VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Bakalauro baigiamasis darbas

Žvejo informacinė sistema

Fisherman’s Information System

Atliko : 4 kurso, 2 grupės studentas
Martynas Gelumbauskas

Darbo vadovas :
dr. Joana Katina

Vilnius
2020

Turinys

Anotacija

Darbo tikslas – sukurti nemokamą žvejams skirtą Android mobiliąją programėlę pavadinimu „Kablys“. Tikslo įgyvendinimui buvo atlikta panašių sistemų analizė ir išsikelti kriterijai, pagal kuriuos buvo suformuluoti sistemos uždaviniai, sudarytas ir aprašytas teorinis sistemos modelis. Pasiektas darbo rezultatas yra sukurta mobilioji programėlė, turinti galimybę žemėlapyje pažymėti pagautas žuvis, galimybę peržiūrėti vandens telkinyje gyvenančias žuvis (naudotojo įvestas), galimybę peržiūrėti ir įvesti žuvų informaciją (svoris, aprašymas) bei jų nuotraukas, galimybę žvejui pažymėti įsigytus žvejybos leidimus ir gauti pranešimus apie jų pabaigą [4], galimybę žvejui patikrinti ar jo pagautas laimikis nepažeidžia žvejybos taisyklių, galimybę pasidalinti laimikiu naudojant socialinius tinklus, vaizdo pamokų peržiūrą, orų prognozę, žuvų kibimo prognozę, kibimo kalendorių, informacinių pranešimų gavimą kai žvejys artėja prie telkinio, kuriame negalima žvejoti ir dienos iššūkio pranešimų gavimą.

Summary

Fisherman’s Information System

The purpose of the work is to create a free Android application meant for fishermen that is called “Kablys”. To achieve the goal similar system analysis and evolution criteria has been carried out, according to which the tasks were formed, the theoretical model of the system was developed and described. The result of the work is the creation of a mobile application with the following functions: tag the captured fish on the map, tag types of fishes that live in there, enter information about them (weight, description), upload a photo. Also, the app would allow you to mark acquired licenses and remind you later that they will expire soon. The app would also have such background functionality: notify when a fisherman approaches a non-fishing area. Further, the app would allow the fisherman to verify if the catch was in compliance with the rules. The app would provide video lessons, show the weather and give a prediction of the fish bite according to the weather and the bite criterias. Also, the app would have a catch calendar that would show in which months what fish would bite and what bait or gear to use. In addition, the app would provide information about the fish themselves and allow fisherman to share their catch through social media.

In conclusion, a mobile app for fishermen has been developed and described and the main goal of creating an app for a fisherman, which would help him with fishing problems, was carried out successfully. The information stored in the app provides user with useful information about fish and subtleties of catching them. During the work similar apps were analyzed, ideas were taken, bad examples were noticed and during the development of the system, database, use-case and activity diagrams were drawn, requirements, functions, algorithms, technologies were described.

Įvadas

Pasaulyje beveik kiekvienas žmogus turi išmanųjį telefoną, todėl didelė tikimybė, kad net eilinis žvejys turės mobilųjį telefoną su 4G internetu. Būtent todėl nuspręsta kurti mobiliąją programėlę, nes tai yra patogu ir prieinama beveik visiems. Žmonės žvejodami turi turėti galimybę greitai ir patogiai pasiekti visą jiems reikiamą ir svarbią informaciją. Visgi, mobilioji programėlė yra daug lengviau pasiekiama ir patogesnė, kadangi internetinė sistema neturėtų lengvo ir interaktyvaus dizaino, juo labiau ji nebūtų tobulai pritaikyta mobiliesiems įrenginiams, kadangi jie yra įvairaus dydžio ekranų ir nevienodo greičio. Kai kurie yra labai lėti, todėl užkrauti internetinę sistemą užtruktų nemažai laiko.

Darbo tikslas – sukurti programėlę, kuri suteiktų žvejui naudingos informacijos ir duotų patarimų, kurie jam suteikti informacijos apie žuvis ir kibimo tendencijas.

Pagrindiniai darbo uždaviniai:

- Išanalizuoti panašias sistemas.
- Sudaryti ir aprašyti teorinį sistemos modelį.
- Aprašyti duomenų bazę ir funkcijas, kurios sąveikautų su ja.
- Sukurti naudotojo autentifikavimo ir jo informacijos pakeitimo funkcionalumą.
- Sukurti algoritmą, kuris paimtų orų duomenis iš interneto, juos atvaizduotų sistemoje ir pateiktų kibimo prognozę.
- Sukurti vaizdo pamokas naudojant *YouTube Android Player API* (API – Application Programming Interface, lietuviškai – programų programavimo sąsaja).
- Integruoti žemėlapį panaudojant *Google Maps API*.
- Sukurti algoritmą, kuris suteiktų informaciją ar pažeidžiamos žvejojimo taisyklės [5].
- Sukurti naudingos informacijos langus.
- Sukurti foninį procesą, kuris siųstų žvejui priminimus ir smagius dienos iššūkius.
- Sukurti pranešimų kanalą, leidžianti programėlei pateikti pranešimus kai programėlė yra įjungta ar net išjungta.

Programėlės kūrimui buvo panaudota Android Studio programavimo aplinka kartu su JAVA programavimo kalba. Duomenys saugomi *SQLite* duomenų bazėje. Naudojamos Google paslaugos ir API raktas, bei *AccuWeather API.*

1. Esamų panašių sistemų analizė

Prieš pradedant programuoti buvo atliktas panašių programėlių tyrimas, kad pamatyti kokias funkcijas naudoja kiti kūrėjai, kokie tų programėlių privalumai ir trūkumai. Gauti rezultatai leidžia spręsti ar ši programėlė atsiliktų nuo kitų rinkos žaidėjų ar ne.

1.1 Sistemų palyginimas

Pirmoji išanalizuota programėlė yra „Fishing and Hunting Solunar Time“. Programėlė rodo dienos reitingą, kuris neturi paaiškinimo kokiu būdu buvo apskaičiuotas ir kodėl išreikštas procentais. Norint pridėti objektą žemėlapyje programėlė reikalauja nusipirkti mokamą versiją. Su žuvimis susijusių funkcijų nėra (patarimai, informacija apie žuvis...). Programėlė neturi naudotojų autentifikacijos.

Antroji išanalizuota programėlė yra „Fishing forecast“. Šios programėlės funkcionalumas yra daug platesnis nei prieš tai analizuotos. Nors žuvų kibimas nėra labai suprantamai atvaizduojamas, bet jis vis dėl to atvaizduojamas, net gi atskirai kiekvienai žuviai. Programėlė turi diagramas, tokias kaip temperatūros pokyčiai, slėgio pokyčiai, bet žinoma paprastam žmogui tai gali būti nesuprantami parametrai. Programėlė taip pat leidžia pažymėti objektus žemėlapyje ir turi literatūrą apie žuvis, bet tekstas apie žuvis yra rusų kalba. Programėlė neturi naudotojų autentifikacijos. Yra nemokama, tačiau yra rodomos reklamos.

Trečioji programėlė yra „Fishbrain“. Ši programėlė turi prisijungimą ir registraciją, integruotą žvejų socialinį ratą, leidžia fiksuoti laimikius, jais dalintis tarp kitų žvejų, todėl programėlė labiau veikia kaip socialinis tinklas. Visgi kibimo parametrai yra pasirinkus „BiteTime“, bet rodo tik vienos žuvies statistiką, kita žuvis reikalauja mokamos paskyros. Programėlė taip pat veikia kaip parduotuvė, nes galima nusipirkti marškinėlius ir kitą atributiką.

Išanalizavus programėles akivaizdu, kad mažas funkcijų asortimentas yra silpnoji dalis, todėl palyginus šias programėles matyti, kad jos nėra tobulos (1 lentelė).

1 lentelė. Panašių programėlių palyginimas

Programėlė	**Registracija**	**Žemėlapis**	**Kibimo prognozė ir orai**	**Informacija apie draudimus**	**Licencijos priminimai**
„Fishing and Hunting Solunar Time“	Nėra	Yra, bet funkcionalumas mokamas	Yra, bet ne detalus	Nėra	Nėra
„Fishing forecast“	Nėra	Yra	Yra	Nėra	Nėra
„Fishbrain“	Yra	Yra	Yra, bet reikia mokėti	Nėra	Nėra
„Kablys“	Yra	Yra	Yra	Yra	Yra

Apibendrinus visas apžvelgtas programėles buvo pastebėta, kad sistemos naudojimas be registracijos ir prisijungimo padaro sistemą paprastesnę, tačiau tai neduoda galimybės saugoti naudotojo duomenų, leisti naudotojui prisijungti ir matyti savo duomenis kituose mobiliuosiuose įrenginiuose. Be to, tai neleidžia personalizuoti duomenų ir pritaikyti funkcijas būtent tam naudotojui. Nėra tokių funkcijų kaip: patikrinti ar pagautą žuvį galima neštis, negalima įsivesti licencijos pabaigos priminimų. Taip pat, šios programėlės pagrindinį funkcionalumą turi paslėptą po mokesčiais.

2. Sistemos architektūra

Sistemos architektūra susideda iš kelių dalių: sistemos reikalavimų, veiklos schemų, panaudos atvejų schemų, technologijų ir programavimo aplinkos pasirinkimo, duomenų bazės modeliavimo, naudotojo sąsajos modeliavimo. Visos šios dalys svarbios norint aiškiai suprasti programos reikalavimus ir tinkamai spręsti iškylančias problemas.

2.1 Sistemos reikalavimai

Šiame skyriuje yra pateikiami sistemos funkciniai ir nefunkciniai reikalavimai, kurie padeda tinkamai įgyvendintini tikslus.

Funkciniai reikalavimai:

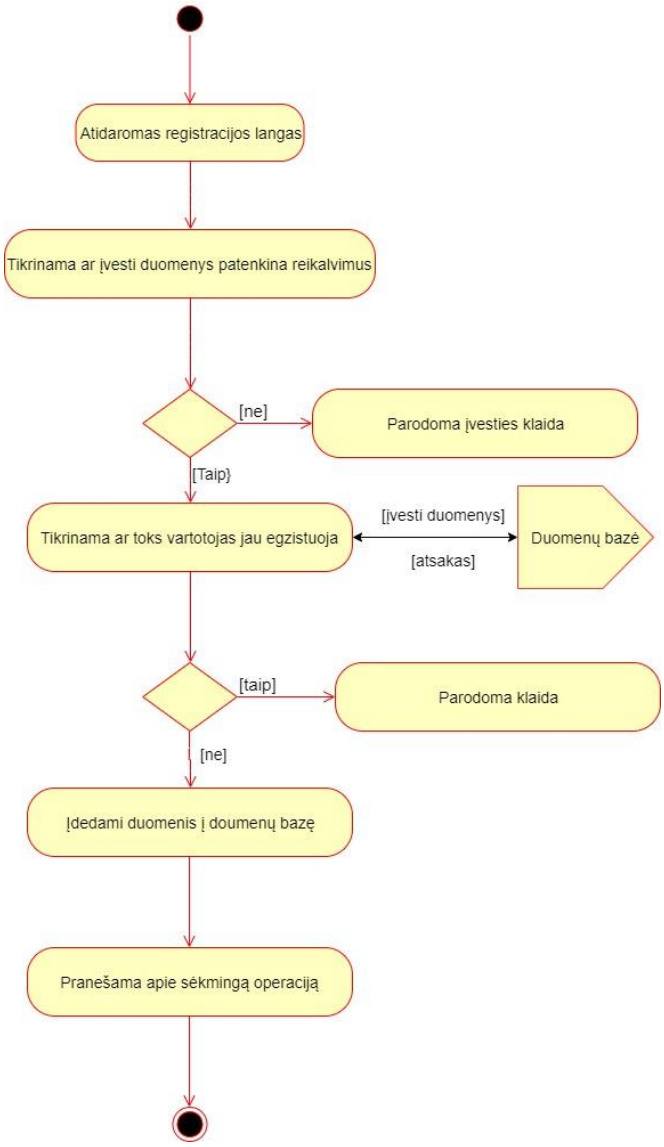
1. Prisijungimo ir registracijos langai.
2. Įvestų duomenų saugojimas duomenų bazėje.
3. Pradinių duomenų įvedimas į duomenų bazę.
4. Foninis procesas atsakingas už datų palyginimus ir priminimų siuntimą naudotojui.
5. Orų duomenų gavimas pasitelkus *AccuWeather API* [1] .
6. Oro slėgio duomenų ir bendros kibimo lentelės panaudojimas prognozuojant kibimą [7, 13].
7. Naudotojo profilio redagavimas ir ištrynimas.
8. Žemėlapio atvaizdavimas ir pagautų žuvų žymėjimas bei saugojimas duomenų bazėje bei vėlesnis atvaizdavimas.
9. Pagautos žuvies patikrinimo funkcija bandant nustatyti ar pagavimas yra legalus.
10. Vaizdo pamokų paleidimo funkcija.
11. Langas informacijai apie žuvis [12].
12. Langas bendrai kibimo lentelei atvaizduoti su galimybe naudotojui pasirinkti mėnesį ir žuvį.

Nefunkciniai reikalavimai:

1. Aiškaus šrifto parinkimas, kad tekstas būtu įskaitomas.
2. Programos dizaino kūrimas pagal modernias tendencijas, naudojant švelnias spalvas ir patogų elementų išdėstymą.

2.2 Veiklos schema

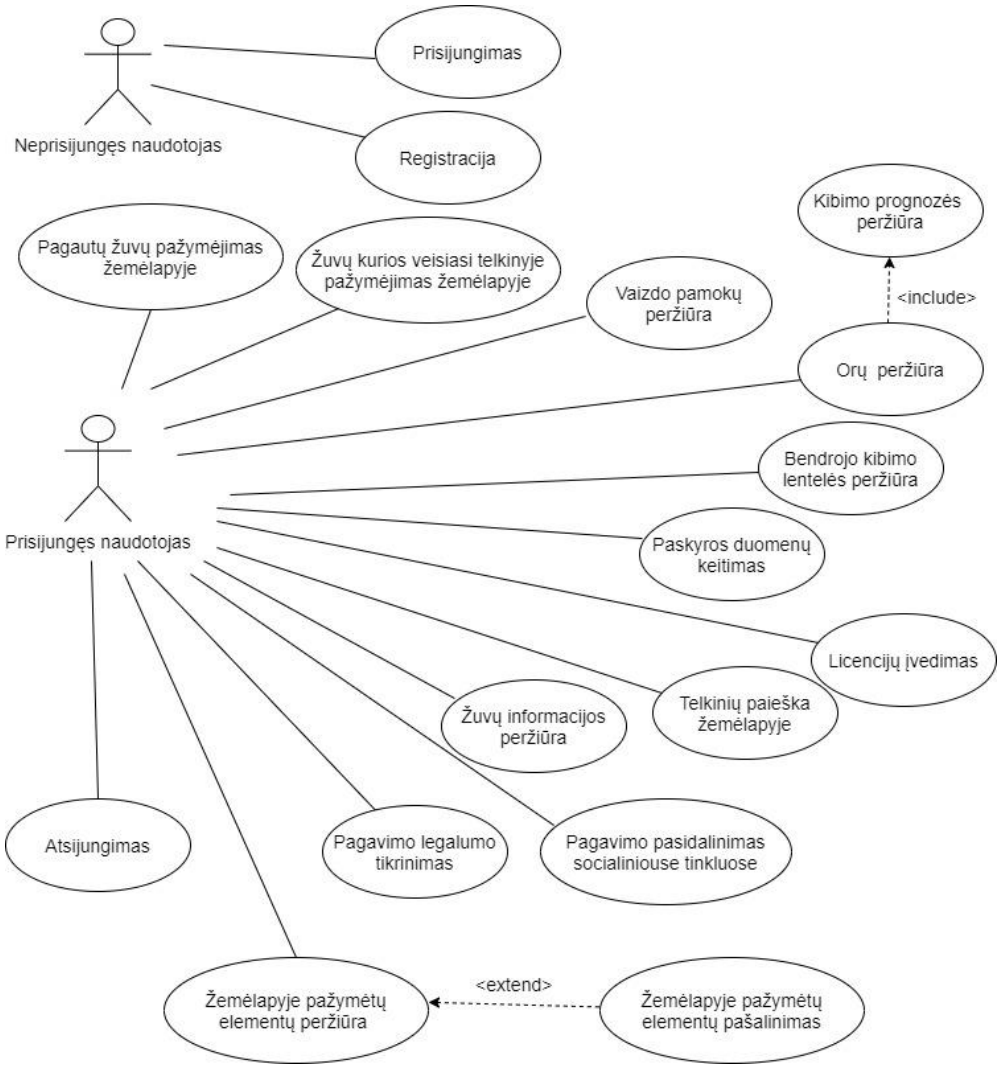
Viena iš svarbiausių veiklų yra registracija ir prisijungimas, kadangi tai leidžia sukurti įrašą apie naudotoją ir vėliau saugoti naudotojo duomenis. Veiklos schema yra labai svarbi kuriant sistemą, kadangi ji atvaizduoja sistemos elgseną. Visgi, svarbu žinoti kaip sistema turi elgtis. Registracijos veiklos schema pateikiama 1 pav. Visu pirma atidaromas registracijos langas, kuriame naudotojas įveda duomenis, yra tikrinama ar slaptažodis nėra per trumpas, ar įvestas legalus elektroninio pašto adresas ir ar slaptažodžiai sutampa. Jeigu kažkas yra blogai – parodoma klaida, jeigu ne – toliau tikrinama ar toks vartotojas jau egzistuoja. Duomenų bazei yra duodama užklausa patikrinti ar tokie duomenys jau egzistuoja, jei egzistuoja, funkcija grąžina atsaką ir naudotojui yra parodoma klaida. Jeigu viskas yra gerai, duomenys yra patalpinami į duomenų bazę ir naudotojui parodoma, kad operacija yra sėkminga.



1 pav. Registracijos veiklos schema

2.3 Sistemos panaudos schema

Sistemoje yra tiktai vienas veikėjas – tai yra pats naudotojas. Naudotojas gali būti dviejų tipų: prisijungęs ir neprisijungęs. Schema pavaizduoja (2 pav.), kad neprisijungęs naudotojas turi tiktai dvi funkcijas (prisiregistruoti arba prisijungti), o prisijungęs vartotojas turi prieigą prie visų funkcijų: atsijungimas, orų ir kibimo peržiūra, licencijų įvedimas ir kt. Funkcija „Žemėlapyje pažymėtų elementų pašalinimas“ yra rezultatas kitos funkcijos „Žemėlapyje pažymėtų elementų peržiūra“, kadangi, norint elementą pašalinti, jį pirmiausia reikia peržiūrėti.



2 pav. Sistemos panaudos schema

2.5 Programavimo aplinka

Šiam darbui įgyvendinti buvo pasirinkta Android Studio programavimo aplinką [3]. Tai yra oficiali Google operacinės sistemos integruota kūrimo aplinka (angl. IDE), sukurta pagal JetBrains IntelliJ IDEA programinę įrangą ir sukurta specialiai Android plėtrai. Android Studio pakeitė Eclipse programavimo aplinką. Android Studio yra naudojama tokių kompanijų kaip Google, Lyft, Square. Android Studio palaiko visas tas pačias Intelli (ir CLion) programavimo kalbas, pvz., Java, C ++ ir dar daugiau su plėtiniais, tokiais kaip Go. Android Studio 3.0 ar naujesnė versija palaiko Kotlin ir visas Java 7 kalbos funkcijas bei Java 8 kalbos poaibį, kuris skiriasi priklausomai nuo platformos versijos.

Android Studio privalumai yra tokie:

- Android Wear programų kūrimo palaikymas.
- Integruotas Google debesų platformos palaikymas.
- ProGuard integracijos ir programų pasirašymo galimybės.
- Lint įrankiai našumui, patogumui, versijų suderinamumui ir kitoms problemoms surasti.
- Šablonų vedliai, skirti sukurti bendrą Android dizainą ir komponentus.
- Android virtualus įrenginys (angl. *emulator*), jei norima paleisti ir derinti programas Android studijoje.
- Turtingo išdėstymo rengyklė, leidžianti naudotojams nuvilkti sąsajos komponentus, galimybė peržiūrėti maketus keliose ekrano konfigūracijose.

2.6 Pasirinktų technologijų analizė

Nors ši programėlė nenaudoja daug technologijų yra svarbu suvokti ką tos technologijos daro ir kaip veikia ir kodėl jos buvo pasirinktos.

2.6.1 JAVA

Android Studio palaiko JAVA programavimo kalbą, o pati programavimo kalba yra labai populiari ir daug kur naudojama, tai padaro šią kalbą puikiu pasirinkimu pradedantiesiems. JAVA [9] yra bendrosios paskirties, objektiškai orientuota programavimo kalba, kurią 1991 m. sukūrė Džeimsas Goslingas ir kiti „Sun Microsystems“ inžinieriai, o 2010 m. ją įsigijo Oracle korporacija. Android Studio aplinkoje JAVA klasės yra atsakingos už programos funkcionalumą ir atvaizdavimą. Pavyzdžiui, funkcija *OnCreate()* yra iškarto paleidžiama iškvietus klasę ir ji nustato lango dizainą ir elementus naudojant *SetContentView().* Dizaino elementai yra projektuojami XML formatu. Toliau

galima aprašyti XML failo elementus klasėje ir jiems nustatyti klausytojus (angl. *listeners*) bei gauti tų elementų reikšmes ir kt.

2.6.2 SQLite

SQLite yra reliacinė duomenų bazių valdymo sistema [10]. SQLite yra C kalba parašyta biblioteka, sukurianti duomenų bazę faile. Duomenys valdomi naudojant SQL (realizuotas SQL 92 standartas) kalbą. C biblioteką parašė D. Richard Hipp. SQLite nėra duomenų bazių serveris. Biblioteką galima naudoti iš daugelio populiariausių programavimo kalbų. SQLite yra tinkamas pasirinkimas, nes tai yra saugiausias ir patikimiausias būdas saugoti duomenis mobiliajame įrenginyje, kadangi nereikia interneto ryšio ir visa duomenų bazė yra laikoma pačiame įrenginyje (DB failas), taip yra išvengiama netikėtumų.

2.6.3 Android palaikymo biblioteka

Kuriant programą Android 9.x, versijai, kad ji veiktų tuose įrenginiuose, kuriuose veikia senesnės Android versijos, to negalima padaryti, kol prie kodo nėra pridedamas atgalinis suderinamumas. Norėdami pateikti šį atgalinį suderinamumą, Android Studio suteikia Android palaikymo bibliotekos paketą (angl. Android Support Library) [2]. Android palaikymo bibliotekos paketas yra kodų bibliotekų rinkinys, kuriame pateikiamos atgalinės Android sistemos API versijos, taip pat funkcijos. Kiekviena palaikymo biblioteka yra suderinama atgal su konkrečiu Android API lygiu.

2.6.4 Google Maps API

Naudojant Android skirtą Maps SDK [6], galima pridėti žemėlapius, pagrįstus Google Maps duomenimis. API automatiškai tvarko prieigą prie Google Maps serverių, duomenų atsisiuntimą, žemėlapio rodymą ir atsakymą į žemėlapio gestus. Taip pat galima naudoti API skambučius (angl. *callbacks*), kad pridėti žymeklius, daugiakampius ir perdangas prie pagrindinio žemėlapio ir pakeisti naudotojo vaizdą apie tam tikrą žemėlapio sritį. Šie objektai suteikia papildomos informacijos apie žemėlapio vietas ir leidžia naudotojui sąveikauti su žemėlapiu. API leidžia prie žemėlapio pridėti šias grafikas:

- Piktogramos, pritvirtintos prie konkrečių vietų žemėlapyje (žymekliai).
- Linijų segmentų rinkiniai (polilinijos).
- Uždari segmentai (daugiakampiai).
- Bitmap grafika, pritvirtinta prie konkrečių vietų žemėlapyje.
- Vaizdų rinkiniai, rodomi ant pagrindinio žemėlapio plytelių (plytelių perdangos).

Apibendrinus, akivaizdu, kad ši technologija yra svarbi ir naudinga, kadangi tai suteikia programėlei patikimą ir tvarkingą žemėlapių paslaugą. O ir faktas, kad technologija leidžia pridėti žymeklius, reguliuoti priretinimą ir kameros poziciją leidžia efektyviai įgyvendinti funkcionalumą.

2.6.5 Youtube Android Player API

YouTube Android Player API [11] suteikia galimybę integruoti vaizdo įrašų atkūrimo funkciją. API apibrėžia YouTube vaizdo įrašų (ir grojaraščių) užkrovimo ir atkūrimo patirties tinkinimo ir valdymo metodus. Taip pat galima užregistruoti įvykių klausytojus, pavyzdžiui, grotuvo vaizdo įrašo užkrovimo ar būsenos pakeitimo. Galiausiai API turi pagalbininko funkciją, kuri palaiko orientacijos pokyčius, taip pat perėjimus į viso ekrano paleidimą. API leidžia kontroliuoti vaizdo įrašų grojimą programiškai, pavyzdžiui, galima atkurti, pristabdyti arba ieškoti konkretaus šiuo metu grojamo vaizdo įrašo taško. API kliento biblioteka sąveikauja su paslauga, kuri platinama kaip YouTube programos dalis, todėl biblioteka turi mažą įtaką programėlės dydžiui.

3. Funkcionalumo įgyvendinimas

Sistema susideda iš nemažai funkcijų, todėl šiame skyriuje aprašytos kelios svarbesnės iš jų: duomenų bazės struktūra, orų informacijos apdorojimas, ribojimų tikrinimas, prisijungimas ir registracijas, naudotojo nustatymai, foninis procesas, naudotojo sąsajos funkcionalumas.

3.1 Duomenų bazė

Duomenų saugojimui buvo panaudota SQLite duomenų bazė. Kadangi ši duomenų bazė yra paprasta ir laikoma telefono atmintyje, ją yra lengva integruoti į sistemą, ypač turint omenyje, kad Android Studio palaiko ją tiesiogiai.

Duomenų bazė susideda iš šių lentelių:

- *Users* lentelėje yra saugomi baziniai naudotojo duomenys: ID (yra padidinamas kiekvieną kartą įdėjus įrašą), Username – slapyvardis, Email – elektroninis paštas, Password – slaptažodis. Slapyvardis yra naudojamas daugelyje funkcijų, norint nustatyti naudotojo tapatybę.

- *Locations* lentelėje yra laikoma informacija apie naudotojo pagautas žuvis, kai jis jas pažymi žemėlapyje. Lentelėje yra saugomi šie duomenys: platuma ir žemuma (naudojama norint pažymėti žymeklius žemėlapyje), žuvies pavadinimas, svoris, aprašymas ir nuotrauka.

- *Permits* yra lentelė, kurioje yra saugomi naudotojo įvesti leidimai. Lentelę sudaro: naudotojo slapyvardis (kad atpažinti kuriam naudotojui priklauso leidimas), leidimo galiojimo pradžios ir pabaigos data (tam, kad vėliau tikrinti ar pasibaigė galiojimas), papildomos pastabos, tekstas pažymėjai ar naudotojas buvo įspėtas (kadangi nenorima, kad naudotojui nuolatos būtų pateikiamas pranešimas).

- *ForbiddenPlaces* yra lentelė, kurioje yra saugomos vietos, kur draudžiama žvejoti. Lentelė saugo šią informaciją: ilguma ir platuma (kad patikrinti naudotojo atstumą iki tos vietos), vietovės pavadinimas.

- *Fishes* yra lentelė, kurioje saugoma informacija apie visas žuvis: žuvies pavadinimas, nuotrauka, aprašymas, rekomenduojamas masalas, patarimai.

- *FishesinPond* yra lentelė, kurioje saugomos naudotojo įvestos žuvys, t. y. kokios žuvys veisiasi vandens telkinyje. Tai yra tokie duomenys: ilguma, platuma, slapyvardis (kad nustatyti, kuris naudotojas pažymėjo ją), žuvys.

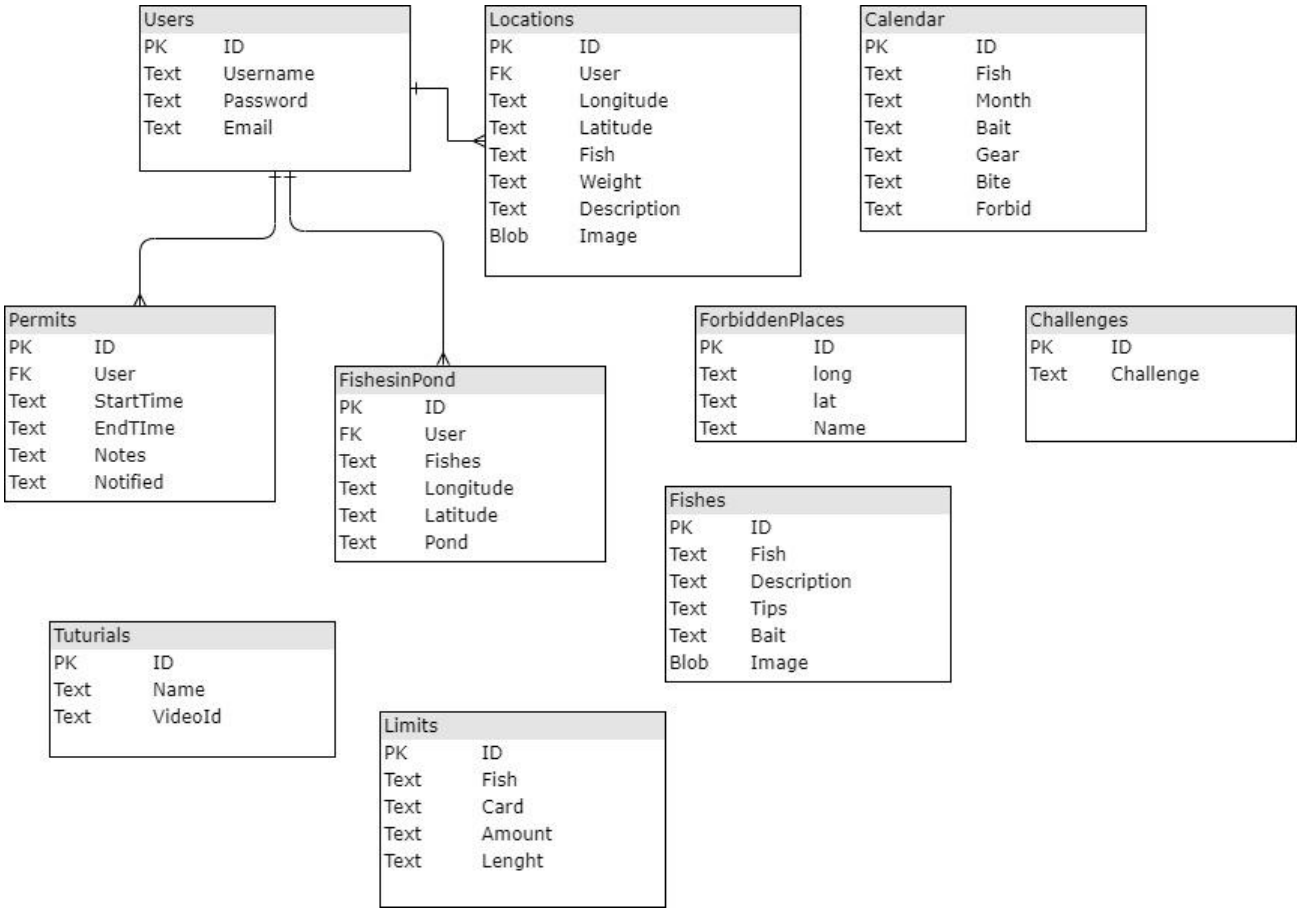
- *Calendar* yra lentelė, kurioje saugoma kalendorinė informacija apie kibimą: žuvies pavadinimas, mėnesis, kibimas (geras, vidutinis, prastas), rekomenduojama žvejojimo įranga tai žuviai tą mėnesį, draudimai.

- *Challenges* yra lentelė, kurioje saugomi dienos iššūkiai.

- *Tutorials* yra lentelė, kurioje saugoma informacija apie vaizdo pamokas: vaizdo įrašo pavadinimas, Youtube vaizdo įrašo ID.

- *Limits* yra lentelė, kurioje saugoma informacija apie draudimus: žuvies pavadinimas (yra apribojimai specifinėms žuvims), rodiklis, kuris pasako ar žuviai reikia leidimo kortelės, maksimalus leistinas kiekis ir minimalus žuvies ilgis.

3 pav. pavaizduoja duomenų bazės schemą. *Users* lentelė turi ryšį vienas – daug su *Locations*, *FishesinPond* ir *Permit* lentelėmis, kadangi vienas naudotojas gali turėti daug įrašų tiek apie žuvis, tiek apie leidimus.



3 pav. Duomenų bazės schema

3.2 Prisijungimas

Visoms duomenų bazės operacijoms yra sukurta *DatabaseAPI* klasė, kuri turi visas funkcijas, susijusias su duomenų baze. Ši klasė yra iškviečiama, kai naudotojas įveda duomenis ir spaudžia prisijungimo mygtuką. „Prisijungti“ mygtukas turi klausytoją, kuris laukia kol naudotojas jį paspaus, jį paspaudus yra iškviečiamas metodas *onClick*(), yra paimami įvesti duomenys ir paduodami funkcijai *CheckUser*() (*DatabaseAPI*), slaptažodis yra užkoduojamas MD5 šifru (saugumo sumetimais, kadangi niekas neturėtų matyti jautrių duomenų). Funkcija bando paimti elektroninį pašto adresą iš *Users* lentelės ir jeigu įvestas naudotojo vardas ir užkoduotas slaptažodis sutampa su slapyvardžiu ir užkuoduotu slaptažodžiu duomenų bazėje, yra gražinamas naudotojo elektroninis paštas. Jeigu funkcijos rezultatas nėra tuščias, reiškia, kad naudotojas egzistuoja ir jis gauna priėjimą prie visų funkcijų. Tuo pat metu yra panaudojama ir kita klasė *SessionManager*. Klasė naudoja *SharedPreferences* sąsają, kuri leidžia įrašyti duomenis (gali būti nustatymai, naudotojo duomenys ir pan.) į programėlės atmintį. Ši sąsaja naudojama tam, kad pažymėti, jog naudotojas yra prisijungęs (naudotojas bus prisijungęs, todėl jis visąlaik pateks į programėlę, nebent atsijungė). Be to, sąsajoje yra saugomas naudotojo slapyvardis ir elektroninis paštas (kad meniu lange būtų atvaizduoti).

3.3 Registracija

Registracijos funkcionalumas veikia taip: naudotojui reikia įvesti elektroninį paštą, slapyvardį ir slaptažodį bei jo patvirtinimą. Visi šie duomenys yra paimami ir tikrinami, bandant nustatyti ar visi laukeliai yra užpildyti, jeigu ne, naudotojui yra pranešama, kad ne visi duomenys įvesti. Neradus klaidų yra tikrinama ar slaptažodžio laukeliai sutampa ir ar slaptažodis yra ilgesnis negu 6 simboliai. Visai tai patikrinus žiūrima ar naudotojas įvedė gerą slaptažodį naudojant *android.util.Patterns.EMAIL_ADDRESS*. Vėliau patikrinama ar toks elektroninis adresas ir slapyvardis nėra jau duomenų bazėje, nes negalima užregistruoti dviejų naudotojų su vienodu adresu ar slapyvardžiu, kartu yra patikrinimas ar yra interneto ryšis, kadangi jo reikės vėliau. Jeigu nėra klaidų, naudotojo duomenys yra išsaugomi duomenų bazėje (slaptažodis užkoduotas MD5 šifru). Kai naudotojas yra sėkmingai užregistruotas, jam į elektroninį paštą yra atsiunčiamas laiškas kuris pasveikina jį prisiregistravus, taip pat yra pateikiamas jo slapyvardis ir slaptažodis, kadangi naudotojas gali pamiršti šiuos duomenis. Visai tai atliekama naudojant *javax.mail* biblioteką, panaudojus savo elektroninio pašto duomenis, ir nurodžius SMTP, SMT Port, SMT Host (Gmail serverio informacija).

3.4 Naudotojo nustatymai

Programėlė turi naudotojo nustatymus, nustatymai leidžia naudotojui pašalinti jo paskyrą, pakeisti slaptažodį, išjungti dienos iššūkius. Paspaudus „Pašalinti paskyra“, naudotojo yra paklausiama ar jis tikrai nori to, naudojant *AlertDialog* (nustatomi teigiamo atsakymo veiksmai). Naudotojui sutikus, sesijos parametrai ir naudotojo duomenys (esantys duomenų bazėje) yra sunaikinami ir ištrinami. Funkcija ištrina naudotoją iš duomenų bazės, kai randa slapyvardį *User* lentelėje. Duomenys yra ištrinami ir iš pagautų žuvų lentelės. Naudotojui paspaudus pakeisti slaptažodį naudotojui yra parodomas naujas dialogas, kuriame jis turi įvesti seną ir naują slaptažodį,

jeigu senas slaptažodis yra teisingas, iškviečiama funkcija, kuri atlieka duomenų atnaujinimą. Atlikus veiksmą naudotojui yra atsiunčiamas elektroninis laiškas su nauja informacija.

3.4 Foninis procesas

Foninis procesas yra atsakingas už licencijų priminimus [4], dienos iššūkius, tikrinimą ar naudotojas artėja prie uždraustos vietovės. Todėl funkcijos turi veikti net programėlei esant uždarytai. 4 pav. pavaizduota funkcija, kuri leidžia palaikyti procesą gyvą net išjungus programėlę, ji yra iškviečiama foninio proceso klasėje vos ją paleidus. Kad procesas būtu paleistas (jeigu jis neveikia) yra naudojama funkcija (6 pav.) *DrawerActivity* klasėje, kuri patikrina ar procesas veikia, jeigu neveikia, procesas yra paleidžiamas. Pagrindinė foninio proceso funkcija yra licencijos patikrinimas, ji paima duomenis (iš duomenų bazės) apie įvestas licencijas, kurios priklauso naudotojui, yra iteruojamas *ArrayList* ir paimamos datos, kurios yra apadoruojamos su *SimpleDateFormat* (nurodomas datos formatas dd/MM/yyy HH:mm – diena, mėnesis, metai, valandos ir minutės). Datos formatas turi funkciją *before*, kuri leidžia palyginti datas. Pav. 5 pavaizduoja kaip yra lyginamos datos, c1 yra esamas laikas pridėjus vieną kalendorinę dieną, jis yra lyginamas su data gauta iš duomenų bazės (licencijos galiojimo pabaigos data), yra gražinama reikšmė didesnė už 0, jeigu lyginama data yra didesnė, tada tikrinama ar naudotojas jau buvo informuotas ir pažiūrima ar licencijos galiojimo laikas nėra pasibaigęs, naudojant *before*.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    return START_STICKY; // turetu padeti neuzdaryti proceso
}
```

4 pav. Foninio proceso gyvavimas

```
SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd/MM/yyy HH:mm", Locale.UK);
Date date = dateFormat.parse(array[1]);
if ((c1.getTime().compareTo(date) > 0) && array[3].equals("0") && !date.before(c2.getTime())) {
```

5 pav. Datų palyginimas

```
// jeigu backgroundo procesas yra isjungtas jis paleidziamas
if (!isMyServiceRunning(BackgroundService.class)) {
    Intent BackgroundService = new Intent( packageContext: this, BackgroundService.class);
    ContextWrapper cont = new ContextWrapper(getBaseContext());
    cont.startService(BackgroundService);
}
```

6 pav. Foninio proceso paleidimas

Kad įspėti naudotoja apie licencijos pasibaigimą, yra sukurtas priminimų kanalas (angl. *Notification Channel*), nurodytas jo pavadinimas ir ID, naudojant ID galima kurti priminimus (7 pav.). Funkcija leidžia nurodyti priminimo ikoną, pavadinimą, stilių ir tekstą.

```
public void createNotification(String message) {

    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this, CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_android)
            .setContentTitle("Priminimas")
            .setStyle(new NotificationCompat.BigTextStyle()
                    .bigText(message))
            .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);

    notificationManager.notify( id: 1, builder.build());
```

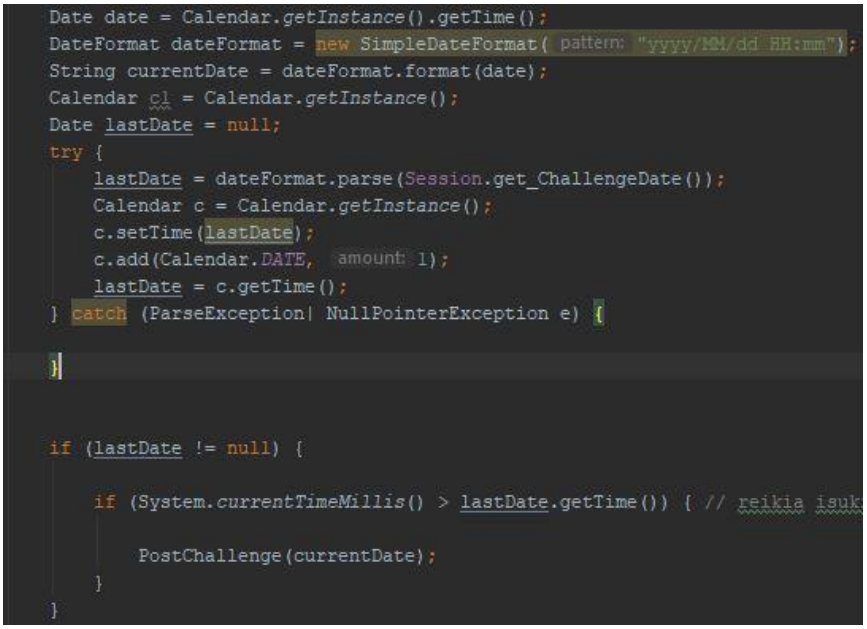
7 pav. Priminimo kūrimas

Funkcija (8 pav.) tikrina naudotojo atstumą iki uždraustų telkinių. Prieš gaunant naudotojo lokaciją yra patikrinama ar naudotas turi įjungęs padėtį, jeigu padėtis įjungta, naudojamas metodas *getSystemService(Context.LOCATION_SERVICE)* ir *getLastKnownLocation(LocationManager.GPS_PROVIDER),* kad gauti tikslią naudotojo vietą. Yra sukuriamas *Location loc1* objektas (*Location* objektas leidžia gauti naudotojo ilgumą ir platumą) iš esamos pozicijos duomenų ir tada sukuriamas antras toks objektas, kuriam nustatomi duomenys, paimti iš duomenų bazės (ilguma ir platuma). Metodas *distanceTo* leidžią gauti atstumą tarp dviejų taškų, padalinus šią reikšmę iš 1000 gaunamas atstumas kilometrais. Paskui yra žiūrima ar naudotojas yra arčiau nei 5 kilometrai nuo tos vietovės, jeigu taip, naudotojas yra įspėjamas pateikus jam priminimą (angl. *notification*). Priminimas jam yra pateikiamas kas 5 minutes (jeigu jo nepamatė ir jis vis artėja prie tos vietovės, be to, visos funkcijos yra paleidžiamos kas 5 minutes, baterijos taupymo sumetimais).

Dienos iššūkio funkcija (9 pav.) paima esamą datą, tada yra suformatuojamas datos formatas, iš *SharedPreferences* (*SessionManager* klasė) yra paima paskutinė priminimo data, prie datos pridedama viena kalendorinė diena ir tada tikrinama ar esama data (laikas) yra didesnė. Jeigu didesnė, yra paimama atsitiktinė reikšmė iš iššūkių masyvo ir pateikiamas priminimas.

```
Location location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
assert location != null;
double latitude;
double longitude;
try
{
   longitude = location.getLongitude();
   latitude = location.getLatitude();
    Location locl = new Location( provider: "");
    locl.setLatitude(latitude);
    locl.setLongitude(longitude);

    Iterator<String[]> itr = ForbiddenLocations.iterator();
    while (itr.hasNext()) {
        String[] array = itr.next();

        Location loc2 = new Location( provider: "");
        loc2.setLatitude(Double.parseDouble(array[0]));
        loc2.setLongitude(Double.parseDouble(array[1]));

        float distance = locl.distanceTo(loc2) / 1000;

        if (distance < 5.0) //mazesnis nei 5km atstumas
```

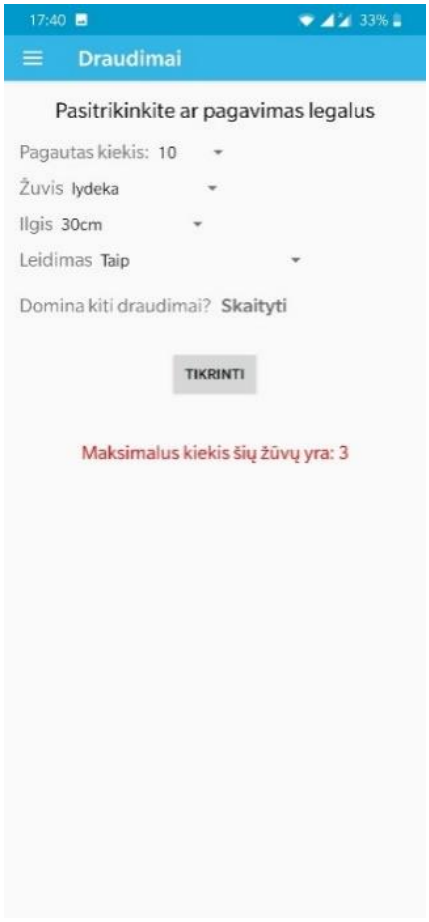
8 pav. Atstumo lyginimas



9 pav. Dienos iššūkio funkcija

3.5 Ribojimų tikrinimas

Tai yra viena iš svarbesnių funkcijų, kuri leidžia patikrinti ar galima parsinešti pagautą laimikį. Visi svarbiausi duomenys (draudžiamos žuvys, minimalūs žuvų ilgiai, mėgėjo kortelės reikalavimas, maksimalūs žuvų kiekiai) iš [5] yra patalpinti į duomenų bazę. 10 pav. pavaizduotas šio funkcionalumo dizainas. Naudotojui pasirinkus visus elementus ir paspaudus „Tikrinti“ funkcija pažiūri ar visi duomenys pasirinkti, jeigu taip, tada yra paimami apribojimai iš duomenų bazės ir iteruojami bei kartu tikrinama ar naudotojo laimikis yra teisėtas. Vykdant tikrinimą reikia pasirūpinti, kad Kg būtu pašalintas iš teksto (draudimai yra tiek kilogramais, tiek žuvų skaičiumi). Yra praeinami *if* tikrinimai, kiekvieno pažeidimo metu tekstas yra papildomas, praėjus viską, yra pateikiamas visas sąrašas pažeidimų. 11 pav. pavaizduoja šią funkciją, patikrinama ar naudotojo pasirinkime yra Kg, jeigu taip, jis yra pašalinimas ir atliekamas patikrinimas su duomenimis iš duomenų bazės, galutinis tekstas yra papildomas. Kartu yra tikrinama ar naudotojas patikrino didesnį nei 5kg žuvų kiekį, kadangi tai yra limitas. Šis skaičiavimas yra atliekamas taip: paspaudus tikrinimo mygtuką ir radus, kad naudotojas pasirinko svorį, tas svoris įdedamas į masyvą, kitą kartą paspaudus mygtuką, tie svoriai yra sudedami ir žiūrima ar svoris viršijo 5kg, jeigu taip, naudotojas yra įspėjamas. Mygtukas „Skaityti“, atidaro visą tekstą apie draudimų subtilybes.



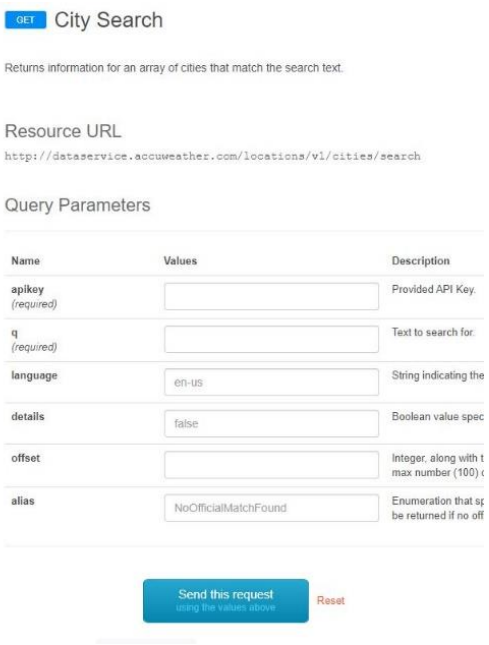
10 pav. Draudimų funkcijos dizainas

```
String am = (String) amount.getSelectedItem();
boolean kg;
Pattern pattr = Pattern.compile( regex: "[Kg]", Pattern.CASE_INSENSITIVE);
Matcher match = pattr.matcher(am);
boolean found = match.find();
if (found)
    kg = true;
else kg = false;


if (kg)
{
    am =  am.replace( target: "kg", replacement: "");
    weights.add(am);

    if (Integer.parseInt(am) > 5)
    {

        textLim += "Negalima pagauti daugiau nei 5Kg (Kuršių marios 7kg) " +"\n" + "\n";
    }

}
```

11 pav. Draudimų funkcijos algoritmas

3.6 Orų funkcija ir kibimo prognozė

AccuWeather yra orų sistema. Kad naudotis ja, reikia prisiregistruoti prie https://developer.accuweather.com konsolės. Gavus API raktą galima naudotis funkcijomis kurios reikalauja rakto. Pavyzdžiui norint gauti Vilniaus miesto kodą, reikia naudoti konsolę (12 pav.).



12 pav. miesto paieška


```
"DailyForecasts": [
  {
    "Date": "2019-11-18T07:00:00+02:00",
    "EpochDate": 1574053200,
    "Temperature": {
      "Minimum": {
        "Value": 38,
        "Unit": "F",
        "UnitType": 18
      },
      "Maximum": {
        "Value": 47,
        "Unit": "F",
        "UnitType": 18
      }
```


13 pav. JSON duomenys

Turint miesto kodą, galima gauti prognozę JSON formatu. AccuWeather leidžia gauti prognozes: 1 dienos, dabartinę, 5 dienų, 15 dienų, 10 dienų, valandines. Buvo panaudota 5 dienų ir dabartinė prognozė, tam kad nustatyti kibimo prognozę. 13 pav. pavaizduota kaip AccuWeather pateikia duomenis, suvedus API raktą ir miesto kodą, tai yra JSON formatu pateikti duomenys [8] . Tam, kad gauti šituos duomenis (programėlėje) reikia funkcijai *getJsonData()* paduoti URL. Ši funkcija naudodama *scanner* nuskaito JSON ir gražina jį kaip String. Vėliau *String* yra apdorojamas sugeneravus *JSONObject* tipo objektą iš *String* (14 pav.), tai leidžia įtaruoti JSON objektus (reikšmes). Trumpai apie JSON, „džeisonas“ – iš anglų kalbos JavaScript Object Notation yra atviro

standarto formatas, perduodantis duomenų objektus, sudarytus iš atributo ir reikšmės porų, lengvai skaitomame tekste. Tai yra pirminis asinchroninio naršyklės/serverio bendravimo (AJAX) duomenų formatas.

```
JSONArray mJsonArray = new JSONArray(weatherResults);
JSONObject mJsonObject = mJsonArray.getJSONObject( index: 0);

JSONObject temperatureObj = mJsonObject.getJSONObject("Temperature");
String minTemperature = temperatureObj.getJSONObject("Metric").getString( name: "Value");
temp.setText(minTemperature + " C");

JSONObject pressureObj = mJsonObject.getJSONObject("Pressure");
String pressureText = pressureObj.getJSONObject("Imperial").getString( name: "Value");
pressure.setText(pressureText + " inHg");
```

14 pav. JSON duomenų apdorojimas

Kad visus šiuos gautus duomenis atvaizduoti, panaudotas sąrašo vaizdas (angl. *ListView*) ir adapteris. Papildomai pasitelkiama *WeatherObject* klasė kaip objektas (15 pav.), kurios elementai yra nustatomi apdorojant JSON (16 pav.), naudojant *get* ir *set*. Duomenys iš *WeatherObject* objekto yra įdedami į *weatherArrayList* sąrašą, šis sąrašas naudojamas adapteriui paduoti duomenis į sąrašo vaizdą.

```
public void setMinTemp(String minTemp) { this.minTemp = minTemp; }

public String getMaxTemp() { return maxTemp; }

public void setMaxTemp(String maxTemp) { this.maxTemp = maxTemp; }
```

15 pav. *WeatherObject* klasė

```
weatherObject.setMinTemp(minTemperature +" C");
```

16 pav. *WeatherObject* klasės duomenų nustatymas

Prognozės algoritmas veikia taip: yra paimama slėgio tendencija esamu laiku (17 pav.), pavyzdžiui, jeigu slėgis kyla ir jis mažesnis negu 30,50 [7], reiškia, kad žuvys kibs gerai. Toliau yra pasitelkiamas bendrasis kibimo kalendorius [13] bandant nustatyti būtent kokios žuvys kibs (18 pav.). Yra iteruojamas kalendorius ir žiūrima kurios šio mėnesio žuvys kimba arba gerai arba vidutiniškai. Kiekvieną kartą iteruojant yra papildomas galutinis žuvų sąrašas. Jeigu paaiškėja, kad pagal slėgio duomenys žuvys nekibs, kalendoriaus nėra iteruojamas ir pranešama naudotojui, kad niekas nekibs.

```
else if (pressureTendency.equals("Rising") && pressure < 30.50)
{
    bite = "Prognozė: Žuvys atkyvėja dėl augančio slėgio, bet aktyvumas dugną mėgstančių žuvų mažėja";
    fishBite.setText(bite);
    d =1;
    fishBite.setTextColor(Color.GREEN);
}

else if (pressureTendency.equals("Falling") && pressure <= 30.50  &&  pressure > 29.60 )
```

17 pav. Kibimo nustatymas naudojant slėgį

```
if (d!=0) {
    ArrayList alreadyPut = new ArrayList();
    Iterator<String[]> itr = Calendar.iterator();
    while (itr.hasNext()) {
        String[] array = itr.next();
        if (array[1].equals(month) && array[2].equals("geras") || array[2].equals("vidutinis")
            fishes += array[0] +  ", ";
            alreadyPut.add(array[0]);
    }

    if (!fishes.equals("")) {
        toFish.setText("Gerai kibs: " + fishes);
        alreadyPut.clear();
        toFish.setTextColor(Color.GREEN);
    }
```

18 pav. Žuvų kibimo nustatymas

3.7 Naudotojo sąsaja

Naudotojo sąsajos modeliavimas yra viena iš svarbiausių dalių, kadangi neapgalvota ir prasta sąsaja gali atbaidyti klientą ir pasirodyti neprofesionali bei nesuteikianti daug pasitikėjimo. Žinoma, ši architektūros dalis svarbi ir tuo, kad yra nusprendžiama, koks funkcionalumas bus įgyvendintas ir kaip komponentai atvaizduos sistemos elementus, kaip jie sąveikaus su naudotojo įvestimi, kokia informacija bus gaunama ir siunčiama kitiems komponentams.

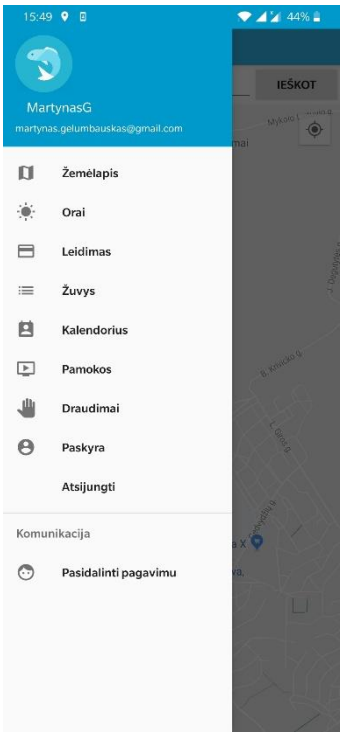
19 pav. Registracijos langas

20 pav. Informacija apie orus ir kibimą

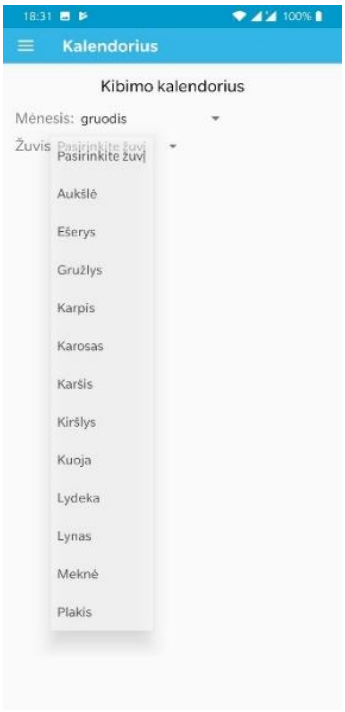
19 pav. pavaizduotas naudotojo registracijos langas, kurį taisyklingai užpildžius, yra sukuriama paskyra (naudotojo duomenys išsaugomi duomenų bazėje). Visos reikšmės yra įrašomos ranka ir vėliau sistema jas patikrina. Laukelis „Slapyvardis“ yra patikrinamas su visais slapyvardžiais, esančiais duomenų bazėje. Jeigu slapyvardis sutampa su jau egzistuojančiu, tai naudotojas informuojamas, kad toksai slapyvardis jau yra užregistruotas. Laukelis „El. paštas“ irgi yra patikrinimas norint nustatyti ar tai yra legalus elektroninis paštas. Jeigu el. paštas neatitinka šablono, tai naudotojas gauna klaidos pranešimą. Laukeliai „Slaptažodis“ ir „Patvirtinti“ yra palyginami vienas su kitu, norint įsitikinti ar naudotojas tikrai nori naudoti tokį slaptažodį. Taip pat reiktų atkreipti dėmesį į sąsajos spalvas, jos yra švelnios akims, fonas neutraliai baltas, atrodo profesionaliai ir nevargina akių. Kiekvienas laukelis turi savo ikoną, tai suteikia programėlei profesionalumo ir aiškumo.

20 pav. pavaizduotas orų langas. Pačiame viršuje yra atvaizduojama dabartinė temperatūra, slėgis, drėgnumas ir ikona, kuri atvaizduoja dabartines sąlygas. Po šia informacija yra pateikiama

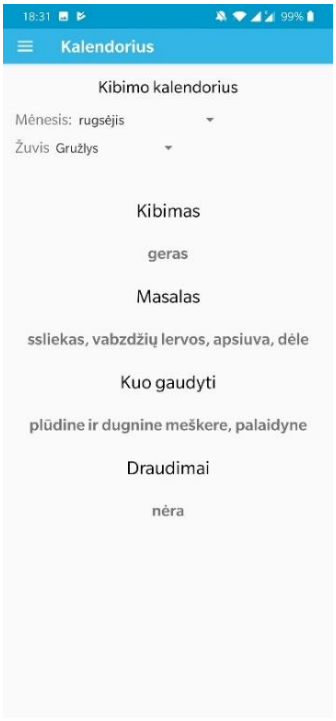
prognozė apie kibimą šiuo laiko momentu. Dėl erdvės trūkumo tekstas yra bėgantis, todėl į vieną eilutę telpa daug informacijos. Tekstas keičia spalvą pagal kibimą: jeigu kibimas yra blogas – tekstas bus raudonas, jeigu geras – tai bus žalias. Pirmoji teksto eilutė nusako dabartinio slėgio poveikį žuvims. Antroji nusako kokios žuvys galimai kibs (pasitelkiant kibimo lentelės informacija). Po kibimo informacija yra penkių dienų orų informacija: mažiausia ir didžiausia temperatūra, data bei ikona, parodanti debesuotumą.



21 pav. Navigacijos meniu



22 pav. Kibimo lentelės meniu



23 pav. Informacija apie kibimą

21 pav. pavaizduotas programėlės meniu langas. Tai yra visiems gerai žinomas meniu lango stilius, kuris yra atidaromas pabraukus pirštu iš kairės į dešinę ir uždaromas pabraukus pirštu iš dešinės į kairę. Meniu viršuje yra atvaizduojamas naudotojo paveiksliukas, jo slapyvardis ir elektroninis paštas, tai leidžia naudotojui žinoti, kad jis yra sėkmingai prisijungęs. Žemiau yra nuorodos į pagrindines programėlės funkcijas, kurios yra atidaromos paspaudus meniu elementą. Kiekvienas elementas turi savo ikoną, tai suteikia profesionalumo dizainui ir vartotojas gali vizualiai greičiau pasirinkti norimą funkciją.

22 pav. pavaizduota kaip yra suprojektuotas kibimo kalendoriaus langas. Langas turi du suktukus (angl. *spinners*). Pirmame suktuke galima pasirinkti norimą mėnesį, antrame – žuvį. Pasirinkus abi reikšmes, bus atvaizduojama informacija apie tos žuvies kibimą tą mėnesį (23 pav.), t. y. ar tai bus geras kibimas ar blogas, rekomenduojamas masalas, rekomenduojama gaudymo įranga ir kokie draudimai gali būti tą mėnesį.

Išvados ir rekomendacijos

Atlikus projektą buvo padarytos šios išvados:

- Panašių sistemų nėra daug, ypač lietuviu kalba, ir nei viena iš jų neturi pilno sąrašo funkcijų, kurios buvo išvardintos ir nagrinėtos, todėl pataisius visas problemas ir patobulinus programėlę, ją galima būtu įkelti į „Play Store“.

- Android Studio yra populiari programavimo aplinka ir ja patogu naudotis, kadangi tai yra Google produktas, kuris yra nuolatos atnaujinamas.

- SQLite yra patogi ir patikima duomenų bazės valdymo sistema, jeigu nėra siekiama duomenų laikyti duomenų serveryje.

- Mobilieji telefonai sparčiai populiarėja, todėl žvejo informacinės sistemos poreikis turi tendenciją augti.

Kad programėlę padaryti geresnę reikėtų sukurti šiuos funkcionalumus:

1. Padaryti žvejų socialinę funkciją (pasidalinti pagavimu su kitais žvejais, aptarti kibimo taktikas).

2. Padaryti slaptažodžio priminimo funkciją.

3. Pataisyti problemą, kad pabaigus žiūrėti vaizdo pamoką programėlė grįžtų į pagrindinį langą.

4. Padaryti kibimo prognozę ateinančioms dienoms.

5. Padaryti miesto pasirinkimą, kad pažiūrėti orus kituose miestuose.

6. Galimybę pažiūrėti kokios žuvys kimba telkiniuose naudojant kitų naudotojų duomenis.

Literatūros šaltiniai

[1] AccuWeather APIs. Prieiga per internetą: https://developer.accuweather.com

[2] Android platform. Libraries. Prieiga per internetą: https://developer.android.com/topic/libraries/support-library/packages

[3] Android Studio. User guide. Prieiga per internetą: https://developer.android.com/studio/intro

[4] Aplinkosaugos leidimų informacinė sistema. Leidimo užsakymas. Žvejo mėgėjo bilietas. Prieiga per internetą: https://alis.am.lt/actionFishingApplicationNew.action?fishingPermitType=1

[5] Delfi. Kablys. Mėgėjų žvejybos taisyklės ir draudimai. Prieiga per internetą: https://kablys.delfi.lt/zvejyba/naujienos/megeju-zvejybos-taisykles-atnaujinta-2019-07-26.d?id=74026588

[6] Google Maps Platform. Maps SDK for Android. Prieiga per internetą: https://developers.google.com/maps/documentation/android-sdk/intro

[7] Slėgio poveikis žuvims. Prieiga per internetą: https://www.fishingskillz.com/best-barometric-pressure-for-fishing/

[8] Wikipedia. JSON. Prieiga per internetą: https://en.wikipedia.org/wiki/JSON

[9] Wikipedia. Java (programming language). Prieiga per internetą: https://en.wikipedia.org/wiki/Java_(programming_language)

[10] Wikipedia. SQLite. Prieiga per internetą: https://en.wikipedia.org/wiki/SQLite

[11] YouTube Android Player API. Prieiga per internetą: https://developers.google.com/youtube/android/player

[12] Žvejo svetainė. Informacija apie žuvis. Prieiga per internetą: http://www.zvejosvetaine.lt/category/apie-zuvis/zuvys/

[13] Žvejo svetainė. Žuvų kibimo lentelė. Prieiga per internetą: http://www.zvejosvetaine.lt/tag/kibimo-lentele/