

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

*Andrej Latyš*

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](#_Toc103163453)

[1. Analitinė dalis 11](#_Toc103163454)

[1.1. Idėjos pagrindimas 11](#_Toc103163455)

[1.2. Alternatyvos 12](#_Toc103163456)

[1.3. Kuriamos programėlės techninė analizė 15](#_Toc103163457)

[1.3.1. Mobiliųjų aplikacijų kūrimo karkaso pasirinkimas 15](#_Toc103163458)

[1.3.2. Išsami „Flutter“ technologijos apžvalga 19](#_Toc103163459)

[1.3.3. Vidinės programos dalies technologijų pasirinkimas 21](#_Toc103163460)

[1.3.4. Išsami vidinės dalies apžvalga 23](#_Toc103163461)

[2. Projektinė dalis 26](#_Toc103163462)

[2.1. Sistemos naudotojai 26](#_Toc103163463)

[2.2. Panaudos atvejų diagrama 26](#_Toc103163464)

[2.3. Funkciniai ir nefunkciniai reikalavimai kuriamai PS 38](#_Toc103163465)

[3. Sistemos architektūra 40](#_Toc103163466)

[3.1. Sistemos serverio architektūra 40](#_Toc103163467)

[3.2. Sistemos įgyvendinimo architektūra 42](#_Toc103163468)

[3.3. Duomenų bazės struktūra ir ryšiai 43](#_Toc103163469)

[5. Testavimas 44](#_Toc103163470)

[5.1. Testavimo metodikos pasirinkimas 44](#_Toc103163471)

[5.2. Automatizuotas testavimas 45](#_Toc103163472)

[5.2.1. Modulinis testavimas 45](#_Toc103163473)

[5.2.2. Integraciniai testai 47](#_Toc103163474)

[5.3. Rankinis end-to-end testavimas 48](#_Toc103163475)

[5.4. Testavimo išvados 52](#_Toc103163476)

[Išvados ir siūlymai 53](#_Toc103163477)

[Literatūros sąrašas 55](#_Toc103163478)

[Priedai 57](#_Toc103163479)

**Iliustracijų sąrašas**

[1 pav. Mobiliųjų programėlių naudojamumas pagal kategorijas (Statista, 2021) 11](#_Toc103165733)

[2 pav. Programėlės “Challenges” vartotojo sąsaja 14](#_Toc103165734)

[3 pav. Programėlės „Microsoft To Do“ vartotojo sąsaja 14](#_Toc103165735)

[4 pav. Programėlės „Habitica“ vartotojo sąsaja 14](#_Toc103165736)

[5 pav. Populiariausi hibridinių ir žiniatinklio mobiliųjų programėlių kūrimo karkasai (Statista, 2021). 16](#_Toc103165737)

[6 pav. „Flutter“ architektūros modelis 20](#_Toc103165738)

[7 pav. „Flutter“ sąveika su „native“ komponentais 20](#_Toc103165739)

[8 pav. „Spring“ karkaso architektūrinis modelis 23](#_Toc103165740)

[9 pav. Panaudos atvejų diagrama 27](#_Toc103165741)

[10 pav. „Užregistruoti paskyrą“ užduoties modelis 28](#_Toc103165742)

[11 pav. „Prisijungti arba išeiti iš grupės“ užduoties modelis 29](#_Toc103165743)

[12 pav. „Prisijungti arba išeiti iš iššūkio“ užduoties 30](#_Toc103165744)

[13 pav. „Kurti įrašus“ užduoties modelis 31](#_Toc103165745)

[14 pav. „Palikti komentarą“ užduoties modelis 32](#_Toc103165746)

[15 pav. „Peržiūrėti iššūkius“ užduoties modelis 33](#_Toc103165747)

[16 pav. „Kurti įrašus grupėje“ užduoties modelis 34](#_Toc103165748)

[17 pav. „Valdyti grupės narius“ užduoties modelis 35](#_Toc103165749)

[18 pav. „Tvarkyti įrašus ir komentarus“ užduoties modelis 36](#_Toc103165750)

[19 pav. „Kurti iššūkį“ užduoties modelis 37](#_Toc103165751)

[20 pav. „Uždaryti iššūkį“ užduoties modelis 38](#_Toc103165752)

[21 pav. Sistemos serverio komponentai 41](#_Toc103165753)

[22 pav. Sistemos įgyvendinimo architektūra 42](#_Toc103165754)

[23 pav. Duomenų bazės struktūra 43](#_Toc103165755)

[24 pav. Testavimo piramidė 44](#_Toc103165756)

[25 pav. Registracijos laukas 49](#_Toc103165757)

[26 pav. Registracijos laukas po email įvedimo 49](#_Toc103165758)

[27 pav. Nesėkmingos registracijos atvejis 50](#_Toc103165759)

[28 pav. Automatizuotų testų paleidimas 52](#_Toc103165760)

[29 pav. Registracijos langas 57](#_Toc103165761)

[30 pav. Feed langas 57](#_Toc103165762)

[31 pav. „Groups“ langas 57](#_Toc103165763)

[32 pav. Įrašai grupėje 57](#_Toc103165764)

[33 pav. „Challenges“ langas 58](#_Toc103165765)

[34 pav. „Create new post“ langas 58](#_Toc103165766)

[35 pav. Komentarai po įrašu 58](#_Toc103165767)

[36 pav. „Notifications“ langas 58](#_Toc103165768)

[37 pav. Naujo iššūkio kūrimas 59](#_Toc103165769)

[38 pav. Naujos grupės kūrimas 59](#_Toc103165770)

**Lentelių sąrašas**

[1 lentelė. Alternatyvių mobiliųjų programų palyginimas 13](#_Toc103163547)

[2 lentelė. Alternatyvų palyginimas pagal iškeltus sisteminius reikalavimus 15](#_Toc103163548)

[3 lentelė. Populiarių žiniatinklio programėlių kūrimo karkasų palyginimas 17](#_Toc103163549)

[4 lentelė. Populiarių hibridinių programėlių kūrimo karkasų palyginimas 18](#_Toc103163550)

[5 lentelė. Vidinių sistemų karkasų palyginimas 22](#_Toc103163551)

[6 lentelė. Naudotojo „Neprisijungęs naudotojas“ detalizavimas 26](#_Toc103163552)

[7 lentelė. Naudotojo „Vartotojas“ detalizavimas 26](#_Toc103163553)

[8 lentelė. Naudotojo „Moderatorius“ detalizavimas 26](#_Toc103163554)

[9 lentelė. Naudotojo „Grupės kūrėjas“ detalizavimas 26](#_Toc103163555)

[10 lentelė. Funkciniai reikalavimai 39](#_Toc103163556)

[11 lentelė Nefunkciniai reikalavimai 40](#_Toc103163557)

[12 lentelė. Modulinis testas Nr.1 45](#_Toc103163558)

[13 lentelė. Modulinis testas Nr.2 46](#_Toc103163559)

[14 lentelė. Modulinis testas Nr.3 46](#_Toc103163560)

[15 lentelė. Integracinis testas Nr. 1 47](#_Toc103163561)

[16 lentelė. Integracinis testas Nr. 2 48](#_Toc103163562)

[17 lentelė. Rankinis testavimas 49](#_Toc103163563)

**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**

Kuriama programėlė turi universalią panaudojimo sritį, dėl teminių apribojimų nebuvimo. Kuriama sistema turės socialinių tinklų charakteristikų, užduočių vykdymo ir vizualizacijos galimybes, turės motyvacinį poveikį vartotojui, naudojant žaidimifikacijos ir viešosios atskaitomybės principus. Geriausiai ši programa atitinka „gyvenimo būdo“ kategoriją, kuri užima 5% mobiliųjų programėlių rinkos „Google Play“ platformoje (Statista, 2021).

Socialinių tinklų idėja yra viešai priimta ir plačiai naudojama. Statistikos portalo „Statista“ 2021-ų 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 tai yra saugiau ir patogiau, be to dabar beveik visos kompanijos suteikia galimybę dirbti nuotoliniu būdu, jei tai yra įmanoma. Tyrimai rodo, kad žmonės, dirbantys namuose, yra laimingesni, nes patiria mažiau streso. Tyrimas, kurį 2020m. atliko „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, kuriomis galima stebėti ir vizualizuoti vykdomo darbo eigą. Didžiausią rinkos dalį užima programuotojų įdiegtos “AGILE” ir “SCRUM” technologijos tokiose platformose kaip “Jira”, “Trello”, “ClickUp” ir daugelyje 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 ir laisvalaikį namie. Kaip rodo 2019–2021 m. izoliacijos per COVID-19 tyrimas, izoliacijos metu pablogėja bendra žmogaus savijauta (E. Colucci, 2022). Nepaisant to, net ir izoliacijoje žmonės stengiasi išlikti motyvuoti ir didinti savo produktyvumą nuolat ieškodami naujų būdų tapti sveikesniems. Tam žmonės naudoja sporto ir sveikos mitybos mobiliąsias programėles. Mano analizė rodo, 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 programelių kūrimui.

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

**Darbo uždaviniai:**

* 1. Išanalizuoti tikslų sekimo ir rezultatų dalijimosi sistemos poreikį.
  2. Išanalizuoti rinkoje 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 4 skyrių, išvados ir 10 priedų. Darbo apimtis – 56 puslapių, neskaitant priedų, tekste yra 38 paveikslų ir 17 lentelių. Rengiant darbą buvo panaudotas 21 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 technologiju pasirinkimas, išsami vidinės dalies apžvalga.

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

Testavimo dalyje atlikta 48 automatinių testų „Spring Boot“ aplinkoje, bei atlikta 12 testų rankinių būdų.

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

Bakalauro baigiamojo darbo įgyvendinimo programinis kodas pasiekiami adresu:

* [*https://github.com/lotro13/Bachelors/tree/master/Application*](https://github.com/lotro13/Bachelors/tree/master/Application)
* [*https://github.com/lotro13/Bachelors/tree/master/backend*](https://github.com/lotro13/Bachelors/tree/master/backend)

# Analitinė dalis

## Idėjos pagrindimas

Išmaniųjų telefonų paklausa stabiliai kyla jau ilgą laiką, ekspertai teigia, kad rinkos augimas išliks toks ir ateinančiais metais. „Gartner“ ataskaita rodo, kad išmaniųjų telefonų pardavimas visame pasaulyje 2021 m. išaugo 10,8% lyginant su 2020 m. (Gartner, 2021). Išmanieji telefonai naudojami bendravimui su kitais žmonėmis, pramogoms, informacijos perdavimui it t.t. Kiekvienas žmogus gali sugalvoti idėją ir sukurti mobiliąją programėlę, kuri bus viešai platinama įvairiose platformose. Tai leidžia integruoti programėles beveik į visus mūsų gyvenimo aspektus, tik svarbu suprasti, kiek tai naudinga ir sąmoninga. Remiantis nauja „App Annie State of Mobile 2022“ ataskaita, programų ir mobiliųjų programėlių naudojimas sparčiai auga visame pasaulyje. Ataskaita rodo, kad 2021 m. buvo parsisiųsta daugiau nei 232 milijardų mobiliųjų programėlių ir jomis besinaudojant praleista 3,8 trilijono valandų. Tyrime analizuotų rinkų vidutinis laikas, praleistas naudojantis programėlėmis 2021 m. viršijo 4 valandas ir 48 minutes per parą – 30 % daugiau nei 2019 m. (AppAnnie, 2022).

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

Chart, bar chart

Description automatically generated

1 pav. Mobiliųjų programėlių naudojamumas pagal kategorijas (Statista, 2021)

Kuriamo darbo idėja yra grupinių iššūkių su konkurenciniu aspektu kūrimas. Žmonės galės spręsti iššūkius, dalintis patirtimi ir nuotraukomis, gaudami už tai taškus. Tai padės jiems susipažinti su naujais žmonėmis, pradėti sveikiau valgyti, sportuoti, dažniau būti lauke ar tiesiog linksmintis, tokiu būdu palaikyti psichinę 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 pasiekti savo tikslą (Clear, 2018). Taip pat žaidimifikacija, tai yra užduočių atlikimas žaidimo forma gali pakelti motyvaciją. 2021-ais metais atliktas eksperimentinis tyrimas rodo, kad žaidimifikuotos riboto laiko užduotys buvo sėkmingai įvykdomos 56.25%, o be žaidimifikacijos tik 31%, be to užduočių atlikimo kokybė atitinkamai buvo vertinama 43% ir 25.75%. Remiantis šiais rezultatais, galima teigti, jog žaidimifikacija motyvuoja atlikti daugiau užduočių, be to padeda jas atlikti greičiau ir kokybiškiau (Changiz Hosseini, 2021). Galimybė padaryti žaidimą iš užduoties atlikimo taip pat gali būti naudojama grupiniame darbe.

## Alternatyvos

Jau daug metų žmonės internete dalijasi savo planais ir pasiekimais: socialiniuose tinkluose, asmeniniuose tinklaraščiuose ir forumuose. Tikslų ir uždavinių vizualizacijos idėja naudojant informacines sistemas taip pat nėra nauja – yra daugybė programų, kurios įvairiomis formomis padeda vartotojams padidinti produktyvumą. Štai keletas pavyzdžių, kuriuos galite rasti programėlių parduotuvėse:

* 1. **„Microsoft To Do“** – tai intelektualioji užduočių tvarkymo programa, palengvinanti dienų planavimą ir plano valdymą. Programa leidžia vartotojams valdyti savo užduotis savo išmaniajame telefone, planšetiniame bei asmeniniame kompiuteryje. Programa yra nemokoma.
  2. **„Challenges - Compete, Get Fit“** – tai sveikos gyvensenos palaikymo programėlė, leidžianti kurti komandas su kitais žmonėmis ir laikytis sveikos gyvensenos tikslų.
  3. **„Habitica“** – tai yra užduočių tvarkyklės paslauga, kurioje pagrindinis dėmesys skiriamas įpročių formavimui ir produktyvumo skatinimui, siekiant paversti darbą žaidimu. Su šia žiniatinklio programa vartotojai gali gauti ir apdovanojimų, ir bausmių, taip pat motyvuoti save socialinio tinklo pagalba. Pavyzdžiui, jei rūkote, herojus praranda sveikatos taškų, o jei darote sportinius pratimus, personažas uždirba atlygį.

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  Virš 5.5 tūkst. | 4.5  Virš 200 tūkst. | 4.3  Virš 21 tūkst. |
| Funkcijos | • Dalintis tikslais ir uždaviniais.  • Kurti grupę bendriems tikslams.  • Galimybė gauti apdovanojimus už pasiekimus.  • Siunčiami priminimai.  • Galima siųsti priminimus kontaktams. | • Kurti, redaguoti, ištrinti užduotis ir sąrašus.  • Dalintis tikslais ir uždaviniais.  • Papildomų užduočių, pastabų kūrimas, galimybė pridėti failus.  • Klasifikuoti darbų sąrašus pagal spalvas.  • Siunčiami priminimai.  •Užduočių komentarai.  • Galimybė 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.  • Siunčiami priminimai.  •Užduočių komentarai  •Produktyvumo analizė.  •Apdovanojimų sistema. |
| Sąsajos nuotraukos | A screenshot of a cell phone  Description automatically generated with medium confidence  2 pav. Programėlės “Challenges” vartotojo sąsaja | Graphical user interface, application  Description automatically generated  3 pav. Programėlės „Microsoft To Do“ vartotojo sąsaja | Graphical user interface, application  Description automatically generated  4 pav. Programėlės „Habitica“ vartotojo sąsaja |

Taigi, šiame skyriuje išanalizavome alternatyvias programas, jų funkcijas ir išvaizdą. „Microsoft To Do“ ir „Habitica“ šiuo metu yra populiariausios tikslų valdymo programos. Pagrindinis šių aplikacijų privalumas toks, kad jos pritaikytos įvairioms operacinėms sistemoms ir turi daugybę funkcijų. „Habitica“ tai naujas požiūris į planavimą kaip į žaidimą. „Challenges - Compete, Get Fit“ yra viena iš populiariausių grupinių sportinių iššūkių programėlių, bet ji yra sukurta tik iOs sistemos mobiliesiems įrenginiams.

Tam, kad galima būtų palyginti alternatyvas su mano kuriama sistema iškeliami tokie reikalavimai sistemai:

* Neturi operacinės sistemos apribojimų.
* Neturi teminio apribojimo.
* Yra galimybė kurti naudotojų grupes.
* Yra galimybė dalintis nuotraukomis ir video failais.
* Yra galimybe palyginti grupės narių pasiekimus.

Kaip matoma žemiau pateiktoje lentelėje (2 lentelė) mano mobilioji aplikacija skirsis nuo pateiktų alternatyvų, taps savotišku socialiniu tinklu, kuriame žmonės galės kurti planus, užduotis 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 | Yra galimybė dalintis nuotraukomis ir video failais | Yra galimybe palyginti grupės narių 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 informacinių technologijų sferos dalis. Vidutiniškai kas mėnesį platformoje „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 sukurti programą, pirmasis kylantis klausimas yra: „kokią technologiją turėčiau naudoti?“. Šis klausimas yra labai svarbus, o 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“ (liet. *gimtosios*) programėles** – kuriamos konkrečiai platformai, skirtos „Android“ arba „iOS“ naudotojams. Jos yra geriau optimizuotos: veikia sklandžiai ir be strigimų, net esant didelei grafinio procesoriaus apkrovai ir sudėtingam skaičiavimo integravimui. Taip pat tokios aplikacijos užima mažiau vietos telefone. Geriausias dalykas tokio tipo programėlėse yra toks, kad jos suteikia prieigą prie integruotų įrenginių ar konkrečios platformos galimybių, pavyzdžiui prie kameros, GPS navigacijos sistemos, operatyviosios atminties ir t.t. Žinoma, yra ir trūkumų. Reikalingos dvi kūrėjų komandos: „Android“ programos paprastai kuriamos „Java“ arba „Kotlin“ programavimo kalbomis, o „iOS“ programos kuriamos „Objective-C“ arba „Swift“, todėl reikia samdyti komandą, turinčią patirties šiomis kalbomis. Dar vienas trūkumas - didelės kūrimo išlaidos: „native“ programėlės idealiai tinka didelėms korporacijoms, turinčioms didelį biudžetą. Kadangi kiekvieną programą reikia kurti nuo nulio, tam, kad ji būtų ekonomiškai sėkminga, reikia paskirti daug išteklių, įskaitant laiką.

**2. Žiniatinklio programėles** – šio tipo programėlė iš esmės yra tinklalapio talpykla, kuri gali turėti dizainą išmaniajam įrenginiui. Vartotojo sąsaja kuriama naudojant standartines žiniatinklio technologijas. Žiniatinklio programėlės darbui naudojama telefono naršyklė. Kadangi, programėle galima naudotis tik prisijungtus prie interneto, jos saugumas yra mažesnis. Pagrindinis tokių programų bruožas yra jų universalumas – galimybė dirbti visuose įrenginiuose be papildomo pritaikymo. Nepriklausomai nuo įdiegtos operacines sistemos, tokios programėles negali naudotis išmaniojo telefono programine įranga. Norint atnaujinti informaciją programėlėje, reikalingas interneto ryšys, todėl veikimo greitį riboja interneto ryšio tiekėjo galimybės. Žemiau (3 lentelė) pateiktas populiariausiu žiniatinklio kūrimo karkasų palyginimas.

**3. Hibridinės programėles** – kitaip nei „native“ programėlės, hibridinės programėlės yra kuriamos abiems platformoms („Android“ ir „iOS“) vienu metu ir rašomos universalia kalba. Naudoja kaip ir “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 (4 lentelė).

**Chart, bar chart

Description automatically generated**

5 pav. Populiariausi hibridinių ir žiniatinklio mobiliųjų programėlių kūrimo karkasai (Statista, 2021).

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

3 lentelė. Populiarių žiniatinklio programėlių kūrimo karkasų palyginimas

| Karkaso pavadinimas | Privalumai | Trūkumai |
| --- | --- | --- |
| React | •Plėtros išlaidos sumažėja maždaug 50%. React Native priežiūra taip pat yra pigesnė.  •Greitas projektų išleidimas.  •Pakanka nedidelės komandos.  •Didžioji dalis kūrimo užduočių atliekama „JavaScript“ kalba.  •Atvirojo kodo privalumas.  •Suteikia kūrėjams galimybę nemokamai naudotis bibliotekomis ir sistemomis. | •Licencijų ir patentų klausimai.  •Prasta 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 įgyvendinama vos keliomis kodo eilutėmis.  •Paprastumas.  •Nereikalauja daug programavimo žinių. | •Karkasas jau baigtas kurti, paskutinė versija buvo išleista 2012-ais metais.  •Mažas greitis.  •Bibliotekos dydis: „jQuery“ bibliotekos dydis yra apie 19 Kb, o tai teoriškai gali sulėtinti tinklalapio įkėlimą, ypač senesniuose kompiuteriuose ir su nestabiliu interneto ryšiu. |
| Express | •Atviras pirminis kodas.  •Išsami dokumentacija.  •Greitas programėlių kūrimas.  •Lengva išmokti. | •Mažas pagrindinių funkcijų kiekis. Visos kitos reikalingos funkcijos turi būti pridėtos išorinių modulių pagalba.  •Turi tik vieną giją veiksmų apdorojimui. tai nėra įprasta ir gali pakenkti nepatyrusio programuotojo produktyvumui. |
| AngularJS | •Didelė bendruomenė.  •Deklaratyvus kodo stilius.  •Universalumas. „AngularJS“ karkasas 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.  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 | Trūkumai |
| --- | --- | --- |
| Flutter | •Perspektyvus ir aktyviai tobulinamas.  •Nuosavas grafinis variklis (nereikia kurti atskiros sąsajos „Android“ ir „iOS“).  •Vieną kartą parašius kodą, jį galima eksportuoti į visas palaikomas platformas.  •Kuriant programėlę ir darant pakeitimus nereikia paleisti programėlės iš naujo, kad pakeitimas būtų atvazduotas.  •Atviras pirminis kodas.  •Gera greitaveika. | •Nestabilumas (neseniai išėjo iš beta versijos).  •Bibliotekų yra mažiau nei „Native“ karkasuose. Tačiau verta paminėti, kad visos svarbiausios bibliotekos jau yra bei nuolat atsiranda naujų.  •Mažai mokomosios medžiagos.  •Mažai papildinių, jau sukurtų sąsajos elementų. |
| Xamarin | •Vienas iš plačiausiai, programinės įrangos kūrėjų ir įvairaus dydžio įmonių naudojamų karkasų, mobiliųjų programėlių kūrimo platformų rinkoje.  •Pilnas kūrimo įrankių paketas.  • „Microsoft“ techninė pagalba.  •Atviras kodas.  •Našumas panasšus į „native“ karkasų.  •Savo vartotojo sąsaja (angl. *UI – user interface*) | •Ne visos bibliotekos yra prieinamos.  •Programėles užima daug vietos.  •Atnaujinimų vėlavimas.  •Mažesnė bendruomenė. |
| Ionic | •Pateikiama daug paruoštų grafinės sąsajos elementų.  •Sukurta su „AngularJS“.  • „Ionic“ vartotojo sąsaja itin patogi.  •Sukurta programėlė tinkama eksportavimui į visas platformas. | •Kuriant programėlę gali atsirasti saugumo problemų, kurias reikia taisyti patiems programuotojams.  •Programėlė neatlaiko daug grandininių elementu.  •Integruoto funkcionalumo apribojimas. |

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

### Išsami „Flutter“ technologijos apžvalga

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

Programos paleidžiamos remiantis atvaizdavimo variklio (sukurto su „C++“) ir „Flutter“ (sukurto su „Dart“) karkaso deriniu. Visi tokiu būdu sukurti failai pridedami prie kiekvienos konkrečios platformos programos ir programinės įrangos kūrimo rinkinio (angl. *software development kit*) programinės įrangos. Žemiau (6 pav.) pateikiamas „Flutter“ architektūros modelis.

.Graphical user interface, chart

Description automatically generated

6 pav. „Flutter“ architektūros modelis

Programėles kūrimas su šiuo karkasu panašus 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.

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

„Dart” yra dinamiška, klasėmis pagrįsta, į objektą orientuota programavimo kalba su uždarymu ir leksine apimtimi. Sintaksės požiūriu ji yra gana panaši į „Java“, „C“ ir „JavaScript“ kalbas. Jei mokate kurią nors iš šių programavimo kalbų, galite lengvai išmokti ir „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 ir „refield generics“ tipo sąsaja. Tai yra kompiliuota kalba ir palaiko dviejų tipų kompiliavimo būdus.

AOT (Ahead-of-Time) – 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.

### Vidinės programos dalies technologijų pasirinkimas

„Backend“ yra programa kuri veikia serverio pusėje, 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 realizuoti API. Pagrindinės tam skirtos technologijos:

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, kuriuose verta naudotis SOAP:
   * kai duomenis galima naudoti pageidaujamomis kalbomis;
   * kai reikia sukurti saugų komunikacijos kanalą tarp naudoto ir sistemos;
   * SOAP palaiko SMTP, JSM, TCP transportavimo protokolus.

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

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

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

5 lentelė. Vidinių sistemų karkasų palyginimas

|  |  |  |
| --- | --- | --- |
| Karkaso pavadinimas | Privalumai | Trukumai |
| Spring | Užima mažai vietos.  Daug modulių su įvairiu funkcionalumu.  Lengvai testuojamas. | Gali buti sudėtinga konfigūruoti.  Neapsaugo nuo „XSS“ atakų. |
| Ruby on Rails | Lengvai naudojamas.  Plačiai paplites, turi daug dokumentacijos ir gerųjų praktikų. | Sunku konfigūruoti.  Lėtai veikia palyginus su alternatyvomis. |
| Meteor | Atviras pradinis kodas.  Turi integruotą ORM duomenims saugoti naudojant MongoDB. | Nepalaiko kitų duomenų bazių.  Naudoja nestandartinius sprendimus. |
| Django | Turi integruotą apsaugą nuo populiariausių įsilaužimo praktikų („XSS“; „SQL Injection“).  Naudoja Python. | Reikalauja optimizacijos, kad padidinti greitaveiką.  Neturi bendrai pripažįstamų geriausių praktikų. |

Savo projekte naudosiu „Spring“ karkasą, todėl, kad jis suteikia didžiausią įvairių funkcijų rinkinį, yra pilnai ir lengvai konfigūruojamas. Šis karkasas naudoja „Java“ programavimo kalbą.

Dabar reikia pasirinkti duomenų bazę, kuri bus naudojama su „Spring“ karkasu. Duomenų bazė naudojama duomenims gauti ir saugoti. Priklausomai nuo to, kokius duomenis 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 tarpussavyje pagal raktą. O nereliacinės („NoSQL“) duomenų bazės saugo reikšmes 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 turime duomenis, iš kurių reikia gauti tam tikrą statistiką arba agreguoti juos į kokį nors naują duomenų tipą, geriau panaudoti reliacinę duomenų bazę (IBM, 2021). Savo projekte naudosiu SQL duomenų bazę, vadinamą „H2“.

### Išsami vidinės dalies apžvalga

„Spring“ - populiariausia „Java“ kalbos programų kūrimo aplinka (Intexsoft, 2020). „Spring“ atsirado 2003 m. kaip lengvesnė „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 (8 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 architektūrinis modelis

#### Pagrindiniai „Spring“ karkaso sluoksniai:

1. **Core Container**

Pagrindinė „Spring Core“ idėja yra IoC („Inversion of Control“), dar yra vadinama priklausomybių injekcija. Tai procesas, kurio metu objektai apibrėžia savo priklausomybes (t.y. kitus objektus, su kuriais jie dirba) tik per konstruktoriaus argumentus. Tada konteineris sukuria šias priklausomybes, kai sukuria „Bean“. Šis procesas iš esmės yra atvirkštinis (tai parodo ir pavadinimas: valdymo inversija), kai pats „Bean“ kontroliuoja savo priklausomybių egzistavimą arba lokaciją, naudodamas tiesioginį klasių konstravimą arba šablonus, pvz., paslaugų lokatoriaus modelį.

„Spring“ objektai, kurie sudaro programos pagrindą ir kuriuos valdo „Spring IoC“ konteineris, vadinami „Bean“. „Bean“ yra objektas, kurį sukuria, surenka ir valdo „Spring IoC“ konteineris ir jo priklausomybės atsispindimos konteinerio naudojamuose konfigūracijos metaduomenyse.  
 SpEL („Spring Expression Language“) yra galinga išraiškų kalba, kuri palaiko užklausų pateikimą ir objekto grafinį manipuliavimą vykdymo metu. Tai yra ypač naudingas įrankis programų testavimui ir gyvam programos 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 nerealicines duomenų bazes, žemėlapių mažinimo sistemas ir debesų duomenų paslaugas. Keletas iš pagrindinių duomenų sluoksinio modulių:

* 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. Joje pateikiami duomenų bazės duomenų užklausų ir atnaujinimo metodai. Ši programavimo technologija yra orientuota 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:
  + JPA („Java Persistence API“): daugiausia naudojama išlaikyti duomenis tarp Java objektų ir reliacinių duomenų bazių. Ji 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 atvaizduoti.
  + Hibernate – tai „Java“ sistema, kuri supaprastina „Java“ programų kūrimą, kad jos galėtų sąveikauti su duomenų baze. Jos esmė yra automatizuotas programos objektų konvertavimas į ir iš reliacinių duomenų bazių 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 (liet. *Atomiškumas*); Consistency (liet. *Nuoseklumas*); Isolation (liet. *Išskyrimas*); Durability (liet. *Patvarumas*)) charakteristikas, kad būtų užtikrintas 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), užtikrinant laisvą šių elementų ryšį.

1. **Test**

Testavimas yra neatsiejama įmonės programinės įrangos kūrimo dalis. Priklausomybės įvedimas padaro jūsų kodą mažiau priklausomą nuo konteinerio, nei naudojant tradicinį „Java EE“ kūrimą. POJO, sudarantys jūsų programą, turėtų būti išbandomi atliekant „Junit“ arba „TestNG“ testus, o bjektai sukuriami naudojant naują operatorių, be Spring ar kito konteinerio. Galima naudoti netikrus objektus (kartu su kitais vertingais testavimo būdais), norint išbandyti kodą atskirai. Jei laikysitės „Spring“ architektūros rekomendacijų, gautas švarus kodų bazės sluoksniavimas ir komponentų sudarymas palengvins vienetų testavimą.

# Projektinė dalis

# Sistemos naudotojai

6 lentelė. Naudotojo „Neprisijungęs naudotojas“ detalizavimas

|  |  |
| --- | --- |
| **Rolė:** | Neprisijungęs naudotojas |
| **Funkcijos:** | Registruotis |
| **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 kūrėjus. Moderatorius tai rolė, kuri palaiko tvarką grupėse (komentaruose ir tarp narių). |

9 lentelė. Naudotojo „Grupės kūrėjas“ detalizavimas

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

# Panaudos atvejų diagrama

Panaudos atvejų diagrama (angl. *use case*) – tai diagramos, kuriose aprašomas sistemos funkcionalumas naudojant aktorius ir galimus panaudojimo atvejus. Panaudos atvejų diagrama parodo, kurie aktoriai sąveikauja su kuriuo 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.

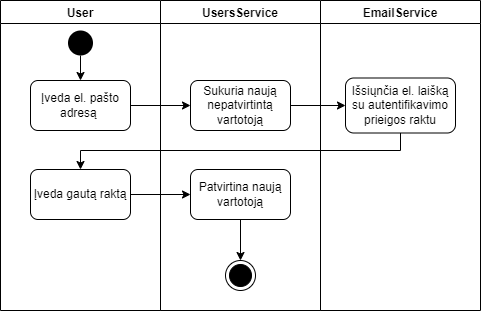
Žemiau (9 pav.) pateikta panaudos atvejų diagrama, kuri susideda iš 12 panaudojimo atvejų.

Diagram

Description automatically generated

9 pav. Panaudos atvejų diagrama

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



10 pav. „Užregistruoti paskyrą“ užduoties modelis

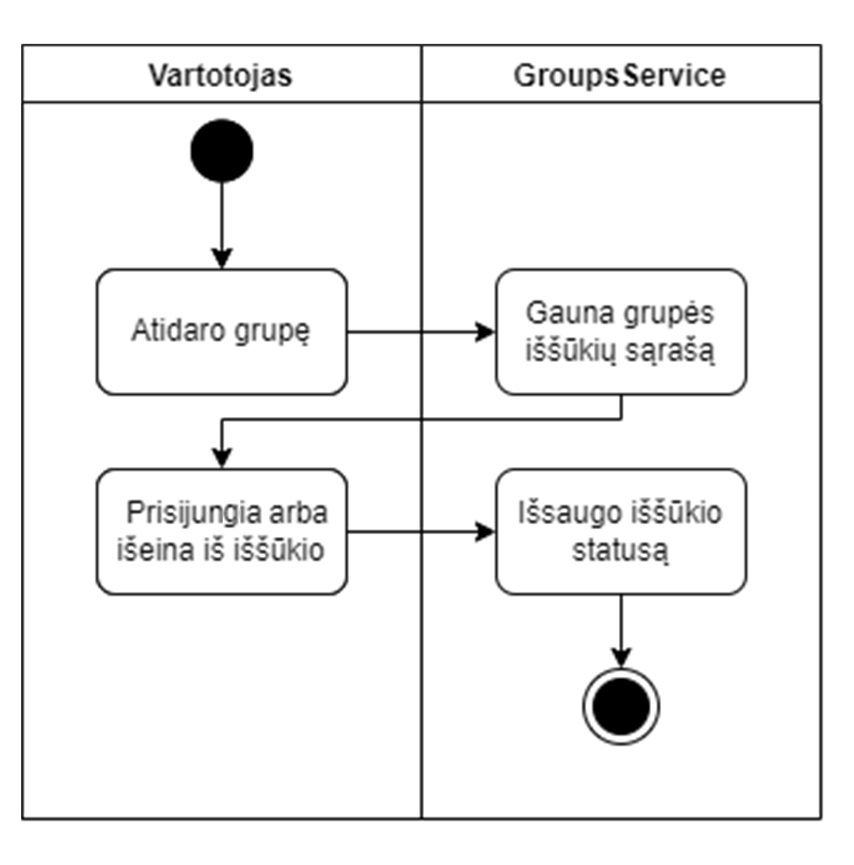
|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 1 | |
| Panaudojimo atvejis | Užregistruoti paskyrą |
| Tikslas | Tam, kad naudotis aplikaciją reikalinga turėti savo paskyrą |
| Vartotojas/Aktorius: | Neprisijungęs vartotojas |
| Prieš - sąlyga: | Vartotojas atidaro mobiliąją aplikaciją. |
| Sužadinimo 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. |

Diagram

Description automatically generated

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

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 2 | |
| Panaudojimo atvejis | Prisijungti arba išeiti iš grupės |
| Tikslas | Tam, kad matyti iššūkių sąrašą grupėje ir dalyvauti juose reikia prisijungti prie grupės. Tam, kad nematyti grupės iššūkių reikia išeiti iš grupės. |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti autorizuotas. |
| Sužadinimo sąlyga: | Norėdamas prisijungti prie grupės vartotojas spaudžia mygtuką „+“;  Norėdamas 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 duomenys. |



12 pav. „Prisijungti arba išeiti iš iššūkio“ užduoties

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 3 | |
| Panaudojimo atvejis | Prisijungti arba išeiti iš iššūkio |
| Tikslas | Tam, kad dalyvauti iššūkiuose reikia būti grupės nariu. |
| 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 prisijungti; 3. Spaudžia mygtuką „Prisijungti prie iššūkio“; |
| Po-sąlyga: | Duomenų bazėje sėkmingai atnaujinti vartotojo iššūkiai. |

Diagram

Description automatically generated

13 pav. „Kurti įrašus“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 4 | |
| 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 mygtuką „+“ ; 3. Įrašo tekstinę žinutę, gali pridėti vaizdo įrašą arba nuotrauką; 4. Spaudžia „Publikuoti“; |
| Po-sąlyga: | Įrašas sėkmingai pridėtas į iššūkį. |

**Diagram

Description automatically generated**

14 pav. „Palikti komentarą“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 5 | |
| 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 su užrašu „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 įrašu. |

Diagram

Description automatically generated

15 pav. „Peržiūrėti iššūkius“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 6 | |
| Panaudojimo atvejis | Peržiūrėti iššūkius |
| Tikslas | Rasti iššūkius kuriose grupės narys nori dalyvauti |
| Vartotojas/Aktorius: | Vartotojas (prisijungęs) |
| Prieš - sąlyga: | Vartotojas turi būti grupės narys |
| Sužadinimo sąlyga: | Vartotojas atidaro grupę |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Vartotojas užeina į grupę 2. Vartotojas gauna grupės iššūkių sąrašą |
| Po-sąlyga: | - |

Text, timeline

Description automatically generated

16 pav. „Kurti įrašus grupėje“ užduoties modelis

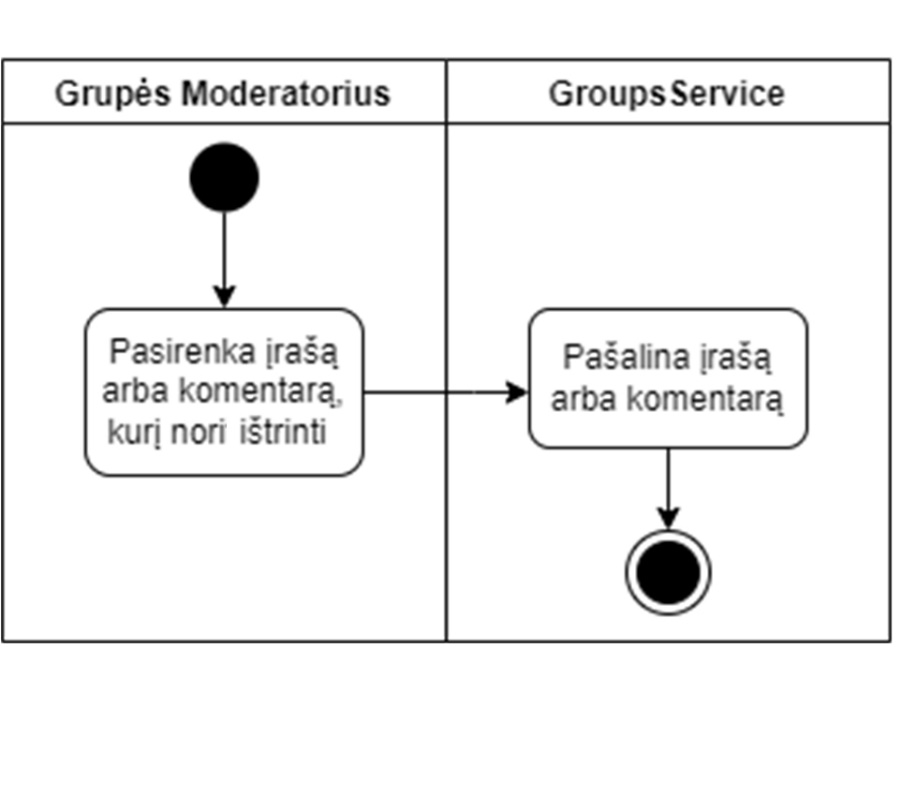
| Specifikacijų lentelė Nr. 7 | |
| --- | --- |
| Panaudojimo atvejis | Kurti įrašus grupėje |
| Tikslas | Informuoti grupės narius. |
| Vartotojas/Aktorius: | Grupės moderatorius |
| Prieš - sąlyga: | Moderatorius užeina į grupę. |
| Sužadinimo sąlyga: | Mygtuko „+“ paspaudimas. |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Moderatorius užeina į grupę; 2. Spaudžia mygtuką „+“; 3. Sukuria naują įrašą grupėje; |
| Po-sąlyga: | Įrašas sėkmingai pridėtas į grupę. Visiems grupės nariams siunčiamas pranešimas apie naują įrašą |

Text, timeline

Description automatically generated

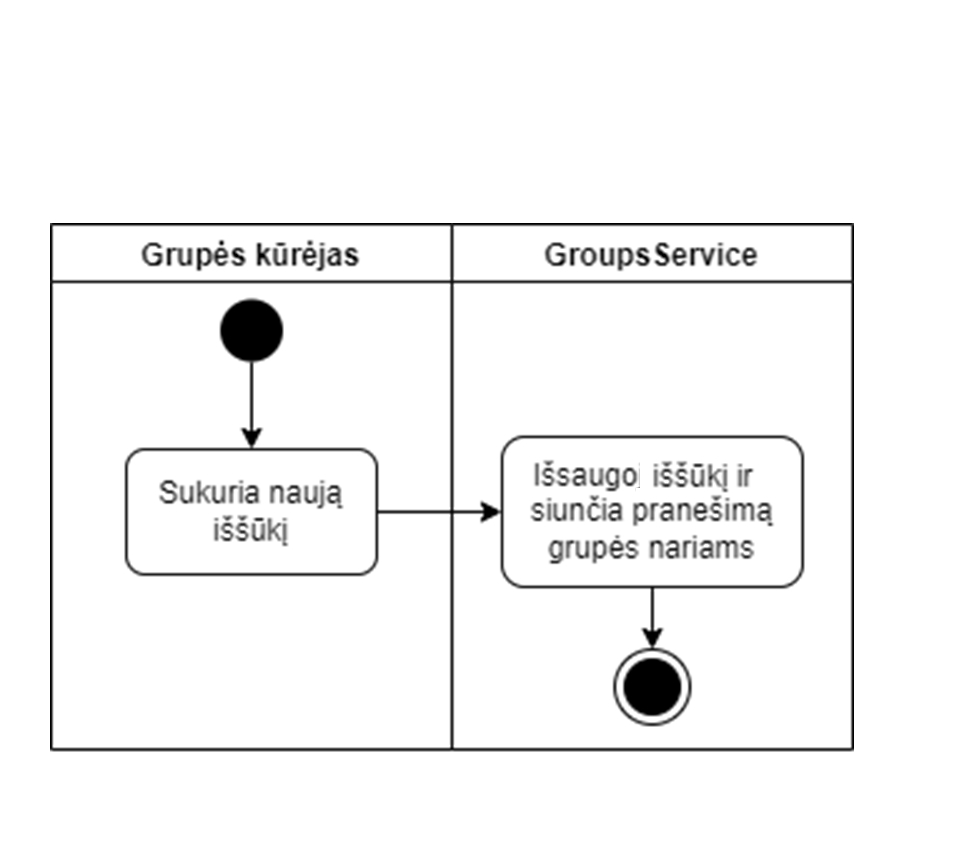
17 pav. „Valdyti grupės narius“ užduoties modelis

| Specifikacijų lentelė Nr. 8 | |
| --- | --- |
| Panaudojimo atvejis | Valdyti grupės narius |
| Tikslas | Valdyti narius, kurie gali prisijungti prie grupės, dalyvauti iššūkiuose, rašyti komentarus |
| Vartotojas/Aktorius: | Grupės moderatorius |
| Prieš - sąlyga: | Grupėje turi būti bent 1 narys (išskyrus moderatorių) |
| Sužadinimo sąlyga: | Mygtuko „+“ paspaudimas. |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Moderatorius užeina į grupę; 2. Spaudžia į mygtuką „ ... „ ; 3. Pasirenka narį šalinimui ir įrašo šalinimo priežastį; |
| Po-sąlyga: | Pasirinktas vartotojas sėkmingai išbrauktas iš grupės narių. |



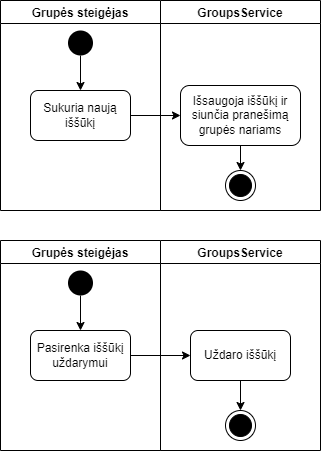
18 pav. „Tvarkyti įrašus ir komentarus“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 9 | |
| Panaudojimo atvejis | Tvarkyti įrašus ir komentarus |
| Tikslas | Trinti nereikalingą informaciją, šlamštą, 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 arba ištrina komentarą. |
| Po-sąlyga: | Įrašas arba komentaras sėkmingai ištrintas arba pakoreguotas. |



19 pav. „Kurti iššūkį“ užduoties modelis

|  |  |
| --- | --- |
| Specifikacijų lentelė Nr. 10 | |
| Panaudojimo atvejis | Kurti iššūkį |
| Tikslas | Motyvuoti narius atlikti užduotis, ką nors daryti |
| Vartotojas/Aktorius: | Grupės kūrėjas |
| Prieš - sąlyga: | - |
| Sužadinimo sąlyga: | Mygtuko „Pridėti naują iššūkį“ paspaudimas. |
| Aprašas (scenarijus): | Pagrindinis scenarijus:   1. Grupės kūrėjas užeina į savo grupę; 2. Paspaudžia mygtuką „Pridėti naują iššūkį“; 3. Įveda iššūkio informaciją (pradžios data, pabaigos data, pavadinimas, taškai ir t.t.); 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 Nr.11 | |
| --- | --- |
| 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“ paspaudimas. |
| 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 sistemos reikalavimus. 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 produktu) ir nefunkciniai reikalavimai (apibūdinantys, kaip sistema ar programinės įrangos produktas turi veikti ir kokias savybes ar charakteristikas turi turėti). Žemiau lentelėse 10 - 11 aprašomi funkciniai ir nefunkciniai reikalavimai kuriamai sistemai.

10 lentelė. Funkciniai reikalavimai

|  |  |  |
| --- | --- | --- |
| Nr. | Formuluotė | Priėmimo kriterijai |
| 1. | Vartotojas turi turėti galimybę prisijungti prie savo paskyros | 1. Vartotojas gali įvesti savo el. paštą; 2. Vartotojas turi gauti laišką su autentifikacijos raktu; 3. Įvedus šį raktą vartotojui turi būti suteikta prieiga prie paskyros; |
| 2. | Vartotojas turi turėti galimybę peržiūrėti savo grupes ir iššūkius | 1. Prisijungęs vartotojas mato savo grupes; 2. Pasirinkęs grupę vartotojas mato iššūkius kuriuose dalyvauja; |
| 3. | Vartotojui turi būti suteikta galimybė kurti ir komentuoti įrašus | 1. Vartotojas turi turėti galimybę sukurti naują įrašą iššūkyje; 2. Grupės kūrėjas turi turėti galimybę sukurti naują įrašą grupėje; 3. Vartotojas gali rašyti komentarus po įrašu arba atsakyti į komentarus; |
| 4. | Vartotojas turi turėti galimybę 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 viešąją 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. | Formuluotė | Priėmimo kriterijai |
| 1. | Vartotojas turi greitai gauti el. laišką su autentifikavimo raktu | Laiškas turi atvykti ne ilgiau kaip per |
| 2. | Mobilioji aplikacija turi veikti „Android“ ir „iOS“ operacinėse sistemose | Sistema tvarkingai veikia visose populiariausiose operacinėse 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 vaizduojami nurodytu formatu. |
| 5. | Užtikrinti naudotojų duomenų saugumą | Sistema turi nepažeisti GDPR (General Data Protection Regulation).  Atsarginė duomenų kopija atnaujinama kiekvieną savaitę. |
| 6. | Po sistemos atnaujinimo failai neturi būti prarasti | Prieš sistemos atnaujinimą visi duomenys bei konfigūraciniai failai turi būti išsaugoti. |

# Sistemos architektūra

# Sistemos serverio architektūra

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

* API kontroleris - tai „RESTful 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 duomenų baze, naudojama saugos raktams validuoti ir saugoti.
* Vartotojų saugykla - naudojama komunikavimui su duomenų baze. Suteikia galimybę gauti, saugoti, atnaujinti vartotojus.
* Iššūkių saugykla - naudojama komunikavimui su duomenų baze. Suteikia galimybę gauti, saugoti, atnaujinti iššūkius.
* Publikacijų saugykla - naudojama komunikavimui su duomenų baze. Suteikia galimybę gauti, saugoti, atnaujinti publikacijas.
* Komentarų saugykla - naudojama komunikavimui su duomenų baze. Suteikia galimybę gauti, saugoti, atnaujinti komentarus.

Diagram

Description automatically generated

21 pav. Sistemos serverio komponentai

# Sistemos įgyvendinimo architektūra

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

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

Diagram

Description automatically generated

22 pav. Sistemos įgyvendinimo architektūra

# Duomenų bazės struktūra ir ryšiai

Diagram

Description automatically generated

23 pav. Duomenų bazės struktūra

Duomenų bazė yra sudaryta iš šių lentelių:

* User – lentelė kuri saugo sistemos naudotojus;
* Group – lentelė saugo sistemoje sukurtas grupes;
* Challenge – lentelė kuri saugo sukurtus iššūkius;
* Post – lentelė kuri saugo sukurtas publikacijas, target\_uuid raktas saugo vietą kur buvo sukurta publikacija, tai gali būti tiek grupė tiek iššūkis;
* Comment – lentelė sauganti naudotojų parašytus komentarus;
* Group\_Members – lentelė saugo grupių narių sąrašą;
* Challenge\_Members – lentelė saugo dalyvaujančių iššūkyje naudotojų sąrašą;

# Testavimas

# Testavimo metodikos pasirinkimas

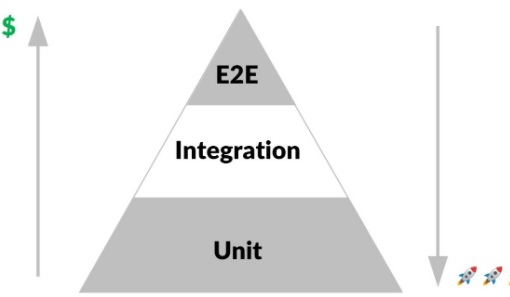
Mobiliųjų programėlių testavimas – tai procesas, kurio metu tikrinamas mobiliesiems įrenginiams sukurtos programinės įrangos funkcionalumas, patogumas naudoti ir suderinamumas. Testavimas gali būti rankinis arba automatinis.

*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 negalime įsitikinti, ar automatizavimas įmanomas. Vienas iš pagrindinių testavimo principų teigia: 100% automatizavimas neįmanomas. Todėl rankinis testavimas yra būtinas.

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

Kai kurios testavimo užduotys, pvz., žemo lygio regresinis testavimas, gali reikalauti daug darbo ir daug laiko, jei jis 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 laiko ir pastangų projekto komandai.

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



24 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, nereikia nuvertinti automatizavimo vertės, taip supaprastinant projekto priežiūrą ir sumažinant jo išlaidas. 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 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, tuomet atliekamas parašyto kodo pertvarkymas, nuolat tikrinant, ar visi testai buvo išlaikyti. Tam, kad išbandyti mobiliąją programėlę buvo parašyti dvejų tipų automatiniai testai.

# Modulinis testavimas

Modulinis testavimas (angl. *unit testing*) - tai yra mažo nepriklausomo kodo bloko testavimo metodas. Mažo bloko kodas daugeliu atvejų yra funkcija. Žodis nepriklausomas reiškia, kad jis nepriklauso nuo kitų projekto kodo dalių. Iš viso buvo parašyta 41 unit testų, žemiau pateikiami trijų testų pavyzdžiai.

12 lentelė. Modulinis testas Nr.1

| Nr. | 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 grupė su nariais, kuriamas grupės iššūkis, tuomet grupės narys bando prisijungti prie iššūkio. | |

13 lentelė. Modulinis testas Nr.2

| Nr. | Pavadinimas: Privačios grupės kūrėjas turi 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 atmeta prašymą įstoti į grupę. | |

14 lentelė. Modulinis testas Nr.3

|  |  |
| --- | --- |
| Nr. | 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, taip siekiama patikrinti duomenų mainus tarp šių modulių. Iš viso buvo parašyta 7 integraciniai testai, žemiau pateikiami dviejų testų pavyzdžiai.

15 lentelė. Integracinis testas Nr. 1

| Nr. | 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 užklausa įstoti į grupę, po to bandoma prisijungti prie iššūkio ir sukurti naują įrašą. | |

lentelė. Integracinis testas Nr. 2

| Nr. | 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 įrašą iššūkyje. | |

# Rankinis end-to-end testavimas

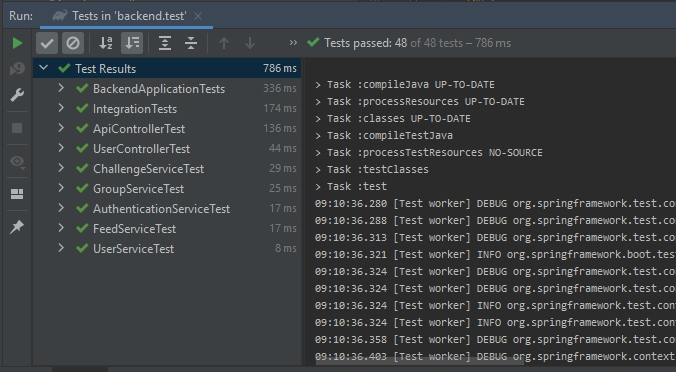
Visapusiškas testavimas, taip pat žinomas kaip „End-to-end“ arba E2E, yra testavimo procesas, kuris išsamiai imituoja vartotojo aplinką. Bandymo metu jis imituoja: pelės paspaudimus, mygtukų paspaudimus, formų pildymą, perėjimą per puslapius ir kitus elgesio veiksnius. Šio testavimo esmė yra išsiaiškinti,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

17 lentelė. Rankinis testavimas

| Nr. | Testavimo atvejis | Laukiamas rezultatas | Gautas rezultatas |
| --- | --- | --- | --- |
| 1. | Testuojamas registracijos langas | Įvedus egzistuojančią pašto adresą turėtų atsirasti laukas skirtas „Token“ įvedimui | Įvedus email adresą atsirado laukas skirtas „Token“ įvedimui  25 pav. Registracijos laukas    26 pav. Registracijos laukas po email įvedimo |
| 2. | Testuojama registracija | Naudotojas įvedus savo email adresą turi gauti autorizacijos raktą į savo paštą. Gautas raktas įvedamas į „Token“ lauką. Toliau spaudžiamas mygtukas „Validate token“. Jeigu autorizavimo raktas buvo teisingas vartotojas matys naują langą „Feed“. Priešingu atveju naudotojas pamatys įrašą kad raktas neteisingas. | Rezultatas sėkmingos registracijos atvejų:  Atidaromas langas „Feed“  **(Žr.29 pav.)**  Nesėkmingos registracijos atvejų:    27 pav. Nesėkmingos registracijos atvejis |
| 3. | „Groups“ lango atidarymas | Paspaudus ant „Gruops“ navigacijos meniu „Grupės“, turėtų atsidaryti naujas „Grupių“ langas, kur rodomos grupės, kuriose dalyvauja vartotojas | Atidaromas langas „Groups“  (Žr. 31 pav.) |
| 4. | Grupės atidarymas | Paspaudus ant grupės pavadinimo, turėtų atsidaryti šios grupės langas su naujienų srautų (angl. *feed*). | Atidaromas grupės naujienomis  (Žr. 32 pav.) |
| 5. | Grupinių iššūkių rodymas | Kai vartotojas prisijungia prie grupės ir atidaro skyrių „Challenges“ turi būti rodomi visi iššūkiai. | Atidarius langą „Challenges“ rodomi visi tos grupės iššūkiai.  (Žr. 33 pav.) |
| 6. | Įstoti į grupę /  Išeiti iš grupės | Vartotojas paspaudžia mygtuką „+“ ir automatiškai prisijungia prie grupės, jei ji neuždaryta. Jei grupė uždaryta, prašymą prisijungti turi patvirtinti šios grupės moderatorius.  /  Vartotojas gali išeiti iš grupės paspausdamas mygtuką „-“. | Paspaudęs mygtuką „+“, vartotojas prisijungia prie neuždarytos grupės:  Jei vartotojas nori prisijungti prie uždaros grupės, jis paspaudžia mygtuką "+" ir laukia, kol šios grupės moderatorius jį priims.  /  Kai vartotojas paspaudžia mygtuką „-“, jis palieka šią grupę. |
| 7. | Prisijungti prie iššūkio/  Išeiti iš iššūkio | Tam, kad prisijungti prie iššūkio, vartotojas turi užeiti į grupę, pasirinkti norimą iššūkį ir spustelėti pliuso ženklą.  /  Norint išeiti iš iššūkio reikia paspausti minuso ženklą. | Grupės narys sekmingai pridėtas prie iššūkio:  /  Grupės narys sekmingai paliko iššūkį: |
| 8. | Sukurti įrašą | Grupės narys gali kurti įrašus iššūkyje. Jis spaudžia mygtuką „Create new post“ ir gali pridėti nuotrauką. Naujas įrašas turi būti pridėtas ir rodomas visiems grupės nariams. | Naujas įrašas sėkmingai sukurtas.  (Žr. 34 pav.) |
| 9. | Palikti komentarą | Grupės narys gali komentuoti įrašus grupėse ir iššūkiuose. Jis spaudžia „All comments“ ir rašo komentarą. Naujas komentaras turi būti pridėtas ir rodomas visiems grupės nariams. | Komentaras sėkmingai pridėtas  **(Žr. 35 pav.)** |
| 10. | Peržiūrėti pranešimus | Vartotojas pereina į „Notifications“, kur rodomi visi pranešimai. | Pranešimų lange „Notifications“ rodomos visos naujienos:  **(Žr. 36 pav.)** |
| 11. | Sukurti grupę | Bet kuris vartotojas gali susikurti savo grupę su pasirinktu tipu (atidaryta/uždaryta). Grupė turi būti sukurta. | Grupė sėkmingai pridėta ir pasiekiama per grupės paiešką  **(Žr. 38 pav. )** |
| 12. | Sukurti iššūkį | Sukurti iššūkį gali tik grupės kūrėjas. Iššūkis turi būti sukurtas ir rodomas grupės nariams. | Iššūkis sėkimgai sukurtas ir matomas grupės nariams.  **(Žr. 37 pav. )** |

# Testavimo išvados

Taigi, iš viso buvo parašyta 48 automatizuoti testai „*SpringBoot“* aplinkoje (28 pav.). Taip pat buvo atlikta 12 testų rankiniu būdu. Iš gautų testavimo rezultatų, galima teigti, kad sistemos funkcionalumas veikia be klaidų, programėlė praeina visus testus, vartotojo sąsaja atrodo vienodai „*Android*“ ir „*iOS*“ operacinėse sistemose. Ištestavus funkcinius ir nefunkcinius reikalavimus, pagal rezultatus, mobilioji programėlė atitinka aprašytus normatyvus.



28 pav. Automatizuotų testų paleidimas

# Išvados ir siūlymai

1. Išanalizavus žmonių poreikį naudotis tikslų siekimo ir kitomis, įvairiai motyvuojančiomis programėlėmis, bei statistiką iš programėlių parduotuvės („Play Market“), buvo nustatyta, kad žmonės aktyviai naudoja programėles, kurios padeda formuoti gerus įpročius sveikatai, sportui, mitybai ir kitiems gyvenimo aspektams bei siekti užsibrėžtų tikslų. Todėl buvo nuspręsta sukurti mobiliąją aplikaciją, kurios idėja yra paversti grupinius iššūkius žaidimu ir pridėti konkurencinį aspektą. Tyrimai rodo, kad užduočių atlikimas žaidimo forma gali padidinti motyvaciją ir padėti laiku atlikti užduotis.

2. Atlikus mano aplikacijos palyginimą su rinkoje siūlomomis alternatyvomis – „Microsoft To Do“, „Challenges – Complete, Get Fit“, „Habitica“, paaiškejo, kokius privalumus ir trūkumus turi kuriama sistema. Nevisos alternatyvos leidžia kurti grupes iššūkiams ir lyginti grupės narių pasiekimus. Pagrindinis rinkoje siūlomų alternatyvų privalumas yra toks, kad jos prieinamos įvairiose operacinėse sistemose ir turi daugybę funkcijų. Šiame darbe aprašoma mobilioji aplikacija skiriasi nuo pateiktų alternatyvų tuo, kad yra nemokoma, primena savotišką socialinį tinklą, kuriame žmonės gali 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 visoms 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 kalba, kurią ir naudojau kuriant programėlę.

4. Projektavimo metu buvo nustatyti vartotojo pageidavimai būsimai sistemai, tai yra 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 dalykinės 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 ir rankinė patikra. Todėl sistemos testavimui buvo naudoti du metodai: automatizuotas ir rankinis testavimas. Mobiliajai programėlei testuoti buvo parašyta 48 automatizuoti testai: 41 modulinių testų (angl. *unit test*) ir 7 integraciniai testai. Rankiniu būdu 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 rankiniu 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. Tam, 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

Ekrano vaizdai

|  |  |
| --- | --- |
| pav. Registracijos langas | pav. Feed langas |
| pav. „Groups“ langas | pav. Įrašai grupėje |
| pav. „Challenges“ langas | pav. „Create new post“ langas |
| pav. Komentarai po įrašu | 36 pav. „Notifications“ langas |
| 37 pav. Naujo iššūkio kūrimas | 38 pav. Naujos grupės kūrimas |