Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

**Институт информационных технологий**

Специальность Программируемые мобильные системы

**КОНТРОЛЬНАЯ РАБОТА**

По курсу ПОМС

Проверил: Писарчик А.Ю.

Выполнил: Павловский Д.В.

Группа 983871

Минск, 2022

**СОДЕРЖАНИЕ**

В[ведение 5](#_Toc79691604)

[1 Архитектура мобильного приложения 4](#_Toc79691611)

[1.1 Объектная модель приложения 4](#_Toc79691612)

[1.2 Описание используемых в проекте шаблонов проектирования 5](#_Toc79691613)

[1.3 Описание используемых в проекте облачных хранилищ 6](#_Toc79691613)

[1.4 Система сборки Gradle 7](#_Toc79691613)

[2 Описание стуктуры кода приложения 15](#_Toc79691608)

[3 Описание графического пользовательского интерфейса 11](#_Toc79691609)

[3.1 Обоснование проекта пользовательского интерфейса 11](#_Toc79691610)

[Заключение 13](#_Toc79691614)

[Список используемых источников 14](#_Toc79691615)

# ВВЕДЕНИЕ

В настоящее время мобильные устройства занимают огромное место в нашей жизни. Люди уже не могут представить себя без мобильника в кармане, без планшета или какой-нибудь портативной приставки. Популярность и развитие мобильных устройств и технологий увеличивается с каждым годом. Развитие мобильных устройств изменяет окружающий нас мир цифровых технологий, теперь каждый человек имеет “персональный компьютер” в своем кармане. [1]

Мобильные приложения представляют собой компьютерные программы, специально созданные для мобильных телефонов. Более 99% рынка мобильных устройств, делят между собой устройства на операционной системе Android и iOS, причем устройства на Android занимают более 86%. Это связано в первую очередь с ценовой политикой и конкуренцией на рынке мобильных устройств, и с тем, что платформа Android представлена в свободном доступе. Смартфоны на операционной платформе Android более доступны и их разрабатывают множество производителей мобильных устройств, таких как Xiaomi, Samsung, Huawei и тд. В отличии от iOS, которые производит только Apple.

Основной целью контрольной работы является разработка графического интерфейса, написание кода и включение в проект облачного хранилища Firebase Realtime для мобильного приложения.

1. **ОПИСАНИЕ АРХТЕКТУРЫ ПРИЛОЖЕНИЯ**

Глава 1 отчета кратко описывает состав приложения.

1. Объектная модель приложения

При разработке приложения для ОС Android ииспользованы среда разработки Android Studio, язык программирования Java, а также Firebase Realtime Database.

Основной приложения является файл AndroidManifest.xml. Посредством данного файла проводятся интеграции сторонних Api и библиотек для разработки дизайна и подключения платформ для хранения данных в удаленных хранилищах. [2]

AndroidManifest.xml:

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <uses-permission android:name="android.permission.INTERNET"/>  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Newspaper"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:windowSoftInputMode="adjustResize"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 <meta-data  
 android:name="android.app.lib\_name"  
 android:value="" />  
 </activity>  
 </application>  
  
</manifest>

Для работы android-приложения используются следующие классы:

* DetailFragment;
* MainFragment;
* Item;
* ItemAdapter;
* Model;
* DetailPresenter;
* MainPresenter;
* MainActivity;

1. Описание используемого в проекте архитектурного шаблона

В проекте использован один из архитектурных шаблонов проектирования, называемый MVP.

Model-View-Presenter (MVP) — шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса.

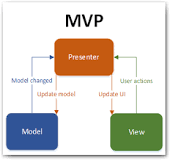


Рисунок 1 – Иллюстрация архитектурного шаблона MVP

Элемент Presenter в данном шаблоне берёт на себя функциональность посредника (аналогично контроллеру в MVC) и отвечает за управление событиями пользовательского интерфейса (например, использование мыши) так же, как в других шаблонах обычно отвечает представление. [3]

MVP — шаблон проектирования пользовательского интерфейса, который был разработан для облегчения автоматического модульного тестирования и улучшения разделения ответственности в презентационной логике (отделения логики от отображения):

1. Модель – данные для отображения;
2. Вид – реализует отображение данных (из Модели), обращается к Presenter за обновлениями, перенаправляет события от пользователя в Presenter;
3. Представитель – реализует взаимодействие между Моделью и Видом и содержит в себе всю логику представления данных о предметной области; при необходимости получает данные из хранилища и преобразует для отображения во View.

1.3 Описание используемых в проекте облачных хранилищ

Разработка надежных и высококачественных приложений для мобильных устройств, предполагает огромную самоотдачу, но что еще более важно, требует мощную и многофункциональную платформу для разработки.

Firebase, предоставляемая компанией Google, является одной из таких платформ, которая завоевала прочные позиции среди разработчиков по всему миру.

Firebase предоставляет разработчикам множество возможностей для создания высоко эффективных и универсальных веб-приложений, а также приложений для платформ Android и iOS.

В то время как существует множество конкурирующих между собой сред для разработки приложений, Firebase всегда использует самые лучшие из доступных на данный момент платформ. [4]

В проекте Firebase Realtime Database служит хранилищем для строковых элементов, содержащих в себе текст, наполняющий тестовые поля приложения. Архитектура, выбранная для хранения данных в Firebase Realtime Database приведена ниже.

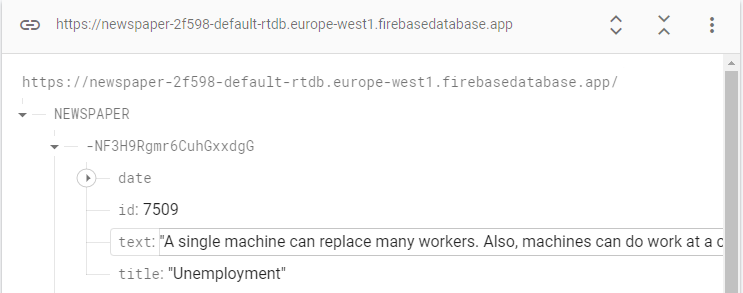


Рисунок 4.1 – Изображение архитектуры, используемой Firebase Realtime Database базы данных

На рисунке 4.1 изображена структура содержимого Firebase Realtime Database.

* 1. Система сборки Gradle

Gradle — [система автоматической сборки](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B8), построенная на принципах [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant) и [Apache Maven](https://ru.wikipedia.org/wiki/Apache_Maven), но предоставляющая [DSL](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) на языках [Groovy](https://ru.wikipedia.org/wiki/Groovy) и [Kotlin](https://ru.wikipedia.org/wiki/Kotlin) вместо традиционной [XML](https://ru.wikipedia.org/wiki/XML)-образной формы представления конфигурации проекта. Gradle использует [направленный ациклический граф](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) для определения порядка выполнения задач.

При разработке проекта используется Gradle v.7.2.2. В среде разработки android studio, Gradle подключается по умолчанию и данной версии достаточно для запуска android приложения. Далее приведен листинг build.gradle файла с текущего проекта:

**build.gradle (проект) :**

buildscript **{** dependencies **{** classpath 'com.google.gms:google-services:4.3.14'  
 **}  
}**// Top-level build file where you can add configuration options common to all sub-projects/modules.  
plugins **{** id 'com.android.application' version '7.2.1' apply false  
 id 'com.android.library' version '7.2.1' apply false  
**}**

**build.gradle (приложение):**

id 'com.android.application'  
 id 'com.google.gms.google-services'  
**}**android **{** namespace 'com.program.newspaper'  
 compileSdk 33  
  
 defaultConfig **{** applicationId "com.program.newspaper"  
 minSdk 21  
 targetSdk 33  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** release **{** minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 **}  
 }** compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_1\_8* targetCompatibility JavaVersion.*VERSION\_1\_8* **}  
}**dependencies **{** implementation 'com.arello-mobile:moxy:1.5.6'  
 annotationProcessor 'com.arello-mobile:moxy-compiler:1.5.6'  
 implementation ('tech.schoolhelper:moxy-x-androidx:1.7.0')**{** exclude group: 'tech.schoolhelper', module:'moxy-x'  
 **}** implementation("androidx.navigation:navigation-fragment:2.5.2")  
 implementation("androidx.navigation:navigation-ui:2.5.2")  
 implementation 'com.squareup.retrofit2:retrofit:2.5.0'  
  
 implementation 'com.google.firebase:firebase-database:20.0.6'  
  
 implementation 'io.reactivex.rxjava2:rxjava:2.2.21'  
 implementation 'com.github.FrangSierra:RxFirebase:1.5.6'  
  
  
 implementation 'androidx.recyclerview:recyclerview:1.2.1'  
  
 implementation 'androidx.appcompat:appcompat:1.5.1'  
 implementation 'com.google.android.material:material:1.6.1'  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
 testImplementation 'junit:junit:4.13.2'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
**}**

# ОПИСАНИЕ СТРУКТУРЫ КОДА приложения

Глава 2 отчета кратко описывает бизнес-логику приложения.

Приложение соответствует архитектурному шаблону MVP. Существует два «presenter», два «fragment» три модели «model» и сопутствующие работе приложения файлы в проекте.

В MainActivity устанавливается .xml-layout, который будет запускаться сразу при запуске приложения.

Поверх MainActivity.xml устанавливается «abs\_layout.xml», который имеет прямое отношение к шаблону MVP. Данный layout располагается согласно контенту поверх «activity\_main.xml».

Далее загружается «fragment\_mail.xml» согласно коду в файле «MainFragment.java». Класс «MainFragment» унаследовал логику «MvpCompatFragment» и реализует методы интерфейса «MainView». Следует описать кратко методы этого файла.

Метод «OnStart(){}» унаследован от библиотеки. Равносилен методу активити «OnCreate(..){}». Реализует методы библиотеки. Выводит данные на «fragment\_main.xml» согласно файлу класса «MainPresenter.java». Далее осуществляется объявление некоторых объектов, расположенных на фрагменте для дальнейшего их использования в коде. Также в коде файла определяется adapter для дальнейшей работы «recyclerview» списка с элементами «recyclerview\_item» и «recyclerview\_item2». Метод «onResume(){}» позволит повторно выгрузить данные при редактировании одного из элементов списка «recyclerview». Метод «startLoad(){}» используется при поиске для демонстрации протекания процесса поиска пользователю, а также на этапе загруски приложения и сопутствующих обращениях к методам файла «MainFragment.java». Метод «updateData(…){}» используется при поиске. Принимает в себя коллекцию найденных данных и выдает их на «fragment\_main.xml». Методы «showEmpty(){}» и «showError(){}» используются по назначению в процессе взаимодействия пользователя и приложения.

Идентичное назначение классов «DetailFragment» и «DetailPresenter». Эти классы описывают архитектурный шаблон MVP. Реализуют соответствующие интерфейсы и библиотеки.

Класс «Item» используется для работы с «presenter». Класс «ItemAdapter» используется с «recyclerview». Класс «Model» используется для работы с данными сервиса «Firebase Realtime».

# ОПИСАНИЕ графического пользовательского интерфейса

Графический интерфейс пользователя это не только решения о цветовой гамме, шрифтах и о конкретном воплощении элементов интерфейса (то есть, собственно о графическом дизайне), а также [изучение потребностей пользователей](https://usabilitylab.ru/services/research/); требования пользователей к системе; создание низко детализированных, а потом и высоко детализированных макетов экрана системы с учетом выявленных требований; [тестирование макеты](https://usabilitylab.ru/blog/mobile-mockup-testing/) на реальных или потенциальных пользователях системы. И лишь протестированные макеты можно отдавать в работу дизайнеру.

1. Обоснование проекта пользовательского интерфейса

В настоящие время разработчики графического интерфейса создают максимально простой и в то же время яркий дизайн, чтобы охватить как можно большее число пользователей и поэтому было решено делать наиболее простой и интуитивно понятный пользовательский интерфейс. Прототип проектируемого приложения представлен на рисунке 3.1.

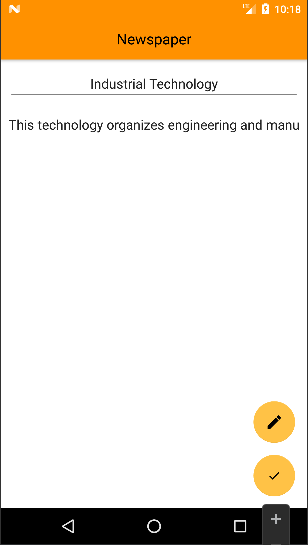
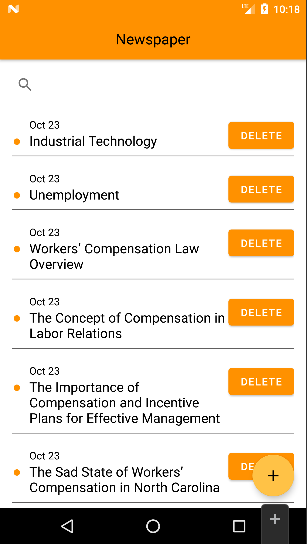


Рисунок 3.1 – Прототип проектируемого приложения

Разработав прототип приложения, осуществляется переход к разработке дизайна.

Ниже представлен стартовое окно приложения. Элементы можно добавить посредством нажатия на кнопку «+» и перехода на соответствуюзее представление, либо удалить, нажав кнопку «DELETE».

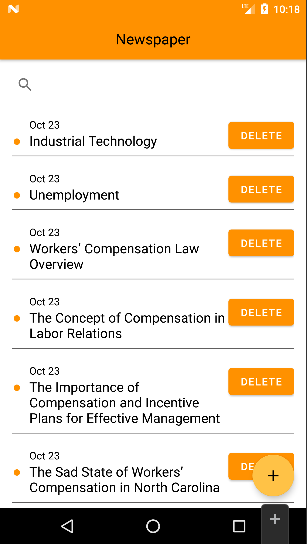


Рисунок 3.2 – Стартовое окно приложения

Ниже представлен вид представления с детальным видом элемента. Пользователь может заполнить соответствующие поля представления и создать элемент посредством нажатия на кнопку «галочки», либо изменить его содержимое посредством нажатия на кнопку с изображением «карандаш».

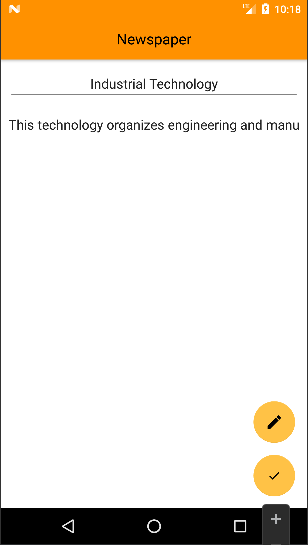


Рисунок 3.3 – Окно детального описания элемента

# ЗАКЛЮЧЕНИЕ

При выполнении контрольной работы стало понятно, что Android является современной операционной системой и в то же время довольно просто разрабатывать приложения под данную ОС используя такую среду разработки как Android Studio. Важным моментом при разработке приложения под Android является то, что Android Studio работает на всех настольных операционных системах, в то время как для разработки под iOS требуется операционная система macOS.

Поэтапно были решены все поставленные задачи, а именно: изучена история, версии, структура и архитектура платформы Android, разработан программный продукт, создана и интегрирована в проект удаленно развернутая Firebase Realtime Database база данных, спроектирован пользовательский интерфейс, изучена архитектура мобильного приложения.

Итогом выполнения работы стало достижение поставленной цели. Был разработан графический интерфейс и написан код для мобильного программного средства.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] СТП 01-2017 [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_122976.pdf

[2] StudBooks.net [Электронный ресурс]. — Режим доступа: <https://studbooks.net/2145389/informatika/platforma_android>

[3] Wikipedia [Электронный ресурс]. — Режим доступа: <https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9_Android>

[4] Wikipedia [Электронный ресурс]. — Режим доступа: <https://ru.wikipedia.org/wiki/Android_SDK#cite_note-2>