Оглавление

[ВВЕДЕНИЕ 4](#_Toc422037528)

[ГЛАВА 1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 6](#_Toc422037529)

[1.1 Анализ предметной области 6](#_Toc422037530)

[1.2 Обзор инструментов для создания мобильных приложений 6](#_Toc422037531)

[1.3 Выбор программных средств разработки 7](#_Toc422037532)

[1.4 Постановка задачи 8](#_Toc422037533)

[ГЛАВА 2. СПЕЦИАЛЬНЫЙ РАЗДЕЛ 10](#_Toc422037534)

[2.1 Разработка структурной схемы программы 10](#_Toc422037535)

[2.2 Разработка схемы базы данных 11](#_Toc422037536)

[2.3 Разработка шаблонов выходных документов 12](#_Toc422037537)

[2.4 Разработка интерфейса программы 14](#_Toc422037538)

[ГЛАВА 3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 16](#_Toc422037539)

[3.1 Технологии создания мобильных приложений 16](#_Toc422037540)

[3.2 Технология ADO 18](#_Toc422037541)

[3.3 Тестирование программы 20](#_Toc422037542)

[ГЛАВА 4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ 22](#_Toc422037543)

[4.1 Системные требования приложения 22](#_Toc422037544)

[4.2 Руководство системного программиста 22](#_Toc422037545)

[4.3 Руководство программиста 22](#_Toc422037546)

[4.4 Руководство оператора 25](#_Toc422037547)

[4.4.1 Основное окно 26](#_Toc422037548)

[4.4.2 Окно выбора/создания файла 27](#_Toc422037549)

[4.4.2 Окно ввода данных 28](#_Toc422037550)

[4.4.3 Окно таймера 30](#_Toc422037551)

[4.4.4 Окно просмотра списка файлов 32](#_Toc422037552)

[4.4.5 Окно просмотра результатов/записей 34](#_Toc422037553)

[ЗАКЛЮЧЕНИЕ 35](#_Toc422037554)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc422037555)

[ПРИЛОЖЕНИЕ 1. БЛОК-СХЕМА 38](#_Toc422037556)

[ПРИЛОЖЕНИЕ 2. КОД ПРОГРАММЫ 39](#_Toc422037557)

# ВВЕДЕНИЕ

Целью данного дипломного проекта является создание статистической программы для отсчета временных данных при проведении каких-либо соревнований по физической культуре.

Проект является достаточно актуальным в наше время, так как для проведения мероприятий с одновременным участием в них нескольких участников, и где важны точные временные данные каждого участника, требуется наличие нескольких таймеров, по таймеру на каждого участника. С обычным секундомером можно легко запутаться, и он не имеет возможности хранить данные постоянно. Для достижения поставленной цели необходимо решить следующие задачи:

* Создание таймера для каждого участника;
* Наглядность таймера, легкое управление таймером и интервалом старта участников;
* Возможность записывать и хранить данные;
* Создание таблиц участников и удобный их просмотр.

Данный проект уже был реализован мной в виде небольшой программы для персональных компьютеров на языке программирования «C#», по просьбе нашего учителя по физической культуре - Никишина Игоря Валентиновича. Но использование такой программы на персональном компьютере не удобно и было решено реализовать подобную программу для мобильных устройств на операционной системе «Android». Таким образом будет достигнута мобильность и удобность в использовании проекта, а также вся записанная информация будет находится под рукой.

Для реализации проекта был выбран язык «C#» и среды разработки «Visual Studio 2012\2013» и «Xamarin Studio». Так как проект планировался для «Android» устройств, в нем дополнительно используются расширяемый язык разметки «XML» и, для работы с базами данных, язык структурированных запросов «SQL». Проект является очень актуальным для меня лично, так как я впервые буду работать с данной средой разработки и с платформой «Android».

Заказчиком программы остался Никишин Игорь Валентинович.

# ГЛАВА 1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## 1.1 Анализ предметной области

Отсчет временных данных применяется практически во всех мероприятиях соревновательного характера, где время и то, насколько ты быстро выполнишь поставленные тебе задачи, являются главными факторами для присуждения мест и выявления победителя. Большинство этих мероприятий носят спортивный характер – кроссы, плавание, различные виды гонок и т.п. Для проведения больших мероприятий, где требуется отсчет временных данных, уже существует множество специального оборудования и программ, но все подобные программы и оборудование являются платными или не имеют конкретно заданной функции отсчета временных данных с определенным интервалом между участниками. Также код большинства программ является закрытым и, так как рассматривались только мобильные программы, написан на языке программирования «Java». Исследовать данный код не представлялось возможным и необходимым, так как данный проект планировался на языке «C#». Все похожие программы брались из официального магазина приложений «Google Play Market». Исследовались программы, предназначенные для физических тренировок и включающие в себя функцию «мультитаймера».

## 1.2 Обзор инструментов для создания мобильных приложений

Существует 3 основные операционные системы для мобильных устройств – «Android», «iOS» и «Windows phone». Для всех существуют разные наборы инструментов для создания приложений.

Для выбранной операционной системы «Android» основным набором инструментов являются:

* Android SDK (software development kit) - комплект средств разработки;
* Java Development Kit - бесплатно распространяемый компанией Oracle Corporation комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE). В состав JDK не входит интегрированная среда разработки на Java;
* Java Runtime Environment - минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки.

Но для создания приложений требуется также среда разработки. Для разработки мобильных приложений на языке «Java» существуют две среды разработки – «Eclipse» и «Android studio». На языке «C#» средой и инструментом для разработки служит «Xamarin Studio». Для ее работы также требуется установка «.NET Framework» и «Microsoft Visual studio 2012/2013».

## 1.3 Выбор программных средств разработки

Для реализации проекта был выбран язык «С#» и среды разработки «Microsoft Visual Studio 2012\2013» и «Xamarin Studio». Системой управления реляционными базами данных была выбрана «SQLite».

Язык «C#» был выбран потому, что он наиболее изучен мной и включает в себя множество различных встраиваемых средств для дополнительных возможностей, он является кроссплатформенным, и разрабатывать приложения можно не только для Windows, но и для многих других операционных систем. Также доступно много литературы, как на русском, так и на английском языках.

«Microsoft Visual Studio 2012\2013» была выбрана, так как является «родной» средой разработки, от разработчиков самого языка, и она является наиболее удобной и функциональной, с поддержкой множества дополнительных средств таких как – «SQLite» и «Xamarin Studio».

«Xamarin Studio» была выбрана потому, что она является единственной средой разработки для операционной системы «Android» на языке «C#» и легко встраивается в «Microsoft Visual Studio 2012\2013». К сожалению, бесплатная версия «Xamarin Studio» имеет ограничение на размер кода программы, включая дополнительные библиотеки, и в ней невозможно реализовать собственный интерфейс различных объектов, сделать кнопку другой формы или в виде картинки. А 30 дневная бесплатная версия студии разрешает использовать скомпилированное приложение только в течении 24 часов, но не имеет никаких ограничений в коде. Так как кроме образовательной части у данного проекта имеется заказчик, было решено разрабатывать программу в полностью бесплатной, но ограниченной версии «Xamarin Studio».

Для локальной базы данных на «Android» устройствах, с поддержкой языка программирования «C#», выбрана «SQLite». Она удобна и легка в использовании, и является единственной доступной СУБД для выбранных мною средств разработки и «Android».

## 1.4 Постановка задачи

Программа должна содержать в себе локальную базу данных в которой должны храниться:

* Фамилия, имя и отчество участников;
* Порядковый номер участников;
* Временные данные, в данном случае секунды и минуты.

Должна быть предусмотрена сортировка этой базы данных по всем ее полям, а также удобный ввод и редактирование существующих и новых записей, с последующим их хранением в памяти устройства, и возможность просмотра любой сохраненной базы данных.

Для записи временных данных должен быть реализован секундомер с функцией автоматического старта для каждой записи в базе данных через указанный промежуток времени (в секундах) и его ручная остановка. Также должна быть предусмотрена остановка таймера для нескольких участников одновременно.

Программа должна предусматривать возможность создания файлов с записями участников заранее, чтобы не приходилось вбивать всех участников прямо перед мероприятием, а также возможность перезаписывать существующие файлы, использовать один файл для многократного использования.

# ГЛАВА 2. СПЕЦИАЛЬНЫЙ РАЗДЕЛ

## 2.1 Разработка структурной схемы программы

Структура программы представляет собой 6 классов типа «Activity», одного класса работы с базой данных, класса адаптера и класса модели адаптера для listview, а также 7 «.axml» файлов разметки, 6 для каждого «Activity» класса и 1 для адаптера listview. Вся структура программы представлена в таблице 2.1.

Таблица 2.1 – Список файлов

|  |  |
| --- | --- |
| Наименование | Описание |
| MainActivity.cs | Основной активный класс |
| DB\_Create.cs | Активный класс заполнения базы данных |
| DB\_View.cs | Активный класс просмотра базы данных |
| FileBrowser.cs | Активный класс просмотра списка файлов баз данных |
| Timer.cs | Активный класс работы таймера |
| Load\_CreateFile.cs | Активный класс создания/выбора файла |
| ListView\_Adapter.cs | Класс адаптера для объекта «ListView» |
| ListView\_Model.cs | Класс модели адаптера для объекта «ListView» |
| WorkWithDB.cs | Класс в котором описаны все методы работы с файлами и базой данных |
| Main.axml | Файл разметки окна для MainActivity.cs |
| DB\_Create.axml | Файл разметки окна для DB\_Create.cs |
| DB\_View.axml | Файл разметки окна для DB\_View.cs |
| File\_Browser.axml | Файл разметки окна для FileBrowser.cs |
| Timer.axml | Файл разметки окна для Timer.cs |
| row.axml | Файл разметки объекта «ListView» описанного в его классах адаптера и модели |
| Load\_CreateFile.axml | Файл разметки для окна Load\_CreateFile.cs |

«Activity» классы отвечают за определенную часть программы в определенный момент времени, то есть при запуске программы активируется «MainActivity.cs» класс, который работает всегда в фоновом режиме, и в котором производится запуск других подобных классов – просмотр файлов («FileBrowser.cs») и заполнение нового файла(«DB\_Create.cs»). Дальше уже в этих классах вызываются новые классы активности, а старые завершаются или остаются работать в фоновом режиме. При закрытии «MainActivity.cs» программа будет закрыта.

Для каждого активного класса существует «.axml» файл разметки. Это файл внешнего вида программы. Каждый такой файл отвечает за интерфейс конкретного окна или же объекта.

В приложении 1 находится краткая блок-схема работы программы.

## 2.2 Разработка схемы базы данных

База данных состоит из двух типов файлов с разными таблицами баз данных. Первый тип имеет структуру базы данных, представленную в таблице 2.2.

Таблица 2.2 – Структура файлов с участниками

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| \_ID | Порядковый номер участника | integer |
| Name | Ф.И.О. участника или любое другое его наименование | ntext |
| Time | Время участника | ntext |

Для каждого нового использования таймера создается файл с данной структурой. Имя файла задается вручную пользователем программы.

Во втором типе базы данных хранятся порядковый номер, наименования, пути и время создания файлов первого типа. Файл с данной структурой создается один раз при первом запуске программы на новом устройстве и имеет название «Records.db». Структура изображена на таблице 2.3.

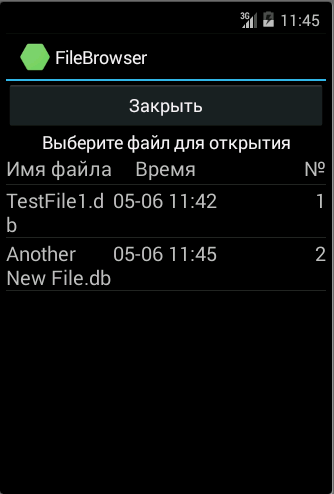
Таблица 2.3 – Структура файла с записями программы

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| \_ID | Порядковый номер файла | integer |
| Name | Имя файла | ntext |
| Time | Время создания файла | ntext |
| Path | Путь к файлу | ntext |

Таблицы этих баз данных никак не пересекаются между собой и работают отдельно друг от друга, в файл «Records.db» записывается только системная информация о файлах и ничего более. Формат файлов – «.db».

## 2.3 Разработка шаблонов выходных документов

Для вывода списка всех файлов и записей конкретного файла был разработан специальный адаптер и его модель для объекта «ListView». На рис. 2.1 изображен интерфейс вывода всех файлов на устройстве.



Рисусок 2.1 – Список файлов

На рис. 2.2 изображен интерфейс вывода записей выбранного файла.

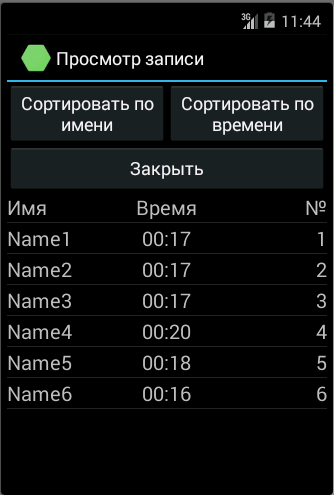


Рисунок 2.2 – Записи выбранного файла

При выводе списка файлов, при нажатие на строку, с конкретным файлом, откроется окно просмотра записей выбранного файла, или же появится окно о подтверждении удаления файла, если список файлов был вызван из основного окна для удаления записей. В окне просмотра записей, при нажатии на соответствующие кнопки, будет произведена сортировка записей на экране.

## 2.4 Разработка интерфейса программы

Так как программа разрабатывалась для мобильных устройств и размеры экрана на таковых имеют маленький размер, интерфейс программы должен быть наглядным, функциональным и минималистичным.

Первым делом нужно было продумать вывод данных на экран, и элементы интерфейса для заполнения данных. Для этого был реализован собственный адаптер для данных, так как стандартные наборы объектов в «Android» не предусматривают вывод таблиц данных и их заполнение. Адаптер представляет из себя строку «ListView», разделенную на столбцы, и для каждого столбца присвоены свои параметры и переменная для хранения данных. Файл разметки данного адаптера изображен на рис. 2.3.

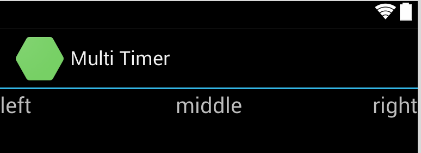


Рисунок 2.3 – Адаптер

Как видно на рисунке, строка разбивается на 3 столбца с названиями «left», «middle» и «right». В данном файле строка представляет из себя 3 объекта типа «TextView». Дальше, с помощью кода, находящегося в файлах адаптера и его модели – «ListView\_Adapter.cs» и «ListView\_Model.cs», три объекта «TextView» превращаются в 1 строку объекта «ListView», и, в дальнейшем, во всем коде программы используются как единая строка. На рис. 2.4 показан итоговый интерфейс данного объекта.



Рисунок 2.4 – Адаптер 2

Все остальные элементы интерфейса и объекты взяты из стандартного набора объектов «Android». Более подробно интерфейс рассмотрен в руководстве оператора. Интерфейс программы не включает в себя стандартные кнопки операционной системы «Android» - «назад», «домой» и «меню», а также экранную клавиатуру, так как на разных мобильных устройствах они отличаются.

# ГЛАВА 3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 3.1 Технологии создания мобильных приложений

Существует множество различных технологий создания мобильных приложений для разных мобильных операционных систем. В основном все технологии для операционной системы «Android» подразумевают знание языка «Java». Данное же приложение было написано на языке «C#» с помощью технологий «Xamarin».

«Xamarin» — это фреймворк для кроссплатформенной разработки мобильных приложений («iOS», «Android», «Windows Phone») с использованием языка «C#».

Фреймворк состоит из нескольких основных частей:

* «Xamarin.IOS» — библиотека классов для «C#», предоставляющая разработчику доступ к «iOS SDK»;
* «Xamarin.Android» — библиотека классов для «C#», предоставляющая разработчику доступ к «Android SDK»;
* Компиляторы для «iOS» и «Android»;
* «IDE Xamarin Studio»;
* Плагин для «Visual Studio».

«Xamarin» основан на open-source реализации платформы «.NET» — «Mono». Эта реализация включает в себя собственный компилятор «C#», среду выполнения, а также основные «.NET» библиотеки. Цель проекта — позволить запускать программы, написанные на «C#», на операционных системах, отличных от «Windows» — «Unix-системах», «Mac OS» и других.

С точки зрения исполнения приложений между «iOS» и «Android» есть одно ключевое различие — способ их предварительной компиляции. Как известно, для выполнения приложений в «Android» используется виртуальная «Java-машина Dalvik». Нативные приложения, которые пишутся на «Java», компилируются в некий промежуточный байт-код, который интерпретируется «Dalvik`ом» в команды процессора в момент исполнения программы(т.е. аналогично тому, как работает CLR в .NET). Это так называемая «Just-in-time» компиляция (компиляция на лету). В «iOS» используется другая модель компиляции — «Ahead-of-Time» (компиляция перед исполнением). «Xamarin» учитывает это различие, предоставляя отдельные компиляторы для каждой из этих платформ, которые позволяют на выходе получать настоящие, нативные приложения, которые выполняются вне контекста браузера и могут использовать все аппаратные и программные ресурсы платформы (9).

Для «iOS» ситуация простая — никакой виртуальной машины нет и программный код должен быть просто заранее скомпилирован в машинный. Для «Android» интереснее. При компиляции приложения происходит перевод кода на C# в промежуточный байт-код, понятный виртуальной машине «Mono» и сама эта виртуальная машина также добавляется в упакованное приложение. И «Mono» и «Dalvik» написаны на Си и работают поверх ядра «Linux». «Android» тоже основана на «Linux». При запуске приложения на «Android» обе виртуальные машины начинают работать бок о бок и обмениваются данными через специальный механизм wrapper`ов (9).

Для каждой платформы «Xamarin» предоставляет возможность использовать нативные средства разработки пользовательского интерфейса и их нативные элементы. Для «Android» создание пользовательского интерфейса может происходить непосредственно в коде или же при помощи декларативного подхода с описанием интерфейса в «XML». Для «iOS» это также либо код, либо использование нативных средств проектирования интерфейса — отдельные xib-файлы или же один большой Storyboard. Редактирование этих файлов происходит в привычной для iOS-разработчика среде «XCode» (9).

«Xamarin» является кроссплатформенной средой, но из-за проблем с уникальностью элементов пользовательского интерфейса на разных платформах переносимость кода вряд ли может превысить 50-60%. Инженеры «Xamarin» это понимают, поэтому стремятся к увеличению этой доли. В качестве достижений в решении этой проблемы можно рассматривать библиотеку «Xamarin.Mobile». Она предоставляет единый для различных платформ API для работы с камерой, контактами и гео-локацией.

Для написания небольших приложений на языке «C#» данная среда является идеальным решением с большим потенциалом и возможностями, но является платной, и не сравнится с функционалом нативных приложений, написанных на языке «Java». Для разработки больших и сложных приложений для платформы «Android» лучше начать изучать язык программирования «Java», и принцип работы этой платформы, тем более, что так разработка не будет требовать никаких платных сред.

## 3.2 Технология ADO

«ADO» (от англ. ActiveX Data Objects — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным, разработанный компанией «Microsoft» («MS Access», «MS SQL Server») и основанный на технологии компонентов «ActiveX». «ADO» позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде.

Объектная модель «ADO» состоит из следующих объектов высокого уровня и семейств объектов:

* Connection (представляет подключение к удалённому источнику данных)
* Recordset (представляет набор строк, полученный от источника данных)
* Command (используется для выполнения команд и SQL-запросов с параметрами)
* Record (может представлять одну запись объекта Recordset или же иерархическую структуру, состоящую из текстовых данных)
* Stream (используется для чтения и записи потоковых данных, например, документов XML или двоичных объектов)
* Errors (представляет ошибки)
* Fields (представляет столбцы таблицы базы данных)
* Parameters (представляет набор параметров SQL-инструкции)
* Properties (представляет набор свойств объекта)

Компоненты «ADO» используются в языках высокого уровня, таких как «VBScript» в «ASP», «JScript» в «WSH», «Visual Basic», «Delphi».

Последней версией «ADO» является версия 2.8. В рамках платформы «Microsoft .NET» интерфейс «ADO» заменён «ADO.NET».

Данная технология устарела и в языке программирования «C#» используется своя технология «ADO.NET».

«ADO.NET» — технология, предоставляющая доступ к данным для приложений, основанных на «Microsoft .NET». Является не развитием более ранней технологии «ADO», а самостоятельной технологией, частью фреймворка «.NET».В отличие от классической «ADO», которая была в основном предназначена для тесно связанных клиент-серверных систем, «ADO.NET» больше нацелена на автономную работу с помощью объектов «DataSet». Эти типы представляют локальные копии любого количества взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты «DataSet» позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым «DataSet», изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных. Но, пожалуй, самое фундаментальное различие между классической «ADO» и «ADO.NET» состоит в том, что «ADO.NET» является управляемой кодовой библиотекой, и, значит, подчиняется тем же правилам, что и любая управляемая библиотека. Типы, составляющие «ADO.NET», используют протокол управления памятью «CLR», принадлежат к той же системе типов (классы, интерфейсы, перечисления, структуры и делегаты), и доступ к ним возможен с помощью любого языка «.NET». Классы» ADO.NET» находятся в сборке «System.Data.dll» (5,6).

Данная технология используется в моей программе совместно с «Mono.Data.Sqlite», которая необходима для работы с данными на «Android» и включает в себя часть «ADO.NET», необходимую для работы с базами данных «SQLite».

## 3.3 Тестирование программы

Программа тестировалась методом «белого ящика» модульным тестированием. Каждый модуль тестировался отдельно и корректировался для устранения найденных ошибок. Затем было проведено интеграционное тестирование, и проверялась работа взаимодействия различных модулей программы между собой. После тестирования была выявлена ошибка, связанная с обновлением интерфейса, а конкретнее, с объектом ListView. Ошибка не позволяет прокручивать список во время работы таймера и постоянно, при попытке прокрутки, возвращает список в исходное положение к первой записи, так как происходит ежесекундное обновление интерфейса данного объекта для вывода на экран времени участников. Данная ошибка связана с операционной системой «Android», и, из-за нехватки знаний, в настоящее время не может быть исправлена. Для удобства просмотра списка участников во время таймера, был реализован метод, который удаляет участников из списка, которые уже были остановлены.

Также после тестирования методом «белого ящика» было проведено интеграционное и системное тестирование методом черного ящика добровольцами, а также заказчиком программы. В 95% случаев программа работала корректно и достаточно быстро, были небольшие недостатки, описанные дальше, но основная функция программы выполнялась верно. В 5% случаев была выявлена ошибка, которая зависит от самого устройства и тем, как он реализует тик таймера, и из-за этого останавливал таймер раньше положенного срока. Проблема больше не возникала после изменения принципа работы таймера программы.

Еще была замечена ошибка, заключающаяся в том, что некорректно выводится время до старта следующего участника, но время старта, а также время участников отображаются корректно. Как и предыдущая ошибка, эта зависит от самого устройства.

Кроме этого была выявлена небольшая ошибка «лишней клавиатуры». При запуске окна таймера появляется клавиатура для ввода ничего, хотя она не должна появляться, и ей там нечего изначально вводить. Единственное решение - это закрытие клавиатуры нажатием кнопки «назад» на устройстве.

Также при очень быстрой остановке участников, примерно 5 человек в секунду, программа может прекратить свою работу.

Для исправления всех этих ошибок требуется более углубленное и детальное изучение работы операционной системы «Android» и технических возможностей мобильных устройств.

# ГЛАВА 4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

## 4.1 Системные требования приложения

Для запуска приложения на мобильном устройстве требуется любое устройство с версией операционной системы «Android 4.1» и выше. Минимальные системные требования для «Android» данной версии:

* 512 MB оперативной памяти;
* Процессор с архитектурой 32-bit ARMv7, MIPS или x86;
* Графический процессор с поддержкой OpenGL ES 2.0.

Для установки приложения на устройство необходимо наличие 10 MB свободного места в памяти устройства.

## 4.2 Руководство системного программиста

Программа реализована для личных мобильных устройств и не нуждается в системном программисте. Для установки программы достаточно перенести установочный «.apk» файл на мобильное устройство и запустить его, предварительно разрешив в настройках устройства установку приложений из неизвестного источника, так как программа не распространяется через официальный магазин приложений «Google Play». При установке программа потребует доступ к записи, хранению и чтению файлов.

## 4.3 Руководство программиста

Для компиляции программы требуется наличие установленной «Xamarin Studio». Для ее установки потребуется предварительно установить «Microsoft Visual Studio 2012/2013». Все остальные необходимые компоненты «Xamarin Studio» скачает и установит сама.

Минимальные системные требования для «Xamarin Studio»:

* Операционная система Microsoft Windows 7 / 8 / 8.1;
* GB оперативной памяти;
* 10 GB свободного места на жестком диске;
* Процессор с тактовой частотой 1.6 GHz и выше;
* Графический процессор с поддержкой OpenGL 2.0

Все основные методы работы с базой данных описаны в файле «WorkWithDB.cs». В таблице 4.1 указаны все методы и их описание.

Таблица 4.1 – Описание методов работы с БД

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование метода | Описание | Входные данные | Выходные данные |
| Getpath(string filename) | Берет заданное имя файла и прописывает путь где файл должен находится. С полученным результатом работают остальные методы. | Имя файла - string |  |
| CreateFile() | Создает файл по указанному пути с таблицей данных. |  |  |
| AddNewLine(string name) | Добавляет в файл новую строку данных с указанным именем. | Наименование участника - string |  |
| DeleteLine(string ID) | Удаляет строку данных из файла с указанным ID. | ID - string |  |
| Read() | Выводит из файла все записи в ListView. |  | Набор данных по созданной модели List<ListView\_Model> |
| ReadFiltered(bool[] a) | Выводит из файла все записи в ListView кроме тех, где id массива соответствует id записи и значение массива = true. | Массив данных с работающими таймерами | Набор данных по созданной модели List<ListView\_Model> |
| Продолжение таблицы 4.1 - Описание методов работы с БД | | | |
| Наименование метода | Описание | Входные данные | Выходные данные |
| UpdateTimer(string ID,string sec,string min) | Обновляет таблицу данных в файле и записывает человеку с указанным ID его время. | ID участника, секунды и минуты - string |  |
| UpdateID() | Обновляет ID участников в таблице данных, для упорядочивания. |  |  |
| IDArray(int idcount, out int[] idarray) | Создает массив всех ID участников. | Кол-во участников - int | Массив со всеми ID участников – int[] |
| OrderByTime() | Метод сортировки записей по времени. |  | Набор данных List<ListView\_Model> |
| OrderByName() | Метод сортировки записей по имени. |  | Набор данных List<ListView\_Model> |
| DeleteFile(string filename) | Удаляет файл по указанному пути. | Путь к файлу - string |  |
| FirstRun() | Создает файл (Records.db) в котором хранятся все пути, наименования и время создания записей (файлов созданных методом CreateFile()) |  |  |
| AddNewRecord(string name) | Добавляет запись о новом файле в Records.db. | Имя файла - string |  |
| UpdateRecordsID() | Обновляет ID файлов в таблице данных, для упорядочивания. |  |  |
| DeleteRecord(string id) | Удаляет файл и запись о нем в Records.db с указанным id. | ID файла - string |  |
| ReadRecords() | Выводит список всех записей о файлах из Records.db в ListView. |  | Набор данных List<ListView\_Model> |
| Продолжение таблицы 4.1 - Описание методов работы с БД | | | |
| Наименование метода | Описание | Входные данные | Выходные данные |
| ReturnPath(string id) | Прописывает путь к файлу с указанным id для дальнейшей работы с ним другими методами. | ID файла - string |  |
| copy() | Копирует выбранный файл |  |  |
| EmptyCheck() | Проверяет есть ли временные данные в выбранном файле |  | bool |
| Zero() | Удаляет все временные данные в файле |  |  |

Для вызова любого метода из любой части программы нужно ввести WorkWithDB.Instance.название\_метода.

Код программы находится в приложении 2, а также проект с программой будет находиться на диске. Так как код программы открытый, разрешается его изучение и доработка, как в плане интерфейса, так и в методах работы с базой данных. Программа делалась в образовательных целях и с конкретно поставленной задачей, и многие методы нуждаются в доработке и соединении в один более крупный метод, для дальнейшего развития программы и ее улучшения.

## 4.4 Руководство оператора

Для запуска программы нужно открыть список установленных приложений на устройстве и кликнуть на соответствующую иконку с названием приложения «Multi Timer», изображенную на рис. 4.1.



Рисунок 4.1 – Иконка программы

После запуска откроется основное окно программы.

При вводе данных в поля для ввода появится встроенная экранная клавиатура. Для того чтобы закрыть клавиатуру ввода данных нужно нажать на устройстве кнопку «Назад». Обычно изображена как стрелочка направленная налево. На каждом устройстве данная кнопка находится по разному и может иметь другой вид!

### 4.4.1 Основное окно

На основном окне программы находятся 3 кнопки «Создать новый таймер», «Просмотр записей» и «Удалить запись». На рис.4.2 изображен интерфейс основного окна.

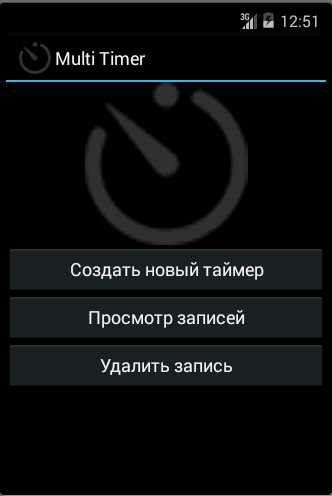


Рисунок 4.2 – Основное окно

При первом запуске программы записей нет и нужно их создать. Для этого нажимаем на кнопку «Создать новый таймер». Откроется окно выбора/создания файла.

При нажатии на кнопки «Просмотр записей» или «Удалить запись» откроется окно просмотра списка файлов.

### 4.4.2 Окно выбора/создания файла

На данном окне, изображенном на рис. 4.3, находятся поле для ввода имени файла и 2 кнопки – «Создать новый файл», «Выбрать файл».

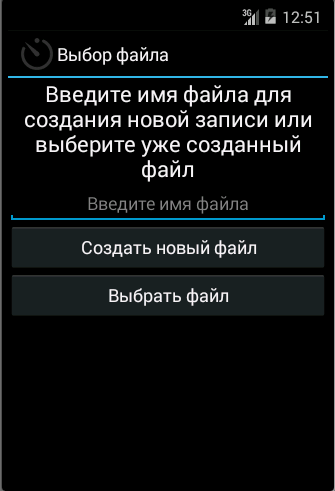


Рисунок 4.3 – Окно выбора/создания файла

Для того, чтобы создать новый файл необходимо ввести уникальное имя в поле для ввода и нажать кнопку «Создать новый файл», после этого откроется окно ввода данных. Если поле для ввода оставить пустым или имя файла не будет уникальным программа выдаст ошибку, как показано на рисунке 4.4.

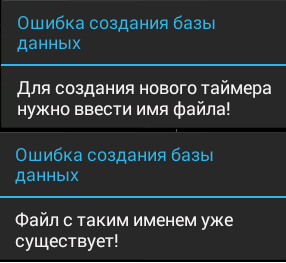
.

Рисунок 4.4 – Ошибки создания файла

Если вы хотите перезаписать старый файл или создали заранее файл для работы с программой, то необходимо нажать на кнопку «Выбрать файл». Откроется окно просмотра списка файлов, где можно выбрать нужный файл.

### 4.4.2 Окно ввода данных

На окне заполнения файла находятся поле для ввода имени участника, 3 кнопки «Добавить», «Продолжить» и «Сохранить и закрыть», а также таблица с добавленными участниками. На рис. 4.5 показано окно ввода данных. Для того, чтобы продолжить работу с программой, нужно добавить минимум 2 участника (максимум 49). Для этого вводим имя участника (по желанию) и нажимаем кнопку добавить, после нажатия на кнопку поле для ввода автоматически очистится. Для удаления ошибочных записей достаточно кликнуть на эту запись в таблице, и появится окно подтверждения удаления записи (см. рис.4.6), при положительном ответе запись будет удалена.

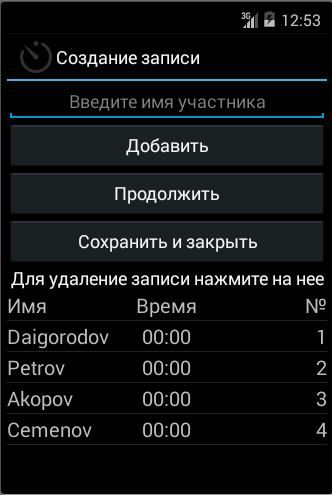


Рисунок 4.5 – Окно ввода данных

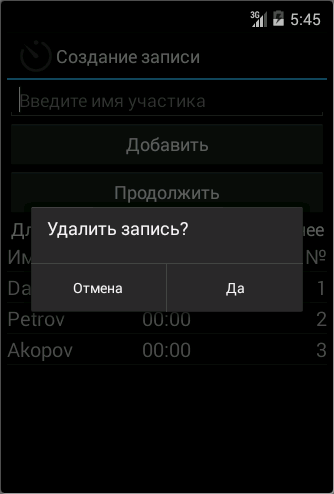


Рисунок 4.6 – Сообщение об удалении записи

После того, как будет добавлено минимальное количество участников можно продолжить работу с программой, для этого нужно нажать на кнопку «Продолжить» и откроется окно работы с таймером, или же нажать на кнопку «Сохранить и закрыть» чтобы сохранить файл для использования в будущем.

### 4.4.3 Окно таймера

На окне работы с таймером находятся 2 кнопки остановки участников «Остановить №» и 2 поля для ввода номера участников, для их остановки. Также существует поле для ввода интервала для таймера и кнопка «Старт». На рис. 4.7 изображено окно таймера, который еще не запущен.

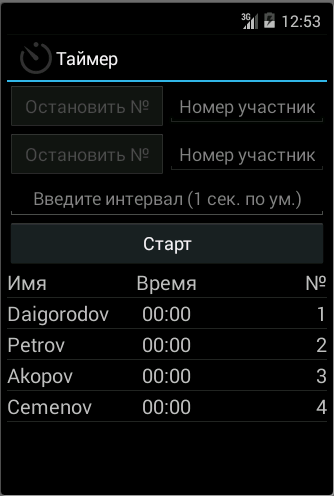


Рисунок 4.7 – Окно таймера (таймер не запущен)

Вводим в соответствующее поле интервал в секундах (по умолчанию 1 секунда) и нажимаем на кнопку «Старт». Таймер для первого участника стартует сразу же после нажатия на кнопку «Старт», для остальных участников таймер будет стартовать через указанный интервал времени в порядке их очереди в таблице (по их номеру). Для остановки участника нужно ввести в одно из полей для остановки участников его порядковый номер и нажать на кнопку слева от этого поля. Для удобства номер участника можно ввести заранее, не останавливая участника. Одновременно можно остановить двух участников благодаря двум кнопкам и полям для ввода. При нажатии на кнопку «Остановить №» поле для ввода номера участника автоматически обнулится. Также участников можно останавливать просто нажимая на них в списке. Так как список обновляется раз в секунду не рекомендуется останавливать в 1 секунду больше трех участников. После того как последний участник будет остановлен откроется окно просмотра результатов/записи. Окно таймера в активном состоянии показано на рис. 4.8.

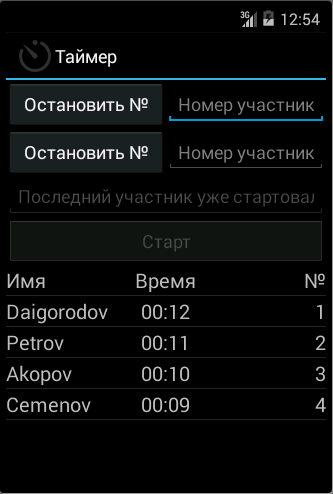


Рисунок 4.8 – Окно таймера (таймер активен)

Таймер может работать в фоновом режиме и программу можно свернуть кнопкой «Домой» на устройстве. Не рекомендуется одновременный запуск нескольких приложений на телефоне при работе таймера так как это может нарушить его работу или остановить.

### 4.4.4 Окно просмотра списка файлов

Окно списка файлов представляет из себя кнопку «Закрыть» и таблицу с созданными файлами. Интерфейс окна изображен на рисунке 4.9.



Рисунок 4.9 – Окно просмотра списка файлов

Если данное окно было вызвано нажатием на кнопку «Просмотр записей» из основного окна, то при нажатии в таблице на файл откроется окно просмотра результатов/записей.

Если окно было вызвано кнопкой «Удалить запись» из основного окна, то при нажатии на запись в таблице появится окно подтверждения удаления, при положительном ответе файл будет удален, и таблица обновится.

Если окно было вызвано из окна выбора/создания файла, то при выборе файла откроется окно работы с таймером с списком участников из данного файла. Время автоматически сбросится до 0 для всех участников, поэтому если файл уже имел временные данные программа выдаст оповещение об этом и попросит подтвердить открытие файла (см. рис. 4.10), при этом будет создана копия файла.

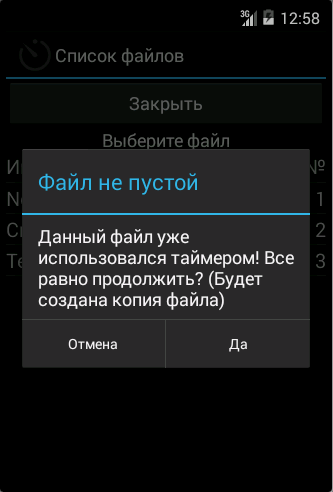


Рисунок 4.10 – Оповещение о файле с данными

При нажатии на кнопку «Закрыть» откроется основное окно.

### 4.4.5 Окно просмотра результатов/записей

На данном окне находятся 2 кнопки сортировки – по имени и по времени, кнопка «Закрыть» и таблица данных. В таблица находятся все участники выбранного файла – их именами, время и порядковый номер. При нажатии на кнопки сортировки таблица обновится в соответствии с выбранной сортировкой. Окно просмотра результатов/записей изображено на рис. 4.11.

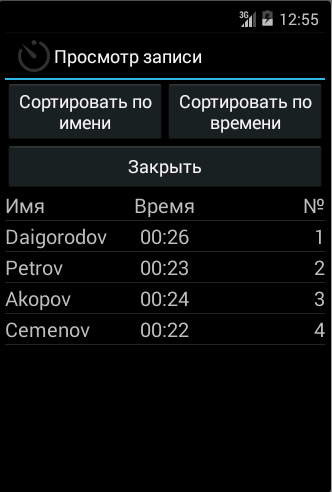


Рисунок 4.11 – Окно просмотра результатов/записей

При нажатии на кнопку «Закрыть» окно будет закрыто и откроется основное окно программы.

# ЗАКЛЮЧЕНИЕ

Целью данной работы было создание мобильного приложения для отсчета временных данных, при проведении каких-либо соревнований по физической культуре, на устройствах с операционной системой «Android». Были преодолены все сложности разработки мобильных приложений и достигнуты все поставленные задачи:

* Изучена новая среда разработки «Xamarin Studio»;
* Изучена мобильная платформа «Android» и ее объекты;
* Была внедрена компактная встраиваемая реляционная база данных – SQLite;
* Изучен язык структурированных запросов «SQL»;
* Изучен расширяемый язык разметки – «XML» для данной платформы;
* Создана своя собственная модель и адаптер для вывода созданных таблиц базы данных на экран.

Программа имеет следующие достоинства и функции:

* Удобный мобильный интерфейс;
* Хранение, чтение и редактирование записей на мобильном устройстве;
* Работа программы в фоновом режиме устройства;
* Вывод результатов и таблиц данных;
* Просмотр всех созданных записей и их удаление;
* Создание файлов участников заранее для использование их таймером в любой другой момент;
* Возможность перезаписывать уже созданные файлы и создание их копий.

Недостатками программы являются:

* Невозможность переносить данные на компьютер;
* Маленький функционал;
* Проблема с просмотром таблицы участников во время работы таймера.

Основной сложностью проекта была новая среда разработки «Xamarin Studio» и сама платформа «Android». Было изучено и решено много проблем, связанных с «окнами» программы и их работой в «фоновом» режиме устройства, работой таймера на разных мобильных устройствах, и выводом данных на экран.

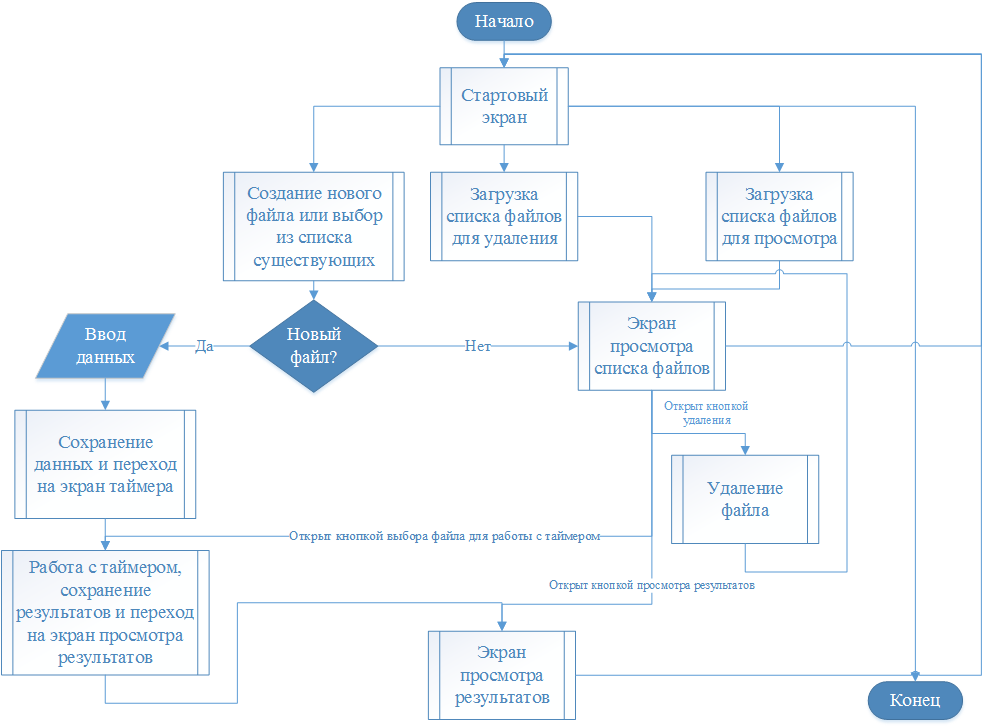
Приложение имеет перспективу в дальнейшей доработке. Первым делом планируется возможность переноса данных на компьютер и возможность их просмотра на нем, также планируется доработка интерфейса и добавление дополнительных функций приложения, таких как обычный секундомер, таймер с обратным отсчетом и тому подобные.

Приложение было передано заказчику на эксплуатацию.

# СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 7.32-2001 Отчет о научно-исследовательской работе;
2. ГОСТ Р 7.0.5-2008 Библиографическая ссылка;
3. Зиборов В.В. Visual C# 2012 на примерах. - СПб.: БХВ-Петербург, 2013;
4. Павловская Т.А. C#. Программирование на языке высокого уровня: учебник для вузов. – СПб.: Питер, 2009;
5. Сахил Малик. Microsoft ADO.NET 2.0 для профессионалов = Pro ADO.NET 2.0. — М.: «Вильямс», 2006. — С. 560.
6. Эндрю Троелсен. Язык Программирования С#2010 и платформа .NET 4.0 = PRO C# 2010 AND THE .NET 4 PLATFORM. — 5-е издание. — Москва • Санкт-Петербург • Киев: «Вильямс», 2011. — С. 1392
7. [developer.android.com](https://developer.android.com/sdk/index.html) — документация по «Android SDK» и разработке приложений под «Android»
8. [developer.xamarin.com](http://developer.xamarin.com/guides/android/) — документация и уроки по работе в среде разработки «Xamarin Studio»
9. [habrahabr.ru](http://habrahabr.ru/post/188130/) — многофункциональный сайт, представляющий собой смешение новостного сайта и коллективного блога, созданный для публикации новостей, аналитических статей, мыслей, связанных с информационными технологиями, бизнесом и Интернетом.
10. [krimel.karelia.ru](http://www.krimel.karelia.ru/file.cgi?id=325) — рекомендации по оформлению рефератов, контрольных, курсовых и дипломных работ (в формате DOC);
11. [msdn.microsoft.com](https://msdn.microsoft.com/) — документация по «.NET» и «C#»;
12. [vsegost.com](http://vsegost.com/Catalog/27/2737.shtml) — межгосударственный стандарт 7.32—2001 на структуру и оформление отчета о научно-исследовательской работе.
13. [w3schools.com](http://www.w3schools.com/) — сайт с учебниками и примерами для web разработчиков на темы «HTML», «XML», «CSS», «JavaScript», «PHP», «SQL» и «JQuery».

# ПРИЛОЖЕНИЕ 1. БЛОК-СХЕМА



# ПРИЛОЖЕНИЕ 2. КОД ПРОГРАММЫ

## MainActivity.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Multi Timer", MainLauncher = true, Icon = "@drawable/Icon")]  
    public class MainActivity : Activity  
    {  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
  
            *// Основной экран*  
            SetContentView (Resource.Layout.Main);  
            *// Создание объектов*  
  
            Button button\_create     =     FindViewById<Button> (Resource.Id.button\_create);  
            Button button\_view         =     FindViewById<Button> (Resource.Id.button\_view);  
            Button button\_delete     =     FindViewById<Button> (Resource.Id.button\_delete);  
            *//EditText edittext\_bdname = FindViewById<EditText> (Resource.Id.editText\_bdname);*  
            *// Создание объектов*  
            WorkWithDB.Instance.FirstRun();  
            button\_create.Click += delegate {  
                StartActivity (typeof(Load\_CreateFile));  
            };  
  
            button\_view.Click += delegate {  
  
                var activity = new Intent (this, typeof(FileBrowser));  
                activity.PutExtra("type","1");  
                StartActivity (activity);  
  
            };  
  
            button\_delete.Click += delegate {  
  
                var activity = new Intent (this, typeof(FileBrowser));  
                activity.PutExtra("type","0");  
                StartActivity (activity);  
            };  
        }  
    }  
}

## DB\_Create.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Создание записи")]              
    public class DB\_Create : Activity  
    {  
  
  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
  
            SetContentView (Resource.Layout.DB\_Create);  
  
            *// Создание объектов*  
            EditText edittext1 = FindViewById<EditText> (Resource.Id.editText\_add);  
            Button button\_add = FindViewById<Button> (Resource.Id.button\_add);  
            Button button\_next = FindViewById<Button> (Resource.Id.button\_next);  
            Button button\_save = FindViewById<Button> (Resource.Id.button\_save);  
            ListView listview\_create = FindViewById<ListView> (Resource.Id.listView\_create);  
            *// Создание объектов*  
            var data = new List<ListView\_Model>{};  
            data = WorkWithDB.Instance.Read ();  
            RunOnUiThread (() => listview\_create.Adapter = new ListView\_Adapter (this, data)); *// Заполнение listview*  
            button\_add.Click += delegate *// Нажатие на кнопку "Добавить"*  
            {   
                WorkWithDB.Instance.AddNewLine(edittext1.Text);  
                data = WorkWithDB.Instance.Read ();  
                RunOnUiThread (() => listview\_create.Adapter = new ListView\_Adapter (this, data));  
                edittext1.Text = "";  
  
            };  
  
            button\_save.Click += delegate *// Нажатие на кнопку "Добавить"*  
            {   
                if(data.Count <= 2)  
                {  
                    var builder = new AlertDialog.Builder (this);  
                    builder.SetMessage ("Для продолжения должно быть минимум 2 участника!");  
                    builder.SetTitle ("Слишком мало участников");  
                    builder.Create ().Show ();  
                }  
                else  
                {  
  
                    if(data.Count > 50)  
                    {  
                        var builder = new AlertDialog.Builder (this);  
                        builder.SetMessage ("Максимальное кол-во участников 49!");  
                        builder.SetTitle ("Слишком много участников");  
                        builder.Create ().Show ();  
                    }  
                    else  
                    {  
                        Finish();  
                    }  
                }  
            };  
  
            listview\_create.ItemClick += (object sender, Android.Widget.AdapterView.ItemClickEventArgs e) => *// Клик на строку в listview*  
            {   
                var builder = new AlertDialog.Builder (this); *// Создание сообщения оповещения*  
                if ((int)listview\_create.GetItemIdAtPosition (e.Position) != 0) {  
                    builder.SetMessage ("Удалить запись?");  
                    builder.SetPositiveButton ("Да", (s, ee) => { *// Действия при нажатие на "Да"*  
                        var t = data[e.Position];  
                        WorkWithDB.Instance.DeleteLine(t.Right);  
                        WorkWithDB.Instance.UpdateID();  
                        data = WorkWithDB.Instance.Read ();  
                        RunOnUiThread (() => listview\_create.Adapter = new ListView\_Adapter (this, data)); *// Заполнение адаптера новой бд*  
                    });  
                    builder.SetNegativeButton ("Отмена", (s, ee) => { *// Действия при нажатии на "Отмена"*  
                        */\* do something on Cancel click \*/*   
                    });  
                    builder.Create ().Show (); *// Создание сообщения оповещения*  
                }  
            };  
  
  
  
            button\_next.Click += delegate *// "Продолжить"*  
            {           
                if(data.Count <= 2)  
                {  
                    var builder = new AlertDialog.Builder (this);  
                    builder.SetMessage ("Для продолжения должно быть минимум 2 участника!");  
                    builder.SetTitle ("Слишком мало участников");  
                    builder.Create ().Show ();  
                }  
                else  
                {  
  
                    if(data.Count > 50)  
                    {  
                        var builder = new AlertDialog.Builder (this);  
                        builder.SetMessage ("Максимальное кол-во участников 49!");  
                        builder.SetTitle ("Слишком много участников");  
                        builder.Create ().Show ();  
                    }  
                    else  
                    {  
                        var Timer\_activity = new Intent (this, typeof(Timer));  
                        StartActivity (typeof(Timer));  
                        Finish();  
                    }  
                }  
            };  
  
        }  
    }}

## Timer.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Таймер")]  
    public class Timer : Activity  
    {  
        System.Timers.Timer timer1 = new System.Timers.Timer ();  
  
        int[] sec = new int[50];  
        int[] min = new int[50];  
        bool[] stopped = new bool[50];  
        int peoplenumber = 1;  
        int peoplecount = 0;  
        int interval = 1;  
        int intervalcount = 1;  
        int[] idarray;  
        List<ListView\_Model> data = new List<ListView\_Model>{};  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
            SetContentView (Resource.Layout.Timer);  
            *// Создание объектов*  
            Button button\_stop1 = FindViewById<Button> (Resource.Id.button\_stop1);  
            EditText edittext\_stop1 = FindViewById<EditText> (Resource.Id.editText\_stop1);  
            Button button\_stop2 = FindViewById<Button> (Resource.Id.button\_stop2);  
            EditText edittext\_stop2 = FindViewById<EditText> (Resource.Id.editText\_stop2);  
            EditText edittext\_interval = FindViewById<EditText> (Resource.Id.editText\_interval);  
            Button button\_start = FindViewById<Button> (Resource.Id.button\_start);  
            ListView listview\_timer = FindViewById<ListView> (Resource.Id.listView\_timer);  
            timer1.Interval = 1000;   
            timer1.Elapsed += OnTimedEvent;  
            timer1.Enabled = false;  
            button\_stop1.Enabled = false;  
            button\_stop2.Enabled = false;  
            edittext\_stop1.Enabled = false;  
            edittext\_stop2.Enabled = false;  
            *// Создание объектов*  
            WorkWithDB.Instance.UpdateID ();  
            WorkWithDB.Instance.Zero ();  
            data = WorkWithDB.Instance.Read ();  
            RunOnUiThread (() => listview\_timer.Adapter = new ListView\_Adapter (this, data));  
            var timer = new System.Timers.Timer();  
            timer.Interval = 1000;   
            timer.Enabled = false;  
            timer.Elapsed += delegate {  
                RunOnUiThread (() => listview\_timer.Adapter = new ListView\_Adapter (this, data));  
                if(data.Count == 1)  
                {  
                    timer.Enabled = false;  
                    timer1.Enabled = false;  
                    StartActivity (typeof(DB\_View));  
                    Finish();  
                }  
            };  
  
  
            WorkWithDB.Instance.IDArray(data.Count-1,out idarray);  
            peoplecount = data.Count - 1;  
            for (int i = 0; i < 50; i++)   
            {  
                stopped [i] = true;      
            }  
            for (int i = 0; i < data.Count-1; i++)   
            {  
                stopped [i] = false;      
            }  
            button\_start.Click += delegate {  
                if (edittext\_interval.Text != "") {  
                    if (Convert.ToInt32 (edittext\_interval.Text) <= 0) {  
                        var builder2 = new AlertDialog.Builder (this);  
                        builder2.SetMessage ("Интервал не может быть меньше или равен 0!");  
                        builder2.SetTitle ("Ошибка интервала");  
                        builder2.Create ().Show ();  
                    } else {  
                        if (Convert.ToInt32 (edittext\_interval.Text) >= 1)  
                            interval = Convert.ToInt32 (edittext\_interval.Text);  
                        intervalcount = interval;  
                        edittext\_interval.Enabled = false;  
                        button\_start.Enabled = false;  
                        timer1.Enabled = true;  
                        timer.Enabled = true;  
                        button\_stop1.Enabled = true;  
                        button\_stop2.Enabled = true;  
                        edittext\_stop1.Enabled = true;  
                        edittext\_stop2.Enabled = true;  
  
  
                    }  
                } else {  
                    intervalcount = interval;  
                    edittext\_interval.Enabled = false;  
                    button\_start.Enabled = false;  
                    timer1.Enabled = true;  
                    timer.Enabled = true;  
                    button\_stop1.Enabled = true;  
                    button\_stop2.Enabled = true;  
                    edittext\_stop1.Enabled = true;  
                    edittext\_stop2.Enabled = true;  
  
                }  
            };  
  
                button\_stop1.Click += delegate {  
                if( edittext\_stop1.Text != "" && Convert.ToInt32(edittext\_stop1.Text) != 0)  
                    {  
                    try  
                    {  
                        stopped[Convert.ToInt32(edittext\_stop1.Text)-1] = true;  
                    }  
                    catch{}  
                    RunOnUiThread (() => listview\_timer.Adapter = new ListView\_Adapter (this, data));  
  
  
                }  
                edittext\_stop1.Text="";  
  
                };  
  
                button\_stop2.Click += delegate {  
                if( edittext\_stop2.Text != "" && Convert.ToInt32(edittext\_stop2.Text) != 0)  
                {  
                    try  
                    {  
                        stopped[Convert.ToInt32(edittext\_stop1.Text)-1] = true;  
                    }  
                    catch{}  
                    RunOnUiThread (() => listview\_timer.Adapter = new ListView\_Adapter (this, data));  
  
                }  
                edittext\_stop2.Text="";  
  
  
            };  
  
            listview\_timer.ItemClick += (object sender, Android.Widget.AdapterView.ItemClickEventArgs e) => *// Клик на строку в listview*  
            {   
                if ((int)listview\_timer.GetItemIdAtPosition (e.Position) != 0) {  
                    var t = data[e.Position];  
                    Console.WriteLine("CHECK!!!!");  
                    Console.WriteLine(listview\_timer.GetItemIdAtPosition(e.Position).ToString());  
                    Console.WriteLine(t.Right.ToString());  
                    stopped[Convert.ToInt32(t.Right)-1] = true;  
                    RunOnUiThread (() => listview\_timer.Adapter = new ListView\_Adapter (this, data));  
  
                }  
            };  
                  
  
        }  
  
        protected override void OnDestroy()  
        {  
            base.OnDestroy();   
            timer1.Enabled = false;  
        }  
  
  
  
        private void OnTimedEvent(object sender, System.Timers.ElapsedEventArgs e)  
        {  
            EditText edittext\_interval = FindViewById<EditText> (Resource.Id.editText\_interval);  
            data = WorkWithDB.Instance.ReadFiltered (stopped);  
            if (intervalcount == 0) {  
                RunOnUiThread (() => edittext\_interval.Text = "Старт!");  
            }   
            if (intervalcount < 0) {  
                RunOnUiThread (() => edittext\_interval.Text = "Последний участник уже стартовал");  
            }  
            if (intervalcount > 0) {  
                RunOnUiThread (() => edittext\_interval.Text = intervalcount.ToString ());  
            }  
  
            if (intervalcount <= 0)   
            {  
                if (peoplenumber < peoplecount)   
                {  
                    peoplenumber++;  
                    intervalcount = interval;  
                }   
  
                else   
                {  
                    intervalcount = 0;  
                }  
            }  
  
            for (int i = 0; i < peoplenumber; i++) {  
                if (stopped [i]) {  
                } else {  
                    sec [i]++;  
                    if (sec [i] == 60) {  
                        min [i]++;  
                        sec [i] = 0;  
                    }  
                    WorkWithDB.Instance.UpdateTimer (idarray [i].ToString (), sec [i].ToString (), min [i].ToString ());  
                }  
            }  
  
  
  
            intervalcount--;  
  
  
        }  
    }  
}

## DB\_View.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Просмотр записи")]              
    public class DB\_View : Activity  
    {  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
            SetContentView (Resource.Layout.DB\_View);  
            Button button\_sort\_time = FindViewById<Button> (Resource.Id.button\_sort\_time);  
            Button button\_sort\_name = FindViewById<Button> (Resource.Id.button\_sort\_name);  
            Button button\_close = FindViewById<Button> (Resource.Id.button\_close);  
            ListView listview\_view = FindViewById<ListView> (Resource.Id.listView\_view);  
            var data = new List<ListView\_Model>{};  
            data = WorkWithDB.Instance.Read ();  
            RunOnUiThread (() => listview\_view.Adapter = new ListView\_Adapter (this, data));  
            button\_sort\_name.Click += delegate {  
                data = WorkWithDB.Instance.OrderByName();  
                RunOnUiThread (() => listview\_view.Adapter = new ListView\_Adapter (this, data));  
            };  
  
            button\_sort\_time.Click += delegate {  
                data = WorkWithDB.Instance.OrderByTime();  
                RunOnUiThread (() => listview\_view.Adapter = new ListView\_Adapter (this, data));  
            };  
  
            button\_close.Click += delegate {  
                Finish();  
            };  
  
        }  
    }  
}

## FileBrowser.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Список файлов")]              
    public class FileBrowser : Activity  
    {  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
            string type = Intent.GetStringExtra ("type");  
            SetContentView (Resource.Layout.File\_Browser);  
            ListView lv\_files     =     FindViewById<ListView> (Resource.Id.listView\_view);  
            Button close     =     FindViewById<Button> (Resource.Id.button\_viewclose);  
            var data = new List<ListView\_Model>{};  
            data = WorkWithDB.Instance.ReadRecords ();  
            RunOnUiThread (() => lv\_files.Adapter = new ListView\_Adapter (this, data));  
            bool yes = false;  
  
            lv\_files.ItemClick += delegate(object sender, Android.Widget.AdapterView.ItemClickEventArgs e) {  
                if ((int)lv\_files.GetItemIdAtPosition (e.Position) != 0)   
                {  
                    if(type == "2")  
                    {  
                        var t = data[e.Position];  
                        WorkWithDB.Instance.ReturnPath(t.Right);  
                        if(WorkWithDB.Instance.EmptyCheck())  
                        {  
                            StartActivity (typeof(Timer));  
                            Finish();  
                        }  
                        else  
                        {  
                            var builder = new AlertDialog.Builder (this); *// Создание сообщения оповещения*  
                            builder.SetMessage ("Данный файл уже использовался таймером! Все равно продолжить? (Будет создана копия файла)");  
                            builder.SetTitle ("Файл не пустой");  
                            builder.SetPositiveButton ("Да", (s, ee) => { *// Действия при нажатие на "Да"*  
                                WorkWithDB.Instance.copy();  
                                StartActivity (typeof(Timer));  
                                Finish();  
                            });  
                            builder.SetNegativeButton ("Отмена", (s, ee) => { *// Действия при нажатии на "Отмена"*  
                                  
                            });  
                            builder.Create ().Show ();  
                            Console.WriteLine(yes.ToString());  
                        }  
  
                    }  
                    if(type == "1")  
                    {  
                        var t = data[e.Position];  
                        WorkWithDB.Instance.ReturnPath(t.Right);  
                        var view\_activity = new Intent (this, typeof(DB\_View));  
                        StartActivity (typeof(DB\_View));  
                    }  
                    if(type == "0")  
                    {  
                        var builder = new AlertDialog.Builder (this); *// Создание сообщения оповещения*  
                        builder.SetMessage ("Удалить запись?");  
                        builder.SetPositiveButton ("Да", (s, ee) => { *// Действия при нажатие на "Да"*  
                            var t = data[e.Position];  
                            WorkWithDB.Instance.DeleteRecord(t.Right);  
                            var builder2 = new AlertDialog.Builder (this);  
                            builder2.SetMessage ("Запись удалена!");  
                            builder2.SetTitle ("Файл удален");  
                            builder2.Create ().Show ();  
                            data = WorkWithDB.Instance.ReadRecords();  
                            RunOnUiThread (() => lv\_files.Adapter = new ListView\_Adapter (this, data));   
                        });  
                        builder.SetNegativeButton ("Отмена", (s, ee) => { *// Действия при нажатии на "Отмена"*  
                            */\* do something on Cancel click \*/*   
                        });  
                        builder.Create ().Show ();  
                    }  
                }  
            };  
  
            close.Click += delegate {  
                Finish();  
            };  
  
  
  
              
        }  
    }  
}

## ListView\_Adapter.cs

using System;  
using Android.App;  
using Android.Content;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Android.OS;  
using System.Collections.Generic;  
  
namespace Multi\_Timer  
{  
    public class ListView\_Adapter : BaseAdapter<ListView\_Model>  
    {  
        private readonly IList<ListView\_Model> \_items;  
        private readonly Context \_context;  
  
        public ListView\_Adapter(Context context, IList<ListView\_Model> items)  
        {  
            \_items = items;  
            \_context = context;  
        }  
  
        public override long GetItemId(int position)  
        {  
  
            return position;  
        }  
  
  
  
        public override View GetView(int position, View convertView, ViewGroup parent)  
        {  
            var item = \_items[position];  
            var view = convertView;  
  
            if (view == null)  
            {  
                var inflater = LayoutInflater.FromContext(\_context);  
                view = inflater.Inflate(Resource.Layout.row, parent, false);  
            }  
  
            view.FindViewById<TextView>(Resource.Id.left).Text = item.Left;  
            view.FindViewById<TextView>(Resource.Id.mid).Text = item.Mid;  
            view.FindViewById<TextView>(Resource.Id.right).Text = item.Right;  
  
            return view;  
        }  
  
  
        public override int Count  
        {  
            get { return \_items.Count; }  
        }  
  
        public override ListView\_Model this[int position]  
        {  
            get { return \_items[position]; }  
        }  
  
  
    }  
}

## ListView\_Model.cs

using System;  
  
namespace Multi\_Timer  
{  
    public class ListView\_Model  
    {  
        public string Left { get; set; }  
        public string Mid { get; set; }  
        public string Right { get; set; }  
    }  
}

## WorkWithDB.cs

using System;  
using Android.App;  
using Android.Content;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Android.OS;  
using System.Collections.Generic;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    public class WorkWithDB  
    {  
        public static SqliteConnection connection;  
        string pathToDatabase = "";  
        static readonly WorkWithDB instance = new WorkWithDB();  
        public static WorkWithDB Instance  
        {  
            get  
            {  
                return instance;  
            }  
        }  
  
        public void getpath(string filename)  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            var directoryname = Path.Combine (docsFolder, "MultiTimerDB");  
            if(!Directory.Exists (directoryname)) Directory.CreateDirectory(directoryname);  
            pathToDatabase = System.IO.Path.Combine (docsFolder, filename);  
        }  
  
        public void CreateFile()  
        {  
            SqliteConnection.CreateFile (pathToDatabase);  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
  
            var command = "CREATE TABLE [People] (\_ID INTEGER PRIMARY KEY AUTOINCREMENT, Name ntext, Time ntext);";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
        }  
  
        public void AddNewLine(string name)  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
  
            var command = "INSERT INTO [People] ([Name], [Time]) VALUES ('" + name + "', '00:00');";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
        }  
  
        public void DeleteLine(string ID)  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
  
            var command = "DELETE FROM [People] WHERE \_ID =" + ID + ";";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
        }  
  
        public List<ListView\_Model> Read()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] from [People]";  
                var r = contents.ExecuteReader ();  
                var data = new List<ListView\_Model>{ new ListView\_Model{ Left = "Имя", Mid = "Время", Right = "№", } };  
                while (r.Read ())   
                {  
                    data.Add (new ListView\_Model { Left = r["Name"].ToString(), Mid = r["Time"].ToString(), Right = r["\_ID"].ToString(), });  
                }  
                connection.Close ();  
                return data;  
            }  
  
        }  
  
        public void Zero()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] from [People]";  
                var r = contents.ExecuteReader ();  
                while (r.Read ())   
                {  
                    var command = "UPDATE [People] SET Time='00:00' WHERE \_ID =" + r["\_ID"].ToString() + ";";  
  
                    using (var c = connection.CreateCommand ())   
                    {  
                        c.CommandText = command;  
                        var rowcount = c.ExecuteNonQuery ();  
                    }  
                }  
                connection.Close ();  
            }  
        }  
  
        public bool EmptyCheck()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            bool check = true;  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] from [People] WHERE \_ID='1'";  
                var r = contents.ExecuteReader ();  
                while (r.Read ())   
                {  
                    if (r ["Time"].ToString () != "00:00") check = false;  
                }  
                connection.Close ();  
                return check;  
            }  
  
        }  
  
        public List<ListView\_Model> ReadFiltered(bool[] a)  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] from [People]";  
                var r = contents.ExecuteReader ();  
                var data = new List<ListView\_Model>{ new ListView\_Model{ Left = "Имя", Mid = "Время", Right = "№", } };  
                while (r.Read ())   
                {  
                    if(a[Convert.ToInt32(r["\_ID"])-1] == false) data.Add (new ListView\_Model { Left = r["Name"].ToString(), Mid = r["Time"].ToString(), Right = r["\_ID"].ToString(), });  
                }  
                connection.Close ();  
                return data;  
            }  
  
        }  
  
  
        public void UpdateTimer(string ID,string sec,string min)  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            if (Convert.ToInt32 (sec) < 10)  
                sec = "0" + sec;  
            if (Convert.ToInt32 (min) < 10)  
                min = "0" + min;  
            var command = "UPDATE [People] SET Time='" + min + ":" + sec + "' WHERE \_ID =" + ID + ";";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
        }  
  
        public void UpdateID()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
  
            var command = "CREATE TABLE [People2] (\_ID INTEGER PRIMARY KEY AUTOINCREMENT, Name ntext, Time ntext);";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
            command = "INSERT INTO [People2] ([Name], [Time]) SELECT [Name], [Time] FROM [People];";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            command = "DROP TABLE [People];";  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            command = "ALTER TABLE [People2] RENAME TO People";  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
        }  
  
        public void IDArray(int idcount, out int[] idarray)  
        {  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID] from [People]";  
                var r = contents.ExecuteReader ();  
                int count = 0;  
                idarray = new int[idcount];  
                while (r.Read ())   
                {      
                    idarray [count] = Convert.ToInt32(r["\_ID"]);  
                    count++;  
                }  
                connection.Close ();  
            }  
        }  
  
        public List<ListView\_Model> OrderByTime()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] FROM [People] ORDER BY Time ASC;";  
                var r = contents.ExecuteReader ();  
                var data = new List<ListView\_Model>{ new ListView\_Model{ Left = "Имя", Mid = "Время", Right = "№", } };  
                while (r.Read ())   
                {  
                    data.Add (new ListView\_Model { Left = r["Name"].ToString(), Mid = r["Time"].ToString(), Right = r["\_ID"].ToString(), });  
                }  
                connection.Close ();  
                return data;  
            }  
        }  
  
        public List<ListView\_Model> OrderByName()  
        {  
            connection = new SqliteConnection ("Data Source=" + pathToDatabase);  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] FROM [People] ORDER BY Name ASC;";  
                var r = contents.ExecuteReader ();  
                var data = new List<ListView\_Model>{ new ListView\_Model{ Left = "Имя", Mid = "Время", Right = "№", } };  
                while (r.Read ())   
                {  
                    data.Add (new ListView\_Model { Left = r["Name"].ToString(), Mid = r["Time"].ToString(), Right = r["\_ID"].ToString(), });  
                }  
                connection.Close ();  
                return data;  
            }  
        }  
  
        public void DeleteFile(string filename)  
        {  
            File.Delete (filename);  
        }  
  
        public void FirstRun()  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            var directoryname = Path.Combine (docsFolder, "MultiTimerDB");  
            if(!Directory.Exists (directoryname)) Directory.CreateDirectory(directoryname);  
            directoryname = Path.Combine (docsFolder, "Records");  
            if(!Directory.Exists (directoryname)) Directory.CreateDirectory(directoryname);  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            if (!File.Exists (path)) {  
                SqliteConnection.CreateFile (path);  
                connection = new SqliteConnection ("Data Source=" + path);  
  
                var command = "CREATE TABLE [Records] (\_ID INTEGER PRIMARY KEY AUTOINCREMENT, Name ntext, Time ntext, Path ntext);";  
  
                connection.Open ();  
  
                using (var c = connection.CreateCommand ()) {  
                    c.CommandText = command;  
                    var rowcount = c.ExecuteNonQuery ();  
                }  
                connection.Close ();  
            }  
        }  
  
        public void AddNewRecord(string name)  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            var command = "INSERT INTO [Records] ([Name], [Time], [Path]) VALUES ('" + name + "', '"+ DateTime.Now.ToString("dd-MM HH:mm") + "', '" + pathToDatabase + "');";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
        }  
  
        public void UpdateRecordsID()  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            var command = "CREATE TABLE [Records2] (\_ID INTEGER PRIMARY KEY AUTOINCREMENT, Name ntext, Time ntext, Path ntext);";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
            command = "INSERT INTO [Records2] ([Name], [Time], [Path]) SELECT [Name], [Time], [Path] FROM [Records];";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            command = "DROP TABLE [Records];";  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            command = "ALTER TABLE [Records2] RENAME TO Records";  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
        }  
  
        public void DeleteRecord(string id)  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT Path FROM [Records] WHERE \_ID =" + id + ";";  
                var r = contents.ExecuteReader ();  
                while (r.Read ())   
                {      
                    DeleteFile(r["Path"].ToString());  
                }  
                connection.Close ();  
            }  
  
            var command = "DELETE FROM [Records] WHERE \_ID =" + id + ";";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
  
            UpdateRecordsID ();  
  
        }  
  
        public List<ListView\_Model> ReadRecords()  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT [\_ID], [Name], [Time] from [Records]";  
                var r = contents.ExecuteReader ();  
                var data = new List<ListView\_Model>{ new ListView\_Model{ Left = "Имя файла", Mid = "Время создания", Right = "№", } };  
                while (r.Read ())   
                {  
                    data.Add (new ListView\_Model { Left = r["Name"].ToString(), Mid = r["Time"].ToString(), Right = r["\_ID"].ToString(), });  
                }  
                connection.Close ();  
                return data;  
            }  
  
        }  
  
        public void ReturnPath(string id)  
        {  
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            connection.Open ();  
            using (var contents = connection.CreateCommand ())   
            {  
                contents.CommandText = "SELECT Path FROM [Records] WHERE \_ID =" + id + ";";  
                var r = contents.ExecuteReader ();  
                while (r.Read ())   
                {      
                    pathToDatabase = r["Path"].ToString();  
                }  
                connection.Close ();  
            }  
        }  
  
        public void copy()  
        {      
            var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
            string path2 = System.IO.Path.Combine (docsFolder, Path.GetFileName(pathToDatabase) + "\_Копия");  
            for(int i=1;i<100;i++)  
            {  
                path2 = System.IO.Path.Combine (docsFolder, Path.GetFileName(pathToDatabase) + "\_Копия\_" + i);  
                if (!File.Exists (path2)) {  
                    path2 = System.IO.Path.Combine (docsFolder, Path.GetFileName (pathToDatabase) + "\_Копия\_" + i);  
                    break;  
                }  
            }  
            File.Copy (pathToDatabase, path2);  
  
            Path.Combine (docsFolder, "MultiTimerDB");  
            Path.Combine (docsFolder, "Records");  
            string path = System.IO.Path.Combine (docsFolder, "Records.db");  
            connection = new SqliteConnection ("Data Source=" + path);  
  
            var command = "INSERT INTO [Records] ([Name], [Time], [Path]) VALUES ('" + Path.GetFileName(path2) + "', '"+ DateTime.Now.ToString("dd-MM HH:mm") + "', '" + path2 + "');";  
  
            connection.Open ();  
  
            using (var c = connection.CreateCommand ())   
            {  
                c.CommandText = command;  
                var rowcount = c.ExecuteNonQuery ();  
            }  
            connection.Close ();  
        }  
  
  
    }  
}

## Load\_CreateFile.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using Android.App;  
using Android.Content;  
using Android.OS;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Mono.Data.Sqlite;  
using System.IO;  
  
namespace Multi\_Timer  
{  
    [Activity (Label = "Выбор файла")]              
    public class Load\_CreateFile : Activity  
    {  
        protected override void OnCreate (Bundle bundle)  
        {  
            base.OnCreate (bundle);  
            SetContentView (Resource.Layout.Load\_CreateFile);  
            Button button\_createnew     =     FindViewById<Button> (Resource.Id.button\_createnew);  
            Button button\_loadFile         =     FindViewById<Button> (Resource.Id.button\_old);  
            EditText edittext\_bdname = FindViewById<EditText> (Resource.Id.editText\_bdname);  
  
            button\_createnew.Click += delegate {  
                if (edittext\_bdname.Text == "") {  
                    var builder2 = new AlertDialog.Builder (this);  
                    builder2.SetMessage ("Для создания нового таймера нужно ввести имя файла!");  
                    builder2.SetTitle ("Ошибка создания базы данных");  
                    builder2.Create ().Show ();  
                } else {  
                    string filename = edittext\_bdname.Text;  
                    var docsFolder = System.Environment.GetFolderPath (System.Environment.SpecialFolder.MyDocuments);  
                    var directoryname = Path.Combine (docsFolder, "MultiTimerDB");  
                    if(!Directory.Exists (directoryname)) Directory.CreateDirectory(directoryname);  
                    string pathToDatabase = System.IO.Path.Combine (docsFolder, filename);  
                    if (File.Exists (pathToDatabase)) {  
                        var builder2 = new AlertDialog.Builder (this);  
                        builder2.SetMessage ("Файл с таким именем уже существует!");  
                        builder2.SetTitle ("Ошибка создания базы данных");  
                        builder2.Create ().Show ();  
                    } else {  
                        WorkWithDB.Instance.getpath(filename);  
                        WorkWithDB.Instance.CreateFile();  
                        WorkWithDB.Instance.AddNewRecord(filename);  
                        StartActivity (typeof(DB\_Create));  
                        Finish();  
                    }  
                }  
            };  
  
            button\_loadFile.Click += delegate {  
                var activity = new Intent (this, typeof(FileBrowser));  
                activity.PutExtra("type","2");  
                StartActivity (activity);  
                Finish();  
            };  
  
  
        }  
    }  
}

## DB\_Create.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <EditText  
        android:inputType="textPersonName"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/editText\_add"  
        android:hint="Введите имя участника"  
        android:gravity="center" />  
    <Button  
        android:text="Добавить"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_add" />  
    <Button  
        android:text="Продолжить"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_next" />  
    <Button  
        android:text="Сохранить и закрыть"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_save" />  
    <TextView  
        android:text="Для удаление записи нажмите на нее"  
        android:textAppearance="?android:attr/textAppearanceMedium"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/textView1"  
        android:gravity="center" />  
    <ListView  
        android:id="@+id/listView\_create"  
        android:layout\_width="fill\_parent"  
        android:layout\_height="fill\_parent"  
        android:layout\_marginTop="0.0dp"  
        android:minWidth="100dp" />  
</LinearLayout>

## DB\_View.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <LinearLayout  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:gravity="center">  
        <Button  
            android:text="Сортировать по имени"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/button\_sort\_name" />  
        <Button  
            android:text="Сортировать по времени"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/button\_sort\_time" />  
    </LinearLayout>  
    <Button  
        android:text="Закрыть"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_close" />  
    <ListView  
        android:id="@+id/listView\_view"  
        android:layout\_width="fill\_parent"  
        android:layout\_height="fill\_parent"  
        android:layout\_marginTop="0.0dp"  
        android:minWidth="100dp" />  
</LinearLayout>

## File\_Browser.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <Button  
        android:text="Закрыть"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_viewclose" />  
    <TextView  
        android:text="Выберите файл"  
        android:textAppearance="?android:attr/textAppearanceMedium"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/textView1"  
        android:gravity="center" />  
    <ListView  
        android:id="@+id/listView\_view"  
        android:layout\_width="fill\_parent"  
        android:layout\_height="fill\_parent"  
        android:layout\_marginTop="0.0dp"  
        android:minWidth="100dp" />  
</LinearLayout>

## Load\_CreateFile.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <TextView  
        android:text="Введите имя файла для создания новой записи или выберите уже созданный файл"  
        android:textAppearance="?android:attr/textAppearanceLarge"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/textView2"  
        android:gravity="center" />  
    <EditText  
        android:inputType="textPersonName"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/editText\_bdname"  
        android:hint="Введите имя файла"  
        android:gravity="center" />  
    <Button  
        android:text="Создать новый файл"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_createnew" />  
    <Button  
        android:text="Выбрать файл"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_old" />  
</LinearLayout>

## Main.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <ImageView  
        android:src="@drawable/icon"  
        android:layout\_width="match\_parent"  
        android:layout\_height="163.2dp"  
        android:id="@+id/imageView1" />  
    <Button  
        android:text="Создать новый таймер"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_create" />  
    <Button  
        android:text="Просмотр записей"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_view" />  
    <Button  
        android:text="Удалить запись"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_delete" />  
</LinearLayout>

## row.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="horizontal"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="wrap\_content">  
    <TextView  
        android:id="@+id/left"  
        android:layout\_height="wrap\_content"  
        android:layout\_width="0px"  
        android:layout\_weight="1"  
        android:text="left"  
        android:minHeight="30dp"  
        android:textSize="20dp" />  
    <TextView  
        android:id="@+id/mid"  
        android:layout\_height="wrap\_content"  
        android:layout\_width="0px"  
        android:layout\_weight="1"  
        android:text="middle"  
        android:height="30dp"  
        android:textSize="20dp"  
        android:gravity="center" />  
    <TextView  
        android:id="@+id/right"  
        android:layout\_height="wrap\_content"  
        android:layout\_width="0px"  
        android:layout\_weight="1"  
        android:text="right"  
        android:height="30dp"  
        android:textSize="20dp"  
        android:gravity="right" />  
</LinearLayout>

## Timer.axml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent">  
    <LinearLayout  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:gravity="center">  
        <Button  
            android:text="Остановить №"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/button\_stop1" />  
        <EditText  
            android:inputType="number"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/editText\_stop1"  
            android:hint="Номер участника" />  
    </LinearLayout>  
    <LinearLayout  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:gravity="center">  
        <Button  
            android:text="Остановить №"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/button\_stop2" />  
        <EditText  
            android:inputType="number"  
            android:layout\_height="wrap\_content"  
            android:layout\_width="0px"  
            android:layout\_weight="1"  
            android:id="@+id/editText\_stop2"  
            android:hint="Номер участника" />  
    </LinearLayout>  
    <EditText  
        android:inputType="number"  
        android:id="@+id/editText\_interval"  
        android:layout\_width="match\_parent"  
        android:layout\_height="40.8dp"  
        android:hint="Введите интервал (1 сек. по ум.)"  
        android:gravity="center" />  
    <Button  
        android:text="Старт"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:id="@+id/button\_start" />  
    <ListView  
        android:id="@+id/listView\_timer"  
        android:layout\_width="fill\_parent"  
        android:layout\_height="fill\_parent"  
        android:layout\_marginTop="0.0dp"  
        android:minWidth="100dp" />  
</LinearLayout>