МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Программное средство Black Box»

Выполнил студент: Сыман Евгений Александрович

(Ф.И.О.)

Руководитель проекта: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой: к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc42330452)

[**1 Обзор аналогов** 4](#_Toc42330453)

[**1.1 Сервис «Jira»** 4](#_Toc42330454)

[**1.2 Web-сервис «Slack»** 5](#_Toc42330455)

[**1.3 Диаграмма «GanttPro»** 6](#_Toc42330456)

[**1.4 Выводы по главе «Обзор аналогов»** 7](#_Toc42330457)

[**2 Проектирование программного продукта** 8](#_Toc42330458)

[**2.1 Проектирование базы данных** 8](#_Toc42330459)

[**2.2 Проектирование мобильного приложения** 10](#_Toc42330460)

[**2.3 Проектирование web-приложения** 10](#_Toc42330461)

[**2.4 Выводы по главе «Проектирование программного продукта»** 11](#_Toc42330462)

[**3 Программная реализация приложений** 12](#_Toc42330463)

[**3.1 Технические средства для разработки** 12](#_Toc42330464)

[**3.2 Разработка баз данных** 12](#_Toc42330465)

[**3.3 Разработка мобильного приложения** 13](#_Toc42330466)

[**3.4 Разработка сервера** 19](#_Toc42330467)

[**3.5 Вывод по главе «Программная реализация приложений»** 21](#_Toc42330468)

[**4 Руководство пользователя** 22](#_Toc42330469)

[**4.1 Выводы по главе «Руководство пользователя»** 24](#_Toc42330470)

[**ЗАКЛЮЧЕНИЕ** 25](#_Toc42330471)

[**Список использованных источников** 26](#_Toc42330472)

[**ПРИЛОЖЕНИЕ A** 27](#_Toc42330473)

[**ПРИЛОЖЕНИЕ Б** 29](#_Toc42330474)

[**ПРИЛОЖЕНИЕ В** 37](#_Toc42330475)

# **ВВЕДЕНИЕ**

Цель: сосредоточить возможности звукозаписи в единый центр управления, что позволит защитить пользователя и его конфиденциальность.

Задачи:

* Изучение аналогичных программных средства и литературные источники по теме курсового проекта.

Данный подход поможет узнать уже реализованные программные продукты, а также их особенности (положительные стороны), недочёты (плохие решения, ошибки) которые не позволяют приложению развиваться либо приносить прибыль. Также это позволит обосновано использовать те или иные решения в разрабатываемом программном продукте.

* Проектирование приложение для управления задачами.

Заранее определённые шаблоны, структуры необходимые для создания программного средства, а также заранее прописанные принципы их взаимодействия. Позволят значительно сократить время разработки и улучшить качество приложения. Данный пункт является неотъемлемой частью создания приложения, которое будет использовать определённый архитектурный стиль или некоторые протоколы взаимодействия.

* Реализация приложения.

На основе описанных в предыдущем пункте взаимодействий и принципов будут использоваться определённые технические решения. Например, использование архитектурного стиля, который был бы оптимальным для реализации взаимодействия.

* Разработка руководства пользователя.

После того ка приложение реализовано необходимо разработать руководство, после которого пользователь, впервые познакомившись с нашим программным продуктом смог в полной мере воспользоваться потенциалом.

# **1 Обзор аналогов**

# **1.1 Rewind**

Rewind – Это приложение работает непрерывно в фоновом режиме и записывает все ваши телефонные слышит, что позволяет сохранить последнюю запись, когда вы слышите что-то вы хотели бы сохранить. Перемотка позволяет пассивно записывать свое окружение. Приложение распространяется бесплатно и не имеет рекламы.

Приложение предоставлено скриншоте ниже (рис 1.1.1).

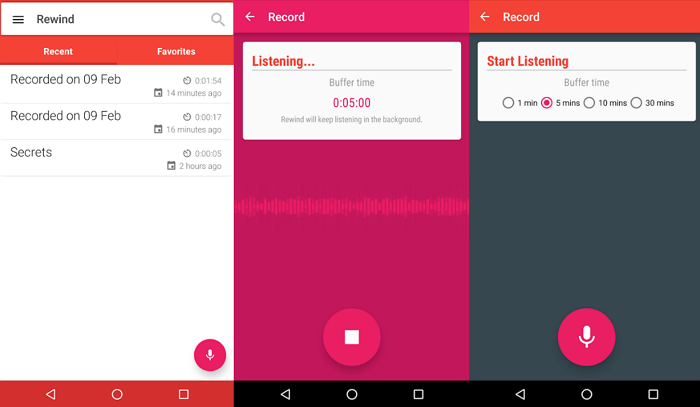


Рисунок 1.1.1 – Приложение «Rewind»

В процессе изучения продукта были выявлены следующие плюсы:

* Поддерживает несколько форматов для записи.
* Возможность записывать разговоры.
* Приложение можно настроить на сохранение только последних 30 минут аудиозаписей.
* Приложение не использует всю память смартфона, оно все же требует значительных ресурсов для работы в фоновом режиме.
* Простота в управлении.

К найденным минусам данного приложения можно отнести:

* Работоспособно далеко не на всех смартфонах.
* Быстро садится батарея в фоновом режиме.

# **1.2 Easy Voice Recorder**

Slack – web-сервис (также доступны приложения для десктопа, iOS и Android) для создания чатов и отдельных групп для ведения обсуждений с командой, клиентами или пользователями.

Скриншот приложения представлен ниже (рис 1.2.1).

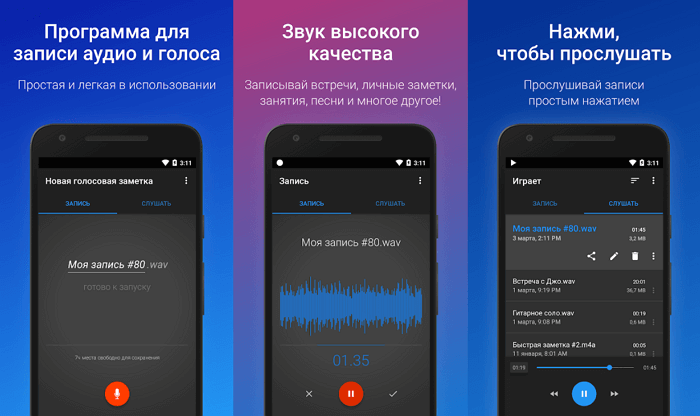


Рисунок 1.2.1 – Приложение Easy Voice Recorder

В приложении были выявлены следующие плюсы:

* Автоматическая загрузка новых записей в Google Drive или Dropbox.
* Запись в MP3 и AAC, дополняет форматы бесплатной версии.
* Поддержка записи с Bluetooth устройства.
* Обрезка и удаление ненужных частей записи в режиме «Редактирования».
* Поддержка папок и сохранение записей на SD-карте.

- Управление диктофоном с помощью панели уведомлений или плагинов для Tasker и Locale.

Так же следующие минусы:

* Нет возможности катализировать интерфейс вызова приложения.
* Нет возможности отключить уведомления.

# **1.3 Диаграмма «GanttPro»**

Приложение, которое включает в себя многофункциональный диктофон и проигрыватель. Отличается лаконичным интерфейсом и многими возможностями записывания разговоров. Размер записи ограничен только местом в вашем накопителе..

Приложение показано на скриншоте (рис 1.3.1):

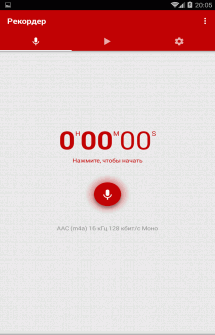
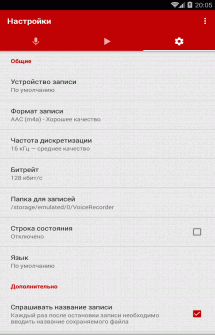
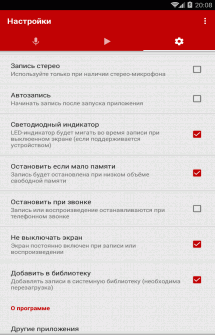
  

Рисунок 1.3.1 – Диаграмма «GanttPro»

Найденные плюсы:

* 4 Различные форматы аудио: MP3, высокое качество PCM (WAV), хорошее качество AAC (m4a/mp4) и AMR (3gp), чтобы сэкономить место
* Регулируемая частота дискретизации от качества телефонной (8 кГц) до качества CD (44 кГц)
* Возможность изменения битрейта от 32 до 320 кбит
* Поддержка стерео и монозаписи
* Live аудио спектр-анализатор
* Запись в фоновом режиме, даже когда экран выключен

Найденные минусами:

* Это приложение, не выделенный рекордер вызов и не может работать на некоторых устройствах.

# **1.4 Выводы по главе «Обзор аналогов»**

Так как в самих пунктах были рассмотрены основные плюсы и минусы таких проектов как «Rewind», «Easy Voice Recorder», «GanttPro». Остаётся остановится на самых интересных.

Общие плюсы: Поддержка записи в различных форматах, перенос данных в облачные хранилища, визуализация работы со звуковыми дорожками, фоновая работа приложения.

Дизайн, большинство приложении подобной тематики не отличаются хорошо подобранной гаммой цветов. Так же стоит отметить некоторые приложения обладают не интуитивно понятным интерфейсом, что значительно затрудняет знакомство с приложением, не говоря уже об работе с ним.

Минусы основным рассмотренных приложений является отсутствие реализации установления промежутка данных и возможности экстренного включения записи.

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

Для начала рассмотрим процесс входа в приложение. Use-case diagram будет содержать 2 актера «Незарегистрированный Пользователь» и «Зарегистрированный Пользователь». Незарегистрированный пользователь должен пройти процесс регистрации.

При входе в приложение зарегистрированному пользователю, будет предложена форма Авторизации

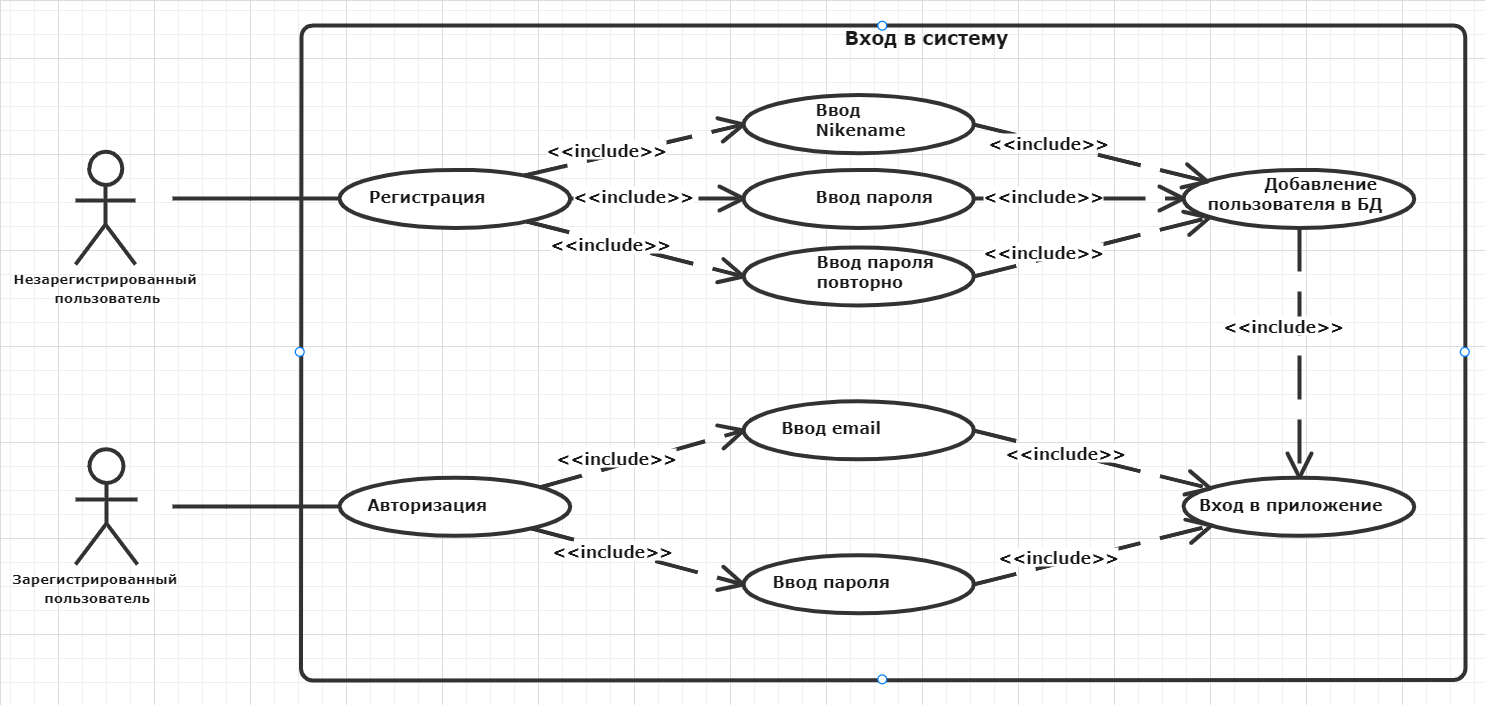


Рисунок 2.1.1 – UML-диаграмма вариантов использования входа в систему

Перед тем как зайти в приложение пользователь должен будет пройти процесс аутентификации. То есть активити на котором будет два поля (для ввода nickname и password). Так как в приложении подразумевается использование акаунтов, при помощи которых к определённому пользователю привязываются его данные, к которым он имеет отношение, что позволяет внести некий уровень защиты.

Основной структурой приложения является запись. Для отображения будет выводится списком, элементы списка будут их представлять (небольшая информация о записях). На этом же экране будет отображается кнопка для создания самого проекта.

При нажатии на кнопку Play будет производится запись до того момента, когда пользователь не остановит её или приложение не будет закрыто, формирование названия проекта будет формироваться на основе временных рамок.

Для вывода всех аудиодорожек, будет выделена отдельная активность. При переходе на неё пользователь сможет производить следующие действия:

* Проигрывать запись
* Удаление
* Останавливать проигрывание

Так же пользователю будут доступны следующие действия (активноти).

После нажатия на хранилище – активность для взаимодействия с облаком пользователя, на которой он сможет:

* Войти в облако
* Просмотреть весь список хранимых файлов
* Загрузить файл, в данном случае аудио
* Скачать аудиодорожку

После нажатия на «Писателя» – активность, позволяющая пользователю устанавливать время запуска записи и период его. После это пользователь сможет прослушать эту запись в хранилище.

Для визуального представления «потенциального» возможностей была нарисована следующая Use-case UML diagram:

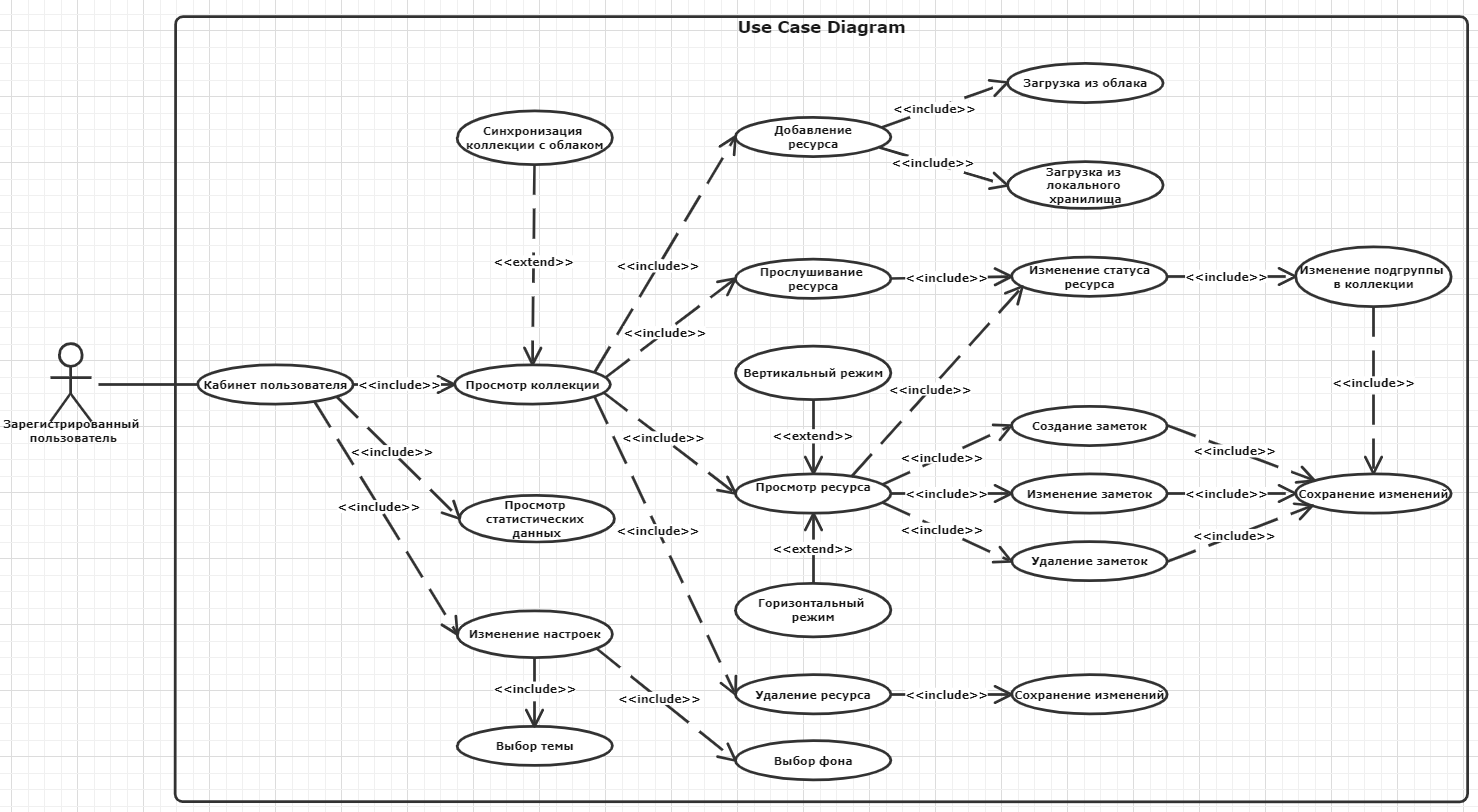


Рисунок 2.1.2 – Use-case UML diagram

При помощи её можно представить примерное представление о работе приложения.

# **2.2 Проектирование базы данных**

После того как мы знаем какие данные нам необходимо хранить и где можно разработать базу данных для наших нужд. Для данного приложения база данных носит тривиальный характер. Реализации может отличатся на разных платформах. Для данной реализации будем использовать всего 1-ть сущности представленных в виде аккаунта пользователя. Зависимости базы данных строились на принципах, описанных в данной литературе [3].

В таблице «Accounts» (для хранения данных для входа в приложение) содержится cстолбцы:

* [Id] - Первичный ключ
  + [Nickname] срока 20 (символов) обязательное,
  + [Password] срока 20 (символов) обязательное,

Схема базы данных представлена на рисунке 2.2.1.

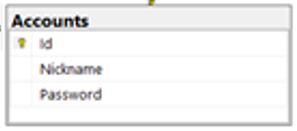


Рисунок 2.2.1 – Схема базы данных

Из схемы видно, что у одной задачи могут быть несколько пользователей, так же, как и у проекта может быть несколько задач (связь один ко многим).

# **2.3 Выводы по главе «Проектирование программного продукта»**

Подведём итоги главы. Наше приложение будет состоять из «Мобильной» части.

Что касается её, был установлен взаимодействие пользователя с приложением, а также принцип построения дизайна, направленного на пользователя.

Общая часть нашего приложения представляет собой база данных, и файлы, хранящиеся на устройстве пользователя. В одном из подпунктов данной главы были определены таблицы, поля, для каждого из которых были выбраны оптимальные типы данных. Так же была представлена схема.

# **3 Программная реализация приложений**

# **3.1 Технические средства для разработки**

При создании курсового проекта использовались следующие технологии:

* Java – строго типизированный объектно-ориентированный язык программирования.

• Flutter - SDK с открытым исходным кодом для создания мобильных приложений от компании Google. Он используется для разработки приложений под Android и iOS.

* SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.
* SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.
* Visual Studio Code - Редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.
* Android Studio – интегрированная среда разработки для работы с платформой Android.
* Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.
* OkHttp – библиотека предоставляющая собой простой и удобный клиент для работы с HTTP запросами.
* DB Browser - это высококачественный, визуальный инструмент с открытым исходным кодом для создания, проектирования и редактирования файлов баз данных, совместимых с SQLite.

# **3.2 Разработка базы данных**

Реализации базы данных на мобильном устройстве воспользуемся SQLite. Так же в целях безопасности будем использовать библиотеку SQLCipher, что позволит защитить данные на устройстве, что значительно усложнит задачу злоумышленникам.

Код для создание мобильной базы данных представлен в Приложении А.

# **3.3 Разработка мобильного приложения**

Для реализации мобильного приложения использовалось такая среда разработки как – Android Studio. Мобильное приложение будет состоять из двух частей «Визуальная» написанная на языке Dart используя при этом Flutter.И «Нативная», к которой в свою очередь будет обращатся визуальная для отображения данных или используя некие программные возможности недоступные ей

Графическая структура проекта представлена на рисунке 3.3.1.

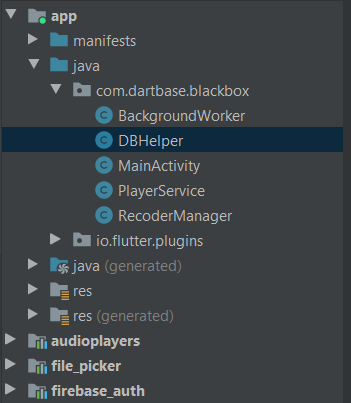


Рисунок 3.3.1 – Графическая структура проекта «Визуальная»

Нативную часть взаимодействует с базой данных. Для этого используется принцип, описанный в источнике [2]. То есть такие действия как создание, удаление, установка значений по умолчанию и обновление базы данных будут реализованы в отдельном классе DBHelper, так же при инициализации или установления связи с базой, данный класс будет использоваться по принципу Singleton. Для получения контекста данных.

|  |  |
| --- | --- |
| DbHelper – представляет собой контекст базы данных | |
| onCreate | Метод создания базы данных при ее отсутствии |
| onUpgrade | Метод вызываемый при смени версии базы данных |
| onConfigure | Метод вызываемый для конфигурации базы данных |
| getReadableDatabase | Метод открывающий соединение с базой данных с помощью пароля, только для чтения |
| getWritableDatabase | Метод открывающий соединение с базой данных с помощью пароля, только для чтения и записи |

**Таблица 3.3.1 – Содержание пакета DBHelper**

Всё операции с базой данных вынесены в отдельный класс.

|  |  |
| --- | --- |
| CRUD – основные операции с базой данных | |
| selectAccounts | Метод для получения всех значений с локальной базы |
| InsertAccount | Метод для добавления пользователя |

**Таблица 3.3.2 – Содержание пакета DBHelper**

В представление сущности, которые находятся в базе данных на сервере.

|  |  |
| --- | --- |
| Account – описание объекта | |
| id | Свойство содержащее идентификатор |
| nickname | Свойство содержащее псевдоним пользователя |
| Password | Свойство содержащее пароль пользователя |

**Таблица 3.3.3 – Содержание пакета DBHelper**

Так В данном проекте используются службы, работающие в фоновом режиме Такие как PlayerService используется для отлавливания действий связанных с медиа. Здесь же он реализован как сервис следящий за значениями физ. кнопок.

|  |  |
| --- | --- |
| PlayerService – сервис для работы с медиа | |
| Конструктор | Метод для установки ресурса |
| onCreate | Метод создания класса |
| onAdjustVolume | Метод для отслеживания значений при нажатии на кнопку |
| onBind | Метод внутреннего инкремента |
| VolumeProviderCompat | Метод устанавливающий обработчик на нажатий |
| onDestroy | Метод уничтожающий сервис |

**Таблица 3.3.4 – Содержание пакета PlayerService**

Следующий класс реализует методы для фоновой работы в определённый период.

|  |  |
| --- | --- |
| BackgroundWorker | |
| Конструктор | Метод для установки ресурса |
| doWork | Основной метод класса при выполнении |
| showNotification | Метод для отображения работы (фоновой) |

**Таблица 3.3.5 – Содержание пакета BackgroundWorker**

Следующий класс реализует сингтон модель, доступ к которой осуществляется через службы записи аудио.

|  |  |
| --- | --- |
| RecoderManager – управление медиа | |
| recordStart | Метод начало записи |
| recordStop | Метод для отслеживания значений при нажатии на кнопку |
| releaseRecorder | Метод используется на создания экземпляра класса |
| releasePlayer | Метод освобождения ресурсов |
| onDestroy | Метод уничтожающий объект |

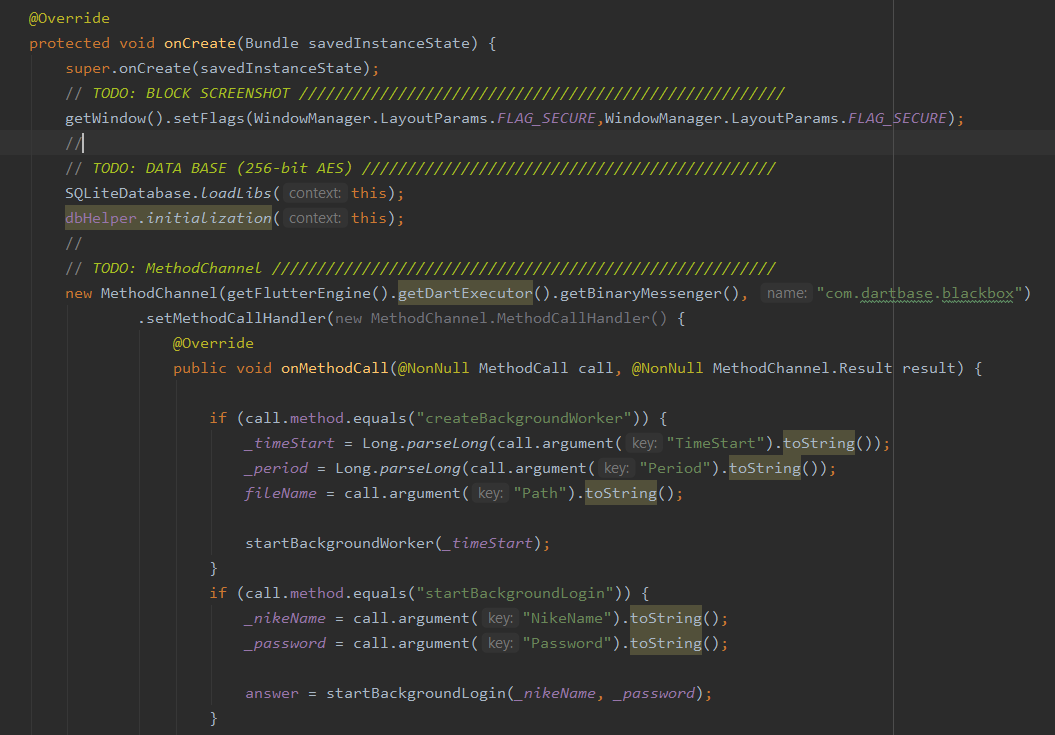
**Продолжение таблицы 3.3.6 – Содержание пакета RecoderManager**

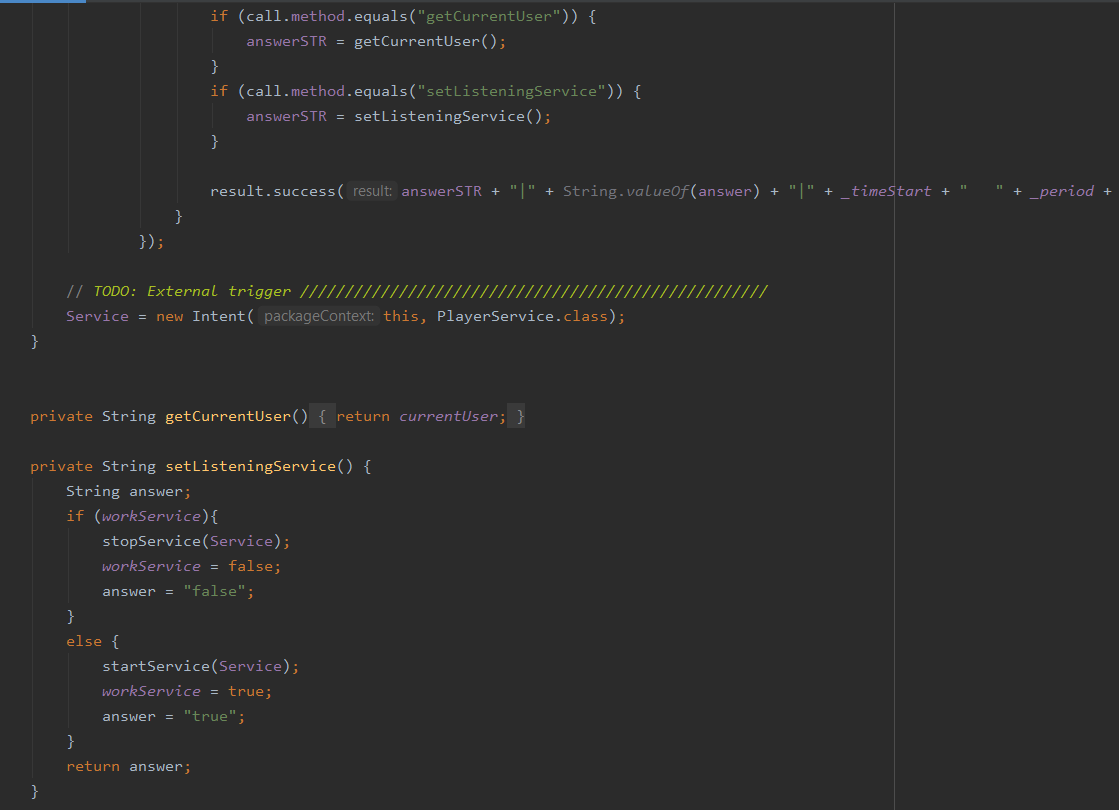
Главная активность — это точка входа для визуальной часть, для получения услуг от нативной.

|  |  |
| --- | --- |
| MainActivity – работа с датой и временем | |
| onCreate | Метод для инициализации основных методов и |
| MethodChannel | Метод обработчик для визуальной части приложения  (выбор метода для его вызова) |
| getCurrentUser | Метод возвращения текущего пользователя для UI |
| setListeningService | Метод для активации сервиса прослушки клавиш |
| startBackgroundWorker | Метод запуск сервиса для фонового прослушивания |
| startBackgroundLogin | Метод внесение пользователя в бузу данных |

**Продолжение таблицы 3.3.7 – Содержание пакета MainActivity**

Основная часть MainActivity представлена на следующем скриншоте:





Для визуального представления приведена диаграмма классов

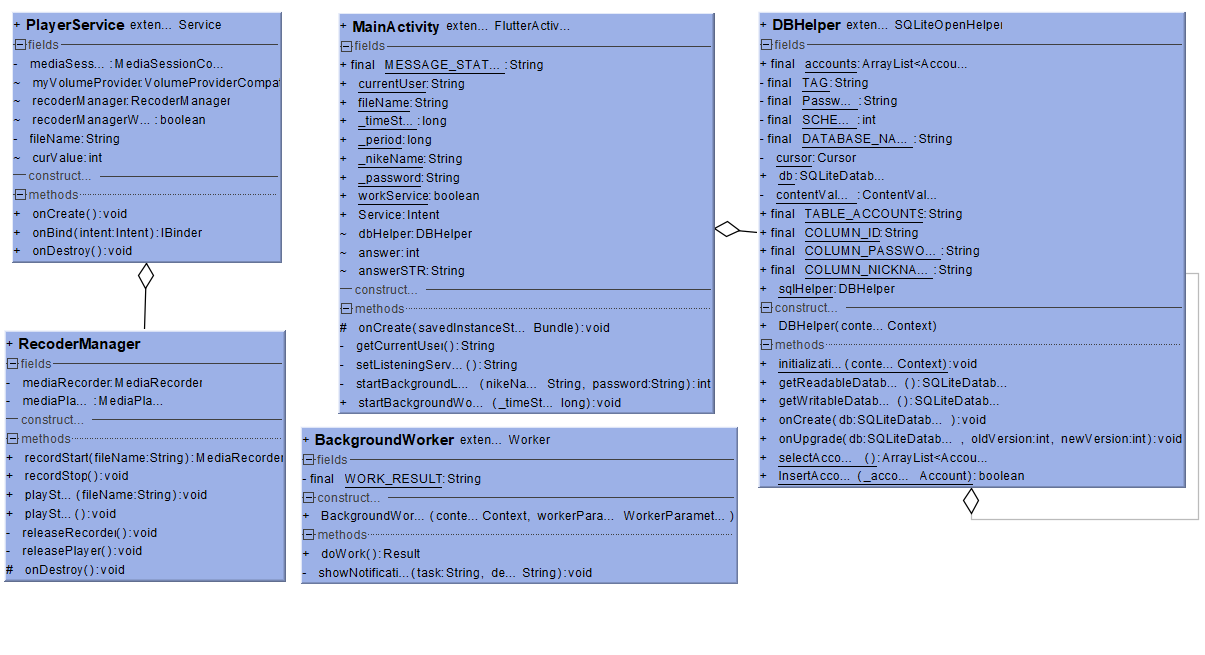


Рисунок 3.3.8 – Диаграмма классов

# **3.4 Разработка визуальной части приложения**

Визуальная часть приложения писалась при помощи SDK Flutter. Далее будут рассмотрены основные функциональные элементы визуальной части приложения. Остальная же часть UI представлена здесь [2].

Графическая структура визуальной части приложения проекта представлена на рисунке 3.4.1.

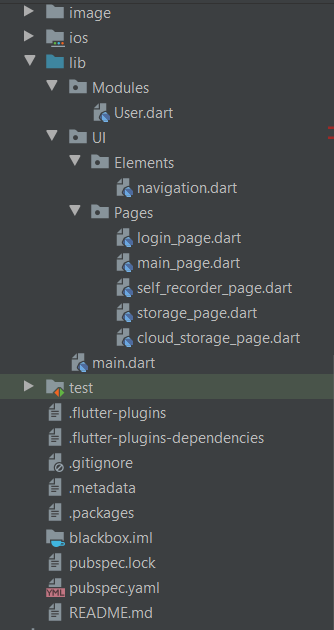


Рисунок 3.4.1 – Графическая структура проекта

Основное взаимодействие строится на основе контроллеров позволяющие обрабатывать запросы. При помощи виджита BottomNavigator реализовано перемещение между функционалом данного приложения. Его структура представлена на следующем.

class \_BottomNavigatorState extends State<BottomNavigator> {  
  
 int \_selectedIndex = 0;  
  
 final List<Widget> \_pages = [  
 MainPage(),  
 SelfRecorderPage(),  
 CloudStoragePage(),  
 StoragePage()  
 ];  
  
 void \_onItemTapped(int index) {  
 setState(() {  
 \_selectedIndex = index;  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return new Scaffold(  
 body: \_pages[\_selectedIndex],  
 bottomNavigationBar: BottomNavigationBar(  
 type: BottomNavigationBarType.fixed,  
 backgroundColor: Colors.*black*,  
 selectedItemColor: Colors.*indigoAccent*[400],  
 unselectedItemColor: Colors.*grey*.withOpacity(.60),  
 selectedFontSize: 14,  
 unselectedFontSize: 14,  
 onTap: \_onItemTapped,  
 currentIndex: \_selectedIndex,  
 items: [  
 BottomNavigationBarItem(  
 title: Text('Main'),  
 icon: Icon(Icons.*mic*),  
 ),  
 BottomNavigationBarItem(  
 title: Text('Recorder'),  
 icon: Icon(Icons.*date\_range*),  
 ),  
 BottomNavigationBarItem(  
 title: Text('Cloud'),  
 icon: Icon(Icons.*cloud\_queue*),  
 ),  
 BottomNavigationBarItem(  
 title: Text('Storage'),  
 icon: Icon(Icons.*library\_books*),  
 ),  
 ],  
 ),  
 );  
 }  
}

За взаимодействия с нативной частью отвечает метод

Пример его использования представлен ниже

Future<void> \_login() async {  
 var methodChannel = MethodChannel("com.dartbase.blackbox");  
 String data = await methodChannel.invokeMethod(  
 "startBackgroundLogin",  
 <String, dynamic>{  
 'NikeName': \_nikeName,  
 'Password': \_password  
 });  
 debugPrint(data);  
  
 for (int i = 0; i < data.length; i++)  
 {  
 if(data[i] == '|'){  
 CodeReturn = data[i+1];  
 break;  
 }  
 }

Также можно ознакомиться с основными классами вызова можно в приложении B.

# **3.5 Программная реализация защиты в приложении**

При хранении данных приложения, в нашем случае базы данных использовался SQLCipher

SQLCipher - это расширение SQLite с открытым исходным кодом, которое обеспечивает прозрачное 256-битное AES-шифрование файлов базы данных. Симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Пример преобразования представлен ниже

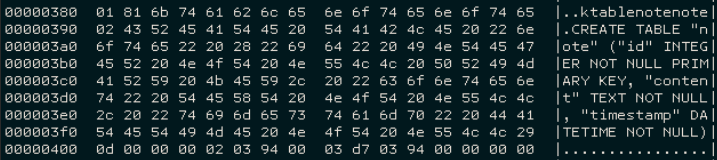


Рисунок 3.5.1 – БД SQLite

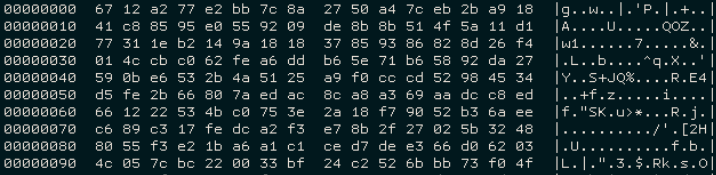


Рисунок 3.5.2 – БД с SQLCipher

import net.sqlcipher.Cursor;  
import net.sqlcipher.database.SQLiteDatabase;  
import net.sqlcipher.database.SQLiteOpenHelper;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class DBHelper extends SQLiteOpenHelper {

Выше представлены необходимые библиотеке для реализации SQLCipher.

Так же было программно заблокировано создании скриншота и записи видео

// *TODO: BLOCK SCREENSHOT //////////////////////////////////////////////////////*getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_SECURE*,WindowManager.LayoutParams.*FLAG\_SECURE*);  
//

Так же в приложении реализованы 2 системы авторизации

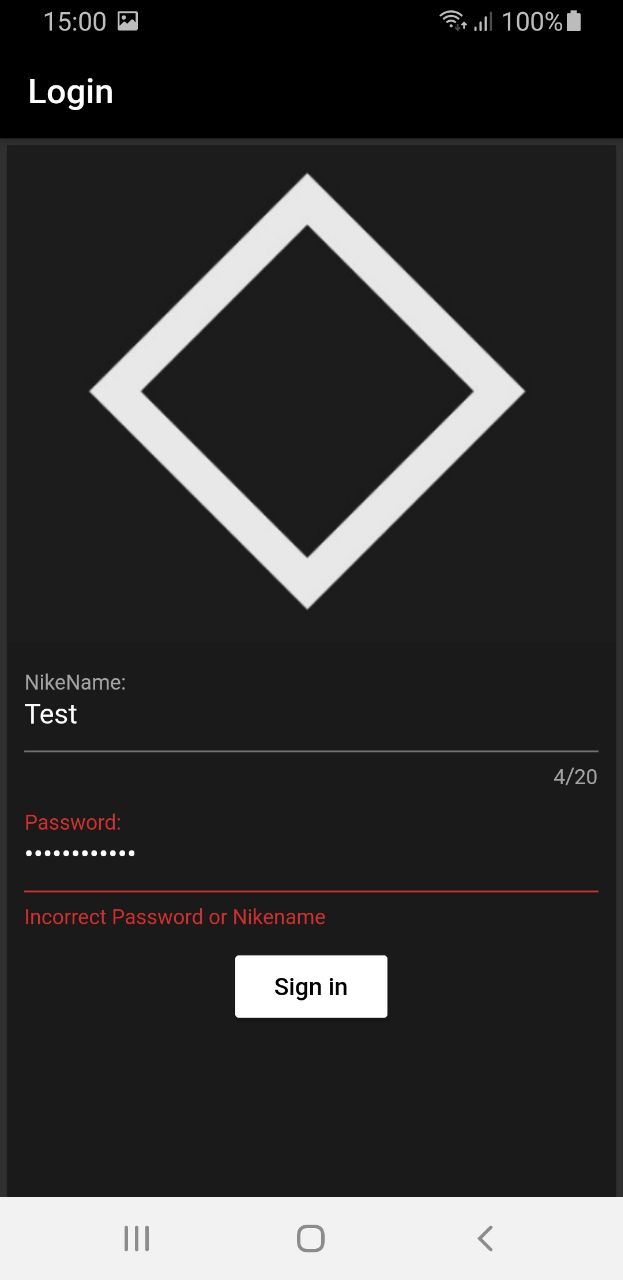
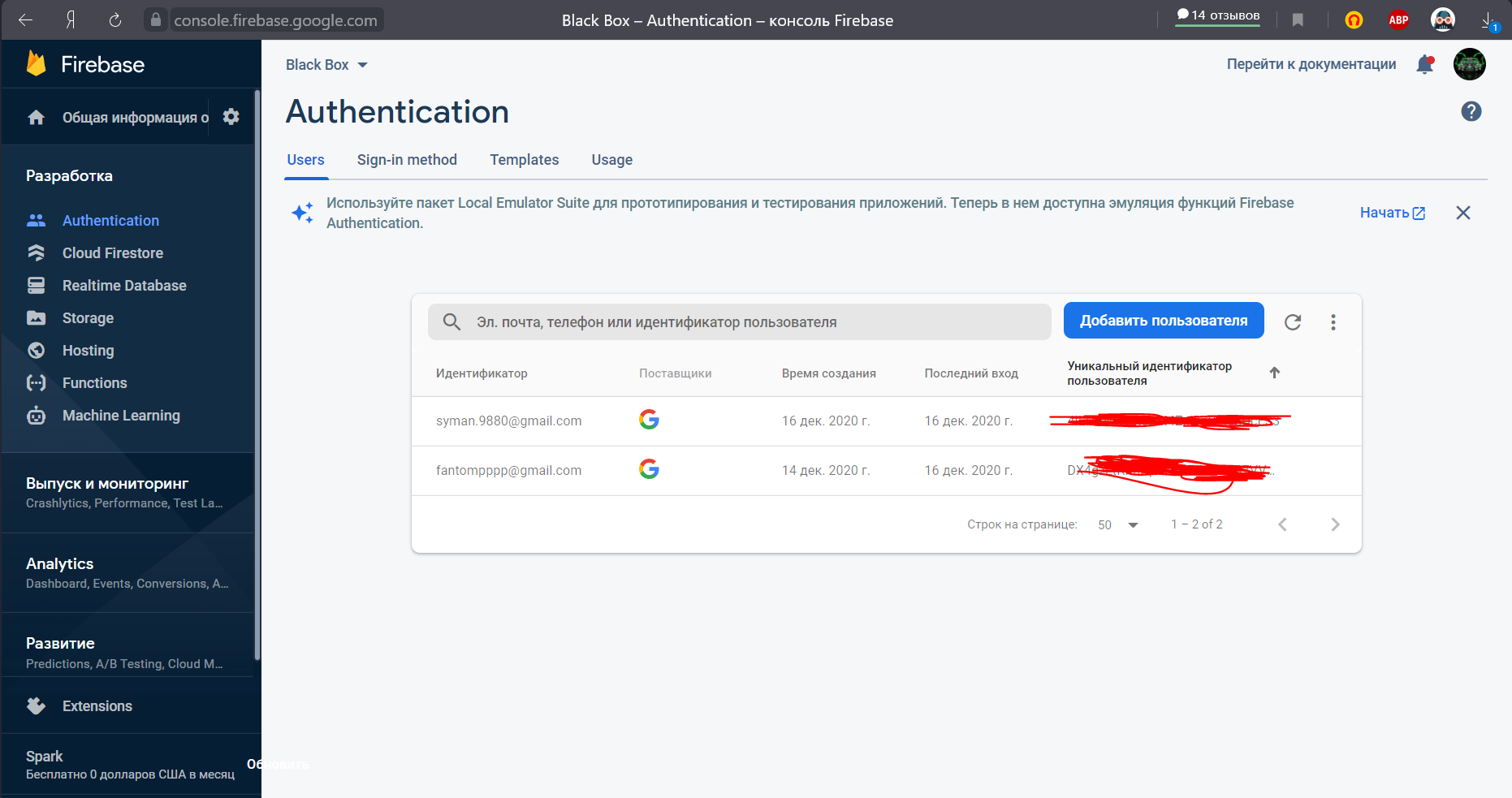


Рисунок 3.5.3 – вход в приложение

Первая в качестве входа через логин и пароль.



Вторая через сервисы Google для получения доступа к облачному хранилищу.

Так же проект, поддерживает обфускацию вернее при генерации (релизной сборки).

buildTypes {

release {

minifyEnabled true

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

С помощью анализатора APK можно увидеть преобразования в коде



# **3.6 Вывод по главе «Программная реализация приложений»**

В данной главе была рассмотрена реализация структуру базы данных.

Так же по мере разработки проекта были представлены отдельные компоненты приложения в виде таблиц. По мимо вызываемых методов в классах были описаны применяемые технологии.

Были перечислены технологии, которые использовались для разработки курсового проекта.

# **4 Руководство пользователя**

Black Box – приложение позволяющее вести звукозапись в обычном и фоновом режиме, создавать расписания включения звукозаписи в фоновом режиме, так же поддерживает, резервное копирование в облачное хранилище,

Шифрование БД и активацию звукозаписи при помощи комбинации клавиш

Необходимые требования для запуска приложения: Android min 6.0, API не ниже 24.

Соответствует ли ваш девайс данным требованиям, можно узнать зайдя в настройки. Раздел о телефоне.

При первом запуске приложения вас попросят зайти в свой аккаунт, так как ваши записи связаны именно с ним. Если у вас нет аккаунта, вам потребуется нажать на кнопку входа два раз (рис. 4.1).

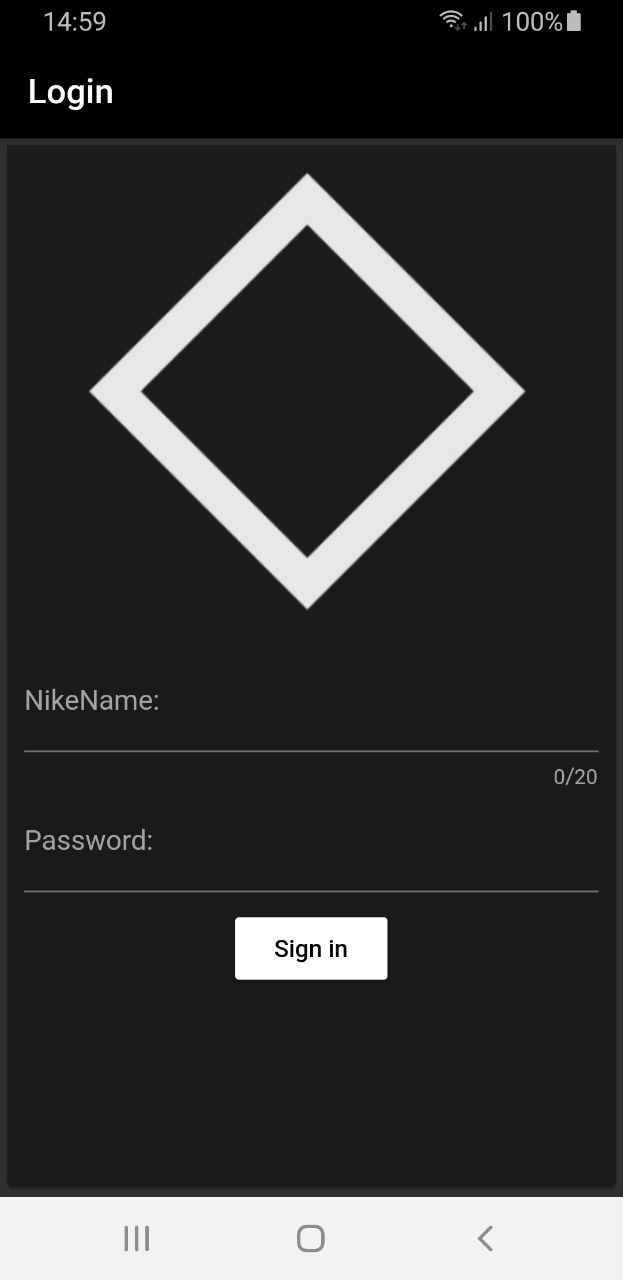
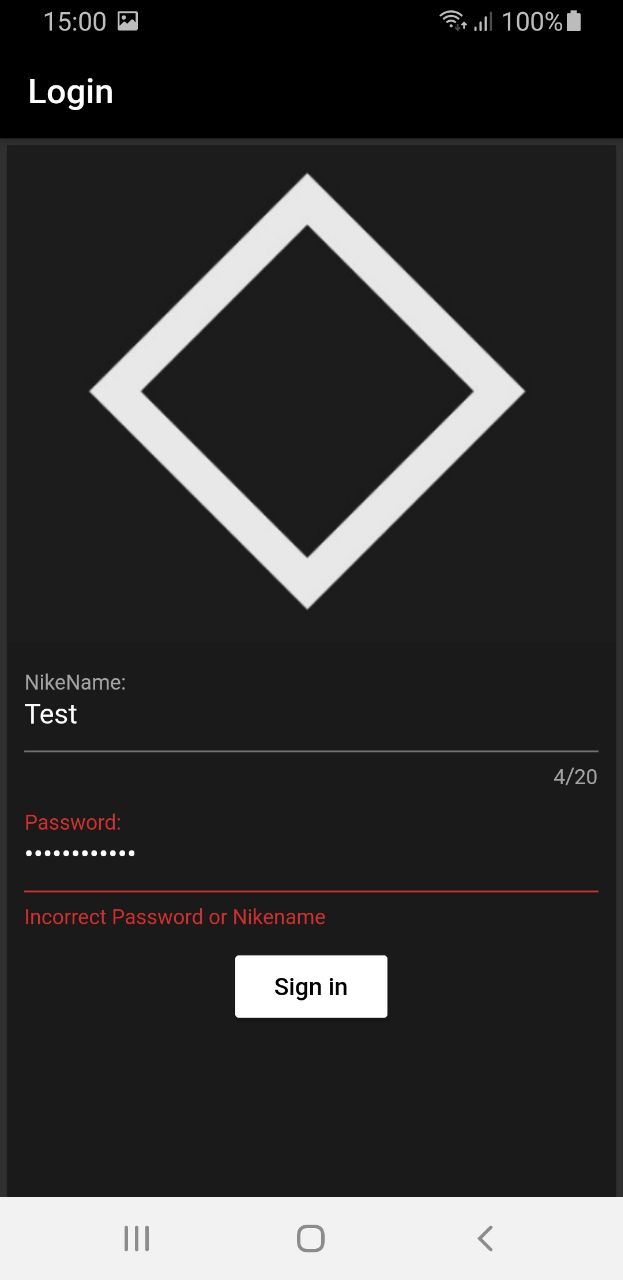
 

Рисунок 4.1 – Sign in

После авторизации приложения вы перейдёте на Главную страницу. В правом нижнем углу присутствует кнопка для начала записи, при нажатии на неё индикатор сверху будет соответствовать громкости звуков, которые он записывает (рис 4.2).

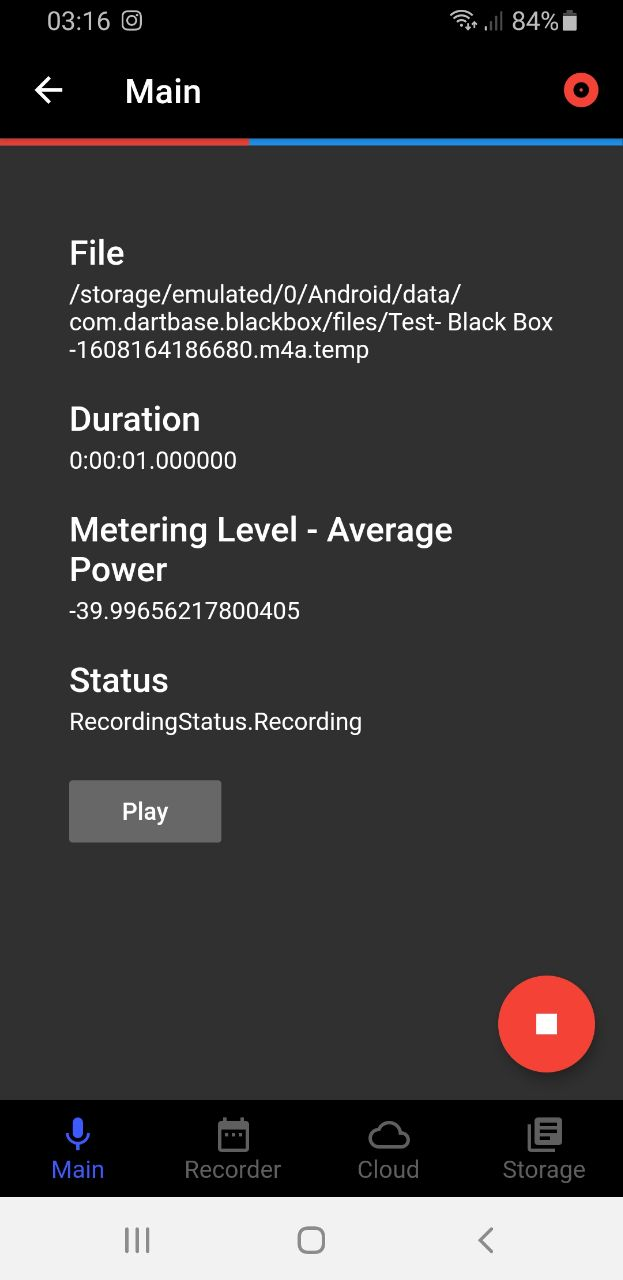
  

Рисунок 4.2 – «Main»

Перейдя на следующую вкладку, пользователь увидит следующую форму для установки Времени и периода записи в фоновом режиме (рис 4.3).

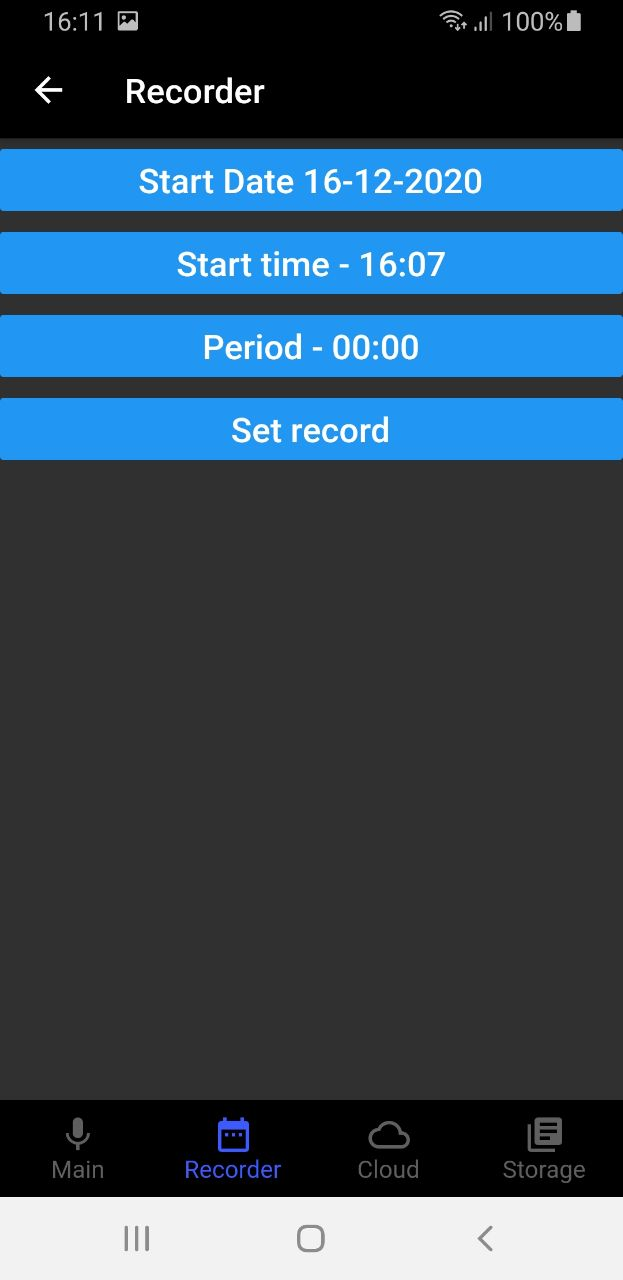


Рисунок 4.3 – «Recorder»

Работа с данной формой происходит следующим образом (рис 4.4)

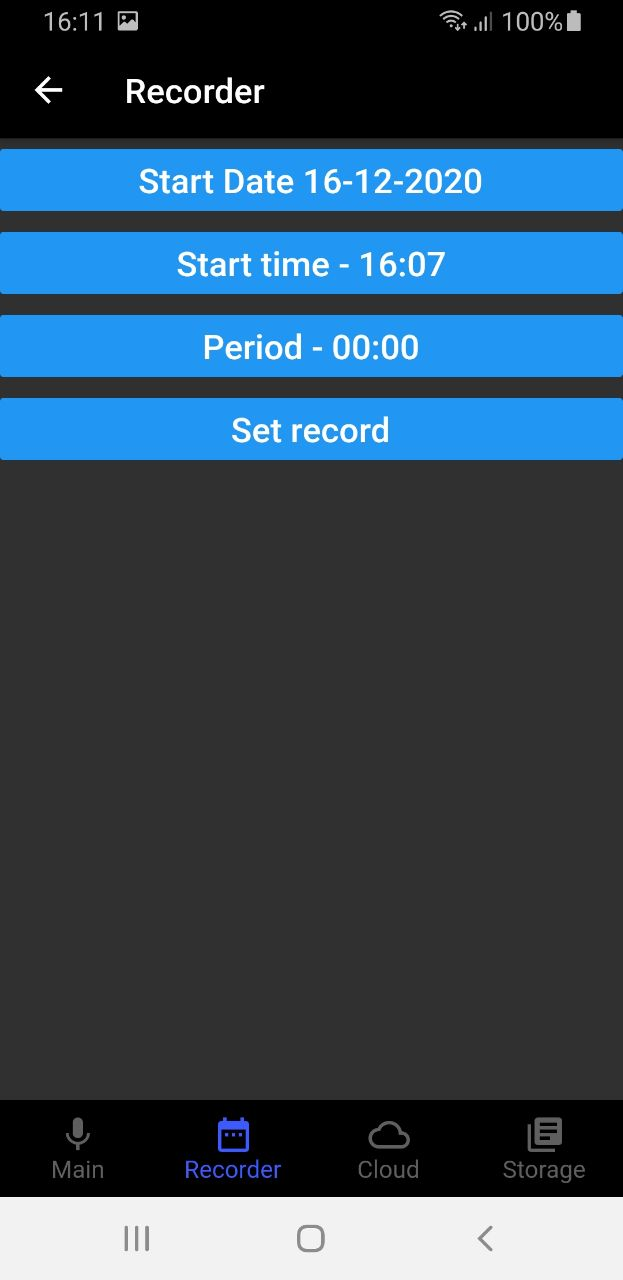
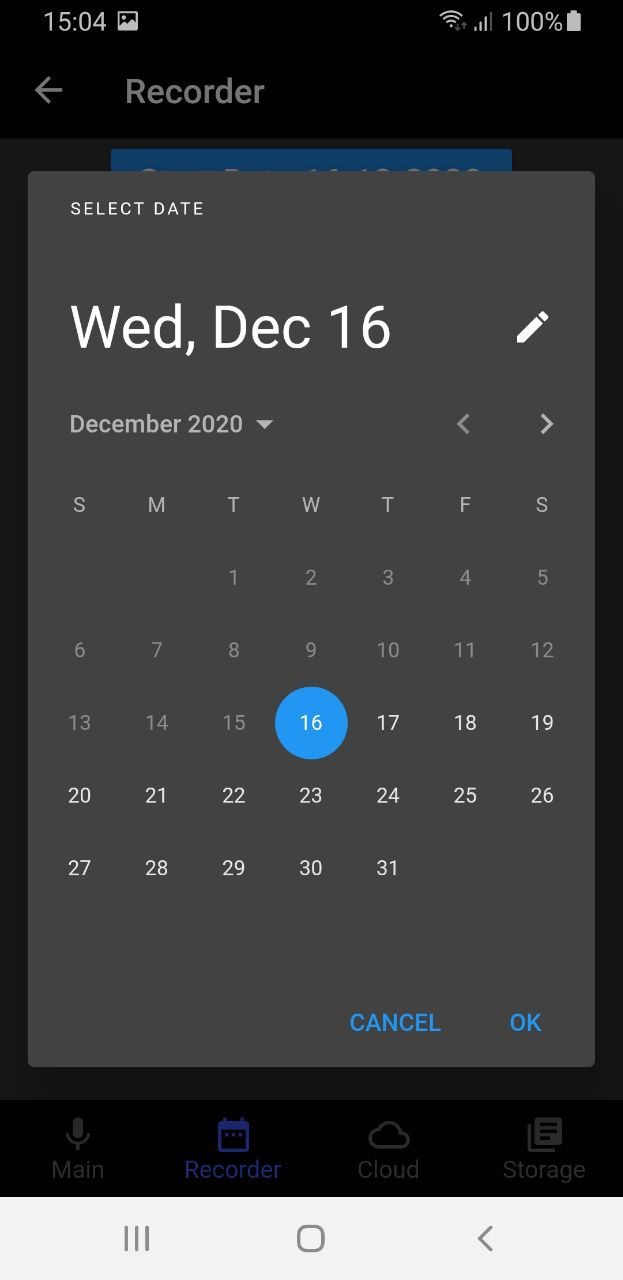
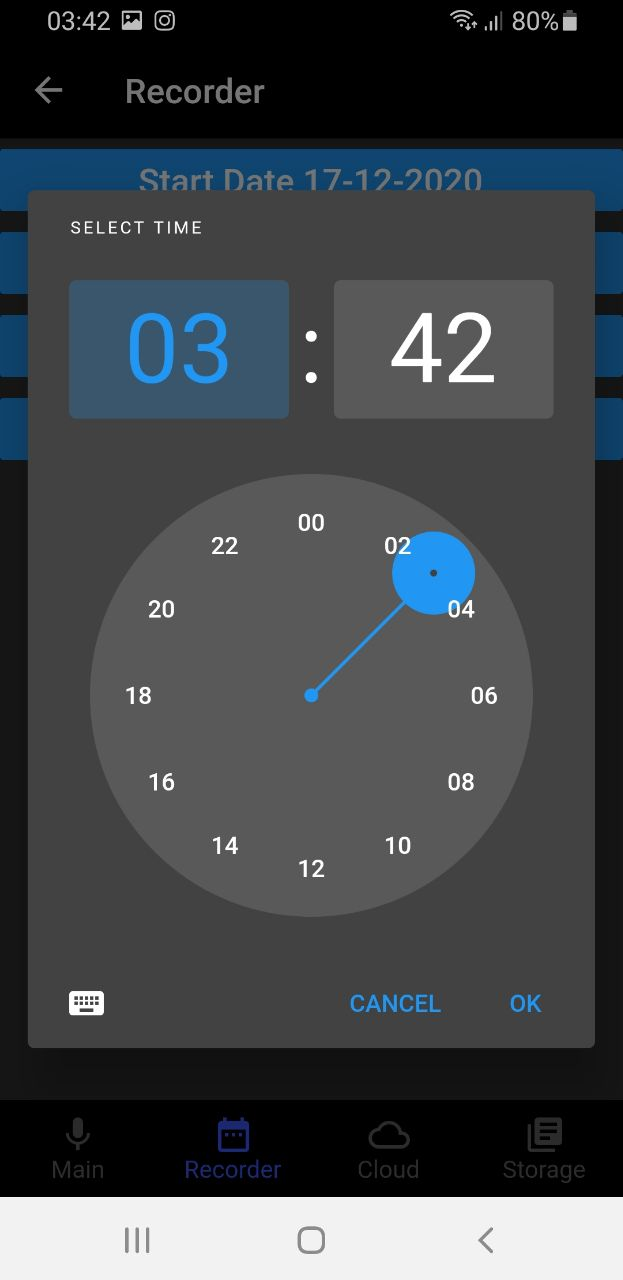
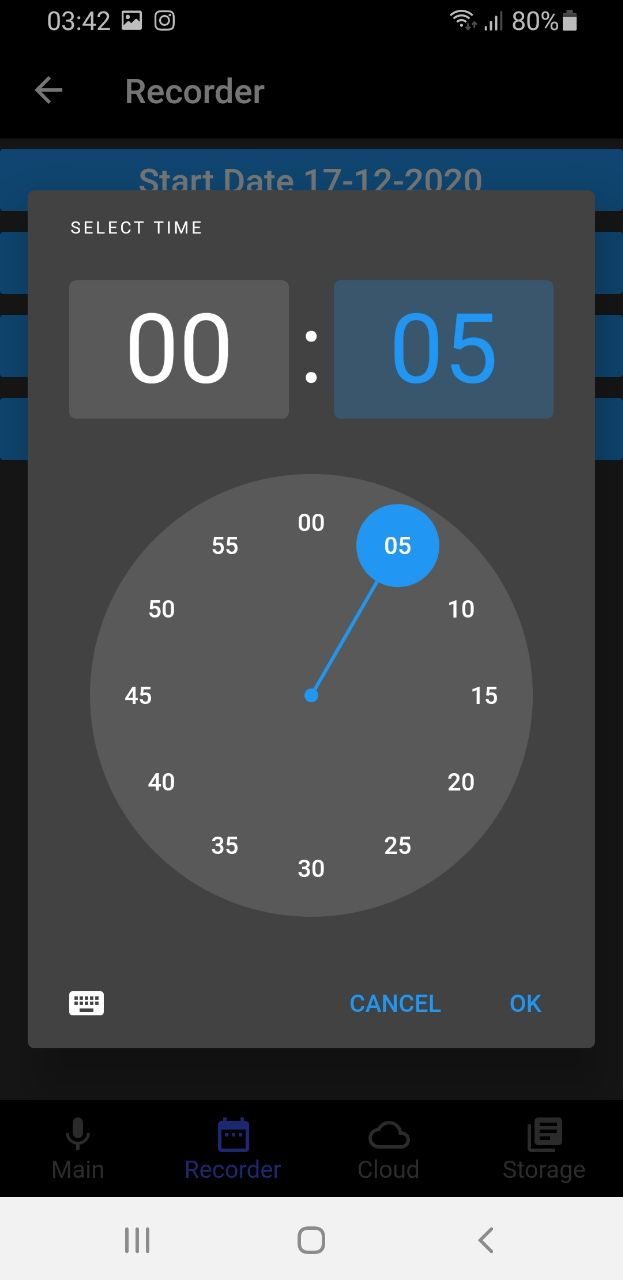
   

Рисунок 4.4 –установка «Recorder»

После заполнения формы такой формы и нажатия на кнопку «Set record», вы получите уведомление, которое сигнализирует о начале записи и её длительности в том момент времени, который вы указали на 3-м слайде (рис. 4.4).

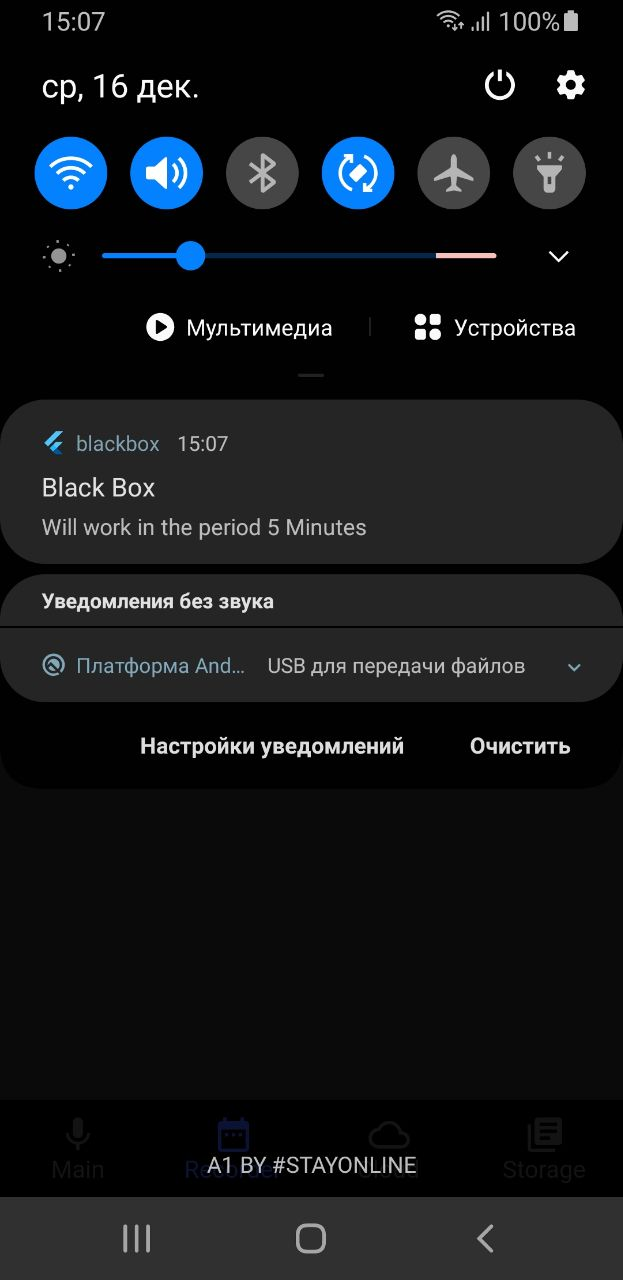


Рисунок 4.4 – Уведомление о начале записи

Следующий пункт меню представляет собой облачное хранилище. После прохождения авторизации и предоставления доступа вы сможете его использовать. (рис. 4.4).

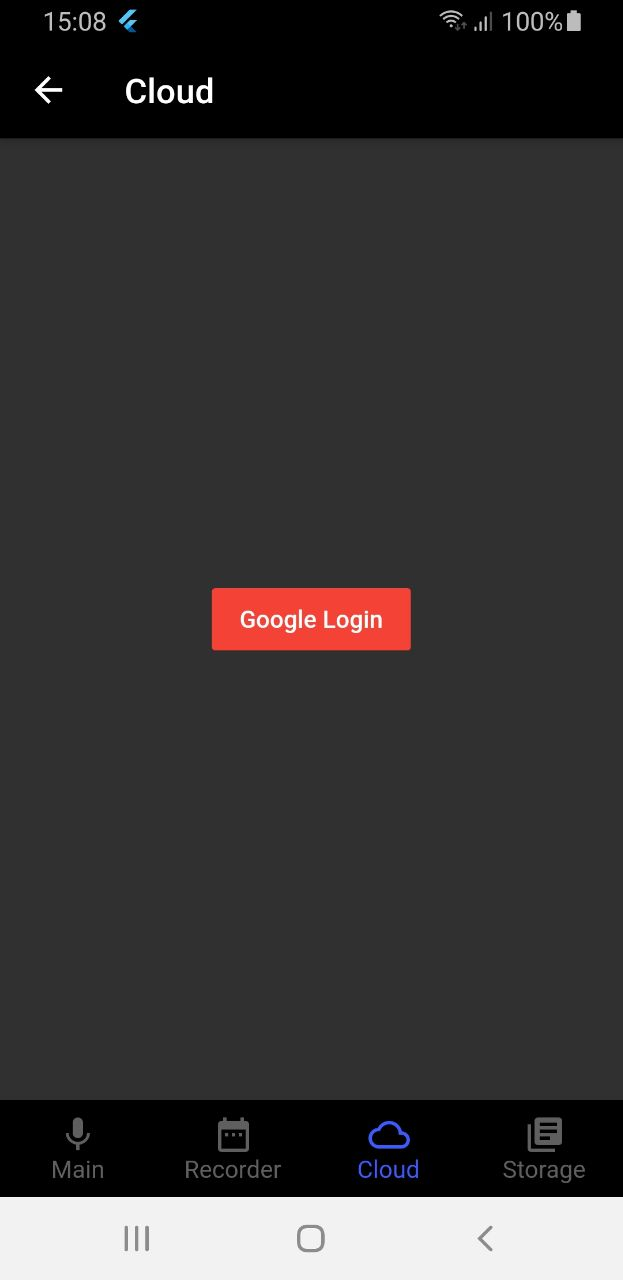
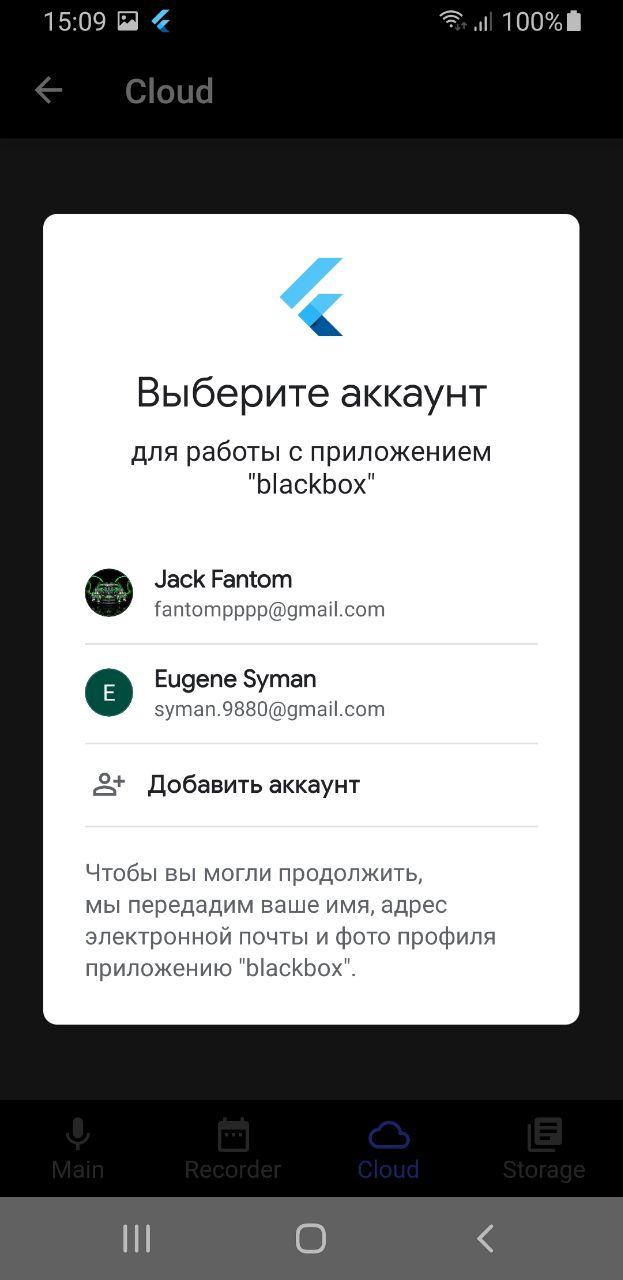
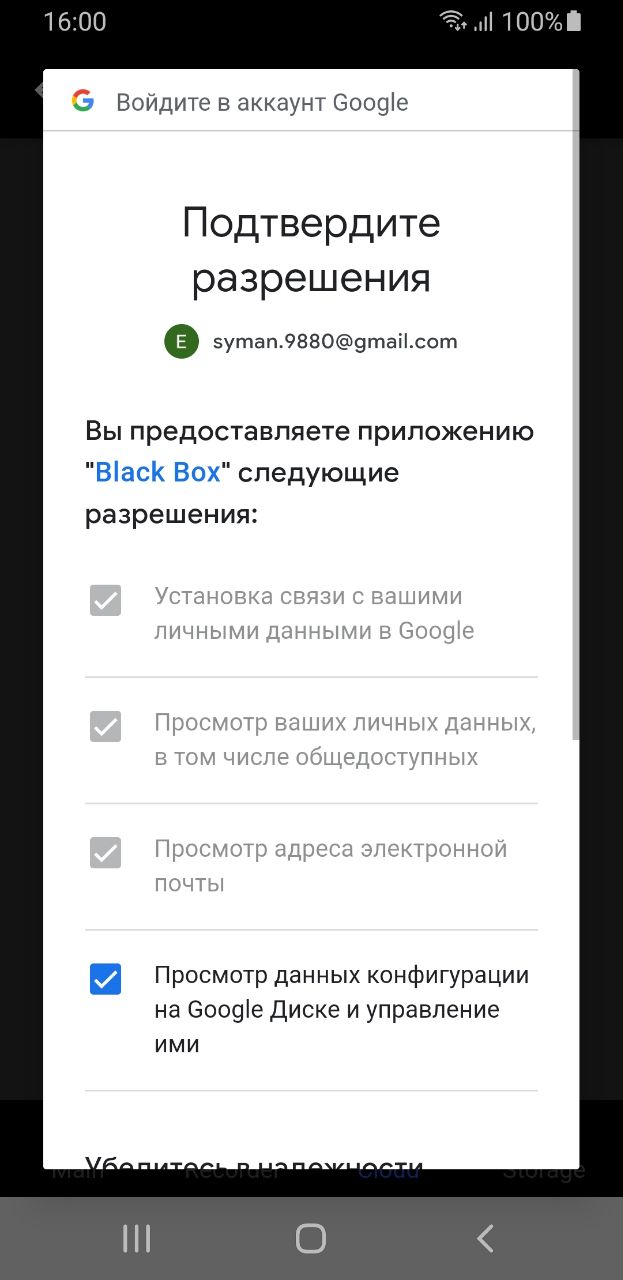
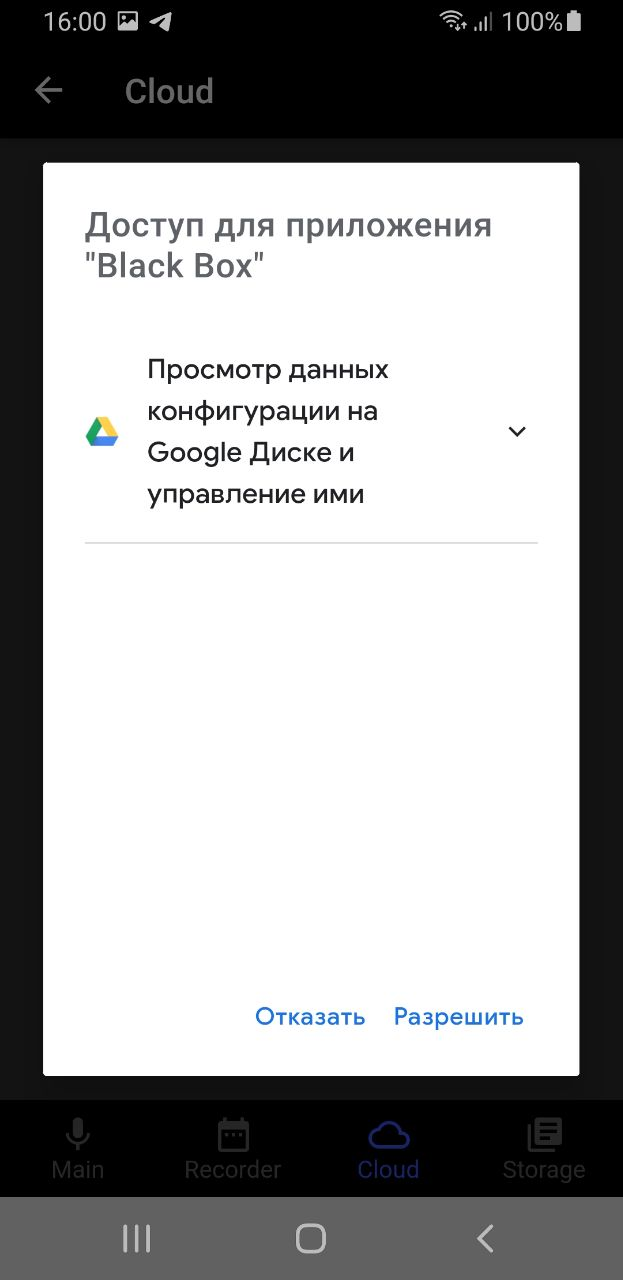
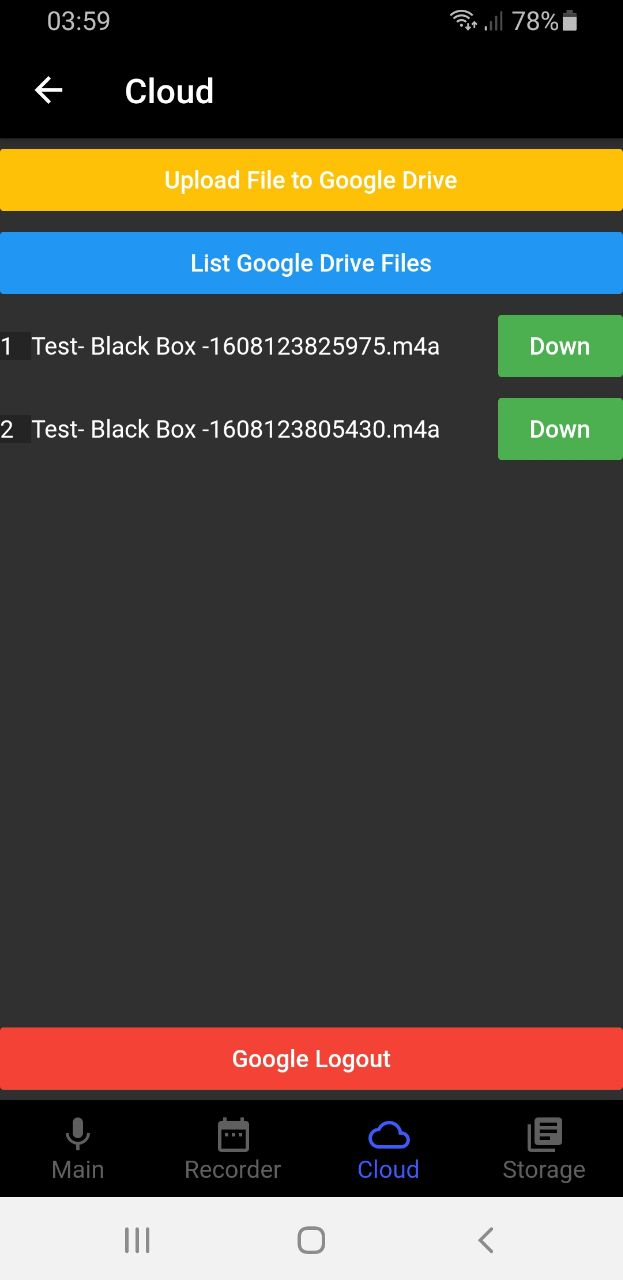
    

Рисунок 4.5 –«Cloud»

Далее «Storage» список в котором отображаются все аудиозаписи пользователя. Для действия на ними свапните элемент влево для воспроизведения, вправо для удаления.

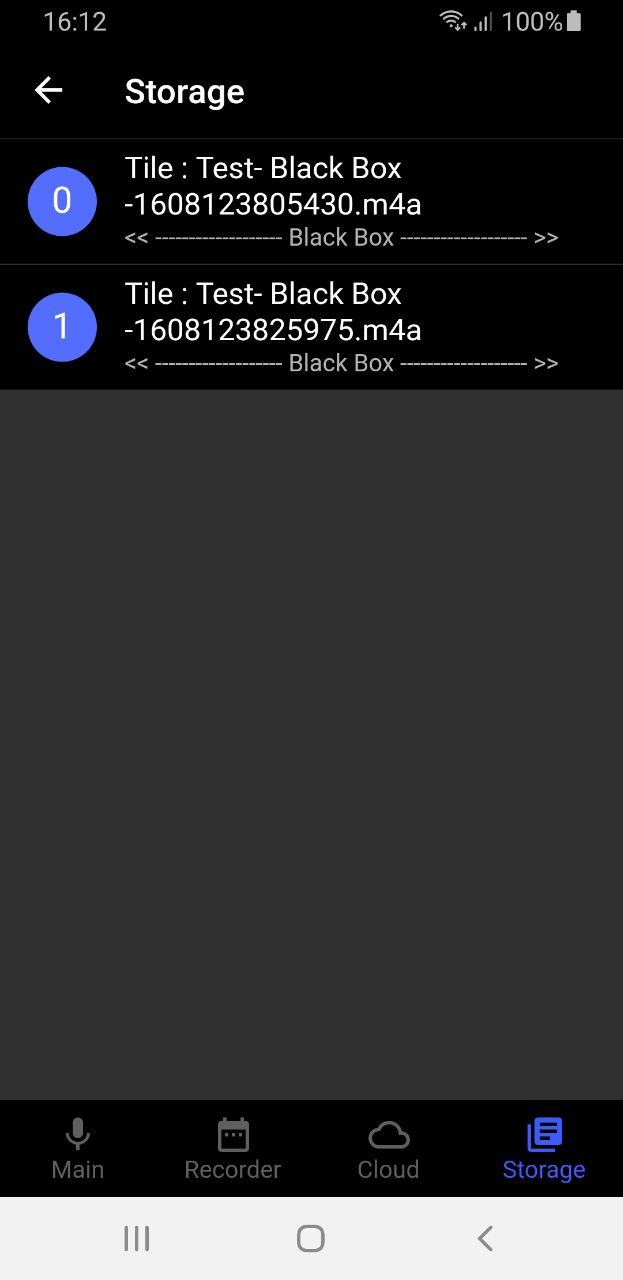
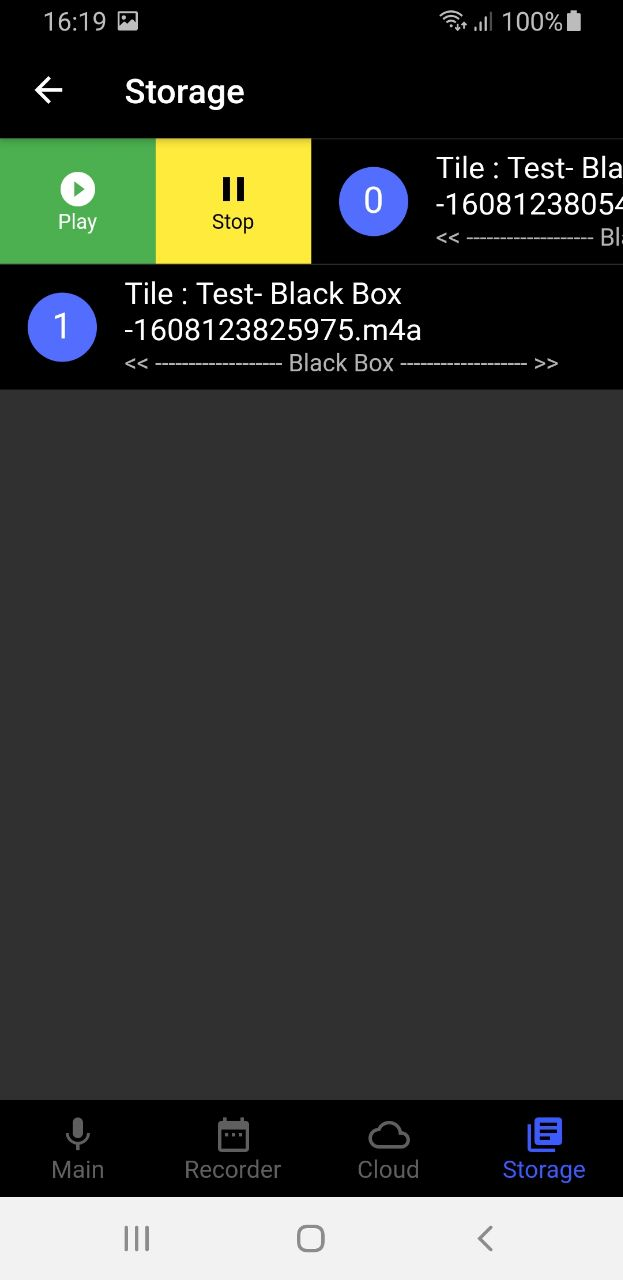
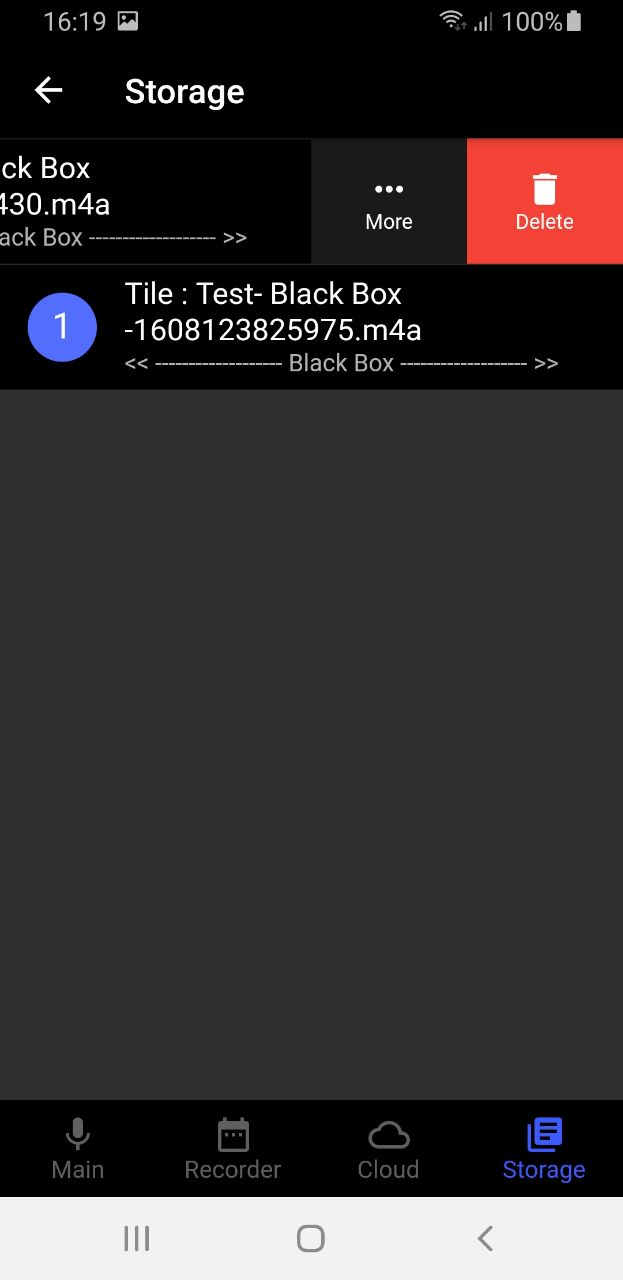
  

Рисунок 4.6 – «Storage»

Дополнительный функционал доступен пользователю после нажатия на кнопку в правом верхнем углу «Main». После нажатия на неё, все последующие нажатия пользователя на кнопки звука ( + , - ) будет обрабатывать само приложение.

При зажатии клавиши вверх, до упора (3 сек), начнётся запись. Которую можно остановить обратным действием, зажать кнопку вниз.

Данная комбинация для активации записи в фоновом режиме работает и при Блокировке экрана

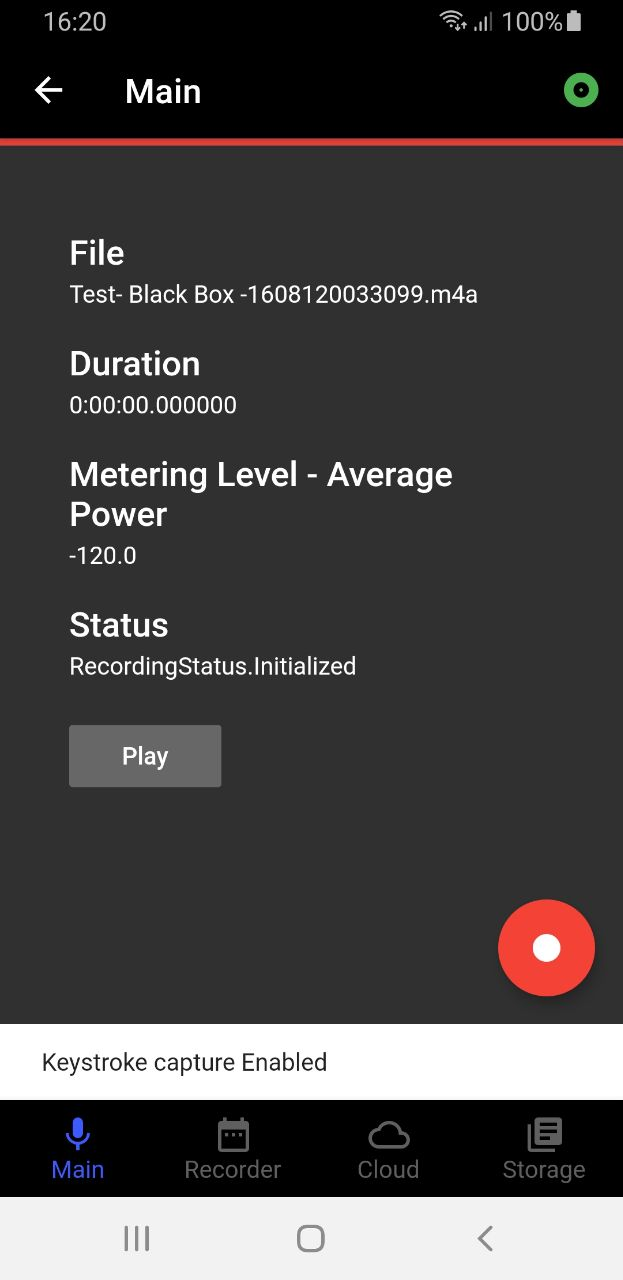
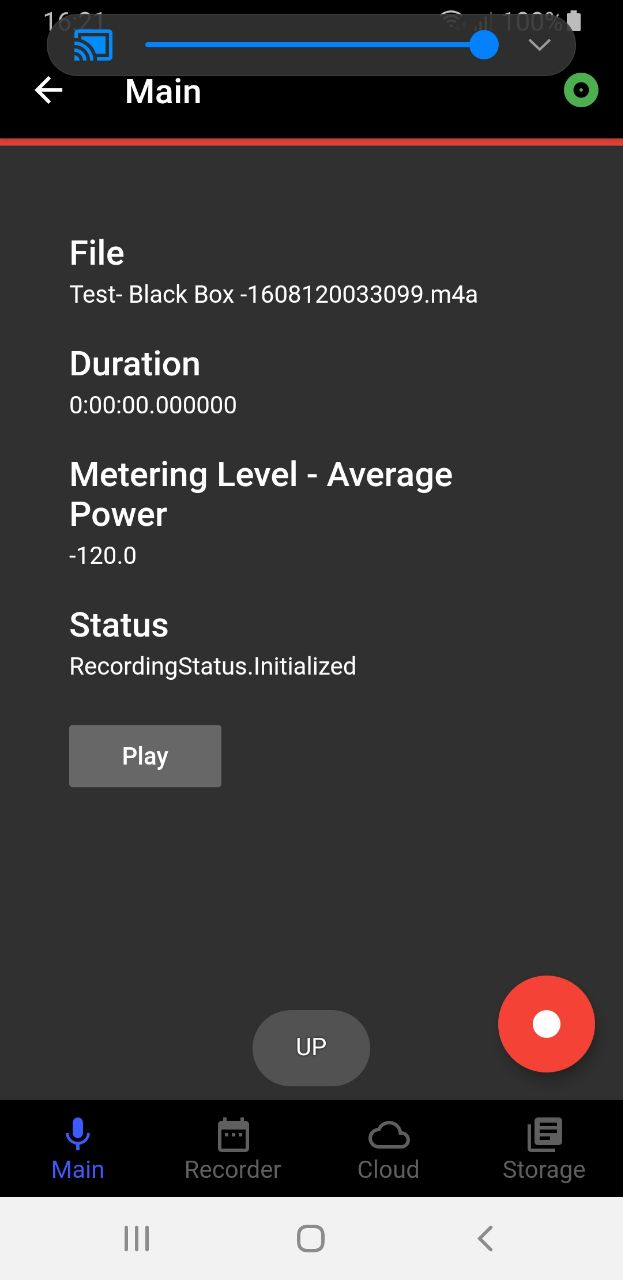
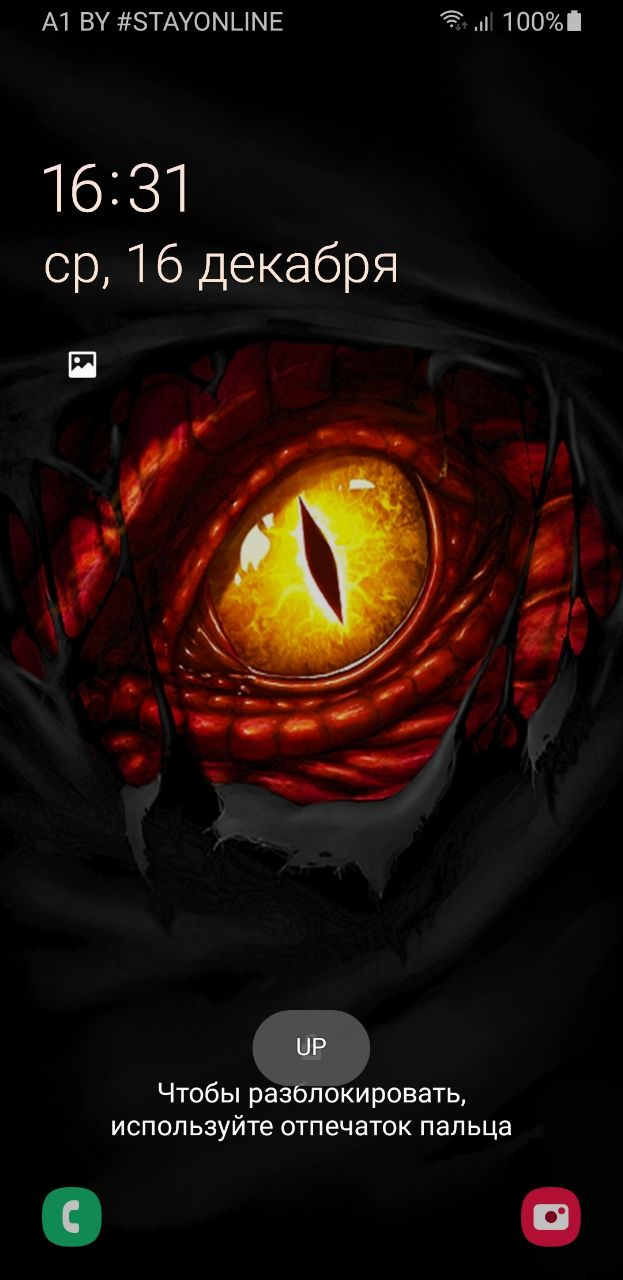
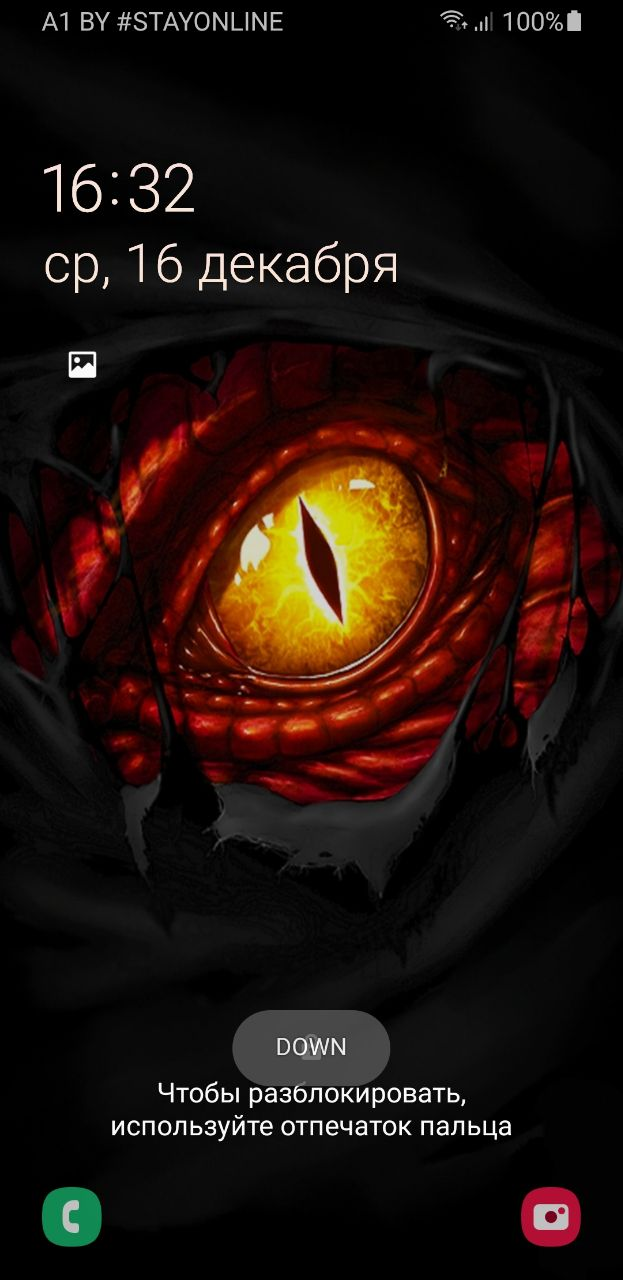
   

Рисунок 4.7 – «Key Сombination»

# **4.1 Выводы по главе «Руководство пользователя»**

Данная глава была посвящена руководству пользователя. Представлены скриншоты, инструкции и небольшое пояснение к ним. Что позволит пользователю использовать весь функционал данного проекта.

Так же инструкция позволяет пользователю узнать минимальные требования для установки приложения на телефон.

# **ЗАКЛЮЧЕНИЕ**

Целью моей курсовой работы было: сосредоточить возможности звукозаписи в единый центр управления, что позволит защитить пользователя и его конфиденциальность.

Были поставлены и решены следующие задачи:

1. Изучение аналогичных программных средства и литературные источники по теме курсового проекта.

Обзор уже реализованных аналогичных программных продуктов, а также их особенностей (положительные стороны), недочётов (ошибок), решений, описанных в литературе. Позволили улучшить не только функциональность, но и дизайн проекта.

1. Проектирование приложение для управления задачами.

Заранее определённые шаблоны, структуры необходимые для создания программного средства, а также заранее прописанные принципы их взаимодействия. Позволили значительно сократить время разработки, избежав при этом нюансы платформы.

1. Реализация приложения.

Для реализации приложения были использованы следующие технологии:

сред разработки Android Studio и Visual Studio Code СУБД SQLite, такие языки программирования как Java и Dart (SDK Flutter) библиотеки SQLCipher, OkHttp. Это позволило создать приемлемую визуальную и функциональную часть курсового проекта.

1. Разработка руководства пользователя.

После создания приложения было разработано руководство пользователя, позволяющее в полной мере использовать основные функции и возможности проекта.

По мимо поставленных задач, разработанное мобильное приложение выполняет следующие функции:

* Звукозапись в обычном и фоновом режиме
* Создание расписания включения звукозаписи в фоновом режиме
* Добавление и использование пользователей (которые будут участвовать в выполнении проекта).
* Активация звукозаписи при помощи комбинации клавиш
* Шифрование БД
* Были предприняты меры по обеспечению безопасности хранения и передачи данных.

# **Список использованных источников**

1 Руководство по работе с StatelessWidget и StatefulWidget [Электронный ресурс]. / flutter.su– Режим доступа: <https://flutter.su/tutorial/2-StatelessWidget-StatefulWidget> – Дата доступа: 29.09.2020.

1. Программирование под ОС Андроид «Работа с базами данных» [Электронный ресурс] / metanit.com. – Режим доступа: <https://metanit.com/java/android/14.1.php> . Дата доступа: 03.09.2020
2. Руководство по проектированию реляционных баз данных [Электронный ресурс] / metanit.com. – Режим доступа: <https://metanit.com/sql/tutorial/1.1.php> . Дата доступа: 07.11.2020
3. Flutter documentation [Электронный ресурс]. / flutter.dev Режим доступа: <https://flutter.dev/docs> – Дата доступа: 12.10.2020.
4. Google Drive Integration In Flutter [Электронный ресурс]. / habrahabr.ru – Режим доступа: <https://www.c-sharpcorner.com/article/google-drive-integration-in-flutter-upload-download-list-files/> – Дата доступа: 29.11.2020.
5. MediaSession [Электронный ресурс]. / developer.android.com – Режим доступа: [https://developer.android.com/reference/androidx/media2/session/MediaSession?hl=ru#KeyEvents](https://developer.android.com/reference/androidx/media2/session/MediaSession?hl=ru%23KeyEvents%20) – Дата доступа: 29.11.2020.

**ПРИЛОЖЕНИЕ A**

Следующий код позволяет создать структуру базы данных, а также дать представление о классе для взаимодействия с ней.

package com.dartbase.blackbox;

import android.accounts.Account;

import android.content.ContentValues;

import android.content.Context;

import android.database.sqlite.SQLiteConstraintException;

import android.util.Log;

import net.sqlcipher.Cursor;

import net.sqlcipher.database.SQLiteDatabase;

import net.sqlcipher.database.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DBHelper extends SQLiteOpenHelper {

public static class Account {

private String nickname;

private String password;

public Account(String \_nickname, String \_password) {

this.nickname = \_nickname;

this.password = \_password;

}

public String getNickname() {

return nickname;

}

public String getPassword() {

return password;

}

}

public static final ArrayList<Account> accounts = new ArrayList<>();

private static final String TAG = "DBHelper";

private static final String Password = "nXn-nwc-yuc-sg4";

private static final int SCHEMA = 1;

private static final String DATABASE\_NAME = "NEXUS";

private static Cursor cursor;

public static SQLiteDatabase db;

private static ContentValues contentValues;

// TODO: ///////// STRUCTURE TABLE

public static final String TABLE\_ACCOUNTS = "Accounts";

public static final String COLUMN\_ID = "\_id";

public static final String COLUMN\_PASSWORD = "Password";

public static final String COLUMN\_NICKNAME = "Nickname";

public DBHelper(Context context) {

super(context, DATABASE\_NAME, null, SCHEMA);

SQLiteDatabase.loadLibs(context);

}

public static DBHelper sqlHelper;

public static void initialization(Context context) {

sqlHelper = new DBHelper(context);

db = sqlHelper.getWritableDatabase();

}

public SQLiteDatabase getReadableDatabase() {

return (super.getReadableDatabase(Password));

}

public SQLiteDatabase getWritableDatabase() {

return (super.getWritableDatabase(Password));

}

@Override

public void onCreate(SQLiteDatabase db) {

try {

// TODO: PRAGMA

db.execSQL("PRAGMA foreign\_keys=on");

// TODO: ///////// TABLES

db.execSQL("CREATE TABLE Accounts (" +

" \_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE," +

" Nickname TEXT NOT NULL UNIQUE," +

" Password TEXT NOT NULL" +

")");

} catch (SQLiteConstraintException e) {

Log.e(TAG, e.toString());

}

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_ACCOUNTS);

onCreate(db);

}

// TODO://///////// CRUD

public static ArrayList<Account> selectAccounts() {

cursor = db.rawQuery("SELECT \* FROM Accounts", null);

while (cursor.moveToNext()) {

accounts.add(new Account(

cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN\_NICKNAME)),

cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN\_PASSWORD))

));

}

cursor.close();

return accounts;

}

public static boolean InsertAccount(Account \_account) {

contentValues = new ContentValues();

contentValues.put(DBHelper.COLUMN\_NICKNAME, \_account.getNickname());

contentValues.put(DBHelper.COLUMN\_PASSWORD, \_account.getPassword());

db.insert(DBHelper.TABLE\_ACCOUNTS, null, contentValues);

cursor = db.rawQuery("SELECT \* FROM Accounts ORDER BY \_id DESC LIMIT 1", null);

cursor.moveToFirst();

return true;

}

}

**ПРИЛОЖЕНИЕ Б**

Код для создания контроллеров

import 'package:blackbox/UI/Elements/navigation.dart';

import 'package:blackbox/UI/Pages/login\_page.dart';

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

theme: ThemeData(

brightness: Brightness.dark,

primaryColor: Colors.black,

accentColor: Colors.indigoAccent[700]),

home: LoginPage(),

);

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

import 'package:blackbox/Modules/User.dart';

import 'package:blackbox/UI/Elements/navigation.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:googleapis/poly/v1.dart';

class LoginPage extends StatefulWidget {

@override

\_LoginPageState createState() => \_LoginPageState();

}

CurrentUser currentUser;

int codeReturn = 0;

String CodeReturn = "";

class \_LoginPageState extends State<LoginPage> {

final formKey = GlobalKey<FormState>();

String \_nikeName = "";

String \_password = "";

@override

Widget build(BuildContext context) {

return Scaffold(

resizeToAvoidBottomPadding: false,

resizeToAvoidBottomInset: true,

appBar: AppBar(

title: Text("Login"),

),

body: SingleChildScrollView(

child: Card(

color: Colors.black26,

child: Column(

children: <Widget>[

Image.asset('image/logo.png'),

Padding(

padding: EdgeInsets.only(

top: 4, bottom: 105, left: 10, right: 10),

child: Form(

key: formKey,

child: Column(

mainAxisSize: MainAxisSize.min,

children: <Widget>[

TextFormField(

maxLength: 20,

decoration: InputDecoration(

labelText: 'NikeName:'

),

onSaved: (input) => \_nikeName = input,

),

TextFormField(

decoration: InputDecoration(

labelText: 'Password:'

),

onSaved: (input) => \_password = input,

validator: (input) => codeReturn == 2 ? "Incorrect Password or Nikename" : null,

obscureText: true,

),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Padding(

padding: const EdgeInsets.all(9.0),

child: RaisedButton(

color: Colors.white,

onPressed: \_submit,

child: Text(

'Sign in',

style: TextStyle(color: Colors.black)

),

),

)

],

)

],

),

),

),

],),

),

),);

}

void \_submit() {

formKey.currentState.save();

print(\_nikeName);

print(\_password);

\_login();

}

Future<void> \_login() async {

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"startBackgroundLogin",

<String, dynamic>{

'NikeName': \_nikeName,

'Password': \_password

});

debugPrint(data);

for (int i = 0; i < data.length; i++)

{

if(data[i] == '|'){

CodeReturn = data[i+1];

break;

}

}

if(CodeReturn == '1'){

debugPrint("Good");

codeReturn = 1;

CurrentUser(\_nikeName, \_password);

Navigator.push(context, MaterialPageRoute(builder: (context) => BottomNavigator()));

formKey.currentState.validate();

}

else if (CodeReturn == '2'){

debugPrint("Bad password");

codeReturn = 2;

formKey.currentState.validate();

}

else if (CodeReturn == '3'){

codeReturn = 3;

debugPrint("Insert");

}

else if (CodeReturn == '4'){

debugPrint("Crash");

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

import 'package:blackbox/Modules/User.dart';

import 'package:blackbox/UI/Pages/%D1%81loud\_storage\_page.dart';

import 'package:blackbox/UI/Pages/main\_page.dart';

import 'package:blackbox/UI/Pages/self\_recorder\_page.dart';

import 'package:blackbox/UI/Pages/storage\_page.dart';

import 'package:flutter/material.dart';

class BottomNavigator extends StatefulWidget {

@override

\_BottomNavigatorState createState() => \_BottomNavigatorState();

}

class \_BottomNavigatorState extends State<BottomNavigator> {

int \_selectedIndex = 0;

final List<Widget> \_pages = [

MainPage(),

SelfRecorderPage(),

CloudStoragePage(),

StoragePage()

];

void \_onItemTapped(int index) {

setState(() {

\_selectedIndex = index;

});

}

@override

Widget build(BuildContext context) {

return new Scaffold(

body: \_pages[\_selectedIndex],

bottomNavigationBar: BottomNavigationBar(

type: BottomNavigationBarType.fixed,

backgroundColor: Colors.black,

selectedItemColor: Colors.indigoAccent[400],

unselectedItemColor: Colors.grey.withOpacity(.60),

selectedFontSize: 14,

unselectedFontSize: 14,

onTap: \_onItemTapped,

currentIndex: \_selectedIndex,

items: [

BottomNavigationBarItem(

title: Text('Main'),

icon: Icon(Icons.mic),

),

BottomNavigationBarItem(

title: Text('Recorder'),

icon: Icon(Icons.date\_range),

),

BottomNavigationBarItem(

title: Text('Cloud'),

icon: Icon(Icons.cloud\_queue),

),

BottomNavigationBarItem(

title: Text('Storage'),

icon: Icon(Icons.library\_books),

),

],

),

);

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

import 'package:blackbox/Modules/User.dart';

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'dart:io' as io;

import 'dart:async';

import 'package:flutter\_audio\_recorder/flutter\_audio\_recorder.dart';

import 'package:audioplayers/audioplayers.dart';

import 'package:path\_provider/path\_provider.dart';

import 'package:blackbox/UI/Elements/navigation.dart';

class MainPage extends StatefulWidget {

@override

\_MainPageState createState() => \_MainPageState();

}

class \_MainPageState extends State<MainPage> {

String \_currentUser = "";

String \_serviceState = "";

Future \_getCurrentUser() async {

\_currentUser = "";

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"getCurrentUser");

for (int i = 0; i < data.length; i++)

{

if(data[i] != "|"){

\_currentUser += data[i];

}

else if(data[i] == "|")

break;

}

debugPrint("CurrentUser: " + \_currentUser);

}

Future<String> \_setListeningService() async {

\_serviceState = "";

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"setListeningService");

for (int i = 0; i < data.length; i++)

{

if(data[i] != "|"){

\_serviceState += data[i];

}

else if(data[i] == "|")

break;

}

debugPrint("ServiceState: " + \_serviceState);

return \_serviceState;

}

Future<String> \_getListeningService() async {

\_serviceState = "";

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"getListeningService");

for (int i = 0; i < data.length; i++)

{

if(data[i] != "|"){

\_serviceState += data[i];

}

else if(data[i] == "|")

break;

}

debugPrint("Get Service State: " + \_serviceState);

return \_serviceState;

}

/// Player ////////////////////////////////////////

FlutterAudioRecorder \_recorder;

Recording \_recording;

Timer \_t;

Widget \_buttonIcon = Icon(Icons.do\_not\_disturb\_on);

String \_alert;

@override

void initState() {

super.initState();

Future.microtask(() {

\_prepare();

});

}

void \_optButton() async {

await \_setListeningService();

if (\_serviceState == 'false') {

setState(() {

color = Colors.red;

scaffoldKey.currentState.showSnackBar(snackBarPayse);

});

}

else {

setState(() {

color = Colors.green;

scaffoldKey.currentState.showSnackBar(snackbarReady);

});

}

}

void \_opt() async {

switch (\_recording.status) {

case RecordingStatus.Initialized:

{

await \_startRecording();

visio = 0;

break;

}

case RecordingStatus.Recording:

{

await \_stopRecording();

visio = 10;

break;

}

case RecordingStatus.Stopped:

{

await \_prepare();

break;

}

default:

break;

}

// 刷新按钮

setState(() {

\_buttonIcon = \_playerIcon(\_recording.status);

});

}

Future \_init() async {

String customPath = '/' + \_currentUser + "- Black Box -";

io.Directory appDocDirectory;

if (io.Platform.isIOS) {

appDocDirectory = await getApplicationDocumentsDirectory();

} else {

appDocDirectory = await getExternalStorageDirectory();

}

// can add extension like ".mp4" ".wav" ".m4a" ".aac"

customPath = appDocDirectory.path +

customPath +

DateTime.now().millisecondsSinceEpoch.toString();

// .wav <---> AudioFormat.WAV

// .mp4 .m4a .aac <---> AudioFormat.AAC

// AudioFormat is optional, if given value, will overwrite path extension when there is conflicts.

\_recorder = FlutterAudioRecorder(customPath,

audioFormat: AudioFormat.AAC, sampleRate: 22050);

await \_recorder.initialized;

}

Future \_prepare() async {

await \_getCurrentUser();

await \_getListeningService();

var hasPermission = await FlutterAudioRecorder.hasPermissions;

if (hasPermission) {

await \_init();

var result = await \_recorder.current();

setState(() {

\_recording = result;

\_buttonIcon = \_playerIcon(\_recording.status);

\_alert = "";

});

} else {

setState(() {

\_alert = "Permission Required.";

});

}

}

Future \_startRecording() async {

await \_recorder.start();

var current = await \_recorder.current();

setState(() {

\_recording = current;

});

\_t = Timer.periodic(Duration(milliseconds: 10), (Timer t) async {

var current = await \_recorder.current();

setState(() {

\_recording = current;

\_power = \_GetPower();

\_t = t;

});

});

}

Future \_stopRecording() async {

var result = await \_recorder.stop();

\_t.cancel();

setState(() {

\_recording = result;

});

}

AudioPlayer player;

void \_play() {

player = AudioPlayer();

player.play(\_recording.path, isLocal: true);

}

Widget \_playerIcon(RecordingStatus status) {

switch (status) {

case RecordingStatus.Initialized:

{

return Icon(Icons.fiber\_manual\_record);

}

case RecordingStatus.Recording:

{

return Icon(Icons.stop);

}

case RecordingStatus.Stopped:

{

return Icon(Icons.replay);

}

default:

return Icon(Icons.do\_not\_disturb\_on);

}

}

final GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();

final SnackBar snackbarReady = const SnackBar(content: Text('Keystroke сapture Enabled'));

final SnackBar snackBarPayse = const SnackBar(content: Text('Keystroke capture Disabled!'));

Color color = Colors.red;

int visio = 10;

double \_GetPower(){

double rez = \_recording.metering.averagePower \* (-0.01);

return rez + visio;

}

double \_power = 1;

@override

Widget build(BuildContext context) {

return Scaffold(

key: scaffoldKey,

appBar: AppBar(

title: const Text('Main'),

actions: <Widget>[

IconButton(

icon: new Icon(

Icons.album,

color: color

),

tooltip: 'Key entry',

onPressed: \_optButton,

)

],

),

body: SingleChildScrollView(

child: Column(

children: <Widget>[

Padding(

padding: const EdgeInsets.only(top: 0.10,left: 0.10, right:0.10, bottom: 10),

child: LinearProgressIndicator(

value: \_power,

backgroundColor: Colors.blue,

valueColor: AlwaysStoppedAnimation<Color>(Colors.red),

),

),

Padding(

padding: const EdgeInsets.all(40.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.start,

children: <Widget>[

Text(

'File',

style: Theme

.of(context)

.textTheme

.title,

),

SizedBox(

height: 5,

),

Text(

'${\_recording?.path ?? "-"}',

style: Theme

.of(context)

.textTheme

.body1,

),

SizedBox(

height: 20,

),

Text(

'Duration',

style: Theme

.of(context)

.textTheme

.title,

),

SizedBox(

height: 5,

),

Text(

'${\_recording?.duration ?? "-"}',

style: Theme

.of(context)

.textTheme

.body1,

),

SizedBox(

height: 20,

),

Text(

'Metering Level - Average Power',

style: Theme

.of(context)

.textTheme

.title,

),

SizedBox(

height: 5,

),

Text(

'${\_recording?.metering?.averagePower ?? "-"}',

style: Theme

.of(context)

.textTheme

.body1,

),

SizedBox(

height: 20,

),

Text(

'Status',

style: Theme

.of(context)

.textTheme

.title,

),

SizedBox(

height: 5,

),

Text(

'${\_recording?.status ?? "-"}',

style: Theme

.of(context)

.textTheme

.body1,

),

SizedBox(

height: 20,

),

RaisedButton(

child: Text('Play'),

disabledTextColor: Colors.white,

disabledColor: Colors.grey.withOpacity(0.5),

onPressed: \_recording?.status == RecordingStatus.Stopped

? \_play

: null,

),

SizedBox(

height: 20,

),

Text(

'${\_alert ?? ""}',

style: Theme

.of(context)

.textTheme

.title

.copyWith(color: Colors.red),

),

],

),

),

],

),

),

floatingActionButton: FloatingActionButton(

backgroundColor: Colors.red,

onPressed: \_opt,

child: \_buttonIcon

),

);

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:intl/intl.dart';

import 'package:path\_provider/path\_provider.dart';

import 'dart:io' as io;

class SelfRecorderPage extends StatefulWidget {

@override

\_SelfRecorderPageState createState() => \_SelfRecorderPageState();

}

class \_SelfRecorderPageState extends State<SelfRecorderPage> {

String \_currentUser = "";

Future \_getCurrentUser() async {

\_currentUser = "";

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"getCurrentUser");

for (int i = 0; i < data.length; i++)

{

if(data[i] != "|"){

\_currentUser += data[i];

}

else if(data[i] == "|")

break;

}

debugPrint("CurrentUser: " + \_currentUser);

}

void \_BackgroundChannel() async {

DateTime \_timeStart = new DateTime(

\_dateTime.year

, \_dateTime.month

, \_dateTime.day

, \_timeOfDay.hour

, \_timeOfDay.minute

);

DateTime difTime = new DateTime(\_currentTime.year,\_currentTime.month,\_currentTime.day);

DateTime periodRange = new DateTime(\_currentTime.year,\_currentTime.month,\_currentTime.day,\_period.hour,\_period.minute);

Duration differenceDate = \_timeStart.difference(\_currentTime);

Duration differencePeriod = periodRange.difference(difTime);

debugPrint(differenceDate.inMinutes.toString());

debugPrint(differencePeriod.inMinutes.toString());

String TimeStart = differenceDate.inMinutes.toString();

String Period = differencePeriod.inMinutes.toString();

await \_getCurrentUser();

String customPath = '/' + \_currentUser + "-Black Box-selfRecorder";

io.Directory appDocDirectory;

if (io.Platform.isIOS) {

appDocDirectory = await getApplicationDocumentsDirectory();

} else {

appDocDirectory = await getExternalStorageDirectory();

}

// can add extension like ".mp4" ".wav" ".m4a" ".aac"

customPath = appDocDirectory.path +

customPath +

DateTime.now().millisecondsSinceEpoch.toString() + ".m4a";

var methodChannel = MethodChannel("com.dartbase.blackbox");

String data = await methodChannel.invokeMethod(

"createBackgroundWorker",

<String, dynamic>{

'TimeStart': TimeStart,

'Period': Period,

'Path': customPath,

});

debugPrint(data);

}

///

DateTime \_dateTime = new DateTime.now();

TimeOfDay \_timeOfDay = new TimeOfDay.now();

TimeOfDay \_period = new TimeOfDay.now();

DateTime \_currentTime = new DateTime.now();

final DateFormat \_dateFormat = DateFormat('dd-MM-yyyy');

final DateFormat \_timeFormat = DateFormat('HH:MM');

///

Future<Null> \_selectDate(BuildContext context) async {

final DateTime picker = await showDatePicker(

context: context,

initialDate: \_dateTime,

firstDate: new DateTime.now(),

lastDate: new DateTime(DateTime.now().year + 1));

if (picker != null && picker != \_dateTime) {

print("Date selected: ${\_dateTime.toString()}");

setState(() {

\_dateTime = picker;

});

}

}

Future<Null> \_selectTime(BuildContext context) async {

final TimeOfDay picker = await showTimePicker(

context: context,

initialTime: \_timeOfDay

);

if (picker != null && picker != \_timeOfDay) {

print("Date selected: ${\_timeOfDay.toString()}");

setState(() {

\_timeOfDay = picker;

});

}

}

Future<Null> \_selectPeriod(BuildContext context) async {

DateTime defTime = new DateTime(\_currentTime.year,\_currentTime.month,\_currentTime.day);

TimeOfDay defPeriod = new TimeOfDay.fromDateTime(defTime);

final TimeOfDay picker = await showTimePicker(

context: context,

initialTime: defPeriod,

);

if (picker != null && picker != \_period) {

print("Date selected: ${\_period.toString()}");

setState(() {

\_period = picker;

});

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("Recorder"),

),

body: new Center(

child: Column(

children: [

FlatButton(

minWidth: 400,

color: Colors.blue,

onPressed: () {

\_selectDate(context);

},

child: Text(

'Start Date ${\_dateFormat.format(\_dateTime)}',

style: Theme

.of(context)

.textTheme

.title,

),

),

FlatButton(

minWidth: 400,

color: Colors.blue,

onPressed: () {

\_selectTime(context);

},

child: Text(

'Start time - ${\_timeOfDay.hour}:${\_timeOfDay.minute}',

style: Theme

.of(context)

.textTheme

.title,

),

),

FlatButton(

minWidth: 400,

color: Colors.blue,

onPressed: () {

\_selectPeriod(context);

},

child: Text(

'Period - ${\_period.hour}:${\_period.minute}',

style: Theme

.of(context)

.textTheme

.title,

),

),

FlatButton(

minWidth: 400,

color: Colors.blue,

onPressed: () {

\_BackgroundChannel();

},

child: Text(

'Set record',

style: Theme

.of(context)

.textTheme

.title,

),

),

],

),

),

);

}

}

**ПРИЛОЖЕНИЕ В**

Код для создания «Фонового функционала» (BackgroundWorker & PlayerService)

package com.dartbase.blackbox;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.content.Context;

import android.os.Build;

import android.util.Log;

import androidx.annotation.NonNull;

import androidx.core.app.NotificationCompat;

import androidx.work.Data;

import androidx.work.Worker;

import androidx.work.WorkerParameters;

import java.util.concurrent.TimeUnit;

import static com.dartbase.blackbox.MainActivity.\_period;

import static com.dartbase.blackbox.MainActivity.fileName;

public class BackgroundWorker extends Worker {

private static final String WORK\_RESULT = "work\_result";

public BackgroundWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {

super(context, workerParams);

}

@NonNull

@Override

public Result doWork() {

Data taskData = getInputData();

String taskDataString = taskData.getString(MainActivity.MESSAGE\_STATUS);

showNotification("Black Box", taskDataString != null ? taskDataString : "Will work in the period " + String.valueOf(\_period) + " Minutes");

Data outputData = new Data.Builder().putString(WORK\_RESULT, "Jobs Finished").build();

RecoderManager recoderManager = new RecoderManager();

recoderManager.recordStart(fileName);

try {

Thread.sleep(\_period \* 60000); ////////////// 60000 for minutes /// 1000 for seconds

recoderManager.recordStop();

} catch (InterruptedException e) {

e.printStackTrace();

}

return Result.success(outputData);

}

private void showNotification(String task, String desc) {

NotificationManager manager = (NotificationManager) getApplicationContext().getSystemService(Context.NOTIFICATION\_SERVICE);

String channelId = "task\_channel";

String channelName = "task\_name";

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

NotificationChannel channel = new

NotificationChannel(channelId, channelName, NotificationManager.IMPORTANCE\_DEFAULT);

manager.createNotificationChannel(channel);

}

NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext(), channelId)

.setContentTitle(task)

.setContentText(desc)

.setSmallIcon(R.mipmap.ic\_launcher);

manager.notify(1, builder.build());

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

package com.dartbase.blackbox;

import android.app.Service;

import android.content.Intent;

import android.media.MediaRecorder;

import android.os.Environment;

import android.os.IBinder;

import android.support.v4.media.session.MediaSessionCompat;

import android.support.v4.media.session.PlaybackStateCompat;

import android.widget.Toast;

import androidx.media.VolumeProviderCompat;

import java.sql.Time;

import java.text.SimpleDateFormat;

import java.util.Date;

import static com.dartbase.blackbox.MainActivity.\_timeStart;

import static com.dartbase.blackbox.MainActivity.currentUser;

public class PlayerService extends Service {

private MediaSessionCompat mediaSession;

VolumeProviderCompat myVolumeProvider;

RecoderManager recoderManager;

boolean recoderManagerWork = false;

private String fileName;

int curValue = 50;

@Override

public void onCreate() {

super.onCreate();

Date dateNow = new Date();

SimpleDateFormat formatForDateNow = new SimpleDateFormat("yyyy\_MM\_dd-hh\_mm\_ss");

recoderManager = new RecoderManager();

recoderManager = new RecoderManager();

fileName = "/storage/emulated/0/Android/data/com.dartbase.blackbox/files/"

+ currentUser

+ "-Black Box-KEY"

+ formatForDateNow.format(dateNow).toString() + ".m4a";

mediaSession = new MediaSessionCompat(this, "PlayerService");

mediaSession.setFlags(MediaSessionCompat.FLAG\_HANDLES\_MEDIA\_BUTTONS |

MediaSessionCompat.FLAG\_HANDLES\_TRANSPORT\_CONTROLS);

mediaSession.setPlaybackState(new PlaybackStateCompat.Builder()

.setState(PlaybackStateCompat.STATE\_PLAYING, 0, 0) //you simulate a player which plays something.

.build());

myVolumeProvider =

new VolumeProviderCompat(VolumeProviderCompat.VOLUME\_CONTROL\_RELATIVE, /\*max volume\*/100, /\*initial volume level\*/50) {

@Override

public void onAdjustVolume(int direction) {

if (direction > 0) {

curValue += 1;

if (curValue >= 100) {

curValue = 100;

Toast.makeText(getApplicationContext(), "UP", Toast.LENGTH\_SHORT).show();

if (recoderManagerWork) {

Toast.makeText(getApplicationContext(), "The recording is already working", Toast.LENGTH\_SHORT).show();

} else {

recoderManagerWork = true;

recoderManager.recordStart(fileName);

}

}

}

else if (direction == 0){

curValue = 50;

}

else if (direction < 0) {

curValue -= 1;

if (curValue <= 0) {

curValue = 0;

Toast.makeText(getApplicationContext(), "DOWN", Toast.LENGTH\_SHORT).show();

if (recoderManagerWork) {

recoderManager.recordStop();

recoderManagerWork = false;

} else {

Toast.makeText(getApplicationContext(), "Press the sound up button (MAX) to start", Toast.LENGTH\_SHORT).show();

}

}

}

}

};

mediaSession.setPlaybackToRemote(myVolumeProvider);

mediaSession.setActive(true);

}

@Override

public IBinder onBind(Intent intent) {

return null;

}

@Override

public void onDestroy() {

super.onDestroy();

mediaSession.release();

}

}