

VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

KOMPIUTERIJOS KATEDRA

Bakalauro baigiamasis darbas

Bičių avilių monitoringas

Atliko:

Titas Kulikauskas parašas

Vadovas:

dr. Tomas Plankis

Vilnius

2018

Turinys

Sutartinis terminų žodynas **4**

Santrauka **5**

Summary **6**

Įvadas **7**

1. Motyvacija ir temos aktualumas **7**

1.1. Tikslai ir sprendžiami uždaviniai 7

1.2. Bitininkavimo apžvalga 7

2. Rinkos apžvalga ir analizė **9**

2.1. Android programėlės 9

2.2. Klasikiniai žurnalai ir apžiūros elementų svarba 12

2.3. Klaidos ir pastebėjimai išbandytose programėlese 15

3. Išmaniųjų programėlių dizaino gairės **16**

4. Įrankių ir technologijų pasirinkimas **16**

4.1. Programavimo aplinka 16

4.2. Testavimo aplinka 17

5. Programėlės dizainas **18**

5.1. Vizualus dizainas ir vartotojo potyris 18

6. Programos prototipas **20**

6.1. Duomenų saugojimas 21

6.2. Programėlės kodas ir klasių struktūra 22

Išvados ir rekomendacijos **24**

Priedai **27**

A. Duomenų struktūros diagrama **27**

B. Programos klasių diagrama **28**

C. Programėlės langų prototipai **29**

D. Programėlės langų prototipai **30**

Sutartinis terminų žodynas

- Monitoringas - (*angl. monitoring*) sistemingas stebėjimas renkant informaciją
- Bitynas - bičių avilių laikymo vieta
- Motinėlė - suaugusi patelė bitė, daugumos, o kartais ir visų, avilio bičių motina
- Fragment - “Android” aplikacijos elementas, naudojamas kaip vartotojo sąsajos elementas daukartiniam panaudojimui
- Activity - Pagrindinis “Android” aplikacijos atvaizdavimo langas, talpinantis Fragment dalis
- Vartotojo potyris [Int15] - (*angl. User Experience*) įdėja nusakanti sklandų ir aiškų vartotojo sąsajos modeliavimą, atsižvelgiant į jos naudojimą
- Programėlė - sukompiliuotas programos kodas skirtas išmaniuosiuose telefonuose
- Rėmelis - struktūrinis avilio elementas, kuris laiko korį avilio viduje
- Akelė - avilio korio dalis. Korys sudarytas iš vienodo dydžio šešiakampių akelių. Akelėse bitės laiko medų arba augina lervas

Santrauka

Darbe buvo išnagrinėti esami bičių avilių priežiūros duomenų rinkimo metodai, įtraukiant klasikinius, popierinius, variantus ir išmaniąsias programėles. Išanalizuota ką jos daro tinkamai, kokius duomenis renka ir kaip juos apipavidalina. Remiantis surinkta informacija aprašytas prototipo kūrimas, atsižvelgiant į vartotojo patyrimų sritį remiantis programėlių kūrimo gairių rekomendacijomis ir preliminaraus prototipo sukūrimas.

Summary

Beehive Monitoring

Main goal of the thesis was to gather information and analyze it about beehive monitoring methods, including both, conventional pen and paper methods, and new age - smartphone applications. Based on the gathered information, analyze what information is usually being collected, why is it important and how can it be presented comfortably for the user, as not to hinder his/her ability to use it e ciently. This step heavily relies on the idea of User Experience, which dictates practical, meaningful and experiential aspects of human-computer interactions. Based on these two criteria - gathered general information about metrics used and paradigms of User Experience - the next step was to design comfortable and well structured application for the chosen smartphone platform. Having it designed, both visually and conceptually, the final step was to create a simple preliminary prototype of the said application.

Įvadas

1. Motyvacija ir temos aktualumas

Bičių svarba ekosistemai ir jų nauda, yra tema kuri įgavo daug dėmesio per apytiksliai pastaruosius dvidešimt metų, o jos aktualumas šiai dienai vis dar auga. Pradedant 1990, buvo pastebėtas globalus bičių populiacijos kritimas ir neįprastai didelis kolonijų mirtingumo skaičius. [Gre14].

Pagrindinis bičių vaidmuo yra kryžminis augalų apdulkinimas [Kri33a]. Tai apima didelius žydinčių augalų ir medžių skaičius, o svarbiausia, įskaitant ir nemažą skaičių žmogaus naudojamų pasėlių, kurie sudaro trečdalį pasaulio suvartojamų produktų [Gre14], tad jų išlikimas yra tiesiogiai susijęs su mūsų maisto poreikių patenkinimu, o šių resursų netekimas turėtų katastrofiškas pasekmes.

Didžiausi žingsniai kurių reikėtų imtis įtraukia bitėms žalingų pesticidų naudojimo mažinimą, ir galimai, nutraukimą, šio užsiėmimo kaip hobio populiarinimą ir ūkininkų skatinimą jas veisti. Šio darbo tikslas ir yra paskatinti mažos skalės bičių priežiūros supaprastinimą, bei galimai įtraukti naujus žmones užsiimti šia veikla pateikiant prieinamą ir supaprastintą požiūrį į šią veiklą mobiliosios programėlės pagalba.

1.1. Tikslai ir sprendžiami uždaviniai

Darbo tikslas yra suprojektuoti ir sukurti išmaniosios programėlės prototipą “Bičių šeimos apžiūros žurnalas”.

Pagrindiniai šio darbo uždaviniai yra:

- Esamos rinkos analizė ir apžiūra, patikrinant kas yra prieinama ir kokius sprendimus jie siūlo
- Duomenų susisteminimas ir argumentavimas kodėl reikalinga sekti tam tikras metrikas, bei suprasti kodėl jos reikalingos prižiūrint bites ir avilius
- Programėlės suprojektavimas, atsižvelgiant į vartotojo patyrimą, pagrindines programėlių kūrimo dizaino gaires ir techninius sprendimus
- Pirminio prototipo kūrimas pagal suprojektuotą jos aprašymą

1.2. Bitininkavimo apžvalga

Egzistuoja kelios lengvai pastebimos grupės kurios užsiima, ar gali užsiimti, bičių veisimu ir priežiūra:

- Profesionalai
- Mėgėjai

Pirmoji grupė nėra tikslinė šio darbo auditorija. Prižiūrint didelius kiekius avilių (skaičiai siekia šimtus avilių, ne vienoje dešimtyje bitynų), atsiranda daugybė logistikos bėdų, įskaitant ir visą jų duomenų sekimą - avilių sandara, bičių šeimų ir motinėlių priežiūra, ligų sekimas ir vaistų pateikimas aviliams. Ir tai vyksta kiekvienam aviliui individualiai. Taip pat neatmetant ir fakto, kad aviliai yra transportuojami dideliais kiekiais prie žydinčių laukų, sulūžusio ar susidėvėjusio inventoriaus sekimas ir priežiūra, bei daugybė kitų problemų, su kuriomis galimai nesusidurs žmogus prižiūrintis iki dešimties avilių savo sodyboje ar ūkyje.

Turint didelius kiekius avilių, mobilioji programėlė tampa archajiškas įrankis, prie kurio bus praleidžiama eksponentiškai daugiau laiko augant avilių skaičiui. Kadangi nebuvo taikoma užpildyti jokio tarpo šioje nišoje, darbe nebuvo nagrinėjami metodai kaip didmenininkai seka ir valdo savo bitynus.

Mėgėjai, dažnai prižiūri vos kelis avilius, neretai su intencija turėti bičių gaminamų produktų savo poreikiams, ar iš sąmoningumo gamtai, be intencijos uždirbti pakankamai, kad tai būtų vienintelis pragyvenimo šaltinis ar apskritai tikėtis kažkokių pajamų. To pasekoje bičių avilių stebėjimas tampa daug intymesnis ir asmeniškesnis dalykas, ir kiekvienam aviliui skiriama daugiau laiko, tad laikas praleidžiamas avilių apžiūrai bus skiriamas vienodai, nepriklausomai nuo pasirinkto duomenų rinkimo metodo apie esamą avilio situacija, galimai net ir bus sutaupomas suprojektavus patogią ir intuityvią programėlę, o ir pačių bitynų skaičius retai siekia daugiau vieną.

Programėlė turėtų patenkinti šiuos poreikius - būti aiški, skiriant dėmesį į ūkininkus, kurie bites augina ilgą laiką, bet tai daro maža skale, ir galimai linkę naudotis jiems įprastus metodus - popierių ir pieštuką ir pakankamai paprasta, kad neatbaidytu duomenų ir informacijos perkrova galimai naujus bitininkus susidomėjusius šiuo hobiu nesenai ir linkusius ieškoti informacijos ir pagalbinių įrankių jiems prieinamesnėje formoje.

2. Rinkos apžvalga ir analizė

Šiuo metu išmaniųjų telefonų rinką besąlygiškai užima “Android” ir “Apple” operacinių sistemų telefonai [MC18]. Asmeninis šių telefonų pasirinkimas yra skonio klausimas, tačiau atsižvelgiant į tai, kad “Android” operacinė sistema apima apie keturis penktadalius visos išmaniųjų telefonų rinkos, ši platforma tampa daug patrauklesnė norint pasiekti kuo daugiau vartotojų.

Dėl asmeninio prieinamumo prie šio tipo įrenginių, ir fakto, kad tai galimai pasiekia daugiau vartotojų, bus nagrinėjamos tik “Android” platformos išmaniosios programėlės bičių avilių stebėjimui ir žurnalo tipo programėlės. Taip pat bus atsižvelgta ir į kelias žiniatinklio tipo aplikacijas, tiesiogiai papildančias fizinių įrenginių funkcionalumą ar turinčias prieigą prie tų pačių duomenų kaip ir jų programėlė, kurios yra pasiekiamos tiesiai iš telefono ar kompiuterio naršyklės. Jos bus traktuojamos kaip lygiavertės programėlės.

Yra kelios pagrindinės grupės programėlių, kurias galima iškart pastebėti:

- Nemokamos
- Mokamos
- Atviro kodo

Dauguma mokamų programėlių ir kompanijų siūlančios šias paslaugas suteikia galimybę kurį laiką nemokamai išmėginti jų produktą (*angl. freetrial*) prieš įsipareigojant jų produktą naudoti už užmokestį.

Atviro kodo programėlių “Android” platformai nepavyko rasti, nors dalis kompanijų skelbiasi esančios atviro kodo. Ką tai įprastai jų kontekste reiškia, tai yra papildomų daviklių prieinamumas gamybai namų sąlygomis ir atvira programavimo sąsaja bendrauti su šiais davikliais. Neretai pasitaiko ir atviro kodo avilio brėžinių, kurie pateikti kaip kompiuterinio dizaino failai įvairiems lazeriniams spausdintuvams.

Šioje skiltyje bus nagrinėjami nemokami ir išmėginimo laiką (*angl. trial period*) turintys produktai skirti “Android” operacinei sistemai, bus palyginti keli žiniatinklio sprendimai ir bus smarkiai atsižvelgta į klasikinius variantus naudojant paruoštus, atspausdintus, popieriaus lapus.

2.1. Android programėlės

Išvis buvo išmėginta ir apžiūrėta 14 skirtingų programėlių - įskaitant ir porą rastų trečiuose šaltiniuose, neesančių “Google Play Store”. Jos visos pasižymi labai panašiu funkcionalumu - dalis kažko neturi, bet turi tai ko neturi likusios ir atvirkščiai. Deja jos visos turi labai nepatogų

ar klaidinantį elementų išdėstymą, ko pasekoje tik pradėjus naudotis programėle būna sunku rasti norimus duomenis, o kartais net suprasti loginę seką kaip patekti iš vienos programos skilties į kitą. Prastas vartotojo patyrimas yra viena iš pagrindinių priežasčių kodėl programėlės būna ištrinamos beveik iškart jas pamėginus. [Lab16] [Var15]. Nesąmoningas vartotojo klaidinimas ir poreikis mokintis konkrečios programėlės logiką, norint ja naudotis, netenkina ir atbaido labai didelę dalį vartotojų. Labai svarbu, kad programėlė būtų kuo aiškesnė, o šio dalyko aiškumas atsiranda iš apibrėžtų standartų ir nusistovėjusių praktikų kurių laikosi dauguma populiariausių programėlių.

Tarp daugybės išmėgintų programėlių ryškiai išsiskiria viena - “Apiary Book” [Ior17]. Ši programėlė, ir kitoje skiltyje nagrinėjami klasikiniai duomenų rinkimo variantai buvo naudojami kaip didžiausi atskaitos taškai ieškant optimalaus sprendimo.

“Apiary book” buvo išbaigčiausia ir funkcionaliausia išmėginta programėlė. Nors ji ir turėjo visas kitų programėlių galimybes paimtas kartu, ji turėjo ir nemažai savos informacijos, kaip pavyzdžiui pelno ir finansų skaičiavimas, avilių perkėlimo sekimas, rankinis avilio svorio sekimas, bičių skaičiaus avilyje spėjimas pagal įvedamus duomenis ar augmenijos žydėjimo kalendorius.

Nors visa tai skamba teoriniame lygmenyje labai patraukliai, bet atbaido prastas didelės informacijos pateikimas ir ranka vedamų duomenų kiekis. Visi šie duomenys turi būti suvedami paties vartotojo ranka, kas tampa labai ilgu procesu arba kurio automatizavimas reikalauja labai konkrečių duomenų iš dedikuoto serverio ir išorinių servisų.

Taip pat dalis minėto funkcionalumo (kaip pavyzdžiui pelno skaičiavimas ir avilių perkėlimo sekimas) yra akivaizdžiai taikoma į profesionalius augintojus. Kaip jau minėta įžangoje, turint didelius, komercinius, kiekius avilių tampa neberealu visus duomenis sekti programėlėje. O turint ne daugiau nei dešimt avilių, jie vargiai atnešt pelno ar bus nugabenti į daugiau nei du taškus prie žydinčių laukų, tad toks funkcionalumas tampa nereikalingas.

Pagrindinės skiltys, kurias turi bent jau didžioji dalis programėlių, ir kurios tenkina ir domina šio darbo kriterijus yra:

- Bityno(-ų) įvedimas
- Avilių ir motinėlių administracija
- Apžiūros ir jų registravimas

Šios skiltys papildo viena kitą. Bitynas yra pagrindinė avilių laikymo vieta. Jame registruojami aviliai. Aviliams priskiriamos motinėlės. Jos gauna daugiau dėmesio ir atskirą skiltį duomenų agregavimui, nes yra kritinė avilio dalis.

Ir pagrindinė akcentuojama šio darbo dalis - apžiūrų žurnalo vedimas. Kiekvienam aviliui galima atlikti apžiūrą, kurią užpildžius duomenys yra susisteminami ir matoma bendra avilio būklė ir sveikata.

Kiekviena iš šių kategorijų renka duomenis, kurių dalis yra universali visur. Tokie laukai kaip "Pavadinimas" yra priskiriami ir bitynui ir aviliui juos identifikuoti programėlėje. Toliau esančioje lentelėje apžvelgiami konkretūs laukai kurie yra mus dominančiose srityse ir atvaizduojami analizuotose programėlėse.

1 lentelė. Bitynai ir Motinėlės

Bitynas	**Motinėlė**
Identifikuojantis pavadinimas	Ar yra spalvos identifikacija?
	Jei taip, kokia spalva
Bityno laikymo vieta	Veislė
	Amžius
Užrašai ir papildoma informacija apie bityną	Šaltinis iš kur gauta
	Data kuomet gauta

2 lentelė. Avilys

Avilys
Identifikacinis pavadinimas
Avilio tipas
Bičių veislė avilyje
Avilio/bičių kilmė
Rėmelių/aukštų skaičius avilyje
Sukūrimo data
Motinėlės priskyrimas

Visos šios kategorijos ir laukai, kaip ir tikėtasi, turi įvedimo, peržiūros ir redagavimo funkcijas, kurios yra pagal nutylėjimą privalomos funkcionaliam programėlės veikimui.

Kadangi apžiūros kategorija yra kritinė šio darbo dalis, jai skiriama ir daugiausia dėmesio. Joje egzistuoja ir daugiausia įvedamų ir sekamų laukų, kurių pagalba matomas bendras avilio vaizdas ir būklė.

Šie laukai tiesiogiai sąveikauja su prieš tai minėtomis kategorijomis. Visa informacija yra vedama konkrečiam aviliui ir atsižvelgiama į motinėlės būklę avilyje.

Kadangi duomenys šiai kategorijai, atsižvelgiant į apžiūrėtų programėlių parametrus, yra padriki ir dažnai nepakankamai pilni, o retais atvejais ir archajiški, šios kategorijos renkami duomenys ir laukai buvo daugiausia atsižvelgti pagal duomenis iš klasikinių - popierinių - bičių avilių žurnalų.

2.2. Klasikiniai žurnalai ir apžiūros elementų svarba

J. Kriščiuno knygoje "Bitininkystė" - išleista 1933m. - jis rašo "Knygose užrašai nepatogu daryti, nes knyga pikiu susitepa ir nešiotis nepatogu" [Kri33b]. Taip pat jis argumentuoja, kad esant galimybei, užrašus patogiausia rašyti tiesiai ant avilio stogo, smarkiai sutrumpinant visus užrašus akronimais ar nukertant galūnes, o jei tokios galimybės nėra, turėti prisegtą segtuvą kuriame laiko užrašus apie tą avilį.

Atsižvelgiant į faktą, kad nuo knygos išleidimo praėjo beveik šimtmetis ir egzistuoja išmanieji telefonai, tai tampa svarus argumentas naudoti juos - duomenys lengvai ir greitai suvedami su minimaliu teksto rašymu, esant reikalui juos galima išplėsti iki bet kokio neapibrėžto dydžio, taip pat, naudojant išmaniuosius telefonus, duomenys tampa pasiekiami ir už bityno ribų, o esant reikalui iš bet kokio kito įrenginio turinčio interneto prieigą (kas taip pat suteikia galimybę dalintis šiais užrašais nevaržomai su kitais užsiimančiais šiuo pomėgiu).

Dauguma nagrinėtų klasikinių variantų rinkti užrašams vienareikšmiškai sutinka dėl duomenų svarbos, tad jie išlieka pastovūs tarp skirtingų šaltinių ir yra išbaigti ir pateikiami sąlyginai vienodai. [uni18] [BKC17] [Son17].

Be įprastinių duomenų - data, oro sąlygos, avilio identifikavimo pavadinimas - renkami duomenys nusakantys avilio temperamentą, motinėlės būklę ir pastebėjimus, perų akių skaičių, ligas ir jų požymius, bei galimai apie pradėtus gydimus, vaistų įdėjimą ar išėmimą iš avilio.

Einant eilės tvarka, kuri turėtų logiškai sekti avilio apžiūrą, pradedama nuo pastebėjimų apie negyvų bičių skaičių prie avilio įėjimo. Mažas skaičius - tarp 1 ir 3 bičių - dar nerodo problemų, bet apžiūros metu į tai reiktų atsižvelgti ir skirti didesnį dėmesį apžiūrai ateinančius apsilankymus. Jei šis skaičius iškart yra didesnis nei penketas, tai jau gali rodyti nemažai problemų, tiek įvairias ligas, tiek galimybę, kad avilys buvo užpultas.

Sekantys dalykai ir bruožai į kuriuos reikėtų atkreipti dėmesį nusako bendrą avilio būklę - ar bitės agresyvios ir koks bendras avilio temparementas? Jei nebuvo sutrukdytas avilys paties bitininko, tai gali vėl rodyti apie avilio užpuolimą, ar tai būtų kitas, stipresnis, avilys, ar įvairūs graužikai. Ar bitės kovoja tarpusavyje? Tai ženklas apie avilio plėšimą, kito, stipresnio, avilio, ir pasako, kad pats avilys kuris buvo užpultas yra nusilpęs.

Atsižvelgus į kritinius avilio pavojus, apžiūra tęsiama. Ar buvo pastebėta motinėlė avilyje?

Sveika motinėlė yra sveiko avilio ženklas. Jos identifikavimui naudojamas spalvinis žymėjimas, padedantis ją lengviau aptikti ir nusakantis jos amžių. Jei motinėlė sveika, o bičių šeima stipri, kiaušinėlių dėjimo struktūra bus vientisa ir tolygi. Tai yra dar vienas apžiūros kriterijus, galimai nusakantis tolimesnius avilio veiksmus, tokius kaip artėjantis spietimasis, pasenusi ir silpstanti motinėlė ir bendras avilio stiprumas.

Svarbu atsižvelgti ir į bendrą avilio populiaciją - kaip smarkiai rėmeliai avilyje yra nusėsti bičių. Jei jis yra vangiai nusėtas bičių, tai galimai nurodo smarkiai silpstantį avilį ir didelį bičių mirtingumą laukuose.

Vienas ginčytinas kriterijus yra tranų akelių skaičius. Nors bendram avilio gerbūviui didelis tranų skaičius nėra kritinis, juos galima naudoti kaip erkių valdymo būdą. Dėl išsiskiriančių nuomonių, šis laukas bus paliekamas kaip neprivalomas vykdant avilio apžiūros dalį programėlėje, kartu su apytiksliu tranų populiacijos skaičiumi avilyje.

Paskutiniai bendrinės apžiūros kriterijai būtų naujų motinėlių akelių skaičius, ir ar jų yra, bei spietimosi požymių pastebėjimas.

Sekantis žingsnis avilio apžiūroje būtų ligų ir kenkėjų pastebėjimas. Bitėms gręsia nemažai ligų, paveikiančių tiek pavienes bites, tiek visą avilį. Dalis šių ligų rodo prastą avilio priežiūra, tačiau į jas visas reikia atitinkamai reaguoti ir jas pastebėjus sekti, visai kaip ir pastebėjus kenkėjus - įvairias erkutes, kandis ar kitus svetimšalius.

Dėl šių dalykų svarbu sekti ar buvo imtasi kažkokių gydimo ir priežiūros veiksmų - sekama kada ir kokie vaistai buvo įdėti į avilį, bei kada išimti.

Galiausiai - ar buvo imtasi kažkokių veiksmų šios apžiūros metu? Sukeista rėmelių tvarka avilyje, išvalytas ar pakeistas avilio dugnas ar pažeistas avilio inventorius. Ar buvo įdėta į avilį maisto? Ir žinoma įvairių smulkių kitų pastebėjimų, tokių kaip nemalonus kvapas sklindantis iš avilio, nevientisas akelių išsidėstymas rėmeliuose ir bendra rėmelių būklė, ar pastebėtas bičių pikio perteklius.

Galutiniai sekami duomenys, kurie bus renkami apžiūros metu yra:

- Apžiūros data
- Oro sąlygos remiantis bityno vieta, kuriam priklauso pasirinktas avilys
- Identifikacinis skaičius ir avilio pavadinimas
- Mirusių bičių skaičius prie avilio įėjimo
- Avilio temperamentas

- Ar yra tarpusavyje kovojančių bičių?
- Ar pastebėta motinėlė?
- Ar pastebėta kiaušinėlių akelėse? Jei taip:

  Kiaušinėlių dėjimo struktūra
- Ar pastebėta motininių akelių? Jei taip:

  Motininių akelių skaičius
- Kaip gausiai rėmeliai aptūpti bičių
- Akelių išsidėstymo struktūra
- Ar yra traninių akelių? Jei taip:

  Traninių akelių skaičius
- Ar pastebėti spietimosi požymiai?
- Ar pastebėta kenkėju ar ligų? Jei taip:

  Ligų pasirinkimas ir kenkėjų pasirinkimas iš sąrašo, pažymėti apie problemas šiame avilyje
- Ar buvo aviliui skirta ar išimta vaistų? Jei taip:

  Kas buvo įdėta ar išimta
- Ar buvo rėmeliai sukeisti vietomis?
- Ar buvo išvalytas, arba pakeistas, avilio pagrindas?
- Ar buvo pažeisto inventoriaus? Jei taip:

  Kas buvo pažeista?
- Ar buvo įdėta į avilį maisto? Jei taip:

  Ko ir kiek buvo įdėta?
- Ar avilys turėjo nemalonų kvapą?
- Bendra rėmelių būklė
- Ar pastebėtas bičių pikio perteklius?

2.3. Klaidos ir pastebėjimai išbandytose programėlese

Iš visų išbandytų programėlių, visos turėjo pastebimų ir grubių trūkumų, kurių bus stengiamasi išvengti darant šios darbo prototipą.

Bene didžiausias ir lengviausiai pastebimas - abejotini dizaino sprendimai. Daliai programėlių efektyviam naudojimui tęsti kliudė prastas elementų ir spalvų pasirinkimas, kaip šviesus fonas ir ant jo pateikiamas labai šviesus tekstas. Nors jis ir įskaitomas, kontrasto trūkumas labai kliudė greitai pastebėti ieškomos skilties.

Taip pat neaiškių ir neapibrėžtų ikonų naudojimas navigacijai. Deja, archajiškos ikonos, kurios nėra jau įprastos daugumoje programėlių (pavyzdžiui namo ikona grįžtanti į pagrindinį langą, ar veržliaraktis nuvedantis į nustatymų langą), ar ikonos be teksto nenusako iš pirmo žvilgsnio ką jos nurodo, ir norint išsiaiškinti reikia išbandyti. Tai atbaido ir klaidina vartotojus, kurie jau nėra įpratę ar išmokę naudotis šią konkrečia programėle.

Įprastų ir jau daugumoje programėlių naudojamų navigacijos kriterijų nesilaikymas taip pat klaidina ir kelia susierzinimą vartotojui, kaip telefono mygtuko “Atgal” paspaudimas išjungiantis programėlę vietoj grįžimo į prieš tai buvusi langą. Nesuprantamas ir neaiškus patekimas į atitinkamas programėles dalis verčia vartotoja arba išmokti šias neįprastas struktūras, arba, kas nutinka žymiai dažniau, nesinaudoti programėle ir ieškoti alternatyvų.

Vienas iš dalykų kuriuo deja pasižymėjo visos mėgintos programėlės buvo didelis laukų kratinys vedant duomenis. Tai sukelia gan daug sumaišties ir atrodo atbaidančiai. Vietoj kelių laukų pateikimo ir vartotojo vedimo per šiuos laukus, buvo pateikiami visi laukai įvedimui viename lange, nereikalingai apkraunantys ekraną ir verčiantys vartotoją praleisti daug laiko ieškant reikiamų duomenų.

Taip pat nereikalingų laukų pateikimas, ir tų pačių klausimų uždavimas kelis kartus sukuriant dubliavimą, kai, pavyzdžiui, įvedus bityno tašką žemėlapyje, prašoma įvesti laiko zoną atskirai. Tai ne tik yra nereikalingas programėlės išpūtimas, bet tuo pačiu ir neduodantis jokios pastebimos naudos.

Programėlių neišbaigtumas buvo taip pat didelė problema - duomenų laukų trūkumas, be galimybės įvesti trūkstamus duomenis į universalų lauką “Užrašai”. Tai pat neišbaigtumas glaudžiai siejosi su duomenų dubliavimu - dalį duomenų galima įvesti tik žiniatinklio programėlės versijoje ir darbą galimą tęsti tik programėlėje, taip verčiant vartotoją žongliruoti keliais duomenų įvedimo šaltiniais, dažnai įvedant tuos pačius duomenis dukart, norint tęsti darbą.

Paskutinis ir bene svarbiausias stabdantis veiksnys šiose programėlėse buvo prastas kodo dizainas ir duomenų saugojimas, sukeliantis neadekvačiai ilgus krovimo laikus pereinant į skirtingas

skiltis.

3. Išmaniųjų programėlių dizaino gairės

Vientisas ir įprastas programėlės dizainas yra svarbus išlaikyti dėmesiui, pritrauki galimus vartotojus ir neatbaidyti jų sudėtingu naudojimu, smarkiai atsižvelgiant į vartotojo patyrimą.

Visos didžiosios išmaniųjų telefonų operacinės sistemos turi detalius savo dizainų aprašymus. Kadangi programėlės prototipas bus kuriamas "Android" platformai, "Google" kompanija turi savo apibrėžta ir aprašytą "Material" [Goo17a] dizaino logistiką. Pagrindinis šio apibendrinto dizaino tikslas yra: "Sukurti vizualią kalbą, kuri apjungia klasikinius gero dizaino principus su išradingumu (inovatyvumu) ir technologijos bei mokslo galimybėmis." [Goo17b]

Pagrindiniai šio dizaino principai diktuoja, kad dizainas turi būti aiškus, neperkrautas ir iš anksto apgalvotas, suteikiant prasmę kiekvienam jo elementui. Jis siekia sukurti vieningą sistemą kuri pateikia vienodą vartotojo patyrimą tarp skirtingų tos pačios platformos įrenginių.

Kuriant programėlės prototipą bus smarkiai remiamasi minėtojo "Material" dizaino gairėmis, pateiktomis jų puslapyje [Goo17a]. Jame detaliai aprašomas spalvų pasirinkimas, pagrindiniai elementai ir jų išdėstymas, bei įvairių animacijų naudojimas norint patraukti vartotojo dėmesį ir suteikti loginio aiškumo programėlei.

Nors programėlė bus kuriama "Android" platformai, bei bus smarkiai atsižvelgiama į šios platformos dizaino kriterijus, buvo trumpai apžvelgti ir "Apple" telefonų ų sistemų dizaino kriterijai. Jie atsižvelgia į kriterijus kurie yra sąlyginai universalūs ir lengvai pritaikomi ne tik jų platformai. Jie diktuoja aiškumą, vientisumą, tiesioginį ir logišką atsaką į vartotojo veiksmus, sukuriant malonų patyrimą. Taip pat pabrėžiamas "pažįstamumas", įprasti navigavimo ir naudojimo gestai, tokie kaip paspaudimas, braukimas ar nutempimas palaikius elementą ilgiau.

4. Įrankių ir technologijų pasirinkimas

4.1. Programavimo aplinka

Dėl jau apibrėžto pasirinkimo kurti programėlę "Android" platformai, bus naudojamos technologijos ir įrankiai skirti šiai platformai. Be abejonės, egzistuoja didelė gausa prieinamų įrankių, skirtų rašyti kodą ir jį kompiliuoti "Android" išmaniesiems telefonams, bet išbaigčiausi ir prieinamiausi įrankiai yra vieni iš patraukliausių.

Ilgą laiko tarpą, pagrindinė "Google" siūloma programavimo aplinka, su integruotu kompilia-

toriumi, buvo “Eclipse” [Fou18], kurią galiausiai pakeitė dabartinė aplinka - “Android Studio” [Goo17c]. Šios abi programavimo aplinkos yra, ar buvo, skatinamos kaip rekomenduojamos ir oficialios, tad jų išbaigtumas, prieinamumas ir bendras patrauklumas programuotojui yra bene didžiausias.

Šios dvi, bei gausi trečių šaltinių pasirinkimo rinka, yra paremta “Android SDK” ir “Android NDK” programavimo aplinkomis, suteikiančiomis pilnas ir reikalingas bibliotekas pilnam platformos išnaudojimui ir kodo kompiliavimui, kad kodas veiktu užtikrintai ir vieningai, tarp skirtingų platformų, kaip galutinis produktas.

Mažiausiai rūpesčių sukelianti aplinka, šiuo atveju, yra paties “Google” siūloma ir aprašoma oficialiame dokumentacijos puslapyje - Android studio [Goo17c]. Atsižvelgiant ir į tai, kad ji veikia nepriekaištingai, ją lengva paleisti (šiuo atveju “Linux” platformoje) ir ji yra rekomenduoja, tai buvo pagrindinis pasirinkimas ieškant įrankių kurti programėlei.

Šiuo metu, “Android Studio”, palaiko dvi programavimo kalbas skirtas programėlių kūrimui - “Java” ir “Kotlin”.

“Java” istoriškai buvo ilgą laiką vienintelė programavimo kalba su kuria buvo galima rašyti kodą išmaniųjų telefonų “Android” programėlėms. Šia kalba gausu kodo pavyzdžių, detalių aprašymų apie įvairias priimtinas praktikas ar įvairių pamokų, detaliai analizuojančių kodo rašymo detales.

Nors “Kotlin” yra gerokai jaunesnė programavimo kalba, pirmosios versijos išleistos 2011 metais, ji paremta tuo pačiu “Java” virtualios mašinos kodo pagrindu, leidžiančiu jai veikti ant tų pačių įrenginių. Tačiau kodas tarpusavyje nėra palaikomas vienas kito dėl skirtingų kompiliatorių.

Dėl kalbos populiarumo ir asmeninės patirties dirbant su ja, buvo pasirinkta “Java” programavimo aplinka.

4.2. Testavimo aplinka

Kuriant aplikacijas įvairiems įrenginiams, yra kritiškai svarbu išmėginti produktą ant įrenginio, kuriam yra skirtas parašytas kodas. Šiuo atveju - išmanieji telefonai.

Kuriant išmaniąsias programėles, egzistuoja du pagrindiniai testavimo būdai - išbandyti ant įvairių išmaniųjų telefonų, su skirtingais parametrais, ekrano dydžiais. Arba išmėginti ant emuliatoriaus, skirto šiai platformai.

“Android studio” turi savo integruotą emuliatorių, kuris sukuria “Android” operacinės sistemos kopiją programuotojo kompiuteryje ir paleidžia išmaniąją programėlę jame. Dėja tai neužtikrina kai kurių dalykų autentiškumo, ypač atsižvelgiant į visus lietimui skirtus daviklius. Taip pat tai

reikalauja nemažai sistemos resursų, tad naudojant ne patį galingiausią kompiuterį, jaučiamas pastebimas sulėtėjimas naudojant programėlės prototipą emuliatoriuje.

Dėl šios priežasties, ir fakto, kad fizinių "Android" operacinių telefonų prieinamumas yra labai didelis, buvo naudotas šis testavimo variantas - sukompiliuotos programėlės paleidimas fiziniame įrenginyje.

5. Programėlės dizainas

Kaip ir bet kuriam projektui, užtikrintam ir konkrečiam rezultatui pasiekti reikalingas projekto planas ir detalus aprašymas. Šios programėlės planą sudaro dvi dalys - vizualus dizainas ir sklandaus vartotojo potyrio užtikrinimas, ir žinoma - programinės įrangos projektavimas.

5.1. Vizualus dizainas ir vartotojo potyris

Šiame poskyryje bus atsižvelgiama į kelis esminius dalykus. Programėlės tėkmę - sklandų jos naudojimą ir logišką veiksmų seką norint pasiekti atitinkama jos skiltį ar rezultatą, nepamirštant vizualiai netrikdančio dizaino sprendimų ir patogaus elementų išdėstymo. Ir bus atsižvelgtą į vartotojo potyrį, kurio tikslas yra sukurti patogią ir lengvai valdomą programėlę.

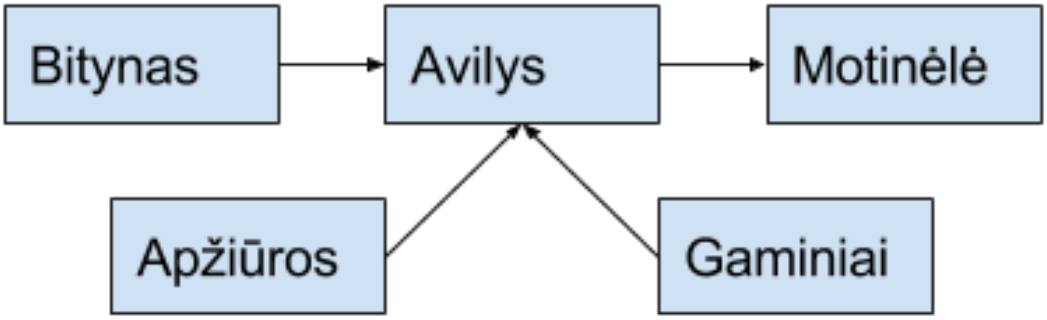
Pagrindiniai programėlės elementai, kurie buvo aptarti praėjusiuose skyriuose, yra:

- Bitynai ir jų valdymas
- Motinėlių ir avilių valdymas ir priskyrimas bitynams
- Apžiūros vykdomos individualiems aviliams

Šioje diagramoje matyti apibendrintas pagrindinių elementų sąveikavimas tarpusavyje. Bitynai turi avilius, kurie turi motinėles. Aviliams atliekamos apžiūros ir iš jų išgaunami įvairūs gaminiai.

Pagrindiniame programėlės lange, tik ją paleidus, turi būti siūlomas pridėti pagrindinis elementas - bitynas. Jį pridėjus, sekantis loginis veiksmas turi sekti avilio sukūrimas. Turint avilius, motinėlės duomenų įvedimas ir jos elemento sukūrimas, o ją turint - priskyrimas konkrečiam aviliui.

Peržiūrint pagrindinius tėvinius elementų duomenis turime pasirinkimą juos pateikti dviem būdais, atsižvelgiant į tai, kad tai yra įvestų bitynų, avilių ar motinėlių sąrašai. Pateikti kaip įprastinį sąrašą arba naudotis "Android" dizaino gairėmis apibrėžtas korteles (*angl. Cards*) [Goo18a].



1 pav. Pagrindinių elementų sąveika

Norėdami pateikti daugiau duomenų, tokių kaip paskutinės apžiūros data atlikta aviliui ar komentarai apie jo būklę, žemėlapio iškarpa pateiktam bitynui ir sukūrimo data, daug patogiau utilizuoti kortelės tipo sąrašo pateikimą. Aviliam ir bitynam bus naudojami jie. Motinėlių, apžiūrų ir gaminių sąrašams bus naudojami įprastiniai sąrašo pateikimo būdai.

Duomenys kurie bus atvaizduojami sąrašuose ir kortelėse turi suteikti aišku ir apibendrintą vaizdą apie už jų esančią informacija. Todėl bus atvaizduojami šie duomenys:

- Bityno kortelė:

  Bityno vietos įrašas lengvai identifikacijai

  Priskirtas identifikacinis pavadinimas

  Avilių skaičius konkrečiame bityne

  Paskutinės šiame bityne atliktos apžiūros data

- Avilio kortelė:

  Avilio identifikacinis pavadinimas

  Kokia motinėlė yra priskirta šiam aviliui ir jos spalvinis identifikavimas

  Paskutinės apžiūros data

  Informacija apie tai ar avilys reikalauja kokių veiksmų ar papildomo dėmesio ateinančios apžiūros metu

- Apžiūrų sąrašas:

Avilio pavadinimas, kuriam buvo atlikta apžiūra

Apžiūros data

Tai yra pagrindinė esminių elementų pateikimo įdėja. Visiem likusiems, smulkiems ir aiškiai aprašytiems elementams bus smarkiai remiamasi dizaino aprašu pateiktu “Material” puslapyje [Goo17a].

Dėl programos universalumo ir prieinamumo bus iškart pasitelkta numatyta programos kelių kalbų palaikymo praktika - teksto elementų sąrašas pateikiamas .xml tipo faile strings.xml [Goo18d].

6. Programos prototipas

Jau aprašytos pagrindinės programos dalys ir preliminari jų numatoma veiksena. Aprašyti dizaino elementai ir šaltiniai kuriais bus remiamasi juos kurti ir maketuoti. Sekantys etapai reikalauja detalaus duomenų struktūrų ir jų laikymo aprašymų, bei loginės kodo sekos aprašo - klasių diagramos.

Atsižvelgiant į naudojamos platformos ypatumus, “Android” SDK turi du mus dominančius ir kitur nenaudojamus elementus - Fragaments ir *Activities*.

Activity - yra klasės tipo objektas “Android” išmaniojoje programėlėje. Kaip šie objektai yra naudojami ir iškviečiami yra kritinis programėlės veikimo principas. Priešingai nei diktuoja dauguma programavimo paradigmų, kuriose programos kodas yra iššaukiamas *main()* dalyje, “Android” sistemos kodas yra inicijuojamas *Activity* dalyje, iškviečiant specifinius atsako metodus, atitinkančius tam tikrus programėlės gyvavimo ciklo etapus [Goo18c].

Fragment - aprašo elgesio dalį programėlės vartotojo sąsajoje esančiame *Activity* dalyje. Keli fragmentai gali priklausyti vienam *Activity* sukuriant unikalų ir modulinį vartotojo sąsajos pateikimą. Tie patys elementai gali būti panaudojami keliose skirtingose programėlės vietose. Fragmentai turi savo gyvavimo ciklą, gali priimti atskirus įvesties atvejus. Fragmentai visada turi priklausyti vienam iš esamų *Activities*, ir jo gyvavimo ciklas priklauso tiesiogiai nuo *Activity* gyvavimo ciklo. [Goo18b]

Užtikrinti sklandžiam programėlės veikimui ir vartotojo potyriui, šie elementai yra kritiškai svarbūs. Naujo *Activity* iššaukimas yra labiau resursų reikalajauntis veiksmas, nei esamų fragmentų pakartotinis naudojimas esamame lange.

Šiuo atveju, *Activities* bus naudojami visiems langams reikalaujantiems naujos informacijos pateikimo, ar kertinėm programos dalims, o *Fragments* kaip elementai kurie rodys esamą infor-

maciją greitam jos pateikimui esamuose *Activity* objektuose ir kaip vartotojo sąsajos elementai. Taip pat kiekvieno naujo *Activity* metu turėtų būti užkraunama informacija su kuria dabar tikimasi dirbti.

Atsižvelgiant į tai, galima labai aiškiai apibrėžti kurie programėlės elementai turėtų būti *Activities* ir kurie *Fragments*.

3 lentelė. Activities ir Fragments elementai

Activities	**Fragments**
Pagrindinis *Activity* langas	Pagrindinio lango elementai: Šoninio meniu juosta, įvairūs navigacijos elementai
Bitynų pridėjimo ir redagavimo langas	Bitynų peržiūros elementai ir kortelės
Avilių pridėjimo ir redagavimo langas	Avilių peržiūros elementai ir kortelės
Apžiūrų pridėjimo ir redagavimo langas	Apžiūrų elementai ir jų sąrašo elementai

6.1. Duomenų saugojimas

Pagrindinis programėlės veikimo principas yra paremtas duomenų agregavimu, atvaizdavimu ir galimu jų manipuliavimu. Kadangi duomenys yra pamažu kaupiami, o ne dinamiškai kuriami kiekvieno paleidimo metu, labai svarbu atsižvelgti į tai kaip jie bus saugomi ir pasiekiami.

Remiantis “Android” programėlių oficialia dokumentacija [Goo18e], jie siūlo kelis išbandytus ir optimalius metodus.

Pats primityviausias metodas - *Shared preferences*. Jis yra viso labo žodyno tipo duomenų saugojimo metodas, kuris yra apdorojamas automatiškai esamų bibliotekų. Duomenys išsaugomi kaip rakto ir reikšmės poros *XML* tipo faile.

Norint išsaugoti didesnes duomenų struktūras, Java objektus ar įvarius paveikslėlius, Shared preferences yra per daug ribojantis savo galimybėmis. Tai palieka tris, gerokai abstraktesnius, apibendrintus metodus duomenų saugojimui:

- Vidinės atminties naudojimas
- Išorinės atminties naudojimas
- Duomenų bazių naudojimas

Kaip matome, egzistuoja ne vienas būdas šiai problemai išspręsti, bet atsižvelgiant į kriterijų, kad programėlė turi veikti pilnavertiškai be interneto prieigos, iš kart galima atmesti dalį metodų,

tokių kaip išorinės duomenų bazės naudojimas ar primityvūs ir pusiau automatizuoti metodai, neskirti saugoti didelėms duomenų struktūroms.

Tarp vidinės ir išorinės atminties naudojimo didžiausias skirtumas yra atminties prieinamumo rizika ir pasirinkimas kur laikyti duomenis. Kaip patys pavadinimai nusako, vidinė atmintis yra įmontuota fiziškai pačiame įrenginyje, ir jos prieinamumas (neatsižvelgiant į nenumatytus atvejus, tokius kaip išoriniai defektai) yra pastovus. Dažnai šios atminties talpa yra mažesnė nei išorinės, bet mūsų atvejų duomenys bus viso labo tekstiniai, ir pavienės iliustracijos, tad tai nėra ribojantis faktorius.

Išorinė atmintis, kaip indikuoja pats pavadinimas, yra atminties rūšis kurią galime pridėti įrenginiui papildomai. Įprastinis išmaniųjų telefonų sprendimas - atminties kortelė. Šios atminties tipui didžiausia problema yra prieinamumas. Atminties kortelė gali būti išimama ar būti pažeista atskirai nuo įrenginio, išskirtiniais atvejais net esant jame.

Nors išorinės atminties prieinamumas teoriškai yra vargiai mažesnis už vidinės, dėl jau minėtos sąlyginai mažos saugomų failų apimties ir labiau garantuotos prieigos bus naudojama vidinė telefono atmintis.

Bus naudojami išoriniai failai, laikomi vidinėje atmintyje, kurie bus serializuojami ir deserializuojami naudojantis Java ir "Android" .

Pagrindinę duomenų struktūrą matoma priede A. Visi anksčiau įvardinti elementai, išskyrus apžiūrų skiltį, yra gan paprasti ir aiškūs. Didžiausią programos dalį apimanti apžiūrų skiltis reikalauja ir daugiausia duomenų sekimo. Šiuo atveju aviliai paveldi informaciją iš bitynų ir motinėlių skilčių, o apžiūros skiltis paveldi informaciją iš avilių.

6.2. Programėlės kodas ir klasių struktūra

Turint aiškų vartotojo sąsajos ir potyrio aprašymą, aprašytas programos skiltis ir duomenų struktūras, lieka aprašyti numatomas klases ir jų metodus naudojamus programėlės veiklos metu.

Pagrindiniai ir apčiuopiamiausi metodai yra elementų kūrimas, trynimas, peržiūra ir redagavimas. Visi pagrindiniai elementai - bitynai, aviliai ir apžiūros - turi turėti šias funkcijas.

Be šių funkcijų taip pat kritiškai svarbu duomenų nuskaitymo ir įrašymo funkcijos, paremtos serializacijos principu.

Kadangi programėlė kuriama “Android” platformai, bus išnaudojama *Fragments* ir *Activities* klasės ir jų metodai. Kadangi *Activities* yra tolygu daugumos programavimo paradigmų naudojamą *main()* funkciją iššaukti pagrindinėms kodo dalims, kodo dalys atitinkančios šią struktūra bus pavadintos pagal *PavadinimasActivity* konvenciją.

Pagrindinės numatomos *Activities* dalys yra:

- Pagrindinis programos langas, kuriame bus atvaizduojama visa esanti informacija:

  Bitynų, avilių ir apžiūrų sąrašų atvaizdavimas

- Duomenų įvedimo ir redagavimo langai (bus daugkartinai panaudotas vienas *Activity* kelioms funkcijoms):

  Bitynų, avilių ir apžiūrų redagavimo ir duomenų įvedimo atvaizdavimas

  Nustatymų langas

Atsižvelgiant į tai, kad *Fragments* yra vartotojo sąsajos elementai, patalpinti į Actvities klases, bus kuriamos atitinkamai šios dalys ir *Fragments* klasėms bus suteiktas pavadinimas pagal konvenciją *PavadinimasFragment*:

- Šoninė meniu juosta (*angl. Navigation Drawer*)
- Bitynų ir avilių kortelės (*angl. Cards*)
- Apžiūrų sąrašo pateikimas
- Pagrindinės lango funkcijos mygtukas (*angl. Floating Action Button*)

Priede B matoma pagrindinių klasių diagrama. Ją projektuojant buvo atsižvelgta į Single Responsibility Principle - vienos atsakomybės principą, kuris diktuoja, kad viena programos dalis ar klasė turi būti atsakinga už vieną funkciją programoje, ir, kad ši atsakomybė būtų visiškai apimta šios klasės [Mar03].

Pagrindinių funkcijų klasės (bitynai, aviliai, apžiūros) turi savo *Activity* klases, kurios turi pagrindines pridėjimo ir redagavimo funkcijas. Atitinkamai kiekvienam pagrindiniam programos elementui bus naudojama ta pati informacija, tiek pridėjimo, tiek ir redagavimo atveju, dėl to pakartotinai naudojamas tas pats *Activity*.

Pagrindinio lango, klasių diagramoje pavadinto *PagrindinisLangas*, paskirtis yra esamų duomenų atvaizdavimas. Jis naudos *Fragment* klases skirtingų elementų atvaizdavimui, kurios yra šoninės menių juosta ir pagrindinio lango veiksmo mygtukas.

Taip pat, programėlės išbaigtumui, yra aprašomas ir nustatymų *Fragment* klasė.

Duomenų šalinimo ir saugojimo funkcijos yra implementuotos pačiuose už juos atsakingų fragmentų klasėse.

Programėlės įgyvendinimo langai matomi priede C ir D.

Išvados ir rekomendacijos

Iš atliktos rinkos apžvalgos matyti, kad dauguma programėlių pasižymi vienodomis bėdomis, o pačios ryškiausios yra vartotojo potyrio ir vartotojo sąsajos elementų nepaisymas. Neatsižvelgiant į šiuos du dizaino elementus, smarkiai kenčia programėlės kokybė, nepriklausomai nuo jos funkcionalumo.

Turint šią informaciją, buvo mėginta konstruktyviai ir detaliai į tai atsižvelgti ir suprojektuoti programėlę kuri neturėtų šių bėdų. Buvo skiriama daug dėmesio vartotojo potyriui ir sklandžiam programėlės tėkmės (perėjimui iš vienos programėlės dalies į kitą) užtikrinimui. Tai buvo mėginta pasiekti atsižvelgiant į paprastus dizaino elementus, tokius kaip teksto ir aplinkos spalvos parinkimas, vengimas naudoti neapibrėžtas ikonas, ir palikti kuo mažiau informacijos vartotojui interpretuoti, ar aiškintis kaip veikia programėlė spėliojimo būdu.

Tolimesnės programėlės plėtojimo galimybės yra gan didelės. Pradedant nuo vartotojo potyrio pagerinimo - programėlės spalvos schemos pasirinkimas, pamažu tampa jau ginčytinai standartiškas dalykas, su vis daugiau programėlių palaikančių šią funkciją. Tamsi ar šviesi spalvų schemos viso labo dauguma atvejų yra asmeninio skonio reikalas, bet jei tai suteikia vartotojui didesnį pasitenkinimą ją naudojant, tai yra gan smarkus argumentas tai implementuoti. Taip pat papildomų kalbų vertimas. Kadangi jau yra numatytas daugiakalbiškumo palaikymas, naudojant strings.xml failą, tai yra lengvai įgyvendinamas dalykas pagal kalbos poreikius.

Tęsiant programėlės pagerinimą, failų saugojimo metodas išoriniame servise, kaip pavyzdžiui debesų technologijos, ar asmeninio failų laikymo servisuose, kaip “DropBox”, panaudojimas suteiktu programėlei didesnį prieinamumą ir užtikrintu duomenų saugumą - praradus ar nepataisomai sugadinus įrenginį, duomenys vistiek liktu pasiekiami.

Funkcionalumo prasme, labai lengva išsiplėsti perkraunant programėlę nereikalinga informacija. Bet atsižvelgiant, kad programėlės idėja yra sukurti ją kuo paprastesnę vartotojui ir sukurti kuo didesnę tėkmę programėlės naudojimo eigai, galima įgyvendinti tokias funkcijas kaip tolimesnių veiksmų siūlymas, atsižvelgiant į surinktus duomenis ar oro sąlygas, bei simptominis požymių vedimas, kuris pasako dabartinę avilio būklę ar gydymo metodus.

Tai būtų pagrindiniai tolimesni siūlymai šiai programėlei, stengiantis jos dirbtinai neišpūsti ir nesukurti informacijos pertekliaus, atbaidančio galimus vartotojus.

Literatūra

[BKC17] BKCorner. Hive inspection form, 2017.
http://www.bkcorner.org/?p=1542.

[Fou18] The Eclipse Foundation, 2018.
https://eclipse.org/downloads/.

[Goo17a] Google. Material design, 2017.
https://material.io/.

[Goo17b] Google. Material guidelines, 2017.
https://material.io/guidelines/.

[Goo17c] Google. The o cial ide for android, 2017.
https://developer.android.com/studio/index.html.

[Goo18a] Google. Creating lists and cards, 2018.
https://developer.android.com/training/material/lists-cards.html.

[Goo18b] Google. Fragments, 2018.
https://developer.android.com/guide/components/fragments.html.

[Goo18c] Google. Introduction to activities, 2018.
https://developer.android.com/guide/components/activities/intro-activities.html.

[Goo18d] Google. Storage options, 2018.
http://developer.android.com/training/basics/supporting-devices/languages.html.

[Goo18e] Google. Storage options, 2018.
https://developer.android.com/guide/topics/data/data-storage.html.

[Gre14] GreenPeace. The bees in decline, 2014.
http://sos-bees.org.

[Int15] Interakcijos.lt. "user experience" lietuviškai, 2015.
http://www.interakcijos.lt/user-experience-lietuviskai/.

[Ior17] Bogdan Iordache. Apiary book, 2017.
https://play.google.com/store/apps/details?id=com.csg.apiarybook&hl=en.

[Kri33a] Jonas Kriščiūnas. *Bitininkystė*. 1933.

[Kri33b] Jonas Kriščiūnas. *Bitininkystė*. 1933.

[Lab16] RainMaker Labs. 8 reasons your app gets uninstalled, 2016. http://rainmaker-labs.com/reasons-your-app-gets-uninstalled.

[Mar03] Robert C. Martin. *Agile Software Development, Principles, Patterns, and Practices*. 2003.

[MC18] Ryan Reith Kathy Nagamine Melissa Chau, Navina Govindaraj. Smartphone os market share, 2017 q1, 2018. https://www.idc.com/promo/smartphone-market-share/os.

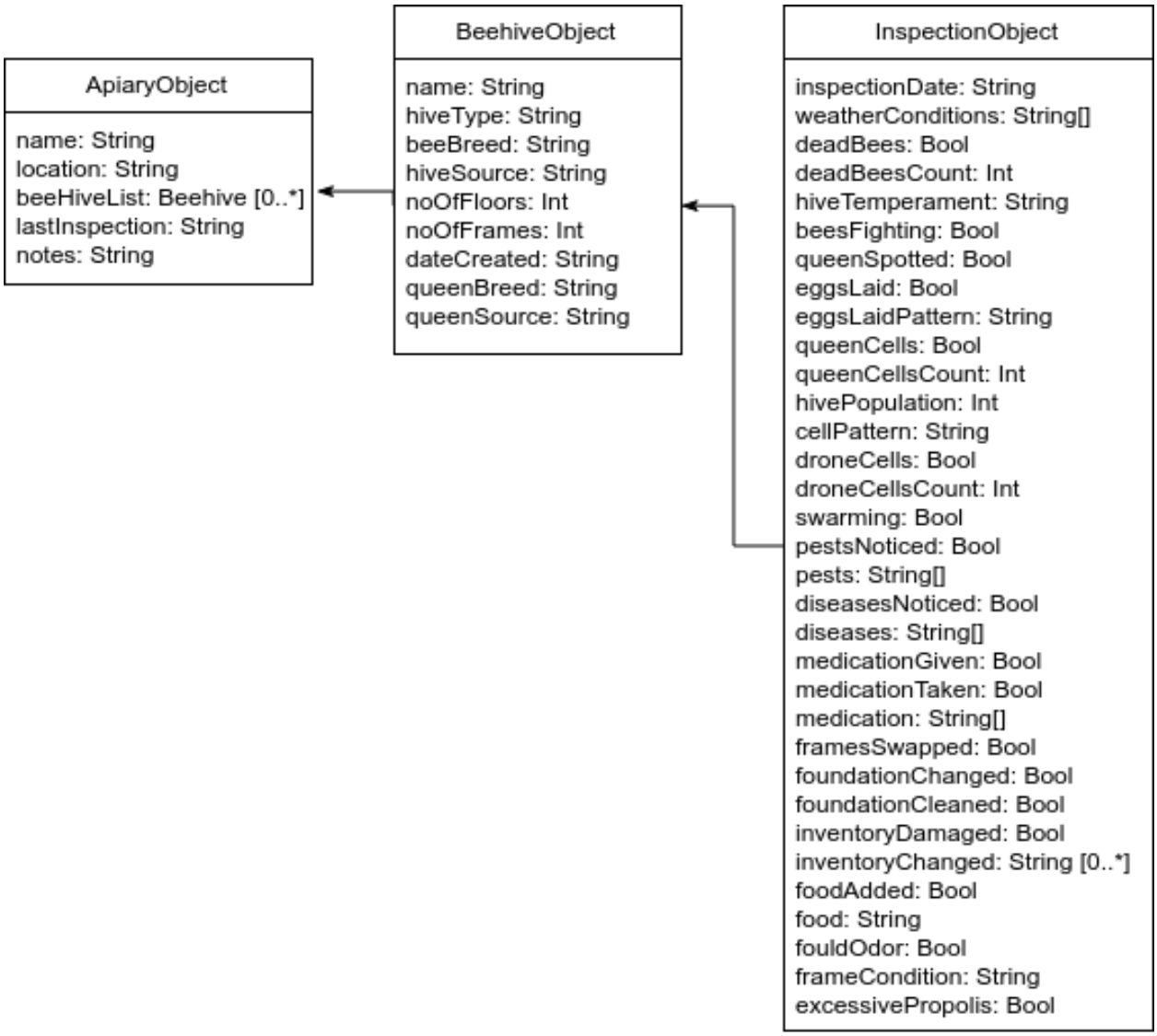
[Son17] Dadant & Sons, 2017. https://www.dadant.com.

[uni18] Florida university, 2018. http://osceola.ifas.ufl.edu/AG/CSA/CommunitySustainedAgriculture_Beekeeping.shtml.

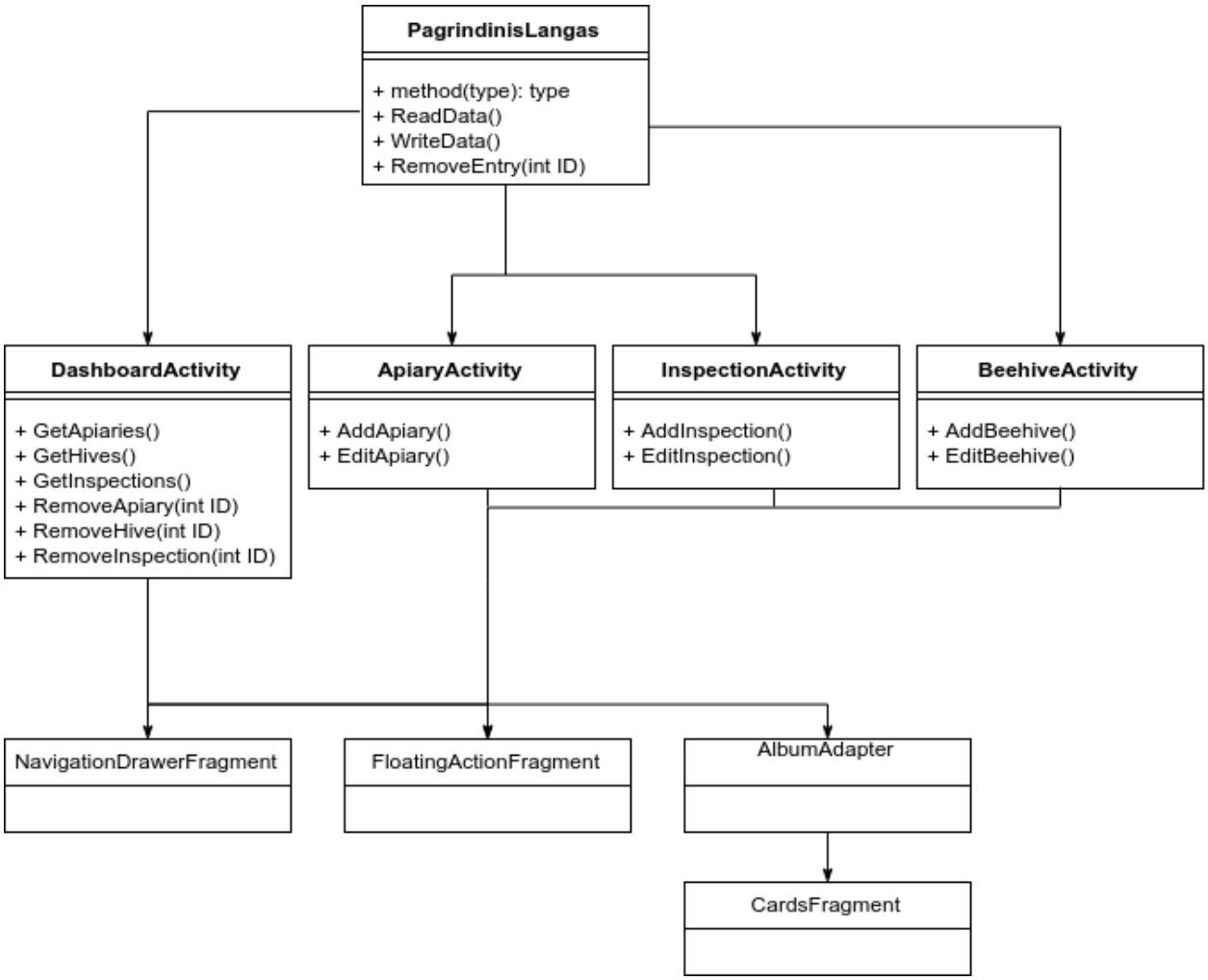
[Var15] Rahul Varshneya. 7 reasons why users delete your mobile app, 2015. https://www.inc.com/rahul-varshneya/7-reasons-why-users-delete-your-mobile-app.html.

Priedai

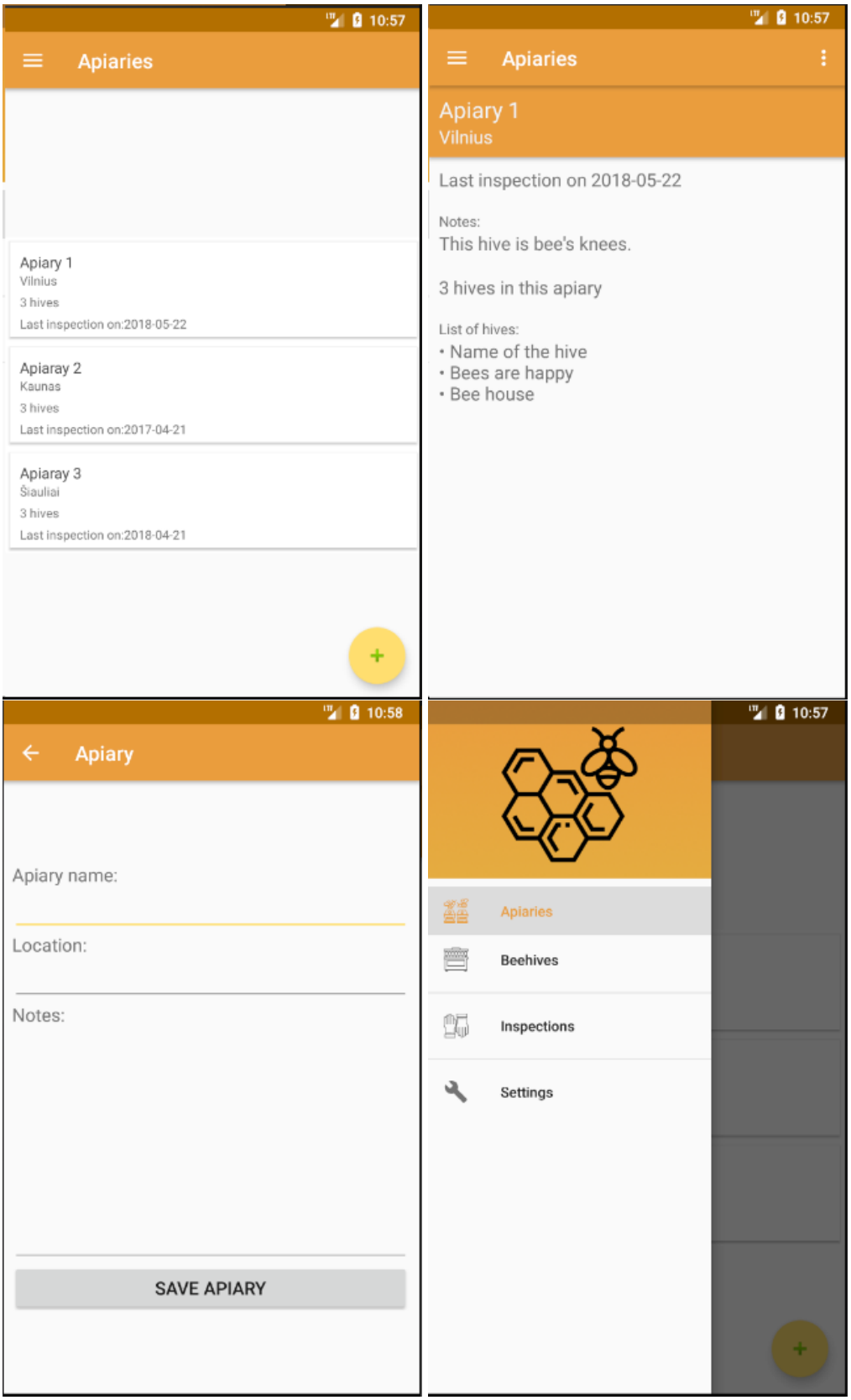
A. Duomenų struktūros diagrama



B. Programos klasių diagrama



C. Programėlės langų prototipai



D. Programėlės langų prototipai

