LATVIJAS UNIVERSITĀTES
DATORIKAS FAKULTĀTE

Lietojumprogrammatūra pasākumu organizēšanai

KVALIFIKĀCIJAS DARBS

Autors: Artūrs Amirovs
Studenta apliecības Nr.: aa16069
Darba vadītājs: M.dat. Emil Syundyukov

RĪGA 2018

ANOTĀCIJA

Kvalifikācijas darbā ir izstrādāta Android lietotne pasākumu organizācijai. Lietotne sastāv no četrām daļām.

Pirmajā (galvenajā) daļā «Pasākumi», lietotājs var apskatīt dažādes aktivitātes un var tām pievienoties. Pasākumi dalās 3 dažādos tipos:

- «Publiskie (Public) pasākumi» – pasākumi, kuros ir piekļuve visiem lietotājiem, ar dalībnieku skaita ierobežojumu.
- «Masu pasākumi» – pasākumi, kuros ir piekļuve visiem lietotājiem, bez dalībnieku skaita ierobežojuma.
- «Slēgtie pasākumi» – pasākumi, kuriem ir ierobežots lietotāju loks.

Otrajā daļā «Manas aktivitātes» lietotājs var izveidot jebkura tipa pasākumu, kontrolēt un rediģēt jau izveidotos pasākumus un apskatīt tekošos pasākumus, ar kuriem lietotājs ir saistīts.

Trešajā daļā "Admininstratora rīkjosla", Administratoram ir piekļuve lietotnes pārvaldīšanas instrumentiem.

Ceturtajā daļā "Reklāmas pārvaldīšanas panelis", reklāmas ievietotājam ir piekļuve visiem instrumentiem darbam ar reklāmu.

Izstrādātas lietojumrogrammatūras mērķis ir sniegt lietotājam platformu, kura ir spējīga apvienot domubiedrus kopīgai laika pavadīšanai. Tāpat, ja negaidīt ir uzrādies brīvais laiks, ar izstrādātas lietojumprogrammatūras palīdzību, ir iespēja viegli atrast interesantu nodarbošanos dažu minūšu laikā.

Atslēgvārdi: Android, Firebase, SQLite, Java, pasākumi, XML, mobilā lietojumprogrammatūra, Android Studio.

ABSTRACT

Mobile application for event organization

In frames of qualification work was developed Android application for event organization. Application consists of four main parts:

In first part “Events” user can observe different activities and join the one he is interested in. There are three different type of events:

- “Public events”– events, that are available for all users and number of participants in such events is limited.
- “Massive events” – events, that are available for all users and number of participants in such events is unlimited.
- “Private events”– events, that are available only for selected users.

In second part “My activity” user can create his own event of any type, manage and edit already create events and view active events that has relations to selected user in anyway.

In third part “Admin control panel” admin has access to application management tools.

In last part “Commercial control panel” advertiser can easily manage ad posts, because of tools that are available for him.

The goal of developed application is to create platform that is capable to unit like-minded people and give them opportunity to organize joined events. Furthermore, if unexpectedly appeared free time, with help of this Application it is possible to find interesting and exciting activity for a couple of minutes.

Keywords: Android, Firebase, SQLite, Java, events, XML, mobile application, Android Studio.

SATURA RĀDĪTĀJS

1 IEVADS

1.1 Tēmas izvēle

Izstrādāta lietojumprogrammatūra šajā darbā risina ļoti svarīgu problēmu, saistītu ar domubiedru meklēšanu kopīgai laika pavadīšanai. Balstoties uz personīgo pieredzi un apkārtējo cilvēku viedokli, var teikt, ka šī problēma ir aktuāla un lietojumprogrammatūrai ir visas iespējas palīdzēt cilvēkiem atrast veidu kā pavadīt brīvo laiku, un apvienot lietotājus ar līdzīgām interesēm.

1.2 Darba mērķis un uzdevumi

Darba uzdevums ir izstrādāt Android lietojumprogrammatūru, kas apvienos domubiedrus kopīgā laika pavadīšanā un palīdzēs jebkuram gribētajam, ja viņam negaidīti uzrādies brīvais laiks, atrast interesantu nodarbošanos vien dažu minūšu laikā. Darba mērķis ir iepazīstināt un savest kopā pilnīgi svešus cilvēkus ar līdzīgām interesēm, lai piedalītos kādā pasākumā.

1.3 Darba struktūra

Darbs sastāv no vairākām daļām – programmatūras prasību specifikācijas, programmatūras projektējuma apraksta, testēšanas dokumentācijas, projekta pārvaldības apraksta, darbietilpības novērtējuma, dažādiem pielikumiem un papildinājumiem.
Projekta izstrādei tika izvēlēta programmēšanas valoda Java un izstrādes vide Android Studio 3.0. Papildus tika izmantota Firebase reāllaika mākoņkrātuve un Room Persistence Library darbam ar datu bāzi.

2 PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

2.1 Ievads

2.1.1 Nolūks

Šis programmatūras prasību specifikācija (PPS) ir paredzēta pasākumu organizācijas Android lietotnes prasību aprakstīšanai. Dokuments ir savstarpēja vienošanās starp pasūtītāju un izpildītāju. Šajā prasību specifikācijā tiek noformulētas sistēmas prasības un raksturota funkcionalitāte.

2.1.2 Darbības sfēra

Lietotnē tiek atspoguļoti visi pasākumi, kurus ir izveidojuši lietotāji. Jebkuram lietotājam ir iespēja pievienoties atvērtajam pasākumam. Pasākumi dalās 3 kategorijās:

- «Slēgtie pasākumi» – pasākumi, kuriem ir ierobežots lietotāju loks.
- «Publiskie (Public) pasākumi» – pasākumi, kuriem piekļuve ir visiem lietotājiem, ar dalībnieku skaita ierobežojumu.
- «Masu pasākumi» – pasākumi, kuriem piekļuve ir visiem lietotājiem, bez dalībnieku skaita ierobežojuma.

Sadaļā «Manas aktivitātes» lietotājs var sekot līdzi pasākumiem, kurus pats ir izveidojis un redzamas dažādas aktivitātes.

2.1.3 Saistība ar citiem dokumentiem

Dokumenta noformēšanā ievērotas standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" prasības.

2.1.4 Definīcijas, akronīmi un saīsinājumi

Android – mobilo ierīču operētājsistēma, kura balstīta uz Linux bāzes.
Java – objektorientēta programmēšanas valoda
Firebase – mākoņservisa datubāze, kas ļauj lietotnes izstrādātajiem glabāt un sinhronizēt datus starp vairākiem klientiem.
API – lietotnes programmas saskarne, kas nosaka funkcionalitāti, kuru nodrošina programma.

Mākoņu glabātuve – attālinātais datu uzglabāšanas modelis

Fragments – android saskarnes komponente iekļaujamā citos fragmentos vai aktivitātēs.

XML – paplašinātā iezīmēšanas valoda;

Activity – lietotnes komponents, kurš izdot ekrānu, un ar kuru lietotāji var sadarboties jebkādu uzdevumu izpildei, piemēram, sastādīt tālruņa numuru, uzņemt bildi, nosūtīt vēstuli vai apskatīt karti

Public – piekļuve ir jebkuram

Private – ar ierobežotu piekļuvi

PPS – Programmatūras prasību specifikācija;

PPA –

Backlog – nepabeigtu darbu vai jautājumu uzkrājums, kurš prasa izskatīšanu

Kanban – izstrādes metodes pārvaldīšana, realizē principu "precīzi laikā" un, kas spēj ietekmēt vienmērīgai slodzes sadalīšanai starp izstrādātājiem

2.2 Vispārējs apraksts

2.2.1 Produkta perspektīva

Perspektīvā lietotne varētu kļūt par galveno sava brīva laika plānotāju. Ar lietojumprogrammatūras palīdzību ikvienam būs iespēja atrast savu komadu jebkura veida aktivitātei. Kā rezultātā, lietotne palīdz iepazīties ar jaunajiem cilvēkiem, taupa laiku savas nodarbes meklēšanā un palīdz atrast interesantas vietas brīva laika pavadīšanai pilsētā.

2.2.2 Produkta funkcijas

Nodaļā "Funkcionālās prasības" detalizēti aprakstītas visas funkcijas. Sistēmas funkcijas ir strukturizētas 4 moduļos:

- Pasākumu modulis – modulis pasākumu pārvaldīšanai lietotnē.
- Administrēšanas modulis – modulis lietotnes pārvaldīšanai un administrēšanai.
- Reklāmas modulis – modulis darbam ar reklāmu lietotnē.
- Autentifikācijas modulis – modulis, kurš atbild par lietotāja piekļuvi lietotnei.

2.2.3 Lietotāju raksturiezīmes

Visām lietotāju grupām ir nepieciešamas elementāras prasmes darbam ar Android mobilo telefonu, lai sekmīgi varētu lietot šo sistēmu.

Lietotāji ir sadalīti 5 grupās, kuriem nepieciešams zināt, kā izmantot atbilstošos moduļus un funkcijas.

- **Neautorizētie lietotāji** – katrs lietotājs, atvērot lietotni, pieder šai grupai. Šī lietotāju grupa nodarbina tikai Autentifikācijas moduļa funkcijas.
- **Reģistrēts lietotājs un Priviliģēts lietotājs** – lietotnes galvenie lietotāji, kuri izmanto Pasākuma moduļa funkcijas.
- **Administrators** – cilvēks, kuram jāzina, kā jārīkojas ar visu moduļu funkcijām, lai nepieciešamības gadījumā palīdzētu lietotājiem ar radušajām problēmām. Pa tiešo Administrators nodarbina Administrēšanas moduļa funkcijas.

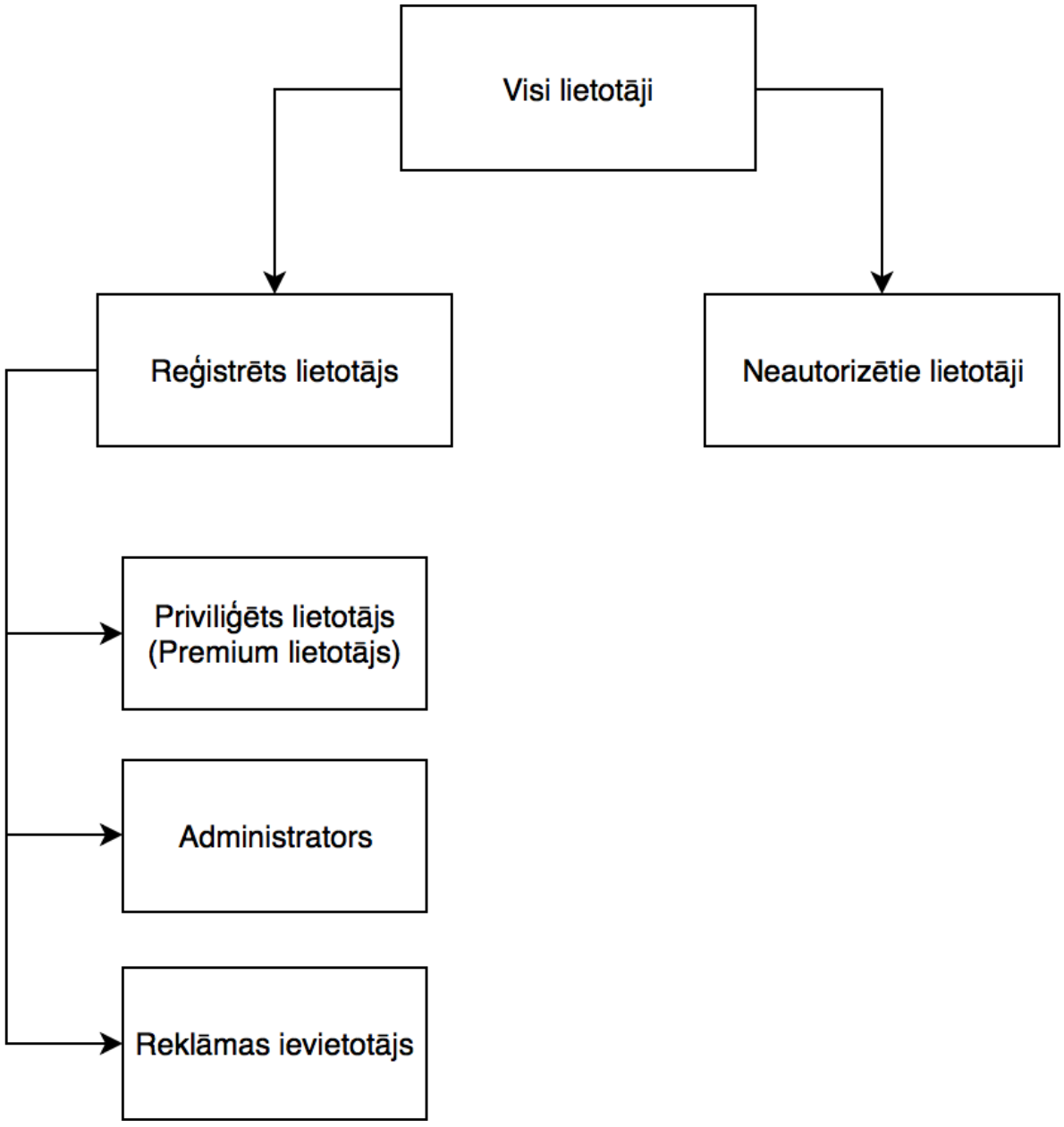
- **Reklāmas ievietotājs** – lietotājs, kurš var veidot reklāmu un aizsūtīt vaicājumu administratoram par reklāmas izvietošanu galvenajā lapā.

2.2.4 Lietotāju grupu apraksts

Lietotāju grupa	Apraksts
Visi lietotāji	Grupai pieder visu lietotāju grupas, kurām ir piekļuve pie lietojumprogrammatūras.
Neautorizētie lietotāji	Šai grupai nav piekļuves nevienai lietotnes funkcijai, izņemot autentifikāciju.
Reģistrēts lietotājs	Autorizēts lietotājs, kuram ir piekļuve visam lietotnes pamata funkcionālam.
Priviliģēts lietotājs (Premium lietotājs)	Autorizēts lietotājs, kuram ir piekļuve visam lietotnes pamata funkcionālam, noņemti ierobežojumi, kuri ir reģistrētam lietotājam, kā arī ir piekļuve papildfunkcijām.
Administrators	Lietotājs, kurš pārvalda un kontrolē lietojumprogrammatūru.
Reklāmas ievietotājs	Lietotājs (juridiska persona), kurš var veidot pieprasījumus reklāmas izvietošanai un organizēt sponsorētus pasākumus.

2.2.5 Lietotāju grupu diagramma

Šī diagramma attēlo lietotāju sadalījumu pēc 2 pazīmēm, reģistrēts (autorizēts) lietotājs vai neautorizēts lietotājs, un vai lietotājs, kurš papildus pieder kādai citai (priviliģētai) grupai.



2.1. att. ***Lietotāju grupu diagramma***

2.2.6 Lietotāju grupu saistība ar moduļiem

2.1. tabula. ***Lietotāju grupu saistība ar moduļiem***

	Funkcija	Identifikators	Visi lietotāji				
			Neautorizētie lietotāji	**Reģistrēts lietotājs**	**Priviliģēts lietotājs**	**Administrators**	**Reklāmas ievietotājs**
Pasākumu modulis	Izveidot pasākumu	DF.PM-01		X	X		X
	Ģenerēt paroli no 4 cipariem	DF.PM-03		X	X		X
	Pievienoties pasākumam	DF.PM-04		X	X		
	Atteikties no pasākuma	DF.PM-05		X	X		
	Dzēst izveidoto pasākumu	DF.PM-06		X	X		X
	Pievienoties pasākumam ar paroli	DF.PM-07		X	X		
Reklāmas modulis	Izveidot reklāmu	DF.RM-01					X
	Dzēst reklāmu	DF.RM-02					X
	Nosūtīt vaicājumu reklāmas izvietošanai	DF.RM-03					X
Administrēšanas modulis	Dzēst lietotāju	DF.AD-03				X	
	Rediģēt lietotāju	DF.AD-04				X	
	Rediģēt pasākumu	DF.AD-05				X	
	Dzēst pasākumu	DF.AD-06				X	

	Funkcija	Identi-fikators	Visi lietotāji				
			Neauto-rizētie lietotāji	Reģistrēts lietotājs	Priviliģēts lietotājs	Admini-strators	Reklāmas ievietotājs
	Apstiprināt reklāmas izvietošanu	DF.AD-07				X	

	Funkcija	Identi-fikators	Visi lietotāji				
Autentifikācijas modulis	Autorizēties lietotnē izmantojot Facebook API	DF.AU-01	X				
	Autorizēties lietotnē izmantojot Google API	DF.AU-02	X				
	Autentificējoties noteikt lietotāja tipu un atvērt atbilstošo saskarni	DF.AU-03	X				
	Jauna lietotāja reģistrēšana	DF.AU-04	X				
	Paroles šifrēšana	DF.AU05	X				

2.2.7 Vispārējie ierobežojumi

Sistēma ir izstrādāta un atbalstīta ar Android platformu, tas nozīmē, ka tikai Android platformas lietotāji varēs piekļūt un izmantot lietotni. Ierīces programmatūras API līmenim ir jābūt vienādam vai lielākam par 21.

2.2.8 Pieņēmumi un atkarības

Bezmaksas abonements Firebase serverim ļauj pieņemt ne vairāk kā 100 vienlaicīgus vaicājumus, tāpēc tiek pieņemts, ka ar doto lietotnes versiju maksimālais vaicājumu daudzums, lai piekļūtu serverim, nepārsniegs skaitli 100. Tāpēc tekošā pasākuma unikālas paroles ģenerēšana ļauj ģenerēt 8999 paroles, tiek pieņemts, ka privāto pasākumu skaits nepārsniegs maksimāli iespējamo unikālo paroles skaitu.

2.3 Funkcionālās prasības

2.3.1 Autentifikācijas modulis

Autentifikācijas modulis paredzēts lietotāju autorizācijai un reģistrācijai. Tāpat Autentifikācijas modulis atbild par atbilstošās saskarnes atvēršanu atbilstošai lietotāju grupai, visas funkcijas šajā modulī ir paredzētas Neautorizētiem lietotājiem.

2.3.1.1 Autorizēties lietotnē izmantojot Facebook API

Identifikators	DF.AU-01
Mērķis	
Funkcija paredzēta, lai lietotājs varētu autorizēties caur Facebook	
Lietotāju grupas	
Neautorizētie lietotāji	
Datu ievade	
Lietotāja vārds un parole	
Datu apstrāde	
Lietotājs autorizējas izmantojot savus datus Facebook tīklā	
Datu izvade	
Ja autorizācija caur Facebook API notika veiksmīgi, tad lietotājs saņem piekļuvi lietotnei, atkarībā no piederības kādai konkrētai grupai. Facebook API atgriež lietotāja fotogrāfiju, vārdu un uzvārdu	
Kļūdu paziņojumi	
Nav, tā kā viss notiek Facebook pusē	

2.3.1.2 Autorizēties lietotnē izmantojot Google API

Identifikators	DF.AU-02
Mērķis	
Funkcija paredzēta lietotāja autorizācijai caur Google kontu	
Lietotāju grupas	
Neautorizētie lietotāji	
Datu ievade	
Lietotāja vārds un parole	
Datu apstrāde	
Lietotājs autorizējas, izmantojot savus datus Google tīklā	
Datu izvade	
Ja autorizācija caur Google API notika veiksmīgi, tad lietotājs saņem piekļuvi lietotnei, atkarībā no savas piederības kādai konkrētai grupai. Google API atgriež lietotāja fotogrāfiju, vārdu un uzvārdu.	
Kļūdu paziņojumi	
Nav, tā kā viss notiek Google pusē	

2.3.1.3 Autentificējoties noteikt lietotāja tipu un atvērt atbilstošo saskarni

Identifikators	DF.AU-03
Mērķis	
Funkcija paredzēta lietotāja tipa noteikšanai autorizācijas laikā un atbilstošās lietotnes saskarnes atvēršanai	
Lietotāju grupas	
Neautorizētie lietotāji	
Datu ievade	
Lietotāja identifikators	
Datu apstrāde	
Funkcija atrod lietotāja tipu datu bāzē	
Datu izvade	
Lietotāja tips	
Kļūdu paziņojumi	

2.3.1.4 Jauna lietotāja reģistrācija

Identifikators	DF.AU-04
Mērķis	
Funkcija paredzēta, lai izveidotu jaunu lietotāju	
Lietotāju grupas	
Neautorizētie lietotāji	
Datu ievade	
Lietotāja izveidošanai nepieciešams aizpildīt sekojošus laukus: • First name (Vārds) – obligāts ievades lauks • Last Name (Uzvārds) – obligāts ievades lauks • Username(Lietotājvārds) – obligāts ievades lauks • Password (Parole) – obligāts ievades lauks	
Datu apstrāde	

Nospiežot pogu "Izveidot pasākumu" notiek sekojošais:
• Pārbauda, vai
  • Vārda lauks nav tukšs,
  • Uzvārda lauks nav tukšs,
  • Lietotājvārds nesatur atstarpes un ir unikāls,
  • parole sastāv vairāk kā no 8 simboliem un satur vismaz vienu burtu un vienu ciparu,
  Ja nē, tad paziņo par kļūdu №1 un un tiek izcelts lauks, kurš ir aizpildīts nepareizi
• Pārbauda, ka lietotāja vārds ir unikāls, ja nav, tad paziņo par kļūdu №2
• Lietotāja parole šifrējas ar DF.AU-05 funkcijas palīdzību
• Lietotājs tiek pievienots datu bāzei.
• Noslēgumā lietotājs tiek pievienots Firebase mākoņu glabātuvē.
• Lietotājam automātiski tiek dots tips: Reģistrēts lietotājs	

Datu izvade	
Reģistrēta lietotāja identifikators	
Kļūdu paziņojumi	
1. "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!" 2. "Šāds lietotāja vārds jau pastāv!"	

2.3.1.5 Paroles šifrēšana

Identifikators	DF.AU-05
Mērķis	
Funkcija paredzēta lietotāja paroles šifrēšanai, lai datu bāzē tas glabātos šifrētā veidā	
Lietotāju grupas	
Neautorizētie lietotāji	
Datu ievade	
Parole	
Datu apstrāde	
Parole, kura pieder funkcijai, tiek šifrēta atbilstoši AES (Advanced Encryption Standard), izmantojot doto šifrēšanas atslēgu	
Datu izvade	
Šifrēta parole	
Kļūdu paziņojumi	

2.3.2 Pasākumu modulis

Pasākumu modulis, ir modulis ar lietotnes pamata funkcijām. Visas funkcijas šajā modulī atbild par pasākumu organizāciju un veido saikni starp lietotāju un pasākumu.

2.3.2.1 Izveidot pasākumu

Identifikators	DF.PM-01
Mērķis	
Funkcija paredzēta, lai izveidotu jaunu pasākumu	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs, Reklāmas ievietotājs	
Datu ievade	
Pasākuma izveidošanai nepieciešams aizpildīt sekojošus laukus: • Event name (Pasākuma nosaukums) – obligāts ievades lauks • Location (Norises vieta) – obligāts ievades lauks • Starting date (Sākuma datums) – obligāta izvēle kalendārā • Ending date (Beigu datums) – obligāta izvēle kalendārā • Description (Apraksts)– obligāts ievades lauks • Minimal participants amount (Minimālais dalībnieku skaits)– obligāts skaitļa ievades lauks • Maximal participants amount (Maksimālais dalībnieku skaits)– obligāts skaitļa ievades lauks • Image (Attēls)– izvēlēts lauks • Event type (Pasākuma tips)– obligāts lauks ar divām izvēles iespējām (public, private)	
Datu apstrāde	
Nospiežot pogu “Izveidot pasākumu” notiek sekojošais: • Pārbauda, vai Pasākuma nosaukuma lauks nav tukšs, • Norises vietas lauks nav tukšs, • Sākuma un Beigu datumam jābūt izvēlētam no kalendāra, • Apraksta lauks nav tukšs, • Minimālo un Maksimālo dalībnieku skaitam jābūt norādītam skaitļos, un Maksimālo dalībniekiu skaitlim jābūt lielākam par Minimālo dalībnieku skaitu, • Attēla laukam jābūt izvēlētam, • pasākuma tipa lauks ir izvēlēts. Ja lauki ir aizpildīti nepareizi, tad paziņo par kļūdu №1 un tiek izcelts nepareizi ievadītais lauks	

• Ja pasākuma tips ir – privāts, tad pasākumam tiek ģenerēta unikāla parole no 4 cipariem, ar funkcijas DF.PM-03 palīdzību (Paroles ģenerēšana no 4 cipariem) • Tad pasākums tiek pievienots datu bāzei ar funkcijas DF.DM-04 palīdzību • Noslēgumā pasākums tiek pievienots Firebase mākoņu glabātuvē, pēc tam ir piekļuve parējiem lietojumprogrammatūras lietotājiem
Datu izvade
Ja pasākuma tips ir Privāts, tad tiek izvadīts dialoga logs paralēli pasākumam, pēc kā citi lietotāji varēs atrast šo pasākumu
Kļūdu paziņojumi
1. "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!"

2.3.2.2 Ģenerēt paroli no 4 cipariem

Identifikators	DF.PM-03
Mērķis	
Funkcija paredzēta četrzīmju paroles ģenerēšanai	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs, Reklāmas ievietotājs	
Datu ievade	
Nav	
Datu apstrāde	
Ar izlīdzinošu funkciju Math.random() tiek ģenerēts skaitlis no 1000 līdz 9999, kurš tiek salīdzināts ar visu paroles sarakstu izveidotiem pasākumiem. Ģenerējam skaitļus, līdz tam brīdim, kamēr iegūsim unikālu	
Datu izvade	
Fukncija izvada unikālu četrzīmju skaitli	
Kļūdu paziņojumi	
Nav	

2.3.2.3 Pievienoties pasākumam

Identifikators	DF.PM-04
Mērķis	
Funkcija paredzēta tam, lai lietotājs varētu pievienoties pasākumam	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs	
Datu ievade	
Funkcija saņem lietotāja un pasākuma identifikatorus	
Datu apstrāde	
Nospiežot taustiņu "Pievienoties", tiek ierakstīta datu bāzē saikne starp Pasākumu un Lietotāju	
Datu izvade	
Kad lietotājs pievienojas pasākumam, taustiņš "Pievienoties" tiek nomainīts uz taustiņu "Atteikties no pasākuma"	
Kļūdu paziņojumi	
Nav	

2.3.2.4 Atteikties no pasākuma

Identifikators	DF.PM-05
Mērķis	
Funkcija paredzēta tam, lai lietotājs varētu atteikties no pasākuma	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs	
Datu ievade	
Funkcija saņem lietotāja un pasākuma identifikatorus	
Datu apstrāde	
Nospiežot taustiņu "Atteikties no pasākuma" datu bāzē tiek dzēsta saikne starp lietotāju un pasākumu	
Datu izvade	
Kad lietotājs atsakās no pasākuma, taustiņš "Atteikties no pasākuma", tiek nomainīts uz taustiņu "Pievienoties"	
Kļūdu paziņojumi	
Nav	

2.3.2.5 Dzēst izveidoto pasākumu

Identifikators	DF.PM-06
Mērķis	
Funkcija paredzēta tam, lai dzēstu izveidoto pasākumu	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs, Reklāmas ievietotājs	
Datu ievade	
Funkcija saņem lietotāja un pasākuma identifikatorus	
Datu apstrāde	
Nospiežot taustiņu "Dzēst izveidoto pasākumu", tiek dzēsts Pasākums un visas saiknes ar datu bāzi	
Datu izvade	
Kad lietotājs dzēš pasākumu, tas pazūd no lietotnes	
Kļūdu paziņojumi	
Nav	

2.3.2.6 Pievienoties pasākumam ar paroli

Identifikators	DF.PM-07
Mērķis	
Funkcija paredzēta tam, lai lietotājs varētu pievienoties privātajam pasākumam, ievadot četrzīmju paroli	
Lietotāju grupas	
Reģistrēts lietotājs, Priviliģēts lietotājs	
Datu ievade	
Funkcija saņem četrzīmju paroli	
Datu apstrāde	
Ievadot 4 ciparus laukā, lietotājs automātiski nonāk slēgtajā lapā, kur var apskatīt pasākumus, kur ar DF.PM-04 funkcijas palīdzību viņš var pievienoties šim pasākumam. Ja parole neatbilst nevienam pasākumam, tad paziņo par kļūdu №1	
Datu izvade	
Veiksmīgi ievadot paroli, tiek atvērts pasākuma detalizēts logs	
Kļūdu paziņojumi	
1. "Ievadīta parole ir nepareiza!"	

2.3.3 Reklāmas modulis

Reklāmas modulis glabā visas funkcijas, kuras ir saistītas ar lietotnes reklāmu. Visas šī moduļa funkcijas paredzētas vienīgi Reklāmas ievetotājam.

2.3.3.1 Izveidot reklāmu

Identifikators	DF.RM-01
Mērķis	
Funkcija paredzēta, lai izveidotu jaunu reklāmu	
Lietotāju grupas	
Reklāmas ievietotājs	
Datu ievade	
Reklāmas izveidošanai nepieciešams aizpildīt sekojošus laukus: • Ad title (Nosaukums) – obligāts ievades lauks • Ad location (Vieta) – obligāts ievades lauks • Ad starting date (Sākuma datums) – obligāta izvēle kalendārā • Ad ending date (Beigu datums) – obligāta izvēle kalendārā • Ad description (Apraksts)– obligāts ievades lauks • Image (Attēls)– obligāts ievades lauks	
Datu apstrāde	

Nospiežot pogu “Izveidot reklāmas publikāciju” notiek sekojošais:
• Pārbauda, vai
  • Nosaukuma lauks nav tukšs,
  • Norises vietas lauks nav tukšs,
  • Sākuma un Beigu datumam jābūt izvēlētam no kalendāra,
  • Apraksta lauks nav tukšs,
  • Attēla laukam jābūt izvēlētam.
  Ja nē, tad paziņo par kļūdu №1 un tiek izcelts nepareizi ievadītais lauks
• Tad publikācija tiek pievienota datu bāzei
• Noslēgumā publikācija tiek pievienota Firebase mākoņu glabātuvē	

Datu izvade
Reklāmas publikācija ir redzama visiem lietotājiem galvenajā lapā
Kļūdu paziņojumi
1. “Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!”

2.3.3.2 Dzēst reklāmu

Identifikators	DF.RM-02
Mērķis	
Funkcija paredzēta tam, lai dzēstu reklāmas publikāciju	
Lietotāju grupas	
Reklāmas ievietotājs	
Datu ievade	
Funkcija saņem reklāmas publikācijas identifikatoru	
Datu apstrāde	
Nospiežot taustiņu "Dzēst reklāmas publikāciju", reklāmas publikācija tiek dzēsta no datu bāzes un Firebase mākoņu glabātuves	
Datu izvade	
Reklāmas publikācija nav redzama lietotājiem galvenajā lapā	
Kļūdu paziņojumi	
Nav	

2.3.3.3 Apstiprināt reklāmas izvietošanu

Identifikators	DF.RM-03
Mērķis	
Funkcija paredzēta, lai aizsūtītu administratoram vaicājumu par reklāmas ievietošanu	
Lietotāju grupas	
Reklāmas ievietotājs	
Datu ievade	
Funkcija saņem reklāmas publikācijas identifikatoru	
Datu apstrāde	
Nospiežot taustiņu "Apstiprināt reklāmu", reklāmas statuss mainās uz "Gaida apstiprinājumu", līdz tam brīdim, kamēr administrators apstiprinās doto publikāciju	
Datu izvade	
Reklāmas publikācija administratoram tiek atspoguļota rīkjoslā, līdz tam brīdim, kamēr viņš to apstiprinās vai noraidīs	
Kļūdu paziņojumi	
Nav	

2.3.4 Administrēšanas modulis

Administrēšanas modulis glabā lietotnes pārvaldīšanas funkcijas. Funkcijas paredzētas tikai Administratoram un nodrošina visas nepieciešamas funkcijas noteiktai grupai lietotnes pārvaldīšanai.

2.3.4.1 Dzēst lietotāju

Identifikators	DF.AD-03
Mērķis	
Funkcija paredzēta tam, lai dzēstu lietotāju	
Lietotāju grupas	
Administrators	
Datu ievade	
Funkcija saņem lietotāja identifikatoru	
Datu apstrāde	
Nospiežot taustiņu "Dzēst lietotāju", lietotājs un visi lietotāja izveidotie pasākumi tiek dzēsti no datu bāzes	
Datu izvade	
Kad lietotājs ir dzēsts, visi pasākumi, kurus viņš ir izveidojis, tiek izdzēsti no lietotnes	
Kļūdu paziņojumi	
Nav	

2.3.4.2 Rediģēt lietotāju

Identifikators	DF.AD-04
Mērķis	
Funkcija paredzēta lietotāja datu maiņai	
Lietotāju grupas	
Administrators	
Datu ievade	
Lietotāja maiņai nepieciešams aizpildīt sekojošus laukus • Vārds • Uzvārds • Lietotāja tips	
Datu apstrāde	
• Visi lauki tiek pārbaudīti, • Vārda lauks nav tukšs, • Uzvārda lauks nav tukšs, Ja kāds lauks ir aizpildīts neprecīzi, tad paziņo par kļūdu №1 un tiek izcelts nepareizi ievadītais lauks Nospiežot taustiņu: "Mainīt lietotāja datus", lietotāja dati tiks nomainīti datu bāzē un Firebase mākoņu glabātuvē	
Datu izvade	
Kad lietotājs ir nomainīts, visur, kur lietotājs bija redzēts līdz šim brīdim, tiks atspoguļoti viņa rediģētie dati	
Kļūdu paziņojumi	
1. "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!"	

2.3.4.3 Rediģēt pasākumu

Identifikators	DF.AD-05
Mērķis	
Funkcija paredzēta jau izveidota pasākuma datu maiņai	
Lietotāju grupas	
Administrators	
Datu ievade	
Pasākumu rediģēšanai, nepieciešams mainīts vismaz vienu no sekojošiem laukiem: • Event name (Nosaukums) • Location (Vieta) • Starting date (Sākuma datums) • Ending date (Beigu datums) • Description (Apraksts) • Minimal participants amount (Minimālais dalībnieku skaits) • Maximal participants amount (Maksimālais dalībnieku skaits) • Image (Attēls) • Event type (Pasākuma tips) • Event password (Pasākuma parole)	
Datu apstrāde	

Nospiežot pogu "Rediģēt pasākumu" notiek sekojošais:
• Pārbauda, vai Pasākuma nosaukuma lauks nav tukšs,
  • Norises vietas lauks nav tukšs,
  • Sākuma un Beigu datumam jābūt izvēlētam no kalendāra,
  • Apraksta lauks nav tukšs,
  • Minimālo un Maksimālo dalībnieku skaitam jābūt norādītam skaitļos, un Maksimālo dalībniekiu skaitlim jābūt lielākam par Minimālo dalībnieku skaitu,
  • Attēla laukam jābūt izvēlētam,
  • pasākuma tipa lauks ir izvēlēts.
  Ja lauki ir aizpildīti nepareizi, tad paziņo par kļūdu №1 un tiek izcelts nepareizi ievadītais lauks
• Ja pasākuma tips ir – private, tad parādās lauks pasākuma paroles maiņai
• Pasākuma dati tiek mainīti datu bāzē	

• Tāpat izmaiņas tiek ierakstītas Firebase mākoņu glabātuvē
Datu izvade
Rediģētais pasākums lietotājiem tiek atspoguļots lietotnē
Kļūdu paziņojumi
1. “Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!”

2.3.4.4 Dzēst pasākumu

Identifikators	DF.AD-03
Mērķis	
Funkcija paredzēta tam, lai dzēstu pasākumu	
Lietotāju grupas	
Administrators	
Datu ievade	
Funkcija saņem pasākuma identifikatoru.	
Datu apstrāde	
Nospiežot taustiņu “Dzēst pasākumu” – Pasākums un visas saiknes starp lietotāju un šo pasākumu tiek dzēstas no datu bāzes – Pasākums tiek dzēsts no Firebase mākoņu glabātuves	
Datu izvade	
Pasākums vairs netiek nekur atspoguļots	
Kļūdu paziņojumi	
Nav	

2.4 Nefunkcionālās prasības

2.4.1 Veiktspējas prasības

Sistēmai jānodrošina gandrīz momentālu datu apmaiņu starp lietotājiem, ar pieļaujamo 3 sekunžu aizkavēšanos. Lietotājam jābūt iespējai strādāt tiešsaistes un bezsaistes režīmā. Ja lietotājs ir bezsaistes režīmā, tad līdz ko viņš pievienosies interneta savienojumam, visi viņa dati atspoguļosies arī citiem lietotājiem.

2.4.2 Drošības prasības

Visām parolēm šifrētā veidā jāglabājas datu bāzē un Firebase mākoņu glabātuvē. Lietotājiem jābūt skaidri sadalītiem grupās un jābūt piekļuvei tikai pie tām funkcijām, kuras ir paredzētas noteiktai grupai, šādā veidā Administratora funkcijām jābūt piekļuvei tikai Administratoram.

2.4.3 Piekļuves prasības

Sistēmai jābūt piekļuvei jebkurā diennakts laikā, 24 stundas diennaktī, 7 dienas nedēļā. Īpaši skolas brīvlaikā, svētku dienās un brīvdienās. Ja sistēmas uzlabošanai nepieciešams pārtraukums lietotnē, tad lietotni var atspējot dienās, kad svētku dienas un brīvdienas nav tuvumā, un veikt atjaunošanu no rīta.

2.4.4 Aparatūras ierobežojumi

Mobilai lietotnei ir jābūt Android operētājsistēmai un tās versijai ir jābūt vienādai vai jaunākai par Android 5.0 *(Lollipop)* . Android ierīcei jābūt piekļuvei internetam, lai sinhronizētos ar serveri. Ja nav interneta savienojuma, tad lietotne glabā visus datus un sūta tos serverim līdz tam brīdim, kad parādīsies savienojums ar internetu.

2.4.5 Uzturamības prasības

Sistēmas kodam jābūt labi nokomentētam, lai izstrādātāja maiņas gadījumā, jauns izstrādātājs spētu ātri visu saprast un sākt darbu ar lietotni. Katrai lietotnes versijai jābūt saglabātai BitBucket repozitorijā, lai nepieciešamības gadījumā būtu iespēja pāriet uz lietotnes vecāko versiju.

3 PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

3.1 Ievads

3.1.1 Nolūks

Programmas projektējuma apraksts (PPA) paredzēts Android lietotnes projektējuma aprakstīšanai un programmatūras prasību specifikācijas izvirzīto prasību realizēšanai. Tas ir paredzēts izstrādātājiem, atvieglojot projekta funkciju implementēšanu, projekta analīzi un lēmumu pieņemšanu tā laikā.

3.2 Dekompozīcijas apraksts

3.2.1 Ievads

Lietotnes pamata struktūrai tika izmantota Android lietotnes standarta struktūra. Lietotāju saskarņu visas klases manto AppCompatActivity vai Fragment klases. Sarakstiem tika izmantotas adapteru klases, kuras ListAdapter klasi..

3.2.2 Moduļu dekompozīcija

3.2.2.1 Klase User

Klase lietotāja objektu izveidošanai.

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

Mainīgie:

Tips	Mainīgais	Apraksts
Long	uid	Identifikators
String	firstName	Vārds
String	lastName	Uzvārds
String	type	Lietotāja privilēģija
String	photoUri	Links uz fotogrāfijām
Float	rating	Atzīme

3.2.2.2 Klase UserDao

Interfeiss Data Access Object (Objekta piekļuve datiem) paredzēts vaicājumu glabāšanai lietotāju tabulās datu bāzē.

Metodes:

Tips	Metode	Apraksts
LiveData <List<String>>	getAllUsers()	Visu lietotāju saraksts, kurš tiek atjaunots uzreiz, kad parādās jauns lietotājs
long	addUser(User user)	Funkcija jauna lietotāja ierakstīšanai datu bāzē
int	update(User user)	Funkcija lietotāja atjaunošanai datu bāzē
void	deleteUser(User user)	Funkcija lietotāja dzēšanai no datu bāzes
User	getUserByUid(String uid)	Funkcija, lai saņemtu lietotāju pēc identifikatora no datu bāzes
void	insertAllUsers(User… users)	Funkcija, lai varētu pievienot uzreiz vairākus lieotāju
User	getUserByUidAndPassword (String uid, String password)	Funkcija, lai atrastu lietotāju pēc ievadīta identifikatora un paroles

3.2.2.3 Klase UserViewModel extends AndroidViewModel

Datu glabāšanas klase lietotnes pastāvēšanas cikla laikā. Dati nav piesaistīti noteiktai Activity. Objektu tipa glabāšanai tiek izmantots LiveData, kuri atjaunojas, kad notiek jebkādas izmaiņas tabulā datu bāzē. Klase paplašina AnroidViewModel klasi no Android bibliotēkas.

Mainīgie:

Tips	Mainīgais	Apraksts
LiveData<List<User>>	usersList	Visu lietotāju saraksts, kurš tiek atjaunots uzreiz, kad parādās jauns lietotājs
AppDatabase	appDatabase	AppDatabase klases objekts

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

3.2.2.4 Klase Event

Klase pasākuma objektu izveidošanai.

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

Mainīgie:

Tips	Mainīgais	Apraksts
Long	uid	Identifikators
String	name	Nosaukums
String	description	Pasākuma apraksts
String	location	Pasākuma norises vieta
String	imageName	Pasākuma bildes nosaukums
Boolean	viewed	Statuss vai pasākums tika apskatīts, vai nē
String	creator	Pasākuma izveidotājs
String	createdTime	Pasākuma izveidošanas datums
String	beginDate	Pasākuma sākuma datums
String	finishDate	Pasākuma beigu datums
String	numberOfParticipantsMin	Minimālais dalībnieku skaits
String	numberOfParticipantsMax	Maksimālais dalībnieku skaits
String	type	Pasākuma tips
String	password	Slēgta pasākuma(private) parole

3.2.2.5 Klase EventDao

Data Access Object (Objekta piekļuve datiem) interfeiss paredzēts pasākumu tabulas vaicājumu glabāšanai datu bāzē.

Metodes:

Tips	Metode	Apraksts
LiveData <List<String>>	getAllPublicEvents()	Visu atvērto pasākumu saraksts, kurš tiek atjaunots uzreiz, kad parādās jauns pasākums

LiveData <List<String>>	getAllEvents	Visu pasākumu saraksts (atvērto un slēgto), kurš paziņo, kad tiks izveidots jauns pasākums
int	countEvents()	Nolasa visus pasākumus tabulā
void	insertAllEvents(Events… event)	Pievienot vairākus pasākumus vienlaicīgi
long	insertEvent(Event event)	Pievienot pasākumus dau bāzē
void	delete(Event event)	Dzēst pasākumu no datu bāzes
Event	getEventById(long id)	Atrast pasākumu pēc identifikatora
Event	getEventByPassword(String password)	Atrast slēgto pasākumu pēc paroles
void	setEventViewed(boolean status, long eventId)	Atzīmēt pasākumu kā apskatītu
void	updateEvent(Event event)	Atjaunot pasākumu datu bāzē
LiveData <List<Event>>	getAllNotViewedEvents()	Saņemt visus neapskatītos pasākumus (Saraksts, kurš tiek atjaunots, kad datu bāzē tie ievietots jauns pasākums)
LiveData <List<Event>>	getEventsForUser(String userId)	Saņemt visus pasākumus, ar kuriem jebkādā veidā ir saistīts dotais lietotājs (Saraksts, kurš automātiski tiek atjaunots, kad datu bāzē tiek izveidots jauns pasākums)
LiveData <List<Event>>	getJoinedEventForUser(String userId)	Saņemt pasākumus, kuriem dotais lietotājs ir pievienojies (Saraksts, kurš automātiski tiek atjaunots, kad datu bāzē tiek izveidots jauns pasākums)
List<Event>	getPrivateEventsForUser(String userId)	Saņemt visus slēgtos pasākumus, ar kuriem ir saistīts lietotājs
List<Event>	getJoinedPrivateEventForUser(String userUid)	Saņemt visus slēgtos pasākumus, kuriem ir pievienojies lietotājs

List<String>	getAllEventPassword()	Saņemt visu slēgto pasākumu paroles
List<Event>	getListOfJoinedEventForUser(String userUid)	Saņemt visu pasākumu sarakstu, kuriem lietotājs ir pievienojies
List<Event>	getListCreatedEvents(String userId)	Saņemt visu lietotāju izveidoto pasākumu sarakstu

3.2.2.6 Klase EventParticipants

Klase objekta izveidošanai, saiknes saglabāšanai starp lietotāju un pasākumu (saistības: daudzi - pret daudziem)

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

Mainīgie:

Tips	Mainīgais	Apraksts
Long	eventId	Pasākuma identifikators
String	userUid	Lietotāja identifikators

3.2.2.7 Klase EventParticipantsDao

Interfeiss Data Access Object (Datu piekļuves objekts) paredzēts vaicājumu glabāšanai datu bāzē starptabulā starp lietotāju un pasākumu.

Metodes:

Tips	Metode	Apraksts
void	joinEvent(EventParticipants eventParticipants)	Metode saiknes veidošanai starp lietotāju un pasākumu
void	deleteJoinedEvent(EventParticipants eventParticipants)	Metode saiknes dzēšanai starp lietotāju un pasākumu

3.2.2.8 Klase EventViewModel extends AndroidViewModel

Datu glabāšanas klase lietotnes cikla pastāvēšanas laikā. Dati nav piesaistīti pie noteiktas Activity. Objektu glabāšanai tiek izmantots LiveData tips, kurš tiek atjaunots, kad tiek veiktas jebkādas izmaiņas datu bāzē. Klase paplašina AnroidViewModel klasi no Android bibliotēkas.

Mainīgie:

Tips	Mainīgais	Apraksts
LiveData<List<Event>>	eventsList	Atvērto pasākumu saraksts (Saraksts automātiski atjaunojas, kad tiek veiktas jebkādas izmaiņas par pasākumu datu bāzē)
LiveData<List<Event>>	allEventsList	Visu pasākumu saraksts (Saraksts, kurš automātiski atjaunojas, kad tiek pievienots jauns pasākums datu bāzē)
LiveData<List<Event>>	notViewedEventsList	Neapskatītu pasākumu saraksts (Saraksts, kurš automātiski atjaunojas, kad tiek pievienots jauns pasākums datu bāzē)
LiveData<List<Event>>	eventsForCurrentUser	Pasākumu saraksts konkrētam lietotājam (Saraksts, kurš automātiski atjaunojas, kad tiek pievienots jauns pasākums datu bāzē)
LiveData<List<Event>>	joinedEventsForCurrentUser	Pasākumu saraksts, kuram ir pievienojies dotais lietotājs (Saraksts, kurš automātiski atjaunojas, kad tiek pievienots jauns pasākums datu bāzē)
AppDatabase	appDatabase	AppDatabase klases objekts

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

3.2.2.9 Klase Ad

Klase reklāmas objektu izveidošanai.

Metodes:

Klase satur visu tās mainīgo uzstādītājmetodes un saņēmējmetodes.

Mainīgie:

Tips	Mainīgais	Apraksts
Long	uid	Identifikators
String	name	Nosaukums
String	description	Reklāmas apraksts
String	location	Pasākuma norises vieta
String	imageName	Reklāmas bildes nosaukums
Boolean	viewed	Statuss vai pasākums tika apskatīts, vai nē
String	creator	Reklāmas veidotājs
String	beginDate	Reklāmas sākuma datums
String	finishDate	Reklāmas beigu datums
boolean	approved	Vai reklāma ir apstiprināta
int	views	Reklāmas apskatīšanas skaits

3.2.2.10 AdDao

Interfeiss Data Access Object (Datu piekļuves objekts), kura paredzēts vaicājumu glabāšanai reklāmas tabulā datu bāzē.

Metodes:

Tips	Metode	Apraksts
List<Ad>	getAllAds()	Saņemt visu reklāmas sludinājumu sarakstu
long	addNewAd(Ad ad)	Pievienot jaunu reklāmas sludinājumu

int	update(Ad ad)	Atjaunot eksistējošu reklāmas sludinājumu
void	deleteAd(Ad ad)	Dzēst reklāmas sludinājumu
Ad	getAdById(String id)	Atrast reklāmas sludinājumu pēc identifikatora

3.2.2.11 Klase AppDatabase extends RoomDatabase

SQLite klase datu bāzes izveidošanai, izmantojot Room Persistence Library.

Mainīgie:

Tips	Mainīgais	Apraksts
AppDatabase	INSTANCE	Datu bāzes izveidots objekts
EventDao	eventDao()	EventDao abstraktās klases vienība
UserDao	userDao()	UserDao abstraktās klases vienība
AdDao	adDao()	AdDao abstraktās klases vienība
EventParticipantsDao	eventParticipantsDao()	EventParticipantsDao abstraktās klases vienība

Metodes:

Tips	Metode	Apraksts
AppDatabase	getAppDatabase	Ja datu bāze ir izveidota, tad saņemt tai linku, ja nav, tad izveidot datu bāzi
void	destroyInstance()	Dzēš datu bāzes linku
void	upsertUser()	Tā ka SQLite neatbalsta Upsert funkcijas, tad šī metode pārbauda vai objekts ir datu bāzē, ja ir, tad to vienkārši atjauno, ja nav, tad pievieno tabulai datu bāzē
void	upsertEvent()	

3.2.2.12 Klase Card

Klase pasākuma interaktīvas kartiņas izveidošanai

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	profileImageView	Vieta profila bildei
TextView	eventNameTextView	Lauks pasākuma nosaukumam
TextView	eventLocationTextView	Lauks, kur tiek atspoguļota pasākuma norises vieta
TextView	numberOfParticipantsTextView	Dalībnieku skaita atspoguļošana
TextView	eventCreatorTextView	Pasākuma autors
Event	mEvent	Pasākuma klases objekts kartiņas aizpildīšanai
Context	mContext	Objekts lietotnes funkciju piekļuvei
SwipePlaceHolderView	mSwipeView	Noteiktas kartiņas objekts no SwipePlaceHolder bibliotēkas

Metodes:

Tips	Metode	Apraksts
void	onResolved()	Metode, kura tiek izsaukta kartiņas atrādīšanas gadījumā
void	onSwipeOut()	Metode, kura tiek izsaukta, novirzot kartiņu pa kreisi
void	onSwpieIn()	Metode, kura tiek izsaukta, novirzot kartiņu pa labi
void	onClick()	Metode, kura darbojas, ja nospiež uz kartiņu

3.2.2.13 Klase MainActivity

Pirmā aktivitāte, kuru lietotājs redz savā lietotnē. Šajā ekrānā lietotājs var ieiet savā lietotnē caur Facebook, Google vai izvēlēties ieejas veidu, ievadot lietotajvārdu un paroli.

Mainīgie:

Tips	Mainīgais	Apraksts
CallbackManager	callbackManager	Menedžeris, atbildes reaģēšanai uz atbildi no FacebookAPI
FirebaseDatabase	database	Links uz objektu, kurš atrodas Firebase mākoņu glabātuvē
DatabaseReference	eventsReference	Links uz pasākumiem, kuri atrodas Firebase mākoņu glabātuvē
DatabaseReference	usersReference	Links uz lietotājiem, kuri atrodas Firebase mākoņu glabātuvē

Metodes:

Tips	Metode	Apraksts
ValueEventListener	addListenerForSingleValueEvent	Datu lejupielādes funkcija no mākoņu glabātuves uz lokālo datu bāzi
void	onFacebookLogin	Ieeja caur Facebook API
void	onGoogleLogin	Ieeja caur Google API
void	loginViaUsernameAndPassword	Atvērt lapu lietotājvārda un paroles ievadīšanai

3.2.2.14 Klase LoginActivity

Šī klase paredzēta laukiem, kur tiks ierakstīts lietotājvārds un parole, lai lietotājam būtu iespēja autorizēties lietotnē.

Mainīgie:

Tips	Mainīgais	Apraksts
EditText	username	Lauks lietotājvārda ievadīšanai
EditText	password	Lauks paroles ievadīšanai
TextView	registrationButton	Teksts, uz kuru nospiežot, tiek atvērta reģistrācijas lapa
Button	signInButton	Autorizācijas taustiņš

Metodes:

Tips	Metode	Apraksts
void	registrationButton: onClick (View view)	Reaģēšana uz taustiņa nospiešanu: tiek atvērta reģistrācijas lapa
void	signInButton: onClick(View view)	Reaģēšana uz taustiņa nospiešanu: tiek atvērta lietotnes pamata daļa
boolean	fieldsValidation()	Funkcija lauku pārbaudei

3.2.2.15 Klase RegistrationActivity

Klase paredzēta ekrāna atspoguļošanai, kur lietotājs var reģistrēt jaunu kontu**:**

Tips	Mainīgais	Apraksts
EditText	fName	Vārds
EditText	lNmae	Uzvārds
EditText	username	Lietotājvārds
EditText	password	Parole
Button	createNewAccountButton	Taustiņš jauna profila izveidošanai

Metodes:

Tips	Metode	Apraksts
boolean	fieldsValidation()	Funkcija lauku pārbaudei
void	createNewAccountButton: onClick()	Funkcija taustiņa reaģēšanai, kad tiek pievienots jauns lietotājs
String	AESCrypt.encrypt()	Funkcija paroles šifrēšanai

3.2.2.16 Klase AESCrypt

Klase paroles šifrēšanai.

Mainīgie:

Tips	Mainīgais	Apraksts
String	ALGORITHM	Rinda, "AES" algoritma nosaukuma glabāšanai
String	KEY	Mainīgais šifrēšanas atslēgas glabāšanai

Metodes:

Tips	Metode	Apraksts
String	encrypt()	Paroles šifrēšanas funkcija
String	decrypt()	Paroles atšifrēšanas funkcija

3.2.2.17 Klase HomeActivity

Galvenā lietotnes Aktivitāte (Activity), kura atbild par fragmentu atspoguļošanu un navigācijas darbību.

Mainīgie:

Tips	Mainīgais	Apraksts
Fragment	homeFragment	Galvenā ekrāna fragments
Fragment	eventFragment	Pasākumu ekrāna fragments

Fragment	addEventFragment	Fragments jauna pasākuma pievienošanai
Fragment	myActivityFragment	Lietotāja aktivitātes fragments
Fragment	myProfileFragment	Informācijas fragments par lietotāju
FragmentTransaction	ft	Klase darbam ar fragmentiem
BottomNavigationViewEx	bottomNavigation	Klase darbam ar apakšējo navigāciju no ārējās bibliotēkas
int	selectedMenuPosition	Klase izvēlētā izvēlnes elementa izsekošanai
LockableViewPager	viewPager	Klase fragmentu izvietošanai. No ārējās bibliotēkas ar pāršķiršanas iespēju izslēgšanu starp fragmentiem
DrawerLayout	mDrawerLayout	Klase sāna izvēlnes atvēršanai, ar administratora paneli
FirebaseDatabase	database	Lins uz objektu, kurš atrodas Firebase mākoņu glabātuvē
DatabaseReference	eventRef	Links uz pasākumiem, kuri atrodas Firebase mākoņu glabātuvē
DatabaseReference	userRef	Links uz lietotājiem, kuri atrodas Firabase mākoņu glabātuvē
List<Event>	allEventsInFirebase	Visu pasākumu saraksts no datu bāzes
User	loggedInUser	Objekts, kurš glabā tekošu lietotāju

Metodes:

Tips	Metode	Apraksts
boolean	onNavigationItemSelected (MenuItem menuItem)	Reakcija uz taustiņa nospiešanu apkšējā izvēlnē
void	onChildAdded (DataSnapshot dataSnapshot)	Reakcija uz ieraksta pievienošanu Firebase mākoņu glabātuvē
void	onChildRemoved (DataSnapshot dataSnapshot)	Reakcija uz ieraksta izdzēšanu Firebase mākoņu glabātuvē

void	onPageSelected(int position)	Reakcija uz lappuses izvēli no izvēlnes
void	setupViewPager (LockableViewPager viewPager)	Uzstādījumi darbam ar fragmentiem
void	openDrawer()	Sāna izvēlnes atvēršana

3.2.2.18 Klase EventDetailsActivity

Pasākuma informācijas atspoguļošanas ekrāns

Mainīgie:

Tips	Mainīgais	Apraksts
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventDescription	Pasākuma apraksts
TextView	eventDate	Pasākuma norises datums
TextView	eventNumberOfParticiapants	Dalībnieku skaits
Event	selectedEvent	Izvēlētais pasākums
Button	joinEventButton	Taustiņš, lai pievienotos pasākumam
Button	leaveEventButton	Taustiņš, lai atteiktos no pasākuma
Button	cancelEventButton	Pasākuma atcelšanas taustiņš

Metodes:

Tips	Metode	Apraksts
void	joinEventButton: onClick()	Nospiežot taustiņu, datu bāzē tiek ierakstīta saikne starp lietotāju un pasākumu
void	leaveEventButton: onClick()	Nospiežot, no datu bāzes tiek dzēsta saikne starp pasākumu un lietotāju
void	cancelEventButton: onClick()	Nospiežot, no datu bāzes tiek dzēsts pasākums

3.2.2.19 Klase CreateEventTemporaryActivity

Jauna pasākuma izveidošanas ekrāns.

Mainīgie:

Tips	Mainīgais	Apraksts
String	EVENT_TYPE_PUBLIC	Konstante
String	EVENT_TYPE_PRIVATE	Konstante
String	eventType	Rinda kurā glabājas pasākuma tips
EditText	eventName	Lauks pasākuma nosaukuma aizpildīšanai
EditText	eventLocation	Lauks norises vietas aizpildīšanai
EditText	eventDescription	Lauks apraksta aizpildīšanai
EditText	eventMinimalNumberOfParticipants	Aizpildīšanas lauks minimālo dalībnieku skaitam
EditText	eventMaximalNumberOfParticipants	Aizpildīšanas lauks maksimālo dalībnieku skaitam
EditText	dateFromEditText	Aizpildīšanas lauks rīkotā pasākuma sākuma datumam (nospiežot, atvēras kalendārs)
EditText	dateEndsEditText	Aizpildīšanas lauks, rīkotā pasākuma beigu datumam (nospiežot, atvēras kalendārs)
ToggleButton	publicEventButton	Taustiņš pasākuma tipa izvēlei
ToggleButton	privateEventButton	Taustiņš pasākuma tipa izvēlei

Spinner	imageSpinner	Izvēlne bildes izvēlēšanai
ImageView	createEventButton	Taustiņš jauna pasākuma izveidošanai
Calendar	myCalendar	Kalendārs pareiza datuma formāta izvēlei
DatePickerDialog	dateStart	Klase, kura ļauj lietotājam izvēlēties pasākuma sākuma datumu no kalendāra
DatePickerDialog	dateEnds	Klase, kura ļauj lietotājam izvēlēties pasākuma beigu datumu no kalendāra

Metodes:

Tips	Metode	Apraksts
String	generatePIN()	Funkcija četrzīmju unikālas paroles ģenerēšanai
void	CustomDialog.showDialog(Context context, String password)	Funkcija dialoga lodziņa atspoguļošanai ar paroli
void	createEventButton: onClick()	Funkcija reaģēšanai uz taustiņu, jauna pasākuma izveidošanai

3.2.2.20 Klase BottomNavigationViewHelper

Klase standarta modificēšanai ButtomNavigationMenuView. Tā kā standarta navigācijai ir nepiemērota animācija un MenuItems obligāti jāsatur tekstu, pretējā gadījumā ikonas ir izvietotas šķībi, tika izveidota klase BottomNavigationViewHelper, lai uzstādītu navigāciju tā, kā tas bija paredzēts.

Mainīgie:

Tips	Mainīgais	Apraksts
BottomNavigationMenuView	menuView	Tiek nodots links uz objektu BottomNavigationMenuView, kuru gribam modificēt

Metodes:

Tips	Metode	Apraksts
public	disableShiftMode (BottomNavigationMenuView view)	Funkcija, lai pārrakstītu standarta navigācijas animāciju
public	centerMenuIcon (BottomNavigationView view)	Funkcija, lai navigācijas ikonas atrastos centrā

3.2.2.21 Klase HomeFragment extends Fragment

Galvenais ekrāns, kuru lietotājs redz tikai tad, kad ir autorizējies lietotnē, tāpat šis ekrāns tiek atspoguļots, kad navigācijā ir izvēlēta pirmā opcija. Ekrāns satur horizontālo sarakstu ar reklāmu un vertikālo sarakstu ar visbiežāk apskatītajiem pasākumiem.

Mainīgie:

Tips	Mainīgais	Apraksts
List<Event>	publicEvent	Pasākuma nosaukums
ViewAdapter	adapterVerticalList	Adapteris, kurš ir atbildīgs par informācijas izvietošanu katra saraksta elementā
ImageView	adminPanelButton	Taustiņš, kurš ļauj nonākt Administrora panelī, ir redzama tikai Administrācijas tipa lietotājiem
ImageView	adContrulPanelButton	Taustiņš, kurš ļauj nonākt reklāmas panelī, redzama tikai Reklāmas

		ievietotāja tipa lietotājiem
List<Ad>	adsList	Reklāmas saraksts
RecyclerView	recyclerView	Lietotāja saskarnes komponents, kurš ļauj veidot pāršķirstamu sarakstu
LinearLayoutManager	horizontalLayoutManagaer	Atbild par komponentu pozicionēšanu RecyclerView
HorizontalRecyclerViewAdapter	adapter	Adapteris atbild par datu izvietošanu katra saraksta elementā
FirebaseDatabase	firebaseDatabase	Links uz Firebase datu mākoņu glabātuvi
ListView	publicEventsListView	Lietotāja interfeisa komponents, kurš ļauj veidot pāršķirstamu sarakstu
View	footer	Lietotāja interfeisa komponents, informācijas atspoguļošanai pēc saraksta pēdēja elementa

Metodes:

Tips	Metode	Apraksts
String	getLoggedUserType()	Saņemt tekoša lietotāja statusu
void	adminPanelButton: onClick(View view)	Atbilde uz taustiņa nospiešanu, Administratora paneļa atvēršanai
void	adControlPanelButton: onClick(View view)	Atbilde uz taustiņa nospiešanu, Reklāmas paneļa atvēršanai

void	onListItemClick(View view, int position)	Atbilde uz reklāmas sludinājuma nospiešanu. Tiek atvērts sludinājuma detalizēts saraksts

Klase ViewAdapter extends BaseAdapter implements ListAdapter

Adapteris atbild par datu izvietošanu katra saraksta elementā

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	eventImage	Pasākuma bilde
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventNumberOfParticipants	Pasākuma dalībnieku skaits
RelativeLayout	listItemContainer	Lietotāja saskarnes komponents, tiek izmantots atbildes saņemšanas izveidošanai, nospiežot uz saraksta elementu
TextView	creator	Izveidot pasākumu

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt saraksta elementa secības numuru
View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam saraksta elementu, kuru viņš redz ekrānā
void	onClick(View view)	Atbilde uz reklāmas sludinājuma nospiešanu. Tiek atvērts sludinājuma detalizēts saraksts

3.2.2.22 Klase EventsFragment extends Fragment

Ekrāns, kurā lietotājs interaktīvā veidā var apskatīt atvērtus pasākumus, pāršķirstot kartiņas ar pasākumiem.

Mainīgie:

Tips	Mainīgais	Apraksts
SwipePlaceHolderView	mSwipeView	Noteiktas kartiņas objekts no SwipePlaceHolder bibliotēkas
Context	mContext	Objekts lietotnes funkciju piekļuvei
ImageButton	acceptButton	Taustiņš, lai pievienotos pasākumam
ImageButton	rejectButton	Taustiņš, lai atteiktos no pasākuma
ImageView	listButton	Taustiņš, lai pārslēgtu uz sarakstu un otrādi
EventsViewModel	eventViewModel	Datu glabāšanas klase lietotnes cikla pastāvēšanas laikā

Metodes:

Tips	Metode	Apraksts
void	listButton: onClick(View view)	Pārslēgšana uz sarakstu un otrādi
void	rejectButton: onClick(View view)	Atbilde uz taustiņa nospiešanu, lai atteiktos no pasākuma
void	acceptButton: onClick(View view)	Atbilde uz taustiņa nospiešanu, lai pievienotos pasākumam
void	onChanged(List<Event> events)	Atbilde uz izmaiņam Pasākuma datu bāzē

3.2.2.23 Klase EventsListFragment

Ekrāns, uz kura lietotājs var apskatīt atvērtus pasākumus saraksta veidā

Mainīgie:

Tips	Mainīgais	Apraksts
ListView	publicEventsListView	Lietotāja saskarnes komponents, kurš ļauj veidot pāršķirstamu sarakstu
ViewAdapter	adapter	Adapteris, kurš atbild par datu izvietošanu katra saraksta elementā
ArrayList<Event>	publicEvents	Saraksts ar visiem atvērtiem pasākumiem
EventsViewModel	eventsViewModel	Datu glabāšanas klase lietotnes pastāvēšanas cikla laikā
ImageView	listButton	Saraksta pārslēgšana uz interaktīvo režīmu
View	footer	Lietotāja interfeisa komponents, informācijas atspoguļošanai pēc saraksta pēdēja elementa

Metodes:

Tips	Metode	Apraksts
void	onChanged(List<Event> events)	Atbilde uz izmaiņam Pasākuma datu bāzē
void	listButton: onClick(View view)	Saraksta pārslēgšana uz interaktīvo režīmu

Klase ViewAdapter extends BaseAdapter implements ListAdapter

Adapteris atbild par datu izvietošanu katra saraksta lementā

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	eventImage	Pasākuma bilde
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventNumberOfParticipants	Pasākuma dalībnieku skaits
RelativeLayout	listItemContainer	Lietotāja saskarnes komponents, tiek izmantots atbildes saņemšanas izveidošanai, nospiežot uz saraksta elementu
TextView	creator	Izveidot pasākumu

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt saraksta elementa secības numuru
View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam saraksta elementu, kuru viņš redz ekrānā
void	onClick(View view)	Atbilde uz reklāmas sludinājuma nospiešanu. Tiek atvērts sludinājuma detalizēts saraksts

3.2.2.24 Klase MyProfileFragment

Ekrāns informācijas atspoguļošanai par lietotāju un atvērto pasākumu atspoguļošanu, kurus lietotājs ir pievienojis vai kurus ir izveidojis. Slēgtie pasākumi tiek atspoguļoti tikai uz šī ekrāna, ar kuriem lietotājs ir saistīts.

Mainīgie:

Tips	Mainīgais	Apraksts
ListView	publicEventsListView	Lietotāja saskarnes komponents, kurš ļauj veidot pāršķirstamu sarakstu
ViewAdapter	adapter	Adapteris atbild par datu izvietošanu katra saraksta elementā
ArrayList<Event>	publicEvents	Saraksts ar atvērtiem pasākumiem, kuri ir saistīti ar lietotāja datiem
ArrayList<Event>	createdEvents	Saraksts ar pasākumiem, kurus ir izveidojis lietotājs.
ArrayList<Event>	joinedEvents	Pasākumu saraksts, kuriem lietotājs ir pievienojies.
ToggleButton	openToggleButton	Taustiņš filtrēšanai vai nu Publisks, vai Privāts
ToggleButton	privateToggleButton	Taustiņš filtrēšanai vai nu Publisks, vai Privāts
ToggleButton	headerOpenToggleButton	Taustiņš filtrēšanai vai nu Publisks, vai Privāts
ToggleButton	headerPrivateToggleButton	Taustiņš filtrēšanai vai nu Publisks, vai Privāts
String	filter	Mainīgais filtra glabāšanai
View	footer	Lietotāja saskarnes komponents, informācijas atspoguļošanai pēc saraksta pēdēja elementa
View	header	Lietotāja saskarnes komponents informācijas atspoguļošanai pirms saraksta pirma elementa
View	header2	Lietotāja saskarnes komponents informācijas

		atspoguļošanai pirms saraksta pirma elementa
EventsViewModel	eventsViewModel	Klase, lai glabātu datus, kurām varat piekļūt jebkurā brīdī

Metodes:

Tips	Metode	Apraksts
void	setListByFilter(String filter)	Nofiltrēt pasākumu sarakstu
void	onCheckedChanged(CompoundButton compoundButton, Boolean b)	Reaģēt uz filtra vērtību

Klase ViewAdapter extends BaseAdapter implements ListAdapter

Adapteris atbild par datu izvietošanu katra saraksta elementā

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	eventImage	Pasākuma bilde
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventNumberOfParticipants	Pasākuma dalībnieku skaits
RelativeLayout	listItemContainer	Lietotāja saskarnes komponents, tiek izmantots atbildes saņemšanas izveidošanai, nospiežot uz saraksta elementu

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt elementa saraksta kārtas numuru

View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam sarakata elementu, kuru redz uz ekrāna
void	onClick(View view)	Atbilde uz saraksta elementa nospiešanu, tiek atvērts pasākuma detalizēts skats

3.2.2.25 Klase ViewPagerAdapter extends FragmentPagerAdapter

Klase (Adapteris) fragmentu pārvaldīšanai lietotnē.

Mainīgie:

Tips	Mainīgais	Apraksts
List<Fragment>	mFragmentList	Visu lietotnes fragmentu saraksts

Metodes:

Tips	Metode	Apraksts
Fragment	getItem(int position)	Saņemt fragmentu noteiktajā pozīcijā
int	getCount()	Atgriezt kopējo fragmentu skaitu
void	addFragment(Fragment fragment)	Pievienot jaunu fragmentu

3.2.2.26 Klase HorizontalRecyclerViewAdapter

Klase (adapteris) atbild par datu izvietošanu katrā elementā.

Mainīgie:

Tips	Mainīgais	Apraksts
List<String>	adPosts	Reklāmas publikāciju saraksts
LayoutInflater	mInflater	Klase View veidošanai no Layout datnes

ItemClickListener	mClickListener	Objekta nospiešanas klausītājs

Metodes:

Tips	Metode	Apraksts
ViewHolder	onCreateViewHolder (ViewGroup parent, int viewType)	Veido elementu no Layout datnes
void	onBindViewHolder (ViewHolder holder, int position)	Piepilda katru saraksta elementu ar datiem
int	getItemCount()	Atgriež elementu skaitu
Ad	getItem(int position)	Saņemt elementu no noteiktas pozīcijas
void	setClickListener (ItemClickListener itemClickListener)	Izveidot reakciju uz nospiešanu

3.2.2.27 Klase AdControlPanelActivity

Klase: Reklāmas paneļa pārvaldīšana – ekrāns, kur Reklāmas ievietotājs var pārvaldīt reklāmas sludinājumus

Mainīgie:

Tips	Mainīgais	Apraksts
List<Ad>	adsList	Reklāmas sludinājumu saraksts
ViewAdapter	adapter	Adapteris atbild par datu izvietošanu katra saraksta elementā
FloatingActionButton	createNewAd	Taustiņš jaunas publikācijas izveidošanai
FirebaseDatabase	database	Links uz Firebase datu bāzi

DatabaseReference	adsFirebase	Links uz reklāmas publikācijām Firebase mākoņu glabātuvē

Metodes:

Tips	Metode	Apraksts
void	createNewAd: onClick(View view)	Atbilde uz taustiņa nospiešanu jaunas reklāmas publikācijas izveidošanu
void	onDataChange (DataSnapshot dataSnapshot)	Reāģēšanas funkcija uz datu izmaiņu veikšanu Firebase mākoņu glabātuvē

Klase ViewAdapter extends BaseAdapter implements ListAdapter

Adapteris atbild par datu izvietošanu katra saraksta elementā.

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	eventImage	Pasākuma bilde
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventNumberOfParticipants	Pasākuma dalībnieku skaits
RelativeLayout	listItemContainer	Lietotāja saskarnes komponents, tiek izmantots atbildes saņemšanas izveidošanai, nospiežot uz saraksta elementu

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt elementa saraksta kārtas numuru

View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam saraksta elementu, kuru redz uz ekrāna
void	onClick(View view)	Atbilde uz saraksta elemena nospiešanu, tiek atvērts pasākuma detalizēts skats

3.2.2.28 Klase AddNewEventFragment

Ekrāns, kurā lietotājs var ievadīt četrzīmju paroli, lai apskatītu slēgto pasākumu, tāpat uz ekrāna ir liela poga "Izveidot jauno pasākumu".

Mainīgie:

Tips	Mainīgais	Apraksts
EditText	pinDigit1	Lauks slēgta pasākuma paroles ievadīšanai
EditText	pinDigit2	
EditText	pinDigit3	
EditText	pinDigit4	
Button	createNewEventButton	Taustiņš ekrāna atvēršanai, jauna pasākuma izveidošanai

Metodes:

Tips	Metode	Apraksts
void	onTextChanged (CharSequence s, int start,int before, int count)	Kad ievadīti 4 cipari, tiek pārbaudīta parole un ja tāds pasākums jau eksistē, tad atvēras nākamais ekrāns
Event	getEventByPassword (String password)	Atrod slēgto pasākumu pēc ievadītas paroles
void	createNewEventButton: onClick(View view)	Atbilde uz taustiņa nospiešanu, jauna pasākuma izveidošanai

3.2.2.29 Klase ChangeUserDataActivity

Ekrānam lietotāja datu rediģēšanai ir piekļuve tikai Administratoram.

Mainīgie:

Tips	Mainīgais	Apraksts
EditText	fName	Lietotāja vārds
EditText	lName	Lietotāja uzvārds
EditText	uid	Lietotāja identifikators
Spinner	userTypeSpinner	Izvēlne ar lietotāju tipu izvēli
Button	changeDataButton	Taustiņš lietotāja izmaiņu apstiprināšanai
ImageView	deleteUserButton	Taustiņš lietotāja dzēšanai
FirebaseDatabase	database	Links uz Firebase Database objektu
User	user	Objekts klases Lietotājs
String	userId	Lietotāja identifikators
Query	findEventsQuery	Vaicājums mākoņu glabātuvei pasākumu saņemšanai, kuri ir saistīti ar lietotāju

Metodes:

Tips	Metode	Apraksts
void	deleteEventsRelatedToUser()	Funkcija, kura dzēš visus pasākumus saistītus ar lietotāju, kuru dzēš ārā
void	deleteUser(User user)	Dzēst lietotāju
void	update(User user)	Atjaunot lietotāju

3.2.2.30 Klase CreateNewAdActivity

Ekrāns jaunu reklāmas publikāciju veidošanai, ir piekļuve tikai grupai Reklāmas ievietotājs.

Mainīgie:

Tips	Mainīgais	Apraksts
EditText	adName	Reklāmas nosaukums
EditText	adLocation	Reklāmas veidošanas vieta
EditText	adDescription	Apraksts
EditText	adFromEditText	Reklāmas veidošanas sākuma datumu
EditText	adEndsEditText	Reklāmas veidošanas beigu datums
Spinner	imageSpinner	Reklāmas bilde
Button	createAdButton	Taustiņš reklāmas izveidošanai
Calendar	myCalendar	Kalendārs pareiza datuma formāta izvēlei
DatePickerDialog	datePicker	Klase, kura ļauj lietotājam izvēlēties datumu no kalendāra

Metodes:

Tips	Metode	Apraksts
void	addNewAd(Ad ad)	Jaunas reklāmas publikācijas pievienošana

3.2.2.31 Klase Custom Dialog

Klase pielāgojama loga uzstādīšanai.

Mainīgie:

Tips	Mainīgais	Apraksts
TextView	firstDigit	Slēgta pasākuma paroles atspoguļošana
TextView	secondDigit	
TextView	thirdDigit	
TextView	fourthDigit	
Button	showEventDetailsButton	Taustiņš, lai atvērtu pasākuma detaļas

Button	editEventButton	Taustiņš, lai atvērtu pasākuma rediģēšanai
Button	deleteEventButton	Taustiņš pasākuma dzēšanai

Metodes:

Tips	Metode	Apraksts
void	showDialog (Context context, String password)	Parādīt slēgta pasākumu paroļu logu
void	showAdminEventControlPanelDialog (Context context, Event event)	Parādīt logu ar paneļa Administrators opcijām

3.2.2.32 Klase EditEventActivity

Klase Pasākuma rediģēšanai, piekļuve ir tikai Administratoram.

Mainīgie:

Tips	Mainīgais	Apraksts
String	EVENT_TYPE_PUBLIC	Konstante
String	EVENT_TYPE_PRIVATE	Konstante
String	eventType	Rinda, kurā glabājas pasākuma tips
EditText	eventName	Lauks pasākuma nosaukuma aizpildīšanai
EditText	eventLocation	Lauks norises vietas aizpildīšanai
EditText	eventDescription	Lauks apraksta aizpildīšanai
EditText	eventMinimalNumberOfParticipants	Aizpildīšanas lauks minimālo dalībnieku skaitam
EditText	eventMaximalNumberOfParticipants	Aizpildīšanas lauks maksimālo dalībnieku skaitam

EditText	dateFromEditText	Aizpildīšanas lauks rīkotā pasākuma sākuma datumam (nospiežot, atvēras kalendārs)
EditText	dateEndsEditText	Aizpildīšanas lauks rīkotā pasākuma beigu datumam (nospiežot, atvēras kalendārs)
ToggleButton	publicEventButton	Taustiņš pasākuma tipa izvēlei
ToggleButton	privateEventButton	Taustiņš pasākuma tipa izvēlei
Spinner	imageSpinner	Izvēlne bildes izvēlēšanai
Calendar	myCalendar	Kalendārs pareiza datuma formāta izvēlei
DatePickerDialog	dateStart	Klase, kura ļauj lietotājam izvēlēties pasākuma sākuma datumu no kalendāra
DatePickerDialog	dateEnds	Klase, kura ļauj lietotājam izvēlēties pasākuma beigu datumu no kalendāra
EditText	eventPassword	Pasākuma parole
Long	eventId	Pasākuma identifikators
Button	saveChangesButton	Taustiņš izmaiņu saglabāšanai

Metodes:

Tips	Metode	Apraksts
Event	getEventById(Event event)	Atrast pasākumus pēc identifikatora

void	saveChanges()	Saglabāt pasākuma izmaiņas
void	updateEvent(Event event)	Atjaunot pasākumu

3.2.2.33 Klase EventsControlPanelActivity

Klase pasākuma pārvaldīšanai, piekļuve ir tikai Administratoram.

Mainīgie:

Tips	Mainīgais	Apraksts
ListView	listView	Lietotāja saskarnes komponents, kurš ļauj veidot pāršķirstamu sarakstu
List<Event>	allEventsList	Visu pasākumu saraksts
ViewAdapter	adapter	Adapteris, kurš atbild par informācijas izvietošanu katra saraksta elementā
EventsViewModel	eventsViewModel	Datu glabāšanas klase lietotnes pastāvēšanas cikla laikā

Metodes:

Tips	Metode	Apraksts
List<Event>	getAllEventsList()	Saņemt visu pasākumu sarakstu

Klase ViewAdapter extends BaseAdapter implements ListAdapter.

Adapteris, kurš atbild par datu izvietošanu katra saraksta elementā.

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	eventImage	Pasākuma bilde
TextView	eventName	Pasākuma nosaukums
TextView	eventLocation	Pasākuma norises vieta
TextView	eventNumberOfParticipants	Pasākuma dalībnieku skaits

RelativeLayout	listItemContainer	Lietotāja saskarnes komponents, tiek izmantots atbildes iegūšanai, nospiežot uz saraksta elementu
TextView	creator	Izveidot pasākumus

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt elementa saraksta kārtas numuru
View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam saraksta elementu, kuru redz uz ekrāna
void	onClick(View view)	Atbilde uz saraksta elementa nospiešanu, tiek atvērts pasākuma detalizēts skats

3.2.2.34 Klase UserSettingsAdminActivity

Klase lietotāju pārvaldīšanai, piekļuve ir tikai lietotnes Administratoram.

Mainīgie:

Tips	Mainīgais	Apraksts
ListView	listView	Lietotāja saskarnes komponents, kurš ļauj veidot pāršķirstamu sarakstu
List<User>	usersList	Visu lietotāju saraksts
ViewAdapter	adapter	Adapteris, kurš atbild par informācijas izvietošanu katra saraksta elementā

Metodes:

Tips	Metode	Apraksts

List<User>	addAll(User user)	Pievienot visus lietotājus

Klase ViewAdapter extends BaseAdapter implements ListAdapter.

Adapteris atbild par datu izvietošanu katra saraksta elementu.

Mainīgie:

Tips	Mainīgais	Apraksts
ImageView	preferenceIcon	Ikona lietotāja rediģēšanai
TextView	userFullName	Lietotāja pilnais vārds
TextView	userUid	Lietotāja identifikators
TextView	userType	Lietotāja tips

Metodes:

Tips	Metode	Apraksts
int	getCount()	Atgriezt elementu skaitu sarakstā
Object	getItem(int i)	Atgriezt saraksta elementu dotajā pozīcijā
long	getItemId(int i)	Atgriezt elementa saraksta kārtas numuru
View	getView(int position, View view, ViewGroup viewGroup)	Atgriež lietotājam saraksta elementu, kuru redz uz ekrāna
void	onClick(View view)	Atbilde uz saraksta elemena nospiešanu, tiek atvērts pasākuma detalizēts skats

4 DATU PROJEKTĒJUMS

4.1 Firebase datu bāzes apraksts

Projektā izmantota“Firebase” mākoņa datubāze, lai dati varētu glabāties un sinhronizēties vienlaicīgi starp vairākiem lietotājiem. Tā ir JSON Objektu koka veida datubāze. Dati tajā tiek glabāti kā objekti. Tā kā datubāze ir koka veida, tā tiek aprakstīta līmeņu veidā.

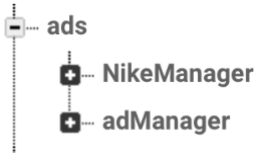
4.1.1 Pirmais līmenis – reklāma (Ads), pasākums (Events) un lietotāji (Users)

Pirmajā līmenī iet sadalījums pēc Reklāmas klasēm, pasākumiem, lietotājiem.

4.1. att. ***Firebase pirmais līmenis.***

4.1.2 Otrais līmenis - Reklāma (Ads)

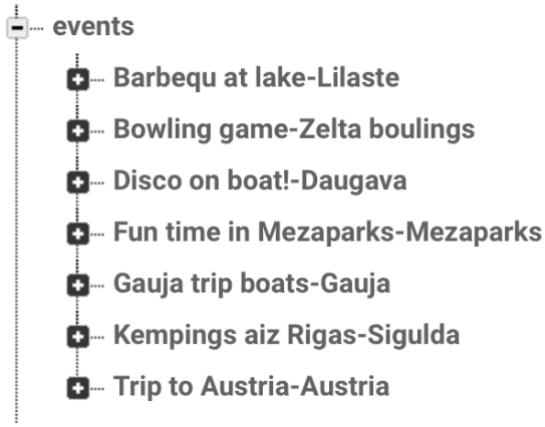
Otrajā līmenī atrodas reklāmas ievietotāja identifikatori.



4.2. att. ***Firebase Ads otrais līmenis.***

4.1.3 Otrais līmenis – Pasākumi (Events)

Otrajā pasākumu līmenī atrodas Pasākumu klases objekti, tie ir sakārtoti pēc kombinācijas "pasākuma nosaukums"-"norises vieta".



4.3. att. ***Firebase Events otraīmenis.***

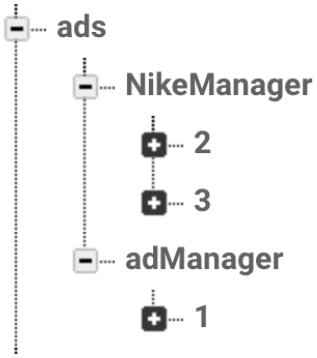
4.1.4 Otrais līmenis - Lietotājs (Users)

Otrajā Lietotājs līmenī glabājas klases lietotājs objekti, tie ir sadalīti pēc unikāla lietotāju identifikatora.

4.4. att. ***Firebase Users otrais līmenis.***

4.1.5 Trešais līmenis – Reklāma (Ads)

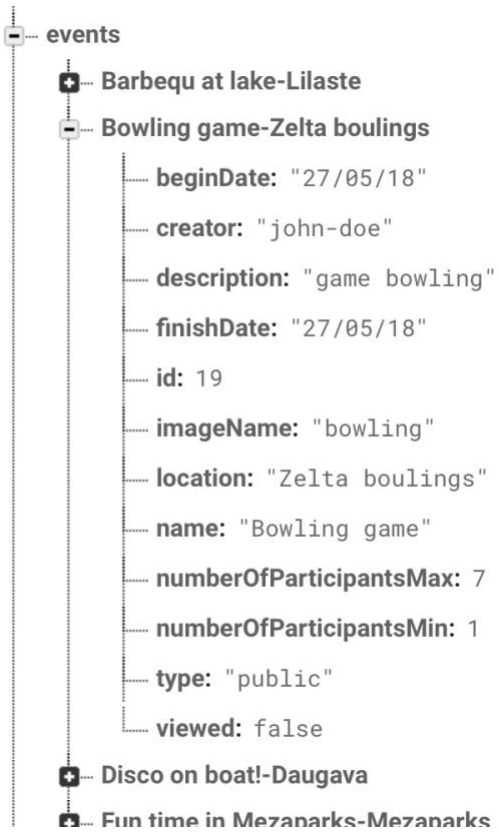
Trešajā reklāmas līmenī atspoguļoti Reklāmas klases objekti, kuri ir sakārtoti pēc unikāla identifikatora.



4.5. att. ***Firebase Ads trešais līmenis***

4.1.6 Trešais līmenis – Pasākums (Events)

Trešajā Pasākumu līmenī glabājas paši pasākuma objekti.



4.6. att. ***Firebase Events trešais līmenis.***

4.1.7 Trešais līmenis – Lietotājs (Users)

Trešajā Lietotājs līmenī, glabājas Lietotāja objekti. Lietotājiem, kuri autorizējās caur Facebook API vai Gmail API nav paroles. Lietotājiem, kuri reģistrējās lietotnē ir lauks paroles ievadīšanai, kurš glabā paroli šifrētā veidā.

4.7. att. ***Firebase Users trešais līmenis.***

4.1.8 Ceturtais līmenis – Reklāma (Ads)

Ceturtajā Reklāmas līmenī glabājas Reklāmas objekti.

4.8. att. ***Firebase Ads ceturtais līmenis.***

4.2 Iekšējā datu bāze SQLite

Lietojumprogrammatūrai izmanto SQLite datu bāzi. Darbam ar datu bāzi izmanto jaunu Google bibliotēku: Room Persistence Library. Ar jauno Room bibliotēku strādāt datu bāzē ir daudz vienkāršāk un patīkamāk. Lai pieslēgtu bibliotēku, nepieciešams pieslēgt atkarību Gradle datne:

```
implementation "android.arch.persistence.room:runtime:1.0.0"
annotationProcessor "android.arch.persistence.room:compiler:1.0.0"
```

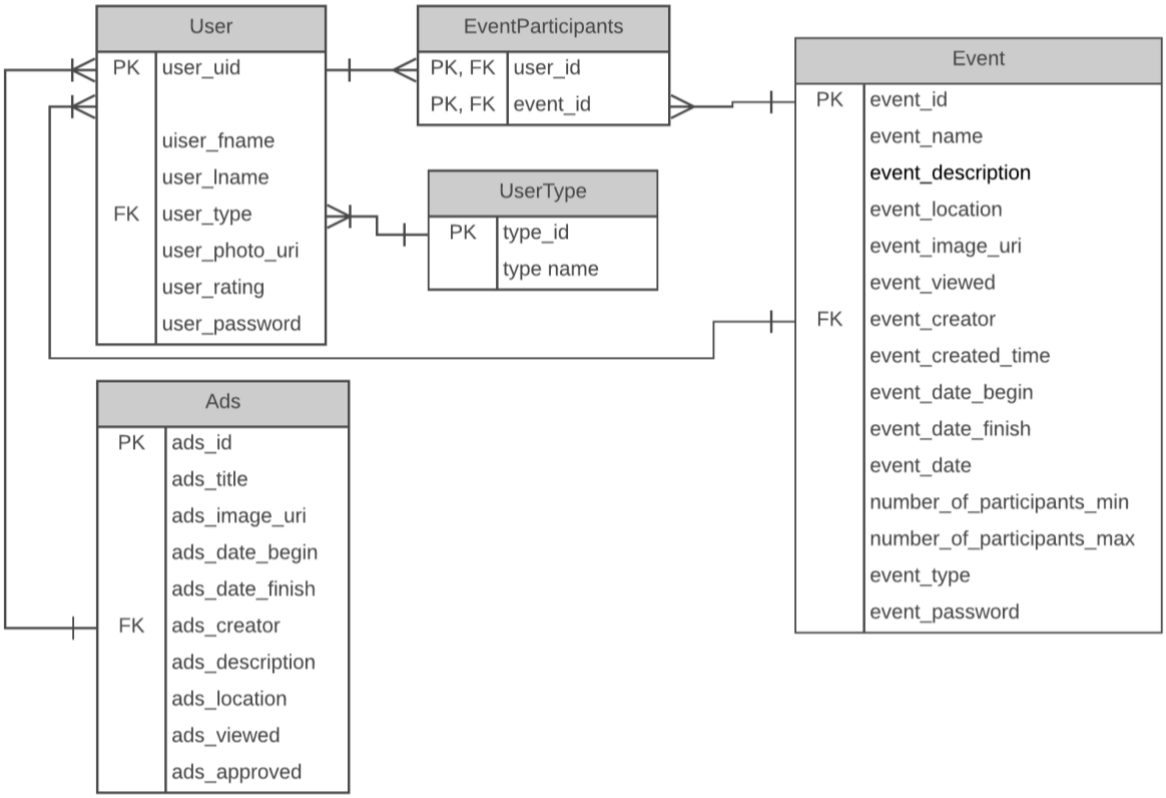
Room satur trīs pamata komponentus: Entity, Dao un Database.

Entity – Entity anotācijai nepieciešams apzīmēt objektu, kuru mēs vēlāmies glabāt datu bāzē, piemēram, Event.java klasi (sk. 2. pielikumu).

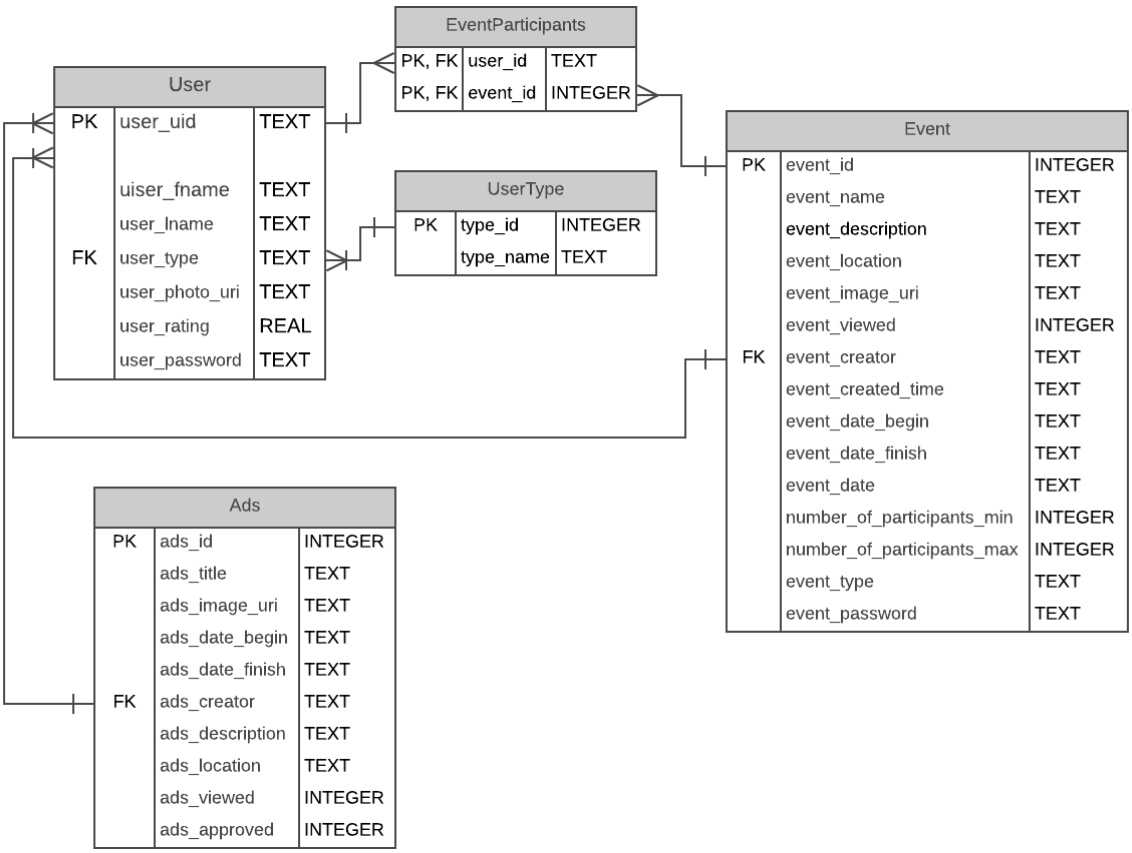
Dao – Dao objektā mēs aprakstīsim darba metodes ar datu bāzi, piemēram, UserDao.java. Dao klases piemērs (sk. 2. pielikumu).

Database – Database anotācija atzīmē pamata klasi darbā ar datu bāzi.Šai klasei jābūt abstraktai un jāmanto RoomDatabase, piemēram, AppDatabase.class (sk. 2. pielikumu):

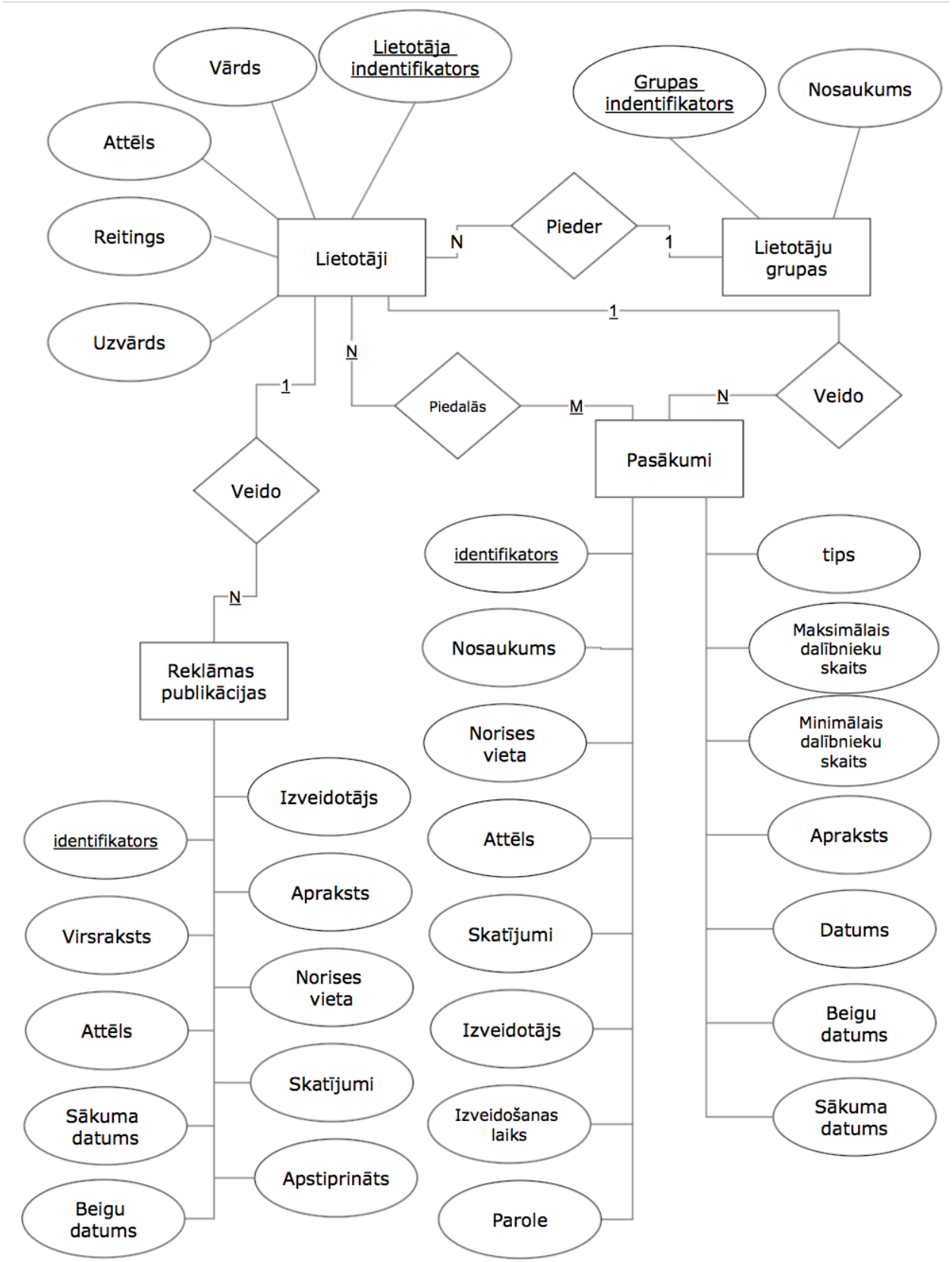
4.2.1 Loģiskais ER modelis



4.2.2 Fiziskais ER modelis



4.2.3 Datubāzes konceptuālais ER modelis



4.2.4 Datu dekompozīcija

Sistēma datus glabā vienā SQLite datubāzē. Datubāze sastāv no 5 tabulām:

4.1. tabula. ***Datu dekompozīcija.***

Tabula	**Apraksts**
User	Tabula visu lietotāju datu glabāšanai.
UserType	Tabula lietotājgrupas tipa piederībai
Event	Tabula pasākumu informācijas glabāšanai.
EventParticipants	Starptabula, glabā informāciju par lietotājiem, kuri piedalās un kādos pasākumos.
Ads	Tabula reklāmas datu glabāšanai, kura ir redzama galvenajā lietotnes ekrānā.

4.2.4.1 Lietotāji

Tabula User paredzēta informācijas glabāšanai par lietotājiem. Taustiņš user_uid – Primāra atslēga, user_type – ārēja atslēga uz tabulu UserType, kura paredzēta lietotāja grupas noteikšanai.

4.2. tabula. ***Lietotāji.***

Lauks	**Datu Tips**	**NULL**	**Pēc noklusējuma**	**Atslēga**	**Apraksts**
user_uid	TEXT	NO	NULL	PK	Lietotāja unikāls identifikators
user_fname	TEXT	YES	NULL		Lietotāja vārds
user_lname	TEXT	YES	NULL		Lietotāja uzvārds
user_type	TEXT	NO	0 (Reģistrēts lietotājs)	FK	Links uz tipu (lietotāja grupu) uz tabulu UserType
user_photo_uri	TEXT	YES	NULL		Lietotāja bilde
user_rating	REAL	YES	NULL		Lietotāja reitings
user_password	TEXT	YES	NULL		Lietotāja parole, vienmēr šifrēta

4.2.4.2 Lietotāju grupas

Tabula UserType glabā visu lietotāju grupas. Primāra atslēga type_id

Lauks	Datu Tips	NULL	Paā	Atslēga	Apraksts
type_id	INTEGER	NO	0	PK	Lietotāju grupas unikāls identifikators
type_name	TEXT	YES	NULL		Lietotāja grupa nosaukums

4.2.4.3 Pasākumi

Tabula Event paredzēta informācijas glabāšanai par pasākumu. Primāra atslēga – event_id, Ārēja atslēg – event_creator, atsēga tabulai User.

Lauks	Datu Tips	NULL	Pēc noklusējuma	Atslēga	Apraksts
event_id	INTEGER	NO	0	PK	Pasākuma identifikators
event_name	TEXT	NO	NULL		Pasākuma nosaukums
event_description	TEXT	YES	NULL		Pasākuma apraksts
event_location	TEXT	YES	NULL		Norises vieta
event_image_uri	TEXT	YES	NULL		Attēls
event_viewed	INTEGER	NO	0		Vai pasākums tika apskatīts vai nē
event_creator	TEXT	NO	NULL	FK	Links uz tabulu User uz pasākuma veidotāju
event_created_time	TEXT	NO	NULL		Pasākuma izveidošanas datums
event_date_begin	TEXT	YES	NULL		Pasākuma sākuma datums

event_date_finish	TEXT	YES	NULL		Pasākuma beigu datums
event_date	TEXT	YES	NULL		Pasākuma datums
number_of_ participants_min	INTEGER	NO	0		Minimālais dalībnieku skaits
number_of_ participants_max	INTEGER	YES	NULL		Maksimālais dalībnieku skaits
event_type	TEXT	NO	PUBLIC		Pasākuma tips
event_password	TEXT	YES	NULL		Pasākuma parole

4.2.4.4 Pasākuma dalībnieki

Tabula EventParticipants paredzēta konkrēta pasākuma (Event) dalībnieku (User) glabāšanai. Šī starptabula Many – Many attiecību nodrošināšanai starp tabulām User un Event. Primāra atslēga – saistviela user_id un event_id, Ārējas atslēgas – user_id tabulā User un event_id tabulā Event.

Lauks	**Datu Tips**	**NULL**	**Pēc noklusējuma**	**Atslēga**	**Apraksts**
user_id	TEXT	NO	NULL	PK, FK	Lietotāja unikāls identifikators
event_id	INTEGER	NO	NULL	PK, FK	Pasākuma unikāls identifikators

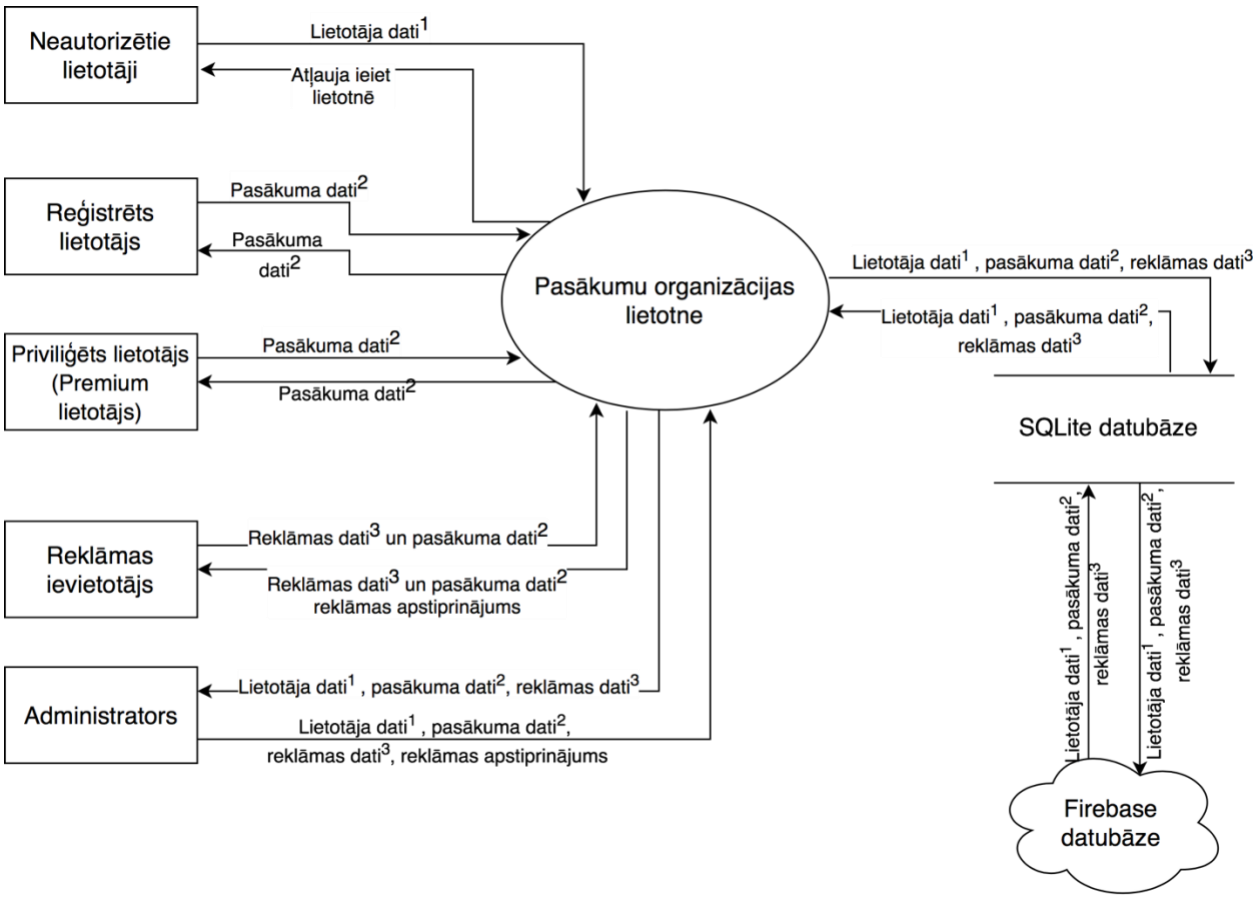
4.2.4.5 Reklāma

Ads tabula paredzēta visas informācijas glabāšanai par reklāmu. Primāra atslēga – ads_id, ārēja atslēga – ad_creator links uz pasākuma izveidotāju tabulā User

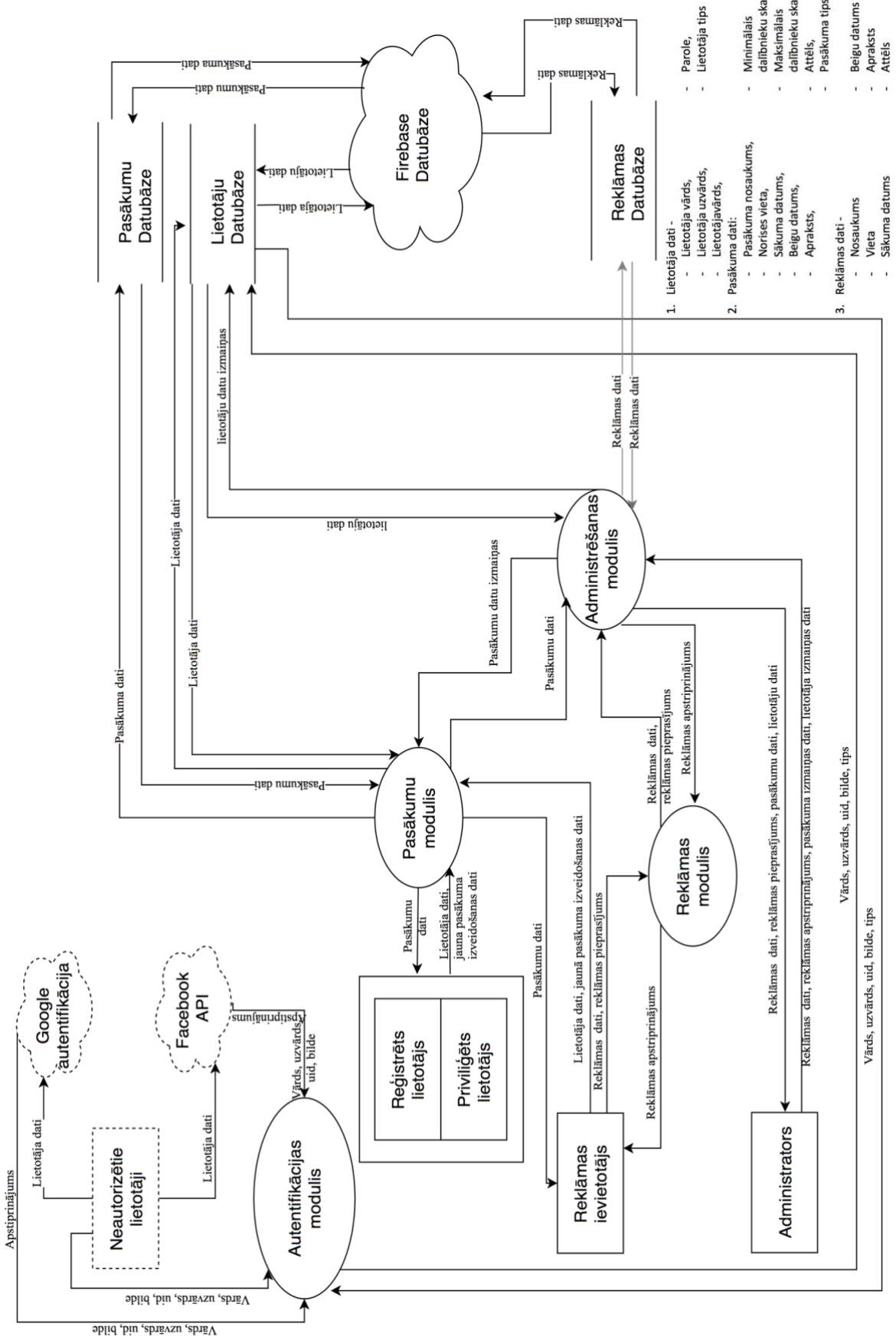
Lauks	Datu Tips	NULL	Pēc noklusējuma	Atslēga	Apraksts
ads_id	INTEGER	NO	NULL	PK	Reklāmas publikācijas unikāls identifikators
ads_title	TEXT	YES	NULL		Reklāmas publikācijas virsraksts
ads_image_uri	TEXT	YES	NULL		Attēls
ads_date_begin	TEXT	YES	NULL		Reklāmas sākuma datums
ads_date_finish	TEXT	YES	NULL		Reklāmas beigu datums
ads_creator	TEXT	NO	NULL	FK	Links uz reklāmas publikācijas izveidotāju tabulā User
ads_descriptipn	TEXT	YES	NULL		Apraksts
ads_location	TEXT	YES	NULL		Norises vieta
ads_viewed	INTEGER	NO	0		Vai lietotājs ir apskatījis reklāmas publikāciju
ads_approved	INTEGER	NO	0		Vai Administrators ir apstiprinājis reklāmas publikāciju

4.3 Datu plūsmu diagrammas

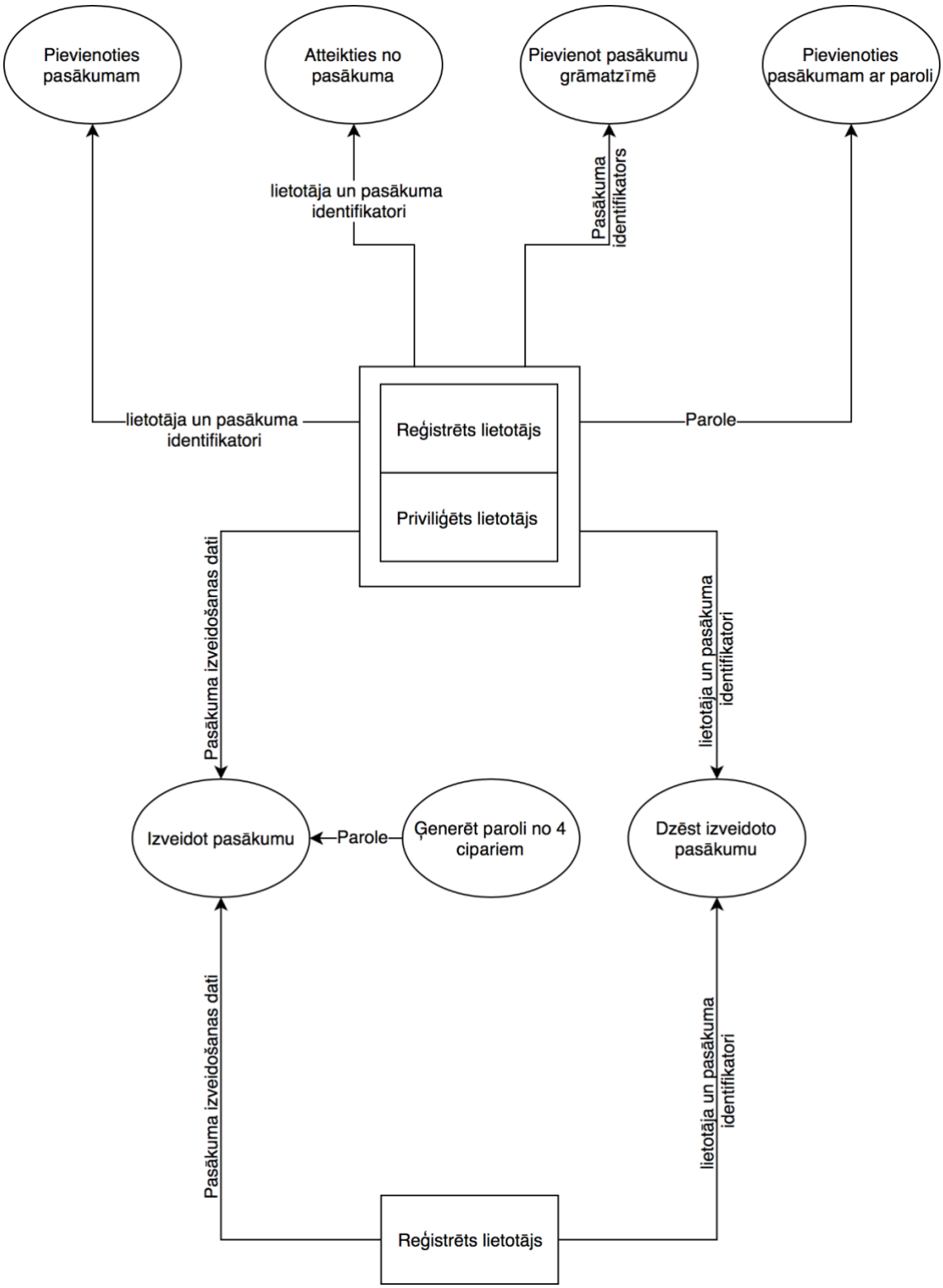
4.3.1 0. līmeņa datu plūsmu diagramma


Neautorizētie lietotāji
Lietotāja dati1
Atļauja ieiet lietotnē
Reģistrēts lietotājs
Pasākuma dati2
Pasākuma dati2
Priviliģēts lietotājs (Premium lietotājs)
Pasākuma dati2
Pasākuma dati2
Reklāmas ievietotājs
Reklāmas dati3 un pasākuma dati2
Reklāmas dati3 un pasākuma dati2 reklāmas apstiprinājums
Administrators
Lietotāja dati1 , pasākuma dati2, reklāmas dati3
Lietotāja dati1 , pasākuma dati2, reklāmas dati3, reklāmas apstiprinājums
Pasākumu organizācijas lietotne
Lietotāja dati1 , pasākuma dati2, reklāmas dati3
Lietotāja dati1 , pasākuma dati2, reklāmas dati3
SQLite datubāze
Lietotāja dati1 , pasākuma dati2, reklāmas dati3
Lietotāja dati1 , pasākuma dati2, reklāmas dati3
Firebase datubāze

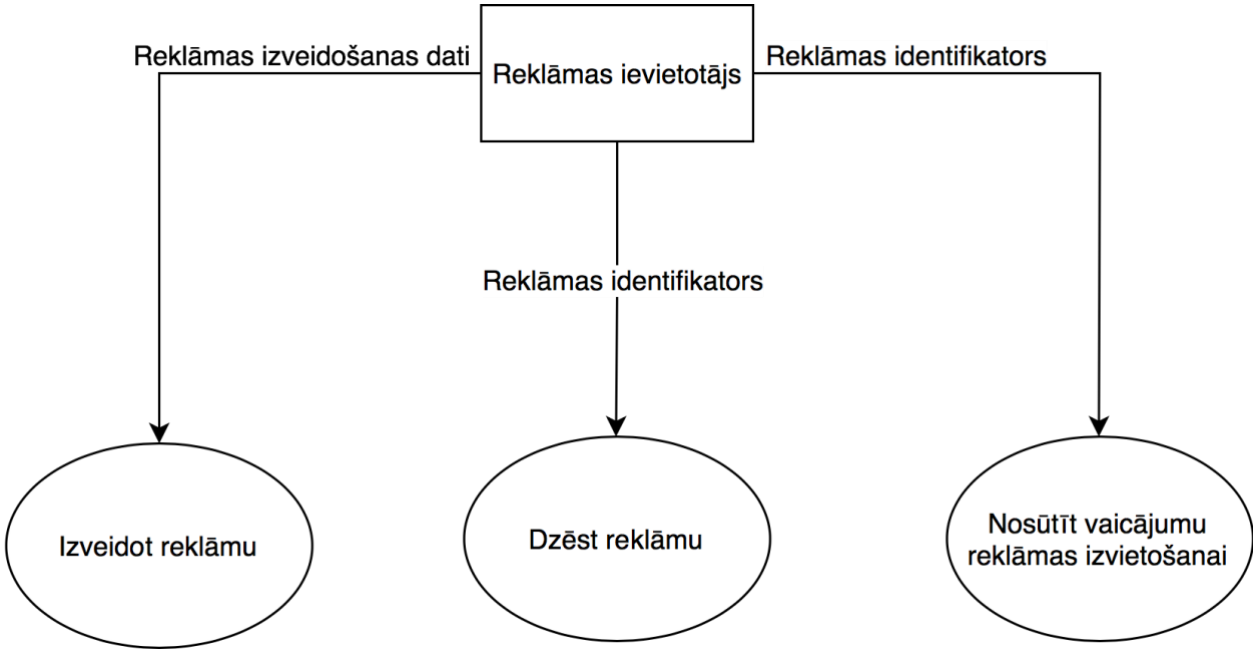
4.3.1 1. Līmeņa datu plūsmu diagramma



4.3.2 2. līmeņa datu plūsmu diagramma– Pasākumu modulis

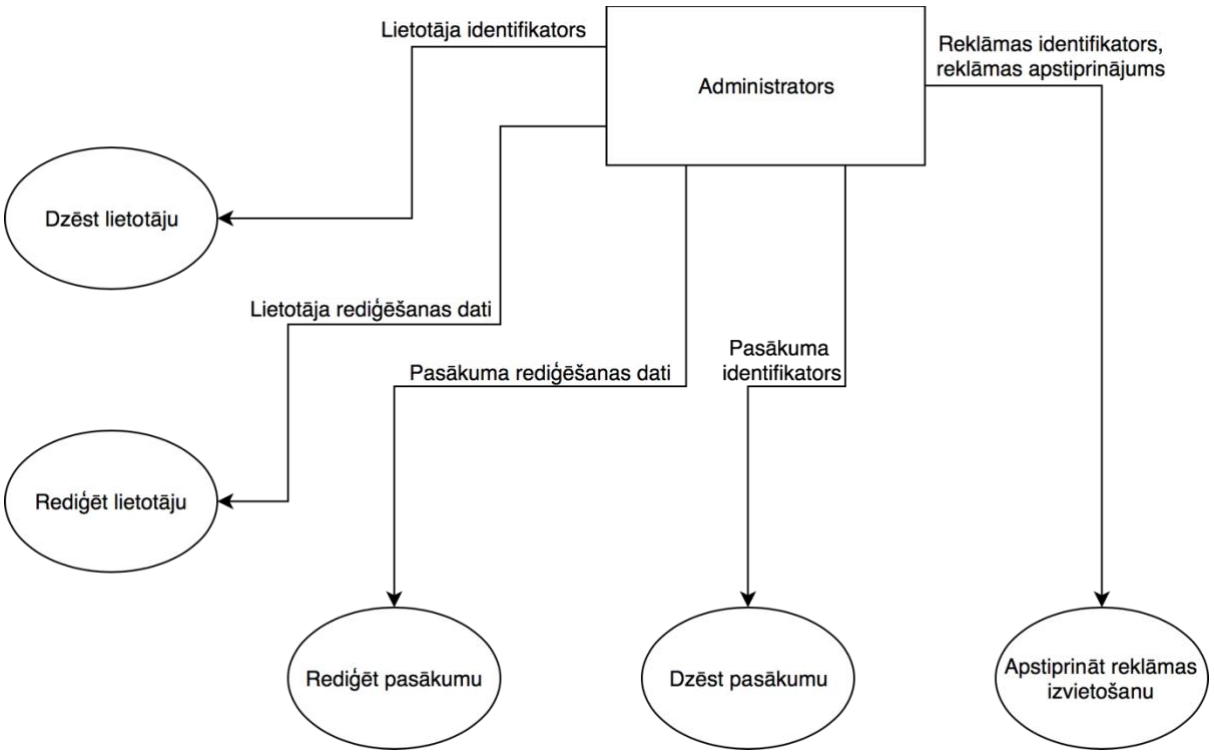


4.3.3 2. līmeņa datu plūsmu diagramma– Reklāmas modulis

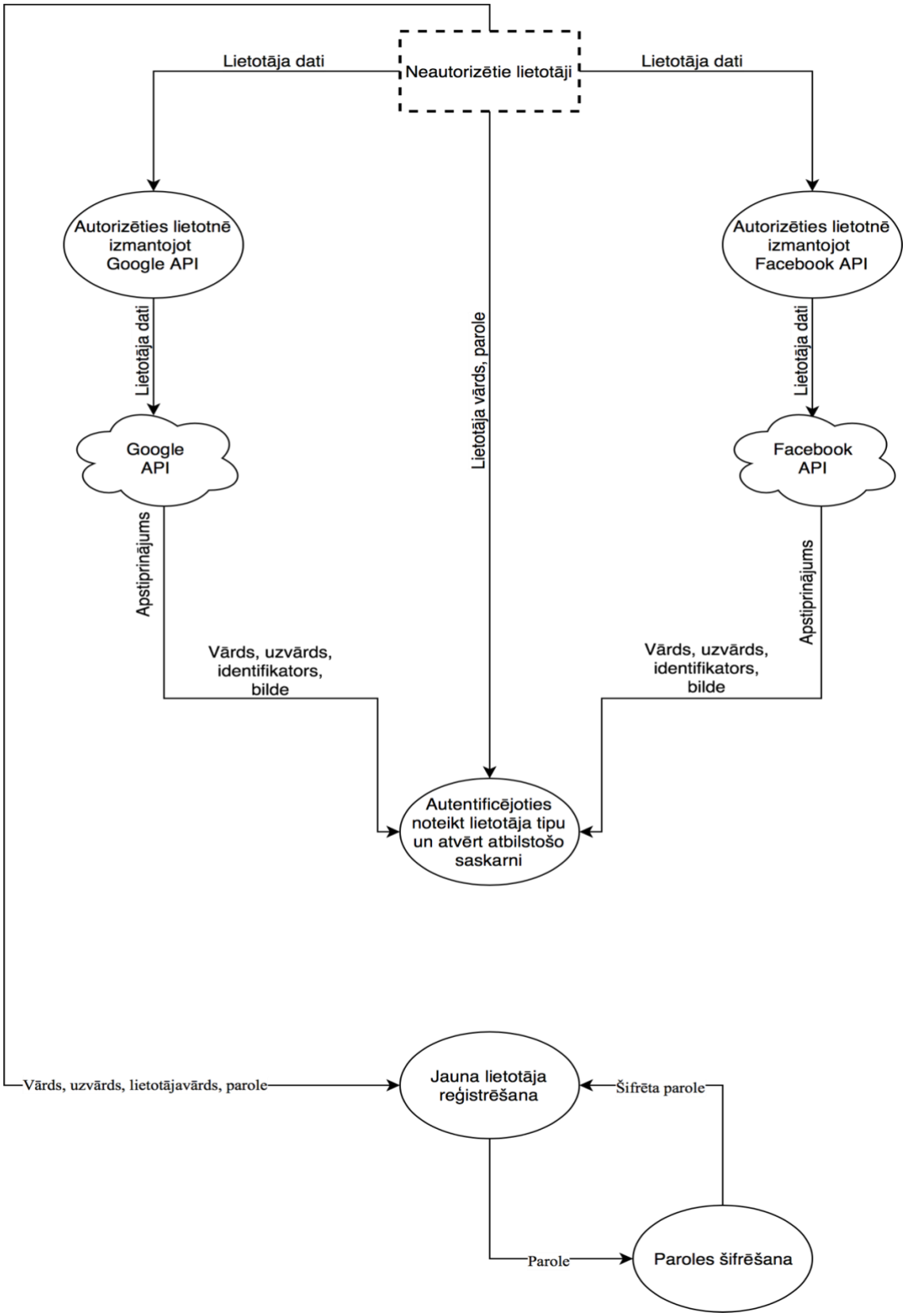


Ilustrācija 4.3-1 2.līmeņa datu plūsmu diagramma – Reklāmas modulis

4.3.4 2. līmeņa datu plūsmu diagramma– Administrēšanas modulis



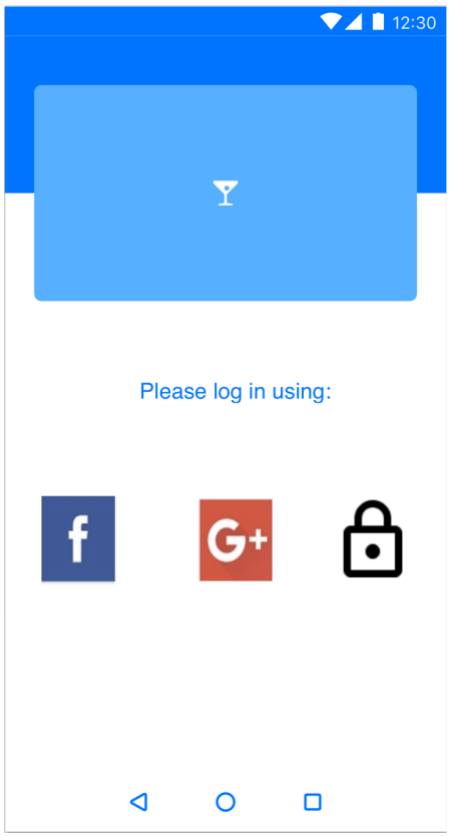
4.3.5 2. līmeņa datu plūsmu diagramma– Autentifikācijas modulis



4.4 Karkasveida kontūru diagramma

Nākamajās daļās tiek aprakstīts kā izskatās klases saistītas ar lietotāja saskarni.

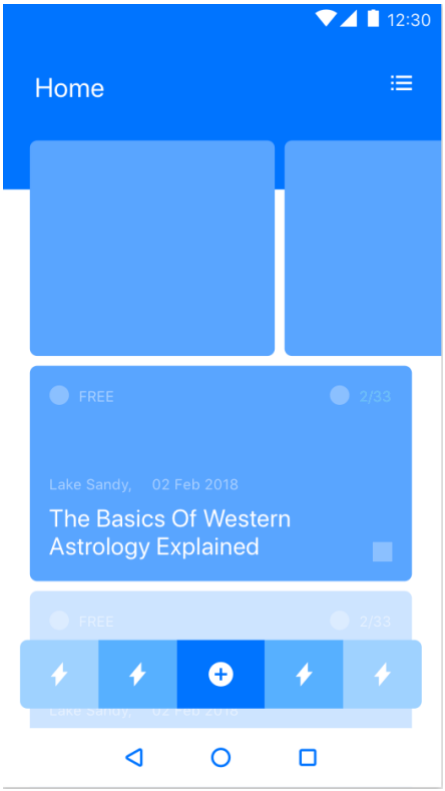
4.4.1 LoginAcitivty



4.9. att. ***Autentifikācija.***

- pieteikšanas ekrāns ar logo
- 2 taustiņi ieejai caur sociālajiem tīkliem un viens taustiņš lai autorizētos ar saviem datiem vai izveidot jaunu kontu, kuri ļauj ieiet lietotnē caur vienu no izvēlētajiem sociālajiem tīkliem.

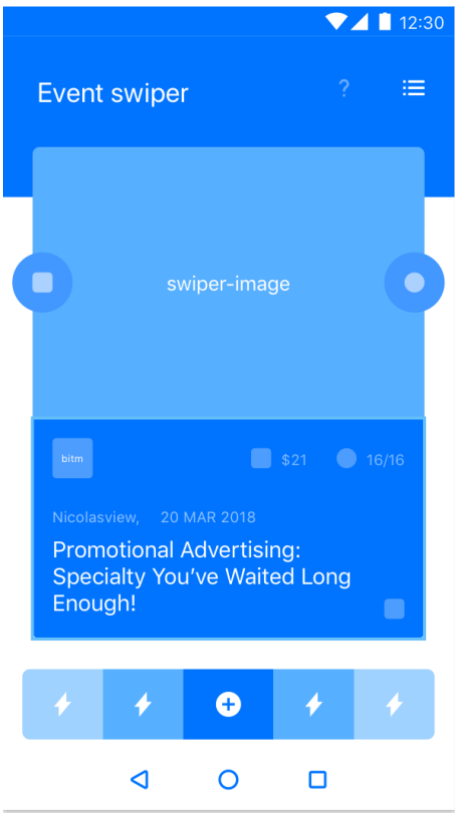
4.4.2 Home Fragment



4.10. att. ***Home Fragment.***

Šajā ekrānā atrodas horizontālais reklāmu saraksts un vertikālais saraksts ar vispopulārākajiem pasākumiem.

4.4.3 EventsFragment



4.11. att. ***Events Fragment.***

Šajā ekrānā atspoguļota vizītkarte ar pasākumiem. Vizītkartē norādīts nosaukums, adrese, pasākuma norises vieta, organizatora vārds, dalībnieku skaits (esošais un maksimālais), iespējams attēlot dalībnieku skaitu ar svītriņu, kura tiks piepildīta pēc cilvēku pieteikšanas pasākumam. Kā arī lietotājs var pievienot attēlu vai fotogrāfiju. Velkot vizītkarti uz labo pusi, lietotājs piekrīt pasākumam, velkot vizītkarti uz kreiso pusi, lietotājs atsakās no pasākuma. Tāpat uz vizītkartes vai zem tās jābūt ikonai, nospiežot uz tās, atvērsies detalizēts skats ar visu informāciju par pasākumu.

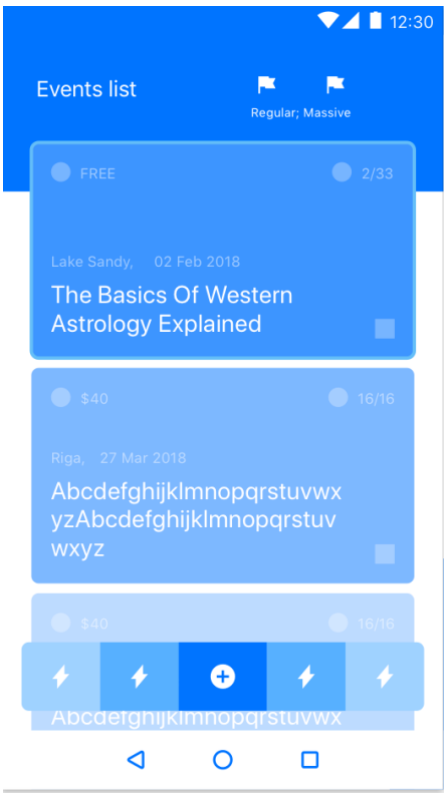
Ir arī ikonas, pēc nospiešanas uz tām lietotājs var:

- Pieteikties pasākumam
- Atteikties no pasākuma
- Pievienot "Apskatīt vēlāk"

Virs vizītkartes atrodas pārslēgšanas taustiņš, nospiežot to, vizītkarte tiks pārslēgta uz saraksta veidu, kas ļaus ātrāk apskatīt pasākumus.

Šī ekrāna mērķis: sniegt lietotājam iespēju interaktīvā veidā un mierīgā tempā apskatīt atvērtos pasākumus.

4.4.4 EventsListFragment

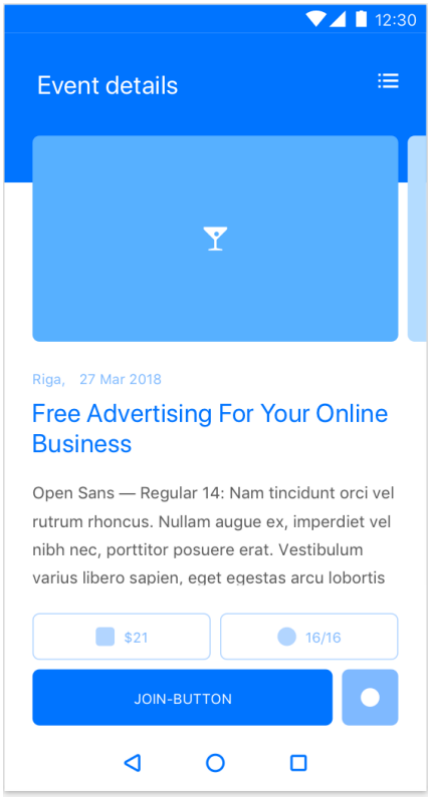


4.12. att. ***Events List Fragment.***

Šajā ekrānā tiek atspoguļoti aktīvie pasākumi saraksta veidā. Katrā no saraksta elementiem jābūt: pasākuma nosaukumam, pasākuma norises vietai, pasākuma dalībnieku skaitam, organizatora vārds, taustiņiem - lai pieteiktos dalībai, atteikšanai no pasākuma un pievienošanai "Apskatīt vēlāk".

Jābūt iespējai atgriezties atpakaļ kartes režīmā, kur noritēs pasākumi. Nospiežot uz saraksta elementu tiks atvērts detalizēts skats ar visu informāciju par pasākumu.

4.4.5 EventDetailsActivity



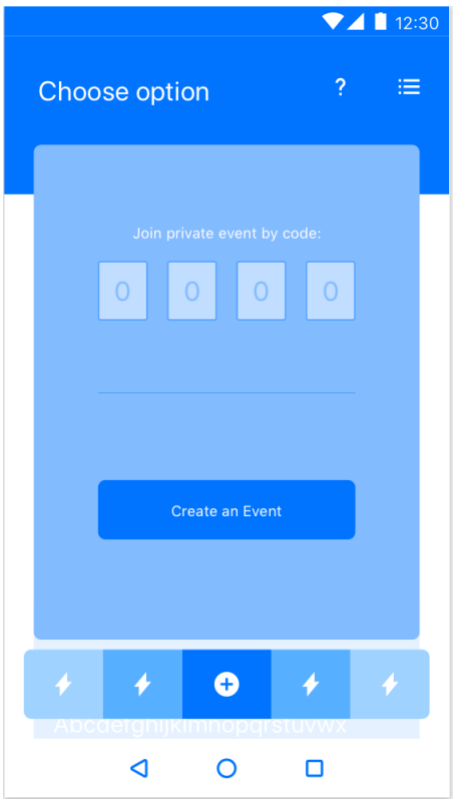
4.13. att. ***Evetns Details Fragment.***

Šajā ekrānā ir atspoguļots pasākuma detalizēts skats. Ekrānā ir redzams:

- pasākuma nosaukums,
- pasākuma norises vieta,
- organizators,
- pasākuma dalībnieku skaits,
- pasākuma apraksts,
- taustiņš, lai pievienotos pasākumam,
- taustiņš, lai atteiktos no pasākuma,
- taustiņš, lai pievienotu pasākumu "Apskatīt vēlāk".

Tāpat jābūt iespējai pievienoties pasākumam. Kā arī jāatspoguļo dalībnieku skaits, cik ņems dalību šajā pasākumā. Šim ekrānam jāsniedz visu nepieciešamo informāciju par pasākumu.

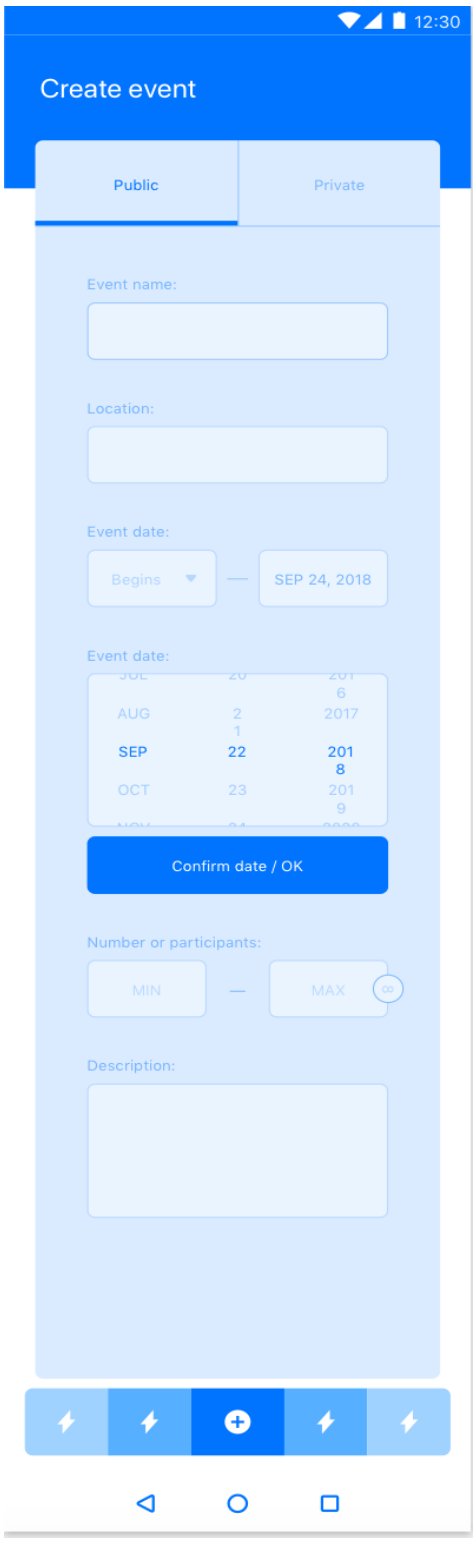
4.4.6 CreateEventFragment



4.13. att. ***Create Event Fragment.***

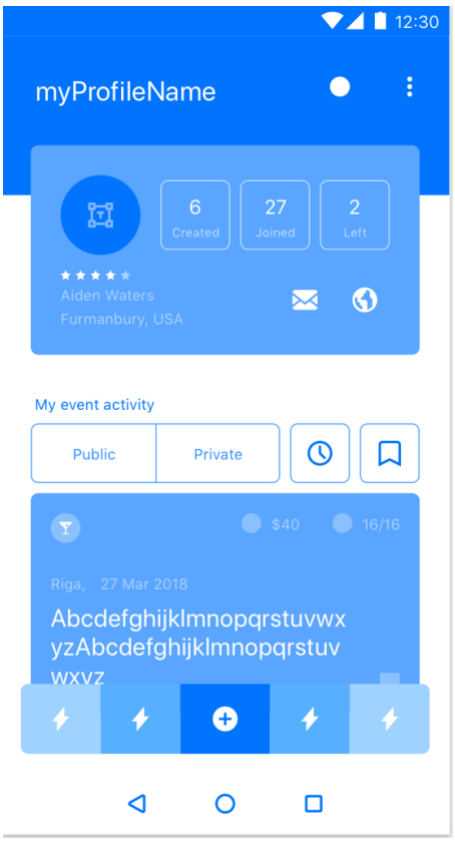
Šājā ekrānā var pievienoties slēgtajam pasākuma, ievadot kodu vai pāriet uz ekrānu, jauna pasākuma izveidošanai.

4.4.7 CreateEventTemporaryActivity



4.14. att. ***Create Event Temporary Fragment.***

4.4.8 MyProfileFragment



4.15. att. ***My Profile Fragment.***

Ekrāns ar informāciju par lietotāja profilu, tāpat uz ekrāna var apskatīt ar pievienotajiem atvērtiem pasākumiem, vai mainot filtru apskatīt slēgtos pasākumus, saistītus ar šo lietotāju.

5 TESTĒŠANAS DOKUMENTĀCIJA

5.1 Ievads

Testēšanas dokumentācijā ir aprakstītas metodes, kā tiek testēta lietotne un šīs metodes rezultāti. Testēšanai tika izmantoti, akcepttesti и funkcionālie testi. Tāpat tika uzrakstīti automātiskie vienībtesti. Testēšana tikai veidota, lai pārliecinātos PPS lietotnes atbilstībai.

5.2 Testēšanas metodika

Manuāla testēšana tika veikta uz šādām ierīcēm: Xiami Redmi 4A un Samsung Galaxy S7 Edge, tāpat uz Android emulatora ierīces LG Nexus 5. Testēšanas veikšanai tika ievadīti pareizie un nepareizie dati, pārbaudot visus iespējamos rezultātus. Manuāla funkcionāla testēšana tika veikta uzreiz, tiklīdz funkcija tika izstrādāta un beigās tika veikta atkārtota testēšana.

5.3 Testēšanas žurnāls

5.3.1 Authentifikācijas modulis

5.3.1.1 Autorizēties lietotnē izmantojot Facebook API

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja vārds un parole	Ja autorizācija notiek caur Facebook API, tad atvēras facebook lapa, kur lietotājs var ievadīt savu lietotājvārdu un paroli, ja autorizācija notika veiksmīga, tad lietotājs var ieiet lietotnē. visas pārbaudes un kļūdas notiek ārpus Facebook	✓	19.05.2018

5.3.1.2 Autorizēties lietotnē izmantojot Google API

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja vārds un parole	Autorizējoties Google API, atvēras Google mājaslapa, kur lietotājs var ievadīt savu lietotājvārdu un paroli, veiksmīgas autorizācijas gadījumā, lietotājs var ieiet lietotnē. Visas pārbaudes un kļūdas notiek ārpus Google	✓	19.05.2018

5.3.1.3 Autentificējoties noteikt lietotāja tipu un atvērt atbilstošo saskarni

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
No datu bāzes tiek ņemts lietotāja tips pēc identifikatora	Ja lietotājs lietotnē autorizējas pirmo reizi, viņa tips: Reģistrēts lietotājs un viņam ir piekļuve pie šīs lietotāju grupas funkcijām	✓	19.05.2018
No datu bāzes tiek ņemts lietotāja tips pēc identifikatora	Ja lietotājs pieder pie grupas Reģistrēts lietotājs, tad viņam ir piekļuve tikai pie šīs grupas funkcijām	✓	19.05.2018
No datu bāzes tiek ņemts lietotāja tips pēc identifikatora	Ja lietotājs pieder pie grupas Priviliģēts lietotājs, tad viņam ir piekļuve tikai pie šīs grupas funkcijām	×	19.05.2018
No datu bāzes tiek ņemts lietotāja tips pēc identifikatora	Ja lietotājs pieder pie grupas Administrators, tad viņam ir piekļuve lietotnes pārvaldīšaani caur Administrācijas paneli	✓	19.05.2018
No datu bāzes tiek ņemts lietotāja tips pēc identifikatora	Ja lietotājs pieder pie grupas Reklāmas ievietotājs, tad viņam ir piekļuve reklāmas publikāciju pārvaldei caur Reklāmas paneli	✓	19.05.2018

5.3.1.4 Jauna lietotāja reģistrācija

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Vārds: John Uzvārds: Smith Lietotāja vārds: john.smith parole: 12345qwerty	Lietotājs ir reģistrēts un pāriet uz autorizācijas lapu, un no turienes var nonākt lietotnē	✓	19.05.2018
Vārds: Uzvārds: Smith Lietotāja vārds: john.smith parole: 12345qwerty	Tā kā vārds nav ierakstīts, tad paziņo par kļūdu: "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!", tāpat lauks vārds tiek izcelts	✓	19.05.2018
Vārds: John Uzvārds: Smith Lietotāja vārds: John Smith parole: 12345qwerty	Lietotāja vārds nevar saturēt atstarpi, tāpēc paziņo par: "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!", tāpat lauks Lietotāja vārds tiek izcelts.	✓	19.05.2018
Vārds: John Uzvārds: Smith Lietotāja vārds: john.smith parole: 12345qwerty	Ja lietotājs ar tādu vārdu jau eksistē, tad paziņo par kļūdu: "Šāds lietotāja vārds jau pastāv!", tāpat lauks Lietotāja vārds tiek izcelts.	✓	19.05.2018
Vārds: John Uzvārds: Smith Lietotāja vārds: john.smith parole: 12345	Parolei jāsatur vismaz 1 ciparu un 1 burtu, tāpēc tiek paziņots par kļūdu: "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!", tāpat lauks Parole tiek izcelts.	✓	19.05.2018

5.3.2 Pasākumu modulis

5.3.2.1 Izveidot pasākumu

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Pasākuma nosaukums Norises vieta Sākuma datums Beigu datums Apraksts Minimālais dalībnieku skaits Maksimālais dalībnieku skaits Attēls Pasākuma tips	Ja kāds lauks nav aizpildīts, tad tās tiek izcelts un paziņo par kļūdu: "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!"	✓	19.05.2018

5.3.2.2 Pievienoties pasākumam

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja un pasākuma identifikātori	Nospiežot taustiņu "Pievienoties", taustiņam jānomainās uz "Atteikties no pasākuma", pasākumam jāpievienojas uz nodaļu "My Profile"	✓	19.05.2018

5.3.2.3 Atteikties no pasākuma

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja un pasākuma identifikātori	Nospiežot taustiņu "Atteikties no pasākuma", taustiņam jānomainās uz "Pievienoties", pasākumam ir jāpazūd no nodalījuma "My Profile"	✓	19.05.2018

5.3.2.4 Dzēst izveidoto pasākumu

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja vārds un parole	Nospiežot taustiņu "Dzēst izveidoto pasākumu", pasākums pazūd no visu pasākumu saraksta	✓	19.05.2018

5.3.2.5 Pievienoties pasākumam ar paroli

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Parole: 1234	Ja parole ir pareiza, atvēras aizvērta pasākuma detalizēts skats	✓	19.05.2018
Parole: 8604	Ja parole ir nepareiza, tad paziņo par kļūdu: "Ievadīta parole ir nepareiza!" un paroles ievades lauks paliek tukšs	✓	19.05.2018

5.3.3 Reklāmas modulis

5.3.3.1 Dzēst reklāmu

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Reklāmas publikācijas identifikators	Nospiežot taustiņu "Dzēst reklāmas publikāciju" reklāmas sludinājums pazūd no galvenās lapas visiem lietotājiem	✓	19.05.2018

5.3.3.2 Apstiprināt reklāmas izvietošanu

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Reklāmas publikācijas identifikators	Nospiežot taustiņu: "Apstiprināt reklāmu" publikācijas statuss mainās uz "Gaida apstiprinājumu"	✓	19.05.2018
Reklāmas publikācijas identifikators	Nospiežot taustiņu "Apstiprināt reklāmu" publikācijas parādās Administratora panelī	✓	19.05.2018

5.3.4 Administrēšanas modulis

5.3.4.1 Dzēst lietotāju

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Lietotāja identifikators	Nospiežot taustiņu "Dzēst lietotāju" lietotājs tiek dzēsts no saraksts	✓	19.05.2018
Lietotāja identifikators	Pēc lietotāja izdzēšanas, visi viņa izveidotie pasākumi tiek dzēsti no lietotnes	✓	19.05.2018

5.3.4.2 Rediģēt lietotāju

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Vārds: John Uzvārds: Smith Lietotāja tips: Reģistrēts lietotājs	Nospiežot taustiņu: "Mainīt lietotāja datus", izvēlētā lietotāja dati tiek nomainīti uz Ievaddati	✓	19.05.2018
Vārds: Uzvārds: Smith Lietotāja tips: Reģistrēts lietotājs	tā kā vārds nav ierakstīts, tad paziņo par kļūdu: "Lūdzu, pārbaudiet vai visi lauki ir aizpildīti pareizi!", tāpat lauks Vārds tiek izcelts	✓	19.05.2018
Vārds: John Uzvārds: Smith Lietotāja tips: Reklāmas ievietotājs	Lietotājam ir mainījies tips, sekojoši, lietotājs ieejot lietotnē, saņems piekļuvi pie reklāmas pārvaldības paneļa	✓	19.05.2018

5.3.4.3 Dzēst pasākumu

Ievaddati	Sagaidāmais rezultāts	Rezultāts	Datums
Pasākumu identifikators	Nospiežot taustiņu "Dzēst pasākumu" pasākumam jāpazūd no lietotnes	✓	19.05.2018

6 PROJEKTA ORGANIZĀCIJA

Projekta darbs tikai organizēts pēc agile metodes - Kanban. Šī metode tika izvēlēta, jo tā visvairāk der projektam, kurā ir viens izstrādātājs, backlog vienmēr darba gaitā tiek papildināts ar uzdevumiem, laiks izstrādei ir ļoti ierobežots.

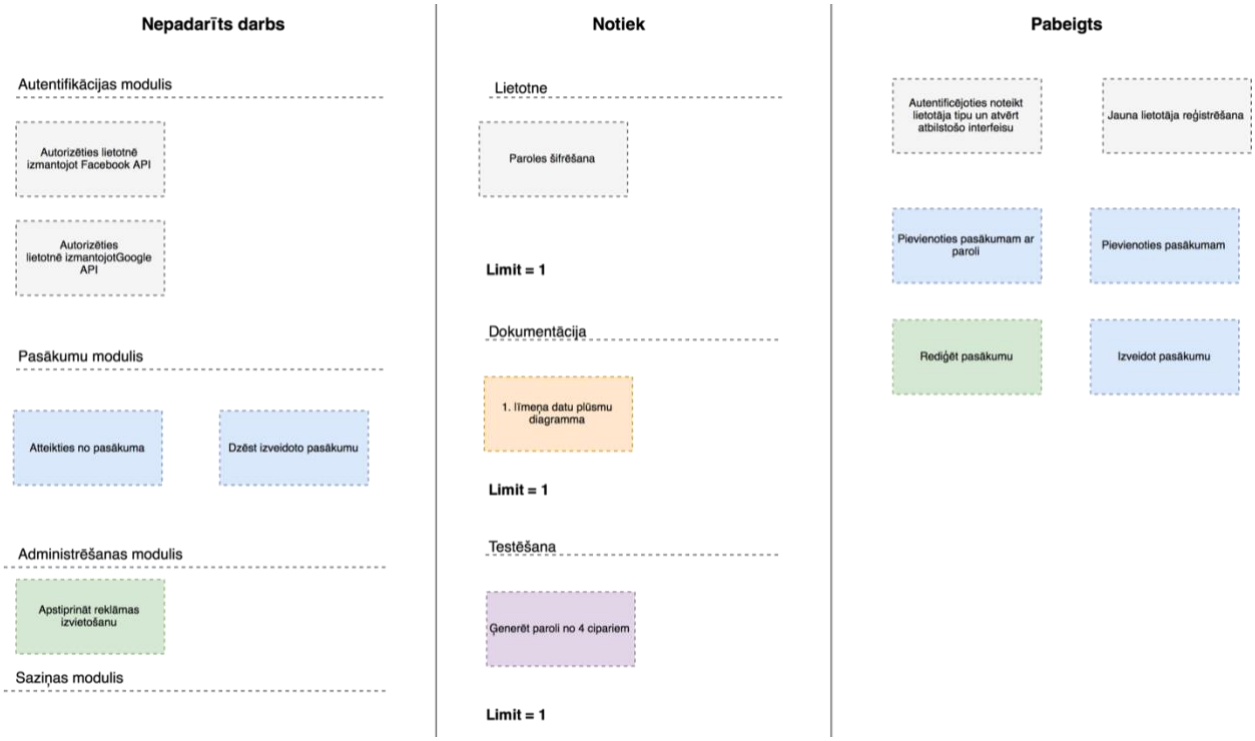
Parastajā variantā, Kanban iekļauj sevī divus vienkāršos noteikumus, kuri ideāli der šim projektam:
Projektam ir noteikts uzdevumu (“backlog”), kurš ir kārtots pēc prioritātes un var mainīties jebkurā laikā

- Uzdevumu skaits, kurus var izpildīt vienlaicīgi ir ierobežots, tas ir, izstrādātajs nevar vienlaicīgi pildīt vairāk uzdevumus nekā ir uzdotais limits. Šī projekta gadījumā, limits ir 1 uzdevums izstrādei, 1 uzdevums dokumentācijai, 1 uzdevums testēšanai.

Projektā bija 3 nodalījumi ar tekošiem:

- izstrāde
- dokumentācija
- testēšana

Zemāk ir parādīta ilustrācija, kura shēmtiski parāda kā izskatās uzdevumu tāfele šajā projektā:



6.1. att. ***Uzdevumu tafele.***

7 DARBIETILPĪBAS NOVĒRTĒJUMS

Kopējo rindu saskaitīšanai tika izmantots kompānijas JetBrains spraudnis “Statistics” priekš Android Studio. Pēc skaitījumiem programma satur 3268 Java koda rindiņas, neieskaitot komentārus un 3367 XML kodu.

Darpietilpības aprēķināšanai izmantoja interneta kalkulatoru “COCOMO II - Constructive Cost Model” uz izrēķinotā darbietilpība ir: 6,4 personmēneši. Īstenībā, projekts kopā ar dokumntāciju un testēšanu aizņēma 3 personmēnešus. Tas ir saistīts ar to, ka Constructive Cost Model izrēķināšanai izmanto tikai vispārējos kritērijus un koda rindiņu skaitu, kurš iekļauj sevī daudz mainīgas uzstādītājmetodes un saņēmējmetodes, mainīgo inicializāciju un darbu ar vizuālajiem elementiem.

8 KVALITĀTES NODROŠINĀŠANA

Lai nodrošinātu sistēmas kvalitāti tika izpildītas sekojošās darbības:

- Izstrādes laikā lietotne tika veidota pēc pieņemtiem Material Designs Style Guides
- Izstrāde tika veikta atbilstoši Google rekomendācijām pēc Android lietotnes izstrādes principiem
- Izstrādes laikā tika izmantota Git kontroles rīka versija.
- Tika veikta regulāra testēšana
- Pateicoties Agile izmantošanas metodikai, atrastas kļūdas uzreiz tika labotas, lai vēlāk nerastos problēmas
- Tāpat Kanban ļauj mainīt pieprasījumus izstrādes laikā, kas palīdzēja optimizēt prasības un ietaupīt daudz laika

9 KONFIGURĀCIJU PĀRVALDĪBA

Sākumā lietotnes versiju pārvaldīšanai veidojās lokālās kopijas. Kad funkcijas palika sarežģītākas un loģiskākas, konfigurācijas pārvaldībai tika izvēlēta Git versiju pārvaldības sistēma, izmantojot BitBucket serveri. Tā kā projekts sastāvēja no 1 izstrādātāja un tika veikts uz viena datora, tad jauna versija tika instalēta uz serveri tikai pēc tam, kad tika veikts kaut kas svarīgs un sarežģīts. Tika izmantots tikai Master Branch, tā kā nebija dažādu koda daļu, kurus vajadzētu sinhronizēt savā starpā, jo pie projekta izstrādes strādāja 1 cilvēks.

Author	Commit	Message
Arturs Amirovs	52d0a60	Created AdControlPanel for managing comercial ads
Arturs Amirovs	87a3ecb	Implemented Admin Panel for Editing and deleting Events
Arturs Amirovs	1bf0833	Added functionality for deleting user with CASCADE deletion of create events by this user
Arturs Amirovs	0c6a734	Added Admin Panel and functionality for user data editing and changing user types
Arturs Amirovs	654c444	Added possobility for registration
Arturs Amirovs	35fbcae	AESCrypt class for password encryption and decryption
Arturs Amirovs	afcd690	Changes to Event database, added feature for joining Private events, now it is possible to join, leave and delete event
Arturs Amirovs	8ddf10a	Improved MyProfileFragment
Arturs Amirovs	20351a4	Crete funtionality for joining, creating events, view for showing Joined/Created events, MyProfile View, integration with User databaseTable
Arturs Amirovs	376d2d7	Initial commit

9.1. att. ***VCS ekrānšāviņš.***

REZULTĀTI

Darba mērķis bija izstrādāt Android lietotni pasākumu organizēšanai. Sistēmai jānodrošina momentālo informācijas apmaiņu par pasākumu starp dažādām ierīcēm ar Firebase mākoņu glabātuves palīdzību un nodrošināt mākoņu glabātuves sinhronizāciju ar lokālo datu bāzi.

Rezultātā tika izstrādāta dokumentācija un lietotne uz tās pamata. Lietotnē ir Administrators ar pilnīgu kontroli par lietotnes datiem, lietotāji var veidot pasākumus, pie kuriem lietotāji no citām ierīcēm var pievienoties. Tāpat tika izveidoti instrumenti reklāmas pārvaldīšanai, kuriem piekļuve ir Reklāmas ievietotājam. Lietotnes kvalitāte bija nodrošināta ar regulāru testēšanu.

SECINĀJUMI

Laiks ir visvērtīgākais resurss, kurš ir mūsu rīcībā, bet ļoti bieži mēs to palaižam vējā, nezinot kā to pareizi pavadīt. Izstrādāta lietotne ir risinājums šai problēmai . Tas ir spējīgs lietotājam palīdzēt sameklēt nodarbi vien dažu minūšu laikā, kopā ar to, atrast vienprātējus un jaunus draugus.

Ja palaist lietotni, tam tāpat jābūt izstrādātam uz iOS platformas, tā kā Android lietotne aizņem tikai pusi no tirgus.

IZMANTOTĀS LITERATŪRAS SARAKSTS

1. Android Firebase dokumentācija. Pieejams: https://firebase.google.com/docs/ . Pārbaudīts: 20.05.2018.
2. Android izstrādātāju dokumentācija. Pieejams: https://developer.android.com/docs/ Pārbaudīts: 20.05.2018.
3. Material dizaina elementi. Pieejams: https://material.io/tools/icons/?style=baseline Pārbaudīts: 20.05.2018.
4. Darbietilpības novērtēšanas rīks COCOMO II Pieejams: http://csse.usc.edu/tools/COCOMOII.php. Pārbaudīts: 20.05.2018.
5. SQLite datubāzes dokumentācija: Room Persistence Library Pieejams: https://developer.android.com/topic/libraries/architecture/room Pārbaudīts: 20.05.2018.
6. PlaceHolderView bibliotēkas dokumentācija. Pieejams: https://github.com/janishar/PlaceHolderView Pārbaudīts: 20.05.2018.

PIELIKUMI

1.Pielikums. Lietotnes koda fragments: Klase MyProfileFragment

```
public class MyProfileFragment extends Fragment {
   private static final String FILTER_PUBLIC = "public";
   private static final String FILTER_PRIVATE = "private";

   private ListView eventsListView;
   private ViewAdapter adapter;
   private ArrayList<Event> eventsList, createdEvents, joinedEvents;
   private ToggleButton openToggleButton, privateToggleButton, headerOpenToggleButton,
headerPrivateToggleButton;
   private String filter = FILTER_PUBLIC;

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      View view = inflater.inflate(R.layout.fragment_my_profile, container, false);

      //Footer for list view
      View footer = inflater.inflate(R.layout.footer_event_list_view, container, false);
      //Header for Profile information above ListView
      View header = inflater.inflate(R.layout.my_profile_header, container, false);
      //Header with navigation buttons on top of ListView and header
      final View header2 = inflater.inflate(R.layout.my_profile_header_2, container, false);

      //Toggle buttons when header2 is not visible
      openToggleButton = (ToggleButton) view.findViewById(R.id.profile_open_toggle_button);
      privateToggleButton = (ToggleButton) view.findViewById(R.id.profile_private_toggle_button);

      //Toggle buttons when header2 is visible
      headerOpenToggleButton = (ToggleButton)
header2.findViewById(R.id.profile_header_open_toggle_button);
      headerPrivateToggleButton = (ToggleButton)
header2.findViewById(R.id.profile_header_private_toggle_button);
```

```
        //List of events that is shown in this view
        eventsListView = (ListView) view.findViewById(R.id.profile_list_view);

        eventsList = new ArrayList<>();
        createdEvents = new ArrayList<>();
        joinedEvents = new ArrayList<>();

        adapter = new ViewAdapter(getContext(), eventsList);
        eventsListView.setAdapter(adapter);

        EventsViewModel eventsViewModel = ViewModelProviders.of(this).get(EventsViewModel.class);

        //Initialize LiveData object for listening changes for created events of this user

eventsViewModel.setEventsForCurrentUser(PreferenceManager.getDefaultSharedPreferences(getContext()).getString(LOGGED_IN_USER_ID, null));
        //Listen for changes and update eventsList when new events are created
        eventsViewModel.getEventsForCurrentUser().observe(this, new Observer<List<Event>>() {
            @Override
            public void onChanged(@Nullable List<Event> events) {

                createdEvents.clear();
                createdEvents.addAll(events);

                //update ListView only if filter is FILTER_PUBLIC, because when filter is FILTER_PRIVATE other list is shown
                if(filter.equals(FILTER_PUBLIC)) {
                    eventsList.clear();
                    eventsList.addAll(createdEvents);
                    eventsList.addAll(joinedEvents);

                    adapter.notifyDataSetChanged();
                }
            }
        });

        //Initialize LiveData object for listening changes for joined events of this user
```

```
eventsViewModel.setJoinedEventsForCurrentUser(PreferenceManager.getDefaultSharedPreferences(
getContext()).getString(LOGGED_IN_USER_ID, null));
      //Listen for the changes and update EventsList when new events are joined or leaved
      eventsViewModel.getJoinedEventsForCurrentUser().observe(this, new Observer<List<Event>>()
{
         @Override
         public void onChanged(@Nullable List<Event> events) {
            joinedEvents.clear();
            joinedEvents.addAll(events);

            //update ListView only if filter is FILTER_PUBLIC, because when filter is
FILTER_PRIVATE other list is shown
            if(filter.equals(FILTER_PUBLIC)) {
               eventsList.clear();
               eventsList.addAll(createdEvents);
               eventsList.addAll(joinedEvents);

               adapter.notifyDataSetChanged();
            }
         }
      });



      eventsListView.addFooterView(footer);
      eventsListView.addHeaderView(header);
      eventsListView.addHeaderView(header2);

eventsListView.setDivider(Objects.requireNonNull(getContext()).getResources().getDrawable(R.draw
able.transperent_color));
      eventsListView.setDividerHeight(8);

      /*  Layouts for handling design related requirements.
         So that Toggle buttons are scrolled with list and at some point they freeze on top.
         Also top bar should decrease in size when list is scrolled */
      final RelativeLayout relativeLayout = (RelativeLayout)
view.findViewById(R.id.my_profila_top_bar);
```

```
final RelativeLayout backgroundRelativeLayout128dp = (RelativeLayout) view.findViewById(R.id.background_top_bar_128dp);
final RelativeLayout backgroundRelativeLayout28dp = (RelativeLayout) view.findViewById(R.id.background_top_bar_28dp);

//object for changing margins/padding of Layout
final ViewGroup.MarginLayoutParams layoutParams = (ViewGroup.MarginLayoutParams) eventsListView
        .getLayoutParams();


eventsListView.setOnScrollListener(new ListView.OnScrollListener() {
    @Override
    public void onScrollStateChanged(AbsListView view, int scrollState) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onScroll(AbsListView view, int firstVisibleItem,
                         int visibleItemCount, int totalItemCount) {
        //if listed is scrolled aka first list item is not visible, then show 28dp Layout, otherwise 128dp
        if (firstVisibleItem != 0) {
            relativeLayout.setVisibility(View.VISIBLE);
            header2.setVisibility(View.GONE);

            backgroundRelativeLayout128dp.setVisibility(View.GONE);
            backgroundRelativeLayout28dp.setVisibility(View.VISIBLE);

            layoutParams.setMargins(0, 0, 0, 0);
            eventsListView.setLayoutParams(layoutParams);

        } else {
            relativeLayout.setVisibility(View.GONE);
            header2.setVisibility(View.VISIBLE);

            backgroundRelativeLayout128dp.setVisibility(View.VISIBLE);
            backgroundRelativeLayout28dp.setVisibility(View.GONE);
```

```
                layoutParams.setMargins(0, 188, 0, 0);
                eventsListView.setLayoutParams(layoutParams);
            }
        }
    });


    openToggleButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if(b) {
                privateToggleButton.setChecked(false);
                headerPrivateToggleButton.setChecked(false);
                headerOpenToggleButton.setChecked(true);

                openToggleButton.setEnabled(false);
                headerOpenToggleButton.setEnabled(false);
                headerPrivateToggleButton.setEnabled(true);
                privateToggleButton.setEnabled(true);

                filter = FILTER_PUBLIC;
            }
            setListByFilter();
        }
    });

    headerOpenToggleButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if(b) {
                privateToggleButton.setChecked(false);
                headerPrivateToggleButton.setChecked(false);
                openToggleButton.setChecked(true);

                openToggleButton.setEnabled(false);
                headerOpenToggleButton.setEnabled(false);
```

```
            headerPrivateToggleButton.setEnabled(true);
            privateToggleButton.setEnabled(true);

            filter = FILTER_PUBLIC;
        }
        setListByFilter();
    }
});

    privateToggleButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if (b) {
                headerPrivateToggleButton.setChecked(true);
                headerOpenToggleButton.setChecked(false);
                openToggleButton.setChecked(false);

                openToggleButton.setEnabled(true);
                headerOpenToggleButton.setEnabled(true);
                headerPrivateToggleButton.setEnabled(false);
                privateToggleButton.setEnabled(false);

                filter = FILTER_PRIVATE;
            }
            setListByFilter();
        }
    });

    headerPrivateToggleButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if(b) {
                privateToggleButton.setChecked(true);
                headerOpenToggleButton.setChecked(false);
                openToggleButton.setChecked(false);
```

```
                openToggleButton.setEnabled(true);
                headerOpenToggleButton.setEnabled(true);
                headerPrivateToggleButton.setEnabled(false);
                privateToggleButton.setEnabled(false);

                filter = FILTER_PRIVATE;
            }
            setListByFilter();
        }
    });

    openToggleButton.setChecked(true);




    return view;
}

private void setListByFilter(){
    if(filter.equals(FILTER_PUBLIC)){
        eventsList.clear();
        eventsList.addAll(createdEvents);
        eventsList.addAll(joinedEvents);

        adapter.notifyDataSetChanged();
    } else {
        //If filters is "Private" then list is filled with private events related to this user
        eventsList.clear();

eventsList.addAll(AppDatabase.getAppDatabase(getContext()).eventDao().getPrivateEventsForUser(
PreferenceManager.getDefaultSharedPreferences(getContext()).getString(LOGGED_IN_USER_ID,
null)));

eventsList.addAll(AppDatabase.getAppDatabase(getContext()).eventDao().getJoinedPrivateEventFor
User(PreferenceManager.getDefaultSharedPreferences(getContext()).getString(LOGGED_IN_USER_
ID, null)));
```

```
            adapter.notifyDataSetChanged();
        }
    }

    private class ViewAdapter extends BaseAdapter implements ListAdapter {
        Context listContext;
        private List<Event> eventsList;

        public ViewAdapter(Context context, List<Event> eventsList){
            listContext = context;
            this.eventsList = eventsList;
        }

        @Override
        public int getCount() {
            return eventsList.size();
        }

        @Override
        public Object getItem(int i) {
            return eventsList.get(i);
        }

        @Override
        public long getItemId(int i) {
            return i;
        }

        private class ViewHolder {
            ImageView eventImage;
            TextView eventName;
            TextView eventLocation;
            TextView eventNumberOfParticipants;
            RelativeLayout listItemContainer;
            ImageButton acceptButton, declineButton;
            TextView creator;
        }
```

```
    @Override
    public View getView(final int position, View view, ViewGroup viewGroup) {
        final ViewHolder holder;
        View viewInf = view;

        if(viewInf == null){
            viewInf = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.event_list_item,
viewGroup, false);
            holder = new ViewHolder();

            holder.eventImage = (ImageView) viewInf.findViewById(R.id.list_event_image_view);
            holder.eventName = (TextView) viewInf.findViewById(R.id.list_event_name);
            holder.eventLocation = (TextView) viewInf.findViewById(R.id.list_event_location);
            holder.eventNumberOfParticipants = (TextView)
viewInf.findViewById(R.id.list_event_number_of_participants);
            holder.listItemContainer = (RelativeLayout)
viewInf.findViewById(R.id.list_item_container);
            holder.creator = (TextView) viewInf.findViewById(R.id.event_creator);

            viewInf.setTag(holder);
        } else {
            holder = (ViewHolder) viewInf.getTag();
        }

        holder.eventName.setText(eventsList.get(position).getName());
        holder.eventLocation.setText(eventsList.get(position).getLocation());

holder.eventNumberOfParticipants.setText(getResources().getString(R.string.my_profile_fragment_ev
ent_participant_number, String.valueOf(eventsList.get(position).getNumberOfParticipantsMin()),
String.valueOf(eventsList.get(position).getNumberOfParticipantsMax())));

        User creator =
AppDatabase.getAppDatabase(listContext).userDao().getUserByUid(eventsList.get(position).getCreat
or());
        if(creator != null)
            holder.creator.setText(getResources().getString(R.string.event_creator_name,
creator.getFirstName(), creator.getLastName()));
```

```
        /*Temporary solution while there is no server for storing images.
        Image is taken from internal storage and user can select image by selecting its name in a Spinner*/
        if (eventsList.get(position).getImageName() != null) {
            switch (eventsList.get(position).getImageName()) {
                case "bowling":
                    holder.eventImage.setImageResource(R.drawable.bowling_image);
                    break;
                case "club":
                    holder.eventImage.setImageResource(R.drawable.club);
                    break;
                case "escape_room":
                    holder.eventImage.setImageResource(R.drawable.escape_room);
                    break;
                case "hockey":
                    holder.eventImage.setImageResource(R.drawable.hockey);
                    break;
                case "hookah":
                    holder.eventImage.setImageResource(R.drawable.hookah);
                    break;
                case "sigulda":
                    holder.eventImage.setImageResource(R.drawable.sigulda);
                    break;
                default:
                    holder.eventImage.setImageResource(R.drawable.image_placeholder);
            }
        }
        //Listener for clicking on a listItem
        holder.listItemContainer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(listContext, EventDetailsActivity.class);
                //Selected event id is passed to EventDetails view, so all event details could be retrived from database
                intent.putExtra(SELECTED_EVENT_ID, eventsList.get(position).getId());
                startActivity(intent);
            }
        });
```

```
            return viewInf;
        }
    }
}
```

2. **Pielikums. Koda fragmenti.**

```
@Entity(tableName = "events", foreignKeys = @ForeignKey(entity = User.class,
        parentColumns = "user_uid",
        childColumns = "event_creator",
        onDelete = CASCADE))
public class Event {

    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "event_id")
    private long id;
    @ColumnInfo(name = "event_name")
    private String name;
    @ColumnInfo(name = "event_location")
    private String location;
    @ColumnInfo(name = "event_number_of_participants")
    private int numberOfParticipants;
    @ColumnInfo(name = "event_image_name")
    private String imageName;
    @ColumnInfo(name = "event_viewed")
    private Boolean viewed = false;
    @ColumnInfo(name = "event_creator")
    private String creator;


    public Event(){}

    public String getCreator() {
        return creator;
    }

    public void setCreator(String creator) {
        this.creator = creator;
    }
    ….
```

```
}
```

DAO fragments:

```
@Dao
public interface EventDao {

    @Query("SELECT * FROM events")
    LiveData<List<Event>> getAllEvents();

//    @Query("SELECT * FROM events where event_id LIKE  :id")
//    Event findById(int id);

    @Query("SELECT COUNT(*) from events")
    int countEvents();

    @Insert
    void insertAllEvents(Event... events);

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    long insertEvent(Event event);

    @Delete
    void delete(Event event);

    @Query("SELECT * from events WHERE event_id = :id")
    Event getEventById(long id);

    @Query("UPDATE events SET event_viewed=:status WHERE event_id = :eventId")
    void updateEvent(Boolean status, Long eventId);

    @Query("SELECT * FROM events WHERE event_viewed = 'false'")
    LiveData<List<Event>> getAllNotViewedEvents();

    @Query("SELECT * FROM events WHERE event_creator=:userId")
    LiveData<List<Event>> getEventsForUser(String userId);
```

```
    @Query("SELECT * FROM events INNER JOIN event_participants ON
event_participants.event_id = events.event_id " +
            "INNER JOIN users ON users.user_uid = event_participants.user_id WHERE user_id
=:userUid")
    LiveData<List<Event>> getJoinedEventForUser(String userUid);

}
```

DB fragments:

```
@Database(entities = {User.class, Event.class, EventParticipants.class}, version = 2)
public abstract class AppDatabase extends RoomDatabase {

    private static AppDatabase INSTANCE;

    public abstract EventDao eventDao();
    public abstract UserDao userDao();
    public abstract EventParticipantsDao eventParticipantsDao();

    public static AppDatabase getAppDatabase(Context context) {
        if (INSTANCE == null) {
            INSTANCE =
                    Room.databaseBuilder(context.getApplicationContext(),
AppDatabase.class,
"leisure-maxx-database")
                            // allow queries on the main thread.
                            // Don't do this on a real app! See PersistenceBasicSample for an example.
                            .allowMainThreadQueries()
                            .build();
        }
        return INSTANCE;
    }

    public static void destroyInstance() {
        INSTANCE = null;
    }}
```

Kvalifikācijas darbs „***Lietojumprogrammatūra pasākumu organizēšanai***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Artūrs Amirovs*** _____ **___.05.2018**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: ***M.dat Emil Syundyukov***_____ **___.05.2018.**

Recenzents: ***M.dat. Pēteris Krastiņš***

Darbs iesniegts 28.05.2018.
Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2018. prot. Nr. _____

Komisijas sekretārs(-e): _____