МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет науки и технологий

имени академика М.Ф. Решетнева»

Направление подготовки

09.04.01 Информатика и вычислительная техника

**КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ**

**«РАЗРАБОТКА ПРИЛОЖЕНИЙ БАЗ ДАННЫХ»**

**ТЕМА: РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ПО ПРОГРАММИРОВАНИЮ «TESTPRO»**

Выполнил:

студент группы МИВ19-01

Е.В. Горшков

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Принял:

доцент, кандидат физ.-мат. наук

Н.В. Киреев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Красноярск 2019

**Оглавление**

[**ВВЕДЕНИЕ** 3](#_Toc28082400)

[**1.** **РАЗРАБОТКА ИНТЕРФЕЙСА ПРОГРАММНОГО ПРОДУКТА** 5](#_Toc28082401)

[**2.** **РАЗРАБОТКА БАЗЫ ДАННЫХ FIREBASE** 17](#_Toc28082402)

[**3.** **РАЗРАБОТКА КЛАССОВ ПРОГРАММНОГО ПРОДУКТА** 18](#_Toc28082403)

[**3.1.** **Main Activity** 18](#_Toc28082404)

[**3.2.** **Categories Activity** 19](#_Toc28082405)

[**3.3.** **Category Adapter** 20](#_Toc28082406)

[**3.4.** **Category Model** 22](#_Toc28082407)

[**3.5.** **Sets Activity** 23](#_Toc28082408)

[**3.6.** **Questions Activity** 24](#_Toc28082409)

[**3.7.** **Questions Model** 29](#_Toc28082410)

[**3.8.** **Score Activity** 31](#_Toc28082411)

[**3.9.** **Bookmark Activity** 31](#_Toc28082412)

[**3.10.** **Bookmarks Adapter** 33](#_Toc28082413)

[**ЗАКЛЮЧЕНИЕ** 35](#_Toc28082414)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 36](#_Toc28082415)

# **ВВЕДЕНИЕ**

История тестов учебных достижений насчитывает, по мнению В.Кадневского, по крайней мере, несколько тысячелетий. В.Аванесов указывает на факты, свидетельствующие о еще более древнем применении тестов.

Древние вавилоняне знали 400 клинописных знаков, использовали шестидесятеричную систему счета, умели вычислять проценты, измерять площадь и объем различных геометрических фигур. Среди изучаемых предметов были те, которые отвечают современному понятию «профессиональная пригодность». За 2200 лет до н.э. в Китае успешно действовала система проверки способностей и отбора персонала для различных государственных должностей. В частности, проверялось умение писать, читать, знать порядок проведения придворных ритуалов и церемоний. В течение последующих 2000 лет в систему отбора чиновников были внесены экзамены по гражданскому праву, военному делу, финансам, сельскому хозяйству, географии.

Одним из первых ученых, попытавшихся измерить различия между людьми в области элементарных психических процессов, был англичанин Френсис Гальтон (Galton F.; 1882-1911).

Большой вклад в развитие тестов интеллекта внес французский психолог Альфред Бине (Binet A., 1857-1911). Совместно с Теодором Симоном (Simon T., 1873-1961) он разработал тест, позволяющий дифференцировать нормальных и умственно отсталых детей.

Для начала давайте обозначим объект нашего исследования. Что же такое тест? Сегодня этот термин встречается довольно часто и в областях далеких от образования.

Тестирование (от слова test - испытание, проверка). Тестирование применяется для определения соответствия предмета испытания заданным спецификациям. В задачи тестирования не входит определение причин несоответствия заданным требованиям. Тестирование - один из разделов диагностики. Качество тестирования и достоверность его результатов в значительной степени зависит от тестера.

Тестирование сейчас применяется в разных областях деятельности: медицине, психологии, программировании. Но чаще всего слово «тест» ассоциируется с методом проверки знаний учащихся.

**Преимущества:**

* Тестирование является более качественным и объективным способом оценивания, его объективность достигается путем стандартизации процедуры проведения, проверки показателей качества заданий и тестов целиком.
* Тестирование - более справедливый метод, оно ставит всех учащихся в равные условия, как в процессе контроля, так и в процессе оценки, практически исключая субъективизм преподавателя
* Тесты - это более объёмный инструмент, поскольку тестирование может включать в себя задания по всем темам курса, в то время как на устный экзамен обычно выносится 2-4 темы, а на письменный - 3-5.
* Тест - это более точный инструмент, так, например, шкала оценивания теста из 20 вопросов, состоит из 20 делений, в то время, как обычная шкала оценки знаний - только из четырёх.
* Тестирование - это более мягкий инструмент, они ставят всех учащихся в равные условия, используя единую процедуру и единые критерии оценки, что приводит к снижению предэкзаменационных нервных напряжений.

**Недостатки:**

* Разработка качественного тестового инструментария - длительный, трудоемкий и дорогостоящий процесс. Стандартные наборы тестов для большинства дисциплин ещё не разработаны, а разработанные обычно имеют очень низкое качество.
* Данные, получаемые преподавателем в результате тестирования, хотя и включают в себя информацию о пробелах в знаниях по конкретным разделам, но не позволяют судить о причинах этих пробелов.
* Тест не позволяет проверять и оценивать высокие, продуктивные уровни знаний, связанные с творчеством, то есть вероятностные, абстрактные и методологические знания.
* В тестировании присутствует элемент случайности. Например, учащийся, не ответивший на простой вопрос, может дать правильный ответ на более сложный. Причиной этого может быть, как случайная ошибка в первом вопросе, так и угадывание ответа во втором. Это искажает результаты теста и приводит к необходимости учета вероятностной составляющей при их анализе.

В совокупности вышесказанного тестирование является более применимым для большого количества студентов и менее затратно с точки зрения проведения проверки знаний. Именно поэтому в данной курсовой работе была выполнена разработка программного продукта в среде Android Studio на языке Java для тестирования студентов технических специальностей по программированию «TestPRO», задачей которой является оценка знаний студентов по дисциплине «Программирование».

# **РАЗРАБОТКА ИНТЕРФЕЙСА ПРОГРАММНОГО ПРОДУКТА**

Для начала работы программы необходимо запустить приложение на android устройстве «TestPRO». Главная форма приложения выглядит следующим образом:

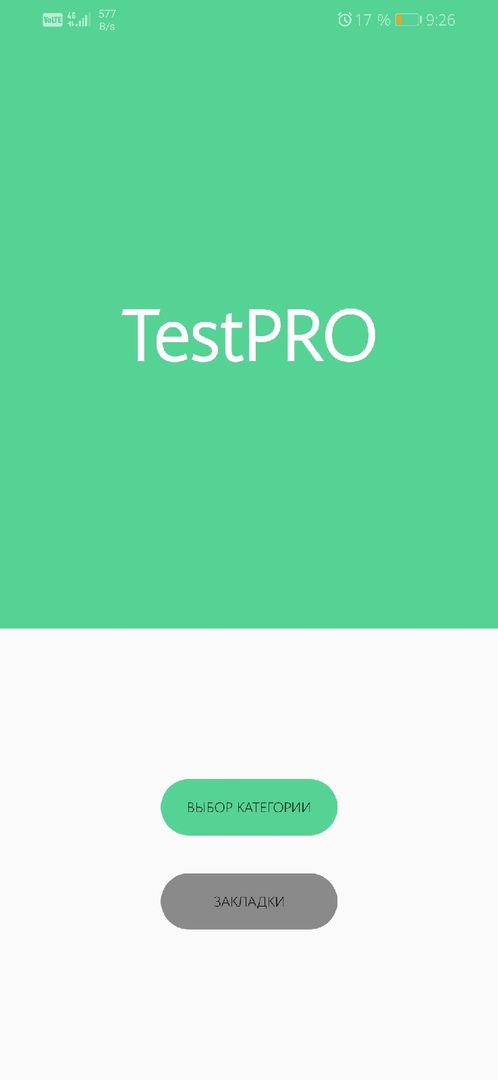


Рисунок 1 – Главная форма приложения.

Ниже представлен код оформления главной формы приложения в формате .xml.

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <FrameLayout  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginBottom="128dp"  
 android:background="@color/colorPrimary"  
 app:layout\_constraintBottom\_toTopOf="@+id/start\_btn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="@string/app\_name"  
 android:textColor="@android:color/white"  
 android:textSize="60dp"  
 android:textStyle="bold" />  
 </FrameLayout>  
  
 <Button  
 android:id="@+id/start\_btn"  
 android:layout\_width="150dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="32dp"  
 android:background="@drawable/rounded\_corners"  
 android:stateListAnimator="@null"  
 android:text="Выбор категории"  
 app:layout\_constraintBottom\_toTopOf="@+id/bookmarks\_btn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/bookmarks\_btn"  
 android:layout\_width="150dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="128dp"  
 android:text="Закладки"  
 android:stateListAnimator="@null"  
 android:backgroundTint="#8A8A8A"  
 android:background="@drawable/rounded\_corners"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
</androidx.constraintlayout.widget.ConstraintLayout>

На данной форме реализованы FrameLayout. Он является самым простым типом разметки. Обычно это пустое пространство на экране, которое можно заполнить только дочерними объектами View или ViewGroup. Все дочерние элементы FrameLayout прикрепляются к верхнему левому углу экрана. В FrameLayout вставлен текст типа TextView, который отцентрирован по центру. Backround rounded\_corners был реализован с целью отображения овалов вместо обычных форм. Код xml:

<?xml version="1.0" encoding="utf-8"?>  
<shape android:shape="rectangle" xmlns:android="http://schemas.android.com/apk/res/android">  
 <corners android:radius="300dp"/>  
 <solid android:color="@color/colorPrimary" />  
</shape>

На главной форме приложения есть две кнопки: «ВЫБОР КАТЕГОРИИ» и «ЗАКЛАДКИ». По нажатию кнопки «ВЫБОР КАТЕГОРИИ» для пользователя представится возможность выбрать категорию тестирования. Результат нажатия на кнопку «ВЫБОР КАТЕГОРИИ» выглядит следующим образом:

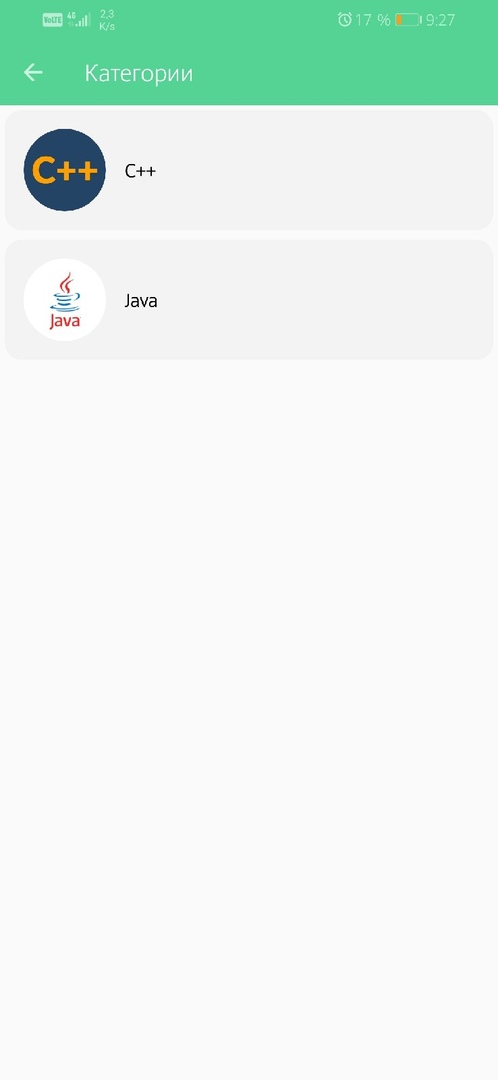


Рисунок 2 – Окно выбора категории тестирования.

Ниже приведен код для данной формы.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".CategoriesActivity">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="@color/colorPrimary" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
</LinearLayout>

Для отображения списка категории был реализован список RecyclerView c параметрами по высоте и длине "match\_parent"(известный как "fill\_parent"), который служит для растягивания компонента по размеру родительского представления, а также список RecylerView для отображения списка категорий.

Если интернет-соединение нестабильно или отсутствует, предусмотрено отображение значка загрузки с сервера. Выглядит это следующим образом:

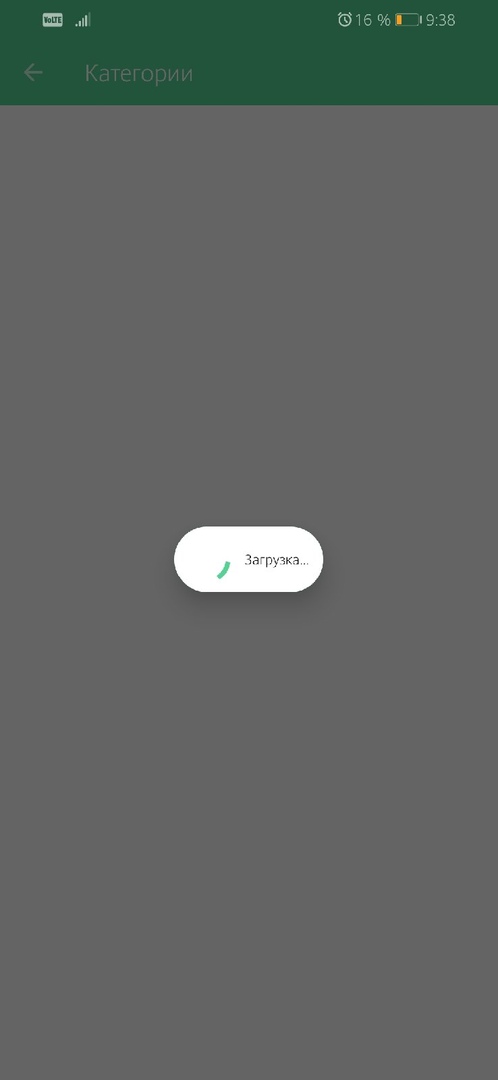


Рисунок 3 – Реализованный loading при обращении к серверу.

Код xml представлен ниже.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal" android:layout\_width="wrap\_content"  
 android:background="@drawable/rounded\_corners"  
 android:backgroundTint="@android:color/white"  
 android:padding="4dp"  
 android:layout\_height="wrap\_content">  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 style="?android:attr/progressBarStyle"  
 android:layout\_width="wrap\_content"  
 android:indeterminateTint="@color/colorPrimary"  
 android:layout\_height="wrap\_content" />  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Загрузка..."  
 android:padding="8dp"  
 android:textColor="@android:color/black"  
 android:layout\_gravity="center"  
 />  
</LinearLayout>

При получении ответа от сервера при нажатии интересующей нас категории тестирования программа выдает список тестов. Результат выполнения данной операции представлено на рисунке 4.



Рисунок 4 – Окно выбора теста.

Код в формате .xml представлен ниже.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".SetsActivity">  
   
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:background="@color/colorPrimary"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark"  
 android:layout\_height="?attr/actionBarSize"/>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Тесты"  
 android:textSize="28sp"  
 android:textStyle="bold"  
 android:textColor="@android:color/black"  
 android:padding="16dp"/>  
  
 <GridView  
 android:id="@+id/gridView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:columnWidth="120dp"  
 android:horizontalSpacing="16dp"  
 android:numColumns="auto\_fit"  
 android:gravity="center"/>  
  
</LinearLayout>

В студии макет LinearLayout представлен двумя вариантами - Horizontal и Vertical. Макет LinearLayout выравнивает все дочерние объекты в одном направлении — вертикально или горизонтально. Направление задается при помощи атрибута ориентации android:orientation:

* android:orientation="horizontal"
* android:orientation="vertical"

Все дочерние элементы помещаются в стек один за другим, так что вертикальный список компонентов будет иметь только один дочерний элемент в ряду независимо от того, насколько широким он является. Горизонтальное расположение списка будет размещать элементы в одну строку с высотой, равной высоте самого высокого дочернего элемента списка.

При выборе интересующего пользователя теста приложение выдаст форму отображения вопроса. Результат выполнения представлен на рисунке 5.



Рисунок 5 – Реализация формы вопросов.

Xml-код представлен ниже.

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".QuestionsActivity">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="0dp"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="@color/colorPrimary"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="@string/app\_name"  
 android:textColor="@android:color/white"  
 android:textSize="16sp"  
 android:textStyle="bold" />  
  
 </androidx.appcompat.widget.Toolbar>  
  
 <LinearLayout  
 android:id="@+id/linearLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/colorPrimary"  
 android:orientation="vertical"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar">  
  
 <TextView  
 android:id="@+id/questions"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp"  
 android:background="@drawable/category\_bg"  
 android:padding="16dp"  
 android:text=""  
 android:textColor="@android:color/black" />  
  
 <TextView  
 android:id="@+id/no\_indicator"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:padding="16dp"  
 android:text="5/10"  
 android:textColor="@android:color/white"  
 android:textSize="16sp"  
 android:textStyle="bold" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/options\_container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:orientation="vertical"  
 android:padding="16dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/linearLayout">  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 android:background="@drawable/rounded\_dorders"  
 android:backgroundTint="#989898"  
 android:textColor="@android:color/black"  
 android:text="" />  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 android:background="@drawable/rounded\_dorders"  
 android:backgroundTint="#989898"  
 android:textColor="@android:color/black"  
 android:text="" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 android:background="@drawable/rounded\_dorders"  
 android:textColor="@android:color/black"  
 android:backgroundTint="#989898"  
 android:text="" />  
  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 android:background="@drawable/rounded\_dorders"  
 android:textColor="@android:color/black"  
 android:backgroundTint="#989898"  
 android:text="" />  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/share\_btn"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="16dp"  
 android:background="@drawable/rounded\_corners"  
 android:backgroundTint="#F3373C"  
 android:text="Выход"  
 android:textColor="@android:color/white"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/next\_btn"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/options\_container" />  
  
 <Button  
 android:id="@+id/next\_btn"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Далее"  
 android:enabled="false"  
 android:alpha="0.7"  
 android:background="@drawable/rounded\_corners"  
 android:backgroundTint="#848484"  
 android:textColor="@android:color/white"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toEndOf="@+id/share\_btn"  
 app:layout\_constraintTop\_toBottomOf="@+id/options\_container" />  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/bookmark\_btn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="16dp"  
 android:clickable="true"  
 android:translationY="24dp"  
 app:layout\_constraintBottom\_toBottomOf="@+id/linearLayout"  
 app:layout\_constraintEnd\_toEndOf="@+id/linearLayout"  
 app:srcCompat="@drawable/bookmarks\_border" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

При прохождении теста пользователю представляется окно результатов ответов на тест. Реализация окна вывода на экран результатов теста представлен на картинке 6.

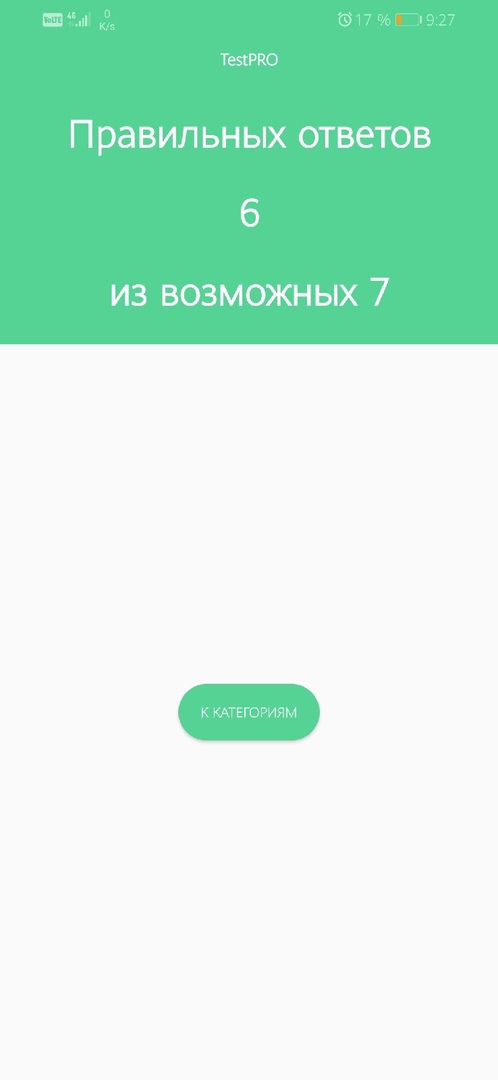


Рисунок 6 – Реализация окна вывода на экран результата прохождения теста

Код xml выполнен следующим образом:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".ScoreActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:background="@color/colorPrimary"  
 tools:layout\_editor\_absoluteX="168dp"  
 tools:layout\_editor\_absoluteY="246dp">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="8dp"  
 android:textSize="16sp"  
 android:textColor="@android:color/white"  
 android:textStyle="bold"  
 android:textAlignment="center"  
 android:text="@string/app\_name" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="24dp"  
 android:textSize="38sp"  
 android:textStyle="bold"  
 android:textColor="@android:color/white"  
 android:textAlignment="center"  
 android:text="Правильных ответов" />  
  
 <TextView  
 android:id="@+id/scored"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="38sp"  
 android:textStyle="bold"  
 android:textColor="@android:color/white"  
 android:textAlignment="center"  
 android:text="5" />  
  
 <TextView  
 android:id="@+id/total"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="24dp"  
 android:textSize="38sp"  
 android:textStyle="bold"  
 android:textColor="@android:color/white"  
 android:textAlignment="center"  
 android:text="из возможных 10" />  
 </LinearLayout>  
  
 <FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1">  
 <Button  
 android:id="@+id/done\_btn"  
 android:layout\_width="120dp"  
 android:layout\_height="wrap\_content"  
 android:text="К категориям"  
 android:textColor="@android:color/white"  
 android:background="@drawable/rounded\_corners"  
 android:layout\_gravity="center" />  
 </FrameLayout>  
</LinearLayout>

При возникновении трудности с ответом пользователь может законспектировать интересующий его вопрос в закладках. Для этого достаточно нажать на кнопку выше вариантов ответа, и вопрос окажется в закладках, кнопка для перехода которой была реализована на главном экране приложения. Результат нажатия на кнопку «ЗАКЛАДКИ» с xml-кодом представлены ниже.

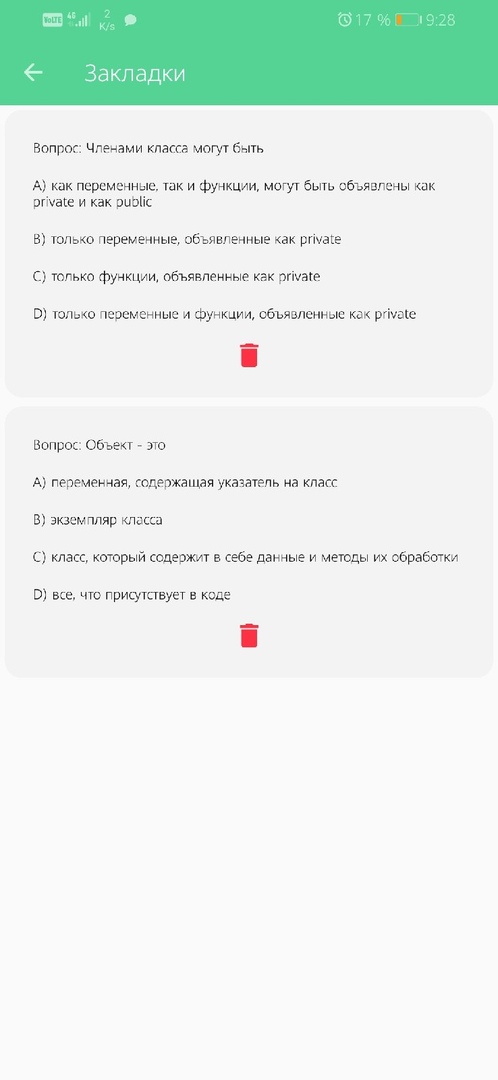


Рисунок 7 – Результат нажатия на кнопку «ЗАКЛАДКИ»

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".BookmarkActivity">  
   
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:theme="@style/ThemeOverlay.AppCompat.Dark"  
 android:background="#55D394"/>  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_bookmarks"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"/>  
  
</LinearLayout>

# **РАЗРАБОТКА БАЗЫ ДАННЫХ FIREBASE**

Для хранения данных теста была реализована база данных реального времени Firebase. Результат использования БД приведен на рисунке 8.

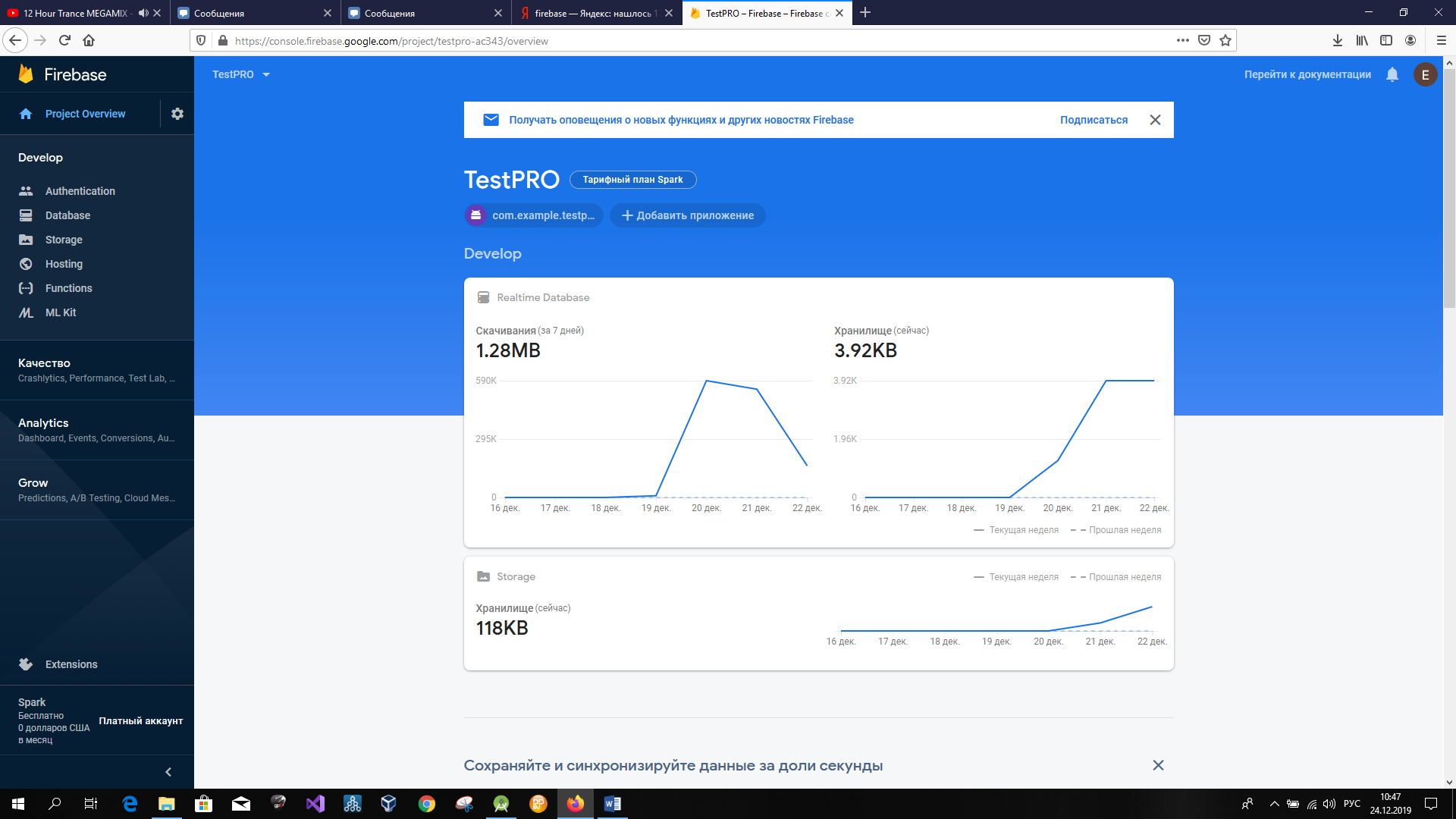


Рисунок 8 – Использование БД

Разработанная БД имеет следующую архитектуру хранения к обращаемым ячейкам для категорий:

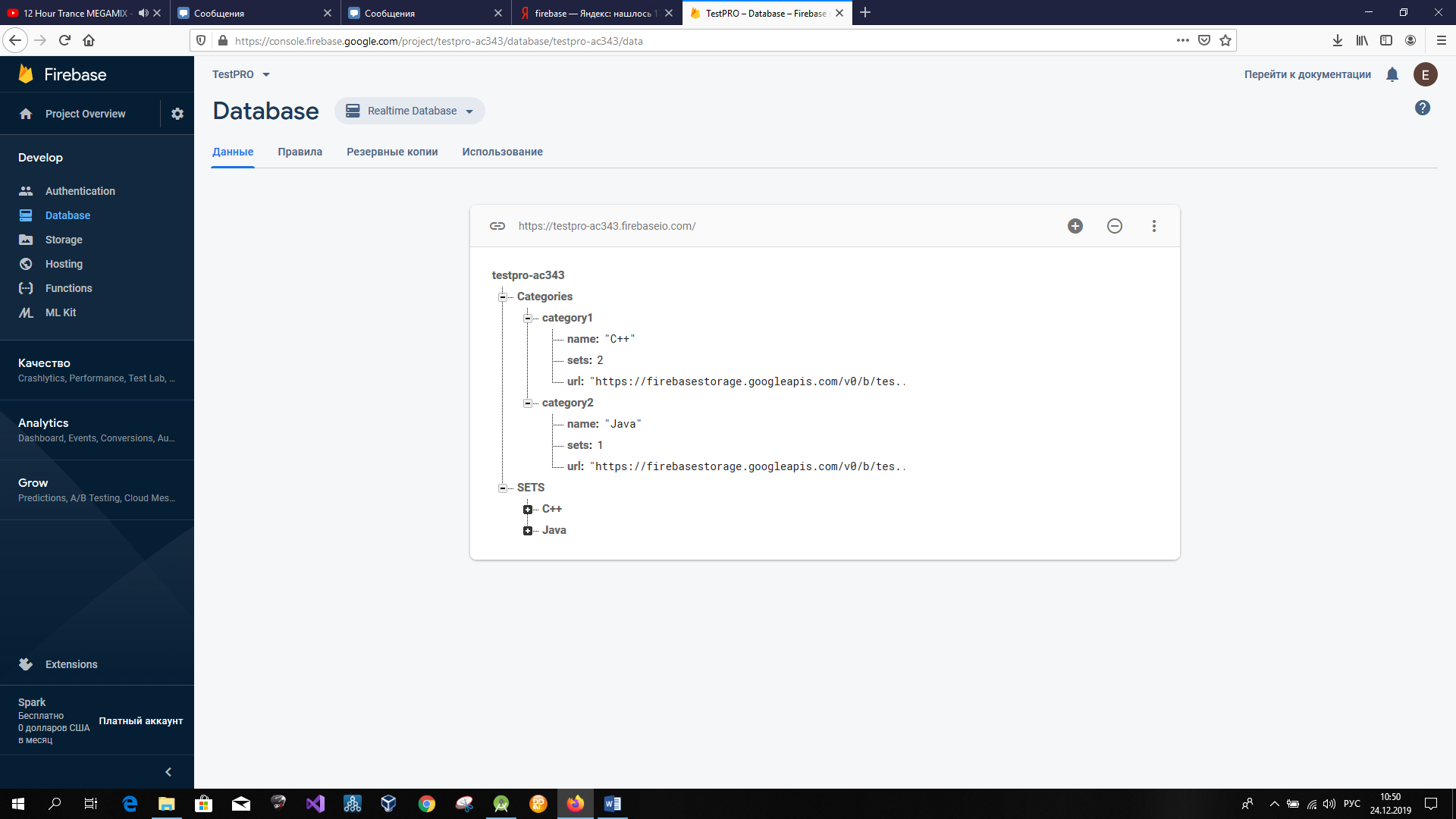


Рисунок 9 – Архитектура хранения данных для категорий

и для хранения вопросов теста:

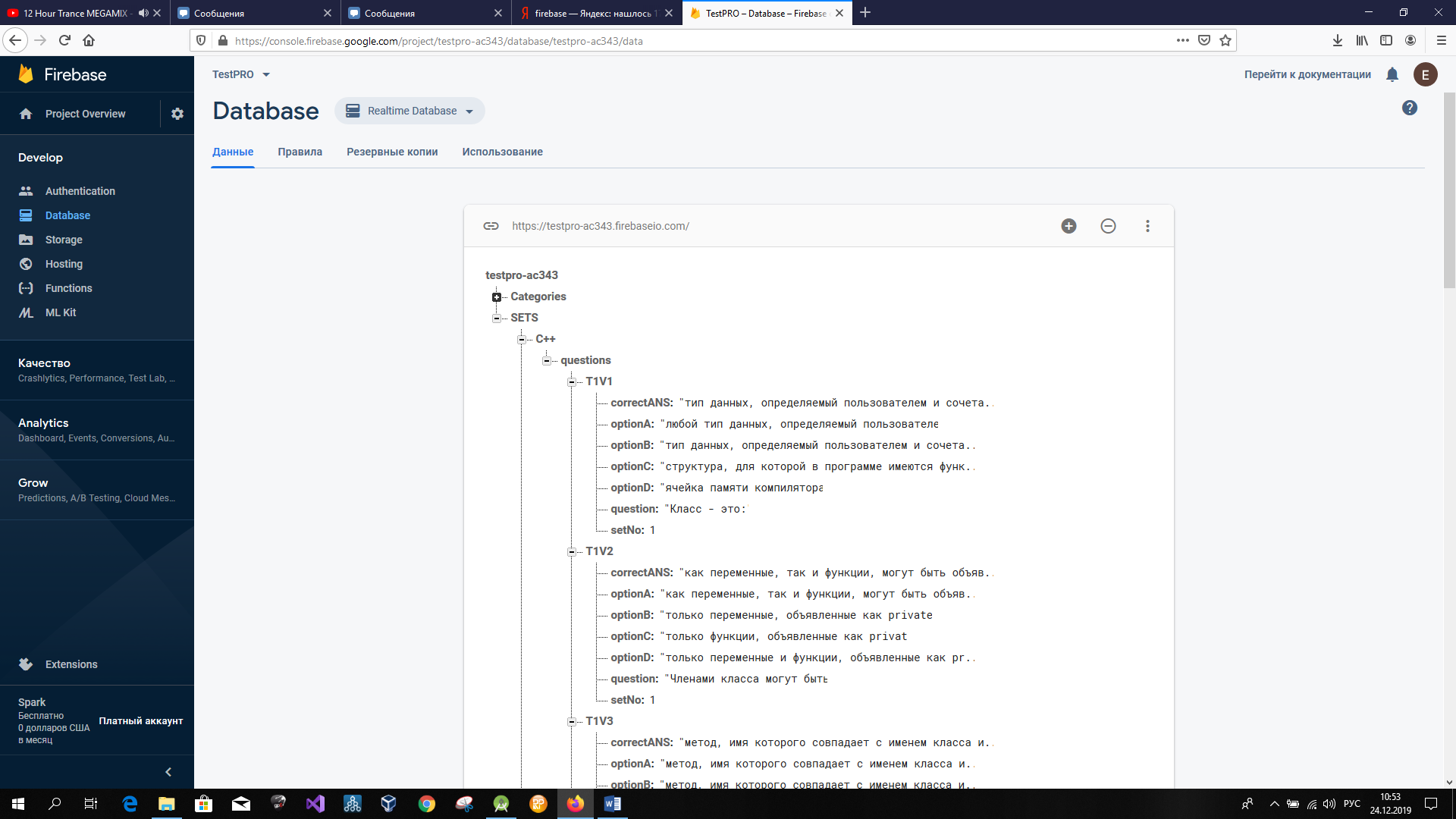


Рисунок 10 - Архитектура хранения данных для вопросов теста

# **РАЗРАБОТКА КЛАССОВ ПРОГРАММНОГО ПРОДУКТА**

* 1. **Main Activity**

package com.example.testpro;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
  
public class MainActivity extends AppCompatActivity {  
  
 private Button startBtn,bookmarkBtn;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 startBtn = findViewById(R.id.*start\_btn*);  
 bookmarkBtn = findViewById(R.id.*bookmarks\_btn*);  
  
 startBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent categoyIntent = new Intent(MainActivity.this,CategoriesActivity.class);  
 startActivity(categoyIntent);  
 }  
 });  
  
 bookmarkBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent bookmarksIntent = new Intent(MainActivity.this,BookmarkActivity.class);  
 startActivity(bookmarksIntent);  
 }  
 });  
 }  
}

На данной форме реализованы кнопки start\_btn и bookmark\_btn для выбора категории и отображения закладок соответственно. Они связывают id кнопки из кода xml и передают информацию для их использования. Реализованы методы setOnClickListener для нажатия кнопки и вывода соответствующих методов и обращение к дочерним классам.

* 1. **Categories Activity**

package com.example.testpro;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.app.Dialog;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.widget.LinearLayout;  
import android.widget.Toast;  
  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class CategoriesActivity extends AppCompatActivity {  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference myRef = database.getReference();  
  
 private Dialog loadingDialog;  
  
 private RecyclerView recyclerView;  
 private List<CategoryModel> list;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_categories*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
  
 setSupportActionBar(toolbar);  
 getSupportActionBar().setTitle("Категории");  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
  
 loadingDialog = new Dialog(this);  
 loadingDialog.setContentView(R.layout.*loading*);  
 loadingDialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.*rounded\_corners*));  
 loadingDialog.getWindow().setLayout(LinearLayout.LayoutParams.*WRAP\_CONTENT*,LinearLayout.LayoutParams.*WRAP\_CONTENT*);  
 loadingDialog.setCancelable(false);  
  
 recyclerView = findViewById(R.id.*rv*);  
  
 LinearLayoutManager layoutManager = new LinearLayoutManager(this);  
 layoutManager.setOrientation(RecyclerView.*VERTICAL*);  
  
 recyclerView.setLayoutManager(layoutManager);  
  
 list = new ArrayList<>();  
 final CategoryAdapter adapter = new CategoryAdapter(list);  
 recyclerView.setAdapter(adapter);  
  
 loadingDialog.show();  
  
 myRef.child("Categories").addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()){  
 list.add(dataSnapshot1.getValue(CategoryModel.class));  
  
 }  
 adapter.notifyDataSetChanged();  
 loadingDialog.dismiss();  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
 Toast.*makeText*(CategoriesActivity.this, databaseError.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 loadingDialog.dismiss();  
 finish();  
 }  
 });  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
  
 if (item.getItemId() == android.R.id.*home*){  
 finish();  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
}

В данном классе реализовано подключение к базе данных Firebase реального времени, а также обозначены основные методы для отображения списка RecycleView и List, загрузка диалогового окна «Загрузка».

* 1. **Category Adapter**

package com.example.testpro;  
  
import android.content.Intent;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
  
import java.util.List;  
  
import de.hdodenhof.circleimageview.CircleImageView;  
  
public class CategoryAdapter extends RecyclerView.Adapter<CategoryAdapter.Viewholder> {  
 private List<CategoryModel> categoryModelList;  
  
 public CategoryAdapter(List<CategoryModel> categoryModelList) {  
 this.categoryModelList = categoryModelList;  
 }  
  
 @NonNull  
 @Override  
 public Viewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*category\_item*,parent,false);  
 return new Viewholder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull Viewholder holder, int position) {  
 holder.setData(categoryModelList.get(position).getUrl(),categoryModelList.get(position).getName(),categoryModelList.get(position).getSets());  
 }  
  
 @Override  
 public int getItemCount() {  
 return categoryModelList.size();  
 }  
  
 class Viewholder extends RecyclerView.ViewHolder{  
  
 private CircleImageView imageView;  
 private TextView title;  
  
 public Viewholder(@NonNull View itemView) {  
 super(itemView);  
  
 imageView = itemView.findViewById(R.id.*image\_view*);  
 title = itemView.findViewById(R.id.*title*);  
 }  
 private void setData(String url, final String title, final int sets){  
  
 Glide.*with*(itemView.getContext()).load(url).into(imageView);  
 this.title.setText(title);  
  
 itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent setIntent = new Intent(itemView.getContext(),SetsActivity.class);  
 setIntent.putExtra("title",title);  
 setIntent.putExtra("sets",sets);  
 itemView.getContext().startActivity(setIntent);  
 }  
 });  
  
 }  
  
 }  
}

В данном модуле было реализовано подключение к библиотеке Glid, которая является ближайшим конкурентом другой популярной библиотеке [Picasso](http://developer.alexanderklimov.ru/android/library/picasso.php) и также предназначена для асинхронной подгрузки изображений, ресурсов или файловой системы, их кэширования и отображения. Также было реализовано использование класса ViewHolder для улучшения производительности работы больших списков. Для использования круглых значков было реализовано CircleImageView библиотеки

implementation 'de.hdodenhof:circleimageview:3.0.1'

, а также использование подгруженных из БД url, title и sets.

* 1. **Category Model**

package com.example.testpro;  
  
public class CategoryModel {  
 private String name;  
 private int sets;  
 private String url;  
  
 public CategoryModel(){  
 //for firebase  
 }  
  
 public CategoryModel(String name, int sets, String url) {  
 this.name = name;  
 this.sets = sets;  
 this.url = url;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getSets() {  
 return sets;  
 }  
  
 public void setSets(int sets) {  
 this.sets = sets;  
 }  
  
 public String getUrl() {  
 return url;  
 }  
  
 public void setUrl(String url) {  
 this.url = url;  
 }  
}

Данный модуль является реализацией структуры хранения данных из Firebase для имени категории, категории и адреса обращения к ней. Был использован constructor для создания getters and setters.

* 1. **Sets Activity**

package com.example.testpro;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.widget.Adapter;  
import android.widget.GridView;  
  
public class SetsActivity extends AppCompatActivity {  
  
 private GridView gridView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_sets*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setTitle(getIntent().getStringExtra("title")) ;  
  
 gridView = findViewById(R.id.*gridView*);  
  
 GridAdapter adapter = new GridAdapter(getIntent().getIntExtra("sets",0),getIntent().getStringExtra("title"));  
 gridView.setAdapter(adapter);  
  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if (item.getItemId() == android.R.id.*home*){  
 finish();  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

В данном модуле реализовано использование GridView для отображения тестов.

* 1. **Questions Activity**

package com.example.testpro;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
import android.animation.Animator;  
import android.app.Dialog;  
import android.content.Context;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.res.ColorStateList;  
import android.graphics.Color;  
import android.os.Bundle;  
  
import android.view.View;  
import android.view.animation.DecelerateInterpolator;  
import android.widget.Button;  
import android.widget.LinearLayout;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.google.gson.Gson;  
import com.google.gson.reflect.TypeToken;  
  
import java.lang.reflect.Type;  
import java.util.ArrayList;  
import java.util.List;  
  
public class QuestionsActivity extends AppCompatActivity {  
 public static final String *FILE\_NAME* = "TestPRO";  
 public static final String *KEY\_NAME* = "QUESTIONS";  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference myRef = database.getReference();  
 private TextView question,noIndicator;  
 private FloatingActionButton bookmarkBtn;  
 private LinearLayout optionsContainer;  
 private Button shareBtn,nextBtn;  
 private int count = 0;  
 private List<QuestionModel> list;  
 private int position = 0;  
 private int score = 0;  
 private String category;  
 private int setNo;  
 private Dialog loadingDialog;  
 private Button cancel;  
 private Button button,button2,button3,button4;  
  
 private List<QuestionModel> bookmarksList;  
  
 private SharedPreferences preferences;  
 private SharedPreferences.Editor editor;  
 private Gson gson;  
 private int matchedQuestionPosition;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_questions*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
 question = findViewById(R.id.*questions*);  
 noIndicator = findViewById(R.id.*no\_indicator*);  
 bookmarkBtn = findViewById(R.id.*bookmark\_btn*);  
 optionsContainer = findViewById(R.id.*options\_container*);  
 shareBtn = findViewById(R.id.*share\_btn*);  
 nextBtn = findViewById(R.id.*next\_btn*);  
 cancel = findViewById(R.id.*share\_btn*);  
  
 preferences = getSharedPreferences(*FILE\_NAME*, Context.*MODE\_PRIVATE*);  
 editor = preferences.edit();  
 gson = new Gson();  
  
  
 button = findViewById(R.id.*button*);  
 button2 = findViewById(R.id.*button2*);  
 button3 = findViewById(R.id.*button3*);  
 button4 = findViewById(R.id.*button4*);  
  
  
  
  
 getBookmarks();  
 bookmarkBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (modelMatch()){  
 bookmarksList.remove(matchedQuestionPosition);  
 bookmarkBtn.setImageDrawable(getDrawable(R.drawable.*bookmarks\_border*));  
 }else{  
 bookmarksList.add(list.get(position));  
 bookmarkBtn.setImageDrawable(getDrawable(R.drawable.*bookmark*));  
 }  
 }  
 });  
  
 category = getIntent().getStringExtra("category");  
 setNo = getIntent().getIntExtra("setNo", 1);  
  
 loadingDialog = new Dialog(this);  
 loadingDialog.setContentView(R.layout.*loading*);  
 loadingDialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.*rounded\_corners*));  
 loadingDialog.getWindow().setLayout(LinearLayout.LayoutParams.*WRAP\_CONTENT*, LinearLayout.LayoutParams.*WRAP\_CONTENT*);  
 loadingDialog.setCancelable(false);  
  
 list = new ArrayList<>();  
  
 loadingDialog.show();  
  
 myRef.child("SETS").child(category).child("questions").orderByChild("setNo").equalTo(setNo).addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 for (DataSnapshot snapshot : dataSnapshot.getChildren()) {  
 list.add(snapshot.getValue(QuestionModel.class));  
 }  
  
 if (list.size() > 0) {  
  
  
 for (int i = 0; i < 4; i++) {  
 optionsContainer.getChildAt(i).setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 checkAnswer((Button) v);  
 }  
 });  
 }  
 playAnim(question, 0, list.get(position).getQuestion());  
 nextBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 nextBtn.setEnabled(false);  
 nextBtn.setAlpha(0.7f);  
 enableOption(true);  
 position++;  
 if (position == list.size()) {  
 Intent scoreIntent = new Intent(QuestionsActivity.this, ScoreActivity.class);  
 scoreIntent.putExtra("score", score);  
 scoreIntent.putExtra("total", list.size());  
 startActivity(scoreIntent);  
 finish();  
 return;  
 }  
 count = 0;  
 playAnim(question, 0, list.get(position).getQuestion());  
 }  
 });  
 } else {  
 finish();  
 Toast.*makeText*(QuestionsActivity.this, "no questions", Toast.*LENGTH\_SHORT*).show();  
 }  
 loadingDialog.dismiss();  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 Toast.*makeText*(QuestionsActivity.this, databaseError.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 loadingDialog.dismiss();  
 finish();  
 }  
 });  
 cancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
 });  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 storeBookmarks();  
 }  
  
 private void playAnim(final View view, final int value, final String data){  
  
 view.animate().alpha(value).scaleX(value).scaleY(value).setDuration(500).setStartDelay(100)  
 .setInterpolator(new DecelerateInterpolator()).setListener(new Animator.AnimatorListener() {  
 @Override  
 public void onAnimationStart(Animator animation) {  
 button.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#989898")));  
 button2.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#989898")));  
 button3.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#989898")));  
 button4.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#989898")));  
 if(value == 0 && count < 4){  
 String option = "";  
 if (count == 0){  
 option = list.get(position).getOptionA();  
 }else if (count == 1){  
 option = list.get(position).getOptionB();  
 }else if (count == 2){  
 option = list.get(position).getOptionC();  
 }else if (count == 3){  
 option = list.get(position).getOptionD();  
 }  
 playAnim(optionsContainer.getChildAt(count), 0,option);  
 count++;  
 }  
 }  
  
 @Override  
 public void onAnimationEnd(Animator animation) {  
 if (value == 0){  
 try {  
 ((TextView)view).setText(data);  
 noIndicator.setText(position+1+"/"+list.size());  
 if (modelMatch()){  
 bookmarkBtn.setImageDrawable(getDrawable(R.drawable.*bookmark*));  
 }else{  
 bookmarkBtn.setImageDrawable(getDrawable(R.drawable.*bookmarks\_border*));  
 }  
 }catch (ClassCastException ex){  
 ((Button)view).setText(data);  
 }  
 view.setTag(data);  
 playAnim(view,1,data);  
 }  
  
 }  
  
 @Override  
 public void onAnimationCancel(Animator animation) {  
  
 }  
  
 @Override  
 public void onAnimationRepeat(Animator animation) {  
  
 }  
 });  
  
 }  
 private void checkAnswer(Button selectOption){  
 enableOption(false);  
 nextBtn.setEnabled(true);  
 nextBtn.setAlpha(1);  
 if (selectOption.getText().toString().equals(list.get(position).getCorrectANS())) {  
 //correct  
 score++;  
 selectOption.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#4CAF50")));  
  
 }else{  
 ///incorrect  
 selectOption.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#ff0000")));  
 Button correctoption = (Button) optionsContainer.findViewWithTag(list.get(position).getCorrectANS());  
 correctoption.setBackgroundTintList(ColorStateList.*valueOf*(Color.*parseColor*("#4CAF50")));  
 }  
  
 }  
 private void enableOption(boolean enable){  
 for (int i = 0; i<4; i++){  
 optionsContainer.getChildAt(i).setEnabled(enable);  
  
 }  
 }  
  
 private void getBookmarks(){  
  
 String json = preferences.getString(*KEY\_NAME*,"");  
  
 Type type = new TypeToken<List<QuestionModel>>(){}.getType();  
  
 bookmarksList = gson.fromJson(json,type);  
  
 if (bookmarksList == null){  
 bookmarksList = new ArrayList<>();  
 }  
 }  
  
 private boolean modelMatch(){  
 boolean matched = false;  
 int i = 0;  
 for (QuestionModel model : bookmarksList){  
 if(model.getQuestion().equals(list.get(position).getQuestion())  
 && model.getCorrectANS().equals(list.get(position).getCorrectANS())  
 && model.getSetNo() == list.get(position).getSetNo()){  
 matched = true;  
 matchedQuestionPosition = i;  
 }  
 i++;  
 }  
 return matched;  
 }  
  
 private void storeBookmarks(){  
  
 String json = gson.toJson(bookmarksList);  
 editor.putString(*KEY\_NAME*,json);  
 editor.commit();  
  
 }  
}

Данный модуль реализует отображение вопросов с БД Firebase. Представлено определение id вопроса, 4 вариантов ответа, правильного ответа и номера теста, а также получение информации в формате JSON, использовав стандартные параметры библиотеки Gson, создав экземпляр одноименного класса и вызвав метод toJson(). Реализовано отображение значка закладок. При прогружении вопроса из БД реализовано отображение загрузки.

myRef.child("SETS").child(category).child("questions").orderByChild("setNo").equalTo(setNo).addListenerForSingleValueEvent(new ValueEventListener()

Данная строчка кода обращается к дочерним ячейкам данных в Firebase. также было реализовано подсвечивание кнопки при выборе ответа (до этого она неактивна), а также счетчик правильных ответов. В любой момент можно выйти из прохождения теста, для этого была реализована кнопка выхода.

На ряду со всем остальным была реализована анимация появления вопросов, а также закрашивание правильных и неправильных ответов на вопрос(CheckAnswer).

* 1. **Questions Model**

package com.example.testpro;  
  
public class QuestionModel {  
 private String question,optionA,optionB,optionC,optionD,correctANS;  
 private int setNo;  
 public QuestionModel(){  
  
 }  
  
 public QuestionModel(String question, String optionA, String optionB, String optionC, String optionD, String correctANS,int setNo) {  
 this.setNo = setNo;  
 this.question = question;  
 this.optionA = optionA;  
 this.optionB = optionB;  
 this.optionC = optionC;  
 this.optionD = optionD;  
 this.correctANS = correctANS;  
 }  
  
 public String getQuestion() {  
 return question;  
 }  
  
 public void setQuestion(String question) {  
 this.question = question;  
 }  
  
 public String getOptionA() {  
 return optionA;  
 }  
  
 public void setOptionA(String optionA) {  
 this.optionA = optionA;  
 }  
  
 public String getOptionB() {  
 return optionB;  
 }  
  
 public void setOptionB(String optionB) {  
 this.optionB = optionB;  
 }  
  
 public String getOptionC() {  
 return optionC;  
 }  
  
 public void setOptionC(String optionC) {  
 this.optionC = optionC;  
 }  
  
 public String getOptionD() {  
 return optionD;  
 }  
  
 public void setOptionD(String optionD) {  
 this.optionD = optionD;  
 }  
  
 public String getCorrectANS() {  
 return correctANS;  
 }  
  
 public void setCorrectANS(String correctANS) {  
 this.correctANS = correctANS;  
 }  
  
 public int getSetNo() {  
 return setNo;  
 }  
}

Данный модуль является реализацией структуры хранения данных из Firebase для вопроса, 4 вариантов ответа, правильного ответа и номера теста. Был использован constructor для создания getters and setters.

* 1. **Score Activity**

package com.example.testpro;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
public class ScoreActivity extends AppCompatActivity {  
 private TextView scored,total;  
 private Button doneBtn;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_score*);  
  
 scored = findViewById(R.id.*scored*);  
 total = findViewById(R.id.*total*);  
 doneBtn = findViewById(R.id.*done\_btn*);  
  
 scored.setText(String.*valueOf*(getIntent().getIntExtra("score",0)));  
 total.setText("из возможных "+String.*valueOf*(getIntent().getIntExtra("total",0)));  
  
 doneBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
 });  
  
 }  
}

Данный класс реализован для отображения корректного отображения результатов прохождения тестов.

* 1. **Bookmark Activity**

package com.example.testpro;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.widget.LinearLayout;  
  
import com.google.gson.Gson;  
import com.google.gson.reflect.TypeToken;  
  
import java.lang.reflect.Type;  
import java.util.ArrayList;  
import java.util.List;  
  
import static com.example.testpro.QuestionsActivity.*FILE\_NAME*;  
import static com.example.testpro.QuestionsActivity.*KEY\_NAME*;  
  
public class BookmarkActivity extends AppCompatActivity {  
  
 private RecyclerView recyclerView;  
 private List<QuestionModel> bookmarksList;  
  
 private SharedPreferences preferences;  
 private SharedPreferences.Editor editor;  
 private Gson gson;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_bookmark*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
  
 setSupportActionBar(toolbar);  
 getSupportActionBar().setTitle("Закладки");  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
  
 recyclerView = findViewById(R.id.*rv\_bookmarks*);  
  
 preferences = getSharedPreferences(*FILE\_NAME*, Context.*MODE\_PRIVATE*);  
 editor = preferences.edit();  
 gson = new Gson();  
  
 getBookmarks();  
  
 LinearLayoutManager layoutManager = new LinearLayoutManager(this);  
 layoutManager.setOrientation(RecyclerView.*VERTICAL*);  
  
 recyclerView.setLayoutManager(layoutManager);  
  
 BookmarksAdapter adapter = new BookmarksAdapter(bookmarksList);  
 recyclerView.setAdapter(adapter);  
  
 }  
 @Override  
 protected void onPause() {  
 super.onPause();  
 storeBookmarks();  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
  
 if (item.getItemId() == android.R.id.*home*){  
 finish();  
 }  
 return super.onOptionsItemSelected(item);  
 }  
 private void getBookmarks(){  
  
 String json = preferences.getString(*KEY\_NAME*,"");  
  
 Type type = new TypeToken<List<QuestionModel>>(){}.getType();  
  
 bookmarksList = gson.fromJson(json,type);  
  
 if (bookmarksList == null){  
 bookmarksList = new ArrayList<>();  
 }  
 }  
  
 private void storeBookmarks(){  
  
 String json = gson.toJson(bookmarksList);  
 editor.putString(*KEY\_NAME*,json);  
 editor.commit();  
  
 }  
}

В данном классе представлено отображение выбранных закладок. Представлено получение информации в формате JSON, использовав стандартные параметры библиотеки Gson, создав экземпляр одноименного класса и вызвав метод toJson().

* 1. **Bookmarks Adapter**

package com.example.testpro;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageButton;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.List;  
  
public class BookmarksAdapter extends RecyclerView.Adapter<BookmarksAdapter.Viewholder> {  
  
 private List<QuestionModel> list;  
  
 public BookmarksAdapter(List<QuestionModel> list) {  
 this.list = list;  
 }  
  
 @NonNull  
 @Override  
 public Viewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*bookmark\_item*,parent,false);  
  
 return new Viewholder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull Viewholder holder, int position) {  
 holder.setData(list.get(position).getQuestion(),list.get(position).getOptionA(),list.get(position).getOptionB(),list.get(position).getOptionC(),list.get(position).getOptionD(),position);  
 }  
  
 @Override  
 public int getItemCount() {  
 return list.size();  
 }  
  
 class Viewholder extends RecyclerView.ViewHolder{  
  
 private ImageButton deleteBtn;  
  
 private TextView question,answer,a,b,c,d;  
  
 public Viewholder(@NonNull View itemView) {  
 super(itemView);  
  
  
 a = itemView.findViewById(R.id.*a*);  
 b = itemView.findViewById(R.id.*b*);  
 c = itemView.findViewById(R.id.*c*);  
 d = itemView.findViewById(R.id.*d*);  
  
 question = itemView.findViewById(R.id.*question*);  
 deleteBtn = itemView.findViewById(R.id.*delete\_btn*);  
 }  
  
 private void setData(String question, String a, String b, String c, String d, final int position){  
 this.a.setText("A) "+a);  
 this.b.setText("B) "+b);  
 this.c.setText("C) "+c);  
 this.d.setText("D) "+d);  
 this.question.setText("Вопрос: "+question);  
  
 deleteBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 list.remove(position);  
 notifyItemRemoved(position);  
 }  
 });  
 }  
 }  
}

Данный модуль представлен для обращения к методу setdata для хранения информации вопроса и описание кнопки удаления закладок.

# **ЗАКЛЮЧЕНИЕ**

В настоящее время в нашу жизнь широко внедряются информационные технологии. На сегодняшний день, наверное, не осталось такой области, где бы они ни применялись.

Постоянно растет объем и сложность обрабатываемой информации, требуются все новые и новые виды ее представления. Как показывает практика, большинство пользователей уже не представляет себе, как бы они выполняли свою работу, с которой прекрасно справлялись еще 5 - 10 лет назад, без помощи компьютера.

В ходе разработки курсового проекта было разработано приложение для тестирования в виде мобильного приложения: разработаны основные классы и методы, а также создан интуитивно понятный интерфейс.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Знакомство с Android Studio [Электронный ресурс] – Режим доступа: <http://startofandroid.com/ru/articles/listofarticles/284-znakomstvo-s-android-studio.html>
2. Введение в разработку Android-приложений [Электронный ресурс] – Режим доступа: <http://www.intuit.ru/studies/courses/4462/988/lecture/14988>
3. Канал «TCODE STUDIO» [Электронный ресурс] – Режим доступа: <https://www.youtube.com/channel/UCNFVIu9YEb1fGPnPz-pTl0g/playlists>