

KAUNO TECHNOLOGIJOS UNIVERSITETAS  
INFORMATIKOS FAKULTETAS

MANGIRDAS KAZLAUSKAS

**AMQP PROTOKOLO PRANEŠIMŲ EILIŲ VALDYMO SAITYNO  
ĮRANKIS**

Baigiamasis bakalauro projektas

Vadovas  
doc. dr. T. Blažauskas

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS  
INFORMATIKOS FAKULTETAS

**AMQP PROTOKOLO PRANEŠIMŲ EILIŲ VALDYMO SAITYNO  
ĮRANKIS**

Baigiamasis bakalauro projektas  
Programų sistemos (kodas 612I30002)

Vadovas  
doc. dr. T. Blažauskas

---

(data, parašas)

Recenzentas  
lekt. T. Uktveris

---

(data, parašas)

Projektą atliko  
M. Kazlauskas

---

(data, parašas)



## KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

Mangirdas Kazlauskas

(Studento vardas, pavardė)

Programų sistemos (kodas 612I30002)

„AMQP protokolo pranešimų eilių valdymo saityno įrankis“

### AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 \_\_\_\_ m. \_\_\_\_\_ d.  
Kaunas

Patvirtinu, kad mano, **Mangirdo Kazlausko**, baigiamasis projektas tema „AMQP protokolo pranešimų eilių valdymo saityno įrankis“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatyta piniginių sumų už šį darbą niekam nesu mokėjės.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

\_\_\_\_\_  
(vardą ir pavardę išrašyti ranka)

\_\_\_\_\_  
(parašas)

## TECHNINĖ UŽDUOTIS

Sukurti pranešimų eilių valdymo įrankį, skirtą *RabbitMQ* ir *ActiveMQ* pranešimų eilių realizacijoms. Įrankis bus naudojamas įmonės programuotojų bei testuotojų vietoje šiuo metu naudojamų mokamų įrankių. Patį įrankį turi sudaryti serverio pusės dalis ir naudotojo sasaja, realizuota kaip saityno programa. Įrankio serverio pusės dalis susideda iš dviejų posistemų:

1. Naudotojų posistemė – atsakinga už naudotojų autentifikavimą, įrankio naudotojų paskyrų sukūrimą, prisijungimo prie pranešimų skirstytojų serverio konfigūracijų valdymą, šių konfigūracijų projektų valdymą;
2. Pranešimų eilių valdymo posistemė – atsakinga už pranešimų eilių ir jose esančios informacijos valdymą.

Naudotojų autentifikavimui (kreipiantis iš saityno programos į serverio pusės dalį) turi būti naudojami JWT žetonai bei naudojamas *IdentityServer 4* karkasas (serverio pusėje).

Saityno įrankio suteikiamas funkcionalumas:

1. Naudotojų paskyrų kūrimas;
2. Prisijungimas prie saityno įrankio tiek su naudotojo paskyra, tiek svečio teisėmis;
3. Prisijungimo prie pranešimų skirstytojo serverio konfigūracijų valdymas: sukūrimas, peržiūra, redagavimas bei pašalinimas;
4. Prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektų valdymas: sukūrimas, peržiūra, redagavimas bei pašalinimas;
5. Konfigūracijų projektų importavimas iš JSON failo bei eksportavimas į JSON failą;
6. Prisijungimas prie pranešimų skirstytojo serverio;
7. Pranešimų eilių valdymas: sukūrimas, peržiūra, pašalinimas, turinio išvalymas;
8. Pranešimų patalpinimas į eilę – tiek sukuriant naują pranešimą, tiek importuojant pranešimus iš JSON failo;
9. Pranešimų eilių turinio valdymas: pranešimų grąžinimas atgal į eilę, pranešimų kopijavimas iš vienos eilės į kitą, pranešimų eksportavimas į JSON failą.

6–9 punktuose išvardintas funkcionalumas galioja tiek *RabbitMQ*, tiek *ActiveMQ* pranešimų skirstytojams bei jų eilėms.

Taip pat turi būti suprojektuota bei realizuota duomenų bazė naudotojų, prisijungimo prie pranešimų skirstytojo serverio konfigūracijų bei jų projektų informacijai saugoti (DBVS galima pasirinkti savo nuožiūra).

Pagrindiniai įrankio projektavimo, realizavimo bei testavimo darbai turi būti užbaigtai iki gegužės 4 dienos. Iki šios datos saityno įrankis turi būti paruoštas naudoti ir sudiegtas į saityno serverį (saityno paslaugų tiekėjas gali būti pasirinktas savo nuožiūra).

Kazlauskas, Mangirdas. AMQP protokolo pranešimų eilių valdymo saityno įrankis. Bakalauro baigiamasis projektas / vadovas doc. dr. Tomas Blažauskas; Kauno technologijos universitetas, informatikos fakultetas.

Mokslo kryptis ir sritis: fiziniai mokslai, programų sistemos

Reikšminiai žodžiai: pranešimų eilė, AMQP, RabbitMQ, ActiveMQ, saityno įrankis

Kaunas, 2018. 69 p.

## SANTRAUKA

Darbe pristatomas AMQP protokolo pranešimų eilių valdymo saityno įrankis. Šiomis dienomis vis labiau populiarėjant mikroservisu architektūra grįstam programinės įrangos kūrimui, didelė problema iškyla sistemą sudarančių komponentų komunikacija. Vienas iš šios problemos sprendimų – komunikacija, naudojant pranešimų eiles. Šis būdas palengvina komponentų tarpusavio komunikaciją, tačiau apsunkina sistemų kūrimo bei testavimo procesą – pranešimų eilių stebėjimui bei valdymui reikalinga papildoma programinė įranga, kartais naudojami net keli skirtini pranešimų eilių valdymo įrankiai. Todėl šio darbo tikslas – palengvinti programuotojų bei testuotojų darbą, sukuriant pranešimų eilių valdymui skirtą saityno įrankį, apjungiantį *RabbitMQ* ir *ActiveMQ* pranešimų skirstytojus bei jų pranešimų eiles.

Darbe pateikiama įrankio kūrimo analizė, apžvelgiamas įrankio aktualumas, rinkos konkurentai. Projekto dalyje identifikuojami įrankio funkciniai ir nefunkciniai reikalavimai, apribojimai, technologijos, projekto kūrimo metodika bei pats įrankio projektas. Testavimo skyriuje sudaromas testavimo planas, apžvelgiamos testavimo technikos. Taip pat pateikiamas vartotojo ir diegimo vadovai.

Darbo pabaigoje apibendrinami projekto kūrimo rezultatai ir išvados.

Kazlauskas, Mangirdas. Message Queuing Management Tool for AMQP Protocol: Bachelor's thesis / supervisor assoc. prof. Tomas Blažauskas. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: physical sciences, software systems

Keywords: message queue, AMQP, RabbitMQ, ActiveMQ, web application

Kaunas, 2018. 69 p.

## SUMMARY

A message queuing management web tool for AMQP protocol is presented in the thesis. Nowadays, when microservice architecture-oriented software development is becoming more and more popular, a big problem of the communication between the system's components occurs. One of the solutions to this problem – communication using message queues. Although this method makes the communication between the system's components easier, the development and testing process of this kind of systems becomes complicated – an extra message queues monitoring and management software is needed, sometimes even a different kind of message queues management tools are used. The objective of the thesis – make a programmers' and testers' work easier by creating a message queuing management web tool, which includes *RabbitMQ* and *ActiveMQ* message brokers and both their message queues.

Thesis begins with analysis part where tool's relevance and competitors are reviewed. Functional and non-functional requirements, restrictions, technologies as well as the software development process, tool's model are presented in the project's section. The tool's testing plan and techniques are reviewed in the testing section. Also, user manual and the deployment guide are provided.

The paper is wrapped up with the results and conclusions of the thesis.

## TURINYS

Lentelių sąrašas .....	9
Paveikslų sąrašas .....	10
Terminų ir santrumpų žodynas .....	12
Įvadas .....	13
1. Analizė .....	14
1.1. Techninis pasiūlymas .....	14
1.1.1. Sistemos apibrėžimas .....	14
1.1.2. Bendras veiklos tikslas .....	14
1.1.3. Sistemos pagrįstumas .....	14
1.1.4. Konkurencija rinkoje .....	15
1.1.5. Prototipai ir pagalbinė informacija .....	17
1.1.6. Sistemos apimtis ir ištakliai, reikalingi sistemai sukurti .....	18
1.2. Galimybių analizė .....	18
1.2.1. Techninės galimybės .....	18
1.2.2. Vartotojų pasiruošimo analizė .....	18
2. Projektas .....	19
2.1. Reikalavimų specifikacija .....	19
2.1.1. Komercinė specifikacija .....	19
2.1.2. Sistemos funkcijos .....	19
2.1.3. Apribojimai .....	31
2.1.4. Vartotojo sąsajos specifikacija .....	31
2.1.5. Realizacijai keliami reikalavimai .....	35
2.1.6. Techninė specifikacija .....	36
2.2. Projektavimo metodai .....	36
2.2.1. Projektavimo valdymas ir eiga .....	36
2.2.2. Projektavimo technologija .....	37
2.2.3. Programavimo kalbos, derinimo, automatizavimo priemonės, operacinės sistemos .....	37
2.3. Sistemos projektas .....	38
2.3.1. Statinis sistemos vaizdas .....	38
2.3.2. Dinaminis sistemos vaizdas .....	43
2.3.3. Duomenų kontrolė .....	53
3. Testavimas .....	54
3.1. Testavimo planas .....	54
3.2. Komponentų testavimas .....	54
3.2.1. Automatinis testavimas .....	54
3.2.2. Rankinis testavimas .....	55
3.3. Testavimo kriterijai .....	57

3.4. Vartotojo sąsajos testavimas .....	58
4. Dokumentacija naudotojui .....	60
4.1. Apibendrintas sistemos galimybių aprašymas .....	60
4.2. Vartotojo vadovas .....	60
4.3. Diegimo vadovas.....	66
5. Rezultatų apibendrinimas ir išvados .....	68
6. Literatūra.....	69

## LENTELIŲ SĄRAŠAS

1.1 lentelė. Rinkos konkurentų ir kuriamo įrankio palyginimas.....	17
3.1 lentelė. <i>RabbitMQ</i> pranešimų eilės pašalinimo atvejis: netinkama užklausos struktūra .....	55
3.2 lentelė. <i>RabbitMQ</i> pranešimų eilės pašalinimo atvejis: netinkama konfigūracija .....	55
3.3 lentelė. <i>RabbitMQ</i> pranešimų eilės pašalinimo atvejis: norimos ištrinti eilės nėra .....	56
3.4 lentelė. Sėkmingas <i>RabbitMQ</i> pranešimų eilės pašalinimo atvejis.....	56
3.5 lentelė. <i>Gherkin</i> sintakse aprašytų testavimo scenarijų pavyzdys .....	58

## PAVEIKSLŲ SĀRAŠAS

1.1 pav. <i>RabbitMQ</i> grafinės naudotojo sąsajos vaizdas (pranešimų eilių langas) .....	15
1.2 pav. <i>ActiveMQ</i> serverio grafinės naudotojo sąsajos vaizdas (pranešimų eilių langas) .....	16
1.3 pav. <i>QueueExplorer</i> įrankio naudotojo sąsajos vaizdas (pranešimų langas) .....	16
2.1 pav. Apibendrinta panaudos atvejų diagrama .....	19
2.2 pav. <i>UML</i> panaudos atvejų diagrama: pranešimų eilių valdymo posistemės panaudos atvejai ....	19
2.3 pav. <i>UML</i> panaudos atvejų diagrama: naudotojų posistemės panaudos atvejai .....	20
2.4 pav. <i>UML</i> veiklos diagrama: užsiregistrnuoti .....	21
2.5 pav. <i>UML</i> veiklos diagrama: prisijungti .....	21
2.6 pav. <i>UML</i> veiklos diagramos: prisijungti kaip svečiui (kairėje) ir prisijungti su naudotojo paskyra (dešinėje) .....	22
2.7 pav. <i>UML</i> veiklos diagrama: importuoti prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektus .....	22
2.8 pav. <i>UML</i> veiklos diagrama: eksportuoti prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektus .....	23
2.9 pav. <i>UML</i> veiklos diagrama: sukurti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	23
2.10 pav. <i>UML</i> veiklos diagrama: peržiūrėti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	24
2.11 pav. <i>UML</i> veiklos diagramos: redaguoti (kairėje) ir pašalinti (dešinėje) prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	24
2.12 pav. <i>UML</i> veiklos diagrama: prisijungti prie pranešimų skirstytojo serverio .....	25
2.13 pav. <i>UML</i> veiklos diagrama: peržiūrėti pranešimų eilių sąrašą .....	25
2.14 pav. <i>UML</i> veiklos diagrama: ištrinti pranešimų eilę .....	26
2.15 pav. <i>UML</i> veiklos diagrama: išvalyti pranešimų eilę .....	26
2.16 pav. <i>UML</i> veiklos diagrama: sukurti naują pranešimų eilę .....	27
2.17 pav. <i>UML</i> veiklos diagrama: siųsti pranešimą (-us) į eilę .....	28
2.18 pav. <i>UML</i> veiklos diagrama: importuoti pranešimus iš failo .....	28
2.19 pav. <i>UML</i> veiklos diagrama: redaguoti pranešimus eilėje .....	29
2.20 pav. <i>UML</i> veiklos diagrama: kopijuoti pranešimus į kitą eilę .....	30
2.21 pav. <i>UML</i> veiklos diagramos: eksportuoti pranešimus į failą (kairėje) ir grąžinti pranešimus į eilę (dešinėje) .....	31
2.22 pav. Prisijungimo langas .....	32
2.23 pav. Registracijos langas .....	32
2.24 pav. Pagrindinis įrankio šablonas .....	32
2.25 pav. Pranešimų skirstytojų konfigūracijų langas (nėra prisijungta prie konkretaus skirstytojo) ..	33
2.26 pav. Pranešimų skirstytojų konfigūracijų langas (yra pasirinktas konkretus skirstytojas) .....	33
2.27 pav. Pranešimų skirstytojų prisijungimo konfigūracijos sukūrimo modalinis dialogo langas ....	34
2.28 pav. Pranešimų eilių sąrašo peržiūros langas .....	34
2.29 pav. Pranešimų peržiūros/redagavimo langas .....	35
2.30 pav. Iteracinio projektavimo modelio schema [7] .....	37
2.31 pav. <i>UML</i> diegimo diagrama: testavimo aplinkos fizinis išdėstymas .....	38
2.32 pav. <i>UML</i> diegimo diagrama: produkcijos aplinkos fizinis išdėstymas .....	39
2.33 pav. Įrankio posistemių naudojamos duomenų bazės schema .....	39
2.34 pav. <i>UML</i> paketų diagrama: bendras projekto vaizdas .....	40
2.35 pav. <i>UML</i> klasių diagrama: būsenos valdymo moduliai bei jų sąsaja su API sluoksniu .....	41
2.36 pav. <i>UML</i> klasių diagrama: saityno programos ir posistemių sąsaja .....	42
2.37 pav. <i>UML</i> klasių diagrama: <i>RabbitMQ</i> pranešimų eilių valdymo posistemės klasės .....	43
2.38 pav. <i>UML</i> sekų diagrama: prisijungti .....	44
2.39 pav. <i>UML</i> sekų diagrama: prisijungti kaip svečiui .....	44
2.40 pav. <i>UML</i> sekų diagrama: prisijungti su naudotojo paskyra .....	45

2.41 pav. <i>UML</i> sekų diagrama: sukurti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	45
2.42 pav. <i>UML</i> sekų diagrama: peržiūrėti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	46
2.43 pav. <i>UML</i> sekų diagrama: pašalinti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	46
2.44 pav. <i>UML</i> sekų diagrama: redaguoti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją .....	47
2.45 pav. <i>UML</i> sekų diagrama: išvalyti pranešimų eilę .....	48
2.46 pav. <i>UML</i> sekų diagrama: siųsti pranešimą (-us) į eilę .....	49
2.47 pav. <i>UML</i> sekų diagrama: importuoti pranešimus iš failo .....	49
2.48 pav. <i>UML</i> sekų diagrama: sukurti naują pranešimų eilę .....	50
2.49 pav. <i>UML</i> sekų diagrama: redaguoti pranešimus eileje .....	51
2.50 pav. <i>UML</i> sekų diagrama: grąžinti pranešimus į eilę .....	52
2.51 pav. <i>UML</i> sekų diagrama: kopijuoti pranešimus į kitą eilę .....	52
2.52 pav. <i>UML</i> sekų diagrama: eksportuoti pranešimus į failą .....	53
3.1 pav. Vienetų testų vykdymo bei API metodų kodo padengimo rezultatai .....	54
3.2 pav. <i>Swagger</i> įrankiu siunčiamos užklausos pavyzdys ir gautas užklausos rezultatas .....	57
4.1 pav. AMQE prisijungimo langas .....	60
4.2 pav. Naudotojo paskyros sukūrimo langas .....	61
4.3 pav. Pagrindinis saityno įrankio langas .....	61
4.4 pav. Projekto sukūrimo langas .....	62
4.5 pav. Pranešimų skirstytojo konfigūracijos sukūrimo langas .....	62
4.6 pav. Projektų meniu .....	63
4.7 pav. Pranešimų eilių lentelės vaizdas .....	63
4.8 pav. Pranešimų nuskaitymo parametrų langas .....	63
4.9 pav. Pranešimų peržiūros/redagavimo langas .....	64
4.10 pav. Pranešimų siuntimo į kitą eilę (eilės pasirinkimo) langas .....	64
4.11 pav. Pranešimų eilės sukūrimo langas .....	64
4.12 pav. Pranešimo (-ų) siuntimo langas .....	65
4.13 pav. Projektų meniu su atsi Jungimo nuo pranešimų skirstytojo serverio mygtuku vaizdas .....	65
4.14 pav. Projektų informacijos importo/eksporto langas .....	66
4.15 pav. Serverio pusės kodo diegimo scenarijai į testavimo (kairėje) ir produkcijos (dešinėje) aplinkas .....	66
4.16 pav. Saityno programos kodo diegimo scenarijai į testavimo (kairėje) ir produkcijos (dešinėje) aplinkas .....	67

## TERMINŲ IR SANTRUMPŪ ŽODYNAS

- API (*angl. Application Programming Interface*) – sistemos ar programos suteikiama sąsaja, kuria naudojantis programuotojai gali pasiekti jos funkcionalumą ar apsikeisti duomenimis.
- AMQP (*angl. Advanced Message Queueing Protocol*) – viešai prieinamas komunikacijos pranešimais, naudojant pranešimų eiles, standartas, skirtas pranešimų skirstytojų programinei įrangai.
- AWS (*Amazon Web Services*) – kompanijos *Amazon* dukterinė įmonė, teikianti debesų kompiuterijos paslaugas.
- DBVS – duomenų bazių valdymo sistema.
- DNS (*angl. Domain Name System*) – hierarchinė srities vardų struktūra, leidžianti į tinklo resursus kreiptis naudojant jiems priskirtą simbolinį vardą, perkoduojant jį į kompiuterio identifikatoriaus (IP) adresą.
- FIFO (*angl. „First In, First Out“*) – eilėje esančios informacijos valdymo principas, kai eilės priekyje esantis (pirmiau į eilę patekęs) elementas apdorojamas pirmiau už esančius eilės gale.
- HTTP (*angl. Hypertext Transfer Protocol*) – protokolas, skirtas pasiekti bei keistis informacija pasaulyje tinkle (WWW).
- Integravimo aplinka – programinė įranga, naudojama kaip programų kūrimo aplinka, turinti daug papildomų priemonių, palengvinančių kitos programinės įrangos kūrimo procesą.
- JSON (*angl. JavaScript Object Notation*) – informacijos saugojimo ir perdavimo atviro kodo formatas, kuriame saugomi duomenų objektai sudaryti iš atributo ir reikšmių porų.
- JWT (*angl. JSON Web Token*) – JSON sintakse grįstas atviro kodo standartas, skirtas prieigos žetonų, saugančių naudotojui priskirtas teises, aprašymui bei kūrimui.
- MOM (*angl. Message-Oriented Middleware*) – programinės arba aparatinės įrangos sluoksnis, atsakingas už pranešimų gavimą bei paskirstymą tarp komunikaciją pranešimais naudojančių sistemų arba sistemos komponentų.
- Pranešimų skirstytojas – programinė įranga, kuri yra atsakinga už komunikacijos pranešimais valdymą: pranešimų paskirstymą gavėjams, pranešimų eilių valdymą, pranešimų siuntimą ir kt.
- SSL (*angl. Secure Sockets Layer*) – protokolas, skirtas internečiame perduodamos informacijos apsaugai, šifravimui.
- SQL (*angl. Structured Query Language*) – standartinė kalba, skirta reliacinių duomenų bazių duomenų aprašymui bei valdymui.
- UML (*angl. Unified Modeling Language*) – modeliavimo ir specifikacijų kūrimo kalba, skirta atvaizduoti bei dokumentuoti objektines programų sistemas.

## ĮVADAS

Didelėms sistemoms kurti vis dažniau pasirenkamas mikroservisu architektūra grįstas programinės įrangos kūrimas, kai keli vienas nuo kito nepriklausomi komponentai veikia individualiai, tačiau komunikuoja tarpusavyje, tokiu būdu sujungdami atskirus komponentus į vieną sistemą. Komponentų komunikacijos realizacija yra tarsi atskira tokio tipo sistemų problema, sprendžiama įvairiais būdais. Vienas jų – komunikacija naudojant pranešimų eiles. Vieni komponentai į eiles talpina pranešimus su tam tikra informacija, kuri turi būti perduodama kitoms sistemos dalims, o kiti – laukia eilėse atsirandančių pranešimų ir juos apdoroja, pvz., atlikdami tam tikrą užduotį. Tačiau toks komunikacijos būdas turi trūkumų: eilėse esančių pranešimų turinys negali būti laisvai peržiūrimas ar koreguojamas, nuskaičius pranešimą reikia pasirūpinti, kad jis būtų grąžintas atgal į eilę. Pranešimų eilių stebėjimui bei valdymui yra naudojama papildoma programinė įranga, tačiau dažniausiai ji būna pritaikyta tik konkretiems pranešimų skirstytojams, pavyzdžiui, *RabbitMQ*, *ActiveMQ*, *MSMQ* ir kt. Didelėse sistemoje kartais yra naudojamos net kelios skirtinges pranešimų eilių realizacijos, kas apsunkina programuotojų bei testuotojų darbą, kai norima pranešimų eiles stebeti ar pakoreguoti eilėse esančią informaciją. Dėl šių priežascių nutarta sukurti universalų pranešimų eilių valdymo saityno įrankį, apimanti *RabbitMQ* bei *ActiveMQ* pranešimų eilių realizacijas.

Darbo tikslas - palengvinti komunikaciją pranešimais naudojančias sistemos kuriančių bei testuojančių programuotojų ir testuotojų darbą, sukuriant pranešimų eilių valdymui skirtą saityno įrankį. Šio tiuko įgyvendinimui iškelti tokie uždaviniai:

- 1) Atlikti konkurentų analizę, išbandyti bei įvertinant rinkoje esančius panašius įrankius;
- 2) Specifikuoti įrankio kūrimo reikalavimus;
- 3) Remiantis analizės metu surinktais duomenimis, sudaryti įrankio projektą;
- 4) Remiantis sudaryti projektu, realizuoti bei ištetsuoti saityno įrankį;
- 5) Dokumentuoti sukurtą įrankį bei jo naudotojo sąsają.

Darbą sudaro analizės, projekto, testavimo ir dokumentacijos dalys. Analizės dalyje apibrėžiama kuriamo įrankio koncepcija, jo kūrimo tikslas ir pagrįstumas, išnagrinėjami rinkoje esantys panašūs įrankiai. Projekto dalyje apibrėžiami įrankio funkciniai ir nefunkciniai reikalavimai, naudotojo sąsajos langų prototipai, kūrimo priemonės, technologijos, projekto valdymo metodika, analizuojamas įrankio statinis bei dinaminis vaizdas. Testavimo dalyje pateikiamas testavimo planas, naudojami testavimo metodai, apžvelgiama testavimo eiga bei rezultatai. Dokumentacijoje pateikiamas vartotojo vadovas, įrankio diegimo bei administravimo vadovai.

Pabaigoje pateikiami darbo rezultatai bei įrankio analizės, projektavimo ir kūrimo eigoje prieitos išvados.

## 1. ANALIZĖ

Šiame skyriuje apžvelgiama kuriamo įrankio idėja, problematika, kūrimo pagrįstumas, nagrinėjami rinkoje esantys konkurentai (panašūs įrankiai), naudojami prototipai, apibrėžiama įrankio apimtis bei jo realizavimo galimybės.

### 1.1. Techninis pasiūlymas

#### 1.1.1. Sistemos apibrėžimas

AMQE (*angl. Advanced Message Queues Explorer*) – pažangus pranešimų eilių valdymo įrankis, leidžiantis stebeti bei valdyti pranešimų eiles, kurias naudoja įvairios komunikacijos pranešimais pagrįstos sistemos. Patį įrankį sudaro naudotojų ir pranešimų eilių valdymo posistemės. Naudotojų posistemės paskirtis – valdyti naudotojų paskyras bei prieigą prie saityno įrankio, kurti ir redaguoti konfigūracijas, skirtas prisijungti prie pranešimų eilių skirstytojų serverių, grupuoti konfigūracijas į projektus. Pranešimų eilių valdymo posistemė suteikia API, skirtą valdyti skirtinges AMQP protokolo pranešimų skirstymo tarpinės programinės įrangos (*angl. MOM – message-oriented middleware*) realizacijas, taip leidžiant įrankio naudotojams valdyti pranešimų eiles (kurti, šalinti, išvalyti ir kt.) bei keisti eilėse esančią informaciją (įdėti naujus ar koreguoti jau esamus pranešimus eilėje).

Taip pat pats įrankis turi grafinę naudotojo sąsają – saityno programą, kuri leidžia naudotojams stebeti ir valdyti pranešimų eiles, nesikreipiant į pranešimų eilių valdymo posistemės suteikiamą API tiesiogiai.

#### 1.1.2. Bendras veiklos tikslas

Bendras veiklos tikslas - sukurti pranešimų eilių valdymo ir stebėjimo įrankį, kuris suteiktų vieningą grafinę sąsają skirtingu pranešimų skirstytojų realizacijų suteikiamų pranešimų eilių valdymui. Įrankis bus naudojamas programuotojų ir testuotojų, kurie kuria, palaiko ar testuoja sistemas, naudojančias komunikaciją pranešimais, ir kuriems reikalingas įrankis, skirtas stebeti bei valdyti pranešimų eiles bei jose esančius pranešimus, įdėti naujus pranešimus į eiles, taip testuojant pranešimų apdorojimą sistemose.

#### 1.1.3. Sistemos pagrįstumas

Šiuo metu didelėms sistemoms kurti vis dažniau yra pasirenkamas mikroservisu architektūra grįstas programinės įrangos kūrimas (kai sistemą sudaro atskirų bei nepriklausomų komponentų - servisu junginys, kur kiekvienas iš jų gali veikti nepriklausomai). Kuriant tokio tipo sistemas, viena didžiausių atsirandančių problemų – komponentų tarpusavio komunikacija. Vienas iš komponentų komunikavimo sprendimų yra komunikacija pranešimais (*angl. message queueing*) [1]. Šio metodo esmė – servisams komunikuoti yra naudojamos bendros pranešimų eilės, į kurias gali būti talpinami ir iš kurių gali būti skaitomi pranešimai, turintys tam tikrą informaciją, pvz., komandos pavadinimą ir tai komandai atliliki reikalingus duomenis. Tokiu būdu vieni servisi talpina pranešimus į atitinkamas eiles, o kiti laukia eilėse atsirandančių pranešimų ir juos apdoroja. Viena didžiausių tokios komunikacijos problemų – pranešimų eilių ir jose esančios informacijos stebėjimas, peržiūra bei koregavimas: nuskaičius pranešimą, jis iš eilės yra pašalinamas ir toliau nebéra apdorojamas, todėl norint tik peržiūrėti pranešime esančią informaciją, reikia papildomai pasirūpinti, kad pranešimas būtų patalpintas atgal į eilę. Taip pat pranešimų eilės veikia FIFO principu, todėl norint peržiūrėti pranešimą, kuris yra ne eilės priekyje, papildomai reikia pasirūpinti prieš jį esančiais pranešimais, kad jie niekur nedingtų ir būtų grąžinti atgal į eilę.

Pranešimų eilių valdymas gali būti atliekamas programiniu būdu, tačiau tai užima daug laiko ir testavimo procesas ilgainiui tampa varginantis. Šioms problemoms spręsti yra naudojama papildoma programinė įranga, kurios suteikiama grafinė sąsaja padeda lengvai valdyti pranešimų eiles bei jų turinį. Tačiau neretai programuotojai ar testuotojai dirba prie skirtingu projektų arba didelė sistema gali naudoti net kelis skirtinges pranešimų skirstymo tarpinės programinės įrangos sprendimus (skirtingas AMQP protokolo realizacijas), kurių eilėms valdyti reikalinga skirtinė programinė įranga

– tai taip pat apsunkina programuotojo bei testuotojo darbą bei papildomai eikvoja techninės įrangos resursus.

Plečiantis projektų apimčiai bei didėjant projektų skaičiui, šios problemos tampa dar didesnės, todėl joms spręsti nutarta sukurti pranešimų eilių stebėjimo bei valdymo įrankį, kuris palengvintų eilėse esančių pranešimų valdymą bei pakeistų skirtinį AMQP realizaciją eilių valdymui skirtą programinę įrangą į vieną centralizuotą saityno įrankį.

#### 1.1.4. Konkurencija rinkoje

Kuriamas pranešimų eilių valdymo įrankis pritaikytas dviem – *RabbitMQ* bei *ActiveMQ* – AMQP protokolo pranešimų skirstymo tarpinės programinės įrangos realizacijoms, todėl konkurentų analizė apima šių dviejų pranešimų skirstytojų serverių suteikiamas valdymo sasajas bei vieną bendrą įrankį, kuriuo galima valdyti tiek *RabbitMQ*, tiek *ActiveMQ* eiles.

*RabbitMQ* serveris gali būti praplėstas papildomais įskiepiams ir suteikti grafinę naudotojo sasają šio pranešimų skirstytojo eilėms valdyti. Būtent šiam tikslui skirtas *rabbitmq-management* įskiepis, kuris yra įrašomas kartu su *RabbitMQ* serveriu, tačiau tam, kad veiktu, turi būti papildomai aktyvuotas [2]. Kaip jau minėta anksčiau, šis įskiepis ne tik suteikia HTTP protokolu pasiekiamą sasają (API) *RabbitMQ* serverio valdymui bei stebėjimui, tačiau kartu turi ir grafinę naudotojo sasają (1.1 pav.), kurios pagalba naudotojai gali valdyti pranešimų eiles, jų turini, keisti pranešimų eilių parametrus, valdyti naudotojų paskyras, stebėti *RabbitMQ* serverio būklę ir kt.

Overview		Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
TestingQueue	AD	idle	0	0	0			

#### 1.1 pav. *RabbitMQ* grafinės naudotojo sasajos vaizdas (pranešimų eilių langas)

Nors įskiepio suteikiamas naudotojo sasajos funkcionalumas yra gana platus, tačiau šios sasajos pagalba nėra galimybės koreguoti eilėse esančių pranešimų – juos galima tik peržiūrėti ir vėliau, priklausomai nuo pasirinkto parametru, arba grąžinti atgal į eilę, arba ne. Šio funkcionalumo būtent ir trūksta programuotojams bei testuotojams, norintiems pakeisti eilėje esančių pranešimų turinį ir taip testuoti komunikaciją pranešimais naudojančių sistemų veikimą.

Savo ruožtu, *ActiveMQ* pranešimų skirstytojo serveris taip pat siūlo grafinę naudotojo sasają šios realizacijos pranešimų eilėms valdyti [3]. Priešingai nei *RabbitMQ* sasaja, šios įjungimui nereikia atlikti jokių papildomų veiksmų, ji veikia automatiškai paleidus *ActiveMQ* serverį (1.2 pav.).

**1.2 pav.** *ActiveMQ* serverio grafinės naudotojo sąsajos vaizdas (pranešimų eilių langas)

Taip pat kaip ir *RabbitMQ* aplinkoje, *ActiveMQ* serverio sąsaja suteikia galimybę valdyti pranešimų eiles: jas kurti, trinti, išvalyti, į eilę patalpinti naujus arba nuskaityti esamus pranešimus pasirinktoje eilėje. Taip pat yra suteikiamas pranešimų eilių prenumeratorių, aktyvių prisijungimų (jvairiai protokolais), pranešimų siuntimo tvarkaraščių valdymo funkcionalumas. Priešingai nei naudojantis *RabbitMQ* sąsaja, *ActiveMQ* eilėse esantys pranešimai gali būti naršomi, t.y. peržiūrimas kiekvienas individualus pranešimas eilėje, juos galima nuskaityti bet kokia tvarka, tačiau pranešimai gali būti peržiūrimi tik po vieną, norint juos pašalinti, jie taip pat ištrinami individualiai. Lygiai taip, kaip ir *RabbitMQ* sąsajos atveju, pranešimų koreguoti negalima, galima tik peržiūrėti jų parametrus bei turinį.

Vienas iš rinkoje siūlomų pranešimų eilių valdymo įrankių – *QueueExplorer* (1.3 pav.). Šis įrankis, priešingai nei anksčiau paminėtos sąsajos, yra darbalaukio programinė įranga, kurią reikia įsidiegti kompiuteryje. *QueueExplorer* leidžia dirbti su *MSMQ*, *Azure Service Bus*, *ActiveMQ* ir *RabbitMQ* (po 2018m. kovo mėnesio atnaujinimo, 4.2 versijos) pranešimų eilių realizacijomis [4].

**1.3 pav.** *QueueExplorer* įrankio naudotojo sąsajos vaizdas (pranešimų langas)

Įrankis suteikia labai platų pranešimų eilių valdymo funkcionalumą, iš esmės praplėsdamas konkrečių pranešimų skirstytojų serverių suteikiamu sąsajų funkcionalumą – su pranešimų eilėmis, pranešimais gali būti atliekami visi tie patys veiksmai (eilių valdymas, pranešimų siuntimas, nustatant konkrečius parametrus ir kt.), tačiau papildomai galima peržiūrėti/koreguoti individualius pranešimus,

jų informaciją, rikiuoti/filtruoti pranešimus pagal konkrečius jų parametrus, testuoti pranešimų skirstytojų serverių veikimą patalpinant didelį pranešimų kiekį į eilę vienu metu, padaryti pranešimų eilių ir jose esančių pranešimų atsargines kopijas ir kt. [5]

*QueueExplorer* suteikia dvi įrankio versijas – *Standard* bei *Professional*. *Professional* versija išplečia *Standard* įrankio galimybes, suteikdama galimybę dirbtį su keliais pranešimų eilių skirstytojų serveriais vienu metu, juos struktūruoti, konfigūruoti pranešimų eilių peržiūros langą pagal savo kriterijus ir kt. [6] Įrankis yra mokamas (po 14 dienų nemokamo bandomojo laikotarpio): *Standard* licencija kainuoja 99\$, o *Professional* – 199\$.

Aprašytos pranešimų eilių valdymo sėsajos (*RabbitMQ* sėsaja ir *ActiveMQ* sėsaja, lentelėje sutrumpintai žymimos atitinkamai *RabbitMQ* ir *ActiveMQ*), *QueueExplorer* ir kuriamas saityno įrankis AMQE buvo lyginami tarpusavyje, atsižvelgiant į jų funkcionalumą, kainą bei kitus aspektus. Šios analizės rezultatai pateikiami 1.1 lentelėje.

### 1.1 lentelė. Rinkos konkurentų ir kuriamo įrankio palyginimas

	RabbitMQ	ActiveMQ	QueueExplorer	AMQE
<i>RabbitMQ</i> pranešimų eilių palaikymas	Yra	Néra	Yra	Yra
<i>ActiveMQ</i> pranešimų eilių palaikymas	Néra	Yra	Yra	Yra
Kitų papildomų realizacijų (pvz., <i>MSMQ</i> ) pranešimų eilių palaikymas	Néra	Néra	Yra	Néra
Pranešimų turinio redagavimo palaikymas	Néra	Néra	Yra	Yra
Prisijungimo prie pranešimų skirstytojų serverių konfigūracijų struktūrizavimas	Néra	Néra	Yra ( <i>Professional</i> versijoje)	Yra
Darbo su keliais pranešimų skirstytojų serveriais vienu metu palaikymas	Néra	Néra	Yra ( <i>Professional</i> versijoje)	Néra
Pranešimų eilių valdymo saityno sėsaja	Yra	Yra	Néra	Yra
Licencijos kaina	Nemokama	Nemokama	<i>Standard</i> – 99\$, <i>Professional</i> – 199\$	Nemokama

Iš konkurentų analizės apibendrinimo lentelės matyti, kad daugiausia funkcionalumo palaiko *QueueExplorer* programinė įranga, tačiau ji yra mokama bei neturi saityno sėsajos (programinę įrangą reikia įdiegti kompiuteryje). Tieki *RabbitMQ*, tieki *ActiveMQ* serverių suteikiamų sėsajų funkcionalumas yra panašus, tik jis apsiriboja ties atitinkamų pranešimų eilių realizacijomis bei neturi pranešimų turinio redagavimo galimybės. Kuriamas įrankis, atsižvelgiant į pranešimų eilių valdymą, apjungia *RabbitMQ* ir *ActiveMQ* pranešimų eilių valdymo sėsajas į vieną nemokamą saityno įrankį, tačiau jo siūlomų funkcijų spektras, lyginant su *QueueExplorer*, yra mažesnis.

Apžvelgus ir įvertinus rinkoje esančius pranešimų eilių valdymo įrankius galima teigti, kad kuriamas įrankis tenkina lūkesčius dėl kelių priežasčių: jis, lyginant su platesnio funkcionalumo įrankiu *QueueExplorer*, yra nemokamas ir pasiekiamas per saityno sėsają - jo nereikia diegti kompiuteryje, o centralizuotas *RabbitMQ* ir *ActiveMQ* pranešimų eilių valdymas bei pranešimų turinio redagavimo galimybė leidžia kuriamam įrankiui lenkti atitinkamų pranešimų skirstytojų serverių sėsajų suteikiamą pranešimų eilių valdymo funkcionalumą.

#### 1.1.5. Prototipai ir pagalbinė informacija

Naudotojų posistemės dalis - autentifikacijos sluoksnis, kurio kūrimui buvo panaudotas atviro kodo *OpenID connect* standartą bei *OAuth 2.0* protokolą realizuojantis karkasas *IdentityServer 4*. Šis karkasas buvo naudojamas saityno įrankio naudotojų autentifikacijai bei prieigos prie pranešimų valdymo posistemės API žetonų suteikimui. Pats karkasas yra lengvai integruojamas, patys kūrėjai yra paruošę ne vieną karkaso integracijos pavyzdį .Net Core aplinkoje (pavyzdžius galima rasti *GitHub* saugykloje: <https://github.com/IdentityServer/IdentityServer4.Samples>).

Įrankio naudotojo sėsaja bei pranešimų eilių valdymo posistemė buvo kuriamą nesiremiant jokiais prototipais ar panašiomis realizacijomis, tačiau buvo naudojamos papildomos bibliotekos, pvz., atviro kodo biblioteka *Dapper*, kuri praplėtia .NET karkasso sėsają *IdbConnection*, taip palengvindama darbą su duomenų baze – vykdant *SQL* užklausas bei priskiriant gautus užklausos rezultatus konkrečioms duomenų objektų klasėms.

## **1.1.6. Sistemos apimtis ir ištekliai, reikalingi sistemai sukurti**

Įrankio serverio pusės dalį sudarys dvi - naudotojų ir pranešimų eilių valdymo – posistemės. Abi posistemės kartu yra pačio įrankio pagrindas, nes saityno programa kreipsis į serverio pusės dalį (šias posistemes) naudojantis jų suteikiama sasaja. Kita svarbi kuriamo įrankio dalis – naudotojo sasaja, kuri kuriamą atskirai nuo jau minėtųjų posistemų. Ši sasaja nėra tiesiog paprastas interneto puslapis, skirtas prieigai prie posistemų API – tai naudotojo naršyklėje veikianti programinė įranga, todėl kuriamų posistemų ir šios sasajos apimtis yra panaši. Ivertinus kuriamo įrankio teikiamo funkcionalumo apimtį bei jo posistemų ir naudotojo sasajos realizavimo aspektus, ivertinta, kad įrankiui sukurti bei dokumentuoti reikia apie 450 valandų. Planuojama, kad įrankio posistemėms sukurti bus parašyta apie 3000 C# kodo eilučių (be automatinių testų), o naudotojo sasajai – apie 4500 kodo eilučių (*HTML 5, CSS 3, JavaScript, Vue.js* komponentai).

Atsižvelgiant į kuriamo įrankio apimtį bei numatytus apribojimus, įrankiui sukurti užtenka vieno programuotojo.

## **1.2. Galimybų analizė**

### **1.2.1. Techninės galimybės**

Įrankis kuriamas naudojant atviro kodo karkasą *ASP.NET Core 2.0* serverio pusės daliai bei *Vue.js* karkasą saityno programos kūrimui. Abu šie karkasai yra populiarūs programuotojų bendruomenėje, yra naudojami įmonėse saityno programų bei įvairių sistemų kūrimui. *RabbitMQ* ir *ActiveMQ* pranešimų eilių realizacijos taip pat yra plačiai bei sėkmingai naudojamos rinkoje, turi išsamią dokumentaciją, yra ištstuotos. Atsižvelgiant į šią informaciją (įrankio kūrimui naudojamas technologijas), nėra jokių techninių kliūčių, kurios trukdytų realizuoti įrankį.

Įrankis bus diegiamas naudojant *AWS* debesų kompiuterijos paslaugas bei pasiekiamas naudojant interneto naršyklę, todėl jokių papildomų infrastruktūros sprendimų, reikalingų įrankio naudojimuisi, įmonei realizuoti nereikia. Jei atsirastų poreikis įrankį įdiegti lokaliuose įmonės serveriuose, tai taip pat nesudarytų jokių kliūčių – saityno programa yra sukompiliuojama į statinius *HTML*, *JavaScript* bei *CSS* failus, todėl gali būti diegama bet kuriame saityno serveryje, o serverio pusės dalis, kaip jau aprašyta anksčiau, yra kuriamą naudojant *ASP.NET Core 2.0* karkasą, kas leidžia serverio pusės dalies kodą diegti tiek *Windows*, tiek *Linux* operacines sistemas palaikančiuose serveriuose.

Atsižvelgiant į įrankio realizavimui naudojamas technologijas bei įrankio diegimui reikalingą infrastruktūrą, galima teigt, kad visos techninės galimybės yra išpildytos, jokių kliūčių įrankio realizavimui nėra.

### **1.2.2. Vartotojų pasiruošimo analizė**

Įrankis realizuoja *RabbitMQ* bei *ActiveMQ* pranešimų eilių valdymą, todėl norint sėkmingai juo naudotis, reikia išmanyti šių tipų pranešimų skirstytojų realizacijas, būti susipažinus su šių tipų pranešimų eilėmis. Patį įrankį naudos įmonės programuotojai bei testuotojai, kuriantys ar testuojantys sistemas, paremtas mikroservisių architektūra ir tarp atskirų sistemos servisių naudojančias komunikaciją pranešimais. Kadangi didžioji dalis įmonės projektų remiasi šia architektūra bei komunikacijos pranešimais principu, įmonės testuotojai bei programuotojai šiuo metu naudoja rinkos analizėje išvardintus panašius įrankius (pvz., minėtajį *QueueExplorer*), todėl galima teigt, kad įmonės darbuotojai turi pakankamai kompetencijos suprasti kuriamą įrankio paskirtį, o pačio įrankio sasają geriau perprasti padės vartotojo vadovas, aprašytas šio dokumento 4.2 skyriuje.

## 2. PROJEKTAS

Šiame skyriuje specifikuojamas įrankio funkcionalumas, kūrimo apribojimai, pateikiami vartotojo sasajos prototipo langų eskizai, apibrėžiami nefunkciniai reikalavimai. Taip pat nagrinėjamos kūrimo priemonės, technologijos, projekto valdymo metodika, analizuojamas įrankio statinis bei dinaminis vaizdas.

### 2.1. Reikalavimų specifikacija

#### 2.1.1. Komercinė specifikacija

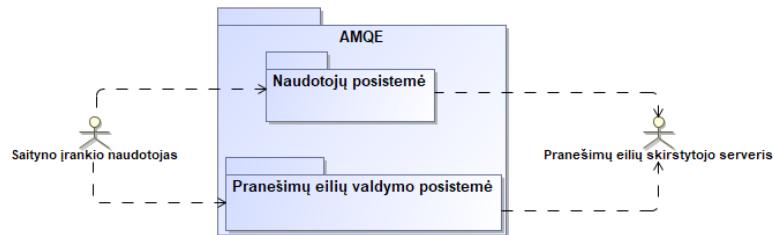
Projekto užsakovas – UAB „CID Lietuva“ – įmonė, kurioje buvo atliekama privalomoji praktika. Projektą vykdo įmonės platformos komanda (*angl. platform squad*), kuri yra atsakinga už įmonės projektuose naudojamų technologijų ir įrankių kūrimą, priežiūrą, projektų diegimą. Verta paminėti, kad nors projekto užsakovas yra pati įmonė, tačiau nutarta, jog sukurtas įrankis vėliau būtų prieinamas kaip atviro kodo programa.

Numatoma projekto pristatymo data – 2018 m. gegužės 4 d. Iki šios datos įrankis turi būti suprogramuotas, ištestuotas ir būti tinkamas naudoti įmonės programuotojų reikmėms.

Projektui kainos apribojimai nėra taikomi, joks biudžetas nėra numatytas.

#### 2.1.2. Sistemos funkcijos

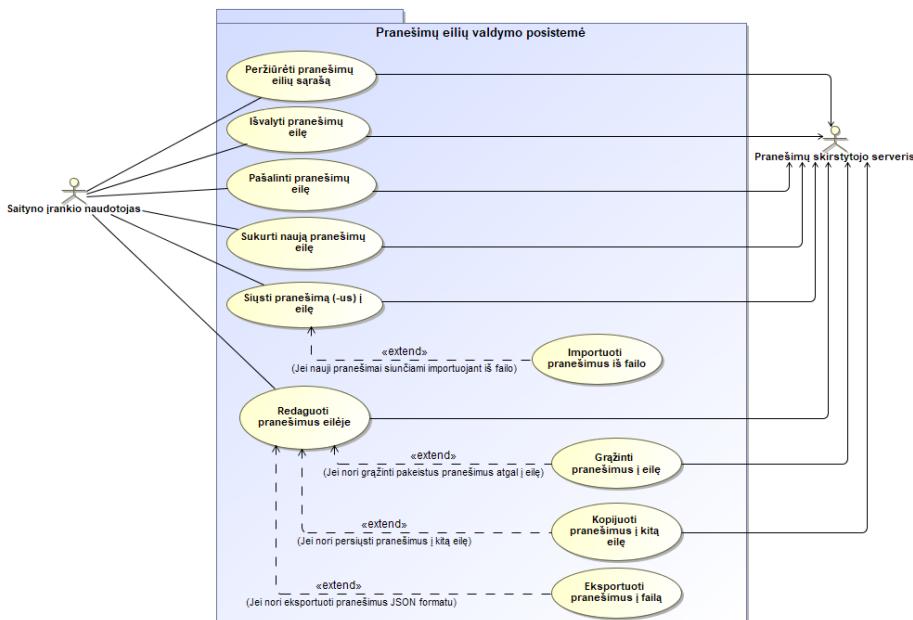
Sistemos funkciniai reikalavimai pavaizduoti UML panaudojimo atvejų diagramomis. 2.1 pav. pateikta apibendrinta panaudos atvejų diagrama.



2.1 pav. Apibendrinta panaudos atvejų diagrama

Apibendrintoje panaudos atvejų diagramoje matyti, kad įrankį sudaro dvi – naudotojų ir pranešimų eilių valdymo – posistemės. Abi posistemės kreipiasi į pranešimų eilių skirstytojo serverį posistemui veiksmams atlikti.

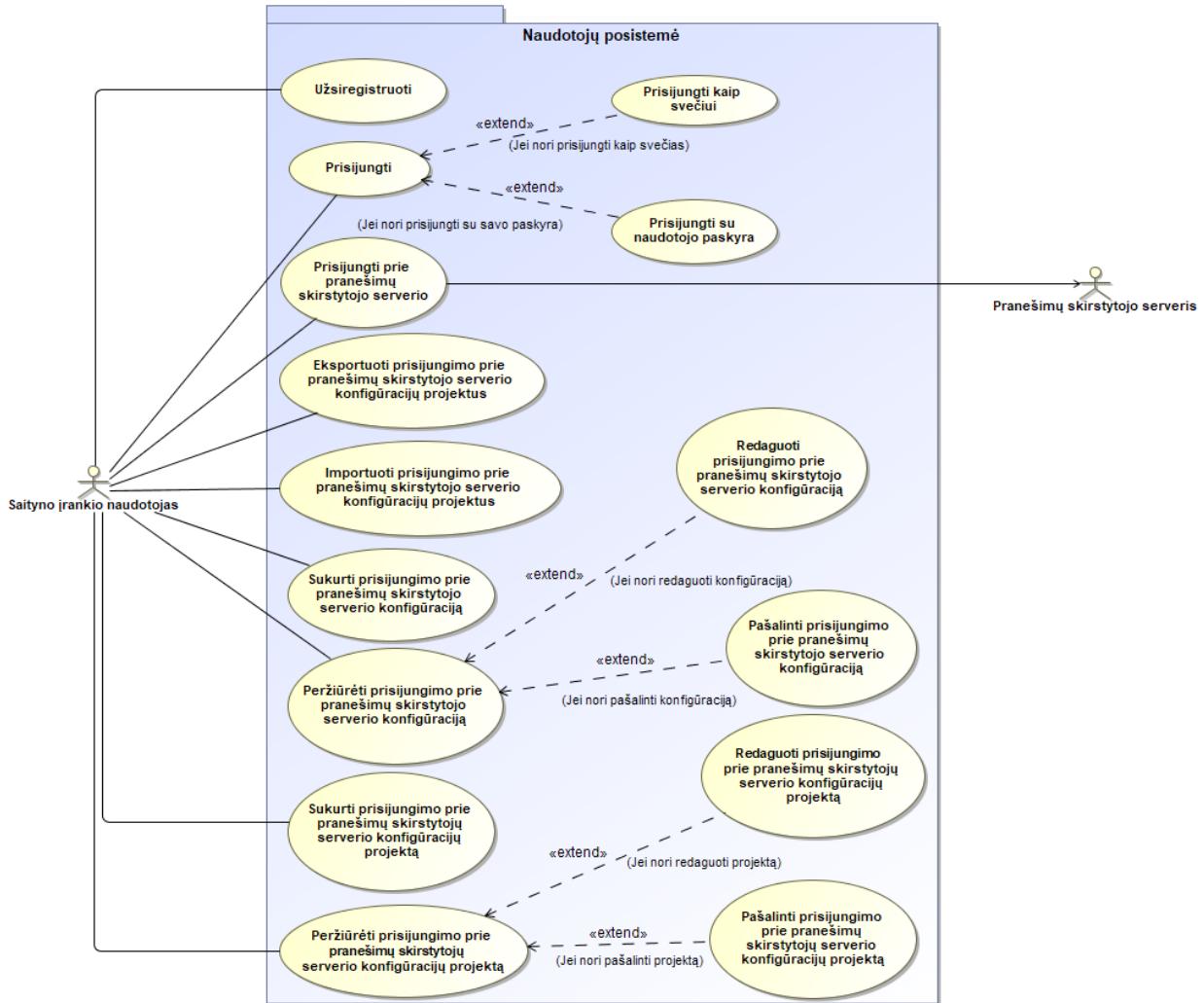
2.2 pav. Pateikta pranešimų eilių valdymo posistemės panaudos atvejų diagrama.



2.2 pav. UML panaudos atvejų diagrama: pranešimų eilių valdymo posistemės panaudos atvejai

Saityno įrankio naudotojas turi galimybę peržiūrėti pranešimų eilių sąrašą, išvalyti, pašalinti, sukurti naują pranešimų eilę, siusti pranešimus į ją (sukuriant naują pranešimą arba importuojant pranešimus iš failo). Taip pat naudotojas gali redaguoti pranešimus eilėje – grąžinti pakeistus pranešimus atgal į eilę, kopijuoti pranešimus į kitą eilę arba eksportuoti į failą. Pranešimų eilės sukūrimo, šalinimo, išvalymo, pranešimų siuntimo/grąžinimo į eilę bei kopijavimo veiksmų atlikimui įrankis kreipiasi į prisijungto pranešimų skirstytojo serverį.

2.3 pav. pateikta naudotojų posistemės panaudos atvejų diagramma.

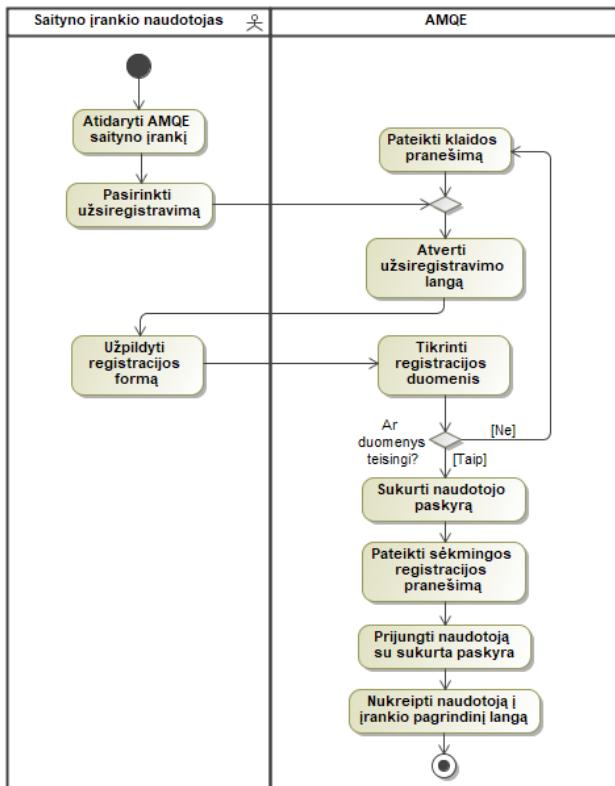


**2.3 pav.** UML panaudos atvejų diagramma: naudotojų posistemės panaudos atvejai

Saityno įrankio naudotojas gali užsiregistravoti, prisijungti prie saityno įrankio su savo paskyra arba kaip svečias. Taip pat naudotojas gali kurti, redaguoti bei pašalinti pranešimų skirstytojo serverio prisijungimo konfigūracijas. Naudodamas šiomis konfigūracijomis, naudotojas turi galimybę prisijungti prie pasirinkto pranešimų skirstytojo serverio (šiam veiksmui atlikti yra kreipiamasi į patį pranešimų skirstytojo serverį). Taip pat pranešimų skirstytojų prisijungimo konfigūracijos gali būti grupuoojamos į projektus – naudotojas turi galimybę kurti, redaguoti bei šalinti tokius konfigūracijų projektus. Pranešimų skirstytojų prisijungimo konfigūracijų projektais taip pat gali būti eksportuojami į JSON failą bei importuojami iš jo.

Panaudos atvejai dokumentuojami *UML* veiklos diagramomis, kurios parodo panaudos atvejų galimus vykdymo scenarijus. 2.4 - 2.12 paveikslėliuose pateiktos naudotojų posistemės panaudos atvejų veiklos diagramos.

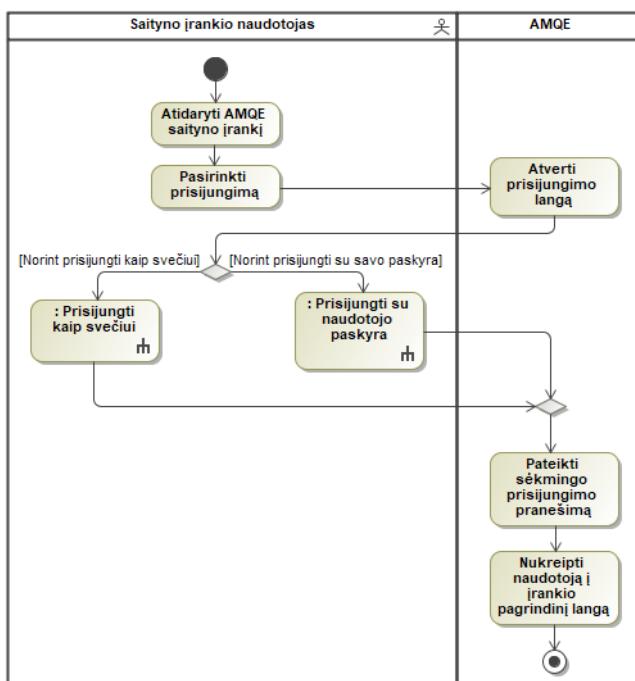
2.4 pav. pavaizduota naudotojo registracijos veiklos diagrama.



2.4 pav. UML veiklos diagramma: užsiregistruoti

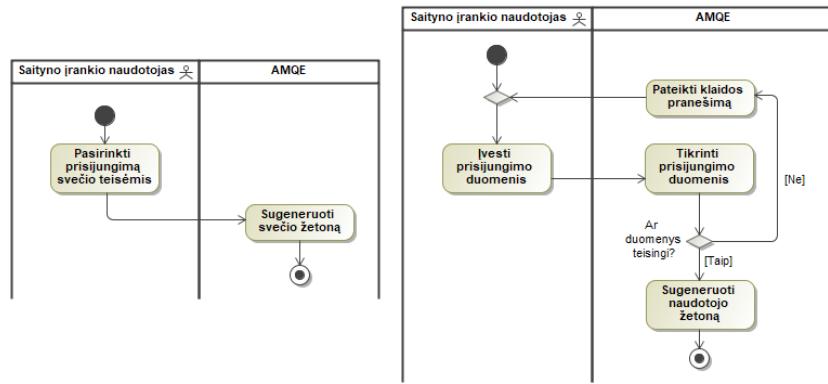
Naudotojas, norėdamas susikurti savo paskyrą, turi atidaryti registracijos puslapį, įvesti reikiamus duomenis registracijos formoje. Jei duomenys yra teisingi, sukuriama nauja paskyra ir naudotojas yra iškart prijungiamas prie įrankio bei nukreipiamas į jo pagrindinį langą, kitu atveju – pateikiamas klaidos pranešimas.

2.5 pav. pateikta naudotojo prisijungimo veiklos diagramma.



2.5 pav. UML veiklos diagramma: prisijungti

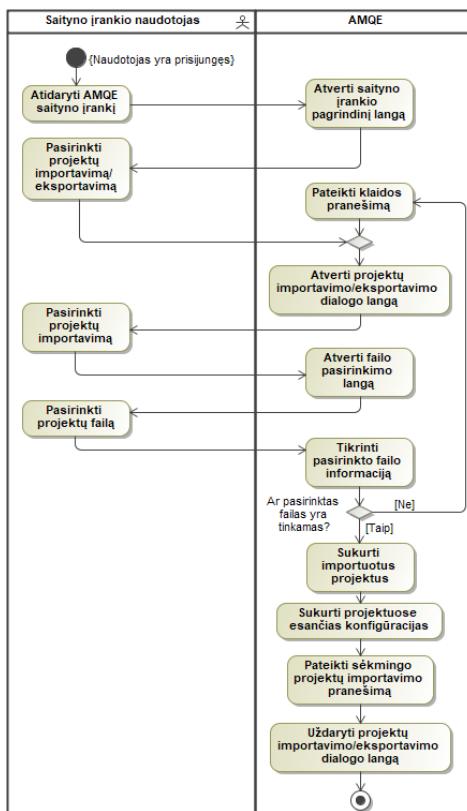
Naudotojui atidarius saityno įrankį, jam pateikiamas prisijungimo langas. Jis gali pasirinkti, ar nori prisijungti kaip svečias, ar nori prisijungti su savo paskyrą (šių veiksmų veiklos diagrammos pateiktos 2.6 pav.). Po sėkmingo prisijungimo vienu iš būdų, naudotojas yra informuojamas apie sėkmingą prisijungimą ir nukreipiamas į pagrindinį saityno įrankio langą.



**2.6 pav.** UML veiklos diagramos: prisijungti kaip svečiu (kairėje) ir prisijungti su naudotojo paskyra (dešinėje)

Naudotojų prijungiant kaip svečią, jam tiesiog sugeneruojanas svečio žetonas (kad būtų galima autorizuoti naudotoją kaip svečią), o prisijungiant su savo paskyrą, naudotojas turi įvesti savo prisijungimo duomenis ir jei duomenys teisingi, naudotojui sugeneruojanas naudotojo žetonas su Jame užšifruota naudotojo informacija (el. paštas, naudotojo identifikacijos ir kt.), kitu atveju – pateikiamas klaidos pranešimas.

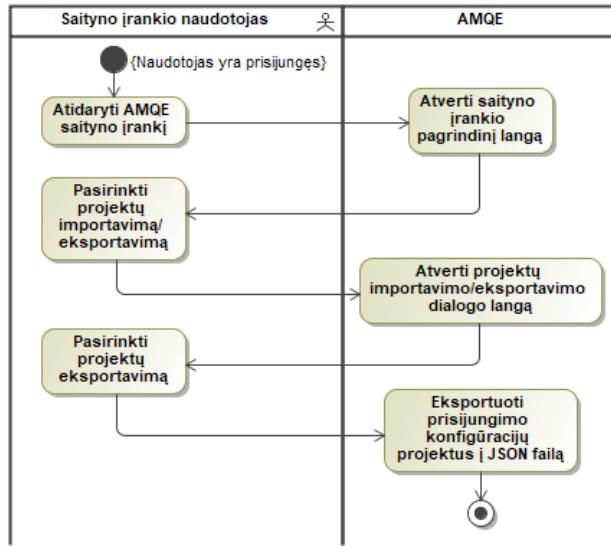
**2.7 pav.** pavaizduota prisijungimo prie pranešimų skirstytojų serverio konfigūracijų projektų importavimo veiklos diagrama.



**2.7 pav.** UML veiklos diagrama: importuoti prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektus

Naudotojas, prisijungęs prie saityno įrankio, atsidaro projektų importavimo/eksportavimo dialogo langą, kuriame pasirenka projektų importavimą. Atsivérus failo pasirinkimo langui, naudotojas pasirenka norimą importuoti projektų failą. Jei failas yra tinkamas (yra visi reikiami duomenys projektų importavimui, failo struktūra teisinga), sukuriami nauji konfigūracijų projektai bei juose esančios konfigūracijos, naudotojas yra informuojamas apie sėkmingai atliktą veiksmą, projektų importavimo/eksportavimo langas yra uždaromas, kitu atveju – pateikiamas klaidos pranešimas.

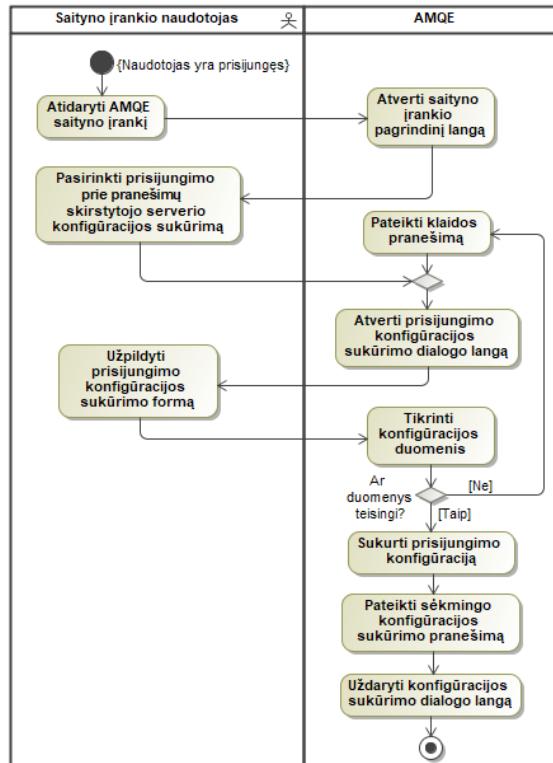
**2.8 pav.** pateikta prisijungimo prie pranešimų skirstytojų serverio konfigūracijų projektų eksportavimo veiklos diagrama.



**2.8 pav.** UML veiklos diagrama: eksportuoti prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektus

Naudotojas, prisijungęs prie saityno įrankio, atsidaro tą patį dialogo langą, kaip ir projektų importavimo atveju, tik Jame pasirenka projektų eksportavimą, o sistema parsunčia į kompiuterį *projects.json* failą, kuriame išsaugoti sukurtų projektų duomenys.

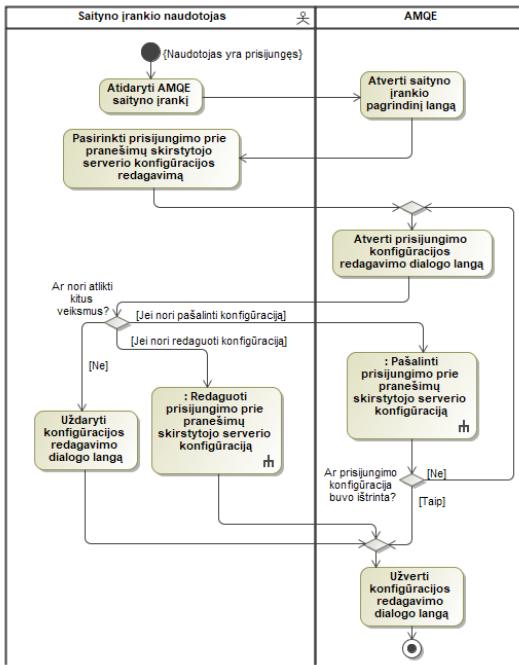
2.9 pav. pavaizduota prisijungimo prie pranešimų skirstytojo serverio konfigūracijos sukūrimo veiklos diagrama.



**2.9 pav.** UML veiklos diagrama: sukurti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

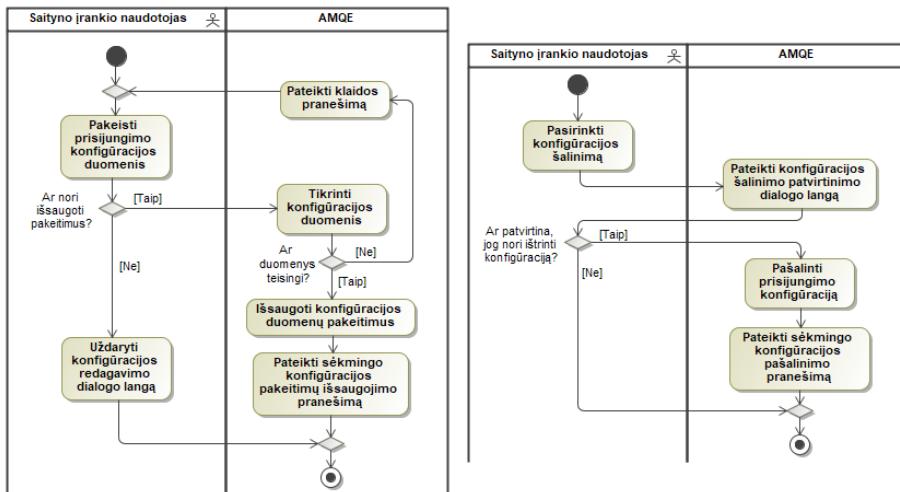
Naudotojas, prisijungęs prie saityno įrankio, atsidaro naujos prisijungimo prie pranešimų skirstytojo serverio konfigūracijos sukūrimo dialogo langą, užpildo konfigūracijos duomenis. Jei duomenys yra teisingi, sukuriama nauja prisijungimo konfigūracija, naudotojui pateikiamas sėkmingo konfigūracijos sukūrimo pranešimas ir konfigūracijos sukūrimo dialogo langas yra uždaromas. Jei duomenys nėra teisingi, naudotojui pateikiamas klaidos pranešimas.

2.10 pav. pateikta prisijungimo prie pranešimų skirstytojo serverio konfigūracijos peržiūros veiklos diagrama.



**2.10 pav.** UML veiklos diagrama: peržiūrėti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Naudotojas, prisijungęs prie saityno įrankio, atsidaro prisijungimo konfigūracijos redagavimo dialogo langą (kuris kartu yra ir peržiūros langas). Atsidaręs langą naudotojas taip pat gali redaguoti arba ištinti peržiūrimą prisijungimo konfigūraciją (šių veiksmų veiklos diagramos pavaizduotos 2.11 pav.). Atlikus veiksmus arba peržiūréjus konfigūraciją, dialogo langas uždaromas.

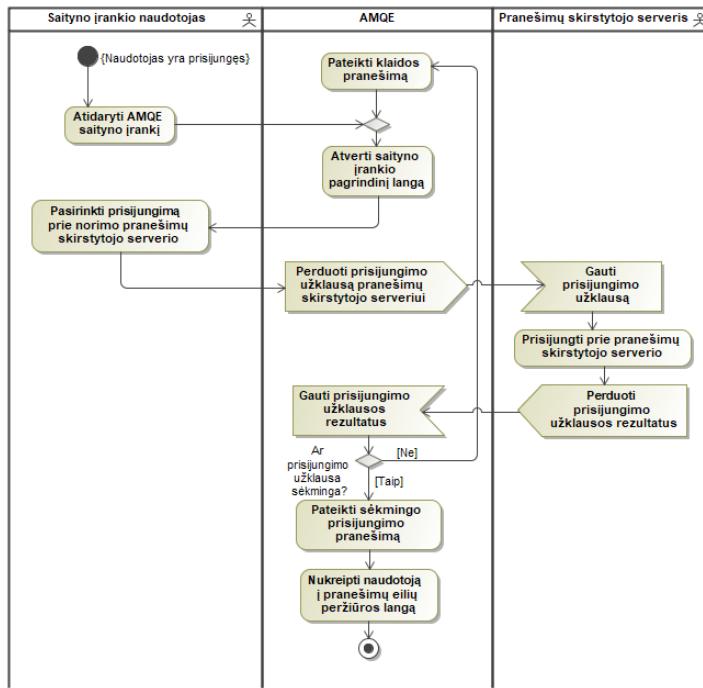


**2.11 pav.** UML veiklos diagramos: redaguoti (kairėje) ir pašalinti (dešinėje) prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Naudotojas, norédamas pakoreguoti konfigūraciją, pakeičia jos duomenis. Norint išsaugoti pakeitimus, pirmiausia yra patikrinami konfigūracijos duomenys ir jei duomenys teisingi, konfigūracija yra atnaujinama, naudotoją informuojant sėkmingo atnaujinimo pranešimu. Jei duomenys nėra teisingi, pateikiamas klaidos pranešimas. Naudotojui nenorint išsaugoti pakeitimų, veiksmų seka yra užbaigiamā uždarant konfigūracijos redagavimo dialogo langą. Konfigūracijos pašalinimo atveju, naudotojui yra pateikiamas patvirtinimo dialogo langas, kuriamo reikia patvirtinti konfigūracijos pašalinimą. Jei naudotojas veiksmą patvirtina, prisijungimo konfigūracija yra pašalinama ir naudotojas informuojamas pranešimu apie sėkmingą jos ištrynimą. Jei pašalinimas nėra patvirtinamas, veiksmų seka yra užbaigiamā.

Prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektų sukūrimo, peržiūros, redagavimo bei šalinimo veiklos diagramų veiksmų sekos yra identiškos atitinkamai pačių prisijungimo konfigūracijų sukūrimo, peržiūros, redagavimo bei šalinimo veiksmų sekoms.

2.12 pav. pavaizduota prisijungimo prie pranešimų skirstytojo serverio veiklos diorama.

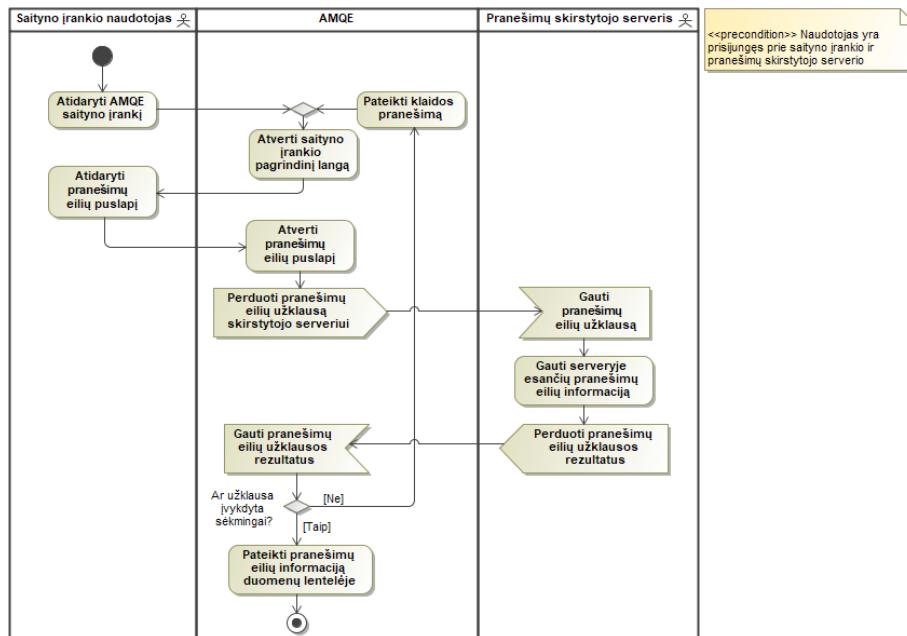


**2.12 pav.** UML veiklos diagrama: prisijungti prie pranešimų skirstytojo serverio

Naudotojas, prisijungęs prie saityno įrankio, pasirenka prisijungimą prie pranešimų skirstytojo serverio. Tada i pranešimų skirstytojo serveri yra siunciama užklausa (signalas), ar su tokiais duomenimis, kokie nurodyti prisijungimo konfigūracijoje, galima prisijungti. Pranešimų skirstytojo serveris perduoda užklauso rezultatus ir jei užklausos rezultatai yra teigiami, naudotojui pateikiamas sėkmindo prisijungimo pranešimas ir jis yra nukreipiamas į pranešimų eilių peržiūros langą. Jei užklausos rezultatai neigiami – naudotojui pateikiamas klaidos pranešimas.

2.13 – 2.21 paveikslėliuose pavaizduotos pranešimų eilių valdymo posistemės veiklos diagramos, kuriomis aprašyti visi šios posistemės panaudos atvejai.

**2.13 pav.** pateikta pranešimų eilių peržiūros veiklos diagrama.

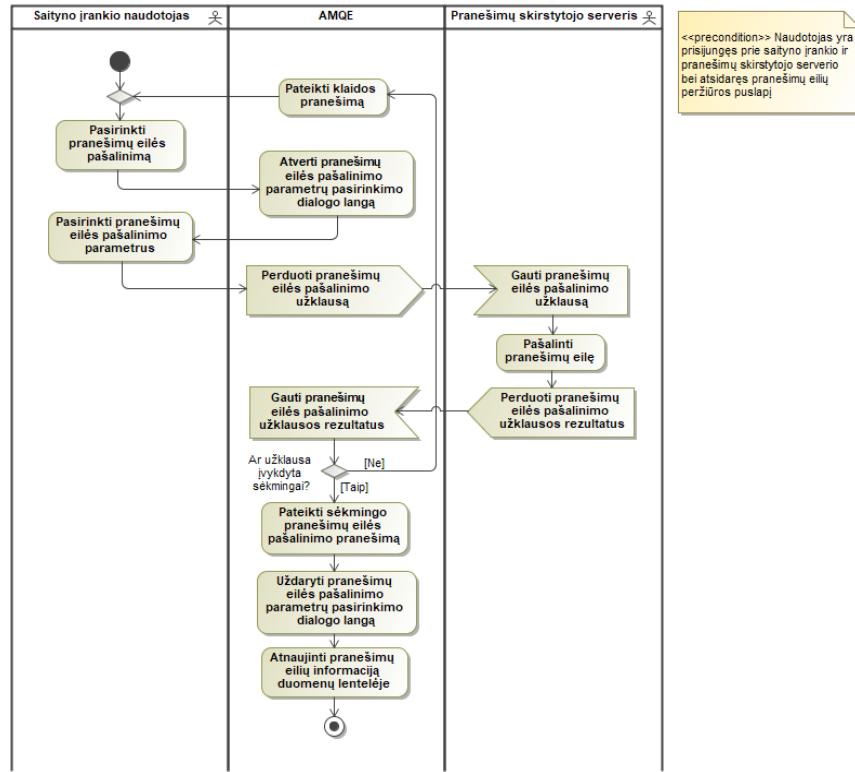


**2.13 pav.** UML veiklos diagrama: peržiūrėti pranešimų eilių sąrašą

Naudotojas, prisijungęs prie saityno įrankio ir pranešimų skirstytojo serverio, atidaro pranešimų eilių langą. Tada i pranešimų skirstytojo serveri yra siunciama užklausa pranešimų eilių informacijai gauti. Pranešimų skirstytojo serveris priima užklausą, gauna servaryje esančią pranešimų

eilių informaciją ir ją perduoda atgal į saityno įrankį. Jei užklausa įvykdyma sėkmingai, gauta informacija pateikiama pranešimų eilių peržiūros puslapyje esančioje duomenų lentelėje, jei ne – naudotojui pateikiamas klaidos pranešimas.

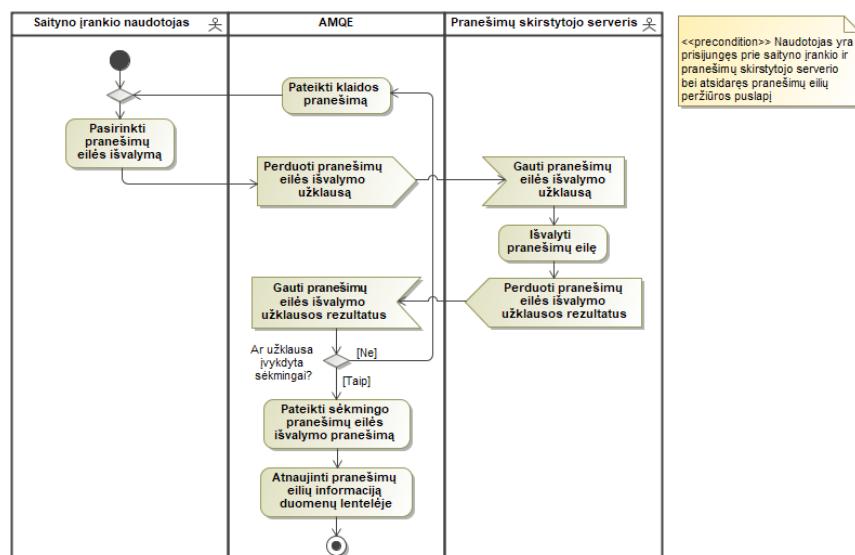
2.14 pav. pavaizduota pranešimų eilės pašalinimo veiklos diagrama.



2.14 pav. UML veiklos diagrama: ištinti pranešimų eilę

Naudotojas, prisijungęs prie saityno įrankio ir pranešimų skirstytojo serverio bei atsidaręs pranešimų eilių peržiūros puslapi, pasirenka eilės pašalinimą. Atidaromas pranešimų eilės pašalinimo parametrų dialogo langas, kuriame naudotojas pasirenka parametrus, kuriais remiantis bus pašalinama pranešimų eilė. Tada pranešimų skirstytojo serverui perduodama pranešimų eilės pašalinimo užklausa, ten ji apdorojama ir įrankiu grąžinamas užklausos įvykdymo rezultatas. Jei užklausa įvykdyma sėkmingai, naudotojui pateikiamas sėkmingai ištintos pranešimų eilės pranešimas, uždaromas eilės pašalinimo parametrų dialogo langas bei atnaujinama pranešimų eilių duomenų lentelė, o kitu atveju – pateikiamas klaidos pranešimas.

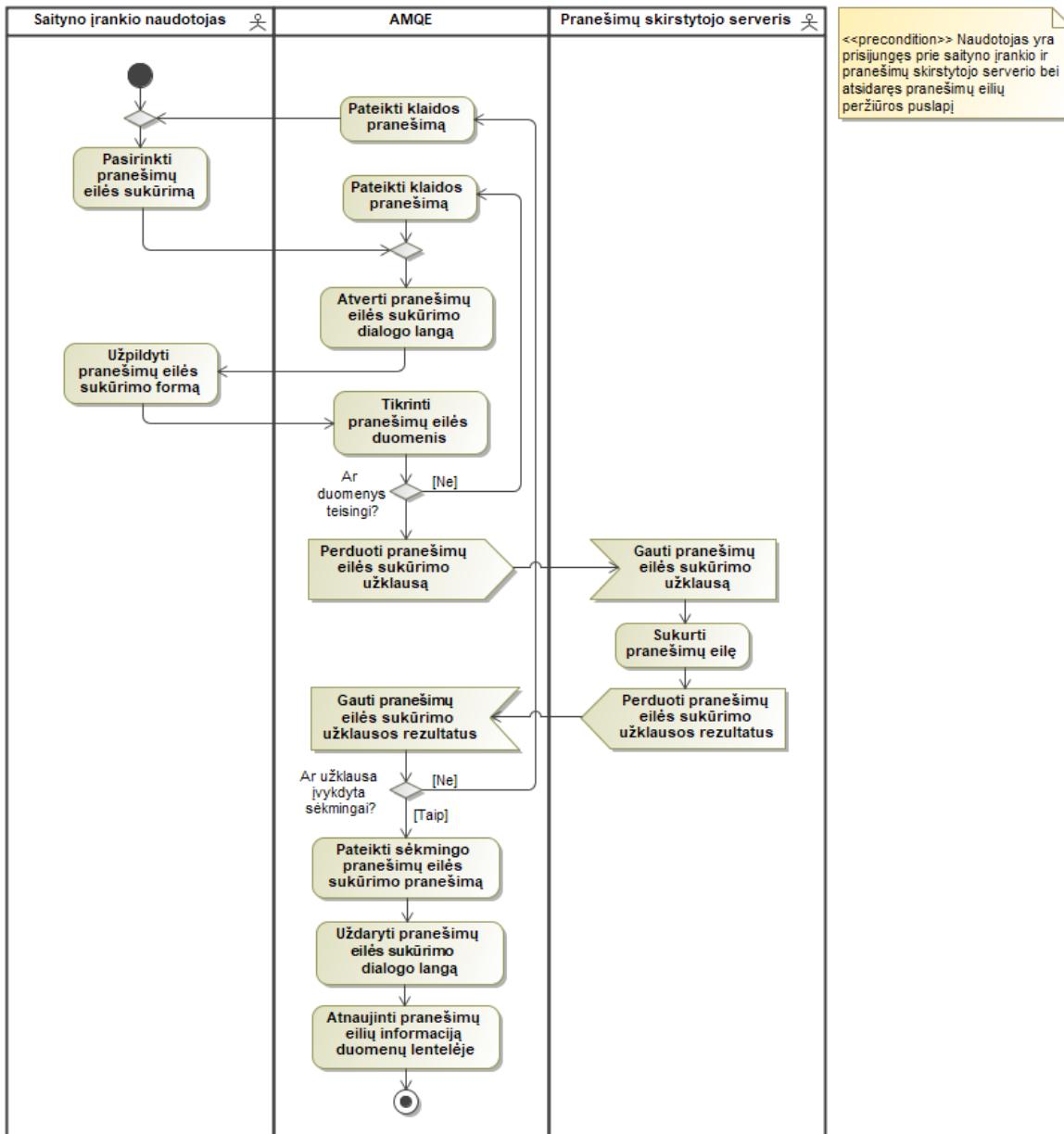
2.15 pav. pateikta pranešimų eilės išvalymo veiklos diagrama.



2.15 pav. UML veiklos diagrama: išvalyti pranešimų eilę

Pranešimų eilės išvalymo veiklos diagramos veiksmai iš esmės sutampa su 2.14 pav. pateiktais pranešimų eilės pašalinimo veiksmais, skiriasi tik tai, kad pranešimų eilės išvalymo atveju naudotojas neturi pasirinkti jokių papildomų parametrų, o pranešimų skirstytojo serveriui yra siunčiama ne pranešimų eilės ištrynimo, bet išvalymo užklausa.

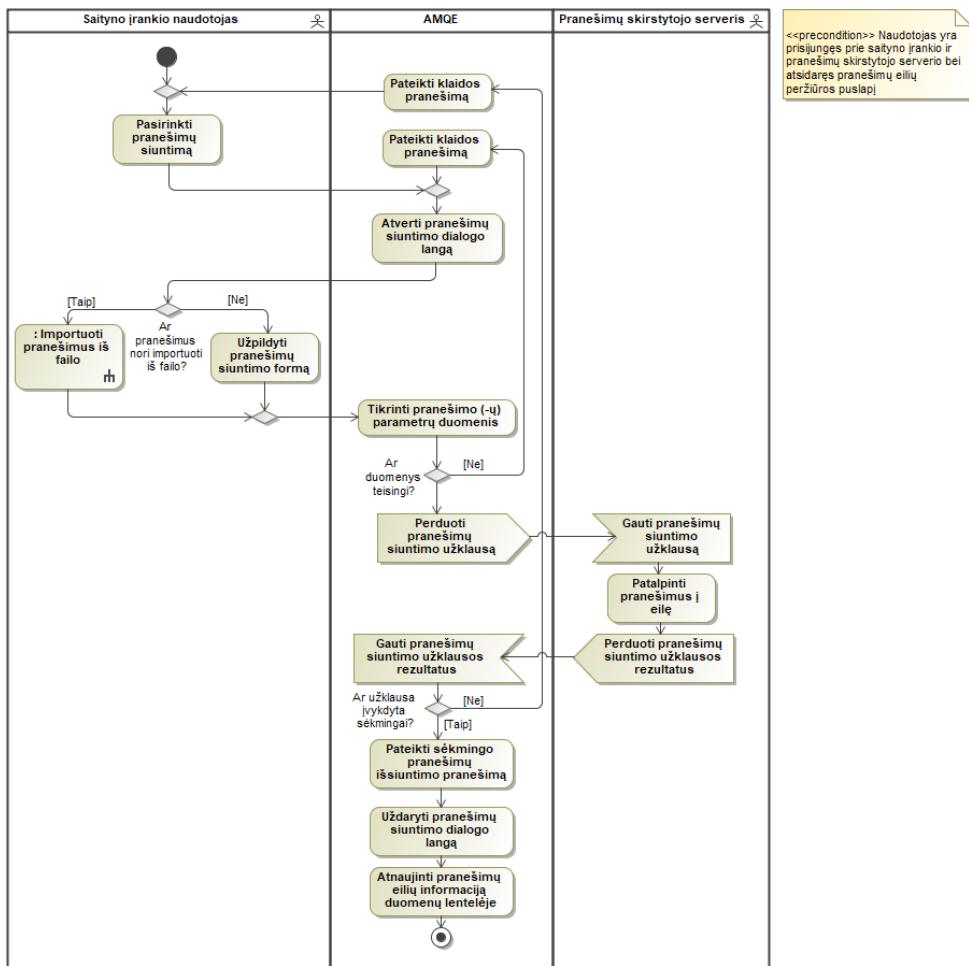
2.16 pav. pateikta naujos pranešimų eilės sukūrimo veiklos diagrama.



2.16 pav. UML veiklos diagrama: sukurti naują pranešimų eilę

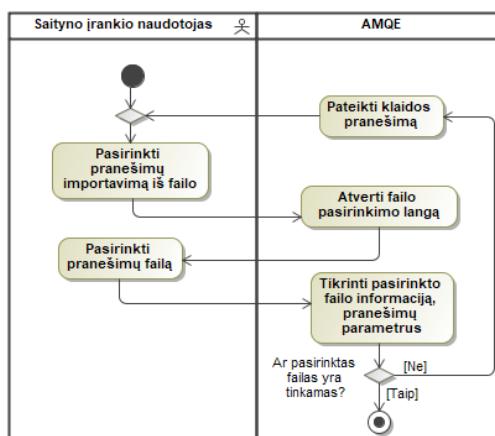
Naudotojas, prisijungęs prie saityno įrankio ir pranešimų skirstytojo serverio bei atsidaręs pranešimų eilių peržiūros puslapij, pasirenka pranešimų eilės sukūrimą. Atsidariusiame pranešimų eilės sukūrimo dialogo lange naudotojas užpilda eilės sukūrimo formą. Jei pranešimų eilės duomenys yra netinkami, pateikiamas klaidos pranešimas, kitu atveju – pranešimų skirstytojo serveriui siunčiama naujos pranešimų eilės sukūrimo užklausa. Serveriui apdorojus užklausą, įrankiui perduodamas užklausos vykdymo rezultatas. Jei užklausos vykdymas nesėkmingas, tada naudotojui pateikiamas klaidos pranešimas, kitu atveju – pateikiamas sėkmingo pranešimų eilės sukūrimo pranešimas, uždaromas pranešimų eilės sukūrimo dialogo langas ir atnaujinama pranešimų eilių informacija duomenų lentelėje.

2.17 pav. pavaizduota pranešimų siuntimo į eilę veiklos diagrama.



**2.17 pav.** UML veiklos diagrama: siusti pranešimą (-us) į eilę

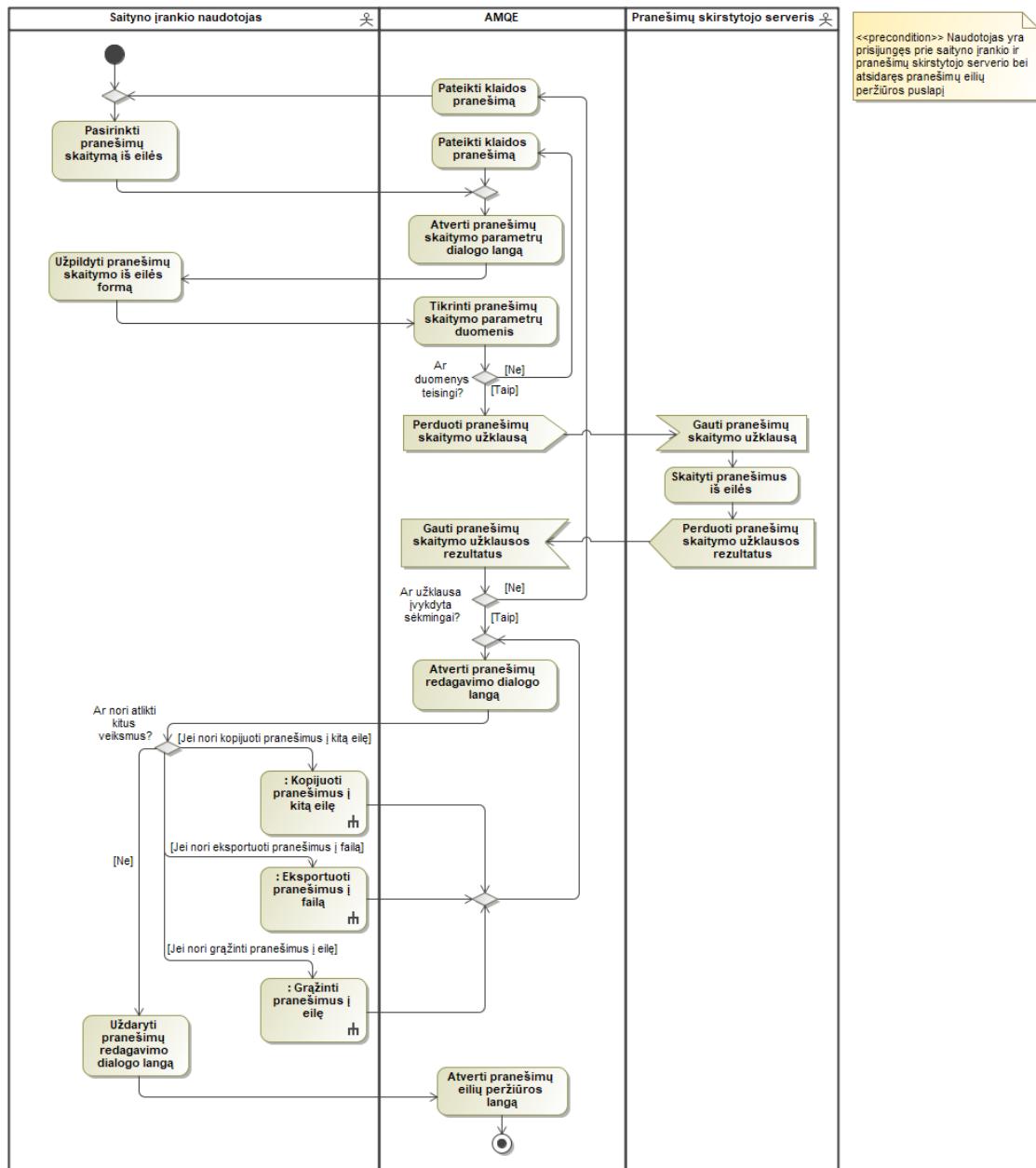
Naudotojas, prisijungęs prie saityno įrankio ir pranešimų skirstytojo serverio bei atsidares pranešimų eilių peržiūros puslapi, pasirenka pranešimų siuntimą. Atsivérusiam pranešimų siuntimo dialogo lange naudotojas pasirenka, ar pranešimus nori importuoti iš failo (pranešimų importavimo iš failo veiklos diagrama pateikta 2.18 pav.), ar sukurti naują pranešimą. Jei naudotojas nori išsiusti naują pranešimą, jam reikia užpildyti pranešimo sukūrimo formą. Jei sukurto arba importuotų pranešimų parametrai yra teisingi, pranešimų skirstytojo serveriui perduodama pranešimo siuntimo užklausa, kurią serveris apdoroja ir grąžina užklausos rezultatus, o jei duomenys neteisingi – pateikiamas klaidos pranešimas. Grąžinus užklausos rezultatą, tikrinama, ar užklausa ivykdyta sekmingai. Jei užklausa nesėkminga – pateikiamas klaidos pranešimas, kitu atveju – pateikiamas sekmingai ivykdyto pranešimų siuntimo pranešimas, uždaromas dialogo langas ir atnaujinama pranešimų eilių informacija duomenų lentelėje.



**2.18 pav.** UML veiklos diagrama: importuoti pranešimus iš failo

2.18 pav. pateiktoje pranešimų importavimo veiklos diagramoje parodyta, kad naudotojas pasirenka pranešimų importavimą, atsivérusiam failo pasirinkimo lange pasirenka norimą importuoti pranešimų failą. Jei failas yra teisingas (failo formatas yra tinkamas, pranešimų informacija korektiška), tada veiksmų seką yra užbaigama ir faile esantys pranešimai yra perduodami tolimesniams jų apdorojimui (parodyta 2.17 pav.), kitu atveju – pateikiamas klaidos pranešimas.

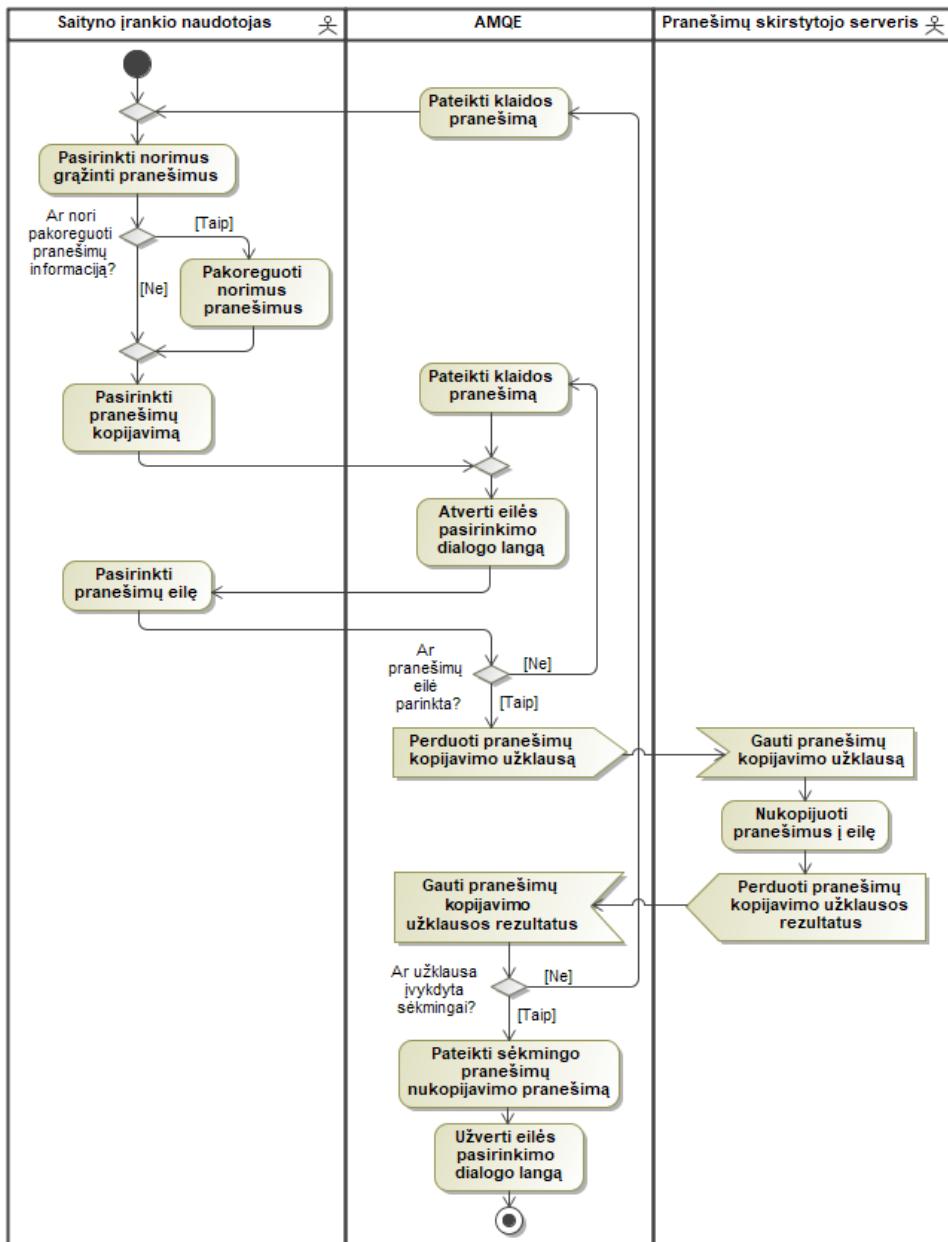
2.19 pav. pateikta pranešimų eilėje esančių pranešimų redagavimo veiklos diagrama.



2.19 pav. UML veiklos diagrama: redaguoti pranešimus eilėje

Naudotojas, prisijungęs prie saityno įrankio ir pranešimų skirstytojo serverio bei atsidaręs pranešimų eilių peržiūros puslapi, pasirenka pranešimų skaitymo iš eilės veiksmą. Atsivérusiam pranešimų skaitymo parametru dialogo lange naudotojas užpilda formos laukus ir jei duomenys teisingi, pranešimų skirstytojo serveriui yra perduodama pranešimų skaitymo užklausa. Pranešimų skirstytojo serveriui apdorojus užklausą, grąžinami jos rezultatai. Jei užklausa nebuvo sėkmingai įvykdymas, pateikiamas klaidos pranešimas, kitu atveju – atveriamas pranešimų redagavimo langas. Šiame lange naudotojas gali pasirinkti norimą veiksmą: kopijuoti pranešimus į kitą eilę, eksportuoti pranešimus į failą arba grąžinti nuskaitytus pranešimus į eilę (šiu veiksmų sekų diagramos pateiktos 2.20 - 2.21 paveikslėliuose). Jei naudotojas jokių veiksmų atliliki nenori, jis gali uždaryti pranešimų redagavimo langą ir atverti pranešimų eilių peržiūros langą.

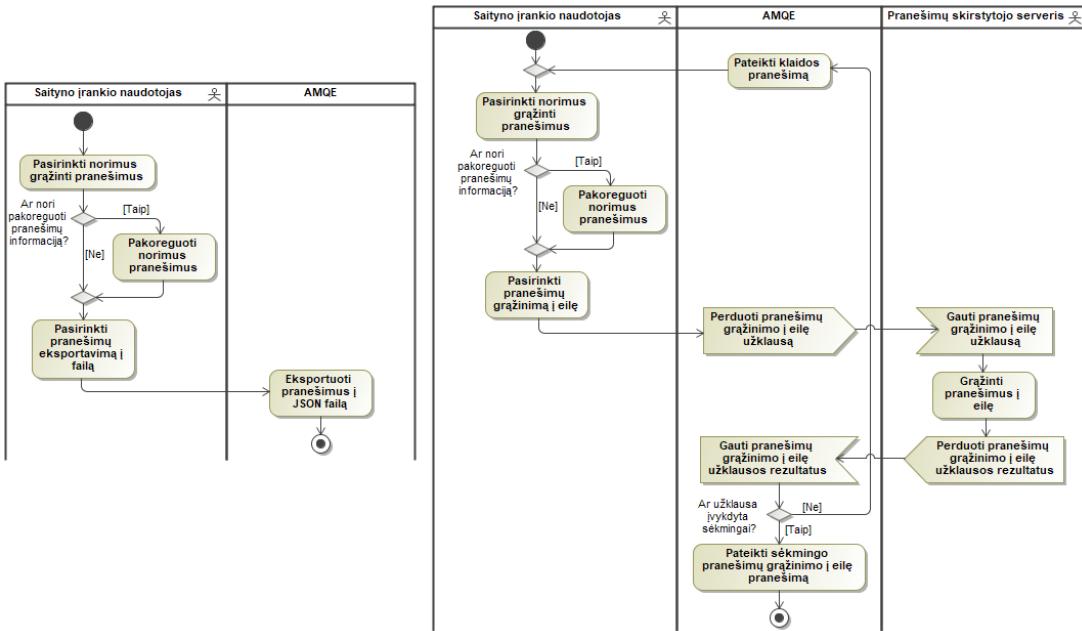
2.20 pav. pateikta pranešimų kopijavimo į kitą eilę veiklos diagrama.



**2.20 pav.** UML veiklos diagrama: kopijuoti pranešimus į kitą eilę

Naudotojas pranešimų redagavimo lange pasirenka norimus kopijuoti pranešimus, jei reikia, pakoreguoja jų informaciją bei pasirenka pranešimų kopijavimą. Atsivérusiamie pranešimų eilės pasirinkimo dialogo lange naudotojas pasirenka pranešimų eilę, į kurią bus nukopijuoti pranešimai. Jei eilė nėra pasirinkta, pranešimai negali būti nukopijuoti ir naudotojui pateikiamas klaidos pranešimas. Jei eilė pasirinkta, pranešimų skirstytojo serveriui siunčiama pranešimų kopijavimo užklausa, kurią apdorojus, grąžinamas užklausos rezultatas. Jei užklausa nebuvo įvykdyta, naudotojui pateikiamas klaidos pranešimas, kitu atveju – pateikiamas sėkmingo pranešimu nukopijavimo pranešimas ir uždaromas eilės pasirinkimo dialogo langas.

**2.21 pav.** pateiktos pranešimų eksportavimo į failą bei grąžinimo atgal į eilę veiklos diagramos.



**2.21 pav.** UML veiklos diagramos: eksportuoti pranešimus į failą (kairėje) ir grąžinti pranešimus į eilę (dešinėje)

Pranešimų eksportavimo į failą atveju, matomu 2.21 pav. kairėje pusėje, naudotojas pasirenka norimus eksportuoti pranešimus, jei reikia, pakoreguoja jų informaciją ir pasirenka pranešimų eksportavimą. Sistema pasirinktus pranešimus eksportuoja į *messages.json* failą.

2.21 pav. dešinėje pusėje pavaizduotoje pranešimų grąžinimo atgal į eilę veiklos diagramoje matoma, kad taip pat, kaip ir pranešimų eksportavimo atveju, naudotojas pasirenka norimus grąžinti pranešimus, jei reikia, pakoreguoja jų informaciją ir pasirenka pranešimų grąžinimą į eilę. Pranešimų skirstytojo serveriui perduodama pranešimų grąžinimo į eilę užklausa, kurią apdorojus, įrankiui grąžinamas užklausos rezultatas. Jei užklausa nebuvo įvykdyta sėkmingai, pateikiamas klaidos pranešimas, kitu atveju – pateikiamas sėkmingo pranešimų grąžinimo atgal į eilę pranešimas.

### 2.1.3. Apribojimai

Tiek kuriamo įrankio serverio pusės (naudotojų ir pranešimų eilių valdymo posistemui) dalį, tiek naudotojo sąsajos projekto diegimui pasirinktos AWS (*Amazon Web Services*) paslaugos. Pasirinkta įrankio DBVS – *PostgreSQL v10.3*, o naudojamos įrankis šios duomenų bazės valdymui – *pgAdmin 4 v2.1*.

Įrankio serverio pusės dalis programuojama *Visual Studio* integravotoje programavimo aplinkoje, *Windows* operacinėje sistemoje, o naudotojo sąsajos (saityno programos) programavimui pasirinktas atviro kodo teksto redaktorius *Atom v1.25.0*.

Kuriamo įrankio testavimui naudojami lokalūs HTTP serveriai: posistemui – *IIS (Internet Information Services)* serveris, o naudotojo sąsajos testavimui – *Node.js* serveris.

Pranešimų eilių valdymo įrankio kūrimo terminai:

- 1) Iki 2018-03-11: projekto reikalavimų bei rinkos analizė, programavimo aplinkos paruošimas;
- 2) Iki 2018-04-08: naudotojų ir pranešimų eilių valdymo posistemui programavimas;
- 3) Iki 2018-04-22: saityno įrankio naudotojo sąsajos programavimas;
- 4) Iki 2018-05-04: serverio pusės komponentų ir naudotojo sąsajos testavimas bei diegimas.

Finansinių apribojimų projekto kūrimui nėra, nes biudžetas nėra numatytas.

### 2.1.4. Vartotojo sąsajos specifikacija

Prieš realizuojant naudotojo sąsają buvo kuriami naudotojo sąsajos langų prototipų eskizai. Šių eskizų kūrimui naudota programinė įranga – *Balsamiq Mockups 3 v3.5.15*. Eskizai buvo kuriami ne visiems, o tik pagrindiniams įrankio langams – kiti langai buvo kuriami atitinkamai pateiktiems

langų eskizams, tik pakeičiant lange esančius elementus, pvz., skiriasi tik modalinių dialogo langų formų laukai bei jų pavadinimai, tačiau išvaizda išlieka ta pati.

2.22 - 2.23 paveikslėliuose pateikiti prisijungimo bei registracijos langų prototipai:

AMQE

Login

E-mail

Password

[Not registered?](#)

2.22 pav. Prisijungimo langas

AMQE

Register

E-mail

Password

Re-entry password

[Already registered?](#)

2.23 pav. Registracijos langas

Kaip matome iš eskizų, pastaruosiuose languose turi būti nuorodos (eskizuose pavaizduotos mėlynu tekstu), leidžiančios nesunkiai pereiti iš prisijungimo lango į registracijos ir atvirkščiai.

2.24 pav. pavaizduotas pagrindinis įrankio langų šablonas, kuris bus naudojamas visuose languose.

AMQE

Project 1

Project 2

Broker 2

Broker 3

Project 3

Project 4

Edit project

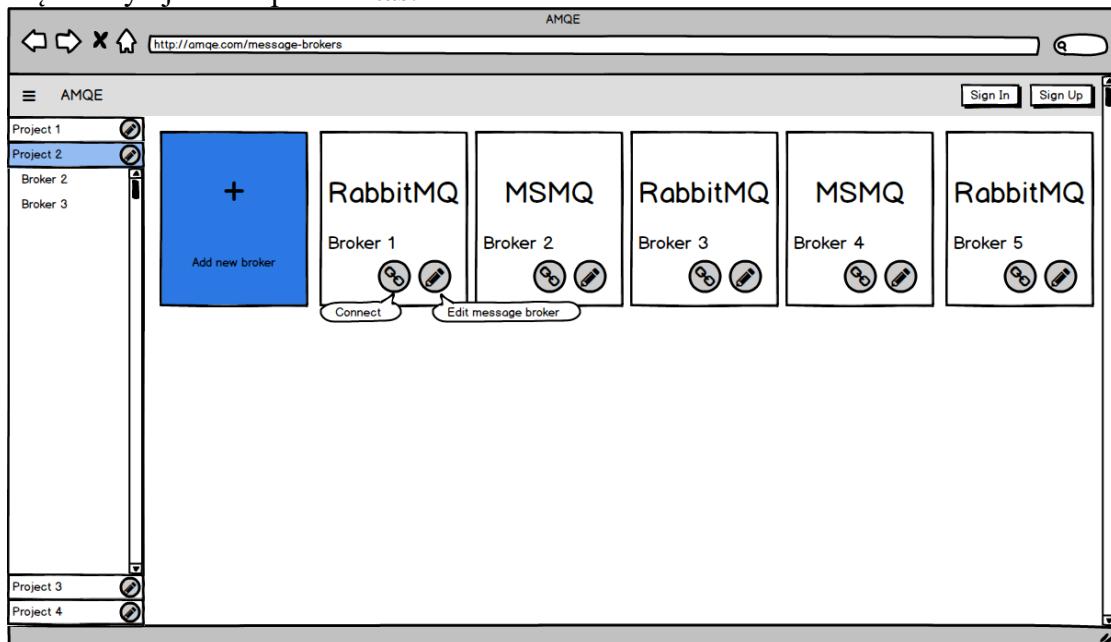
Sign In Sign Up

2.24 pav. Pagrindinis įrankio šablonas

2.24 pav. galima matyti, kad įrankio šabloną sudaro 3 pagrindiniai elementai:

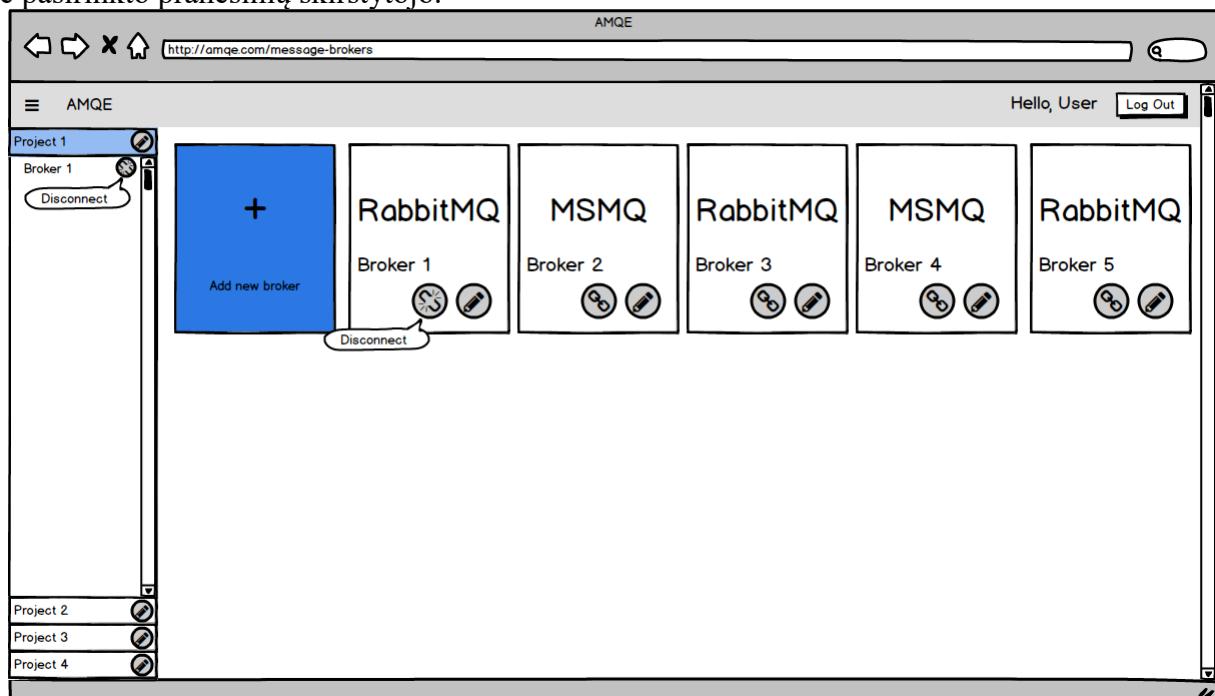
- 1) Priemonių juosta viršuje su prisijungimo bei registracijos mygtukais (arba atsijungimo mygtuku, jei naudotojas yra prisijungęs) dešinėje bei šoninio projektų meniu suskleidimo mygtuku kairėje pusėje;
- 2) Projektų pasirinkimo meniu kairėje pusėje;
- 3) Pagrindinis langas, kuriamo bus pateikiamas pasirinkto puslapio vaizdas.

2.25 - 2.26 paveikslėliuose pavaizduotas pranešimų skirstymo programinių įrangų konfigūracijų langas dviem atvejais: kai yra prisijungta prie konkretaus pranešimų skirstytojo ir kai pranešimų skirstytojas nėra pasirinktas.



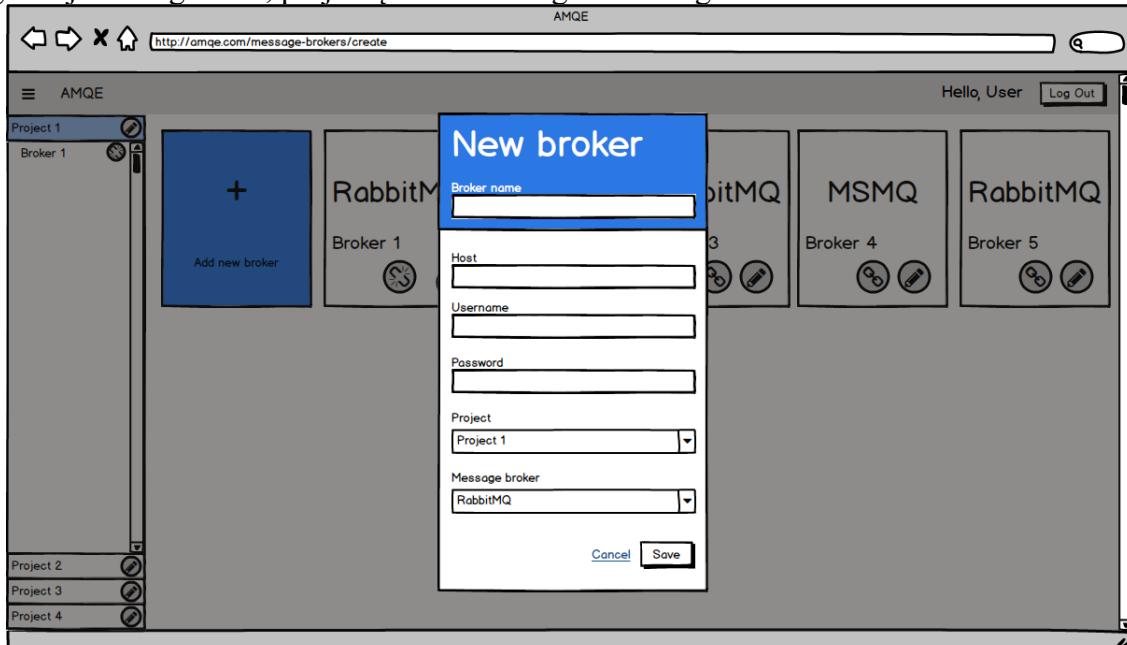
2.25 pav. Pranešimų skirstytojų konfigūracijų langas (nėra prisijungta prie konkretaus skirstytojo)

2.25 pav. matoma, kad pranešimų skirstytojų konfigūracijos naudotojams yra pateikiamos kortelėmis su mygtukais, kurie leidžia redaguoti konkretios konfigūracijos informaciją arba prisijungti prie pasirinkto pranešimų skirstytojo.



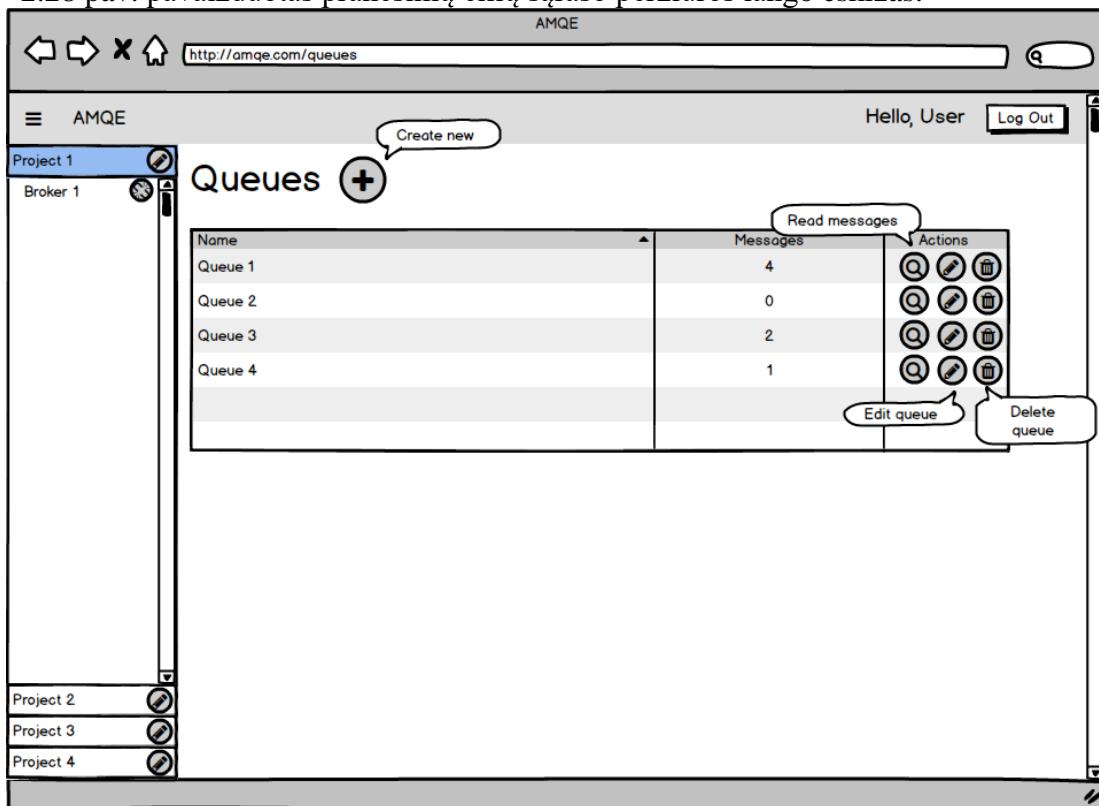
2.26 pav. Pranešimų skirstytojų konfigūracijų langas (yra pasirinktas konkretus skirstytojas)

2.27 pav. pateiktas modalinio dialogo lango pavyzdys, kaip turi atrodyti naujos pranešimų skirstytojo prisijungimo konfigūracijos sukūrimas. Remiantis šiuo eskizu, taip pat sukurti konfigūracijos redagavimo, projektų kūrimo/redagavimo langai.



**2.27 pav.** Pranešimų skirstytojų prisijungimo konfigūracijos sukūrimo modalinis dialogo langas

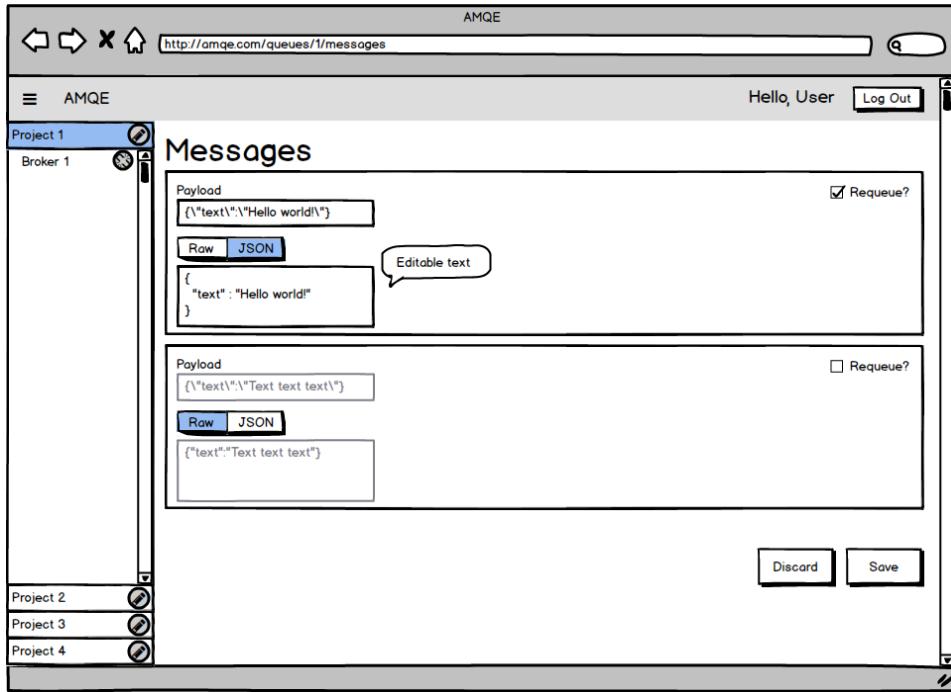
2.28 pav. pavaizduotas pranešimų eilių sąrašo peržiūros lango eskizas.



**2.28 pav.** Pranešimų eilių sąrašo peržiūros langas

Pranešimų eilių peržiūros lange eilės yra pateikiamos duomenų lentelės forma. Taip pat šiame lange galima pasirinkti naujos pranešimų eilės sukūrimo mygtuką arba vieną iš pranešimų eilės valdymo veiksnių: pranešimų peržiūrą, eilės parametrų redagavimą arba eilės pašalinimą.

2.29 pav. pateiktas pranešimų redagavimo/peržiūros langas.



**2.29 pav.** Pranešimų peržiūros/redagavimo langas

Matoma, kad pranešimų peržiūrai ir redagavimui yra naudojamas tas pats langas. Kiekvienas pranešimas yra vaizduojamas atskiroje kortelėje, kurioje galima matyti originalų žinutės tekštą arba suformatuotą ir pateiktą, pvz., JSON formatu. Taip pat kortelėse galima pasirinkti, ar atitinkamas pranešimas turi būti grąžintas į eilę. Peržiūrėjus/pakeitus pranešimus, reikia pasirinkti vieną iš galimų veiksnių: išsaugoti pakeitimus ir grąžinti pasirinktus pranešimus į eilę arba pakeitimus atmesti ir pranešimų atgal į eilę nebegrąžinti.

### 2.1.5. Realizacijai keliами reikalavimai.

Kuriamam pranešimų eilių valdymo įrankiui iškelti tokie nefunkciniai reikalavimai:

1. Realizavimui:
  - a. Įrankiu turi būti galima valdyti bent šių *AMQP* protokolo realizacijų pranešimų eiles: *RabbitMQ* ir *ActiveMQ*;
  - b. Naudotojų ir pranešimų eilių valdymo posistemui API metodų kodo padengimas automatiniais testais turi siekti bent 90%.
2. Sistemos priežiūrai:
  - a. Įrankio programos kodo dalijimuisi ir kodo istorijai saugoti turi būti naudojama versijų kontrolės sistema *Git*;
  - b. Įrankis turi turėti testavimo ir produkčijos aplinkas tiek saityno programos, tiek serverio pusės dalies kodo talpinimui bei vykdymui;
  - c. Įrankio diegimui į testavimo bei produkčijos aplinkas turi būti paruošti automatinio diegimo scenarijai.
3. Saugumui:
  - a. Visi pranešimų eilių valdymo posistemės API metodai turi būti prieinami tik autentikuotiems naudotojams;
  - b. Kliento – serverio komunikacijos apsauga turi būti pagrįsta JWT žetona;
  - c. Konfigūracijos duomenys, skirti prisijungti prie pranešimų eilių skirstymo programinės įrangos, duomenų bazėje turi būti saugomi prieš tai juos užšifruojant.
4. Kultūriniai-politiniai:
  - a. Įrankio saityno programos naudotojo sėsaja turi būti pateikiama anglų kalba.

## **2.1.6. Techninė specifikacija.**

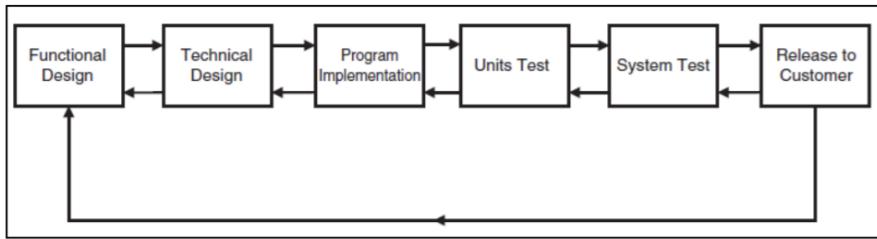
Kaip pateikta 2.1.5 skyriuje (sistemos priežiūrai keliamuose nefunkciniuose reikalavimuose), įrankio talpinimui bus naudojamos dvi aplinkos – testavimo ir produkcijos. Tieka saityno programa, tiek įrankio serverio pusės dalis diegama naudojant AWS debesų kompiuterijos paslaugas. Detalesnė kiekvienos aplinkos specifikacija:

- a. Testavimo aplinka:
  1. Įrankio posistemų serveris:
    - 1.1. AWS EC2 (virtualių serverių paslauga) – virtualaus serverio, kuriame talpinami projekto failai, sukūrimui;
    - 1.2. Serverio parametrai: t2.micro virtualaus serverio tipas – 1 virtualus procesorius, 1 GB operatyviosios atminties, disko talpa - 30 GB;
    - 1.3. ActiveMQ ir RabbitMQ serveriai.
  2. Duomenų bazės serveris:
    - 2.1. AWS RDS (virtualus duomenų bazės serveris) – įrankio duomenų bazės talpinimui;
    - 2.2. Serverio parametrai: t2.db.micro virtualaus serverio tipas - 1 virtualus procesorius, 1 GB operatyviosios atminties, disko talpa – 20 GB;
    - 2.3. Naudojama PostgreSQL DBVS.
  3. Saityno programos talpykla:
    - 3.1. AWS S3 (failų saugykla) – patalpinus sukompliuotus projekto failus, paslauga suteikia nuorodą, kuria naudojantis galima prieiti prie patalintų failų, mūsų atveju, pačios saityno programos – papildomas diegimas į serverį nėra reikalingas.
- b. Produkcijos aplinka:
  1. Įrankio posistemų serveriai:
    - 1.1. AWS EC2 (virtualių serverių paslauga) – virtualių serverių, kuriuose talpinami projekto failai, sukūrimui;
    - 1.2. Dvi serverių kopijos, veikiančios vienu metu. Apkrova tarp jų paskirstoma naudojant AWS ELB (apkrovos balansavimo serviso) paslaugą;
    - 1.3. AWS Route53 (DNS valdymas) – domeno nukreipimui į apkrovos balansavimo servisą;
    - 1.4. Serverių parametrai: t2.micro virtualaus serverio tipas – 1 virtualus procesorius, 1 GB operatyviosios atminties, disko talpa - 30 GB.
    - 1.5. ActiveMQ ir RabbitMQ serveriai.
  2. Duomenų bazės serveris:
    - 2.1. AWS RDS (virtualus duomenų bazės serveris) – įrankio duomenų bazės talpinimui;
    - 2.2. Serverio parametrai: t2.db.micro virtualaus serverio tipas - 1 virtualus procesorius, 1 GB operatyviosios atminties, disko talpa – 20 GB;
    - 2.3. Naudojama PostgreSQL DBVS;
  3. Saityno programos talpykla:
    - 3.1. AWS S3 (failų saugykla) – saityno programos failų saugojimui;
    - 3.2. AWS Cloudfront (CDN) – HTTP užklausų nukreipimui į HTTPS bei failų kopijavimui tarp fizinių AWS serverių įvairiuose pasaulio regionuose;
    - 3.3. AWS Route53 (DNS valdymas) – domeno nukreipimui į saityno programą;
    - 3.4. AWS Certificate Manager - SSL sertifikato gavimui bei HTTPS protokolo užtikrinimui.

## **2.2. Projektavimo metodai**

### **2.2.1. Projektavimo valdymas ir eiga**

Projektas kuriamas taikant iteracinių projektavimo modelį (2.30 pav.).



**2.30 pav.** Iteracinio projektavimo schema [7]

Šis projektavimo modelis pasirinktas dėl to, kad projekto kūrimo nėra būtina pradėti nuo visų reikalavimų apibrėžimo, projekto specifikacijos – reikalavimai gali būti keičiami bei papildomi kiekvienos iteracijos metu. Taip pat taikant iteracinių modelių projektas kuriamas palaipsniui, todėl galima lengviau pastebėti programos defektus, blogus projekto architektūros sprendimus ankstyvosiose kūrimo fazėse, daugiau laiko yra skiriama pačio projekto projektavimui, kūrimui, testavimui nei jo dokumentacijai.

Projekto kūrimo eigoje iteracijos buvo taikomos atskirų AMQE įrankių sudarančiu posistemų/komponentų kūrimui, bendriems įrankio projektavimo darbams. Kiekvienna iteracija truko po 2 savaites, o bendras atlirkų iteracijų skaičius – 7 (pradedant 2018-02-01 ir baigiant 2018-05-04). Buvo atlirktos tokios įrankio projektavimo bei kūrimo iteracijos:

- 1) AMQP protokolo bei komunikacijos pranešimais sprendimų analizė;
- 2) Įrankio posistemų projektų konfigūracija bei paruošimas, duomenų bazės projektavimas bei realizavimas;
- 3) Naudotojų posistemės projektavimas, kūrimas bei testavimas;
- 4) Įrankio pranešimų eilių valdymo posistemės projektavimas, kūrimas bei testavimas (RabbitMQ integracija);
- 5) Įrankio pranešimų eilių valdymo posistemės projektavimas, kūrimas bei testavimas (ActiveMQ integracija);
- 6) Įrankio naudotojo sąsajos (saityno programos) projektavimas bei kūrimas;
- 7) Baigiamieji darbai: įrankio posistemų ir naudotojo sąsajos projekto diegimas, kodo optimizavimas.

## 2.2.2. Projektavimo technologija

Įrankio projekto modeliams kurti buvo naudojamas standartinės modeliavimo kalbos – *UML* – elementų poaibis: panaudos atvejų, klasių, paketų, veiklų, sekų bei diegimo diagramos. Šių tipų diagramų kūrimui naudojamas *Magic Draw 18.5* modeliavimo įrankis.

## 2.2.3. Programavimo kalbos, derinimo, automatizavimo priemonės, operacinės sistemos

Įrankio naudotojų bei pranešimų eilių valdymo posistemės suprogramuotos naudojant *C#* programavimo kalbą (7.1 versiją) bei *ASP.NET Core 2.0* karkasą. Programavimas bei posistemų testavimas buvo atliekamas *Microsoft Visual Studio 2017* integruotoje programavimo aplinkoje, kartu naudojant *ReSharper* įskiepi. Naudota *Microsoft Windows 10* operacinė sistema. Lokaliam posistemui testavimui naudotas *IIS 10 Express* HTTP servisas. Duomenų bazė valdymo sistema – *PostgreSQL* (10.1 versija), naudotas įrankis duomenų bazės valdymui ir koregavimui – *pgAdmin 4* (2.1 versija). Posistemių testavimo aplinkos operacinė sistema – *Microsoft Windows Server 2016 Base* (naudojamos *AWS EC2* paslaugos), serveryje veikia *IIS 10.0* HTTP servisas, o duomenų bazė patalpinta atskirame serveryje, naudojant *AWS RDS* debesų kompiuterijos paslaugas.

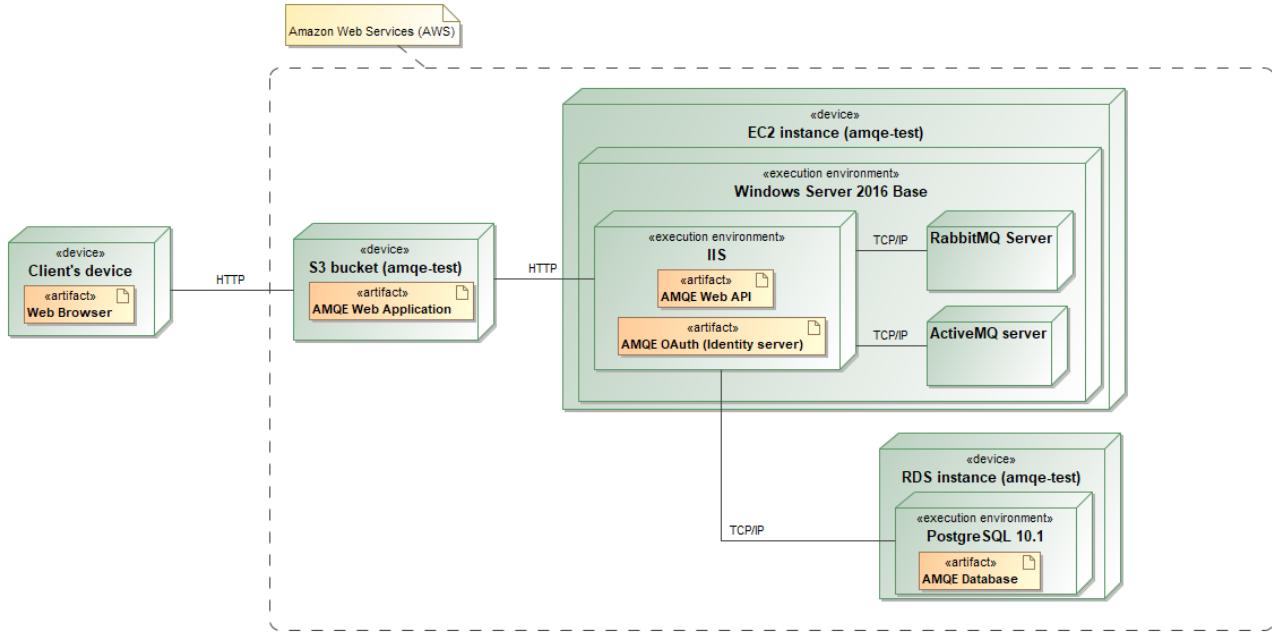
Įrankio saityno sąsajos programavimui naudojama *JavaScript* programavimo kalba (*ECMAScript 6* versija) bei *HTML 5* žymėjimo ir *CSS 3* (kartu su *Stylus* preprocesoriumi) stilių aprašymo kalbos. Taip pat naudojamas *JavaScript* programavimo kalbos karkasas *Vue.js*. Programavimo bei testavimo darbai buvo vykdomi naudojant lokalų *Node.js* serverį (5.8 versiją), *Atom* teksto redaktorių. Kadangi saityno programos kodas prieš diegimą yra sukompiliuojamas į statinius *.html*, *.css* bei *.js* tipų failus, todėl tokio tipo svetainės talpinimui (testavimo aplinkoje) pasirinkta naudoti *AWS S3* debesų kompiuterijos paslaugas.

Įrankio serverio pusės dalies ir saityno programos diegimas į testavimo bei produkcijos aplinkas yra automatizuotas, naudojant saityno įrankį *Buddy*. Plačiau apie ši įrankį bei automatizuotą AMQE diegimo procesą aprašyta diegimo vadove, 4.3 skyriuje.

## 2.3. Sistemos projektas

### 2.3.1. Statinis sistemos vaizdas

2.31 - 2.32 paveikslėliuose pateikiamos *UML* diegimo diagramos, vaizduojančios fizinių sistemų išdėstyti testavimo ir produkcijos aplinkose.

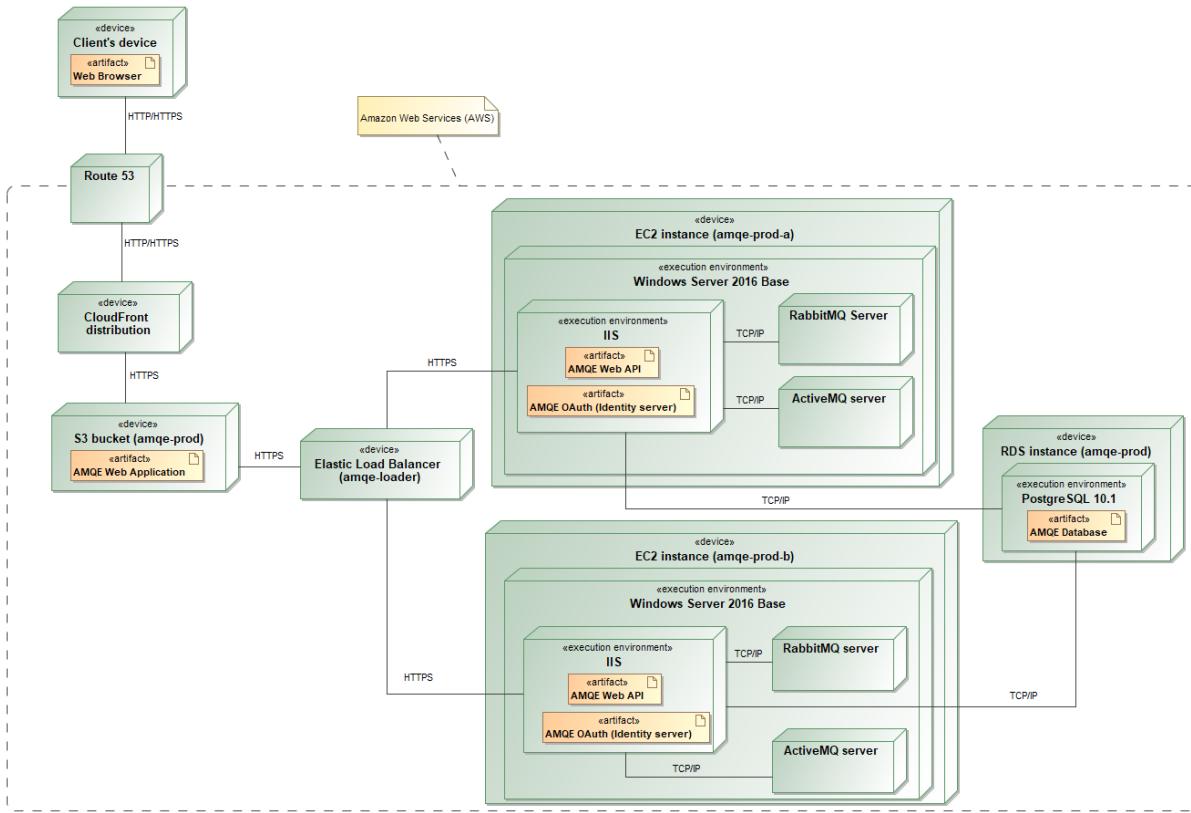


**2.31 pav.** *UML* diegimo diagrama: testavimo aplinkos fizinius išdėstymus

Testavimo aplinkoje AMQE įrankio infrastruktūra yra padalinta į 3 atskirus serverius:

- 1) S3 bucket (amqe-test): naudojama AWS *S3* paslauga, kurios sukurtame konteineryje (pavadinimu *amqe-test*) talpinami įrankio saityno programos failai;
- 2) EC2 instance (amqe-test): naudojama AWS *EC2* paslauga, kuria naudojantis sukurtas virtualus serveris (pavadinimu *amqe-test*), kuriame talpinami įrankio posistemų failai;
- 3) RDS instance (amqe-test): naudojama AWS *RDS* paslauga, kuria naudojantis sukurtas duomenų bazės serveris (pavadinimu *amqe-test*), kuriame talpinama AMQE duomenų bazė;

Taip pat testavimo aplinkoje veikia papildomi *RabbitMQ* bei *ActiveMQ* pranešimų skirstytojų serveriai, naudojami šių realizacijų pranešimų eilėms saugoti bei valdyti.

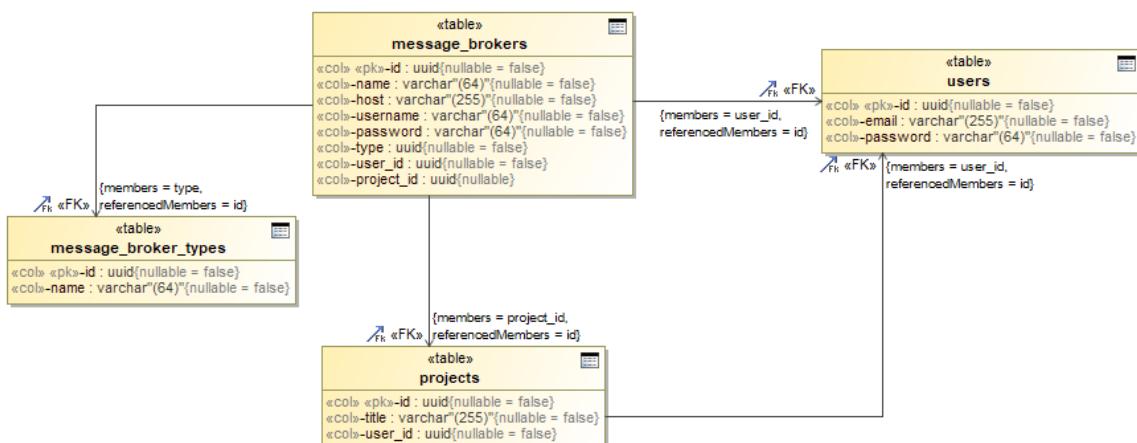


**2.32 pav.** UML diegimo diagramma: produkcijos aplinkos fizinių išdėstymas

Įrankio infrastruktūra testavimo ir produkcijos aplinkose turi panašumą ir skirtumą. Kaip ir testavimo aplinkoje, produkcijos aplinkoje įrankio saityno programa yra talpinama AWS S3 konteineryje (pavadinimu *amqe-prod*), duomenų bazę patalpinta AWS RDS paslaugos sukurtame serveryje (pavadinimu *amqe-prod*), veikia tokie pat pranešimų skirstytojų serveriai.

Apžvelgiant skirtumus, produkcijos aplinkoje veikia dvi vienodos AWS EC2 paslaugos sukurtų virtualių serverių kopijos (*amqe-prod-a* ir *amqe-prod-b*), talpinančios įrankio posistemų failus. Apkrovą tarp šių serverių valdo AWS *Elastic Load Balancer* paslauga (sukurtas konkretus servisas *amqe-loader*). Taip pat naudojamas AWS *Route 53* servisas, nukreipiantis DNS užklausas į reikiamus AWS servisus. AWS *CloudFront* paslauga naudojama dėl dviejų priežasčių: pirmoji – HTTP užklausų peradresavimas į HTTPS protokolą, o kita – įrankio saityno programos failų paskirstymas/kopijavimas tarp fizinių AWS serverių skirtinguose pasaulyje regionuose, taip sutrumpinant užklausų siuntimo bei jų atsakymo gavimo laiką (saityno programos naudotojų užklausos yra nukreipiamos į arčiausiai jų esantį fizinį serverį).

Naudotojų ir pranešimų eilių valdymo posistemės naudoja tą pačią duomenų bazę, kurios schema pateikta 2.33 pav.

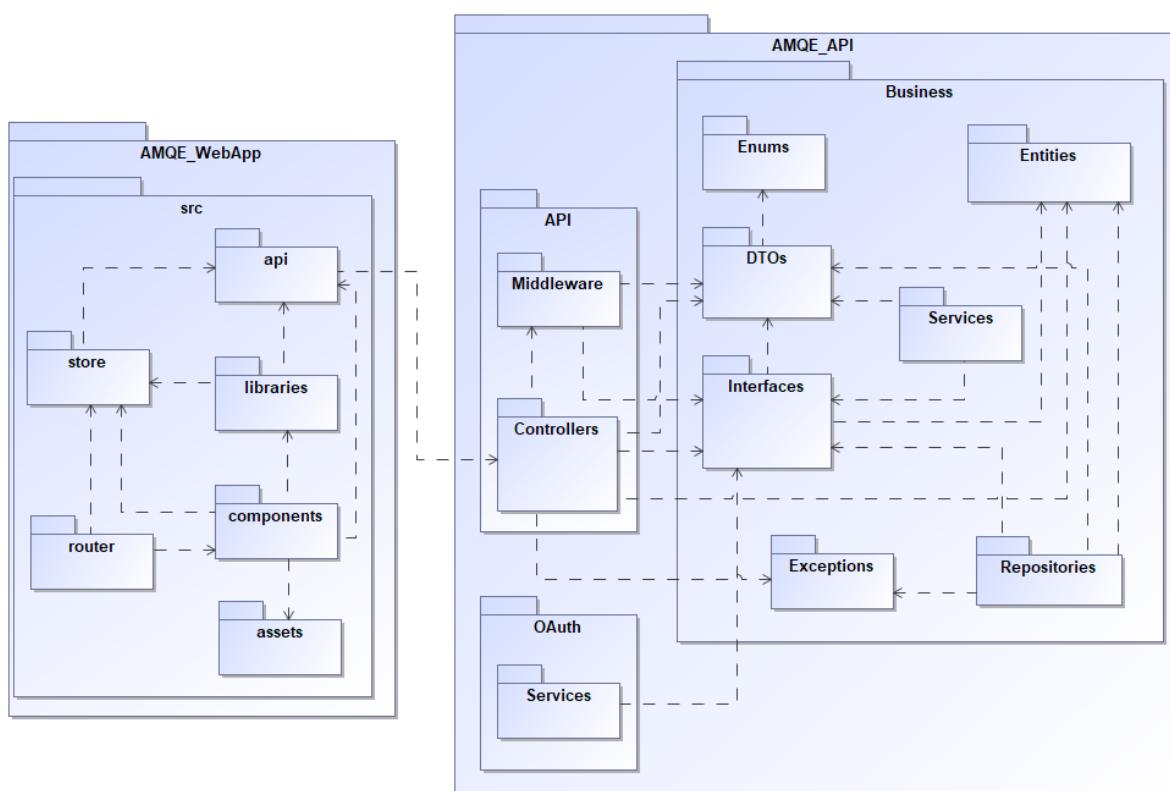


**2.33 pav.** Įrankio posistemui naudojamos duomenų bazės schema

AMQE duomenų bazę sudaro 4 lentelės. Lentelėje *users* saugomi įrankio naudotojus identifikuojantys duomenys: el. paštas bei slaptažodis. Kiekvienas sistemos naudotojas gali sukurti kelis projektus. Projekto informacija (projekto pavadinimas bei naudotojo identifikatorius) talpinama *projects* lentelėje. Taip pat naudotojai gali kurti pranešimų skirstytojų konfigūracijas – duomenis, reikalingus prisijungti prie pranešimų skirstytojų serverių. Šie duomenys yra saugomi *message\_brokers* lentelėje: pranešimų skirstytojo serverio pavadinimas, tipas, serverio adresas, prisijungimo prie serverio vardas bei slaptažodis, naudotojo, kuriam priklauso konfigūracija, identifikatorius ir projekto identifikatorius, nurodantis, kuriam projektui konfigūracija yra priskirta (jei konfigūracija nėra priskirta jokiam projektui, šis laukas yra tuščias). Lentelėje *message\_broker\_types* saugomi pranešimų skirstytojų tipai (konkrečios realizacijos). Šioje lentelėje kol kas saugomi du įrankyje naudojami tipai: *RabbitMQ* ir *ActiveMQ*.

AMQE įrankį sudarančios dalys pavaizduotos paketų bei klasių diagramomis. Šios diagramos ne tik apibrėžia projekto struktūrą, bet taip pat parodo, kokia yra sąsaja tarp atskirų įrankių sudarančiu dalių bei komponentų.

2.34 pav. pateikta paketų diagrama, vaizduojanti aukšto lygmens projekto struktūrą, nesigilinant į paketuose esančias klasės bei jų tarpusavio ryšius.



2.34 pav. UML paketų diagrama: bendras projekto vaizdas

Bendroje projekto paketų diagramoje matyti, kad įrankių sudaro dvi atskiros dalys: saityno programa *AMQE\_WebApp* bei įrankio posistemui (serverio pusės) projektas *AMQE\_API*.

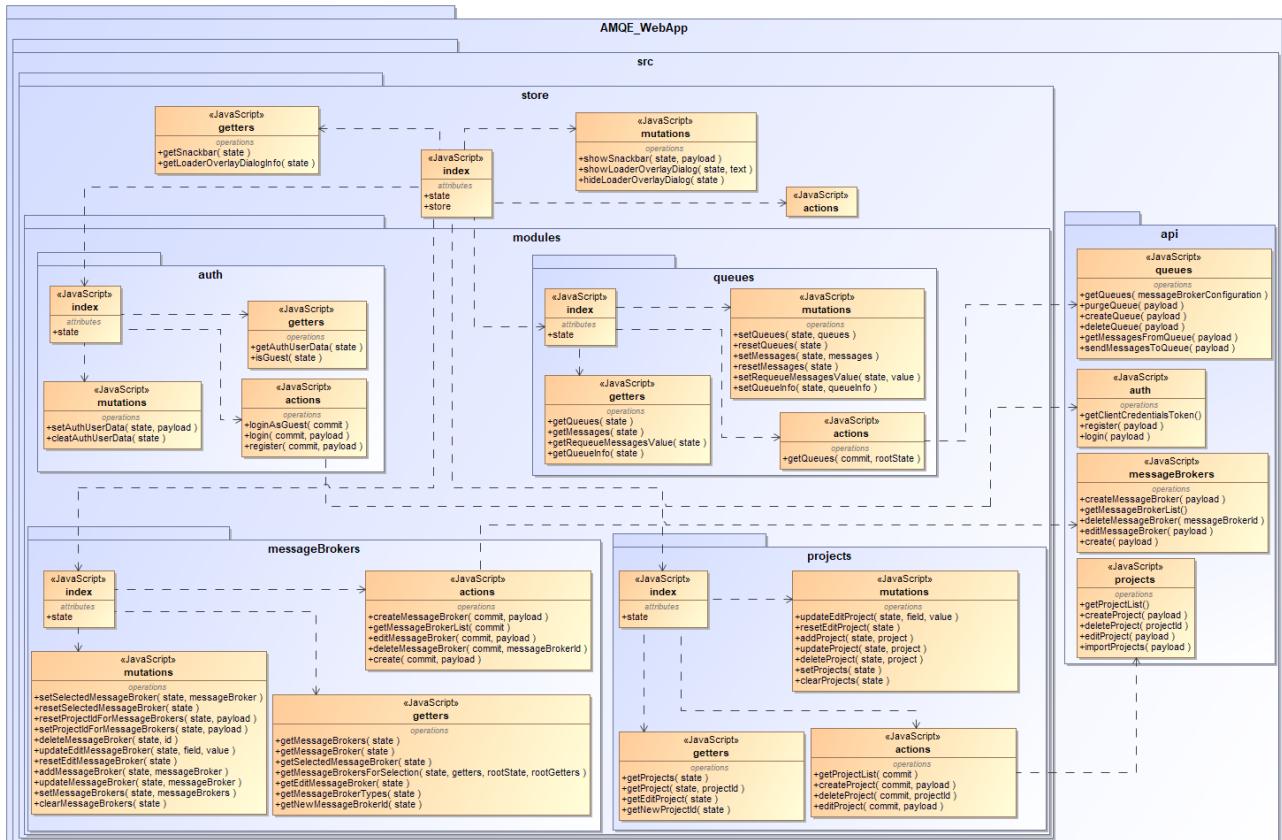
Saityno programos komponentai į paketus sugrupuoti pagal jų tipą bei paskirtį:

- *api* – komponentai, atsakingi už saityno programos komunikaciją su serverio pusės dalimi;
- *store* – saityno programos būseną saugantys moduliai;
- *libraries* – papildomos bibliotekos, pagalbinės funkcijos, konstantos;
- *components* – saityno programą sudarantys komponentai, puslapiai, dialogo langai;
- *router* – komponentai, atsakingi už navigaciją saityno programoje;
- *assets* – statiniai failai, naudojami saityno sėriajos atvaizdavimui (paveikslėliai, piktogramos, stilių failai ir kt.).

Įrankio posistemės buvo kuriamos išskaidant jas sluoksniais į kelis lygmenis:

- Sąsajos lygmuo *API* – sąsajos sluoksnis, saugantis valdiklius bei jų metodus prieigai prie posistemui metodu. Sluoksnį sudaro valdikliai (*Controllers*) bei tarpinės programinės įrangos sluoksnis *Middleware*;
- Autentifikacijos lygmuo *OAuth* – autentifikacijos sluoksnis, užtikrinantis, kad posistemui metodai būtų naudojami tik autentifikuotų naudotojų. Pagrindinė dalis – *Services*, atsakinga už naudotojų autentifikavimo, prieigos žetonų sukūrimo logiką;
- Verslo logikos lygmuo *Business* - svarbiausias posistemui lygmuo, saugantis posistemėse naudojamus objektus (*DTOs*, *Enums*), esybes (*Entities*), sąsajas (*Interfaces*), jų realizacijas (*Repositories*) bei servisus (*Services*), klaidų valdymo objektus (*Exceptions*).

Labai svarbi saityno programos dalis – komponentų būsenos bei bendrai naudojamų duomenų valdymas. 2.35 pav. pateikta saityno programos būsenos valdymo komponentų ir modulių klasų diagrama bei jų sąveika su saityno programos API sluoksniu.



**2.35 pav.** UML klasų diagrama: būsenos valdymo moduliai bei jų sąsaja su API sluoksniu

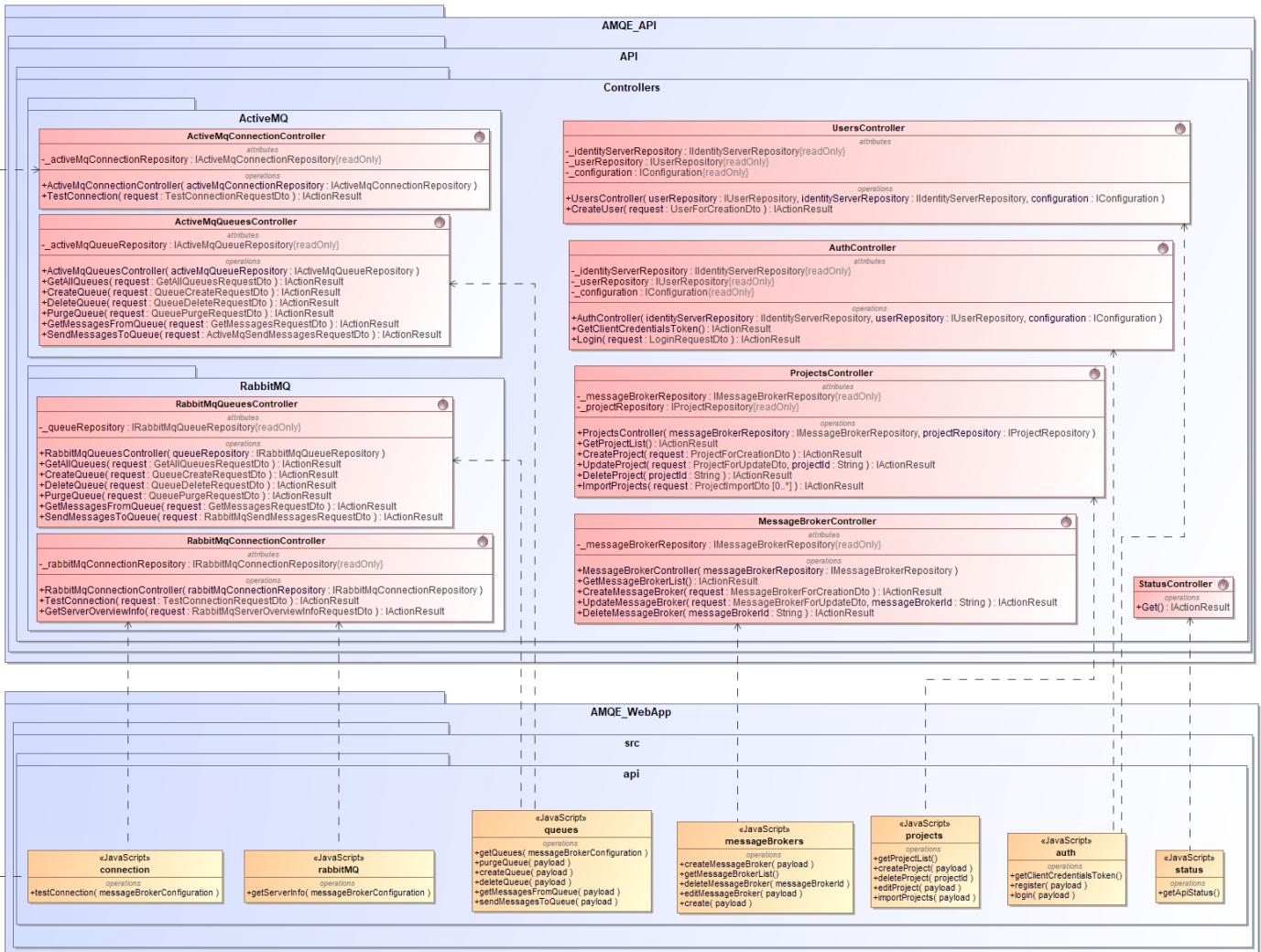
Už būsenos valdymą atsakingas sluoksnis – *store*, kuris valdo bei saugo konkrečiu momentu esančią įrankio komponentų būseną. Būsenos „saugykla“ yra išskaidyta į atskirus modulius pagal jų paskirtį, valdomas sistemos dalis:

- *auth* – modulis, atsakingas už naudotojo būsenos saugojimą ir valdymą (ar yra autentifikuotas, ar prisijungęs su savo paskyra ir t.t.);
- *messageBrokers* – modulis, atsakingas už pranešimų skirstytojų konfigūracijų būsenos saugojimą ir valdymą;
- *queues* – modulis, atsakingas už pranešimų eilių bei jose esančio turinio būsenos saugojimą ir valdymą;
- *projects* – modulis, atsakingas už pranešimų skirstytojų konfigūracijų projektų būsenos saugojimą ir valdymą.

Kiekvienas modulis (bei pati saugykla *store*) taip yra išskaidytas į 4 atskiras klasės: *getters* atsakinga už dabartinės modulio būsenos grąžinimą programai, *mutations* – už modulio būsenos pakeitimą (šioje vietoje būsenos pakeitimo veiksmai negali būti asinhroniniai), *actions* – už modulio būsenos pakeitimą, naudojant asinhroninius veiksmus (pvz., kreipiantis į serverio pusės dalį ir

gaunant informaciją), *index* – sujungia visus modulio veiksmus į vieną saugyklą, bei čia saugo visą modulio būseną, kuriai taikomi aprašyti veiksmai. Pačios saugyklos *store* viduje esanti *index* klasė apjungia visų atskirų modulių būsenų saugyklas į vieną, taip suteikiant galimybę kiekvienam moduliuui bendrauti tarpusavyje bei centralizuoti visas būsenas vienoje vietoje, kas leidžia saityno programoje gauti būseną iš kiekvieno norimo modulio, naudojant tik šakninetę *store* klasę *index*. Toks būsenos saugyklos išskaidymas moduliais leidžia lengviau prižiūrėti bei plėsti saityno aplikaciją, būsenų valdymo atskyrimas pagal prasmę padeda patogiau struktūrizuoti programos kodą bei lengviau ji suprasti. Taip pat verta paminėti, kad kiekvienas modulis saityno programos API sluoksnyje turi atskirai priskirtą klasę, kurią naudoja kiekviename modulyje esanti *actions* klasė. Tokiu būdu API veiksmai yra patogiau struktūruojami, vykdant asinhroninius veiksmus galima keisti tik konkretaus modulio būseną, bereikalingai nesikreipiant į visus modulius apjungiančias klasės.

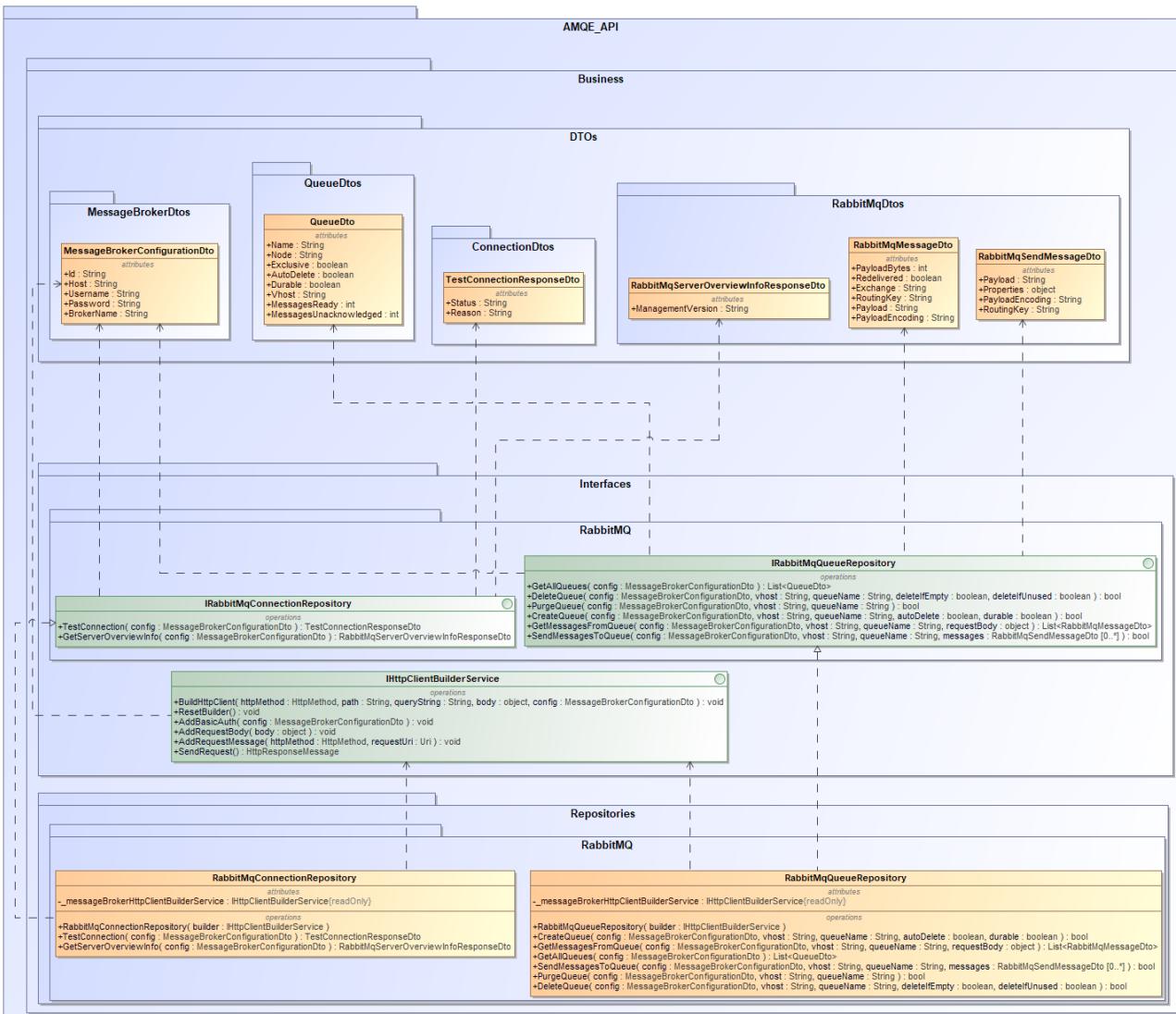
2.36 pav. pavaizduota saityno programos API pakete esančių klasių ir projekto posistemui valdiklių klasių diagrama.



2.36 pav. UML klasių diagrama: saityno programos ir posistemui sąsaja

Saityno programos API komponentai yra išskaidyti į atskirus *JavaScript* failus pagal prasmę ir funkcionalumą, atitinkantį posistemui valdiklių suteikiamus metodus. Tokiu būdu projekto kūrimo metu išlaikomas vientisumas bei tvarkinga tiek saityno programos, tiek serverio pusės dalies kodo struktūra.

Kaip pavyzdys 2.37 pav. pateikta pranešimų eilių valdymo posistemės dalis, atsakinga už RabbitMQ pranešimų eilių valdymo logiką.



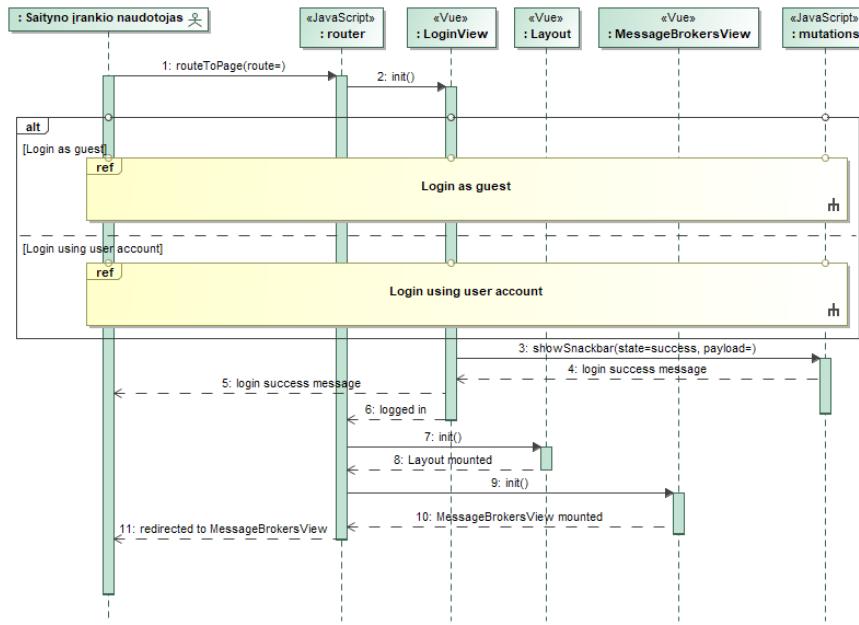
**2.37 pav.** UML klasų diagrama: *RabbitMQ* pranešimų eilių valdymo posistemės klasės

RabbitMQ pranešimų eilių valdymo logiką aprašo sasajos, esančios *Interfaces*, *RabbitMQ* pakete bei jos metodus realizuojančios repozitorijos (*angl. repositories*), patalpintos *Repositories*, *RabbitMQ* pakete, naudojami reikalingi duomenų objektai iš *DTOs* paketo. Toks paketė ir juose esančių sasajų, klasių pavadinimų vientisumas leidžia nesunkiai suprasti, kurios klasės, sasajos ar jų realizacijos priklauso kuriai pranešimų skirstytojų realizacijai, kodas išlikę aiškus net ir tada, kai didėtų klasių skaičius, pvz., pridedant papildomas pranešimų skirstytojų realizacijas.

### 2.3.2. Dinaminis sistemos vaizdas

Dinaminis sistemos vaizdas pavaizduotas sekų diagramomis. Šios diagramos parodo sąveiką tarp atskirų įrankio saityno programos ir serverio pusės dalies klasių.

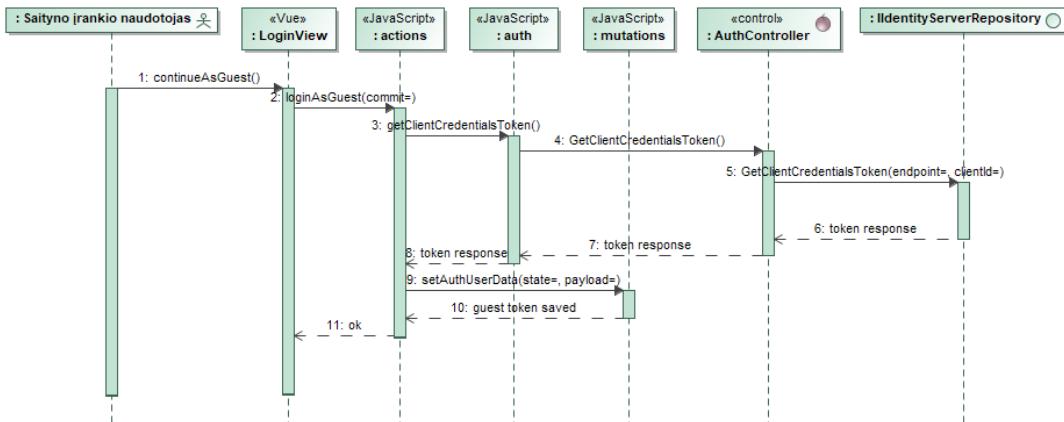
2.38 – 2.44 paveikslėliuose pateiktos naudotojų posistemės sekų diagramos. 2.38 pav. pavaizduota naudotojo prisijungimo sekų diagrama.



2.38 pav. UML sekų diagrama: prisijungti

Naudotojas, naudojant saityno aplikacijos navigatorių (*router*), yra nukreipiamas į prisijungimo puslapį (kuris kreipimosi metu yra sukuriamas). Naudotojas pasirenka vieną iš būdų, kaip nori prisijungti prie įrankio (kaip svečias arba su savo paskyra). Šių pasirinkimų sekų diagramos pavaizduotos 2.39 - 2.40 paveikslėliuose. Prisijungus vienu iš pateiktų būdų, naudotojui yra pateikiamas sėkmingo prisijungimo pranešimas, sukuriamas puslapio šablonas (*Layout*) bei pranešimų skirstytojų konfigūracijų puslapis *MessageBrokerView*, į kurį naudotojas yra nukreipiamas.

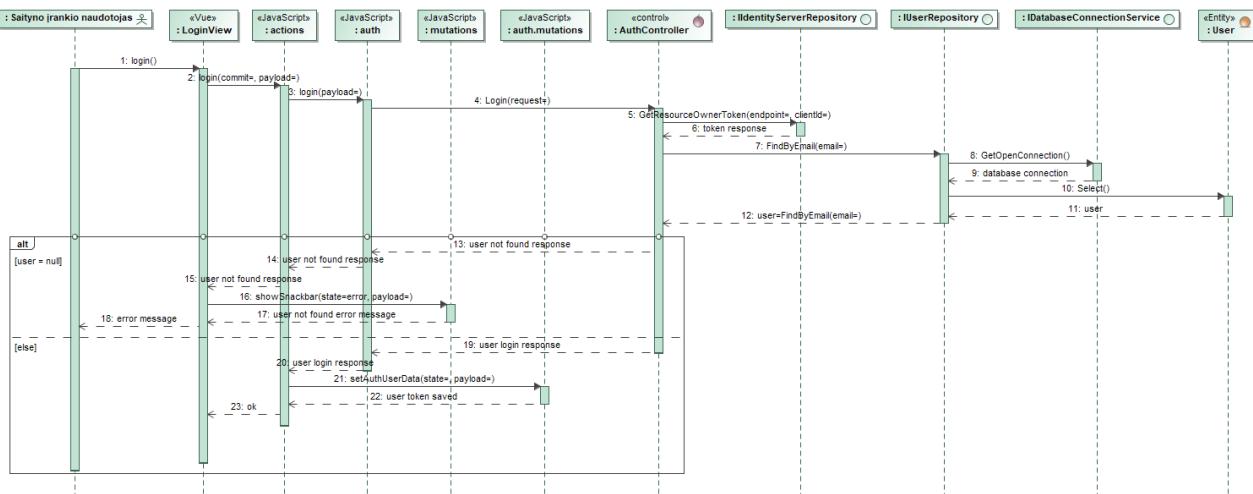
2.39 pav. pavaizduota naudotojo prisijungimo svečio teisėmis sekų diagrama.



2.39 pav. UML sekų diagrama: prisijungti kaip svečiu

Naudotojui pasirinkus prisijungimo kaip svečiu būdą, iš saityno įrankio API sluoksnio (*auth*) siunčiama užklausa į serverio pusės dalį (*AuthController*), kuris, naudojantis *IdentityServer* sąsaja, gauna svečio JWT žetoną. Žetonas yra parsiunčiamas atgal į saityno programą ir išsaugomas (kreipiantis į būsenos išsaugojimo sluoksnį, konkrečiau – *mutations*).

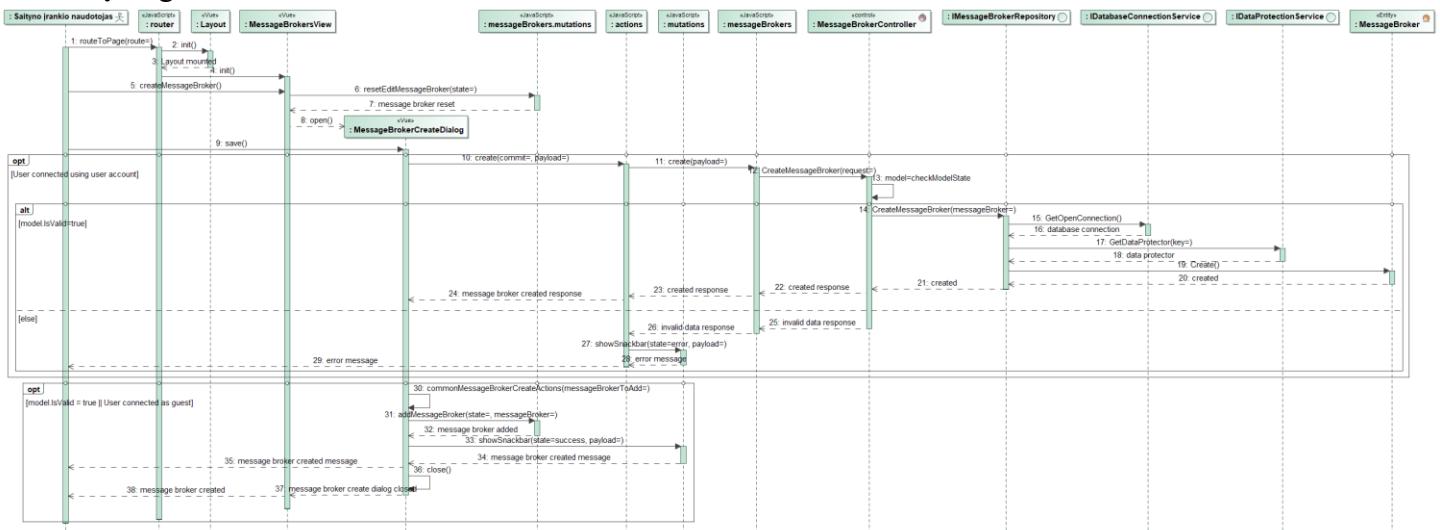
2.40 pav. pateikta naudotojo prisijungimo su savo paskyra sekų diagrama.



2.40 pav. UML sekų diagrama: prisijungti su naudotojo paskyra

Taip pat, kaip ir svečio prisijungimo atveju, naudotojui pasirinkus prisijungimą su savo paskyra, iš saityno įrankio API sluoksnio (*auth*) siunčiama užklausa į serverio pusės dalį, tačiau kreipiamasi į prisijungimo su naudotojo duomenimis metodą *Login*. Vėliau kreipiamasi į *IdentityServer* sasajos sluoksnį, sugeneruojanas JWT žetonas. Vėliau tikrinami naudotojo duomenys (ar toks naudotojas egzistuoja) – kreipiamasi į *IUserRepository*, duomenų bazėje tikrinama, ar naudotojas, su užklausoje gautu el. pašto adresu, egzistuoja. Jei tokio naudotojo nėra, saityno programai perduodama klaida, naudotojui pateikiamas klaidos pranešimas, kad negalima prisijungti. Jei užklausa serverio pusės dalyje įvykdoma sėkmingai, sugeneruotas žetonas yra išsaugomas saityno programos būsenos saugojimo sluoksnyje (*auth.mutations*).

2.41 pav. pateikta prisijungimo prie pranešimų skirstytojo serverio konfigūracijos sukūrimo sekų diagrama.

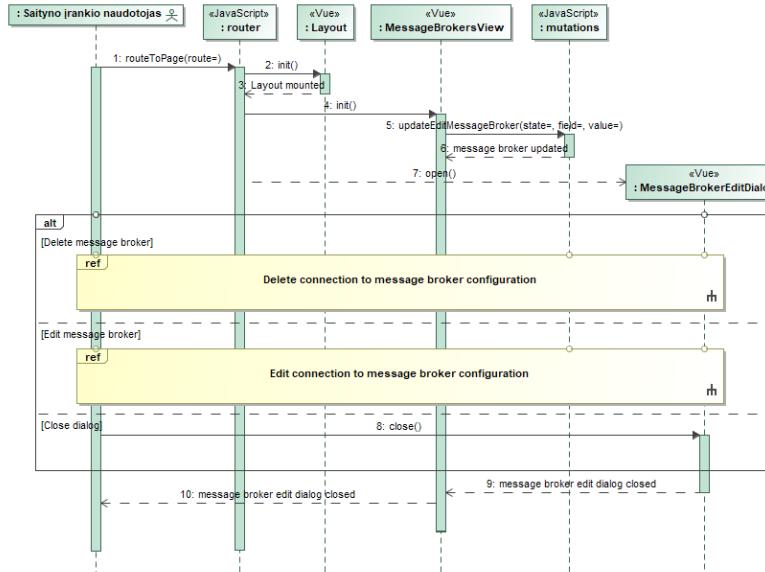


2.41 pav. UML sekų diagrama: sukurti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Naudotojas atsidaro pranešimų skirstytojų konfigūracijų puslapį (pagrindinį saityno įrankio langą), pasirenka naujos konfigūracijos sukūrimą. Prieš atidarant konfigūracijos sukūrimo dialogo langą, yra išvalomi formos laukai (funkcija *resetEditMessageBroker*). Užpildžius laukus ir pasirinkus išsaugojimą, jei naudotojas yra prisijungęs su savo paskyra, saityno įrankio API sluoknis (*messageBrokers*) siunčia užklausą į pranešimų skirstytojų valdiklį, kuriame patikrinami užklausos duomenys. Jei duomenys yra teisingi, kreipiamasi į *IMessageBrokerRepository* sasają. Pastarojoje pirmiausia yra sukuriamas prisijungimas prie duomenų bazės, naudojant *IDatabaseConnectionService* sasają, po to yra užšifruojami pranešimų skirstytojo konfigūracijos prisijungimo duomenys, naudojant *IDataProtectionService* sasają. Užšifravus duomenis, konfigūracija yra išsaugoma duomenų bazėje ir saityno programai grąžinamas sėkmingos užklausos įvykdymo atsakymas. Jei užklausos duomenys nėra teisingi, saityno programai grąžinama klaida, o naudotojui pateikiamas klaidos pranešimas.

Vėliau yra vykdomi bendrieji konfigūracijos išsaugojimo saityno programoje veiksmai (jei naudotojas yra prisijungęs su svečio paskyra, prieš tai minėti veiksmai nėra atliekami, nes duomenų bazėje konfigūracija nėra išsaugoma). Konfigūracijos duomenys saityno programoje yra išsaugomi tada, kai naudotojas yra prisijungęs su svečio paskyra arba prieš tai vykdytos užklausos duomenys yra teisingi. Tokiu atveju, saityno programos būsenos saugojimo sluoksnyje (*mutations*) yra išsaugoma nauja konfigūracija, naudotojui yra pateikiamas sėkmingai sukurtos konfigūracijos pranešimas bei uždaromas konfigūracijos sukūrimo dialogo langas.

2.42 pav. pateikta pranešimų skirstytojo konfigūracijos peržiūros sekų diagrama.



2.42 pav. UML sekų diagrama: peržiūrēti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Naudotojas, atsidaręs pagrindinį saityno įrankio puslapį (pranešimų skirstytojų konfigūracijų langą), pasirenka norimos konfigūracijos peržiūrą. Prieš atidarant konfigūracijos peržiūros/redagavimo dialogo langą *MessageBrokerEditDialog*, šiame lange yra pakeičiami formos duomenys (funkcija *updateMessageBroker*). Atsidarius dialogo langui, naudotojas gali pasirinkti vieną iš veiksmų: konfigūracijos pašalinimą arba konfigūracijos pakeitimą išsaugojimą (šių veiksmų sekų diagramos pateiktos 2.43 - 2.44 paveikslėliuose) arba tiesiog peržiūrēti konfigūracijos duomenis ir uždaryti dialogo langą. Atlikus vieną iš veiksmų, konfigūracijos peržiūros/redagavimo dialogo langas yra uždaromas.

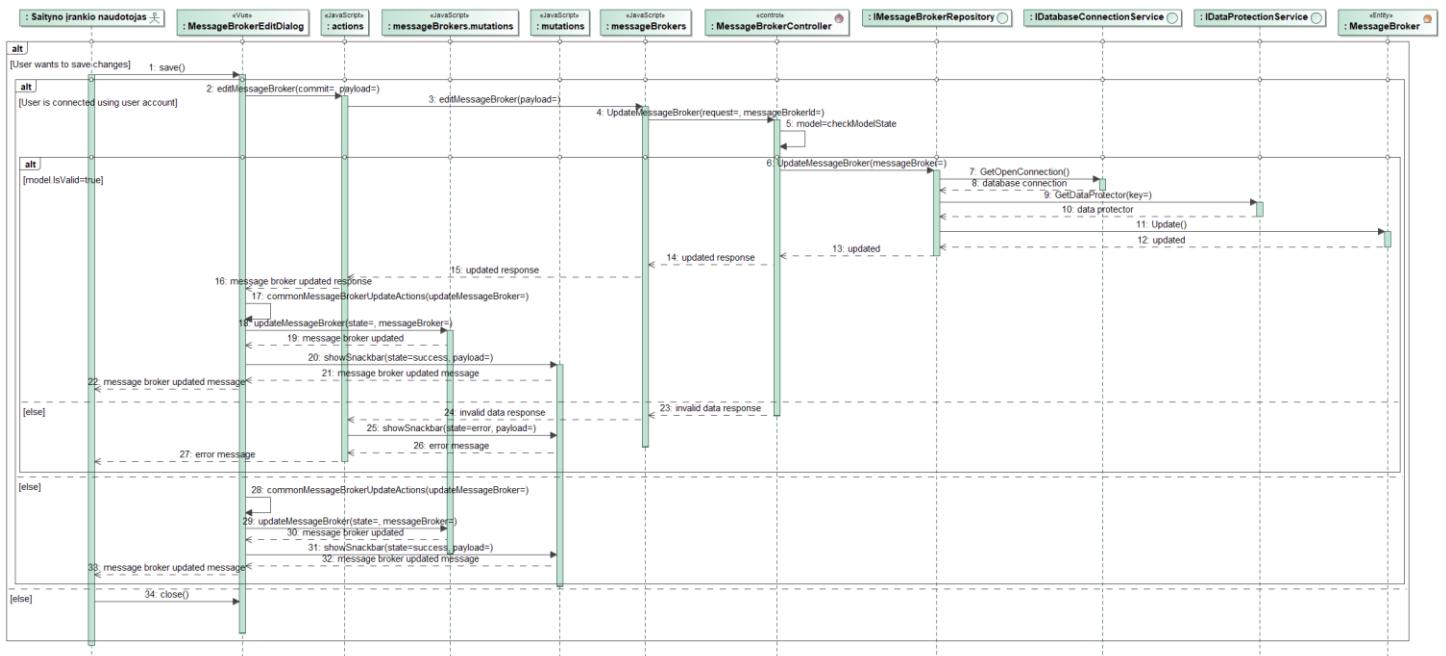
2.43 pav. pateikta prisijungimo prie pranešimų skirstytojo serverio konfigūracijos pašalinimo sekų diagrama.



2.43 pav. UML sekų diagrama: pašalinti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Naudotojas prisijungimo prie pranešimų skirstytojo konfigūracijos peržiūros/redagavimo dialogo lange pasirenka konfigūracijos pašalinimą. Naudotojui pateikiamas šalinimo patvirtinimo dialogo langas, kuriamo jis turi patvirtinti, ar tikrai nori ištrinti konfigūraciją. Jei pašalinimo veiksmas yra atšaukiamas, patvirtinimo dialogo langas yra uždaromas ir naudotojas gali toliau peržiūrėti/redaguoti konfigūraciją. Kitu atveju, jei naudotojas yra prisijungęs svečio teisėmis, saityno programoje konfigūracija yra ištrinama, kreipiantis į programos būsenos valdymo sluoksnį (*messageBrokers.mutations*), atliekami bendri pranešimų skirstytojo konfigūracijos pašalinimo veiksmai ir naudotojui yra pateikiamas sėkmindo konfigūracijos pašalinimo pranešimas bei uždaromas konfigūracijos peržiūros/redagavimo dialogo langas. Naudotojui prisijungus su savo paskyra, prisijungimo prie pranešimų skirstytojo serverio konfigūracija turi būti pašalinta ne tik iš saityno programos, bet ir iš duomenų bazės. Dėl šios priežasties, pirmiausia yra kreipiamasi į saityno programos API sluoksnį (*actions*, tada *messageBrokers*), iš kurio yra siunčiama užklausa į pranešimų skirstytojų valdiklį serverio pusės dalyje. Serverio pusėje kreipiamasi į pranešimų skirstytojų valdymo sasają *IMessageBrokerRepository*, naudojant *IDatabaseConnectionService* sasają sukuriamas prisijungimas prie duomenų bazės ir galiausiai prisijungimo konfigūracija pašalinama iš duomenų bazės. Serverio pusės dalis saityno programai parsiunčia sėkmindo pašalinimo atsakymą, konfigūracija taip pat ištrinama iš saityno programos (kreipiantis į būsenos valdymo sluoksnį) ir naudotojui pateikiamas sėkmindo konfigūracijos pašalinimo pranešimas, po to uždarant konfigūracijos peržiūros/redagavimo dialogo langą.

2.44 pav. pateikta prisijungimo prie pranešimų skirstytojo serverio konfigūracijos redagavimo sekų diagrama.



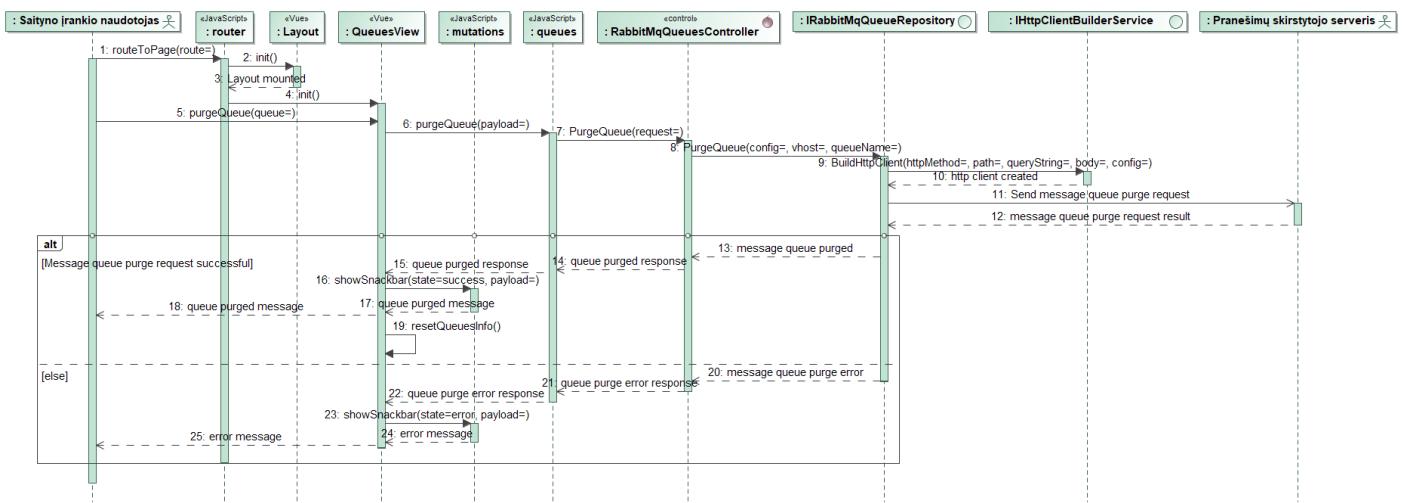
2.44 pav. UML sekų diagrama: redaguoti prisijungimo prie pranešimų skirstytojo serverio konfigūraciją

Jei naudotojas nori atliliki konfigūracijos pakeitimus, pakeičia jos duomenis ir pasirenka konfigūracijos išsaugojimą, kitu atveju – tiesiog uždaruoja konfigūracijos peržiūros/redagavimo dialogo langą. Jei naudotojas yra prisijungęs svečio teisėmis, atliekami bendri konfigūracijos duomenų pakeitimo veiksmai (naudojant saityno programos būsenos valdymo sluoksnį *messageBrokers.mutations*), naudotojui pateikiamas sėkmindo konfigūracijos duomenų pakeitimo pranešimas. Taip pat, kaip ir konfigūracijos pašalinimo atveju, jei naudotojas yra prisijungęs su savo paskyra, tai konfigūracija turi būti kartu pašalinta ir iš duomenų bazės. Tam iš saityno programos API sluoksnio *messageBrokers* siunčiama užklausa į pranešimų skirstytojų valdiklį serverio pusėje *MessageBrokerController*. Pirmiausia valdiklyje patikrinamas duomenų teisingumas. Jei duomenys nėra tinkami, tada naudotojui grąžinamas klaudingų duomenų atsakymas, pateikiamas klaidos pranešimas. Kitu atveju, kreipiamasi į pranešimų skirstytojų valdymo sasają *IMessageBrokerRepository*, kuri, naudojant *IDatabaseConnectionService* sasają, pirmiausia suria

prisijungimą prie duomenų bazės. Prieš išsaugant atnaujintus duomenis duomenų bazėje, juos reikia užšifruoti – tam naudojama *IDataProtectionService* sasaja. Galiausiai duomenys yra išsaugomi duomenų bazėje, į saityno programą parsiunčiamas sėkmingo duomenų atnaujinimo rezultatas, atliekami reikiams konfigūracijos duomenų pakeitimai saityno programoje, o naudotojui pateikiamas sėkmingo konfigūracijos duomenų atnaujinimo pranešimas.

Taip pat, kaip ir veiklos diagramose, prisijungimo prie pranešimų skirstytojo serverio konfigūracijų projektų sukūrimo, peržiūros, redagavimo bei šalinimo sekų diagramos yra beveik identiškos atitinkamai pačių prisijungimo konfigūracijų sukūrimo, peržiūros, redagavimo bei šalinimo sekų diagramoms, todėl jos nėra pateikiamas.

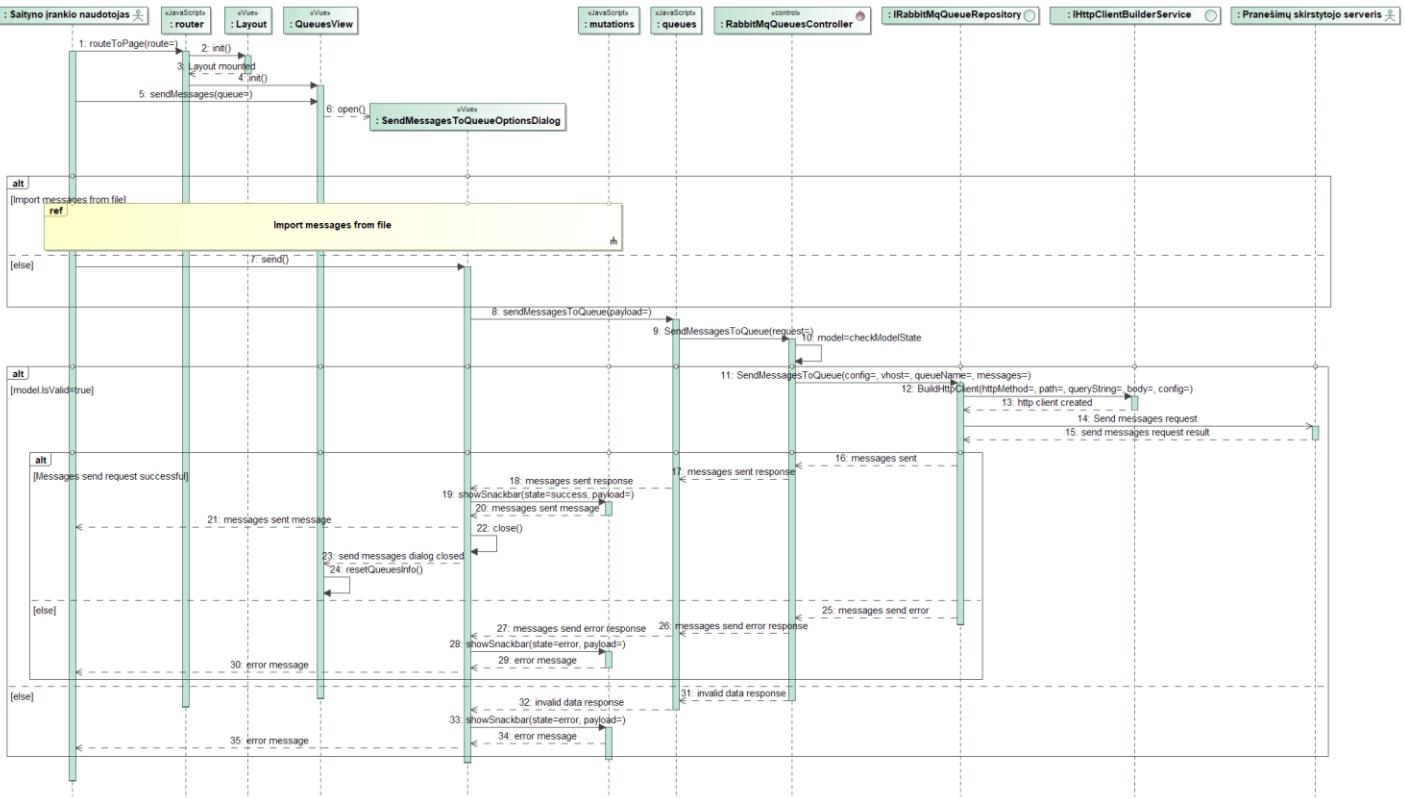
2.45 - 2.52 paveikslėliuose pateiktos pranešimų eilių valdymo posistemės sekų diagramos. Šiuose pavyzdžiuose yra valdomos *RabbitMQ* pranešimų eilės, todėl serverio pusės dalies kodą atitinka šiai realizacijai skirti valdikliai, sasajos. *ActiveMQ* pranešimų eilių valdymui visa saityno programos logika išlieka ta pati, skiriasi tik serverio pusėje esantys valdikliai, sasajos. 2.45 pav. pavaizduota pranešimų eilės išvalymo sekų diagrama.



2.45 pav. UML sekų diagrama: išvalyti pranešimų eilę

Naudotojas, atsidaręs pranešimų eilių peržiūros puslapį (navigatorius *router* pirmiausia atveria puslapio šabloną – *Layout*, kuriame pateikia pranešimų eilių langą *QueuesView*), pasirenka pranešimų eilės išvalymą. Saityno programos pranešimų eilių API sluoksnis *queues* kreipiasi į serverio pusėje esantį *RabbitMQ* pranešimų eilių valdiklį *RabbitMqQueuesController*. Valdiklis kreipiasi į *IRabbitMqQueueRepository* sasają, sukuriamas HTTP klientas (naudojant *IHttpClientBuilderService* sasają), skirtas užklausos siuntimui į pranešimų eilių skirstytojo serverį. Nusiuntus užklausą į pranešimų skirstytojo serverį ir gavus atsakymą, yra tikrinamas jo rezultatas. Jei rezultatas neigiamas (pranešimų eilė nebuvvo sėkmingai išvalyta), saityno programai yra grąžinamas nesėkmingo eilės išvalymo rezultatas, naudotojui pateikiamas klaidos pranešimas, o jei užklausa įvykdymo sėkmingai – saityno programai grąžinamas sėkmingo užklausos įvykdymo rezultatas, naudotojui pateikiamas sėkmingo pranešimų eilės išvalymo pranešimas bei atnaujinama pranešimų eilių informacija duomenų lentelėje.

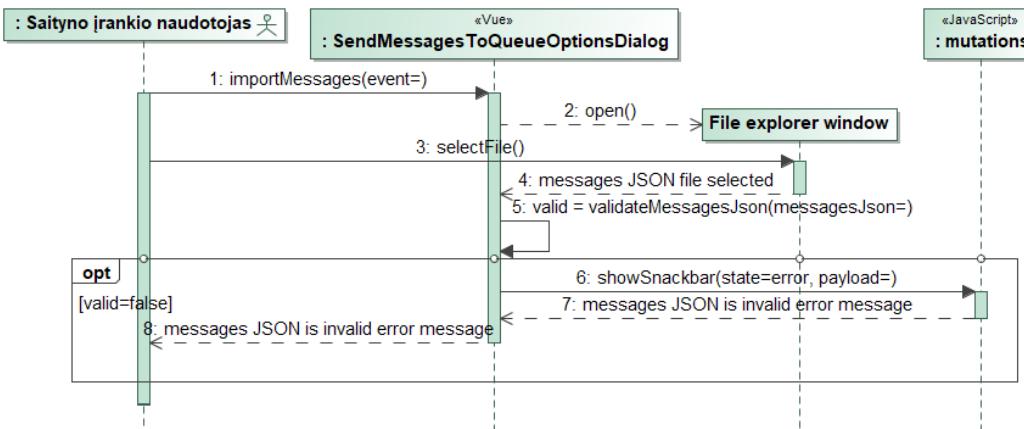
2.46 pav. pavaizduota pranešimų siuntimo į eilę sekų diagrama.



2.46 pav. UML sekų diagrama: siųsti pranešimą (-us) į eilę

Naudotojas, atsidarės pranešimų eilių peržiūros puslapį, pasirenka pranešimų siuntimą į eilę. Atsidariusiame pranešimų siuntimo į eilę dialogo lange, naudotojas turi pasirinkti, ar nori siųsti naujai sukurtą pranešimą, ar nori importuoti pranešimus iš failo (failo importavimo sekų diagrama pateikta 2.47 pav.). Jei naudotojas nusprenčia siųsti naują pranešimą, pirmiausia įveda jo duomenis ir pasirenka pranešimų siuntimą. Iš saityno programos pranešimų eilių API sluoksnio *queues* siunčiama užklausa į serverio dalį, *RabbitMqQueuesController* valdiklį. Valdiklyje pirmiausia patikrinamas užklausos duomenų teisingumas. Jei užklausos duomenys neteisingi, saityno programai grąžinamas neteisingų duomenų rezultatas, naudotojui parodomas klaidos pranešimas, kitu atveju – valdiklis kreipiasi į *IRabbitMqQueueRepository* sąsaja, kuri pirmiausia sukuria HTTP klientą kreipimuisi į pranešimų skirstytojo serverį, o vėliau suformuotą užklausą pastarajam serveriui išsiunčia. Jei užklausos vykdymo metu įvyko klaida, saityno programai grąžinamas klaidos rezultatas, naudotojui parodomas klaidos pranešimas, kitu atveju – grąžinamas sėkmingas pranešimų siuntimo rezultatas, naudotojui parodomas pranešimas, jog pranešimai buvo sėkmingai išsiųsti, pranešimų siuntimo į eilę dialogo langas yra uždaromas bei atnaujinama duomenų lentelėje esančių pranešimų eilių informacija.

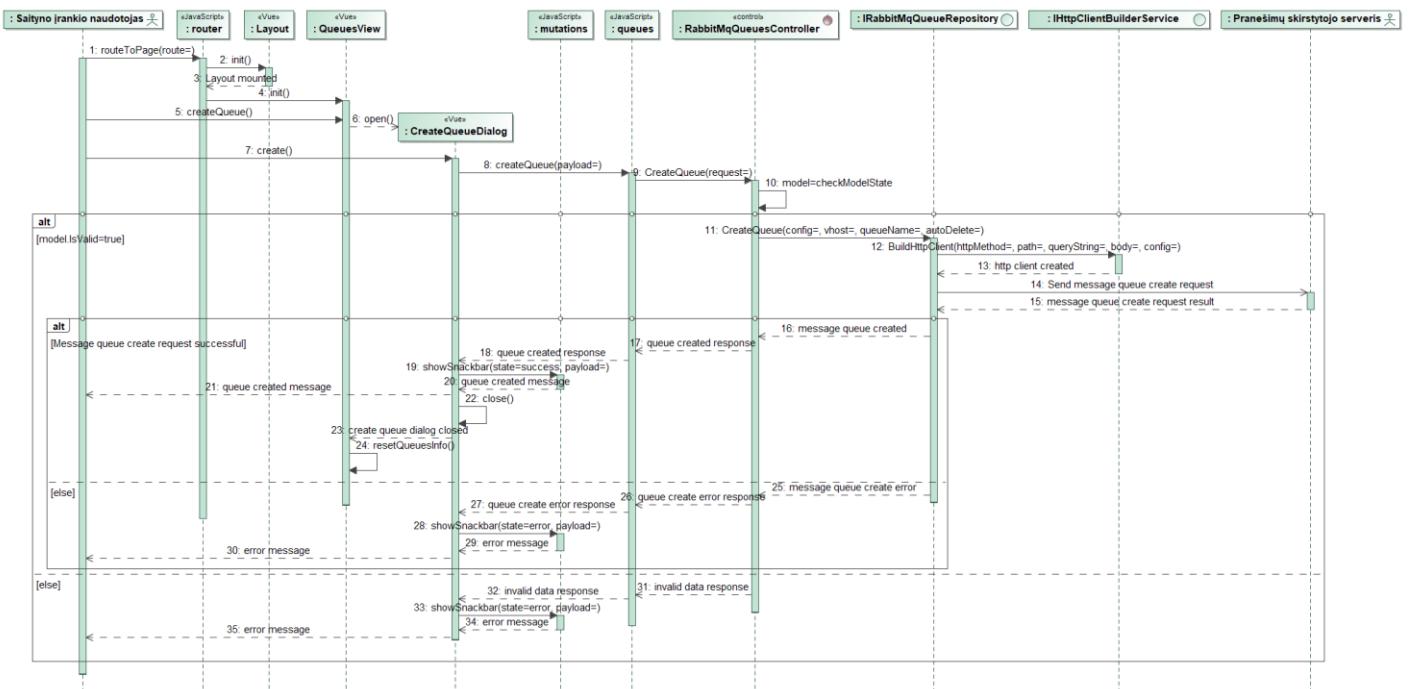
2.47 pav. pateikta pranešimų importavimo iš failo sekų diagrama.



2.47 pav. UML sekų diagrama: importuoti pranešimus iš failo

Pateiktoje pranešimų importavimo iš failo sekų diagramoje matyti, kad įrankio naudotojui pasirinkus pranešimų importavimo veiksmą, atidaromas operacinės sistemos failo pasirinkimo langas. Šiame lange naudotojas pasirenka pranešimų JSON failą (su plėtiniu *.json*). Pasirinkus failą, tikrinami jo duomenys – ar pasirinktas pranešimų JSON failas yra tinkamas (ar tinkama duomenų struktūra, ar pateikti visi reikiams pranešimų parametrai). Jei failas yra tinkamas, veiksmų seka užsibaigia ir vyksta tolimesnis pranešimų apdorojimas, pateiktas 2.47 pav. pavaizduotoje sekų diagramoje, kitu atveju – naudotojui pateikiamas klaidos pranešimas, informuojantis apie importuotą netinkamą pranešimų failą.

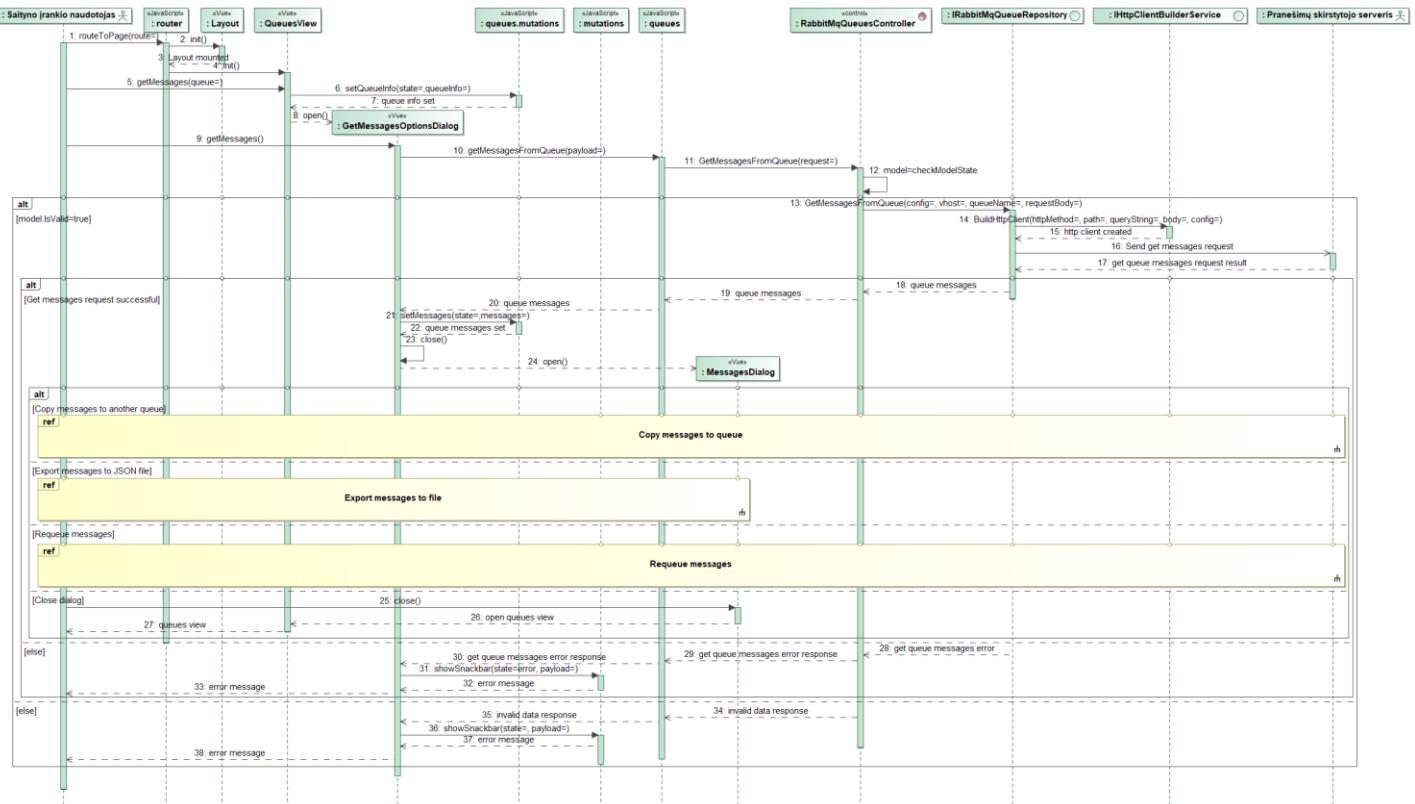
2.48 pav. pateikta pranešimų eilės sukūrimo sekų diagrama.



2.48 pav. UML sekų diagrama: sukurti naujų pranešimų eilę

Naudotojas, atsidarės pranešimų eilių peržiūros puslapį, pasirenka naujos pranešimų eilės sukūrimą. Atdidariusiame pranešimų eilės sukūrimo dialogo lange suvedama reikiama informacija, parametrai ir pasirenkamas eilės sukūrimas. Saityno programos pranešimų eilių API sluoksnius *queues* siunčia eilės sukūrimo užklausą į serverio pusėje esantį *RabbitMqQueuesController* valdikli. Valdiklyje pirmiausia patikrinamas pranešimų eilės sukūrimo duomenų teisingumas. Jei užklausa nėra tinkama, saityno programai grąžinamas klaidos rezultatas ir pateikiamas klaidos pranešimas. Jei pranešimų eilės sukūrimo duomenys teisingi, kreipiamasi į *IRabbitMqQueueRepository* sėsają, kuri pasirūpina HTTP kliento sukūrimu ir pranešimų eilės sukūrimo užklausos siuntimu į pranešimų skirstytojo serverį. Jei užklausa nebuvo įvykdyma sėkminges, saityno programai grąžinamas klaidos rezultatas, naudotojui pateikiamas klaidos pranešimus, o sėkminges pranešimų eilės sukūrimo atveju – saityno programai grąžinamas sukurtos eilės rezultatas, naudotojui pateikiamas sėkminges eilės sukūrimo pranešimas bei atnaujinama duomenų lentelėje esančių pranešimų eilių informacija.

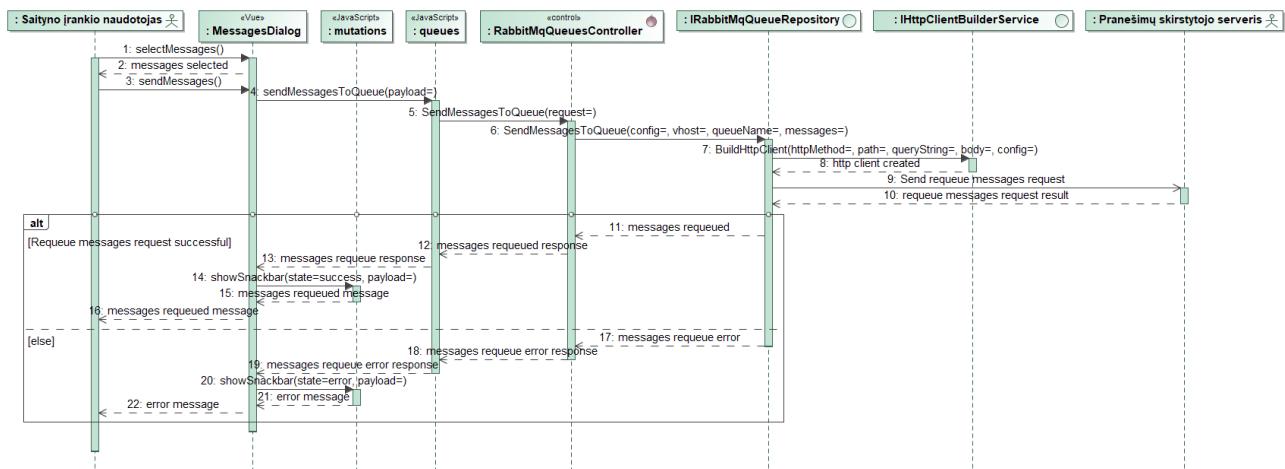
2.49 pav. pavaizduota pranešimų redagavimo eilėje sekų diagrama.



2.49 pav. UML sekų diagrama: redaguoti pranešimus eilėje

Naudotojas, atsidaręs pranešimų eilių peržiūros langą, pasirenka eileje esančių pranešimų redagavimą. Saityno būsenos saugojimo sluoksnyje *queues.mutations* išsaugoma pasirinktos eilės informacija, atveriamas pranešimų nuskaitymo (redagavimo) parametru dialogo langas. Pasirinkus norimus parametrus, saityno programos API sluoksnis *queues* kreipiasi į serverio pusės valdiklį *RabbitMqQueuesController*. Valdiklis pirmiausia patikrina, ar siunčiamos užklausos parametrai yra teisingi – jei neteisingi, saityno programai grąžinamas netinkamų duomenų rezultatas, naudotojui pateikiamas klaidos pranešimas, o jei teisingi, naudojant *IRabbitMqQueueRepository* sasaja siunčiamą užklausa į pranešimų skirstytojo serverį pranešimams iš eilės nuskaityti. Ivykdžius užklausą tikrinamas jos rezultatas – jei užklausos apdoroti nepavyko, saityno programai grąžinamas klaidos rezultatas, naudotojui pateikiamas klaidos pranešimas. Jei pranešimai buvo nuskaityti sėkmingai, saityno programai grąžinama iš eilės nuskaitytų pranešimų informacija, pranešimai yra išsaugomi saityno programoje, uždaromas pranešimų nuskaitymo parametru dialogo langas ir atidaromas kitas dialogo langas (*MessagesDialog*), skirtas pačių pranešimų informacijos redagavimui bei kitų veiksmų pasirinkimui. Naudotojas gali pasirinkti vieną iš veiksmų – kopijuoti pranešimus į kitą eilę, grąžinti nuskaitytus pranešimus į eilę arba eksportuoti pranešimus į JSON failą (šiuu veiksmu sekų diagramos pateiktos 2.50 - 2.52 paveikslėliuose). Jei naudotojas jokių veiksmų atliliki nenori, tada pasirenkamas pranešimų redagavimo dialogo lango uždarymas, grįztama į pranešimų eilių peržiūros langą.

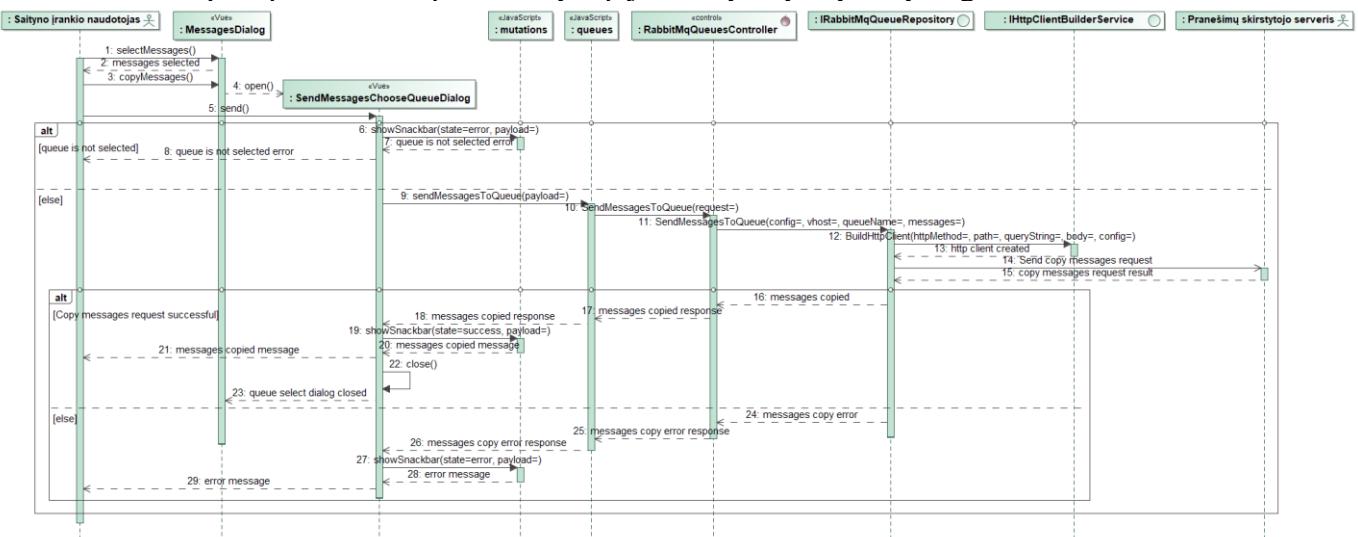
2.50 pav. pateikta pranešimų grąžinimo į eilę sekų diagrama.



2.50 pav. UML sekų diagrama: grąžinti pranešimus į eilę

Naudotojas, norėdamas grąžinti pranešimus į eilę, pirmiausia pasirenka norimus grąžinti pranešimus, jei reikia, pakoreguoja jų informaciją ir pasirenka pranešimų grąžinimo veiksmą. Saityno programos API sluoksnis *queues* siunčia užklausą į *RabbitMqQueuesController*, kuris, naudojant *IRabbitMqQueueRepository* sėsą, sukuria HTTP klientą ir, naudojant jį, siunčia pranešimų grąžinimo į eilę užklausą pranešimų skirstytojo serveriui. Nesėkmingo užklausos įvykdymo atveju, saityno programai grąžinamas klaidos rezultatas, naudotojui pateikiamas klaidos pranešimas. Jei pranešimai buvo sėkmingai grąžinti atgal į eilę, saityno programai grąžinamas sėkmingas rezultatas, naudotojui pateikiamas sėkmingo pranešimų grąžinimo į eilę pranešimas.

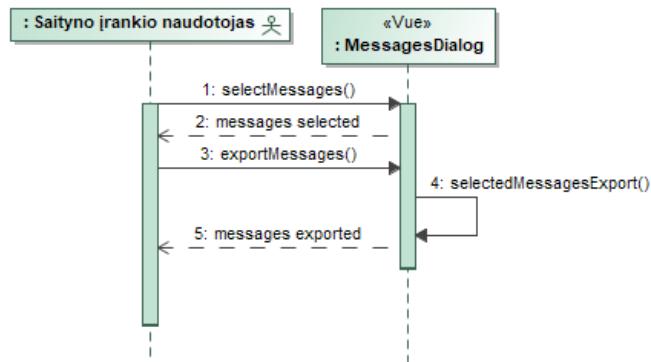
2.51 pav. Pavaizduota pranešimų kopijavimo į kitą eilę sekų diagrama.



2.51 pav. UML sekų diagrama: kopijuoti pranešimus į kitą eilę

Naudotojas, norėdamas nukopijuoti pranešimus į kitą eilę, pirmiausia pasirenka norimus kopijuoti pranešimus, jei reikia, pakoreguoja jų informaciją ir pasirenka pranešimų kopijavimo veiksmą. Atveriamas eilės pasirinkimo dialogo langas, kuriama naudotojas pasirenka eilę, į kurią bus nukopijuoti parinkti pranešimai. Dialogo lange pasirinkus kopijavimo veiksmą, tikrinama, ar buvo parinkta pranešimų eilė. Jei ji parinkta nebuvvo, naudotojui pateikiamas klaidos pranešimas, kitu atveju, saityno programos API sluoksnis *queues* siunčia užklausą į serverio pusės dalį, *RabbitMqQueuesController* valdiklį. Valdiklis, naudojant *IRabbitMqQueueRepository* ir *IHttpClientBuilderService* sėsą, sukuria HTTP klientą ir naudojant jį siunčia pranešimų kopijavimo į kitą eilę užklausą pranešimų skirstytojo serveriui. Jei užklausa nebuvvo įvykdyma sėkmingai, saityno programai grąžinamas klaidos rezultatas, naudotojui pateikiamas klaidos pranešimas, o sėkmingo užklausos įvykdymo atveju – grąžinamas sėkmingo užklausos įvykdymo rezultatas, naudotojui pateikiamas sėkmingo pranešimų nukopijavimo pranešimas, uždaromas pranešimų eilės pasirinkimo dialogo langas.

2.52 pav. pateikta pranešimų eksportavimo į failą sekų diagrama.



**2.52 pav.** UML sekų diagrama: eksportuoti pranešimus į failą

Naudotojas, norėdamas eksportuoti pranešimus į JSON failą, pirmiausia pasirenka norimus eksportuoti pranešimus, jei reikia, pakeičia jų turinį, pasirenka pranešimų eksportavimo veiksmą. Norimų eksportuoti pranešimų informacija yra suformuojama į vientisą JSON objektą ir eksportuojama į *messages.json* failą.

### 2.3.3. Duomenų kontrolė

Pirminė duomenų kontrolė yra vykdoma dar saityno programoje, informacijos įvedimo formose. Čia naudotojas privalo įvesti informaciją pagal reikalaujamas taisykles (simbolių eilutės ilgio, privalomų laukų, skaičių formatų kontrolės ir kt.). Jei naudotojo įvesti duomenys yra neteisingi, jis yra informuojamas klaidos pranešimo tekstu prie atitinkamo laukelio, blokuojamas užklausos išsiuntimo mygtuko paspaudimas (mygtukas neveiks tol, kol visų formos laukelių reikšmės nebus teisingos).

Siunčiamų užklausų duomenų kontrolė į posistemių valdiklius yra vykdoma tarpinės programinės įrangos (angl. *middleware*) lygyje. Čia, jei metodas reikalauja autentikuotos užklausos, yra tikrinama, ar kartu su užklausa buvo siunčiamas JWT žetonas, kuriame patalpinta autentikuoto naudotojo informacija. Jei užklausos žetonas yra nebegaliojantis arba jo žetono struktūra yra neteisinga, naudotojui grąžinamas HTTP 401 atsakymas,

Visoms užklausoms į įrankio posistemių valdiklius yra taikoma užklausų struktūros duomenų kontrolė, t.y., ar užklausos formatas, struktūra, metodas yra teisingi. Jei užklausa šios kontrolės nepraeina – grąžinamas HTTP 400 atsakymas, informuojantis apie neteisingą užklausą.

Galiausiai, duomenų kontrolė yra vykdoma *PostgreSQL* duomenų bazės lygyje, naudojant apibrėžtus stulpelių tipus, laukų unikalumo aprivojimus.

### 3. TESTAVIMAS

Šiame skyriuje pateikiamas sudarytas įrankio testavimo planas, kurio atskiri punktai nagrinėjami detaliau komponentų ir vartotojo sąsajos testavimo bei testavimo kriterijų skyreliuose.

#### 3.1. Testavimo planas

Kuriamo įrankio testavimas buvo atliekamas šia tvarka:

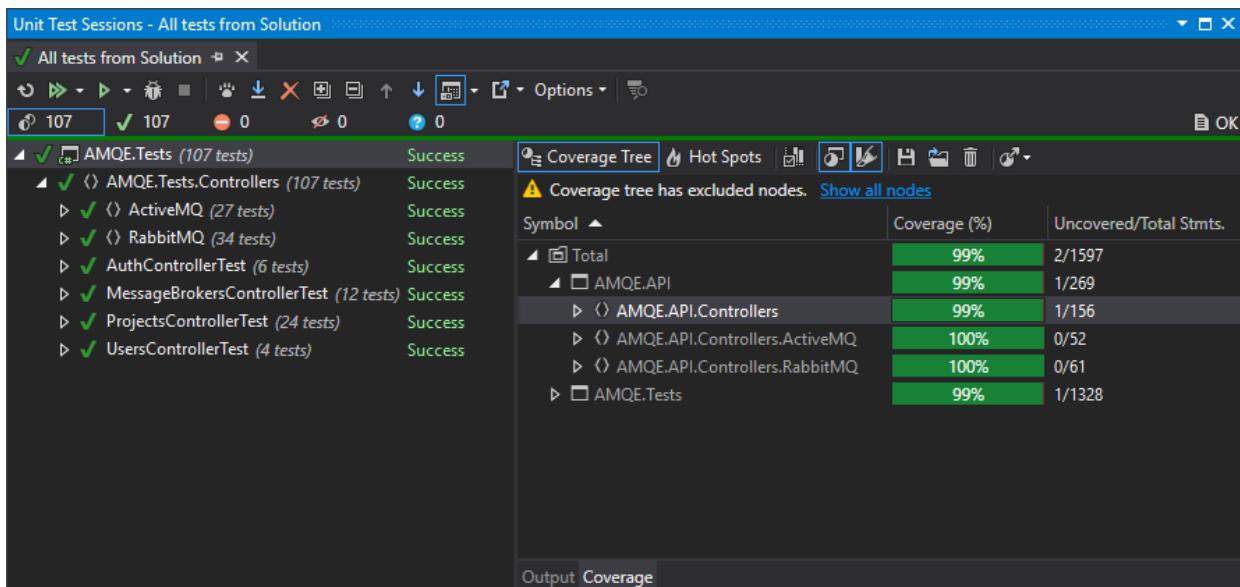
- 1) Statinė serverio pusės kodo analizė;
- 2) Automatinis posistemių valdiklių testavimas;
- 3) Rankinis posistemių valdiklių testavimas;
- 4) Statinė kliento pusės kodo analizė;
- 5) Rankinis saityno įrankio testavimas.

Statinė kodo analizė (tieki serverio, tieki kliento pusės kodo) buvo vykdoma tuo pačiu metu, kaip ir rašant programos kodą. Pirmiausia ji buvo atliekama naudojant *Visual Studio 2017* integruotos programavimo aplinkos įskiepij *ReSharper* serverio pusės bei *ESLint* įrankį saityno programos statinei kodo analizei atliki. Po kiekvienos projekto kūrimo iteracijos (pradedant trečiąja) buvo atliekama statinė kodo analizė be jokio įrankio – programos kodo peržiūra. Jos metu buvo peržiūrimes kodas ir identifikuojamos tos klaidos, kurių negali aptikti statinės kodo analizės įrankiai (pvz., gerųjų programavimo praktikų naudojimas). Plačiau apie konkrečius kodo peržiūros kriterijus aprašyta 3.3 skyriuje.

#### 3.2. Komponentų testavimas

##### 3.2.1. Automatinis testavimas

Kuriamo įrankio posistemių valdiklių testavimui buvo kuriami automatiniai vienetų testai. Kadangi įrankio saityno programa kreipiasi į posistemių suteikiamą API, svarbiausia automatiniais testais buvo padengti būtent naudotojų bei pranešimų eilių valdymo posistemių API valdiklius, jų metodų logiką. 3.1 pav. pavaizduoti vienetų testų vykdymo rezultatai.



3.1 pav. Vienetų testų vykdymo bei API metodų kodo padengimo rezultatai

Tam, kad automatiniais vienetų testais būtų padengti visi įmanomi API metodų vykdymo eigos keliai, iš viso parašyti 107 testai, o posistemių API valdiklių programinis kodas padengtas 99% (remiantis *dotCover* įskiepio rezultatų duomenimis, matomais 3.1 pav.). Vienetų testų vykdymo rezultatai parodo, kad testavimas atliktas sėkmingai, nenumatyta funkcijų elgsena nebuvo aptikta. Taip pat automatinis vienetų testavimas sumažino galimų klaidų ar nenumatytos funkcijų elgsenos atsiradimo riziką, jei būtų atliekami posistemių API valdiklių programos kodo pakeitimai ateityje.

### 3.2.2. Rankinis testavimas

Rankinis įrankio API serviso testavimas buvo atliekamas peržiūrint programos kodą, taip identifikuojant visus galimus API metodų vykdymo kelius, siunčiant HTTP užklausą į testuojamą metodą ir tikrinant, ar gautas užklausos atsakymas sutampa su kodo peržiūros metu nustatytu laukiamu rezultatu. HTTP užklausų siuntimas į įrankio API serviso metodus buvo vykdomas naudojant API dokumentavimo bei testavimo įrankį *Swagger*, kuris automatiškai sugeneruoja užklausos struktūrą kiekvienam sukurtam valdiklio API metodui, taip pagreitindamas rankinjų API metodų testavimo procesą.

Peržiūrint kiekvieno API metodo programos kodą buvo sudaromi testavimo scenarijai, nurodantys, kokie užklausos duomenys turi būti siunčiami ir koks yra laukiamas užklausos rezultatas. Scenarijus sudaro konkretūs testavimo atvejai, kurie, priklausomai nuo siunčiamos užklausos duomenų, grąžina konkretų programos kode numatyta rezultatą. 3.1 - 3.4 lentelėse aprašyti galimi *RabbitMQ* eilės pašalinimo metodo scenarijus atvejai: testuojamo metodo pavadinimas, scenarijus apibūdinimas (kokis atvejis yra testuojamas), užklausos duomenys ir laukiami rezultatai.

#### 3.1 lentelė. *RabbitMQ* pranešimų eilės pašalinimo atvejis: netinkama užklausos struktūra

Testuojamo API metodo pavadinimas	DeleteQueue
Scenarijus apibūdinimas	<i>RabbitMQ</i> pranešimų eilė neturi būti pašalinta, nes užklausos struktūra nėra tinkama arba jos reikšmė yra <i>null</i> .
Užklausos duomenys	{ }
Laukiamas rezultatas	Grąžinamas HTTP 400 rezultatas su klaidos pranešimo tekstu "Bad request: request information is invalid".

#### 3.2 lentelė. *RabbitMQ* pranešimų eilės pašalinimo atvejis: netinkama konfigūracija

Testuojamo API metodo pavadinimas	DeleteQueue
Scenarijus apibūdinimas	<i>RabbitMQ</i> pranešimų eilė neturi būti pašalinta, nes pranešimų skirstytojo konfigūracijos duomenų struktūra nėra tinkama arba jos reikšmė yra <i>null</i> .
Užklausos duomenys	{ "vhost": "/", "name": "TestQueue", "deleteIfEmpty": true, "deleteIfUnused": true, "messageBrokerConfiguration": null }
Laukiamas rezultatas	Grąžinamas HTTP 400 rezultatas su klaidos pranešimo tekstu "Bad request: Message broker information is invalid.".

**3.3 lentelė.** *RabbitMQ* pranešimų eilės pašalinimo atvejis: norimos ištrinti eilės nėra

Testuojamo API metodo pavadinimas	DeleteQueue
Scenarijaus apibūdinimas	<i>RabbitMQ</i> pranešimų eilė neturi būti pašalinta, nes pranešimų eilės informacija yra neteisinga (negalima rasti nurodytos pranešimų eilės).
Užklausos duomenys	{ "vhost": "/", "name": "TestQueue", "deleteIfExists": true, "deleteIfUnused": true, "messageBrokerConfiguration": { "id": null, "host": "http://localhost:15672", "username": "guest", "password": "guest" } }
Laukiamas rezultatas	Grąžinamas HTTP 500 rezultatas su klaidos pranešimo tekstu "Not Found".

**3.4 lentelė.** Sėkmingas *RabbitMQ* pranešimų eilės pašalinimo atvejis

Testuojamo API metodo pavadinimas	DeleteQueue
Scenarijaus apibūdinimas	<i>RabbitMQ</i> pranešimų eilė turi būti pašalinta.
Užklausos duomenys	{ "vhost": "/", "name": "TestQueue", "deleteIfExists": true, "deleteIfUnused": true, "messageBrokerConfiguration": { "id": null, "host": "http://localhost:15672", "username": "guest", "password": "guest" } }
Laukiamas rezultatas	Grąžinamas HTTP 204 rezultatas.

3.1 - 3.4 lentelėse aprašytois užklausos buvo siunčiamos naudojant *Swagger* įrankio grafinę sąsają. Šiuo įrankiu siunčiamos užklausos pavyzdys (*RabbitMQ* pranešimų eilės pašalinimo užklausa, kai pranešimų skirstytojo konfigūracija yra netinkama) pateiktas 3.2 pav.

The screenshot shows the Swagger UI interface for a DELETE request to the endpoint `/api/rabbitmq/queues/vhost/name`. The request body is defined as follows:

```
{
  "vhost": "/",
  "name": "TestQueue",
  "deleteIfEmpty": true,
  "deleteIfUnused": true,
  "messageBrokerConfiguration": null
}
```

The response content type is set to `application/json`. The server response shows a `400` status code with the error message `"Bad request: Message broker information is invalid."`.

### 3.2 pav. Swagger įrankiu siunčiamos užklausos pavyzdys ir gautas užklausos rezultatas

Tokie rankinio API valdiklių logikos testavimo scenarijai buvo sudaryti kiekvienam API metodui. Iš viso aprašyti 28 testavimo scenarijai, kurie bendrai apima 107 testavimo atvejus – tiek pat, kiek ir sukurta automatinių vienetų testų. Atlikus rankinį API metodą testavimą nebuvo rasta jokių nenumatytyų programos elgsenos atvejų, metodai grąžino tokius rezultatus, kokie buvo numatyti scenarijų testavimo atvejuose.

### 3.3. Testavimo kriterijai

Kaip jau minėta anksčiau, buvo atliekamas tiek automatinis, tiek rankinis naudotojo sąsajos ir serverio pusės kodo testavimas. Taip pat taikytas vienas iš statinės kodo analizės metodų, nereikalaujantis jokio papildomo įrankio – programos kodo peržiūra. Tam, kad kodo peržiūros procesas būtų nuoseklus, jis buvo suskaidytas į atskirus žingsnius, kurie atitinka statinės kodo analizės kriterijus:

- 1) Kodo formatavimo tikrinimas – ar kodas rašomas nuosekliai, ar yra taikomos teisingos kodo rašymo taisyklės (pvz., kintamujų pavadinimams naudojamas *camelCase*, klasių pavadinimai prasideda didžiaja raide ir kt.);
- 2) Kodo architektūros tikrinimas – ar programos kodas yra išskaidytas į komponentus arba failus, ar laikomasi sluoksniuotos architektūros principo;
- 3) Gerujų programavimo praktikų naudojimas – ar kode nėra „stebuklingų skaičių“ (kai vietoj kintamujų programos kode yra naudojamos kietai įsiūtos reikšmės ar skaičiai, neturintys jokios semantinės prasmės bei jų nustatytos reikšmės paaiškinimo), ar taikomas kintamujų tipų tikrinimas, ar tinkamai išnaudojamos projekte naudojamų karkasų/bibliotekų galimybės (pvz., nėra iš naujo be reikalo aprašomos jau sukurtos funkcijos);
- 4) Kodo saugumo tikrinimas – ar kodas yra apsaugotas nuo SQL injekcijų, ar į naršyklės bei konsolės langus nėra išvedamos kintamujų reikšmės arba testavimo duomenys/rezultatai,

programos kodo fragmentai, ar tikrinamos įvedimo laukų reikšmės bei jų tipai, ar serverio pusės dalies API valdikliai gali būti pasiekiami neautentifikuotiemis naudotojams.

Taikant rankinį testavimą (pvz., rankinį sistemos funkcionalumo testavimą, atliekant aprašytus scenarijus) konkretūs kriterijai taikyti nebuvu – vykdant konkrečius scenarijų veiksmus buvo stebima sistemos reakcija, kartu tikrinant, ar įrankio rezultatai atitinka aprašytus dokumentacijoje. Scenarijus buvo laikomas atlirk tiki tada, kai įvykdyti visi skirtinių scenarijaus atvejai, o jų rezultatai sutampa su nurodytais dokumentacijoje.

### 3.4. Vartotojo sasajos testavimas

Saityno programos sasajos testavimas buvo atliekamas remiantis kuriamo įrankio panaudos atvejais, veiklos diagramomis – ar numatytais funkcionalumas veikia taip, kaip tikimasi. Prieš atliekant rankinį programos testavimą buvo sudaromi scenarijai. Kadangi įrankio funkcionalumas yra gana platus, o panaudos atvejai gali turėti kelis skirtinges vykdymo eigos kelius, galimų saityno sasajos testavimo scenarijų skaičius taip pat auga ir tampa gana didelis. Dėl šios priežasties testavimo scenarijų aprašymui pasirinkta naudoti *Gherkin* kalbos sintaksę, skirtą aprašyti programinės įrangos elgseną (šiuo atveju – testavimo scenarijus). Šios kalbos sintaksė padeda sutrumpinti bei struktūrizuoti testavimo scenarijus, o aprašyti scenarijai yra lengvai aprašomi bei skaitomi tiek programuotojams, tiek testuotojams.

3.5 lentelėje pateiktas pranešimų skirstytojų konfigūracijų projektų sukūrimo testavimo scenarijų pavyzdys, aprašytas *Gherkin* sintakse.

### 3.5 lentelė. *Gherkin* sintakse aprašytų testavimo scenarijų pavyzdys

Pranešimų skirstytojų konfigūracijos projektų sukūrimo testavimo scenarijai (pateikti <i>Gherkin</i> sintakse)
<b>Feature:</b> Project create
<b>Scenario:</b> Open project create dialog
<b>Given</b> the user is in the main application window
<b>And</b> can see the projects side menu
<b>When</b> the user presses the „Add new project“ button
<b>Then</b> the user sees project create dialog with title „New project“
<b>Scenario:</b> Close project create dialog
<b>Given</b> the user sees project create dialog
<b>When</b> the user presses the „x“ button at the top right of the project create dialog
<b>Then</b> the project create dialog closes
<b>Scenario:</b> Creating a new project (without title)
<b>Given</b> the user sees project create dialog
<b>When</b> the user presses the „save“ button
<b>Then</b> input field's label with name „Project title“ turns red
<b>And</b> „save“ button becomes disabled
<b>Scenario:</b> Creating a new project (unsuccessful)
<b>Given</b> the user sees project create dialog
<b>When</b> the user enters <title> as „Project title“
<b>And</b> presses the „save“ button
<b>Then</b> red alert with error reason text is shown at the right bottom corner
<b>Examples:</b>
title
Error project
Any Project
Random project
<b>Scenario:</b> Creating a new project (successful)
<b>Given</b> the user sees project create dialog
<b>When</b> the user enters <title> as „Project title“
<b>And</b> presses the „save“ button
<b>Then</b> a new project with title <title> is created
<b>And</b> green alert with text „Project created!“ is shown at the right bottom corner
<b>And</b> project create dialog closes

**Examples:**

title	
New project	
Test Project	
Random name	

**Scenario:** Creating a new project (with message brokers assignment)

**Given** the user sees project create dialog

**And** the following message broker configurations (without assigned projects) exist

name	host	type	username	password	project
MB 1	http://localhost:15672	RabbitMQ	guest	guest	null
MB 2	http://localhost:15672	RabbitMQ	guest	guest	null
MB 3	http://localhost:15672	RabbitMQ	guest	guest	null

**When** the user enters <title> as „Project title“

**And** selects <selected message brokers> in „Message brokers“ select input

**And** presses the „save“ button

**Then** a new project with title <title> is created

**And** green alert with text „Project created!“ is shown at the right bottom corner

**And** project create dialog closes

**And** <selected message brokers> are assigned to the created project

**Examples:**

title	selected message brokers	
New project	MB 1	
Test Project	MB 1, MB 2	
Random name	MB 1, MB 2, MB 3	

Kaip matome iš *Gherkin* kalba aprašyto testavimo scenarijų pavyzdžio, testuojanas funkcionalumas turi jam priskirtus testavimo scenarijus (*Scenario*), kurie aprašo būtinas sąlygas, norint įvykdyti testą (*Given*), pačia veiksmų seką (*When*), laukiamus rezultatus (*Then*) ir, jei reikia, testavimo eigoje reikalingų duomenų pavyzdžius (*Examples*).

## 4. DOKUMENTACIJA NAUDOTOJUI

Šiame skyriuje pateikiamas apibendrintas kuriamo įrankio funkcijų aprašymas, vartotojo vadovas, skirtas supažindinimui su įrankio naudotojo sąsaja bei diegimo vadovas, kuriame pateikiama naudoto automatinio diegimo įrankio *Buddy* informacija, automatizuoto diegimo scenarijai.

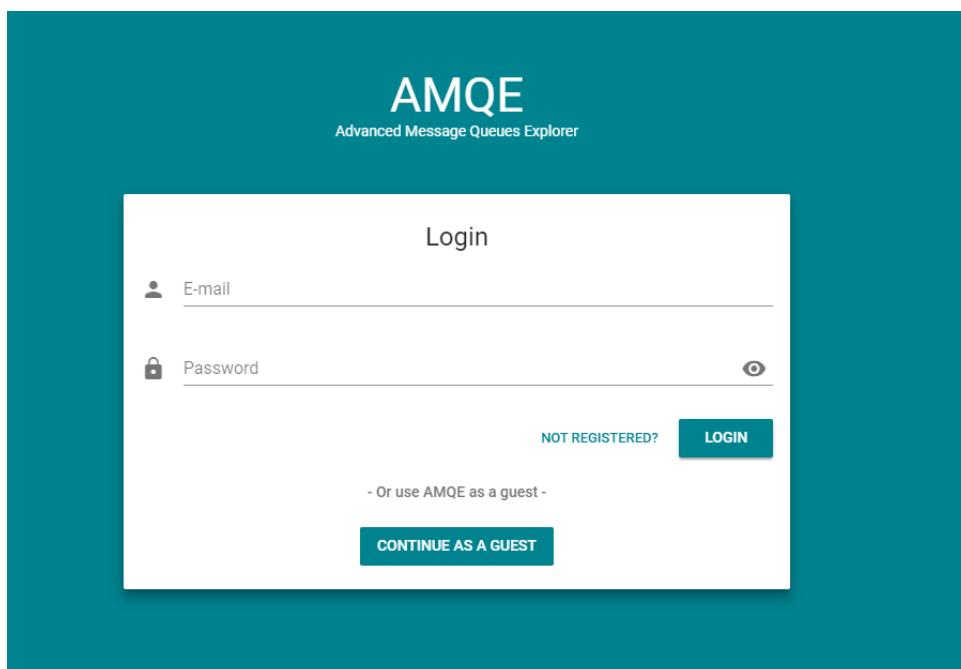
### 4.1. Apibendrintas sistemos galimybių aprašymas

Realizuotas įrankis suteikia naudotojui galimybę prisijungti prie pranešimų skirstytojo serverio ir matyti jame esančias pranešimų eiles bei atlikti veiksmus su jomis: siųsti bei gauti pranešimus, kopijuoti pranešimus iš vienos eilės į kitą, sukurti naują, ištrinti esamą eilę, išvalyti eilėje esančius pranešimus. Tam, kad prisijungtų prie pranešimų skirstytojo serverio, naudotojas turi sukurti prisijungimo konfigūraciją, kurios informaciją vėliau gali koreguoti. Taip pat šias konfigūracijas galima grupuoti bei struktūruoti kuriant konfigūracijų projektus. Visus šiuos veiksmus naudotojas gali atlikti naudodamas svečio teisėmis, tačiau jei reikia, kad sukurtos prisijungimų konfigūracijos bei sukurti projektais išsisaugotų, naudotojas gali susikurti savo paskyrą.

### 4.2. Vartotojo vadovas

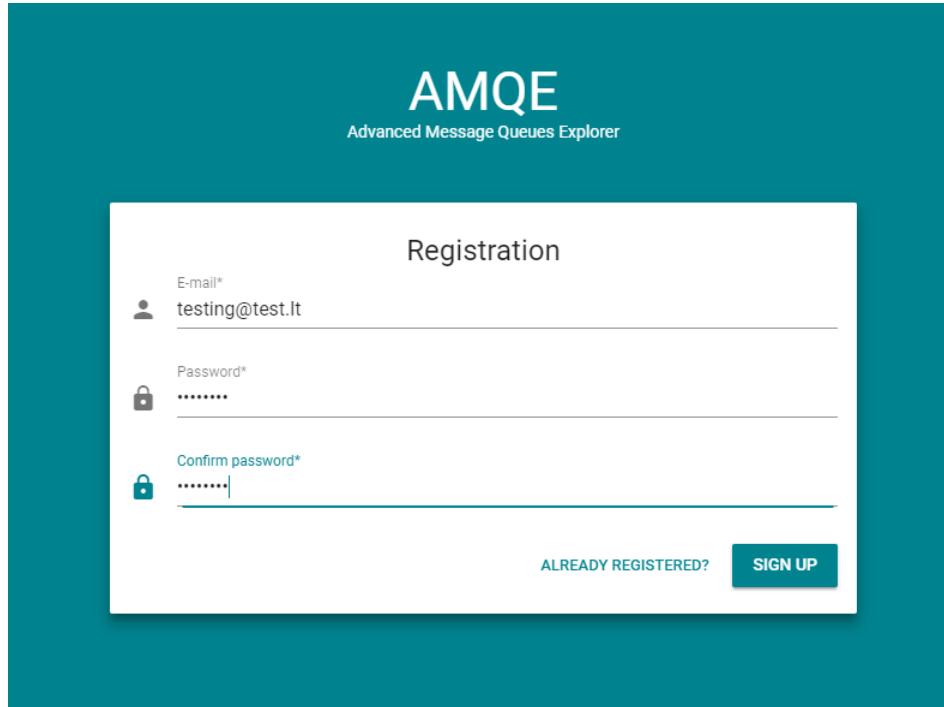
Dokumentacijoje pateikiamas konkretus įrankio naudojimosi scenarius – naudotojas susikuria naują paskyrą, sukuria naują pranešimų skirstytojų konfigūracijų projektą bei jam priskiria taip pat naujai sukurtą pranešimų skirstytojo konfigūraciją. Vėliau naudotojas prisijungia prie pasirinkto pranešimų skirstytojo, jau turinčio šias pranešimų eiles: *Work Queue* bei *Error Queue*. Naudotojas nuskaito visus pranešimus iš *Error Queue* eilės, pakoreguoja jų informaciją ir juos perkelia į eilę *Work Queue*. Vėliau sukuriama nauja eilė *Test Queue*, į ja patalpinami 5 nauji pranešimai. Tada ištrinama nenaudojama eilė *aliveness-test* ir naudotojas atsijungia nuo pranešimų skirstytojo. Galiausiai sukurto projekto informacija (kartu su priskirta pranešimų skirstytojo konfigūracija) yra eksportuojama ir išsaugoma *projects.json* faile, naudotojas atsijungia nuo savo paskyros ir baigia darbą. Šis įrankio naudojimosi scenarius atskleidžia pagrindinių įrankio funkcionalumą, parodomas pagrindinių langų vaizdas, tad išnagrinėjus ir išmèginus aprašytą scenarijų, naudotojas jau bus susipažinės su įrankio sąsaja ir galės naudotis įrankiu savo reikmėms.

Pirmausia yra atveriamas AMQE saityno įrankio langas, kuriame pasirenkamas mygtukas „Not Registered?“, nukreipiantis naudotoja į registracijos langą (4.1 pav.).



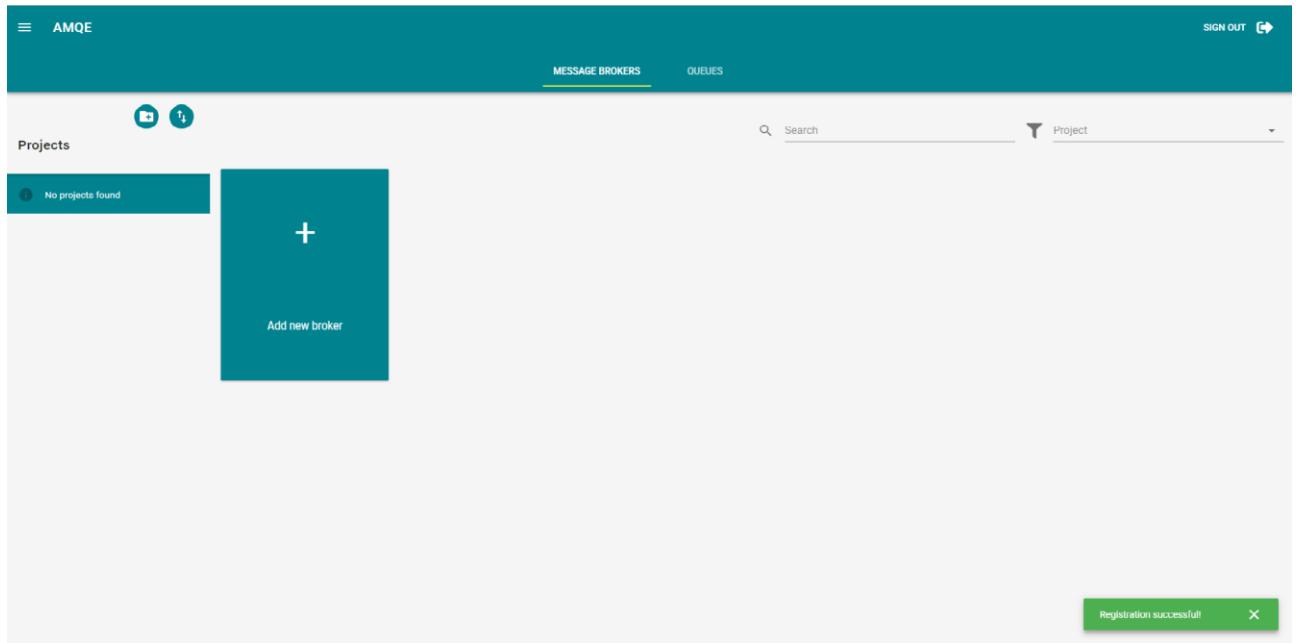
4.1 pav. AMQE prisijungimo langas

Naudotojo paskyros sukūrimo (registracijos) lange suvedama naudotojo informacija – el. pašto adresas ir slaptažodis - bei paspaudžiamas mygtukas „Sign Up“ (4.2 pav.).



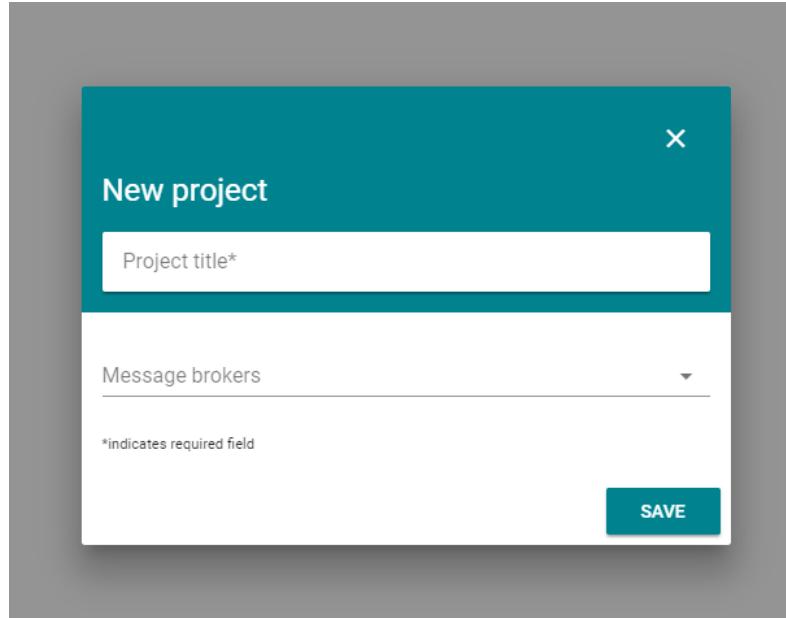
**4.2 pav.** Naudotojo paskyros sukūrimo langas

Sukūrus paskyrą, naudotojas yra nukreipiamas į pagrindinį saityno įrankio langą, apatiniaame dešiniajame kampe rodomas pranešimas, informuojantis apie sėkmingą registraciją (4.3 pav.).



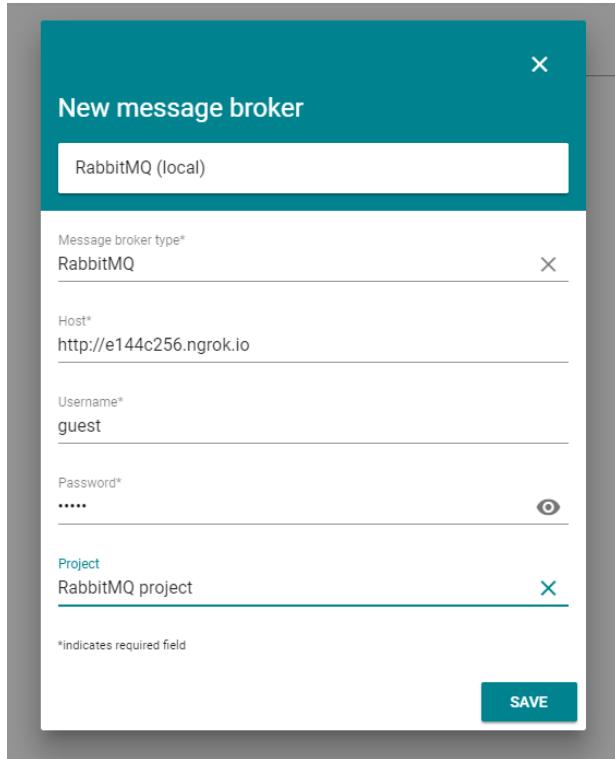
**4.3 pav.** Pagrindinis saityno įrankio langas

Norint sukurti naują pranešimų skirstytojų konfigūracijų projektą, kairėje pusėje esančiame projektų meniu viršuje pasirenkamas mygtukas „Add New Project“, atidaromas projekto sukūrimo langas (4.4 pav.). Atsidariusiame lange įvedamas projekto pavadinimas ir paspaudžiamas mygtukas „Save“.



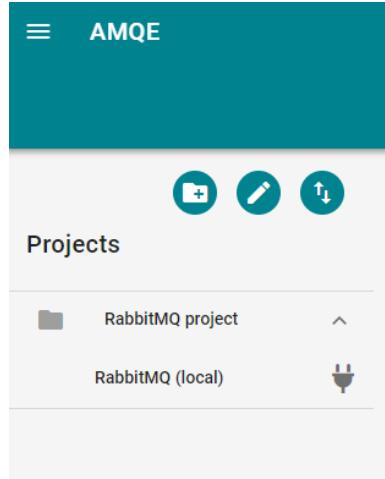
**4.4 pav.** Projekto sukūrimo langas

Naudotojas, norédamas sukurti naują pranešimų skirstytojo konfigūraciją, pasirenka kortelę „Add new broker“, o atsivérusiaiame lange suveda reikalingą konfigūracijos informaciją bei spausdžia mygtuką „Save“ (4.5 pav.).



**4.5 pav.** Pranešimų skirstytojo konfigūracijos sukūrimo langas

Sukūrus projektą bei prie jo priskyrus pranešimų skirstytojo konfigūraciją, kairėje pusėje esančiam projekto meniu galima pasirinkti pastarają konfigūraciją, taip prisijungiant prie pranešimų skirstytojo serverio (4.6 pav.).



**4.6 pav.** Projektų meniu

Prisijungus prie pranešimų skirtystojo, matomos pranešimų eilės, jų informacija (4.7 pav.). Prie kiekvienos eilės matomi mygtukai, leidžiantys atlikti tam tikrus veiksmus kiekvienai eilei: siušti naujus arba nuskaityti eilėje esamus pranešimus, išvalyti eilės turinį arba pašalinti pačią eilę.

Queues			
Queue name	All messages	Ready messages	Actions
Error Queue	2	2	
Work Queue	0	0	
aliveness-test	0	0	
<a href="#">+ Add new queue</a>			

Rows per page: 5 < 1-3 of 3 >

**4.7 pav.** Pranešimų eilių lentelės vaizdas

Matome, kad *Error Queue* pranešimų eilėje yra 2 pranešimai, todėl jie yra nuskaitomi paspaudus mygtuką „Get messages from queue“ (4.8 pav.). Atsivérus pranešimų nuskaitymo parametru langui, pasirenkamas nuskaitomų pranešimų skaičius, jų koduotė bei pranešimų nuskaitymo elgsena – ar nuskaičius pranešimus jie turi būti automatiškai grąžinti į eilę, ar ne. Paspaudžiamas mygtukas „Get messages“.

X

### Get messages from queue

**Options:**

Message count\*

The maximum number of messages to get from the queue

Encoding\*

Acknowledgement mode\*

\*indicates required field

**4.8 pav.** Pranešimų nuskaitymo parametru langas

Atsiveria pranešimų peržiūros langas, kuriamo pasirenkami norimi peržiūrėti/redaguoti pranešimai (4.9 pav.).

**Messages (2)**

[SELECT ALL](#)

Selected

[ORIGINAL](#) [JSON](#)

Message payload

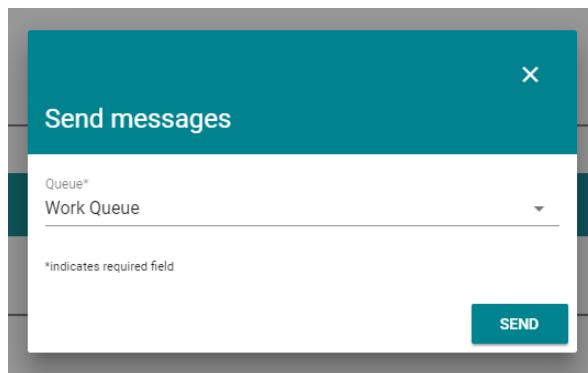
```
{
  "error": "Invalid payload JSON.",
  "payload": {
    "name": "test",
    "email": "test@test.lt"
  }
}
```

Select to view content

[EXPORT](#) [SEND](#) [REQUEUE](#)

**4.9 pav.** Pranešimų peržiūros/redagavimo langas

Pasirinkus pranešimą ir paspaudus mygtuką JSON (paverčia žinutės tekštą į JSON formatą, suformatuoją ji patogiam objekto duomenų skaitymui), matoma, kad žinutės tekstas nėra tinkamas JSON objektas, todėl redaguojant pranešimą yra ištaisomos klaidos. Pakoregavus žinutes, jos yra perkeliamos į *Work Queue* pranešimų eilę (pasirenkamas mygtukas „Send“, atsivérusiamame pranešimų siuntimo lange pasirenkama eilė *Work Queue* – 4.10 pav.).



**4.10 pav.** Pranešimų siuntimo į kitą eilę (eilės pasirinkimo) langas

Išsiuntus pranešimus, pranešimų peržiūros/redagavimo langas yra uždaromas, grįztama į eilių peržiūros langą. Pranešimų eilių lentelės apačioje pasirenkamas mygtukas „Add new queue“, atsivérusiamame lange įvedama kuriamos pranešimų eilės informacija, parenkami jos parametrai bei paspaudžiamas mygtukas „Create“ (4.11 pav.).

**Create queue**

Queue name\*

Auto delete

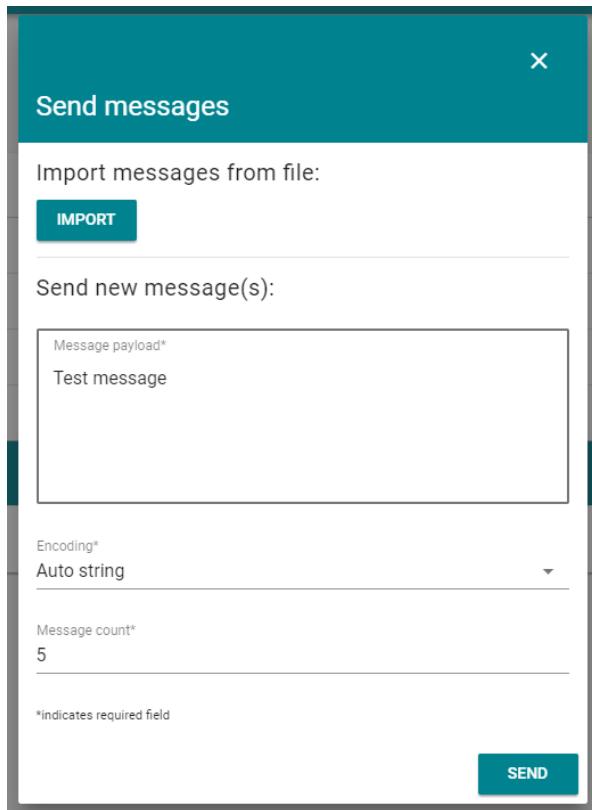
Durable

\*indicates required field

[CREATE](#)

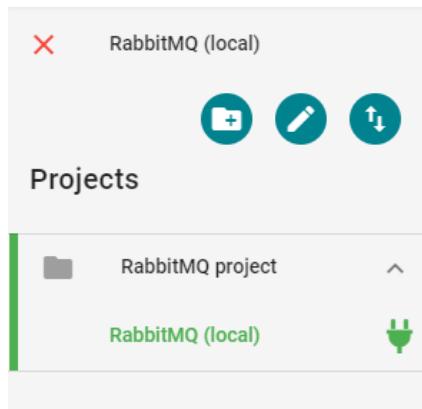
**4.11 pav.** Pranešimų eilės sukūrimo langas

Pranešimų eilių lentelėje atsiradus naujai eilei, paspaudžiamas mygtukas „Send messages to queue“, atsidariusiame pranešimų siuntimo lange įvedamas pranešimo tekstas, pasirenkama jo koduotė, įvedamas norimas išsiųsti pranešimų skaičius (4.12 pav.).



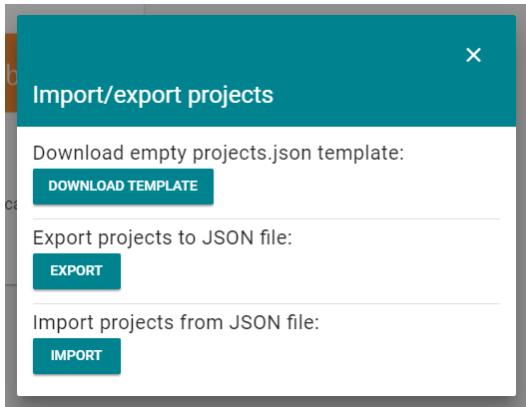
4.12 pav. Pranešimo (-ų) siuntimo langas

Naudotojui ištrynus nenaudojamą pranešimų eilę *aliveness-test* (prie eilės pasirenkamas mygtukas „Delete queue“), atsijungama nuo pranešimų skirstytojo serverio – kairėje pusėje esančiam projektų meniu pasirenkamas raudonas kryželis, esantis prie prisijungto pranešimų skirstytojo pavadinimo (4.13 pav.).



4.13 pav. Projektų meniu su atsijungimo nuo pranešimų skirstytojo serverio mygtuku vaizdas

Norėdamas eksportuoti sukurto projekto bei prie jo priskirto pranešimų skirstytojo informaciją, naudotojas projekto meniu mygtukų juosteje pasirenka mygtuką „Import/export projects“, atsivérusiam lange paspaudžiamas mygtukas „Export“ (4.14 pav.).



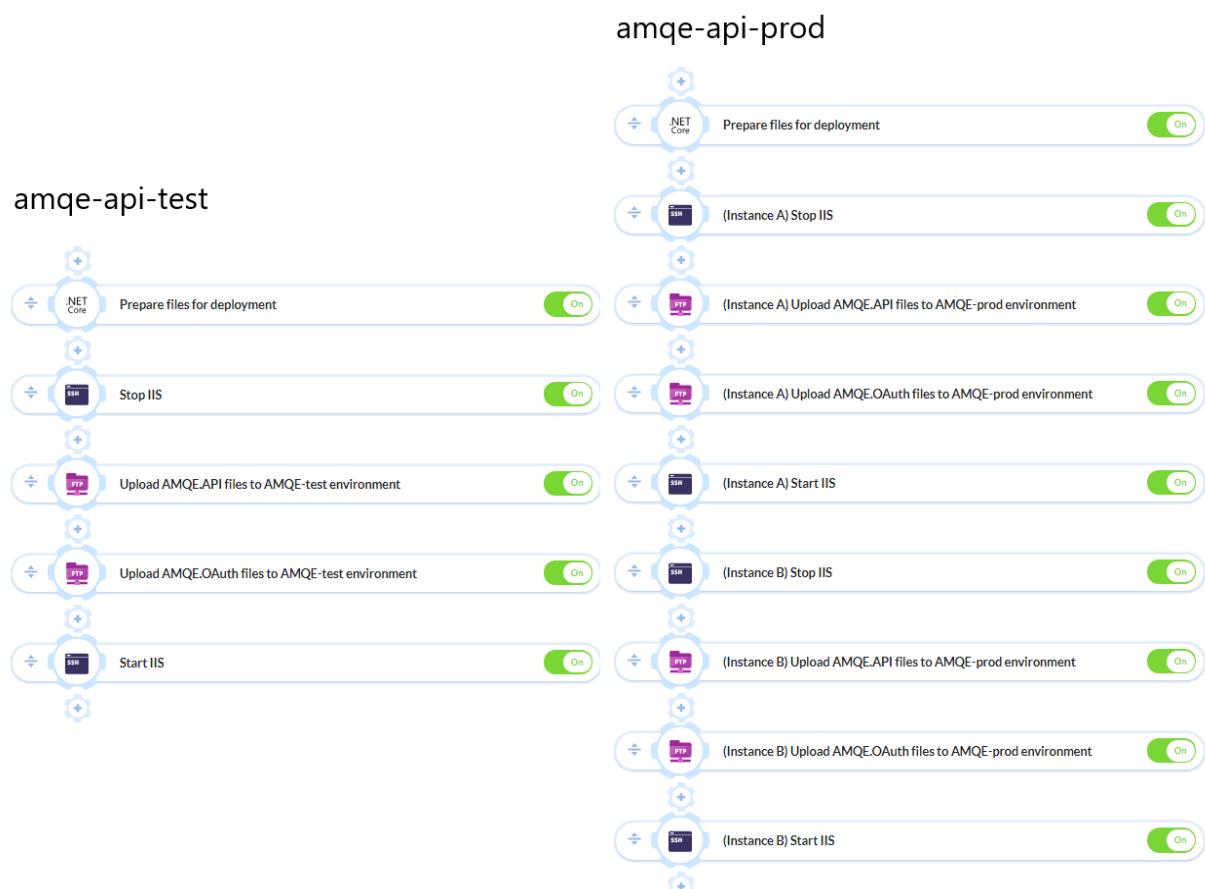
**4.14 pav.** Projektų informacijos importo/eksporto langas

Projekto informacija yra išsaugoma kompiuteryje, *projects.json* faile. Galiausiai darbas yra baigiamas, naudotojas atsijungia nuo savo paskyros viršutinėje juostoje pasirinkdamas mygtuką „Sign Out“.

### 4.3. Diegimo vadovas

Kaip jau minėta anksčiau, kuriamo įrankio (serverio pusės dalies ir saityno programos) diegimas į testavimo bei produkcijos aplinkas yra automatizuotas, naudojant *Buddy* saityno įrankį. Šis įrankis stebi projekto programos kodo pokyčius *Git* versijų kontrolės sistemos kodo talpykloje *GitHub* ir, atsiradus kodo pakeitimams, automatiškai aktyvuoją atitinkamą iš anksto paruoštą diegimo scenarijų [8]. Automatinio diegimo scenarijų sudaro atskiri žingsniai, kurie yra vykdomi iš eilės. Jei kažkuriuo diegimo žingsnio įvykdysti nepavyksta – scenarijus, o kartu ir diegimo procesas, yra nutraukiamas.

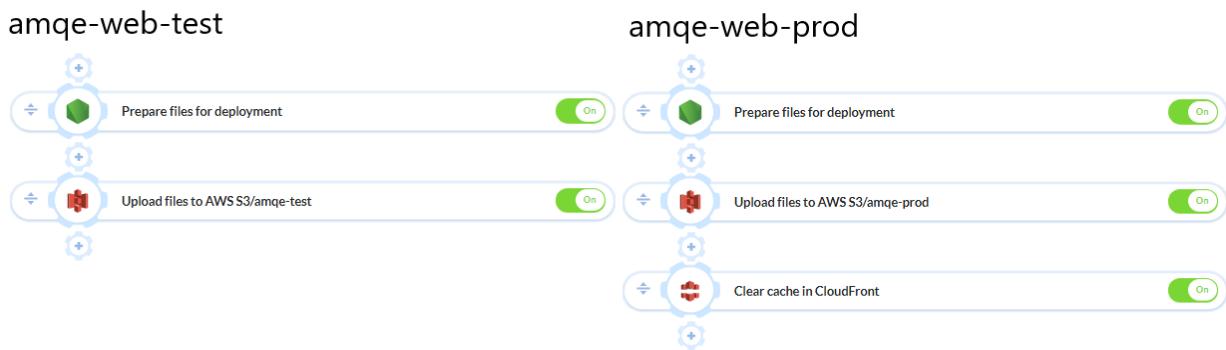
4.15 – 4.16 paveikslėliuose pateikti serverio pusės dalies kodo bei saityno programos diegimo scenarijai į testavimo ir produkcijos aplinkas.



**4.15 pav.** Serverio pusės kodo diegimo scenarijai į testavimo (kairėje) ir produkcijos (dešinėje) aplinkas

4.15 pav. matoma, kad diegimo scenarijus į testavimo aplinką susideda iš šių žingsnių: pirmiausia yra paruošiami diegimo failai (sukompiliuojamas programos kodas), tada sustabdomas veikiantis *IIS* servisas, serveryje atnaujinami API bei OAuth projektų failai ir galiausiai *IIS* servisas vėl yra paleidžiamas – diegimo procesas užbaigiamas.

Produkcijos aplinkos diegimo scenarijus iš esmės sutampa su testavimo aplinkos diegimo scenarijumi, skiriasi tik tai, kad produkcijos aplinkoje įrankio serverio pusės dalies kodas yra diegiamas į dvi serverio kopijas (pirmiausia į vieną, po to į kitą), tačiau diegimo žingsniai yra tie patys. Toks scenarijus produkcijos aplinkoje leidžia atnaujinti šios aplinkos kodą be jokio įrankio veikimo sustabdymo – įrankio posistemui serveriai yra sustabdomi ir atnaujinami po vieną, todėl apkrovos balansavimo servisas įrankio naudotojų užklausas nukreipia į tuo metu veikiantį serverį.



**4.16 pav.** Saityno programos kodo diegimo scenarijai į testavimo (kairėje) ir produkcijos (dešinėje) aplinkas

4.16 pav. matome, kad saityno programos diegimo scenarijus į testavimo aplinką susideda iš 2 žingsnių – programos failų paruošimo diegimui (kodas yra sukompiliuojamas bei suspaudžiamas, kiek galima labiau sumažinant programos failų apimtį) ir jų perkėlimo į AWS *S3* konteinerį. Produkcijos aplinkoje pirmieji diegimo žingsniai sutampa su esančiais testavimo aplinkoje, tačiau po jų įvykdymo produkcijos aplinkoje papildomai dar yra išvalomas AWS *CloudFront* paslaugos podėlis, taip atnaujinant programos failus fiziniuose AWS serveriuose įvairiuose pasaulio regionuose, iš kurių saityno programos failai yra persiunčiami įrankio naudotojams.

Automatiniai diegimo scenarijai į testavimo aplinkas yra aktyvuojami tada, kai pastebimi bet kokie kodo pokyčiai atitinkamą *Github* saugykloje esančių projektų *master* šakoje. Automatinis diegimas į produkcijos aplinkas vyksta tada, kai sukuriama nauja projekto šaka, pavadinimu *release-x.y* kur *x.y* atitinka naujos įrankio versijos numerį.

## 5. REZULTATŪ APIBENDRINIMAS IR IŠVADOS

Atlikus darbą prieita prie šių išvadų:

1. Reikalavimų bei konkurentų analizės metu nustatyta, kad rinkoje yra mažai universalų pranešimų eilių valdymo įrankių, palengvinančių programuotojų ir testuotojų darbą stebint bei koreguojant pranešimų eiles bei jose esančią informaciją. Dėl šios priežasties nuspręsta kurti universalų pranešimų eilių valdymo įrankį, palaikantį *RabbitMQ* bei *ActiveMQ* pranešimų eilių realizacijas.
2. Įrankio projektavimo metu įsigilinta į *RabbitMQ* bei *ActiveMQ* pranešimų skirstytojų realizacijas, išsiaiškinti jų panašumai bei trūkumai. Tai padėjo lengviau suprojektuoti, o vėliau ir realizuoti įrankio serverio pusės dalį, palaikyti vientisą programos kodo struktūrą.
3. Darbo metu pritaikytas iteracinis programinės įrangos kūrimo procesas tiko šio įrankio projektavimui bei realizacijai, nes palaipsniui renkami reikalavimai atskiroms įrankio dalims bei jų realizacija, testavimas leido geriau įsigilinti į kiekvieną įrankio komponentą atskirai, taip identifikuojant galimas realizacijos klaidas ateityje.
4. Įrankio serverio pusės dalies projektavimo metu įsigilinta į sluoksniuotosios architektūros (*angl. n-tier architecture pattern*) šabloną, priklausomybės injekcijos (*angl. dependency injection*), kūrėjo (*angl. builder*) bei strategijos (*angl. strategy*) projektavimo pavyzdžius. Išnagrinėtieji šablonai ir pavyzdžiai buvo sėkmingai pritaikyti realizuojant įrankio serverio pusės dalį, padėjo struktūrizuoti programos kodą bei pagerino jo skaitomumą, išplečiamumo galimybes.
5. Pasiektas 99% įrankio serverio pusės valdiklių kodo padengimas testais ne tik užtikrina įrankio posistemų suteikiamo API patikimumą, tačiau ir leis išvengti galimai atsirandančių programos vykdymo klaidų ateityje, plečiant įrankio funkcionalumą.
6. Ateityje numatoma įrankių tobulinti ir pralėsti jo funkcionalumą: pridėti *MSMQ* pranešimų eilių palaikymą, suteikti galimybę naudotojams keisti pranešimų eilėse esančių pranešimų parametrus ir antraštės, integruoti pranešimų eilių informacijos atnaujinimą realiu laiku.
7. Sukurtas pranešimų eilių valdymo įrankis sėkmingai naudojamas *CID* įmonių grupės programuotojų bei testuotojų ir tikimasi, kad praplėtus jo funkcionalumą, jis visiškai pakeis įmonės projektų kūrimo bei testavimo metu naudojamą mokamą pranešimų eilių valdymo programinę įrangą.

## 6. LITERATŪRA

- [1] Implementing event-based communication between microservices (integration events).  
*Microsoft Docs.* [Tinkle] Microsoft, 2017 m. lapkričio 12 d. [Cituota: 2018 m. gegužės 7 d.]  
<https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/multi-container-microservice-net-applications/integration-event-based-microservice-communications>.
- [2] RabbitMQ - Management Plugin. *RabbitMQ.* [Tinkle] Pivotal, 2017 m. [Cituota: 2018 m. gegužės 7 d.] <https://www.rabbitmq.com/management.html>.
- [3] Apache ActiveMQ™ -- Web Console. *Apache ActiveMQ.* [Tinkle] Apache, 2011 m.  
[Cituota: 2018 m. gegužės 7 d.] <http://activemq.apache.org/web-console.html>.
- [4] QueueExplorer - manage queues like they are files, for MSMQ, Azure Service Bus, RabbitMQ, and ActiveMQ. *QueueExplorer.* [Tinkle] Cogin. [Cituota: 2018 m. gegužės 7 d.]  
<https://www.cogin.com/mq/index.php>.
- [5] QueueExplorer Features. *QueueExplorer.* [Tinkle] Cogin. [Cituota: 2018 m. gegužės 7 d.]  
<https://www.cogin.com/mq/qfeatures.php>.
- [6] QueueExplorer Professional. *QueueExplorer.* [Tinkle] Cogin. [Cituota: 2018 m. gegužės 7 d.]  
<https://www.cogin.com/mq/professional.php>.
- [7] *A Survey of Software Development Process Models in Software.* Iqbal H. Sarker, Faisal Faruque, Ujjal Hossen, Atikur Rahman. 11, Bangladesh : International Journal of Software Engineering and Its Applications, 2015 m., T. 9.
- [8] Development, Deployment & DevOps Automation Platform - Buddy. *Buddy tinklalapis.*  
[Tinkle] Buddy. [Cituota: 2018 m. gegužės 7 d.] <https://buddy.works/>.