Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Уфимский государственный нефтяной технический университет

Кафедра вычислительной техники и инженерной кибернетики

КУРСОВАЯ РАБОТА

по дисциплине

Мобильные приложения и программирование устройств

ТАБЛИЦЫ ШУЛЬТЕ

Выполнил студент группы БПО 18-01 Р.И. Хузина

Принял ст. преподаватель Е.В. Дружинская

Дата представления работы: \_\_23.05.2021\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Результат: \_\_\_\_\_\_\_\_\_

Уфа, 2021 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc72694628)

[1. Обзор предметной области 3](#_Toc72694629)

[2. Проектирование приложения 4](#_Toc72694630)

[2.1 Функциональное моделирование 4](#_Toc72694631)

[2.2 Эскизирование экранов 5](#_Toc72694632)

[2.3 Логическое моделирование 6](#_Toc72694633)

[3. Программная реализация 8](#_Toc72694634)

[3.1 Главное меню 8](#_Toc72694635)

[3.2 Выбор таблицы 8](#_Toc72694636)

[3.3 Окно тренировок 9](#_Toc72694637)

[3.4 Результаты 11](#_Toc72694638)

[3.5 Активности с результатами по размерностям 13](#_Toc72694639)

[3.6 Информация по записи из ListView 14](#_Toc72694640)

[3.7 Работа с базой данных. Класс ResultsDBHelper 15](#_Toc72694641)

[3.8 Пользовательский курсор 15](#_Toc72694642)

[3.9 Секундомер 16](#_Toc72694643)

[ЗАКЛЮЧЕНИЕ 19](#_Toc72694644)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_Toc72694645)

# **ВВЕДЕНИЕ**

За последние годы сотовые телефоны стали неотъемлемой частью нашей жизни. И это неудивительно, ведь высокий ритм жизни, реализация повседневных целей и задач диктует нам необходимость быть всегда на связи и иметь под рукой верного помощника. Вместе с увеличением роли мобильных телефонов в жизни человека, становится более актуальной разработка приложений на них.

Android  — самая популярная операционная система и платформа для приложений, насчитывающая больше двух миллиардов активных пользователей. На ней работают совершенно разные устройства, от «интернета вещей» и умных часов до телевизоров, ноутбуков и автомобилей, но чаще всего Android используют на смартфонах и планшетах. [1]

Мобильные приложения - компьютерная программа, созданная специально для использования в мобильном телефоне, смартфоне или коммуникаторе, которая предназначена для выполнения той или иной задачи. [2]. В большинстве всех мобильных устройств установлена ОС Android. Поэтому разработка приложений является перспективным направлением.

В качестве темы индивидуального проекта было решено взять реализацию таблиц Шульте в среде Android Studio. Целевой аудиторией данного приложения будут являться люди всех возрастов, так как приложение относится к категории развивающих.

Целью данного проекта является написание многофункционального приложения, а в следствии и закрепление полученных знаний.

Задачи данного проекта:

1. Создать корректно функционирующее приложение;
2. Разработать привлекательный дизайн;
3. Обеспечить приложению многоплатформенность;
4. Сделать приложение интуитивно понятным для людей всех возрастов.
5. **Обзор предметной области**

В качестве темы индивидуального проекта были выбраны таблицы Шульте. Таблицы Шульте — таблицы со случайно расположенными объектами, служащие для проверки и развития быстроты нахождения этих объектов в определённом порядке.

При работе с таблицами Шульте развивается прежде всего объемное (параллельное) внимание. Символы в поле вашего зрения считываются зрительной памятью одновременно, а не каждый в отдельности. Это способствует ускорению интеллектуального развития. [3]

При регулярном использовании данный тренинг расширяет периферийное зрение и это позволяет повысить скорость чтения как за счет охвата большего пространства читаемого текста, так и за счет освоения параллельного режима (а не сканирующего, последовательного) распознавания печатных символов. [4]

Был проведен анализ существующих приложений. В состав всех приложений, помимо других второстепенных, входят два основных компонента: тренировка на таблице и результаты.

При проектировании приложения необходимо аналогично обязательно реализовать тренировки и анализ данных.

Проанализировав отзывы людей по существующим приложениям можно выделить следующие моменты, которые стоит учесть при создании приложения:

1. Слишком яркие цвета использовать не стоит, так как глаза будут находиться в постоянном напряжении;
2. Во времени должны отображаться миллисекунды;
3. Желательно реализовать возможность сброса попытки.

Таким образом, при создании приложения стоит учесть отмеченные моменты.

1. **Проектирование приложения**
   1. **Функциональное моделирование**

При открытии приложения перед пользователем будет открыто меню с тремя кнопками:

* тренировка;
* результаты;
* выход.

**Кнопка «Тренировка».** У пользователя будет выбор размера таблицы. После чего начнется сама тренировка. На экране пользователя высветится таблица и начнется отсчет времени, отображенный над таблицей, под таблицей есть две кнопки: «сохранить результат» и «назад». При нажатии на цифру, задний фон будет подсвечиваться другим цветом. При достижении конца таблицы отсчет времени останавливается. Есть возможность сохранить результат, нажав на кнопку «сохранить результат».

**Кнопка «Результаты».** Здесь будет отображено число, размерность таблицы и время, затраченное на таблицу. Также будет отображен лучший результат.

**Кнопка «Выход».** Завершение работы приложения.

* 1. **Эскизирование экранов**

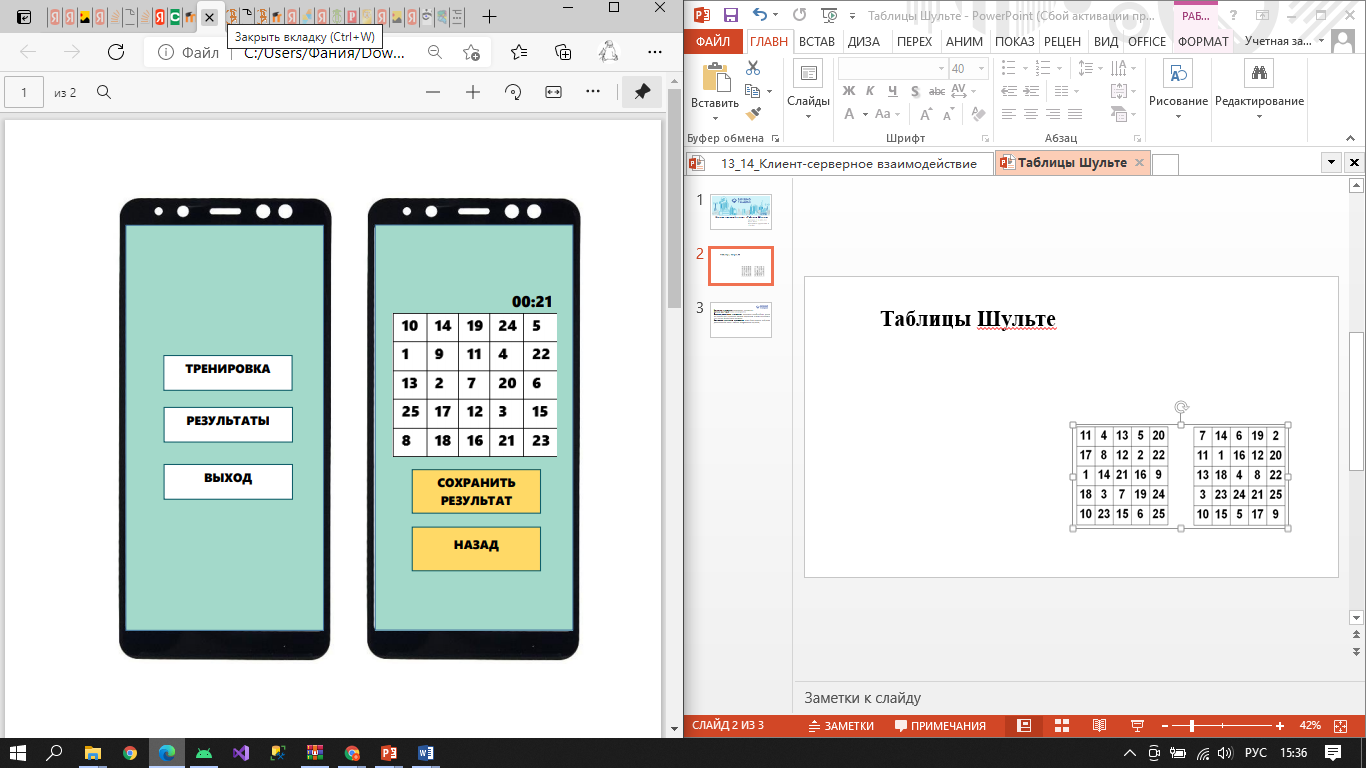
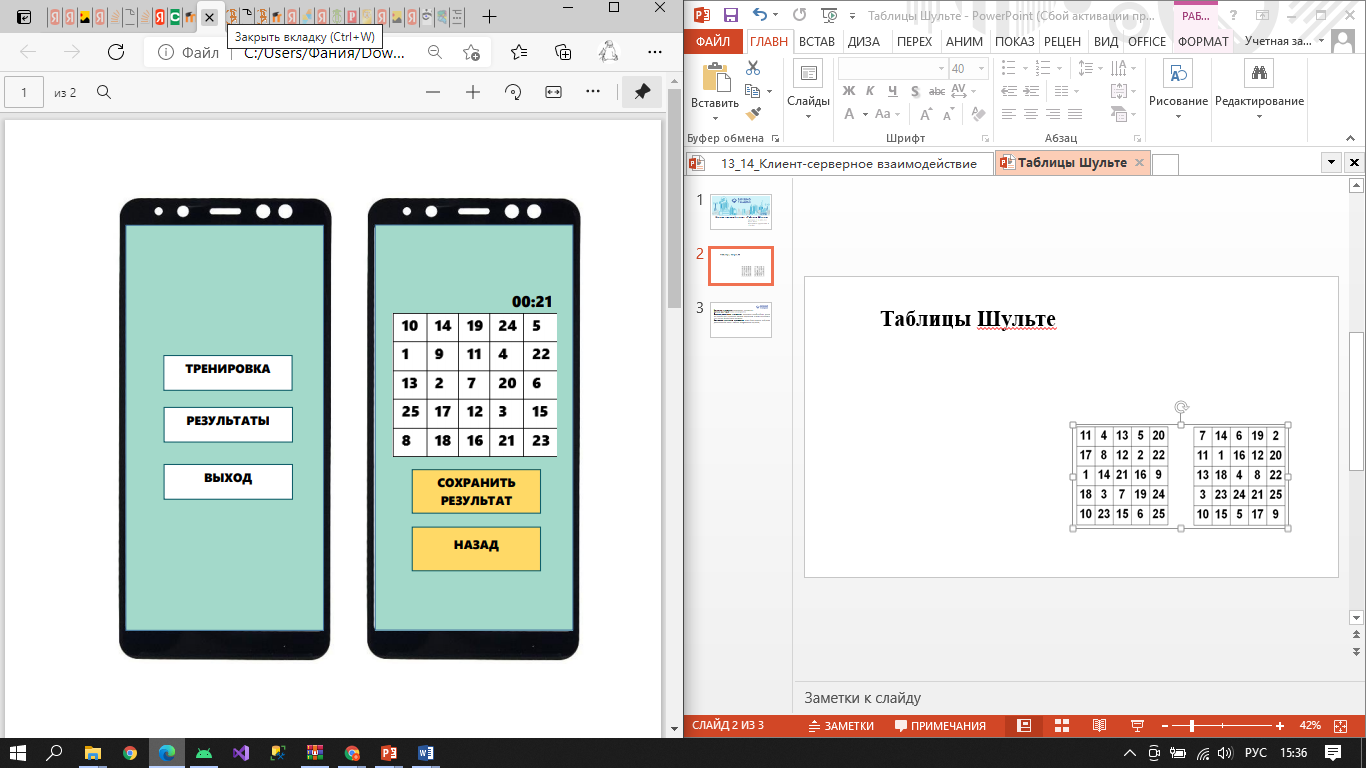
 

Рисунок 1. Главное меню Рисунок 2. Тренировка

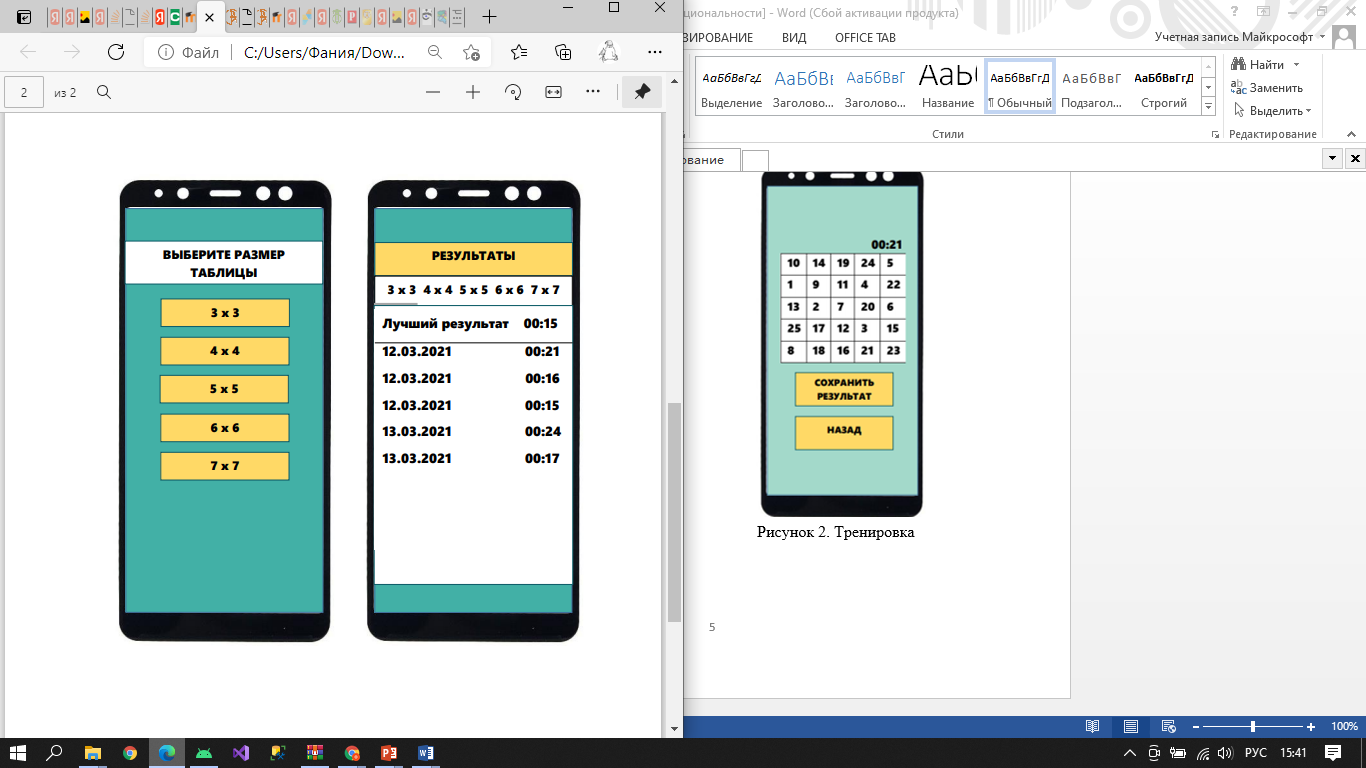
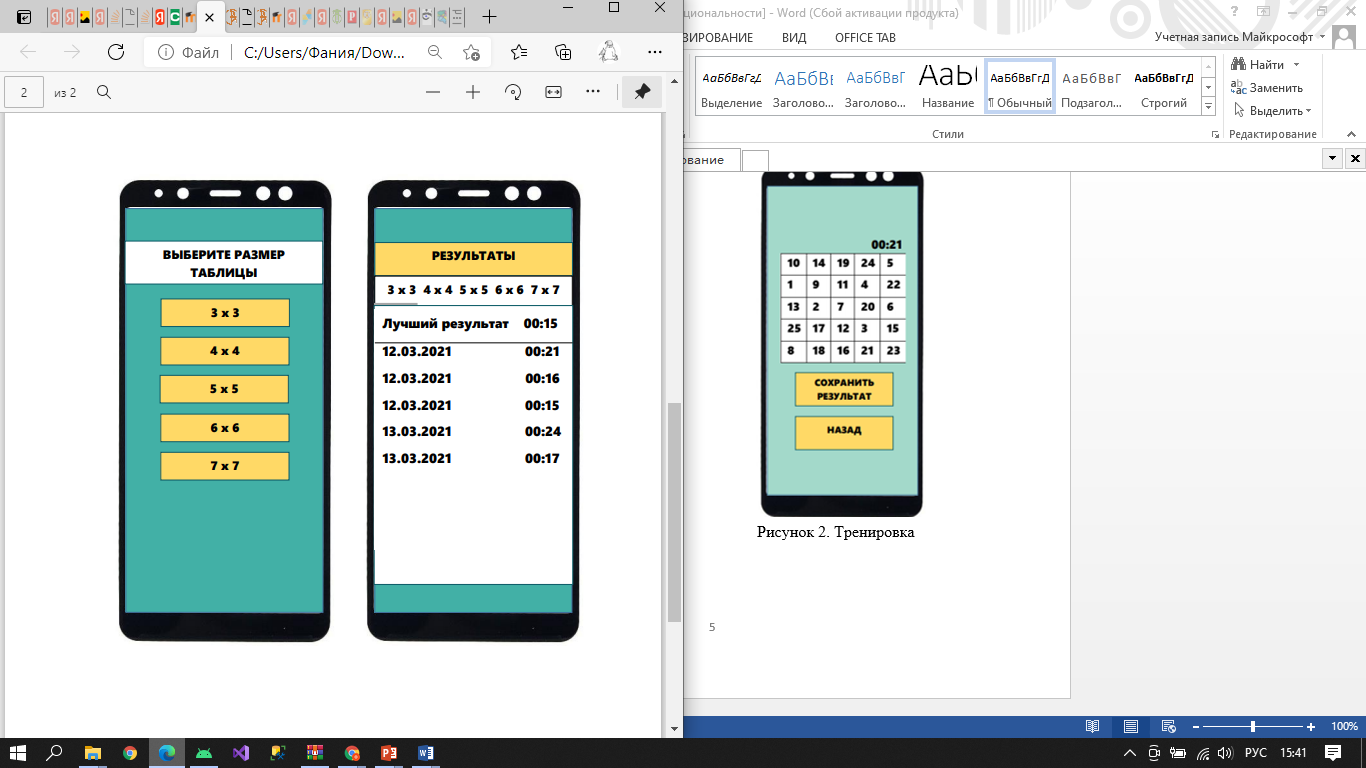
 

Рисунок 3. Выбор размерности таблицы Рисунок 4. Анализ результатов

* 1. **Логическое моделирование**
* **Класс MainActivity**- расширяет класс AppCompatActivity. В данной активности реализуется меню приложения, состоящее из трех кнопок. Кнопка «Тренировка» запускает активность Choice, кнопка «Результаты» запускает активность Results, по кнопке «Выход» завершается работа приложения.
* **Класс Choice** – расширяет класс AppCompatActivity. В данной активности реализуется выбор размерности таблицы. Кнопка «3х3» осуществляет запуск активности Training3. Кнопка «4х4» осуществляет запуск активности Training4. Кнопка «5х5» осуществляет запуск активности Training5. Кнопка «6х6» осуществляет запуск активности Training6. Кнопка «7х7» осуществляет запуск активности Training7.
* **Классы Training(n) (3<=n<=7)**- расширяет класс AppCompatActivity. В данной активности реализована непосредственно сама тренировка. Кнопка «Сохранить результат» через объект Intent и метод putExtra() передает данные в класс TabResults(n). Кнопка «Назад» осуществляет запуск активности MainActivity.
* **Класс Results**- расширяет класс TabActivity. Данная активность содержит все результаты тренировок. Вкладка «3х3» запускает активность TabResults3. Вкладка «4x4» запускает активность TabResults4. Вкладка «5x5» запускает активность TabResults5. Вкладка «6x6» запускает активность TabResults6. Вкладка «7x7» запускает активность TabResults7.
* **Классы TabResults(n) (3<=n<=7)**- расширяют классы AppCompatActivity. В данных активностях содержатся сохраненные результаты тренировок. Получают данные из соответствующих им классов Training(n).

1. **Программная реализация**
   1. **Главное меню**

Главное меню реализовано в классе Start\_window, расширяющий класс AppCompatActivity.

Реализованы обработчики нажатия, реализующие переходы на соответствующие активности, для следующих кнопок:

* Тренировка

**public void** onClickStart1(View view){  
 Intent intent=**new** Intent(**this**,Chose\_window.**class**);  
 startActivity(intent);  
}

* Результаты

**public void** onClickStart2(View view){  
 Intent intent=**new** Intent(**this**, Results\_main.**class**);  
 startActivity(intent);  
}

* Выход

**public void** onClickStart3(View view){  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);  
  
}

Переопределен метод нажатия на кнопку навигации «Назад». При двойном нажатии в течении двух секунд приложение закрывается.

**public void** onBackPressed() {  
 **if** (*back\_pressed* + 2000 > System.*currentTimeMillis*()){  
 **super**.onBackPressed();  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);}  
 **else**{ Toast toast=Toast.*makeText*(**this**, **"Для выхода нажмите еще один раз"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();}  
  
 *back\_pressed* = System.*currentTimeMillis*();  
}

* 1. **Выбор таблицы**

Реализованы обработчики нажатия кнопок, реализующие переход на активность, где реализована таблица, размерностью, указанной на кнопке. Так как методы аналогичные, приведем код для кнопки 3х3.

**public void** onClickChoose3(View view){  
 Intent intent=**new** Intent(**this**,Window\_3x3.**class**);  
 startActivity(intent);  
}

Переопределен метод нажатия на кнопку навигации «Назад». При двойном нажатии в течении двух секунд приложение закрывается.

**public void** onBackPressed() {  
 **if** (*back\_pressed* + 2000 > System.*currentTimeMillis*()){  
 **super**.onBackPressed();  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);}  
 **else**{ Toast toast=Toast.*makeText*(**this**, **"Для выхода нажмите еще один раз"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();}  
  
 *back\_pressed* = System.*currentTimeMillis*();

## **Окно тренировок**

Метод установки кнопок. Реализуется создание ArrayList с количеством элементов, соответствующих размерности таблицы. Метод Shuffle перемешивает значения. Затем кнопкам по порядку устанавливаются значения перемешанного ArrayList.В соответствии с шириной кнопки устанавливается размер текста на кнопке. Разметка TableLayout устанавливается видимой, кнопки красятся в белый цвет. Часть кода приведенного ниже аналогична для всех активностей, реализующих таблицы разных размерностей.

**public void** setButton(**int** width,**int** height){  
 **int**[] arr=**new int**[9];  
 String[]arr2=**new** String[9];  
 ArrayList<Integer> arrayList=**new** ArrayList<>();  
 **for**(**int** i=0;i<9;i++){  
 arr[i]=i+1;  
 arrayList.add(arr[i]);  
 }  
 Collections.*shuffle*(arrayList);  
 **for** (**int** i = 0; i != arrayList.size(); i++) {  
 arr[i] = arrayList.get(i);  
 arr2[i]=Integer.*toString*(arr[i]);  
 }  
  
 **bt1**.setText(arr2[0]);**bt1**.setTextSize(width/4);  
 **bt2**.setText(arr2[1]);**bt2**.setTextSize(width/4);  
 **bt3**.setText(arr2[2]);**bt3**.setTextSize(width/4);  
 **bt4**.setText(arr2[3]);**bt4**.setTextSize(width/4);  
 **bt5**.setText(arr2[4]);**bt5**.setTextSize(width/4);  
 **bt6**.setText(arr2[5]);**bt6**.setTextSize(width/4);  
 **bt7**.setText(arr2[6]);**bt7**.setTextSize(width/4);  
 **bt8**.setText(arr2[7]);**bt8**.setTextSize(width/4);  
 **bt9**.setText(arr2[8]);**bt9**.setTextSize(width/4);  
 TableLayout table=findViewById(R.id.***table3***);  
 table.setVisibility(View.***INVISIBLE***);  
  
 **bt1**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt2**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt3**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt4**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt5**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt6**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt7**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt8**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
 **bt9**.setBackgroundColor(Color.*parseColor*(**"#ffffff"**));  
  
}

Метод, реализующий запуск секундомера.

**public void** onclickStartClock3(View view){  
 **clock** = findViewById(R.id.***simpleChronometer3***);  
 Toast toast=Toast.*makeText*(**this**, **"Найдите числа от 1 до 9"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();  
 **clock**.setBase(SystemClock.*elapsedRealtime*());  
 **clock**.start();  
 TableLayout table=findViewById(R.id.***table3***);  
 table.setVisibility(View.***VISIBLE***);  
 Button bt=(Button)findViewById(R.id.***start3***);  
 bt.setVisibility(View.***INVISIBLE***);  
}

Метод, реализующий обработку нажатия на кнопку в таблице. При значении текста на кнопке равного размерности таблицы секундомер останавливается. При нажатии на кнопку происходит покраска кнопки в соответствующий цвет.

**public void** onClick(View view){  
 Button btn=(Button)view;  
 **if**(btn.getText().toString().equals(Integer.*toString*(**i**))){btn.setBackgroundColor(Color.*parseColor*(**"#F0D5F8"**));**i**++;  
 **if**(**i**==**N**){**clock**.stop();**save**.setVisibility(View.***VISIBLE***);}}  
}

Метод, реализующий обновление таблицы. Часть кода приведенного ниже аналогична для всех активностей, реализующих таблицы разных размерностей.

**public void** onClickRefresh3(View view){  
 **i**=1;  
 setButton(**width**,**height**);  
 **clock** = findViewById(R.id.***simpleChronometer3***);  
 Button bt=(Button)findViewById(R.id.***start3***);  
 bt.setVisibility(View.***VISIBLE***);  
 **clock**.setBase(SystemClock.*elapsedRealtime*());  
 **clock**.stop();  
**save**.setVisibility(View.***INVISIBLE***);  
  
 }

Метод, реализующий сохранение результата в базу данных. При сохранении передаются тип (размерность таблицы, например «3х3»), дата (текущая дата на устройстве), и время с секундомера.

**public void** onClickSave3(View view){  
 String time= (String) **clock**.getText();  
 Date currentDate = **new** Date();  
*// Форматирование времени как "день.месяц.год"* DateFormat dateFormat = **new** SimpleDateFormat(**"dd.MM.yyyy"**, Locale.*getDefault*());  
 String date = dateFormat.format(currentDate);  
 ContentValues cv = **new** ContentValues();  
 cv.put(ResultsDBHelper.***RESULTS\_COLUMN\_TYPE***, **"3x3"**);  
 cv.put(ResultsDBHelper.***RESULTS\_COLUMN\_DAY***, date);  
 cv.put(ResultsDBHelper.***RESULTS\_COLUMN\_TIME***, time);  
 **db**.insert(ResultsDBHelper.***TABLE\_RESULTS***, **null**, cv);  
 Toast toast=Toast.*makeText*(**this**, **"Успешно сохранено"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();  
  
  
 }

Переопределен метод нажатия на кнопку навигации «Назад». При двойном нажатии в течении двух секунд приложение закрывается.

**public void** onBackPressed() {  
 **if** (*back\_pressed* + 2000 > System.*currentTimeMillis*()){  
 **super**.onBackPressed();  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);}  
 **else**{ Toast toast=Toast.*makeText*(**this**, **"Для выхода нажмите еще один раз"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();}  
  
 *back\_pressed* = System.*currentTimeMillis*();

Обработчик нажатия кнопки назад. Перебрасывает на активность с выбором размерностей.

**public void** onClickBack3(View view){  
 Intent intent=**new** Intent(**this**,Chose\_window.**class**);  
 startActivity(intent);  
}

* 1. **Результаты**

Установка активностей в TabHost. В методе onCreate().

TabHost tabHost = getTabHost();  
4TabHost.TabSpec tabSpec;  
  
tabSpec = tabHost.newTabSpec(**"tag3"**);  
tabSpec.setIndicator(**"3x3"**);  
tabSpec.setContent(**new** Intent(**this**, Results\_3x3.**class**));  
tabHost.addTab(tabSpec);  
  
tabSpec = tabHost.newTabSpec(**"tag4"**);  
tabSpec.setIndicator(**"4x4"**);  
tabSpec.setContent(**new** Intent(**this**, Results\_4x4.**class**));  
tabHost.addTab(tabSpec);  
tabSpec = tabHost.newTabSpec(**"tag5"**);  
tabSpec.setIndicator(**"5x5"**);  
tabSpec.setContent(**new** Intent(**this**, Results\_5x5.**class**));  
tabHost.addTab(tabSpec);  
  
tabSpec = tabHost.newTabSpec(**"tag6"**);  
tabSpec.setIndicator(**"6x6"**);  
tabSpec.setContent(**new** Intent(**this**, Results\_6x6.**class**));  
tabHost.addTab(tabSpec);  
  
tabSpec = tabHost.newTabSpec(**"tag7"**);  
tabSpec.setIndicator(**"7x7"**);  
tabSpec.setContent(**new** Intent(**this**, Results\_7x7.**class**));  
tabHost.addTab(tabSpec);

Обработчик нажатия на кнопку удаления всех записей из базы данных.

**public void** onClickDeleteAll(View view){  
 **dialog**.show();  
}

Вызывает диалоговое окно с двумя кнопками согласия/несогласия. В случае согласия очищается база данных. В случае несогласия диалоговое окно скрывается.

**dialog** = **new** AlertDialog.Builder(Results\_main.**this**);  
 **dialog**.setTitle(**"Внимание!"**);  
 **dialog**.setMessage(**"Желаете ли Вы удалить все результаты?"**);  
 **dialog**.setPositiveButton(**"Да"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
**db**.delete(ResultsDBHelper.***TABLE\_RESULTS***,**null**,**null**);  
 finish();  
 startActivity(getIntent());  
  
 }  
 });  
 **dialog**.setNegativeButton(**"Отмена"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.dismiss();  
 }  
 });

Кнопка назад возвращает на активность главного меню.

**public void** onClickBackFromResults(View view){  
 Intent intent=**new** Intent(**this**,Start\_window.**class**);  
 startActivity(intent);  
}

Переопределен метод нажатия на кнопку навигации «Назад». При двойном нажатии в течении двух секунд приложение закрывается.

**public void** onBackPressed() {  
 **if** (*back\_pressed* + 2000 > System.*currentTimeMillis*()){  
 **super**.onBackPressed();  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);}  
 **else**{ Toast toast=Toast.*makeText*(**this**, **"Для выхода нажмите еще один раз"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();}  
  
 *back\_pressed* = System.*currentTimeMillis*();

* 1. **Активности с результатами по размерностям**

Слушатель для нажатия на элемент ListView. Запускает активность об соответствующей записи.

**listView**.setOnItemClickListener((parent, view, position, id) -> {  
 Intent i = **new** Intent(**this**, Date\_information.**class**);  
  
 i.putExtra(**"id"**, id);  
 TextView dayText = view.findViewById(R.id.***data***);  
 i.putExtra(**"date"**, dayText.getText());  
  
 TextView timeText = view.findViewById(R.id.***time***);  
 i.putExtra(**"time"**, timeText.getText());  
  
  
 startActivityForResult(i, **REQUEST\_ACCESS\_TYPE**);  
  
 setListView();  
});

@Override  
**public void** onActivityResult(**int** requestCode, **int** resultCode, Intent data){  
  
 **if**(requestCode==**REQUEST\_ACCESS\_TYPE**){  
 **if**(resultCode==***RESULT\_OK***){  
 setListView();  
 Toast.*makeText*(**this**, **"Запись удалена "**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }  
 **else**{  
 **super**.onActivityResult(requestCode, resultCode, data);  
 }  
}

Установка данных из базы данных.

**private void** setDB(){  
 *//открыть подключение к бд* **databaseHelper** = **new** ResultsDBHelper(**this**);  
 **db** = **databaseHelper**.getReadableDatabase();  
  
 *//обновить list view* setListView();  
}

Установка данных в listview.

**private void** setListView(){  
 *//получить данные в курсор* **resultsCursor** = ResultsDBHelper.*getResultsByType*(**db**,**"3x3"**);  
 *//создаем адаптер и помещаем в него курсор* **resultsAdapter** = **new** MySimpleCursorAdapter(  
 **this**, **resultsCursor**, ResultsDBHelper.*headersWithIdAll*, ResultsDBHelper.*guiListAll*,  
 0);  
 *//listview* **listView**.setAdapter(**resultsAdapter**);  
}

Установка времени для лучшего результата.

**int** check=**resultsCursor**.getCount();  
**if**(check!=0){  
**while**(**resultsCursor**.moveToNext()) {  
 String t = **resultsCursor**.getString(**resultsCursor**.getColumnIndex(ResultsDBHelper.***RESULTS\_COLUMN\_TIME***));  
 **arrayList**.add(t);  
}  
 Collections.*sort*(**arrayList**);  
 **best**.setText(**arrayList**.get(0));}  
**else best**.setText(**"-"**);

Переопределен метод нажатия на кнопку навигации «Назад». При двойном нажатии в течении двух секунд приложение закрывается.

**public void** onBackPressed() {  
 **if** (*back\_pressed* + 2000 > System.*currentTimeMillis*()){  
 **super**.onBackPressed();  
 **this**.finish(); Intent intent = **new** Intent(Intent.***ACTION\_MAIN***); intent.addCategory(Intent.***CATEGORY\_HOME***); intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***); startActivity(intent);}  
 **else**{ Toast toast=Toast.*makeText*(**this**, **"Для выхода нажмите еще один раз"**, Toast.***LENGTH\_SHORT***);  
 toast.setGravity(Gravity.***BOTTOM***,0,0);  
 toast.show();}  
  
 *back\_pressed* = System.*currentTimeMillis*();

* 1. **Информация по записи из ListView**

Установка данных, полученных из предыдущей активности

**id** = getIntent().getLongExtra(**"id"**, 0);  
**date** = findViewById(R.id.***selected\_data***);  
**time** = findViewById(R.id.***seelcted\_time***);  
**delete** = findViewById(R.id.***delete***);  
**date**.setText(getIntent().getStringExtra(**"date"**));  
**time**.setText(getIntent().getStringExtra(**"time"**));

Обработчик нажатия на кнопку Удалить, удаляющий данную запись из базы данных.

**public void** onClickDelete(View view){  
 **db**.delete(ResultsDBHelper.***TABLE\_RESULTS***, ResultsDBHelper.***RESULTS\_COLUMN\_ID*** + **"="** + **id**, **null**);  
 setResult(***RESULT\_OK***, **new** Intent());  
 finish();  
}

Кнопка назад.

**public void** onClickBackFromSelect(View view)  
{  
 finish();  
}}

* 1. **Работа с базой данных. Класс ResultsDBHelper**

Название таблицы и столбцов

**static final** String ***TABLE\_RESULTS*** = **"results"**;  
**public static final** String ***RESULTS\_COLUMN\_ID*** = **"\_id"**;  
**public static final** String ***RESULTS\_COLUMN\_TIME*** = **"time"**;  
**public static final** String ***RESULTS\_COLUMN\_TYPE*** = **"type"**;  
**public static final** String ***RESULTS\_COLUMN\_DAY*** = **"day"**;

Создание таблицы

@Override  
**public void** onCreate(SQLiteDatabase db) {  
 db.execSQL(**"CREATE TABLE results ("** +  
 ***RESULTS\_COLUMN\_ID*** + **" INTEGER PRIMARY KEY AUTOINCREMENT, "** +  
 ***RESULTS\_COLUMN\_TYPE*** + **" TEXT, "** +  
 ***RESULTS\_COLUMN\_DAY*** + **" TEXT, "** +  
 ***RESULTS\_COLUMN\_TIME*** + **" TEXT);"**);  
  
}

Запрос в базу данных на уничтожение таблицы

@Override  
**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 db.execSQL(**"DROP TABLE IF EXISTS "**+ ***TABLE\_RESULTS***);  
 onCreate(db);  
}

Получение результатов по типу

**public static** Cursor getResultsByType(SQLiteDatabase db, String name){  
 **return** db.query(***TABLE\_RESULTS***, *headersWithIdAll*, **"type = ?"**,  
 **new** String[]{name}, **null**, **null**, **null** );  
}

* 1. **Пользовательский курсор**

Установка значений в ListView

**public** MySimpleCursorAdapter(Context context, Cursor c, String[] from, **int**[] to, **int** flags) {  
 **super**(context, R.layout.***row\_layout***, c, from, to, flags);  
}  
  
@Override  
**public void** bindView(View view, Context context, Cursor cursor) {  
 String day = cursor.getString(cursor.getColumnIndex(ResultsDBHelper.***RESULTS\_COLUMN\_DAY***));  
 String time = cursor.getString(cursor.getColumnIndex(ResultsDBHelper.***RESULTS\_COLUMN\_TIME***));  
  
  
 TextView dayText = view.findViewById(R.id.***data***);  
 dayText.setText(day);  
  
 TextView timeText = view.findViewById(R.id.***time***);  
 timeText.setText(time);  
  
  
  
}  
  
@Override  
**public** View newView(Context context, Cursor cursor, ViewGroup parent) {  
 LayoutInflater inflater = (LayoutInflater) context.getSystemService(context.***LAYOUT\_INFLATER\_SERVICE***);  
 **return** inflater.inflate(R.layout.***row\_layout***, parent, **false**);  
}

* 1. **Секундомер**

В качестве секундомера c миллисекундами используется класс Chronometer, расширяющий класс TextView.

Ссылка на источник: [Millisecond-Chronometer/Chronometer.java at master · antoniom/Millisecond-Chronometer (github.com)](https://github.com/antoniom/Millisecond-Chronometer/blob/master/src/gr/antoniom/chronometer/Chronometer.java)

**public class** Chronometer **extends** TextView {  
 @SuppressWarnings(**"unused"**)  
 **private static final** String ***TAG*** = **"Chronometer"**;  
 **public interface** OnChronometerTickListener {  
 **void** onChronometerTick(Chronometer chronometer);  
 }  
 **private long mBase**;  
 **private boolean mVisible**;  
 **private boolean mStarted**;  
 **private boolean mRunning**;  
 **private** OnChronometerTickListener **mOnChronometerTickListener**;  
 **private static final int *TICK\_WHAT*** = 2;  
 **private long timeElapsed**;  
  
 **public** Chronometer(Context context) {  
 **this** (context, **null**, 0);  
 }  
 **public** Chronometer(Context context, AttributeSet attrs) {  
 **this** (context, attrs, 0);  
 }  
 **public** Chronometer(Context context, AttributeSet attrs, **int** defStyle) {  
 **super** (context, attrs, defStyle);  
 init();  
 }  
 **private void** init() {  
 **mBase** = SystemClock.*elapsedRealtime*();  
 updateText(**mBase**);  
 }  
 **public void** setBase(**long** base) {  
 **mBase** = base;  
 dispatchChronometerTick();  
 updateText(SystemClock.*elapsedRealtime*());  
 }  
 **public long** getBase() {  
 **return mBase**;  
 }  
  
 **public void** setOnChronometerTickListener(  
 OnChronometerTickListener listener) {  
 **mOnChronometerTickListener** = listener;  
 }  
 **public** OnChronometerTickListener getOnChronometerTickListener() {  
 **return mOnChronometerTickListener**;  
 }  
 **public void** setStarted(**boolean** started) {  
 **mStarted** = started;  
 updateRunning();}  
 @Override  
 **protected void** onDetachedFromWindow() {  
 **super** .onDetachedFromWindow();  
 **mVisible** = **false**;  
 updateRunning();  
 }  
 @Override  
 **protected void** onWindowVisibilityChanged(**int** visibility) {  
 **super** .onWindowVisibilityChanged(visibility);  
 **mVisible** = visibility == ***VISIBLE***;  
 updateRunning();  
 }  
 **private synchronized void** updateText(**long** now) {  
 **timeElapsed** = now - **mBase**;  
 DecimalFormat df = **new** DecimalFormat(**"00"**);  
 **int** hours = (**int**)(**timeElapsed** / (3600 \* 1000));  
 **int** remaining = (**int**)(**timeElapsed** % (3600 \* 1000));  
 **int** minutes = (**int**)(remaining / (60 \* 1000));  
 remaining = (**int**)(remaining % (60 \* 1000));  
 **int** seconds = (**int**)(remaining / 1000);  
 remaining = (**int**)(remaining % (1000));  
 **int** milliseconds = (**int**)(((**int**)**timeElapsed** % 1000) / 100);  
 String text = **""**;  
 **if** (hours > 0) {  
 text += df.format(hours) + **":"**;}  
 text += df.format(minutes) + **":"**;  
 text += df.format(seconds) + **":"**;  
 text += Integer.*toString*(milliseconds);  
 setText(text)}  
 **private void** updateRunning() {  
 **boolean** running = **mVisible** && **mStarted**;  
 **if** (running != **mRunning**) {  
 **if** (running) {  
 updateText(SystemClock.*elapsedRealtime*());  
 dispatchChronometerTick();  
 **mHandler**.sendMessageDelayed(Message.*obtain*(**mHandler**,  
 ***TICK\_WHAT***), 100);  
 } **else** {  
 **mHandler**.removeMessages(***TICK\_WHAT***);  
 }  
 **mRunning** = running;  
 }  
 }  
 **private** Handler **mHandler** = **new** Handler() {  
 **public void** handleMessage(Message m) {  
 **if** (**mRunning**) {  
 updateText(SystemClock.*elapsedRealtime*());  
 dispatchChronometerTick();  
 sendMessageDelayed(Message.*obtain*(**this** , ***TICK\_WHAT***),  
 100)  
 };  
 **void** dispatchChronometerTick() {  
 **if** (**mOnChronometerTickListener** != **null**) {  
 **mOnChronometerTickListener**.onChronometerTick(**this**);  
 }  
 **public long** getTimeElapsed() {  
 **return timeElapsed**;  
 }  
  
}

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была написана мобильное приложение, реализующее таблицы Шульте. При использовании поставленных задач, была достигнута цель, поставленная в начале проекта – «написание многофункционального приложения, а в следствии и закрепление полученных знаний.»

В получившемся приложении можно тренироваться и хранить результаты этих тренировок. При желании, можно очистить все результаты, либо определенные записи с результатами.

Приложению соответствует условиям, поставленным на этапе проектирования. Функционал приложения в ходе создания несколько расширился, вобрав в себя некоторые функции, которые не были учтены в первоначальном проектировании. Основная концепция приложения не изменилась.

Дизайн приложения, разработанный на этапе проектирования практически не изменен. Изменились положения и вид кнопок. Были добавлены изображения, которые, по моему мнению, улучшают дизайн этого приложения.

Данное приложение корректно функционирует. Все кнопки в приложении являются рабочими.

Данное приложение может быть модифицировано. Можно отметить такие моменты, которые можно реализовать в следующих обновлениях:

1. Добавление графиков, которые наглядно покажут результаты тренировок.
2. Перемешивание таблицы после каждого нажатия на элемент таблицы;
3. Реализовать таблицы с буквами;

Подводя итоги, можно сказать, что в выполненном проекте реализованы все задачи, которые стояли перед этапом разработки. Приложение хорошо функционирует, и может быть улучшено в будущем.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Как работает Android. Часть 1 [Электронный ресурс]. – URL:

[https://habr.com/ru/company/solarsecurity/blog/334796/](https://habr.com/ru/company/solarsecurity/blog/334796/%20) (дата обращения 22.05.2021)

1. Мобильные приложения [Электронный ресурс]. – URL:

[Мобильные приложения (tadviser.ru)](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F) (дата обращения 22.05.2021)

1. Таблицы Шульте: Коротко об известном [Электронный ресурс]. – URL:

[https://brainapps.ru/blog/2020/10/tablica-shulte-glavnoe/](https://brainapps.ru/blog/2020/10/tablica-shulte-glavnoe/%20) (дата обращения 22.05.2021)

1. Таблицы Шульте и динамическое зрение [Электронный ресурс]. – URL:

<http://bessmerten.ru/speed_reading/shulte01.html> (дата обращения 23.05.2021)