**Turinys**

[1. Įžanga 3](#_Toc71140805)

[1.1. Problema **Error! Bookmark not defined.**](#_Toc71140806)

[1.2. Suplanuotas pakeitimas (angl. Change request) 3](#_Toc71140807)

[2. Statinis modelis (angl. Static model) **Error! Bookmark not defined.**](#_Toc71140808)

[2.1. Žodynas **Error! Bookmark not defined.**](#_Toc71140809)

[2.2. Dalykinės srities modelis **Error! Bookmark not defined.**](#_Toc71140810)

[3. Panaudojimo atvejai (angl. Use Cases) **Error! Bookmark not defined.**](#_Toc71140811)

[3.1. Užduočių diagrama 10](#_Toc71140812)

[3.2. Bendravimo scenarijus **Error! Bookmark not defined.**](#_Toc71140813)

[3.3. Grafiniai modeliai **Error! Bookmark not defined.**](#_Toc71140814)

[3.4. Išbaigtumo diagramos **Error! Bookmark not defined.**](#_Toc71140815)

[4. Poveikio analizė (angl. Impact analysis) **Error! Bookmark not defined.**](#_Toc71140816)

[5. Atsekamumas (angl. Traceability) 11](#_Toc71140817)

[6. Rezultatai 12](#_Toc71140818)

[7. Išvados 12](#_Toc71140819)

# Įžanga

Šiame dokumente bus aprašomas sistemos architektūrinis stilius, naudojami architektūriniai šablonai, funckiniai ir nefunkciniai sistemos reikalavimai. Taip pat į sistemą bus pažvelgta iš skirtingų pjūvių (angl. Viewpoints) ir perspektyvų bei aprašytas pasirinktas sistemos testavimo būdas. Dokumento gale pagal aprašytus duomenis bus pateikiami rezultatai ir išvados.

## Kontekstas

Norint, kad sistema išliktų konkurencinga rinkoje ir būtų kuriama pagal nustatytus standartus, privaloma apibrėžti tinkamą architektūrinį stilių ir šablonus. Siekiant sistemą apsaugoti nuo įsilaužimų, užtikrinti sklandų ir efektyvų veikimą didelio vartotojų srauto metu, į ją reikia pažvelgti skirtingais pjūviais, apibendrinant perspektyvomis. Kad vystymo procesas būtų sklandus, sistemos saugykloje (angl. Repository) būtina įdiegti automatinę testavimo aplinką.

## Sistemos architektūra

Sistemos architektūra aprašo pagrindines sistemos struktūras ir jų kūrimo discipliną. Kiekvieną struktūrą sudaro programos elementai, jų tarpusavio ryšiai ir elementų bei jų santykių savybės.

### Architektūrinis stilius

Architektūrinis stilius – daugkartiniai dizaino sprendimai ir suvaržymai, taikomi architektūrai, norint pritaikyti pasirinktas pageidaujamas savybes. Kainų palyginimo svetainė ,,Smartshop” yra sukurta remiantis ,,Layered” ir ,,Data-centric” stiliais.

#### ,,Layered” stilius

,,Layered” stilius – keturių pakopų: pristatymo, verslo, atkaklumo ir duomenų bazės modelis, kur komponentai yra išdėstyti horizontaliais sluoksniais. Visi komponentai yra tarpusavyje susiję, tačiau nepriklauso vienas nuo kito. Kainų palyginimo svetainė „SmartShop“ naudoja šį šabloną, dėl lengvo sistemos funkcionalumo įgyvendinimo.

#### ,,Data-centric” stilius

,,Data-centric” stilius – stilius, orientuotas į duomenis, kai duomenys laikomi svarbiausiu sistemos objektu. Puikiai tinka kainų palyginimo svetainės „Smartshop“ idėjai, kur duomenys iš įvairių internetinių svetainių yra surenkami ir lyginami ir pateikiamas geriausias pagal kainą produktas. Taip pat, „Data-centric“ stilius vadovaujasi SSOT (angl. single source of truth) principu – sukuriamas vienas duomenų modelis, kurį naudoja visa sistema. Taip išvengiama duomenų dubliavimo, nes duomenys bus išsaugojami tik vieną kartą. „Smartshop“ sistema vadovaujasi šiuo principu naudodama vieną centrinę duomenų bazę visų duomenų saugojimui.

### Architektūrinis šablonas

Architektūrinis šablonas – pakartotinai naudojamas programos sandaros problemos sprendimas. Architektūriniai modeliai sprendžia įvairias programų inžinerijos problemas. Kainų palyginimo svetainė „SmartShop“ naudoja MVC (angl. Model-View-Controller) šabloną.

#### MVC

,,SmartShop“ sistemoje MVC šablonas naudojamas vartotojo sąsajų, kurios programos logiką sujungia į tris tarpusavyje susijusius elementus kūrimui. Šablonas pasirinktas dėl greito programos kūrimo ir atnaujinimo. Kadangi MVC modelis išskirsto programos veikimo logiką į tris tarpusavyje susijusius elementus, kiekvieną programos elementą galima kurti atskirai ir pasiskirstyti programos kūrimo darbus. Kita priežastis – atnaujinimai. Platformos gyvavimo laikas yra gana trumpas, o tai lemia neišvengiamus pokyčius ir būtinus atnaujinimus. MVC šablonas leidžia atlikti nedidelius pakeitimus nekeičiant viso sistemos modelio.

## Reikalavimai sistemai

Siekiant užtikrinti efektyvų ir tinkamą sistemos veikimą, būtina realizuoti atitinkamus funkcinius ir nefunkcinius reikalavimus.

### Funkciniai reikalavimai

* + 1. Programa iš bendradarbiaujančių e-parduotuvių turi surasti produktą pagal produkto pavadinimą.
       1. Vartotojas turi galimybę susiaurinti paiešką nurodydamas:
          1. kainos intervalą;
          2. parduotuves, kuriose produktas būtų ieškomas.
    2. Programoje registruotas vartotojas gali:
       1. įsiminti produktus;
       2. pamiršti produktus;
       3. parašyti atsiliepimą apie vieną e-parduotuve vieną kartą;
       4. pranešti programos valdytojams apie aptiktą klaidą.
    3. Rastas produktas pateikiamas lentelėje su:
       1. pardavėjo pavadinimu;
       2. produkto nuotrauka;
       3. produkto pavadinimu;
       4. kaina;
       5. mygtuku, leidžiančiu išsaugoti produktą.
    4. Programoje įvykusi klaida turi būti išsaugota duomenų bazėje.
    5. Programos administratorius turi galimybę:
       1. Peržiūrėti klaidos įrašą duomenų bazėje;
       2. Pašalinti klaidos įrašą duomenų bazėje;
       3. Sukurti naują vartotoją;
       4. Pašalinti vartotoją;
       5. Pakeisti vartotojo rolę;
       6. Bendrauti su vartotoju per pokalbių langą.
    6. Vartotojas turi galimybę tiesiogiai bendrauti su programos administratoriumi.
       1. Vartotojas susirašinėja su administratoriumi per dešinėje pusėje esantį dialogo langą;
       2. Vartotojas mato ir anksčiau parašytas žinutes;
       3. Bendrauti su programos administratoriumi gali ir registruoti, ir neregistruoti vartotojai;
    7. Programoje turi būti galimybė peržiūrėti konkretaus produkto kainos kitimo grafiką.
       1. Produktų, išsaugotų duomenų bazėje, kainos turi būti atnaujinamos kas 24 valandas;
       2. Produkto kainų grafikas turi būti vaizduojamas modaliniame lange;
       3. Grafike turi būti galimybė nustatyti norimą laiko intervalą. Minimalus laiko intervalas - 1 mėnesis.

### Nefunkciniai reikalavimai

* + 1. Saugumo reikalavimai:
       1. Užtikrinti registruotų vartotojų duomenų apsaugą ir privatumą;
       2. Užtikrinti programos administratoriaus paskyros saugumą;
       3. Laikytis BDAR[[1]](#footnote-1) (angl. GDPR) reikalavimų;
       4. Slaptažodžiai privalo būti šifruojami (angl. hashed);
    2. Patikimumo reikalavimai:
       1. Programos atnaujinimai turi būti tvarkingi ir neturi pakeisti ar kitaip trikdyti programos veikimo.
    3. Prieinamumo reikalavimai:
       1. Programa turi būti prieinama bet kokio tipo išmaniajam įrenginiui.
    4. Naudojimosi reikalavimai:
       1. Pokalbio (dialogo) langas turi būti matomas kiekviename puslapyje.
       2. Visi puslapiai turi tilpti į ekrano plotį. Nėra slankjuostės (angl. scroll bar).

# Pjūviai (angl. Viewpoints)

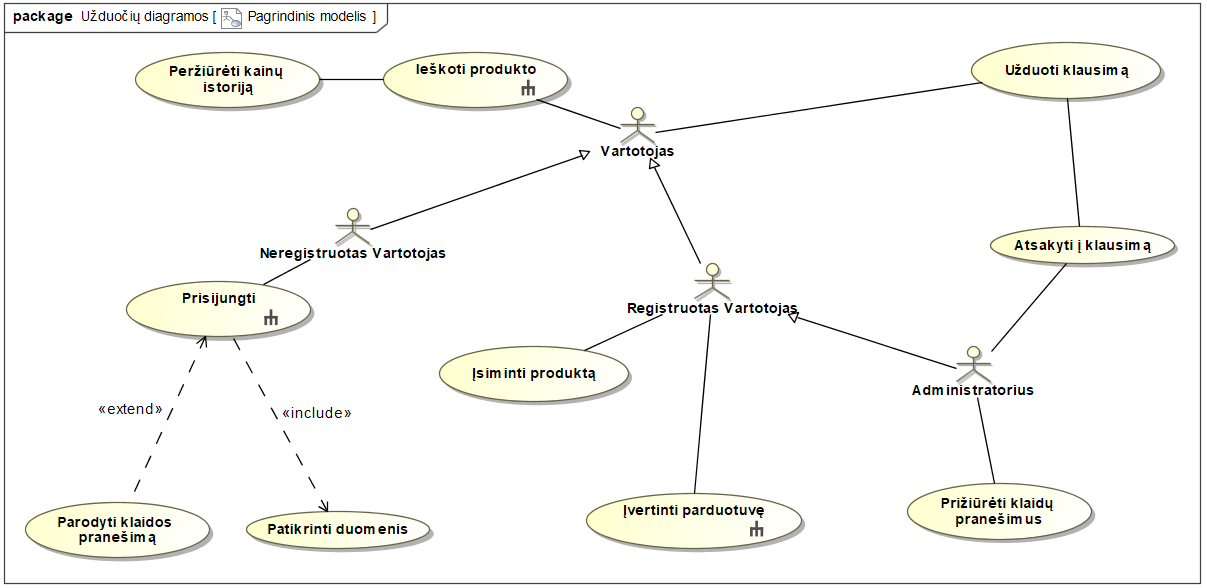
Pjūvis – tai būdas organizuoti reikalavimus programinės įrangos sistemai, remiantis tam tikra perspektyva, pavyzdžiui, vartotojo ar vadovo požiūriu. Pagrindinė į pjūvius orientuotos (angl. viewpoint-oriented) analizės stiprybė – pripažįstamas kelių skirtingų perspektyvų egzistavimas bei sudaromos tinkamos sąlygos konfliktų tarp suinteresuotųjų šalių keliamų reikalavimų išryškinimui. Pjūviai taip pat naudojami kaip būdas klasifikuoti įvairius suinteresuotųjų šalių tipus ir kitus reikalavimų šaltinius.

## Konteksto pjūvis (angl. Context Viewpoint)

Konteksto pjūvis apibrėžia santykius, priklausomybes ir sąveiką tarp sis­te­mos­ ir ­jos­ ap­lin­kos­ – žmonių, sistemų ir išorinių subjektų, su kuriais ji tarpusavyje sąveikauja. Juo remiantis, apibrėžiama, ką sistema daro ir ko nedaro, kur yra ribos tarp sistemos ir išorinio pasaulio, kaip sistema sąveikauja su kitomis sistemomis, organizacijomis ir žmonėmis peržengdama minėtas ribas.

### Užduočių diagrama

Užduočių diagrama – UML diagrama, vaizduojanti vartotojo ir skirtingų naudojimo atvejų, su kuriais yra susijęs šis vartotojas, santykį. Šioje diagramoje daugiausiai dėmesio skiriama būtent programos vartotojui, tad vaizduojami svarbiausi tikslai, kuriuos vartotojas gali pasiekti, naudodamasis sistema.



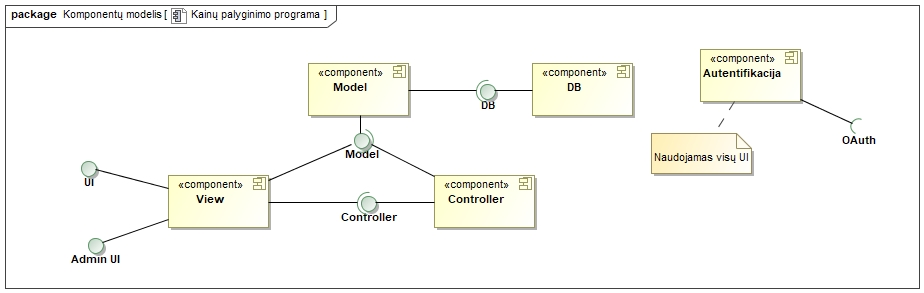
**1 Pav. Sistemos naudotojų galimybės**

,,SmartShop” sistemos naudotojų tikslus vaizduoja viena užduočių diagrama (1 Pav.). Galimi diagramos scenarijai:

* **Vartotojas** turi galimybę ieškoti produkto ir peržiūrėti jo kainos kitimo istoriją ir gali įsiminti produktą ir vieną kartą palikti atsiliepimą apie patikusią parduotuvę;
* **Administratorius** turi galimybę peržiūrėti sistemos klaidų pranešimus bei atsakinėti į užduotus vartotojo klausimus.

### Komponentų modelis (angl. Component diagram)

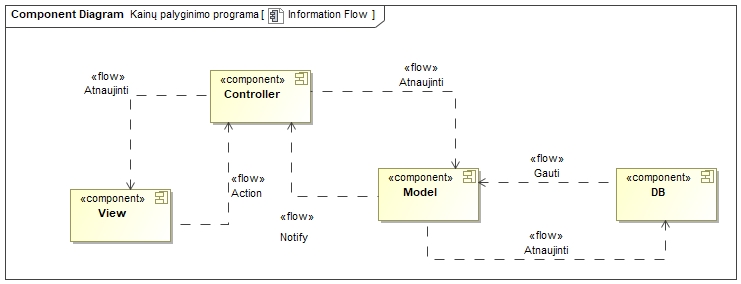
Komponentų modelis skirtas pagrindinių sistemos komponentų ir ryšio tarp jų vaizdavimui. Jame taip pat iliustruojama sistema iš programuotojo perspektyvos ir programinės įrangos valdymas. Sistemai sukurti pasitelkta plačiai naudojama MVC architektūra (2 Pav.).



**2 Pav. Kainų palyginimo sistemos komponentų modelis**

### Informacijos srautas (angl. Information flow diagram)

Siekiant dar aiškiau parodyti, kaip siunčiami duomenys tarp modelio, vaizdo (angl. View) ir kontrolerio, pateikiama informacijos srauto (angl. Information Flow) diagrama (3 Pav.).



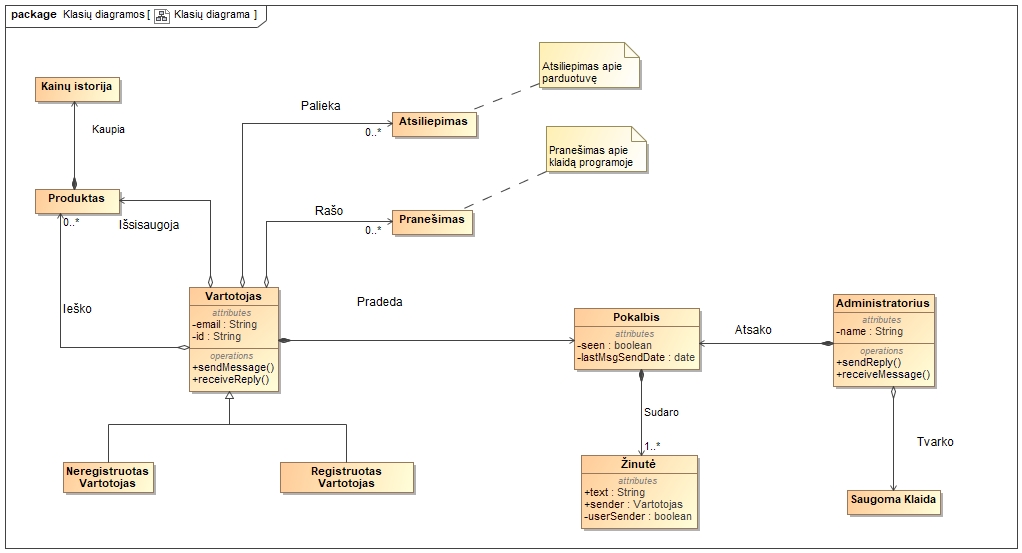
**3 Pav. Informacijos srautas sistemoje**

## Funkcinis pjūvis (angl. Functional Viewpoint)

Funkcinis sistemos pjūvis aprašo architektūrinius elementus, kurie yra atsakingi už sistemos funkcionalumą. Pjūvyje dokumentuojama sistemos funkcinė struktūra, įtraukiant pagrindinius funkcinius elementus, jų atsakomybę, sąsajas, kurias jie atskleidžia, ir sąveiką tarp jų. Kartu parodoma, kaip sistema atliks jai reikalingas funkcijas.

### Klasių diagrama

,,SmartShop“ sistemoje pagrindinė klasė yra produktas. Klasių diagrama vaizduoja pagrindinės produkto klasės ir kitų klasių, papildančios šios pagrindinės klasės funkcionalumą, sąryšius (4 Pav.).



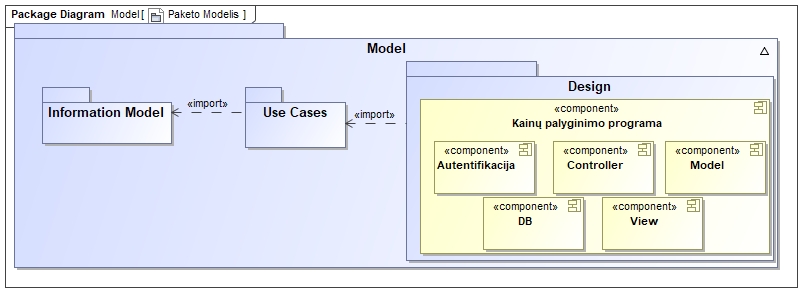
**4 Pav. Kainų palyginimo sistemos klasių diagrama**

## Realizacijos pjūvis (angl. Development Viewpoint)

Planuojant programinės įrangos projektavimą ir sukūrimą, reikia tvarkyti kūrimo aplinką. Būtina pagalvoti apie kodo struktūrą ir priklausomybes, sudedamųjų dalių sukūrimą ir konfigūravimą, visos sistemos projektavimo apribojimus ir visos sistemos standartus, kad būtų užtikrintas techninis vientisumas. Realizacijos pjūvis atsakingas už šiuos sistemos kūrimo proceso aspektus. Pjūvyje dėmesys yra sutelkiamas į architektūriškai reikšmingus klausimus, problemas ir ypatybes.

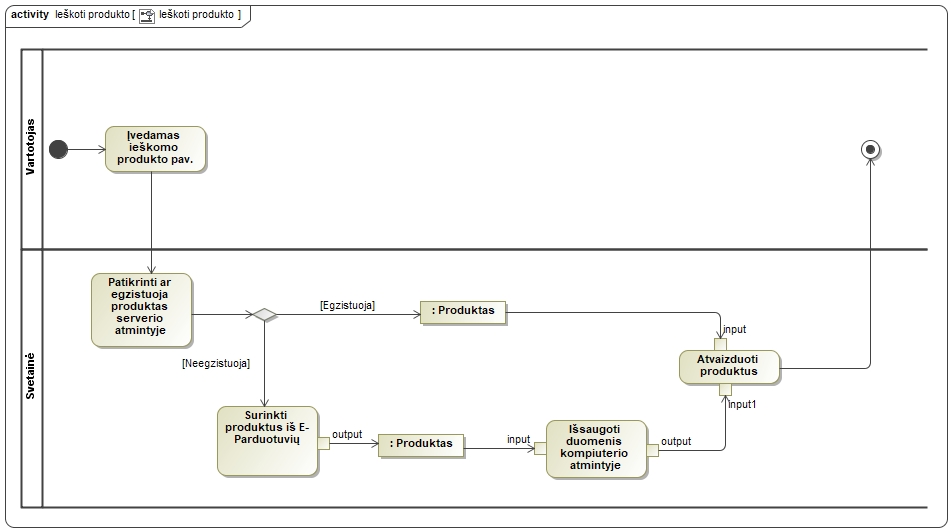
### Paketo modelis (angl. Package diagram)

Paketo modelio diagrama skirta pavaizduoti informacijos modelio, užduočių diagramų ir programos dizaino sąryšį (5 Pav.). Programos dizaino paketas yra išskaidytas į penkis atskirus komponentus, atsakingus už vartotojo autentifikaciją, duomenų bazę ir programos puslapį.

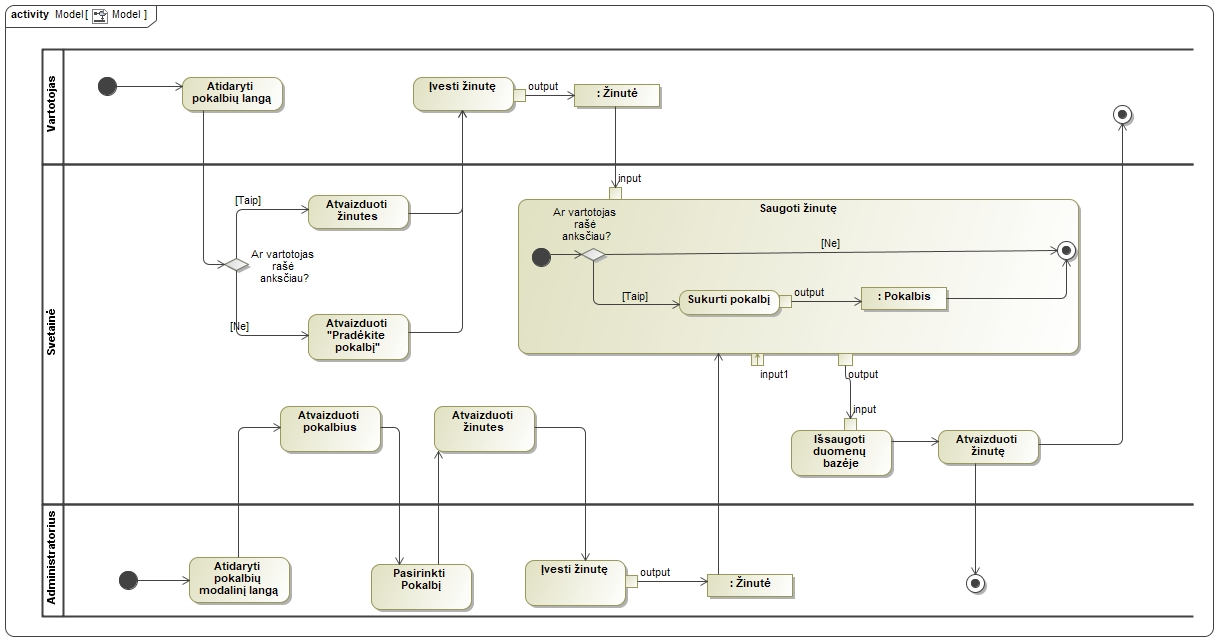


**5 Pav. Sistemos paketo modelis**

Vartotojas prie sistemos prisijungia suvesdamas savo paskyros duomenis (elektroninį paštą ir slaptažodį) (3 Pav.). Jei įvesti duomenys teisingi – išsiunčiamas JWT[[2]](#footnote-2) (JSON web token), jeigu ne – siunčiamas klaidos pranešimas. Toliau vaizduojamas produkto paieškos procesas (6 Pav.). Vartotojui įvedus ieškomo produkto pavadinimą, prašymas nukeliauja į ,,SmartShop“ aplikaciją kurioje visų pirma patikrinama, ar ieškomas produktas jau yra išsaugotas serverio atmintyje. Jei taip, tuomet jis paprasčiausiai pateikiamas vartotojui. Jeigu produktas yra nerastas, produkto ieškoma bendradarbiaujančiuose e-parduotuvėse, kuris bus išsaugoti kompiuterio atmintyje ir taip pat pateiktas vartotojui.



**6 Pav. Produkto paieškos procesas**

Toliau pateikiamas galimas pokalbio tarp vartotojo ir administratoriaus scenarijus (7 Pav.). Matoma, jog pokalbis yra visada inicijuojamas vartotojo. Vartotojas siunčia pranešimą ir per ,,SmartShop“ sistemą pranešimas yra persiunčiamas administratoriui, kuris realiu laiku gali atsakyti vartotojui.

**7 Pav. Pokalbio scenarijus**

# Perspektyvos

Architektūrinė perspektyva – veiklos ir gairių rinkinys, naudojamas siekiant užtikrinti, sistema pasižymi tam tikru susijusių kokybės savybių rinkiniu, į kurį reikia atsižvelgti iš daugelio sistemos architektūrinių vaizdų. Kainų palyginimo platformoje „SmartShop“ įžvelgiamos keturios pagrindinės perspektyvos: saugumo, našumo ir plėtimo, prieinamumo ir atsparumo, vystymo.

## Saugumo perspektyva

Konteksto pjūvis. Saugumas šiandieniniame pasaulyje yra vienas iš svarbiausių programos aspektų. Kainų palyginimo platforma „SmartShop“ turi būti pasirengusi apsaugoti vartotojų ir programos duomenis. Didžiausios grėsmės „SmartShop“ sistemai – duomenų vagystė (įmonės „CityBee“ vartotojų nutekinimo atvejis) arba SQL injekcijos.

Funkcinis pjūvis. Sistemos jautriausia vieta yra centrinė duomenų bazė, kurioje saugomi visi duomenys: produkto paieškos, vartotojų asmeniai duomenys (elektroniai paštas ir slaptažodis) ir kt. Pašaliniam asmeniui įsibrovus į duomenų bazę, būtų nutekinti visi turimi duomenys. Kita problema gali būti vartotojo autentifikavimas. Vartotojui prisijungus, slapukuose išsaugomas tik jo elektroninis paštas, naudojamas pakartotinei autentifikacijai (vartotojui vėliau jungiantis prie sistemos). Paskyras, autentifkuojamas tik per elektroninį paštą nulaužti yra labai lengva.

Realizacijos pjūvis. Turi būti aiškios gairės, pagal kokius saugumo standartus turi būti apsaugota sistema. „SmartShop“ naudojasi OWASP[[3]](#footnote-3) (angl. open web application security project) metodika.

## Našumo ir plėtojimosi perspektyva

Konteksto pjūvis. Vieni iš svarbiausių kainų palyginimo svetainių prioritetų: dalyvaujančių parduotuvių skaičius, paieškos greitis ir vartotojo potyris. Šie veiksniai užtikrina sklandžią plėtrą. PAPILDYTI

Funkcinis pjūvis. Vienas iš būdų įgyvendinti greitesnį paieškos algoritmą reikalautų dviejų duomenų bazių sukūrimo. Vienoje būtų saugomi vartotojų duomenys, o kitoje produkto paieškos reikalavimai. Kad greičiau vykdoma, galima produktus parse‘inti paraleliai. Taip pat, reikia pakeisti esamą produkto internetinėse parduotuvės paiešką, nes ji priklauso nuo internetinės parduotuvės elementų išdėstymo ir tariama, jog jis nesikeis. Parduotuvei pakeitus savo internetinio puslapio dizainą (pavyzdžiui informacijos apie produktus išdėstymą), produkto kaina nebus nuskaitoma.

Realizacijos pjūvis. Norint padidinti platformos vystymosi galimybes, reikia peržvelgti turimą kodą ir jį optimizuoti: peržiūrėti dabartinį paieškos algoritmą ir jį padaryti spartesnį, peržvelgti platformos turinį, padaryti duombazes užklausas efektyvesnes – sukurti prekių indeksavimą, kad prekių paieška būtų greitesnė.

## Prieinamumo ir atsparumo perspektyva

Konteksto pjūvis. „SmartShop“, kaip ir kitos kainų palyginimo platformos yra priklausomos nuo bendradarbiavimo su internetinėmis parduotuvėmis. Atsisakius bendradarbiauti nors vienai iš jų, sumažėja produkto pasirinkimo galimybės ir suprastėja paieškos rezultatai. Kitas išorinis veiksnys – duomenų bazė. Jai sutrikus, nėra jokios atsarginės kopijos, kuri galėtų laikinai pakeisti centrinę duomenų bazę, tad vartotojai negalės naudotis platforma. Taip pat, nėra žinomas sistemos elgesys didelių apkrovų metu (pavyzdžiui vienu metu prisijungusių didelio kiekio vartotojų).

Funkcinis pjūvis. Siekiant padidinti „SmartShop“ sistemos atsparumą ir prieinamumą reikia susimąstyti apie eilės struktūrą: sulaukus didelio kiekio užklausų vienu metu, vartotojus dėti į eiles pagal jų prisijungimo laiką ir leisti prisijungti naujiems vartotojams, vos tik atsiranda laisvos vietos. Toks funkcionalumas reikalautų investicijų, bet padidintų platfromos prieinamumą vartotojams. Taip pat, numatytu laiku reikia nustatyti mėnesinį platformos išjungimą, kad būtų atlikti techniniai darbai. Galimas vartotojo neveiksnumo limitas: vartotojui kurį laiką (pavyždžiui, 5 minutes) sistemoje neatliekant jokių veiksmų, būtų išsiunčiama užklausa, dėl veiklos pratęsimo. Tokiu būdu būtų taupomi sistemos resursai.

Realizacijos pjūvis. Norint užtikrinti atsparumą nuo išorinių veiksnių, „SmartShop“ platforma turėtų ateityje svarstyti papildomos duomenų bazės įsigijimą, kurioje būtų saugomos atsarginės kopijos vartotojų duomenų, jeigu pagrindinėje duomenų bazėje įvyktų gedimas arba laikinai iki kol gedimas bus pašalintas.

## Vystymosi perspektyva

Konteksto pjūvis. Dėl trumpo gyvavimo laiko, „SmartShop“ platforma neišvengiamais turės pereiti keletą atnaujinimų ciklų. Vienas iš tokių galimų PAPILDYTI

Realizacijos pjūvis. Kainų palyginimo platformos „SmartShop“ turi keletą alternatyvių krypčių. Viena iš jų – įsitvirtinti kaip pagrindinė kainų palyginimo platforma kokioje Lietuvos nišinėje, pavyzdžiui, elektroninių prekių rinkoje. Kitas būdas padidinti savo turimų elektroninių parduotuvių kiekį ir produktų rūšį taip stengiantis pasiekti kuo platesnį vartotojų ratą. Trečias kelias būtų lyginti Lietuvos ir kitų šalių produktų kainas. Taip būtų siekiama vartotojui parodyti kainų skirtumus įvairiose šalyse.

# Testavimas (angl. Testability)

Siekiant užtikrinti, kad sistema veiktų taip, kaip tikimasi, ir pagerinti bendrą jos kokybę, buvo įdiegta automatizuoto testavimo sistema, naudojanti modulių ir integravimo testus (angl. Unit and integration tests). Automatizuoti testai leidžia nedelsiant aptinka sistemos kūrimo metu nepastebėtas klaidas. Taip sutaupoma brangaus laiko, kuris būtų skiriamas pakeitimus testuoti rankiniu būdu.

## Testavimo aplinka

Sistemos testavimui yra pasitelktas „GitLab Runner” įrankis. Keliant atnaujintus duomenis (kodą) į pagrindinę GIT saugyklos šaką, bus vykdomos instrukcijos, kurios paleidžia sistemos aplikaciją, veikiančią pagal ką tik atnaujintą programos kodą. Aplikacija veikia virtualioje mašinoje, kuri yra patalpinta Vilniaus Universiteto debesų kompiuterijos platformoje. Dabartinę ,,SmartShop“ sistemą galima pasiekti adresu <http://193.219.91.103:12808/>.

## Testavimo įrankiai

Testavimo sistemą sukūrta naudojant xUnit projekto šabloną. Duomenų, kontrolerių bei metodų testavimui papildomai į projektą įdiegta Moq sistema (angl. Moq framework), šiuo metu viena iš plačiausiai naudojamų testavimo sistemų. Duomenų kaupimui leidžiant testus pasirinkta naudoti vadinamoji duomenų bazė atmintyje (angl. In-memory database) – duomenų bazės valdymo sistema, kuri pirmiausia naudoja pagrindinę kompiuterio atmintį duomenims saugoti.

## Automatiniai testai

Kuriant testus, pagrindinis prioritetas buvo nustatyti sistemos dalis, kurias svarbiausia ištestuoti per trumpą laiką, nes nebuvo pakankamai išteklių, norint testavimą įdiegti visai sistemai. Nuspręsta, kad šio darbo metu testai bus rašomi tik atliekamam programos pakeitimui, t.y, pokalbių sistemos funkcionalumui. Vėliau testai turėtų apimti anksčiau įdiegtas sistemos funkcijas.

# Atsekamumas (angl. Traceability)

Buvo sukurta atsekamumo matrica, skirta užtikrinti, kad visi funkciniai reikalavimai yra apžvelgiami konteksto, funkciniame ir realizacijos pjūviuose (Lentelė 1.).

**Lentelė 1. Funkcinių reikalavimų ir pjūvių atsekamumo matrica**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pjūvis** | **Diagrama** | FR1 | FR2 | FR3 | FR4 | FR5 | FR6 | FR7 |
| **Konteksto** | *Užduočių diagrama* | ***X*** | ***X*** |  |  | ***X*** | ***X*** | ***X*** |
| *Komponentų modelis* | ***X*** |  |  |  |  |  |  |
| *Informacijos srautas* | ***X*** |  |  |  |  |  |  |
| **Funkcinis** | *Klasių diagrama* | ***X*** |  | ***X*** | ***X*** |  | ***X*** | ***X*** |
| **Realizacijos** | *Veiklos diagramos* |  |  | ***X*** |  |  | ***X*** |  |
| *Paketo modelis* | ***X*** |  |  |  |  |  |  |

# Rezultatai

Apibendrinant, buvo pasiekti šie darbo rezultatai:

1. Aprašyti architektūriniai stiliai ir šablonai;
2. Aprašytos architektūrines perspektyvos. Perspektyvos aptartos per keturis svarbiausius programos punktus: saugumo, našumo ir atsparumo, prieinamumo ir atsparumo bei vystymosi;
3. Aprašyta konteksto, funkciniai ir vystymosi pjūviai;
4. Aprašyta programos testavimo būdas, eiga;
5. Sukurti modulių testai;
6. Sukurta atsekamumo KOKIA matrica;
7. Įgyvendintas užsakovų reikalavimas: vartotojas realiu laiku bendraujaua su programos administratoriumi per pokalbių langą Dėl pokalbių lango.

# Išvados

Remiantis darbo, kuriame buvo ... , galima daryti išvadas, kad:

1. Sistema nuo šiol pritaikyta testavimui;
2. Sistemos kūrimas vadovaujasi tam tikrais architektūriniais stiliais;

Apibendrinant galima teigti, kad ,,SmartShop‘‘ sistema pasiruošusi realeas‘ui SUTVARKYTI KĄ

1. Bendrasis duomenų apsaugos (sutrumpintai BDAR) reglamentas pateikia išsamius įmonėms ir organizacijoms taikomus asmens duomenų rinkimo, saugojimo ir valdymo reikalavimus. [↑](#footnote-ref-1)
2. JSON žiniatinklio ženklas (angl. JSON Web Token) – tai interneto siūlomas duomenų kūrimo (saugojimo) su pasirinktu parašu ir (arba) šifravimu standartas. [↑](#footnote-ref-2)
3. OWASP - tarptautinė ne pelno siekianti organizacija, skirta interneto programų saugumui. [↑](#footnote-ref-3)