

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Andrej Latyš

**BENDRŲ TIKSLŲ ĮGYVENDINIMO BEI DALINIMOSI MOBILI PROGRAMĖLĖ**

**MOBILE APPLICATION FOR TEAM CHALLENGE AND SHARING**

Baigiamasis bakalauro darbas

Informacinių sistemų studijų programa, valstybinis kodas 612I20003

Informacijos sistemų studijų kryptis

Vilnius, 2022

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU

Katedros vedėjas

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Parašas)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Vardas, pavardė)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Data)

Andrej Latyš

**BENDRŲ TIKSLŲ ĮGYVENDINIMO BEI DALINIMOSI MOBILI PROGRAMĖLĖ**

**MOBILE APPLICATION FOR TEAM CHALLENGE AND SHARING**

Baigiamasis bakalauro darbas

Programų inžinerijos studijų programa, valstybinis kodas 612I30003

Programų sistemų studijų kryptis

**Vadovas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

**Konsultantas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

**Konsultantas**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

Vilnius, 2019

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Fundamentinių mokslų FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU

..........…....……………....studijų kryptis

.................………...…......studijų programa, valstybinis kodas ………………………..…specializacija

Katedros vedėjas

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Parašas)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Vardas, pavardė)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Data)

**BAIGIAMOJO BAKALAURO DARBO (PROJEKTO)**

**UŽDUOTIS**

………........................Nr. ...............

Vilnius

Vardenis Pavardenis

Studentui (ei)...............…………................................….........................................................................…........…....  
 (Vardas, pavardė)

Baigiamojo darbo (projekto) tema: ......….........................................................................................................

......................................................................................................................................................................

patvirtinta 201…m. ………………..…… d. dekano potvarkiu Nr. ………….

Baigiamojo darbo (projekto) užbaigimo terminas 2019 m. birželio 1 d.

BAIGIAMOJO DARBO (PROJEKTO) UŽDUOTIS:

Duomenys: .................................................................................................................................……............................

..............................................................................................................................................................……...............

...................................................................................................................................................................……..........

Aiškinamasis raštas: ..................................................................................................................................…….............

.......................................................................................................................................................................……......

...................................................................................................................................................................……..........  
...............................................................................................................................................................……..............

Baigiamojo bakalauro darbo (projekto) konsultantai: …………………………………………………….…………..

................................................................................................................................................................…….............  
 (Pareigos, vardas, pavardė)

Vadovas ................................ ..........................................................  
 (Parašas) (Moksl. laipsnis, vardas, pavardė)

Užduotį gavau

…………………………………..

(Parašas)

..................................................

(Vardas, pavardė)

.......................................................

(Data)

**Turinys**

[Įvadas 8](#_Toc102735523)

[1. Analitinė dalis 11](#_Toc102735524)

[1.1. Idėjos pagrindimas 11](#_Toc102735525)

[1.2. Alternatyvos 12](#_Toc102735526)

[1.3. Kuriamos programėlės techninė analizė 15](#_Toc102735527)

[1.3.1. Mobiliųjų aplikacijų kūrimo karkaso pasirinkimas 15](#_Toc102735528)

[1.3.2. Išsami „Flutter“ technologijos apžvalga 19](#_Toc102735529)

[1.3.3. Vidinies programos dalies technologiju pasirinkimas 21](#_Toc102735530)

[1.3.4. Išsami vidinies dalies apžvalga 23](#_Toc102735531)

[2. Projektinė dalis 26](#_Toc102735532)

[2.1. Sistemos naudotojai 26](#_Toc102735533)

[2.2. Panaudos atvejų diagrama 26](#_Toc102735534)

[2.3. Funkciniai ir nefunkciniai reikalavimai kuriamai PS 36](#_Toc102735535)

[3. Sistemos architektūra 38](#_Toc102735536)

[3.1. Sistemos serverio architektūra 38](#_Toc102735537)

[3.2. Sistemos įgyvendinimo architektūra 39](#_Toc102735538)

[3.3. Duomenų bazės struktūra ir ryšiai 40](#_Toc102735539)

[4. Testavimas 41](#_Toc102735540)

[4.1. Testavimo metodikos pasirinkimas 41](#_Toc102735541)

[4.2. Automatizuotas testavimas 42](#_Toc102735542)

[4.2.1. Modulinis testavimas 42](#_Toc102735543)

[4.2.2. Integraciniai testai 44](#_Toc102735544)

[4.3. Rankinis testavimas 46](#_Toc102735545)

[4.3.1. End-to-end testavimas 46](#_Toc102735546)

[4.4. Testavimo išvados 46](#_Toc102735547)

[Išvados ir siūlymai 48](#_Toc102735548)

[6. Literatūros sąrašas 50](#_Toc102735549)

**Iliustracijų sąrašas**

[1 pav. Mobiliuju programeliu naudojamumas pagal kategorijas (Statista, 2021) 10](#_Toc102734918)

[2 pav. Programėlės “Challenges” vartotojo sąsaja 13](#_Toc102734919)

[3 pav. Programėlės „Microsoft To Do“ vartotojo sąsaja 13](#_Toc102734920)

[4 pav. Programėlės „Habitica“ vartotojo sąsaja 13](#_Toc102734921)

[5 pav. Populiariausi hibridiniu ir web mobiliųjų programėlių sukūrimo karkasai (Statista, 2021). 15](#_Toc102734922)

[6 pav. „Flutter“ architekturos modelis 19](#_Toc102734923)

[7 pav. „Flutter“ sąveika su „native“ komponentais 19](#_Toc102734924)

[8 pav. Spring karkaso architekturinis modelis 22](#_Toc102734925)

[9 pav. Panaudos atvejų diagrama 26](#_Toc102734926)

[10 pav. „Užregistruoti paskyrą“ užduoties modelis 27](#_Toc102734927)

[11 pav. „Prisijungti arba išeiti iš grupės“ užduoties modelis 27](#_Toc102734928)

[12 pav. „Prisijungti arba išeiti iš iššūkio“ užduoties modelis 28](#_Toc102734929)

[13 pav. „Kurti įrašus“ užduoties modelis 29](#_Toc102734930)

[14 pav. „Palikti komentarą“ užduoties modelis 30](#_Toc102734931)

[15 pav. „Peržiūrėti iššūkius“ užduoties modelis 30](#_Toc102734932)

[16 pav. „Kurti įrašus grupėje“ užduoties modelis 31](#_Toc102734933)

[17 pav. „Valdyti grupės narius“ užduoties modelis 32](#_Toc102734934)

[18 pav. „Tvarkyti įrašus ir komentarus“ užduoties modelis 33](#_Toc102734935)

[19 pav. „Kurti iššūkį“ užduoties modelis 33](#_Toc102734936)

[20 pav. „Uždaryti iššūkį“ užduoties modelis 34](#_Toc102734937)

[21 pav. Sistemos serverio komponentai 38](#_Toc102734938)

[22 pav. Sistemos įgyvendinimo architektūra 39](#_Toc102734939)

[23 pav. Testavimo piramidė 41](#_Toc102734940)

**Lentelių sąrašas**

[1 lentelė. Alternatyvių mobiliųjų programų palyginimas 13](#_Toc102734995)

[2 lentelė. Alternatyvų palyginimas pagal iškeltus sisteminius reikalavimus 15](#_Toc102734996)

[3 lentelė. Populiarių žiniatinklio programėlių kūrimo karkasų palyginimas 17](#_Toc102734997)

[4 lentelė. Populiarių hibridinių programėlių kūrimo karkasų palyginimas 18](#_Toc102734998)

[5 lentelė. Vidiniu sistemu karkasų palyginimas 22](#_Toc102734999)

[6 lentelė. Naudotojo „Neprisijungęs naudotojas“ detalizavimas 26](#_Toc102735000)

[7 lentelė. Naudotojo „Vartotojas“ detalizavimas 26](#_Toc102735001)

[8 lentelė. Naudotojo „Moderatorius“ detalizavimas 26](#_Toc102735002)

[9 lentelė. Naudotojo „Grupės kūrėjas“ detalizavimas 26](#_Toc102735003)

[10 lentelė Funkciniai reikalavimai 36](#_Toc102735004)

[11 lentelė Nefunkciniai reikalavimai 37](#_Toc102735005)

[12 lentelė. Modulinis testas №1 42](#_Toc102735006)

[13 lentelė. Modulinis testas №2 43](#_Toc102735007)

[14 lentelė. Modulinis testas №3 44](#_Toc102735008)

[15 lentelė. Integracinis testas №1 44](#_Toc102735009)

[16 lentelė. Integracinis testas №2 45](#_Toc102735010)

**Santrumpos**

SQL – Structured Query Language – struktūrinių užklausų kalba

GPU – Graphics processing unit – grafikos apdorojimo blokas

SDK – Software development kit – programinės įrangos kūrimo rinkinys

FPS – Frames per second – kadrai per sekundę

ARM – Advanced RISC machines – pažangios RISC mašinos

API – Application programming interface – taikomųjų programų programavimo sąsaja

XML – Extensible markup language – išplečiama žymėjimo kalba

HTTP – HyperText Transfer Protocol – hiperteksto perdavimo protokolas

SMTP – Simple Mail Transfer Protocol – paprastas el-pašto siuntimo protokolas

JSM – Jabber session manager – komunikavimo seanso vadovas

TCP – Transmission control protocol – duomenų perdavimo valdymo protokolas

XSS – Cross-site scripting

POJO – Plain old Java object – paprastas senas java objektas

# Įvadas

**Darbo aktualumas**

Kuriamos sistemos dalykine sritis yra labai plati, pagrindinis jos privalumas yra teminių apribojimų nebuvimas. Sistema turės socialinių tinklų charakteristikas, užduočių vykdymo ir vizualizacijos galimybė, turės motyvacinį poveikį naudojant žaidimifikacijos ir viešosios atskaitomybes principus. Geriausiai ši programa apibūdina kategoriją "gyvenimo būdas", kuri užima 5% mobiliųjų programėlių rinkos „Google Play“ platformoje (Statista, 2021).

Socialinių tinklų idėja yra viešai priimta ir plačiai naudojama. Statistikos portalo „Statista“ 2021 metų sausio duomenimis, apie 53% pasaulio gyventojų naudojasi socialiniais tinklais, iš jų 89% socialiniais tinklais naudojasi per išmanųjį telefoną (Statista, 2021). Nors bendravimas nėra pagrindinis kuriamos sistemos tikslas, naudotojams bus suteikta galimybė bendrauti, viešai ir grupėse, kad jiems nereikėtų ieškoti išorinių komunikacijos būdų.

JAV statistika rodo, kad daugiau nei dvi dienas per savaitę nuotoliniu būdu dirbančių žmonių skaičius, lyginant su 2019 m., išaugo beveik dvigubai (Statista, 2020). Daug kas suprato kad jiems tai yra saugiau ir patogiau, dabar beveik visos kompanijos suteikia galimybę dirbti nuotoliniu būdu, jei tai yra įmanoma. Tyrimai rodo, kad žmonės, dirbantys namuose, yra laimingesni, nes patyria mažiau streso. Tyrimas, kurį atliko 2020m. „FlexJob“, rodo, kad 65% darbuotojų mano, jog jie būtų produktyvesni, jeigu dirbtų iš namų, o ne biure (FlexJobs, 2020). Išpopuliarėjęs darbas nuotoliniu būdu sukuria paklausą sistemoms, kurias gali stebėti ir vizualizuoti esamą darbo eigą. Didžiausią rinkos dalį užima programuotojų įdiegtos “AGILE” ir “SCRUM” technologijos tokiose platformose kaip “Jira”, “Trello”, “ClickUp” ir daugelis kitų populiarių servisų. Kompanijos „VersionOne“ ataskaitoje pateikiama informacija apie tai, kad 98% tyrime dalyvavusių įmonių jau naudoja „Agile“ metodus (VersionOne, 2021).

Izoliacija namuose kaip papildoma saugumo priemonė taip pat reiškia, kad žmonės dažniau leidžia laisvalaikį namuose. Kaip rodo 2019–2021 m. izoliacijos per COVID-19 tyrimas rodo, kad izoliacijos metu pablogėja bendra žmogaus savijauta (E. Colucci, 2022). Net ir izoliacijoje žmonės stengiasi išlikti motyvuoti ir didinti savo produktyvumą nuolat ieškodami naujų būdų tapti sveikesniems. Tam žmonės dažnai naudoja sporto ir sveikos mitybos mobiliąsias programėles. Mano analizė parodė, kad bendradarbiavimo servisų neprograminiams ir nesportiniams tikslams trūksta. Atsižvelgiant į tokius duomenis, pastebėjau kad atsirado poreikis pramogas perkelti į nuotolį. Kuriama sistema neturės teminių apribojimų ir suteiks naudotojams įrankius ir galimybę siekti bendrų tikslų, dalyvauti socialiniuose iššūkiuose ir dalintis patirtimi.

**Darbo objektas –** programinės įrangos kūrimo technologijos ir praktikos bei jų pritaikymas mobiliųjų programėlių skūrimui.

**Darbo tikslas** – Sukurtį bendrų tikslų siekimo ir rezultatų dalijimosi sistemą mobiliesiems įrenginiams.

**Darbo uždaviniai:**

* 1. Išanalizuoti poreikį naudoti tikslų sekimo ir rezultatų dalijimosi sistemas.
  2. Išanalizuoti rinkos siūlomas alternatyvas.
  3. Išnagrinėti mobiliųjų programėlių kūrimo technologijas.
  4. Suformuluoti funkcinius ir nefunkcinius reikalavimus.
  5. Suprojektuoti sistemą atsižvelgiant į keliamus reikalavimus ir realizuoti ją programiškai.
  6. Atlikti sistemos testavimą ir pateikti išvadas.

**Temos pavadinimas -** Bendrų tikslų įgyvendinimo bei dalinimosi mobili programėlė.

**Darbo naujumas bei praktinė vertė:**

Tyrimo objektas nėra visiškai naujas, panašios programos jau yra sukurtos. Tačiau alternatyvos dažniausiai turi teminį apribojimą, o kuriama sistema gali būti pritaikyta įvairiausioms sritims. Tai gali būti draugiški iššūkiai, labdara, savanorystė, sporto renginiai, meno projektai ir kt. Sistema leis žmonėms vienytis ir siekti bendrų tikslų greičiau ir smagiau.

**Darbo struktūra**

Bakalauro baigiamąjį darbą sudaro x skyrių, išvados ir x priedų. Darbo apimtis – x puslapių, neskaitant priedų, tekste yra x numeruotų formulių, x paveikslų ir x lentelių. Rengiant darbą buvo panaudotas x literatūros šaltinis.

Įvade nagrinėjamas darbo aktualumas, tyrimų objektas, formuluojamas darbo tikslas bei uždaviniai.

Analitinėje dalyje nagrinėjama: idėjos pagrindimas, alternatyvos, kuriamos programėlės techninė analizė, mobiliųjų aplikacijų kūrimo karkaso pasirinkimas, išsami “Flutter” technologijos apžvalga, vidinies programos dalies technologijų pasirinkimas, išsamia vidinies dalies apžvalga.

Projektinėje dalyje atprašyti sistemos naudotojai, pateikiama panaudos atvejų diagrama, suformuluoti funkciniai ir nefunkciniai reikalavimai kuriamai sistemai, pavaizduota sistemos architektūrą, serverio architektūrą ir duomenų bazės struktūra ir ryšiai.

Testavimo dalyje atlikta 77 automatinių testų „Spring Boot“ aplinkoje, bei atlikta XX testų rankinių būdų.

Darbo įgyvendinimui panaudota lokali programavimo aplinka, „IntelJ IDEA“ programavimo platforma„ Flutter“ karkasas, „Java“ ir „Dart“ programavimo kalbos, „JUnit“ testavimo karkasas.

Bakalauro baigiamo darbo įgyvendinimo programinis kodas pasiekiami adresu: URL

# Analitinė dalis

## Idėjos pagrindimas

Išmaniųjų telefonų paklausos augimo tendencija išlaikoma ilgą laiką, ekspertai teigia, kad rinka auga ir tokia išliks ateinančiais metais. „Gartner“ ataskaita rodo, kad išmaniųjų telefonų pardavimas visame pasaulyje 2021 m. išaugo 10,8% lyginant su 2020m (Gartner, 2021). Mes naudojam jį bendravimui su kitais žmonėmis, pramogoms, informacijos perdavimui it t.t. Kiekvienas žmogus gali sugalvoti ir sukurti mobiliąją programėlę, kuri bus viešai platinama įvairiose platformose. Tai leidžia integruoti programėles į beveik visus mūsų gyvenimo aspektus, tik reikia suprasti, kiek tai naudinga ir sąmoninga. Remiantis nauja „App Annie State of Mobile 2022“ ataskaita, programų ir mobiliųjų programėlių naudojimas labai išaugo visame pasaulyje. Ataskaita rodo, kad 2021 m. buvo atsiųsta daugiau nei 232 milijardų programėlių ir 3,8 trilijono valandų buvo praleista mobiliesiems programėlėms. Tyrime analizuotų rinkų vidutinis laikas, praleistas programėlėse, 2021 m. viršijo 4 valandas ir 48 minutes – 30 % daugiau nei 2019 m. (AppAnnie, 2022).

Mobiliųjų programų kūrimo sričių yra daug, tarp jų yra mus motyvuojančios programėles. Norint pasiekti tikslą, būtina motyvacija. Juk būtent ji padeda eiti savo svajonės link ir suteikia impulsą net tomis akimirkomis, kai energija ir jėgų išsenka. Pradėti naują darbą, projektą ar įprotį yra sunku ir reikia įdėti daug pastangų . Tačiau yra daug metodų kaip lengviau tai padaryti, vienas iš tokiu - progreso sekimas (Fokkens, 2021) ir žaidimifikacija (Encarnação, Reuter, Dias, & Amorim, 2021). Galima rasti daug tokių programėlių, kurie padeda pasiekti tikslus arba formuoti sveikus įpročius. 2021 sausio mėnesio apklausa parodė, kad apie 30% (1 pav.) visų respondentų, amžiaus nuo 16 iki 64 metų, naudoja programėlės sveikatai, sportui ir mitybai (Statista, 2021).

Chart, bar chart

Description automatically generated

1 pav. Mobiliuju programeliu naudojamumas pagal kategorijas (Statista, 2021)

Kuriamo darbo idėja yra grupinių iššūkių sukūrimas su konkurenciniu aspektu. Žmonės galės vykdyti iššūkius, dalintis patirtimi ir nuotraukomis, taip pat gaunant už tai taškus. Tai padės jiems susipažinti su naujais žmonėmis, pradėti sveikiau valgyti, sportuoti, dažniau būti lauke ir tiesiog linksmintis, kas padės palaikyti gerą psichikos sveikatą (Stock, 2021). Konkurencija yra svarbi šios sistemos dalis, kaip rodo Iuliana-Tania Zlatcu ir Roxana Clodnitchi tyrimas, konkurencija gali padidinti tiek asmens, tiek įmonės produktyvumą (Iuliana-Tania ZLATCU & CLODNITCHI, 2018). Tarp kitų motyvacijos technikų yra socialinė atsakomybė ir žaidimifikacija. Socialinės atsakomybės principas – viešas savo tikslo skelbimas, sukuriant netiesioginę atsakomybę už savo tikslo pasiekimą (Clear, 2018). Taip pat žaidimifikacija, užduočių atlikimas žaidimo forma gali padidinti motyvaciją. 2021 metais atliktas eksperimentinis tyrimas rodo kad su žaidimifikacija laiku atliktu užduočių dalys yra 56.25% o be 31%, taip pat užduočių atlikimo kokybė buvo 43% ir 25.75% atitinkamai. Remiantis šiais rezultatais, galima teigti, kad zaidimifikacija motyvuoja atlikti daugiau užduočių, be to padeda vykdyti jas atlikti greičiau ir kokybiškiau (Changiz Hosseini, 2021). Galimybė padaryti žaidimą iš užduoties atlikimo taip pat gali būti naudojama grupiniame darbe.

## Alternatyvos

Savo planais ir pasiekimais žmonės jau daug metų dalijasi internete: socialiniuose tinkluose, asmeniniuose tinklaraščiuose ir forumuose. Taip pat idėja vizualizuoti tikslus ir uždavinius naudojant informacines sistemas nėra nauja – yra daugybė programų, kurios įvairiomis formomis padeda vartotojams padidinti produktyvumą. Štai keletas pavyzdžių, kuriuos galite rasti programų parduotuvėse:

* 1. **“Microsoft To Do”** - tai intelektualioji užduočių tvarkymo programa, palengvinanti dienų planavimą ir valdymą. Programa leidžia vartotojams valdyti savo užduotis iš savo išmaniojo telefono, planšetinio kompiuterio ir kompiuterio. Programa yra nemokoma.
  2. **“** **Challenges - Compete, Get Fit” –** tai sveikos gyvensenos palaikymo programėlė, leidžianti kurti komandas ir laikytis sveikos gyvensenos tikslų.
  3. **„Habitica“** – tai yra užduočių tvarkyklės paslauga, kurioje pagrindinis dėmesys skiriamas įpročiams ir produktyvumui, siekiant paversti darbą žaidimu. Su šia žiniatinklio programa vartotojai galės sulaukti ir apdovanojimų, ir bausmių, motyvuodami save ir socialinio tinklo pagalba. Pavyzdžiui, jei rūkote, herojus praranda sveikatą, o jei darote pratimus, personažas uždirba.

1 lentelė. Alternatyvių mobiliųjų programų palyginimas

| Pavadinimas | Challenges - Compete, Get Fit | Microsoft To Do | Habitica |
| --- | --- | --- | --- |
| Platformos |  |  |  |
| Kaina (1 mėn.) | nemokoma  5$ premium | nemokoma | 9.00 $ |
| Įvertinimas ir atsisiuntimų skaičius | 4.7  5.5+ tukst. | 4.5  200+ tukst. | 4.3  21+ tukst |
| Funkcijos | • Dalintis tikslais ir uždaviniais.  • Kurti grupė bendriems tikslams.  • Galimybė gauti apdovanojimus už pasiekimus.  • Siunčia priminimus.  • Galima siusti priminimus kontaktams | • Kurti, redaguoti, ištrinti užduotis ir sąrašus.  • Dalintis tikslais ir uždaviniais.  • Papildomų užduočių, pastabų kūrimas, galimybė pridėti failus.  • Klasifikuoti darbų sąrašus pagal spalvas.  • Siunčia priminimus.  •Užduočių komentarai  • Dalinti užduotis į atskirus etapus, kuriuos lengviau valdyti. | • Kurti, redaguoti, ištrinti užduotis ir sąrašus.  • Dalintis tikslais ir uždaviniais.  • Papildomų užduočių, pastabų kūrimas, galimybė pridėti failus.  • Klasifikuoti darbų sąrašus pagal spalvas.  • Siunčia priminimus.  •Užduočių komentarai  •Produktyvumo analizė  •Apdovanojimų sistema |
| Sąsajos nuotraukos | 2 pav. Programėlės “Challenges” vartotojo sąsaja | 3 pav. Programėlės „Microsoft To Do“ vartotojo sąsaja | 4 pav. Programėlės „Habitica“ vartotojo sąsaja |

Taigi, šiame skyriuje išanalizavome alternatyvias programas, jų funkcijas ir išvaizdą. „Microsoft To Do“ ir „Habitica“ šiuo metu tai yra populiariausios tikslų valdymo programos. Pagrindinis šių aplikacijų privalumas tai, kad jie prieinami įvairiose operacinėse sistemose ir turi daugybę funkcijų. „Habitica“ tai naujas požiūris į planavimą kaip į žaidimą. O “Challenges - Compete, Get Fit” yra viena iš populiariausių grupinių sportinių išsuksiu programėlių, bet ji yra sukurta tik iOs sistemos mobiliesiems įrenginiams.

Kad galėčiau palyginti alternatyvas su mano sistema buvo iškelti tokie reikalavimai sistemai:

* Neturi mobiliosios sistemos apribojimų.
* Neturi teminio apribojimo.
* Yra galimybė kurti naudotojų grupes.
* Galima dalintis nuotraukomis ir video failais.
* Yra galimybe paliginti grupės nariu pasiekimus.

Kaip galima pamatyti iš 2 lenteles mano mobilioji aplikacija skirsis nuo pateiktų alternatyvų, taps savotišku socialiniu tinklu, kuriame žmonės galės kurti ir vienytis siekdami savo tikslų.

2 lentelė. Alternatyvų palyginimas pagal iškeltus sisteminius reikalavimus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programėle | Neturi mobiliosios sistemos apribojimų | Neturi teminio apribojimo | Yra galimybė kurti naudotojų grupes | Galima dalintis nuotraukomis ir video failais | Yra galimybe paliginti grupės nariu pasiekimus |
| Challenges - Compete, Get Fit | Ne | Taip | Ne | Taip | Ne |
| Microsoft To Do | Taip | Taip | Taip | Taip | Ne |
| Habitica | Taip | Ne | Ne | Ne | Taip |
| Kuriama sistema | Taip | Taip | Taip | Taip | Taip |

## Kuriamos programėlės techninė analizė

### Mobiliųjų aplikacijų kūrimo karkaso pasirinkimas

Mobiliųjų aplikacijų kūrimas yra labai didelė ir vis dar auganti IT sferos dalis. Vidutiniškai kas mėnesį „Google Play“ išleidžiama daugiau nei 100 000 naujų programėlių (Statistica, 2021), o „Apple App Store“– daugiau nei 30 000 naujų programėlių (Matters, 2022). Kai programuotojas nori parašyti programą, pirmasis klausimas yra: "kokią technologiją turėčiau naudoti?". Šis klausimas yra labai svarbus ir atsakymas priklauso nuo pasirinktos platformos (įrenginio operacines sistemos) bei programuotojo patirties. Priklausomai nuo technologijos, mobiliųjų programų kūrimo karkasus galima suskirstyti į tris pagrindinius tipus:

**1. “Native” programėles** – kuriamos konkrečiai platformai, skirtos „Android“ arba „iOS“ naudotojams. Yra geriau optimizuoti: veikia sklandžiai ir be strigimų, net esant didelei GPU apkrovai ir sudėtingam skaičiavimo integravimui. Taip pat tokios aplikacijos užima mažiau vietos telefone. Geriausias dalykas tokio tipo programėlėje yra tai, kad ji suteikia prieigą prie integruotų įrenginių ar konkrečios platformos galimybių, pavyzdžiui prie kameros, gps, operatyvios atminties ir t.t. Žinoma, yra ir trukumų. Reikalingos dvi kūrėjų komandos: „Android“ programos paprastai kuriamos „Java“ arba „Kotlin“ programavimo kalba, o „iOS“ programos kuriamos „Objective-C“ arba „Swift“, todėl reikia samdyti komandą, turinčią patirties šiomis kalbomis. Dar vienas trukumas - didelės kūrimo išlaidos: “native” programėles idealiai tinka didelėms korporacijoms, turinčioms didelį biudžetą. Kadangi kiekvieną programą reikia kurti nuo nulio, tam, kad ji būtų ekonomiškai sėkminga, reikės daug išteklių, įskaitant laiką.

**2. Žiniatinklio programėles** – iš esmes programėle yra konteineris tinklalapiu, kuris gali turėti dizainą išmaniajam įrenginiui. Vartotojo sąsaja kuriama naudojant standartines žiniatinklio technologijas. Žiniatinklio programėles darbui naudoja telefono naršyklę. Tas faktas, kad programėle galima naudotis tik internete sumažina saugumą. Pagrindinis tokių programų bruožas yra jų universalumas – galimybė dirbti visuose įrenginiuose be papildomo pritaikymo. Nepriklausomai nuo įdiegtos operacines sistemos, tokios programėles negali naudoti išmaniojo telefono programinės įrangos. Norint atnaujinti informaciją programėlėje, reikalingas interneto ryšys, veikimo greitį riboja paslaugų teikėjo interneto ryšio galimybės. Žemiau (1 lentele) pateiktas populiariausiu žiniatinklio kūrimo karkasų palyginimas.

**3. Hibridinės programėles** – kitaip nei “native” programėlės, hibridinės programėlės yra kuriamos abiem platformoms (“Android” ir “iOS”) vienu metu ir rašomos universalia kalba. Naudoja kai “Native” taip ir žiniatinklio programėlių savybes. Hibridinėms programoms reikalingas interneto ryšys, nes žiniatinklio dalis atnaujinama internetu. Tarp pagrindinių jų kūrimo įrankių yra „Flutter“, „Ionic“, „Xamarin“, jų palyginimas yra pateiktas žemiau (2 lentele).

**Chart, bar chart

Description automatically generated**

5 pav. Populiariausi hibridiniu ir web mobiliųjų programėlių sukūrimo karkasai (Statista, 2021).

Mobiliosioms programėlėms kurti yra daug karkasų. Kiekvienas iš jų turi savo ypatybes, privalumus ir trūkumus. Toliau pateiktose lentelėse (3-4) pateikiami dažniausiai pasitaikantys karkasai su jų privalumais ir trūkumais.

3 lentelė. Populiarių žiniatinklio programėlių kūrimo karkasų palyginimas

| Karkaso pavadinimas | Privalumai | Trukumai |
| --- | --- | --- |
| React | •Plėtros išlaidos sumažėja maždaug 50%. React Native priežiūra taip pat yra pigesnė.  •Greitas projektų išleidimas  •Reikalinga nedidelė komanda  •Didžioji dalis kūrimo užduočių atliekama JavaScript.  •Atvirojo kodo privalumas  •Suteikia kūrėjams galimybę nemokamai naudotis bibliotekomis ir sistemomis. | •Licencijų ir patentų klausimai  •Bloga dokumentacija  •Dokumentacijos problema yra susijusi su nuolatiniu naujų įrankių išleidimu.  •Nestabilumas, suderinamumo problemos ir klaidos |
| jQuery | •Suderinamumas  •jQuery sintaksę palaiko visos šiuolaikinės interneto naršyklės  •Kodo kompaktiškumas.  •Tai, ką turite parašyti „JavaScript“ atskirose funkcijose, „jQuery“ yra įgyvendinta vos keliomis kodo eilutėmis.  •Paprastumas  •Nereikalauja daug programavimo žinių. | •Karkasas jau pabaigtas kurtis, paskutine versija buvo išleista 2012 metais.  •Letas greitis  •Bibliotekos dydis  • „jQuery“ bibliotekos dydis yra apie 19 Kb, o tai teoriškai gali sulėtinti tinklalapio įkėlimą, ypač senesniuose kompiuteriuose ir nestabiliuose tinklo ryšiuose. |
| Express | •Atviras pirminis kodas  •Gera dokumentacija  •Greitas programėlių kūrimas  •Lengva išmokti | •Mažas pagrindinių funkcijų kiekis  •Visos kitos reikalingos funkcijos turės būti pridėtos išorinių modulių pagalba.  •Turi tik viena gyja veiksmu apdorojimui  •Tai nėra įprasta ir gali pakenkti nepatyrusio programuotojo produktyvumui. |
| Angular | •Didelė bendruomenė  •Deklaratyvus kodo stilius  •Universalumas  •AngularJS sukurtas kaip bendros paskirties sistema, todėl su juo galite kurti beveik bet kokio tipo žiniatinklio programas.  •Testavimo paprastumas  •Leidžia efektyviai atlikti automatizuotą testavimą. | •Greitis  •Dėl architektūrinių savybių gali būti problemos su greitaveika.  •Bloga dokumentacija  •Dažnai atidarę dokumentaciją galite pamatyti tik klasės metodų pavadinimą, be jokio aprašymo, ką ir kaip jie daro.  •Sudėtinga sintaksė |

4 lentelė. Populiarių hibridinių programėlių kūrimo karkasų palyginimas

| Karkaso pavadinimas | Privalumai | Trukumai |
| --- | --- | --- |
| Flutter | •Perspektyvus ir aktyviai tobulinamas.  •Nuosavas grafinis variklis (nereikia kurti atskiros sąsajos Android ir iOS).  •Vieną kartą pasirašius kodą, jį galima eksportuoti į visas palaikomas platformas.  •Kuriant programėle ir padarius pakeitimą nereikia paleisti programėle iš naujo kad pamatyti pasikeitimą.  •Atviras pirminis kodas.  •Gera greitaveika. | •Nestabilumas (neseniai išėjo iš beta versijos).  •Bibliotekų yra mažiau nei “Native” karkasuose. Bet verta paminėti kad visos svarbiausios bibliotekos jau yra, nuolat atsiranda naujų.  •Mažai mokomosios medžiagos.  •Mažai papildiniu, jau sukurtu sąsajos elementu. |
| Xamarin | •Viena iš plačiausiai programinės įrangos kūrėjų ir didelių bei mažų įmonių naudojamų mobiliųjų programėlių kūrimo platformų rinkoje.  •Pilnas kūrimo įrankių paketas.  • „Microsoft“ techninė pagalba  •Atviras kodas  •Našumas artimas “native” karkasams  •Savo vartotojo sąsaja (angl, UI – user interface) | •Ne visos bibliotekos yra prieinamos.  •Programėles užima daug vietos.  •Atnaujinimų vėlavimas.  •Mažesnė bendruomenė |
| Ionic | •Pateikiama daug paruoštų grafiniu sąsajos elementų.  •Sukurta su AngularJS.  • „Ionic“ vartotojo sąsaja itin patogi.  •Sukurti programėle viena karta ir eksportuoti visiems platformoms. | •Kuriant programėle gali atsirasti saugumo problemos, kurias reikės taisyti programuotojams.  •Programėle neatlaiko daug gradinių elementu.  •Integruoto funkcionalumo apribojimas |

Išanalizavęs galimas mobiliųjų aplikacijų kūrimo technologijas, pasirinkau „Flutter“. „Flutter“ naudoja „Dart“ programavimo kalbą ir yra hibridinis programėlių kūrimo įrankis. Kaip galima pamatyti, „Flutter“ naudojimas sparčiai išaugo 2021 metais ir aplenkė „React Native“ (5 pav.). Pagrindinės priežastys, kodėl pasirinkau „Flutter“, yra geras greitis, galimybė sukurti vieną programėlę visiems išmaniųjų įrenginių operacinėms sistemoms ir tai, kad jau turiu patirties kuriant mobiliąją programėlę su šia technologija.

### Išsami „Flutter“ technologijos apžvalga

“Flutter” - nemokamas atvirojo kodo mobiliojo vartotojo sąsajos kūrimo rinkinys, sukurtas „Google“ ir išleistas 2017 m. gegužės mėn. „Flutter“ yra naudojamas tokioms programėlėms sukurti kaip „Alibaba Gourp“, „BMW“, „Google Pay“, „Stadia“, „Tencent“, „Toyota“, „eBay“ ir t.t.. (Flutter, 2022) Darbo su „Flutter“ ypatybė yra ta, kad skirtingų platformų programos gali turėti tą patį kodą. Kuriant su “Flutter” naudojama programavimo kalba, vadinama “Dart”. Tai taip pat „Google“ kalba, sukurta 2011 m. spalio mėn., tačiau pastaraisiais metais ji labai patobulėjo. „Statista“ svetainės duomenimis, nuo 2019 metų hibridinių platformų kūrėjų, naudojančių „Flutter“, skaičius pasaulyje išaugo nuo 30% iki 42%, taip aplenkdamas savo konkurentą „React Native“ (38%) (Statista, 2021).

Programos paleidžiamos remiantis atvaizdavimo variklio (sukurta C++) ir „Flutter“ (sukurta ant Dart) deriniu. Visi tokiu būdu sukurti failai pridedami prie kiekvienos konkrečios platformos programos ir SDK rinkinių programinės įrangos. Žemiau (4 pav.) pateikiamas „Flutter“ architektūros modelis.

.Graphical user interface, chart

Description automatically generated

6 pav. „Flutter“ architekturos modelis

Tai panašu į žaidimo kūrimą: žaidimas nepaskirsto savo karkaso, o funkcionalumas vykdomas naudojant žaidimo variklį. Tas pats ir „Flutter“ programinei įrangai – visos programos, pagrįstos „Flutter SDK“, pakeičia dalis vietinių sistemų „Flutter“ elementais.

Diagram

Description automatically generated

7 pav. „Flutter“ sąveika su „native“ komponentais

Nors tai gali turėti įtakos galutinės programos dydžiui, našumas vis tiek yra gana geras – atvaizdavimas atliekamas iki 120 FPS greičiu.

Dėl įprasto ARM procesorių kompiliavimo, paprasto atvaizdavimo ir integruotų valdiklių bei įrankių rinkinio „Flutter“ supaprastina kūrimo procesą.

„Dart” yra dinamiška, klasėmis pagrįsta, į objektą orientuota programavimo kalba su uždarymu ir leksine apimtimi. Sintaksės požiūriu jis yra gana panašus į „Java“, „C“ ir „JavaScript“. Jei žinote kurią nors iš šių programavimo kalbų, galite lengvai išmokti „Dart“ programavimo kalbą.

„Dart“ yra atvirojo kodo programavimo kalba, plačiai naudojama kuriant mobiliąsias programas, modernias žiniatinklio programas, darbalaukio programas ir daiktų internetą (IoT), naudojant „Flutter“ sistemą. Ji taip pat palaiko keletą pažangių sąvokų, tokių kaip sąsajos, mišiniai, abstrakčios klasės, refield generics ir tipo sąsaja. Tai yra kompiliuota kalba ir palaiko dviejų tipų kompiliavimo būdus.

AOT (prieš laiką) – jis konvertuoja Dart kodą į optimizuotą JavaScript kodą, naudodamas dar2js kompiliatorių ir veikia visose šiuolaikinėse žiniatinklio naršyklėse. Jis sukompiliuoja kodą kūrimo metu.

JOT (Just-In-Time) – konvertuoja baitų kodą į mašinos kodą (savąjį kodą), bet tik reikiamą kodą.

### Vidinies programos dalies technologiju pasirinkimas

„Backend“ yra programa kuri veikia serverio puseje, ji saugo ir tvarko duomenis. Tai yra programinės įrangos dalis, kuri neturi tiesioginio kontakto su vartotojais. Vartotojai prieigą prie funkcijų gauna netiesiogiai, naudodamiesi sukurta sąsaja – API. Yra keletas sprendimų kaip galima ralizuoti API, pagrindinios technologijos tai:

1. SOAP (Simple Object Access Protocol) – XML pagrindu sukurtas siuntimo protokolas, skirtas informacijos siuntimui tarp kompiuterių, kurio pagrindinė idėja yra iškviesti nuotolines procedūras per HTTP. SOAP pagrįstos sistemos yra pageidautinos tais atvejais, kai naudojamos įmonės programos, kuriose reikia integruoti išankstines paslaugas. SOAP išlaiko transakcijų būsenas. Štai keletas pagrindinių naudojimo atvejų, kuriose verta naudotis SOAP:
   * kai duomenis galima naudoti pageidaujamomis kalbomis.
   * Kai reikia sukurti saugu komunikacijos kanala tarp naudoto ir sistemos.
   * SOAP palaiko SMTP, JSM, TCP transportavimo protokolus.

Taigi išvardintuose scenarijuose SOAP žiniatinklio paslauga bus geriausias pasirinkimas, kitiems scenarijoms reikia naudoti REST.

1. REST (Representational State Transfer) – yra architektūrinis stilius, apibrėžiantis apribojimų rinkinį, kuris bus naudojamas kuriant žiniatinklio paslaugas. Jis suteikia laisvą sujungimo galimybė, mastelio padidinima ir paprastumą. Ši technologija turėtų būti naudojama šiais atvejais :
   * Kai reikia sukurti paprastą sąsają bendrauti su sistema.
   * Kai užklausoma informacija nėra dinamiškai generuojama.
   * Kai programuotojas ir vartotojas tarpusavyje susitaria dėl perduodamo turinio ir nėra griežto formato duomenų keitimui.
   * Kai yra apribojimai duomenų pralaidumui, čia yra ypač aktualu naudojant mobilųjį internetą.

Po tyrimo nusprendžiau, kad savo vidinies programos interfeisui naudosiu REST architektūrą RESTfulAPI kurimui. Šis API veiks kaip tarpininkas tarp vidinės sistemos ir mobiliosios aplikacijos. Žemiau esančioje lentelėje pateikiami serverių karkasai, jų privalumai ir trūkumai.

5 lentelė. Vidiniu sistemu karkasų palyginimas

|  |  |  |
| --- | --- | --- |
| Karkaso pavadinimas | Privalumai | Trukumai |
| Spring | Mažai sveria  Daug moduliu su ivairiu funkcionalumu  Lengvai testuojamas | Gali buti sudetinga konfiguruoti.  Neapsaugo nuo XSS ataku. |
| Ruby on Rails | Lengvai naudojamas  Plačiai paplites, turi daug dokumentacijos ir geru praktiku. | Sunku konfiguruoti  Letai veikia palyginus su alternativomis |
| Meteor | Atviras pradinis kodas  Turi integruota ORM duoemnis saugoti naudojant MongoDB | Nepalaiko kitu duomenu baziu.  Naudoja nestandartinius sprendimus |
| Django | Turi integruotos apsauga nuo populiariausiu isilauzimo praktiku (XSS; SQL Injection)  Naudoja Python | Reikalauja optimizacijos kad padidinti greitaveika  Neturi bendrai pripazistu geriausiu praktiku |

Savo projekte naudosiu „Spring“ karkasą, del to kad suteikia suteikia didžiausią įvairių funkcijų rinkinį, yra pilnai ir lengvai konfigurojimas. Jis naudoja „Java“ programavimo kalba. Dabar reikia pasirinkti duomenų bazę, kurią bus naudojama su „Spring“ karkasu. Duomenų bazė naudojama duomenims gauti ir saugoti. Priklausomai nuo to, kokius duomenis jame reikia saugoti ir kaip su jais dirbti, duomenų bazės skirstomos į reliacines ir nereliacines. Reliacinės duomenų bazės dar vadinamos lentelių duomenų bazėmis, nes visi jose esantys duomenys gali būti pateikiami skirtingose ​​lentelėse. Lenteles galima sujungti tarpusavyje pagal raktą. O nereliacinės (NoSQL) duomenų bazės saugo reikšmės kaip objektų sąrašą. Skirtingais atvejais naudojami skirtingi tipai, priklausomai nuo saugomų duomenų parametrų, tokių kaip duomenų kiekis, homogeniškumas, kaip jie bus naudojami sistemoje. Pavyzdžiui, yra daug vienodų objektų, kurie bus gaunami pilnai iš duomenų bazės kaip sąrašas, šiuo atveju geriau naudoti NoSQL duomenų bazę. O kai mes turime duomenis, iš kurių mums reikia gauti tam tikrą statistiką arba agreguoti juos į kazkokį naują duomenų tipą, geriau panaudoti reliacinę duomenų bazę (IBM, 2021). Savo projekte naudosiu SQL duomenų bazę, vadinamą H2.

### Išsami vidinies dalies apžvalga

„Spring“- populiariausia Java programų kūrimo aplinka (Intexsoft, 2020). „Spring“ atsirado 2003 m. kaip lengva „Java Enterprise“ platformos versija. Tai yra lengvai naudojama žiniatinklio programų kūrimo platforma. „Spring“ palaiko keletą programavimo kalbų: „Java“, „Kotlin“ ir „Groovy“.

Kaip parodyta schemoje žemiau (6 pav), “Spring” sudaryta iš daugybės modulių. Tarp jų yra pradiniai moduliai, be kurių „Spring“ aplikacijos tiesiog nepasileis, taip pat yra pagalbinių projektų, kurie papildo programą tam tikru funkcionalumu: pavyzdžiui, „Spring Data Flow“ srautiniam duomenų apdorojimui, „Security“ saugumui arba „Cloud“ paskirstytoms sistemoms. Ši struktūra leidžia kūrėjams efektyviai kurti ir prižiūrėti programas naudojant tik jiems reikalingus įrankius.



8 pav. Spring karkaso architekturinis modelis

#### Pagrindiniai „Spring“ karkaso sluoksniai:

1. **Core Container**

Pagrindine „Spring Core“ ideja yra IoC („Inversion of Control“) kas dar yra vadinama priklausomybiu injekcija. Tai procesas, kurio metu objektai apibrėžia savo priklausomybes (ty kitus objektus, su kuriais jie dirba) tik per konstruktoriaus argumentus. Tada konteineris sukuria šias priklausomybes, kai sukuria „Bean“. Šis procesas iš esmės yra atvirkštinis (taigi ir pavadinimas, valdymo inversija), kai pats „Bean“ kontroliuoja savo priklausomybių egzistavimą arba lokacija, naudodamas tiesioginę klasių konstravimą arba šablonus, pvz., paslaugų lokatoriaus modelį.  
 „Spring“ objektai, kurie sudaro programmos pagrindą ir kuriuos valdo „Spring IoC“ konteineris, vadinami „Bean“. „Bean“ yra objektas, kurį sukuria, surenka ir valdo „Spring IoC“ konteineris ir jų priklausomybės atsispindi konteinerio naudojamuose konfigūracijos metaduomenyse.  
 SpEL („Spring Expression Language“) yra galinga išraiškų kalba, kuri palaiko užklausų pateikimą ir objekto grafiko manipuliavimą vykdymo metu. Ipač naudingas irankis programu testavimui ir givam programmos derinimui.

1. **Data Access / Integration**

„Spring Data“ tikslas yra pateikti žinomą ir nuoseklų „Spring“ pagrindu sukurtą programavimo modelį prieigai prie duomenų, išlaikant specialias pagrindinės duomenų saugyklos savybes. Tai leidžia lengvai naudoti duomenų prieigos technologijas, reliacines ir nesusijusias duomenų bazes, žemėlapių mažinimo sistemas ir debesų duomenų paslaugas. Keleta is pagrindiniu duomenu sluoksinio moduliu:

* JDBC („Java Database Connectivity“) – yra taikomųjų programų programavimo sąsaja (API), apibrėžianti, kaip klientas gali pasiekti duomenų bazę. Tai duomenų prieigos technologija, naudojama „Java“ duomenų bazės ryšiui palaikyti. Jame pateikiami duomenų bazės duomenų užklausų ir atnaujinimo metodai ir yra orientuotos į reliacines duomenų bazes. JDBC siūlo natūralią Java sąsają darbui su SQL.
* ORM – Spring-ORM yra technika arba dizaino šablonas, naudojamas prieigai prie reliacinės duomenų bazės iš objektinės kalbos. ORM (Object Relation Mapping) apima daugybę patvarumo technologijų. Jie yra tokie:
* JPA (Java Persistence API): ji daugiausia naudojama išlaikyti duomenis tarp Java objektų ir reliacinių duomenų bazių. Jis veikia kaip tiltas tarp objektinių domenų modelių ir reliacinių duomenų bazių sistemų.
* JDO (Java Data Objects): tai vienas iš standartinių būdų pasiekti nuolatinius duomenis duomenų bazėse, naudojant paprastus senus Java objektus (POJO) nuolatiniams duomenims pavaizduoti.
* Hibernate – tai „Java“ sistema, kuri supaprastina „Java“ programų kūrimą, kad jos galėtų sąveikauti su duomenų baze. Jos esme yra automatizuotas programos objektu konvertavimas i ir iš reliaciniu duomenu baziu formato.
* Transactions – modulis valdo pakeitimus, kuriuos atliekate vienoje ar keliose sistemose. Tai gali būti duomenų bazės, pranešimų brokeriai ar bet kokios kitos programinės įrangos sistemos. Pagrindinis operacijos tikslas yra pateikti ACID (Atomicity; Consistency; Isolation; Durability) charakteristikas, kad būtų užtikrintas jūsų duomenų nuoseklumas ir galiojimas.

1. **Spring Web (MVC / Remoting)**

„Spring WEB“ sluoksnis suteikia modelio peržiūros valdiklio (MVC) architektūrą ir paruoštus komponentus, kurie gali būti naudojami kuriant lanksčias ir laisvai susietas žiniatinklio programas. Dėl MVC modelio atskiriami skirtingi programos aspektai (įvesties logika, verslo logika ir vartotojo sąsajos logika), kartu užtikrinant laisvą šių elementų ryšį.

1. **Test**

Testavimas yra neatsiejama įmonės programinės įrangos kūrimo dalis. Priklausomybės įvedimas turėtų padaryti jūsų kodą mažiau priklausomą nuo konteinerio, nei tai būtų naudojant tradicinį Java EE kūrimą. POJO, sudarantys jūsų programą, turėtų būti išbandomi atliekant JUnit arba TestNG testus, o objektai sukuriami naudojant naują operatorių, be Spring ar kito konteinerio. Galite naudoti netikrus objektus (kartu su kitais vertingais testavimo būdais), norėdami išbandyti savo kodą atskirai. Jei laikysitės „Spring“ architektūros rekomendacijų, gautas švarus kodų bazės sluoksniavimas ir komponentų sudarymas palengvins vienetų testavimą. Pvz., galite išbandyti paslaugų sluoksnio objektus, užklijuodami arba tyčiodamiesi iš DAO arba saugyklos sąsajų, nereikės pasiekti nuolatinių duomenų, kai vykdomi vienetų bandymai.

# Projektinė dalis

# Sistemos naudotojai

6 lentelė. Naudotojo „Neprisijungęs naudotojas“ detalizavimas

|  |  |
| --- | --- |
| **Rolė:** | Neprisijungęs naudotojas |
| **Funkcijos:** | Registruotis arba prisijungti per „Facebook“, „Google“. |
| **Svarba:** | Rolė yra priskiriama automatiškai kiekvienam neprisijungusiam naudotojui. |

7 lentelė. Naudotojo „Vartotojas“ detalizavimas

|  |  |
| --- | --- |
| **Rolė:** | Vartotojas |
| **Funkcijos:** | Prisijungti prie paskyros, dalyvauji grupiniuose iššūkiuose, rašyti komentarus, gauti taškus už įvykdytus iššūkius. |
| **Svarba:** | Rolė yra priskiriama automatiškai kiekvienam naujai užsiregistravusiam naudotojui. „Vartotojas“ tai pagrindiniai naudotojai kurie dalyvauja iššūkiuose. |

8 lentelė. Naudotojo „Moderatorius“ detalizavimas

|  |  |
| --- | --- |
| **Rolė:** | Moderatorius |
| **Funkcijos:** | Prižiūri vartotojus grupėje, priima naujus narius ir šalina iš grupės vartotojus. Tvarko komentarus ir įrašus. |
| **Svarba:** | Rolė priskiria grupės kūrėjas. Moderatorius tai rolė, kurį palaiko tvarka grupėse (komentaruose ir nariuose). |

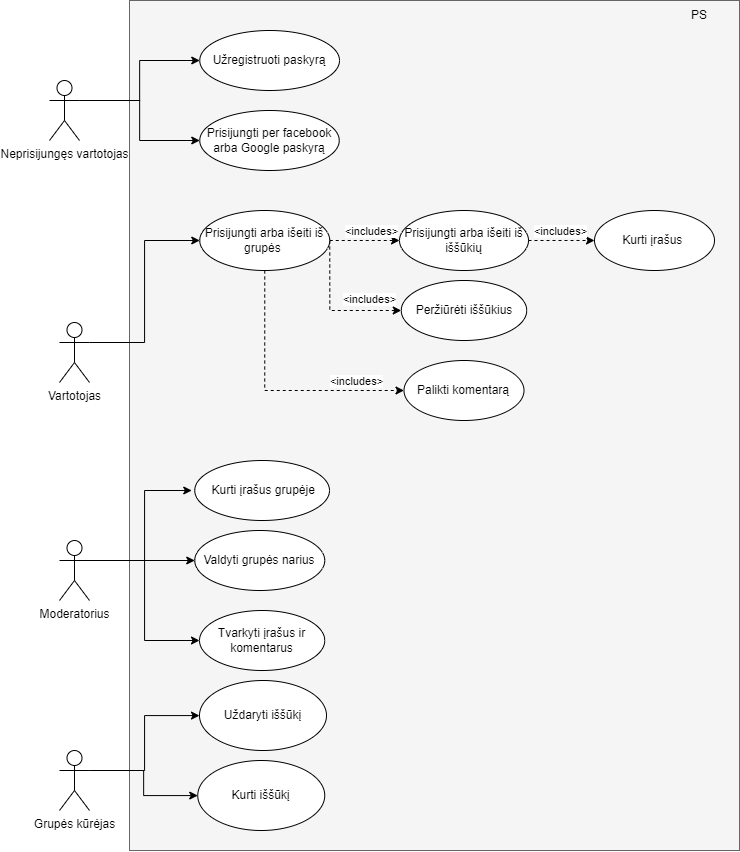
9 lentelė. Naudotojo „Grupės kūrėjas“ detalizavimas

|  |  |
| --- | --- |
| **Rolė:** | Grupės kūrėjas |
| **Funkcijos:** | Kurti iššūkius, pradėti, sustabdyti arba baigti iššūkį, redaguoti įrašus, gali suteikti vartotojui rolė „Moderatorius“, redaguoti iššūkį. |
| **Svarba:** | Rolė yra suteikiama vartotojams, kurie sukuria grupę. |

# Panaudos atvejų diagrama

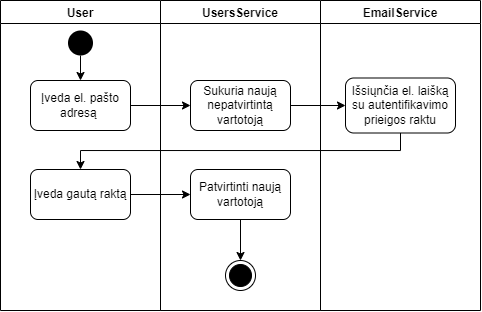
Panaudos atvejų diagrama (angl. *use case*) – tai diagramos, kurios aprašo sistemos funkcionalumą panaudojant aktorius ir galimus panaudojimo atvejus. Panaudos atvejų diagrama parodo, kurie aktoriai sąveikauja su kiekvienu panaudos atveju. Aktoriai aprašo sistemos vartotojus. Panaudos atvejai atvaizduoja veiksmus ,atliekamus vieno ar daugiau aktorių, siekiant konkretaus tikslo. Įtraukimas (angl. *include*) yra naudojamas norint parodyti, kaip panaudojimo atvejis yra suskaidomas į mažesnes veiklas.

Žemiau (9 pav.) pateikta panaudojimo atvejų diagrama, kuri susideda iš 12 panaudojimo atvejų.



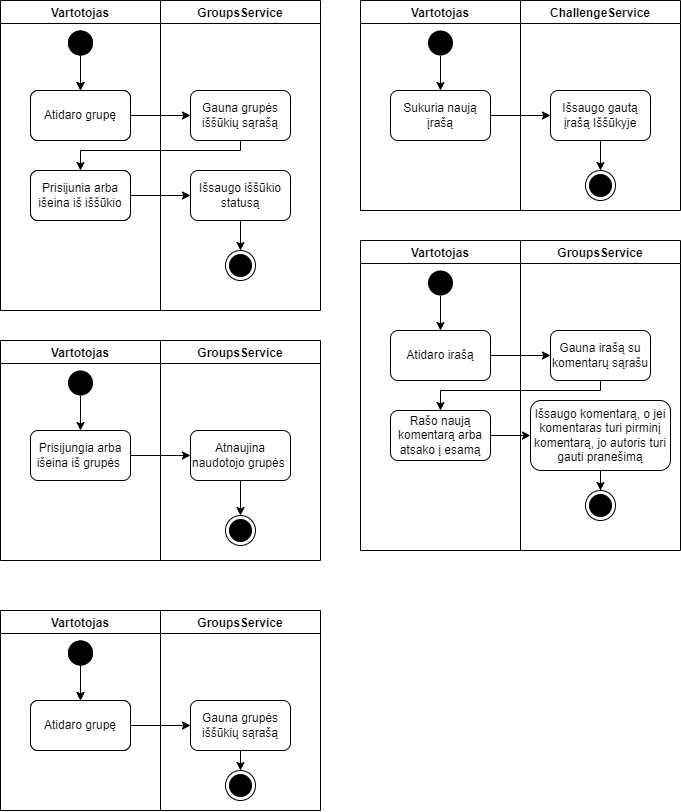
9 pav. Panaudos atvejų diagrama

Kiekvienas 9 paveikslėlyje pavaizduotas panaudojimo atvejis detalizuojamas specifikacijų lentelėmis ir užduočių modeliais:



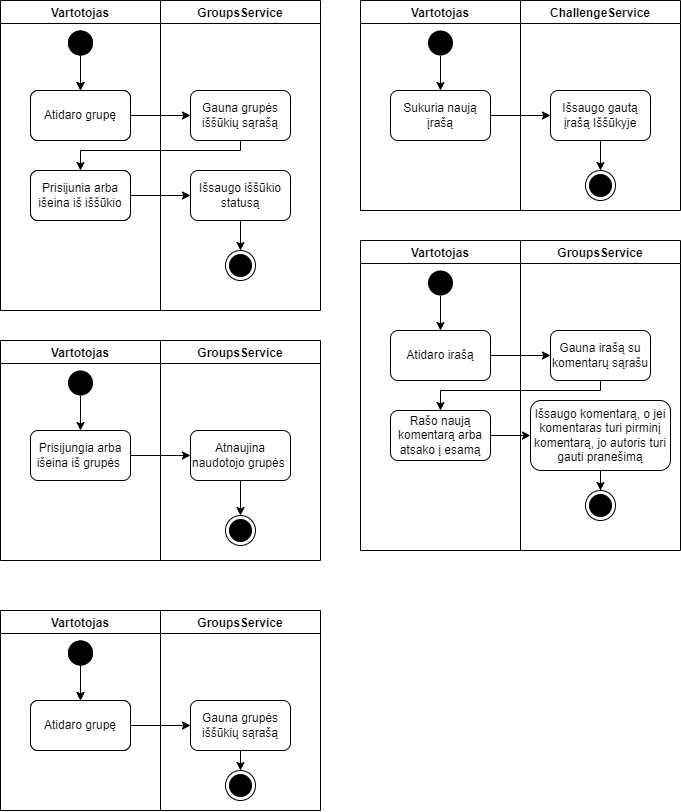
10 pav. „Užregistruoti paskyrą“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 1 | |
| Panaudojimo atvejis | Užregistruoti paskyrą |
| Tikslas | Tam, kad aplikacija rodytų paskaitų tvarkaraštį, ji turi žinoti grupę, kurioje mokosi studentas |
| Vartotojas/Aktorius: | Neprisijungęs vartotojas |
| Prieš - sąlyga: | Vartotojas atidaro mobilioji aplikaciją |
| Sužadinimo sąlyga: | Mygtuko „registruotis“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas įveda savo el.paštą ir spaudžia mygtuką „gauti raktą“; 2. Į el.paštą gauna laišką su autorizacijos raktu; 3. Vartotojas įveda raktą; |
| Po-sąlyga: | Į duomenų bazę įtrauktas naujas vartotojas |



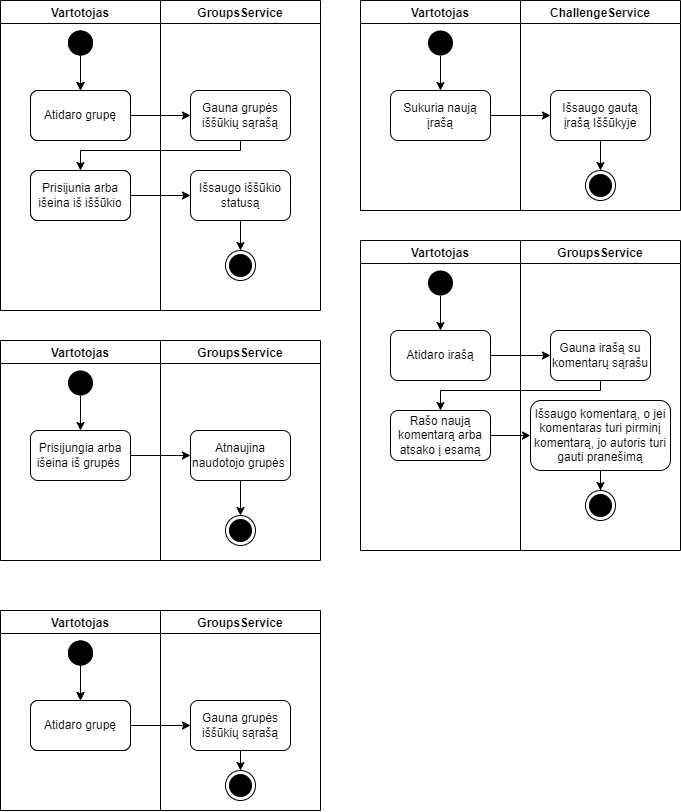
11 pav. „Prisijungti arba išeiti iš grupės“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 3 | |
| Panaudojimo atvejis | Prisijungti arba išeiti iš grupės |
| Tikslas | Tam, kad matyti iššūkių sąrašą grupėje ir dalyvauti juose reikia prisijungti prie grupės. Ir apskritai, tam kad nematyti kažkokios grupės iššūkius reikia išeiti iš grupės. |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti autorizuotas |
| Sužadinimo sąlyga: | Norint prisijunti prie grupės vartotojas spaudžia mygtuką „+“;  Norint išeiti iš grupės vartotojas spaudžia mygtuką „Išeiti iš grupės“ |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas atidaro grupę; 2. Spaudžia mygtuką „+“ arba „Išeiti iš grupės“; |
| Po-sąlyga: | Duomenų bazėje sėkmingai atnaujinti vartotojo grupės |



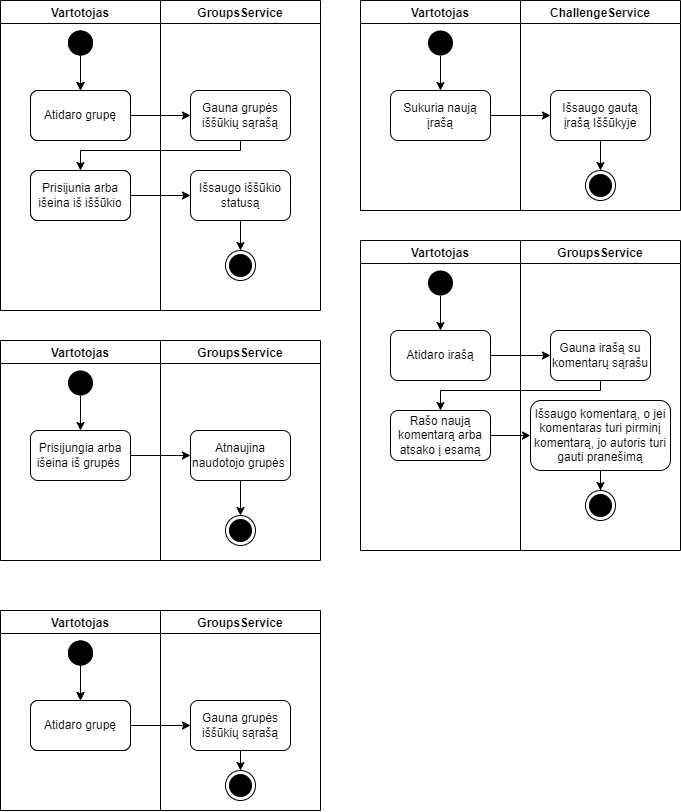
12 pav. „Prisijungti arba išeiti iš iššūkio“ užduoties modelis

| Specifikacijų lentelė № 4 | |
| --- | --- |
| Panaudojimo atvejis | Prisijungti arba išeiti iš iššūkio |
| Tikslas | Tam, kad dalyvauti iššūkiuose reikia būti grupės narių |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti grupės narys |
| Sužadinimo sąlyga: | Mygtuko „Prisijunti prie iššūkio“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas atidaro grupę; 2. Pasirenka iššūkį prie kurio nori prisijunti ; 3. Spaudžia mygtuką „Prisijunti prie iššūkio“; |
| Po-sąlyga: | Duomenų bazėje sėkmingai atnaujinti vartotojo iššūkiai |



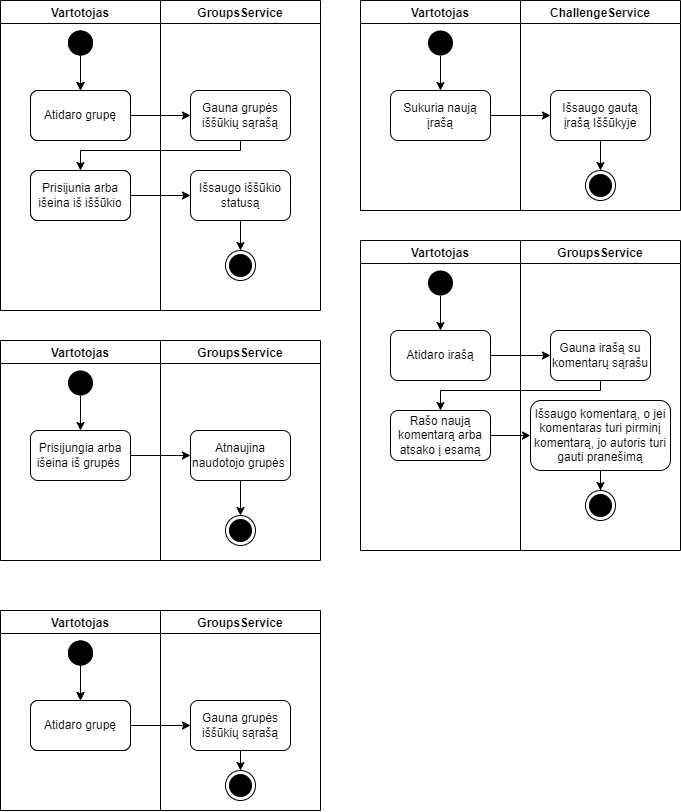
13 pav. „Kurti įrašus“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 5 | |
| Panaudojimo atvejis | Kurti įrašus |
| Tikslas | Informuoti grupės narius apie iššūkio eigą |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti grupės narys ir dalyvauti iššūkyje |
| Sužadinimo sąlyga: | Mygtuko „+“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas užeina į grupę, pasirenka iššūkį; 2. Vartotojas spaudžia į mygtuką „+“ ; 3. Įrašo tekstinę žinutę, gali pridėti vaizdo įrašą arba nuotrauką; 4. Spaudžia „Publikuoti“; |
| Po-sąlyga: | Įrašas sėkmingai pridėtas į iššūkį |



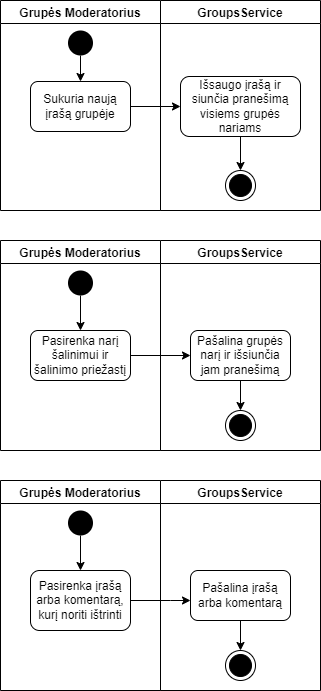
14 pav. „Palikti komentarą“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 6 | |
| Panaudojimo atvejis | Palikti komentarą |
| Tikslas | Pareikšti savo nuomonę, pagirti dalyvius, pakritikuoti ir t.t |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti grupės narys ir dalyvauti iššūkyje |
| Sužadinimo sąlyga: | Vartotojas spaudžia mygtuką po įrašo „Atsakyti“ |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas užeina į grupę, pasirenka iššūkį; 2. Atidaro įrašą; 3. Rašo komentarą arba atsako į esamą; |
| Po-sąlyga: | Komentaras sėkmingai paliktas po įrašų |



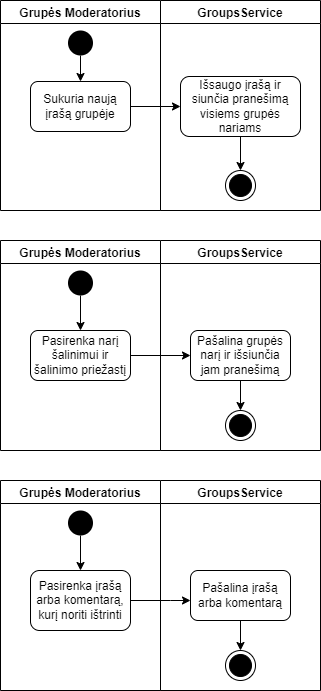
15 pav. „Peržiūrėti iššūkius“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 7 | |
| Panaudojimo atvejis | Palikti komentarą |
| Tikslas | Pareikšti savo nuomonę, pagirti dalyvius, pakritikuoti ir t.t |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti grupės narys ir dalyvauti iššūkyje |
| Sužadinimo sąlyga: | Vartotojas spaudžia mygtuką po įrašo „Atsakyti“ |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas užeina į grupę, pasirenka iššūkį; 2. Atidaro įrašą; 3. Rašo komentarą arba atsako į esamą; |
| Po-sąlyga: | Komentaras sėkmingai paliktas po įrašų |



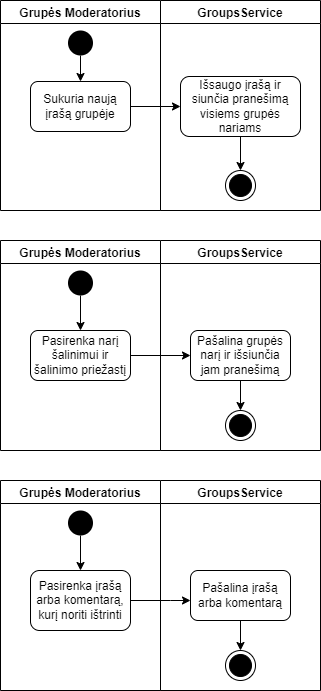
16 pav. „Kurti įrašus grupėje“ užduoties modelis

| Specifikacijų lentelė № 8 | |
| --- | --- |
| Panaudojimo atvejis | Kurti įrašus grupėje |
| Tikslas | Informuoti grupės narius |
| Vartotojas/Aktorius: | Grupės moderatorius |
| Prieš - sąlyga: | Moderatorius užeina į grupę |
| Sužadinimo sąlyga: | Mygtuko „+“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Moderatorius užeina į grupę; 2. Spaudžia mygtuką „+“; 3. Sukuria naują įrašą grupėje; |
| Po-sąlyga: | Įrašas sėkmingai pridėtas į grupę. Visiems grupėms nariams siunčiamas pranešimas apie naują įrašą |



17 pav. „Valdyti grupės narius“ užduoties modelis

| Specifikacijų lentelė № 9 | |
| --- | --- |
| Panaudojimo atvejis | Valdyti grupės narius |
| Tikslas | Reguliuoti narius, kurie gali prisijungti prie grupes, dalyvauti iššūkiuose, rašyti komentarus |
| Vartotojas/Aktorius: | Grupės moderatorius |
| Prieš - sąlyga: | Grupėje turi būti bent 1 narys (išskyrus moderatoriaus) |
| Sužadinimo sąlyga: | Mygtuko „+“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Moderatorius užeina į grupę; 2. Spaudžia į mygtuką „ ... „ ; 3. Pasirenka narį šalinimui ir rašo šalinimo priežastį; |
| Po-sąlyga: | Pasirinktas vartotojas sėkmingai išbrauktas iš grupes narių |



18 pav. „Tvarkyti įrašus ir komentarus“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 10 | |
| Panaudojimo atvejis | Tvarkyti įrašus ir komentarus |
| Tikslas | Trinti nereikalinga informaciją, spamą, reklamą ir t.t |
| Vartotojas/Aktorius: | Grupės moderatorius |
| Prieš - sąlyga: | Moderatorius užeina į grupę |
| Sužadinimo sąlyga: | Mygtuko „Redaguoti“ arba „Trinti“ paspaudimas |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Moderatorius užeina į grupę/ iššūkį; 2. Spaudžia mygtuką „Redaguoti“ arba „Trinti“ 3. Redaguoja komentarą arba ištrina. |
| Po-sąlyga: | Įrašas arba komentaras sėkmingai ištrintas arba pakoreguotas |



19 pav. „Kurti iššūkį“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė № 11 | |
| Panaudojimo atvejis | Kurti iššūkį |
| Tikslas | Motyvuoti narius kažką daryti |
| Vartotojas/Aktorius: | Grupės kūrėjas |
| Prieš - sąlyga: | - |
| Sužadinimo sąlyga: | Mygtuko „Pridėti naują iššūkį“ |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Grupės kūrėjas užeiną į savo grupę; 2. Paspaudžia mygtuką „Pridėti naują iššūkį“; 3. Įveda informaciją apie iššūkį (pradžios data, pabaigos data, pavadinimas, taškai ...); 4. Spaudžia mygtuką „Išsaugoti“; |
| Po-sąlyga: | Naujas iššūkis sėkmingai išsaugotas ir pridėtas į sąrašą. Visi grupės nariai gauna pranešimą apie naują iššūkį. |



20 pav. „Uždaryti iššūkį“ užduoties modelis

| Specifikacijų lentelė № 12 | |
| --- | --- |
| Panaudojimo atvejis | Uždaryti iššūkį |
| Tikslas | Jeigu iššūkis baigtas arba atšauktas jis turi būti uždarytas |
| Vartotojas/Aktorius: | Grupės kūrėjas |
| Prieš - sąlyga: | Iššūkis turi egzistuoti |
| Sužadinimo sąlyga: | Mygtuko „Uždaryti“ |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Grupės kūrėjas užeiną į savo grupę; 2. Paspaudžia mygtuką „Uždaryti“; 3. Įveda informaciją apie iššūkio uždarymą ; |
| Po-sąlyga: | Iššūkio statusas pasikeičia į „Baigtas“ |

# Funkciniai ir nefunkciniai reikalavimai kuriamai PS

Kuriant naują informacinę sistemą ar diegiant esamą visada būtina nustatyti nefunkcinius reikalavimus savo sistemai. Reikalavimus programinės įrangos produktams ar informacinėms sistemoms galima suskirstyti į dvi dideles grupes. Tai funkciniai reikalavimai (apibūdinantys, ką reikia įdiegti produkte ar sistemoje, įskaitant veiksmus, kuriuos vartotojai turi atlikti sąveikaudami su jais) ir nefunkciniai reikalavimai (apibūdinantys, kaip sistema ar programinės įrangos produktas turi veikti ir kokias jo savybes ar charakteristikas turi atlikti). Žemiau leneliuose 10-11 aprašomi funkciniai ir nefunkciniai reikalavimai kuriamai sistemai.

10 lentelė Funkciniai reikalavimai

| Nr. | Formulote | Priėmimo kriterijai |
| --- | --- | --- |
| 1. | Vartotojas turi turėti galimybe prisijungti prie savo paskyros | 1. Vartotojas gali įvesti savo el. paštą; 2. Vartotojas turi gauti laišką su autentifikacijos raktų; 3. Įvedus šį raktą vartotojui turi būti suteikta prieiga prie paskyros; |
| 2. | Vartotojas turi turėti galimybe peržiūrėti savo grupes ir iššūkius | 1. Prisijungęs vartotojas mato savo grupės; 2. Pasirinkus grupę vartotojas mato iššūkius kuriose dalyvauja; |
| 3. | Vartotojui turi būti suteikta galimybe kurti ir komentuoti įrašus | 1. Vartotojas turi galimybę sukurti naują įrašą iššūkyje; 2. Grupės kūrėjas turi turėti galimybę sukurti naują įrašą grupėje; 3. Vartotojas gali rašyti komentarus po įrašų arba atsakyti į komentarus; |
| 4. | Vartotojas turi galimybė įstoti į grupę arba palikti ją | 1. Vartotojas gali prisijungti prie viešos grupės; 2. Vartotojas gali pateikti prašymą įstoti į privačią grupę; 3. Vartotojas gali palikti grupę; |
| 5. | Grupės kūrėjas gali kurti iššūkius ir valdyti juos | 1. Grupės kūrėjas gali kurti naujus iššūkius savo grupėje; 2. Grupės kūrėjas gali pradėti, sustabdyti, baigti iššūkį; |
| 6. | Vartotojas gali kurti grupę | 1. Prisijungęs vartotojas gali kurti naują privačią grupę; 2. Prisijungęs vartotojas gali kurti naują viešoji grupę; |
| 7. | Grupės nariai gali peržiūrėti savo rezultatus grupėje | 1. Už pabaigtus iššūkius vartotojas gauna taškus; 2. Grupėje vartotojas mato savo ir grupės narių taškų skaičių; 3. Savo paskyroje vartotojas mato visų savo surinktų taškų sumą; |

lentelė Nefunkciniai reikalavimai

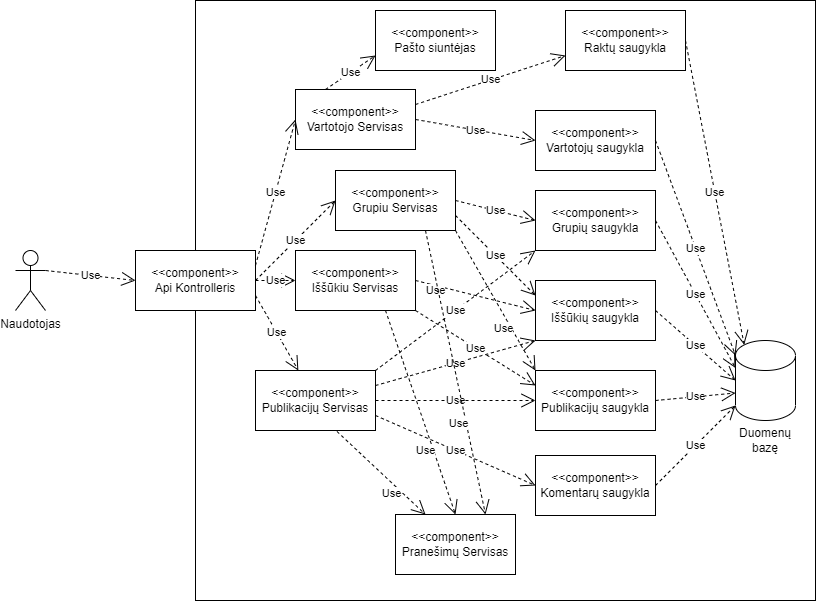
| Nr. | Formulote | Priėmimo kriterijai |
| --- | --- | --- |
| 1. | Vartotojui turi greitai gauti el. laišką su autentifikavimo raktų | Laiškas turi atvykti ne ilgiau kaip per |
| 2. | Mobilioji aplikacija turi veikti „Android“ ir „iOS“ operacinėse sistemose | Sistema tvarkingai veikia visose populiariausiose operaciniuose sistemose |
| 3. | Sistema turi palaikyti struktūrizuotą duomenų bazę „H2“ | Duomenų bazė sukonfigūruota ir surišta su aplikacijos back-end. |
| 4. | Sistema turi vaizduoti datas „YYYY-MM-DD“ formatu | Kuriamoje sistemoje laikas ir data vaizduojamas nurodytu formatu |
| 5. | Užtikrinti naudotojų duomenų saugumą | Sistema turi nepažeisti GDPR (General Data Protection Regulation).  Atsargine duomenų kopija atnaujinama kiekviena savaitė. |
| 6. | Po sistemos atnaujinimo failai neturi būti prarasti | Prieš sistemos atnaujinimą visi duomenys bei konfigūraciniai failai turi būti išsaugoti. |

# Sistemos architektūra

# Sistemos serverio architektūra

Ant 21 pav. parodyta kuriamos sistemos serverio komponentų schema. Sistemos serveryje iš viso yra 13 komponentų. Kiekvieno komponento funkcijos aprašytos žemiau.

* Api kontroleris - tai restfull api kuris naudojamas komunikacijai su serveriu naudojant http protokolą.
* Vartotojo servisas - atsako už vartotojų valdymą.
* Pašto siuntėjas - siunčia laiškus į el. paštus
* Grupių servisas - atsako už grupių valdymą, t.y narių tvarkymą, grupių ir iššūkių kūrimą.
* Iššūkių servisas - atsako už iššūkių valdymą, naujienų juostos atvaizdavimą, valdo iššūkių būseną
* Publikacijų servisas - naudojamas naujų publikacijų kūrimui ir publikavimui, naujienų lentos agregavimui
* Pranešimų servisas - naudojamas pranešimams vartotojui kurti ir siųsti.
* Raktų saugykla - komunikuoja su DB ir naudojama saugos raktams validuoti ir saugoti.
* Vartotojų saugykla - naudojamas komunikavimui su duomenų bazė. Suteikia galimybę gauti, saugoti, atnaujinti vartotojus.
* Iššūkių saugykla - naudojamas komunikavimui su duomenų bazė. Suteikia galimybę gauti, saugoti, atnaujinti iššūkius.
* Publikacijų saugykla - naudojamas komunikavimui su duomenų bazė. Suteikia galimybę gauti, saugoti, atnaujinti publikacijas.
* Komentarų saugykla - naudojamas komunikavimui su duomenų bazė. Suteikia galimybę gauti, saugoti, atnaujinti komentaras.

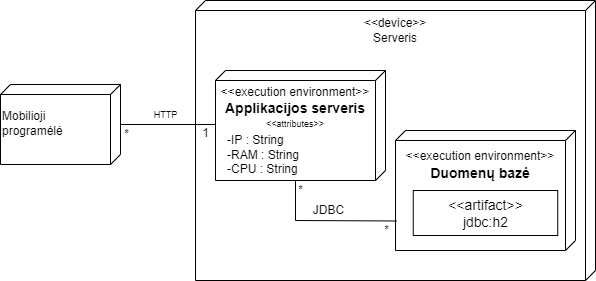


21 pav. Sistemos serverio komponentai

# Sistemos įgyvendinimo architektūra

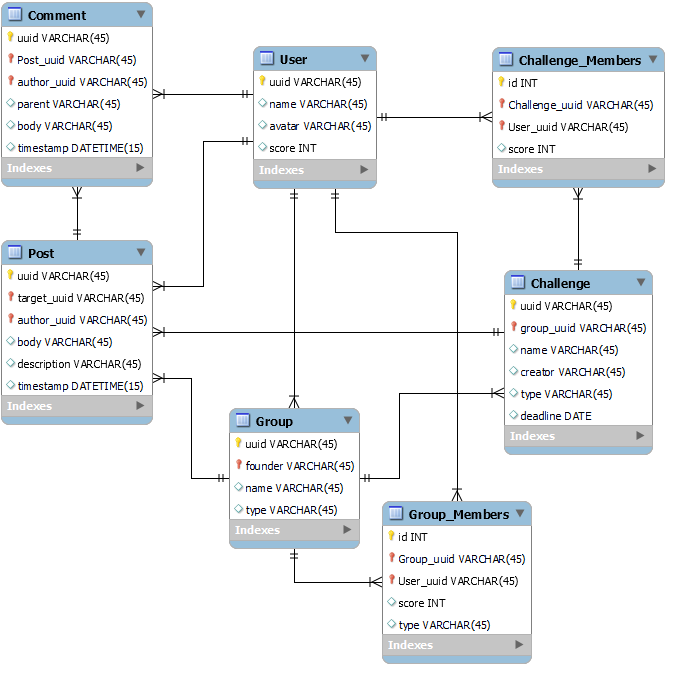
Sistema bus sudaryta iš trejų pagrindinių komponentų - mobilioji aplikacija, vidinės sistemos ir duomenų bazės. Mobilioji aplikacija komunikuoja su serveriu per internetą, http protokolą. „jdbc“ - tai sąsaja, kuri "Java" naudoja komunikacijai su duomenų baze. Aplikacijos serveris yra apibūdinamas tremtis charakteristikomis: IP, RAM, CPU:

* IP adresas – unikalus numeris, identifikuojantis jūsų kompiuterį tinkle ir internete;
* RAM - operatyvioji atmintis;
* CPU - centrinis procesorius;



22 pav. Sistemos įgyvendinimo architektūra

# Duomenų bazės struktūra ir ryšiai



Duomenu baze yra sudaryta is tokiu lenteliu kaip: User – lentele kuri saugo sistemos naudotojus; Group – lenetele saugo sistemoje sukurtas grupes; Challenge – lentele kuri saugo sukurtus issukius; Post – lentele kuri saugo sukurtas publikacijas, target\_uuid raktas saugo vieta kur buvu sukurta publikacija, tai gali buti kaip grupe taip ir issukis; Comment – lentele sauganti naudotoju parasytus komentarus; Group\_Members – lentele saugo grupiu nariu sarasa; Challenge\_Members – lentele saugo naudotoju sarasa dalivaujanciu issukije;

# Testavimas

# Testavimo metodikos pasirinkimas

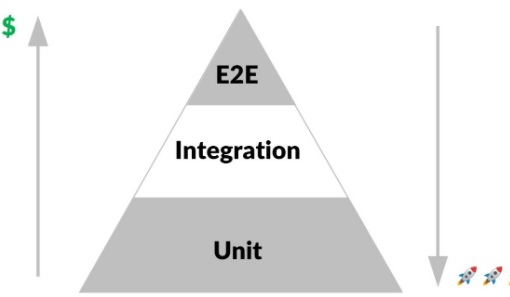
Mobiliųjų programėlių testavimas – tai procesas, kurio metu tikrinamas mobiliesiems įrenginiams sukurtos programinės įrangos funkcionalumas, patogumas naudoti ir suderinamumas. Testavimas gali būti rankinis arba automatinis.

*Rankinio testavimo* metu testuotojai atlieka testus rankiniu būdu, nenaudodami jokių automatizavimo įrankių. Rankinis testavimas yra žemo lygio ir paprasčiausias testavimo būdas, nereikalaujantis daug papildomų žinių. Tačiau prieš automatizuodami bet kurios programos testavimą, pirmiausia turime atlikti keletą rankinių testų. Rankinis testavimas reikalauja didelių pastangų, tačiau be jų negalėsime įsitikinti, kad automatizavimas iš esmės įmanomas. Vienas iš pagrindinių testavimo principų sako: 100% automatizavimas neįmanomas. Todėl rankinis testavimas yra būtinas.

*Automatizuotas testavimas* apima specialios programinės įrangos naudojimą, kad būtų galima kontroliuoti testų vykdymą ir palyginti laukiamą faktinį programos rezultatą. Šio tipo testavimas padeda automatizuoti užduotis, kurios dažnai pasikartoja.

Kai kurios testavimo užduotys, pvz., žemo lygio regresijos testavimas, gali reikalauti daug darbo ir daug laiko, jei jos atliekamos rankiniu būdu. Be to, rankinis testavimas gali būti nepakankamai efektyvus, kad būtų galima rasti kai kurių klasių klaidas. Tokiais atvejais automatizavimas gali padėti sutaupyti projekto komandos laiką ir pastangas.

Žemiau (23 pav) pateikta testavimo piramidė pagal kurį rašysiu testus savo mobiliajai programėlei. Piramides užduotis – sugrupuoti testus pagal skirtingus detalumo lygius. Unit testų turėtų būti daugiausia (apie 80%), jie greičiausiai vykdomi. Integracinių testų apie 10%, jie vykdomi lečiau ir naudoja daugiau resursų. Piramides viršuje – end to end testavimas, čia testų skaičius turėtų būti kuo mažesnis.



23 pav. Testavimo piramidė

Taigi, išanalizavus testavimo metodikas, buvo nuspręsta sukurtą mobiliąją aplikaciją testuoti automatizuotu būdu – rašant testus. Sukūrus automatizuotus testus, juos galima bet kada paleisti iš naujo, jie paleidžiami ir vykdomi greitai ir tiksliai. Taigi, jei reikia dažnai kartoti bandymus, automatizavimo vertės supaprastinant projekto priežiūrą ir sumažinant jo išlaidas negalima pervertinti. Galų gale, net ir minimalūs pataisymai ir kodo pakeitimai gali sukelti naujų klaidų. Taip pat programėle reikalauja ir rankinio testavimo, kai kurie atvejai bus testuojami rankiniu būdu, pavyzdžiui, kiek laiko užima registracijos laiško siuntimas į el. paštą.

# Automatizuotas testavimas

Savo programėles kūrimui aš naudojau bandomosios plėtros (TDD) metodiką. Test-driven development arba testavimu pagrįstas kūrimo procesas – tai programinės įrangos kūrimo metodika, kuri remiasi trumpų kūrimo ciklų kartojimu: iš pradžių parašomas testas, apimantis norimą pakeitimą, po to parašomas programos kodas, įgyvendinantis norimą sistemos elgesį, o tada atliekamas parašyto kodo pertvarkymas, nuolat tikrinant, ar visi testai buvo išlaikyti. Buvo parašyti dvejų tipų automatiniai testai, skirti išbandyti mobiliąją programėlę.

# Modulinis testavimas

Modulinis testavimas (angl. unit testing) - tai yra mažo nepriklausomo kodo bloko testavimo metodas. Mažo bloko kodas daugeliu atvejų bus funkcija. Žodis nepriklausomas reiškia, kad jis nepriklauso nuo kitų projekto kodo dalių. Iš viso buvo parašyta 40 unit testų, žemiau pateikiami trijų testų pavyzdžiai.

12 lentelė. Modulinis testas №1

| № | Pavadinimas: Grupės narys turi turėti galimybę prisijungti prie aktyvaus grupės iššūkio |
| --- | --- |
| 1 | @Test  public void group\_member\_should\_be\_able\_to\_join\_active\_group\_challenge() {  User founder = User.initialize("mock@");  final UUID founderUuid = founder.getUuid();  final UUID newUserUuid = UUID.randomUUID();  Group group = Group.initialize(founderUuid, "Name", AccessType.PUBLIC)  .addGroupMember(newUserUuid);  groupRepository.save(group);  Challenge challenge = Challenge.initialize(founderUuid, group.getUuid(), "name", LocalDateTime.now(), AccessType.PUBLIC);  challengesRepository.save(challenge);  challengesService.joinChallenge(newUserUuid, challenge.getUuid());  Challenge savedChallenge =  challengesRepository.findByUuid(challenge.getUuid()).get();  assertThat(savedChallenge.getParticipants(), hasItem(newUserUuid));  } |
| Aprašymas: Iš pradžių kuriama grupę su nariais, kuriamas grupės iššūkis ir bandoma grupės nariui prisijungti prie iššūkio. | |

13 lentelė. Modulinis testas №2

| № | Pavadinimas: Privačios grupės kūrėjas turėtų turėti galimybę atmesti prašymą prisijungti |
| --- | --- |
| 2 | @Test  public void private\_group\_founder\_should\_be\_able\_to\_decline\_join\_request() {  User founder = User.initialize( "mock@");  final UUID founderUuid = founder.getUuid();  Group group = Group.initialize(founderUuid, "Name", AccessType.PRIVATE);  usersRepository.save(founder);  groupRepository.save(group);  final User mock = User.initialize("mocker@");  groupsService.tryToJoinGroup(mock.getUuid(), group.getUuid());  groupsService.declineJoinRequest(founderUuid, group.getUuid(), mock.getUuid());  Group savedGroup = groupRepository.findByUuid(group.getUuid()).get();  var members = savedGroup.getMembers();  assertThat(members, not(hasItem(mock.getUuid())));  var requests = savedGroup.getPendingRequests();  assertThat(requests, hasSize(0));  } |
| Aprašymas: Kuriamas naujas naudotojas, privati grupė, grupės kūrėjas, naudotojas bando prisijungti prie privačios grupės. Grupės kūrėjas atšaukia prašymą įstoti į grupę. | |

14 lentelė. Modulinis testas №3

|  |  |
| --- | --- |
| № | Pavadinimas: Grupės narys turi turėti galimybę gauti aktyvių grupės iššūkių sąrašą |
| 3 | @Test  public void group\_member\_should\_be\_able\_to\_get\_list\_of\_active\_group\_challenges() {  User founder = User.initialize("mock@");  final UUID founderUuid = founder.getUuid();  Group group = Group.initialize(founderUuid, "Name", AccessType.PRIVATE);  groupRepository.save(group);  Challenge challenge = Challenge.initialize(founderUuid, group.getUuid(), "name", LocalDateTime.now(), AccessType.PUBLIC);  groupsService.createNewChallenge(challenge);  var availableChallenges = groupsService.getAvailableChallenges(founderUuid, group.getUuid());  assertThat(availableChallenges, hasSize(1));  } |
| Aprašymas: Kuriamas naujas iššūkis ir yra įsitikinama kad jį gali matyti kiti grupės nariai. | |

# Integraciniai testai

Integravimo testavimas – tai testavimo tipas, kai programinės įrangos moduliai logiškai sujungiami ir testuojami kaip grupė. Paprastai programinės įrangos produktą sudaro keli programinės įrangos moduliai, parašyti skirtingų programuotojų. Integracinio testavimo tikslas – nustatyti šių programinės įrangos modulių sąveikos klaidas ir visų pirma siekiama patikrinti duomenų mainus tarp šių modulių. Iš viso buvo parašyta 7 integracinių testų, žemiau pateikiami dvejų testų pavyzdžiai.

15 lentelė. Integracinis testas №1

| № | Pavadinimas: Prisijungimas prie grupės, prie iššūkio ir vertinamo įrašo kūrimas |
| --- | --- |
| 1 | @Test  public void joining\_group\_then\_joining\_challenge\_and\_creating\_rated\_post() throws  Exception {  mockMvc.perform(  get("/api/groups/" + group1.getUuid() + "/join?userUuid=" + user2.getUuid()))  .andExpect(status().isOk());  // requesting to join public challenge  mockMvc.perform(get("/api/challenges/" + challenge1.getUuid() +   "/join?userUuid=" + user2.getUuid()))  .andExpect(status().isOk());  // creating ranked post  final UUID newPostUuid = UUID.randomUUID();  final Post newPost = Post.create(newPostUuid, user2.getUuid(),  challenge1.getUuid(), "newPost", "description", "image", 0);  mockMvc.perform(post("/api/challenges/createRatedPost")  .contentType(MediaType.APPLICATION\_JSON)  .content(gson.toJson(newPost)))  .andExpect(status().isOk());  final Challenge savedChallenge = challengesRepository.findByUuid(challenge1.getUuid()).get();  final Integer userScore = savedChallenge.getScoreboard().get(user2.getUuid());  assertThat(userScore, greaterThan(0));  final var savedPost = postsRepository.findByUuid(newPostUuid);  assertThat(savedPost.isPresent(), equalTo(true));  } |
| Aprašymas: Siunčiama užklausą įstoti į grupę, po to bandoma prisijungti prie iššūkio ir sukurti naują įrašą. | |

16 lentelė. Integracinis testas №2

| № | Pavadinimas: Prisijungti prie grupės, tada prisijungti prie iššūkio ir komentuoti po įrašu |
| --- | --- |
| 2 | @Test  public void joining\_group\_then\_joining\_challenge\_and\_comment\_under\_post() throws Exception {  // requesting to join public group  mockMvc.perform(  get("/api/groups/" + group1.getUuid() + "/join?userUuid=" + user2.getUuid()))  .andExpect(status().isOk());  // requesting to join public challenge  mockMvc.perform(get("/api/challenges/" + challenge1.getUuid() +  "/join?userUuid=" + user2.getUuid()))  .andExpect(status().isOk());  // write comment under post  final Comment newComment = Comment.initialize(user2.getUuid(),  post1.getUuid(), "Some random comment");  mockMvc.perform(post("/api/feed/comments")  .contentType(MediaType.APPLICATION\_JSON)  .content(gson.toJson(newComment)))  .andExpect(status().isOk());  final var savedComments = commentsRepository.findByTarget(post1.getUuid());  assertThat(savedComments, hasSize(1));  } |
| Aprašymas: Bandoma įstoti į grupę, po to į iššūkį ir pakomentuoti iššūkyje įrašą. | |

# Rankinis testavimas

# End-to-end testavimas

Visapusiškas testavimas, taip pat žinomas kaip „End-to-end“ arba E2E, yra testavimo procesas, kuris išsamiai imituoja vartotojo aplinką. Bandymo metu jis imituoja: pelės paspaudimus, mygtukų paspaudimus, formų pildymą, puslapių perėjimą ir kitus elgesio veiksnius. Šio testavimo esmė yra išsiaiškinti, ar galutiniam klientui skirta programa veikia taip, kaip buvo apskaičiuota iš pradžių. End-to-end testavima nuspendžiau testuoti rankinių būdų ...............

# Testavimo išvados

Taigi, iš viso buvo parašyta 77 automatizuotu testų „*SpringBoot“* aplinkoje. Taip pat buvo atlikta X testų rankiniu būdu. Iš gautų testavimo rezultatų, galima teigti, kad sistemos funkcionalumas veikia be klaidų, programėlę praeina visus testus, vartotojo sąsaja atrodo vienodai „*Android*“ ir „*iOS*“ operacinėse sistemose. Ištestavus funkcinius ir nefunkcinius reikalavimus, pagal rezultatus, mobilioji programėle atitinka aprašytus normatyvus.

# Išvados ir siūlymai

1. Išanalizavus žmonių poreikį naudoti tikslų įvairiausias motyvuojančios programėles bei statistiką iš programėlių parduotuves (Play Marker), buvo nustatyta kad žmonės aktyviai naudoja programėles sveikatai, sportui, mitybai ir kitus programėles, kurie padeda formuoti sveikus įpročius ir sekti savo tikslą. Todėl buvo nuspręsta sukurti mobiliąją aplikaciją, kurios idėja yra paversti grupines iššūkius žaidimų ir pridėti konkurencinį aspektą. Tyrimai rodo, kad užduočių atlikimas žaidimo forma gali padidinti motyvaciją ir padėti laiku atlikti užduotis.

2. Atlikus mano aplikacijos palyginimus su rinkoje siūlomais alternatyvomis – „Microsoft To Do“, Challenges – Complete, Get Fit“, „Habitica“ , paaiškejo, kokias privalumus ir skirtumus turi sukurta sistema. Ne visas alternatyvas leidžia kurti grupės iššūkiams ir lyginti grupės narių pasiekimus. Pagrindinis siūlomų rinkoje alternatyvų privalumas yra tai, kad jie prieinami įvairiose operacinėse sistemose ir turi daugybę funkcijų Mobilioji aplikacija skiriasi nuo pateiktų alternatyvų, tai yra nemokoma programėlę, savotiškas socialinis tinklas, kuriame žmonės galės kurti ir vienytis siekdami savo tikslų.

3. Darbo metu buvo tiriami mobiliųjų aplikacijų kūrimo teoriniai aspektai bei aplikacijų kūrimo technologijos. Išanalizavęs mobiliųjų aplikacijų kūrimo technologijas, pasirinkau „Flutter“. „Flutter“ naudoja „Dart“ programavimo kalbą ir yra hibridinis programėlių kūrimo įrankis. Pagrindinės priežastys, kodėl pasirinkau „Flutter“, yra geras greitis, galimybė sukurti vieną programėlę visiems išmaniųjų įrenginių operacinėms sistemoms ir tai, kad jau turiu patirties kuriant mobiliąją programėlę su šia technologija. Taip pat atlikus backend dalies analizę buvo nustatyta, kad populiariausia programėlių kūrimo aplinka yra „Spring“ kartu su Java programavimo kalbą, kuri ir panaudojau kuriant programėlę.

4. Projektavimo metu buvo nustatyti vartotojo pageidavimai būsimai sistemai, kokias funkcijas turės atlikti kuriama programinė įranga bei kokie yra sistemos apribojimai. Tam tikslui buvo apibrėžti 7 funkciniai ir 6 nefunkciniai reikalavimai bei nustatyti priėmimo kriterijai. Sukurta mobilioji programėlė atitinka nurodytus reikalavimus, tai buvo patikrinta testavimo metu. Reikalavimų specifikacijoje aprašyti funkciniai ir nefunkciniai reikalavimai sistemai yra realizuoti programiškai.

5. Šio projekto tikslas buvo suprojektuoti ir realizuoti bendrų tikslų siekimo ir rezultatų dalijimosi sistemą mobiliesiems įrenginiams. Atlikus dalykines srities analizę buvo parinktos tinkamos technologijos sistemai realizuoti. Projektui įgyvendinti buvo nuspręsta sukurti mobiliąją aplikaciją populiariausioms operacinėms sistemoms – Android ir IOS. Sistemos projektavimui buvo sukurta panaudos atvejų diagrama, serverio architektūros diagrama, sistemos architektūros įgyvendinimo diagrama ir fizinė duomenų bazės diagrama. Šio baigiamojo darbo rezultatas – pilnai suprojektuota ir realizuota mobilioji aplikacija *IŠŠŪKIS* skirta *Android* ir IOS operacinėms sistemoms.

6. Išanalizavus testavimo principus buvo išaiškinta, kad visų testų automatizavimas neįmanomas ir būtina taip pat ir rankinę patikrą. Todėl sistemos testavimui buvo naudoti du metodai: automatizuotas testavimas ir rankinis. Mobiliajai programėlei testuoti buvo parašyta 77 automatizuotų testų: 70 modulinių testų (angl. unit test) ir 7 integracinių testų. Rankinių būdų buvo padarytas end-to-end programėlės testavimas, kuris patikrina visą programėlę nuo pradžios iki galo kartu su jos integravimu su išorinėmis sąsajomis. Mobilioji programėlė IŠŠŪKIS sėkmingai praėjo visus automatizuotus ir rankinių būdu atliktus testus.

**Siūlymai**

1. Aplikacijos talpinimas *Play Market* ir *AppStore* aplikacijų parduotuvėse. Aplikacijos registracija *Play Market* kainuoja 25$ (vienkartinis mokestis), o registracija *AppStore* 99$/per metus.

2. Kad kitų šalių vartotojai galėtų naudotis programa, būtina lokalizuoti jos turinį, sąsają, APK failus ir puslapį Google Play ir AppStore programėlių parduotuvėse.

# Literatūros sąrašas

1. AppAnnie. (2022 m. sausio 13 d.). *State of Mobile 2022*. Nuskaityta iš https://www.appannie.com/en/go/state-of-mobile-2022/
2. Changiz Hosseini, O. H. (2021). An experimental study on the effects of gamification on task performance. *Procedia Computer Science*, 1004-1005.
3. Clear, J. (2018). Motivation: The Scientic Guide on How to Get and Stay Motivated.
4. E. Colucci, S. N. (2022). COVID-19 lockdowns’ effects on the quality of life, perceived health and well-being of healthy individuals: A longitudinal comparison of pre-lockdown and lockdown states of well-being,. *Archives of Gerontology and Geriatrics*, 5-7.
5. Encarnação, R., Reuter, J., Dias, M. F., & Amorim, M. (2021). Gamification as a driver of motivation in the organizations: A Bibliometric Literature Review. *Association for Computing Machinery*, 168-172.
6. FlexJobs. (2020). *FlexJobs Survey Finds Employees Want Remote Work Post-Pandemic*. Nuskaityta iš flexjobs.com: https://www.flexjobs.com/blog/post/flexjobs-survey-finds-employees-want-remote-work-post-pandemic/
7. Flutter. (2022 m. 01 22 d.). *Showcase*. Nuskaityta iš Flutter Corporation Tinklalapis: https://flutter.dev/showcase
8. Fokkens, D. (2021). “IF IT’S NOT TRACKED, IT DOESN’T COUNT!” – EXPLORING DEPENDENCY EFFECTS REGARDING. 24.
9. Gartner. (2021). *Gartner Says Worldwide Smartphone Sales Grew 10.8% in Second Quarter of 2021*. Nuskaityta iš https://www.gartner.com/en/newsroom/press-releases/2021-09-01-2q21-smartphone-market-share
10. IBM. (2021). *https://www.ibm.com*. Nuskaityta iš SQL vs. NoSQL Databases: What's the Difference: https://www.ibm.com/cloud/blog/sql-vs-nosql
11. Intexsoft. (2020). *JAVA FRAMEWORKS FOR WEB APPLICATIONS: WHAT TO USE IN 2020*. Nuskaityta iš Indexsoft: https://intexsoft.com/blog/10-java-frameworks-for-web-applications-what-to-use-in-2020/
12. Iuliana-Tania ZLATCU, & CLODNITCHI, R. (2018). The Impact of Competition on Productivity. *Review of International Comparative Management* , 418-419.
13. Matters, 4. (2022). *Store Stats 2022*. Nuskaityta iš https://42matters.com/ios-apple-app-store-statistics-and-trends
14. Statista. (2020 m. April). *Change in remote work trends due to COVID-19 in the United States in 2020*. Nuskaityta iš https://www.statista.com: https://www.statista.com/statistics/1122987/change-in-remote-work-trends-after-covid-in-usa/
15. Statista. (2021). *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021*. Nuskaityta iš https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/
16. Statista. (2021). *Global digital population as of January 2021*. Nuskaityta iš Statista: https://www.statista.com/statistics/617136/digital-population-worldwide/
17. Statista. (2021). *Most popular app categories worldwide during 3rd quarter 2020, by usage reach*. Nuskaityta iš https://www.statista.com: https://www.statista.com/statistics/1252652/top-apps-categories-by-global-usage-reach/
18. Statista. (2021). *Most popular Google Play app categories as of 1st quarter 2021, by share of available apps*. Nuskaityta iš Statista: https://www.statista.com/statistics/279286/google-play-android-app-categories/
19. Statistica. (2021). *Average number of new Android app releases via Google Play per month from March 2019 to November 2021*. Nuskaityta iš https://www.statista.com/statistics/1020956/android-app-releases-worldwide/
20. Stock, S. B. (2021). Going outdoors, neighbourhood satisfaction and mental health and wellbeing during a COVID-19 lockdown: A fixed-effects analysis. 16-17.
21. VersionOne. (2021). *Versionone 11th annual state of agile report*. Nuskaityta iš https://digital.ai/resource-center/analyst-reports/state-of-agile-report

# Priedai