Kauno technologijos universitetas

Informatikos fakultetas

Prieigos valdymas debesų paslaugų stebėjimo sistemoje

Baigiamasis magistro projektas

Benas Taurosevičius

Projekto autorius

Prof. dr. Algimantas Venčkauskas

Vadovas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Prieigos valdymas debesų paslaugų stebėjimo sistemoje

Baigiamasis magistro projektas

Informacijos ir informacinių technologijų sauga (6211BX008)

Benas Taurosevičius

Projekto autorius

Prof. dr. Algimantas Venčkauskas

Vadovas

Doc. Rimantas Kavaliūnas

Recenzentas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Benas Taurosevičius

Prieigos valdymas debesų paslaugų stebėjimo sistemoje

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Beno Taurosevičiaus, baigiamasis projektas tema „Prieigos valdymas debesų paslaugų stebėjimo sistemoje“ 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 nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

(vardą ir pavardę įrašyti ranka)	(parašas)

Taurosevičius, Benas. Prieigos valdymas debesų paslaugų stebėjimo sistemoje. Magistro baigiamasis projektas / vadovas prof. dr. Algimantas Venčkauskas; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): informatikos inžinerija, informatikos mokslai.

Reikšminiai žodžiai: prieigos valdymas, debesų kompiuterija, stebėjimas, Prometheus, Alertmanager.

Kaunas, 2020. 77 p.

Santrauka

Pagrindinė šio darbo paskirtis yra suprojektuoti ir realizuoti prieigos valdymą debesų paslaugų stebėjimo sistemoje *Prometheus Alertmanager* taip joje užtikrinant informacijos saugumo triadą CIA – konfidencialumą, vientisumą ir prieinamumą.

Darbo įvade yra detalizuojamas projekto tikslas ir išskiriami uždaviniai.

Sekančiame skyriuje yra analizuojamos debesų paslaugų stebėjimo sistemos. Toliau yra apibendrinami prieigos valdymo mechanizmai. Šio skyriaus gale yra pateiktas siekiamas sprendimas bei pateikiamos analizės išvados.

Projekto skyriuje yra detalizuotas kuriamo sprendimo projektas bei jo realizacija.

Darbe taip pat yra pristatomi su projektu atlikti eksperimentai.

Darbo gale pateikti ateities darbai ir projekto išvados.

Atlikus analizę, realizavus projekto prototipą ir atlikus eksperimentus buvo padaryta išvada, kad suprojektuota sistema įgyvendina informacijos saugumo triadą CIA debesų paslaugų stebėjimo sistemoje *Prometheus Alertmanager*.

Taurosevičius, Benas. Access Control in Cloud Monitoring System. Master's Final Degree Project/ supervisor prof. dr. Algimantas Venčkauskas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics Engineering, Computing.

Keywords: access control, cloud computing, monitoring, Prometheus, Alertmanager.

Kaunas, 2020. 77 pages.

Summary

The purpose of this paper is to design and implement an access control in a cloud monitoring system *Prometheus Alertmanger* thus ensuring the CIA information security triad – Confidentiality, Integrity and Availability.

The introductory section details the project's goal and objectives.

Cloud monitoring solutions are analyzed in the following section. Then the access control mechanisms are summarized. The intended solution and the conclusions of the analysis are presented at the end of this section.

The detailed system's model and the realization are presented in the design section.

Experiment with the created system are also presented in the paper.

Future work and project's conclusions are presented at the end of the paper.

After the analysis, realization of the project's prototype and experiments, it was concluded that the designed system implements the information security triad CIA in the cloud service monitoring system *Prometheus Alertmanager*.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Terminų sąrašas

API gateway – mikroservisų architektūros šablonas, priklausantis ribinių (angl. *boundary*) šablonų grupei. Jis veikia kaip tarpininkas tarp išorinio kliento ir patikimoje aplinkoje (angl. *trusted environment*) veikiančių programų.

AWS – *Amazon Web Services*. Vienas iš didžiausių debesų paslaugų teikėjų;

CIA – konfidencialumas, vientisumas ir pasiekiamumas (angl. *Confidentiality, Integrity and Availability*). Pagrindinės informacijos saugos sudedamosios dalys;

Cloud Native Computing Foundation (CNCF) – įgimtos debesų technologijos fondas, kuris prižiūri atviro kodo programinę įrangą ir technologijas kaip konteineriai, mikroservisai bei paslaugų tinklai (angl. *service mesh*) ir skatina jų naudojimą.

Docker – pagrindinė programinė konteinerių platforma, kuri leidžia lengvai kurti bei vykdyti konteinerius.

Docker Swarm – *Docker* klasterizavimo ir orkestravimo įrankis. Kurtas kaip atskira platforma, tačiau buvo integruotas į patį Docker variklį kaip atskiras veikimo režimas.

HTTP – *Hyper Text Transfer Protocol*. Programos lygmens duomenų apsikeitimo protokolas.

HTTPS – *Hyper Text Transfer Protocol Secure*. Saugus ir šifruotas HTTP protokolo praplėtimas naudojantis TLS.

YAML – *Yaml Ain't Markup Language*. Žmonėms draugiškas duomenų serializavimo standartas skirtas visoms programavimo kalboms.

Json – *JavaScript* objektų notacija (angl. *JavaScript Object Notation*). Tekstinis duomenų saugojimo ir apsikeitimo formatas.

Konteineris – standartizuotas programinės įrangos vienetas. Tai yra izoliacijos ir abstrakcijos sluoksnis tarp operacinės sistemos ir programinės įrangos. Konteinerių technologijos leidžia supakuoti programinę įrangą bei visas jos priklausomybes (angl. *dependency*) ir bibliotekas (angl. *library*) į savarankiškus modulį.

OIDC – OpenID Connect. Paprastas identiteto sluoksnis ant OAuth2 standarto. Jis leidžia klientams patvirtinti galutinio vartotojo tapatybę išvengiant vartotojų slaptažodžių saugojimo ir valdymo.

OpenAPI standartizuotas REST API specifikavimo formatas.

Orkestracija – automatinis programinės įrangos paslaugų konfigūravimas, koordinavimas ir valdymas.

Perspėjimas (angl. *alert*) – stebimų duomenų nukrypimas nuo nustatytų rėžių stebėjimo sistemoje

Pranešimas (angl. *notification*) – Vieno ar kelių pranešimų grupė, kurie yra siunčiami gavėjui tam tikru pranešimų kanalu: e. paštu, SMS žinute ir t. t.

RBAC – rolėmis grįstas prieigos valdymas (angl *Role Based Access Control*)

REST –REpresentational State Transfer. Architektūrinis interneto paslaugų kūrimo stilius, veikiantis HTTP protokolu.

Reverse proxy – atvirkštinis įgaliotasis serveris. Serveris, kuris priešakyje kitų interneto programų serverių ir persiunčia klientų užklausas tiems serveriams. Dažniausiai naudojami krūvio padalinimui, TLS šifravimui ir/arba kaip podėlis.

TLS – *Transport Layer Security*, nebenaudojamo SSL protokolo įpėdinis. Kriptografinis protokolas skirtas užtikrinti ryšio kanalo saugumą kompiuterių tinkle.

URL - universalus ištekliaus adresas (angl. *Uniform Resource Locator*). Ištekliaus aprašymo metodas tinkle.

VCS – versijų valdymo sistema (angl. V*ersion Control System*). Sistema leidžianti saugoti failus, jų versijas ir metaduomenis apie perėjimus tarp versijų. Pavyzdžiai: git, subversion, mercurial.

Webhook – būdas perspėti antrą sistemą apie įvykį pirmoje sistemoje naudojant HTTP ar protokolą.

Įvadas

Didėjant debesų, jų teikiamų paslaugų kiekiui ir sudėtingumui, debesų valdymas darosi vis sudėtingesnis. Šį procesą palengvina debesų paslaugų stebėjimo sistemos, kurios leidžia išgauti debesies būklę iš jo teikiamų komponentų ir centralizuotai ją peržiūrėti. Tačiau stebėjimas yra tik dar viena paslauga, kuri privalo būti saugi.

Kiekvienas debesų stebėjimo sprendimas į saugą žvelgia kitaip. Viena iš labiausiai pažengusių ir modernių atviro kodo stebėjimo sistemų *Prometheus* [1], prieigos valdymo yra visiškai neįgyvendinusi [2].

Šio darbo tikslas – suprojektuoti ir realizuoti programą *Alertmanager Proxy*, kuri įgyvendins prieigos valdymą debesų paslaugų stebėjimo sistemos *Prometheus* dalyje *Alertmanager*, kuri yra atsakinga už pranešimų išsiuntimą.

Šiam tikslui pasiekti buvo iškelti uždaviniai:

1. atlikti debesų paslaugų, jų stebėjimo sistemų ir saugos problemų analizę;
2. nustatyti kaip galima įdiegti prieigos valdymą į *Prometheus Alertmanager*;
3. įvertinti kuriamos sistemos reikalavimus ir suprojektuoti kuriamą įrankį;
4. realizuoti įrankio prototipą;
5. kokybiškai ir kiekybiškai įvertinti sukurtą prototipą.

Pirmame darbo skyriuje pateikta debesų paslaugų, stebėjimo sistemų ir prieigos valdymo analizė bei siekiamo sprendimo aprašymas. Sekančiame skyriuje pateiktas kuriamo sprendimo projektas ir aprašyta prototipo realizacija. Trečiame skyriuje pateikta prototipo analizė – testavimas bei atlikti eksperimentai. Dokumento pabaigoje pateikti ateities darbai bei padarytos išvados.

1. Debesų kompiuterijos stebėjimo sistemų analizė

1.1. Problema ir analizės tikslas

Debesų paslaugomis dažniausiai naudojasi vidutinio ir didelio dydžio organizacijos, kuriose yra daug padalinių, komandų ir darbuotojų. Organizacijos dažnai naudoja keletą įvairių debesų paslaugų vienu metu, taip pat skirtingos komandos gali naudoti tas pačias paslaugas. Viena iš tokių paslaugų yra debesų stebėjimas. Kiekvieno padalinio ir kiekvienos komandos darbuotojas dažniausiai turi skirtingas pareigas bei atsakomybes, taip pat turi ir skirtingą prieigą prie konfidencialios informacijos. Dėl to yra reikalinga prieigos kontrolė prie resursų, kurie yra pasiekiami naudojant debesų paslaugų stebėjimo sistemas.

Vienoje iš moderniausių atviro kodo sistemų *Prometheus* prieigos valdymas yra neįgyvendintas, todėl šios analizės tikslas – nustatyti kaip galima įdiegti prieigos valdymą į *Prometheus*, išanalizuojant debesų kompiuterijos stebėjimo sistemas ir išnagrinėjant prieigos valdymo metodus.

1.2. Debesų stebėjimo analizė

1.2.1. Debesų kompiuterija

Pagal NIST [3] debesų kompiuterija yra modelis, leidžiantis visur prieinamą, patogią ir pagal poreikį prieinamą tinklo prieigą prie bendro konfigūruojamų kompiuterinių išteklių (pvz., tinklų, serverių, saugyklų, programų ir paslaugų), kuriuos galima greitai sukurti ir sunaikinti naudojant minimalias valdymo pastangas ar paslaugų teikėjo įsikyšimą. Šį debesies modelį sudaro penkios pagrindinės charakteristikos, trys paslaugų modeliai ir keturi diegimo modeliai.

Charakteristikos:

1. Apsirūpinimas pagal pareikalavimą (angl. *on-demand self-service*)
2. Plati tinklo prieiga (angl. *broad network access*)
3. Išteklių telkimas (angl. *resource pooling*)
4. Spartus elastingumas (angl. *rapid elasticity*)
5. Matuojamas aptarnavimas (angl. *measured service*)

Paslaugų modeliai:

1. *Software as a service*
2. *Platform as a Service*
3. *Infrastructure as a Service*

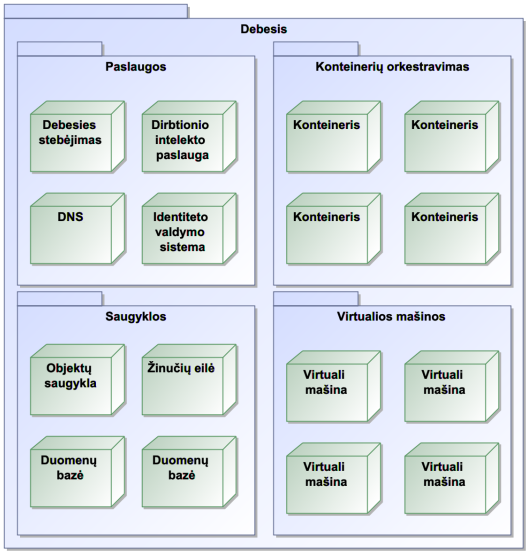
Diegimo modeliai:

1. Privatus debesis
2. Bendruomenės debesis
3. Viešas debesis
4. Hibridinis debesis

Įgimtos debesų technologijos (angl. *cloud native technologies*) suteikia organizacijoms galimybę kurti ir valdyti programas šiuolaikinėje, dinamiškoje aplinkoje – debesyse. Konteineriai, paslaugų tinkleliai (angl. *service meshes*), mikro paslaugos (angl. *micro services*), nekintama (angl. *immutable*)

infrastruktūra ir deklaratyvūs API yra šio požiūrio pavyzdžiai [4]. Šie metodai įgalina silpnai sujungtas (angl. *loosely coupled*) sistemas, kurios yra atsparios, valdomos ir stebimos. Kartu su tvirta automatizacija tai inžinieriams leidžia dažnai ir nuspėjamai atlikti didelio poveikio pakeitimus su minimalia našta.

Organizacijos debesų infrastruktūra gali būtis sudaryta iš kelių paslaugų: virtualių mašinų, konteinerių orkestracijos sprendimų (angl. *container orchestration*), duomenų bazių ir kitų paslaugų. Tokia infrastruktūra pateikta paveiksle 1.1. Debesyje virtualių mašinų ar konteinerių kiekis gali dinamiškai keistis, nuo to taip pat keičiasi ir įvairių sistemų patiriamos apkrovos.



1.1 pav. Organizacijos debesų kompiuterijos naudojamos paslaugos

Aptariant šiuos debesų paslaugų teikiamus privalumus reikia atsižvelgti ir į trūkumus. Dėl spartaus elastingumo ir didelės dinamikos darosi sudėtinga turėti bendrą vaizdą apie veikiančios sistemos būklę. O neturint informacijos apie sistemos būklę yra neįmanoma užtikrinti paslaugų pasiekiamumo ar informacijos konfidencialumo.

1.2.2. Debesų stebėjimas

Kaip teigia IBM [5], „debesų stebėjimas yra sudarytas iš veiklų ir procesų, kurios yra skirtos debesų paslaugų ir programų analizei, sekimui ir valdymui“.

Didžiąją dalį šių veiklų ir procesų galima automatizuoti naudojant debesų paslaugų stebėjimo sistemas.

Debesų paslaugų stebėjimo sistema turi atlikti šias funkcijas:

1. rinkti informaciją apie sistemą realiu laiku;
2. įgalinti sistemos perspėjimus įvykus tam tikroms, nustatytoms, sąlygoms;
3. siųsti pranešimus įvykus perspėjimams;
4. grafiškai pateikti surinktą informaciją;
5. generuoti ataskaitas apie atskirus sistemos komponentus ar bendrą sistemą už tam tikrą laikotarpį.

Turint sistemą, kuri atlieka šias funkcijas galima:

1. užtikrinti teikiamų paslaugų pasiekiamumą;
2. informuoti atsakingus asmenis apie esamus ir galimus defektus;
3. aptikti našumo ir prieinamumo problemas prieš joms pasiekiant galutinį vartotoją;
4. greičiau spręsti iškilusias problemas, nes galima peržiūrėti sistemos būklę ir greičiau nustatyti pagrindinę problemos priežastį;
5. atskleisti debesų infrastruktūrų problemines zonas;
6. nustatyti tinkamo dydžio debesies infrastruktūrą, kad būtų galima efektyviai palaikyti programų darbo krūvius;
7. gerinti debesų programų ir tinklų saugumą;
8. sumažinti netikėtų debesų kompiuterijos paslaugų kainų išaugimo galimybę dėl padidinto architektūros matomumą;
9. automatinį mastelio keitimą (angl. *scaling*).

Taigi gera stebėjimo sistema didina organizacijos produktyvumą šiais aspektais:

1. pagerina organizacijos aparatinės įrangos naudojimą kontroliuojant jos gerą veikimą jeigu yra naudojamas privatus ar hibridinio debesies modelis;
2. užkerta kelią incidentams, o kai šie incidentai įvyksta, jie nustatomi greičiau, o tai taupo laiką ir pinigus;
3. vengiant paslaugos nutrūkimo ar sutrumpinant jo sprendimo laiką, pagerės ir organizacijos įvaizdis, ir klientų aptarnavimas;
4. tinkamo sistemų veikimo kontrolei skiriama mažiau laiko, daugiausia todėl, kad stebėjimo sistema tuo pasirūpina pati.

1.3. Debesų stebėjimo sistemų palyginimas

Yra daug įvairių debesų stebėjimo sistemų, kiekviena iš jų veikia kitokiu principu ir kitaip įgyvendina prieigos kontrolę, kelias populiariausiai apžvelgsime šiame poskyryje.

1.3.1. Amazon CloudWatch

Amazon CloudWatch teikia stebėjimo paslaugą programoms ir resursams, kurie veikia *Amazon Web Services (AWS)* platformoje. Sistema automatiškai integruojasi su *AWS* teikiamomis paslaugomis, todėl yra paprastas pradinis sistemos paleidimas.

Identiteto valdymas yra užtikrinamas naudojant *AWS IAM* politikas, todėl yra labai lankstus ir galingas [6].

Naudojant šią sistemą, galima siųsti pranešimus keturiais būdais:

1. kaip e. laišką;
2. naudojant HTTP Webhook;
3. iškviečiant *AWS Lambda* funkciją;
4. įdedant į *AWS SQS* eilę.

Amazon CloudWatch programa yra teikiama kaip paslauga ir yra uždaro kodo. Duomenys saugomi iki 15 mėnesių [7].

1.3.2. Nagios

Nagios yra atviro kodo sistemų stebėjimo platforma kuri buvo pritaikyta debesų kompiuterijos stebėjimui.

Nagios teikia tik vieną standartinį pranešimų kanalą – e. paštą [8], tačiau sistemą galimą praplėsti ir pranešimus siųsti kitokiais kanalais.

Taip pat ši programa yra skirta įvertinti tik esamą stebimos sistemos būklę, todėl *Nagios* nekaupia istorinių duomenų [9].

Nagios pasižymi dideliu potencialu išplėtimui, todėl šios sistemos diegimas ir konfigūravimas yra sudėtingas.

Šis įrankis yra konfigūruojamas statiškai, todėl nėra pritaikytas efektyviai veikti dinamiškai besikeičiančiose debesų kompiuterijos aplinkose [10].

Nagios autentikavimui įgyvendinti naudoja *Apache Httpd CGIauth* [11], o autorizacija yra realizuota principu ACL su grupėmis.

1.3.3. Prometheus

Prometheus yra atvirojo kodo sistemų stebėjimo ir perspėjimo įrankių rinkinys, iš pradėtas kurti 2012 metais įmonės *SoundCloud*. Dabar tai yra atvirojo kodo projektas, prižiūrimas nepriklausomai nuo bet kurios įmonės, nes *Prometheus* 2016 metais prisijungė prie *Cloud Native Computing Foundation* kaip antrasis pagrindinis projektas po *Kubernetes*.

Pagrindinės *Prometheus* savybės:

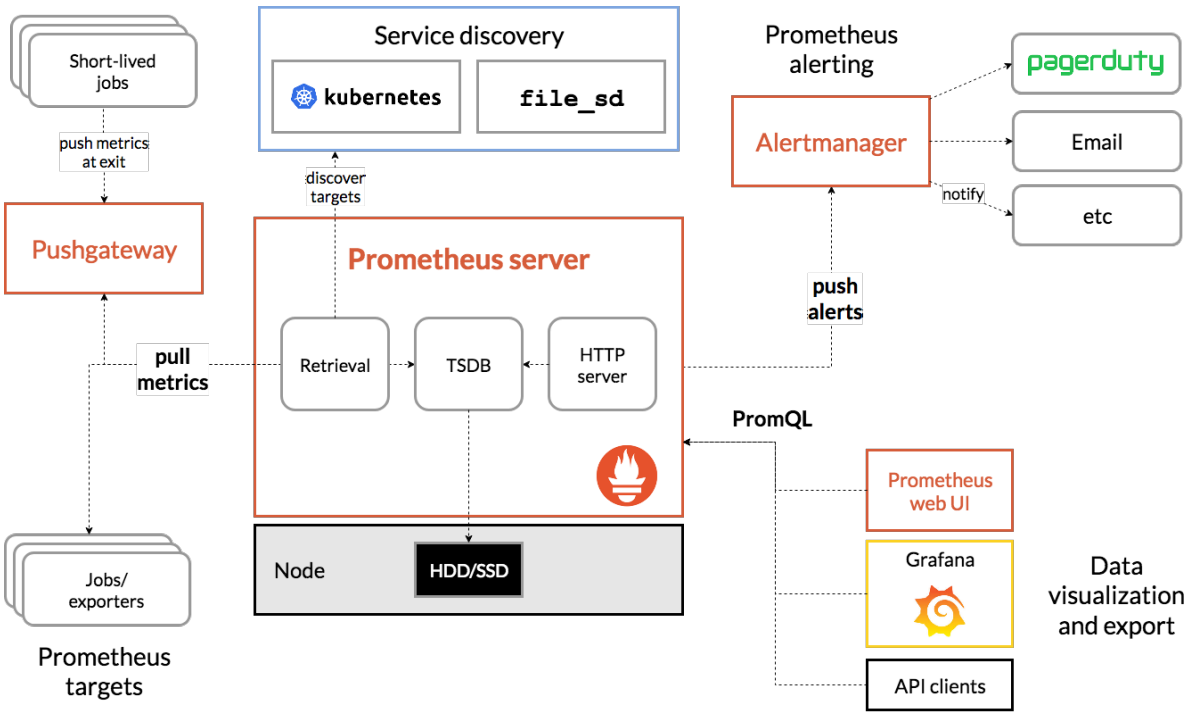
1. daugiamatis duomenų modelis su laiko eilučių duomenimis, identifikuojamais pagal metrikos pavadinimą ir rakto/vertės poras;
2. *PromQL*, lanksti užklausų kalba, skirta išnaudoti šį dimensialumą;
3. laiko eilučių rinkimas vyksta naudojant traukimo (angl. *pull*) modelį HTTP protokolu;
4. laiko eilučių surinkimas naudojant stūmimo (angl. *push*) modelį yra taip pat palaikomas per tarpinę programą *Pushgateway*;
5. metrikas atiduodantys taikiniai gali būti aptinkami naudojant paslaugų aptikimą (angl. *service discovery*) arba statinę konfigūraciją;
6. keli grafikų ir prietaisų skydelių (angl. *dashboard*) atvaizdavimo būdai.

Visi *Prometheus* sistemos komponentai gali dirbti didelio pasiekiamumo (angl. *availability*) režimu.

Prometheus buvo kurta siekiant patikimumo, kad tai būtų sistema, į kurią galima eiti kitų paslaugų sutrikimo metu ir, kad ją naudojant būtų galima greitai diagnozuoti problemas. Kiekvienas *Prometheus* serveris yra autonominis, nepriklausomas nuo tinklo saugyklos ar kitų nuotolinių paslaugų. Šia sistema galima pasikliauti tada, kai sugenda kitos infrastruktūros dalys. Taip pat sistema *Prometheus* yra paprastai diegiama.

Vienas iš didžiausių *Prometheus* sistemos trūkumų yra identiteto ir prieigos valdymo nebuvimas. *Prometheus* kūrėjai daro prielaidą, kad visi, kas turi prieiga prie *Prometheus* HTTP sąsajos, gali matyti visą informaciją [2]. Šis trūkumas yra sutinkamas ne tik pačioje *Prometheus* programoje, bet ir pagalbinėse *Prometheus* sistemos programose *Pushgateway* ir *Alertmanager*.

Architektūra. 1.2 pav. yra pateikta *Prometheus* architektūros diagrama.



1.2 pav. Bendrinė Prometheus architektūra [12]

Prometheus programa, kuri yra pavaizduota diagramos centre, aptinka visus metrikas galinčius atiduoti taikinius naudodama paslaugų aptikimą ar statinę konfigūraciją ir iš jų renka metrikas. Taip pat metrikos yra renkamos ir iš *Pushgateway* programos, į kurią metrikas siunčia trumpalaikiai (angl. *short-lived*) procesai, kaip „Cron užduotys“.

Prometheus nustatytu metu vykdo aprašytas taisykles. Šios taisyklės yra dviejų tipų: įrašymo ir perspėjimo. Įrašymo taisyklės nurodo kaip agreguoti ir saugoti naujas laiko eilutes iš esamų duomenų, o perspėjimų taisyklės nurodo kokioms sąlygoms įvykus reikia generuoti perspėjimus ir juos siųsti į *Alertmanager*.

Surinktiems duomenims vizualizuoti gali būti naudojami *Prometheus* tinklo vartotojo sąsaja (angl. *web UI*), programa *Grafana* ar kiti API vartotojai.

Alertmanager

Alertmanager yra programa, kuri priima perspėjimus iš *Prometheus* sistemos, juos apdoroja ir išsiunčia nustatytiems gavėjams kaip pranešimus [13].

Alertmanager taip pat teikia ir nutildymo funkciją – galima išjungti pranešimų generavimą įvykus tam tikriems perspėjimams pagal nurodytas taisykles. Šie nutildymo egzemplioriai yra vadinami tylomis (angl. *silences*) .

Alertmanager, kaip ir *Prometheus*, gali dirbti didelio pasiekiamumo režimu. Tokiu atveju yra paleidžiami keli *Alertmanager* egzemplioriai (angl. *instance*) kurie bendrauja tarpusavyje. Perspėjimai iš *Prometheus* turi būti siunčiami į visus paleistus *Alertmanager*, o patys *Alertmanager* bendraudami tarpusavyje atlieka pranešimų nedubliavimą (gavus keletą tokių pačių perspėjimų, yra sukuriamas tik vienas pranešimas) ir išsiuntimą.

Alertmanager gali siųsti pranešimus 9-ais kanalais [14]:

1. e. paštu;
2. į incidentų valdymo paslaugą *PagerDuty*;
3. į incidentų valdymo paslaugą *VictorOps*;
4. į pokalbių paslaugą *HipChat*;
5. į pokalbių paslaugą *Slack*;
6. į pokalbių paslaugą *WeChat*;
7. į pranešimų ir budėjimo paslaugą *OpsGenie*;
8. į pranešimų pristatymo paslaugą *Pushover*;
9. naudojant HTTP Webhook.

Didžiausia Alertmanager problema – visiškai neturi autentifikacijos bei autorizacijos mechanizmų, todėl negalima užtikrinti vartotojo peržiūrimų duomenų konfidencialumo bei vientisumo.

1.3.4. Elastic Stack

Elastic Stack yra atviro kodo įrankių komplektas sudarytas iš *Elasticsearch*, *Logstash* ir *Kibana* [15].

Šio komplekto pagrindinis panaudos atvejis yra žurnalo įrašų centralizavimas ir analizė, tačiau šis komplektas taip pat palaiko ir metrikų, programų veikimo laiko (angl. *uptime*), tinklo ir trasavimo (angl. *tracing*) duomenų surinkimą ir analizę.

Visi duomenys yra saugomi *Elasticsearch* paieškos ir analitikos variklyje, už duomenų atvaizdavimą yra atsakinga *Kibana*, o *Logstash* yra duomenų priėmimo, apdorojimo ir saugojimo į *Elasticsearch* kanalas.

Elastic Stack pranešimus gali siųsti 4-iais kanalais [16]:

1. e. paštu;
2. į incidentų valdymo paslaugą *PagerDuty*;
3. į pokalbių paslaugą *Slack*;
4. naudojant HTTP Webhook.

Nemokama versija teikia labai ribotą funkcionalumą. Identiteto valdymas ir pranešimų siuntimas prieinamas tik mokamoje versijoje.

Programoje Kibana autentifikacijai galima naudoti 7 būdus [17]:

1. Bazinė autentifikacija – naudoja *Elasticsearch* sukonfigūruotas autentifikacijos taisykles;
2. Žetono autentifikacija – taip pat naudoja *Elasticsearch* sukonfigūruotas autentifikacijos taisykles, tačiau kitokį duomenų perdavimo būdą;
3. Viešojo rakto Infrastruktūros autentifikacija naudojant klientų TLS sertifikatus;
4. SAML vieną prisijungimą;

5. OpenID Connect vieną prisijungimą;
6. Kerberos vieną prisijungimą;
7. HTTP autentifikaciją.

Autorizacija mechanizmas *Elastic Stack* sistemoje yra įgyvendintas RBAC principu [18].

1.3.5. Palyginimas

Lyginamoji analizė atlikta įvertinus šiame poskyryje pateiktas sistemas dešimtbalėje sistemoje pagal dešimt kriterijų:

- **Analizė realiu laiku** – kaip greitai duomenys atsiranda stebėjimo sistemoje nuo jų įvykimo – 9 balai – iki minutės , 10 balų – iki sekundės.
- **Sistemos perspėjimai** – vertinamas perspėjimų generavimo lankstumas. 0 balų už perspėjimų nebuvimą, 10 balų už generavimą naudojant lanksčias taisykles bei perspėjimų aptikimą naudojant mašininį mokymąsi.
- **Pranešimai** – vertinamas pranešimų siuntimo kanalų kiekis. Po vieną balą už kanalą.
- **Grafinis atvaizdavimas ir ataskaitų generavimas** – vertinamas sistemų grafinio atvaizdavimo galimybės. 5 balai už esamos situacijos pateikimą, 8 balai už duomenų pateikimą naudojant grafikus, 10 balų už lankstų duomenų pateikimą naudojant įvairius grafikus ir diagramas už pasirenkamą laikotarpį.
- **Istorinių duomenų prieinamumas** – vertinama kaip ilgai sistemoje yra prieinami istoriniai duomenys. Skiriamas 1 balas už esamos situacijos pateikimą, 5 balai už ribotą duomenų prieinamumą tam tikram laikotarpiui su duomenų rezoliucijos sumažinimo (angl. *downsampling*) taisyklėmis, 5 balai už neribotą duomenų pasiekiamumą be duomenų rezoliucijos sumažinimo taisyklių, 8 balai už neribotą duomenų prieinamumą tam tikram laikotarpiui su duomenų rezoliucijos sumažinimo taisyklėmis.
- **Diegimo ir priežiūros paprastumas** – vertinamas sistemos diegimo ir priežiūros sudėtingumas. 0 balų – visą diegimo darbą ir konfigūravimo darbą reikia atlikti rankiniu būdu, nėra jokių automatizavimo priemonių, 10 balų – sistema yra įdiegiama ir konfigūruojama automatiškai.
- **Išplečiamumas ir integracija** – vertinama keliais būdais sistemą galima išplėsti bei integruoti su kitomis sistemomis. Skiriami 5 balai už esamų integracijų kiekį ir 5 balai už galimybę programuoti integracijas naudojant technologijas kaip HTTP Webhook ar kt.
- **Kodo atvirumas** – 0 balų – sistema yra uždaro kodo, 5 balai – dalis sistemos yra atviro kodo, 10 balų - visa sistema yra atviro kodo.
- **Kaina** – kiek kainuoja pilnas sistemos funkcionalumas: 0 balų – sistema neturi nemokamos versijos, 10 balų – sistema visiškai nemokama.
- **Prieigos valdymas** – 0 balų – sistema visiškai neturi autentifikavimo ir autorizacijos mechanizmų, 10 balų – mechanizmai labai lankstūs.

1 lentelėje pateikiamas palyginimo rezultatas.

1 lentelė Debesų paslaugų stebėjimo sistemų palyginimas

Kriterijus	Amazon CloudWatch	Nagios	Prometheus	Elastic Stack
Analizė realiu laiku	10	10	9	9

Sistemos perspėjimai	8	8	9	10
Pranešimų kanalai	4	1	9	4
Grafinis atvaizdavimas ir ataskaitų generavimas	8	5	10	10
Istorinių duomenų prieinamumas	5	1	5	8
Diegimo ir priežiūros paprastumas	7	2	8	6
Išplečiamumas ir integracijos	5	10	10	5
Kodo atvirumas	0	10	10	8
Kaina	3	10	10	1
Prieigos valdymas	10	3	0	5
Viso	**60**	**60**	**80**	**66**

Pagal rezultatus matome, kad *Prometheus* sistema surinko daugiausiai taškų nors visai neturi identiteto valdymo, todėl buvo nuspręsta sukurti įrankį, kuris padės išspręsti šią problemą *Prometheus* programoje.

1.4. Prieigos valdymo analizė

Prieigos valdymas susideda iš autentifikacijos ir autorizacijos mechanizmų. Šių mechanizmų apibendrinimas pateikiamas šiame poskyryje.

1.4.1. Autentifikacijos mechanizmų analizė

Mikroservisų architektūroje autentifikaciją galima įgyvendinti keliais būdais:

1. **Skirtingos programos – skirtingi vartotojai.** Kiekviena programa turi duomenų bazę kurioje yra saugomi vartotojai ir taip pat turi savo logiką pagal kurią autentifikuoja vartotoją;
2. **Skirtingos programos – tie patys vartotojai, skirtingi prisijungimo mechanizmai.** Viena centrinė vartotojų duomenų bazė, kiekviena programa turi savo logiką, pagal kurią jungiasi prie duomenų bazės ir autentifikuoja vartotojus;
3. **Skirtingos programos – tie patys vartotojai, tas pats prisijungimo mechanizmas.** Viena centrinė standartizuota vartotojų identiteto valdymo sistema, kiekviena programa turi standartinius logiką ir protokolą (OpenID Connect, LDAP, SAML), pagal kuriuos jungiasi prie duomenų bazės ir autentifikuoja vartotojus;
4. **Skirtingos programos – tie patys vartotojai, tas pats centralizuotas prisijungimo mechanizmas.** Viena centrinė standartizuota vartotojų bazė, kurioje saugomi vartotojai, o API prieigos vartų (angl. *API gateway*) programa atlieka autentifikavimo funkcionalumą. Naudojant šį sprendimą programoje nebereikia autentifikavimo logikos, nes yra pasitikima, kad iki programos atkeliavusi užklausa jau yra autentifikuota [19]. Šis mechanizmas yra paremtas HTTP užklausų perėmimų ir modifikavimu, todėl jis palaiko daugumą autentifikacijos standartų kaip OpenID Connect (OIDC), LDAP, HTTP basic auth ir kt. Taip pat šis autentifikacijos mechanizmas naudoja HTTP antraštes (angl. *headers*) vartotojo informacijos perdavimui į programą, todėl programa gali identifikuoti kurio vartotojo užklausą apdoroja ir pagal tai pati spręsti ar užklausą autorizuoti, atmesti ar dalinai autorizuoti.

1.4.2. Autorizacijos mechanizmų analizė

Prieigos kontrolės sąrašas

Prieigos kontrolės sąrašo (angl. *ACL – access control list*) mechanizmas gali būti apibūdinamas, kaip leidimų sąrašas prie objekto – t. y. kiekvienas objektas turi turėti sąrašą subjektų, kurie gali juo operuoti.

Savarankiška prieigos kontrolė

DAC (angl. *Discretionary Access Control*) mechanizmas, apriboja subjekto prieigą prie objekto, pagal subjekto identitetą ir / arba grupes kurioms jis priklauso. Savarankiškumas ateina iš to, kad subjektai gali perduoti savo teises kitiems subjektams (kartais ir netiesiogiai). Šis mechanizmas yra daugiausiai naudojamas, kai kiekvienas objektas turi savo savininką – subjektą.

Rolėmis grįsta prieigos kontrolė

RBAC (angl. *role based access control*) yra centralizuotas autorizacijos mechanizmas, kurį sudaro leidimai ir juos apjungiančios grupės – rolės. Rolės yra priskiriamos subjektams (vartotojams) ir sprendimo priėmimo taške yra tikrinama, ar subjektas turi leidimą atlikti veiksmą su objektu įvertinant visas subjekto grupes.

Atributais grįsta prieigos kontrolė

Atributais grįsta prieigos kontrolė (angl. *ABAC – attribute based access control*) apibrėžia prieigos kontrolės mechanizmą, pagal kurį prieigos teisės vartotojams suteikiamos naudojant strategijas, kurios kartu sujungia atributus [20]. Politika gali naudoti bet kokio tipo atributus (vartotojo atributus, išteklių atributus, objekto, aplinkos atributus ir t. t.). Šis mechanizmas surenka atributus apie subjektą, subjekto grupes, objektą, objekto grupes, vykdymo aplinką ir kt. Ir pagal saugos politikoje aprašytą logiką nusprendžia ar subjektas yra autorizuotas atlikti veiksmą su objektu.

1.5. Siekiamas sprendimas

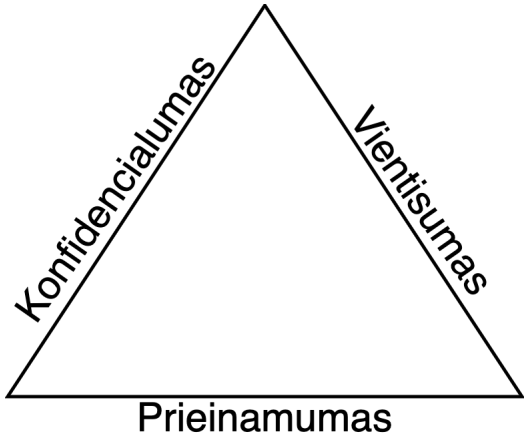
Įvertinus keturias stebėjimo sistemas buvo nuspręsta sukurti įrankį, kurį būtų galima integruoti į *Prometheus* sistemą ir joje užtikrinti CIA triadą – konfidencialumą, vientisumą ir prieinamumą.

Prometheus sistema yra labai plati, todėl pasirinkta sukurti įrankį, kuris įgyvendins CIA triadą *Prometheus Alertmanager* programoje.

Bus sukurta *Alertmanager Proxy* programa, kuri veiks kaip nematomas tarpininkas tarp vartotojo ir *Prometheus Alertmanager* sistemos.

Programa *Alertmanager* yra pasiekiama HTTP protokolu ir turi *OpenAPI* standartizuotą dokumentaciją. *Alertmanager Proxy* perims užklausas, einančias tarp kliento ir *Alertmanager*, jas autorizuos pagal nustatytas taisykles. Kai kurioms užklausoms *Alertmanager Proxy* gali pateikti tik dalį rezultatų.

CIA triada (1.3 pav.), yra sudaryta iš trijų komponentų – prieinamumo, konfidencialumo ir vientisumo. Šios triados pagrindinis tikslas yra užtikrinti informacijos saugumą.



1.3 pav. CIA triada

Kaip minėta skyrelyje 1.3.3, *Alertmanager* programoje jau yra įgyvendintas didelis prieinamumas, todėl šio darbo tikslas yra įgyvendinti *Alertmeneger* duomenų konfidencialumą ir vientisumą.

Konfidencialumo kriterijus bus įgyvendintas naudojant RBAC autorizaciją ir pačios programos diegimas panaudojant motociklo priekabos (angl. *sidecar*) šabloną. Vientisumą užtikrins perduodamų duomenų šifravimas TLS protokolu naudojant API prieigos vartus. Prieinamumo kriterijų taip pat leis padidinti konfigūracijos kaip kodo naudojimas. Šių sprendimų aprašymas bei pasirinkimo motyvacija pateikti sekančiuose skyreliuose.

1.5.1. Autentifikacija naudojant API prieigos vartus

Apžvelgus autentifikacijos mechanizmus skyriuje 1.4.1 buvo nuspręsta pasirinkti ketvirtąjį mechanizmą – perkelti autentifikaciją arčiau vartotojo – į API prieigos vartus.

Taip pasirinkta, nes bet kokia didesnė organizacija dažnai jau turi tam tikrą darbuotojų duomenų bazę. Tai gali būti *Microsoft Active Directory*, *Azure AD*, *Amazon IAM*, *Okta*, *Auth0* ar netgi pilnai išorinės sistemos, kurios teikia identiteto paslaugas kaip *Google Identity Platform*, *Microsoft Office 365 Azure AD* ir kt.

Dar viena priežastis – yra keletas programų, kurios veikia kaip įėjimo vartai į sistemą – *Containous Traefik*, *Netflix OSS Zuul* ar *Istio*. Naudojant šias programas galima lengvai įgyvendinti tokį mechanizmą.

Be autentifikacijos perkėlimo logikos iškėlimo iš programos, API prieigos vartų šablonas įgaliną šį funkcionalumą:

1. vienas įeigos taškas visoms programoms;
2. užklausų nukreipimas ir paskirstymas teisingoms programoms;
3. SSL/TLS terminacija (angl. SSL/TLS termination);
4. užklausų auditas;
5. užklausų validacija;
6. ir kt.

1.5.2. RBAC principu pagrįsta autorizacija

Įvertinus keturis populiariausius prieigos kontrolės mechanizmus buvo nuspręsta įgyvendinti rolėmis grįstą prieigos kontrolę, nes ją įgyvendinti ji yra sąlyginai paprasta, tačiau suteikia daugiausiai galimybių.

1.5.3. Diegimo principas sidecar

Sidecar (liet. motociklo priekaba) yra vienas iš architektūrinių dekompozicinių projektavimo šablonų [21].

Naudojant šį šabloną, programa turi palaikyti pagrindinės programos sąsają. Ši programa priima užklausas, jas apdoroja, persiunčia pagrindinei programai, apdoroja grąžintą rezultatą ir galiausiai jį persiunčia klientui. Šio šablono naudojimas leidžia praplėsti pagrindinės programos funkcionalumą nemodifikuojant pagrindinės programos.

Šio projekto pagrindinis komponentas *Alertmanager Proxy* bus diegiamas kaip *sidecar* programa, o *Alertmanager* bus pagrindinė programa.

Alertmanager Proxy bus atsakinga už:

- užklausų autorizavimą;
- metaduomenų pridėjimą prie Alertmanager duomenų struktūrų.

1.5.4. Konfigūracija kaip kodas

Konfigūracija kaip kodas (angl. *configuration-as-code*) yra programų konfigūracijos duomenų valdymo procesas [22]. Šio proceso pagrindas yra visą programos konfigūraciją saugoti failuose. Toks procesas suteikia šiuos privalumus [23]:

- Įgalina automatizaciją, nes nereikia kiekvieno paleidimo metu rankomis konfigūruoti sistemos;
- Įgalina konfigūracijos versijavimą, nes šie falai gali būti saugomi versijos kontrolės sistemoje (VCS – angl. *version control system*);
- Pakeitimų atsekamumą – kadangi konfigūracija yra saugoma VCS, galima matyti visus konfigūracijos pakeitimus bei kas ir kada juos atliko;
- Leidžia padidinti pasiekiamumą, nes įvykus sutrikimui, programą galima paleisti daug greičiau, nes nereikia jos iš naujo konfigūruoti.

1.6. Analizės išvados

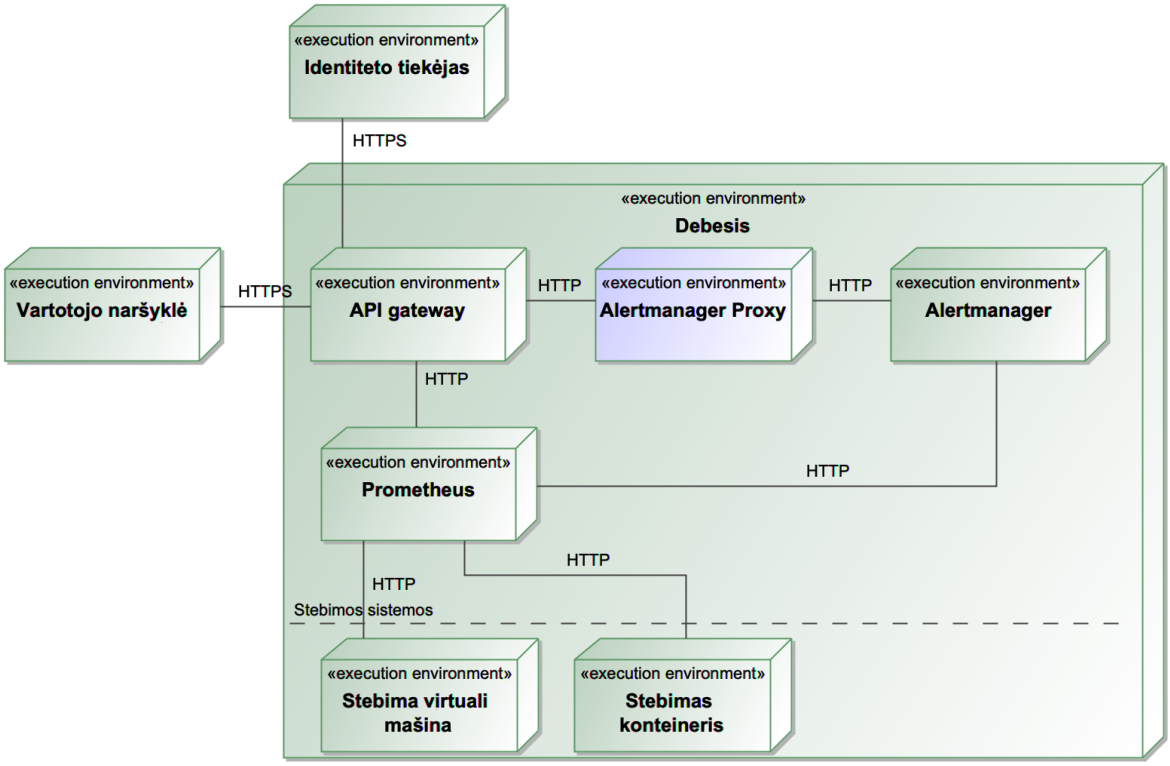
1. Atlikta debesų kompiuterijos analizė ir rasta, kad stebėjimo sistemos gali padidinti debesų kompiuterijos saugumą.
2. Išanalizuotas debesų paslaugų stebėjimas ir palygintos kelios tą atliekančios sistemos, visose rasta trūkumų, tačiau *Prometheus* sistemoje rasta, kad ji neturi jokių prieigos valdymo mechanizmų.
3. Peržvelgti keli pagrindiniai prieigos valdymo mechanizmai.
4. Nuspręsta suprojektuoti programą, kuri padidins *Prometheus* sistemos saugumą įgyvendindama CIA triadą.

2. Projektas

Šiame skyriuje pateiktas *Alertmanager Proxy* programos projektas ir prototipas.

2.1. Bendras sistemos vaizdas

Kaip minėta analizės dalyje, *Alertmanager Proxy* bus tarpininkas tarp vartotojo naršyklės ir *Alertmanager*, kuris bus atsakingas už užklausų autorizavimą. Už užklausų autentifikavimą bus atsakingi API prieigos vartai ir identiteto tiekėjas. Taip pat sistemoje bus įdiegtas ir *Prometheus*, kuris rinks informaciją iš stebimų paslaugų – programų, virtualių mašinų, konteinerių ir kt. Sprendimo diegimo diagrama pateikta 2.1 paveiksle.



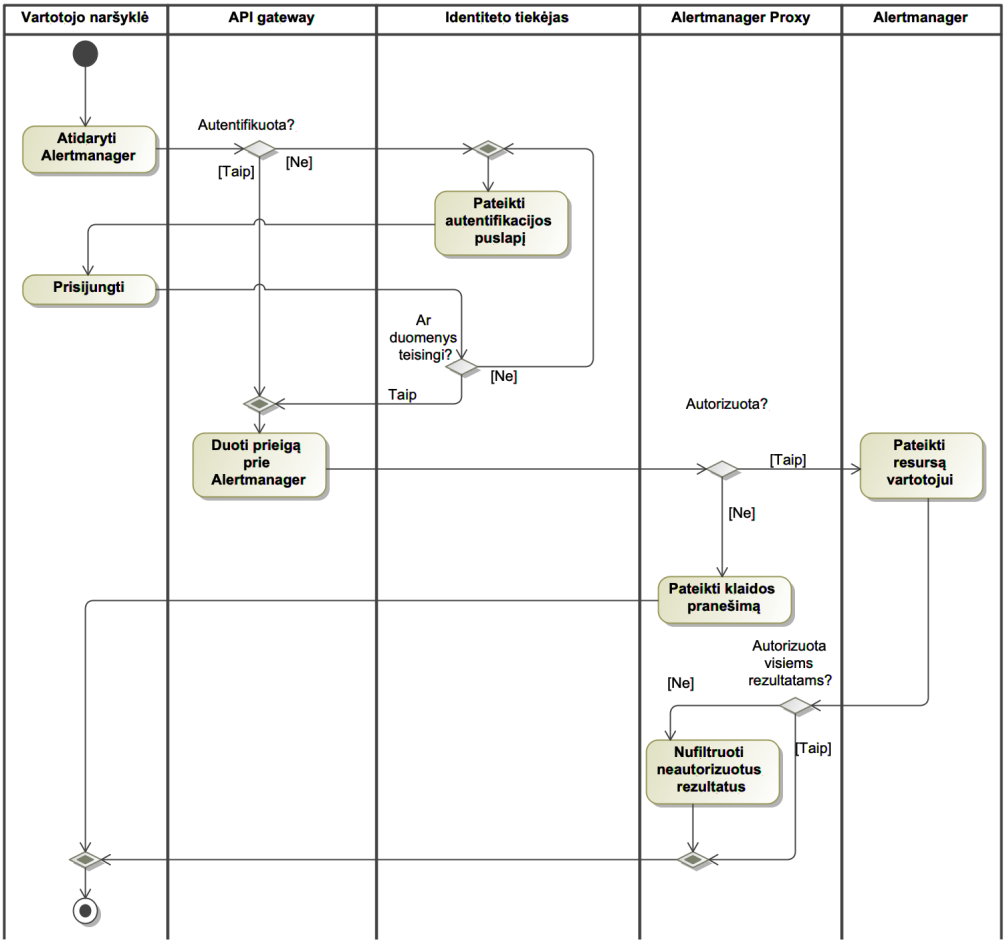
2.1 pav. Alertmanager pasiekiamumas įdiegus kuriamą sprendimą

Vartotojui norint peržiūrėti *Alertmanager* pateikiamus resursus, bus vykdoma tokia veiksmų seka:

1. vartotojas interneto naršyklėje atsidaro *Alertmanager Proxy;*
2. API prieigos vartai patikrina ar vartotojas autentifikuotas;
3. jei vartotojas neautentifikuotas:
   - vartotojas nukreipiamas į identiteto paslaugos teikėjo prisijungimo langą;
   - vartotojas prisijungia įvesdamas savo autentifikacijos duomenis;
   - jei vartotojo duomenys yra neteisingi, jis yra nukreipiamas atgal į trečią žingsnį;
4. vartotojui yra suteikiama prieiga prie *Alertmanager Proxy;*
5. jeigu vartotojas yra neautorizuotas pasiekti norimus resursus, jam yra pateikiamas klaidos pranešimas ir veikla yra baigiama;
6. *Alertmanager Proxy* kreipiasi į *Alertmanager* ir gauna visus užklaustus resursus;

7. Jeigu vartotojas nėra autorizuotas visų gautų resursų peržiūrai, neautorizuoti resursai yra nufiltruojami;
8. Likę resursai yra pateikiami vartotojui.

Šis procesas yra pavaizduotas veikos diagrama 2.2 pav.



2.2 pav. Vartotojo autentifikavimas naudojant API prieigos vartus

Resursų kūrimo, atnaujinimo ir trynimo operacijos veiks analogiškai. Visos operacijos yra detalizuotos šiame skyriuje.

2.2. Reikalavimų specifikacija

2.2.1. Alertmanager funkcijos

Alertmanger komunikacijai naudoja HTTP REST API, kuri yra pilnai dokumentuota naudojant OpenAPI standartą. Tai reiškia, kad iš *Alertmanager* OpenAPI dokumentacijos galima sugeneruoti kliento kodą.

Šioje dokumentacijoje [24] yra aprašyti 9 *Alertmanager* teikiami metodai:

1. sukurti pranešimą;
2. peržiūrėti pranešimus;
3. peržiūrėti pranešimų grupes;

4. peržiūrėti gavėjus;
5. peržiūrėti sistemos būklę;
6. peržiūrėti tylas;
7. peržiūrėti tylą;
8. sukurti tylą;
9. sunaikinti tylą.

Modeliuojant *Alertmanager* panaudos atvejus buvo nuspręsta šeštąjį ir septintąjį metodus sujungti į vieną panaudos atvejį, nes jie atlieką tą patį veiksmą. Visi kiti aprašyti metodai buvo tiesiogiai sumodeliuoti į panaudos atvejus. Šiuos metodus buvo nuspręstą įdėti į API posistemę panaudos atvejų diagramoje (2.3 pav.). Taip pat buvo išskirti ir kiti *Alertmanager* panaudos atvejai – pranešimų priėmimas iš *Prometheus* programos bei pranešimų kanalų ir pranešimų grupavimo valdymo panaudos atvejai.

2.2.2. Alertmanager Proxy autorizacijos modelis

Leidimai bus priklausomi nuo organizacijos struktūros. Kiekvienas leidimas nurodys subjekto (leidimo turėtojo, vartotojo) ir objekto (perspėjimo) priklausomybę komandai. Tai yra kiekvienai programos vykdymo aplinkoje esančiai komandai reikės sukurti po vieną leidimą.

Rolės. *Alertmanager Proxy* turės dviejų tipų roles:

- statinė administratoriaus rolė. Šią rolę turintys vartotojai turės turės pilną prieigą prie viso sistemos funkcionalumo;
- dinaminės komandų rolės. Jos priklausys nuo organizacijoje esančių padalinių ir komandų. Viena rolė galės turėti keletą leidimų.

Kiekvienas vartotojas galės turėti keletą rolių.

Alertmanager tylas galės kurti visi vartotojai, tačiau jas modifikuoti bei trinti galės tik jas sukūrę vartotojai ir administratoriai

Autorizacijos posistemė bus įgyvendinta tik teigiamų leidimų principų, tai yra leidimai bus sumuojami be galimybės juos atimti.

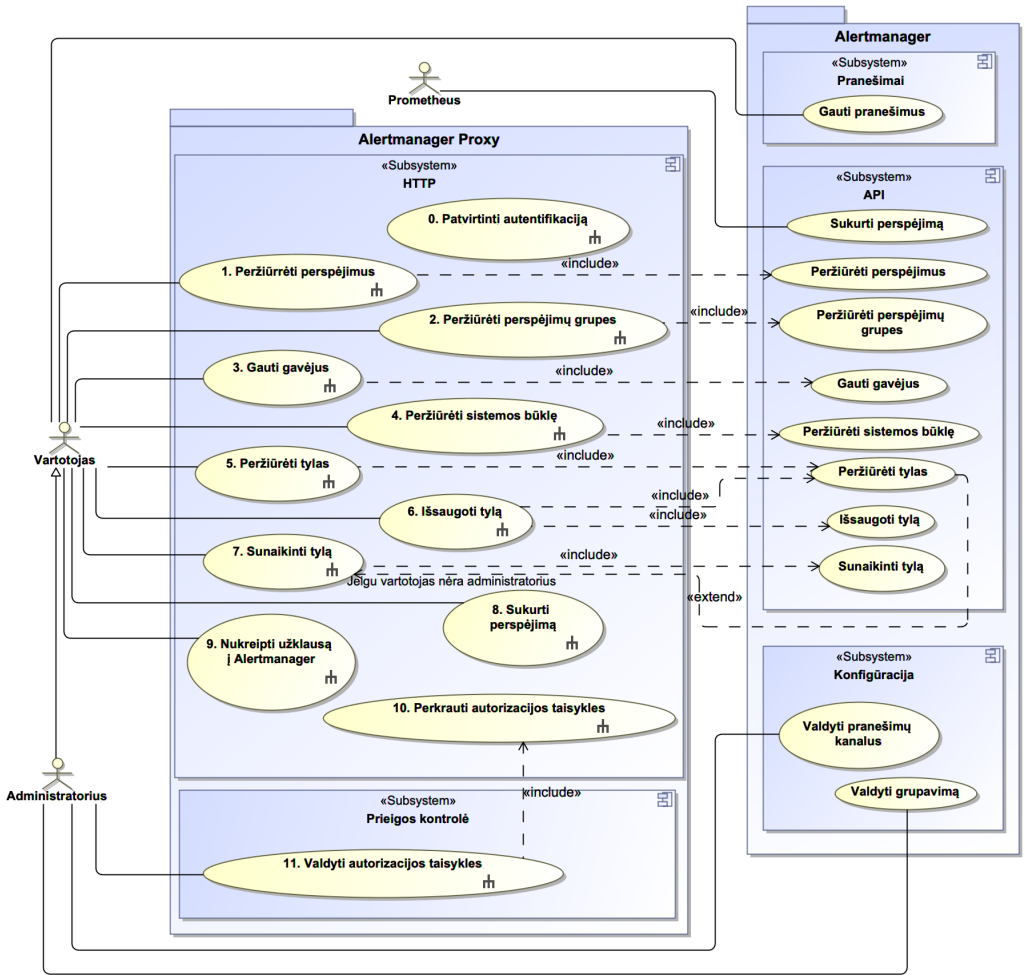
2.2.3. Alertmanager Proxy funkcijos

Alertmanager Proxy įgyvendins 7-is iš 8-ių panaudos atvejų – visus išskyrus pranešimų kūrimą. Neįgyvendinti pranešimų kūrimo buvo nuspręsta, nes yra daroma prielaida, kad *Alertmanager* bus diegiamas saugioje aplinkoje kartu su programa *Prometheus*, kuri yra pagrindinis, ir daugeliu atveju vienintelis, pranešimų kūrėjas, todėl *Prometheus* kurs pranešimus tiesiai į *Alertmanager*.

Taip pat sistemoje atsirado dar keletas papildomų panaudos atvejų:

- patvirtinti autentifikaciją;
- nukreipti užklausą į *Alertmanager*;
- valdyti autorizacijos taisykles;
- perkrauti autorizacijos taisykles.

Visi programos *Alertmanager Proxy* panaudos atvejai yra pateikti 2.3 paveiksle. Iš karto po diagrama yra visų panaudos atvejų aprašymai pateikti lentelėse.



2.3 pav. Alertmanager Proxy panaudos atvejai

2 lentelė „0. Patvirtinti autentifikaciją“ panaudos atvejo aprašas

PA 0. Patvirtinti autentifikaciją		
Tikslas Patvirtinti vartotojo autentifikaciją		
Aprašymas Šis panaudos atvejis yra vykdomas panaudos atvejų 1-10 pradžioje, žymėjimas diagramoje nėra įtrauktas, kad nekomplikuoti diagramos papildomais ryšiais. Vykdyti užklausą yra leidžiama tik jeigu ją vykdantis vartotojas yra autentifikuotas.		
Prieš sąlyga		-
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	-
	Specializuoja PA	1. Peržiūrėti pranešimus

		2. Peržiūrėti pranešimų grupes 3. Peržiūrėti gavėjus 4. Peržiūrėti sistemos būklę 5. Peržiūrėti tylas 6. Išsaugoti tylą 7. Sunaikinti tylą 8. Sukurti perspėjimą 9. Nukreipti užklausą į Alertmanager 10. Perkrauti autorizacijos taisykles
Po sąlygą		Vartotojas yra autentifikuotas ir jam yra leidžiama atlikti sekantį veiksmą.

3 lentelė „1. Peržiūrėti perspėjimus“ panaudos atvejo aprašas

PA 1. Peržiūrėti perspėjimus		
Tikslas Peržiūrėti aktyvius perspėjimus ir visą papildomą informaciją.		
Aprašymas Vartotojui rodomi tik tie perspėjimai, kurie turi bent vieną sutampančią žymą su vartotojo leidimu. Perspėjimą sudaro: • anotacijos; • gavėjai; • piršto antspaudas; • pradžios data; • atnaujinimo data; • pabaigos data; • būklė; • žymos; • kūrėjo URL.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

4 lentelė „2. Peržiūrėti perspėjimų grupes“ panaudos atvejo aprašas

PA 2. Peržiūrėti perspėjimų grupes
Tikslas Peržiūrėti aktyvius perspėjimus sugrupuotus pagal gavėjus.
Aprašymas Vartotojui rodomi tik tos perspėjimų grupės, kurios turi bent vieną sutampančią žymą su vartotojo leidimu nepaisant pačių perspėjimų žymų. Perspėjimų grupę sudaro:

• perspėjimų sąrašas; • gavėjas; • žymos.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

5 lentelė „3. Peržiūrėti gavėjus“ panaudos atvejo aprašas

PA 3. Peržiūrėti gavėjus		
Tikslas Gauti ir peržiūrėti visą gavėjų sąrašą		
Aprašymas Vartotojui pateikiamas nefiltruotas gavėjų sąrašas. Šiame sąraše yra pateikiami tik gavėjų pavadinimai, be jų konfigūracijos, todėl šį sąrašą galės pasiekti visi autentifikuoti vartotojai.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

6 lentelė „4. Peržiūrėti sistemos būklę“ panaudos atvejo aprašas

PA 4. Peržiūrėti sistemos būklę
Tikslas Peržiūrėti *Alertmanager* būklę ir konfigūraciją

Aprašymas
Administratoriams bus pateikta pilna *Alertmanager* konfigūracija, kurią sudaro:
• *Alertmanager* klasterio informacija:
  ○ pavadinimas;
  ○ būklė;
  ○ bendradarbių (angl. *peer*) adresai;
• *Alertmanager* versijos informacija:
  ○ versija, revizija, šaka, sukūręs vartotojas, sukūrimo data, go versija;
• *Alertmanager* konfigūracijos failas;
• *Alertmanager* paleidimo data.
 	
Vartotojui, kuris neturi administratoriaus teisių, bandant peržiūrėti konfigūraciją bus rodoma tik dalis informacijos:

• klasterio pavadinimas; • klasterio būklė; • *Alertmanager* paleidimo data.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

7 lentelė „5. Peržiūrėti tylas“ panaudos atvejo aprašas

PA 5. Peržiūrėti tylas		
Tikslas Peržiūrėti visas sukurtas tylas		

Aprašymas
Tylas peržiūrėti galės visi autentifikuoti vartotojai.
Tylą sudaro:
• unikalus id;
• tylos būklė;
• pradžios data;
• atnaujinimo data;
• pabaigos data;
• sąrašas šablonų (angl. *matcher*), vieną šabloną sudaro:
  o pavadinimas;
  o reikšmė;
  o ar reikšmė yra regex išraiška;
• kūrėjas;
• komentaras.		

Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

8 lentelė „6. Išsaugoti tylą“ panaudos atvejo aprašas

PA 6. Išsaugoti tylą
Tikslas Sukurti naują arba atnaujinti esamą tylą, kuri blokuos naujų pranešimų siuntimą
Aprašymas

Saugoti tylas galės visi vartotojai, tačiau kuriant tylą per *Alertmanager Proxy* bus pridedama papildoma informacija : • į kūrėjo lauką visada bus įvedamas vartotojo, kuris sukūrė tylą, id; • komentarų lauke bus saugoma visa tylos pakeitimų istorija.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		Sistemoje sukurta nauja tyla

9 lentelė „7. Sunaikinti tylą“ panaudos atvejo aprašas

PA 7. Sunaikinti tylą		
Tikslas Nebeblokuoti pranešimų sutampančių su tylos šablonais (angl. *matcher*).		
Aprašymas Jei vartotojas yra sistemos administratorius arba naikinamos tylos kūrėjas, tai tyla yra sunaikinama, jei ne – vartotojui grąžinamas klaidos pranešimas.		
Prieš sąlyga		Vartotojas yra autentifikuotas Vartotojas yra administratorius, arba tylos kūrėjas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		Egzistuojančios tylos būsena pakeista iš aktyvios į nebegaliojančią

10 lentelė „8. Sukurti perspėjimą“ panaudos atvejo aprašas

PA 8. Sukurti perspėjimą		
Tikslas Neleisti jokiam vartotojui sukurti perspėjimų.		
Aprašymas Yra nuspręsta neleisti kurti perspėjimų naudojant *Alertmanager Proxy* programą, todėl vykdant šį panaudos atvejį vartotojui yra visada grąžinamas klaidos pranešimas.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		-

11 lentelė „9. Nukreipti užklausą į Alertmanager“ panaudos atvejo aprašas

PA 9. Nukreipti užklausą į Alertmanager		
Tikslas Leisti vartotojui pasiekti *Alertmanager* programos vartotojo sąsają.		
Aprašymas Jeigu užklausa nebuvo apdorota kitų panaudos atvejų, tai ją reikia nukreipti į *Alertmanager* ir vartotojui grąžinti visus gautus rezultatus.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Vartotojas
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	-
Po sąlygą		Egzistuojančios tylos būsena pakeista iš aktyvios į nebegaliojančią

12 lentelė „10. Perkrauti autorizacijos taisykles“ panaudos atvejo aprašas

PA 10. Perkrauti autorizacijos taisykles		
Tikslas Užkrauti naujas autorizacijos taisykles		
Aprašymas Sistema patikrina ar taisyklės yra parašytos be klaidų, jei taip – jos yra užkraunamos, jei ne – grąžinamas klaidos pranešimas ir yra paliekamos galioti senos taisyklės.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Administratorius
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	0. Patvirtinti autentifikaciją
	Specializuoja PA	11. Valdyti autorizacijos taisykles
Po sąlygą		Užkrautos naujos autorizacijos taisyklės

13 lentelė „11. Valdyti autorizacijos taisykles“ panaudos atvejo aprašas

PA 11. Valdyti autorizacijos taisykles
Tikslas Modifikuoti sistemos roles, leidimus ir vartotojų roles
Aprašymas Valdyti rolėmis grįstos autorizacijos taisykles: • suteikti vartotojui rolę • anuliuoti vartotojo rolę; • sukurti leidimą; • modifikuoti leidimą; • trinti leidimą; • sukurti rolę;

• modifikuoti rolę; • trinti rolę. Visos šios taisyklės yra saugomos tekstiniame faile, todėl šių teisių keitimo operacija susidaro iš failo redagavimo bei autorizacijos posistemės perkrovimo.		
Prieš sąlyga		Vartotojas yra autentifikuotas
Aktorius		Administratorius
Susiję panaudojimo atvejai	Išplečiantys PA	-
	Apimami PA	10. Perkrauti autorizacijos taisykles
	Specializuoja PA	-
Po sąlygą		Atnaujintos autorizacijos taisyklės

2.2.4. Apribojimai

Prometheus ir *Alertmanager* dažniausiai būna diegiami kaip *Docker* konteineriai tiesiogiai, arba orkestruojami aplinkose kaip *Kubernetes* ar *Docker Swarm*. Dėl šios priežasties *Alertmanager Proxy* diegimo artefaktas turi taip pat būti prieinamas kaip *Docker* konteineris.

Su *Alertmanager Proxy* bendradarbiaujančios programos:

- *Alertmanager*;
- *Prometheus*;
- API prieigos vartų programa;
- identiteto paslaugos teikėjas.

Autorizacijos mechanizmų realizavimui pasirinktas *Casbin* [25] karkasas, nes jis suteikia didelį lankstumą:

- galima pasirinkti kokį prieigos kontrolės mechanizmą naudoti: ACL, RBAC ar ABAC;
- karkasas yra įgyvendintas kaip bibliotekos kelioms programavimo kalboms: *Java*, *Golang*, *C#*, *Javascript*, *Python* ir kt.;
- *Casbin* nesaugo pačių vartotojų, o saugo tik vartotojų ir leidimų ryšius. Tai leidžia labai paprastai aprašyti visas vartotojų teises vienoje vietoje.
- Vartotojų ir leidimų ryšius galima saugoti *csv* formato faile, SQL ar NoSQL tipo duomenų bazėje, objektų saugykloje (angl. *object storage*) kaip *MinIO* ar *Amazon S3* ar rakto-reikšmės saugykloje kaip *Redis* ar *etcd*.

Programa *Alertmanager Proxy* bus realizuota naudojant Java programavimo klabą, todėl bus naudojama *jCasbin* biblioteka [26].

Casbin konfigūracija bus saugoma failuose:

- autorizacijos modelis saugomas conf tipo faile;
- vartotojų-rolių, rolių-leidimų bei rolių-rolių ryšiai bus saugomi csv faile.

Alertmanager Proxy turės veikti kaip HTTP serveris, todėl yra reikalingas tinklo programų karkasas. Šiam projektui įgyvendinti pasirinktas *sparkjava* karkasas.

2.2.5. Vartotojo sąsajos specifikacija

Kadangi sistema yra kuriama naudojant *sidecar* šabloną, tai vartotojo sąsaja bus identiška paties *Alertmanager* sąsajai, o skirsis tik pateikiama informacija.

2.2.6. Realizacijai keliami reikalavimai

1. vidutiniškai programa turi atsakyti per 324 milisekundes [27], tam kad patektų tarp 20% greičiausių internetu pasiekiamų paslaugų;
2. Sistema turi būti lengvai ir greitai diegiama;
3. Sistema turi palaikyti galimybę lengvai keisti vartotojų autorizacijos konfigūraciją;
4. Sistema turi palaikyti autentifikacijos mechanizmą naudojant trečiosios šalies identiteto valdymo sistemą, kuri palaiko OIDC standartą.

2.2.7. Techninė specifikacija

Pagrindinė diegimo aplinka *Docker Swarm* klasteris, tačiau sistemą bus galima diegti bet kokioje aplinkoje, kuri palaiko konteinerių orkestravimą, pavyzdžiu *Kubernetes*, *Rancher* ar *Hashicorp Nomad*.

Diegimo aplinka testavimui bus *Docker* bei *Docker-Compose*. Šie įrankiai pasirinkti, nes jie veikia *Windows*, *Mac OS* ir *Linux* operacinėse sistemose.

Norint įgyvendinti patikimą pasiekiamumą iš interneto naudojant šifruotą HTTPS protokolą, reikės domeno bei DNS A tipo įrašo *Alertmanager Proxy* programai.

Programavimo kalbos, derinimo, automatizavimo priemonės, operacinės sistemos

1. *Java 14* – programavimo kalba ir vykdymo aplinka;
2. *Casbin* – autorizacijos karkasas;
3. *Docker Swarm* – debesies vykdymo aplinka;
4. *Traefik* – API prieigos vartų sprendimas;
5. *Let's Encypt* – nemokamas, automatizuotas sertifikatų autoritetas (angl. *CA – certificate authority*);
6. *SonarQube* – statinės analizė įrankis;
7. *Gradle* – projekto valdymo įrankis;
8. *OpenAPI – Alertmanager* API specifikacija ir sugeneruotas kliento kodas;
9. *Postman* – HTTP paslaugų rankinio testavimo įrankis;
10. *IntelliJ IDEA* – programavimo aplinka;
11. *Docker* – konteinerizavimo platforma.

2.3. Projektavimo metodai

Įrankio projekto kūrimui naudojome UML 2 notaciją ir diagramas. Panaudos atvejų, komponentų, paketų, klasių, veiklos ir sekų diagramos nubraižytos naudojant *MagicDraw* modeliavimo įrankį.

Panaudos atvejų specifikavimui yra naudojamos lentelės.

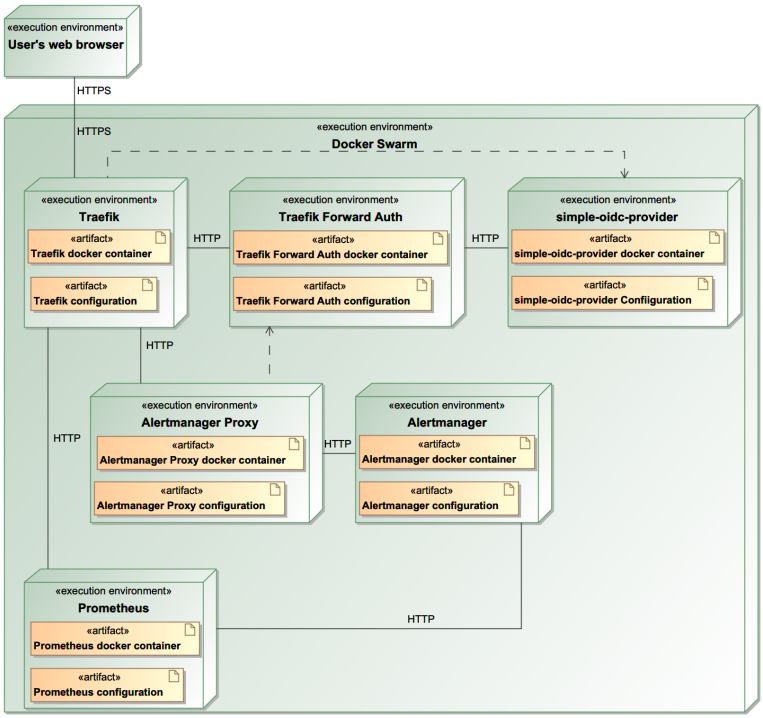
2.4. Sistemos prototipas

Visą projekto prototipo sistemą sudaro šie komponentai:

1. *Alertmanager Proxy*
2. *Alertmanager*
3. *Prometheus*
4. *Traefik*
5. *Traefik Forward Auth*
6. *Simple Oidc Provider*

Visas šias programas, galima diegti kaip *Docker* konteinerius, todėl galima naudoti tokias konteinerių orkestravimo aplinkas kaip *Kubernetes, Hashicorp Nomad* ar *Docker Swarm*.

Docker Swarm ir vienas iš paprasčiausiai įdiegiamų ir paruošiamų naudoti produkcijos aplinkoje konteinerių orkestratorius, taip pat jis palaiko konfigūravimą kaip kodą. Dėl šių priežasčių jis ir buvo pasirinktas kaip prototipo vykdymo aplinka. Principinė sistemos diegimo schema pateikta 2.4 paveiksle.



2.4 pav. Prototipo diegimo schema

Šiame poskyryje pateikta visų komponentų aprašymas bei *Alertmanager Proxy* prototipo realizacijos statinio ir dinaminio vaizdų diagramos su aprašymais.

2.4.1. Sistemos komponentai

Traefik

Iš 1.5.1 skyrelyje minėtų API prieigos vartų teikiamų funkcijų, šiame projekte bus reikalingos 5-ios:

1. vienas įeigos taškas visoms programoms;

2. užklausų nukreipimas ir paskirstymas teisingoms programoms;
3. SSL/TLS terminacija (angl. SSL/TLS termination);
4. užklausų auditas;
5. autentifikacija.

Pirmus keturis punktus įgyvendina *Containaous Traefik* atvirkštinis įgaliotasis serveris (angl. *reverse proxy*).

Papildomos priežastys, kodėl pasirinkta naudoti būtent Traefik yra:

1. priklauso *Cloud Native Computing Foundation*, todėl yra užtikrinama jo palaikymas (angl. *support*);
2. *Traefik* yra atviro kodo ir nemokamas;
3. yra sukurtas lengvai integruotis su dauguma debesų kompiuterijos infrastruktūros technologijų kaip *Kubernetes*, *Docker Swarm*, *Hashicorp Consul*, *Amazon ECS* ir kt.

Traefik Forward Auth

Pasitelkus įskiepį *Traefik Forward Auth* [28], programoje *Traefik* galima įgyvendinti autentifikacijos mechanizmą.

Traefik Forward Auth pasirinktas, nes ši programa atitinka šiuos keliamus reikalavimus:

1. palaiko identiteto valdymo standartą OIDC;
2. yra lengvai diegiamas ir galima konfigūruoti kokias užklausas padaryti viešomis (praleisti be autentifikacijos);
3. perduoda vartotojo identitetą paslaugai naudodamas „X-Forwarded-User“ HTTP antraštę (angl. *header*);
4. gali būti atsakingu už vieno arba kelių paslaugų autentifikavimą.

Simple Oidc Provider

Taip pat šiame projekte yra reikalinga ir identiteto valdymo sistema, kuri palaikytų OIDC standartą. Autentifikavimo paslaugos teikėjas prototipui pasirinktas *Qlik* kompanijos kuriamas įrankis *simple-oidc-provider* kuris pilnai palaiko *OpenID Connect* standartą ir vartotojų aprašymą JSON faile bei leidžia greitai vystyti programą ir paprastai keisti vartotojų bei jų prisijungimo informaciją, kuri yra taip pat konfigūruojama kaip kodas.

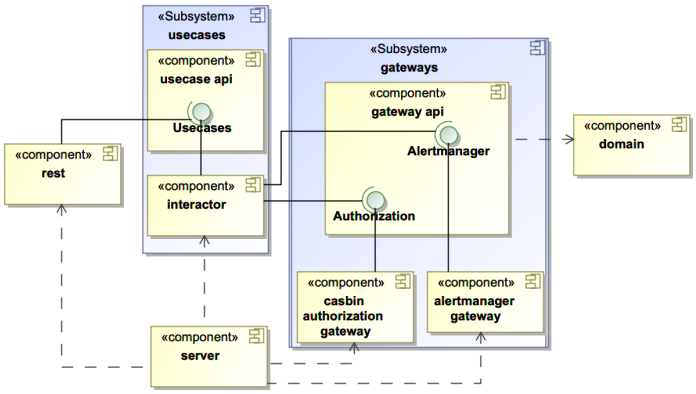
2.4.2. Statinis Alertmanager Proxy vaizdas

Alertmanager Proxy yra suprojektuota naudojant *Clean Architecture* [29].
Sistema yra suskirstyta į 8 atskirus komponentus:

- **server** – šiame komponente yra visų komponentų tarpusavio susiejimo ir programos paleidimo logika;
- **rest** komponente yra aprašyta programos HTTP API prieigos keliai ir įgyvendinta pradinė duomenų kontrolė, šis paketas naudoja *usecase api* komponentą;
- **usecae api** komponente pateikiamos panaudos atvejų sąsajos (angl. *interface*);
- **interactor** pakete yra pagrindinė programos panaudos atvejų logika. Šio komponento klasės realizuoja *usecase api* komponento sąsajas. Šis komponentas naudoja *gateway api* komponento klases bendravimui su išorinėmis esybėmis;

– **gateway api** pakete yra aprašytos išorinių sistemų sąsajos. Šis komponentas naudoja *domain* paketą;
– **casbin authorization gateway** – šiame komponente yra vykdoma autorizacijos logika;
– **alertmanager gateway** – šis komponentas yra atsakingas už programos bendravimą su programa *Alertmanager*;
– **domain** – šiame komponente yra aprašytos duomenų struktūros, kuriomis manipuliuoja *gateaway* ir *interactor* komponentai.

Komponentai ir jų ryšiai pateikiami 2.5 paveiksle.



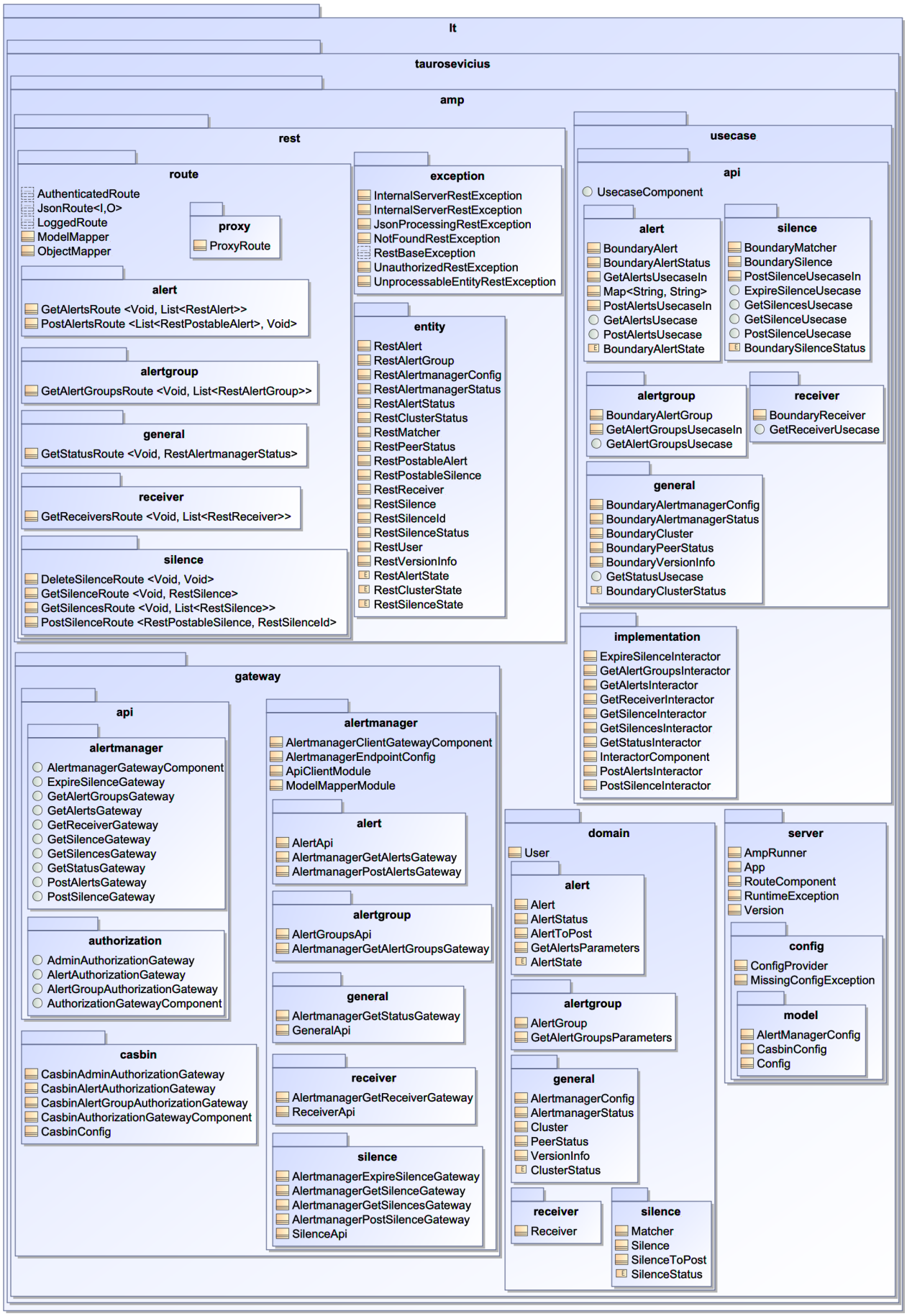
2.5 pav. Alertmanager Proxy Komponentų diagrama

Visi komponentai yra sukuriami iš atitinkamų paketų. Šie ryšiai pateikiami 14 lentelėje.

14 lentelė Alertmanager Proxy komponentų ir paketų ryšiai

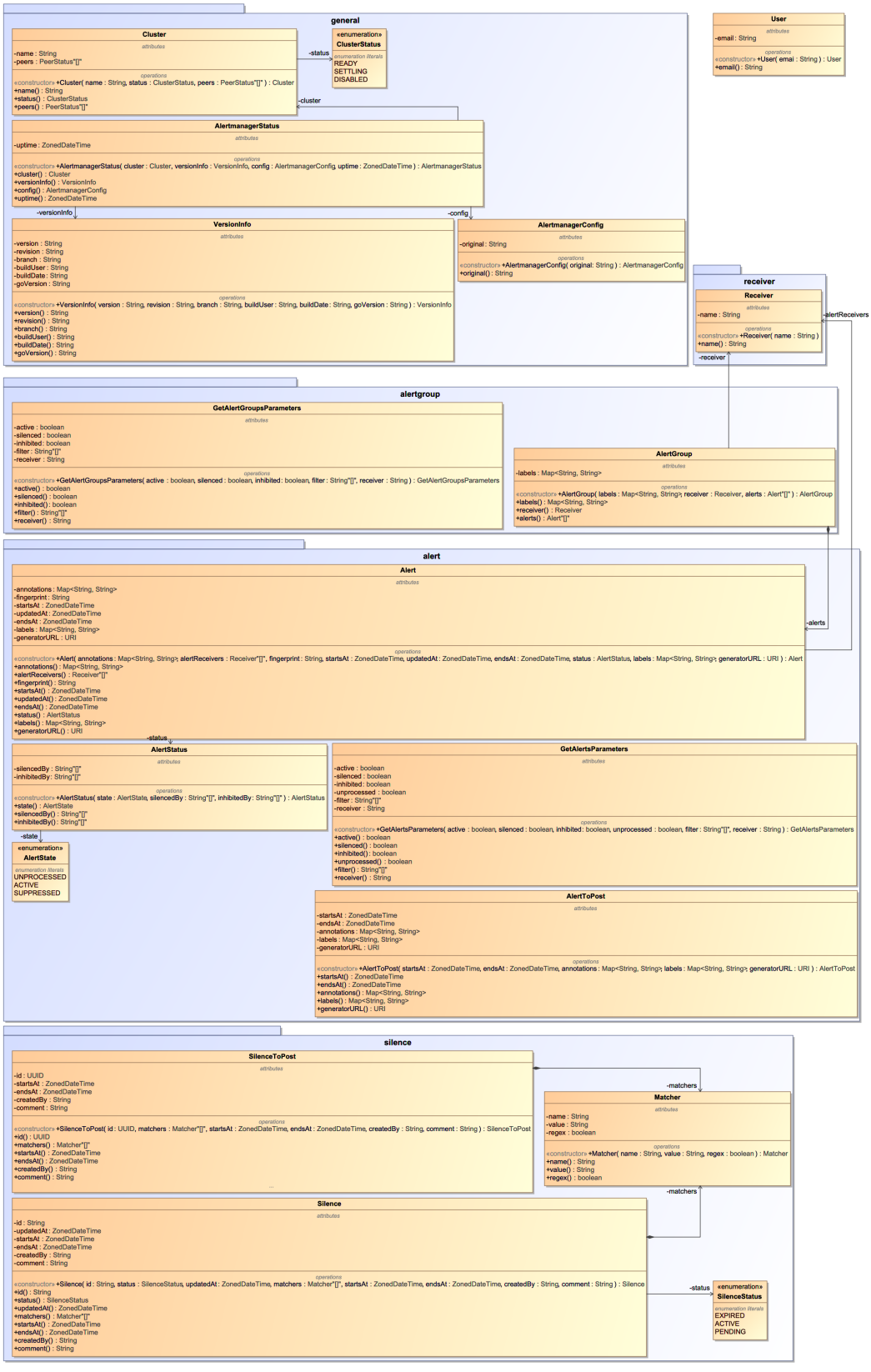
Komponentas	**Paketas**
server	lt.taurosevicius.amp.server
rest	lt.taurosevicius.amp.rest
usecase api	lt.taurosevicius.amp.usecase.api
interactor	lt.taurosevicius.amp.usecase.implementation
gateway api	lt.taurosevicius.amp.gateway.api
casbin authorization gateway	lt.taurosevicius.amp.gateway.casbin
alertmanager gateway	lt.taurosevicius.amp.gateway.alertmanager
domain	lt.taurosevicius.amp.domain

Paveiksle 2.6 yra pateikta sistemos paketų diagrama.



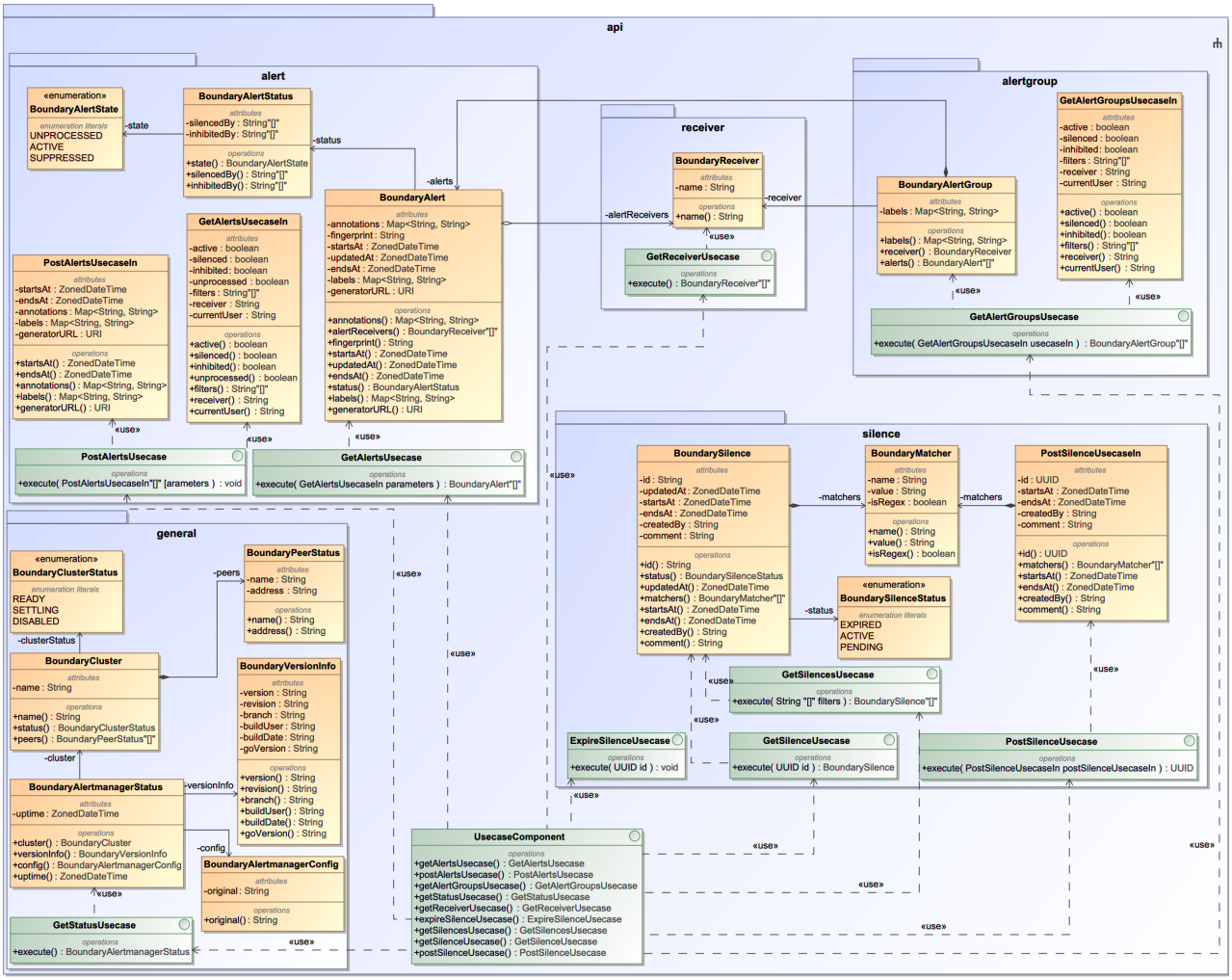
2.6 pav. Alertmanager Proxy paketų diagrama

Paveiksle 2.7 yra pateikta programos *domain* paketo klasių diagrama. Šiame pakete yra aprašytos pagrindinės *Alertmanager Proxy* duomenų struktūros. Jos yra suskirstytos į penkis sub-paketus pagal teikiamą funkcionalumą: *general*, *receiver*, *alertgroup*, *alert* ir *silence*. Pagrindiniame pakete taip pat yra viena klasė *User*, kuri yra naudojama kaip vartotojo atitikmuo programos kontekste.



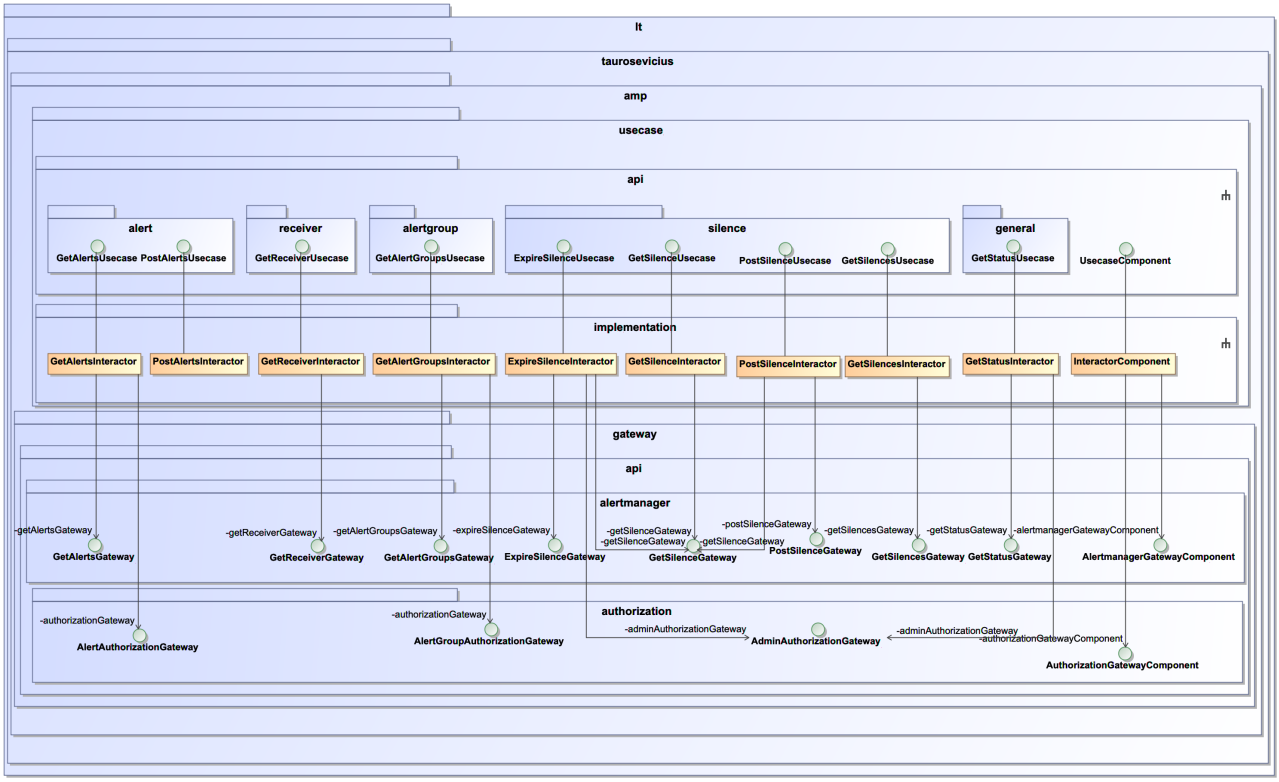
2.7 pav. Paketo lt.taurosevicius.amp.domain klasių diagrama

Klasių diagramoje 2.8 yra *usecase api* komponentą sudarančios klasės. Jos, kaip ir *domain*, paketo klasės yra suskirstytos į 5 sub-paketus. Visos pakete esančios klasės yra nemodifikuojamos duomenų struktūros, kurios yra sukuriamos naudojant konstruktorių. Konstruktoriai yra nepateikti šioje ir visos sekančiose diagramose, nes užima daug vietos ir nesuteikia naudos. Taip pat kiekviename iš paketų yra pateikiamos ir panaudos atvejų sąsajos (angl. *interface*). Sąsaja *UsecaseComponent* yra atsakinga už panaudos atvejų sukūrimą, čia yra naudojamas gamyklos projektavimo šablonas (angl. *factory design pattern*).



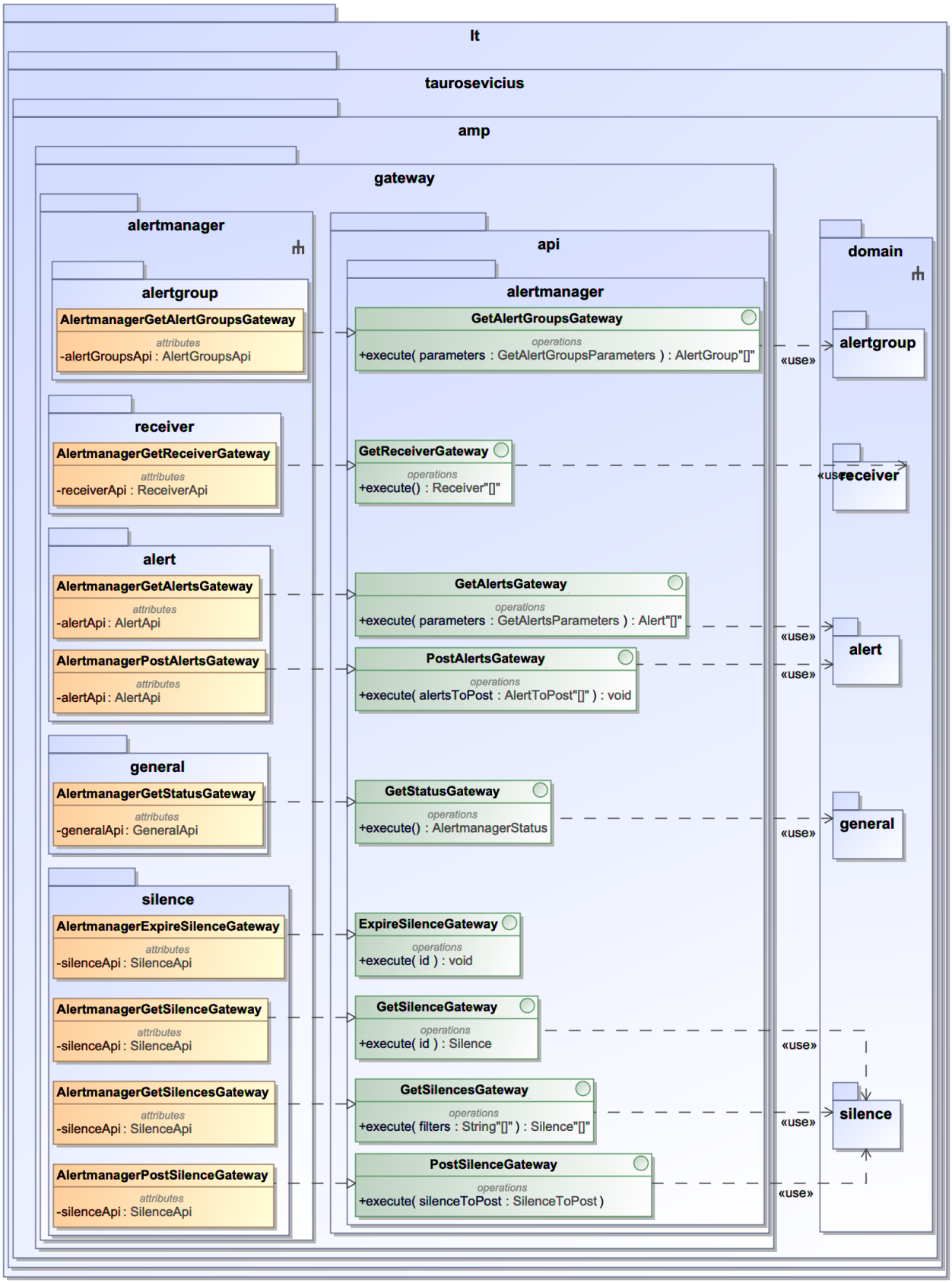
2.8 pav. Paketo lt.taurosevicius.amp.usecase.api klasių diagrama

Diagramoje 2.9 yra atvaizduota komponento *interactor*, kurį sudaro paketas *lt.taurosevicius.amp.usecase.implementation*. Šio paketo klasėse yra pagrindinė programos logika. Taip pat diagramoje yra pateiktos ir *lt.taurosevicius.amp.gateway.api* pakete esančios sąsajos ir jų panaudojimas.



2.9 pav. Paketo lt.taurosevicius.amp.usecase.implementation klasių diagrama

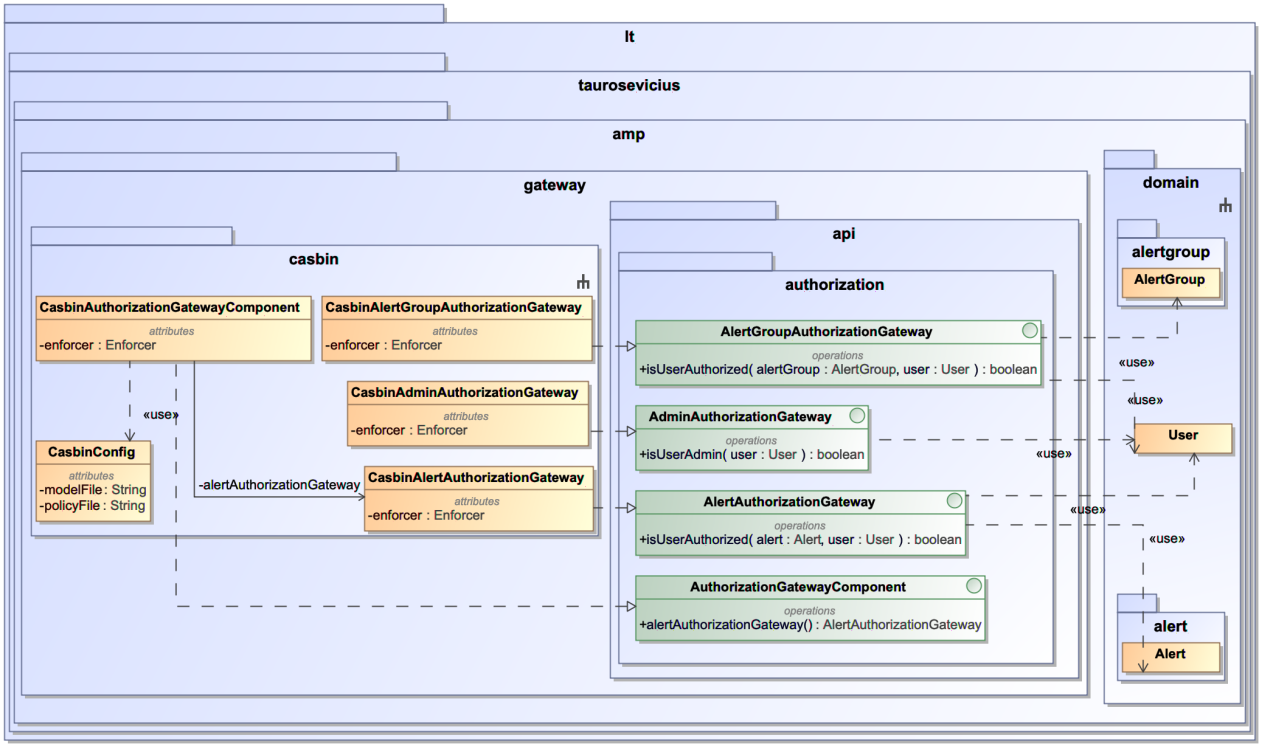
2.10 paveiksle pateikta *lt.taurosevicius.amp.gateway.alertmanager* paketo diagrama. Kiekviena paketo klasė turi *API* objektą iš *Alertmanager* kliento bibliotekos. Šis paketas yra atsakingas už programos bendradarbiavimą su programa *Alertmanager*.



2.10 pav. Paketo lt.taurosevicius.amp.gateway.alertmanager klasių diagrama

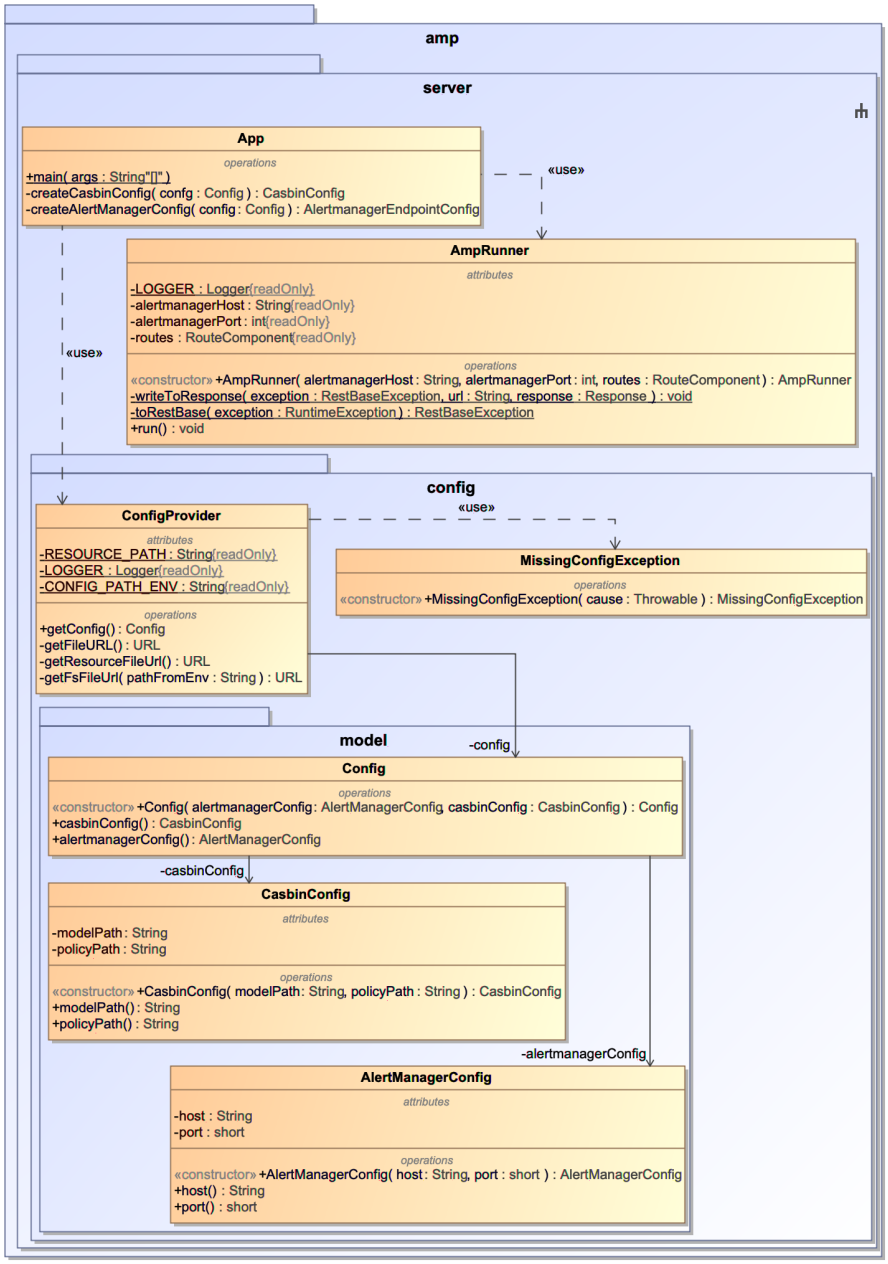
Paveiksle 2.11 yra atvaizduota paketų *lt.taurosevicius.amp.gateway.api.authorization* ir *lt.taurosevicius.amp.gateway.casbin* klasių diagramą. *casbin* pakete esančios klasės realizuoja

api.authorization pakete esančias sąsajas (angl. *interface*). Šių klasių logika yra atsakinga už programos autorizaciją.



2.11 pav. Paketo lt.tauroseviciius.amp.gateway.casbin klasių diagrama

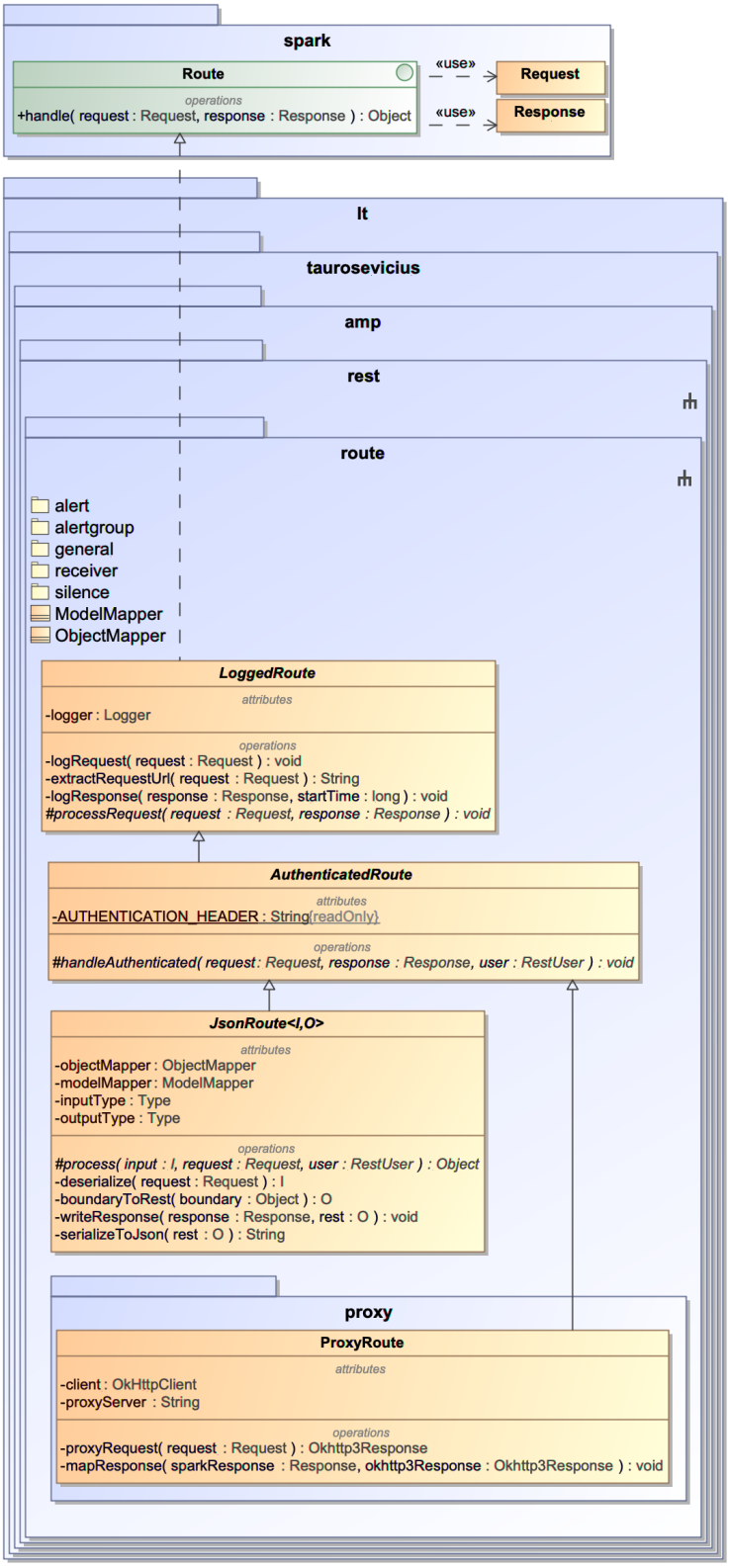
Paketo *lt.tauroseviciius.amp.server* klasių diagrama yra pateikta paveiksle 2.12. Šiame pakete esančios klasės yra atsakingos už programos konfigūracijos nuskaitymą ir programos paleidimą. Iš šio paketo klasių sudarytas komponentas *server* priklauso nuo visų kitų programos komponentų.



2.12 pav. Paketo lt.taurosevicius.amp.server klasių diagrama

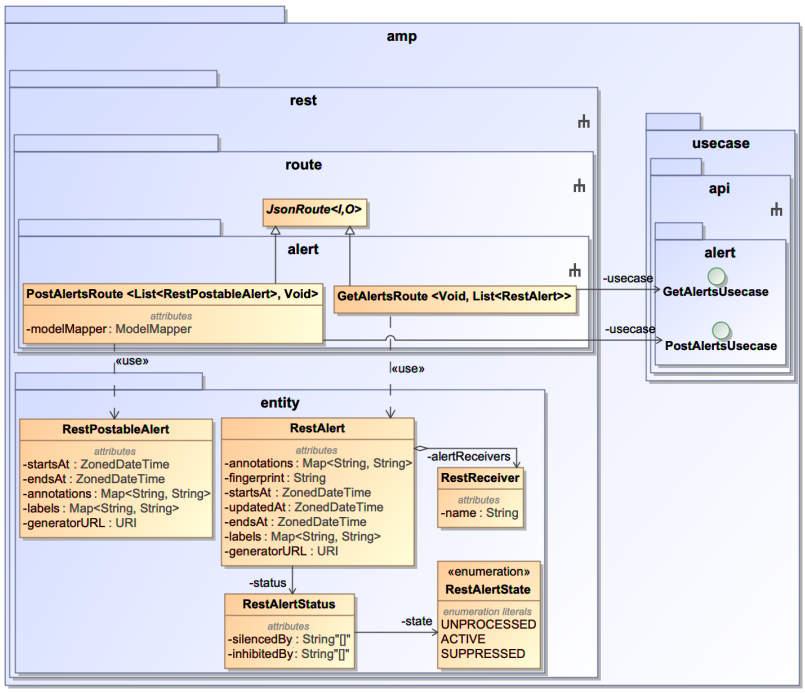
Visos likusios diagramos atspindi *route* komponento klases. Naudojant HTTP biblioteką *sparkJava*, programai gavus HTTP užklausą, ji yra apdorojama iškviečiant *spark.Route* sąsajos realizaciją. Pirmoje diagramoje (2.13 pav.) yra pavaizduoti HTTP užklausų bendrinės doroklių (angl. *handler*) klasės. Klasė *LoggedRoute* įrašo į programos žurnalą įrašą apie vykdomą operaciją. Klase *AuthenticatedRoute* yra atsakinga už HTTP užklausos autentifikavimo patvirtinimą ir vartotojo ID pavertimą į objektą. Klasės *JsonRoute* paskirtis yra paversti gautos užklausos Json dokumentą į *Java* objektą ir gautą atsakymą (Java objektą) paversti į Json dokumentą, kuris bus grąžintas vartotojui. Ši klasė turi du bendrinius (angl. *genereic*) parametrus *I* ir *O*, jie nurodo kokio formato bus HTTP užklausos kūnas (angl. *request body*) ir atsakymo kūnas (angl. *response body*). Jeigu yra naudojamas tipas *Void*, vadinasi ši užklausa ar atsdakymas neturės kūno.

Šioje diagramoje taip pat yra pateikta ir klasė *ProxyRoute*, kuri yra atsakinga už visų neaprašytų užklausų peradresavimą į *Alertmanager*.

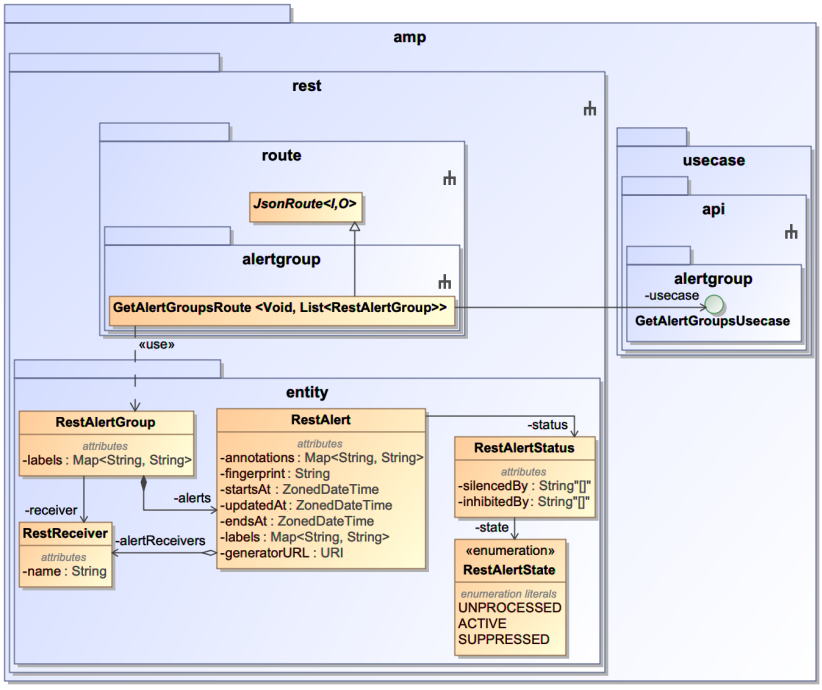


2.13 pav. Paketo lt.taurosevicius.amp.rest.route klasių diagrama

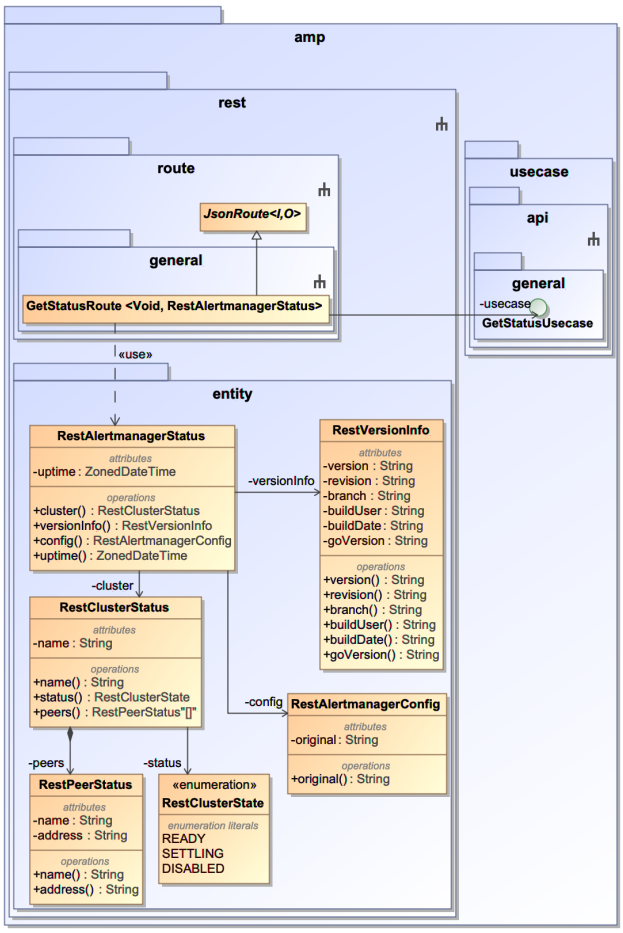
Diagramose 2.14-2.18 yra pateiktos paketo *lt.taurosevicius.amp.rest.route* sub-paketų *alert*, *alertgroup*, *general*, *receiver* ir *silence* klasių diagramos.



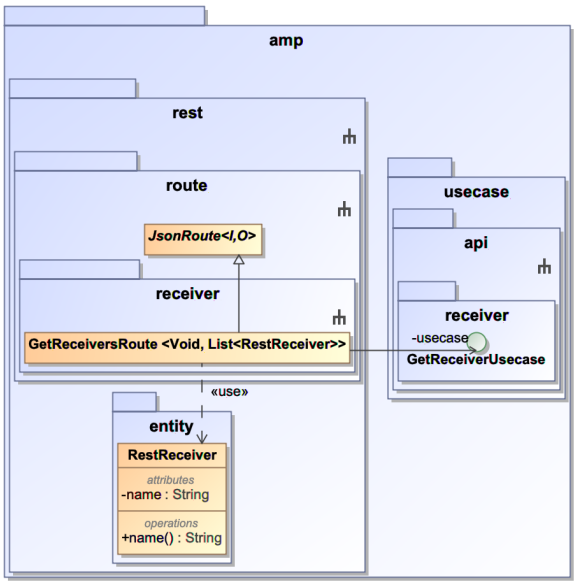
2.14 pav. Paketo lt.taurosevicius.amp.rest.route.alert klasių diagrama



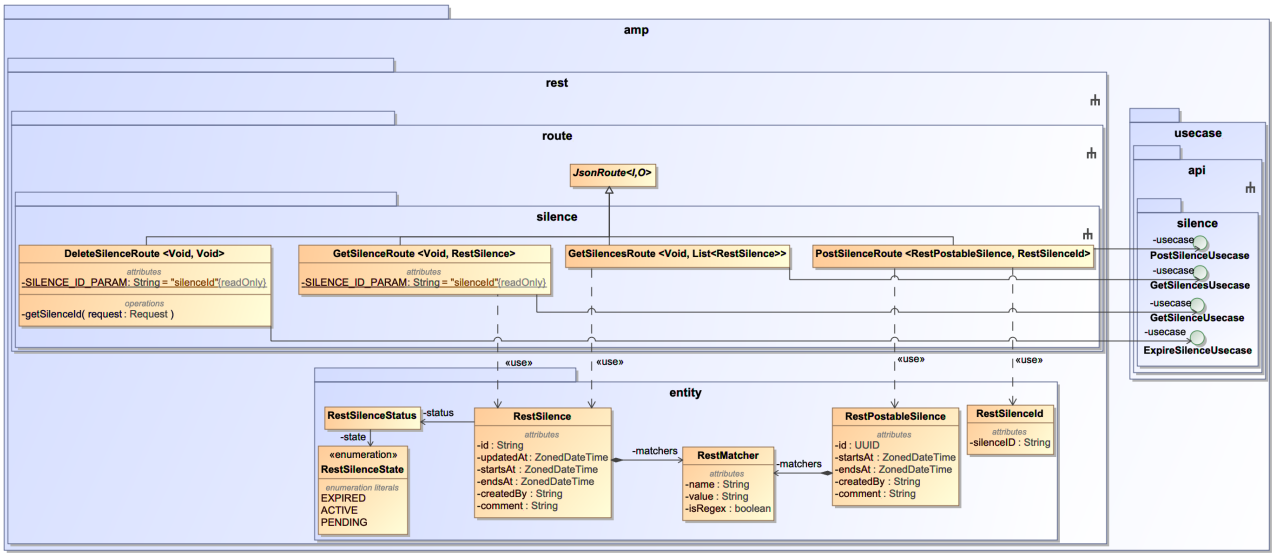
2.15 pav. Paketo lt.taurosevicius.amp.rest.route.alertgroup klasių diagrama



2.16 pav. Paketo lt.taurosevicius.amp.rest.route.general klasių diagrama



2.17 pav. Paketo lt.taurosevicius.amp.rest.route.receiver klasių diagrama



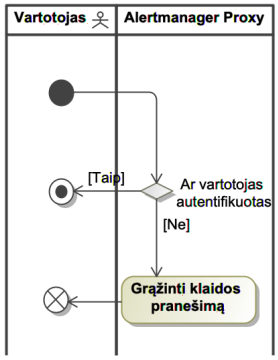
2.18 pav. Paketo lt.taurosevicius.amp.rest.route.silence klasių diagrama

2.4.3. Dinaminis Alertmanager Proxy vaizdas

Lentelėse 15-26 yra pateikiamos *Alertmanager Proxy* panaudos atvejų veiklos, atvaizduotos UML veiklos diagramomis.

15 lentelė „0. Patvirtinti autentifikaciją“ panaudos atvejo veiklos diagramos aprašymas

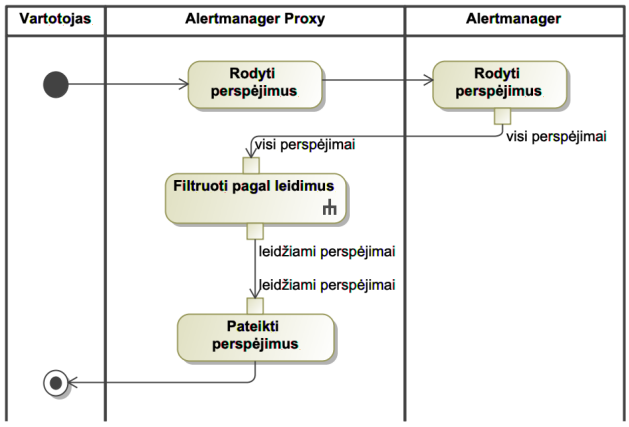
VD 0. Patvirtinti autentifikaciją		
Aprašymas Vartotojas pateikia HTTP užklausą. Programa patikrina ar užklausa turi antraštę „X-Forwarded-User“. Jeigu tokia antraštė neegzistuoja, yra grąžinama klaida. Jei egzistuoja, yra sukuriamas vartotojo objektas, kuris yra perduodamas į sekančią veiklą. Ši veikla yra vykdoma prieš veiklas 1-9.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	1. Peržiūrėti perspėjimus 2. Peržiūrėti perspėjimų grupes 3. Peržiūrėti gavėjus 4. Peržiūrėti sistemos būklę 5. Peržiūrėti tylas 6. Išsaugoti tylą 7. Sunaikinti tylą 8. Sukurti perspėjimą 9. Nukreipti užklausą į Alertmanager
Veiklos diagrama		



2.19 pav. Panaudos atvejo „0. Patvirtinti autentifikaciją“ veiklos diagrama

16 lentelė „1. Peržiūrėti perspėjimus“ panaudos atvejo veiklos diagramos aprašymas

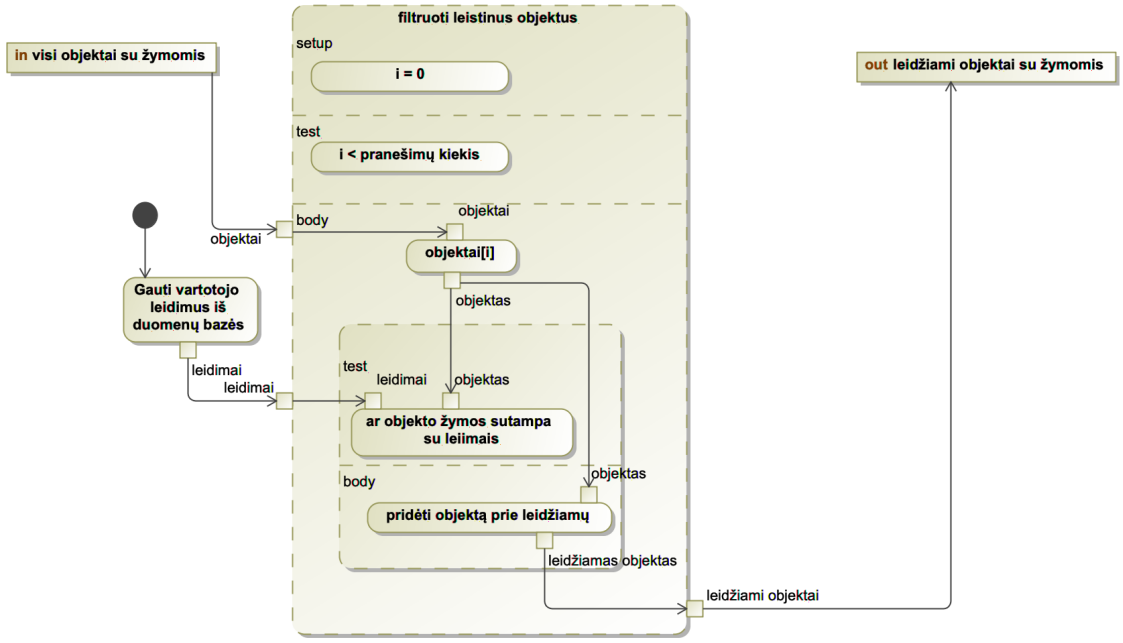
VD 1. Peržiūrėti perspėjimus		
Aprašymas Vartotojas užklausia perspėjimų. Programa užklausia perspėjimų iš *Alertmanager*. Visi perspėjimai, kurių vartotojas neturi teisių peržiūrėti, yra nufiltruojami. Filtruotas perspėjimų sąrašas pateikiamas vartotojui.		
Susijusios veiklos diagramos	Apimami PA	1.1. Filtruoti pagal leidimus
	Specializuoja PA	-
Veiklos diagrama		



2.20 pav. Panaudos atvejo „1. Peržiūrėti perspėjimus“ veiklos diagrama

17 lentelė „1.1. Filtruoti pagal leidimus“ veiklos diagramos aprašymas

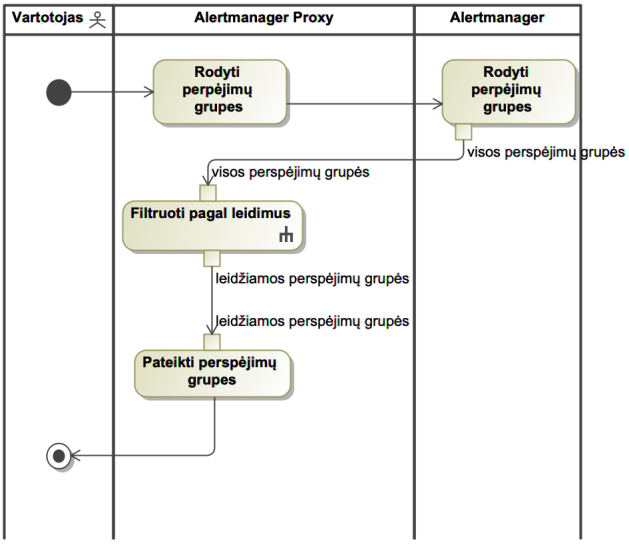
VD 1.1. Filtruoti pagal leidimus		
Aprašymas Yra perrenkamas visas objektų sąrašas. Jeigu objekto žymų sąraše pateikta padalinio žyma sutampa su vartotojo padaliniu, tai pranešimas yra įdedamas į leidžiamų objektų sąrašą. Šis sąrašas yra grąžinamas.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	1. Peržiūrėti perspėjimus, 2. Peržiūrėti perspėjimų grupes
Veiklos diagrama		



2.21 pav. „1.1. Filtruoti pagal leidimus“ veiklos diagrama

18 lentelė „2. Peržiūrėti perspėjimų grupes“ panaudos atvejo veiklos diagramos aprašymas

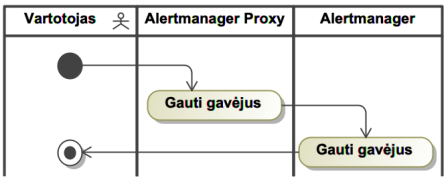
VD 2. Peržiūrėti perspėjimų grupes		
Aprašymas Vartotojas užklausia perspėjimų grupių. Programa užklausia perspėjimų grupių iš *Alertmanager*. Visos perspėjimų grupės, kurių vartotojas neturi teisių peržiūrėti, yra nufiltruojamos. Filtruotas sąrašas pateikiamas vartotojui.		
Susijusios veiklos diagramos	Apimami PA	1.1. Filtruoti pagal leidimus
	Specializuoja PA	-
Veiklos diagrama		



2.22 pav. Panaudos atvejo „2. Peržiūrėti perspėjimų grupes“ veiklos diagrama

19 lentelė „3. Peržiūrėti gavėjus“ panaudos atvejo veiklos diagramos aprašymas

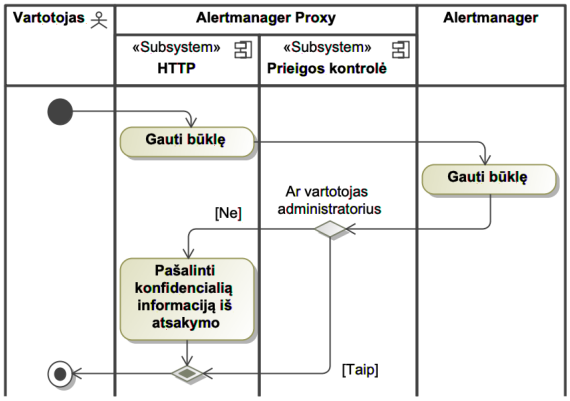
VD 3. Peržiūrėti gavėjus		
Aprašymas Programa užklausią visų gavėjų iš *Alertmanager* ir juos grąžina vartotojui.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama		



2.23 pav. Panaudos atvejo „3. Peržiūrėti gavėjus“ veiklos diagrama

20 lentelė „4. Peržiūrėti sistemos būklę“ panaudos atvejo veiklos diagramos aprašymas

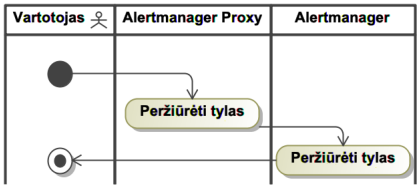
VD 4. Peržiūrėti sistemos būklę		
Aprašymas Programa užklausia *Alertmanager* būklės. Jeigu vartotojas nėra administratorius, tai dalis informacijos yra pašalinama. Apdorota būklė yra grąžinama vartotojui.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama		



2.24 pav. Panaudos atvejo „4. Peržiūrėti sistemos būklę“ veiklos diagrama

21 lentelė „5. Peržiūrėti tylas“ panaudos atvejo veiklos diagramos aprašymas

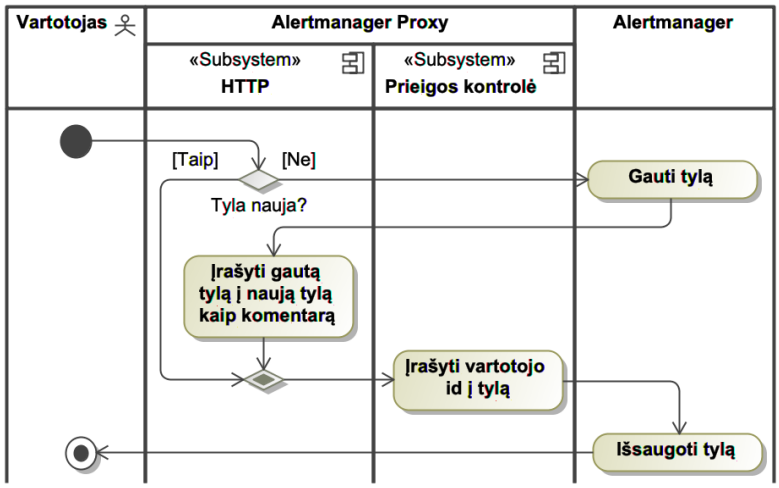
VD 5. Peržiūrėti tylas		
Aprašymas Programa užklausią tylų iš *Alertmanager* ir jas grąžina vartotojui.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama		



2.25 pav. Panaudos atvejo „5. Peržiūrėti tylas“ veiklos diagrama

22 lentelė „6. Išsaugoti tylą“ panaudos atvejo veiklos diagramos aprašymas

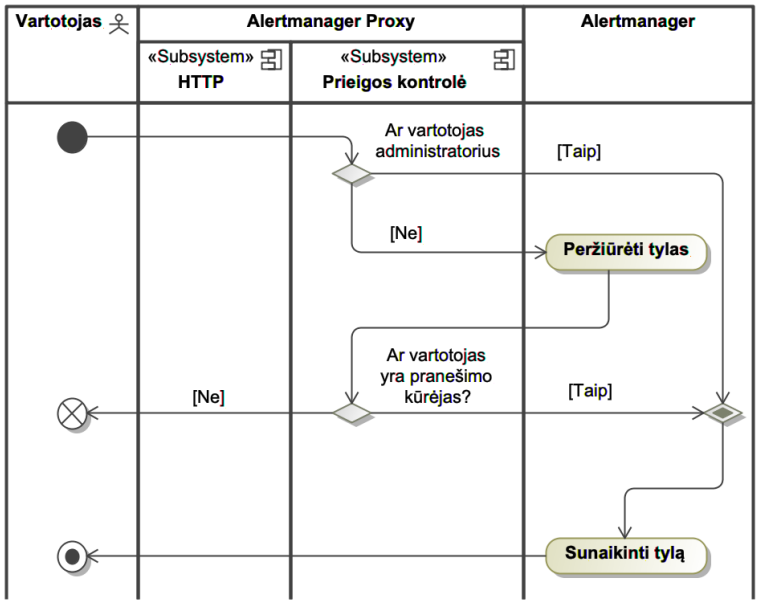
VD 6. Išsaugoti tylą		
Aprašymas Sistema patikrina ar vartotojo saugoma tyla yra nauja, ar tai yra modifikuojama jau esanti tyla. Jeigu tyla nėra nauja, tai programa gauna tylą iš *Alertmanager* ir ją įrašo į saugomą tylą kaip komentarą. Į tylos kūrėjo lauką yra įrašomas kuriančiojo vartotojo *id*, nepaisant kas prieš tam lauke buvo įrašyta. Ši tyla yra išsaugoma į programą *Alertmanager*.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama		



2.26 pav. Panaudos atvejo „6. Išsaugoti tylą“ veiklos diagrama

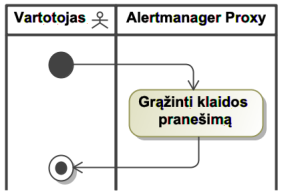
23 lentelė „7. Sunaikinti tylą“ panaudos atvejo veiklos diagramos aprašymas

VD 7. Sunaikinti tylą		
Aprašymas Tylas sunaikinti gali tik administratoriai ir tylos kūrėjai, todėl programa, gavusi tylos naikinimo užklausą, patikrina ar atliekantis veiksmą vartotojas yra administratorius. Jeigu nėra, tai sistemą užklausia tylos iš *Alertmanager* ir patikrina ar tylos kūrėjas yra tas pats vartotojas kuris atlieka tylos naikinimo operaciją. Jeigu užklausą atliekantis vartotojas nėra ir tylos kūrėjas, tai jam yra grąžinamas klaidos pranešimas. Jeigu vartotojas atitinką nors vieną anksčiau minėtą sąlygą, tai tyla yra sunaikinama iš *Alertmanager*.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama		

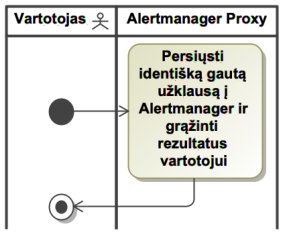


2.27 pav. Panaudos atvejo „7. Sunaikinti tylą“ veiklos diagrama

24 lentelė „8. Sukurti perspėjimą“ panaudos atvejo veiklos diagramos aprašymas

VD 8. Sukurti perspėjimą		
Aprašymas Programa *Alertmanager Proxy* neturi galimybės kurti naujų perspėjimų, nes jie yra kuriami tiesiogiai programoje *Alertmanager*.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama  **2.28 pav.** Panaudos atvejo „8. Sukurti perspėjimą“ veiklos diagrama		

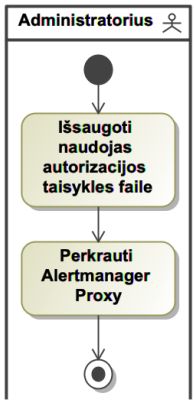
25 lentelė „9. Nukreipti užklausą į Alertmanager“ panaudos atvejo veiklos diagramos aprašymas

VD 9. Nukreipti užklausą į Alertmanager		
Aprašymas Jeigu per HTTP sąsaja siųsta užklausa buvo neatpažinta, tai užklausą reikia persiųsti programai *Alertmanager*. Šiuo būdu bus pasiekiama programos *Alertmanager* tinklinė (angl. *web*) vartotojo sąsaja.		
Susijusios veiklos diagramos	Apimami PA	-
	Specializuoja PA	-
Veiklos diagrama  **2.29 pav.** Panaudos atvejo „9. Nukreipti užklausą į Alertmanager“ veiklos diagrama		

26 lentelė „11. Valdyti autorizacijos taisykles“ panaudos atvejo veiklos diagramos aprašymas

VD 11. Valdyti autorizacijos taisykles
Aprašymas Prieigos taisyklės yra saugomos *csv* formato faile, failo vieta yra nurodoma programos paleidimo metu konfigūracijos faile. Norėdamas pakeisti autorizacijos taisykles, administratorius turi redaguoti šį failą ir iš naujo paleisti programą *Alertmanager Proxy*. Autorizacijos taisyklių užkrovimas yra vykdomas programos paleidimo metu, o programa vidutiniškai pasileidžia per 6 sekundės. Dėl šių priežasčių buvo nuspręsta prototipe neįgyvendinti panaudos atvejo „10. Perkrauti autorizacijos taisykles“.

Susijusios veiklos diagramos	Apimami PA	10. Perkrauti autorizacijos taisykles
	Specializuoja PA	-
Veiklos diagrama		



2.30 pav. Panaudos atvejo „11. Valdyti autorizacijos taisykles“ veiklos diagrama

3. Eksperimentai

Sukurtai sistemai atliktas kokybinis ir kiekybinis įvertinimas. Kiekybinė analizė atlikta statinės analizės ir testavimo pavidalu ir yra pateikta skyrelyje 3.1. Kokybinė analizė atlikta kaip eksperimentai ir pateikta skyrelyje 3.2.

3.1. Testavimas

3.1.1. Testavimo planas

Programos testavimas buvo atliekamas keliais būdais:

1. statinė analizė;
2. automatiniai vienetų testai visiems panaudos atvejams;
3. integracinis testavimas buvo atliktas rankiniu būdu;
4. apkrovos testavimas.

3.1.2. Testavimo kriterijai

Alertmanager Proxy statinei analizei buvo pasirinkta naudoti du įrankius:

1. Integruotą kūrimo aplinką *IntelliJ IDEA*, kuri turi virš 800 taisyklių *Java* programavimo kalbai [30] bei įvairias klaidas ir pranešimus parodo realiu laiku – rašant programinį kodą
2. Kodo kokybės ir saugumo įrankį *SonarQube*, kuris atlieka statinę analizę naudodamas 562 kodo taisykles *Java* kalbai [31] bei atvaizduoja programos kodo padengimą testais.

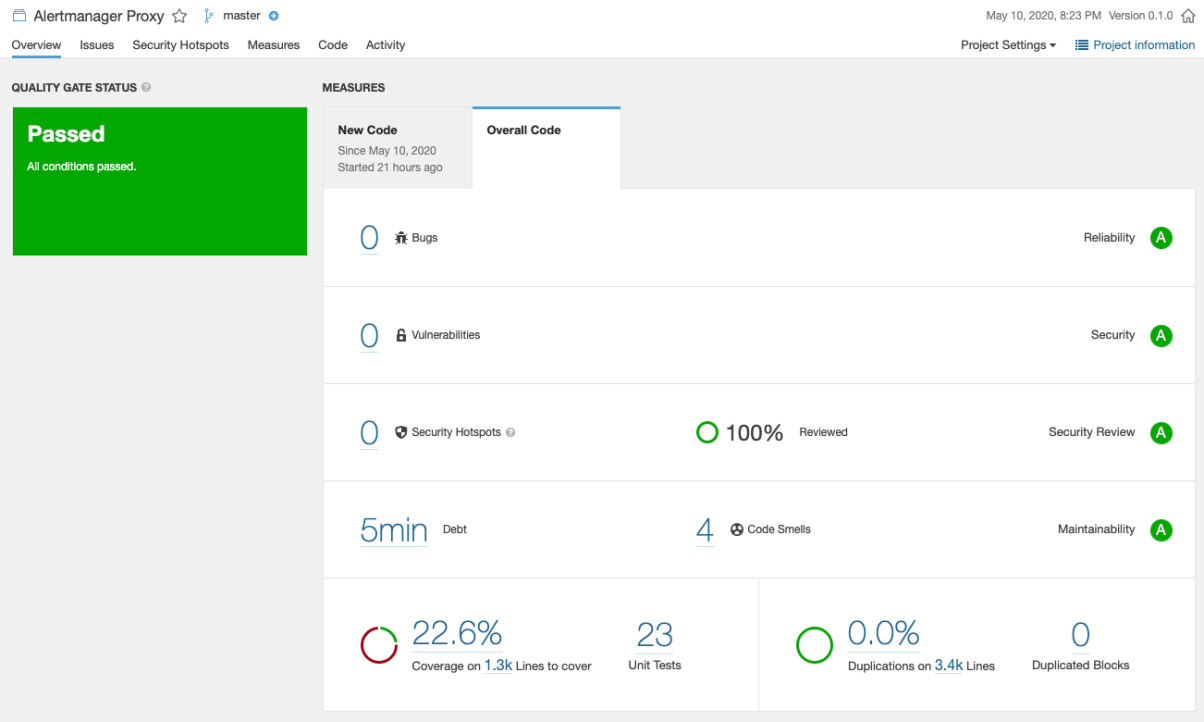
Vienetų testai leidžiami kiekvieną kartą paleidžiant programą kūrimo aplinkoje, todėl jie turėjo būti greiti. Buvo nuspręsta, kad bendra jų veikimo trukmė neturėtų viršyti vienos minutės.

Ne mažiau nei 80% panaudos atvejų kodo turi būti padengta testais.

Nefunkcinio reikalavimo dėl atsakymo laiko per 324 milisekundes (skyrelis 2.2.6) atitikimas buvo testuojamas naudojant apkrovos testavimą.

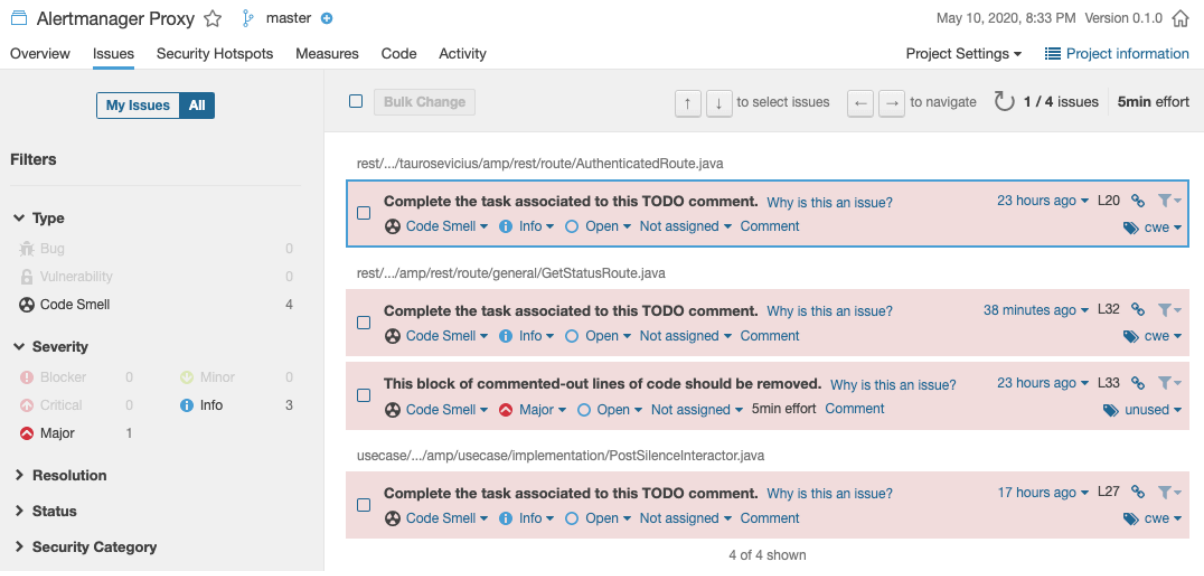
3.1.3. Statinė analizė

Statinė analizė buvo vykdoma viso programos kūrimo metu. Jos rastos problemos buvo peržiūrimos ir tvarkomos. 3.1 pav. pateikta *SonarQube* ekrano kopija su analizės rezultatais programos kūrimo pabaigoje.



3.1 pav. SonarQube rezultatai Alertmanager Proxy

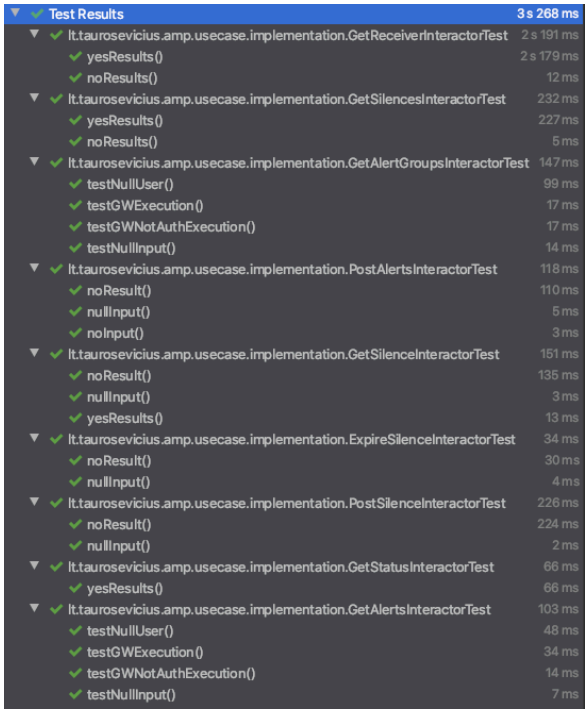
3.2 pav. rodomi statinės analizės metu rasti dviejų rūšių defektai – palikti TODO tipo komentarai ir paliktas užkomentuotas kodas.



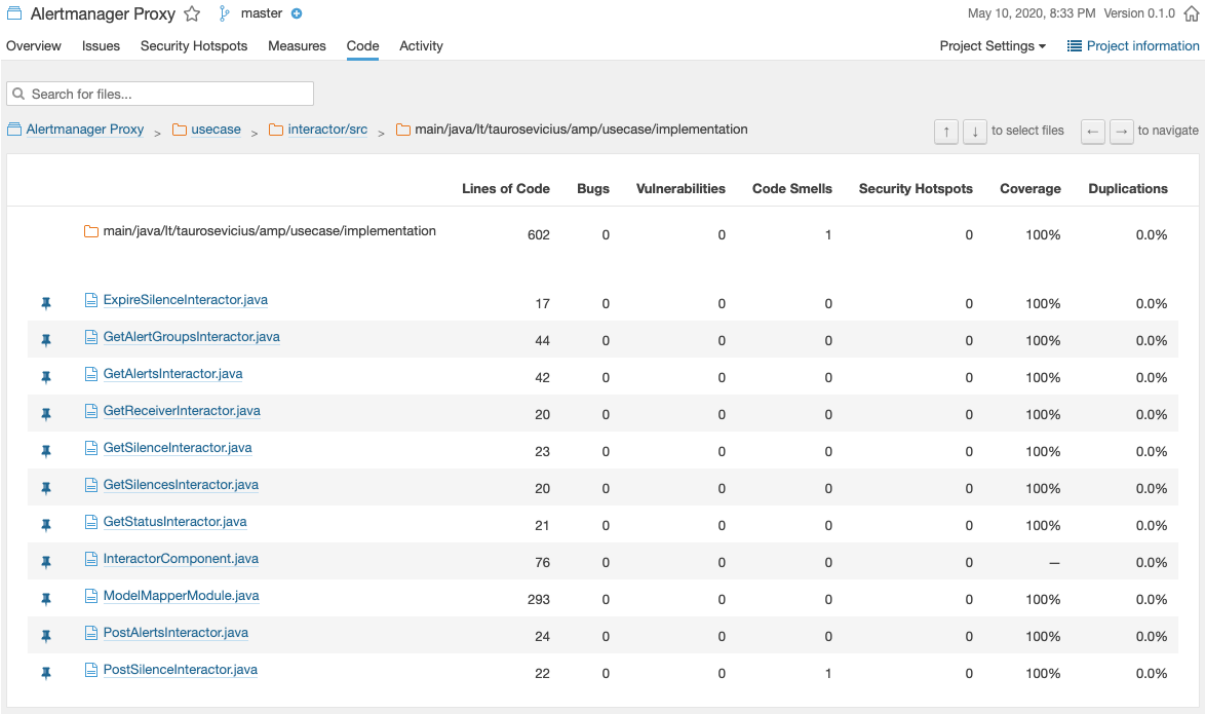
3.2 pav. SonarQube rasti defektai

3.1.4. Vienetų testavimas

Vienetų testai panaudos atvejams (paketui *lt.taurosevicius.amp.usecase.implementation*) buvo rašomi kartu su pačios programos kodu naudojant *Test Driven Development* praktiką, todėl buvo pasiektas 100% programinio kodo padengimas testais. Testų rezultatai pateikti 3.3 pav., o kodo padengimas pateiktas paveiksle 3.4.



3.3 pav. Paketo lt.taurosevicius.amp.usecase.implementation vienetų testų vykdymas



3.4 pav. Klasių pakete lt.taurosevicius.amp.usecase.implementation padengimas testais

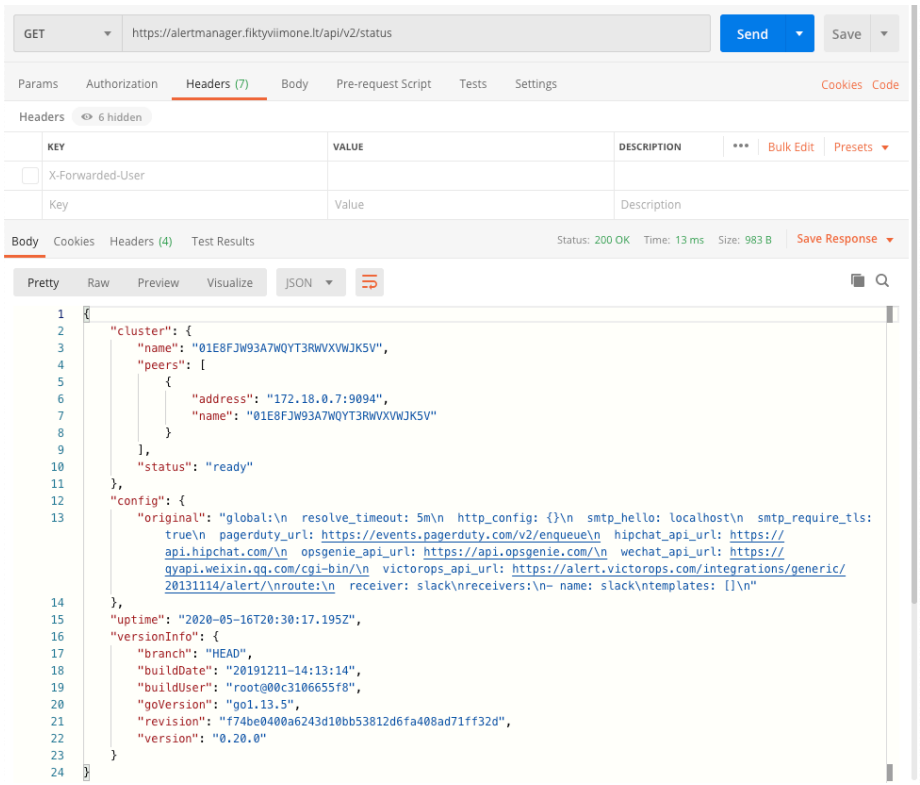
3.1.5. Rankinis testavimas

Testuojant rankiniu būdu buvo naudojama programa *Postman.* Taip pat testavimo procesą labai palengvino *Alertmanager* aprašas OpenAPI standartu, kuris buvo importuotas į *Postman*.

Testavimui buvo paleista visa sistema ir buvo daromos užklausos į programas *Alertmanager* bei *Alertmanager* Proxy ir buvo lyginami jų rezultatai.

Vienas iš rankinių testų buvo gauti būklę:

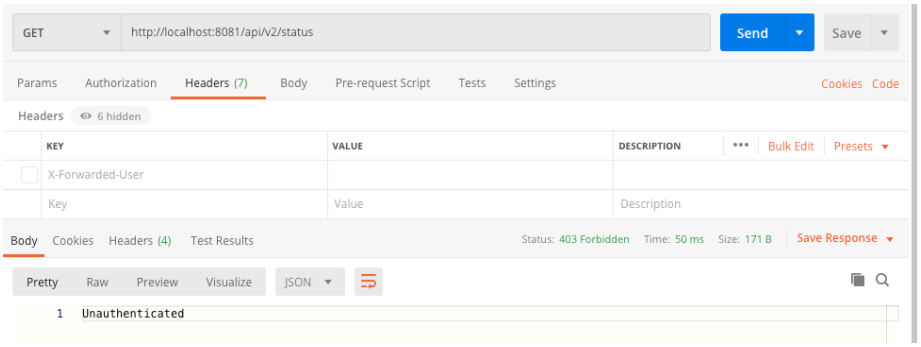
Iš pradžių buvo atlikta užklausa į paleistą *Alertmanager*, kuris buvo pasiekiamas adresu https://alertmanager.fiktyviimone.lt/. Užklausa ir rezultatai pateikti 3.5 pav**.** Šiame paveiksle matome, kad atsakymo būklė yra „200 OK“ ir kūnas (angl. *body*) yra json tipo dokumentas kuriame yra pateikta *Alertmanager* informacija.



3.5 pav. Gauti būklę užklausa ir rezultatai iš Alertmanager

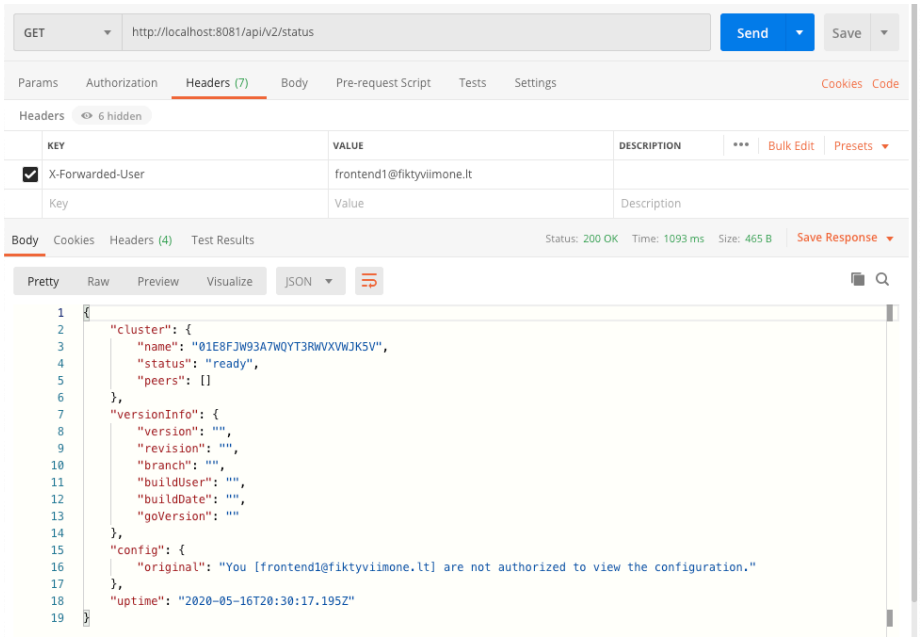
Tada buvo paleista *Alertmanager Proxy* programa, kuri buvo sukonfigūruota naudoti prieš tai minėtą *Alertmanager* kaip šaltinį.

Toliau buvo atlikta užklausa į *Alertmanager Proxy* (3.6 pav.). Atsakymo būklė „403 Forbidden“, o užklausos kūne yra tekstas „Unauthenticated“ o tai rodo, kad užklausa buvo neautentifikuota.



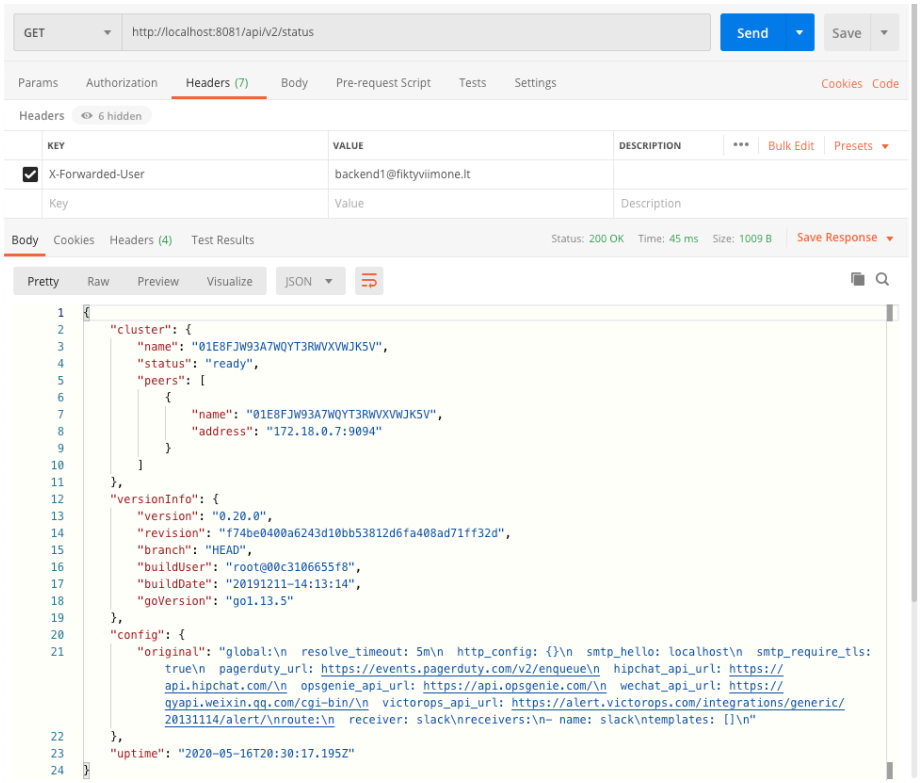
3.6 pav. Gauti būklę užklausa ir rezultatai iš Alertmanager Proxy

Sekančiame teste darėme užklausą į *Alertmanager Proxy* kaip autentifikuotas tačiau neautorizuotas vartotojas. Į užklausą pridėjome antraštę „X-Forwarded-User“ su sistemoje neautorizuoto vartotojo el. pašto adresu „frontend1@fiktyviimone.lt“. 3.7 pav. pateiktame atsakyme matome, kad užklausos kodas yra „200 OK“ ir yra pateiktas dokumentas su *Alertmanager* būkle, tačiau ji skiriasi nuo pradinės reikšmės 3.5 pav. Šiame atsakyme yra pateiktas tik *Alertmanager* pavadinimas, jo būklė ir nuo kada veikia *Alertmanager*, konfigūracijos vietoje yra pateiktas tekstas, kad vartotojas daręs užklausą nėra autorizuotas peržiūrėti konfigūraciją.



3.7 pav. Gauti būklę užklausa ir rezultatai iš Alertmanager Proxy jungiantis su neautorizuotu vartotoju

Paskutinis testas buvo atlikta į *Alertmanager Proxy* naudojant autorizuotą vartotoją „backend1@fiktyviimone.lt“. Užklausos rezultatai pateikti 3.8 pav. Atsakymo būklė taip pat yra „200 OK“, o kūne yra pateiktas tas pats dokumentas, kaip ir pirmoje užklausoje į *Alertmanager*.



3.8 pav. Gauti būklę užklausa ir rezultatai iš Alertmanager Proxy jungiantis su autorizuotu vartotoju

3.1.6. Apkrovos testavimas

Alertmanager Proxy programos stabilumui ir naudojamų resursų kiekiui išmatuoti buvo atliktas apkrovos testavimas.

Testavimui buvo pasirinkti 4 panaudos atvejai:

1. peržiūrėti perspėjimų grupes;
2. peržiūrėti sistemos būklę;
3. išsaugoti tylą;
4. peržiūrėti tylas.

Naudojant HTTP API kiekvienas iš šių panaudos atvejų buvo užklausiamas 10 tūkstančių kartų. Tos pačios užklausos buvo siunčiamos ir tiesiai į *Alertmanager*.

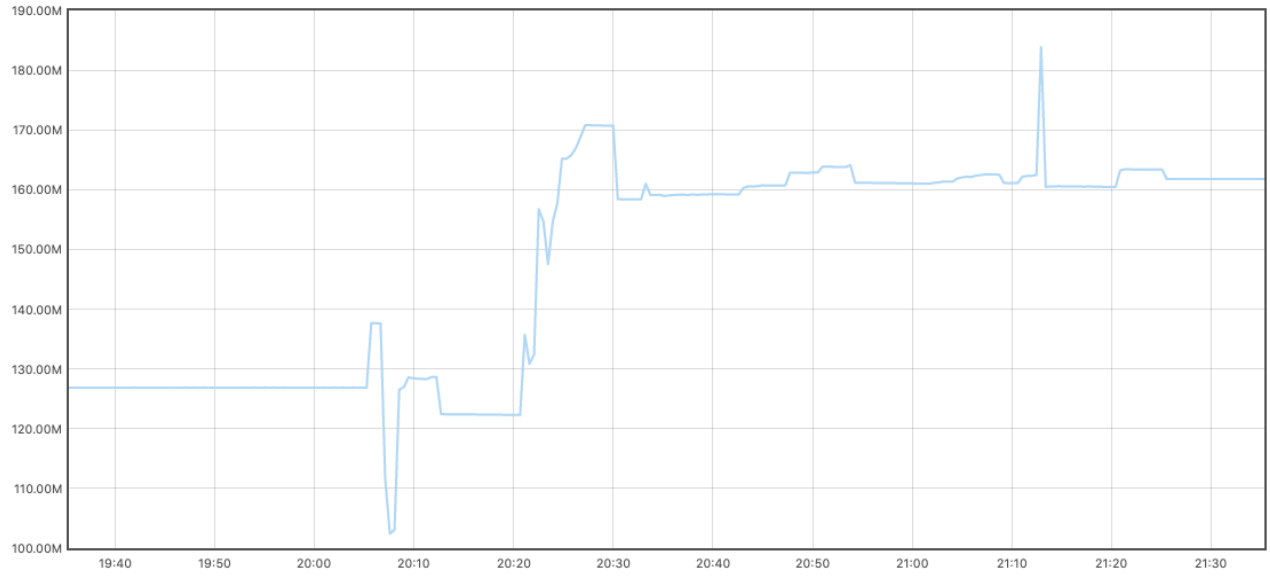
Testavimo metu nei vienas sistemos komponentas nesustreikavo, visos HTTP užklausos grąžino atsakymą „200 OK“. 27 lentelėje pateiktos vidutinės užklausų trukmės, bei kiek kartų užklausos užtruko ilgiau kreipiantis tiesiogiai į *Alertmanager*. Iš gautų rezultatų galima daryti išvadą, kad Alertmanager Proxy užklausas gali apdoroti iki 5 kartų ilgiau, nei Alertmanager, tačiau net ir ilgiausios užklausos atsakė 1,8 karto greičiau nei apibrėžta nefunkciniame reikalavime – 324 milisekundės.

27 lentelė Alertmanager Proxy apkrovos testavimo rezultatai

Testas	Alertmanager Proxy, ms	Alertmanager, ms	Užtruko ilgiau, kartai
Peržiūrėti perspėjimų grupes	43,3796	9,3731	4,6281
Peržiūrėti sistemos būklę	35,3539	18,2894	1,933

Išsaugoti tylą	171,5877	99,6358	1,7221
Peržiūrėti tylas	32,2223	10,8441	2,9714

Taip pat testavimo metu buvo sekamas programos naudojamos atminties kiekis, rezultatai pateikti 3.9 pav. Testavimas buvo pradėtas 20 valandą ir 5 minutės ir buvo vykdomas iki 21:27. Iš grafiko pastebime, kad sunaudotos atminties kiekis testavimo metu neviršijo 190 MB.



3.9 pav. Alertmanager Proxy naudojamos atminties kiekis apkrovos teso metu

3.2. Eksperimentas

Norint išbandyti įrankio saugą buvo įvykdyti trys eksperimentai. Šiame poskyryje pateikti jų rezultatai.

Pasiruošimas

Eksperimento metu buvo įdiegta sistema fiktyviai įmonei. Ji teikia savo paslaugas internete naudodama vieną interneto programą (angl. *web-service*), kuri yra vadinama *Monitored app*. Ji yra pasiekiama adresu https://fiktyviimone.lt/.

Ši programa jau veikė *Docker Swarm* klasteryje, todėl į jį buvo įdiegta ir visa stebėjimo sistema:

- *Alertmanager Proxy*
- *Alertmanager*
- *Traefik*
- *Traefik Forward Auth*
- *Simple Oidc provider*
- *Prometheus*

Programų diegimo konfigūracija pateikta pirmame priede.

Monitored app buvo papildyta metrikų atidavimu *Prometheus* suprantamu formatu, o *Prometheus* stebėjo šią programą. Ši programa atiduoda trejas metrikas:

- **orders_total** – Skaitiklio (angl. *counter*) tipo rodiklis, rodo kiek programoje buvo padaryta užsakymų nuo paleidimo;
- **order_total_errors** – Skaitiklio tipo rodiklis, rodo kiek programoje įvyko klaidų nuo paleidimo. Turi tris vertes:
    - **validation** – klaidos atliekant užsakymo validaciją, dažniausiai atsitinka dėl blogos suprogramuotos logikos;
    - **payment** – vartotojui nepavyko sumokėti už užsakymą – natūrali eiga, tačiau naudinga vedant statistiką;
    - **shipping** – klaida pristatant užsakymą pirkėjui;
- **user_sessions** – Matuoklės (angl. *gauge*) tipo rodiklis rodo kiek programoje yra prisijungusių vartotojų.

Šias metrikas naudosime sukurtos sistemos išbandymui *Prometheus* sistemoje sukonfigūruodami 3 perspėjimus:

- **order_validation_errors** – per daug užsakymų su validacijos klaidomis – prieinama vidinės dalies komandai;
- **order_shipping_errors** – Per daug klaidų pristatant užsakymą – prieinama aptarnavimo komandai;
- **low_order_over_5min** – Per mažai užsakymų – prieinama išorinės dalies programuotojų komandai.

Įmonėje yra 3 padaliniai, kiekviename iš padalinių dirba po 2 darbuotojus:

- vidinės dalies (angl. *back-end*) programavimo padalinys;
- išorinės dalies (angl. *front-end*) programavimo padalinys;
- klientų aptarnavimo (angl. *support*) padalinys.

Sukurtoje sistemoje kiekvienas padalinys atspindės komandą. Administratoriais bus visi vidinės dalies programuotojai. Šių rolių priskyrimai vartotojams yra pateikti lentelėje 28.

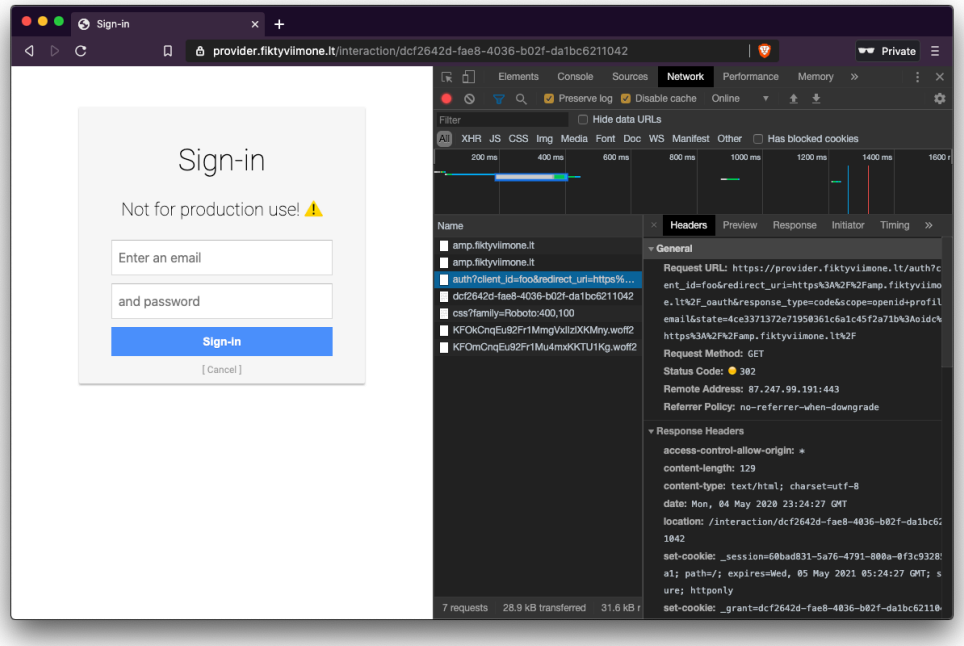
28 lentelė Fiktyvios įmonės rolių priskyrimai darbuotojams

***Rolės pavadinimas* / vartotojo id**	**Priklauso rolei**
back-end	admin
backend1@fiktyviimone.lt	back-end
backend2@fiktyviimone.lt	back-end
frontend1@fiktyviimone.lt	front-end
frontend2@fiktyviimone.lt	front-end
support1@fiktyviimone.lt	support
support 2@fiktyviimone.lt	support

Vartotojų ir rolių priskyrimas naudotas šiame eksperimente *casbin* formatu yra pateiktas antrame priede.

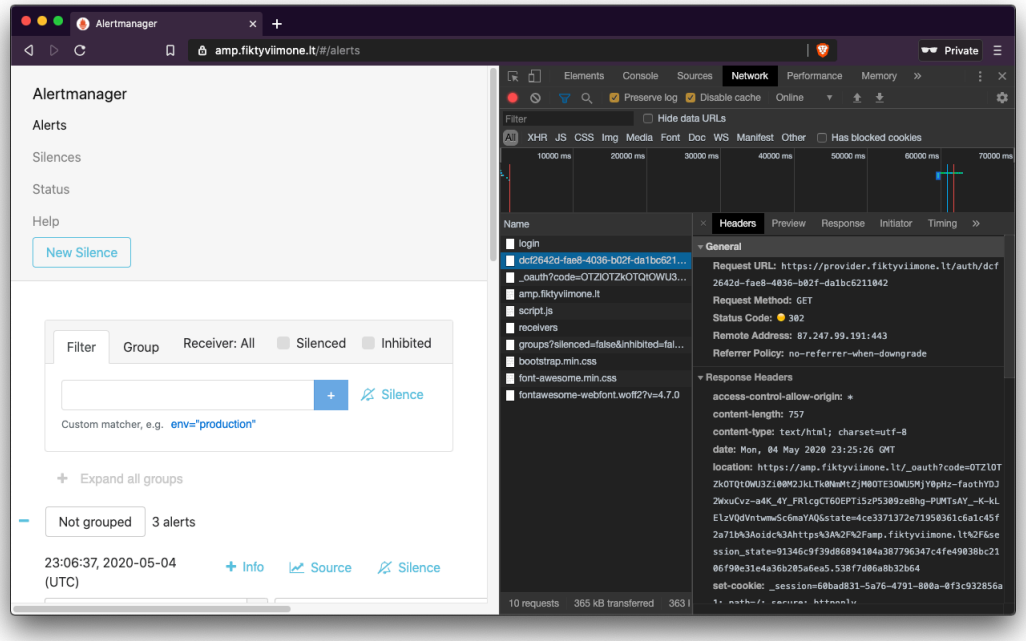
Sistemos paleidimas. Sudarius visų programų konfigūraciją, jos buvo paleistos *Docker Swarm* klasteryje. Praėjus trims minutėms nuo programos paleidimo *Alertmanager Proxy* buvo pasiekiamas adresu https://amp.fiktyviimone.lt/. Iš karto buvo pradėti vykdyti eksperimentai.

3.2.1. Prisijungimas



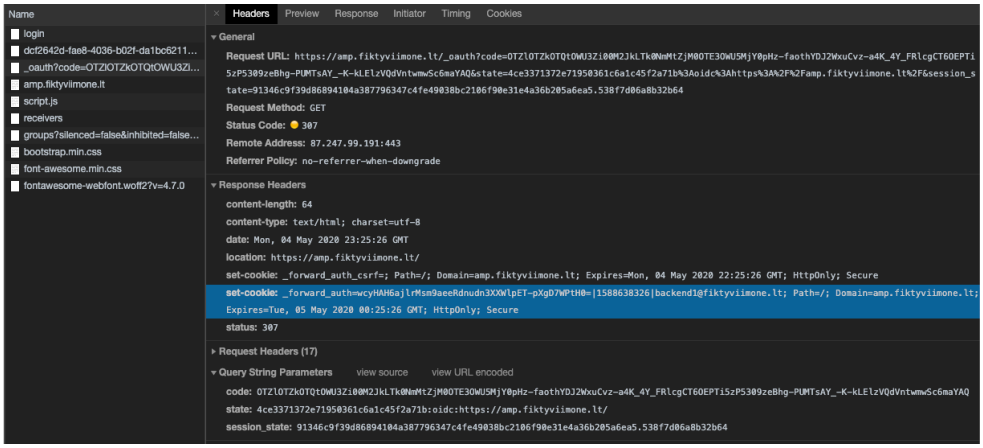
3.10 pav. Prisijungimo langas į Alertmanager proxy ir atliktų užklausų sąrašas

1. Vartotojas eina adresu http://amp.fiktyviimone.lt
2. Naršyklė nukreipiama į šifruotą prieigą per https: https://amp.fiktyviimone.lt
3. *Traefik Forward Auth* patikrina ar vartotojas neprisijungęs ir tada jį nukreipia į prisijungimą: https://provider.fiktyviimone.lt/auth ir į užklausos parametrus prideda visą reikalingą informaciją.
4. *Simple-oidc-provider* pateikia vartotojui prisijungimo langą.
5. Vartotojas įveda savo vartotojo vardą bei slaptažodį ir spaudžia *Sign-In* (liet. prisijungti). Pavyzdys ir užklausų seka pateikta 3.10 pav.



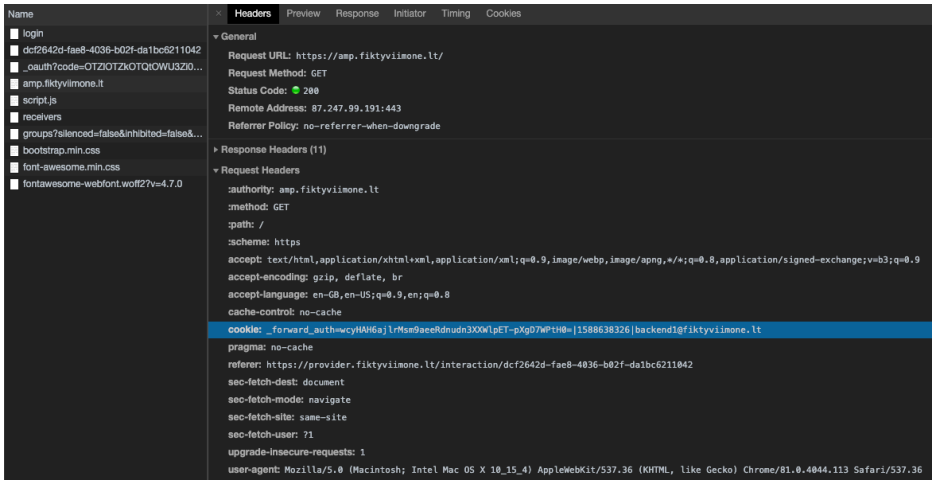
3.11 pav. Alertmanager proxy pradinis langas ir atliktų užklausų sąrašas nuo prisijungimo mygtuko paspaudimo

6. *Simple-oidc-provider* patikrina ar teisingi vartotojo pateikti duomenys ir jį nukreipia atgal į *Alertmanager Proxy* (3.11 pav.) https://amp.fiktyviimone.lt/_oauth ir užklausos parametruose pateikia autentifikacijos parametrus.



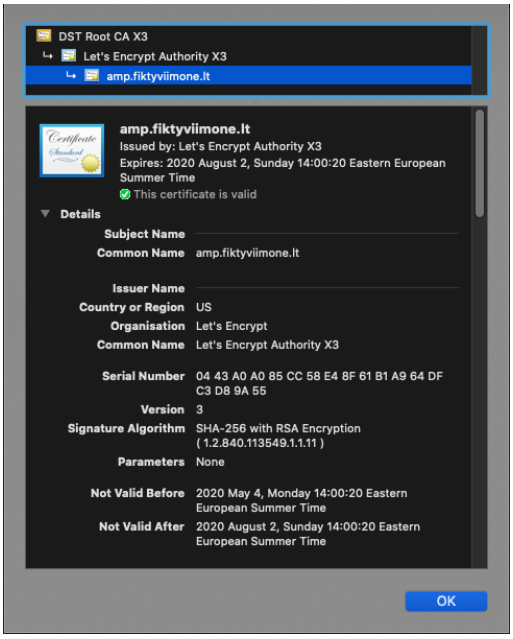
3.12 pav. Traefik-forward-auth atsakymo į užklausą detalės

7. Keliu https://amp.fiktyviimone.lt/_oauth yra pasiekiamas *Traefik Forward Auth*, kuris patikrina ar yra teisingi pateikti autentifikacijos parametrai ir naršyklėje įrašo slapukus (angl. *cookie*), kurie toliau bus naudojami autentifikacijai patvirtinti ir galiausiai vartotojas yra nukreipiamas į pradinį puslapį https://amp.fiktyviimone.lt (3.12 pav.).



3.13 pav. Užklausos į Alertmanager proxy analizė

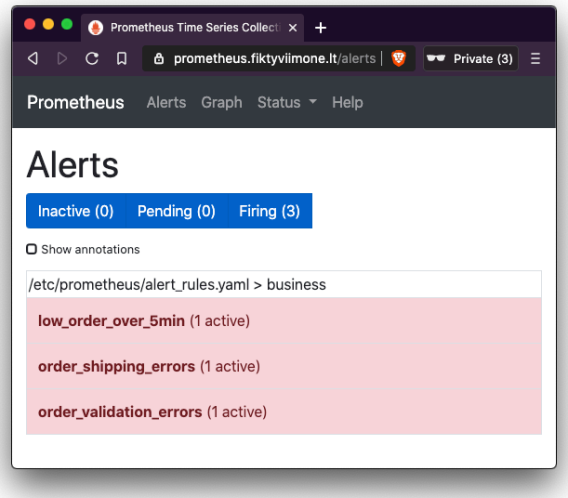
8. Naršyklė, kaip ir antrame žingsnyje, atidaro puslapį https://amp.fiktyviimone.lt, tačiau jau užklausoje siunčia slapuką **_forward_auth** (3.13 pav.), kuriame yra įrašyti autentifikacijos duomenys. Užklausą patikrina *Traefik Forward Auth* įskiepis, ir ji yra perduodama pačiam *Alertmanager Proxy* ir vartotojui yra rodomas *Alertmanager* langas.
9. Atsidarius programai patikriname domeno TLS sertifikatą (3.14 pav.). Sertifikatas yra galiojantis.



3.14 pav. Programos, pasiekiamos adresu amp.fiktyviimone.lt sertifikato informacija

3.2.2. Pranešimų peržiūra

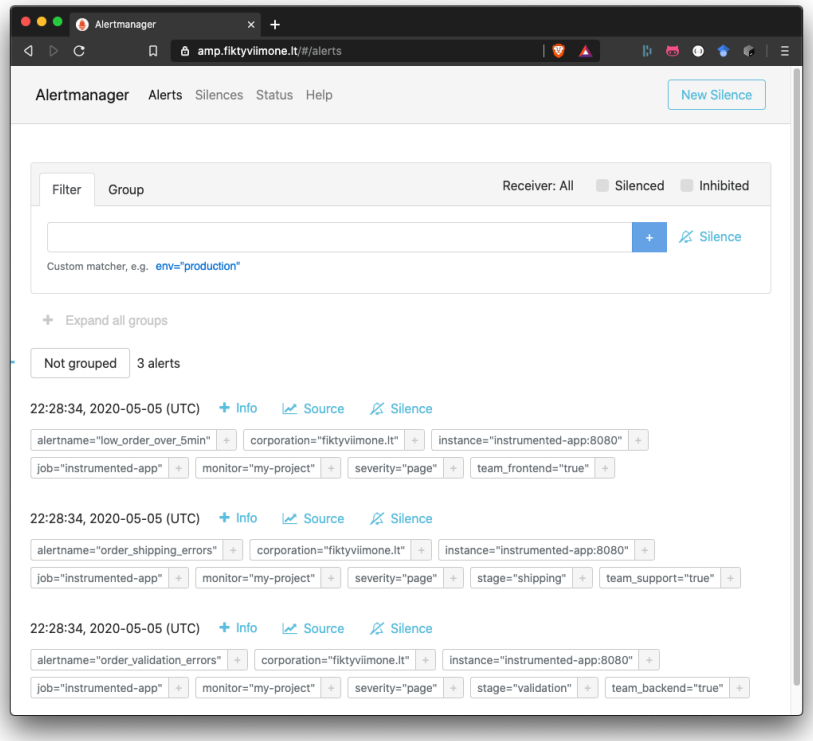
Kaip minėta sistemoje buvo sukurti 3 skirtingi perspėjimai – kiekvienai komandai po vieną. Taip pat sistemoje buvo padarytos dirbtinės sąlygos, kad visi šie pranešimai būtų aktyvioje būsenoje (angl. *firing*). Prisijungę prie *Prometheus* 3.15 paveiksle matome, kad visi 3 pranešimai yra aktyvioje būsenoje.



3.15 pav. Pranešimai Prometheus programoje

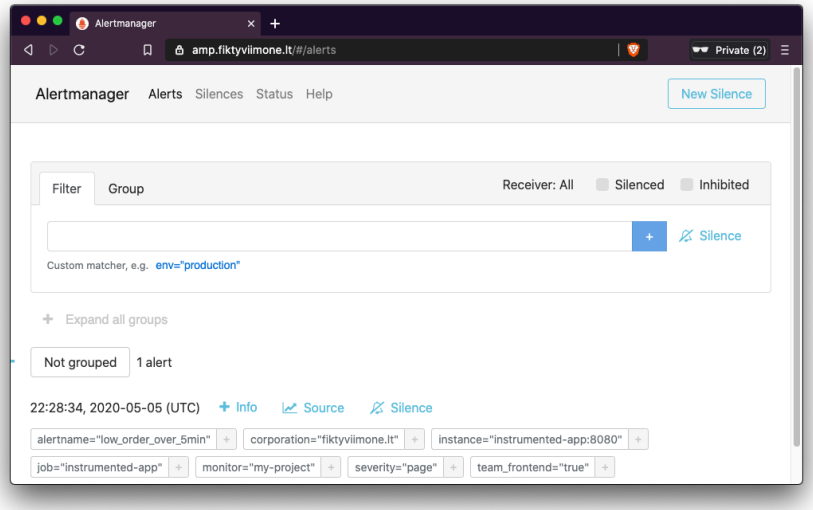
Į sistemą *Alertmanager Proxy* prisijungiame su 3 vartotojais:

1. Vidinės dalies programuotojų grupei priklausantis darbuotojas mato visus pranešimus (3.16 pav.), nes jis yra ir sistemos administratorius;



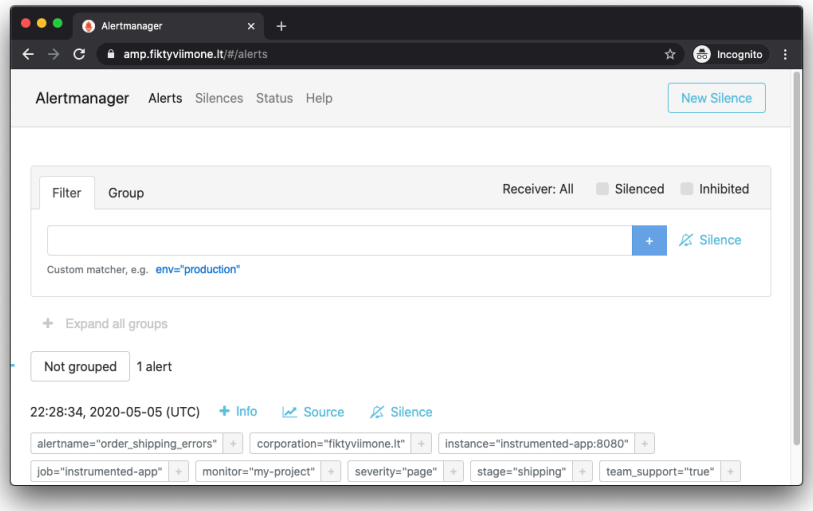
3.16 pav. Backend grupės vartotojo matomi pranešimai

2. Išorinės dalies programuotojų grupei priklausantis darbuotojas mato tik 1 perspėjimą (3.17 pav.);



3.17 pav. Frontend grupės vartotojo matomi pranešimai

3. Klientų aptarnavimo grupei priklausantis darbuotojas taip pat mato tik 1 perspėjimą, tačiau skirtingą nuo išorinės dalies programuotojų grupės darbuotojo (3.18 pav.):

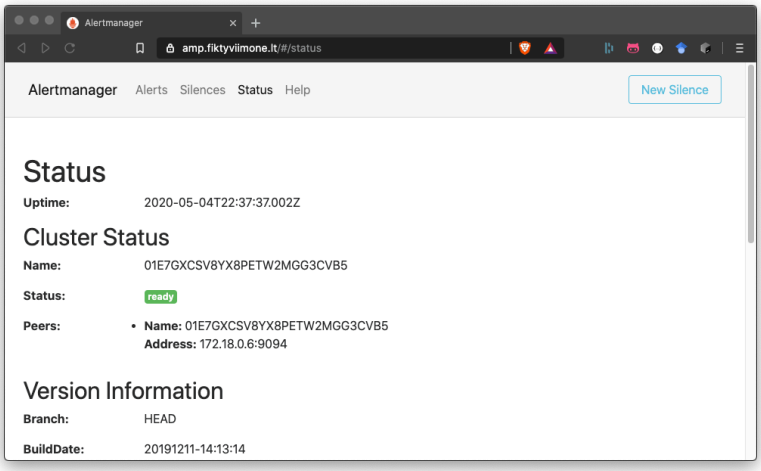


3.18 pav. Support grupės vartotojo matomi pranešimai

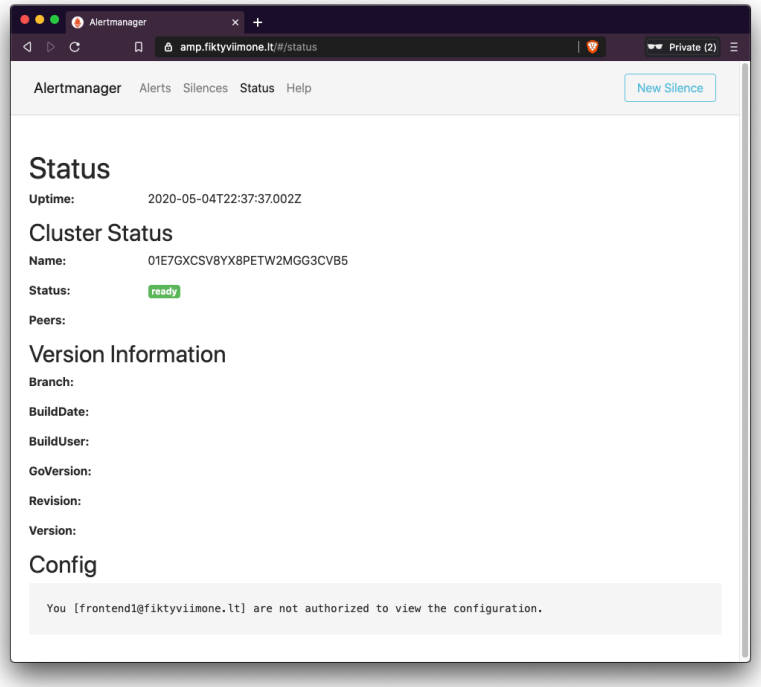
3.2.3. Būklės peržiūra

Šiame bandyme atlikome tokį patį eksperimentą kaip ir rankinio testavimo skyrelyje 3.1.5.

Nuėję į būklę (angl. *status*) su backend1 vartotoju gauname prieigą prie visos konfigūracijos (3.19 pav.), o su frontend1 vartotoju matome tik dalį informacijos ir pranešimą, kad nesame autorizuoti (3.20 pav.).



3.19 pav. Backend grupės vartotojo matoma Alertmanager būklė ir konfigūracija



3.20 pav. Frontend grupės vartotojo matoma Alertmanager būklė ir konfigūracija

Ateities darbai

1. Prototipe įgyvendinti panaudos atvejį „10. Perkrauti autorizacijos taisykles“
2. Atskleisti programos kodą įkeliant jį į versijų kontrolės sistemą *Github*.
3. Paruošti programos diegimo ir konfigūracijos dokumentaciją *Docker Swarm* ir *Kubernetes* platformoms, ją taip pat publikuoti su programos kodu.
4. Palaikyti sukurta programą ir atnaujinti kartu su *Alertmanager* programa.
5. Sukurti naują projektą – padaryti autorizacijos sluoksnį pačiam *Prometheus*.

Išvados

1. Atlikus debesų paslaugų stebėjimo sistemų saugos problemų analizę buvo pastebėtas *Prometheus* sistemos trūkumas – sistema neturi identiteto valdymo, todėl buvo nuspręsta sukurti įrankį kuris įgyvendins autentifikacijos ir autorizacijos mechanizmus *Prometheus* sistemos dalyje *Alertmanager*.
2. Išanalizavus prieigos valdymo mechanizmus buvo padaryta išvada, kad kuriamai sistemai labiausiai tiktų autentifikacijos mechanizmas naudojant API prieigos vartų šabloną, o autorizacijai – rolėmis grįsta prieigos kontrolė.
3. Kuriant programos *Alertmanager Proxy* projektą, buvo sukurtas programos autorizacijos modelis.
4. Atlikus apkrovos testavimą pastebėta, kad sistema naudoja tik 200 MB vykdomosios atminties, todėl programą galima naudoti debesų kompiuterijos sistemose, kur yra labai riboti resursai. Taip pat programa yra pakankamai greita – vidutiniškai į pateiktas užklausas atsako greičiau nei per 180 milisekundžių.
5. Turint sistemos konfigūraciją, užregistruotą DNS įrašą ir veikiantį Docker Swarm klasterį, sistemos diegimas užtruko 3 minutes, todėl galima teigti, kad programa yra greitai diegiama.
6. Atlikus eksperimentus nustatyta, kad sukurtas programos prototipas padidina *Prometheus Alertmanager* saugumą įgyvendindamas CIA triadą.
7. Programoje *Alertmanager* konfidencialumas yra užtikrinamas naudojant *Alertmanager Proxy* ir jame suprogramuotą rolėmis grįstą prieigos kontrolę.
8. Duomenų vientisumo kriterijus užtikrintas naudojant TLS šifravimą, kuris įgyvendintas naudojant automatinį TLS sertifikatų išdavimą pasinaudojant *Traefik* API prieigos vartus.

Literatūros sąrašas

1. TRAKADAS, Panagiotis, et al. Scalable monitoring for multiple virtualized infrastructures for 5G services. Iš: SoftNetworking 2018, The International Symposium on Advances in Software Defined Networking and Network Functions Virtualization. 2018. p. 1-4
2. PROMETHEUS. Security model [interaktyvus]. 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://prometheus.io/docs/operating/security/
3. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Special Publication 800-145. The NIST definition of cloud computing. U. S. Department of Commerce, 2011
4. KOHN, Dan, et al. CNCF Cloud Native Definition [interaktyvus], 2018 [žiūrėta 2020 m. gegužės 2 d.] Pasiekiama: https://github.com/cncf/toc/blob/master/DEFINITION.md
5. IBM CLOUD EDUCATION. Cloud Monitoring [interaktyvus]. 2019 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.ibm.com/cloud/learn/cloud-monitoring
6. VARIA, Jinesh, MATHEW, Sajee. Overview of Amazon Web Services [interaktyvus]. Amazon Web Services, 2018 [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://media.amazonwebservices.com/AWS_Overview.pdf
7. AMAZON WEB SERVICES, CloudWatch extends Metrics retention and new User Interface [interaktyvus], 2016, [žiūrėta 2020 m. gegužės 2 d.] Pasiekiama: https://aws.amazon.com/about-aws/whats-new/2016/11/cloudwatch-extends-metrics-retention-and-new-user-interface/
8. NAGIOS, Notifications [interaktyvus], 2016 [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/notifications.html
9. NAGIOS, Understanding retention.dat and modified_attributes [interaktyvus], 2016 [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://support.nagios.com/kb/article.php?id=522&show_category=36
10. PANDORA FMS TEAM, 9 reasons not to install Nagios in your company [interaktyvus], 2016, [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://pandorafms.com/blog/9-reasons-not-to-install-nagios-in-your-company/
11. NAGIOS, Authentication and Authorization In The CGIs [interaktyvus], 2016 [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/cgiauth.html
12. PROMETHEUS. Overview [interaktyvus]. 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://prometheus.io/docs/introduction/overview/
13. PROMETHEUS. Alertmanager [interaktyvus]. 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://prometheus.io/docs/alerting/alertmanager/
14. PROMETHEUS. Alertmanager Configuration [interaktyvus]. 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://prometheus.io/docs/alerting/configuration/#receiver
15. ELASTIC, What is the ELK Stack? [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.elastic.co/what-is/elk-stack
16. ELASTIC, Action and connector types [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.elastic.co/guide/en/kibana/7.6/action-types.html
17. ELASTIC, Authentication in Kibana [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.elastic.co/guide/en/kibana/7.6/kibana-authentication.html
18. ELASTIC, Role-based access control [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.elastic.co/guide/en/kibana/7.6/development-security-rbac.html

19. HE, Xiuyu ir YANG, Xudong, 2017. Authentication and Authorization of End User iš Microservice Architecture. Journal of Physics: Conference Series [interaktyvus]. spalio 2017. Vol. 910, p. 12060. DOI 10.1088/1742-6596/910/1/012060. Prieiga per interneta: http://dx.doi.org/10.1088/1742-6596/910/1/012060
20. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Special Publication 800-162. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. U. S. Department of Commerce, 2011
21. MICROSOFT. Sidecar pattern [interaktyvus]. 2017 [žiūrėta 2020 m. gegužės 2 d.]. Prieiga per internetą: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar
22. GOEBELBECKER, Eric. Configuration as Code: Everything You Need to Know [interaktyvus], 2018 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://rollout.io/blog/configuration-as-code-everything-need-know/
23. ATLASSIAN. What is configuration as code? [interaktyvus], 2018 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://confluence.atlassian.com/bamboo/what-is-configuration-as-code-894743909.html
24. INDEN, Max. Alertmanager OpenAPI specification. 2019 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://github.com/prometheus/alertmanager/blob/v0.20.0/api/v2/openapi.yaml
25. CASBIN. Casbin [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://casbin.org/
26. CASBIN. jCasbin [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://github.com/casbin/jcasbin
27. LITTLEDATA. What is the average server response time? [interaktyvus] 2020 [žiūrėta 2020 m. gegužės 7 d.] Prieiga per internetą: https://www.littledata.io/average/server-response-time
28. SEDDON, Thom, Traefik Forward Auth [interaktyvus]. 2018 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://github.com/thomseddon/traefik-forward-auth/blob/v2.1.0/README.md#forwarded-headers
29. MARTIN, Robert C., Clean Architecture: A Craftsman's Guide to Software Structure and Design.Prentice Hall Press, ISBN 978-0-13-449416-6
30. JETBRAINS. List of Java inspections [interaktyvus], 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://www.jetbrains.com/help/idea/list-of-java-inspections.html
31. SONARSOURCE. Java Rules [interaktyvus], 2020 [žiūrėta 2020 m. gegužės 2 d.] Prieiga per internetą: https://rules.sonarsource.com/java

Priedai

1 priedas. Sistemos diegimo konfigūracija į Docker Swarm klasterį

```
version: "3.8"
volumes:
  prometheus_data: {}
services:
  traefik:
    image: traefik:v2.2
    command:
    - "--accesslog.format=json"
    - "--accesslog=true"
    - "--accesslog.filepath=/config/access.log"
    - "--providers.docker.swarmmode"
    - "--providers.docker.exposedByDefault=false"
    - "--providers.docker.network=prom_default"
    - "--entryPoints.web.address=:80"
    - "--entryPoints.websecure.address=:443"
    - "--certificatesresolvers.letsencrypt.acme.httpchallenge=true"
    - "--certificatesresolvers.letsencrypt.acme.httpchallenge.entrypoint=web"
    - "--certificatesresolvers.letsencrypt.acme.email=benas.taurosevicius@ktu.edu"
    - "--certificatesresolvers.letsencrypt.acme.storage=/config/acme.json"
    volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - "./traefik:/config"
    ports:
    - "80:80"
    - "443:443"
    deploy:
      labels:
      - "traefik.enable=true"
      - "traefik.http.routers.traefik.entrypoints=websecure"
      - "traefik.http.routers.traefik.tls.certresolver=letsencrypt"
      - "traefik.http.routers.traefik.rule=Host(`traefik.fiktyviimone.lt`)"
      - "traefik.http.routers.traefik.service=api@internal"
      - "traefik.http.services.dummy-svc.loadbalancer.server.port=9999"
      # global forward-auth middleware
      - "traefik.http.middlewares.forward-auth-
verify.forwardauth.address=http://forward-auth:4181"
      - "traefik.http.middlewares.forward-auth-
verify.forwardauth.trustForwardHeader=true"
      - "traefik.http.middlewares.forward-auth-
verify.forwardauth.authResponseHeaders=X-Forwarded-User"
      # global redirect to https
      - "traefik.http.routers.catch-all.rule=hostregexp(`{host:.+}`)"
      - "traefik.http.routers.catch-all.entrypoints=web"
      - "traefik.http.routers.catch-all.middlewares=redirect-to-https"
      - "traefik.http.middlewares.redirect-to-https.redirectscheme.scheme=https"
  forward-auth:
    image: thomseddon/traefik-forward-auth:2.1.0
    environment:
    - "PROVIDERS_OIDC_ISSUER_URL=https://provider.fiktyviimone.lt"
    - "PROVIDERS_OIDC_CLIENT_ID=foo"
    - "PROVIDERS_OIDC_CLIENT_SECRET=bar"
    - "SECRET=thequickbrownfoxjumpsoverthelazydog"
    - "AUTH_HOST=auth.fiktyviimone.lt"
    - "COOKIE_DOMAIN=.fiktyviimone.lt"
    - "LIFETIME=3600"
    - "DEFAULT_PROVIDER=oidc"
    deploy:
```

```
      labels:
      - "traefik.enable=true"
      - "traefik.http.routers.forward-auth.entrypoints=websecure"
      - "traefik.http.routers.forward-auth.tls.certresolver=letsencrypt"
      - "traefik.http.routers.forward-auth.rule=Host(`auth.fiktyviimone.lt`)"
      - "traefik.http.routers.forward-auth.middlewares=forward-auth-verify"
      - "traefik.http.services.forward-auth.loadbalancer.server.port=4181"

  oidc-provider:
    image: qlik/simple-oidc-provider
    environment:
    - "REDIRECTS=\
      https://auth.fiktyviimone.lt/_oauth,\
      https://prometheus.fiktyviimone.lt/_oauth,\
      https://amp.fiktyviimone.lt/_oauth"
    - "IDP_NAME=https://provider.fiktyviimone.lt"
    - "USERS_FILE=/simple-oidc-provider/users.json"
    - "PORT=80"
    volumes:
    - "./simple-oidc-provider:/simple-oidc-provider"
    deploy:
      labels:
      - "traefik.enable=true"
      - "traefik.http.routers.oidc-provider.entrypoints=websecure"
      - "traefik.http.routers.oidc-provider.tls.certresolver=letsencrypt"
      - "traefik.http.routers.oidc-provider.rule=Host(`provider.fiktyviimone.lt`)"
      - "traefik.http.services.oidc-provider.loadbalancer.server.port=80"
  prometheus:
    image: prom/prometheus
    volumes:
    - ./prometheus/:/etc/prometheus/
    - prometheus_data:/prometheus
    command:
    - "--config.file=/etc/prometheus/prometheus.yaml"
    - "--storage.tsdb.path=/prometheus"
    - "--web.console.libraries=/usr/share/prometheus/console_libraries"
    - "--web.console.templates=/usr/share/prometheus/consoles"
    deploy:
      labels:
      - "traefik.enable=true"
      - "traefik.http.routers.prometheus.rule=Host(`prometheus.fiktyviimone.lt`)"
      - "traefik.http.routers.prometheus.entrypoints=websecure"
      - "traefik.http.routers.prometheus.tls.certresolver=letsencrypt"
      - "traefik.http.routers.prometheus.middlewares=forward-auth-verify"
      - "traefik.http.services.prometheus.loadbalancer.server.port=9090"
  monitored-app:
    image: monitored_app
    command:
    - "--listen=0.0.0.0:8080"
    deploy:
      labels:
      - "traefik.enable=true"
      - "traefik.http.routers.app.rule=Host(`fiktyviimone.lt`)"
      - "traefik.http.routers.app.entrypoints=websecure"
      - "traefik.http.routers.app.tls.certresolver=letsencrypt"
      - "traefik.http.services.app.loadbalancer.server.port=8080"
  alertmanager:
    image: prom/alertmanager
    volumes:
    - "./alertmanager/:/etc/alertmanager/"
    ports:
    - "127.0.0.1:9093:9093" # For testing only
```

```
    command:
    - "--config.file=/etc/alertmanager/config.yaml"
    - "--storage.path=/alertmanager"
  alertmanager-proxy:
    image: alertmanager-proxy:latest
    environment:
    - "CONFIG_PATH=/amp/config.yaml"
    volumes:
    - "./amp:/amp"
    ports:
    - "8081:8081"
    deploy:
      resources:
        limits:
          memory: 200mb
      labels:
      - "traefik.enable=true"
      - "traefik.http.services.amp.loadbalancer.server.port=8081"
      - "traefik.http.routers.amp.rule=Host(`amp.fiktyviimone.lt`)"
      - "traefik.http.routers.amp.entrypoints=websecure"
      - "traefik.http.routers.amp.tls.certresolver=letsencrypt"
      - "traefik.http.routers.amp.middlewares=forward-auth-verify"
```

2 priedas. Fiktyvios įmonės *Alertmanager Proxy* programos autorizacijos politikos casbin csv formatu

```
p, admin, admin, execute
p, backend, backend, execute
p, frontend, frontend, execute
p, support, support, execute

g, backend, admin

g, backend1@fiktyviimone.lt, backend
g, backend2@fiktyviimone.lt, backend
g, frontend1@fiktyviimone.lt, frontend
g, frontend2@fiktyviimone.lt, frontend
g, support1@fiktyviimone.lt, support
g, support2@fiktyviimone.lt, support
```