**Технически университет-София**

**Програмиране на мобилни устройства**

***Курсов проект***

**"Приложение за гласуване, анкети, референдуми"**

**Разработил:** **Одобрил:**  Георги Иванов

Факултет: ФКСТ

Специалност: КСИ

фак.№: 121215116

Група: 37 Дата: 16.09.2018г.

**Съдържание**

**Увод**……………………………………………………………………

**Първа глава** - Анализ на съществуващи разработки……………

1.1 myvote.io……………………………………………………..

1.2 voxvote.com…………………………………………………

1.3 doodle.com………………………………………………..

**Втора глава** - Проектиране………………………………………..

2.1 Изисквания към продукта……………………………….

2.2 Избор на език и среда……………………………………

2.3 Кой ще използва продукта?……………………………..

2.4 Какви данни ще се използват?…………….……………

2.5 Как ще бъдат достъпни функционалностите от потребителя?………………………………………………

**Трета глава** - Реализация………………………………………….

3.1 Структура на проекта……………………………………

3.2 Описание на моделите(основните класове) на приложението………………………………………….

3.2.1 Vote.java…………………………………………

3.2.2 Question.java……………………………………

3.2.3 Referendum.java…………………………………….

3.2.4 Voting.java………………………………………….

3.2.5 Poll.java…………………………………………..

3.3 AppController……………………………………………….

3.4 Описание на activity-тата в приложението……………...

3.4.1 HomeActivity.java……………………………………

3.4.1.1 Метод protected void onCreate(Bundle savedInstanceState)…………………………..

3.4.1.2 Метод private boolean isNetworkConnected()………………………..

3.4.1.3 Метод private void setUpVotes()……………

3.4.1.4 Метод private void loadDataFromFiles(final String votesPathFile, final Class voteType)…..

3.4.1.5 Метод public boolean onNavigationItemSelected(MenuItem item)…

3.4.1.6 Метод public void onBackPressed()………..

3.4.2 VotingActivity.java…………………………………...

3.4.2.1 Метод private void loadListFragment()…..

3.4.2.2 Метод public void onItemSelected (Vote vote)………

3.4.2.3 Метод private void loadFragment(Fragment fragmentToLoad, Vote vote)……………………………………..

3.4.2.4 Метод public void onBackPressed()……….

3.5 Описание фрагментите в приложението…………………

3.5.1 ListFragment………………………………………..

3.5.1.1 Метод public void onCreate(Bundle savedInstanceState)………………………..

3.5.1.2 Метод private ArrayList<Vote> getVotesByType(Class type)……………..

3.5.1.3 Под-клас VotesAdapter…………………..

3.5.2 TakeReferendumFragment………………………..

3.5.3 TakeVotingFragment………………………………

3.5.4 TakePollFragment…………………………………

3.5.5 ReferendumResultsFragment……………………….

3.5.6 VotingResultsFragment…………………………….

3.5.7 PollResultsFragment……………………………….

**Четвърта глава** - Потребителско ръководство

4.1 Начин на инсталация…………………………………….

4.2 Нужни ресурси………………………………………….

**Заключение**……………………………………………………….

**Използвана литература и ресурси**……………………………..

**Увод**

С развиването на модерните технология и интернет, в последните двадесет години, устройства, лаптопи и смарт-телефони стават все по-голяма част от живота на средностатистическия човек, до степен до която тяхната липса е вече немислима възможност. Прогресирането на ИТ и софтуерната сфера е толкова силно, че буквално ежедневно се създават нови приложения, които да улеснят живота на човека в модерния свят, замествайки вече остарелите и ненужни технология на вчерашния свят. Google Maps и приложения от подобен характер, за локализация и изготвяне на маршрути, отдавна замениха GPS-сите, които всички ползвахме преди десетина години. Хартиените карти са демоде, което е станало трудно за намиране. Човек няма нужда да чете вестници или телевизия, когато всички новини са на един клик разстояние и могат да бъде наблюдавани в реално време. Streaming платформи като Spotify и Apple Music отдавна направиха ненужно свалянето на музика в хранилището на телефона или записването на дискове, за да може да слушаме музика докато се разхождаме или пътуваме с кола.

Примерите са безкрайни и се увеличават неимоверно всяка секунда.

Технологиите фундаментално са навлезли и в политическата система на много държави и макар, че все още не е силно разпространено е напълно сигурно, че след години ще можем да гласуваме на следващите избори/референдум от комфорта на собствения си дом, забравяйки за дългите опашни в кварталните училища, безкрайното лутане докато намерим стаята, която отговаря на нашия жилищен регион, и тъмните стаички, в които се крием за да дадем своят вот. Вместо това ще можем с един клик да дадем своя глас и да продължим спокойно деня си, без да трябва да правим планове, кога да отидем да гласуваме, така че да приключим най-бързо и безболезнено. Комфорта обаче няма да е единствения плюс на подобна технология. По-голямата сигурност предоставена от софтуерното бъдеще ще надвишава всяка такава на настоящето. Неща като фалшифициране на вотове и човешки грешки ще бъдат премахнати или поне сведени до минимум. Тоновете хартия, които се използват и изхвърлят при всички избори вече няма да са нужни, по-този начин добавяйки и един екологичен плюс в полза на модерното.

Целта на курсовия проект е не да създаде това бъдеще, а да покаже колко лесно и удобно е за създаване едно приложение за гласуване.???

**Първа глава**

**Анализ на съществуващи разработки**

Идеята за подобно приложение не е нова и можем лесно да намерим други подобни разработки в интернет. Ето няколко примера, изтъквайки техните положителни и отрицателни страни:

**1.1** **myvote.io**

Web базирано приложение за гласуване, което позволява създаването на гласувания и анкети, и наблюдаването на им в реално време. Има версии за iOS и Android, поддържа се както на мобилни устройства, така и на таблети и персонални компютри.

**1.2 voxvote.com**

Мобилно приложение за гласуване. Използва се най-често за задаване на въпроси по време на реални събития като конференции и срещи. Водещият на срещата създава гласуването, а участващите в публиката, чрез приложението, отговарят или гласуват на зададения от него въпрос.

Минус е факта, че можеш да имаш само по десет въпроса на създаден event и имаш право само на 5 event-а. За наличието на повече се заплаща

**1.3 doodle.com**

Doodle е уеб-сайт за създаване на анкети. Негов голям плюс е, че е можеш да продължаваш да ползваш безплатните му функции, колкото често искаш, без да се налага заплащане. Дизайна му е доста минималистичен и “user friendly”. Не е нужно регистрирането на акаунт и предоставянето на мобилен телефон или и-мейл адрес за неговото ползване, което го прави лесно за употреба

**Втора глава**

**Проектиране**

В следната глава ще се разгледат изискванията към продукта, както и важни въпроси свързани с неговото проектиране.

**2.1 Изисквания към продукта**

Да се създаде приложение за Android устройства, написано на Java, което да позволява на ползващия го да дава своя вот в гласувания, анкети и референдуми, намиращи се в приложението. Приложението трябва да:

* + Има Login страницата, чрез която потребителите да влизат в системата.
  + Има както нормални потребители, така и администратори.
  + Има лесно и удобно меню за навигиране между отделните вотове.
  + Има отделни страници за всеки вид вот.
  + Позволява на потребителя да дава своя вот в дадено гласуване.
  + Позволява на потребителя да види общите резултати от гласуването както в проценти, така и в прости цифри.
  + Позволява на администраторите да създават нови анкети, гласувания и референдуми.
  + Праща push нотификация, когато нов вот е създаден.

**2.2 Избор на език и среда**

Езикът избран за създаването на приложението е Java, а средата - Android Studio. То е най-разпространената комбинация от език и среда за създаване на Android приложение, която с години се е доказвала отново и отново като водещ метод за разработка на мобилни приложения за устройства с операционна система Android.

Избраната база данни е SQLite, тъй като лесна за интегриране в андройд приложения.

Предишен опит в ползването както на езика, така и на средата, също изигра важна роля в тяхното избиране.

**2.3 Кой ще използва продукта?**

Продукта може да се използва както от хора, които искат просто да дадат своя глас в неангажираща анкета, така и от хора, които искат да дадат своя вот във важен референдум.

**2.4 Какви данни ще се използват?**

Данната, в своята основа, ще бъде вот. Вот, който сам по себе си трябва да представлява въпрос и възможни опции за отговор, от които потребителя трябва да избере една. Вотът трябва да има три разновидности:

* + Гласуване - В гласуването трябва да имаме един общ въпрос и N на брой опции за възможен отговор. От тях потребителя може и трябва да избере само една.
  + Анкета - В анкетата трябва да имаме едно общо заглавие и N на брой въпроси, всеки от които трябва да има M на брой опции за отговор. Потребителя може и трябва да избере само по една опция, като отговор на всеки един въпрос от анкетата за да може да гласува успешно.
  + Референдум - В референдума трябва да имаме един въпрос и два възможни отговора - “Да” и “Не”, от които потребителя да избере само един.

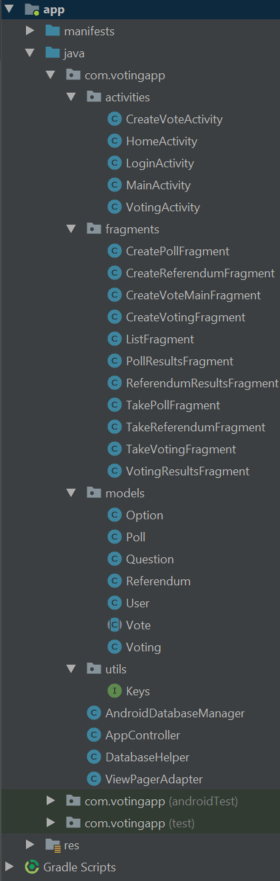
**2.5 Как ще бъдат достъпни функционалностите от потребителя?**

**Трета глава**

**Реализация**

В тази глава ще разгледаме как е реализиран проекта. Ще покажем обектите, които изграждат приложението, activity-тата, от който се състои VotingApp, и фрагментите, който се използват. Ще дадем примери и за потребителския интерфейс, валидации и програмен код.

**3.1 Структура на проекта**



***Фиг. 3.1.*** *Структура на проекта в Android Studio*

Основните папки и файлове в проекта са:

* java - com.votingapp – В нея се намират всички java файлове използвани в приложението.
  + В тази папка се съдържа и AppController.java- който е отговорен за създаване на самото приложение, DatabaseHelper – грижещ се за базата данни SQLite частта на приложението.
  + activities - В нея се съдържат acitivity-тата, които изграждат приложението. Това са HomeActivity.java, VotingActivity.java, MainActivity.java, LoginActivity.java, CreateVoteActivity.java
  + fragments – В нея се намира файла ListFragment.java, който се използва за показване на списъка от гласувания/анкети/референдуми в VotingActivity.java. Файловете TakeVotingFragment.java, TakePollFragment.java и TakeReferendumFragment.java, които се използват за даване на вот в гласуване, анкети и референдум респективно. Файловете VotingResultsFragment.java, PollResultsFragment.java и ReferendumResultsFragment.java които се използват за показване на резултати от даденото гласуване, анкета или референдум.
  + models – В тази папка се съдържат класовете-модели, от които се изграждат обектите изграждащи приложението. Това са Option.java, Poll.java, Question.java, Referendum.java, UserProfile.java, Vote.java, Voting.java.
  + utils – В utils се намира Key.java, който съдържа всички ключове използвани за комуникиране и изпращане на информация между activity-та и фрагменти.
* res – Папка с ресурсите на приложението.
  + drawable – Съдържа всички xml икони, които се използват предимно за NavigationDrawer-а.
  + layout – В тази папка се намират всички layout файлове, които изграждат потребителския интерфейс на всяко activity и фрагмент.
  + menu – В нея се намира съдържанието на основното меню, което се визуализира под формата на NavigationDrawer.
  + mipmap – Съдържа иконата за стартиране на приложението.
  + values – Папката съдържа всички цветове, стилове, размери и низове нужни на приложението.

**3.2 Описание на моделите(основните класове) на приложението**

**3.2.1 Vote.java**

Най-основния модел в приложението е абстрактният клас Vote.java. Неговата идея е да бъде изграждащата частица на всеки вид вот(гласуване/анкета/референдум). Класът е абстрактен и не могат да бъдат създавани обекти директно от него. Той служи единствено като основа! Vote.java съдържа поле “title” от тип String, което държи заглавието на вота, get-ър и set-ър за полети, default конструктор, както и конструктор за инициализиране на title.

**3.2.2 Question.java**

Двата основни елемента, които изграждат един вот са – въпрос и възможни отговори. Класът Question представлява точно този въпрос. В сегашния си вариант класа съдържа само едно поле от тип String, което държи текста на въпроса(questionText), get и set методи за инициализира и достъпване на член-променливата и два конструктора: един празен и един инициализиращ questionText.

Моля да се има предвид, че questionText полето на Question.java не се припокрива с полето title на Vote.java, тъй като заглавието на вота и текста на въпроса могат да се различават. Също така, когато имаме анкета имаме едно генерално заглавие на анкетата, което се пази в title на Vote.java, а отделните въпроси в анкета се пазят в член-променливите questionText на обектите от тип Question(За повече информация виж 3.2.5 Poll.java)

В бъдещи разработки могат да се добавят и допълните полета към класа Question, като например:

* important – Поле, което да означава, че въпроса е от висока важност.
* hot – Променлива, която да показва, че много хора гласуват в даден вот.
* category – Член, който да разделя въпросите в различни видове категории.

**3.2.3 Option.java**

Другия основен елемент на всеки вот – възможните отговори е представен чрез класа Option.java. Класа съдържа текста на дадената опция(optionText), броя пъти, които е избран този отговор(int променливата timesSelected) и променлива, която обозначава дали опцията е избрана от текущия потребител(selectedByCurrentUser).

Класът съдържа метода *public void increaseTimesSelected(),* който се вика, когато някой избере дадената опция във вот. Той инкрементира променливата timesSelected, по този начин показвайки, че опцията е избрана още веднъж от даден потребител.

**3.2.3 Referendum.java**

Класът Referendum е репрезентацията на референдум, на който се гласува с „Да“ и „Не“. Класа наследява абстрактния(основния) клас Vote.java и съдържа обект от тип Question, в който се пази информацията за въпроса на референдума и два член-променливи обекта от тип Option:

* optionYes – Опцията „Да“. Съгласяване с въпроса зададен на референдума.
* optionNo - Опцията „Не“. Несъгласие с референдума.

**3.2.4 Voting.java**

Voting класа на приложението представлява гласуване и наследява основния клас Vote.java. Надгражда го като добавя член-променлива от тип Question и списък от възможни отговори(ArrayList от Option). Това се налага, тъй като дадено гласуване не е ограничено от брой възможни отговори.

**3.2.5 Poll.java**

Моделът Poll.java репрезентира анкета. Както и предишните два вида вот, той също наследява абстрактния клас Vote.java. Класът има една член-променлива “pollContent” тип HashMap<Question, ArrayList<Option>>. Това е Map, който като ключ ползва обект от тип Question(въпрос), а като стойност има свързан списък от възможни отговори(Option)

**3.3 AppController**

AppController-а е класа, който стартира приложението. Той extend-ва класа Application и е активен през целия живот на приложението. AppController-а е пълен със статични член-променливи и методи, които ние можем да достъпваме по всяко време. Можем да гледаме на AppController-а като на място за глобални методи и променливи. В него намираме ArrayList<Vote> на име “votes”. Променлива, в която се съхраняват всички вотове без значение дали са референдуми, анкети или гласувания. Това става лесно и възможно благодарение на полиморфизма, който ни позволява Java.

Имаме метода onCreate(), който създава приложението и го стартира.

Имаме и метод public static int getTotalNumberOfVotes(ArrayList<Option> options), които при подаден списък от опции калкулира общия брои на всички гласове(Сбор от всички гласували за всички опции). Метода се използва когато се смятат процентите при извеждане на резултатите от даден вот.

**3.4 Описание на activity-тата в приложението**

**3.4.1 HomeActivity.java**

HomeActivity.java е activity-то, от което започва работа приложението. То е първото activity, което се зарежда, когато пуснем приложението. В него има toolbar където е изписано името на приложението, странично меню(NavigationDrawer), надпис ??? И бутон за обновяване на гласуванията/анкетите/референдумите. Следва да разгледаме по-важните методи по отделно.

**3.4.1.1 Метод protected void onCreate(Bundle savedInstanceState)**

Този метод се извиква автоматично при стартирането на activity-то. В него първо се инициализира toolbar-а и NavigationDrawer-a(Страничното меню)

*Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);*

*setSupportActionBar(toolbar);*

*DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);*

*ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close);*

*drawer.setDrawerListener(toggle);*

*toggle.syncState();*

*NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view);*

*navigationView.setNavigationItemSelectedListener(this);*

Създава се onClickListener на бутона за обновяване на вотовете

*Button loadDataButton = (Button) findViewById(R.id.load\_data\_button);*

*loadDataButton.setOnClickListener(new View.OnClickListener() {*

*@Override*

*public void onClick(View view) {*

*setUpVotes();*

*}*

*});*

**3.4.1.2 Метод private boolean isNetworkConnected()**

Метод, който връща дали Android устройството ни има връзка с интернет. Връща true при наличие на връзка и false при липса на такава.

**3.4.1.3 Метод private void setUpVotes()**

Методът първоначално извиква isNetworkConnected(). При липсата на интернет връзка вади съобщение.

*CharSequence text = "Липса на интернет връзка...";*

*Toast toast = Toast.makeText(this, text, Toast.LENGTH\_SHORT);*

*toast.show();*

При наличието на връзка с интернет извиква метода loadDataFromFiles(final String votesPathFile, final Class voteType) с трите link-а към трите файла съдържащи информацията за съответните видове вотове.

**3.4.1.4 Метод private void loadDataFromFiles(final String votesPathFile, final Class voteType)**

Метод, който по подаден линк към файл намиращ се в Google Drive и тип на вота, който да очаква от този файл. Зарежда, парсва и налива информацията в приложението. Информацията във файловете е в JSON формат. Пример:

*[*

*{*

*"title" : "Да се прекрати ли мандата на правителството?",*

*"optionNo" :*

*{*

*"optionText" : "Не",*

*"timesSelected" : 523*

*},*

*"optionYes" :*

*{*

*"optionText" : "Да",*

*"timesSelected" : 324*

*},*

*"question" :*

*{*

*"questionText" :"Да се прекрати ли мандата на правителството?"*

*}*

*},*

*……*

*/\* Файлът продължава натам подобаващо \*/*

Във всеки файл се съдържа масив от обекти кореспондиращи съответно на класовете в папка model за гласувания, референдуми и анкети. Информацията се взима от файловете под формата на String съдържащ JSON. Той бива парснат по следния начин:

*ObjectMapper mapper = new ObjectMapper();*

*Voting[] votings = mapper.readValue(json, Voting[].class);*

*AppController.votes.addAll(Arrays.asList(votings));*

Резултата се присвоява на масив от съответния тип вот и се добавя към всички вотове в AppController.votes.

**3.4.1.5 Метод public boolean onNavigationItemSelected(MenuItem item)**

Метод, който се извиква когато бъде избрана опция от страничното меню. Методът стартира желаното activity от потребителя.

*Intent votingActivityIntent = new Intent(HomeActivity.this, VotingActivity.class);*

*votingActivityIntent.putExtra(Keys.VOTING\_ACTIVITY\_FRAGMENT, Keys.LIST\_VOTINGS);*

*startActivity(votingActivityIntent);*

**3.4.1.6 Метод public void onBackPressed()**

Метода се активира, когато се натисне бутона за връщане назад на Android устройството. Ако в момента на натискане на бутона е отворен NavigationDrawer-а(страничното меню) - то се затваря. В противен случай се затваря приложението

*DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);*

*if (drawer.isDrawerOpen(GravityCompat.START)) {*

*drawer.closeDrawer(GravityCompat.START);*

*} else {*

*super.onBackPressed();*

*}*

**3.4.2 VotingActivity.java**

VotingActivity е activity-то, което отговаря за всички видове вотове и нещата, които се случват с тях. Неговите основни методи са

**3.4.2.1 Метод private void loadListFragment()**

Метода се извиква от метод onCreate на activity-то при неговото създаване. Когато се създаде VotingActivity към него се предава чрез Keys типа на вот, чиито списък да се визуализира в activity-то. Първото нещо, което loadListFragment() прави е да вземе този тип и да го сложи в Bundle, който после да предаде на ListFragment при неговото създаване и да замени FrameLayout-а на activity\_voting.xml с ListFragment-а.

*String defaultFragment = (String) getIntent().getSerializableExtra(Keys.VOTING\_ACTIVITY\_FRAGMENT);*

*Bundle bundle = new Bundle();*

*if (Keys.LIST\_VOTINGS.equals(defaultFragment)) {*

*bundle.putSerializable(Keys.VOTING\_ACTIVITY\_FRAGMENT, Keys.LIST\_VOTINGS);*

*} else if (Keys.LIST\_POOLS.equals(defaultFragment)) {*

*bundle.putSerializable(Keys.VOTING\_ACTIVITY\_FRAGMENT, Keys.LIST\_POOLS);*

*} else if (Keys.LIST\_REFERENDUMS.equals(defaultFragment)) {*

*bundle.putSerializable(Keys.VOTING\_ACTIVITY\_FRAGMENT, Keys.LIST\_REFERENDUMS);*

*}*

*ListFragment listFragment = new ListFragment();*

*listFragment.setArguments(bundle);*

*FragmentTransaction transaction = getFragmentManager().beginTransaction();*

*transaction.replace(R.id.list\_content\_fragment, listFragment, "listFragment");*

*transaction.commit();*

**3.4.2.2 Метод public void onItemSelected(Vote vote)**

Методът отговаря за това, което се случва, когато се избере вот от VotesAdapter-а на ListFragment-a(Моля погледнете ???). onItemSelected проверява типа на избрания вот и дали потребителя вече не е гласувал по същия въпрос. Ако не е - извиква метода loadFragment с нов фрагмент за гласуване. Ако потребителя вече е гласува тогава пак се извиква горния метод, но с фрагмент за показване на резултати.

**3.4.2.3 Метод private void loadFragment(Fragment fragmentToLoad, Vote vote)**

Метода приема като аргументи фрагмента, който да зареди и вота, който да му предаде. Той взима вота и го слага в Bundle, който предава на фрагмента, след което заменя FrameLayout-а на activity\_voting.xml с желания фрагмент.

*Bundle bundle = new Bundle();*

*bundle.putSerializable(Keys.VOTE\_OBJECT, vote);*

*fragmentToLoad.setArguments(bundle);*

*transaction = getFragmentManager().beginTransaction();*

*transaction.replace(R.id.list\_content\_fragment, fragmentToLoad);*

*transaction.commit();*

**3.4.2.4 Метод public void onBackPressed()**

Метода отговаря за това, което се случва при натискането на бутона за връщане назад. Проверява дали активния в момента фрагмент е listFragment. Ако това е така метода приключва activity-то и връща потребителя назад към HomeActivity. В противен случай, когато фрагмента не е listFragment, се зарежда отново listFragment. Това е нужно за да се избегне възможността след като се гласува по даден въпрос и се покажат резултатите, потребителя да се върне отново на фрагмента за гласуване.

*Fragment currentFragment = getFragmentManager().findFragmentByTag("listFragment");*

*if (currentFragment != null && currentFragment.isVisible()) {*

*finish();*

*}else {*

*loadListFragment();*

*}*

**3.5 Описание фрагментите в приложението**

Всички фрагменти работят на принципа, че заменят FrameLayout-а на activity\_voting.xml на VotingActivity. Всички фрагменти “живеят” върху VotingActivity.

**3.5.1 ListFragment**

Фрагмента се използва за визуализиране на списъка от гласувания/анкети/референдуми. Основните му методи и начин на работа са

**3.5.1.1 Метод public void onCreate(Bundle savedInstanceState)**

В него от аргументите се взима, под формата на String, типа вот, който потребителя иска да визуализира(анкета/референдум/гласуване). На базата на този тип се сменя заглавието на activity-то в toolbar-а, така че то да отговаря на типа вот. Извиква се метода getVotesByType(Class type), който връща списък от всички вотове по подаден тип. Списъка се добавя в VotesAdapter-а на ListFragment(Моля вижте ???)

**3.5.1.2 Метод private ArrayList<Vote> getVotesByType(Class type)**

Метода получава променлива от тип Class. Взима всички вотове, намиращи се във AppController.votes, итерира ги и тези, отговарящи на подадения тип, добавя в отделен списък. Накрая връща списъка за по-нататъшно използване.

*ArrayList<Vote> votesByType = new ArrayList();*

*AppController.getInstance();*

*Iterator it = AppController.votes.iterator();*

*while (it.hasNext()) {*

*Vote vote = (Vote) it.next();*

*if (vote.getClass().equals(type)) {*

*votesByType.add(vote);*

*}*

*}*

*return votesByType;*

**3.5.1.3 Под-клас VotesAdapter**

Идеята на VotesAdapter е да съдържа в себе си списъка от вотове, които се показват в фрагмента. Класа extend-ва ArrayAdapter<Vote> и съдържа член променливите Context context и List<Vote> values. Използва list\_view\_item.xml отговарящ за все отделен елемент в списъка като на него set-ва заглавието на всеки вот в метод public View getView(int position, View convertView, ViewGroup parent).

**3.5.2 TakeReferendumFragment**

Фрагмента, който се използва, когато потребителя иска да гласува на даден референдум. XML-а отговарящ за дизайна на фрагмента е fragment\_take\_referendum.xml. В него имам TextView, което държи заглавието на референдума, RadioGroup-а от два RadioButton-а, както и бутон за гласуване в референдума. RadioButton-ните са за двата възможно отговора на референдума - “Да” и “Не”.

Избраният от списъка референдум се взима от аргументите чрез

*final Referendum referendum = (Referendum) getArguments().getSerializable(Keys.VOTE\_OBJECT);*

При натискане на бутона “Гласувай” първо се взима избрания от двата RadioButton-а. Ако нито един не е избран се изважда съобщение за грешка:

*CharSequence text = "Моля отговорете на референдума!";*

*Toast toast = Toast.makeText(getActivity(), text, Toast.LENGTH\_SHORT);*

*toast.show();*

Ако има избрана опция се взима текста на селектирания RadioButton и се проверява дали тя е “Да” или “Не”. В зависимост от това се изпълнява или

*referendum.getOptionYes().increaseTimesSelected();*

*referendum.getOptionYes().setSelectedByCurrentUser(true);*

или

*referendum.getOptionNo().increaseTimesSelected();*

*referendum.getOptionNo().setSelectedByCurrentUser(true);*

Първия ред инкрементира съответно опция “Да” или “Не” на референдум обекта, по този начин зачитайки гласуването на потребителя. Вдига се и един флаг на име selectedByCurrentUser, който показва коя опция е избрал гласуващия. Този флаг се използва при визуализиране на резултатите в следващия фрагмент.

Следва създаване на ReferendumResultsFragment, Bundle, в който се слага referendum обекта и зареждане на новосъздадения фрагмент.

**3.5.3 TakeVotingFragment**

Това е фрагмента, който отговаря за гласуванията. Неговия интерфейс се определя от fragment\_take\_voting.xml.

Взимаме гласуването от аргументите

*final Voting voting = (Voting) getArguments().getSerializable(Keys.VOTE\_OBJECT);*

Слагаме името на референдума

*((TextView) view.findViewById(R.id.votingТitle)).setText(voting.getTitle());*

*getActivity().setTitle(voting.getTitle());*

Завъртаме цикъл минаващ през всички опции, които има потребителя като отговори и ги добавяме към интерфейса под формата на RadioButton-и в RadioGroup-а.

*final RadioButton[] options = new RadioButton[voting.getOptions().size()];*

*for (int i = 0; i < voting.getOptions().size(); i++) {*

*options[i] = new RadioButton(getActivity());*

*options[i].setLayoutParams(new LinearLayout.LayoutParams(-1, -1));*

*options[i].setText((voting.getOptions().get(i)).getOptionText());*

*radioGroup.addView(options[i]);*

*}*

*takeVotingLinearLayout.addView(radioGroup);*

Тъй като в fragment\_take\_voting.xml имаме LinearLayout това ни позволява да добавяме елементи, който да се редят един под друг. Това става възможно чрез избиране на ориентацията на LinearLayout-а.

*android:orientation="vertical"*

Добавяме и бутон за гласуване

*Button saveButton = new Button(getActivity());*

*saveButton.setLayoutParams(new LinearLayout.LayoutParams(-1, -1));*

*saveButton.setText("Гласуване");*

*takeVotingLinearLayout.addView(saveButton);*

При натискане на бутона се проверява дали е избрана опция и се изкарва съобщение при липсата на избор.

Проверяват се всички опции, чрез завъртането на цикъл, и на тази, която отговаря на избраната от потребителя, и се инкрементира променливата брояща броя пъти, в които е избрана опцията и флага за избрана опция става true.

*for (Option option : voting.getOptions()) {*

*if (option.getOptionText().equals(radioButton.getText())) {*

*option.increaseTimesSelected();*

*option.setSelectedByCurrentUser(true);*

*}*

*}*

Идентично като в TakeReferendumFragment се създава нов фрагмент за резултат(в случая VotingResultsFragment), който се стартира, след като му се подаде Bundle с гласуването

**3.5.4 TakePollFragment**

Това е фрагмента за гласуване в анкети. Той работи по идентичен начин с фрагмента за гласувания с изключение, че освен добавяне на всички опции от вота на екрана на потребителя, се добавят всички въпроси в анкетата и всички опции на всички въпроси под формата на RadioGroup-и с RadioButton-и.

**3.5.5 ReferendumResultsFragment**

ReferendumResultFragment се зарежда след като гласуваме в даден референдум. За неговата визуализация отговаря fragment\_referendum\_results.xml, който съдържа LinearLayout с три TextView-та: едно за заглавието на референдума и два за двете възможни опции(“Да” и “Не”).

Референдум обекта се взима от аргументите

*final Referendum referendum = (Referendum) getArguments().getSerializable(Keys.VOTE\_OBJECT);*

Извиква се метода

*AppController.getTotalNumberOfVotes(options);*

На него се подава списък съдържащ двете опции. Метода събира timesSelected променливите на двата обекта и по този начин разбира колко общо човека са гласували на референдума. Този резултат се връща от метода и се присвоява на променливата totalNumberOfVotes в onCreate на ReferendumResultsFragment.

Пресмята се колко процента от гласували са избрали “Да” и колко опция “Не”

*String optionTextPercentageYes = new DecimalFormat("##.##").format(((referendum.getOptionYes().getTimesSelected() / (double) totalNumberOfVotes) \* 100)) + "%";*

Проверява се коя от двете опции е избрал потребителя и избраната се удебелява

*if(referendum.getOptionYes().isSelectedByCurrentUser())*

*optionYesTextView.setTypeface(null, Typeface.BOLD);*

**3.5.6 VotingResultsFragment**

Фрагмента отговаря за визуализирането на резултатите от гласуването, което е направил потребителя.

Заглавието на гласуването се изписва на екрана, а след него се завърта цикъл с всички опции в гласуването. За всяка опция се пресмятат колко пъти е избирана до сега в проценти. Избраната от потребителя опция се удебелява

**3.5.7 PollResultsFragment**

Фрагмента е идентичен на VotingResultsFragment. Той отговаря за показване на резултатите за гласуваната анкета. Различава се от фрагмента за резултати на гласувания с това, че се завърта цикъл за всеки въпрос от анкетата и след него за всички опции на въпроса. Идентично се смятат проценти и се удебеляват избраните от потребителя отговори.

**Четвърта глава**

**Потребителско ръководство**

**4.1 Начин на инсталация**

Приложението може да се инсталира като се вземе готовия .apk файл и се сложи на устроиството. Алтернативно може да инсталирате VotingApp чрез source code, който може да fork-нете от GitHub страницата

<https://github.com/georgi-ivanov-tues/VotingApp_TU_Project>

или в предоставения диск.

След като се сдобиете с папката на проекта отворете Android Studio и отидете на **File -> New -> Import Project** и укажете пътя до папката на проекта. След това използвайте командата **Build -> Make Projecte**, свържете устройството си и изпълнете команда **Run -> Run.**

**4.2 Нужни ресурси**

**Заключение**

Създадено е приложение за гласувания, анкети и референдуми на съвсем базово ниво. То може да се използва единствено за демонстративни цели и има нужда да се добави още много функционалност, дизайн и време преди приложението да е конкурентноспособно на вече съществуващи решения.

**Използвани ресурси**

Използвано IDE:

* + Android Studio: <https://developer.android.com/studio/>

Използвани езици:

* + Java: <https://java.com/en/>

Използвано хранилище:

* + GitHub: <https://github.com>

Примерни приложения:

* + MyVote: <https://www.myvote.io/>
  + VoxVote: <http://www.voxvote.com/>
  + Doodle: <https://doodle.com/>