasības versiju projekta izveides laikā, t.i., veidojot jaunu projektu un atzīmējot tā vēlamās prasības, tām ir iespējams veikt pielāgojumus projekta vajadzībām, nerediģējot projektu prasības pamatformu un attiecīgi neietekmējot citus projektus. Lietotājam iespējams izvēlēties arī kādu no iepriekš konfigurētajām prasības versijām, kas izmantota kādā citā projektā, kā arī balstīt jauno versiju uz kādas citas. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.

Ievade
• Versijas nosaukums/identifikators (teksta lauks, neobligāts) Pārējie ievaddati sakrīt ar *1.3.1.14 Projektu prasības izveide* ievaddatiem. Lauki pēc noklusējuma aizpildīti ar esošajiem bāzes projektu prasības datiem, vai arī kādas citas izvēlētās prasības versijas datiem.
Apstrāde
Sistēma pārbauda ievades lauku datu atbilstību. Versijas nosaukuma nenorādīšanas gadījumā tas tiek aizvietots ar jaunā projekta automātiski ģenerēto projekta numuru kā projekta prasības versijas piederības identifikatoru. Sistēma izveido jaunu projekta prasību, kuru atzīmē kā attiecīgās bāzes prasības versiju, saglabā datus.
Izvade
Veiksmīgas izveides gadījumā atgriež projekta izveides skatā ar norādītu jauno prasības versiju. Kļūdas gadījumā atpakaļ uz projektu prasības regulēšanas skatu, norādot kļūdainos ievades laukus.

1.3.1.19 Darba uzdevumu rediģēšana

Mērķis
Rediģē darba uzdevuma norādīto stundu patēriņu. Pārējo atribūtu izmaiņas nav paredzētas, tādā gadījumā jāveido jauns darba uzdevums. Ņemot vērā darba uzdevumu tiešo atkarību no tos ietverošajām projektu prasībām, ir ļoti svarīgs rediģēšanas konteksts. Rediģējot darba uzdevumu, kas tikko pievienots izveidē esošai projektu prasībai, tā ir vienkārša datu atjaunināšanas funkcija. Ja darba uzdevums tiek rediģēts projektu prasības rediģēšanas vai regulēšanas kontekstā, tā ir augstāka – projektu prasību līmeņa – izmaiņa, kas var izsaukt projektu prasību rezerves kopiju izveidi, kuras laikā attiecīgi tiek dublēts arī rediģējamais darba uzdevums, pirms tajā neatgriezeniski ir veiktas izmaiņas. Pēc būtības šī pārbaude notiek augstākā līmenī un faktiski funkcija nodarbojas tikai ar darba uzdevumu datu saglabāšanu, tomēr šie procesi ir cieši saistīti un darba uzdevuma rediģēšana ir plašākas funkciju hierarhijas pamats. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
• Stundas (vidējais stundu patēriņš uz šo uzdevumu, skaitļa lauks, obligāts)
Apstrāde
Sistēma pārbauda lauka esamību un datu atbilstību. Atkarībā no darba uzdevuma ietverošās projektu prasības konteksta (veidojot vai neveidojot darba uzdevuma rezerves kopiju), tiek saglabātas darba uzdevuma izmaiņas.

Izvade
Darba uzdevuma veiksmīgas rediģēšanas gadījumā atgriež atjaunotos datus projektu prasības rediģēšanas skatam. Kļūdas gadījumā atgriež kļūdas ziņojumu par nederīgajiem ievadlaukiem.

1.3.1.20 Darba uzdevumu dzēšana

Mērķis
Dzēš darba uzdevumu. Izšķir funkcijas darbības kontekstu līdzīgi kā *1.3.1.19 Darba uzdevumu rediģēšana*, vajadzības gadījumā veidojot augstāka līmeņa – projektu prasības un tai piederošo darba uzdevumu rezerves kopijas. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē, kā arī atkārtots lietotāja apstiprinājums darba uzdevuma dzēšanai.
Apstrāde
Sistēma atrod darba uzdevumu un pārbauda tā lietojumu. Atkarībā no darba uzdevuma ietverošās projektu prasības konteksta (veidojot vai neveidojot darba uzdevuma rezerves kopijas), dotais darba uzdevums tiek dzēsts.
Izvade
Darba uzdevuma veiksmīgas dzēšanas gadījumā atgriež atjaunotos datus projektu prasības rediģēšanas skatam.

1.3.1.21 Projektu prasības dzēšana

Mērķis
Dzēš kādu no sistēmā definētajām projektu prasībām. Līdzīgi projektu prasības rediģēšanas funkcijai, arī šai ir jāpārbauda dzēšamās prasības lietojums kādā no esošajiem projektiem un vajadzības gadījumā jāveic tās rezerves kopijas izveide, pirms tiek dzēsta pamatentītija. Pieejama reģistrētiem lietotājiem ar administratora tiesībām.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē, kā arī atkārtots lietotāja apstiprinājums projektu prasības dzēšanai.

Apstrāde
Sistēma pārbauda, vai konkrētā projektu prasība šajā stāvokli netiek izmantota kādā no esošajām projektu tāmēm. Tādā gadījumā šī projektu prasības versija tiek dublēta un saglabāta kā vecāka prasības kopija, kas vairs nav aktīvā lietošanā, taču tiek izmantota iepriekšējos projektos, lai tie nemainītu savas īpašības. Kad vajadzīgie dati ir saglabāti, vai, ja šāda nepieciešamība nav bijusi, projekta prasība savā pamatformā tiek dzēsta no datubāzes un attiecīgi vairs netiek atspoguļota aktīvajā projekta prasību hierarhijas kokā.
Izvade
Veiksmīgas dzēšanas gadījumā pāradresē uz projektu prasību skatu ar atjauninātajiem datiem.

1.3.1.22 Projekta tāmes izklājlapu pieprasīšana

Mērķis
Pieprasa no sistēmas lejupielādēt tāmes dokumentu, pieejama reģistrētiem lietotājiem, kā arī nereģistrētiem lietotājiem, kas piekļuvuši tāmei caur publisko tāmes īssaiti.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē
Apstrāde
Sistēma pārbauda, vai tāmes fails eksistē, vajadzības gadījumā vēlreiž palaiž tā ģenerēšanu, kā arī pārliecinās par lietotāja tiesībām piekļūt failam. Ja lietotājs ir pierakstījies sistēmā, vai arī tas ir nācis caur projekta tāmes īssaiti, tiek padots fails.
Izvade
Lejupielādes tiesību gadījumā – Excel izklājlapu formāta fails (.xlsx) Autorizācijas kļūdas gadījumā pāradresē uz sistēmas pierakstīšanās formu.

1.3.1.23 Projekta tāmes apraksta dokumentu pieprasīšana

Mērķis
Pieprasa no sistēmas lejupielādēt tāmes apraksta dokumentus kādā no pieejamiem formātiem, pieejama reģistrētiem lietotājiem, kā arī nereģistrētiem lietotājiem, kas piekļuvuši tāmei caur publisko tāmes īssaiti.

Ievade
Bez ievadlaukiem, pogas grafiskajā saskarnē – attiecīgi .docx vai .pdf ieguvei.
Apstrāde
Sistēma pārbauda, vai apraksta dokumenti eksistē, vajadzības gadījumā vēlreiž palaiž to ģenerēšanu, kā arī pārliecinās par lietotāja tiesībām piekļūt failiem. Ja lietotājs ir pierakstījies sistēmā, vai arī tas ir nācis caur projekta tāmes īssaiti, tiek padots viens no pieprasītajiem failiem.
Izvade
Lejupielādes tiesību gadījumā – teksta dokuments (.docx vai .pdf) Autorizācijas kļūdas gadījumā pāradresē uz sistēmas pierakstīšanās formu.

1.3.1.24 Projekta tāmes īssaites pieprasīšana

Mērķis
Pieprasa no sistēmas īssaiti uz tāmi, ar ko var dalīties un kas sniedz iespēju nereģistrētiem lietotājiem apskatīt tāmi un tās dokumentus, funkcija pieejama reģistrētiem lietotājiem.
Ievade
Bez ievadlaukiem, poga grafiskajā saskarnē
Apstrāde
Sistēma atrod konkrētajai tāmei ģenerēto nejaušo simbolu kombināciju, kas tiek izmantota kā piekļuves žetons un ievietota īssaitē.
Izvade
Projekta tāmes īssaite.

1.3.1.25 Projekta tāmes viesa pieprasījums

Mērķis
Pieprasa no sistēmas konkrētās tāmes skatu un failu lejupielādes tiesības, izmantojot tās īssaiti, taču nodrošinot nereģistrētu lietotāju norobežošanu no citiem datiem un sistēmas funkcijām.
Ievade
Netieši – projekta tāmes piekļuves žetons tāmes īssaites sastāvā.

Apstrāde
Sistēma pārbauda, vai īssaitē esošais piekļuves žetons (simbolu kombinācija) ir piederošs kādai projekta tāmei. Ja tā, lietotāja sesijā tiek noteikts speciāls viesa statuss šai tāmei, kas dod pieeju tikai ierobežotam konkrētās tāmes skatam.
Izvade
Nepierakstījušamies lietotājiem ar viesa statusu – ierobežots tāmes skats, bez izvēlnēm un pieejas pārējai sistēmai, taču ar failu lejupielādes iespējām. Pierakstījušamies lietotājiem – parasts tāmes skats, ignorējot viesa statusu. Neautorizētiem lietotājiem – neatrastas lapas kļūda.

Kļūdu paziņojumi

Individuāli nederīgo ievades lauku kļūdu paziņojumi saskaņoti netiek, to noformēšana atstāta izstrādātāja kompetencē. Paziņojumiem vēlams būt īsiem un vienkāršiem, piemēram, par nepareizu ievades lauka vērtību – formā “Nepareizs [ievades lauka nosaukums]” –, vai “Jāaizpilda obligāti” obligātajiem laukiem.

1.3.2 Drošība

Sistēmai jānodrošina pamata līmeņa drošības pasākumi pret galvenajiem tīmekļa vides apdraudējumiem – CSRF un XSS uzbrukumiem, SQL injekciju veikšanu [2].

Lietotāju paroles datubāzē jāglabā jaucējsummas veidā.

Padziļinātu drošības pasākumu ieviešana nav vitāli nepieciešama, pamatojoties uz sistēmas mazo mērogu, kas izriet no minimālā lietotāju skaita un tiešā veidā vērtīgu datu, kuru noplūde varētu nest reālus zaudējumus, neesamības.

Sistēmas funkcionalitātei un datiem jābūt pieejamiem tikai reģistrētiem un autorizētiem lietotājiem ar attiecīgu pieeju, kas, ja nav norādīts citādāk, jāpārbauda, izpildot lietotāju pieprasījumus. Sistēmas lietotāju lokam jābūt ierobežotam un administratora kontrolētam. Jaunu lietotāju reģistrācijai jābūt iespējamai tikai ar administratora speciālu uzaicinājumu. Brīvas pieejas reģistrācija nav paredzēta.

1.3.3 Lietotāja saskarne

Sistēmas lietotāja saskarne ir tīmekļa vietne, ar galveno uzsvaru uz vienkāršu un ērtu lietojamību. Lietošanas gatavībā uzstādītai un ar izejas datiem nokomplektētas sistēmas lietošanai jābūt vieglai un praktiskā pielietojumā ievērojami ātrākai par dokumentu izveidi manuāli, lai atsvērtu sistēmas izveidi un racionalizētu ikdienas lietošanu.

Viegli lietojams šīs sistēmas kontekstā – saprotami organizēts un darbībā vienkāršs sistēmas lietotāju auditorijas atzinumā, empīriskā ceļā. Neprasa nekādas programmēšanas iemaņas vai no lietojamības viedokļa specifisku jaunu darbību apguvi.

Ieteicama Javascript un AJAX pieprasījumu lietošana saskarnes izveidē, lai padarītu lietošanu estētiskāku un dinamiskāku lietotājam. Datu attēlojuma tabulām jābūt kārtojamām, dalāmām lappusēs un viegli uztveramām. Tā kā sistēma ir slēgtas lietošanas, vietnei nav jānodrošina meklēšanas dzinēju robotu pārvietošanās iespējas.

Vietne primāri paredzēta lietošanai no pārlūka datorā, mobilo ierīču atbalsts nav jānodrošina, taču vietnei jābūt pilnvērtīgi lietojami datoru ekrānos sākot no vismaz 1024 pikseļu pārlūka loga platuma.

Sistēmas skati ir tiešā programmprodukta izstrādātāja kompetencē, iepriekšēja to plānošana un skicēšana netiek paredzēta. Pielāgojumi un izmaiņas tiks veiktas izstrādes laikā, rīkojot apspriedes par saskarnes lietojamību kopā ar pasūtītājiem.

2. Programmatūras projektējuma apraksts

2.1 Ievads

2.1.1 Dokumenta nolūks un darbības sfēra

Šis dokuments paredzēts sistēmas *Specomatic* ieplānotā projektējuma un izvēlēto izstrādes risinājumu aprakstīšanai.

Tā galvenās funkcijas ir kalpot kā vispārīgam ceļvedim sistēmas organizācijā un kā atbalsta dokumentam sistēmas programmatūras izstrādē, nodrošinot sistēmas moduļu plānojumu, informāciju par vēlamo implementācijas funkcionalitāti un lietojamību.

Dokumentā aprakstītā sistēma *Specomatic* kalpo kā projektu tāmju pārvaldības un izveides rīks, kam jāatvieglo un jāpaātrina projektu tāmēšanas un pārvaldības dzīvescikls. Šī dokumenta saturs fokusējas uz konkrētu sistēmas projektējumu un apraksta pieņemtos sistēmas izstrādes un uzbūves lēmumus, balstoties uz dokumentā *1. Programmatūras prasību specifikācija* uzstādītajām funkcionālajām un nefunkcionālajām projekta prasībām un sistēmas darbības un lietojamības vēlamajām prioritātēm.

Dokuments kalpo arī kā atbalsts iespējamu sistēmas paplašinājumu un uzlabojumu plānošanai un projektēšanai nākotnē.

2.1.2 Saistība ar citiem dokumentiem

Dokumenta izveidē un organizācijā ņemti vērā standarta LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai" [3] ieteikumi un galvenās vadlīnijas.

Dokumenta saturs un sistēmas plānojuma lēmumi balstīti uz iepriekš aprakstīto sistēmas programmatūras prasību specifikāciju.

2.2 Dekompozīcijas apraksts

2.2.1 Moduļu dekompozīcija

2.2.1.1 Ārējie moduļi

Sistēma izstrādāta izmantojot Laravel 4 [4] izstrādes ietvaru. Izmantotie ietvara moduļi tiek uzskatīti par daļu no izstrādes vides un sīkāk aprakstīti netiek.

Laravel 4 moduļi veic arī sistēmas drošības nodrošināšanu, pārbaudot sistēmas pieprasījumus un aizsargājot no CSRF un XSS uzbrukumiem, kā arī veic saziņu ar datubāzi, izsargājoties no SQL injekcijām. Šie ir gatavi risinājumi, kuri tiek pieņemti kā uzticami un uzskatīts, ka tālāka testēšana nav nepieciešama.

Sistēmas darbībā funkcionalitātes nodrošināšana tiek izmantoti vairāki gatavi trešo pušu programmatūras moduļi, kuru funkcijas tiek izmantotas sistēmas moduļos datu apstrādē servera pusē:

- *PHPOffice PHPExcel* [5] PHP klase, kas nodrošina funkcijas tiešai manipulācijai ar *Microsoft Excel* izklājlapu formāta failiem un *Specomatic* sistēmā ar tās palīdzību tiek rakstīti dati, veidojot tāmju failus.
- *PHPOffice PHPWord* [6] PHP klase, kas nodrošina funkcijas tiešai manipulācijai ar dokumentu failiem – *Microsoft Word* dokumentiem, kā arī PDF. Tiek izmantota veidojot projekta aprakstu dokumentus.
- *DomPDF* modulis, tiek izmantots kā PDF failu rakstīšanas dzinējs, saglabājot dokumentus PDF formātā.
- *Carbon* trešās puses PHP *DateTime* klases paplašinājums

Klienta puses programmatūrā saskarņu izstrādē izmantoti sekojoši trešo pušu moduļi:

- *Bootstrap* CSS/Javascript modulis vizuālo elementu izstrādei
- *jQuery* Javascript bibliotēka
- *jQuery Datatables* datu tabulu izveidei lietotāja saskarnēs

2.2.1.2 Sistēmas moduļi

Pēc to lietojuma loģikas sistēma iedalīta sekojošos moduļos:

- Projektu un tāmju modulis – ietver aktīvo un arhivēto projektu tabulāro uzskaitījumu, individuālos tāmju skatus, manipulācijas ar projektu entītijām un to izveidi, prasību uzstādīšanu, utilizē projektu prasību moduļa datus.
- Dokumentu ģenerēšanas modulis – ietver tāmju dokumentu veidošanu, utilizē ārējos moduļus dokumentu manipulācijām, sistēmā cieši integrēts ar projektu un tāmju moduli, kurš sniedz savus datus, taču loģikā nošķir funkcionālā atšķirība.
- Klientu modulis – ietver klientu tabulāro uzskaitījumu un manipulācijas ar klientu entītijām, ciešas attiecības ar projuktu un tāmju moduli, veic izsaukumus klientu projektu atlasei.
- Projektu prasību modulis – ietver projektu prasību tabulāro uzskaitījumu un to detalizētu apskati, manipulācijas ar entītijām, kā arī darba uzdevumu apakšmodeli – to attēlojumu un izveidi, operācijas un savstarpējās interkcijas starp darba uzdevumiem un tos ietverošajām projektu prasībām.
- Lietotāju modulis – ietver lietotāju reģistrācijas, pierakstīšanās un uzaicināšanas funkcionalitāti, sistēmas funkciju piekļuves pārbaudes un viesu statusa kontroli.

Izvērsts modeļu plānojuma un lietotāja darba plūsmas apraksts nodaļā *2.3 Detalizētais projektējums*, kurā aprakstītas arī vairākas atsevišķi izdalītas palīgklases.

2.2.2 Datu dekompozīcija

Specomatic sistēmas datu modelim izveidota uz MySQL datubāžu pārvaldības sistēmas strādājoša relāciju datubāze, kas uztur sekojošās sistēmas moduļus apkalpojošās tabulas:

N.B. Tabulu kolonnas, kam pie tipa norādīts *NULL*, nav obligātas, iespējams tās atstāt tukšas, t.i., ar *NULL* vērtību.

Sekojot Laravel izstrādes ietvara praksei, izmantojot datubāzes shēmas izstrādes moduli, visas datubāzes tabulas, arī starptabulas, realizācijā ir aprīkotas ar standartizētu "id" lauku – primāro atslēgu –, kā arī entītijas izveides un pēdējās rediģēšanas laika "zīmogiem" (*timestamps*).

2.2.2.1 "Users" tabula

Sistēmas lietotāju tabula. Glabā visus sistēmā reģistrētos lietotājus.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
remember_token	Varchar(100) *NULL*	Sistēmas nozīmes atribūts, lietotāja atcerēšanās žetons.
firstname	Varchar(255) *NULL*	Vārds
lastname	Varchar(255) *NULL*	Uzvārds
email	Varchar(255) *NULL*	E-pasts
username	Varchar(255)	Lietotājvārds, obligāts
password	Varchar(64)	Lietotāja paroles *Bcrypt* jaucējsumma, obligāts.
token	Varchar(255) *NULL*	Uzģenerētais lietotāja reģistrācijas žetons, tiek izmantots reģistrācijas saitē.

role	Tinyint(4)	Pēc noklusējuma: 2 Lietotāja loma. 0 – *master* statuss, galvenais administrators; 1 – administratora statuss; 2 – novērotāja statuss, redz datus, taču datu liegta lielākā daļa datu manipulāciju iespējas
status	Tiniyint(1)	Pēc noklusējuma: 0 Lietotāja statuss. 0 – vēl nepabeigta reģistrācija; 1 – pabeigta reģistrācija;
position_id	Unsigned int(10) *NULL* Foreign Key (“Positions” tabulas id kolonna)	Ārējā atslēga, norāda uz lietotāja piederību kādai speciālista pozīcijai. Neaktīva.

Piezīmes

Kolonnai “position_id” pašreizējā sistēmas versijā nav funkcionālas nozīmes, tā ieviesta pamatojoties uz funkcionāliem paplašinājumiem tuvā nākotnē, kad paredzēts ieviest iespēju uzņēmuma darbiniekiem kā speciālistiem apskatīt projektus, kuros tie iesaistīti.

2.2.2.2 “Clients” tabula

Glabā informāciju par uzņēmuma klientiem – projektu pasūtītājiem.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
contact	Varchar(255)	Klienta kontaktpersona (vārds, uzvārds), obligāts
email	Varchar(255) *NULL*	Saziņai paredzētais epasts
phone	Int(11)	Saziņai paredzētais telefona numurs, obligāts
name	Varchar(255)	Klienta vārds / nosaukums (ja plašāk kā vienas personas kontekstā, piemēram, uzņēmuma nosaukums), obligāts

2.2.2.3 “Projects” tabula

Glabā informāciju par sistēmā izveidotajiem projektiem.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
name	Varchar(255)	Projekta nosaukums, obligāts
nr	Varchar(255) *NULL*	Projekta numurs
client_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Clients” id kolonna)	Ārējā atslēga, norāda uz klientu, kam pieder šis projekts.
category_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Categories” id kolonna)	Ārējā atslēga, norāda uz kategoriju, kurā ietilpst šis projekts.
archived	tinyint(1)	Pēc noklusējuma: 0 Norāda projekta statusu: 0 – aktīvs 1 – arhivēts (neaktīvs)
created_by	Unsigned int(10) *NULL* Foreign Key (tabulas “Users” id kolonna)	Ārējā atslēga, norāda uz sistēmas lietotāju, kurš izveidojis projektu
edited_by	Unsigned int(10) *NULL* Foreign Key (tabulas “Users” id kolonna)	Ārējā atslēga, norāda uz sistēmas lietotāju, kurš pēdējais rediģējis projektu

2.2.2.4 “Estimates” tabula

Glabā informāciju par sistēmā izveidoto projektu tāmēm.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
project_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Projects” id kolonna)	Ārējā atslēga, norāda uz projektu, kuram pieder attiecīgā tāme.
random	Varchar(255) *NULL*	Nejauši ģenerētā simbolu virkne, ko lieto tāmes īssaitē un kas funkcionē kā viesu piekļuves žetons.

2.2.2.5 “Func-to-est” tabula (starptabula *many-many* tabulu relācijai)

Glabā informāciju par projektu prasību piederību projektu tāmēm – “prasība < > tāme” formas pārus.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.

func_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Functions” id kolonna)	Ārējā atslēga, norāda uz relācijā iesaistīto projektu prasību.
est_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Estimates” id kolonna)	Ārējā atslēga, norāda uz relācijā iesaistīto projekta tāmi, kas izmanto iepriekšminēto projektu prasību.

2.2.2.6 “Functions” tabula

Glabā informāciju par sistēmā definētajām projektu prasībām un to hierarhiju.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
parent_id	Unsigned int(10) *NULL* Foreign Key (šīs pašas tabulas id kolonna)	Ārējā atslēga, norāda uz citu “Functions” tabulas entītiju – projektu prasības “vecāku”, t.i., hierarhijā augstāk stāvošu prasību, kurai šī ir apakšprasība. Augstākā līmeņa funkcijām kolonna saglabā *NULL* vērtību.
fork_base	Unsigned int(10) *NULL* Foreign Key (šīs pašas tabulas id kolonna)	Ārējā atslēga, norāda uz citu “Functions” tabulas entītiju. Tiek lietota specifiski regulētām projektu prasību versijām, norāda uz attiecīgās projektu prasības pamatformu – bāzi, uz kuras veidota šī prasības versija. Projektu prasību pamatformām un pamatformu kopijām kolonna saglabā *NULL* vērtību.

forkname	Varchar(255) *NULL*	Specifiski regulētas projektu prasības versijas nosaukums. Projektu prasību pamatformām un pamatformu kopijām kolonna saglabā *NULL* vērtību.
type	tinyint(4)	Pēc noklusējuma: 0 Norāda projektu prasības tipu: 0 – projektu prasības pamatforma 1 – speciālversija (regulēta forma) 2 – pamatformas kopija, datu saglabāšanas nolūkiem
name	Varchar(255)	Projektu prasības nosaukums, obligāts
description	Text *NULL*	Projektu prasības apraksts
designs	Int(11) *NULL*	Dizaina skatu skaits
design_guide	Text *NULL*	Norādījumi dizaineriem
text_guide	Text *NULL*	Norādījumi tekstu autoriem

2.2.2.7 “Tasks” tabula

Glabā informāciju par sistēmā definēto projektu prasību saturošajiem darba uzdevumiem.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.

func_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Functions” id kolonna)	Ārējā atslēga, norāda uz projektu prasību, kas satur šo darba uzdevumu.
position_id	Unsigned int(10) *NULL* Foreign Key (tabulas “Positions” id kolonna)	Ārējā atslēga, norāda uz speciālista pozīciju, atbildīgu par šī darba uzdevuma izpildi.
name	Varchar(255)	Darba uzdevuma nosaukums, obligāts
hours	Int(11) *NULL*	Darba uzdevuma izpilde vidēji veltāmais stundu skaits, izmantojams tāmes aprēķinos.
description	Text *NULL*	Darba uzdevuma apraksts

2.2.2.8 “Positions” tabula

Glabā informāciju par speciālistu pozīcijām.

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
name	Varchar(255)	Speciālista pozīcijas nosaukums, obligāts
rate	Int(11) *NULL*	Speciālista darba stundas likme

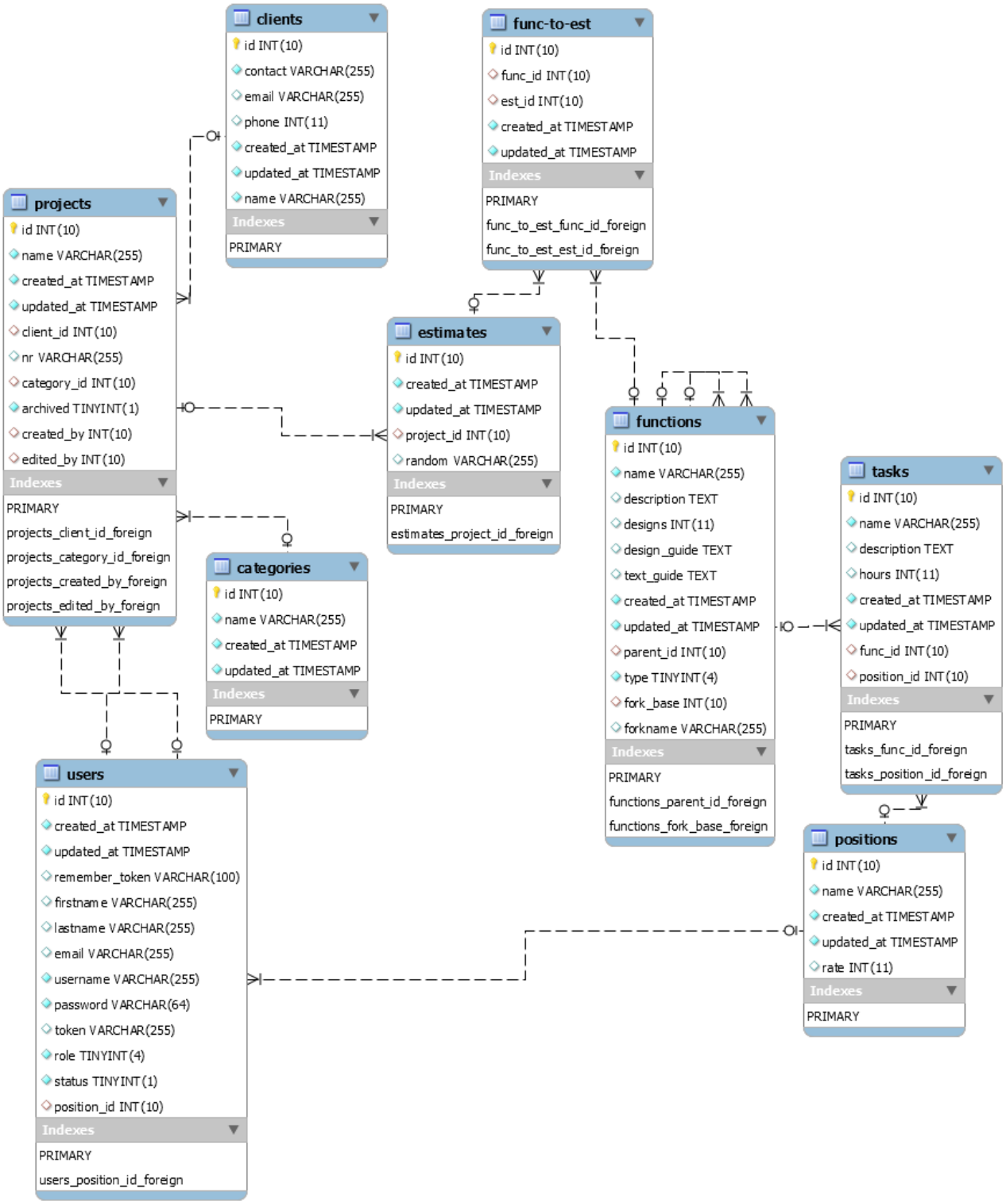
2.2.2.9 “Categories” tabula

Glabā informāciju par projektu kategorijām

Kolonna	Tips	Apraksts
id	Unsigned int(10) Primary Key	*Auto increment.* Automātiski piešķirts entītijas identifikācijas numurs..
created_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē izveides laiku.
updated_at	timestamp	Pēc noklusējuma: 0000-00-00 00:00:00 Sistēmas nozīmes atribūts, fiksē pēdējo izmaiņu laiku.
name	Varchar(255)	Projektu kategorijas nosaukums, obligāts

2.2.3 Datu atkarības

Datubāzes realizācijas ER modelis



2.3 Detalizētais projektējums

Nodaļa apraksta programmprodukta moduļu projektējumu un to ietverošo klašu un metožu darbību. Uzsvars tiek likts uz sistēmas darba plūsmu, izklāstot darbības principus no saskarnes skatpunkta, lai sniegtu pilnvērtīgāku un plašāku skatījumu uz sistēmas projektējumu un moduļu mijiedarbību. Detalizētais projektējums lietojams kopā ar programmatūras prasību specifikācijas konkrētajām funkcionālajām prasībām, kas attiecināmas uz moduļu projektējumu un sniedz vadlīnijas to vēlamajos realizācijas principos.

Sistēmas faili glabājas:

- Kontrolieri – *app/controllers/*
- Modeļi – *app/models*
- Skati – *app/views*
- Vietnes maršruti definēti *app/routes.php* failā
- Javascript, CSS un multimediju faili izvietoti *public/assets* direktorijā.

Visiem sistēmas galvenajiem kontrolieriem, ja vien nav nepieciešams cits risinājums, pieprasījumi tiek pārbaudīti ar lietotāju autorizācijas filtru, kas neautorizētos lietotājus pāradresē uz pierakstīšanās skatu.

Svarīgākā sistēmai specifiskā moduļu funkcionalitāte vieglākai uztveramībai papildināta ar sistēmas ekrānšāviņiem.

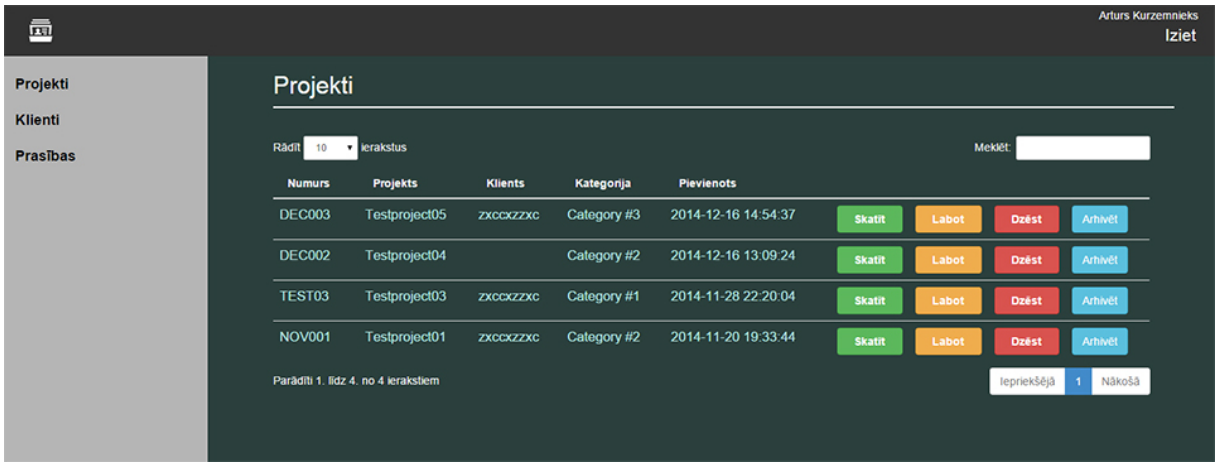
2.3.1 Projektu un tāmju modulis

Sistēmas centrālais modulis.

2.3.1.1 Projektu skats

Pēc noklusējuma vietnes pirmā lapa ir projektu tabulārais kopskats, mājas lapas izsaukumi tiek pāradresēti uz */projects* maršrutu. Šo pašu izsaukumu veic arī sānu izvēlnes "Projekti" saite.

Tiek izsaukta kontroliera *ProjectController.php getIndex* metode, kas atgriež *projects* skatu ar sistēmā esošo projektu datiem.



Attēls 1. Projektu skats

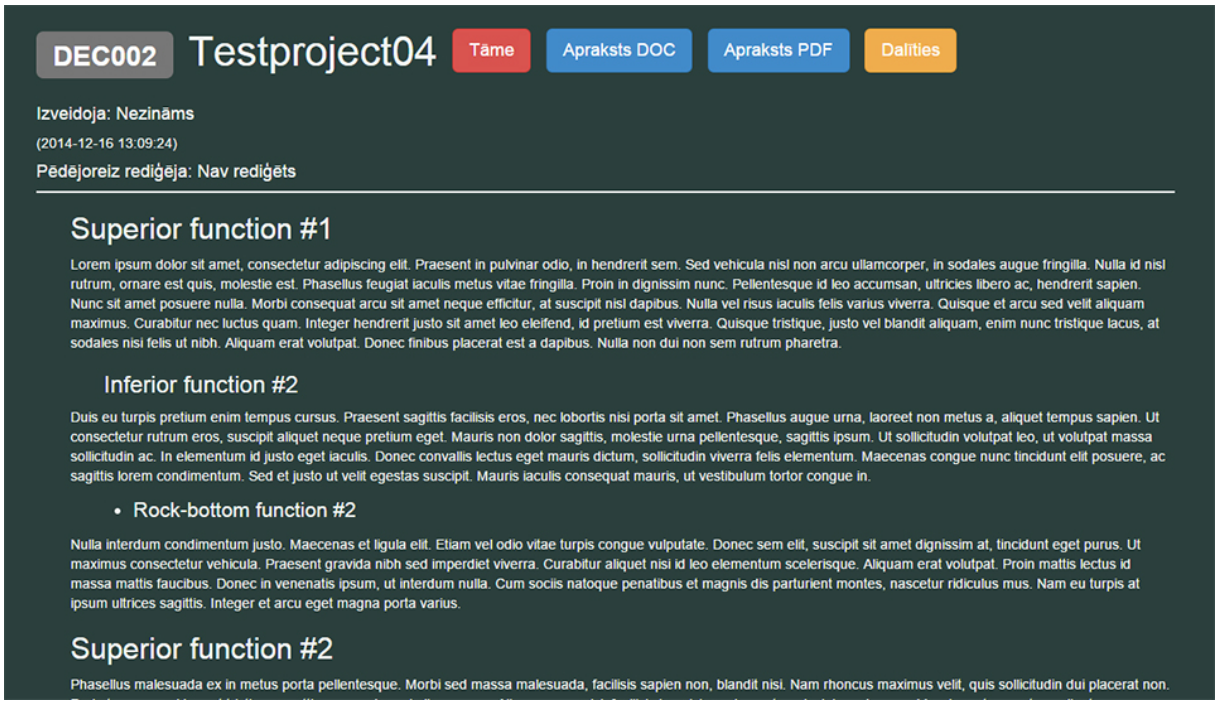
2.3.1.2 Projekta tāmes skats

Projektu "Skatīt" saites norāda uz projekta aktīvo tāmi, maršruts */estimates/[id]*, kur "id" ir aktīvās tāmes identifikators, izsauc kontroliera *EstimateController getEstimate* metodi, parametros nododot tāmes identifikatoru. Metode atgriež galveno tāmes skatu, aizpildot to ar projekta nosaukumu, numuru, autora un rediģēšanas datiem, kā arī tāmes datiem – izmantotajām projektu prasībām un to aprakstiem, dizaina un tekstu autoru ceļvežiem. Saskarnē izvietotas pogas tāmes dokumentu lejupielādei, reģistrētiem lietotājiem arī dalīšanās saites poga, kas uznirstošajā logā parāda tāmes īssaiti.

Skatam viesa statusā var piekļūt ar īssaiti, kas izmanto maršrutu */share/[random]*, kur "random" ir tāmei uzģenerētā simbolu virkne – tās identifikators. Ar to kā parametru tiek izsaukta kontroliera *EstimateController getShare* metode, kas pārbaude, vai dotā virkne atbilst kādai tāmei, tādā gadījumā reģistrējot lietotāja sesijas 'guest' mainīgajā tāmess identifikatoru,

norādot viesa statusu šai tāmei, tad izsaucot *getEstimate* metodi attiecīgajai tāmei. Nederīgas virknes gadījumā tiek atgriezta HTTP 404 kļūda.

GetEstimate metode neautorizētiem lietotājiem ar viesa statusu atgriež nepilnīgu tāmes skatu bez sistēmas izvēlnēm un 'Dalīties' pogas.



Attēls 2. Projektu tāmes skats

Augšējās dokumentu izvēlnes pogas veic izsaukumus, attiecīgi:

- "Tāme" - */estimate/[id]/excel*, izmanto kontroliera *EstimateController.php getExcel* metodi
- "Apraksts DOC" - */estimate/[id]/doc*, izmanto kontroliera *EstimateController.php getDoc* metodi
- "Apraksts PDF" - */estimate/[id]/pdf*, izmanto kontroliera *EstimateController.php getPdf* metodi

Visi trīs izsaukumi ievadē pieņem tāmes identifikācijas numuru "id", tālāk veicot kontroliera *EstimateController.php getFile* privātās metodes izsaukumu, padodot tai tāmes identifikācijas numuru un nepieciešamo faila veidu. Šī metode pārbauda failu eksistenci, vajadzības gadījumā cenšas tos pārģenerēt, kā arī piekļuves tiesības failiem. Nepielaides gadījumā lietotāju pāradresē uz pierakstīšanās skatu.

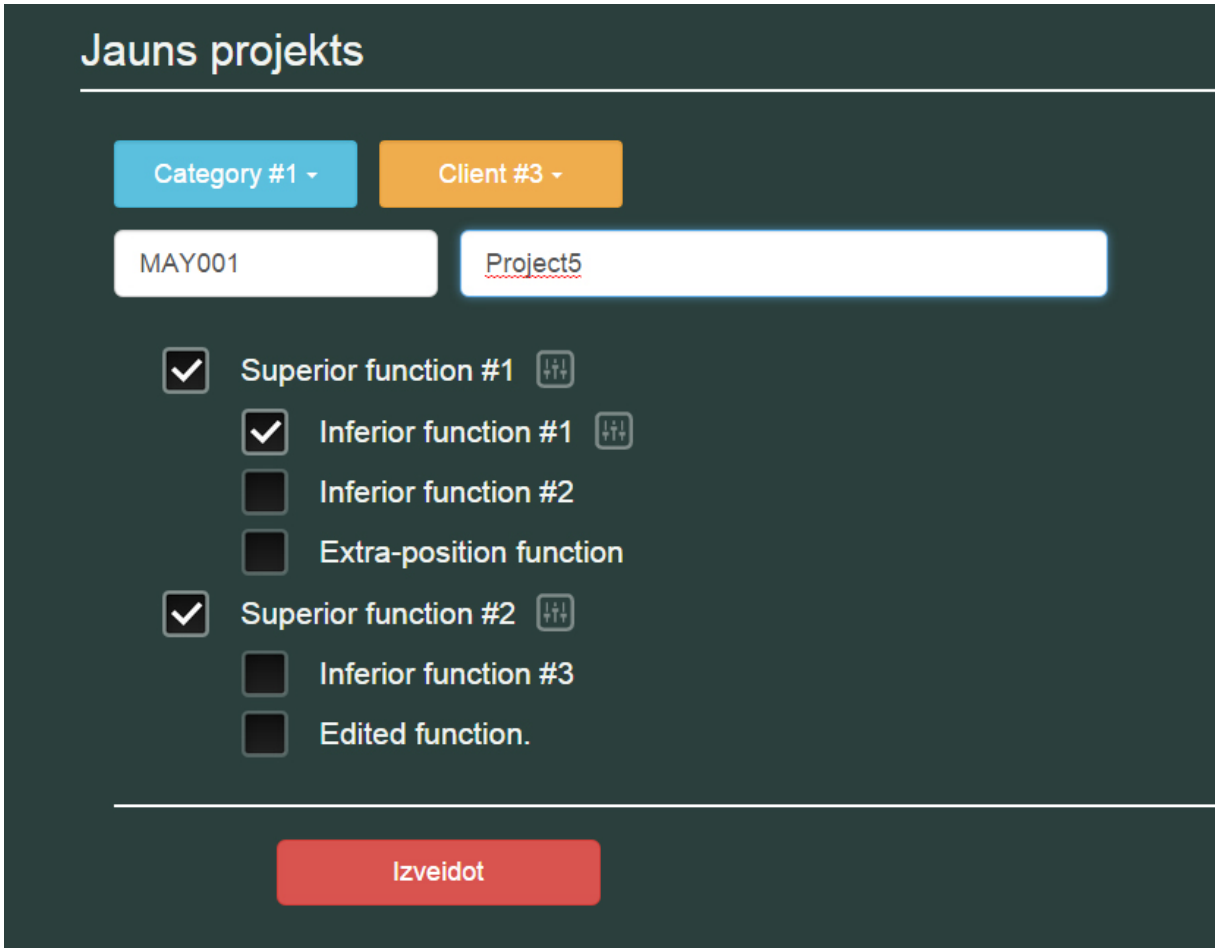
2.3.1.3 Projektu izveide un rediģēšana

Attēls 3. Sānu apakšizvēlne

Sānu izvēlnē, apakšizvēlnē zem "Projekti" saites, novietota saite "Pievienot", kas pieprasa jauna projekta un tāmes izveides skatu maršrutā */projects/create*, kas izsauc kontroliera *ProjectController.php getCreate* metodi. Tā sagatavo datus jauna projekte izveidei – esošo klientu sarakstu, projektu kategorijas, speciālistu pozīcijas, uzģenerē automātisko jaunā projekta numuru –, atgriežot projektu izveides skatu.

No projektu kopskata tabulas izvēloties projekta 'Labot' saiti, maršrutā */projects/[id]/edit* tiek izsaukta kontroliera *ProjectController.php getEdit* metode ar "id" kā tāmes identifikācijas numura parametru. Tā izmanto to pašu projektu izveides skatu, taču pielāgotā rediģēšanas režīmā (norādot atšķirīgus ievades formu parametrus), atspoguļošanai padodot arī rediģējamā projekta datus.

Abām šīs metodes izveides/rediģēšanas skatam nodod arī projektu prasību hierarhiju speciāli izveidotā un sakārtotā Laravel *Eloquent* moduļa kolekcijas objekta formā, kura sagatavošanai metodes izsauc sistēmas palīgmoduli – *functionTree* metodi *app/classes/Sort.php,* kas rekursīvi izstaigā katras projektu prasības (tiek ņemtas vērā tikai pamatformas) dziļākos līmeņus un sakārto tās vajadzīgajā secībā, lai tās pēc tam tiktu pareizi atspoguļotas lietotāja saskarnē.



Attēls 4. Projektu izveides skats

Projektu izveides skatā projektam no izkrītošās izvēlnes izvēlama kategorija, kā arī klients. Izvēlnes apakšā ir "Cita" poga, kas atver uznirstošo logu un ļauj izveidot jaunu kategoriju, tajā ievadāms kategorijas nosaukums. Izveide tiek veikta ar AJAX Post izsaukumu, kuru apkalpo kontroliera *CategoryController.php postNew* metode, kas veic datu pārbaudi un atgriež atbildi par izveides sekmēm. Analogā veidā ar uznirstošā loga ievades formu tiek pievienots arī jauns klients, ko apstrādā kontroliera *ClientController.php postNew* metode (klientu modulis), pārbaudot ievadi un saglabājot klientu datubāzē.

Projektu izveides formā tiek norādīts projekta numurs un nosaukums, numurs pēc noklusējuma aizpildīts ar sistēmas ģenerēto. Zem šīm ievadēs atrodas projektu prasību izvēlne, kur katrai prasībai atbilst sava izvēles rūtiņa. Izvēloties kādu no vēlamajām prasībām, no tās izslīd apakšprasību izvēlne, ja attiecīgajai prasībai hierarhijā ir definētas dziļāka līmeņa prasības. Katrai izvēlētajai prasībai tiek parādīta regulēšanas poga, kas atver uznirstošo logu ar projektu prasības regulēšanas skatu, kas sakrīt ar projektu prasības izveides un rediģēšanas skatu, kas aprakstīts tālāk tekstā.

Jaunizveidojamo projektu dati tiek sūtīti maršrutā */projects/new* kā Post izsaukums, apstrādi veic *ProjectController.php postNew* metode, kas pieņem un apstrādā datus, izveido jaunu projektu, kā arī tam piesaistītu tāmes entītiju, ciklā iziet caur izvēlēto projektu prasību sarakstu un izveido jaunas relācijas starp attiecīgajām projektu prasībām un jauno tāmi. Beigās tiek izsaukta tāmes dokumentu ģenerēšana un lietotājs pāradresēts uz projekta tāmes skatu.

Projektu rediģēšanu veic *ProjectController.php postEdit* metode, kas veic apstrādi analogi izveides metodei, taču pārrakstot esošo entītiju datus.

Izveides un rediģēšanas laikā projektu numura un nosaukuma pieejamība tiek automātiski pārbaudītas ievades laikā, izmantojot AJAX izsaukumus, attiecīgi ar *ProjectController.php postCheckNumber* un *postCheckName* metodēm, kas JSON atbildēs atgriež pieejamības statusu un saskarne neļauj iesūtīt formu ar nederīgu projekta numuru vai nosaukumu.

2.3.1.4 **Projektu dzēšana**

Projektu skata 'Dzēst' saite pēc atkārtota dzēšanas apstiprinājuma izsauc *ProjectController.php postDelete* metodi, kas vispirms dzēš relācijas starp projekta tāmi un projektu prasībām, pēc tam dzēšot projekta un tā tāmes entītitijas (izmantojot projekta modelī implementēto *Project.php deleteProject* metodi).

2.3.1.5 Projektu arhivēšana

Projektu skata "Arhivēt" saites izsauc *ProjectController.php postArchive* metodi, padodot projekta identifikācijas numuru. Metode darbojas abos virzienos, t.i., samaina projekta aktivitātes statusu uz pretēju vērtību. Attiecīgi tas apkalpo arī izsaukumus izņemšanai no arhīva, pēc tam pāradresējot lietotāju uz iepriekšējo skatu (attiecīgi uz projektu skatu vai arhivēto projektu skatu, atkarībā no izsaukuma mērķa).

2.3.1.6 Projektu arhīvs

Arhīvā esošie projekti apskatāmi izsaucot projektu skatu, izmantojot sānu izvēlnes "Projekti" apakšizvēlnes "Arhīvs" saiti, kas izmanto */projects/archive* maršrutu. To apkalpo *ProjectController.php getArchive* metode, kas pēc būtības darbojas tāpat kā parastā projektu skata izsaukums, tikai sākotnēji atlasa projektus, kam datubāzē statuss norādīts kā neaktīvs. Arhīva skatā "Arhivēt" saiti aizvieto "Atjaunot" saite, kas izsauc iepriekš aprakstīto *postArchive* metodi.

2.3.2 Dokumentu ģenerēšanas modulis

Tāmju izveides un rediģēšanas apstrādē, kā arī izsaukumos no failu pieprasījumu apstrādes metodēm, ja vajadzīgie faili nav atrasti, tiek izsaukta tāmes dokumentu ģenerēšana. Programmas pirmkoda struktūrā funkcijas nodrošina pašam tāmes modeļa klasei *Estimate* (*Estimate.php*) izstrādātas metodes, taču dokumentāri aprakstāmas kā speciāls modulis sava funkcionālā nozīmīguma un algoritmiskās sarežģītības dēļ.

2.3.2.1 Excel izklājlapu ģenerēšana

Excel izklājlapu izveidi veic *Estimate.php generateExcel* metode. Tāmes izklājlapas izveide tiek veikta uz veidnes *estimates/template.xlsx*, kurā ir doti noformējuma pamatelementi, standarta speciālistu pozīciju kolonnas, sagatavoti bloki kopējiem tāmes aprēķiniem, kā arī dotas noklusētās pamatprasības.

SĀKOTNĒJAIS IZMAKSU APRĒĶINS / TĀME SASKAŅĀ AR SPECIFIKĀCIJU

AĢENTŪRAS DARBS: Darba nosaukums

Pozīcija	Iesaistīto speciālisti un to darba stundu skaits											Darbu izmaksas
	Projektu vadītājs	Projektu direktors / Stratēģis	Radošais direktors	Dizaineris	Tekstu & ideju autors	Tekstu redaktors	Programmētājs	Animators	3D dizaineris	Sociālo mediju stratēģis	Sociālo mediju projektu vadītājs	
Prasību analīze, sākotnējais izmaksu aprēķins, tehniskās specifikācijas izveide, gala tāmes sastādīšana	3		1				1					[illegible]
Radošās idejas piedāvājums												0
Arhitektūras-struktūras izstrāde	1		1	1								[illegible]
Dizaina virziena piedāvājums (izpēte un skices)	1	1	1	1	1	1	1	1	1	1	1	[illegible]
Dizaina izstrāde (xx skat.)												0
Responsivitātes dizaina versiju izstrāde												0
Satura migrēšana												0
Testēšana un uzlabojumi												0
Sākotnējā satura ievade (1 valoda)												0
Lietotāju apmācības												0
Projekta vadība												0
Speciālista darba stundu skaits	5	1	3	2	1	1	2	1	1	1	1	
Speciālista darba stundas likme EUR	[illegible]											
Speciālista darba izmaksas EUR	[illegible]											
											KOPĀ AĢENTŪRAS DARBS EUR:	[illegible]

RAZOŠANAS IZMAKSAS

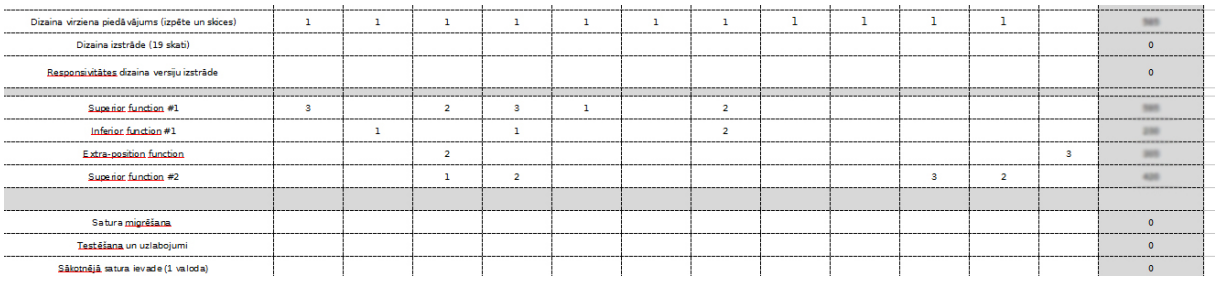
Pozīcija	Sajā tāmes daļā ieklauts	Vienības	Cena EUR	Summa EUR
Hostings	Izmaksa 7 EUR/mēnesī, atkarībā no mēnešu skaita	1	7	7,00
Domēna iegāde	Fiksēta registrācijas un gada maksa, atkarīga no registratūras ?	0	0	0,00
Foto / video iegāde		0	0	0,00
Aģentūras komisija 10%				0,70
			KOPĀ RAŽOŠANAS IZMAKSAS EUR:	7,70
			KOPĀ EUR:	[illegible]
			PVN 21%	[illegible]
			KOPĀ AR PVN EUR:	[illegible]

Attēls 5. Tāmes izklājlapas veidne (sensitīvie dati slēpti)

Metode pēc nepieciešamības pārbīda izklājlapas šūnas, atbrīvojot vietu pievienojamajām projektu prasībām un papildus speciālistu pozīcijām. Katrai izvēlētajai projektu prasībai tiek izveidota sava rinda, kur tiek norādīts tās nosaukums, saskaitīti tajā ietverto darba uzdevumu darba stundu skaits katram speciālistam, kas tiek ierakstīts attiecīgā speciālista kolonnā

prasības rindā. Rindas beigās, balstoties uz katra speciālista stundas likmi un saskaitīto stundu skaitu, tiek aprēķinātas konkrētās projektu prasības darba izmaksas un pievienotas arī kopējai summai.

Beigās tāme tiek saglabāta *estimates/* mapē ar faila nosaukuma shēmu *Taame_*[projekta numurs].*xlsx*



Dizaina virziena piedāvājums (izpēte un skices)	1	1	1	1	1	1	1	1	1	1	1		[illegible]
Dizaina izstrāde (19 skati)													0
Responsivitātes dizaina versiju izstrāde													0
Superior function #1	3		2	3	1		2						[illegible]
Inferior function #1		1		1			2						[illegible]
Extra-position function			2									3	[illegible]
Superior function #2			1	2						3	2		[illegible]
Satura migrēšana													0
Testēšana un uzlabojumi													0
Sākotnējā satura ievade (1 valoda)													0

Attēls 6. Tāmes dati izklājlapā (sensitīvie dati slēpti)

Izklājlapu apstrādē tiek izmantots *PHPOffice PHPExcel* modulis, kā arī to izmantojoša palīgklase *Excel*, kas realizēta *app/classes/Excel.php*. Klases metodes veic tāmes rindu izmaksu summēšanu, kā arī izklājlapas šūnu bloku pārbīdes.

2.3.2.2 Apraksta dokumentu ģenerēšana

Tāmi aprakstošo teksta dokumentu izveidi veic *Estimate.php generateDocs* metode. Tā nebalstās uz šablona, bet gan veido jaunu dokumentu, norāda vajadzīgās galvenes un definē dokumenta stilus.

Dokumenta saturs pēc būtības satur to pašu informāciju, kas sniegta projekta tāmes skatā – uzskaita izvēlētās projektu prasības, izvada to aprakstus, kā arī sniedz dizaineru un tekstu autoru norādes.

Metode dokumenta izveidei un apstrādei izmanto *PHPOffice PHPWord* moduli. Apraksta dokuments tiek saglabāts *estimates/* mapē ar faila nosaukuma shēmu *Apraksts_*[projekta numurs].*docx*

Kad Word dokuments ir saglabāts, metode uzstāda *DomPDF* moduli kā PDF failu rakstīšanas dzinēju *PHPWord* moduļa uzstādījumos un saglabā dokumentu arī kā PDF failu ar faila nosaukuma shēmu *Apraksts_*[projekta numurs].*pdf*

2.3.3 Klientu modulis

2.3.3.1 Klientu skats

Klientu moduļa galvenais skats ir klientu tabulārā uzskaitījuma skats, kurš pieejams no sistēmas sānu izvēlnes "Klienti" saitē, kas ved uz maršrutu */clients*. To apstrādā klientu kontroliera *ClientController.php getIndex* metode, kas no datubāzes iegūst klientu informāciju un atgriež lietotājam klientu skatu.

2.3.3.2 Klienta projektu atlase

Katra klienta tabulas rindā pieejama "Projekti" saite, kas norāda uz maršrutu */clients/*[id]*/projects.* Maršruta forma un lietojuma loģika norāda uz funkcijas piederību klientu modulim, koda struktūrā gan tā ir realizēta projektu kontrolierī, jo pēc programmas funkcionalitātes ir ciešāk saistīta ar to. Attiecīgi izsaukumu saņem *ProjectController.php getClientProjects* metode, kas atlasa projektus, kam kā klients ir norādīts "id" identifikatorā dotais, un atgriež projektu skatu ar vajadzīgajiem projektiem.

2.3.3.3 Klienta izveide

Jaunu klientu pievienošanas process aprakstīts iepriekš – *2.3.1.3 Projektu izveide un rediģēšana.* Tiek veikta no projektu izveides vai rediģēšanas skata, uzreiz ar kādu klientam piederīgu projektu. No klientu kopējā skata pieejama tikai klientu dzēšana un rediģēšana.

Attēls 7. Klientu skats ar rediģēšanas uznirstošo logu

2.3.3.4 Klienta rediģēšana

Izvēloties klienta "Labot" saiti, tiek atvērts uznirstošais logs ar klienta datu formu, kāds tiek lietots arī klienta izveidei no projektu izveides vai rediģēšanas skata. Forma tiek aizpildīta ar klienta esošajiem datiem, izmantojot AJAX Post pieprasījumu */clients/info-id* maršrutā, padodot klienta identifikācijas numuru. To saņem *ClientController.php postInfoId* metode, kas atrod vajadzīgo lietotāju un ar JSON atbildes palīdzību atgriež datus, ar kuriem tiek aizpildīti formas lauki.

Veicot klienta datu izmaiņas, pašu datu saglabāšanu veic *ClientController.php postEdit* metode, kas pārbauda ievadītos datus un saglabā klienta entītijas izmaiņas.

2.3.3.5 Klienta dzēšana

Klients tiek dzēsts no sistēmas, izvēloties "Dzēst" saiti klientu tabulas attiecīgā klienta rindā un apstiprinot atkārtoto vaicājumu par darbības veikšanu. Šo funkciju pilda *ClientController.php postDelete* metode, kam tiek padots dzēšamā klienta identifikācijas numurs, pēc kā tas tiek atrasts. Vispirms tiek izsaukta dzēšana visiem klienta projektiem (kas attiecīgi ķēdē izsauc iesaistīto projektu prasību un tāmes relāciju dzēšanu, kā arī projekta tāmes dzēšanu), pirms no datubāzes tiek dzēsta arī pati klienta entītija.

Metode lietotāju atgriež klientu skatā.

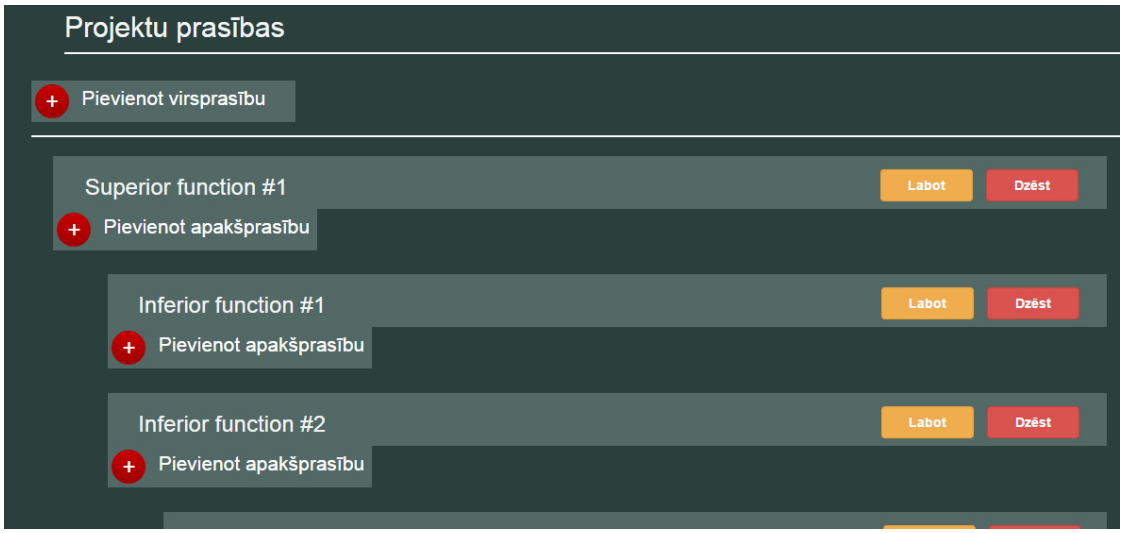
2.3.4 Projektu prasību modulis

2.3.4.1 Projektu prasību skats

Galvenais moduļa skats, pieejams no sānu izvēlnes "Prasības" saites, maršrutā */functions.*

Piezīme: koda izstrādes ietvaros projektu prasībām kā entītijām izmantots termins "funkcijas", atsaucoties uz tām kā projektu funkcionalitātes vienībām.

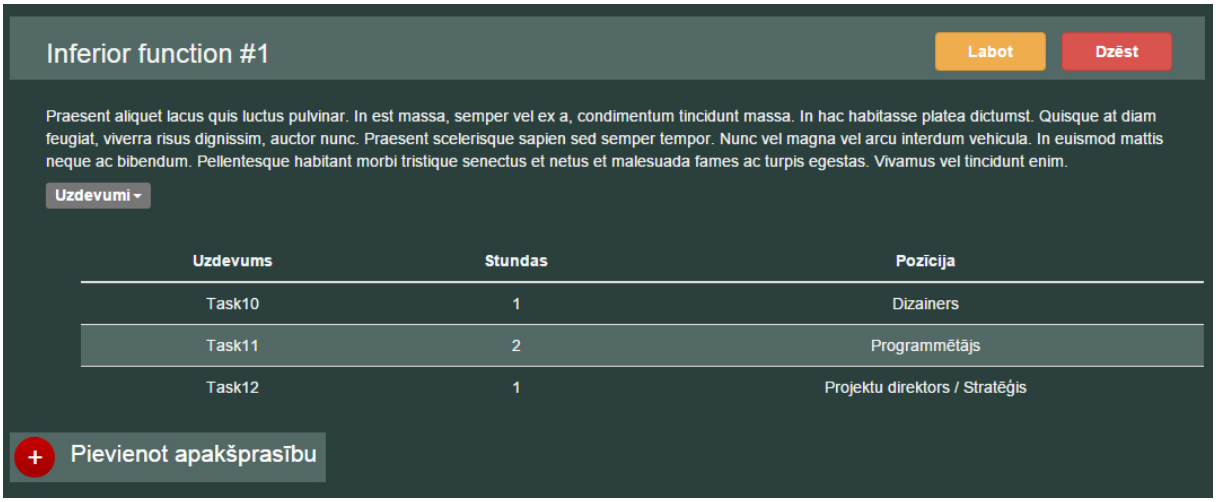
Skatu atgriež *WorkFunctionController.php getIndex* metode.



Attēls 8. Projektu prasību skats

Tā izmanto iepriekš aprakstīto *Sort* palīgmoduli, lai sagatavotu projektu prasību hierarhiju tās vizuālai atveidei.

Skatā tiek parādīts pilns projektu prasību koks, dziļākiem līmeņiem palielinot atkāpi. Katra projektu prasība ir atverama ar peles klikšķi – izslīd prasības informācijas panelis, kurā parādīts prasības apraksts, kā arī dziļāka līmeņa izslīdošais "Uzdevumi" panelis, kuru atverot, tiek parādīti prasībā ietilpstošie darba uzdevumi, tos izpildošie speciālisti un paredzētās darba stundas.



Attēls 9. Projektu prasības izvēršana

2.3.4.2 Projektu prasību un darba uzdevumu pievienošana

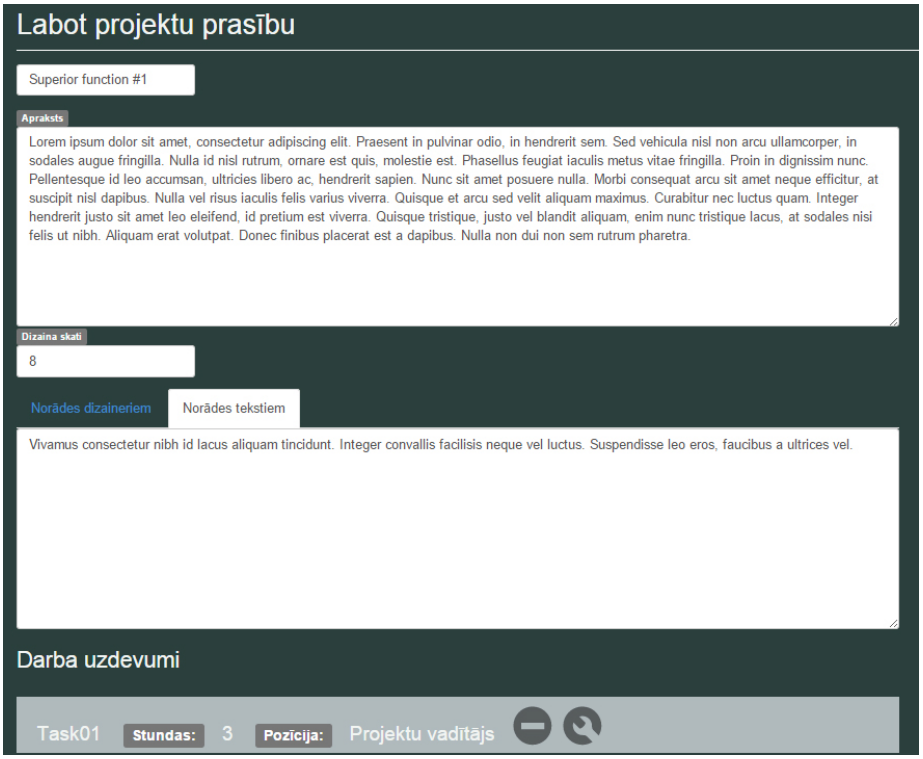
Projektu prasību pievienošanas skats izsaucams no projektu prasība skata, izvēloties kādu no prasību pievienošanas saitēm saskarnē. Saraksta augšā atrodas "Pievienot virsprasību" saite, kas izsauc izveides skatu bez definēta prasības vecāka, attiecīgi tiek pievienota jauna augstākā līmeņa projektu prasība.

Katrai esošajai projektu prasībā sarakstā ir sava "Pievienot apakšprasību" saite, kuru izvēloties tiek izsaukts izveides skats jaunai prasībai, kas hierarhijā atradīsies tieši zem šīs, t.i., būs tās bērns.

"Pievienot virsprasību" norāda uz */functions/new/master* maršrutu, savukārt "Pievienot apakšprasību" – uz */functions/new/[id]*, kur "id" ir vecāka prasības identifikācijas numurs.

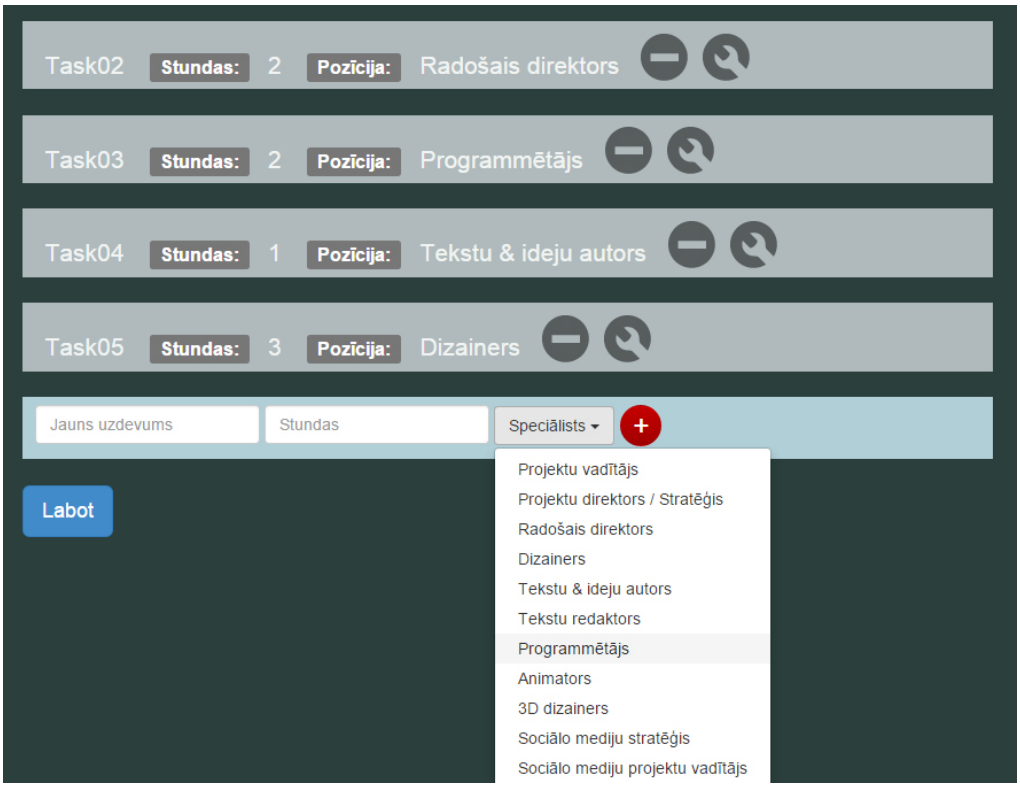
Abos gadījumos pieprasījuma apstrādi veic *WorkFunctionController.php getNew* metode, kas kā vienīgo parametru saņem šo virsprasības identifikācijas numuru. Ja tas vienāds ar "master", metode prasības izveides skatam atdod tukšu slēpto "parent" ievades lauku. Ja šis parametrs ir skaitlis, tiek uztverts, ka tas ir kādas prasības identifikācijas numurs, tādā gadījumā metode pārliecinās, vai projektu prasība ar šādu numuru eksistē, un nosaka tās identifikācijas numuru kā noklusētās ievades "parent" vērtību.

Lietotājam tiek atgriezts jaunas prasības izveidei pielāgots projektu prasības formas skats.



Attēls 10. Projektu prasības forma

Formas sastāvā ir projektu prasības pamatinformācijas ievades lauki (skat. attēlā), zem tiem atrodas darba uzdevumu pievienošanas panelis. Darba uzdevuma pievienošana jaunajai projektu prasībai notiek, ievadot darba uzdevuma nosaukumu, darba stundu skaitu un izvēloties attiecināmo speciālistu, beigās piespiežot pievienošanas “+” pogu.



Attēls 11. Darba uzdevumu forma

Speciālista pozīcija tiek izvēlēta no izkrītošās izvēlnes, kurā uzskaitīti datubāzē esošie, kā arī saraksta apakšā ir izvēle "Cits", kas atver uznirstošo logu jauna speciālista pievienošanai. Tajā norāda speciālista pozīcijas nosaukumu un stundas likmi. Speciālista pievienošana tiek veikta ar AJAX Post izsaukumu, kuru apstrādā pozīciju kontroliera *PositionController.php postNew* metode, kas pārbauda datus, saglabā jauno pozīciju datubāzē un atgriež jaunās pozīcijas identifikācijas numuru JSON atbildē, kas ļauj saskarnē pievienot jauno speciālistu tūlītējai izvēlei jaunā darba uzdevuma izveidē.

Darba uzdevuma izveidi nodrošina uzdevumu kontroliera *TaskController.php postNew* metode, kurai ar AJAX Post izsaukumu tiek nodots jaunizveidojamā darba uzdevuma nosaukuma, stundu skaits un izvēlētā speciālista identifikācijas numurs. Jaunais darba uzdevums, izveidojot norādi uz speciālista pozīcijas entītiju, tiek saglabāts datubāzē un JSON atbildē projektu prasības izveides formai atgriezts jaunā uzdevuma identifikācijas numurs, kas tiek pievienots kā slēpts ievades lauks.

Kad projektu prasības forma aizpildīta, vēlamie darba uzdevumi pievienoti, "Izveidot" poga nodod ievadītos datus prasību kontroliera *WorkFunctionController.php postNew* metodei maršrutā */functions/new*. Tā saņem un pārbauda datus, izveido jaunu projektu prasības entītiju datubāzē. Katram ievadē padotajam darba uzdevuma identifikācijas numuram tiek piemeklēts

tam atbilstošais darba uzdevums, kuram kā ietverošā projektu prasība tiek norādīta jaunizveidotā un darba uzdevuma izmaiņas saglabātas.

Veiksmīgas izveides gadījumā lietotājs tiek pāradresēts uz projektu prasību skatu.

2.3.4.3 Darba uzdevumu rediģēšana un dzēšana projektu prasību izveides kontekstā

Pievienojot jaunu darba uzdevumu projektu prasības izveides laikā, projektu prasībai tas tiek piesaistīts tikai projektu prasības apstrādes laikā *WorkFunctionController.php postNew* metodē. Kamēr jaunās projektu prasības dati vēl nav iesniegti apstrādei, tai pievienotie darba uzdevumi jau ir izveidoti datubāzē, taču ar tukšiem ietverošās prasības laukiem, kas nozīmē, ka šie uzdevumi ir brīvi rediģējami un dzēšami, jo vēl netiek izmantoti nevienā projektā un nevar tikt apdraudēta datu konsistence.

Attiecīgi, izvēloties darba uzdevuma rediģēšanas kontaktpogu, darba uzdevuma ailes stundu pozīcija kļūst rediģējama. Veicot tajā izmaiņas, uzdevuma identifikācijas numurs un stundu skaits AJAX Post pieprasījumā tiek nosūtīts darba uzdevumu kontroliera *TaskController.php postEdit* metodi. Tā kā uzdevums šajā brīdī vēl ir neizmantots, tiek veiktas vienkāršas datubāzes datu izmaiņas uzdevuma entītijai.

Līdzi, izvēloties darba uzdevuma dzēšanas kontaktpogu, darba uzdevums tiek izņemts no jaunai projektu prasībai pievienoto darba uzdevumu saraksta un tiek veikts AJAX Post pieprasījums *TaskController.php postDelete* metodei, kas pēc tai padotā identifikācijas numura atrod un dzēš no datubāzes attiecīgo darba uzdevumu.

2.3.4.4 Projektu prasību rediģēšana

Projektu prasības rediģēšana pieejama, spiežot "Labot" saiti nepieciešamajai projektu prasībai kopējā skatā. Saites maršruts */functions/edit/[id]*, kur "id" ir rediģējamās prasības identifikācijas numurus, izsauc *WorkFunctionController.php getEdit* metodi, kas saņem identifikācijas numuru un atrod atbilstošo projektu prasību, lietotājam atgriežot rediģēšanai pielāgotu projektu prasību formas skatu ar rediģējamās prasības datiem (tai skaitā arī ietilpstošajiem darba uzdevumiem).

Formas ievade tiek veikta tāpat kā projektu prasības izveidē, taču atšķiras darba uzdevumu apstrāde. Pēc noklusējuma formas datiem ir pievienoti esošo darba uzdevumu slēptie ievades lauki. Jaunu darba uzdevumu pievienošanas apstrādē atšķirību nav. Darba uzdevumu rediģēšanas un dzēšanas skaidrojumu skatīt *2.3.4.5 Darba uzdevumu rediģēšana un dzēšana projektu prasību rediģēšanas un regulēšanas kontekstā.*

Atjauninātie projektu prasības dati tiek nodoti apstrādei *WorkFunctionController.php postEdit* metodei, kas pārbauda datus, kā arī nosaka, vai rediģējamā projektu prasība ir izmantota kādā projektā.

Ja tā nav izmantota, dati tiek saglabāti esošajā entītijā. Tiek salīdzināti pašreizējā darba uzdevumu informācija un atjauninātā darba uzdevumu informācija. Darba uzdevumi, kuru identifikācijas numuri nav ievadē, t.i., lietotāja saskarnes pusē tikuši dzēsti vai aizvietoti ar rediģēto kopiju, tiek dzēsti no datubāzes, to vietā nākot atjauninātajiem darba uzdevumiem.

Ja projektu prasība ir izmantota, tiek izsaukta tās modeļa dublēšanas metode *WorkFunction.php forkPreserve,* kas izveido projektu prasības kopiju, atzīmē tās statusu kā datu saglabāšanas kopiju un modificē visas projektu tāmju relācijas ar prasības oriģinālformu, kā attiecināmo projektu prasību tajās norādot jauno kopiju. Metode izveido arī darba uzdevumu kopijas, tās piešķirot jaunizveidotajai projektu prasības kopijai. Visos projektos, kur tikusi izmantota projektu prasības oriģinālforma, tā aizvietota ar pēc īpašībām identisku kopiju.

Projektu prasības oriģinālformā, kā aprakstīts iepriekš, dati tiek pārrakstīti ar jaunajiem ievaddatiem.

2.3.4.5 Darba uzdevumu rediģēšana un dzēšana projektu prasību rediģēšanas un regulēšanas kontekstā

Veicot minētās operācijas darba uzdevumiem, kuri kopā ar ietverošajām projektu prasībām ir tikuši izveidoti iepriekš, pastāv iespēja, ka šīs projektu prasības ir izmantotas kādā projektā un izmaiņas minētajos darba uzdevumos nozīmētu attiecīgo projektu prasību īpašību maiņu un iespējamu datu nekonsistenci.

Izvēloties darba uzdevuma rediģēšanas kontaktpogu projektu prasību rediģēšanas vai projektu prasību regulēšanas skatā, darba uzdevuma ailes stundu pozīcija kļūst rediģējama un izmaiņas tāpat kā iepriekš apskatītajā gadījumā veic darba uzdevumu kontroliera *TaskController.php postEdit* metode.

Atšķiras metodes darbība. Tā kā pastāv iepriekš aprakstītās datu nekonsistences risks, tiek izveidota rediģējamā darba uzdevuma kopija – atsevišķa entītija –, kurai tiek veiktas nepieciešamās izmaiņas, oriģinālo darba uzdevuma formu saglabājot neskartu. Šajā gadījumā projektu prasības forma saņem šīs kopijas identifikācijas numuru, ar kuru aizstāj esošo darba uzdevumu.

Dzēšanas gadījumā netiek veikts reāls kontroliera metodes izsaukums, bet gan izņemts darba uzdevuma slēptais ievades lauks.

Tālāka darba uzdevumu apstrāde nav darba uzdevumu kontroliera , bet gan projektu prasību kontroliera pārziņā, aprakstīta *2.3.4.4 Projektu prasību rediģēšana.*

2.3.4.6 Projektu prasību dzēšana

Projektu prasību skatā izvēloties "Dzēst" kontaktpogu un apstiprinot, ka prasība tiešām ir jādzēš (brīdinājums, ka tas nozīmē arī apakšprasību dzēšanu), tiek padots izsaukums *WorkFunctionController.php postDelete* metodei. Līdzīgi rediģēšanas gadījumam, ir svarīgi, vai attiecīgā projektu prasība ir vai nav izmantota kādā projektā.

Ja nav izmantota, kas nozīmē arī to, ka tās apakšprasības nav izmantotas, tiek izsaukta rekursīva prasību dzēšana, kas izdzēš visas apakšprasības un arī konkrēto prasību, katras prasības dzēšanas procesā vispirms izsaucot tajā ietverto darba uzdevumu dzēšanu.

Ja tā ir izmantota, reāla dzēšana no datubāzes nenotiek, taču šai prasībai un visām tām apakšprasībām statuss tiek norādīts kā datu saglabāšanas kopija, kas nozīmē, ka tās tiek izņemtas no aktīvās projektu prasību hierarhijas un nav izvēlamas jaunu projektu veidošanā.

2.3.4.7 Projektu prasību regulēšana

Kā minēts *2.3.1.3 Projektu izveide un rediģēšana* moduļa funkciju aprakstā, projektu formas skatā katrai izvēlētajai prasībai ir pieejama kontaktpoga tās regulēšanai, kas atver uznirstošo logu. Šis logs satur projektu prasības formu regulēšanas režīmā, tā dati tiek nodoti ar AJAX Post pieprasījumu.

Šī forma ir domāta mikroizmaiņu veikšanai projektu prasībās, pēc noklusējuma aizpildīta ar regulējamās projektu prasības datiem, lai uz vietas tās pielāgotu attiecīgajam projektam. Formas augšpusē papildus pievienots regulētās versijas nosaukuma ievades lauks, kā arī izkrītošā izvēlne ar iepriekš veidotajām šīs projektu prasības regulētajām versijām, kuras identificē to versiju nosaukumi. Šo sarakstu izveido regulēšanas loga atvēršanas laikā, izmantojot AJAX Get pieprasījumu *WorkFunctionController.php getFork* metodi, kurai tiek padots regulējamās prasības (vai tās versijas, ja ir jau izvēlēta kāda cita, kas nav oriģinālforma) identifikācijas numurs. Metode JSON atbildes veidā atgriež gan identifikācijas numuram atbilstošās projektu prasības datus, ar kuriem automātiski tiek aizpildīta forma, gan sarakstu ar visām bāzes prasības versijām, par katru atgriežot nosaukumu un identifikācijas numuru.

Attēls 12. Projektu prasības regulēšanas logs

Ja kādā no laukiem vai darba uzdevumos dati tikuši modificēti (neattiecas, ja tiek izvēlēta kāda no iepriekš veidotajām versijām, pie nosacījuma, ka arī tā netiek tālāk pārveidota), iesniedzot formu, tiek izsaukta *WorkFunctionController.php postFork* metode, kas pēc formas projekta prasības datiem un darba uzdevumu datiem izveido jaunu projektu prasības entītiju, kurai kā versijas bāzes norādi atzīmē prasības oriģinālformu, un norāda statusu kā speciālversijai. Oriģinālformas darba uzdevumi, kas norādīti arī jaunajā regulētajā ievadē, tiek kopēti un kopijas piešķirtas prasības speciālversijai. Jaunie darba uzdevumi tiek piešķirti, kopijas neveidojot. Kad dati saglabāti, projekta formai tiek atgriezts jaunās versijas identifikācijas numurs, kuru izmantot tālākā apstrādē.

Ja tiek izvēlēta kāda no iepriekš veidotajām versijām, tiek atkārtots augstāk aprakstītais *WorkFunctionController.php getFork* AJAX Get pieprasījums, atjauninot formu ar izvēlētas versijas datiem un nomainot projekta formā izvēlēto prasības identifikācijas numuru pret šīs izvēlētas versijas identifikācijas numuru.

Ja datu izmaiņas nav veiktas, izmaiņu kontaktpoga aizver uznirstošo logu.

2.3.5 Lietotāju modulis

2.3.5.1 Lietotāju pierakstīšanās

Visi sistēmā nepierakstītu lietotāju pieprasījumi (izņemot tāmes viesa skata pieprasījumu) tiek pāradresēti uz lietotāju pierakstīšanās skatu maršrutā */login*, ko nodrošina *HomeController.php getLogin* metode. Lietotājus, kas jau ir pierakstījušies, metode pāradresē uz projektu skatu.

Pierakstīšanās apstrādi veic *HomeController.php postLogin* metode, kas pārbauda lauku esamību un cenšas autorizēt lietotāju, izmantojot iesūtītos datus. Veiksmīgas pierakstīšanās gadījumā lietotājs tiek pāradresēts uz viņa iepriekš ieplānoto maršrutu, vai arī projektu skatu, ja tāds nav bijis.

Neveiksmīgas pierakstīšanās gadījumā pāradresē atpakaļ uz pierakstīšanās skatu.

2.3.5.2 Lietotāju izrakstīšanās

Sistēmas augšējās izvēlnes "Iziet" saite izsauc *HomeController.php postLogout*, kas izraksta lietotāju un pāradresē uz pierakstīšanā skatu.

2.3.5.3 Lietotāju skats

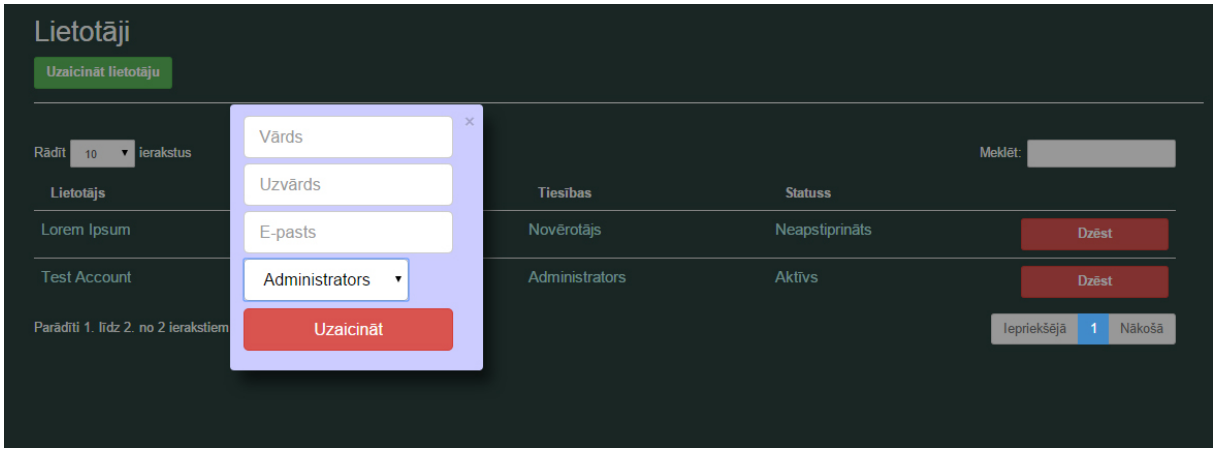
Sistēmas augšējās izvēlnes kreisajā pusē redzama kontaktpogas ikona, kas ved uz */admin/users* maršrutu. Uz to atbild *AdminController.php getUsers* metode, kas atgriež sistēmas lietotāju tabulāro skatu, kur redzama katra lietotāja informācija, kā arī dzēšanas iespēja, kas gan iespējota tikai galvenajam administratoram. Virs tabulas atrodas "Uzaicināt lietotāju" saite, kas atver uznirstošo logu jauna lietotāja uzaicināšanai.

2.3.5.4 Lietotāja uzaicināšana

Uznirstošā loga uzaicinājuma formā tiek ievadīts uzaicināmā lietotāja vārds, uzvārds, e-pasts un izvēlēts tiesību līmenis. Apstrādi veic *AdminController.php postInvite* metode, kas datubāzē izveido jaunu lietotāja entītiju, norāda tai nereģistrēta lietotāja statusu un uzģenerē nejaušu 10 simbolu virkni – reģistrācijas žetonu –, kas tiek iekļauts reģistrācijas saitē formā */register/[token]*, kur "token" ir reģistrācijas žetons. Šī saite tiek izsūtīta uz norādīto e-pastu.

2.3.5.5 Lietotāja dzēšana

Lietotāju skata "Dzēst"saite izsauc maršrutu */admin/delete-user/[id]*, kur "id" ir dzēšamā lietotāja identifikācijas numurs. To ar Post izsaukumu pieņem *AdminController.php postDeleteUser* metode, kas dzēš lietotāju no datubāzes un pāradresē atpakaļ uz lietotāju skatu.



Attēls 13. Lietotāju skats ar uzaicinājuma formu

2.3.5.6 Lietotāja reģistrācija

Dodoties uz uzaicinājuma e-pastā doto uzaicinājuma saiti (*/register/[token]*), pieprasījumu apstrādā *HomeController.php getRegister* metode, kas pieņem reģistrācijas žetonu un pārbauda, vai datubāzē ir nereģistrēts lietotājs ar šādu reģistrācijas žetonu. Ja žetons ir nepareizs, tiek atgriezta HTTP 404 kļūda. Pretējā gadījumā lietotājam atgriež reģistrācijas formu. Iesniedzot reģistrācijas datus, tos apstrādā *HomeController.php postRegister* metode, kas pārbauda lauku esamību un pareizību, sajauc paroli ar Laravel nodrošināto *Bcrypt* jaucējfunkciju, pēc tam saglabā lietotāja datus datubāzē, atzīmē lietotāju kā reģistrētu un pāradresē uz lietotāja sākumskatu (projektu skatu).

3. Testēšanas dokumentācija

Sistēmas svarīgākajai moduļu funkcionalitātei pēc izstrādātāja ieskatiem izveidotas atsevišķas vienībtestēšanas klases *app/tests/* mapē. Pārējie testi veikti manuāli pārbaudot sistēmas elementu funkcionalitāte un vērtējot sistēmas darbību dažādos ievades un lietojuma gadījumos, veicot gan mērķtiecīgus manuālos testus, gan lietotāju testus, kuru laikā potenciālie lietotāji iepazinušies ar sistēmas darbību un veikuši funkcionalitātes testēšanu.

Sistēmas lietotāja saskarne, balstoties uz lietotāju atsauksmēm, atzīta par atbilstošu.

3.1 Projektu izveide un rediģēšana, kategoriju un klientu pievienošana

Apraksts	**Paredzētais rezultāts**	**Rezultāts**
Iesniedz projektu, nenorādot tā kategoriju.	Kļūdas paziņojums, lūdz izvēlēties kategoriju, neļauj iesniegt projektu.	Iziets
Iesniedz projektu, nenorādot klientu.	Kļūdas paziņojums, lūdz izvēlēties klientu, neļauj iesniegt projektu.	Iziets
Iesniedz projektu, neizvēloties nevienu projektu prasību.	Kļūdas paziņojums, lūdz izvēlēties vismaz vienu projektu prasību, neļauj iesniegt projektu.	Iziets
Iesniedz projektu, nenorādot projekta nosaukumu.	Kļūdas paziņojums, lūdz ievadīt projekta nosaukumu, neļauj iesniegt projektu.	Iziets
Pievienojot kategoriju, nenorāda nosaukumu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Pievienojot klientu, nenorāda klienta nosaukumu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Pievienojot klientu, nenorāda kontaktpersonu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Pievienojot klientu, nenorāda telefona numuru.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Neiziets, ļauj ievadīt tukšu. Labots.
Veic izmaiņas projektā.	Atbilstoši pārģenerēti tāmes dokumenti.	Iziets

Apraksts	Paredzētais rezultāts	Rezultāts
Izsauc projekta arhivāciju no projektu kopskata.	Projekts izņemts no projektu kopskata, parādās projektu arhīvā.	Iziets
Izsauc projekta atjaunošanu no projektu arhīva skata.	Projekts izņemts no projektu arhīva, parādās projektu kopskatā.	Iziets

3.2 Projektu prasību pievienošana, rediģēšana un dzēšana, darba uzdevumu operācijas

Apraksts	Paredzētais rezultāts	Rezultāts
Iesniedz projektu prasību, nenorādot tās nosaukumu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Neiziets, ļauj ievadīt tukšu. Labots.
Iesniedz projektu prasību bez neviena pievienota darba uzdevuma.	Kļūdas paziņojums, lūdz pievienot vismaz vienu darba uzdevumu, neļauj iesniegt formu.	Iziets
Pievieno darba uzdevumu, nenorādot tā nosaukumu.	Neļauj pievienot, norāda neaizpildīto lauku.	Iziets
Pievieno darba uzdevumu, nenorādot tam paredzēto stundu skaitu.	Neļauj pievienot, norāda neaizpildīto lauku.	Iziets
Pievieno darba uzdevumu bez izvēlētas speciālistu pozīcijas.	Neļauj pievienot, norāda neaizpildīto lauku.	Iziets
Rediģē darba uzdevumu, atstāj neaizpildītu paredzēto stundu skaitu.	Atstāj iepriekšējo vērtību.	Iziets
Dzēš visus darba uzdevumus kādai projektu prasībai.	Neļauj dzēst pēdējo, kļūdas paziņojums.	Neiziets, ļauj izdzēst visus. Labots.
Rediģē projektu prasību, atstāj neaizpildītu nosaukuma lauku.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Neiziets, ļauj ievadīt tukšu. Labots.
Rediģē kādā projektā izmantotu projektu prasību.	Projektu prasības izmaiņas tiek saglabātas, tiek izveidota rezerves kopija un iesaistīto projektu tāmes savas īpašības nemaina.	Iziets
Dzēš kādā projektā izmantotu projektu prasību.	Projektu prasības un tās apakšprasību statuss atzīmēts kā rezerves kopija, iesaistīto	Iziets

Apraksts	Paredzētais rezultāts	Rezultāts
	projektu tāmes savas īpašības nemaina.	
Pievieno prasību caur "Pievienot virsprasību" atgriezto formu.	Tiek izveidota augstākā līmeņa projektu prasība. Atspoguļojas projektu prasību kopskatā.	Iziets
Pievieno prasību caur "Pievienot apakšprasību" atgriezto formu.	Tiek izveidota zemāka līmeņa projektu prasība. Atspoguļojas projektu prasību kopskatā, kur atrodas zem citas prasības, kam tika izsaukta apakšprasības pievienošana.	Iziets

3.3 Klientu rediģēšana un dzēšana

Apraksts	Paredzētais rezultāts	Rezultāts
Veic klienta rediģēšanu, atstāj neaizpildītu klienta, nosaukumu	Neļauj iesniegt formu un saglabāt izmaiņas, norāda neaizpildīto lauku.	Iziets
Veic klienta rediģēšanu, atstāj neaizpildītu klienta kontaktpersonu,	Neļauj iesniegt formu un saglabāt izmaiņas, norāda neaizpildīto lauku.	Iziets
Veic klienta rediģēšanu, atstāj neaizpildītu klienta telefona numuru.	Neļauj iesniegt formu un saglabāt izmaiņas, norāda neaizpildīto lauku.	Iziets
Dzēš klientu.	Izdzēš gan klientu, gan tā projektus.	Iziets

3.4 Tāmes dokumentu lejupielāde un dalīšanās

Apraksts	Paredzētais rezultāts	Rezultāts
Pieprasa tāmes Excel izklājlapas lejupielādi.	Pārlūks sāk vajadzīgā faila lejupielādi.	Iziets
Pieprasa tāmes apraksta Word dokumenta lejupielādi.	Pārlūks sāk vajadzīgā faila lejupielādi.	Iziets

Apraksts	Paredzētais rezultāts	Rezultāts
Pieprasa tāmes apraksta PDF dokumenta lejupielādi.	Pārlūks sāk vajadzīgā faila lejupielādi.	Iziets
Nepierakstījies sistēmā, izmanto derīgu tāmes īssaiti.	Pielaiž pie tāmes un dokumentu lejupielādes viesa skatā, bez sānu izvēlnēm un pieejas pārējai sistēmai.	Iziets
Iepriekš pierakstījies sistēmā, izmanto derīgu tāmes īssaiti.	Pielaiž pie tāmes un dokumentu lejupielādes parastā lietotāja skatā.	Neiziets, atgriež viesa skatu. Labots
Dodas uz tāmes īssaiti ar kļūdainu viesa žetonu tās sastāvā.	HTTP 404 kļūda.	Iziets

3.5 Lietotāju funkcijas

Apraksts	Paredzētais rezultāts	Rezultāts
Uzaicina lietotāju, nenorāda lietotāja vārdu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Uzaicina lietotāju, nenorāda lietotāja uzvārdu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Uzaicina lietotāju, nenorāda lietotāja epastu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Uzaicina lietotāju, izmantojot derīgu epasta adresi.	Lietotājs saņem uzaicinājuma epastu ar reģistrācijas saiti.	Iziets
Dodas uz saņemtu reģistrācijas saiti.	Sistēma atgriež reģistrācijas formu.	Iziets
Dodas uz reģistrācijas saiti ar kļūdainu reģistrācijas žetonu tās sastāvā.	HTTP 404 kļūda.	Iziets
Veic lietotāja reģistrāciju, nenorāda lietotājvārdu.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Veic lietotāja reģistrāciju, nenorāda paroli.	Neļauj iesniegt formu, norāda neaizpildīto lauku.	Iziets
Veic lietotāja reģistrāciju, nenorāda atkārtoto paroli vai arī tā nesakrīt.	Neļauj iesniegt formu, norāda paroļu nesakritību.	Iziets

4. Projekta organizācija

Sistēmas izstrāde tika veikta patstāvīgi, paralēli izstrādei konsultējoties ar pasūtītājiem par prasību izmaiņām un uzstādīto prasību realizācijas formu un lietojamību. Projekts tika izstrādāts ilgākā laika periodā, nepilna laika režīmā. Izstrādes ciklu var iedalīt vairākos posmos.

- Pirmajā posmā tika veikts sistēmas sākotnējais uzmetums. Noritēja pārrunas ar sistēmas pasūtītājiem, kas izklāstīja un izskaidroja pamatprasības, kā arī iepazīstināja ar primitīvu sistēmas prototipu, kas kalpoja vēlamā projektu izveides mehānisma demonstrācijai. Balstoties uz šajā posmā gūto informāciju, tika noformulētas un saskaņotas pamatprasības, konsultējoties ar citiem programmētājiem un veicot izpēti tīmekļa resursos, tika pieņemti lēmumi par tehnisko rīku izvēli.
- Otrajā posmā tika izveidots uz esošajām prasībām balstīts sistēmas pamata un to apkalpojošās datubāzes projektējums, sākta izstrāde un sākotnējo moduļu testēšana, periodiski konsultējoties ar pasūtītājiem par izstrādāto moduļu lietderību un lietojamību.
- Trešajā posmā, kad sistēmas galvenā funkcionalitāte bija saskaņota, izveidota un pamatlīmenī notestēta, sākās paildzināta rakstura izstrāde, periodiski konsultējoties ar pasūtītājiem, kuri uzstādīja papildus prasības funkcionalitātes paplašināšanai, kas tika konkretizētas, saskaņotas un uz to pamata paplašināts sistēmas projektējums, veikta papildus moduļu izstrāde.
- Ceturtajā posmā tika saskaņots sistēmas pirmās versijas prasību kopsavilkums un pārspriests projektējums, veikta plašāka lietotāja puses sistēmas testēšana, tika identificēti un veikti nepieciešamie labojumi un mazāka mēroga izmaiņas sistēmā.

Sistēmas pašreizējā versija nav projekta gala versija, tā vairāk ir funkcionalitātes un lietojamības testa versija, tās izstrādē nākotnē tiks turpināta, paplašinot funkcionalitāti, kuras atsevišķas iestrādnes ir paredzētas jau šajā versijā.

5. Kvalitātes nodrošināšana

Izstrādātās sistēmas kvalitātes nodrošināšanā loma bija gan izvēlētajiem tehniskajiem risinājumiem, gan sistēmas testēšanai un kopīgai lietojamības analīzei. Sistēmas izstrādei tika izvēlēts Laravel 4 izstrādes ietvars, kas balstās uz MVC projektēšanas šablona, palīdzot sistēmu veidot modulārāku un ilgtspējīgāku. Tika veikta citu sistēmu un ietvara dokumentācijas izpēte, cenšoties sekot tā izstrādes principu ieteiktajai praksei, pareizi strukturējot kodu un ievērojot modeļu, skatu un kontrolieru loģikas dalījumu. Izstrādes rezultāti tika testēti manuālajos un vienībtestos, attiecīgi melnkastes un baltkastes testos, kuri plašāk aprakstīti testēšanas dokumentācijas nodaļā. Manuālajā testēšanā un lietojamības vērtēšanā tika piesaistīti arī pasūtītāji un potenciālie sistēmas lietotāji, ar kuriem tika apspriesti galvenokārt lietojamības ērtuma un sistēmas funkciju jautājumi.

Sistēmas programmatūras prasību specifikācijas un programmatūras projektējuma apraksta izstrādē tika sekots valsts standartu LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis” un LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai" sniegtajām vadlīnijām, palīdzot strukturēt dokumentus un atvieglojot saskaņošanas procesu, kā arī cenšoties sekot gūtajām zināšanām par programminženierijas labo praksi [7].

Sistēmas koda izstrādes jautājumos par atsevišķiem tehniskiem lēmumiem un moduļu realizāciju tika veiktas konsultācijas arī ar pieredzējušākiem programmētājiem, pamatā gan uzsvaru liekot uz personīgu tēmas, ietvara standartu un prakses izpēti.

6. Konfigurācijas pārvaldība

Sistēmas versiju pārvaldībai tika izvēlēts versiju kontroles rīks *git*, kas ļāva regulāri saglabāt izmaiņas, nodrošināt datus un uzturēt informāciju par stabilām sistēmas versijām. Nodevumi tika veikti regulāri, gan moduļu izstrādes laikā, norādot komentārus par attiecīgās versijas nestabilitāti, gan pabeidzot moduļus. Kā *git* rīka izmantotais repozitorijs sākotnēji tika lietots Latvijas Universitātes nodrošinātais *git.df.lu.lv*, taču izstrādes beigu fāzē nodevumu vēsture tika migrēta uz *BitBucket*, kur pieejama pilna vēsture un pirmkods.

7. Darbietilpības novērtējums

Sākotnējais darbietilpības novērtējums veikts ar COCOMO II modeli [8][9], izmantojot funkcijpunktu skaitīšanu. Tika pieņemts, ka katram funkcijpunktam atbilst 30 programmkoda rindas, kas ir mūsdienīgām objektorientētām valodām piemērojamā konstante.

	Vienkārši	**Sarežģīti**	**Kopā**
Ievades formas	8 (3FP)	2 (6FP)	36
Failu izvade	2 (4FP)	1 (8FP)	16
Datubāzes vaicājumi	22 (3FP)	2 (6FP)	78
Failu ievade	1 (7FP)	0 (14FP)	7
Saskarne	6 (5FP)	3 (10FP)	60

Funkcijpunktu kopsumma: 197FP

Koda rindu aprēķins: 197 * 30 = 5910 LOC

Darbietilpības aprēķins: 3.6 * 5.91(KLOC) * 1.2 = 25.53 personmēneši

Aprēķinātais apjoms jau sākotnēji bija atzīstams par projekta apjomiem neadekvātu, nepiemērotu viena cilvēka izstrādātai nelielai sistēmai.

Konsultējoties ar pieredzējušākiem programmētājiem un ņemot vērā minimālo pieredzi šādu sistēmu izstrādē un tehnoloģiju izmantošanā, aptuvenā darbietilpības prognoze darba izstrādei, neskaitot dokumentēšanu un testēšanu, tika novērtēts kā 2.5 personmēneši.

Reālais izstrādei patērētais laiks bija 3 personmēneši, kuru laikā tika izstrādātas ap 1800 rindām PHP koda, vairāk kā 800 rindiņas Javascript koda, aptuveni 700 rindiņas HTML (ar *Blade* veidnēm) koda un arī ap 700 rindiņām CSS koda, kopā ap četriem tūkstošiem dažādu valodu koda, neskaitot komentārus un tukšās rindiņas. PHP koda apjomu ievērojami ļāva samazināt ietvara izmantošana. Papildus tika patērēti arī 0.5 personmēneši sistēmas testēšanai un dokumentēšanai.

8. Programmatūras pirmkoda paraugi

8.1 *Estimate* – tāmes modeļa klase

Fails: *app/models/Estimate.php*

N.B. Sistēmas modeļi manto Laravel ietvara *Eloquent* ORM klasi, nodrošinot ietvarā iebūvēto abstrakcijas līmeni modeļu interakcijām ar datubāzi.

```php
<?php

/*
 * Estimate model
 * by Arturs Kurzemnieks
 *
 * Document generation implemented inside the model itself, since it uses its own data.
 */
class Estimate extends Eloquent {
    protected $table = 'estimates';

    public function project() {
        // defines relation with its parent project
        return $this->belongsTo('Project','project_id');
    }

    public function work_functions() {
        // defines relations with the project requirements it uses
        return $this->belongsToMany('WorkFunction','func-to-est','est_id','func_id');
    }

    public function extraPositions() {
        /*
        *    checks for any extra specialist positions (other than the company's standard ones) used
        *   returns the number of them and their corresponding entity's ID
        */
        $positions = array();
        $count = 0;
        foreach($this->work_functions()->get() as $func) {
            foreach($func->tasks()->get() as $task) {
                /*
                 * iterates through all the tasks of all the work requirements used by the estimate
                 * and checks their position usage
                 */
                if($task->position_id > 11 && !in_array($task->position_id, $positions)) {
                    array_push($positions, $task->position_id);
                    $count++;
                }
            }
        }
        return ['positions' => $positions, 'count' => $count];
    }

    public function designCount() {
        // counts the design view total that's estimated for the project
        $total = 0;
        $functions = $this->work_functions()->get();
        foreach($functions as $func) {
            $total += $func->designs;
```

```
        }
        return $total;
    }

    public function functionCount() {
        // counts the number of project requirements used by the estimate
        $total = 0;
        foreach($this->work_functions()->get() as $func) {
            $total++;
        }
        return $total;
    }

    public function generateExcel() {
        /*
         * calculates required data and generates
         * an Excel spreadsheet for the estimate
         */
        $estimate = $this;
        $pos_info = $estimate->extraPositions();
        $functions = $estimate->work_functions()->get();
        // prepares row and column positions based on project requirements and
specialist count
        $sum_col = 12 + $pos_info['count'];
        $rate_row = 25 + $estimate->functionCount();
        // retrieves the estimate template
        $reader = new PHPExcel_Reader_Excel2007();
        $excel = $reader->load('../estimates/template.xlsx');
        $excel->getProperties()->setCreator('WRONG')
                               ->setTitle('Taame - ' . $estimate->project->name);
        $excel->setActiveSheetIndex(0);
        $sheet = $excel->getActiveSheet();
        $pluscol = 12;
        // adds extra specialist positions to the template if there are any
        if($pos_info['count'] > 0) {
        $sheet->insertNewColumnBefore('M',$pos_info['count']);
            foreach($pos_info['positions'] as $pos) {
                $currentcol = PHPExcel_Cell::stringFromColumnIndex($pluscol);
                $sheet->duplicateStyle($sheet->getStyle('L9'),$currentcol.'9')
                        ->duplicateStyle($sheet->getStyle('L10'),$currentcol.'10')
                        ->duplicateStyle($sheet->getStyle('L17'),$currentcol.'17')
                        ->duplicateStyle($sheet->getStyle('L18'),$currentcol.'18')
                        ->setCellValue($currentcol.'10',Position::find($pos)->name)
                        -
>setCellValue($currentcol.'24','=SUM('.$currentcol.'11:'.$currentcol.'23)')
                        ->setCellValue($currentcol.'25',Position::find($pos)->rate)
                        -
>setCellValue($currentcol.'26','='.$currentcol.'25*'.$currentcol.'24');

                $pluscol++;
            }
        }
        // Inserts design view total into the template
        $sheet->setCellValue('A15','Dizaina izstrāde (' . $estimate->designCount()
. ' skati)')
              ->insertNewRowBefore(18,$estimate->functionCount());
        // main loops for shifting cells and inserting data
        $row = 18;
        foreach($functions as $func) {
            //inserts project requirement data
            $sheet->duplicateStyle($sheet-
>getStyle('A11'),'A'.$row.':'.PHPExcel_Cell::stringFromColumnIndex($pluscol       -
1).$row)
                  ->setCellValue('A'.$row,$func->name)
                  ->getRowDimension($row)->setRowHeight($sheet-
>getRowDimension(11)->getRowHeight());
            for($col = 1; $col <= 11 + $pos_info['count']; $col++) {
                $sheet->setCellValueByColumnAndRow($col,$row,
```

```
                                                    ($func->positionHours()[$col]  ==
0) ? null : $func->positionHours()[$col]);
                }
                // calculates total, uses custom made helper class, found @
app/classes/Excel.php
                $sheet-
>setCellValue(PHPExcel_Cell::stringFromColumnIndex($sum_col).$row,Excel::rowSum($ra
te_row,$sum_col,$row));
                $row++;
            }
            for($row = 11; $row <= 23 + $estimate->functionCount(); $row++) {
                if($row == 17) { $row = $row + $estimate->functionCount() + 2; }
                // calculates each requirement's subtotal, uses custom made helper
class, found @ app/classes/Excel.php
                $sheet-
>setCellValue(PHPExcel_Cell::stringFromColumnIndex($sum_col).$row,Excel::rowSum($ra
te_row,$sum_col,$row));
            }
            if($pos_info['count'] > 0) {
                // shifts cells to accommodate space for extra specialist positions
                Excel::moveBlock($sheet,'K'.strval(27           +           $estimate-
>functionCount()).':L'.strval(39 + $estimate->functionCount()),
                            $pos_info['count']);
                for($row  =  31  +  $estimate->functionCount();  $row  <=  33  +  $estimate-
>functionCount(); $row++) {
                    $sheet-
>setCellValue(PHPExcel_Cell::stringFromColumnIndex($sum_col).$row,

'='.PHPExcel_Cell::stringFromColumnIndex(PHPExcel_Cell::columnIndexFromString('L')
+                                $pos_info['count']                                -
1).$row.'*'.PHPExcel_Cell::stringFromColumnIndex(PHPExcel_Cell::columnIndexFromStri
ng('K') + $pos_info['count'] - 1).$row);
                }
            }
            $sheet->setCellValue('B8',$estimate->project->name)
                ->duplicateStyle($sheet->getStyle('A8'),'B8');
            // creates an excel writer instance and saves the document identified by
the project number
            $writer = new PHPExcel_Writer_Excel2007($excel);
            $filename = 'Taame_'.$estimate->project->nr.'.xlsx';
            $writer->save('../estimates/'.$filename);
            $excel->disconnectWorksheets();
            unset($excel);
        }
        public function generateDocs() {
            /*
             * generates description documents for the estimate
             */
            $estimate = $this;
            // creates a phpword instance, sets headers and main styles
            $document = new \PhpOffice\PhpWord\PhpWord();
            header('Content-Type: text/html; charset=utf-8');
            $document->setDefaultFontName('DejaVu');
            $document->addFontStyle('main',array('name' => 'DejaVu', 'size' => 10));
            $document->addParagraphStyle('main-p',array('align' => 'left', 'spaceAfter'
=> 100));
            $document->addTitleStyle('head',array('name'  =>  'DejaVu',  'size'  =>  16,
'bold' => true),array('spaceAfter' => 50));
            $document->addTitleStyle('function',array('name' => 'DejaVu', 'size' => 14,
'bold' => true),array('spaceAfter' => 20));

            $properties = $document->getDocumentProperties();
            $properties->setCreator('WRONG');
            $properties->setCompany('WRONG');
            $properties->setTitle('Description'.$estimate->project->name);
            $section = $document->addSection();
            \PhpOffice\PhpWord\Shared\Html::addHtml($section,        '<meta        http-
equiv="Content-Type" content="text/html; charset=utf-8"/>');
```

```
        $section->getSettings()->setMarginLeft(500);
        $section->getSettings()->setMarginRight(500);
        // writes content
        $section->addTitle($estimate->project->name,'head');
        $section->addText('','main',array('spaceAfter' => 100));
        $relations = EstimateFunction::where('est_id', '=', $estimate->id)->get();
        foreach($relations as $rel)
        {
            $function = WorkFunction::find($rel->func_id);
            $section->addTitle($function->name,'function');
            $section->addText($function->description,'main','main-p');
        }
        $section->addText('','main',array('spaceAfter' => 200));
        $section->addPageBreak();
        $section->addTitle('Norādes dizaineriem','function');
        foreach($relations as $rel)
        {
            $function = WorkFunction::find($rel->func_id);
            $section->addListItem($function->design_guide,0,'main',3);
        }
        $section->addText('','main',array('spaceAfter' => 100));
        $section->addTitle('Norādes tekstiem','function');
        foreach($relations as $rel)
        {
            $function = WorkFunction::find($rel->func_id);
            $section->addListItem($function->text_guide,0,'main',3);
        }
        $filename = 'Apraksts_'.$estimate->project->nr;
        // creates new phpword writes for word document format
        $objWriter      =      \PhpOffice\PhpWord\IOFactory::createWriter($document,
'Word2007');
        $objWriter->save('../estimates/'.$filename.'.docx');
        // sets the pdf writer engine to DomPDF and remakes the phpword writer for
pdf writing
        \PhpOffice\PhpWord\Settings::setPdfRendererPath('../vendor/dompdf/dompdf');
        \PhpOffice\PhpWord\Settings::setPdfRendererName('DomPDF');
        $objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($document, 'PDF');
        $objWriter->save('../estimates/'.$filename.'.pdf');
    }
}
```

8.2 *ClientController* – klienta kontroliera klase

Fails: *app/controllers/ClientController.php*

```
<?php
/*
 * Client controller
 * by Arturs Kurzemnieks
 */
class ClientController extends BaseController {
    // standard user authentification check for all controller methods
    public function __construct() {
        $this->beforeFilter('auth');
    }

    public function getIndex() {
        // responds to /clients index call, returns the clients view with client
data
        $clients = Client::all();
        return View::make('clients',array('clients' => $clients));
    }

    public function postCheckName() {
        /*
         *  responds to /clients/check-name AJAX post call
         *  Checks if the client name provided is not in use already
         */
        if(Request::ajax()) {
            $tempname = Input::get('client_name');
            if (Client::whereName($tempname)->count() > 0) {
                $status = 0;
            } else {
                $status = 1;
            }
            return Response::json(array('available' => $status));
        }
    }

    public function postInfo() {
        /*
         *  responds to /clients/info AJAX post call
         *  returns client data based on client's name
         */
        if(Request::ajax()) {
            $client = Client::whereName(Input::get('client'))->first();
            $response = array('contact' => $client->contact,
                'phone' => $client->phone,
                'email' => $client->email);
            return Response::json($response);
        }
    }
    public function postInfoId() {
        /*
         * responds to /clients/info-id AJAX post call
         * returns client data based on client's ID
         */
        if(Request::ajax()) {
            $client = Client::findOrFail(Input::get('client_id'));
            $response = array('contact' => $client->contact,
                'phone' => $client->phone,
                'email' => $client->email,
                'name' => $client->name);
            return Response::json($response);
        }
        else {
            App::abort(404);
```

```
        }
    }

    public function postNew() {
        /*
         * responds to /clients/new AJAX post call
         * creates a new client entity
         * returns the client's name
         */
        if(Request::ajax()) {
            $data = Input::all();
            $rules = array('name' => 'required', 'email' => 'email', 'phone' =>
'required|numeric');
            $validator = Validator::make($data,$rules);
            if($validator->fails()) {
                return  Response::json(array('errors'  =>  $validator->messages()-
>all()));
            }
            else {
                $newclient = new Client();
                $newclient->name = $data['name'];
                $newclient->contact = $data['contact'];
                $newclient->phone = $data['phone'];
                $newclient->email = $data['email'];
                $newclient->save();
                return Response::json(array('name' => $newclient->name));
            }
        }
        else {
            App::abort(404);
        }
    }

    public function postEdit() {
        /*
         * responds to /clients/edit AJAX post call
         * checks data and edits the client's entity
         * returns full client data for view updating
         */
        if(Request::ajax()) {
        $data = Input::all();
        $rules = array('name' => 'required', 'email' => 'email', 'phone' =>
'required|numeric');
        $validator = Validator::make($data,$rules);
            if($validator->fails()) {
                return  Response::json(array('errors'  =>  $validator->messages()-
>all()));
            }
            else {
                $client = Client::findOrFail($data['client_id']);
                $client->name = $data['name'];
                $client->contact = $data['contact'];
                $client->phone = $data['phone'];
                $client->email = $data['email'];
                $client->save();
                return Response::json(array(
                    'name' => $client->name,
                    'contact' => $client->contact,
                    'email' => $client->email,
                    'phone' => $client->phone,
                    'id' => $client->id));
            }
        }
        else {
            App::abort(404);
        }
    }
```

```
    public function postDelete() {
        /*
         * responds to /clients/delete POST call
         * deletes all the client's projects (utilizes Project model safe delete)
         * then the client itself
         */
        $client_id = Input::get('delete_id');
        $client = Client::findOrFail($client_id);
        $client->projects()->get()->each(function($project) {
            $project->deleteProject();
        });
        $client->delete();
        return Redirect::to('clients');
    }

}
```

8.3 Projekta izveides skats (izmanto *Blade* veidņu valodu)

Fails: *app/views/create.blade.php*

```
@extends('layouts.main')
@section('section-main')
<div class='section-head'><h2>{{ isset($edit) ? 'Rediģēt projektu' : 'Jauns projekts'
}}</h2></div>
<div class='form-group project-form'>
    {{ Form::open(array('url' => isset($edit) ? '/projects/edit' : 'projects/new', 'id' =>
'projectform')) }}
    <div class='form-group'>

    <div class='dropdown'>
        <button class='btn btn-info form-control input-lg dropdown-toggle' type='button'
id='categoryMenu' data-toggle='dropdown'>
            <span id='category-name'>{{ isset($edit) ? $project->categories->name :
'Kategorija' }}</span>
            <span class='caret'></span>
        </button>
        <ul class='dropdown-menu' id='category-select' role='menu'
arialabelledby='dropdownMenu2'>
        @foreach($categories as $cat)
            <li role='presentation' class='category-option'><a role='menuitem' tabindex='-1'
href='#'>{{ $cat->name }}</a></li>
        @endforeach
            <li role="presentation"><a role="menuitem" tabindex="-1" href="#" id='add-
category' class='btn btn-default'>
                <div class='btn plus-button'>+</div>Cita...
            </a></li>
        </ul>
        {{ Form::hidden('category', isset($edit) ? $project->categories->name : null ) }}
    </div>
    <div class="dropdown">
  <button class="btn btn-warning form-control input-lg dropdown-toggle" type="button"
id="clientMenu" data-toggle="dropdown">
      <span id="client-name">{{ isset($edit) ? $project->clients->name : 'Klients' }}</span>
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" id="client-select" role="menu" aria-labelledby="dropdownMenu1">
        <li role="presentation"><a role="menuitem" tabindex="-1" href="#" id='add-client'
class='btn btn-default'>
                <div class='btn plus-button'>+</div>Jauns klients
            </a></li>
      @foreach ($clients as $client)
        <li role='presentation' class="client-option"><a role='menuitem' tabindex='-1'
href='#' >{{$client->name}}</a></li>
      @endforeach
  </ul>
        {{ Form::hidden('client', isset($edit) ? $project->clients->name : null) }}
    </div>
    </div>
```

```
    <div class='form-group'>
    {{ Form::text('nr', isset($edit) ? $project->nr : $number,
                    array('class' => 'form-control input-lg',
                          'id' => 'project-nr',
                          'placeholder' => 'Numurs' )) }}
    {{ Form::text('name', isset($edit) ? $project->name : null,
            array('class' => 'form-control input-lg',
                  'id' => 'project-name',
                  'placeholder' => 'Projekta nosaukums')) }}
    </div>
    <div class='form-errors'></div>
<div class='form-group function-boxes'>
    @foreach ($functions as $func)
    <div class='function-box func-level-{{$func->level}}' data-level={{ $func->level }}>
        <div class='checkbox'>
            {{ Form::checkbox('funcs[]',$func->id, isset($edit) ? in_array($func->id,
                        $checked->lists('id')) ? true : false : false,
                        array('class' => 'funcbox', 'id' => $func->id . '-box')) }}
            <label for='{{ $func->id }}-box' class='funcbox-label'>{{ $func->name }}</label>
            <svg                version="1.1"               xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
    width="32px"  height="32px"  viewBox="0  0  512  512"  enable-background="new  0  0  512  512"
xml:space="preserve" class="function-details">
<path    id="pen-8-icon"    class="svg-white"    d="M334.502,147.189h23.4v105.122h-23.4V147.189z
M244.592,147.189h23.4v45.777h-23.4V147.189z
    M154.098,147.189h23.4v105.122h-23.4V147.189z       M200.604,310.227h-23.105v54.584h-23.4v-
54.584h-23.109v-40.365h69.615V310.227z
    M291.1,250.881h-23.107v113.93h-23.4v-113.93h-23.107v-40.364H291.1V250.881z
M381.01,310.341h-23.107v54.47h-23.4v-54.47h-23.107
   v-40.365h69.615V310.341z           M382,90c22.092,0,40,17.908,40,40v252c0,22.092-17.908,40-
40,40H130c-22.092,0-40-17.908-40-40V130
   c0-22.092,17.908-40,40-40H382z    M462,125c0-41.422-33.578-75-75-75H125c-41.422,0-75,33.578-
75,75v262c0,41.422,33.578,75,75,75h262
   c41.422,0,75-33.578,75-75V125z"/>
</svg>
        </div>
    </div>
    @endforeach
</div>
<footer>
    @if(isset($edit))
        {{ Form::hidden('edit_project', $project->id) }}
    @endif
    {{ Form::submit(isset($edit) ? 'Labot' : 'Izveidot',array('class' => 'btn btn-danger btn-
lg')) }}
    {{ Form::close() }}
</footer>
    </div>
<div id='new-client' class='popup-box popup-standard'>
    <a class='close'>&times;</a>
    {{ Form::open(array('url' => 'clients/new', 'id' => 'clientform')) }}
    {{ Form::text('name',null,array('placeholder' => 'Klienta nosaukums',
                                    'id' => 'new-client-name',
                                    'class' => 'form-control input-lg')) }}
    {{ Form::text('contact',null,array('placeholder' => 'Kontaktpersona',
                                    'class' => 'form-control input-lg',
                                    'id' => 'new-client-contact')) }}
    {{ Form::text('phone',null,array('placeholder' => 'Telefona numurs',
                                    'class' => 'form-control input-lg')) }}
    {{ Form::email('email',null,array('placeholder' => 'E-pasts',
                                      'class' => 'form-control input-lg')) }}
    {{ Form::submit('Pievienot',array('class' => 'btn btn-danger btn-lg')) }}
    {{ Form::close() }}
</div>
<div id='new-category' class='popup-box popup-standard'>
    <a class='close'>&times;</a>
    {{ Form::open(array('url' => 'categories/new', 'id' => 'categoryform')) }}
    {{ Form::text('name',null,array('placeholder' => 'Kategorijas nosaukums',
                                    'id' => 'new-category-name',
                                    'class' => 'form-control input-lg')) }}
    {{ Form::submit('Pievienot',array('class' => 'btn btn-danger btn-lg')) }}
    {{ Form::close() }}
</div>
<div id='client-info'>
    <span class='lead' id='client-info-name'></span><br>
    <div      class='label      label-default'>Kontaktpersona:</div><span      id='client-info-
contact'></span><br>
```

```
    <div class='label label-default'>Telefons:</div><span id='client-info-phone'></span><br>
    <div class='label label-default'>E-pasts:</div><span id='client-info-email'></span>
</div>
<div id='function-fork' class='popup-box popup-standard'>
    <a class='close'>&times;</a>
    @include('functionform-content')
    <div id='fork-projects' class='popup-box popup-standard'>
        <span class='lead'>Konfigurācija projektos:</span>
    </div>
</div>
{{ HTML::script('assets/js/projects-create.js') }}
@stop
```

9. Izmantotā literatūra

1. ”Programmatūras prasību specifikācijas ceļvedis”. LVS 68:1996 standarts.
2. http://www.symantec.com/connect/articles/five-common-web-application-vulnerabilities
3. “Ieteicamā prakse programmatūras projektējuma aprakstīšanai”. LVS 72:1996 standarts.
4. http://laravel.com/docs/4.2/
5. https://github.com/PHPOffice/PHPExcel
6. https://github.com/PHPOffice/PHPWord
7. Pressman, R.S. Software engineering A Practitioner’s Approach. - New York : McGraw-Hill, Publishing Company, 2005
8. http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html
9. http://it.toolbox.com/blogs/just-make-it-work/making-estimates-of-software-development-effort-27551

Kvalifikācijas darbs „***Projektu pārvaldības sistēma 'Specomatic'*** ” izstrādāts Latvijas Universitātes Datorikas fakultātē.

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

Autors: **Artūrs Kurzemnieks _____ ___.05.2015**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: **M. dat. Artūrs Lavrenovs _____ ___.05.2015.**

Recenzents: **M.dat. Jānis Mārtužs**

Darbs iesniegts 01.06.2015.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: Imants Gorbāns _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): _____