Kauno technologijos universitetas

Informatikos fakultetas

Android programėlių kūrimo metodo naudojančio „Model Driven Engineering“ principu analizė ir tyrimas

Baigiamasis magistro projektas

Ernestas Vyšniauskas

Projekto autorius

doc. Šarūnas Packevičius

Vadovas

KAUNAS, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Android programėlių kūrimo metodo naudojančio „Model Driven Engineering“ principu analizė ir tyrimas

Baigiamasis magistro projektas

Programų sistemų inžinerija (6211BX011)

Ernestas Vyšniauskas

Projekto autorius

doc. Šarūnas Packevičius

Vadovas

prof. Tomas Blažauskas

Recenzentas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Ernestas Vyšniauskas

Android programėlių kūrimo metodo naudojančio „Model Driven Engineering“ principu analizė ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Ernestas Vyšniauskas

Patvirtinta elektroniniu būdu

Vyšniauskas Ernestas. Android programėlių kūrimo metodo naudojančio „Model Driven Engineering“ principu analizė ir tyrimas. Magistro baigiamasis projektas. Vadovas doc. Šarūnas Packevičius; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Programų sistemų inžinerija, informatikos inžinerija

Reikšminiai žodžiai: *Android mobilios programėlės, kodo generavimas, modelių transformacijos, programinės įrangos modeliavimas, į modelį orientuota inžinerija*

Kaunas, 2021. 59 p.

SANTRAUKA

Mobiliųjų programėlių kūrimo pramonė vis labiau auga dėl intensyvaus programų taikymo mobiliuosiuose įrenginiuose, dauguma jų naudoja „Android“ operacinę sistemą. Tačiau kuriant programėles mobiliosioms platformoms iškyla papildomi rūpesčiai, tokie kaip kodo efektyvumas, sąveika su įrenginio ištekliais, trumpas laiko tarpas pateikimui į rinką.

MDE kartu su UML galėtų suteikti abstrakciją ir automatizavimą mobiliosios programinės įrangos kūrėjams. Automatizuotas kodo generavimas iš „Android“ programėlės modelių gali užtikrinti kodo efektyvumą, teisingą resursų naudojamą ir sutrumpinti projekto realizavimo laikotarpį.

Dėl to yra siūlomas įrankis, kuris būtų pagrįstas MDE metodika ir galėtų atlikti transformacijos procesą, tarp pateiktų „Android“ programėlės modelių, kurio rezultatas būtų „Android“ programėlės kodas.

Vyšniauskas Ernestas. Analysis and Research of the Android Applications Development Method Using the Model Driven Engineering. Master's Final Degree Project. Supervisor doc. Šarūnas Packevičius; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Software Engineering, computer engineering

Keywords: *Android mobile applications, code generation, model driven engineering, model transformations, software modeling*

Kaunas, 2021. 59 p.

SUMMARY

The mobile app development industry is growing due to the intensive application of mobile devices, most of which use the Android operating system. However, mobile platforms' development raises additional concerns such as code performance, interoperability with device resources, and a short time to market.

MDE along with UML could provide abstraction and automation for mobile software developers. Automated code generation from Android mobile application models could ensure code efficiency, correct use of resources, and shorten project implementation time.

Therefore, a tool is proposed based on the MDE methodology and could perform the transformation process between the submitted Android mobile application models, resulting in Android mobile application code.

Turinys

Lentelių sąrašas ... **8**
Paveikslų sąrašas ... **9**
Santrumpų ir terminų sąrašas ... **11**
1. Įvadas ... **13**
2. Rinkos, projektavimo metodologijos ir technologijų analizė ... **14**
2.1. Analizės tikslas ... 14
2.2. Srities apžvalga ... 14
2.3. „Android“ programėlių rinkos analizė ... 14
2.4. Sistemos modeliavimas ... 16
2.4.1. MDE *(angl. Model-driven engineering)* ... 16
2.4.2. MDA *(angl. Model-driven architecture)* ... 17
2.5. Pagrindiniai MDE standartai ir kalbos ... 17
2.5.1. Metamodelis ... 18
2.5.2. „Microsoft“ domeno specifinė kalba DSL ... 18
2.6. Metodai, technologijos, sprendimai kuriant „Android“ programėles naudojant MDE ... 19
2.7. Egzistuojančių programų sistemų palyginimas ... 21
2.8. Analizės išvados ... 22
3. Projektinė dalis ... **23**
3.1. Sistemos paskirtis ... 23
3.2. Sistemos tikslai ... 23
3.3. Apribojimai sprendimui ... 23
3.4. Diegimo aplinka ... 23
3.5. Nefunkciniai reikalavimai ... 24
3.5.1. Reikalavimai sistemos išvaizdai ... 24
3.5.2. Reikalavimai panaudojamumui ... 24
3.5.3. Reikalavimai vykdymo charakteristikoms ... 24
3.5.4. Reikalavimai veikimo sąlygoms ... 24
3.5.5. Reikalavimai saugumui ... 25
3.6. Funkciniai reikalavimai ... 25
3.6.1. Panaudojimo atvejų specifikacija ... 28
3.7. Sistemos statinis vaizdas ... 42
3.8. Pagrindinių komponentų detalizavimas ... 42
3.8.1. WEB paslaugos komponentas ... 42
3.8.2. Vykdomasis failas „.jar“ generatoriaus komponentas ... 44
3.9. Kodo generavimo procesas ... 44
3.9.1. Sukurti „MagicDraw“ profiliai ... 45
3.9.2. Kodo generavimo eiga ... 48
4. Sukurtos programinės sistemos analizavimas ... **49**
4.1. Programinės sistemos bandymas ... 49
4.2. Sistemos vertinimo rezultatai ... 52
4.3. Siūlomi pakeitimai, atnaujinimai ... 53
4.4. Atlikti atnaujinimai ... 53
5. Eksperimentinis „Android“ programėlių kūrimo tyrimas ... **56**
5.1. Tyrimo eiga ... 56

5.2. Programėlių aprašymai 56
5.2.1. „Užrašinė“ programėlės aprašymai 56
5.2.2. „Minimalistinis užduočių sekimo“ programėlės trumpoji specifikacija 56
5.3. Gauti rezultatai 57
5.4. Rezultatų tikslumas 57
Išvados 58
Literatūros sąrašas 59
Priedai 61
1 priedas. „Užrašinė“ programėlės klasių diagrama ir sugeneruotos programėlės vaizdai 61
2 priedas. „Užduočių sekimas“ programėlės klasių diagrama ir sugeneruotos programėlės vaizdai 63
3 priedas. „Užrašinė“ programėlės Java failų struktūra ir sukurtos programėlės vaizdai 66
4 priedas. „Užduočių sekimas“ programėlės Java failų struktūra ir sukurtos programėlės vaizdai 68

Lentelių sąrašas

1 lentelė Populiariausios kategorijos 2021 neoficialiais duomenimis [4] 15
2 lentelė Įrankių palyginimas 21
3 lentelė Panaudojimo atvejų aprašymas 26
4 lentelė „Registruotis“ PA specifikacija 28
5 lentelė „Prisijungti“ PA specifikacija 29
6 lentelė „Atsijungti“ PA specifikacija 30
7 lentelė „Peržiūrėti saugyklų sąrašą“ PA specifikacija 31
8 lentelė „Kurti saugyklą“ PA specifikacija 32
9 lentelė „Trinti saugyklą“ PA specifikacija 33
10 lentelė „Peržiūrėti saugyklą“ PA specifikacija 34
11 lentelė . „Įkelti naują modelių projekto failą“ PA specifikacija 35
12 lentelė „Generuoti „Android“ programėlės kodą“ PA specifikacija 36
13 lentelė „Generuoti „Android“ programėlę“ PA specifikacija 37
14 lentelė „Peržiūrėti naudotojų sąrašą“ PA specifikacija 38
15 lentelė „Užšaldyti/atstatyti naudotoją“ PA specifikacija 39
16 lentelė „Peržiūrėti saugyklų sąrašą“ PA specifikacija 40
17 lentelė „Užšaldyti / atstatyti saugyklą“ PA specifikacija 41
18 lentelė Stereotipų aprašymai 45
19 lentelė Duomenų tipų „MagicDraw“ profilis 47
20 lentelė Sugeneruotos „Užrašų“ programėlės rezultatai 52
21 lentelė Atnaujintų stereotipų aprašymai 53
22 lentelė Eksperimento rezultatai 57

Paveikslų sąrašas

1 pav. Duomenų rinkinio dydis ir „Google Play“ bei 16 Kinijos rinkų ypatybės [3] 15
2 pav. MDE proceso atvaizdavimas 16
3 pav. Metamodelio loginis ryšys su modeliu ir realiu pasauliu [28] 18
4 pav. Klasės diagrama – struktūros vaizdas projekto „*Snake*“ [23] 19
5 pav. Siūlomas metodas „Android“ programėlių kūrimui [24] 20
6 pav. Tyrimo rezultatai [24] 21
7 pav. Panaudojimo atvejų modelis 26
8 pav. „Registruotis“ PA 28
9 pav. „Prisijungti“ PA 29
10 pav. „Atsijungti“ PA 30
11 pav. „Peržiūrėti saugyklų sąrašą“ PA 31
12 pav. „Kurti saugyklą“ PA 32
13 pav. „Trinti saugyklą“ PA 33
14 pav. „Peržiūrėti saugyklą“ PA 34
15 pav. „Įkelti naują modelių projekto failą“ PA 35
16 pav. „Generuoti „Android“ programėlės kodą“ PA 36
17 pav. „Generuoti „Android“ programėlę“ PA 37
18 pav. „Peržiūrėti naudotojų sąrašą“ PA 38
19 pav. „Užšaldyti/atstatyti naudotoją“ PA 39
20 pav. „Peržiūrėti saugyklų sąrašą“ PA 40
21 pav. „Užšaldyti / atstatyti saugyklą“ PA 41
22 pav. Komponentų diagrama 42
23 pav. WEB paslaugos paketų diagrama 43
24 pav. WEB paslaugos komponento klasių diagrama 43
25 pav. Generatorių komponento paketų diagrama 44
26 pav. „Android“ kodo generatoriaus klasių diagrama 44
27 pav. Įrankio „MagicDraw“ stereotipų profilis 46
28 pav. „MagicDraw“ kintamųjų tipų profilis 47
29 pav. Programėlės „Užrašinė“ klasių diagrama 49
30 pav. Saugyklos peržiūros langas 49
31 pav. Saugyklos peržiūros langas pradėjus programėlės generavimo procesą 50
32 pav. Saugyklos peržiūros langas sugeneravus programėlę ir programėlės kodą 51
33 pav. Sugeneruotos „Užrašinės“ programėlės vaizdai 52
34 pav. Atnaujintas įrankio „MagicDraw“ stereotipų profilis 54
35 pav. „Užrašinė“ programėlės klasių diagrama 61
36 pav. „Užrašinė“ programėlės sugeneruota Java failų struktūra 61
37 pav. „Užrašinė“ programėlės sugeneruota naudotojo sąsaja 62
38 pav. „Užduočių sekimo“ programėlės klasių diagrama 63
39 pav. „Užduočių sekimo“ programėlės sugeneruota Java failų struktūra 63
40 pav. „Užduočių sekimo“ programėlės sugeneruota naudotojo sąsaja, peržiūrint užduotis 64
41 pav. „Užduočių sekimo“ programėlės sugeneruota naudotojo sąsaja, peržiūrint kategorijas 65
42 pav. „Užrašinė“ programėlės sukurta Java failų struktūra 66
43 pav. „Užrašinė“ programėlės sukurta naudotojo sąsaja 67
44 pav. „Užduočių sekimo“ programėlės sukurta Java failų struktūra 68

45 pav. „Užduočių sekimo“ programėlės sukurta naudotojo sąsaja, peržiūrint užduotis..... 69
46 pav. „Užduočių sekimo“ programėlės sukurta naudotojo sąsaja, peržiūrint kategorijas..... 70

Santrumpų ir terminų sąrašas

Santrumpos:

CRUD – (angl. creare, read, update, delete)

KissMDA – (angl. Keep It Simple Stupid, MDA) karkaso pavadinimas.

M2M – (angl. Model to Model)

M2T – (angl. Model to Text)

MDA – (angl. Model-driven architecture)

MDE – (angl. Model-driven engineering)

MIT – (angl. Massachusetts Institute of Technology)

MOF – (angl. Meta-Object Facility)

MOFM2T – (angl. Meta-Object Facility Model to Text)

OMG – (angl Object Management Group)

SDK – (angl. Software development kit)

UML – (angl. The Unified Modeling Language)

XML – (angl. Extensible Markup Language)

Terminai:

„Android SDK“ – kūrimo įrankių rinkinys, naudojamas kuriant „Android“ platformos programas. Į tai gali įeiti: įvairios bibliotekos, derintuvė (angl. *debugger*), emuliatorius, dokumentacija apie „Android API“, kodo pavyzdžiai.

„Cartridge“ – taisyklių rinkinys, nusakantis kaip transformuoti atitinkamus modelių stereotipus.

KissMDA – MDA karkasas parašytas ant Java ir suteikianti Java API.

MDA (*angl. Model-driven architecture*) – į modelį orientuota architektūra yra programinės įrangos projektavimo metodika, taikoma programinių sistemų kūrimui. MDA pateikia gaires, kaip struktūrizuoti specifikacijas, kurios pateikiamos kaip modeliai. Visas dėmesys yra skiriamas programos modeliams, kurie vėliau yra naudojami automatinio kodo generavime atitinkamoms platformoms. Šis standartas yra lankstesnis norint programą išleisti ant kelių platformų.

MDE (*angl. Model-driven engineering*) – į modelį orientuota inžinerija (MDE), tai požiūris į programinės įrangos kūrimą, kai modeliai yra pagrindiniai kūrimo proceso rezultatai. Programinės įrangos dėka modeliai vėliau yra automatiškai transformuojami į vykdomas programas ar programos kodą.

Saugykla (*angl. Vault*) – projekte traktuojama kaip naudotojo vieta, kuri yra skirta įkelti sukurtus modelius ir valdyti kuriamą programėlę. Visa informacija apie programėlės generavimą yra saugoma prie naudotojo saugyklos.

Modelis – elementas, kuris laiko informaciją apie vieną „Android“ programėlės modelį, pavyzdžiui tai gali būti UML klasė su atitinkamais atributais.

XML (*angl. Extensible Markup Language*) – žymėjimo kalba, apibrėžianti dokumentų kodavimo taisyklių rinkinius, kuriuos supranta kompiuteris, tačiau juos lengviau skaityti ir žmonėms.

1. Įvadas

Mobiliųjų programėlių kūrimo pramonė vis labiau auga dėl intensyvaus programų taikymo mobiliuosiuose įrenginiuose, iš kurių dauguma naudoja „Android“ operacinę sistemą. Pagal 2018 metų sausio mėnesio duomenis „Google Play“ pasiekė daugiau kaip 2.0 mln. patvirtintų ir patikrintų mobiliųjų programėlių, kurias naudotojai lengvai gali parsisiųsti ir naudoti.

Problema. Tačiau kuriant programėles mobiliosioms platformoms iškyla papildomi rūpesčiai, tokie kaip: kodo efektyvumas, sąveika su įrenginio ištekliais, trumpas laiko tarpas pateikimui į rinką. Būtent šis projektas yra orientuojamas į šių problemų sprendimą.

Į modelį orientuota inžinerija (angl. Model-driven enginnering) tai – požiūris į programinės įrangos kūrimą, kai pagrindiniai kūrimo proceso rezultatai yra modeliai, o ne programos dalys [1], [2]. Vėliau modeliai yra automatiškai transformuojami į programą ar programos kodą. Į modelį orientuota inžinerija (MDE) kartu su UML, kaip jau naudojama programinės įrangos inžinerijoje, galėtų suteikti abstrakciją ir automatizavimą mobiliosios programinės įrangos kūrėjams. Dažnai projektuojant dideles sistemas yra naudojamos vizualios diagramos ar modeliai, norint pavaizduoti programos logiką ar veikimą, kadangi prie projekto dirba įvairaus lygio ir krypčių specialistai, nuo verslo analitikų, projektuotojų, dizainerių iki programuotojų.

Tikslas. Norima sukurti įrankį pagrįstą MDE principu, kuris leistų „Android“ programėlės suprojektuotus modelius automatiškai transformuoti į „Android“ programėlę ar programėlės kodą. Tai palengvintų kūrėjų darbą ir pagerintų produkto kokybę, nes sugeneruotas kodas jau būtų optimizuotas ir galimi resursai būtų tinkamai naudojami (baterija, procesorius, atmintis). Šiam tikslui yra keliami darbo uždaviniai:

1. išanalizuoti MDE metodiką ir esamų technologijų padėtį;
2. suprojektuoti ir realizuoti programų sistemą;
3. išanalizuoti sukurtą programų sistemą ir atlikti atnaujinimo, papildymo darbus;
4. eksperimentiškai ištirti ir įvertinti sukurtą metodiką bei sistemą.

Darbo struktūra: Antrajame „Rinkos, projektavimo metodologijos ir technologijų analizė“ skyriuje yra gilinamasi į „Android“ programėlių rinką, atliekama MDE metodikos analizė ir analizuojami moksliniai straipsniai, kuriuose bandoma pritaikyti MDE metodiką „Android“ programėlių kūrime. Trečiajame „Projektinė dalis“ skyriuje yra pateikiama informacija apie iškeltus sistemos funkcinius ir nefunkcinius reikalavimus, detalizuojama sistemos architektūra ir veikimas. Ketvirtajame „Sukurtos programinės sistemos analizavimas“ skyriuje buvo išbandoma sukurta programinė sistema, analizuojami rezultatai, siūlomi pakeitimai, detalizuojama atliktų pakeitimų eiga ir rezultatai. Penktajame „Eksperimentinis „Android“ programėlių kūrimo tyrimas“ skyriuje eksperimentiniu tyrimo būdu buvo pagrįsta ar sukurta programinė sistema, kuri yra pagrįsta MDE metodika, pagreitina „Android“ programėlių kūrimą lyginant kūrimo procesą su tradiciniu programėlių kūrimu.

2. Rinkos, projektavimo metodologijos ir technologijų analizė

2.1. Analizės tikslas

Analizės metu siekiama išanalizuoti į modelį orientuotos inžinerijos (MDE) požiūrį ir rasti galimus sprendimo būdus, norint pritaikyti MDE „Android“ programėlių kūrimo procese.

Pirmiausia siekiama išanalizuoti į modelį orientuotos inžinerijos požiūrį, metodiką, pranašumus ir trūkumus. Toliau siekiama išnagrinėti esamus sprendimus ir egzistuojančią programinę įrangą, kuri yra pagrįsta MDE metodika. Išanalizuoti moksliniuose darbuose siūlomus sprendimus ir bandymus, pritaikant MDE metodiką „Android“ programėlių kūrime. Taip pat reikia identifikuoti galimas problemas, kurios gali iškilti norint kurti „Android“ programėles, taikant MDE metodika pagrįstus įrankius ar sprendimus.

Analizės uždaviniai:

1. Išanalizuoti į modelį orientuotos inžinerijos požiūrį, privalumus, trūkumus.
2. Išanalizuoti į modelį orientuotos inžinerijos metodikos pritaikymą „Android“ programėlių kūrime.
3. Išanalizuoti ir palyginti galimas sistemas pagrįstas MDE metodika.

2.2. Srities apžvalga

Šioje analizėje yra analizuojamas ir pristatomas vienas iš programų sistemų kūrimo procesų, kuris kilo iš sistemų modeliavimo, tai į modelį orientuota inžinerija (MDE). Taip pat analizuojama „Android“ programėlių rinka ir susiję tyrimai bei bandymai, norint pritaikyti į modelį orientuotą inžineriją (MDE) „Android“ programėlių kūrime.

2.3. „Android“ programėlių rinkos analizė

Kaip žinoma „Android“ programėlių rinka sparčiai auga ir plinta. „Android“ operacinė sistema vis labiau tobulėja ir vis plačiau naudojama interaktyviuose įrenginiuose.

2018 metais buvo atliktas didelės apimties palyginimas tarp Kinijos ir „Google Play“ rinkų, kurioje bandoma palyginti „Android“ programėlių rinką tarp didžiausių „Android“ programėlių platintojų, palyginimo rezultatai pateikiami 1 paveikslėlyje. Pateiktame duomenų rinkinyje galima matyti, kad „Android“ programėlių kūrėjų yra daugiau negu 1 milijonas, o pačių programėlių visose rinkose yra daugiau negu 6.2 milijono [3].

Market	Type	Size (#Apps)	Aggregated Downloads	#Developers	% Unique Developers	Openness	Copyright Check	App Vetting	Security Check	Vetting Time	Quality Rating	Incentive#1	Incentive#2	Incentive#3	Privacy Policy	Advertisement	In-app Purchase
Google Play	Official	2,031,946	193 B	538,283	57.04	✓	✓	✓	✓	Hours	✓		✓	✓	✓	✓	✓
Tencent Myapp	Web Co.	636,265	82 B	294,950	10.61	✓	✓	✓	✓	1 day	✓	✓	✓	✓		✓	
Baidu Market	Web Co.	227,454	94 B	107,698	15.10	✓	✓	✓	✓	1-3 days		✓				✓	
360 Market	Web Co.	163,121	50 B	90,226	6.80	✓	✓	✓	✓	1 day	✓	✓	✓			✓	✓
OPPO Market	HW Vendor	426,419	57 B	209,197	14.37	Partial[1]	✓	✓	✓	1-3 days		✓				✓	
Xiaomi Market	HW Vendor	91,190	-	55,669	5.78	✓	✓	✓	✓	1-3 days		✓					
MeiZu Market	HW Vendor	80,573	19 B	50,451	0.58	✓	✓	✓	✓	1-3 days		✓					
Huawei Market	HW Vendor	51,303	83 B	32,927	5.66	✓	✓	✓	✓	3-5 days		✓	✓	✓		✓	
Lenovo MM	HW Vendor	37,716	24 B	24,565	0.79		✓	✓	✓	2 days		✓					
25PP	Specialized	1,013,208	56 B	470,073	19.06	✓	✓	✓	✓	1-3 days		✓	✓			✓	
Wandoujia	Specialized	554,138	38 B	291,114	0.97	✓	✓	✓	✓	1-3 days		✓	✓				
HiApk	Specialized	246,023	17 B	115,191	3.65	✓	N/A	N/A	N/A	N/A							
AnZhi	Specialized	223,043	12 B	74,145	21.93	✓	✓	✓	✓	1-3 days		✓					
LIQU	Specialized	179,147	26 B	101,336	6.10	✓	✓	✓	✓	N/A		✓					
PC Online	Specialized	134,863	0.2 B	65,225	2.58	✓	N/A	N/A	N/A	N/A							
Sougou	Specialized	128,403	3 B	66,759	4.04	✓	✓	✓	✓	1 day		✓	✓			✓	
App China	Specialized	42,435	-	23,699	3.22	✓	✓	✓	✓	1-3 days						✓	
Total		6,267,247	754 B	1,035,992													

1 pav. Duomenų rinkinio dydis ir „Google Play“ bei 16 Kinijos rinkų ypatybės [3]

Neoficialiais duomenimis ir dabar yra teigiama, kad „Google Play“ platformoje yra daugiau negu 2.9 milijono „Android“ programėlių. Populiariausių programėlių kategorijos yra: „Education“, „Bussness“, „Music and Audio“, „Tools“ ir „Entertainment“, žiūrėti 1 lentelę [4].

1 lentelė Populiariausios kategorijos 2021 neoficialiais duomenimis [4]

Kategorija	Viso programėlių	Programėlių kiekis turinčios > 50K atsisiuntimų
Education	281954	19081
Business	200743	5482
Music & Audio	198934	11773
Tools	167167	20906
Entertainment	164482	20485
Lifestyle	143782	11750
Books & Reference	134446	12546
Food & Drink	129276	2831
Shopping	121663	6216
Productivity	112273	7011
Personalization	111368	14067
Health & Fitness	100448	5963
Finance	87184	10368
Travel & Local	85453	4976
Communication	68671	5376
Arcade	60042	5279
Social	59568	5191
Puzzle	59411	7395

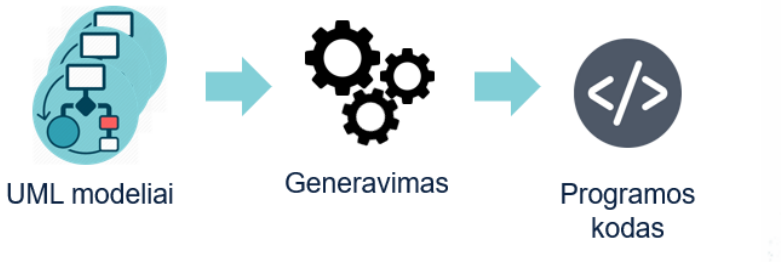
2.4. Sistemos modeliavimas

Sistemos modeliavimas padeda abstrakčiai atvaizduoti sistemą naudojant grafinį žymėjamą, kas būtent žmogui padeda greičiau suprasti sistemą ar pavaizduotą procesą. Šiuolaikinėje rinkoje sistema gali turėti neišpasakytą dydį, dėl to yra naudojamas grafinis žymėjamas, pavyzdžiui UML (angl. *Unified Modeling Language*) [1]. Palyginimui galima sakyti, kad sistemos modeliavimas yra tas pats kaip pastatų ar namų projektavimas.

Būtent šiuolaikinėje rinkoje, kuomet prie vienos sistemos gali dirbti tūkstančiai žmonių iš įvairių rinkos sričių, yra reikalingas bendras komunikavimo metodas ir tai gali suteikti sistemų modeliavimas, naudojant standartais pagrįstas metodikas ar grafinius žymėjimus. Tačiau suprojektuoti ir dokumentuoti sistemos neužtenka, ją reikia ir realizuoti, būtent šioje vietoje atsiranda procesai, kurie gali palengvinti, net retkarčiais pagreitinti realizavimą. Vienas iš šių procesų yra MDE.

2.4.1. MDE *(angl. Model-driven engineering)*

Į modelį orientuota inžinerija (MDE) – požiūris į programinės įrangos kūrimą, kai modeliai yra pagrindiniai kūrimo proceso rezultatai [1], [2]. Programinės įrangos dėka modeliai vėliau yra automatiškai transformuojami į vykdomas programas ar programos kodą, žiūrėti 2 paveikslą. MDE šalininkai argumentuoja, kad tai yra vienas iš intelektualiausių programinės įrangos abstrakcijos lygių.



2 pav. MDE proceso atvaizdavimas

Viena iš priežasčių kodėl MDE sulaukia išskirtinio dėmesio yra, tai, kad pagrindinis proceso produktas yra modeliai, kurie yra kuriami ir atnaujinami visuose programos kūrimo bei palaikymo etapuose [5], [6]. Tačiau MDE vis dar yra ankstyvoje vystymosi stadijoje, kadangi nėra aišku ar šis procesas turi kokį poveikį sistemų inžinerijos praktikose. Šiuo metu nuomonės yra skirstomos į už ir prieš MDE:

Už MDE – MDE pagrindu sukurta inžinerija leidžia inžinieriams galvoti apie aukščiausio lygio abstrakcijos sistemas, nesirūpinant dėl jų įdiegimo detalių. Tai sumažina klaidų tikimybę, pagreitina projektavimo ir diegimo procesą bei leidžia sukurti daugkartinio naudojimo nuo platformos nepriklausomus taikymo modelius. Naudojant galingus įrankius, sistemos gali būti kuriamos iš to paties modelio skirtingoms platformoms. Norint pritaikyti sistemą kitai platformos technologijai, tereikia aprašyti pasirinktos programinės platformos transformacijas. Paruošus kelių platformų transformacijos taisykles, atsiranda galimybė greitai atlikti pakeitimus tarp kelių platformų atnaujinant modelius [7], [8].

Prieš MDE – tai buvo aptarta, kad sistemos architektūra gali būti puikiai atvaizduota modeliais. Tačiau yra sunku nustatyti ar modelio palaikomos abstrakcijos yra tinkamos abstrakcijos, kurias reikia įgyvendinti. Problema yra ta, kad galima sukurti informatyvų modelį, tačiau funkcionalumui realizuoti gali būti panaudotas nestandartinis (*angl. off-the-shelf*) sprendimas su atitinkama konfigūracija ir tai gali sukelti įvairias problemas norint pritaikyti MDE principus ir standartus [7], [8].

Tačiau negalima sakyti, kad MDE nėra panaudojamas. OMG grupė savo tinklalapyje yra pranešę apie reikšmingas MDE/MDA sėkmės istorijas, o metodais naudojasi didelės įmonės, tokios kaip IBM ir Siemens. Šie metodai buvo sėkmingai naudojami kuriant dideles, ilgalaikes programinės įrangos sistemas, tokias kaip oro eismo valdymo sistemas [7].

2.4.2. MDA *(angl. Model-driven architecture)*

MDA (Model-driven architecture) yra viena iš MDE atšakų, kuri buvo pristatyta „Object Management Group“ (OMG) 2001 metais [7]. Pagrindinė MDA idėja yra naudoti modelius kaip pagrindinius kūrimo artefaktus ir tokiu būdu galima atskirti platformos specifinius duomenis nuo programinės įrangos kūrimo proceso. Kuriant programas be konkrečių platformos sąlygų, sukurtas programas yra lengviau ir pigiau perkelti į skirtingas platformas [9]. Tai supaprastina visą kūrimo procesą, kadangi užtenka sistemą sumodeliuoti kartą ir išleisti programą ant skirtingų platformų su tais pačiais modeliais. Norint tai atlikti tereikia paruošti atitinkamas modelių transformacijas į norimą platformą, dažniausiai tai leidžia atlikti pasirinkta programinė įranga.

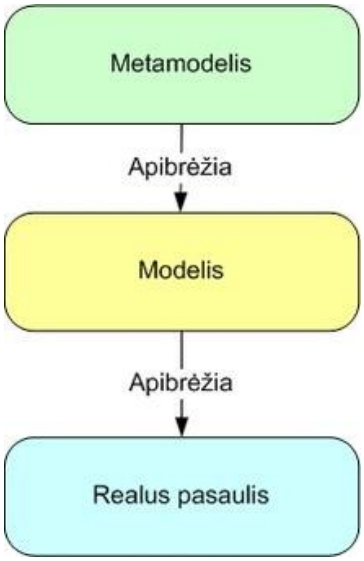
2.5. Pagrindiniai MDE standartai ir kalbos

Vieni iš pagrindinių MDE standartų ir kalbų:

- Metamodelis arba surogatinis modelis – modelio modelis, kiekvienas modelis turi sekti su savo metamodeliu [10].
- Modelio transformacijos kalba – kalba, leidžianti transformuoti modelį. Pavyzdžiui: ATL, *GReAT*, JTL, *Kermeta*, *Lx*, objektų valdymo grupės (OMG) standartas QVT, M2M *Eclipse*, paremtas QVT standartu [11], [14].
- Modelio į tekstą transformavimo kalbos leidžia modeliuoti kodą. Pavyzdžiui: MOFM2T (*angl. Meta-Object Facility Model to Text*) paremtas QVT standartu, M2T (*angl. Model to Text*) EGL „*angl. Eclipse Epsilon Generation Language*“ [12], [14].
- Domeno specifinė kalba DSL (*angl. Domain specific Language*) leidžia kurti metamodelius. Pavyzdžiui: MOF: OMG, JMI Java API, skirta manipuliuoti MOF modeliais [11], [12], [14], [15], [16].
- Keitimasis XML metaduomenimis (XMI) – OMG standartas, leidžiantis modelį konvertuoti į XML [11], [12], [14].
- Žiniatinklio ontologijos kalba (OWL) – semantinė žymėjimo kalba ontologijoms publikuoti ir dalintis internete [13], [14].
- Bendrasis sandėlio metamodelis CWM (*angl. The Common Warehouse MetaModel*) – standartizuoja išsamų, visapusišką metamodelį, kuris įgalina „*data mining*“ [14].

2.5.1. Metamodelis

Metamodelis – modeliavimo kalbos modelis, kuriame apibrėžtos esminės kalbos savybės. Metamodelis turi būti tikslus išreikštinių darinių (*angl. artifacts*) ir taisyklių apibrėžimas, kuris reikalingas semantinių arba dalykinės srities modelių kūrimui. Metamodelį sudaro: koncepcijos, kurias pats metamodelis palaiko, tekstinė sintaksė, grafinė sintaksė, semantika [17]. Kaip ir ankščiau buvo aptarta, kad modeliai apibrėžia realų pasaulį, tačiau metamodeliai apibrėžia modelius, žiūrėti 3 paveikslą.



3 pav. Metamodelio loginis ryšys su modeliu ir realiu pasauliu [28]

Metamodeliai yra naudojami norint abstrakčiai apibūdint modelius. Pavyzdžiui metamodelis gali būti matematinis ryšys arba algoritmas, vaizduojantis įvesties ir išvesties ryšius.

2.5.2. „Microsoft“ domeno specifinė kalba DSL

Skirtingai nuo bendrosios programavimo ar modeliavimo paskirties kalbų, tokių kaip C # ar UML, domenui būdinga kalba (DSL), skirta išreikšti teiginius tam tikroje probleminėje erdvėje ar srityje [18]. Būtent „Microsoft“ teikia įrankius galinčius padėti specifikuoti atitinkamą DSL, kuris būtų naudojamas automatinio kodo generavimui:

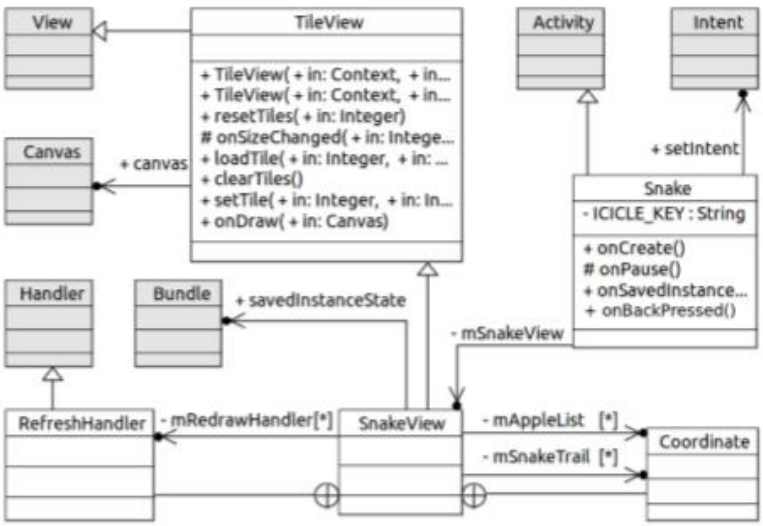
- Suteikia galimybę dirbti su domeno kalba per grafinį dizainerį (nuoseklųjį XML formatu), su kuriuo galima apibrėžti ir redaguoti DSL [10], [19].
- Leidžiama nustatyti dizainerio apibrėžimus naudojant patentuotą XML formatą. Šis formatas yra šaltinis generuoti kodą (be jokio rankinio įsikišimo) [10], [19].
- Yra galimybė naudotis iš anksto nustatytais kodo generatoriais, kurie gali naudoti sukurtą DSL [15].
- Galimybė naudotis sukurtais karkasais, pagal šabloną, šiam metodui naudoti yra reikalingi domeno modeliai, norint generuoti kodą pagal šabloną [15].

2.6. Metodai, technologijos, sprendimai kuriant „Android“ programėles naudojant MDE

Šiuo metu yra labai platus įrankių pasirinkimas, kurie palaiko programų sistemų modeliavimą naudojant UML notacijas ir gaires. Kai kurie iš tokių įrankių gali sugeneruoti kodą iš UML modelių, pavyzdžiui „UModel“ [20] ar „Modelio“ [21]. Tačiau tokie įrankiai yra susitelkę į tradicinį programų sistemų kūrimo procesą, kuomet modeliavimo programa gali sugeneruoti sumodeliuotas klases ir metodus, nepriklausomai nuo programavimo kalbos. Sugeneruoti rezultatai perduodami programuotojui, kuris turėtų integruoti gautus rezultatus.

Norint palengvinti „Android“ programėlių kūrimo procesą buvo išleista daug įvairių įrankių, kurie yra panašūs į vizualų programavimą. Vienas iš pavyzdžių būtų „App Inventor“ [22], kurį galima pasiekti per internetinę naršyklę ir vizualiai sukurti asmeninę „Android“ programėlę. Tačiau kaip ir minėta šis metodas yra labiau panašus į vizualų programavimą negu į modeliavimą. Naudotojui yra leidžiama vizualiai sukurti savo programėlę naudojant naudotojo sąsajos laukus, specifikuojant kiekvieno lauko funkcijas iš galimų funkcijų sąrašo dėliojant kodo blokus.

Šiuo metu ieškant įrankio, kuris būtų pagrįstas MDE principu ir generuotų pilnai „Android“ programėles, sėkmingai aptikti nepavyko, gal to priežastis būtų ta, kad MDE yra jaunas procesas, kuris vis dar auga ir tobulėja. Taip pat gal to priežastis, kad nėra rasto bendro sprendimo, kaip geriau pritaikyti MDE principus „Android“ programėlių kūrime. Tačiau pavyko aptikti bandymus norint pritaikyti MDE principus „Android“ programėlių kūrimui. A. Parados ir L. Brisolaros straipsnyje apie „*A model driven approach for „Android“ applications development*“ yra aptariamas vienas iš sprendimų, kaip buvo pasiekta sėkmingai pritaikyti MDE principus „Android“ programėlės kūrime. A. Parados ir L. Brisolaros straipsnyje apie „*A model driven approach for „Android“ applications development*“ sprendimas buvo praplėsti „*GenCode*“ įrankį, kad naudojantis įrankiu iš pateiktų modelių būtų galima automatiškai sugeneruoti „Android“ programėlės kodą. Modeliams realizuoti buvo pasirinkta naudoti UML klasių ir veiksmų sekų diagramas norint atvaizduoti programos logiką ir sudėtį. Sprendimui patikslinti buvo atlikta atvejo analizė, kuriai buvo pasirinktas atviro kodo projektas „*The Snake*“, žiūrėti 4 paveikslą.

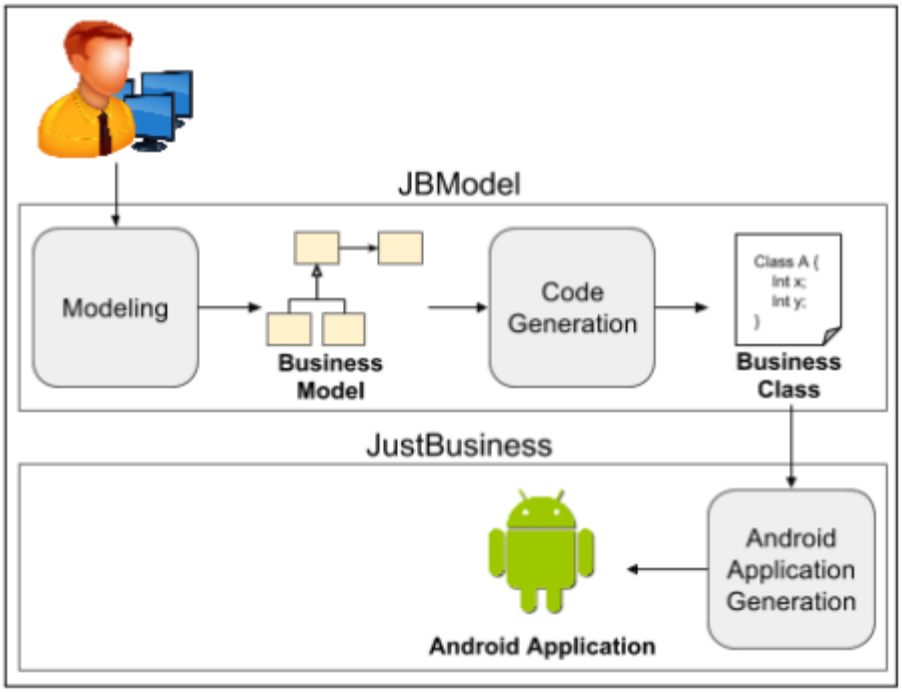


4 pav. Klasės diagrama – struktūros vaizdas projekto „*Snake*“ [23]

Projektas buvo sėkmingas ir iš pateiktų modelių pavyko automatiškai sugeneruoti „Android Java“ programos kodą. Tačiau A. Parada ir L. Brisolara teigia, kad „*GenCode*“ atliko tik transformacijos procesą tarp UML modelių ir „Android“ programos kodo. Nebuvo atliktas kodo optimizavimo procesas, kuris taupytų įrenginio resursus [23].

F. Fabiano, Paulo Henrique M. Maia straipsnyje apie „*JustModeling: An MDE Approach to Develop Android Business Applications*“ pristatomas detalesnis sprendimo būdas, pritaikant MDE principus „Android“ programėlių kūrime.

Naudojantis „JBModel“ programų sistemų modeliavimo programa ir metamodeliu yra modeliuojama „Android“ programėlės klasių diagrama ir asociacijos tarp klasių. Užbaigus modeliavimo darbus modelis yra transformuojamas į „JustBusiness“ kodą, naudojant integruotą M2T transformacijos būdą „JBModel“. „JustBusiness“ kodas yra naudojamas „Android“ programėlės generavimui „JustBusiness“ įrankyje, žiūrėti 5 paveikslėlį [24].



5 pav. Siūlomas metodas „Android“ programėlių kūrimui [24]

Sukurtam sprendimui buvo iškeltas tyrimas, norint įsitikinti ar sukurtas sprendimas pagreitina programėlių kūrimą. Tyrime buvo lyginama vienos „Android“ programėlės kūrimo laikas naudojant skirtingus kūrimo būdus: tradicinis programėlės kūrimo būdas, „JustBussiness“ programėlių kūrimo būdas ir tik modeliavimo būdas naudojantis „JBModel“ ir „JustBussiness“, žiūrėti 6 paveikslėlį [24].

Type of Development	Traditional	JustBusiness	JustModeling
Time	720 minutes	27 minutes	18 minutes
Lines written by Developer	4315	376	0
Files created by Developer	66	4	1
Total files in the Project	66	66	66

6 pav. Tyrimo rezultatai [24]

F. Fabiano, Paulo Henrique M. Maia teigia, kad tyrimo rezultatai yra teigiami ir pasiūlytas sprendimas įrodo, kad kuriant programėles modeliavimo būdu, naudojant „JBModel“ ir „JustBusiness“, dramatiškai sumažėja programėlės kūrimo laikas ir programuotojo parašytų programos kodo eilučių kiekis. Žinoma sukurtas sprendimas dabar leidžia tik sumodeliuoti klasių struktūrą, tačiau papildomų metodų įterpimas reikalauja rankinio įterpimo, taip pat ne visos anotacijos „JBModel“ įrankyje yra galimos, kurios reikalingos „JustBusiness“ įrankiui, kas taip pat gali reikalauti rankinio įsikišimo [24].

2.7. Egzistuojančių programų sistemų palyginimas

Atlikus rinkos analizę nebuvo rasta pilnai užbaigtų įrankių, kurie iškart be papildomos konfigūracijos ar pakeitimų (*angl. out of the box*) generuotų „Android“ programėles iš suprojektuotų programėlės modelių. Šiuo metu rinka yra labiau susitelkusi į vizualaus programavimo įrankius, kas yra labiau sukurtų funkcijų pasirinkimas kuriant mobilios platformos programėles. Tačiau buvo rasti vieni iš populiaresnių ir aktyvesnių transformacijos karkasų, kurie yra pagrįsti MDE principu, tai: „AndroMDA“ [25], „OpenMDX“ [26] ir „TigerTeam – TRIMM – Model Driven Generator“ [27]. Visi šie įrankiai yra atviro kodo, turi atitinkamas konfigūracijas, palaiko daug programavimo kalbų (Java SE, Java EE, .Net, PHP). Yra galimybė apsirašyti asmenines transformacijas, kurios leistų sugeneruoti norimos platformos kodą iš pateiktų modelių. Taip pat yra galimybė naudoti metamodelį, kuris padėtų apibrėžti modelius, palengvinti modelių atpažinimą transformacijos procese.

2 lentelė Įrankių palyginimas

Kriterijus/sistema	**„AndroMDA“**	**„OpenMDX“**	**„TigerTeam - TRIMM“**	**KissMDA**
Atviro kodo	+	+	+	+
UML2/3	+	Tik UML 2	+	Tik UML 2
Nuotolinė versija	-	-	-	-
Naudotojo sąsaja	Tekstinė sąsaja	Tekstinė sąsaja	Tekstinė sąsaja	Tekstinė sąsaja
Papildoma	Reikalauja didelės konfigūracijos	-	-	-
Kaina	Nemokama	Nemokama	Nemokama	Nemokama

Su pateiktais įrankiais yra galimybė generuoti „Android“ programėlių kodą iš pateiktų modelių, tačiau norint pasinaudoti bent vienu iš įrankių, reikėtų sukurti į „Android“ platformą orientuotas transformacijos taisykles, pagal pasirinkto įrankio dokumentaciją, kurios nusakytų kaip turi būti generuojamas programėlės kodas iš pateiktų modelių. Taip pat įrankiai turi atitinkamas konfigūracijas, kurios yra būtinos norint pradėti darbą su pasirinktai įrankiais. Nei vienas įrankis neturi savo grafinės naudotojo sąsajos. Įrankiais galima naudotis tik parsisiuntus lokaliai arba pasitalpinti serveryje.

2.8. Analizės išvados

- Atsižvelgiant į MDE metodikos privalumus ir trūkumus manau, kad MDE metodika gali pagreitinti „Android“ programėlių kūrimo procesą. Tačiau norint realizuoti MDE transformacijos procesą, nenustačius kuriamų „Android“ programėlių funkcinių ribų, gali apsunkėti modeliavimo procesas, kadangi visa informacija apie programėlę turėtų būti pateikiama per modelius.

- Atlikus įvairių literatūros šaltinių analizę, buvo nustatyta kokie modeliai taikomi norint pritaikyti MDE metodiką „Android“ programėlių kūrimui, tai buvo klasių diagramos, veiksmų sekų diagramos.

- Sprendimui realizuoti bus pasinaudojama „KissMDA“ karkasu, norint greičiau ir stabiliau atlikti sistemos realizavimo darbus. Šis karkasas buvo pasirinktas dėl lengvesnės konfigūracijos, transformacijos proceso kūrimo. O kaip modeliavimo įrankiu bus naudojamas „MagicDraw 19“.

3. Projektinė dalis

3.1. Sistemos paskirtis

Dažnai vienos programėlės tipas yra išleidžiamas ant kelių platformų, tačiau tai gali reikalauti daug įvairių resursų (laiko, pinigų, žmogiškų išteklių). Dėl šios priežasties ir dėl sparčiai augančios mobiliųjų programėlių rinkos yra reikalingas įrankis, kuris galėtų palengvinti ir paspartinti „Android“ programėlių kūrimo procesą. Įrankiui norima pritaikyti MDE principą, kuris orientuoja programinės įrangos kūrimą aplink modelius, tai leistų programinės įrangos kūrėjui labiau orientuotis į realizuojamą programą, o platformos aspektus palikti įrankiui.

Atlikus rinkos, projektavimo metodologijos ir technologijų analizę buvo pastebėta, kad norint pritaikyti MDE geriau yra žinoti generuojamos programėlės ribas ar funkcionalumą, kad supaprastinti projektavimo procesą ir transformacijos realizavimą. Dėl šios priežasties tikimasi sukurti įrankį gebantį generuoti „Android“ programėles bent su CRUD (angl. *create, read, update, delete*) funkcionalumu.

3.2. Sistemos tikslai

Tikslas: Pagreitinti „Android“ programėlių kūrimo procesą.

Nauda: „Android“ programėles galima generuoti automatiškai.

Tenkinimo kriterijus: „Android“ programėlės kūrimas trunka trumpiau mobilių programėlių inžinieriui (modeliuojant programėlę ir pasinaudojant įrankiu), negu „Android“ programėlės programuotojui, programuojant tradiciniu būdu. Tenkinimo kriterijui išpildyti bus bandoma palyginti „Užrašinės“ programėlės sukūrimo laikus. Naudojantis įrankiu, skirtumas turi būti bent 30 % greičiau, negu nenaudojant įrankio.

3.3. Apribojimai sprendimui

Šiame skyriuje yra pateikiami apribojimai sprendimui:

1. **Apribojimas:** Projekto realizavime negalima naudoti jokių komercinių sprendimų, nuo kurių įrankis būtų priklausomas, norint juo naudotis.
   1.1. **Pagrindimas:** įrankis negali būti priklausomas nuo komercinių produktų, kadangi projektas bus skelbiamas kaip „atviro kodo“ po MIT licencija.
   1.2. **Tenkinimo kriterijus:** projekte nebuvo panaudoti komerciniai sprendimai, kurie sutrukdytų projektą paskelbti kaip „atviro kodo“ projektu po MIT licencija.

3.4. Diegimo aplinka

Projektas skirstomas į dvi dalis:

- WEB dalis – sistemos dalis, suteikianti naudotojo sąsają ir galimybę naudotis įrankiu per naršyklę, taip pat suteikia saugyklas, naudotojų paskyrų ir administravimo funkcionalumą.
- Kodo generatorius – komponentas, kuris atsakingas už programėlių kodo ir papildomų resursų generavimą.

Įrankį bus galima diegtis lokaliai arba į nutolusį serverį. Diegimo aplinkoje yra reikalinga bent PHP 7.2, duomenų bazių valdymo sistema (DBVS) „MySQL“ 8, Java SE 1.8.0_281, „Android“ platformos „SDKmanager“ su įdiegtais platformos įrankiais 30.0.5.

3.5. Nefunkciniai reikalavimai

3.5.1. Reikalavimai sistemos išvaizdai

Išvaizdos reikalavimai. Šviesi grafinė naudotojo sąsaja. Šviesioje grafinėje naudotojo sąsajoje elementai yra labiau pastebimi, stipriai neapkrauna akių, lengviau skaityti tekstą šviesioje darbo vietoje.

Stiliaus reikalavimai. Sistemos klaidos ar pranešimai apie klaidą turi būti pateikti raudoname fone. Klaidų pranešimai turi patraukti naudotojo dėmesį, norint priversti naudotoją pastebėti, kad yra klaida ir atliekamas veiksmas nėra galimas. Sistemos pranešimai apie sėkmingą operaciją turi būti pateikti žaliame fone. Pranešimai apie sėkmingą operaciją ar veiksmą turi būti ne tik pateikti tekstu, bet ir spalva. Žalia spalva mūsų gyvenime asocijuojasi su leidimu, teisingumu ir kita. Tai greitas signalas parodyti, kad veiksmas atliktas be kliūčių.

3.5.2. Reikalavimai panaudojamumui

Naudojimosi paprastumo reikalavimai. Sistema turi pateikti patvirtinimo pranešimus prieš atliekant kūrimo, atnaujinimo, trynimo operacijas. Kadangi naudotojai gali netyčia paspausti arba atlikti sistemos veiksmą per klaidą, nenorėdami to atlikti.

Mokymosi reikalavimai. Įrankiu jau turėtų sugebėti naudotis dalykinės srities naudotojai po susipažinimo su „greitąja“ dokumentacija. Nes įrankiu gali naudotis įvairios IT patirties turintys naudotojai iš dalykinės srities, dėl to įrankis neturi būti apkrautas nereikalingais elementais, operacijos žingsniai turi būti aiškūs naudotojui, tačiau sudėtingose operacijose turi būti aiški dokumentacija kuri visuomet galėtų pagelbėti naudotojui kaip teisingai atlikti atitinkamą operaciją įrankyje.

Suteikiami patogumai. Įrankyje prie veiksmų turėtų būti nuorodos į dokumentaciją į atitinkamą veiksmą. Nes įrankiu gali naudotis įvairios IT patirties turintys naudotojai iš dalykinės srities, dėl to atliekant atitinkamus veiksmus gali būti sudėtinga, o pagalbinės nuorodos į dokumentaciją naudotojams gali pagelbėti greičiau atlikti atitinkamus veiksmus įrankyje.

3.5.3. Reikalavimai vykdymo charakteristikoms

Reikalavimai užduočių vykdymo greičiui. Naudotojui generuojant programėlės kodą ar programėlę neturi trukti ilgiau negu 10 minučių, jei nurodytoje saugykloje modelių yra mažiau arba lygų 50. Tikimasi, kad darbas su įrankiu turi būti greitas ir efektyvus, sistema neturėtų stabdyti darbo proceso.

3.5.4. Reikalavimai veikimo sąlygoms

Reikalavimai darbui su gretimomis sistemomis. Įrankis turi būti pasiekiamas per „Chrome“, „Firefox“ ir „Safari“ internetines naršykles. Įrankis turėtų būti projektuojamas taip, kad būtų galima

įrankį pasiekti per internetinę naršyklę ir nereikalautų atskiro įdiegimo į kiekvieną įrenginį, norint jį naudoti tarp daug įrenginių.

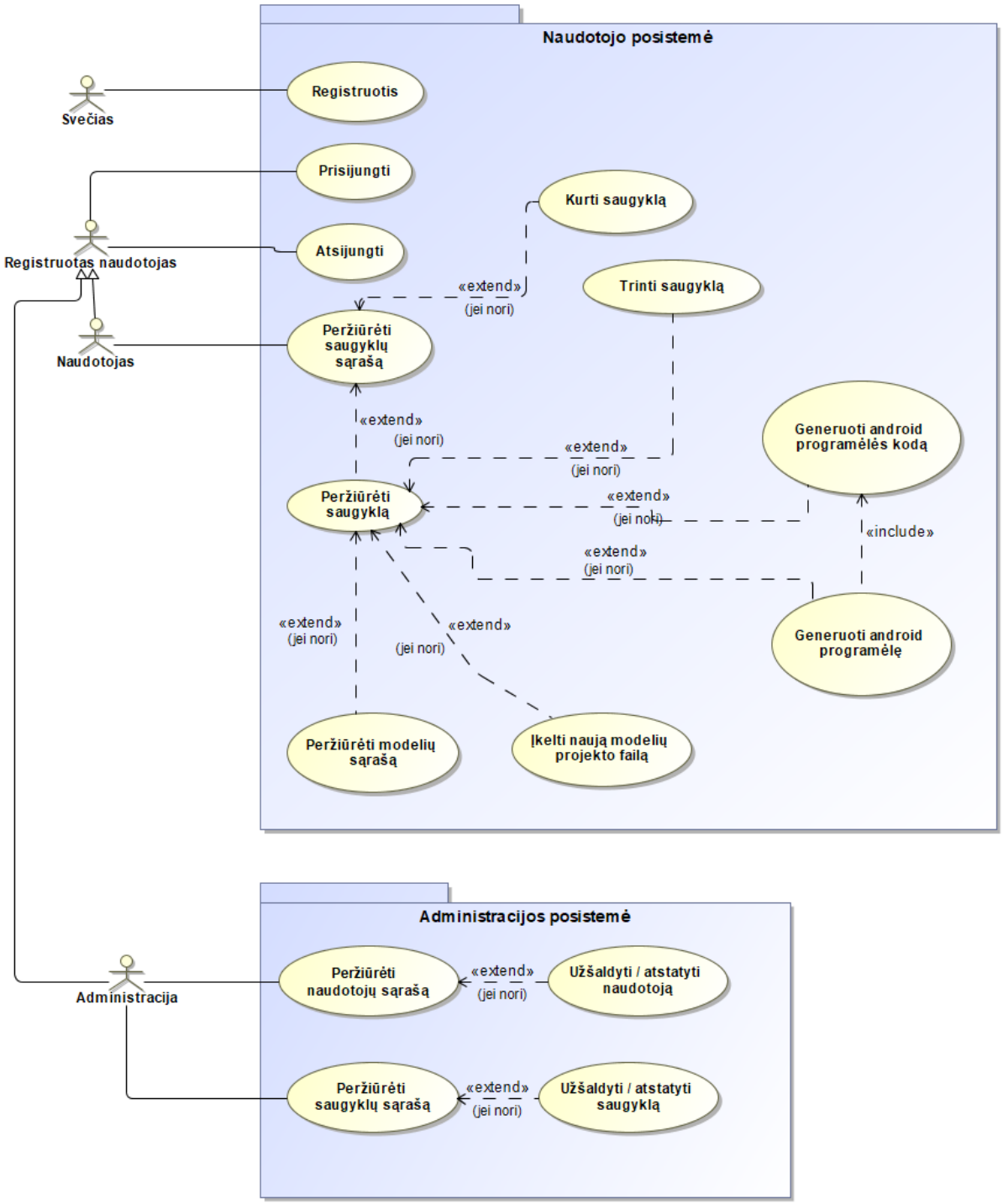
3.5.5. Reikalavimai saugumui

Prieigos reikalavimai (teisės). Administracija gali matyti dalį įrankio duomenų, tačiau naudotojai gali matyti tik viešai paskelbtą informaciją arba tik savo informaciją. Sistema yra padalyta į posistemes, kuriose egzistuoja atitinkamo lygio naudotojai turintys skirtingas teises sistemoje, tačiau turintys prieigą prie tų pačių sisteminių modelių.

Vientisumo (integralumo) reikalavimai. Įrankis neturi priimti nežinomų failų formatų ar papildomų failų, kurie nėra reikalingi transformacijos procese.

3.6. Funkciniai reikalavimai

Sistema susideda iš dviejų posistemių. Naudotojų posistemėje naudotojams yra suteikiama valdyti asmenines naudotojų saugyklas sistemoje ir generuoti „Android“ programėlių kodą ar pačias programėles. Administracijos posistemė skirta administruoti sistemos naudotojus ir naudotojų saugyklas. Yra pateikiama panaudojimo atvejų diagrama, kurioje galima matyti išdėstytas posistemes ir panaudojimo atvejus, žiūrėti 7 paveikslėlį. Detalesnė informacija apie panaudojimo atvejus yra pateikiama 3 lentelėje ir panaudojimo atvejų specifikacija yra pateikiama 3.6.1 skyriuje.



7 pav. Panaudojimo atvejų modelis

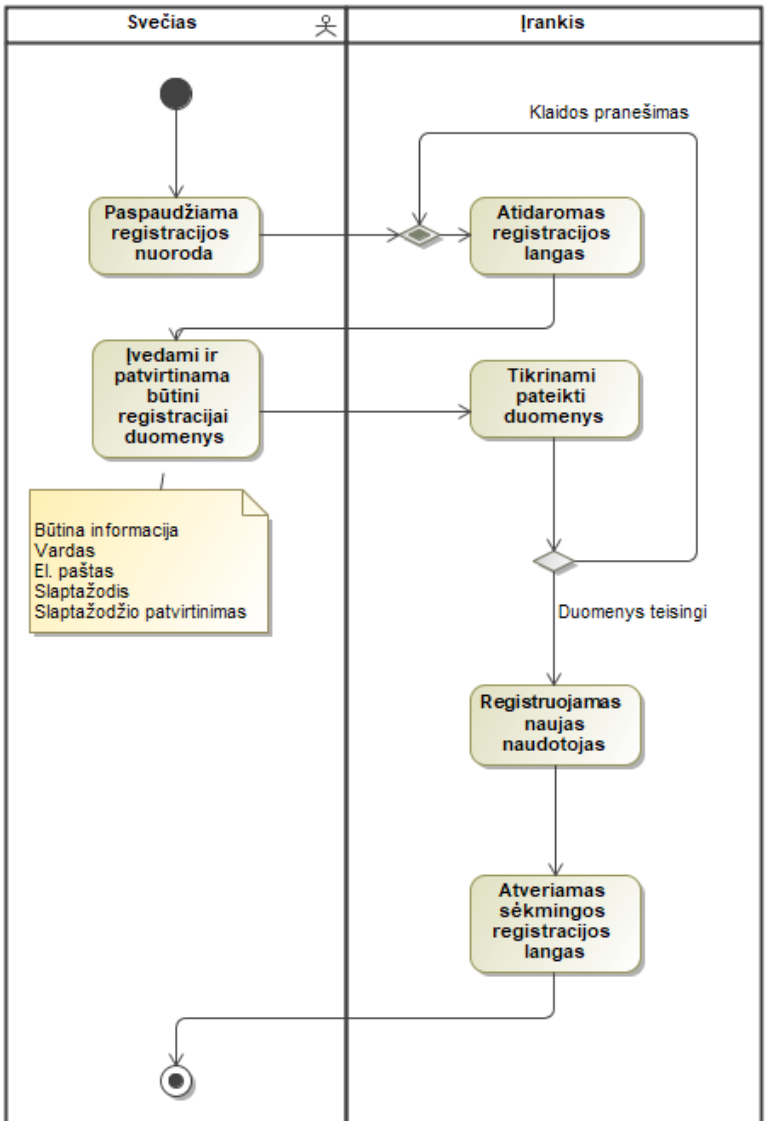
3 lentelė Panaudojimo atvejų aprašymas

Panaudojimo atvejo numeris	Panaudojimo atvejis	Aprašymas
1	Registruotis	Galimybė užregistruoti naudotoją sistemoje.
2	Prisijungti	Prisijungimas prie sistemos, tiek registruotiems naudotojams, tiek administratoriams.
3	Atsijungti	Atsijungimas nuo sistemos.

Panaudojimo atvejo numeris	Panaudojimo atvejis	Aprašymas
4	Peržiūrėti saugyklų sąrašą	Sukurtų asmeninių (prisijungusio naudotojo) saugyklų peržiūra.
5	Kurti saugyklą	Sukuriama projekto saugyklą, kurioje saugomi sugeneruoti programėlės failai.
6	Trinti saugyklą	Pašalinama saugykla, įkelti failai ir sugeneruoti įrankio failai (jei tokie yra).
7	Peržiūrėti saugyklą	Saugyklos informacijos peržiūra.
8	Peržiūrėti modelių sąrašą	Peržiūrėti įkeltų modelių sąrašą prie saugyklos.
9	Įkelti naują modelių projekto failą	Galimybė įkelti modelių failus į saugyklą.
10	Generuoti „Android“ programėlės kodą	Generuojamas „Android“ programėlės kodas.
11	Generuoti „Android“ programėlę	„Android“ programėlės generavimas iš sugeneruoto kodo.
12	Peržiūrėti naudotojų sąrašą	Galimybė peržiūrėti visų naudotojų sąrašą.
13	Užšaldyti / atstatyti naudotoją	Užšaldyti arba atstatyti naudotojo paskyrą.
14	Peržiūrėti saugyklų sąrašą	Peržiūrėti saugyklų sąrašą.
15	Užšaldyti / atstatyti saugyklą	Užšaldyti arba atstatyti saugyklą.

3.6.1 skyriuje yra pateikiami panaudojimo atvejų specifikacijos, žiūrėti 8 – 21 paveikslėlius ir 4– 17 lenteles.

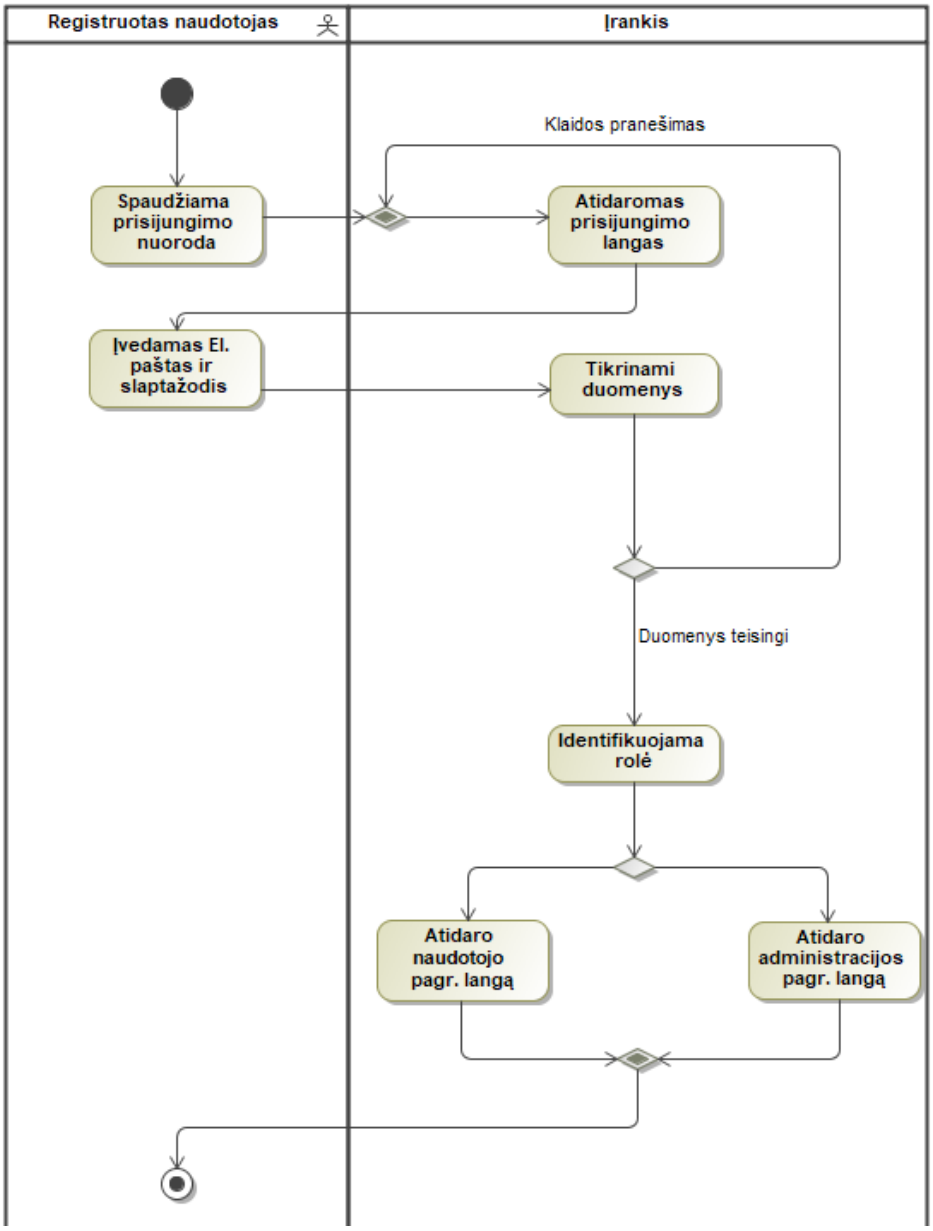
3.6.1. Panaudojimo atvejų specifikacija



8 pav. „Registruotis“ PA

4 lentelė „Registruotis“ PA specifikacija

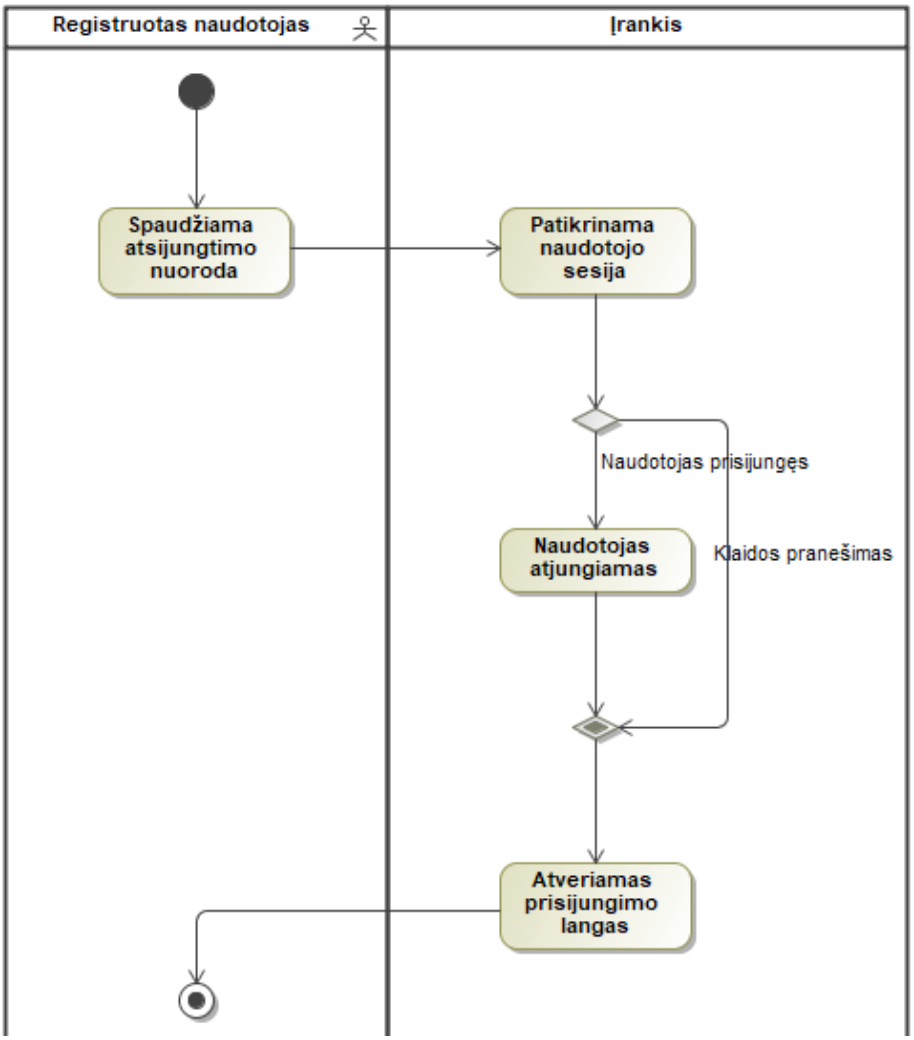
PA. PA1 Registruotis		
Tikslas. Leisti svečiui užsiregistruoti į sistemą		
Aprašymas. Svečias atsidaręs registracijos formą ją užpildo ir pateikia sistemai.		
Prieš sąlygą		Svečias neužsiregistravęs sistemoje.
Aktorius		Svečias
Sužadinimo sąlyga		Svečias paspaudžia ant registracijos nuorodos
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Svečias užsiregistravo į sistemą.



9 pav. „Prisijungti“ PA

5 lentelė „Prisijungti“ PA specifikacija

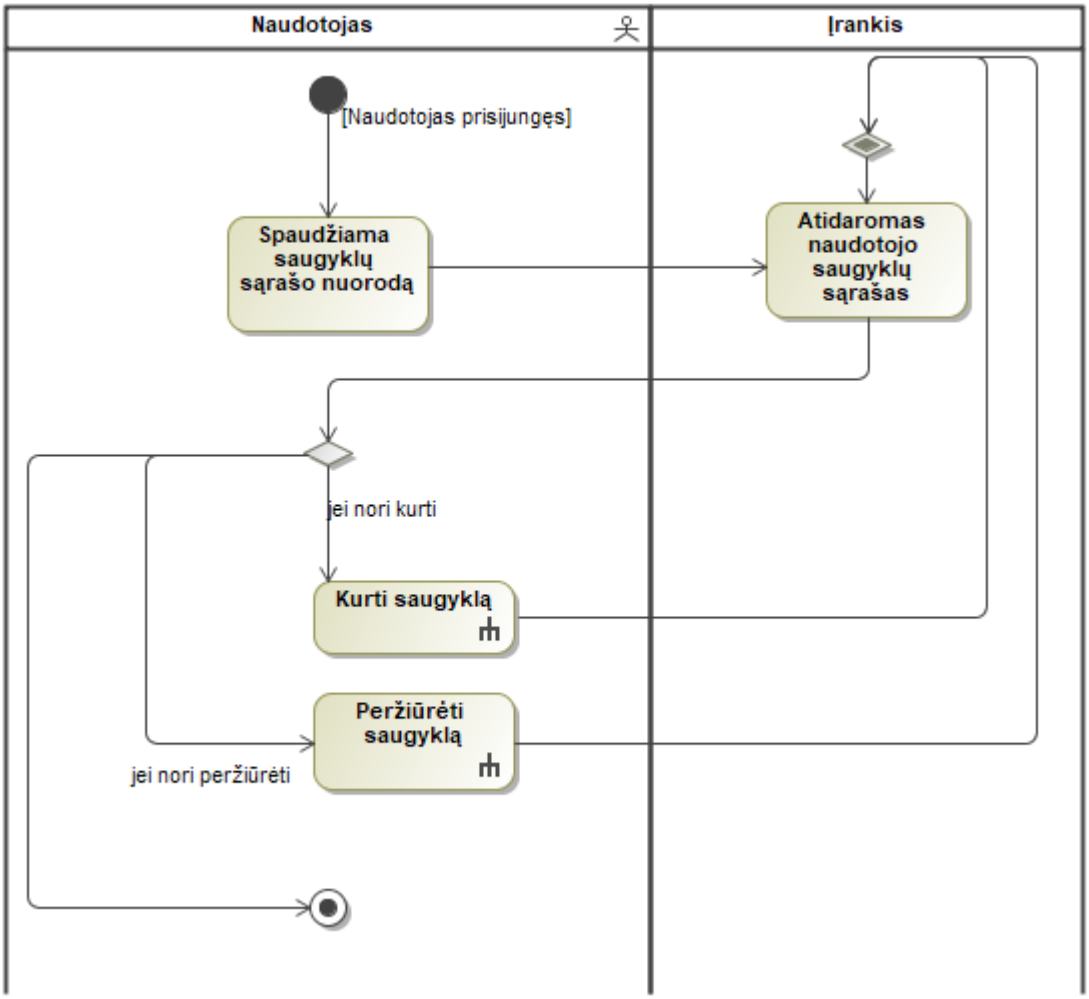
PA. PA2 Prisijungti		
Tikslas. Leisti registruotam naudotojui prisijungti prie sistemos		
Aprašymas. Registruotas naudotojas atsidaręs prisijungimo formą užpildo ją ir pateikia sistemai		
Prieš sąlygą		Naudotojas užsiregistravęs sistemoje
Aktorius		Registruotas naudotojas
Sužadinimo sąlyga		Registruotas naudotojas paspaudžia ant prisijungimo nuorodos
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Registruotas naudotojas prisijungia prie sistemos



10 pav. „Atsijungti“ PA

6 lentelė „Atsijungti“ PA specifikacija

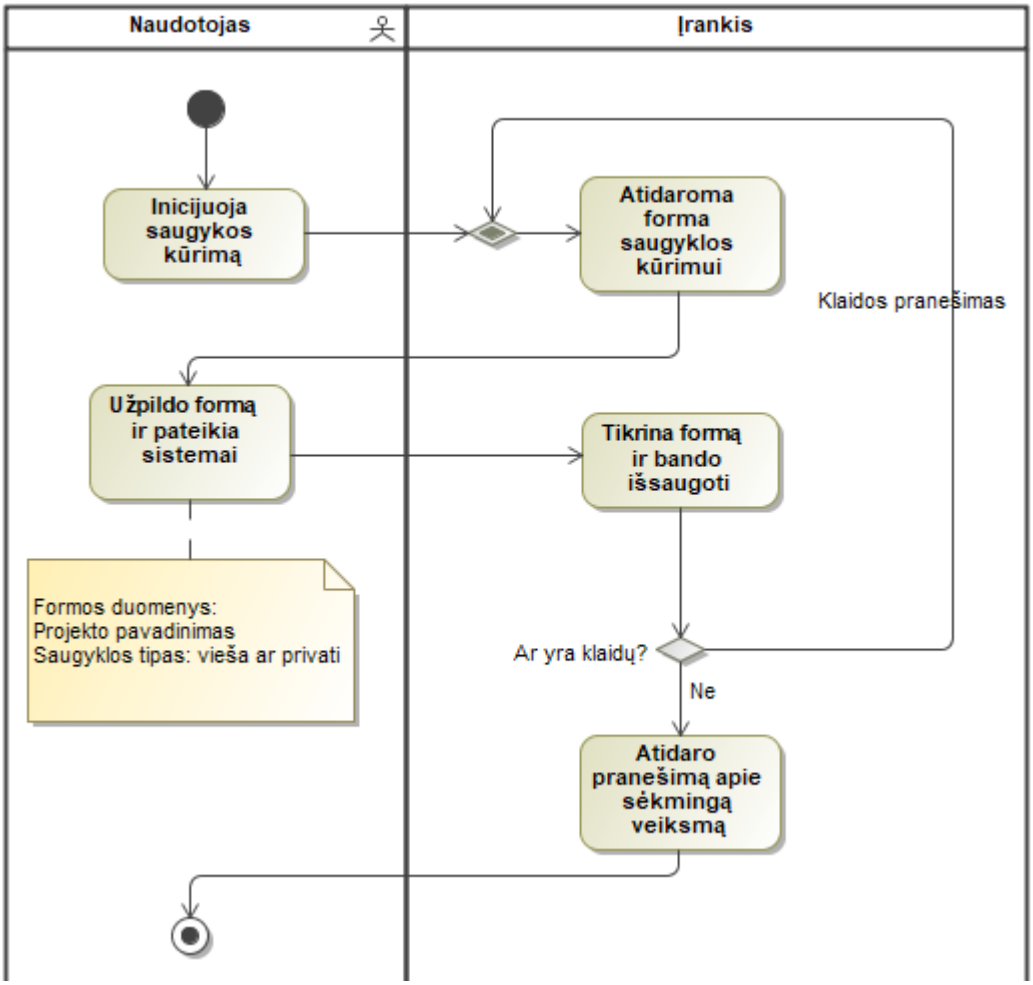
PA. PA3 Atsijungti		
Tikslas. Leisti registruotam naudotojui atsijungti nuo sistemos		
Aprašymas. Paspaudus atsijungimo nuorodą naudotojas yra atjungiamas iš sistemos		
Prieš sąlygą		Registruotas naudotojas prisijungęs prie sistemos
Aktorius		Registruotas naudotojas
Sužadinimo sąlyga		Registruotas naudotojas paspaudžia atsijungimo nuorodą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Registruotas naudotojas yra atjungtas iš sistemos



11 pav. „Peržiūrėti saugyklų sąrašą“ PA

7 lentelė „Peržiūrėti saugyklų sąrašą“ PA specifikacija

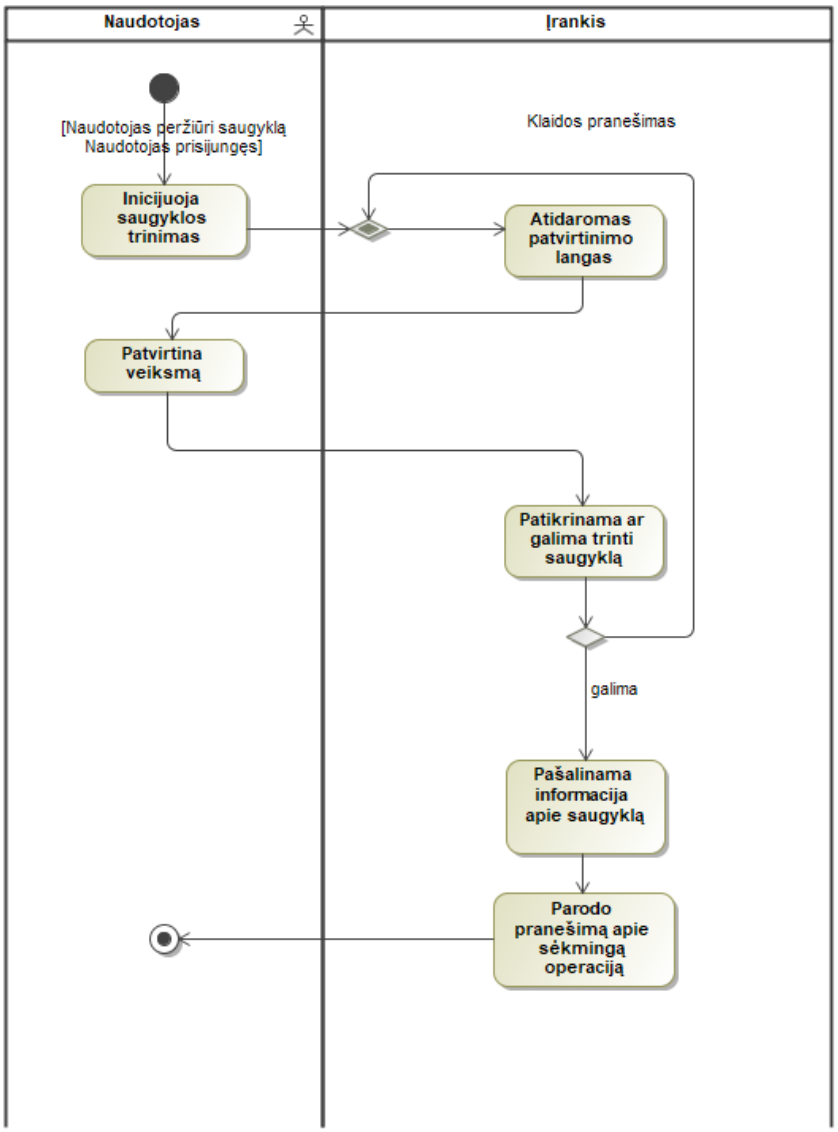
PA. PA4 Peržiūrėti saugyklų sąrašą		
Tikslas. Peržiūrėti naudotojo saugyklas ir atlikti atitinkamus veiksmus su jomis		
Aprašymas. Naudotojas paspaudęs nuorodą peržiūri saugyklų sąrašą		
Prieš sąlygą		Naudotojas prisijungęs
Aktorius		Naudotojas
Sužadinimo sąlyga		Spaudžia ant saugyklų sąrašo nuorodos
Susiję panaudojimo atvejai	**Išplečiantys PA**	Kurti saugyklą Peržiūrėti saugyklą
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Naudotojas peržiūrėjo saugyklų sąrašą



12 pav. „Kurti saugyklą“ PA

8 lentelė „Kurti saugyklą“ PA specifikacija

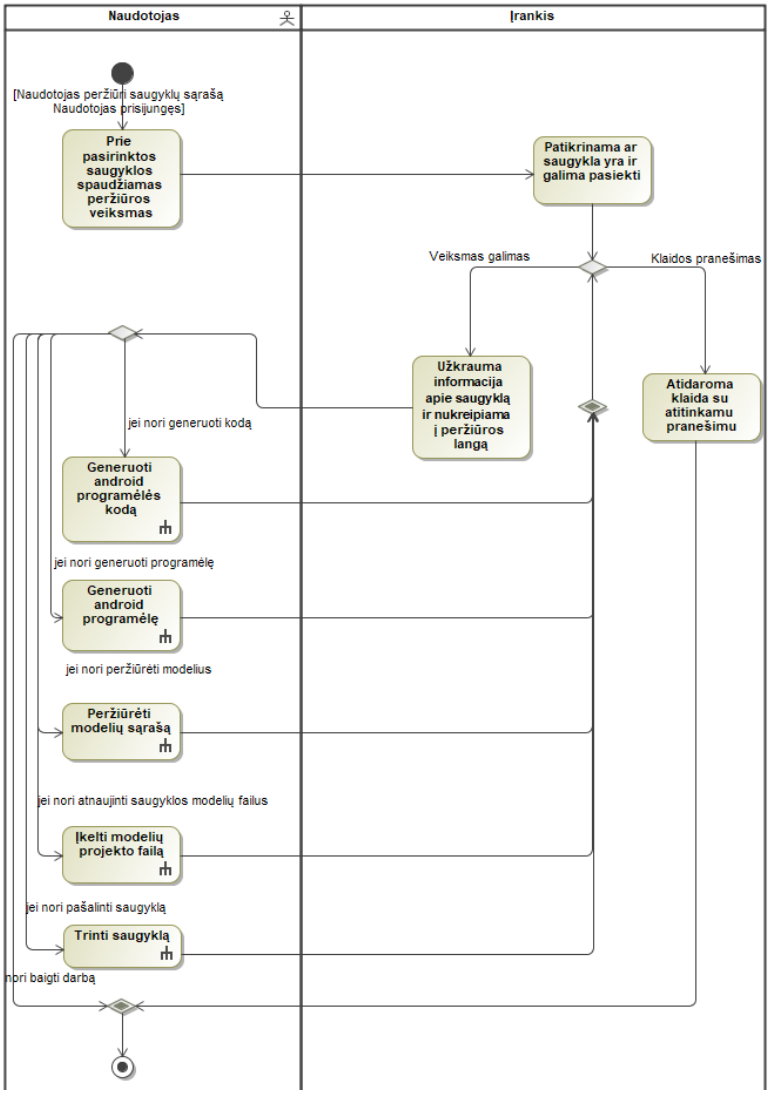
PA. PA5 Kurti saugyklą		
Tikslas. Naudotojas nori sukurti naują saugyklą		
Aprašymas. Naujos saugyklos kūrimas naujai programėlei generuoti		
Prieš sąlygą		Naudotojas prisijungęs Naudotojas peržiūri saugyklų sąrašą
Aktorius		Naudotojas
Sužadinimo sąlyga		Naudotojas (mygtuko / nuorodos) paspaudimu inicijuoja naujos saugyklos sukūrimą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Saugykla sėkmingai sukurta



13 pav. „Trinti saugyklą“ PA

9 lentelė „Trinti saugyklą“ PA specifikacija

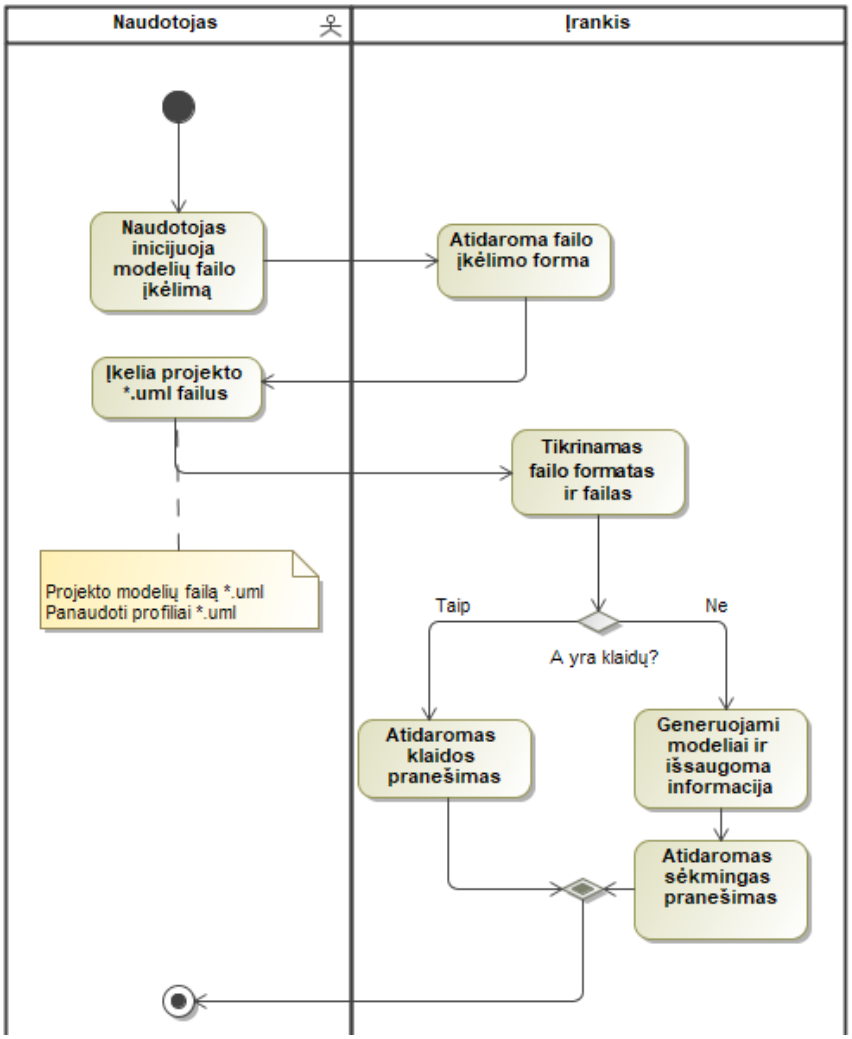
PA. PA6 Trinti saugyklą		
Tikslas. Naudotojas nebenori tęsti darbo su nurodyta saugykla ir jis nori ją ištrinti		
Aprašymas. Saugyklos pašalinimas		
Prieš sąlygą		Naudotojas prisijungęs Naudotojas peržiūri saugyklų sąrašą
Aktorius		Naudotojas
Sužadinimo sąlyga		Naudotojas (mygtuko / nuorodos) paspaudimu inicijuoja saugyklos pašalinimą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Susiję failai ir informacija su saugyklą sėkmingai pašalinti



14 pav. „Peržiūrėti saugyklą“ PA

10 lentelė „Peržiūrėti saugyklą“ PA specifikacija

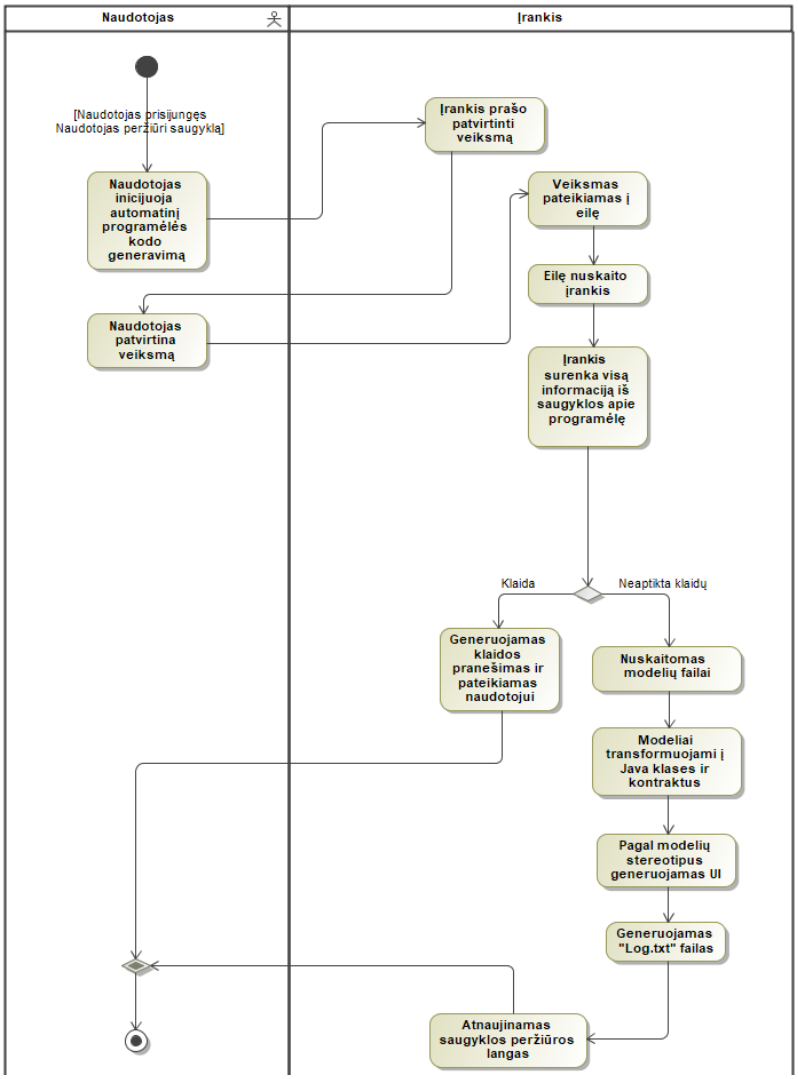
PA. PA7 Peržiūrėti saugyklą		
Tikslas. Naudotojas nori peržiūrėti saugyklą		
Aprašymas. Peržiūrėti saugyklą ir atlikti papildomus veiksmus		
Prieš sąlygą		Naudotojas prisijungęs Naudotojas peržiūri saugyklų sąrašą
Aktorius		Naudotojas
Sužadinimo sąlyga		Spaudžia saugyklos peržiūros nuorodą
Susiję panaudojimo atvejai	**Išplečiantys PA**	Generuoti „Android“ programėlės kodą Generuoti „Android“ programėlę Peržiūrėti modelių sąrašą Įkelti modelių projekto failą Trinti saugyklą
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Naudotojas peržiūrėto saugyklą



15 pav. „Įkelti naują modelių projekto failą“ PA

11 lentelė . „Įkelti naują modelių projekto failą“ PA specifikacija

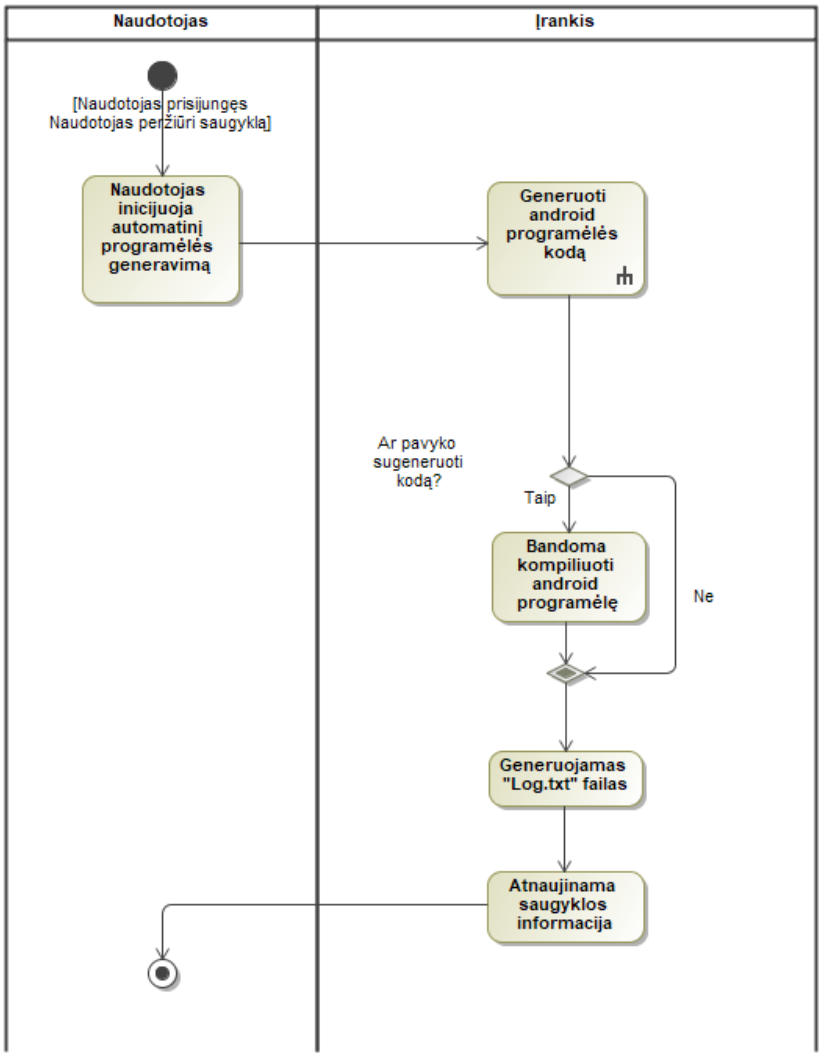
PA. PA9 Įkelti naują modelių projekto failą		
Tikslas. Norima importuoti „Android“ programėlės modelius naudojant projekto failą		
Aprašymas. Norima importuoti „Android“ programėlės modelius iš kitos programos		
Prieš sąlygą		Naudotojas prisijungęs Peržiūri modelių sąrašą
Aktorius		Naudotojas
Sužadinimo sąlyga		Naudotojas paspaudžia (mygtuką / nuorodą) inicijuodamas naujo modelio kūrimą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Failas sėkmingai įkeltas ir iš jo sukurti modeliai



16 pav. „Generuoti „Android“ programėlės kodą“ PA

12 lentelė „Generuoti „Android“ programėlės kodą“ PA specifikacija

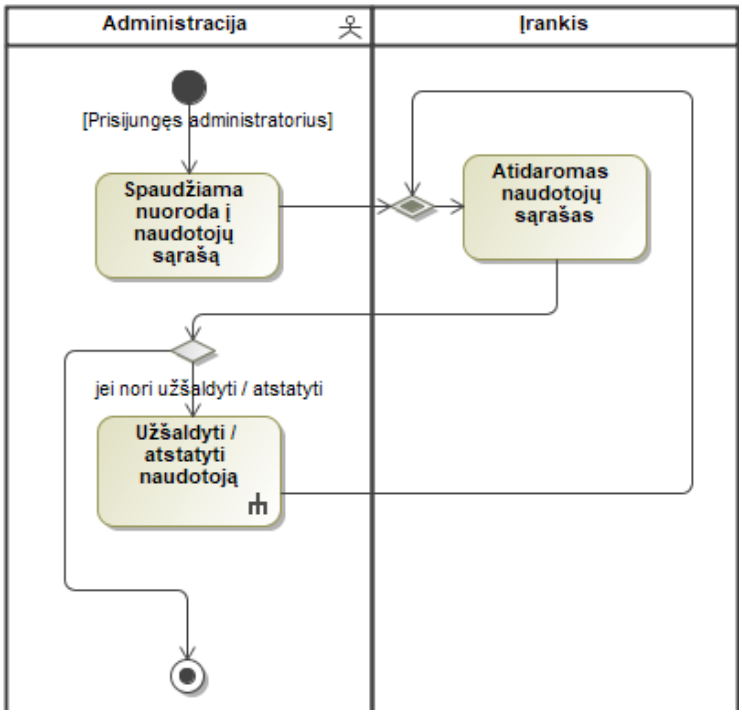
PA. PA10 Generuoti „Android“ programėlės kodą		
Tikslas. Naudotojas nori generuoti „Android“ kodą iš modelių		
Aprašymas. Automatinis kodo generavimas iš saugyklos modelių		
Prieš sąlygą		Naudotojas prisijungęs Naudotojas peržiūri saugyklą
Aktorius		Naudotojas
Sužadinimo sąlyga		Naudotojas paspaudžia (mygtuką / nuorodą) inicijuodamas modelio pašalinimą
Susiję panaudojimo atvejai	**Išplečiantys PA**	Generuoti „Android“ programėlę
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Sėkmingai sugeneruotas kodas ir pateiktas atsisiųsti



17 pav. „Generuoti „Android“ programėlę“ PA

13 lentelė „Generuoti „Android“ programėlę“ PA specifikacija

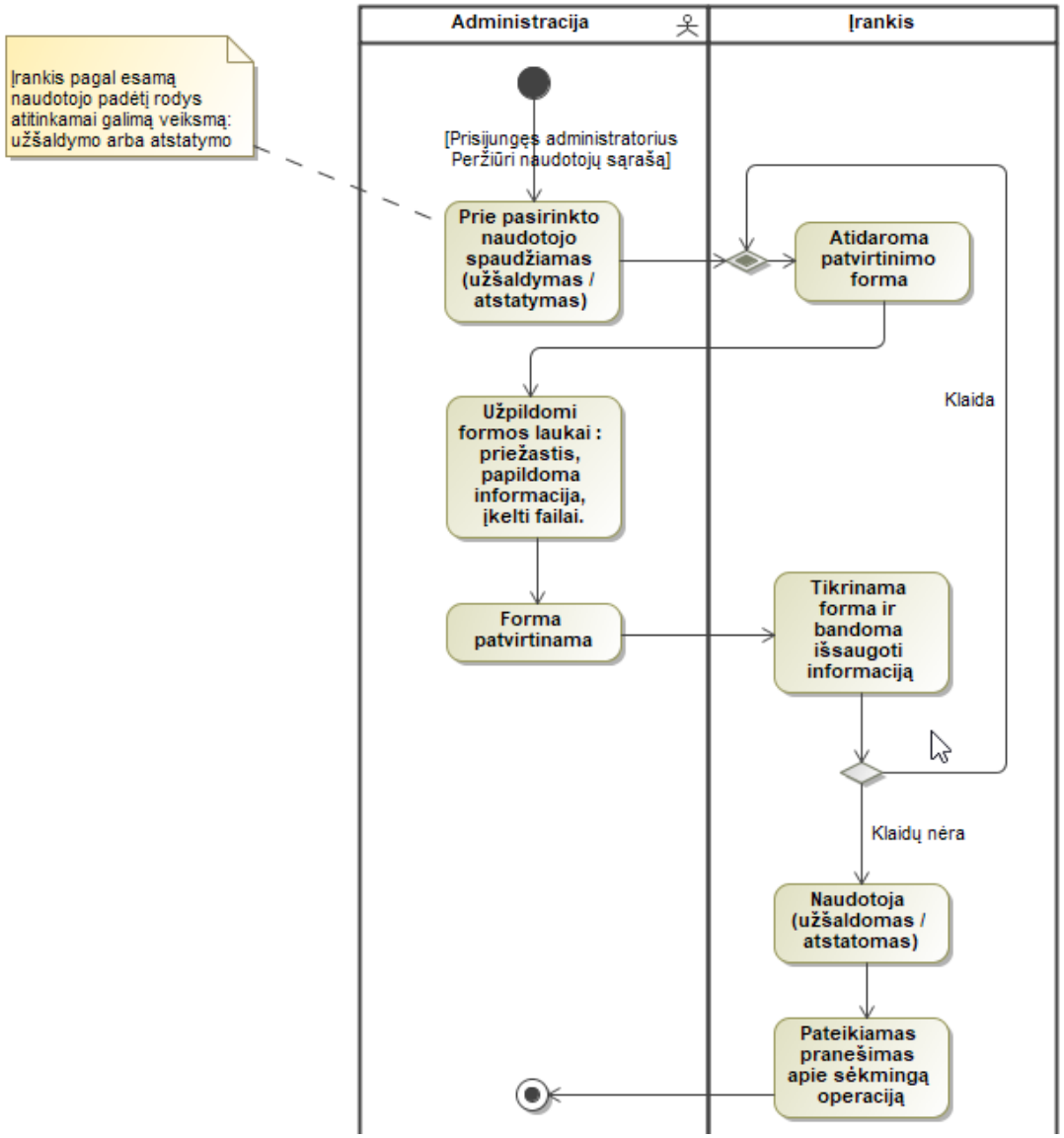
PA. PA12 Generuoti „Android“ programėlę		
Tikslas. Naudotojas nori sugeneruoti „Android“ programėlę iš sugeneruoto kodo		
Aprašymas. „Android“ programėlės generavimas iš sugeneruoto kodo		
Prieš sąlygą		Naudotojas prisijungęs Naudotojas sėkmingai sugeneravo „Android“ programėlės kodą
Aktorius		Naudotojas
Sužadinimo sąlyga		Naudotojas paspaudžia (mygtuką / nuorodą) inicijuodamas „Android“ programėlės generavimą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	Generuoti „Android“ programėlės kodą
	Specializuoja PA	
Po sąlyga:		Sėkmingai sugeneruota „Android“ programėlė ir pateikta atsisiųsti



18 pav. „Peržiūrėti naudotojų sąrašą“ PA

14 lentelė „Peržiūrėti naudotojų sąrašą“ PA specifikacija

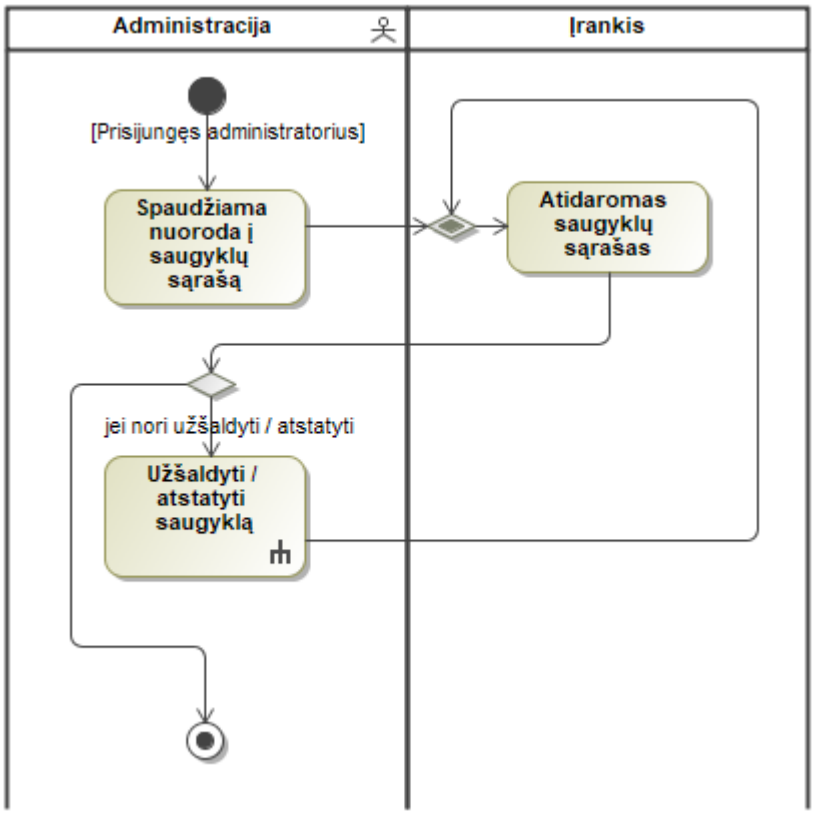
PA. PA12 Peržiūrėti naudotojų sąrašą		
Tikslas. Administratorius nori peržiūrėti naudotojų sąrašą		
Aprašymas. Naudotojų sąrašo peržiūrėjimas		
Prieš sąlygą		Administratorius prisijungęs
Aktorius		Administratorius
Sužadinimo sąlyga		Spaudžia nuorodą į naudotojų sąrašą
Susiję panaudojimo atvejai	**Išplečiantys PA**	Užšaldyti / atstatyti naudotoją
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Naudotojų sąrašas peržiūrėtas



19 pav. „Užšaldyti/atstatyti naudotoją“ PA

15 lentelė „Užšaldyti/atstatyti naudotoją“ PA specifikacija

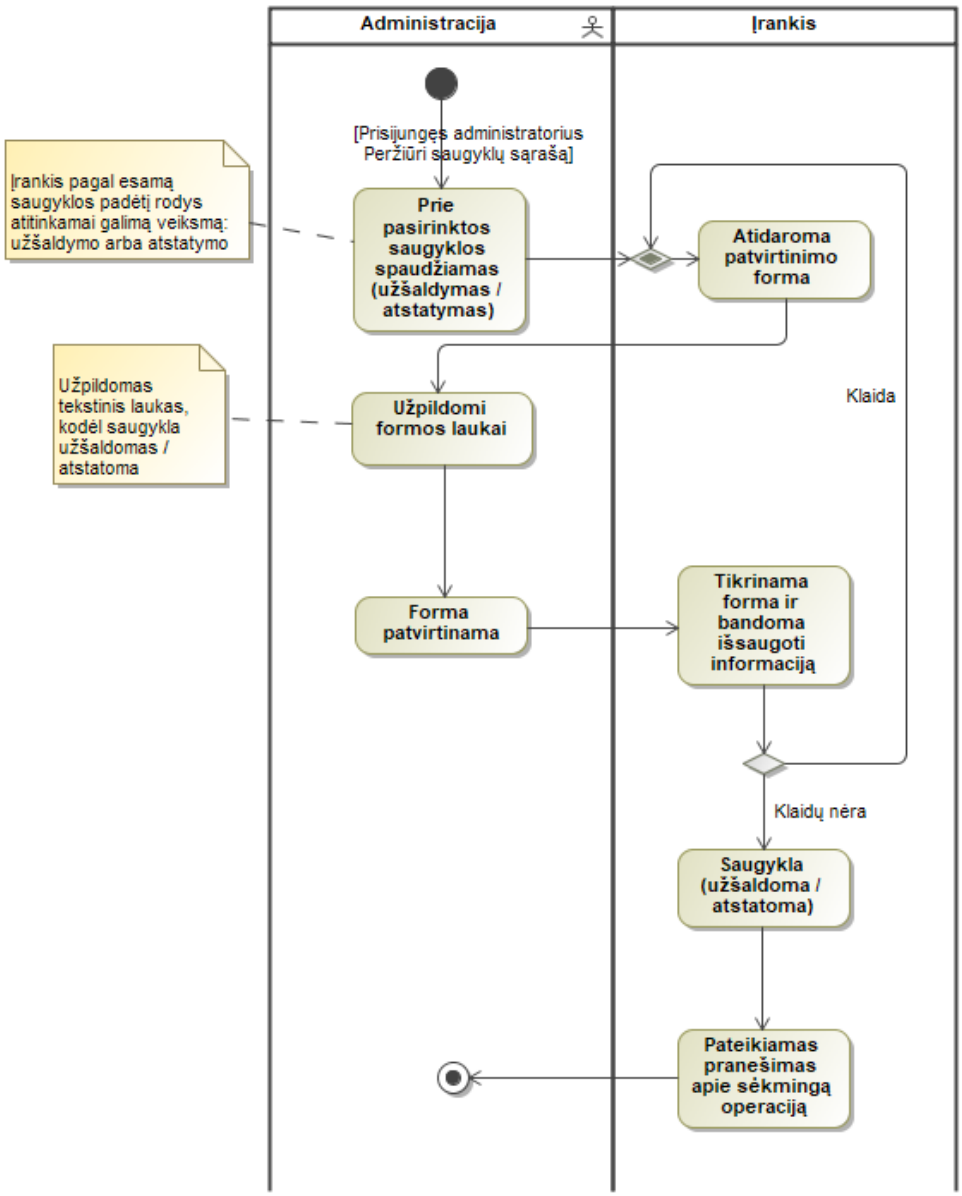
PA. PA13 Užšaldyti/atstatyti naudotoją		
Tikslas. Administratorius nori užšaldyti arba atstatyti naudotojo paskyrą		
Aprašymas. Naudotojo paskyros užšaldymas arba atstatymas		
Prieš sąlygą		Administratorius prisijungęs Administratorius peržiūri naudotojų sąrašą
Aktorius		Administratorius
Sužadinimo sąlyga		Administratorius paspaudžia (mygtuką / nuorodą) inicijuodamas (užšaldymo / atstatymo) veiksmą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Sėkmingai atstatyta arba užšaldyta naudotojo paskyra



20 pav. „Peržiūrėti saugyklų sąrašą“ PA

16 lentelė „Peržiūrėti saugyklų sąrašą“ PA specifikacija

PA. PA14 Peržiūrėti saugyklų sąrašą		
Tikslas. Administratorius nori peržiūrėti saugyklų sąrašą		
Aprašymas. Naudotojų saugyklų sąrašo peržiūrėjimas		
Prieš sąlygą		Administratorius prisijungęs
Aktorius		Administratorius
Sužadinimo sąlyga		Spaudžia nuorodą į saugyklų sąrašą
Susiję panaudojimo atvejai	**Išplečiantys PA**	Užšaldyti / atstatyti saugyklą
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Peržiūrėtas saugyklų sąrašas



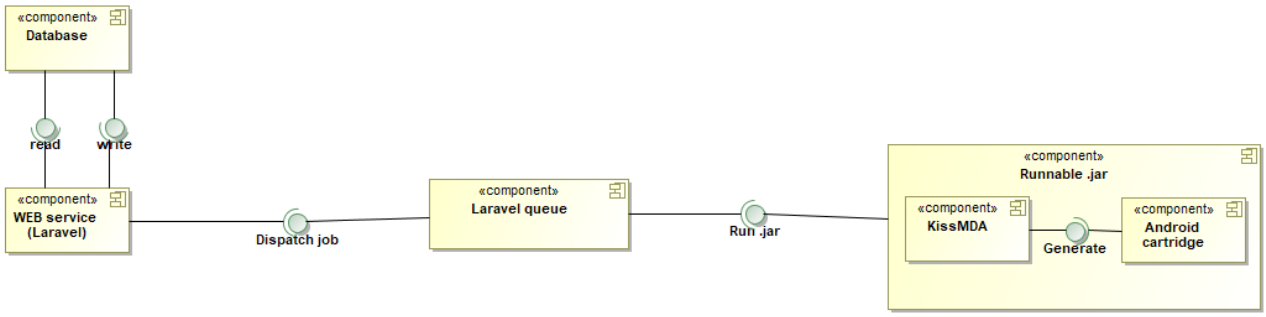
21 pav. „Užšaldyti / atstatyti saugyklą“ PA

17 lentelė „Užšaldyti / atstatyti saugyklą“ PA specifikacija

PA. PA15 Užšaldyti / atstatyti saugyklą		
Tikslas. Administratorius nori užšaldyti arba atstatyti saugyklą		
Aprašymas. Naudotojo saugyklos užšaldymas arba atstatymas		
Prieš sąlygą		Administratorius prisijungęs Administratorius peržiūri saugyklų sąrašą
Aktorius		Administratorius
Sužadinimo sąlyga		Administratorius paspaudžia (mygtuką / nuorodą) inicijuodamas (užšaldymo / atstatymo) veiksmą
Susiję panaudojimo atvejai	**Išplečiantys PA**	
	Apimami PA	
	Specializuoja PA	
Po sąlyga:		Sėkmingai atstatyta arba užšaldyta naudotojo saugykla

3.7. Sistemos statinis vaizdas

Įrankio architektūra yra pagrįsta komponentų architektūros metodika. Komponentais pagrįsta architektūra suteikia įrankiui lankstumą, leidžianti pakeisti norimus komponentus kitais ar prijungti papildomus, leidžia kai kuriuos komponentus naudoti atskirai, pateikia komponentų diagramą 22 paveikslėlyje.



22 pav. Komponentų diagrama

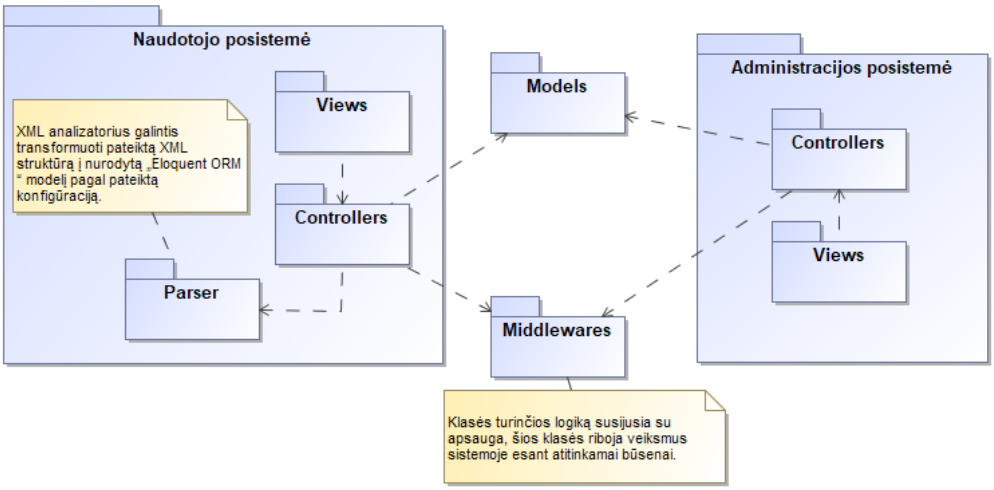
Įrankis yra sudarytas iš kelių komponentų. Žiūrėti 22 pav. Šie komponentai yra:

- WEB paslauga (angl. service) – realizuota naudojant „Laravel“ karkasą.
- Eilė (*angl. queue*) – kodo generavimo procesas ir programėlės generavimo procesas yra išskaldyti į atskirus procesus, kurie yra statomi į eilę ir vykdomi, norint suteikti naudotojui grįžtamąjį ryšį, kad procesas yra vykdomas. Įvykus bet kuriam iš išvardintų procesų yra pateikiamas rezultatas ir įvykių žurnalas (angl. *log*).
- Vykdomasis failas „*.jar*“ failas – komponentas atsakingas už kodo generavimą ir papildomų resursų generavimą, reikalingų norint sukompiliuoti „Android“ programėlę.

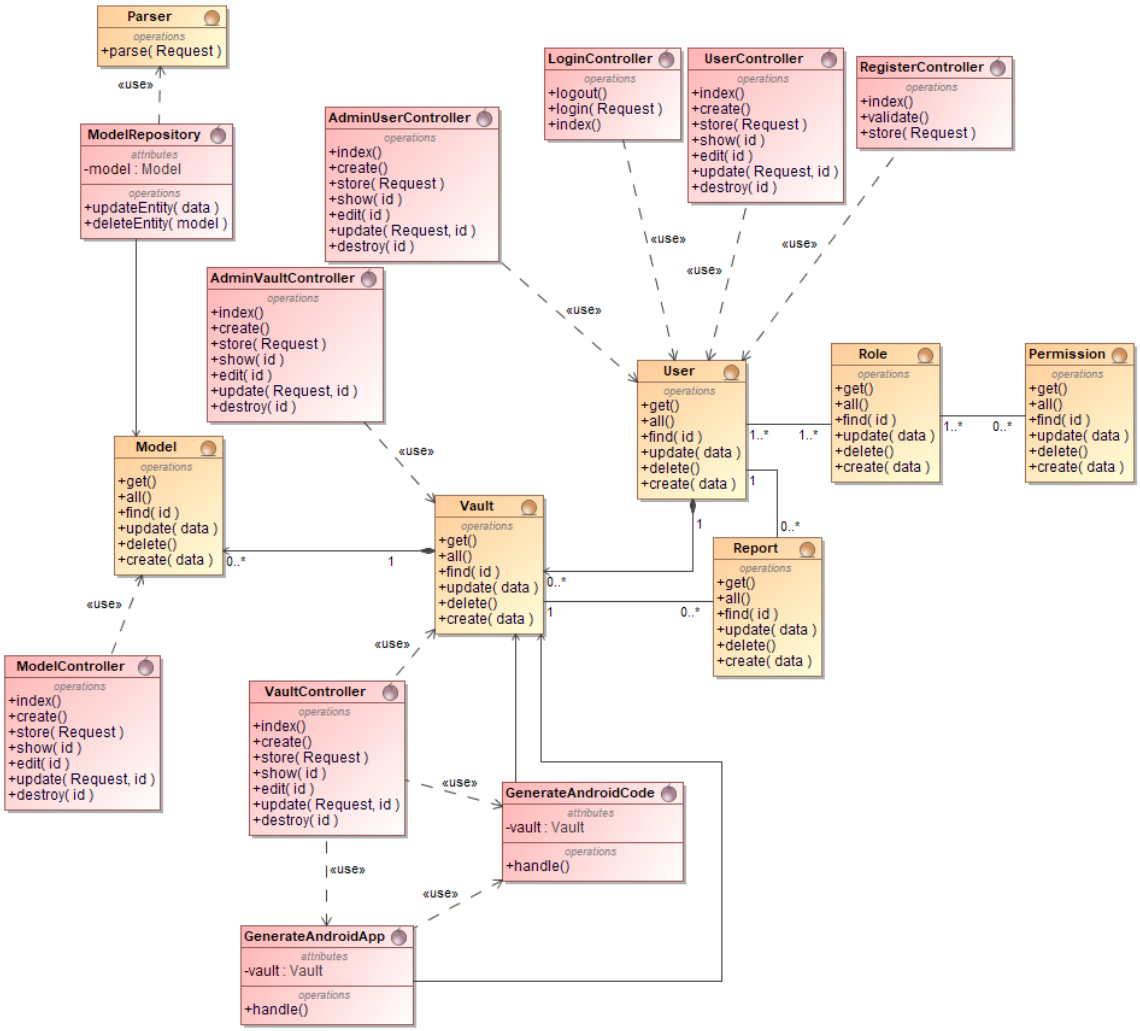
3.8. Pagrindinių komponentų detalizavimas

3.8.1. WEB paslaugos komponentas

WEB paslauga yra išskirta į dvi posistemes, tai į naudotojo posistemę ir į administracijos posistemę. Žiūrėti 23 paveikslėlį. WEB paslaugos architektūrai buvo pasirinkta naudoti MVC (angl. *model-view-controller*) architektūra. Posistemės dalijasi modeliais, taip išvengiant kodo pasikartojimo.



23 pav. WEB paslaugos paketų diagrama



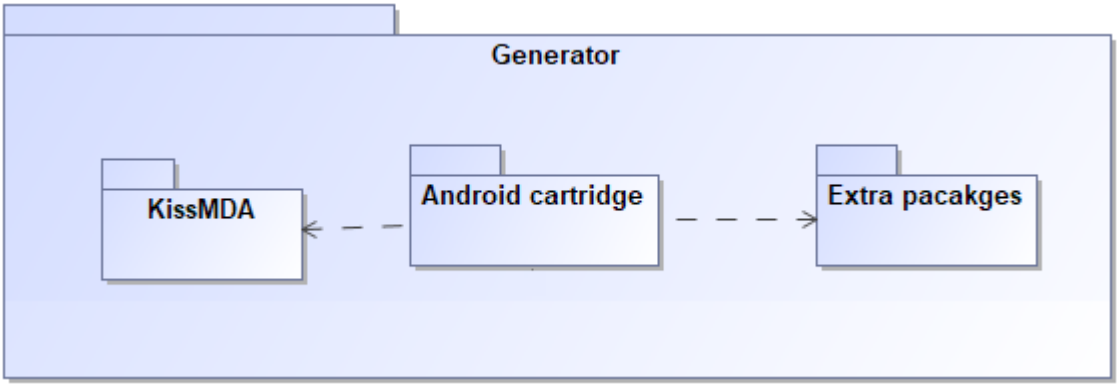
24 pav. WEB paslaugos komponento klasių diagrama

Pateikiama klasių diagrama tarp modelių ir valdiklių (angl. *controllers*). Naudotojo sąsaja yra realizuota naudojant „Laravel“ integruotą „Blade“ šablonų variklį, kuris palengvina naudotojų sąsajos aprašymą ir pateikimo procesą. Bendradarbiavimo diagramose nurodoma kurie naudotojo

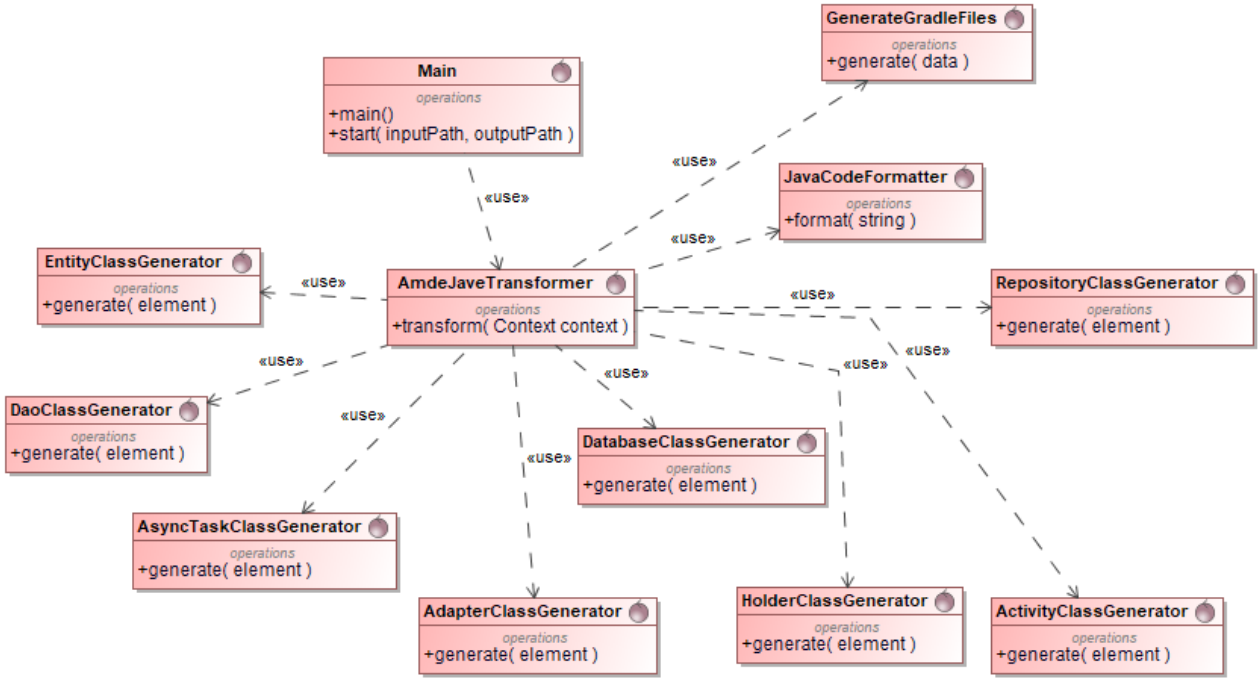
sąsajos langai bus rodomi naudotojui, vaizdus realizuojant per aukščiausią generavimo funkciją „*render()*“.

3.8.2. Vykdomasis failas „.jar“ generatoriaus komponentas

Kodo generavimui buvo pasinaudota „KissMDA“ karkasu. Buvo aprašytos „Android“ programėlių transformacijos taisyklės ir procesas, pateikiama generatoriaus paketų diagrama 25 paveikslėlyje. Taip pat pateikiama „Android cartrige“ paketo klasių diagrama 26 paveikslėlyje.



25 pav. Generatorių komponento paketų diagrama



26 pav. „Android“ kodo generatoriaus klasių diagrama

3.9. Kodo generavimo procesas

Šiame skyriuje yra detalizuojami pagrindiniai aspektai, kurie yra reikalingi kodo generavimo procese. Pirmiausia yra pristatomi sukurti modeliavimo profiliai, kurie yra naudojami atpažinti modeliuose pateiktą informaciją. Vėliau yra trumpai detalizuojama generavimo proceso vykdymo eiga.

3.9.1. Sukurti „MagicDraw“ profiliai

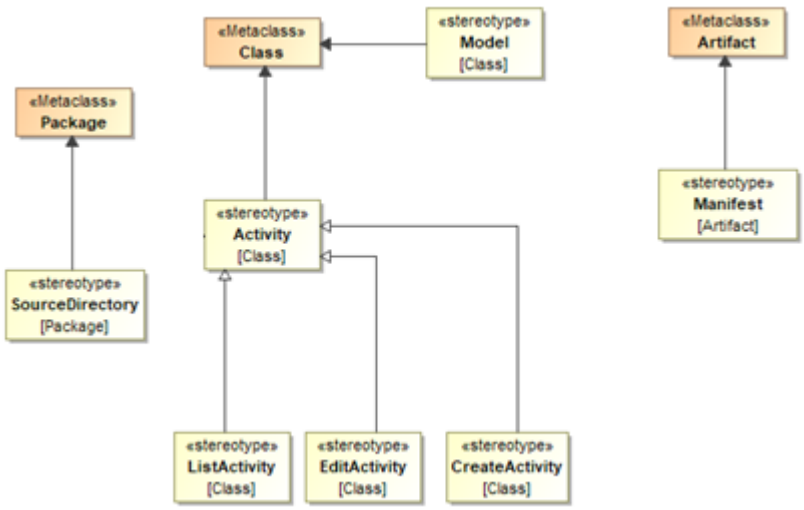
Norint, kad įrankis teisingai suprastų pateiktą klasių diagramą ir sugebėtų sugeneruoti „Android“ programėlės kodą ar pačią programėlę, yra būtina naudoti pateiktą „MagicDraw“ profilį. Šis profilis nustato papildomus stereotipus kuriuos vėliau galima suteikti klasėms, klasių diagramoje.

18 lentelėje pateikia informacija apie profilį ir trumpai aprašoma kokius failus generatorius generuos pagal suteiktą stereotipą. Profilio diagrama pateikta 27 paveikslėlyje. Kad supaprastinti ir apsibrėžti kodo generavimo ribas kaip kodas turi būti generuojamas buvo atsižvelgta į „Google“ teikiamas „Android“ programėlių architektūros rekomendacijas ir buvo pasirinkta kodą generuoti pagal MVVM (angl. *Model–view–viewmodel*) architektūrą.

18 lentelė Stereotipų aprašymai

Meta klasė	Stereotipas	Aprašymas	Generuojama
„Package“	„SourceDirectory“	Stereotipas suteikiamas „package“ elementui norint nusakyti įrankiui nuo kur turi būti skaitomi moduliai.	Generuojami visi katalogų lygiai.
„Artifact“	„Manifest“	Stereotipas nurodomas klasei, kuri atstovauja „Android manifest.xml“ failą.	Generuojamas „manifest.xml“ failas
„Class“	„Model“	Stereotipas suteikiamas klasei, kuri atstovauja „Android“ modulį.	Automatiškai generuojamos atitinkamos klasės pagal MVVM architektūrą. Pagr. klasė aprašanti duomenų bazės lentelę. „ListAdapter“ ir „ViewHolder“. Saugyklos klasė su visais CRUD (*angl. create, read, delete, update*) metodais. DAO (*angl. data-access-object*).
	„Activity“	Stereotipas suteikiamas klasei, kuri atstovauja „Android activity“ klasę.	Generuojamos visos galimos „Android activity“ nurodytai klasei. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„ListActivity“	Stereotipas suteikiamas klasei, kuri atstovauja sąrašo „Android activity“ klasę.	Generuojama „Adnrdoid activity“, kurioje bus rodomas tik modelių sąrašas su nurodytais parametrais. Klasė turi turėti sąryšį su klase, kuri

Meta klasė	Stereotipas	Aprašymas	Generuojama
			turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„CreateActivity“	Stereotipas suteikiamas klasei, kuri atstovauja kūrimo formos „Android activity“ klasę.	Generuojama „Adnrdoid activity“, kurioje bus rodoma tik kūrimo forma į kurią reikia pateikti atitinkamą informaciją. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„EditActivity“	Stereotipas suteikiamas klasei, kuri atstovauja keitimo formos „Android activity“ klasę.	Generuojama „Adnrdoid activity“, kurioje bus rodoma tik keitimo forma į kurią galima pateikti tik atitinkamą informaciją. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.



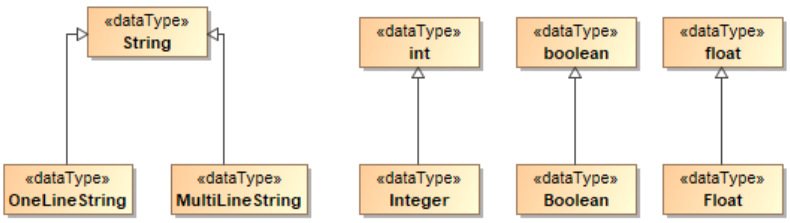
27 pav. Įrankio „MagicDraw“ stereotipų profilis

Tačiau, kad generatorius suprastų objektų diagramoje nurodytų klasių kintamuosius, buvo sukurtas papildomas „MagicDraw“ profilis, kuriame nurodyti visi galimi kintamųjų tipai kuriuose galima

generuoti, 19 lentelėje pateikiama detalesnė informacija apie generuojamus kintamųjų laukų tipus ir 28 paveikslėlyje pateikiama profilio diagrama.

19 lentelė Duomenų tipų „MagicDraw“ profilis

Duomenų tipas	Aprašymas	Generuojama
„String“	Galima nurodyti klasėms turinčioms „Entity“, „ListActivity“, „CreateActivity“, „EditActivity“ stereotipus.	Generuojamas „String“ laukas, naudotojo sąsajoje generuojamas „EditText“ laukas, pagal nutylėjimą kuris yra nustatomas, kad bus vienos eilutės.
„OneLineString“	Galima nurodyti tik klasėms turinčioms „ListActivity“, „CreateActivity“ ir „EditActivity“ stereotipus.	Generuojamas „EditText“ laukas su nustatytus, kad bus galima rašyti tik vieną eilutę.
„MultiLineString“	Galima nurodyti tik klasėms turinčioms „ListActivity“, „CreateActivity“ ir „EditActivity“ stereotipus.	Generuojamas „EditText“ laukas su nustatytus, kad bus galima rašyti tekstą į kelias eilutes naudojant „enter“, tai pat tekstas bus atvaizduojamas kaip paragrafas.
„int“	Galima nurodyti klasėms turinčioms „Entity“ stereotipą.	Generuojamas „int“ laukas „Entity“ klasėje.
„Integer“	Galima nurodyti klasėms turinčioms „Entity“ stereotipą.	Generuojamas „Integer“ laukas „Entity“ klasėje.
„boolean“	Galima nurodyti klasėms turinčioms „Entity“ stereotipą.	Generuojamas „boolean“ laukas „Entity“ klasėje.
„Boolean“	Galima nurodyti klasėms turinčioms „Entity“, „ListActivity“, „CreateActivity“, „EditActivity“ stereotipus.	Generuojamas „Boolean“ laukas „Entity“ klasėje, tai pat naudotojo sąsajoje generuojamas „CheckBox“ laukelis.
„float“	Galima nurodyti klasėms turinčioms „Entity“ stereotipą.	Generuojamas „float“ laukas „Entity“ klasėje.
„Float“	Galima nurodyti klasėms turinčioms „Entity“ stereotipą.	Generuojamas „Float“ laukas „Entity“ klasėje.



28 pav. „MagicDraw“ kintamųjų tipų profilis

3.9.2. Kodo generavimo eiga

Kodo generavimo procesas buvo išskaldytas į dvi dalis. Pirmoje dalyje kodas yra generuojamas pasitelkiant AST (angl. abstract syntax tree) metodiką. Pagal pateiktą klasių diagramą įrankis tikrina modelį sukurtų objektų nustatytus stereotipus, duomenų tipus, kintamuosius. Visą informaciją fiksuoja ir fiksavimo metu vykdo atitinkamų klasių generavimą. Pirmiausiai yra sugeneruojamos Java „Entity“ klasės, vėliau šių klasių DAO (angl. data access object) sąsajos (angl. interfaces). Surinkta informacija iš modelio yra pateikiama į kitą etapą, kuriame yra naudojami kodų šablonai norint sugeneruoti pagalbines Java klases, kurios yra reikalingos norint pritaikyti MVVM architektūroje, tai būtų „EntityRepository“, „EntityViewModel“, „EntityAdapter“.

Sugeneravus duomenų lygį ir pagrindines „Java“ klases yra generuojama naudotojo sąsaja. Naudotojo sąsajos funkcionalumui, tai „Activity“ klasėms sugeneruoti buvo naudojami kodo šablonai su iš anksto numatytu funkcionalumu. O naudotojo sąsajos elementai yra generuojami XML formatu, kurie jau prieš tai turi būti užfiksuoti po „Entity“ klasių sukūrimo.

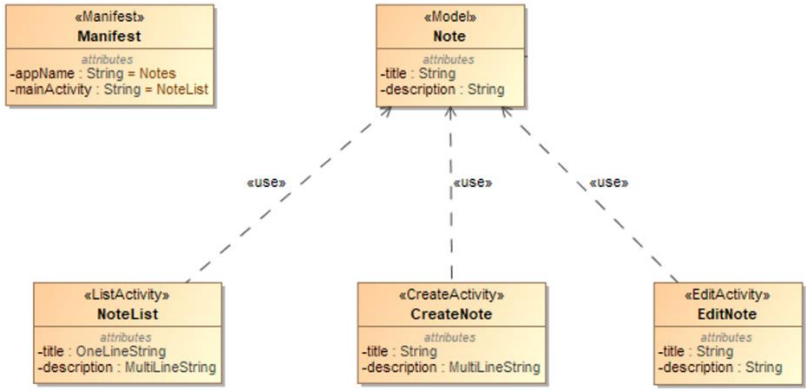
Sugeneravus visas „Android“ programėles klases ir naudoto sąsają projekte yra sugeneruojamas „AndroidManifest.xml“, kuriame aprašoma pagrindinė informacija apie programėlę. Sugeneravus „Manifest“ failą yra perkeliami iš anksto numatyti naudotojo sąsajos pagalbiniai failai, standartizuoti paveikslėliai.

4. Sukurtos programinės sistemos analizavimas

Šiame skyriuje yra aprašomas sukurtos sistemos analizavimas ir bandymai, gauti rezultatai ir galimi tobulinimai, praplėtimai.

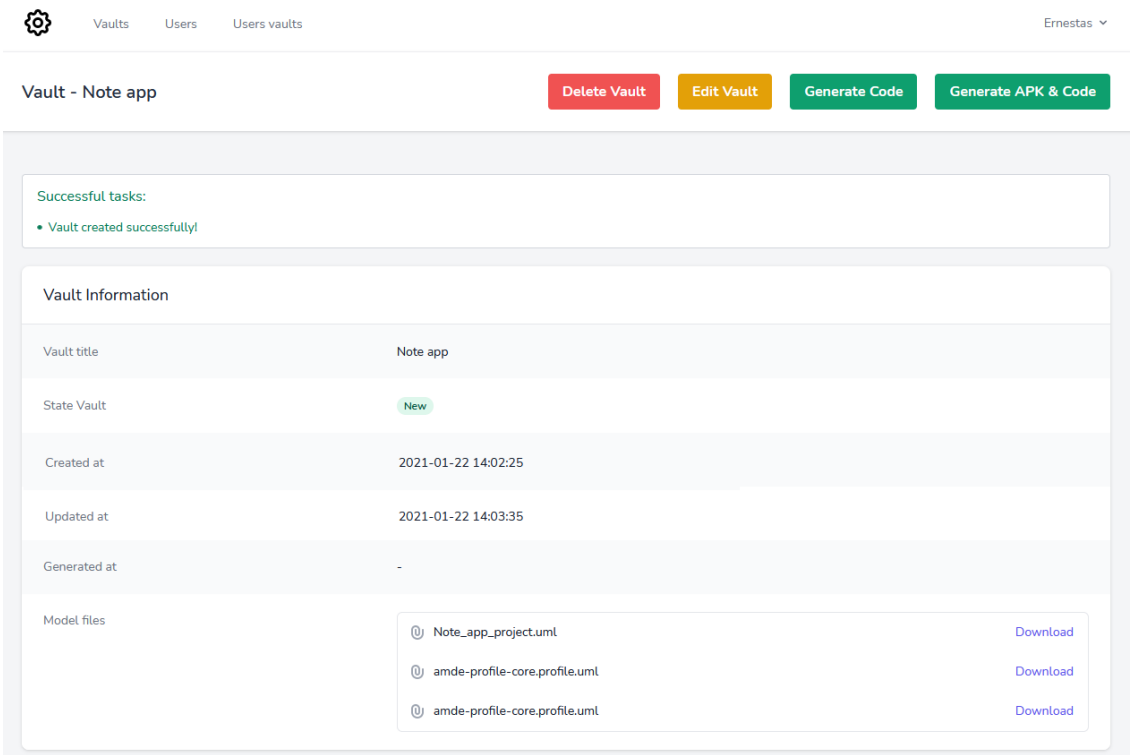
4.1. Programinės sistemos bandymas

Įrankiu buvo bandoma sugeneruoti „Užrašinės“ programėlę, pateikiama 29 paveikslėlyje sumodeliuota klasių diagrama pagal įrankio naudojimo nurodymus.



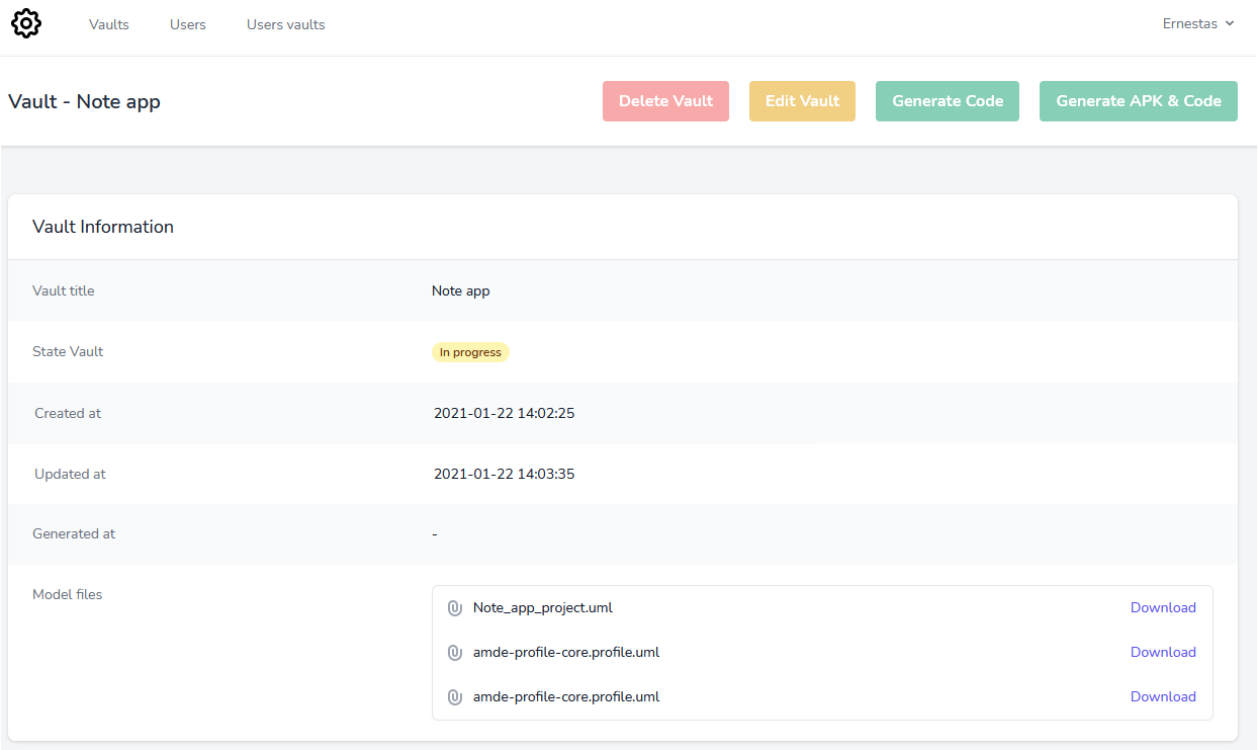
29 pav. Programėlės „Užrašinė“ klasių diagrama

Sukūrus klasių diagramą, diagrama buvo išeksportuota į „Eclipse UML 2 (v3.x) XML“ formatą, norint ją pateikti įrankiui. Projekto išeksportuoti XML failai buvo įkelti į sukurtą saugyklą, pateikiamas saugyklos langas 30 paveikslėlyje.



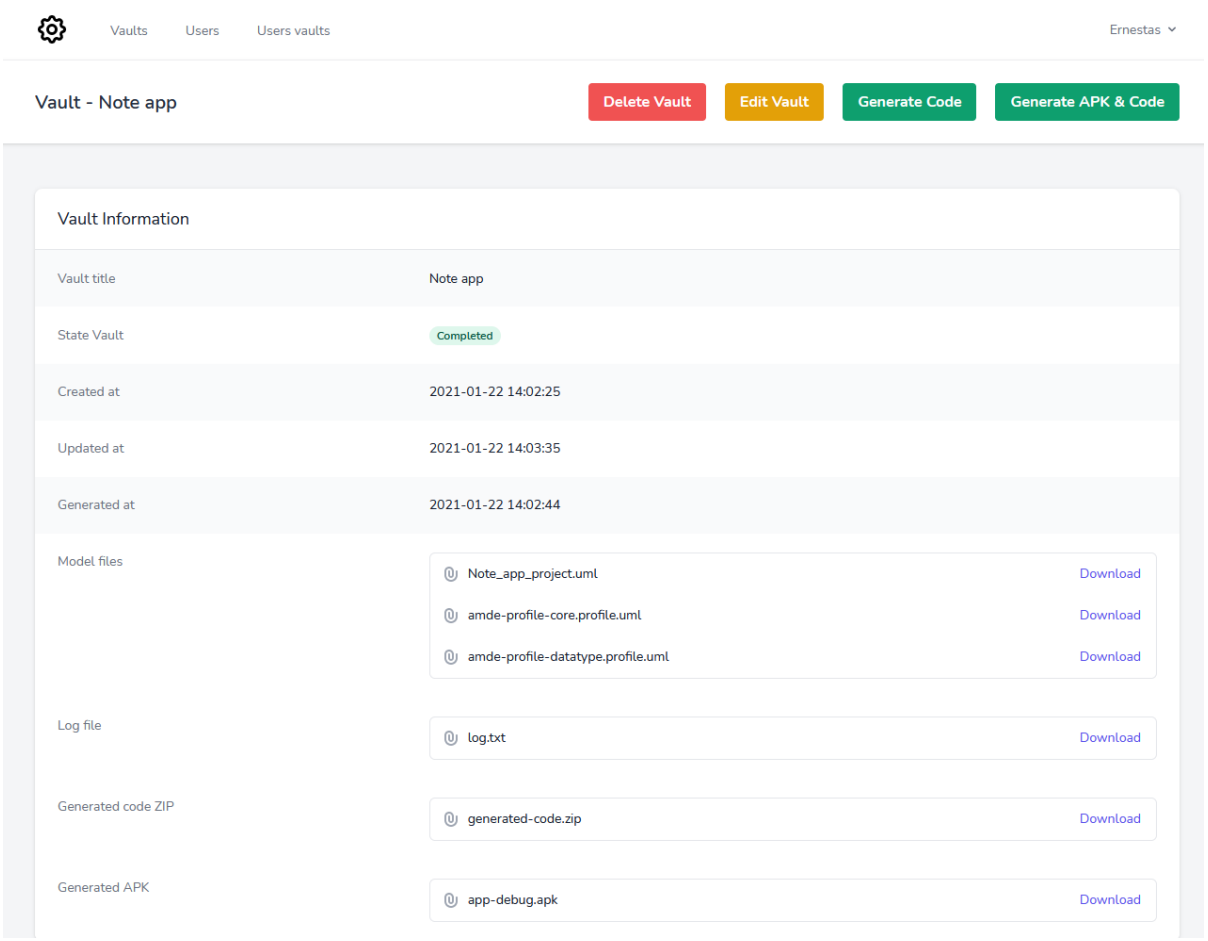
30 pav. Saugyklos peržiūros langas

Vėliau buvo pradėtas programėlės ir kodo generavimo procesas. Pradedant programėlės ir kodo generavimo procesą, saugykla yra trumpam užšaldoma norint užbaigti generavimo procesą, žiūrėti 31 paveikslėlį.



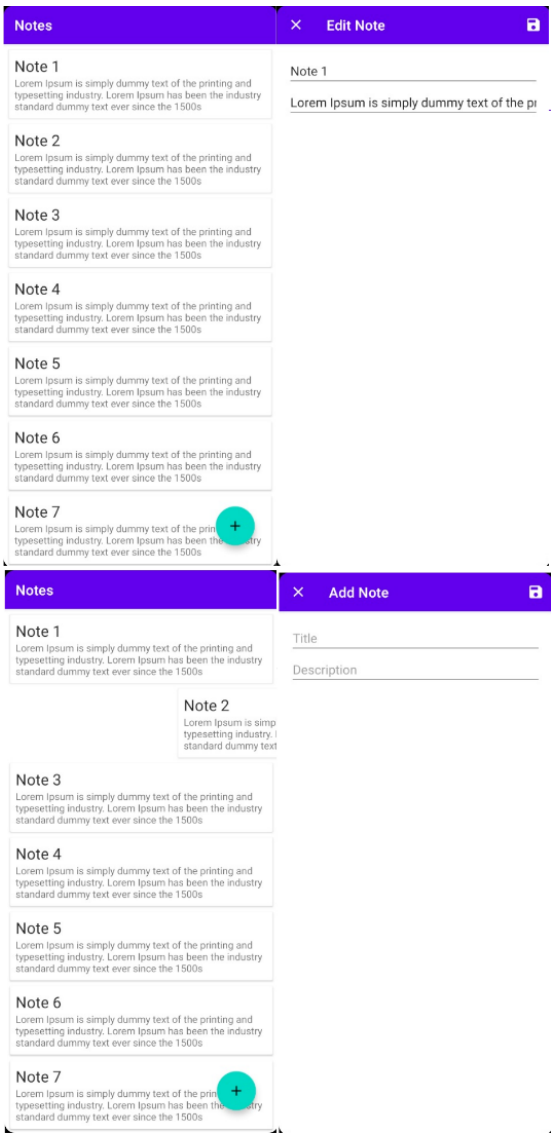
31 pav. Saugyklos peržiūros langas pradėjus programėlės generavimo procesą

Generavimo procesui pasibaigus pateikiamas rezultatų langas, kuriame yra pateikiama informaciją apie sugeneruotą programėlę, programėlės kodas ir įvykių žurnalas, žiūrėti 32 paveikslėlį.



32 pav. Saugyklos peržiūros langas sugeneravus programėlę ir programėlės kodą

Gauti rezultatai buvo atsisiųsti ir detaliau išanalizuoti, kas buvo sugeneruota. Pateikiami sugeneruotos programėlės vaizdai 33 paveikslėlyje ir 20 lentelėje detalesnė informacija kas buvo sugeneruota.



33 pav. Sugeneruotos „Užrašinės“ programėlės vaizdai

20 lentelė Sugeneruotos „Užrašų“ programėlės rezultatai

Java failų skaičius	15
XML failų skaičius	26
Java kodo eil. skaičius	756
XML eil. skaičius	298

4.2. Sistemos vertinimo rezultatai

Šiuo metu įrankis geba generuoti „Android“ programėlės kodą pagal pateiktą klasių diagramą, tam atlikti buvo naudojama AST metodika ir papildomi šabloninio kodo failai. Tačiau įrankyje nėra pilnai realizuotas generavimo procesas, įrankis nepalaiko:

- UML asociacijų tarp modelių (vienas su vienu, vienas su daug, daug su daug).
- Nėra galimybės generuoti meniu, navigacijos.
- Nepalaiko nuotolinių resursų, paslaugų (API).
- Įrankis palaiko tik kelis duomenų tipus.

Visgi įrankį galima naudoti, kaip greito kodo generatorių MVVM projektams, tačiau dėl nepilnai realizuoto generavimo yra reikalingas žmogaus įsikišimas. Siūloma atlikti įrankio praplėtimo darbus bandyti praplėsti įrankio funkcionalumą.

4.3. Siūlomi pakeitimai, atnaujinimai

Šiuo metu įrankyje trūksta tam tikro funkcionalumo, norint generuoti platesnio panaudojimo programėles, dėl to yra siūloma atlikti įrankio praplėtimo darbus.

- Įrankis turėtų generuoti navigaciją ir meniu, kuris leistų naviguoti tarp skirtingų modelių ir modelių vaizdų.
- Įrankis turėtų palaikyti bent vieno tipo asociacinį ryšį tarp modelių, kad būtų galima sugeneruoti sudėtingesnio tipo programėles.
- Siūloma atnaujinti naudotojo sąsajos generavimo būdą, keičiant „Activity“ klasių generavimą į „Fragment“ klasių generavimą. Tai leistų lengviau integruoti meniu generavimą ir „Android Navigation API“.

Pasiūlyti pakeitimų ir atnaujinimo darbai buvo priimti ir vykdomi, detaliau apie atliktus pakeitimus 4.4 skyriuje.

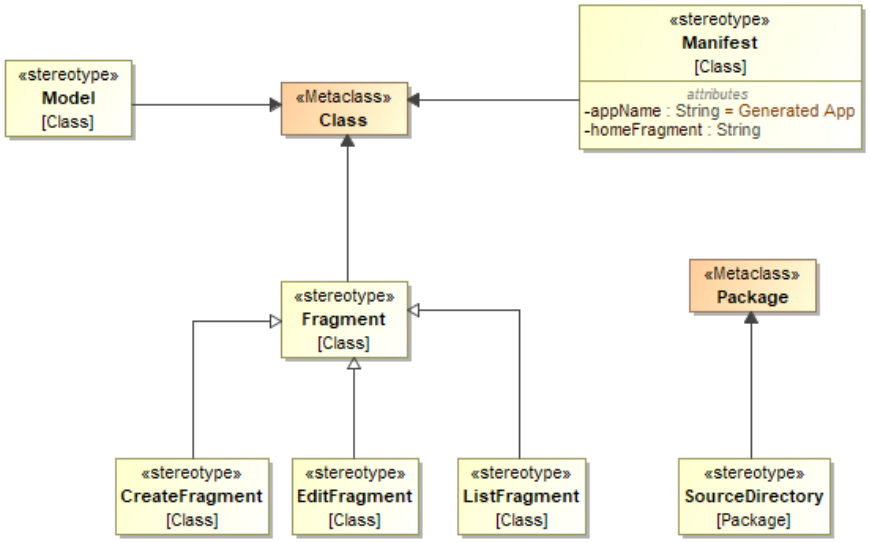
4.4. Atlikti atnaujinimai

Įrankyje buvo atnaujintas naudotojo sąsajos generavimo procesas. Šiam atnaujinimui atlikti buvo atnaujintas „MagicDraw“ profilis, generavimo taisyklės, bei naudojami kodo šablonai. Profilyje buvo atsisakyta „Activity“ stereotipų ir vietoje jų buvo integruoti „Fragment“ stereotipai. Detalesnė informacija apie profilio pakeitimus pateikta 21 lentelėje, taip pat pateikiama atnaujinto profilio diagrama 34 paveikslėlyje.

21 lentelė Atnaujintų stereotipų aprašymai

Meta klasė	Stereotipas	Aprašymas	Generuojama
Class	„Fragment“	Stereotipas suteikiamas klasei, kuri atstovauja „Android fragment“ klasę.	Generuojamos visos galimos „Android fragment“ nurodytai klasei. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„ListFragment“	Stereotipas suteikiamas klasei, kuri atstovauja sąrašo „Android fragment“ klasę.	Generuojama „Adnrdoid fragment“ klasė, kurioje bus rodomas tik modelių sąrašas su nurodytais parametrais. Klasė turi turėti sąryšį su klase, kuri

Meta klasė	Stereotipas	Aprašymas	Generuojama
			turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„CreateFragment“	Stereotipas suteikiamas klasei, kuri atstovauja kūrimo formos „Android fragment“ klasę.	Generuojama „Android fragment“ klasė, kurioje bus rodoma tik kūrimo forma į kurią reikia pateikti atitinkamą informaciją. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.
	„EditFragment“	Stereotipas suteikiamas klasei, kuri atstovauja keitimo formos „Android fragment“ klasę.	Generuojama „Adnrdoid fragment“ klasė, kurioje bus rodoma tik keitimo forma į kurią galima pateikti tik atitinkamą informaciją. Klasė turi turėti sąryšį su klase, kuri turi stereotipą „Model“, norint nusakyti iš kokio modulio turi būti skaitoma informacija.



34 pav. Atnaujintas įrankio „MagicDraw“ stereotipų profilis

Baigus naudotojo sąsajos generavimo atnaujinimo darbus buvo praplėstas įrankis, kad palaikytų „vieną su daug“ ryšį tarp „Entity“ klasių. Klasių diagramoje užtenka nurodyti ryšį tarp klasių ir

nustatyti daugialypumą (angl. *multiplicity*). Įrankis automatiškai sugeneruoja reikalingas klases, laukus ir funkcijas.

Vėliau buvo integruota meniu ir navigacijos generavimas. Buvo pasirinkta pasinaudoti „Android“ platformos siūlomu navigacijos komponentu (angl. *navigation component*). Šiuo metu per klasių diagramą meniu ir navigacijos informacijos nėra galimybės perduoti į generavimo procesą, tačiau buvo pritaikyta taisyklė pagal kurią šiuo metu įrankis generuoja visoms „Entity“ klasėms meniu punktą į sąrašą, jei „Entity“ klasė turi sugeneruotą „ListFragment“ klasę. Taip pat yra generuojamas navigacijos komponentas, kuriame apsirašo galimi keliai tarp „fragment“ klasių. Šiuo metu navigacija yra generuojama pagal šią taisyklę – iš „ListFragment“ į „CreateFragment“ arba iš „ListFragment“ į „EditFragment“.

5. Eksperimentinis „Android“ programėlių kūrimo tyrimas

Šiame skyriuje aprašomas eksperimentinis tyrimas ir gauti rezultatai. Eksperimentinio tyrimo metu buvo bandoma ištirti ar sukurtas įrankis pagrįstas MDE metodika pagreitina „Android“ programėlių kūrimo procesą. Tyrimo metu buvo išmatuojama kaip greitai galima sukurti „Android“ programėles tradiciniu būdu ir naudojantis sukurtu įrankiu.

5.1. Tyrimo eiga

Tyrimo metu buvo bandoma sukurti „Užrašinė“ ir „Minimalistinis užduočių sekimo“ programėles, pagal iškeltas specifikacijas, dvejais kūrimo būdais: tradiciniu būdu programuojant ir naudojantis sukurtu įrankiu, kuris yra pagrįstas MDE metodika.

5.2. Programėlių aprašymai

Šiame skyriuje pateikiama tyrimo metu kariamų programėlių pagrindiniai specifikacijos aspektai, pagal kuriuos buvo kuriamos programėlės.

5.2.1. „Užrašinė“ programėlės aprašymai

Programėlė turi būti kuriama vadovaujantis „Google“ teikiamomis gairėmis ir naudojant MVVM architektūrą. Pagrindinis programėlės funkcionalumas – valdyti asmeninių užrašų knygutę. Prie užrašo galima saugoti pavadinimą ir aprašymą. Visa informacija turi būti saugoma lokaliai telefone. Naudotojo sąsajos dizainui specifiniai reikalavimai nėra keliami. Tačiau programėlei yra keliami funkciniai reikalavimai kurie yra:

- Peržiūrėti užrašų sąrašą.
- Kurti naują užrašą.
- Redaguoti užrašą.
- Trinti esamą užrašą.

5.2.2. „Minimalistinis užduočių sekimo“ programėlės trumpoji specifikacija

Programėlė turi būti kuriama vadovaujantis „Google“ teikiamomis gairėmis ir naudojant MVVM architektūrą. Pagrindinis programėlės funkcionalumas yra valdyti užduočių kategorijas ir užduotis. Užduotys gali būti priskirtos sukurtoms kategorijoms.

Naudotojo sąsajos dizainui specifiniai reikalavimai nėra keliami. Tačiau programėlei yra keliami funkciniai reikalavimai kurie yra:

- Peržiūrėti užduočių sąrašą.
- Kurti naujas užduotis.
- Redaguoti esamas užduotis.
- Trinti esamas užduotis.
- Peržiūrėti kategorijų sąrašą.
- Kurti naujas kategorijas.
- Redaguoti esamas kategorijas.
- Trinti esamas kategorijas.

5.3. Gauti rezultatai

Atlikus eksperimentą buvo gauti rezultatai, kurie pateikti 22 lentelėje.

22 lentelė Eksperimento rezultatai

	„Užrašinė“ tradiciniu būdu	„Užrašinė“ naudojantis sistema	„Užduočių sekimo“ tradiciniu būdu	„Užduočių sekimo“ naudojantis sistema
Programuotojo sukurtų Java failų kiekis	10	0	22	0
Sugeneruotų Java failų kiekis	-	10	-	19
Programuotojo sukurtų XML failų kiekis	37	0	37	0
Sugeneruotų XML failų kiekis	-	34	-	38
Viso failų	47	44	59	57
Programuotojo parašytų Java kodo eilučių kiekis	697	0	1920	0
Sugeneruoto Java kodo eilučių kiekis	-	618	-	1297
Kūrimo laikotarpis	112 minučių	7 minutės	182 minučių	8 minutės

Pagal gautus rezultatus naudojantis sukurta sistema, kuri pagrįsta MDE metodika, pasirinktų „Android“ programėlių kūrimas užtruko 95 % greičiau negu lyginant su tradiciniu programėlių kūrimu. Taip pat reikėtų atkreipti dėmesį, kad sugeneruotų programėlių failų ir programos eilučių kiekiai yra mažesni.

5.4. Rezultatų tikslumas

Eksperimentinį tyrimą atliko pats studentas, kurio metu studentas sukūrė ankščiau specifikuotas programėles tiek tradiciniu būdu ir naudojantis įrankiais, kartu surinko visas metrikas. Dėl to norima paminėti, kad rezultatai gali kisti priklausomai nuo programėlių kūrėjo patirties, tiek asmens, kuris naudosis įrankiu. Taip pat rezultatus gali stipriai įtakoti programėlių funkciniai ir nefunkciniai reikalavimai, kas gali lemti galutinius rezultatus.

Išvados

1. Atlikus MDE metodikos analizę buvo pastebėta, kad bandant realizuoti įrankį pagrįstą MDE ir nenustačius kuriamų „Android“ programėlių funkcinių ribų ar dalykinės srities, gali apsunkėti modeliavimo procesas ir transformacijos realizavimas. Nenurodžius programėlių funkcinių ribų visa informacija apie programėlę turėtų būti pateikiama per modelius, kas gali prailginti sistemos modeliavimo procesą, taip pat tai žymiai apsunkintų transformacijos proceso realizavimą. Dėl to buvo pasirinkta apibrėžti generuojamų programėlių funkcinius reikalavimus ir buvo tikimasi, kad sukurtas įrankis sugebės automatiškai sugeneruoti „Android“ programėlės CRUD funkcionalumą, kuris yra beveik visada realizuojamas kiekvienoje programėlėje.

2. Atsižvelgiant į mokslinėje literatūroje siūlomus ir realizuotus sprendimus, buvo pasirinkta transformacijos procesui naudoti klasių diagramas norint sugeneruoti „Android“ programėles. Nes šiose diagramose galima perduoti programėlės struktūros aspektus, kuriems galima pritaikyti transformacijos taisykles.

3. Realizuotas įrankis atitinka užsibrėžtus reikalavimus. Sėkmingai pavyksta generuoti „Android“ programėles iš pateiktų klasių diagramų.

4. Analizuojant ir bandant įrankį buvo aptikti įrankio trūkumai, dėl to buvo pateikti atitinkami atnaujinimo darbai. Atnaujinimo darbai buvo atlikti sėkmingai, kurie praplėtė įrankio funkcionalumą ir optimizavo naudoto sąsajos transformacijos procesą.

5. Atlikus eksperimentinį tyrimą buvo nustatyta, kad sukurtas įrankis pagreitina „Android“ programėlių kūrimą 95 %. Tačiau rezultatus gali stipriai įtakoti programėlių kūrėjų patirtis, naudotojo patirtis, kuris bandys generuoti programėles sukurtu įrankiu, iškelti funkciniai ir nefunkciniai programėlės reikalavimai.

6. Atsižvelgus į gautus rezultatus, MDE metodiką pavyko pritaikyti „Android“ programėlių kūrimo procesui pagreitinti, tačiau kaip ir buvo minėta, buvo nustatytos „Android“ programėlių funkcinės ribos, kas palengvino transformacijos proceso realizavimą. Tačiau tai susiaurino galimų „Android“ programėlių kūrimo tipus, taikant įrankį programėlių kūrimo procese.

Literatūros sąrašas

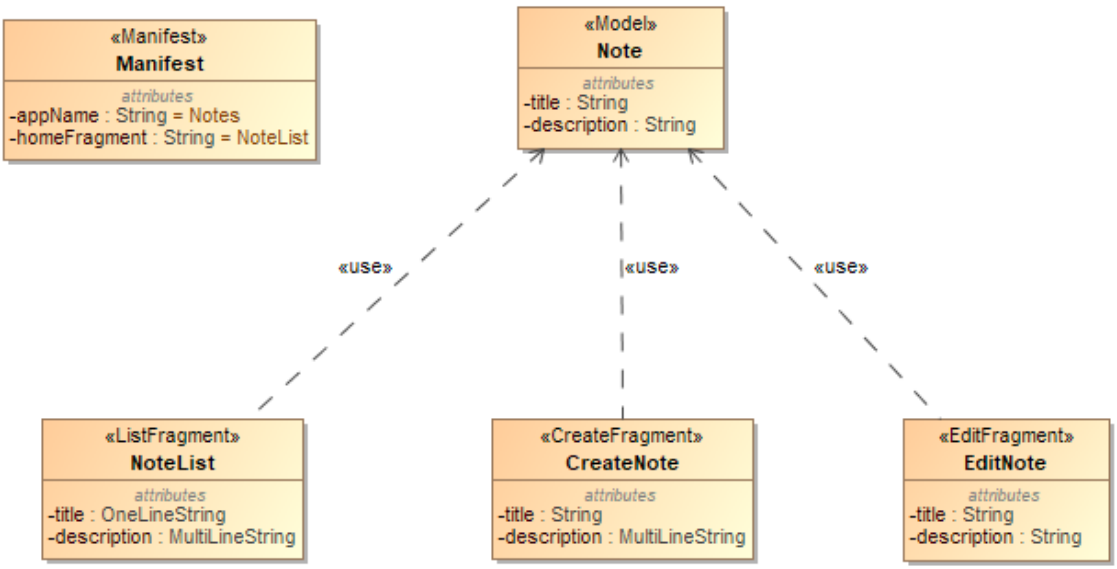
1. S. Kent, „*Model Driven Engineering*” Lecture Notes in Computer Science Integrated Formal Methods, p. 286–298, 2002.
2. D. C. Schmidt, "*Guest Editor's Introduction: Model-Driven Engineering*" in *Computer*, vol. 39, no. 2, pp. 25-31, Feb. 2006, doi: 10.1109/MC.2006.58.
3. Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, Guoai Xu, „*Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets*“, 2018.
4. AppBrain, „Most popular Google Play categories“, 2021. (Prieiga per internetą: https://www.appbrain.com/stats/android-market-app-categories) [Žiūrėta 2021-05]
5. Miller. J., Mukerji. J, „MDA guide version" 1.0.1. Technical report, Object Management Group (OMG) (2003)
6. B. Selic, "*The pragmatics of model-driven development*" in IEEE Software, vol. 20, no. 5, pp. 19-25, Sept.-Oct. 2003, doi: 10.1109/MS.2003.1231146.
7. I. Sommerville. „*Software Engineering (9th Edition)*“, p. 138-141, 2011.
8. Vicente García Díaz, Edward Rolando Núñez Valdez, Jordán Pascual Espada, B. Cristina Pelayo García Bustelo, Juan Manuel Cueva Lovelle, Carlos Enrique Montenegro Marín. „*A brief introduction to model-driven engineering*“. Tecnura, 18(40), p. 127-142, 2014.
9. OMG. MDA Guide version 1.0.1, 2003. (prieiga per internetą: https://www.omg.org/news/meetings/workshops/UML_2003_Manual/00-2_MDA_Guide_v1.0.1.pdf) [žiūrėta 2019-08]
10. Ibn Batouta, Zouhair & Dehbi, Rachid & Talea, Mohamed & Hajoui, Omar. „*Multi-criteria analysis and advanced comparative study between automatic generation approaches in software engineering*“. 81. 609-620, 2015.
11. Krzysztof Czarnecki & Simon Helsen. „*Classification of model transformation approaches. In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*“ (Vol. 45, No. 3, pp. 117, Oct. 2003).
12. OMG, „*MOF Model to Text Transformation Language v1.0*“, (Prieiga per internetą: http://www.lifl.fr/~dumoulin/enseign/pje/doc s/MTL-08-01-16.pdf) [žiūrėta 2020-09]
13. Djurić, D., Gašević, D., & Devedžić, V. „*Ontology modeling and MDA“*. Journal of Object technology, 4(1), p. 109-128, 2005.
14. OMG, „*MDA specifications*“, (Prieiga per internetą: http://www.omg.org/mda/specs.htm#CWM) [žiūrėta 2020-09]
15. Meijler, T. D., Nytun, J. P., Prinz, A., & Wortmann, H. (2010). „*Supporting finegrained generative model-driven evolution. Software & Systems Modeling“*, 9(3), p. 403-42
16. Pelechano, V., Albert, M., Muñoz, J., & Cetina, C. „*Building Tools for Model Driven Development. Comparing Microsoft DSL Tools and Eclipse Modeling Plug-ins*“. In DSDM, Otc. 2006.
17. Sandeep, Shukla, first, and , “*Metamodeling: What is it good for?*”, vol. 26, May 2009.
18. Microsoft, „*About Domain-Specific Languages*“. (prieiga per internetą: https://docs.microsoft.com/en-us/visualstudio/modeling/about-domain-specific-languages?view=vs-2019) [žiūrėta 2020-09]

19. Microsoft, „*Overview of Domain-Specific Language Tools*“. (prieiga per internetą: https://docs.microsoft.com/en-us/visualstudio/modeling/overview-of-domain-specific-language-tools?view=vs-2019) [žiūrėta 2020-09]
20. Altova. „*UModel*“. (prieiga per internetą: http://www.altova.com/umodel/uml-code-generation.html) [žiūrėta 2020-09]
21. Modeliosoft. „*Modelio*“ (prieiga per internetą: https://www.modelio.org/) [žiūrėta 2020-10]
22. Massachusetts Institute of Technology, MIT App Inventor. (prieiga per internetą: http://appinventor.mit.edu) [žiūrėta 2020-10]
23. Parada Abilio & Brisolara Lisane. „*A Model Driven Approach for Android Applications Development*“. 10.1109/SBESC.2012.44. 2012.
24. Freitas Fabiano & Paulo Henrique M. Maia. „*JustModeling: An MDE Approach to Develop Android Business Applications*“. 10.1109/SBESC.2016.016. 2016.
25. AndroMDA. „*AndroMDA*“ (prieiga per internetą: https://www.andromda.org/) [žiūrėta 2019-08]
26. Openmdx. „OpenMDX“ (prieiga per internetą: http://www.openmdx.org/) [žiūrėta 2019-08]
27. TigerTeam Aps. „TigerTeam – TRIMM – Model Driven Generator“ (prieiga per internetą: https://trimm.tigerteam.dk/) [žiūrėta 2019-08]
28. Metamodelis (prieiga per internetą: https://lt.wikipedia.org/wiki/Vaizdas:Metamodelis.jpg) [žiūrėta 2019-08]

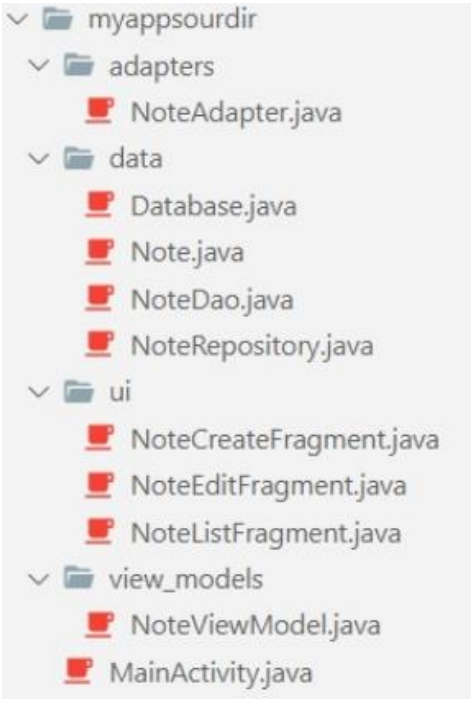
Priedai

1 priedas. „Užrašinė“ programėlės klasių diagrama ir sugeneruotos programėlės vaizdai

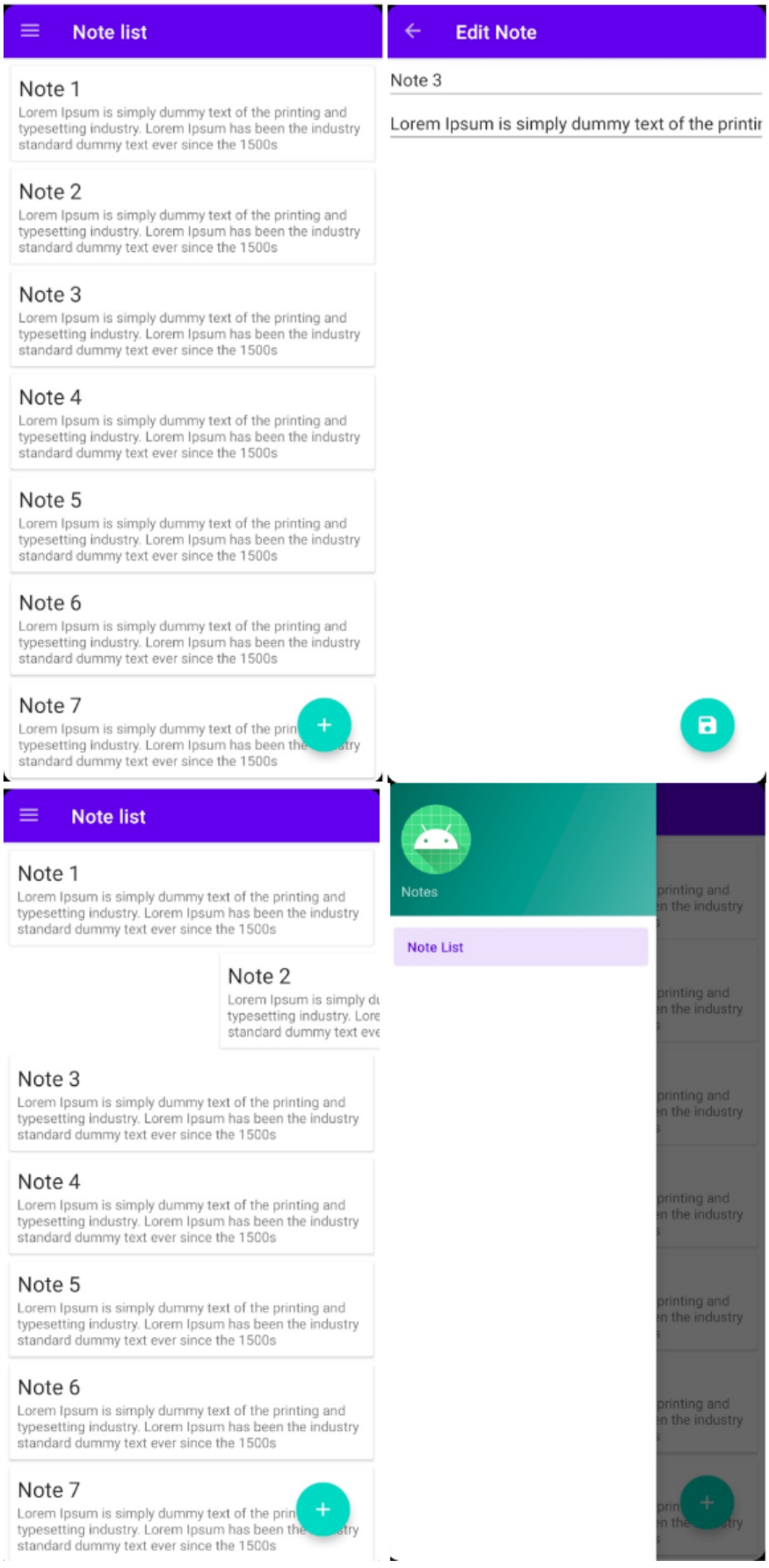
Toliau yra pateikiama „Užrašinė“ programėlės klasių diagrama, projekto Java sugeneruotų failų struktūra, sugeneruotos programėlės vaizdai su testavimo duomenimis, žiūrėti 35 – 37 pav.



35 pav. „Užrašinė“ programėlės klasių diagrama



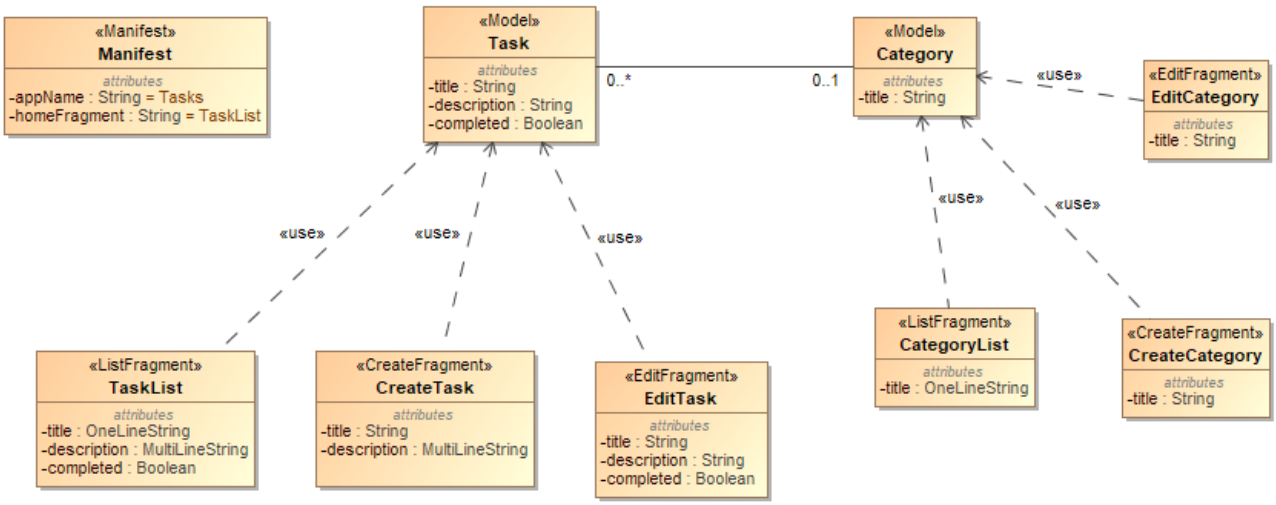
36 pav. „Užrašinė“ programėlės sugeneruota Java failų struktūra



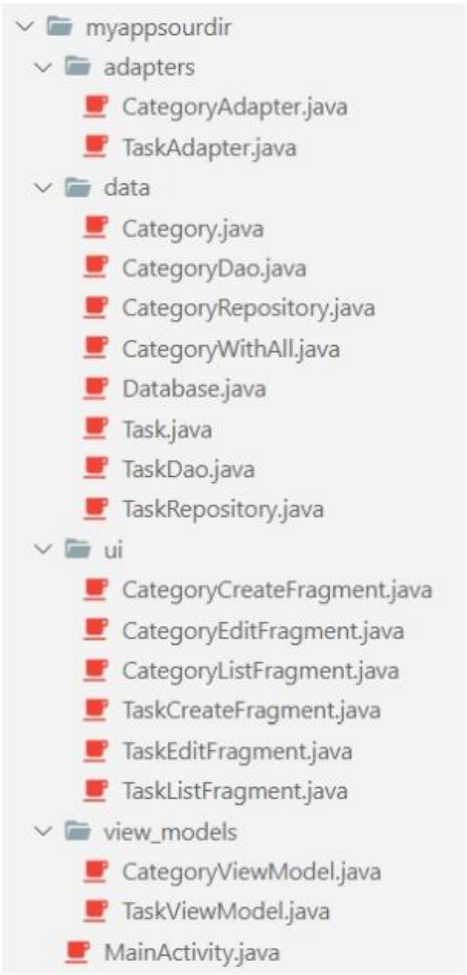
37 pav. „Užrašinė“ programėlės sugeneruota naudotojo sąsaja

2 priedas. „Užduočių sekimas“ programėlės klasių diagrama ir sugeneruotos programėlės vaizdai

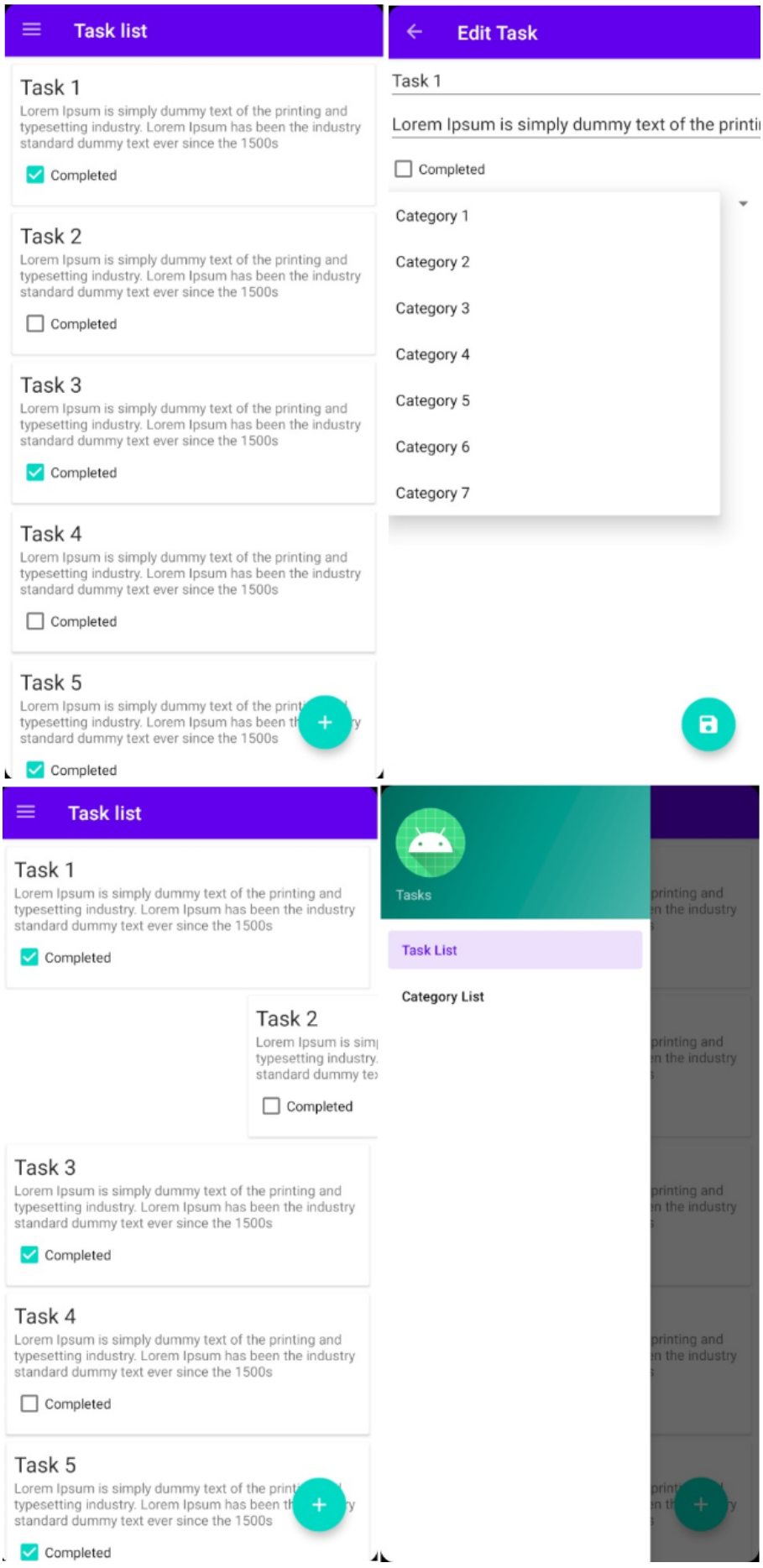
Toliau yra pateikiama „Užduočių sekimo“ programėlės klasių diagrama, projekto Java sugeneruotų failų struktūra, sugeneruotos programėlės vaizdai su testavimo duomenim, žiūrėti 38 – 41 pav.



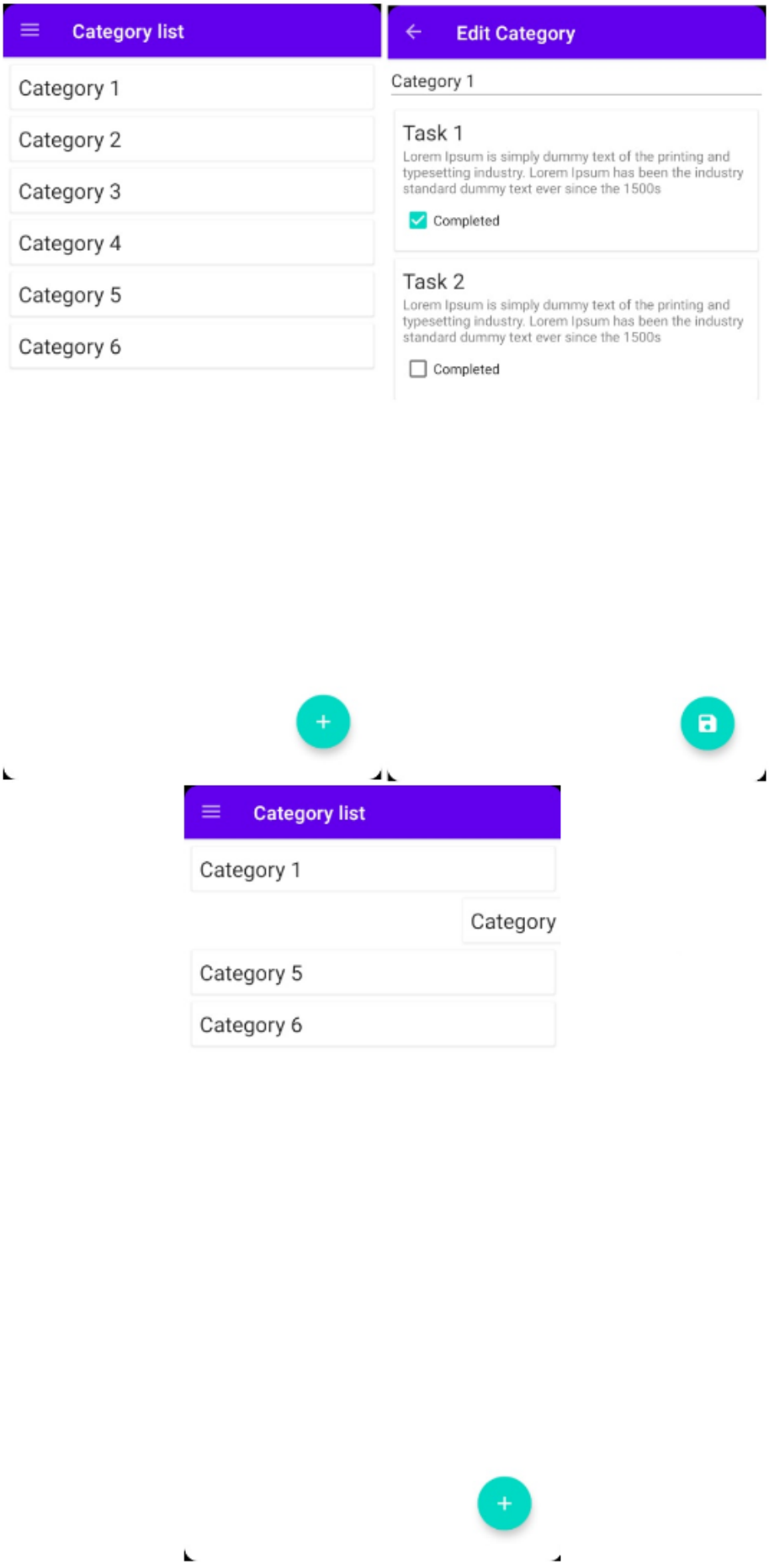
38 pav. „Užduočių sekimo“ programėlės klasių diagrama



39 pav. „Užduočių sekimo“ programėlės sugeneruota Java failų struktūra



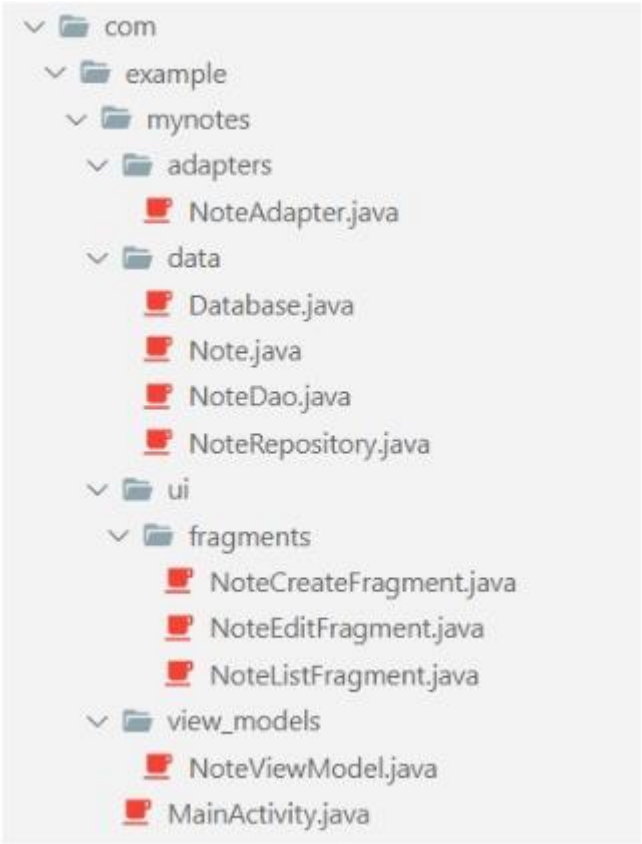
40 pav. „Užduočių sekimo“ programėlės sugeneruota naudotojo sąsaja, peržiūrint užduotis



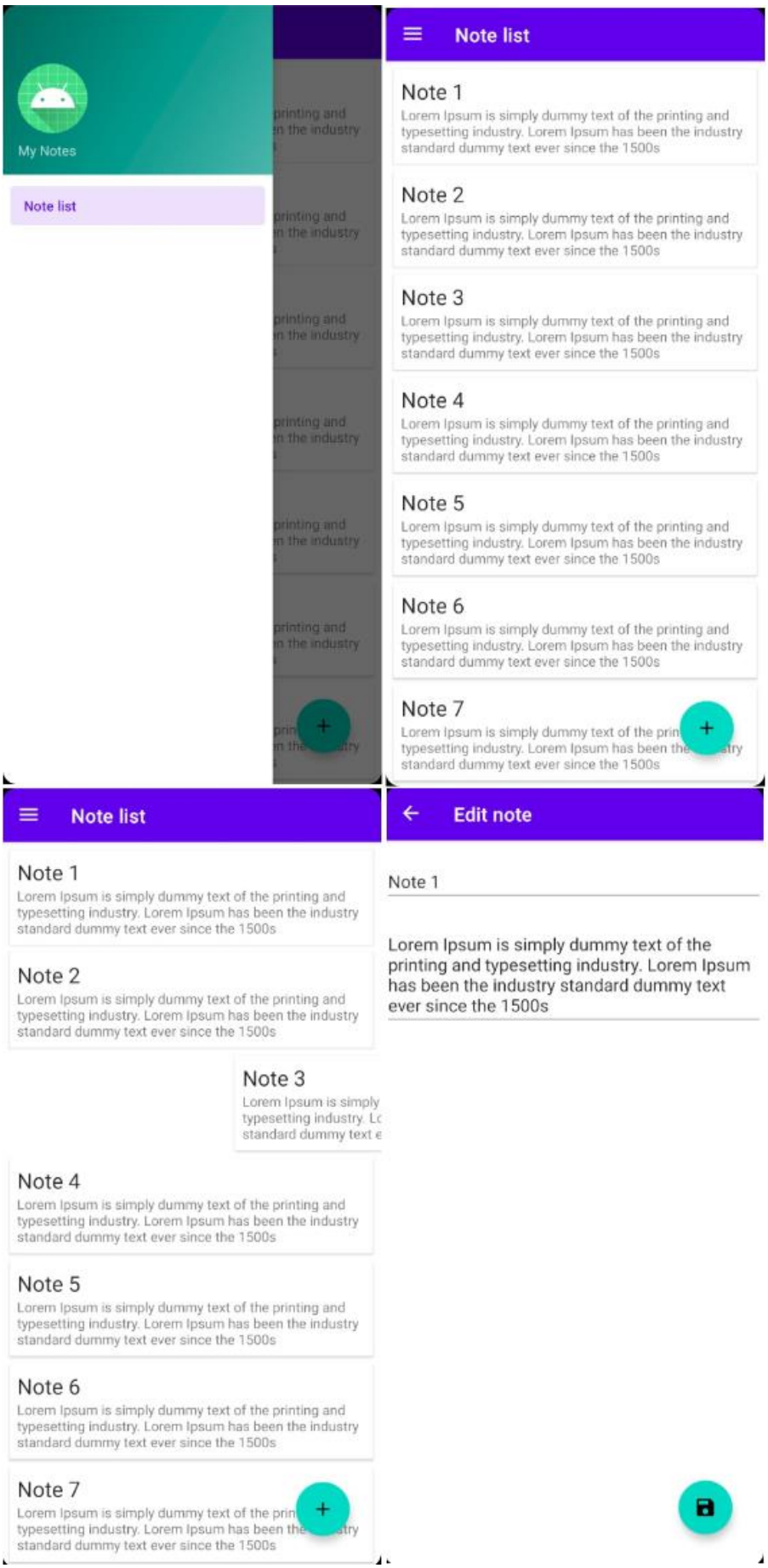
41 pav. „Užduočių sekimo“ programėlės sugeneruota naudotojo sąsaja, peržiūrint kategorijas

3 priedas. „Užrašinė“ programėlės Java failų struktūra ir sukurtos programėlės vaizdai

Toliau yra pateikiama „Užrašinės“ programėlės projekto Java failų struktūra, programėlės vaizdai su testavimo duomenim, žiūrėti 42 – 43 pav. Programėlė buvo kurta tradiciniu būdų.



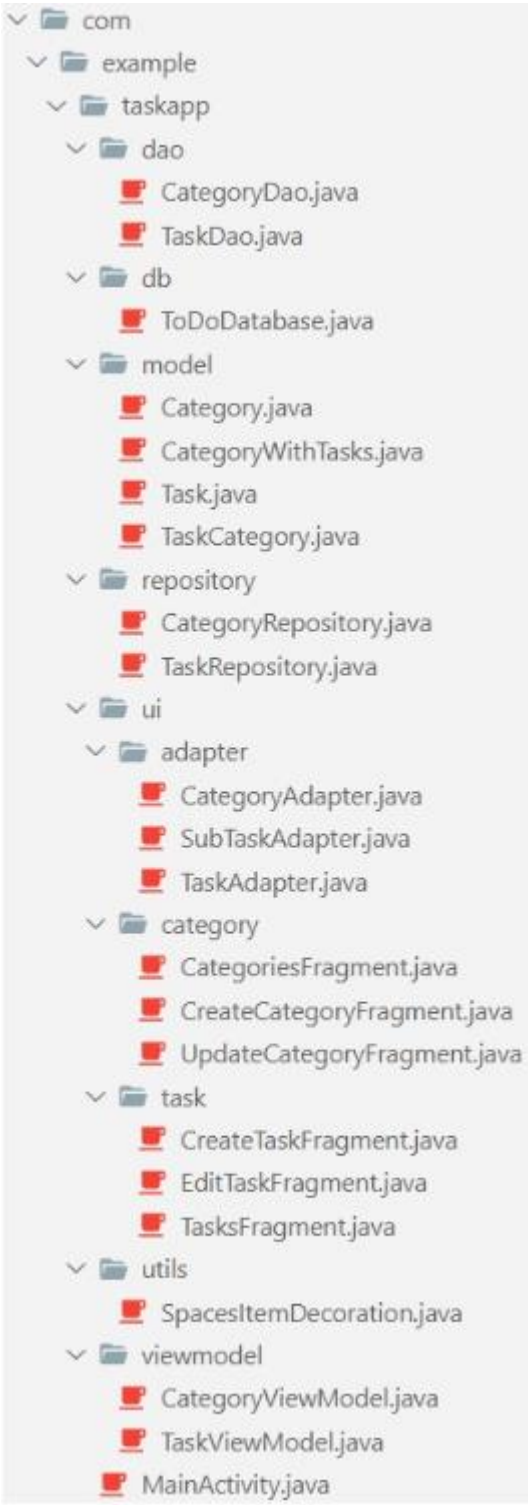
42 pav. „Užrašinė“ programėlės sukurta Java failų struktūra



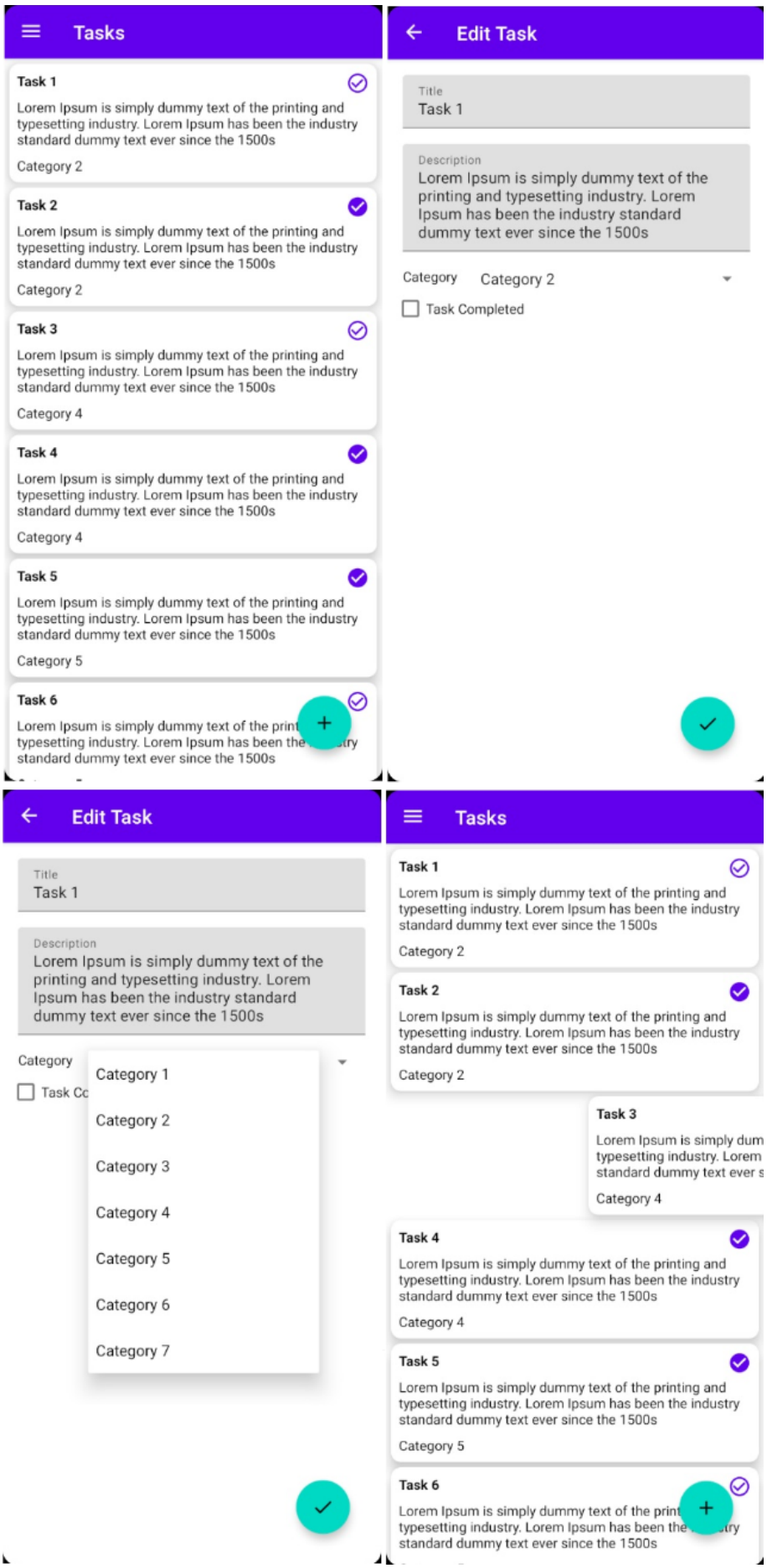
43 pav. „Užrašinė“ programėlės sukurta naudotojo sąsaja

4 priedas. „Užduočių sekimas“ programėlės Java failų struktūra ir sukurtos programėlės vaizdai

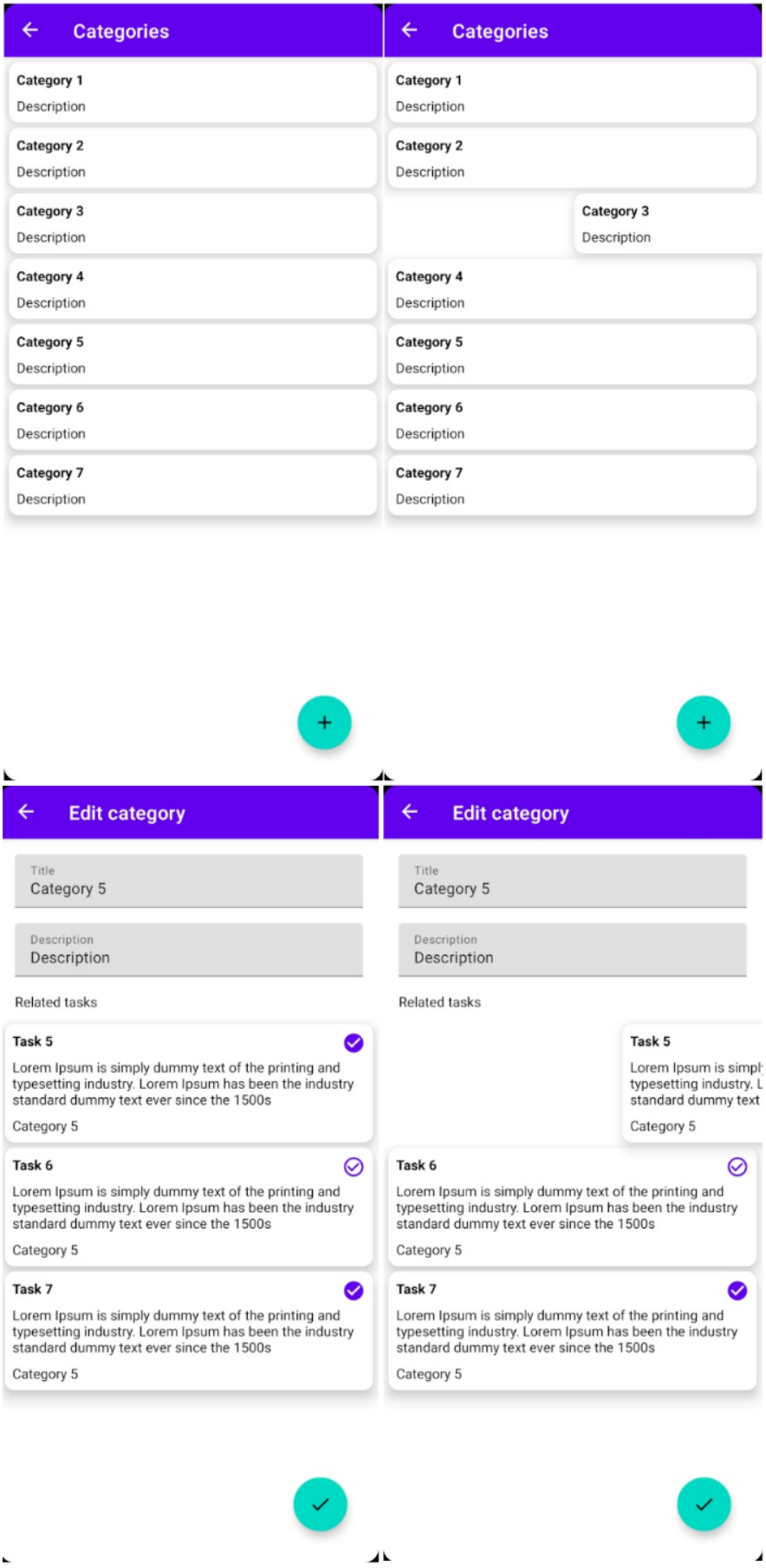
Toliau yra pateikiama „Užduočių sekimo“ programėlės projekto Java failų struktūra, programėlės vaizdai su testavimo duomenimis, žiūrėti 44 – 46 pav. Programėlė buvo kurta tradiciniu būdu.



44 pav. „Užduočių sekimo“ programėlės sukurta Java failų struktūra



45 pav. „Užduočių sekimo“ programėlės sukurta naudotojo sąsaja, peržiūrint užduotis



46 pav. „Užduočių sekimo“ programėlės sukurta naudotojo sąsaja, peržiūrint kategorijas