KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

GINTARAS VOLKVIČIUS

PROBLEMŲ SPRENDIMŲ PASITELKIANT KVANTINIUS ALGORITMUS PASLAUGŲ SERVISAS

Baigiamasis bakalauro projektas

Vadovas
doc. dr. T. Blažauskas

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

PROBLEMŲ SPRENDIMŲ PASITELKIANT KVANTINIUS ALGORITMUS PASLAUGŲ SERVISAS

Baigiamasis bakalauro projektas

Programų sistemos (kodas 612I30002)

Vadovas
doc. dr. T. Blažauskas

_____
(data, parašas)

Recenzentas
K. Ryselis

_____
(data, parašas)

Projektą atliko
G. Volkvičius

_____
(data, parašas)

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

Gintaras Volkvičius

(Studento vardas, pavardė)

Programų sistemos (kodas 612I30002)

„Baigiamojo projekto pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Gintaro Volkvičiaus**, baigiamasis projektas tema „Problemų sprendimų pasitelkiant kvantinius algoritmus paslaugų servisas“ 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)*

TECHNINĖ UŽDUOTIS

Sukurti uždavinių/problemų sprendimų saityno paslaugų servisą, kuris jas spręstų pasitelkdamas kvantinius algoritmus. Jos paskirtis – supažindinti naudotojus su problemomis ir jų sprendimo algoritmais, kurie gali pasinaudoti kvantiniams kompiuteriams skirtais algoritmais. Šie algoritmai gali paspartinti problemų išsprendimo laiką, lyginant su tik klasikiniams kompiuteriams skirtais algoritmais. Šis paslaugų servisas apima šių problemų sprendimo realizacijas:

1. Sveiko skaičiaus skaidymas pirminiais daugikliais;
2. Diskretaus logaritmo skaičiavimas.

Prie bendros kuriamos sistemos kartu su minėtu saityno paslaugų servisu įeina ir asmeninio sertifikavimo paslaugų teikėjo servisas, skirtas autentifikuoti ir autorizuoti problemų sprendimo saityno paslaugų servisą, bei saityno taikomoji programa, skirta problemų sprendimo saityno paslaugų serviso aprašymui bei demonstravimui.

Sistemoje turi būti realizuotas šis funkcionalumas:

1. Problemų sprendimo užklausų kūrimas su norimais problemų parametrais;
2. Problemų sprendimo gavimas;
3. Problemų sprendimo eigos paaiškinimai;
4. Asinchroninis problemų sprendimas;
5. Galimybė atšaukti problemų sprendimą;
6. Naudotojų autentifikacija ir jų prieigos prie resursų teisių tikrinimas.

Dėl duomenų saugumo, visos sistemos dalys turi būti sukonfigūruotos naudoti ir tarpusavyje bendrauti HTTPS protokolu.

Volkvičius, Gintaras. Problemų sprendimų pasitelkiant kvantinius algoritmus paslaugų servisas. Bakalauro baigiamasis projektas / vadovas doc. dr. Tomas Blažauskas; Kauno technologijos universitetas, informatikos fakultetas.

Mokslo kryptis ir sritis: fiziniai mokslai, informatikos kryptis.

Reikšminiai žodžiai: API, kvantiniai algoritmai, kriptografija, matematika.

Kaunas, 2018. 53 p.

SANTRAUKA

Darbe pristatomas problemų sprendimo paslaugų servisas pasitelkiantis kvantinius algoritmus. Įvade apžvelgiamas darbo aktualumas, iškeliamas tikslas bei tam tikslui pasiekti skirti uždaviniai. Darbo aktualumo tema plėtojama analizės dalyje, kuriame nusakomas bendras sistemos veiklos tikslas. Šioje dalyje taip pat nagrinėjami rinkoje esantys konkurentai bei pasirinktos problemos bei jų sprendimo algoritmai.

Projektavimo dalyje pateikiami sistemos funkciniai ir nefunkciniai reikalavimai, sistemos apribojimai ir reikalavimai techninei įrangai. Po to apžvelgiamos technologijos, naudotos projektavimo metu bei analizuojamas sistemos projektas.

Testavimo dalyje pristatomas testavimo planas bei detalizuojamas jo išpildymas bei rezultatai.

Dokumentacijos dalyje pateikiama sistemos techninė dokumentacija, paprastas naudotojo vadovas bei kaip reiktų įdiegti bei administruoti sistemą.

Darbo pabaigoje pateikiamas rezultatų apibendrinimas ir išvados.

Volkvičius, Gintaras. Problem Solving Service that Uses Quantum Algorithms: Bachelor's thesis / supervisor assoc. prof. Tomas Blažauskas. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: physical sciences, informatics.

Keywords: API, quantum algorithms, cryptography, mathematics.

Kaunas, 2018. 53 p.

SUMMARY

A problem solving service that uses quantum algorithms is presented in the thesis. The paper begins with introduction, which includes the thesis' relevance, its main goal and tasks, how this goal should be achieved. These topics are further reviewed in analysis section, which includes the main objective of the system. This section also includes competitors analysis and analysis of selected problems and their algorithms.

System's modelling section includes functional and non-functional requirements, as well as restrictions and technical requirements of the system. After that, used technologies are revised. Finally, the system's analysis is presented.

Testing section includes testing plan and its execution, as well as execution results.

System's documentation section includes system's technical and user-friendly documentation, as well as system deployment and maintenance guidelines.

At the end of the paper results and conclusions of the thesis are presented.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

- API (angl. *Application Programming Interface*) – taikomųjų programų programavimo sąsaja. Tai sąsaja, suteikianti programuotojui naudotis programos, teikiančios API, funkcijomis.
- HTTP (angl. *HyperText Transfer Protocol*) – protokolas, skirtas keistis informacija pasauliniame tinkle.
- OAuth – standartas, aprašantis naudotojų autorizaciją.
- GCD (angl. *Greatest Common Divisor*) – didžiausias bendras daliklis.
- DBVS – duomenų bazių valdymo sistema.
- UML (angl. *Unified Modeling Language*) – modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

ĮVADAS

Pastaruoju metu vis daugiau kalbų sukasi apie kvantinius kompiuterius bei kaip jie greitu metu galės padėti spręsti problemas, kurios su klasikiniais kompiuteriais yra neįmanomos. Taip pat kalbama apie tai, jog atėjus kvantinių kompiuterių erai, visos dabartinės kriptografinės sistemos tiesiog žlugs, kadangi jų raktai galės būti „nulaužti“ per keletą valandų ar dienų, kai tuo tarpu dabar klasikiniams kompiuteriams prireiktų dešimčių tūkstančių metų. Kvantiniuose kompiuteriuose bus vykdomi specialūs kvantiniai algoritmai, kurie išnaudos kvantines mechanikos fenomenus kvantiniuose kompiuteriuose. Tačiau kaip tie kvantiniai algoritmai veikia, kaip jie vykdomi? Norint atsakyti į šį klausimą ir kuriama darbe aprašoma sistema – problemų sprendimų pasitelkiant kvantinius algoritmus paslaugų servisas.

Darbo tikslas – susipažinti ir supažindinti naudotojus su problemomis, kurių sprendimai gali būti pagreitinti naudojant kvantinius algoritmus, bei jų sprendimo būdais pasitelkiant šiuos algoritmus. Tikslui pasiekti iškelti tokie uždaviniai:

1. Ištirti panašias rinkoje esančias sistemas;
2. Išanalizuoti pasirinktas problemas ir jų sprendimus pasitelkiant kvantinius algoritmus;
3. Sudaryti kuriamos sistemos projektą;
4. Realizuoti sistemą pagal sukurtą sistemos projektą;
5. Ištestuoti sistemą;
6. Paruošti sukurtos sistemos dokumentaciją.

Darbas susideda iš analizės, sistemos projektavimo, testavimo ir dokumentacijos skyrių. Analizės skyriuje ištiriamos rinkoje esančios panašios sistemos, išanalizuojamos pasirinktos problemos bei jų sprendimo algoritmai, taip pat aptariamas sistemos kūrimo tikslas bei pagrįstumas. Sistemos projektavimo skyriuje pristatomas sistemos projektas, sistemos statinis ir dinaminis vaizdas, funkciniai ir nefunkciniai sistemos reikalavimai, projektavimo metodika bei naudotos technologijos. Testavimo skyriuje pristatomas sudarytas testavimo planas bei jo įgyvendinimo rezultatai. Dokumentacijos skyrius susideda iš sukurto API specifikacijos, sukurtos saityno taikomosios programos naudotojo vadovo, visos sistemos diegimo ir priežiūros dokumentacijos.

Dokumento pabaigoje pateikiamas rezultatų apibendrinimas ir išvados.

1. ANALIZĖ

1.1. Techninis pasiūlymas

1.1.1. Sistemos apibrėžimas

Problemų sprendimų pasitelkiant kvantinius algoritmus paslaugų servisas – tai saityno paslaugų servisas, sprendžiantis konkrečias problemas , pasitelkdamas kvantinius algoritmus ir juos vykdydamas kvantinio kompiuterio simuliatoriuje. Spręsdamas problemas servisas paaiškina kiekvieną algoritmo žingsnį, taip leisdamas įsigilinti, kaip šios problemos sprendžiamos pasitelkiant kvantinius algoritmus.

1.1.2. Bendras veiklos tikslas

Bendras veiklos tikslas – susipažinti ir supažindinti sistemos naudotojus su problemomis, kurių sprendimai gali būti pagreitinti naudojant kvantinius algoritmus, bei jų sprendimo būdais pasitelkiant šiuos algoritmus.

1.1.3. Sistemos pagrįstumas

Informacijos apie kvantinius kompiuterius, algoritmus ir kaip jie problemas, kurias klasikinis kompiuteris spręstų tūkstančius metų, kvantinis kompiuteris galėtų išspręsti per keletą dienų, yra daug. Į šias problemas įeina ne tik jau minėtos realizuojamos sveikojo skaičiaus skaidymo pirminiais daugikliais [1] bei diskretaus logaritmo skaičiavimo [2] problemos, o taip pat ir molekulinių procesų kvantinės mechanikos modeliavimo [3] bei reakcijos mechanizmų sudėtingose cheminėse sistemose išsiaiškinimo [4] problemos. Ir tai tikrai nėra visos problemos, kurias kvantiniai kompiuteriai gali ar numanomai galės išspręsti eksponentiškai greičiau, nei klasikiniai kompiuteriai. Tačiau klausimas – kaip? Yra parašyta mokslinių publikacijų, aiškinančių algoritmus bei jų žingsnius [5] [6], tačiau jos yra gana moksliškos ir jas suprasti yra išties sudėtinga.

Todėl šis paslaugų servisas ir yra sukurtas taip, jog spręsdamas problemas kiekvieną žingsnį bendrai ir konkrečiame kontekste (pagal pateiktus duomenis) aprašytų kuo aiškiau ir paprasčiau. Žinoma, kai kurių matematinių operacijų, sąvokų ar sutrumpinimų paprasčiau jau ir neįmanoma parašyti. Tačiau net ir naudotojams, iš pirmo karto nesuprantantiems algoritmų žingsnių, daug medžiagos studijuoti nereikės, norint perprasti algoritmus.

1.1.4. Konkurencija rinkoje

Egzistuoja aibė įvairių portalų, skaičiuojančių skaičiaus pirminius daugiklius ar diskrečius logaritmus, pasitelkdami žinomus greičiausius algoritmus, skirtus klasikiniams kompiuteriams. Tačiau nepavyko rasti nė vieno portalo, kuris šias problemas spręstų panaudodamas algoritmus, skirtus kvantiniams kompiuteriams ar jų simuliatorius. Vienas iš portalų, pasitelkiančių klasikinius algoritmus, yra įvairių problemų sprendimų portalas, sukurtas Dario Alpern [7]. Šiame portale realizuoti tiek skaičių skaidymo pirminiais daugikliais, tiek diskretaus logaritmo skaičiavimo sprendimai.

Skaičių skaidymas pirminiais daugikliais yra realizuotas elipsinių kreivių skaičiavimo algoritmu [8]. Šio algoritmo greitis yra sub-eksponentinis, t. y. algoritmo veikimo laikas augs greičiau nei bet kokio polinominio algoritmo veikimo laikas, bet augimas vis tiek bus žymiai lėtesnis nei bet kokio eksponentinio algoritmo. Tuo tarpu mano sprendimas pasitelks P. Shor algoritmą, kurio veikimo greitis yra polinominis [5]. Tačiau dėl sudėtingumo simuliuojant kvantinį kompiuterį bei ribotų techninių galimybių, pastarasis algoritmas veiks nepalyginamai ilgiau.

Diskretaus logaritmo paieška minėtame portale įgyvendinta Pohlig–Hellman algoritmu [9], kai tuo tarpu mano sprendime bus naudojama modifikuota P. Shor algoritmo versija.

Portale be problemų sprendimo atsakymų, yra pateikiama papildoma informacija, kuri yra susijusi su sprendimu. Pavyzdžiui, skaičiuojant diskretų logaritmą, apskaičiavus pateikiamas ne tik

atsakymas (diskretus logaritmas), bet ir jo periodas (daugiau apie tai 1.3.3 skyrelyje). Taip pat vykdymo metu yra šiek tiek rodomi vykdomi žingsniai, tačiau jie niekur neišsaugomi. Be to, kiek galima pamatyti, tie žingsniai yra gan sunkiai suprantami, kadangi detalesnei paaiškinimai nepateikiami. Kita vertus, mano paslaugų servisas išsaugos visus vykdymo pranešimus, kurie yra lengvai suprantami.

Bendras abiejų sistemų palyginimas pateiktas 1.1 lentelėje.

1.1 lentelė. Rinkos konkurento ir kuriamos sistemos palyginimas

Lyginimo kriterijai	Sistema A	Kuriama sistema
Skaičiaus skaidymas pirminiai daugikliais (SSPD)	Realizuota	Realizuota
SSPD algoritmas	Elipsinių kreivių metodas	P. Shor algoritmas
Diskretaus logaritmo skaičiavimas (DLS)	Realizuota	Realizuota
DLS algoritmas	Pohlig–Hellman algoritmas	P. Shor algoritmo modifikacija
Sprendimų paaiškinimai	Iš dalies realizuota	Realizuota

1.1.5. Prototipai ir pagalbinė informacija

Kuriant visas sistemos dalis (asmeninio sertifikavimo paslaugų teikėjo servisą, saityno paslaugų servisą problemų sprendimui bei saityno taikomąją programą pastarojo serviso funkcionalumui pademonstruoti) nebus naudojamasi jokiais galutiniais produktais, tačiau bus pasinaudota įvairiais karkasais, palengvinančiais sistemos kūrimą.

Visos sistemos dalys kuriamos naudojantis *ASP.NET Core 2* karkasu, palengvinančių saityno paslaugų servisų ir taikomųjų programų kūrimą. Asmeninio sertifikavimo paslaugų teikėjo servisas realizuojamas pasinaudojus *IdentityServer4* karkasu, kuris skirtas *ASP.NET Core 2* karkasui bei kuriame yra realizuotas *OpenID Connect* standartas, paremtas *OAuth 2.0* standartu [10]. Tai palengvina asmeninio sertifikavimo paslaugų teikėjo serviso sukūrimą.

Kvantiniai skaičiavimai saityno paslaugų servise problemų sprendimui bus realizuojami pasinaudojus *Microsoft Quantum Development Kit* kūrimo įrankiu, kuris suteikia kvantinio kompiuterio / procesoriaus simuliatorių bei naują kvantiniams algoritmams aprašyti skirtą programavimo kalbą *Q#* [11]. Asinchroniniam problemų sprendimui realizuoti bus pasinaudota *Hangfire* karkasu, kuris suteikia sąsaja kurti bei valdyti foninius darbus (angl. *background jobs*).

1.1.6. Sistemos apimtis ir ištekliai, reikalingi sistemai sukurti

Kaip minėta anksčiau, visą sistemą sudaro 3 dalys:

1. Asmeninio sertifikavimo paslaugų teikėjo serviso;
2. Saityno paslaugų serviso problemų sprendimui;
3. Saityno taikomosios programos.

Asmeninio sertifikavimo paslaugų teikėjo servisas yra atsakingas už naudotojų autentifikavimą bei prieigos prie problemų saityno serviso ir taikomosios programos prieigą. Saityno paslaugų servisas yra atsakingas už problemų sprendimą asinchroniškai naudojantis kvantiniais algoritmais. Saityno taikomoji programa yra skirta pastarojo serviso funkcionalumui demonstruoti, tad ji tik perduoda naudotojo nurodytus duomenis pastarajam servisui bei apdoroja gaunamus rezultatus iš jo.

Buvo įvertinta, jog sistemai sukurti reikės apie 240 darbo valandų ir jog be automatinių testų ją turėtų sudaryti apie 7000 *C#* kodo eilučių, iš kurių daugiau nei pusė bus saityno paslaugų serviso problemų sprendimui realizacija, bei apie 100 *Q#* kodo eilučių kvantiniams algoritmams realizuoti. Sistemai sukurti užtenka vieno žmogaus.

1.2. Galimybių analizė

1.2.1. Techninės galimybės

Didžioji sistemos dalis bus realizuota taikant rinkoje patikrintas technologijas: *ASP.NET Core 2*, *IdentityServer4*, *Hangfire* karkasus. Vienintelis *Microsoft Quantum Development Kit* kūrimo įrankis kartu su *Q#* programavimo kalba yra ganėtinai naujas, pirmoji versija buvo išleista vos 2017 metų gruodį. Taip pat, tiek pirmoji, tiek dabartinė – planuojama naudoti versija – yra vadinamosios peržiūros stadijos (angl. *preview release*), t. y. kūrimo įrankis vis dar kuriamas ir gali turėti nežinomų klaidų, tačiau yra suteikta išankstinė prieiga naudotojams, norintiems išbandyti jį. Nepaisant to, manoma, jog kuriamai sistemai užteks dabartinio esamo kūrimo įrankio funkcionalumo.

Apibendrinus galima teikti, jog visos techninės galimybės yra išpildytos ir kliūčių sukurti sistemai nėra.

1.2.2. Vartotojų pasiruošimo analizė

Kuriamos sistemos saityno paslaugų serviso problemų sprendimui dalis bus API, kuris bus pasiekiamas per HTTP/HTTPS sąsają, todėl norint sėkmingai juo naudotis būtina patirtis naudojantis ar integruojant tokio tipo saityno paslaugas.

Būtent dėl to ir yra kuriama saityno taikomoji programa, kuri naudoja API ir pateikia dauguma jo funkcijų draugišku mažiau patyrusiam naudotojui būdu – per grafinę sąsają. Todėl saityno taikomąja programa galės naudotis visi naudotojai, turintys įprasto darbo su kompiuteriu patirties.

Kita vertus, dėl pačios sistemos bendro veiklos tikslo, norint suprasti pateiktas problemas bei jų sprendimų paaiškinimus reiktų turėti bent jau vidurinį išsilavinimą.

1.3. Teorinė dalis

1.3.1. Kvantiniai algoritmai

Kvantiniai algoritmai – tai yra tokie algoritmai, kurie yra vykdomi pagal kvantinių skaičiavimų modelį. Kvantiniai skaičiavimai skiriasi nuo paprastų skaičiavimų tuo, jog remiasi kvantinės mechanikos fenomenais – tokiais kaip superpozicija ir kvantinis susiejimas [12]. Lyginant su klasikiniais algoritmais, tiek vieni, tiek kiti yra sudaryti iš baigtinės instrukcijų sekos. Klasikinio algoritmo atveju, šios instrukcijos yra vykdomos mums puikiai pažystamuose klasikiniuose kompiuteriuose. Tuo tarpu kvantinių algoritmų instrukcijos yra vykdomos kvantiniuose kompiuteriuose. Klasikiniuose kompiuteriuose mažiausias informacijos vienetas yra bitas, kuris gali turėti reikšmę 0 arba 1, o kvantiniuose kompiuteriuose – kubitas (angl. *qubit*). Kubitas, priešingai nei bitas, gali įgyti ne tik 0 arba 1 reikšmę, tačiau ir bet kurią kitą reikšmę tarp šių reikšmių – tuomet kubitas būna taip vadinamoje superpozicijoje [13].

1.3.2. Sveiko skaičiaus skaidymo pirminiais daugikliais problema ir skaičiavimo algoritmas

Sveikojo skaičiaus skaidymo pirminiais daugikliais problema formuluojama taip – turint skaičių N, rasti jo pirminius daugiklius. Šiuo atveju ši problema šiek tiek konkretizuojama – turint skaičių N, rasti jo pirminius daugiklius p ir q. Problema kyla iš to, jog funkcija

$$(p, q) \Rightarrow p \cdot q$$

yra vadinama vienos krypties funkcija, t. y. turint p ir q, lengva suskaičiuoti rezultatą, tačiau turint rezultatą, sunku suskaičiuoti p ir q (laiko prasme). Žinoma, su nedideliais skaičiais, tai nėra didelė problema, tačiau su pakankamai dideliais skaičiais tai tampa tiesiog neįmanoma. Todėl RSA sistemos naudoja nuo 1024 iki 4096 bitų ilgio raktus N, kuriuos nulaužti prireiktų tūkstančių metų.

Efektyviausias žinomas klasikinis algoritmas šiai problemai spręsti yra bendrasis skaičių lauko rėtis (angl. *general number field sieve*) [14]. P. W. Shor 1994 metais išleido publikaciją, kurioje aprašomas eksponentiškai greitesnis kvantinis algoritmas šiai problemai spręsti [5].

Šį algoritmą galima suskirstyti į tokius žingsnius:

1. Pasirenkame $a < N$;
2. Patikriname, ar a yra reliatyviai pirminis N, t. y. $GCD(a, N) = 1$;
3. Jei lygybė nėra teisinga, mes ką tik atsitiktinai atspėjome vieną iš N daugiklių, tad algoritmas yra baigtas;
4. Randame periodą funkcijos $(x) \Rightarrow a^x \ mod \ N$;
5. Patikriname, ar periodas r yra lyginis ir ar $\left(a^{\frac{r}{2}} + 1\right) mod \ N \ \neq 0$;
6. Jei bent viena iš sąlygų nėra teisinga, reikia pasirinkti naują a ir kartoti algoritmą;
7. Kitu atveju, $p = GCD(a^{\frac{r}{2}} - 1, N)$ ir $q = GCD(a^{\frac{r}{2}} + 1, N)$.

Paskutiniame žingsnyje naudojamos išraiškos gaunamos iš anksčiau priimtų sąlygų ir gautų rezultatų. Tarkime, jog r yra funkcijos $(a, N, x) \Rightarrow a^x \ mod \ N$ pagal x periodas. Tuomet

$$a^0 \ mod \ N = 1,$$
$$a^{0+r} \ mod \ N = 1,$$
$$a^r \ mod \ N = 1,$$
$$a^r - 1 \ mod \ N = 0.$$

Tai reiškia, jog egzistuoja kažkoks k, jog

$$a^r - 1 = k \cdot N.$$

Be to mes teigėme, jog r yra lyginis. Taip pat, $N = p \cdot q$, tad

$$\left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) = k \cdot p \cdot q.$$

Dabar galima teigti, jog p dalina viena iš dauginamųjų kairėje lygybės pusėje, o q dalina kitą dauginamąjį kairėje lygybės pusėje. Nei p nei q nedalina to paties dauginamojo, kadangi nei vienas kairėje esantis dauginamasis nėra dalinamas N. Taip yra todėl, nes mes teigėme, jog

$$\left(a^{\frac{r}{2}} + 1\right) mod \ N \ \neq 0,$$

bei todėl, nes

$$a^r \ mod \ N = 1.$$

Kadangi periodas r yra mažiausia reikšmė didesnė už 0, sukuria ši lygybė yra teisinga, todėl

$$a^{\frac{r}{2}} \ mod \ N \ \neq 1,$$
$$\left(a^{\frac{r}{2}} - 1\right) mod \ N \ \neq 0.$$

Dabar, turėdami po du skaičius, kuriuos dalina p ir q, galima rasti juos rasdami didžiausią bendrą daliklį tarp tų skaičių.

Ketvirtas žingsnis algoritme yra tas žingsnis, kuris yra atliekamas kvantiniame kompiuteryje ir kurį P. W. Shor aprašė. Šis žingsnis remiasi kvantine Fourier transformacija [15].

1.3.3. Diskretaus logaritmo problema ir skaičiavimo algoritmas

Diskretaus logaritmo problema formuluojama taip – turint skaičius *r*, g ir *N*, kur *r* yra reliatyviai pirminis skaičiui *N* ir g yra *N* primityvi šaknis, rasti skaičių *k* tokį, jog

$$r = g^k \bmod N.$$

Skaičius *k* šiuo atveju yra vadinamas šios lygybės diskrečiu logaritmu [16]. Kaip ir su sveiko skaičiaus skaidymu pirminiais daugikliais, šiuo atveju problema yra ta pati. Funkcija

$$(g, k, N) \Rightarrow g^k \bmod N$$

yra vienos krypties funkcija, todėl ji yra naudojama įvairiuose kriptografinėse sistemose.

Efektyviausias žinomas klasikinis algoritmas bendram atvejui yra toks pat, kaip ir sveiko skaičiaus skaidymo pirminiais daugikliais – bendrasis skaičių lauko rėtis [14], tačiau egzistuoja algoritmų, kurie prie specialių sąlygų veikia greičiau. Vienas tokių yra jau minėtas Pohlig–Hellman algoritmas [9]. Kadangi diskretaus logaritmo problemą galima pakeisti funkcijos periodo radimo problema, P. W. Shor aprašytas periodo radimas kvantiniame kompiuteryje gali būti pritaikomas ir čia.

Visą algoritmą galima suskirstyti į tokius žingsnius:

1. Randame periodą funkcijos $(a, b) \Rightarrow g^a \cdot r^b \bmod N$;
2. Tuomet $k = -\frac{a}{b} \bmod N$.

Antrame žingsnyje pateiktos lygybės įrodymas yra nesudėtingas - radę periodą *(a, b)*, atliekame tokius pakeitimus:

$$g^0 \cdot r^0 \bmod N = 1,$$
$$g^{0+a} \cdot r^{0+b} \bmod N = 1,$$
$$g^a \cdot r^b \bmod N = 1.$$

Iš problemos apibrėžimo

$$r = g^k \bmod N,$$

tad

$$g^a \cdot r^b \bmod N = g^a \cdot g^{k \cdot b} \bmod N = g^{a+k \cdot b} \bmod N,$$
$$g^{a+k \cdot b} \bmod N = 1,$$
$$g^{a+k \cdot b} \bmod N = g^0 \bmod N,$$
$$a + k \cdot b = 0 \bmod N,$$
$$k = -\frac{a}{b} \bmod N.$$

Kaip minėjome anksčiau, sistemoje bus naudojama modifikuota šio algoritmo versija. Pagrindinis pakeitimas yra tas, jog vietoj periodo ieškojimo dviejų kintamųjų funkcijai, periodas yra ieškomas funkcijoms

$$(a) \Rightarrow g^a \bmod N,$$
$$(b) \Rightarrow r^b \bmod N.$$

Tuomet *(a, b)* yra periodo kartotinis, kurį bandoma sumažinti bandant *a* ir *b* padalinti iš *GCD(a, b)*, arba bent per pusę, jeigu skaičiai yra lyginiai. Tuomet diskretų logaritmą galima rasti pagal lygybę

$$a + b \cdot k = a \cdot m;$$

čia m – periodo kartotinio (a, b) daliklis, iš kurio padalinus gaunamas periodas.

Tuomet algoritmas spėjimo būdu bando rasti m ir apskaičiuoti diskretų logaritmą k. Nepavykus atspėti m iš $N / 2$ kartų, algoritmas sustoja, kadangi spėti tampa nebeefektyvu.

2. PROJEKTAS

2.1. Reikalavimų specifikacija

2.1.1. Komercinė specifikacija

Tai yra universitetinis projektas, specialiai kuriamas bakalauro baigiamajam darbui. Projekto užsakovas – bakalaurinio darbo vadovas doc. dr. Tomas Blažauskas. Projekto vykdytojas - universiteto studentas Gintaras Volkvičius. Projekto naudotojai: kitų projektų kūrėjai, norintys integruoti ar naudotis problemų sprendimo paslaugų serviso API; darbo su kompiuteriu įgūdžių turintys žmonės, norintys pasidomėti problemų sprendimais pasitelkiant kvantinius algoritmus.

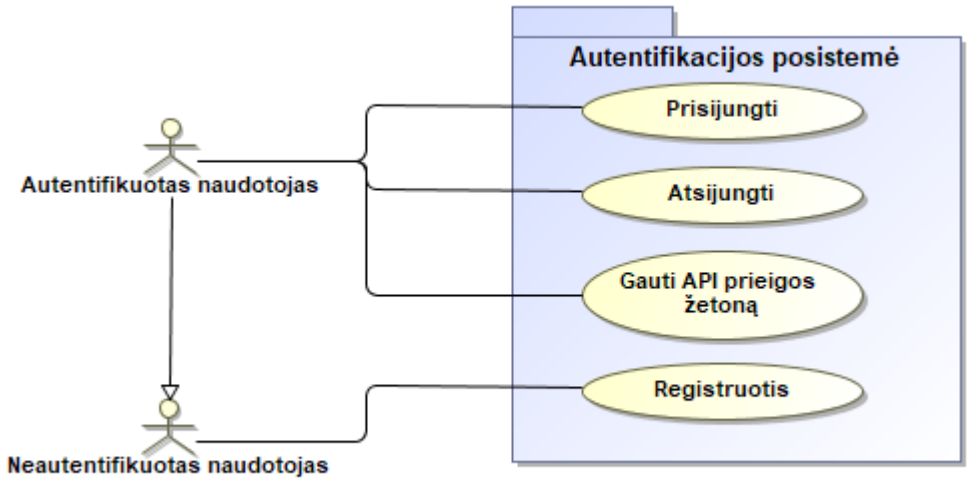
Projekto biudžeto nėra, kadangi tai yra asmeninis projektas. Numatyta projekto baigimo data – 2018 m. gegužės 21 d. Iki šios dienos turi būti atlikti visi programavimo, testavimo bei dokumentavimo darbai.

2.1.2. Sistemos funkcijos

Sistemos funkciniai reikalavimai išskirstyti į tris posistemes:

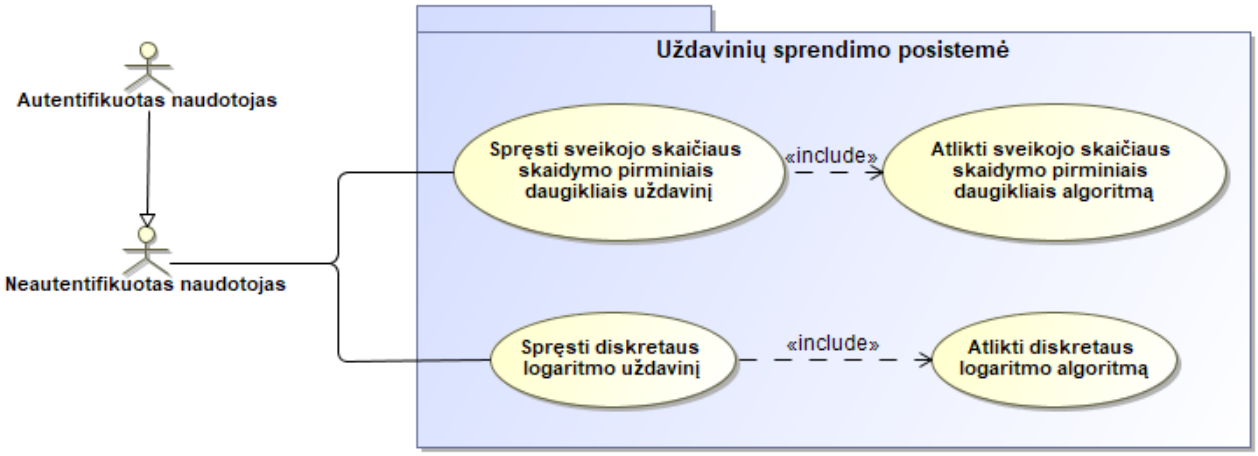
1. Autentifikacijos;
2. Uždavinių sprendimo;
3. Uždavinių sprendimų valdymo.

Funkciniai reikalavimai pavaizduoti pasitelkus UML panaudojimo atvejų diagramas (2.1 – 2.3 pav.).



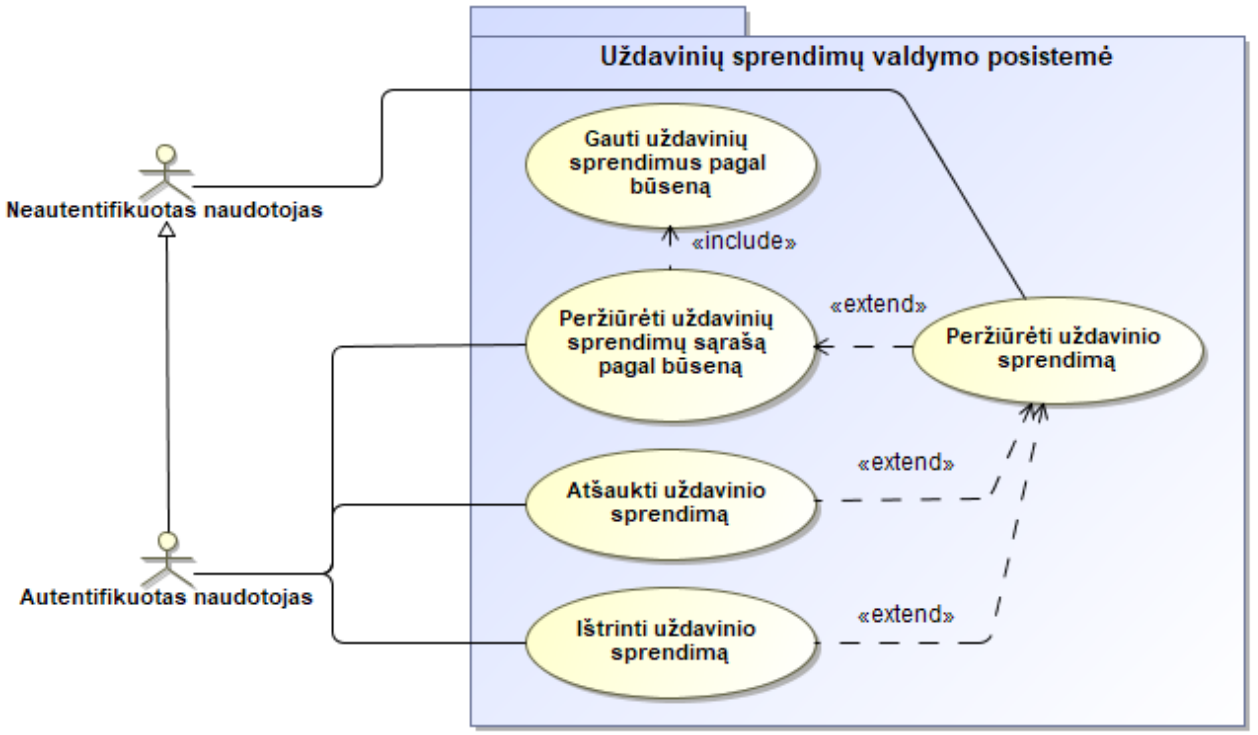
2.1 pav. Autentifikacijos posistemės panaudojimo atvejai

Autentifikacijos posistemė skirta su naudotojo autentifikacija skirties veiksmams – prisijungti ir atsijungti prie saityno taikomosios programos per asmeninį sertifikavimo paslaugų teikėjo servisą, registruotis pastarajame paslaugų servise bei gauti prieigos žetoną, skirtą problemų sprendimų API naudojimui už saityno taikomosios programos ribų. Neautentifikuotas naudotojas (negalintis patvirtinti savo tapatybės per asmeninį sertifikavimo paslaugų teikėjo servisą) gali tik registruotis, o autentifikuotas naudotojas gali atlikti visas posistemės funkcijas, įskaitant ir naujos tapatybės kūrimą (pakartotinę registraciją).



2.2 pav. Uždavinių sprendimo posistemės panaudojimo atvejai

Uždavinių sprendimo posistemė skirta uždavinių sprendimui. Posistemę sudaro pagrindinės dvi funkcijos – sveikojo skaičiaus skaidymo pirminiais daugikliais uždavinio sprendimas bei diskretaus logaritmo uždavinio sprendimas. Uždavinių sprendimas vykdomas asinchroniškai, todėl abi šios funkcijos atlieka po papildomą funkciją, kuri iškviečiama sukuriant naują foninį darbą (procesą), kai tuo tarpu naudotojui yra grąžinamas tarpinis uždavinio sprendimo rezultatas (detaliau apie tai 2.3.2 skyrelyje). Tiek autentifikuotas, tiek neautentifikuotas naudotojas gali atlikti šias funkcijas, t. y. spręsti uždavinius.



2.3 pav. Uždavinių sprendimų valdymo posistemė

Uždavinių sprendimų valdymo posistemė, kaip nusako pats pavadinimas, skirta valdyti išspręstus ar vis dar sprendžiamus uždavinius. Tiek išspręstiems, tiek vis dar sprendžiamiems uždaviniams galimos šios funkcijos – uždavinio sprendimo peržiūrėjimas bei uždavinių sprendimų sąrašo peržiūrėjimas pagal būseną (į kurį įeina sisteminė funkcija, skirta gauti uždavinių sprendimus

pagal būseną). Papildomai išspręstiems (tiek sėkmingai, tiek ne) uždaviniams galima uždavinio sprendimo panaikinimo funkcija, o vis dar sprendžiamiems uždaviniams – sprendimo atšaukimo funkcija. Neautentifikuotas naudotojas, galėdamas spręsti uždavinius, gali peržiūrėti ir konkretaus uždavinio sprendimą. Likusios funkcijos yra išskirtinės autentifikuoto naudotojo funkcijos.

2.1.3. Apribojimai

Kadangi visos sistemos dalys yra kuriamos pasitelkus *ASP.NET Core 2* karkasu, sistema gali būti diegiama tiek *Windows*, tiek *Linux*, tiek *macOS* aplinkose. Taip pat, tiek *IdentityServer4* karkasas, naudojamas asmeninio sertifikavimo paslaugų teikėjo kūrimui, tiek *Hangfire* karkasas, skirtas foninių darbų valdymui, yra pilnai palaikomi su *ASP.NET Core 2* karkasu. Duomenų bazei valdyti naudojama *Microsoft SQL Server 2017* DBVS. *Microsoft SQL Server 2017* nėra tiesiogiai palaikoma *macOS*, tačiau yra programinės įrangos produktų, leidžiančių tai padaryti. Vienas tokių – *Docker* [17].

Sistema kuriama naudojant *Microsoft Visual Studio 2017* integruotą kūrimo aplinką kartu su *ReSharper Ultimate* įskiepiu bei IIS HTTPS serverį lokaliam sistemos diegimui.

Kaip jau buvo minėta, sistemos kūrimo baigimo data - 2018 m. gegužės 21 d., projekto biudžeto nėra. Tarpiniai sistemos kūrimo terminai nebuvo sudaryti.

2.1.4. Vartotojo sąsajos specifikacija

Jokie reikalavimai grafinei naudotojo sąsajai nebuvo iškelti.

2.1.5. Realizacijai keliami reikalavimai.

Sistemai buvo iškelti šie nefunkciniai reikalavimai:

1. Reikalavimai realizacijai:
    1.1. Uždavinių sprendimas turi būti vykdomas asinchroniškai;
    1.2. Problemų sprendimo servisas neturi saugoti jokios būsenos (angl. *stateless service*);
    1.3. Problemų sprendimo servisas visą informaciją apie naudotoją turi gauti kartu su užklausa per nuorodos prieigos žetoną (angl. *reference token*), kuris būna išduodamas ir tikrinamas asmeninio sertifikavimo paslaugų teikėjo servise;
2. Reikalavimai saugumui:
    2.1. Autentifikuoto naudotojo funkcijos turi būti apsaugotos integruotu *ASP.NET Core 2* autentifikacijos mechanizmu, kaip tapatybės teikėją naudojant sukurtą asmeninio sertifikavimo paslaugų teikėjo servisą;
    2.2.

2.1.6. Techninė specifikacija.

Kuriama sistema bus talpinama *Microsoft Azure Cloud* platformoje. Kiekviena iš sistemos dalių turi turėti saityno serverius, o problemų sprendimo bei asmeninio sertifikavimo paslaugų teikėjo servisai – ir po vieną duomenų bazės serverį. Detalesnė techninė specifikacija kiekvienai sistemos daliai:

1. Problemų sprendimo servisas:
    1.1. Viena *Azure* virtuali mašina (*H-series*, dydis: *Standard_H16*) [18];
    1.2. Vienas *Azure* duomenų bazės serveris (*Standard service tier, S0*) [19].
2. Asmeninio sertifikavimo paslaugų teikėjo servisas:
    2.1. Viena *Azure* virtuali mašina (*B-series*, dydis: *Standard_B1s*) [20];
    2.2. Vienas *Azure* duomenų bazės serveris (*Standard service tier, S0*) [19].
3. Saityno taikomoji programa:
    3.1. Viena *Azure* virtuali mašina (*B-series*, dydis: *Standard_B1s*) [20].

2.2. Projektavimo metodai

2.2.1. Projektavimo valdymas ir eiga

Kuriant sistemą buvo taikomas iteracinis projektavimo modelis. Iš viso buvo atliktos 4 dviejų savaičių iteracijos. Pirmosios trys iteracijos buvo skirtos visų trijų sistemos dalių projektavimui, realizavimui ir testavimui. Pirmoji iteracija buvo skirta problemų sprendimo serviso projektavimui, realizavimui bei testavimui. Antroji iteracija buvo skirta asmeninio sertifikavimo paslaugų teikėjo serviso projektavimui, realizavimui bei testavimui. Trečioji iteracija buvo skirta saityno taikomosios programos projektavimui, realizavimui bei testavimui. Paskutinė iteracija buvo skirta visų trijų sistemų apjungimui bei pastebėtų projektavimo ir realizavimo klaidų taisymui.

2.2.2. Projektavimo technologija

Sistemos projektas sukurtas naudojant UML standarto grafinius elementus. Buvo suprojektuotos panaudojimo atvejų, diegimo, komponentų, paketų, klasių bei sekų diagramos. Diagramų projektavimui buvo naudojamas *MagicDraw* projektavimo įrankis.

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

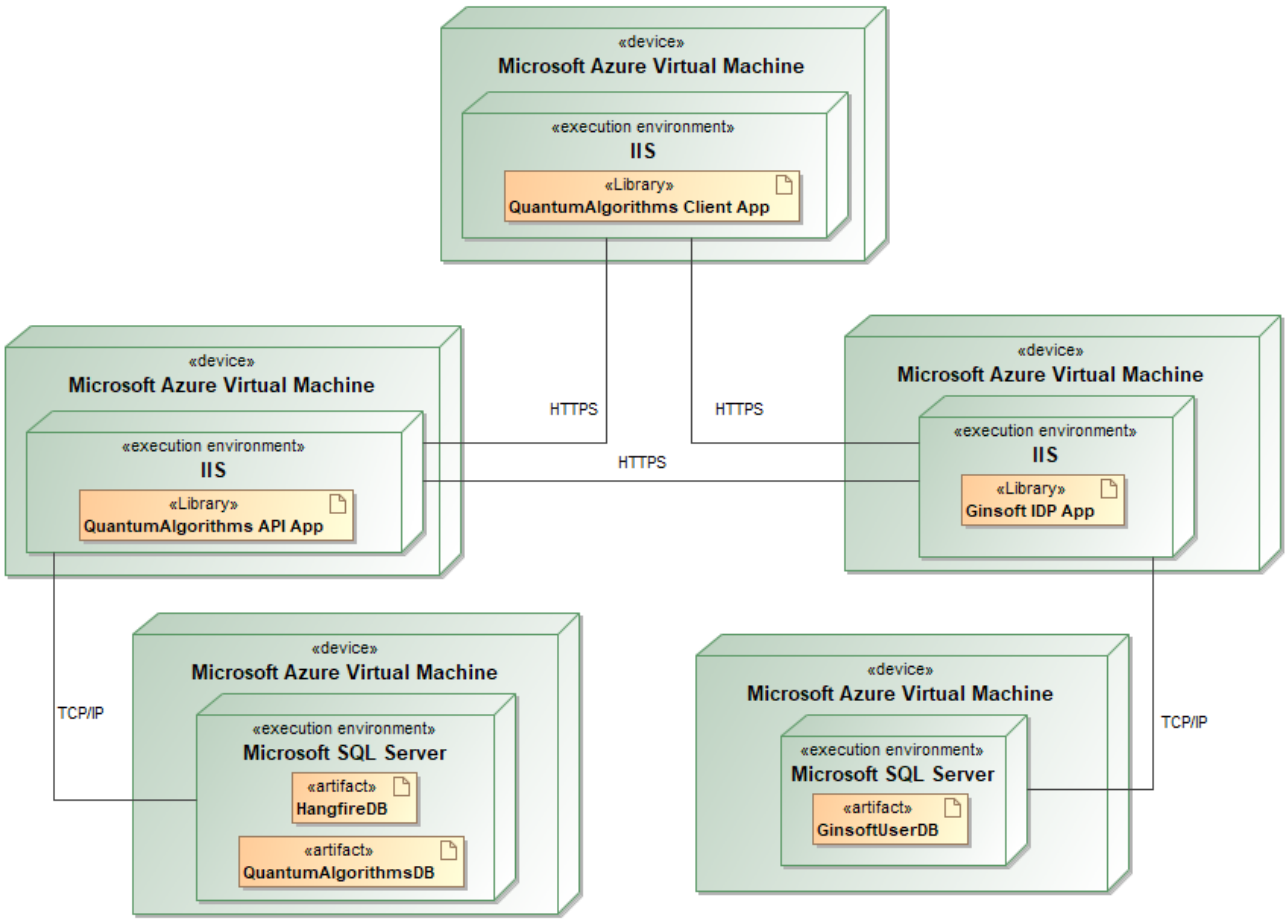
Sistema buvo sukurta naudojant *C#* bei *Q#* programavimo kalbas. Sistema buvo projektuojama ir programuojama *Windows 10* operacinėje sistemoje. Programavimui buvo naudota *Microsoft Visual Studio 2017 Enterprise* integruota kūrimo aplinka. Duomenų bazėms valdyti buvo naudojama *Microsoft SQL Server 2017* DBVS.

Komponentų testai (angl. *unit tests*) sukurti naudojant *NUnit 3.x* karkasą, testai buvo vykdomi su *Microsoft Visual Studio 2017* integruotos kūrimo aplinkos *ReSharper Ultimate* įskiepio integruotu komponentų testų vykdytoju, kodo padengimas buvo skaičiuojamas su tuo pačiu įskiepiu.

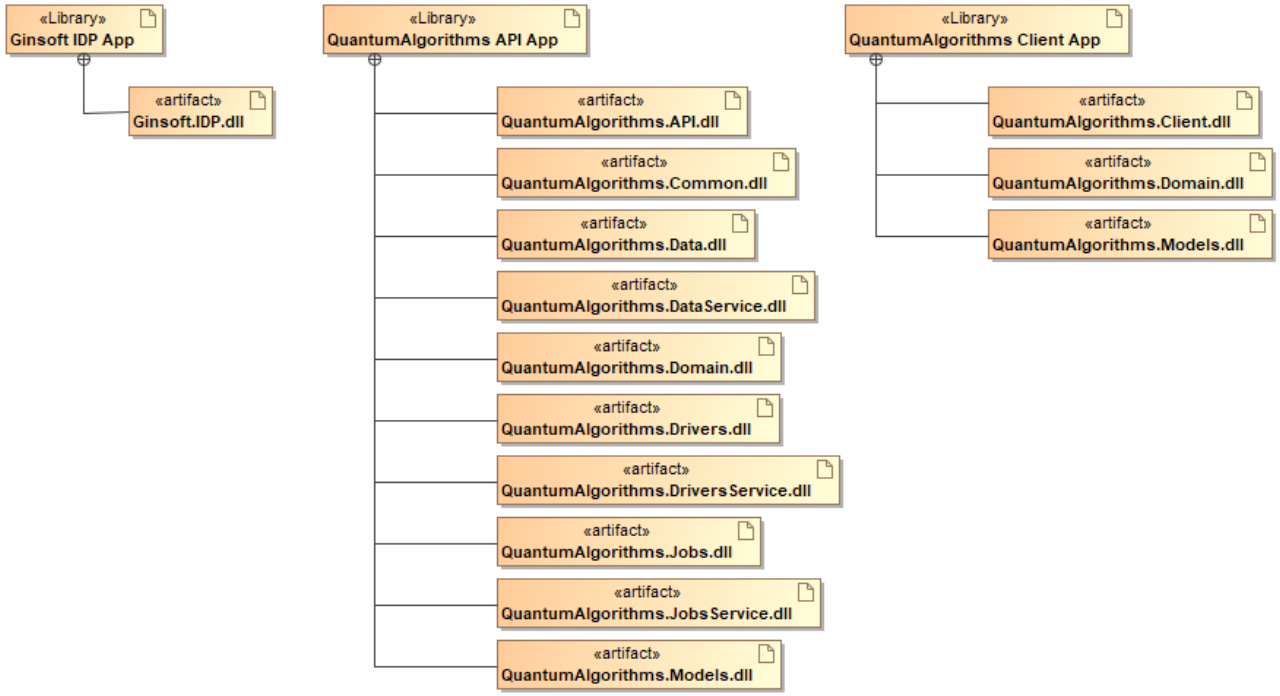
2.3. Sistemos projektas

2.3.1. Statinis sistemos vaizdas

Statinio sistemos vaizdo apžvalgą geriausia pradėti nuo jos diegimo diagramos, kadangi ji greičiausiai supažindina su bendra sistemos sudėtimi ir jos pasiskirstymu tarp įrenginių. Sistemos diegimo bei sistemos dalių naudojamų artefaktų diagramos pavaizduotos 2.4 pav. ir 2.5 pav..



2.4 pav. Sistemos diegimo diagrama

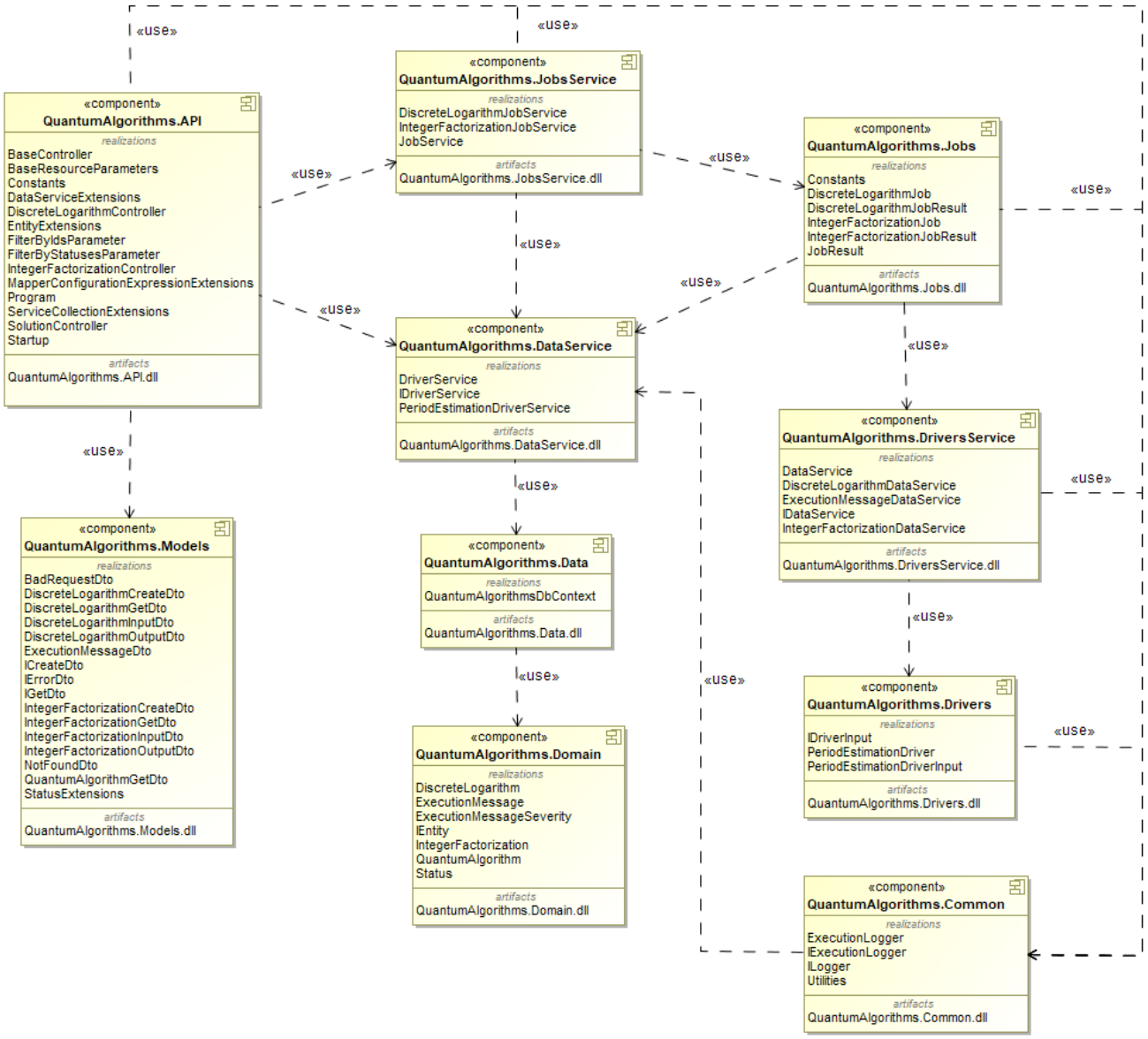


2.5 pav. Sistemos dalių naudojamų artefaktų diagrama

Kaip galima pastebėti diegimo diagramoje, trys sistemos dalys yra paskirstytos tarp 5 skirtingų įrenginių. Po du įrenginius – servisui bei duomenų bazės serveriui – skirta problemų sprendimo servisui ir asmeninio sertifikavimo paslaugų teikėjui bei vienas įrenginys skirtas saityno

taikomajai programai. Visos trys sistemos dalys yra paskirstytos skirtinguose įrenginiuose, nes problemų sprendimo servisui reikalingi dideli techniniai pajėgumai atlikti skaičiavimus, o asmeninio sertifikavimo paslaugų teikėjas bendrame kontekste gali aptarnauti ir kitus servisus bei taikomąsias programas, neįeinančias į šios sistemos sudėtį.

Toliau bus vaizduojamos tik pagrindinės – problemų sprendimų serviso – dalies statinio vaizdo diagramos. Esminės loginės problemų sprendimų serviso dalys ir sąveika tarp jų vaizduojama komponentų diagrama. Komponentų diagrama pavaizduota 2.6 pav..



2.6 pav. Problemų sprendimo serviso komponentų diagrama

QuantumAlgorithms.API komponentas – tai serviso įėjimo taškas. Jame yra kontroleriai, atsakingi už HTTP užklausų priėmimą, bei atsakymo grąžinimą. Komponentas gaunamą informaciją interpretuoja ir atsakymus naudotojui grąžina pagal *QuantumAlgorithms.Models* komponente aprašytus duomenų perdavimo objektus (angl. *data transfer objects, DTO*).

QuantumAlgorithms.DataService komponentas – tai duomenų servisas, kurį naudoja kiti komponentai talpinti ar pasiekti duomenų bazės duomenis. Kiekviena duomenų bazės loginė esybė turi savo duomenų serviso implementaciją, kuri kitiems komponentams yra pateikiama priklausomybės injekcijos principu (angl. *dependency injection*), kai kiti komponentai paprašo tam tikros duomenų bazės loginės esybės duomenų serviso. Šis komponentas duomenims pasiekti naudoja *QuantumAlgorithms.Data* komponentą, kuriame yra laikomas duomenų bazės kontekstas, iš kurio

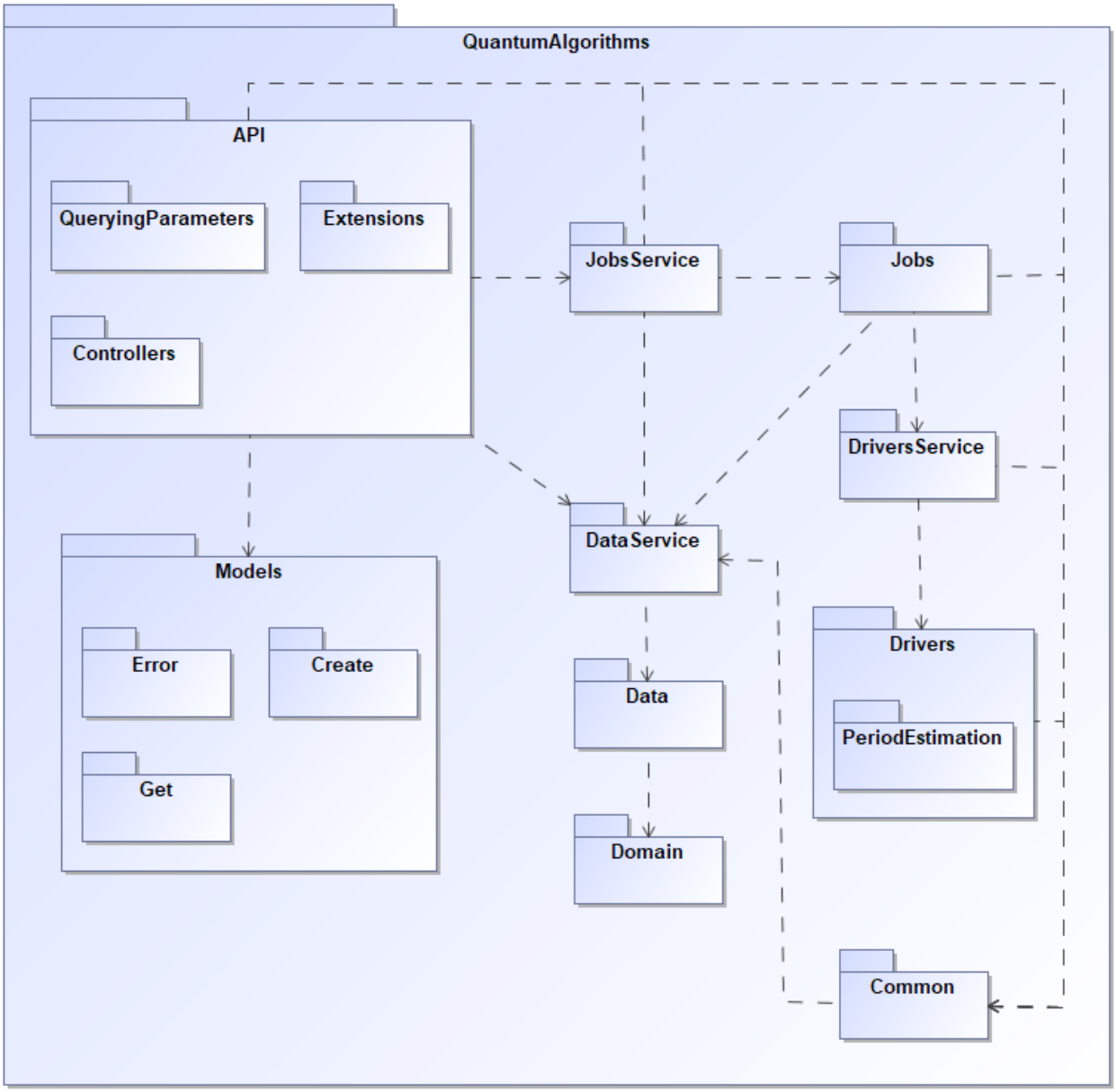
tiesiogiai galima pasiekti visas esybes. Duomenų bazės konteksto loginė schema yra sudaryta iš *QuantumAlgorithms.Domain* komponente aprašytų klasių.

QuantumAlgorithms.JobsService komponentas atsakingas už problemos sprendimo inicializavimą, apdorojimą ir užbaigimą. Po naudotojo problemos sprendimo užklausos, foniniame darbe yra inicializuojamas problemos darbo servisas, kuris registruoja darbo pradžią, pradeda patį problemos sprendimo darbą, kuris yra gaunamas iš *QuantumAlgorithms.Jobs* komponento, bei po darbo baigimo registruoja darbo būseną,

QuantumAlgorithms.DriversService komponentas, panašiai kaip *QuantumAlgorithms.JobsServicve* komponentas, yra inicializuojamas naujame foniniame darbe. Išskirtinai šis servisas yra atsakingas už kvantinių algoritmų skaičiavimų inicializavimą, apdorojimą ir užbaigimą. Darbai, kuriuos šis komponentas inicializuoja, yra *QuantumAlgorithms.Drivers* komponente.

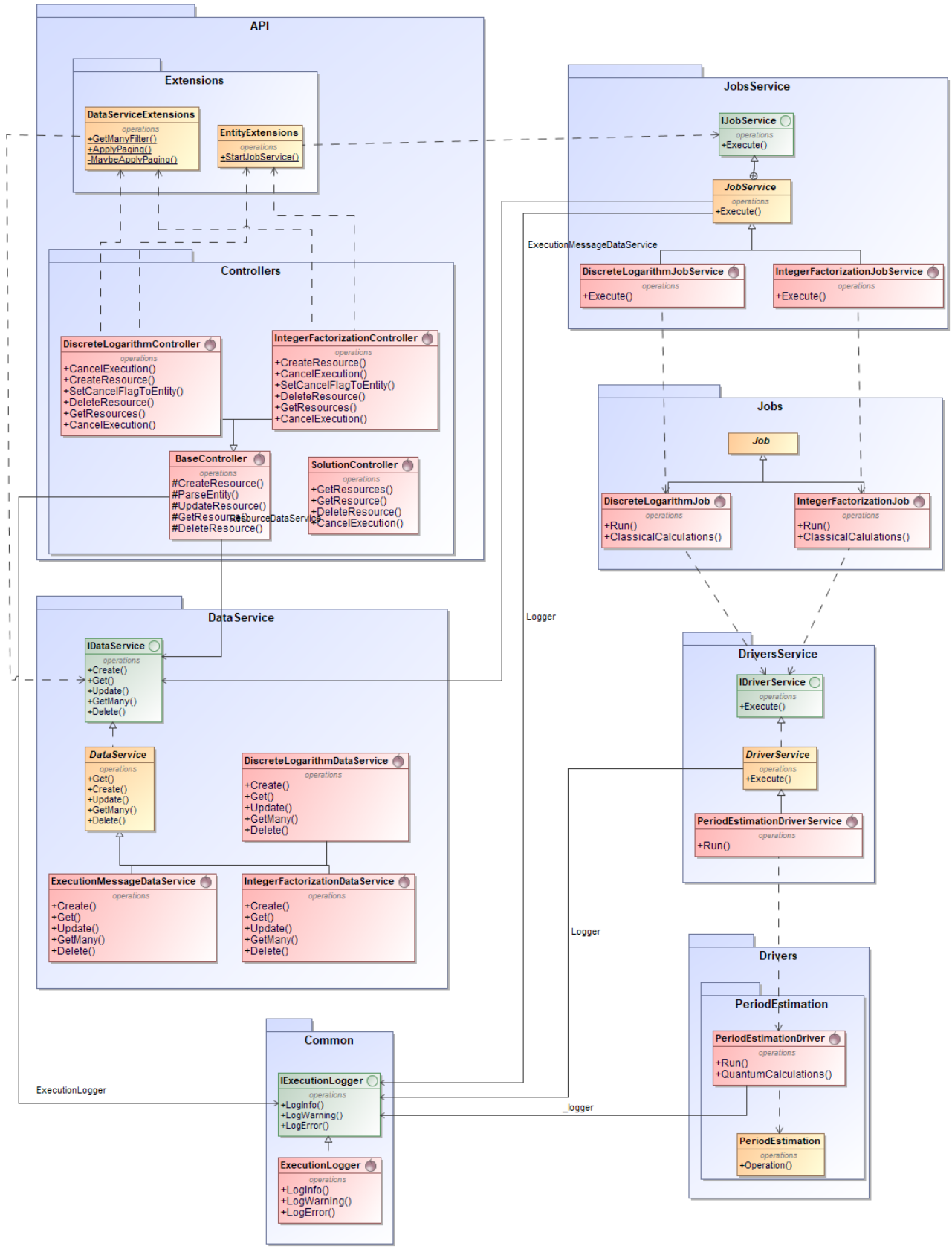
Paskutinis komponentas – *QuantumAlgorithms.Common* – yra atsakingas už bendrų klasių ir funkcijų pateikimą visiems likusiems komponentams. Bene svarbiausia iš jų – problemų sprendimo eigos registravimo funkcija.

Toliau pateikiama paketų diagrama, kuri leidžia atskleisti šiek tiek detalesnį problemų sprendimo serviso vaizdą (2.7 pav.).



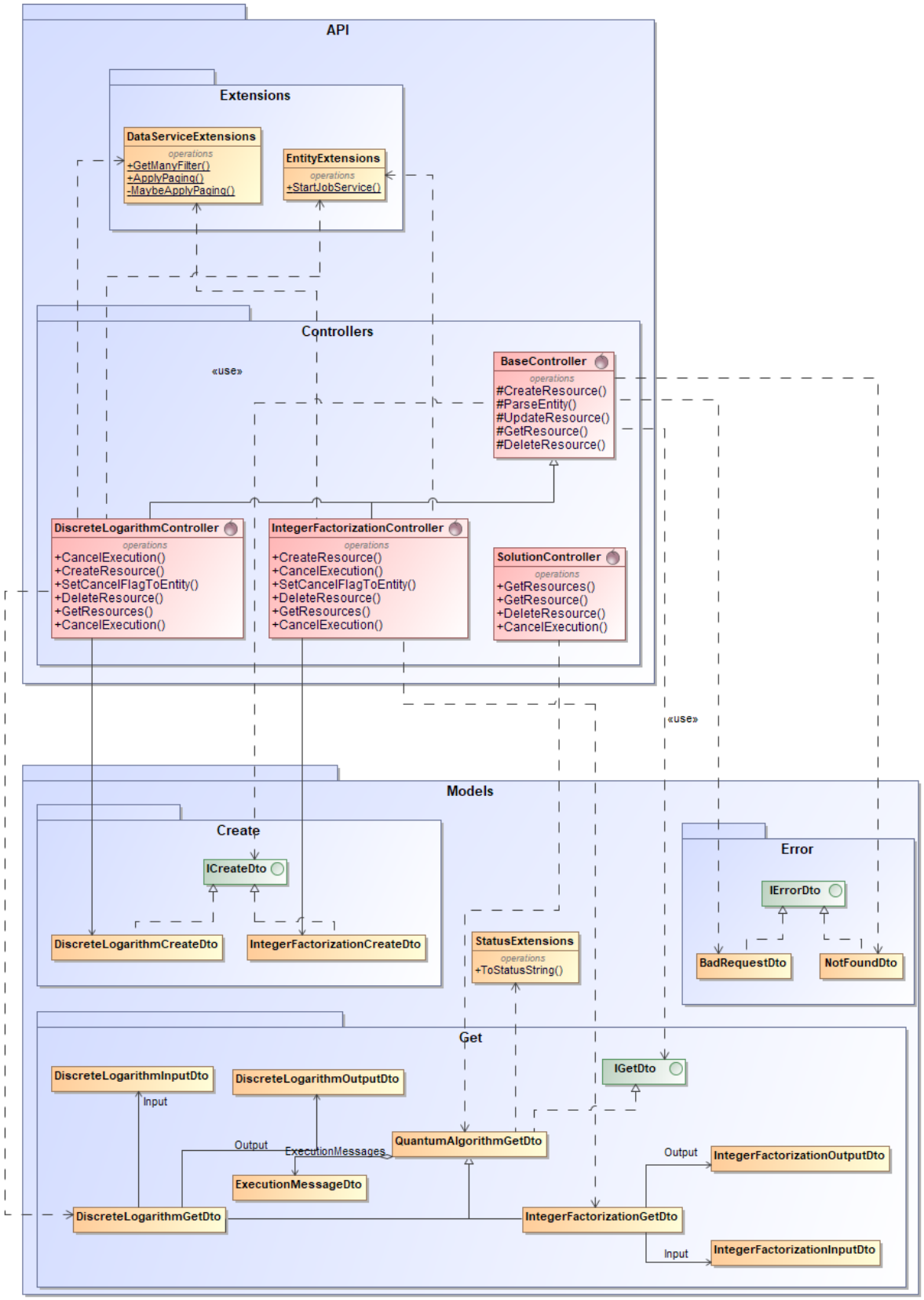
2.7 pav. Problemų sprendimo serviso paketų diagrama

Toliau pateikiamos problemų sprendimo serviso klasių diagramos (2.8 pav., 2.10 pav. ir 2.9 pav.).



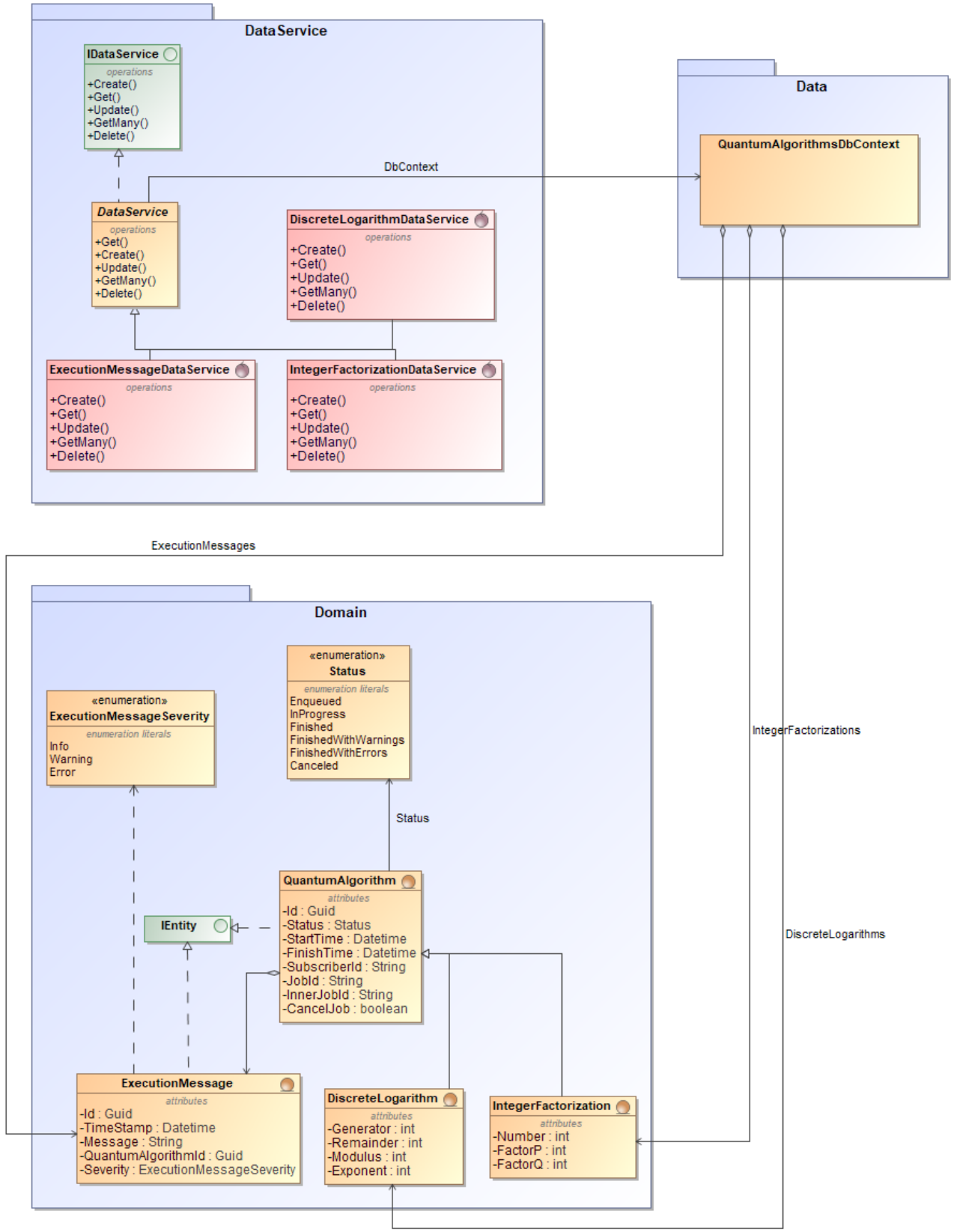
2.8 pav. Problemų sprendimo serviso API, servisų, darbų bei bendro funkcionalumo komponentų pagrindinių klasių diagrama

Šioje klasių diagramoje pateiktos pagrindinės klasės, dalyvaujančios pagrindinių funkcijų – problemų sprendimų – vykdymo eigoje. Iš klasių diagramos galima pastebėti, kaip jau ir buvo minėta anksčiau, jog problemų sprendimo serviso klasės turi priklausomybę ne nuo konkrečių duomenų ar darbų servisų, bet nuo apibrėžtos sąsajos. Priklausomai nuo to, kokio konkretaus serviso yra reikalaujama, jis būna paduodamas klasei per konstruktorių priklausomybės injekcijos principu.



2.9 pav. Problemų sprendimo serviso API ir modelių komponentų klasių diagrama

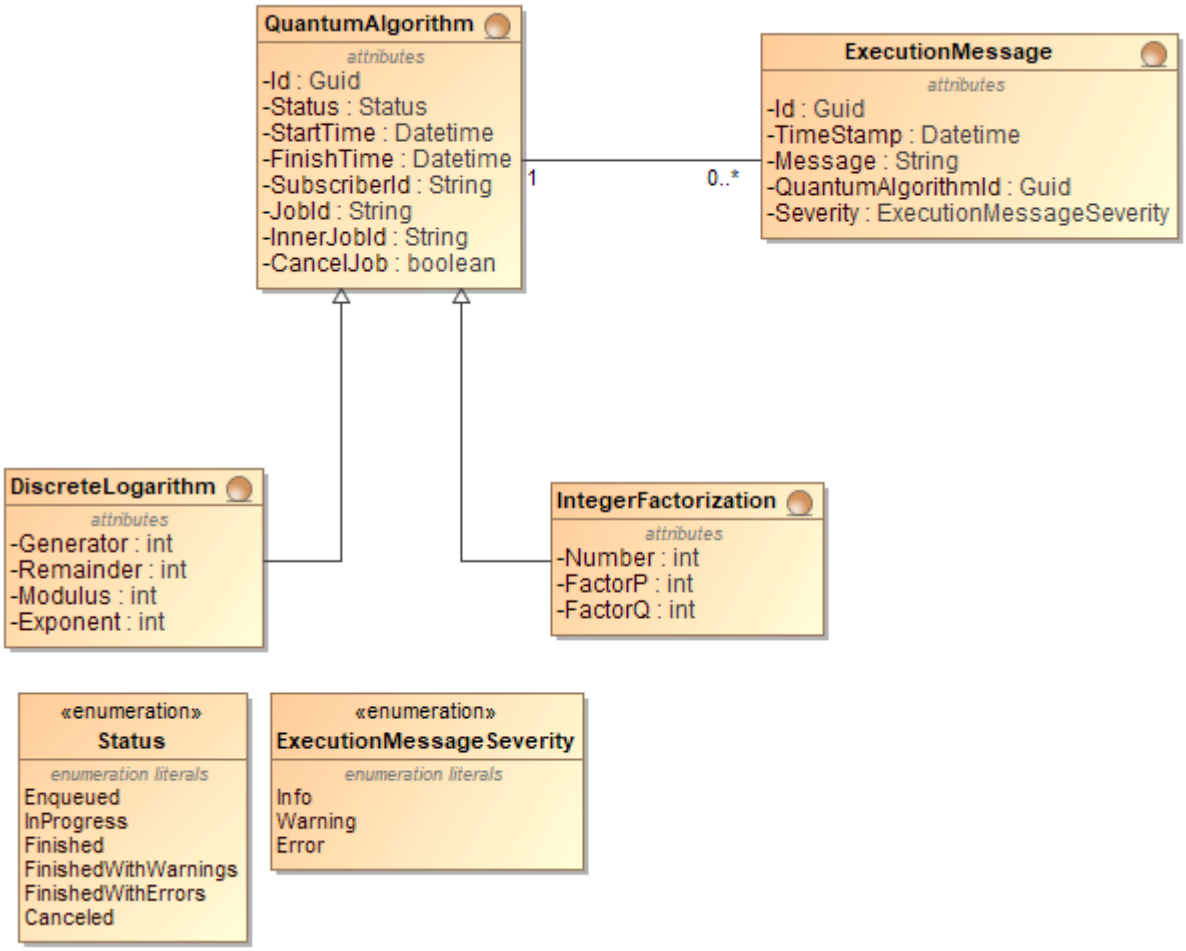
Šioje klasių diagramoje detaliau pavaizduojami ryšiai tarp kontrolerių bei duomenų perdavimo objektų. Abstrakti kontrolerio klasė *BaseController* yra priklausoma nuo duomenų perdavimo objektų sąsajos klasių, kai tuo tarpu konkretūs kontroleriai – nuo konkrečių klasių.



2.10 pav. Problemų sprendimo serviso duomenų serviso ir priklausomų komponentų klasių diagrama

Šioje klasių diagramoje galime detaliau panagrinėti, nuo ko priklauso duomenų servisai. Taip pat galima pamatyti, jog egzistuoja trys pagrindinės duomenų bazės loginės esybės.

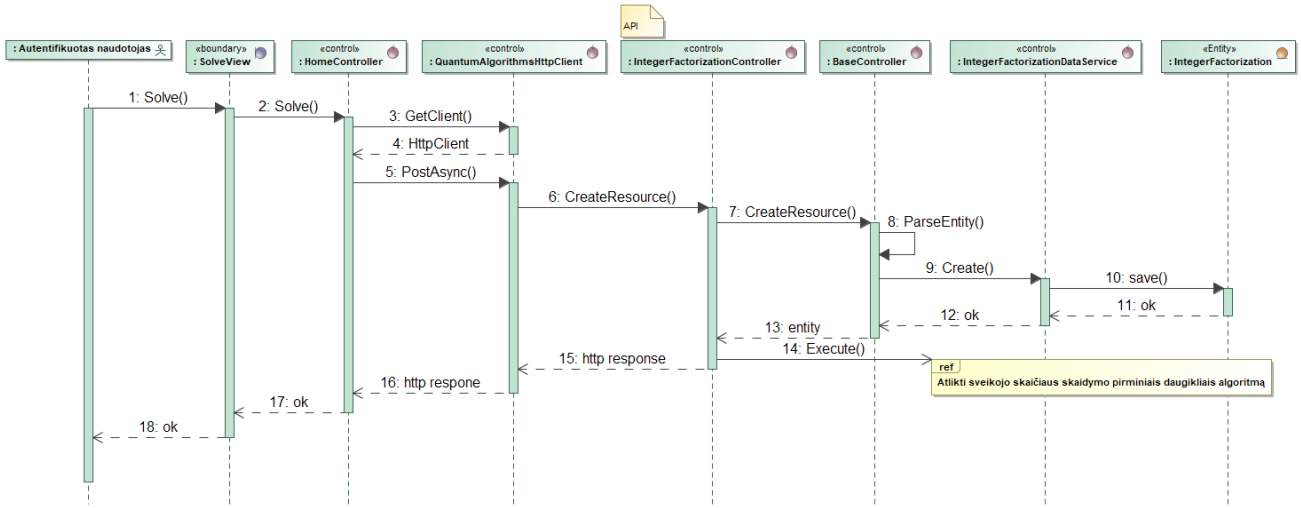
Kalbant apie duomenų bazė, tikslus duomenų bazės esybių-ryšių modelis pavaizduotas 2.11 pav..



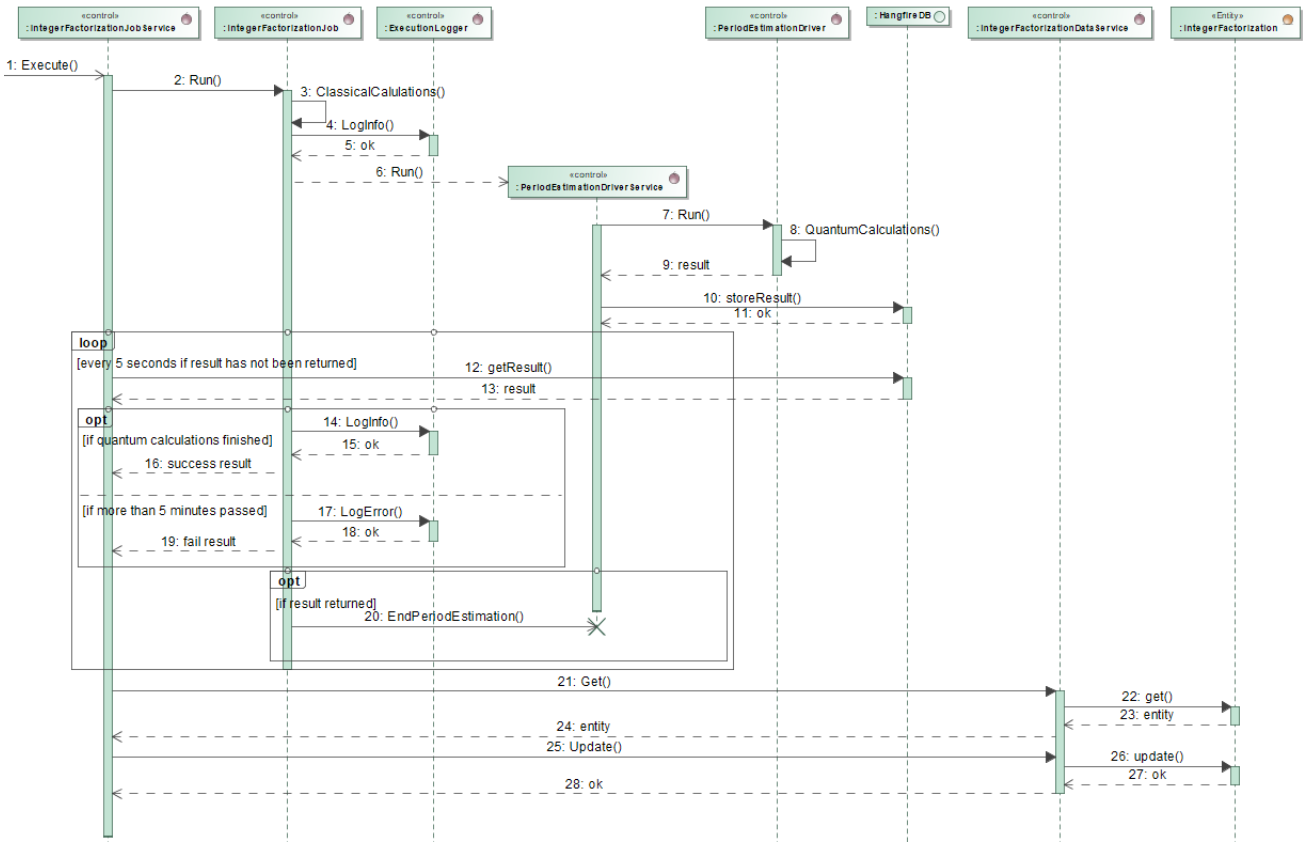
2.11 pav. Problemų sprendimo serviso duomenų bazės esybių-ryšių modelis

2.3.2. Dinaminis sistemos vaizdas

Sistemos dinaminis vaizdą nuspręstą atskleisti pagrindinių posistemių – uždavinių sprendimo ir uždavinių sprendimų valdymo – panaudojimo atvejų sekų diagramomis. Sekų diagramos pavaizduotos 2.12 – 2.20 pav..



2.12 pav. „Spręsti sveikojo skaičiaus skaidymo pirminiais daugikliais uždavinį“ panaudojimo atvejo sekų diagrama



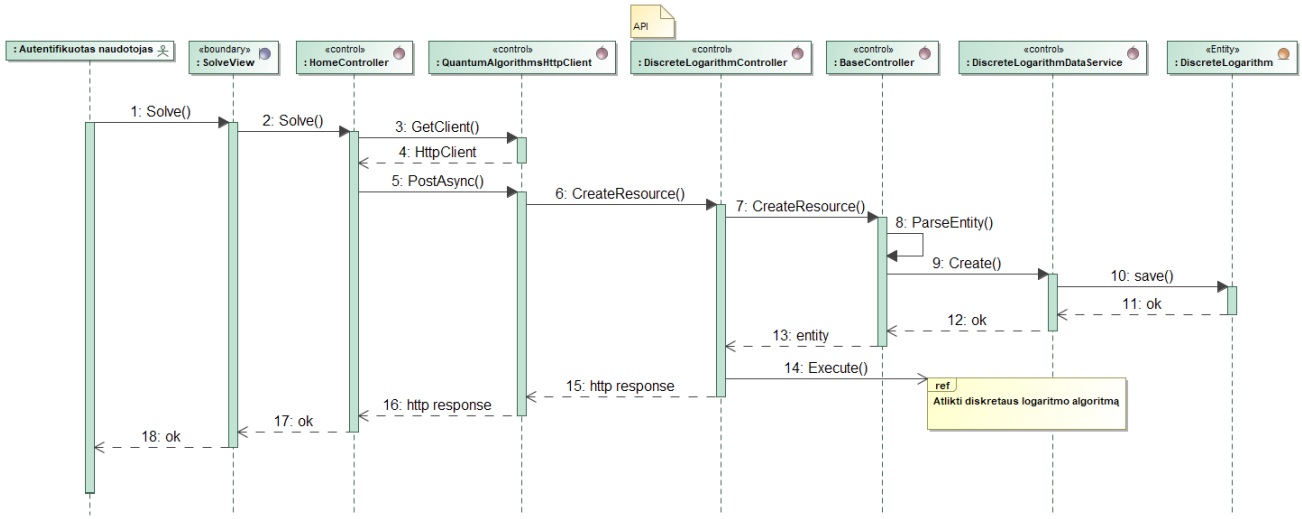
2.13 pav. „Atlikti sveikojo skaičiaus skaidymo pirminiais daugikliais algoritmą“ panaudojimo atvejo sekų diagrama

Pastarosios dvi sekų diagramos pilnai nusako sveikojo skaičiaus skaidymo pirminiais daugikliais funkcijos vykdymo eigą. Gavus užklausą iš naudotojo, duomenų bazėje yra sukuriama uždavinio sprendimo esybė, kurioje bus saugoma informaciją apie uždavinio sprendimo eigą. Tuomet yra inicijuojamas foninis darbas, kuris pradeda vykdyti uždavinio sprendimo eigą. Tuo tarpu naudotojui yra grąžinamas rezultatas su išsaugotos esybės informacija ir identifikatoriumi, pagal kurį galima gauti informaciją apie sprendimą.

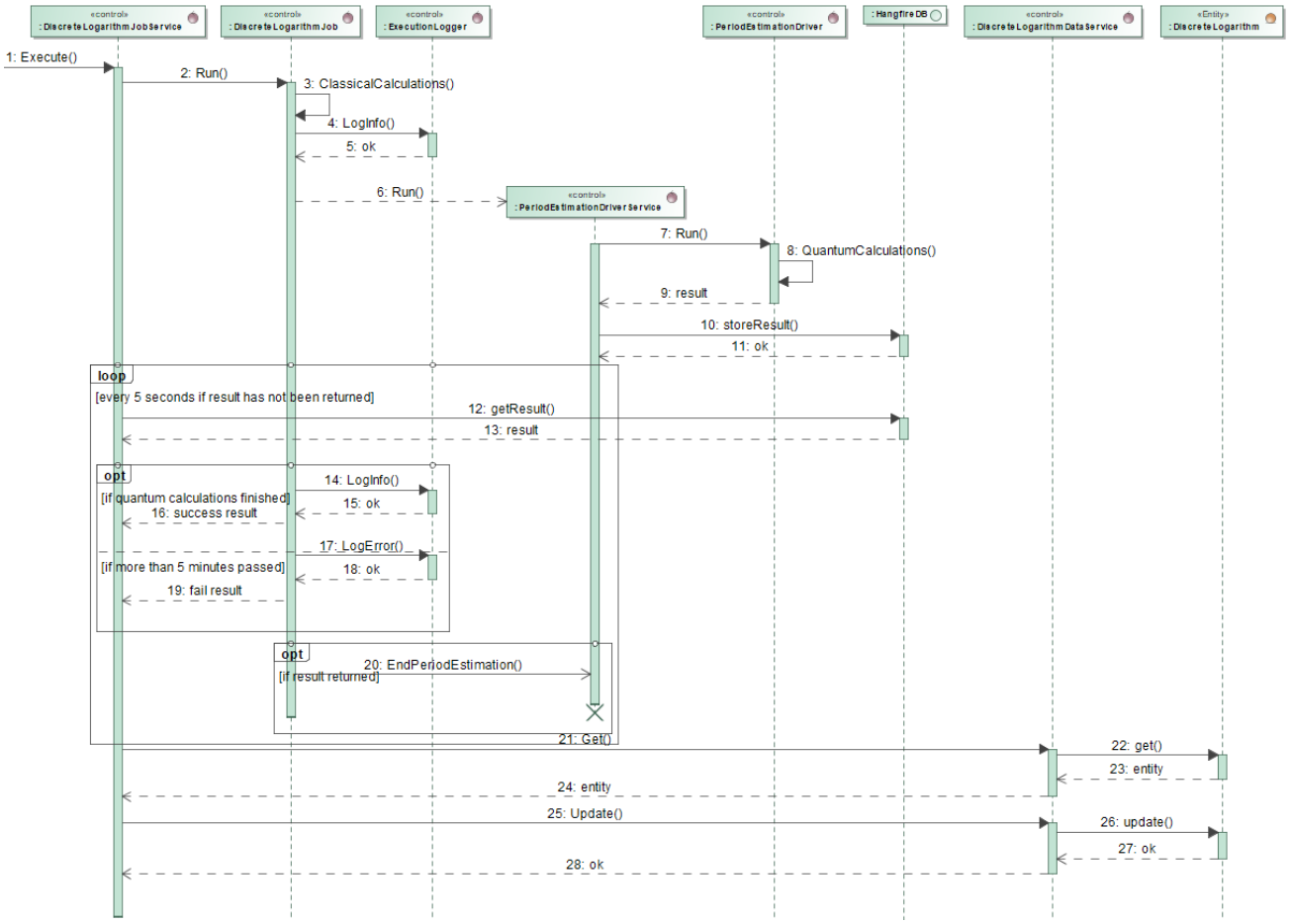
Inicijuotas uždavinio sprendimo algoritmas pirmiausia atlieka baigtinį skaičių veiksmų, kuriuos galima atlikti klasikiniame kompiuteryje. Tuomet yra sukuriamas naujas foninis darbas kvantiniams skaičiavimams atlikti, o tuo tarpu pirminis darbas aktyviai laukia kvantinių skaičiavimų pabaigos. Jei per 5 min. rezultatas yra negaunamas, pirminis darbas nutraukia kvantinius skaičiavimus

ir yra grąžinamas nesėkmės rezultatas. Priešingu atveju, jei rezultatas yra sulaukiamas, grąžinamas sėkmingas rezultatas.

Tuomet pirminis darbas atnaujina informaciją duomenų bazėje ir pats baigia savo darbą.

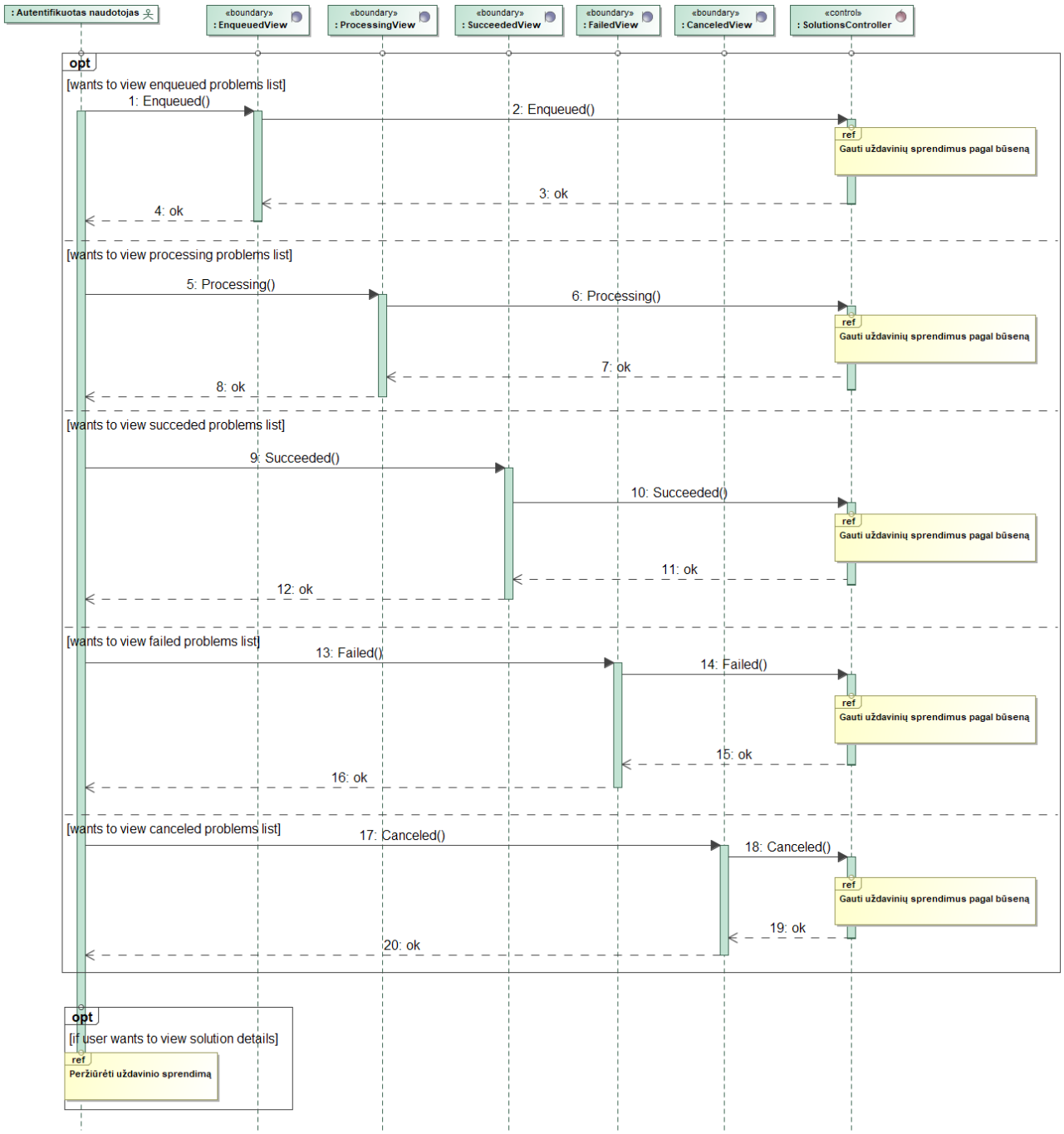


2.14 pav. „Spręsti diskretaus logaritmo uždavinį“ panaudojimo atvejo sekų diagrama

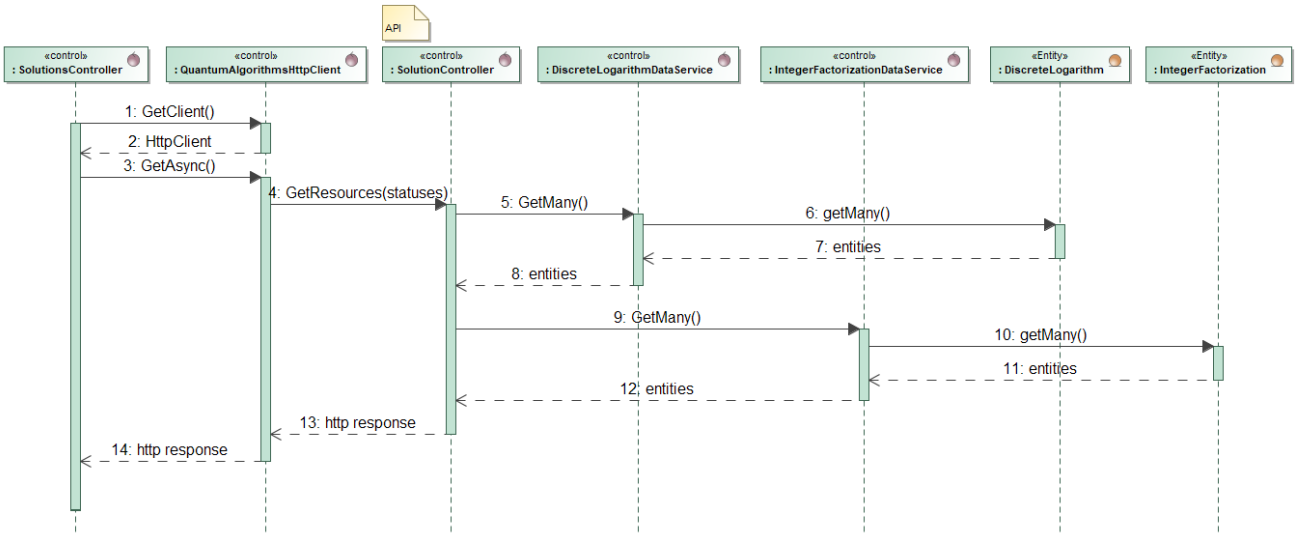


2.15 pav. „Atlikti diskretaus logaritmo algoritmą“ panaudojimo atvejo sekų diagrama

Šios dvi sekų diagramos pilnai nusako diskretaus logaritmo skaičiavimo funkcijos vykdymo eigą. Šios funkcijos vykdymo eiga yra beveik identiška sveikojo skaičiaus skaidymo pirminiais daugikliais funkcijos vykdymo eigai.

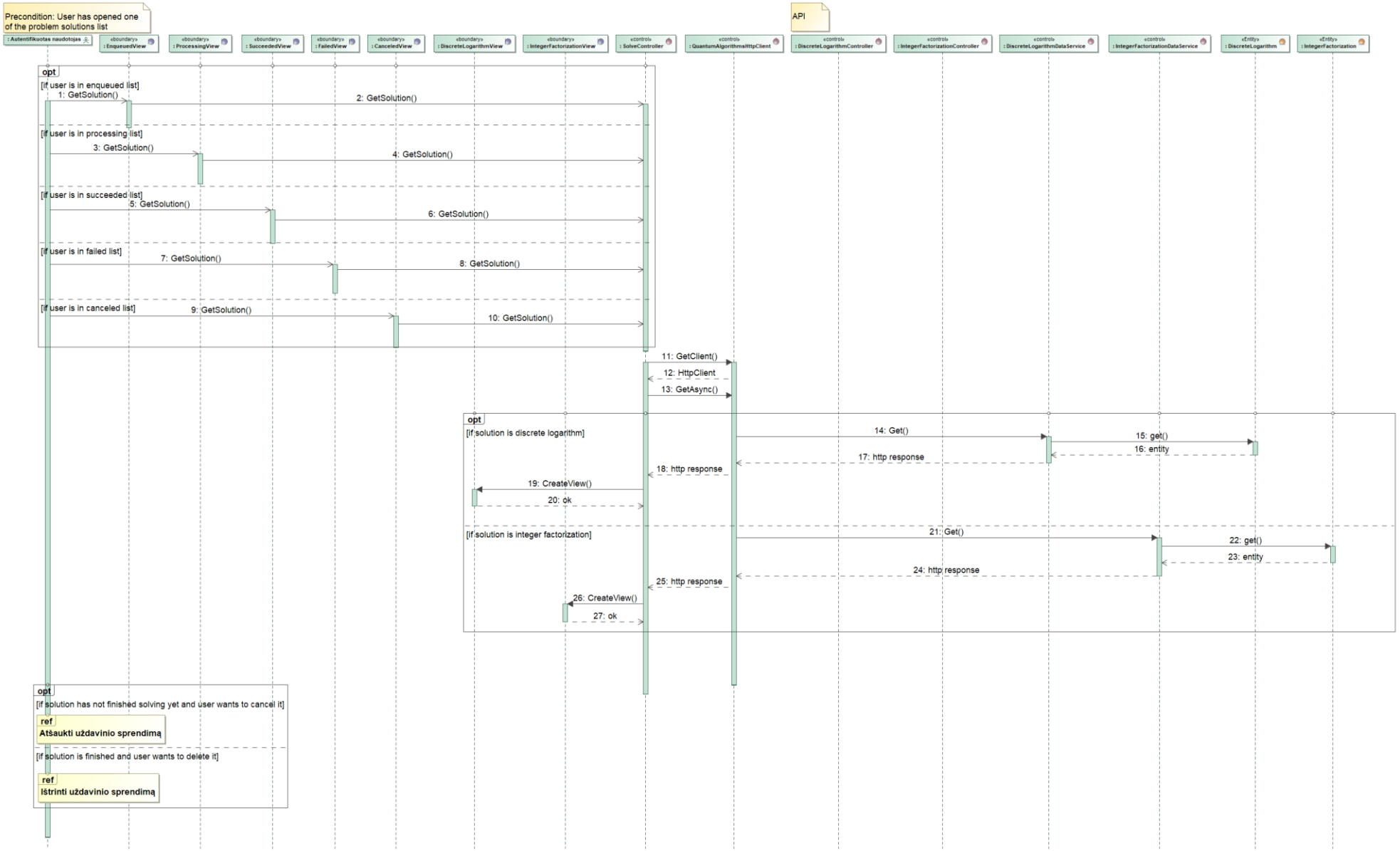


2.16 pav. „Peržiūrėti uždavinių sprendimų sąrašą pagal būseną“ panaudojimo atvejo sekų diagrama

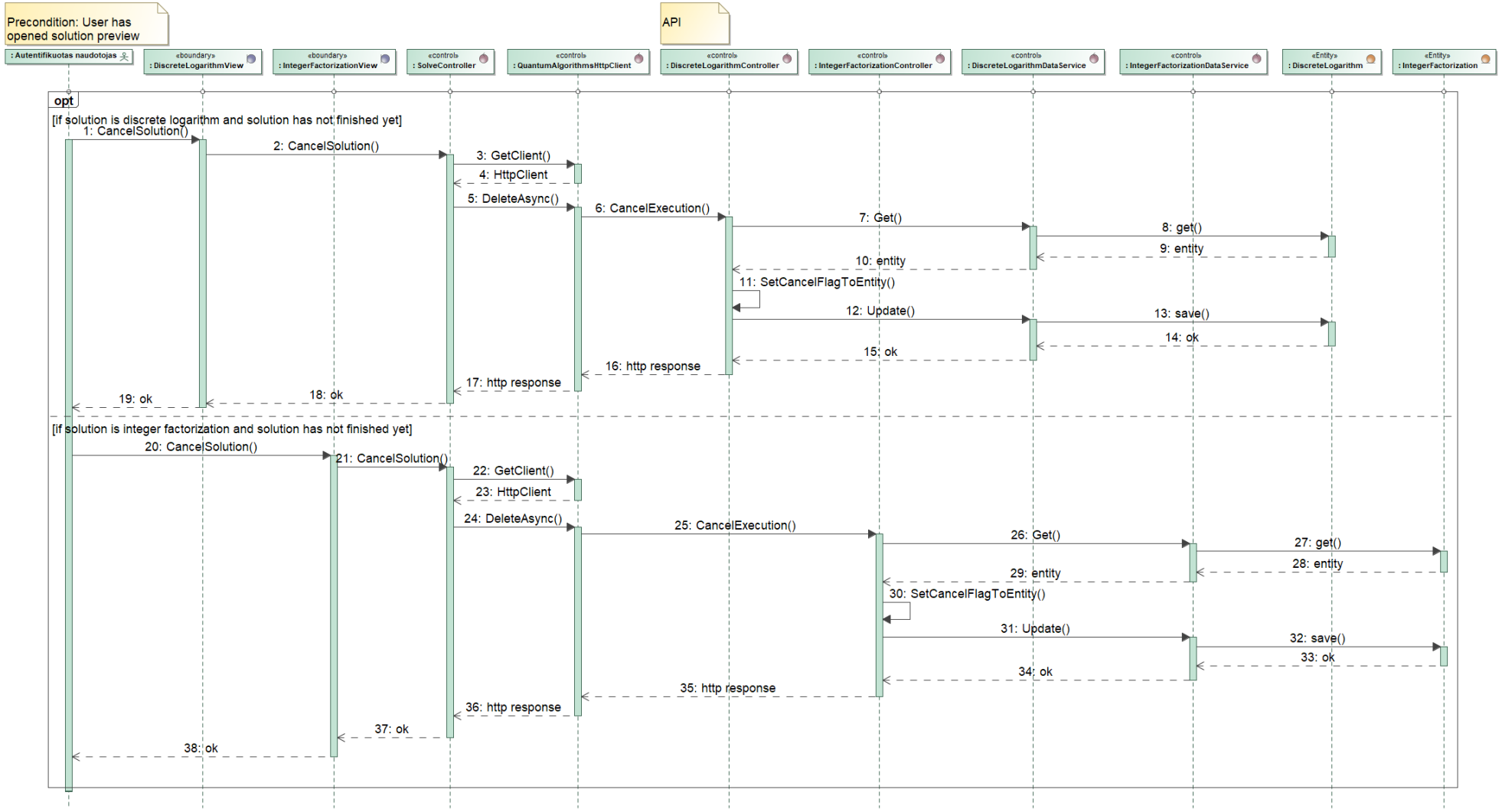


2.17 pav. „Gauti uždavinių sprendimus pagal būseną“ panaudojimo atvejo sekų diagrama

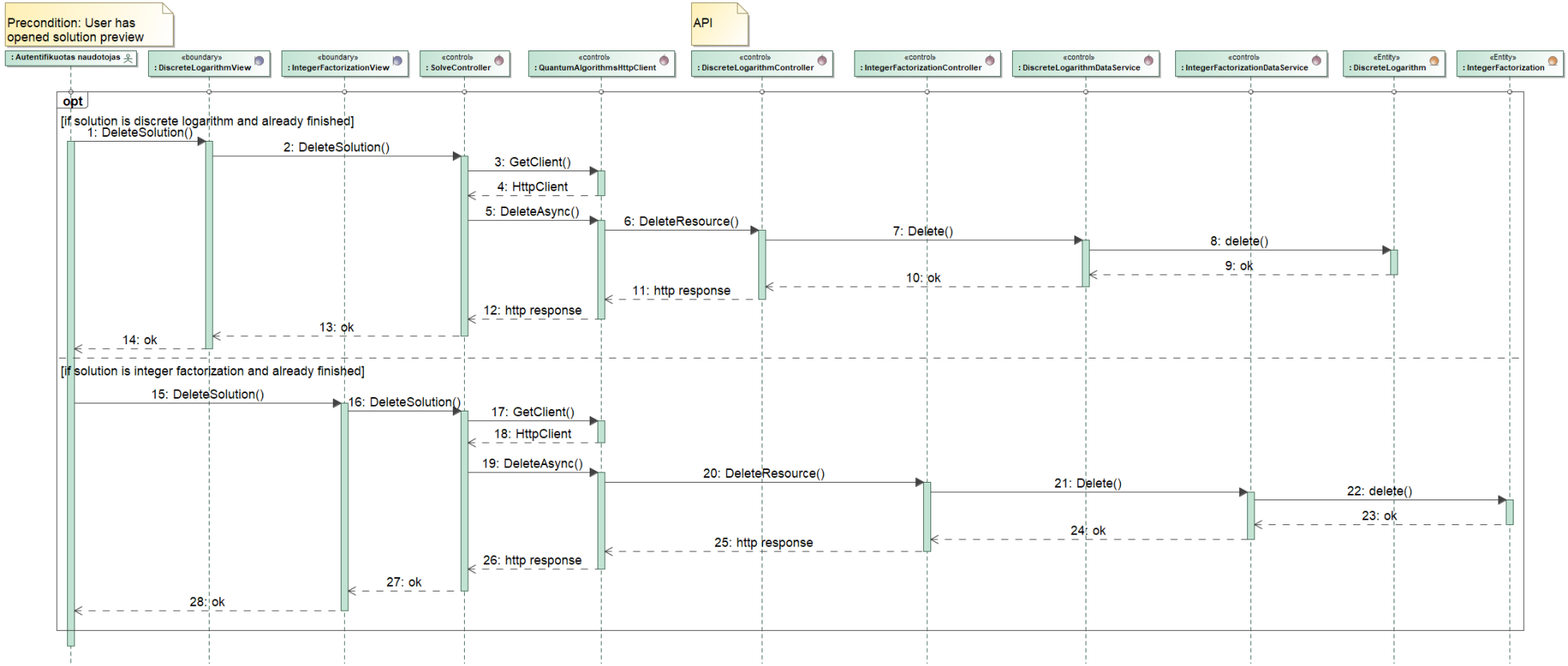
Šios dvi sekų diagramos pilnai apibrėžia uždavinių peržiūros pagal būseną funkciją. „API“ komentaras diagramoje rodo tašką, kuriame iš saityno taikomosios programos yra pereinama prie problemų sprendimo serviso. Saityno taikomoji programa bendravimui su problemų sprendimo servisu tam tikslui skirtą ir sukonfigūruotą *QuantumAlgorithmsHttpClient* klasę, kuri pati pasirūpina prieigos žetono pridėjimu prie užklausos.



2.18 pav. „Peržiūrėti uždavinio sprendimą“ panaudojimo atvejo sekų diagrama



2.19 pav. „Atšaukti uždavinio sprendimą“ panaudojimo atvejo sekų diagrama



2.20 pav. „Ištrinti uždavinio sprendimą“ panaudojimo atvejo sekų diagrama

Paskutinės trys sekų diagramos nusako problemos sprendimo peržiūrėjimo funkcijos bei papildomų galimų funkcijų su sprendimu – uždavinio sprendimo atšaukimo bei uždavinio sprendimo ištrynimo – vykdymo eigą.

2.3.3. Duomenų kontrolė

Problemų sprendimo servise problemų sprendimo užklausoms yra atliekama uždavinio duomenų kontrolė – kadangi kvantinių procesų simuliavimas užima daug resursų ir laiko, yra nustatyti apribojimai, kurie apriboja visus parametrus rėžiuose nuo 2 iki 100. Kadangi visi parametrai yra sveikieji skaičiai, kitos duomenų kontrolės problemų sprendimo užklausoms nėra. Tokie patys apribojimai yra nustatyti ir saityno taikomojoje programoje, kur parametrai tikrinami *Javascript* pagalba.

Problemų sprendimo servise sprendimų gavimo užklausoms yra nustatyti korektiško sprendimo identifikatoriaus apribojimai – bandant pasiekti ar ištrinti neegzistuojantį sprendimą, grąžinamas HTTP 404 kodas. Tačiau tai negalioja užklausoms, kur bandoma gauti keletą sprendimų iš karto.

3. TESTAVIMAS

Aprašoma su sukurtos įrangos testavimu susijusi informacija (8 – 12 lapai). Skyriaus struktūra ir pavadinimas priklauso nuo baigiamojo darbo specializacijos ir pačios temos specifikos.

Nurodomas įrangos testavimo planas, testavimo duomenų rinkiniai ir gauti rezultatai. Nurodoma sistemos specifikacija ir sąlygos, prie kurių buvo atliekamas testavimas.

3.1. Testavimo planas

Suprojektavus ir realizavus sistemą buvo sudarytas toks testavimo planas:

1. Atlikti statinę kodo analizę visoms sistemoms dalims;
2. Sukurti komponentų testus (angl. *unit tests*) pagrindinės dalies – problemų sprendimo serviso – komponentams ištestuoti;
3. Rankiniu būdu ištestuoti problemų sprendimo serviso API funkcijas;
4. Rankiniu būdu ištestuoti visus panaudojimo atvejus per saityno taikomąją programą.

3.2. Testavimo kriterijai

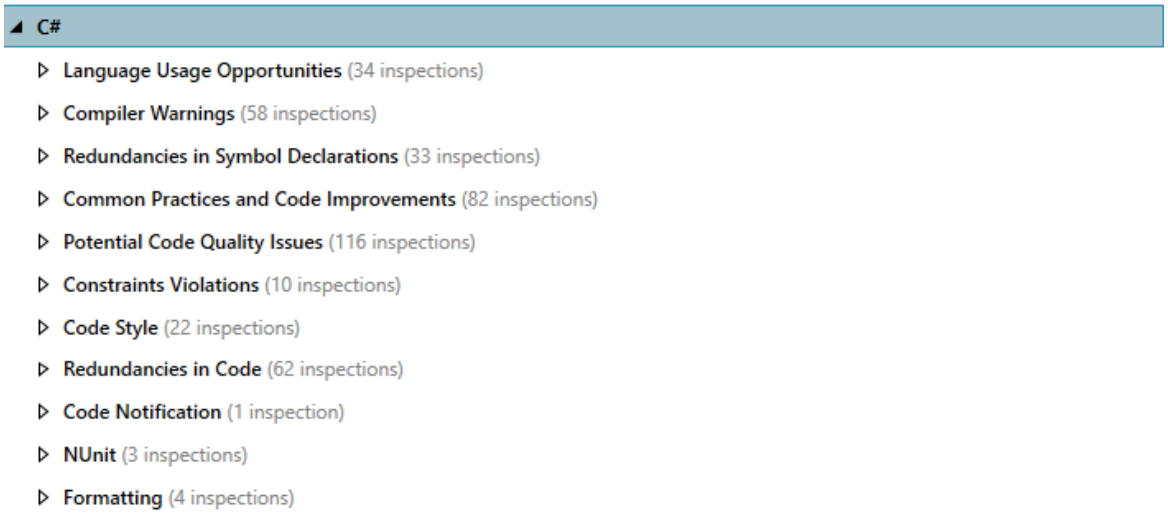
Sistema buvo testuojama pagal šiuos kriterijus:

1. Statinė kodo analizė turi būti 100 % sėkminga, t. y. neturi būti jokių statinės kodo analizės klaidų parašytame kode;
2. Komponentų testai turi padengti bent 60 % problemų sprendimo serviso komponentų;
3. Visi komponentų testai turi būti sėkmingi;
4. Problemų sprendimo serviso API funkcijos turi veikti pagal specifikaciją;
5. Saityno taikomoji programa turi korektiškai perduoti duomenis problemų sprendimo servisui bei korektiškai atvaizduoti gautus rezultatus iš jo.

3.3. Automatinis testavimas

3.3.1. Statinė kodo analizė

Statinė kodo analizė buvo atliekama pasitelkus statinės kodo analizės įrankiu, kuris yra kartu įdiegiamas su *ReSharper Ultimate* įskiepiu *Microsoft Visual Studio 2017* integruotai kūrimo aplinkai. Statinė kodo analizė buvo atliekama su numatytomis įrankio taisyklėmis, kurių yra 425. Taisyklių kategorijos pavaizduotos 3.1 pav..



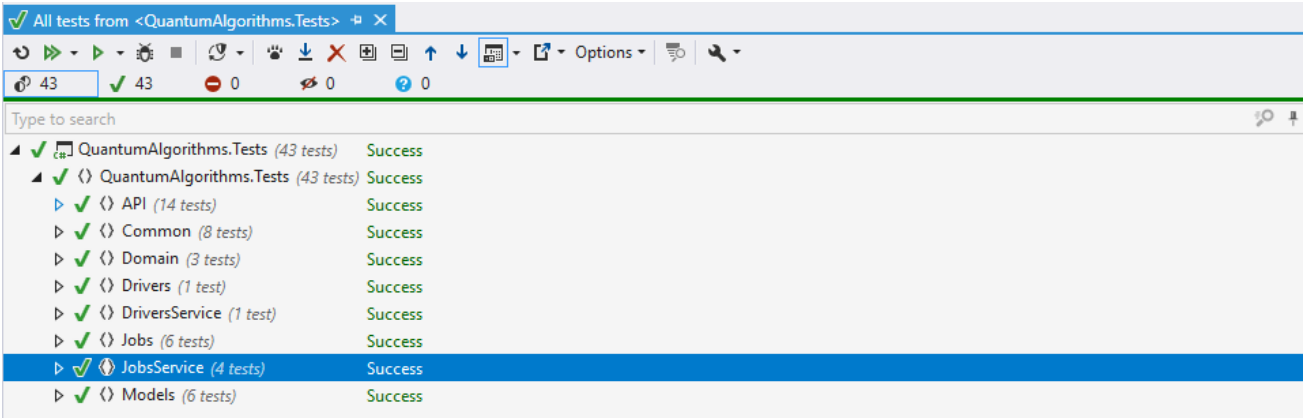
3.1 pav. Statinės kodo analizės įrankio taisyklių kategorijos

Atlikus statinę kodo analizę, buvo rastos dvi klaidos, tačiau jos buvo iš trečiųjų šalių bibliotekų. Todėl galime teigti, jog statinės kodo analizės kriterijai buvo įvykdyti.

3.3.2. Komponentų testai

Komponentai buvo sukurti pasitelkus *NUnit 3.x* karkasą. Testai buvo vykdomi su integruotu *ReSharper Ultimate* įskiepio testų vykdytojų, kodo padengimas buvo skaičiuojamas su tuo pačiu įskiepiu.

Iš viso buvo sukurti 43 komponentų testai, testuojantys problemų sprendimo serviso komponentus. Visi komponentų testai buvo įvykdyti sėkmingai. Vykdymo rezultatai pateikti 3.2 pav..



3.2 pav. Komponentų testų vykdymo rezultatai

Įvykdžius komponentų testus, buvo apskaičiuotas bendras 68 % kodo padengimas testais. Kodo padengimo rezultatai pateikti 3.3 pav..

Symbol	Coverage (%)	Uncovered/Total Stmt
Total	68%	412/1293
QuantumAlgorithms.API	48%	156/299
QuantumAlgorithms.Common	70%	17/56
QuantumAlgorithms.Domain	100%	0/44
QuantumAlgorithms.Drivers	49%	19/37
QuantumAlgorithms.DriversService	100%	0/13
QuantumAlgorithms.Jobs	34%	201/303
QuantumAlgorithms.JobsService	93%	4/54
QuantumAlgorithms.Models	86%	11/79

3.3 pav. Komponentų testų kodo padengimo rezultatai

Pagal komponentų testų vykdymo bei jų kodo padengimo rezultatus, galime teigti, jog komponentų testų kriterijai buvo įvykdyti.

3.4. Rankinis testavimas

3.4.1. Problemų sprendimo serviso API testavimas

Problemų sprendimo serviso API buvo testuojamas pasitelkus *Postman* programinę įrangą, kuri yra skirta HTTP užklausų siuntimui bei atsakymų gavimui. Testavimas buvo integracinis – t. y. užklausos buvo siunčiamos į lokaliai įdiegtą problemų sprendimo servisą. Prieš testavimą buvo paruošta testavimo aplinka – sugeneruota problemų sprendimų.

Buvo testuojamas kiekvienas prieinamas API galinis taškas (angl. *endpoint*). Tokių galinių taškų iš viso yra 16. Kiekvienam API galiniam taškui buvo sudaryta po vieną ar daugiau testavimo

atvejų. Iš viso buvo sudaryti 48 testavimo atvejai. Pavyzdiniai vieno API galinio taško testavimo atvejai pateikti 3.1 – 3.6 lentelėse.

3.1 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis be prieigos žetono

API galinis taškas	GET .../Api/DiscreteLogarithm
Prieigos žetonas	Nėra
Tikėtinas HTTP statuso kodas	401
Tikėtinas atsakymo turinys	Nėra

3.2 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis su negaliojančiu prieigos žetonu

API galinis taškas	GET .../Api/DiscreteLogarithm
Prieigos žetonas	Pateikiamas negaliojantis
Tikėtinas HTTP statuso kodas	401
Tikėtinas atsakymo turinys	Nėra

3.3 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis su galiojančiu prieigos žetonu

API galinis taškas	GET .../Api/DiscreteLogarithm
Prieigos žetonas	Pateikiamas galiojantis
Tikėtinas HTTP statuso kodas	200
Tikėtinas atsakymo turinys	Sąrašas diskretaus logaritmo uždavinio sprendimų, priklausančių prieigos žetonu autentifikuotam naudotojui

3.4 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis su tuščiu identifikatorių filtru

API galinis taškas	GET .../Api/DiscreteLogarithm?ids=[]
Prieigos žetonas	Pateikiamas galiojantis
Tikėtinas HTTP statuso kodas	200
Tikėtinas atsakymo turinys	Tuščias sąrašas

3.5 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis su identifikatorių filtru (1)

API galinis taškas	GET .../Api/DiscreteLogarithm?ids=[05CE212D-4D5B-4F70-A0D1-C92C5F43D94F, 8AC79F80-7F01-423F-AC37-4BBFDEAA3C68]
Prieigos žetonas	Pateikiamas galiojantis
Tikėtinas HTTP statuso kodas	200
Tikėtinas atsakymo turinys	Sąrašas iš dviejų diskretaus logaritmo uždavinių sprendimų

3.6 lentelė. Diskretaus logaritmo sprendimų gavimo testavimo atvejis su identifikatorių filtru (2)

API galinis taškas	GET .../Api/DiscreteLogarithm?ids=[05CE212D-4D5B-4F70-A0D1-C92C5F43D94F, 29C8E25D-113F-442C-A1C0-262BFA4BDAEA, invalid]
Prieigos žetonas	Pateikiamas galiojantis
Tikėtinas HTTP statuso kodas	200
Tikėtinas atsakymo turinys	Sąrašas iš vieno diskretaus logaritmo uždavinio sprendimo (antras identifikatorius identifikuoja kito naudotojo uždavinio sprendimą)

Ištestavus visus testavimo atvejus, visuose buvo gauti tikėtini rezultatai, todėl problemų sprendimo serviso testavimo kriterijai buvo įvykdyti.

3.4.2. Panaudojimo atvejų per saityno taikomąją programą testavimas

Panaudojimo atvejai per saityno taikomąją programą buvo testuojami tiesioginiu būdų, t. y. vykdant įprasto naudotojo veiksmus saityno taikomojoje programoje. Testavimas buvo integracinis – t. y. veiksmai buvo vykdomi su lokaliai įdiegta visa sistema.

Panaudojimo atvejai buvo testuojami sudarant testavimo scenarijus. Testavimo scenarijus apima vieną ar daugiau panaudojimo atvejų. Iš viso buvo sudaryta 16 tokių testavimo scenarijų. Vienas pavyzdinis testavimo scenarijus pateikiamas 3.7 lentelėje.

3.7 lentelė. Panaudojimo atvejų testavimo per saityno taikomąją programą scenarijus

Pradinės sąlygos	
Naudotojas turi turėti paskyra su socialinio tinklo *Facebook* prisijungimu asmeninio sertifikavimo paslaugų teikėjo servise. Naudotojas turi būti prisijungęs prie *Facebook* socialinio tinklo. Naudotojas turi būti atsijungęs nuo saityno taikomosios programos ir atsidaręs pagrindinį jos langą.	
Veiksmas	**Tikėtinas rezultatas**
Paspaudžiamas „Log In“ meniu punktas.	Atidaromas asmeninio sertifikavimo paslaugų serviso prisijungimo langas.
Paspaudžiamas mygtukas „Facebook“ prie „External Login“ srities.	Naudotojas prijungiamas ir yra grąžinamas į pagrindinį svetainės langą.
Paspaudžiamas „Solve“ meniu punktas.	Atidaromas problemų sprendimo langas.
Įvedamas skaičius 77 prie sveikojo skaičiaus skaidymo daugikliais uždavinio ir paspaudžiamas mygtukas „Solve“	Atidaromas problemos sprendimo langas.
Paspaudžiamas „Solutions“ meniu punktas.	Atidaromas vykdomų problemų sprendimų langas.
Paspaudžiamas pirmasis problemos sprendimas.	Atidaromas tas pats vykdomas sprendimas.
Paspaudžiamas mygtukas „Cancel solution“.	Atsidaro atšaukimo patvirtinimo dialogas.
Patvirtinamas sprendimo atšaukimas.	Atidaromas tas pats atšauktas sprendimas.
Paspaudžiamas „Canceled“ punktas kairėje.	Atidaromas atšauktų sprendimų sąrašas.
Paspaudžiamas pirmasis problemos sprendimas.	Atidaromas tas pats atšauktas sprendimas.
Paspaudžiamas „Log Out“ meniu punktas.	Naudotojas atjungiamas ir yra grąžinamas į pagrindinį svetainės langą.
Du kartus paspaudžiamas naršyklės „atgal“ mygtukas	Pasirodo klaidingo puslapio pranešimas.

Ištestavus visus testavimo scenarijus, visuose buvo gauti tikėtini rezultatai, todėl saityno taikomosios programos testavimo kriterijai buvo įvykdyti.

4. DOKUMENTACIJA NAUDOTOJUI

4.1. Apibendrintas sistemos galimybių aprašymas

Realizuota sistema teikia API sveiko skaičiaus skaidymo pirminiais dauginamaisiais ir diskretaus logaritmo skaičiavimo uždaviniams spręsti bei valdyti uždavinių sprendimus.

Taip pat, ši sistema teikia ir saityno taikomąją programą, kuri naudojasi teikiamu API ir pateikia jo funkcijas per grafinę sąsają.

4.2. API specifikacija

Šiame skyrelyje 4.1 – 4.10 lentelėse pateikiama problemų sprendimo serviso API specifikacija.

4.1 lentelė. Visų diskretaus logaritmo sprendimų gavimo specifikacija

Kelias	/Api/DiscreteLogarithm		
HTTP metodas	GET		
Paskirtis	Gauti visus diskretaus logaritmo sprendimus		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Page	Ne	Sveikas skaičius	Puslapio numeris (resursų sudalinimas į puslapius)
PageSize	Ne	Sveikas skaičius	Puslapio dydis (resursų sudalinimas į puslapius)
Ids	Ne	Simbolių eilutė	Identifikatorių masyvas, pagal kuriuos filtruojama
Statuses	Ne	Simbolių eilutė	Būsenų masyvas, pagal kuriuos filtruojama
Atsakymo kodas	**Apibrėžimas**		
200	Sprendimai sėkmingai pasiekti		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		

Atsakymo modelis (200)	<pre>[
	  {
	  "id": "string",
	  "input": {
	  "generator": 0,
	  "result": 0,
	  "modulus": 0,
	  },
	  "output": {
	  "discreteLogarithm": 0,
	  },
	  "status": 0,
	  "statusString": "string",
	  "startTime": "2018-05-18T01:40:43.3109247",
	  "finishTime": "2018-05-19T01:40:43.3109247",
	  "messages": [
	  {
	  "timeStamp": "2018-05-18T23:43:39.0776347",
	  "message": "string",
	  "severity": 0,
	  "severityString": "string"
	  }
	  ]
	  }
	]</pre>		

4.2 lentelė. Diskretaus logaritmo sprendimo pradėjimo specifikacija

Kelias	/Api/DiscreteLogarithm
HTTP metodas	POST
Paskirtis	Pradėti diskretaus logaritmo sprendimą
Galimi užklausos turinio tipai	`application/json, application/xml`

Užklausos modelis	`{`
	`  "generator": 0,`
	`  "result": 0,`
	`  "modulus": 0`
	`}`

Atsakymo kodas	**Apibrėžimas**
201	Sprendimas sėkmingai pradėtas
400	Blogas užklausos modelis
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)

Atsakymo modelis (201)	`{`
	`  "id": "string",`
	`  "input": {`
	`  "generator": 0,`
	`  "result": 0,`
	`  "modulus": 0,`
	`  },`
	`  "output": {`
	`  "discreteLogarithm": 0,`
	`  },`
	`  "status": 0,`
	`  "statusString": "string",`
	`  "startTime": "2018-05-18T01:40:43.3109247",`
	`  "finishTime": "2018-05-19T01:40:43.3109247",`
	`  "messages": [`
	`  {`
	`  "timeStamp": "2018-05-18T23:43:39.0776347",`
	`  "message": "string",`
	`  "severity": 0,`
	`  "severityString": "string"`
	`  }`
	`  ]`
	`}`

4.3 lentelė. Konkretaus diskretaus logaritmo sprendimo gavimo specifikacija

Kelias	/Api/DiscreteLogarithm/{Id}		
HTTP metodas	GET		
Paskirtis	Gauti konkretų diskretaus logaritmo sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
200	Sprendimas sėkmingai pasiektas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

Atsakymo modelis (200)	`{`
	`  "id": "string",`
	`  "input": {`
	`  "generator": 0,`
	`  "result": 0,`
	`  "modulus": 0,`		

	<pre> },
	  "output": {
	  "discreteLogarithm": 0,
	  },
	  "status": 0,
	  "statusString": "string",
	  "startTime": "2018-05-18T01:40:43.3109247",
	  "finishTime": "2018-05-19T01:40:43.3109247",
	  "messages": [
	  {
	  "timeStamp": "2018-05-18T23:43:39.0776347",
	  "message": "string",
	  "severity": 0,
	  "severityString": "string"
	  }
	  ]
	}</pre>

4.4 lentelė. Diskretaus logaritmo sprendimo ištrynimo specifikacija

Kelias	/Api/DiscreteLogarithm/{Id}		
HTTP metodas	DELETE		
Paskirtis	Ištrinti konkretų diskretaus logaritmo sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
204	Sprendimas sėkmingai ištrintas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

4.5 lentelė. Diskretaus logaritmo sprendimo atšaukimo specifikacija

Kelias	/Api/DiscreteLogarithmRun/{Id}		
HTTP metodas	DELETE		
Paskirtis	Atšaukti konkretų diskretaus logaritmo sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
204	Sprendimas sėkmingai atšauktas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

4.6 lentelė. Visų sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimų gavimo specifikacija

Kelias	/Api/IntegerFactorization		
HTTP metodas	GET		
Paskirtis	Gauti visus sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimus		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Page	Ne	Sveikas skaičius	Puslapio numeris (resursų sudalinimas į puslapius)
PageSize	Ne	Sveikas skaičius	Puslapio dydis (resursų sudalinimas į puslapius)
Ids	Ne	Simbolių eilutė	Identifikatorių masyvas, pagal kuriuos filtruojama
Statuses	Ne	Simbolių eilutė	Būsenų masyvas, pagal kuriuos filtruojama
Atsakymo kodas	**Apibrėžimas**		
200	Sprendimai sėkmingai pasiekti		

401	Neautorizuota prieiga prie resursų (blogas/nesamas prieigos žetonas)

Atsakymo modelis (200)	[
	  {
	  "id": "string",
	  "input": {
	  "number": 0
	  },
	  "output": {
	  "p": 0,
	  "q": 0
	  },
	  "status": 0,
	  "statusString": "string",
	  "startTime": "2018-05-18T01:40:43.3109247",
	  "finishTime": "2018-05-19T01:40:43.3109247",
	  "messages": [
	  {
	  "timeStamp": "2018-05-18T23:43:39.0776347",
	  "message": "string",
	  "severity": 0,
	  "severityString": "string"
	  }
	  ]
	  }
	]

4.7 lentelė. Sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimo pradėjimo specifikacija

Kelias	/Api/IntegerFactorization
HTTP metodas	POST
Paskirtis	Pradėti sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimą
Galimi užklausos turinio tipai	application/json, application/xml

Užklausos modelis	{
	  "number": 0
	}

Atsakymo kodas	**Apibrėžimas**
201	Sprendimas sėkmingai pradėtas
400	Blogas užklausos modelis
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)

Atsakymo modelis (201)	{
	  "id": "string",
	  "input": {
	  "number": 0
	  },
	  "output": {
	  "p": 0,
	  "q": 0
	  },
	  "status": 0,
	  "statusString": "string",
	  "startTime": "2018-05-18T01:40:43.3109247",
	  "finishTime": "2018-05-19T01:40:43.3109247",
	  "messages": [
	  {
	  "timeStamp": "2018-05-18T23:43:39.0776347",

	<pre>"message": "string",
	"severity": 0,
	"severityString": "string"
	}
	]
	}</pre>

4.8 lentelė. Konkretaus sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimo gavimo specifikacija

Kelias	/Api/IntegerFactorization/{Id}		
HTTP metodas	GET		
Paskirtis	Gauti konkretų sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
200	Sprendimas sėkmingai pasiektas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

Atsakymo modelis (200)	<pre>{
	"id": "string",
	"input": {
	"number": 0
	},
	"output": {
	"p": 0,
	"q": 0
	},
	"status": 0,
	"statusString": "string",
	"startTime": "2018-05-18T01:40:43.3109247",
	"finishTime": "2018-05-19T01:40:43.3109247",
	"messages": [
	{
	"timeStamp": "2018-05-18T23:43:39.0776347",
	"message": "string",
	"severity": 0,
	"severityString": "string"
	}
	]
	}</pre>		

4.9 lentelė. Sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimo ištrynimo specifikacija

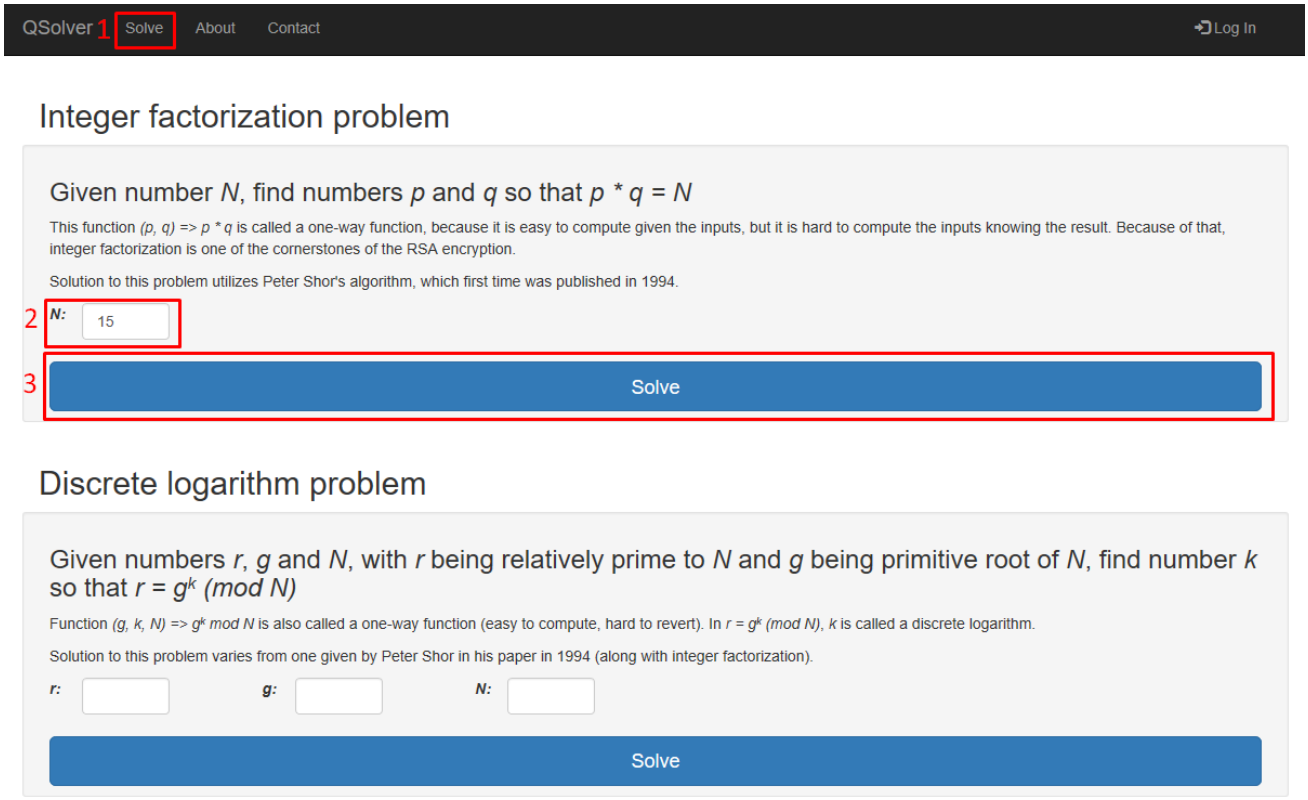
Kelias	/Api/IntegerFactorization/{Id}		
HTTP metodas	DELETE		
Paskirtis	Ištrinti konkretų sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
204	Sprendimas sėkmingai ištrintas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

4.10 lentelė. Sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimo atšaukimo specifikacija

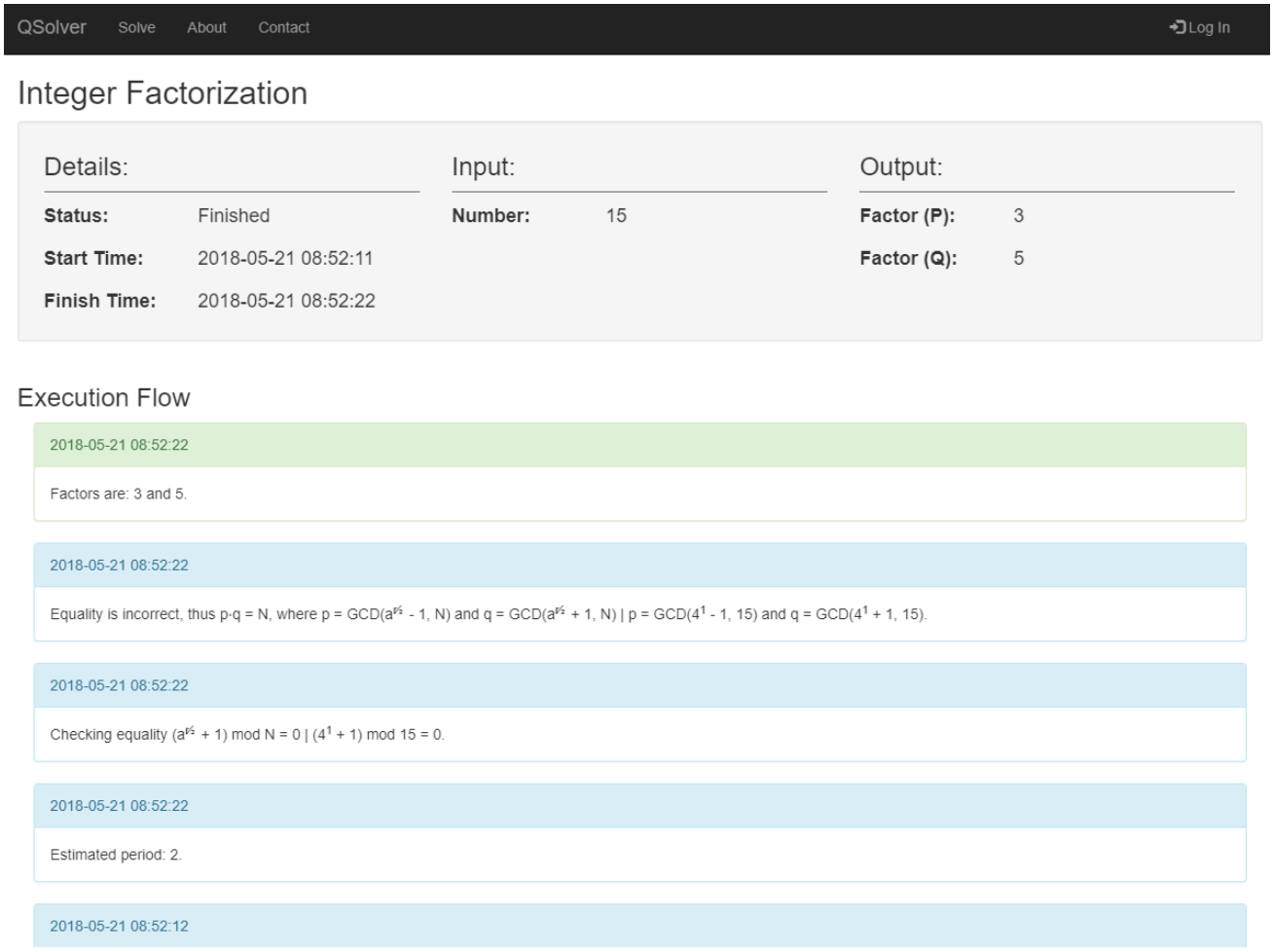
Kelias	/Api/IntegerFactorizationRun/{Id}		
HTTP metodas	DELETE		
Paskirtis	Atšaukti konkretų sveikojo skaičiaus skaidymo pirminiais skaičiais sprendimą		
Parametras	**Ar būtinas**	**Tipas**	**Apibrėžimas**
Id	Taip	Simbolių eilutė	Sprendimo identifikatorius
Atsakymo kodas	**Apibrėžimas**		
204	Sprendimas sėkmingai atšauktas		
401	Neautorizuota prieiga (blogas/nesamas prieigos žetonas)		
404	Sprendimas su nurodytu identifikatoriumi nerastas		

4.3. Vartotojo vadovas

Norint pradėti naudotis sistema, nereikia net užsiregistruoti. Problemas galima spręsti ir neprisijungus. Norint tai padaryti, meniu juostoje reikėtų spausti punktą „Solve“, tuomet atsidarytų langas, kuriame galima pasirinkti, kurį uždavinį norima spręsti. Pasirinkus uždavinį ir suvedus jo duomenis, po konkrečiu uždaviniu reikia paspausti mygtuką „Solve“. Tuomet būsite nukreipti į uždavinio sprendimo langą. Visa eiga parodyta 4.1 pav. ir 4.2 pav..

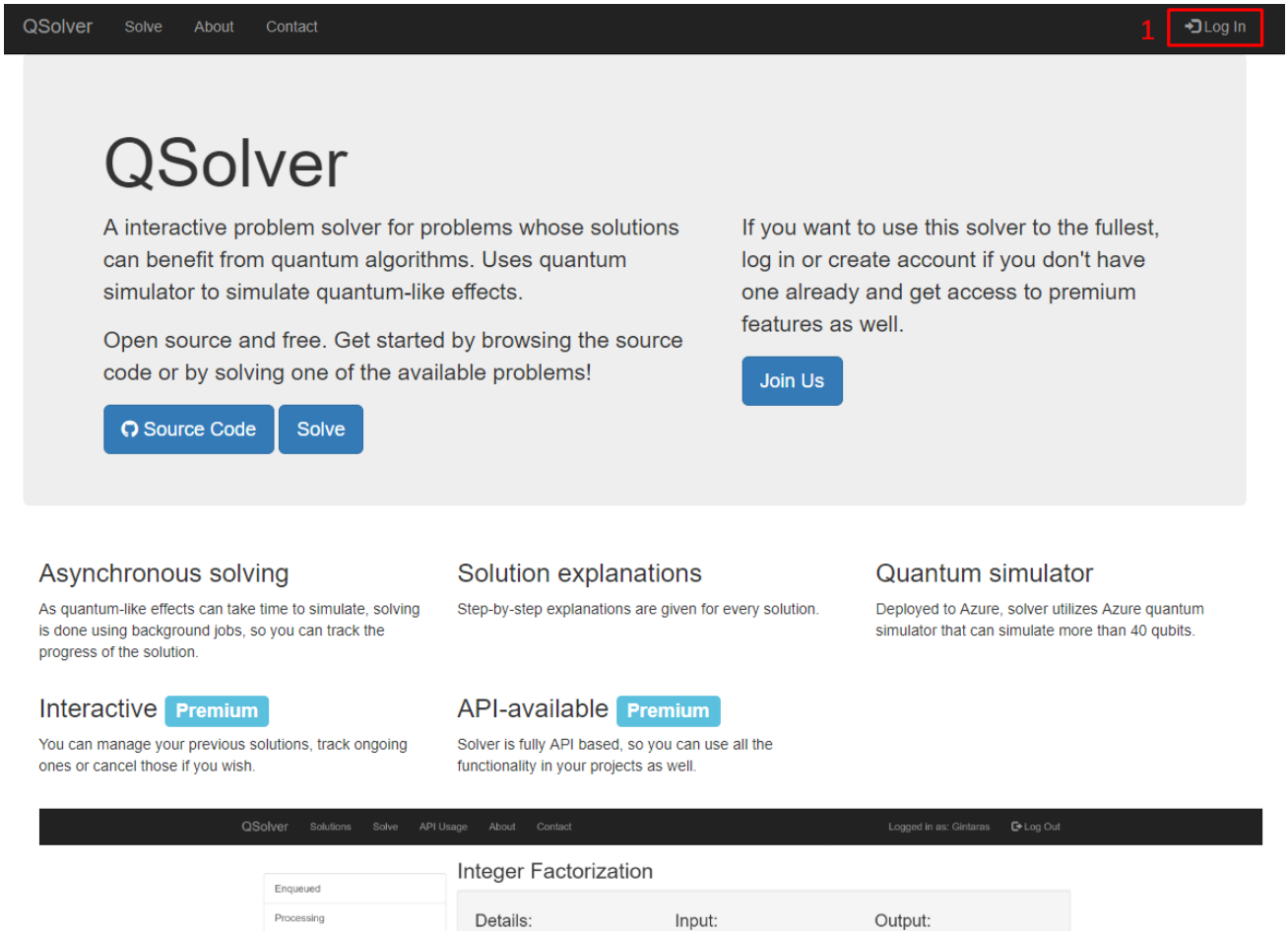


4.1 pav. Problemos sprendimo žingsniai

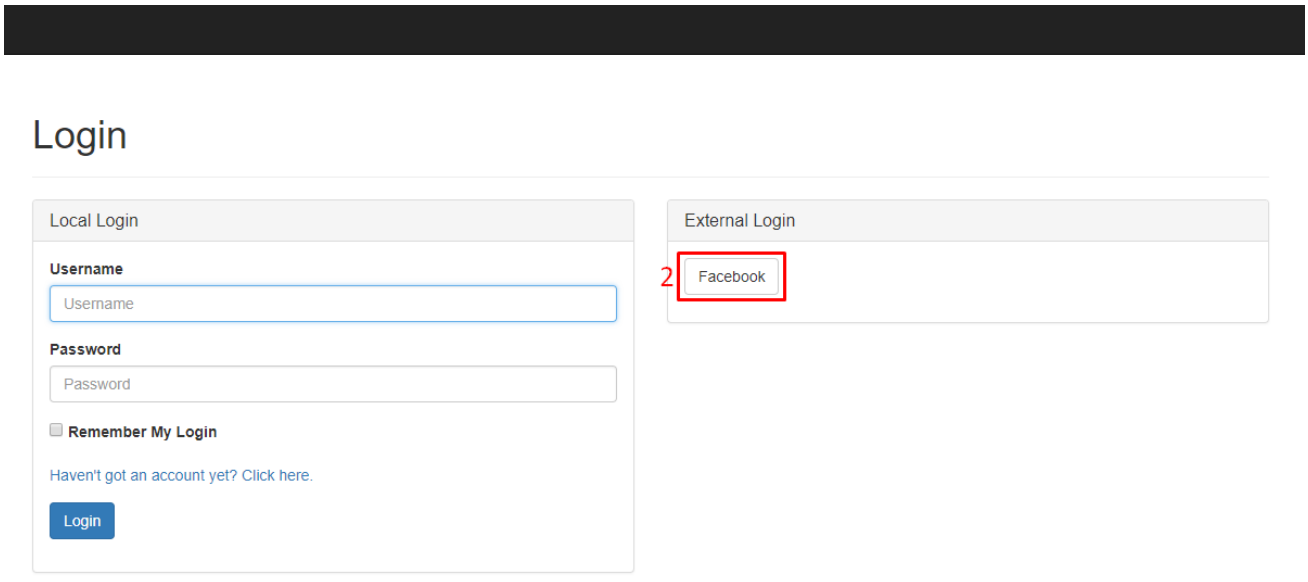


4.2 pav. Problemos sprendimo langas

Tačiau norėdami valdyti savo sprendimus – juos peržiūrėti, tvarkyti pašalinant, atšaukti vykdymo eigoje – turite prisijungti. Prie sistemos labai patogu prisijungti, kadangi ji palaiko *Facebook* prisijungimą. Norėdami prisijungti, spauskite meniu punktą „Log In“ (4.3 pav.), o tuomet naujame atsidariusiame lange spauskite mygtuką „Facebook“ (4.4 pav.).

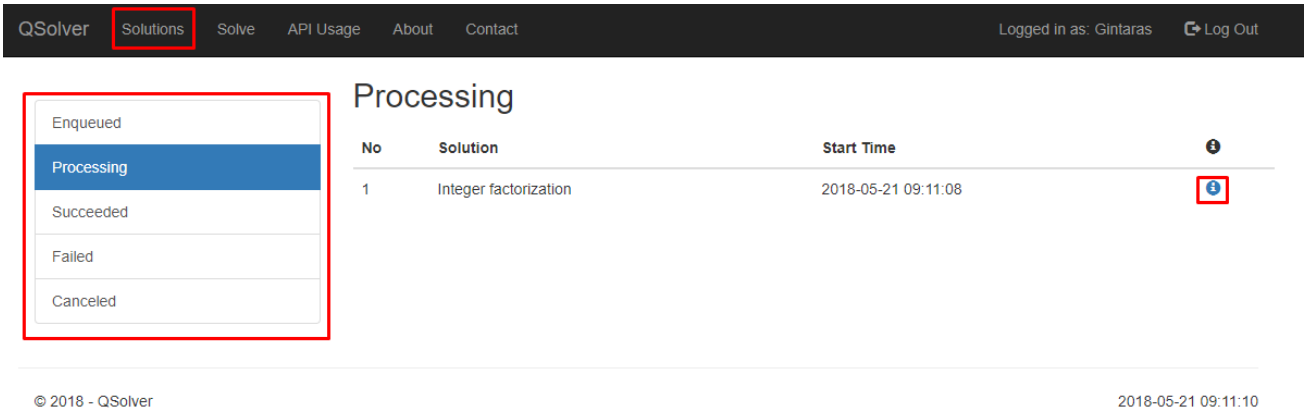


4.3 pav. Prisijungimas prie sistemos su „Facebook“ paskyra (1)



4.4 pav. Prisijungimas prie sistemos su „Facebook“ paskyra (2)

Dabar, kai esate prisijungę, galite peržiūrėti visus savo sprendimus (kuriuos darėte prisijungęs) paspausdami meniu punktą „Solutions“. Atsidariusiame lange galite matyti šiuo metu vykdomus sprendimus, o kairėje lango pusėje esančioje panelėje galite pasirinkti, kuriuos sprendimus norite peržiūrėti. Detaliai peržiūrėti sprendimą galite paspaude informacijos ženkliuką esantį prie sprendimo (4.5 pav.).



4.5 pav. Sprendimų peržiūrėjimas

4.4. Diegimo vadovas

Pagal techninę specifikaciją bei diegimo diagramą, sistema turi būti talpinama *Microsoft Azure Cloud* platformoje. Kadangi projektas yra atviro kodo ir yra patalpintas *GitHub* talpykloje, paprasčiausią sistemą įdiegti taip:

1. *Azure* portale pagal techninę specifikaciją susikurti numatytas virtualias mašinas, DBVS serverius;
2. Asmeninėje *GitHub* talpykloje susikurti šios talpyklos (https://github.com/GintV/quantum-algorithms) pagrindinės šakos kopiją;
3. *Azure* portale nustatyti, jog sistemos dalys būtų diegiamos iš asmeninės talpyklos.

Detalesnę informaciją apie tai, kaip reiktų 1 ir 3 žingsnius galima rasti *Azure* portale, o 2 žingsnį – *GitHub* svetainėje.

4.5. Administravimo vadovas

Kadangi sistema yra patalpinta *Microsoft Azure Cloud* platformoje, *Azure* portale galima rasti visą informaciją apie sistemos dabartinę būseną, jos užimtumą, nepavykusias užklausas bei ištikusias klaidas. Taip pat galima rasti kiekvienos sistemos dalies žurnalą, kuriame galima rasti įrašus apie sistemos veikimo informaciją.

5. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

Atlikus darbą prieitą prie tokių išvadų:

- Ištyrus panašias rinkoje esančias sistemas, nebuvo rasta nė viena sistema, kuri įgyvendintų vienokių ar kitokių problemų sprendimą pasitelkiant ir vykdant kvantinius algoritmus kvantiniame simuliatoriuje. Tai buvo pagrindinė priežastis, dėl kurios ši sistema buvo sukurta.
- Išanalizavus pasirinktas problemas ir jų sprendimus pasitelkiant kvantinius algoritmus buvo susidurta su problema, jog šiuo metu įgyvendinti periodo radimo funkciją dviejų kintamųjų funkcijai yra ganėtinai sunku arba išvis neįmanoma. Dėl šios priežasties, diskretaus logaritmo skaičiavimo algoritmas buvo modifikuotas ir pritaikytas veikti su periodo radimo funkcija vieno kintamojo funkcijai.
- Sistemos projektavimo ir realizavimo metu buvo susidurta su problema, jog *ASP.NET Core 2* ir kitos *ASP.NET* taikomosios programos esant neaktyvumui (nesant jokios užklausoms) gali save pačios sunaikinti, tad vykdomi foniniai skaičiavimų darbai tiesiog dingtų. Šiai problemai spręsti nuo savarankiško foninių darbų valdymo buvo pereita prie *Hangfire* karkaso, kuris padeda kurti ir valdyti foninius darbus, bei neleidžia taikomajai programai pačiai savęs susinaikinti, kol yra aktyvių darbų.
- Kuriant komponentų testus su *NUnit 3.x* karkasu, buvo sužinota ir išmokta, kaip specialiai reikia susikonfigūruoti testavimo projektą, jog būtų galima naudotis šiuo karkasu, o testus vykdyti su *ReSharper* įskiepio integruotu testų vykdytoju.
- Ruošiant sistemos dokumentacija buvo koncentruojamasi tiek į patyrusius sistemos naudotojus, kurie naudosis API funkcijomis, tiek į nepatyrusius – kurie naudotis saityno taikomąja programa, todėl sudaryta tiek techninė API specifikacija, tiek paprastas naudotojo vadovas.

6. LITERATŪRA

[1] Wikipedia, „Integer factorization - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Integer_factorization. [Kreiptasi 12 05 2018].

[2] Wikipedia, „Discrete logarithm - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Discrete_logarithm. [Kreiptasi 12 05 2018].

[3] P. O'Malley, R. Babbush, I. Kivlichan, J. Romero, J. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, E. Jeffrey, E. Lucero, A. Megran, J. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. White, P. Coveney, P. Love, H. Neven, A. Aspuru-Guzik ir J. Martinis, „Phys. Rev. X 6, 031007 (2016) - Scalable Quantum Simulation of Molecular Energies,“ [Tinkle]. Available: https://journals.aps.org/prx/abstract/10.1103/PhysRevX.6.031007. [Kreiptasi 12 05 2018].

[4] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker ir M. Troyer, „Elucidating reaction mechanisms on quantum computers | PNAS,“ [Tinkle]. Available: http://www.pnas.org/content/early/2017/06/30/1619152114. [Kreiptasi 12 05 2018].

[5] P. W. Shor, „Algorithms for Quantum Computation: Discrete Log and Factoring,“ AT&T Bell Labs, Murray Hill, NJ, 1994.

[6] E. Farhi, J. Goldstone ir S. Gutmann, „A Quantum Algorithm for the Hamiltonian NAND Tree,“ Boston, MA, 2007.

[7] D. Alpern, „Java programs written by Dario Alejandro Alpern,“ [Tinkle]. Available: https://www.alpertron.com.ar/JAVAPROG.HTM. [Kreiptasi 14 05 2018].

[8] Wikipedia, „Lenstra elliptic-curve factorization - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Lenstra_elliptic-curve_factorization. [Kreiptasi 14 05 2018].

[9] Wikipedia, „Pohlig–Hellman algorithm - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Pohlig%E2%80%93Hellman_algorithm. [Kreiptasi 14 05 2018].

[10] B. Allen ir D. Baier, „Welcome to IdentityServer4 - IdentityServer4 1.0.0 documentation,“ [Tinkle]. Available: http://docs.identityserver.io/en/release/index.html. [Kreiptasi 14 05 2018].

[11] Microsoft, „The Q# Programming Language | Microsoft Docs,“ [Tinkle]. Available: https://docs.microsoft.com/en-us/quantum/quantum-qr-intro?view=qsharp-preview. [Kreiptasi 14 05 2018].

[12] N. Gershenfeld ir I. L. Chuang, „Quantum Computing with Molecules,“ 1998.

[13] Wikipedia, „Qubit - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Qubit. [Kreiptasi 14 05 2018].

[14] Wikipedia, „General number field sieve - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/General_number_field_sieve. [Kreiptasi 14 05 2018].

[15] Wikipedia, „Quantum Fourier transform - Wikipedia,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Quantum_Fourier_transform. [Kreiptasi 14 05 2018].

[16] W. M. „Discrete Logarithm -- from Wolfram MathWorld,“ [Tinkle]. Available: http://mathworld.wolfram.com/DiscreteLogarithm.html. [Kreiptasi 14 05 2018].

[17] Docker, „Docker - Build, Ship, and Run Any App, Anywhere,“ [Tinkle]. Available: https://www.docker.com/. [Kreiptasi 20 05 2018].

[18] Microsoft, „Azure Windows VM sizes - HPC | Microsoft Docs,“ [Tinkle]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-hpc. [Kreiptasi 20 05 2018].

[19] Microsoft, „Azure SQL Database DTU-based resource limits | Microsoft Docs,“ [Tinkle]. Available: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-dtu-resource-limits. [Kreiptasi 20 05 2018].

[20] Microsoft, „Azure Windows VM sizes - General purpose | Microsoft Docs,“ [Tinkle]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-general. [Kreiptasi 20 05 2018].