МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«PassManager»

Выполнил студент Самолазов Герман Русланович

(Ф.И.О.)

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

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

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

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

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

Курсовой проект защищен с оценкой

Минск 2020

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc59073297)

[**1 Обзор аналогов приложений «PassManager»** 4](#_Toc59073298)

[**1.1** **Приложение «SafeInCloud»** 4](#_Toc59073299)

[**1.2 Приложение «LastPass»** 5](#_Toc59073300)

[**1.4 Выводы по главе «Обзор аналогов приложений «PassManager»»** 6](#_Toc59073301)

[**2 Проектирование программного продукта** 7](#_Toc59073302)

[**2.1 Проектирование мобильного приложения** 7](#_Toc59073303)

[**2.2 Проектирование базы данных** 8](#_Toc59073304)

[**3 Программная реализация приложений** 9](#_Toc59073305)

[**3.1 Технические средства для разработки** 9](#_Toc59073306)

[**3.2 Разработка базы данных** 9](#_Toc59073307)

[**3.3 Разработка мобильного приложения** 9](#_Toc59073308)

[**3.5 Вывод по главе «Программная реализация приложений»** 13](#_Toc59073309)

[**4 Обеспечение безопасности приложения** 14](#_Toc59073310)

[**4.1 Генерация надежных паролей** 14](#_Toc59073311)

[**4.2 Защита базы данных** 15](#_Toc59073312)

[**4.3 Запрет на запись и снимки экрана** 15](#_Toc59073313)

[**4.4 Биометрическая аутентификация** 15](#_Toc59073314)

[**4.4 Выводы по главе «Обеспечение безопасности приложения»** 15](#_Toc59073315)

[**5 Руководство пользователя** 16](#_Toc59073316)

[**5.1 Выводы по главе «Руководство пользователя»** 19](#_Toc59073317)

[**ЗАКЛЮЧЕНИЕ** 20](#_Toc59073318)

[**Список использованных источников** 21](#_Toc59073319)

[**ПРИЛОЖЕНИЕ А** 22](#_Toc59073320)

[**ПРИЛОЖЕНИЕ Б** 23](#_Toc59073321)

[**ПРИЛОЖЕНИЕ В** 24](#_Toc59073322)

# **ВВЕДЕНИЕ**

Цель курсового проекта: удобное и безопасное хранение аккаунтов и паролей к ним пользователя, а также генерировать надежные пароли.

Задачи курсового проекта:

1. Изучить аналогичные программные средства по теме курсового проекта.

Данная задача позволит нам изучить похожие программные средства для анализа, где пользователям может быть неудобно в использовании приложения. Также определить, чем новое приложение будет отличатся от уже имеющихся аналогов.

1. Спроектировать приложение.

Проектирование приложения позволит нам увидеть прозрачную картину предстоящей работы целиком и разрабатывать приложение с четким пониманием что, зачем и для кого делается, а также сколько это займет времени.

1. Реализовать приложение.

В курсовом проекте будет реализовано мобильное приложение для хранения и генерации паролей. В век развития информационных технологий этот вариант будет более оптимальный и удобный, так как вся информация будет хранится в одном приложении на телефоне, и пользователь легко сможет им воспользоваться, где бы он не был.

1. Сделать приложение безопасным.

Это один из важных этапов в процессе создания продукта. Не уделив ему должного внимания, нельзя добиться высокого качества на выходе. Безопасность необходима во всех приложениях, особенно в приложениях, оперирующих пользовательскими конфиденциальными данными.

1. Разработать руководство пользователя.

В рамках работы данного курсового проекта будет разработано руководство пользователя, призванное помочь пользователю разобраться с работой программы и её основными функциями. Важно разработать руководство пользователя, даже если вы уверены, что с вашим продуктом легко работать, а его интерфейс интуитивно понятен – всегда найдутся пользователи, которые любят читать инструкции, прежде чем начать освоение продукта.

# **1 Обзор аналогов приложений «PassManager»**

# **Приложение «SafeInCloud»**

Приложение позволяет хранить и генерировать пароли, обеспечена безопасность и возможность подстановки данных в поля сайтов в браузере.

Приложение предоставлено скриншоте ниже (рис 1.1.1).

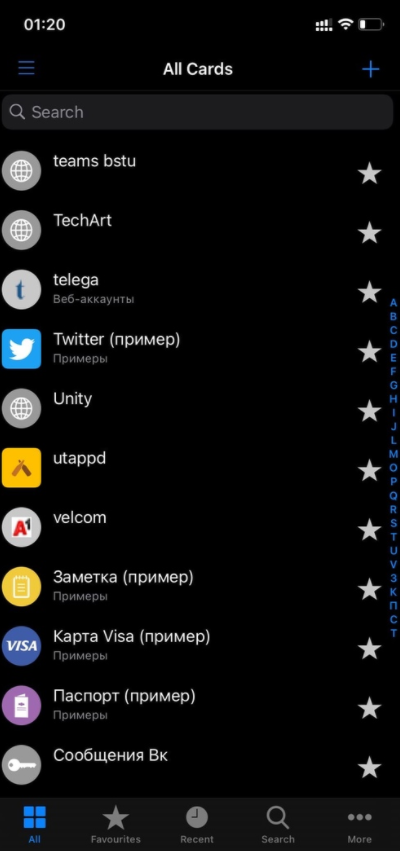


Рисунок 1.1.1 –Приложение «SafeInCloud»

Позволяет выбирать «любимые» пароли, показывает предыдущие и позволяет искать по списку паролей. Минусы приложения: перегруженность интерфейса, ненадежная генерация паролей, отсутствует удобная биометрическая аутентификация.

# **1.2 Приложение «LastPass»**

Популярное приложение хранения паролей и аккаунтов.

Общее представление о приложении можно получить, посмотрев скриншот (рис 1.2.1).

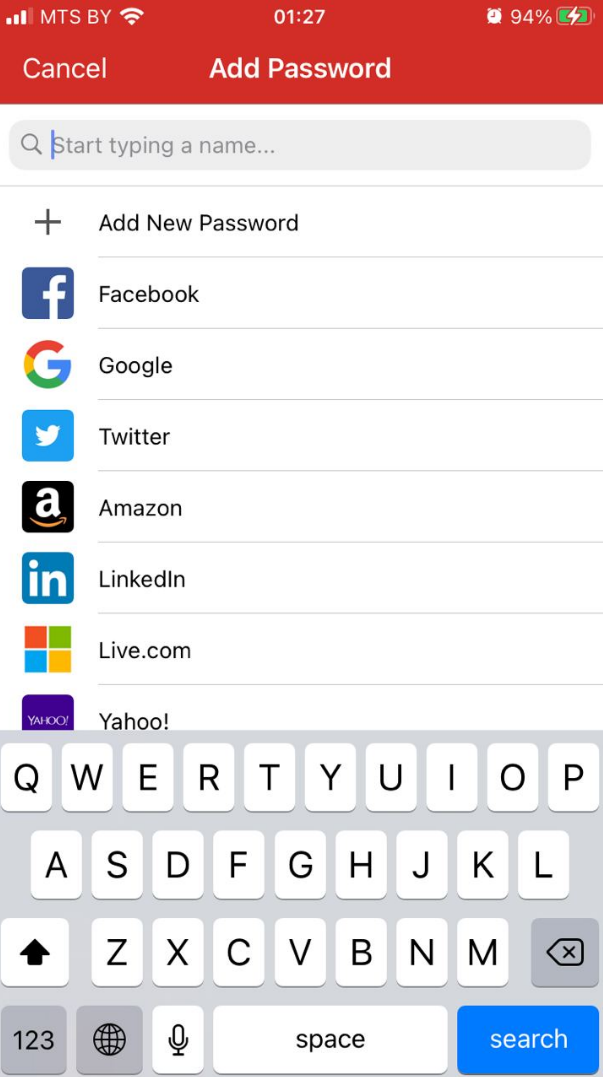


Рисунок 1.2.1 – Приложение «LastPass»

Данное приложение полностью бесплатно. Имеет возможности поиска, сортировки паролей, подстановки на сайты в браузере и другие приложения, установленные на мобильном устройстве. Минус приложения: отсутствие удобной биометрической аутентификации.

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

В данной главе были изучены два приложения, которые являются аналогами приложения «SafeIncCloud», «LastPass». Были рассмотрены и оценены преимущества и недостатки приложений, проведен анализ, какие функции наиболее необходимы для удобного и безопасного использования приложения, а от каких стоит отказаться в пользу улучшения пользовательского опыта.

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать мобильное приложение позволяющего пользователю добавлять, изменять и просматривать свои аккаунты и генерировать пароли.

На странице логина пользователь имеет возможность аутентифицироваться с помощью отпечатка пальца или пароля, который задается при первом запуске приложения.

На главной странице приложение должно позволить пользователю выбрать: просмотреть аккаунт из имеющегося списка или добавить его.

Если пользователь выбирает просмотр определенного аккаунта, то ему отображаются данные по аккаунту (название, логин, пароль, описание), а также имеется возможность редактировать или удалить аккаунт.

Если выбирается создание нового аккаунта, то пользователю предлагается заполнить поля (название, логин, пароль, описание), сгенерировать надежный пароль и сохранить данные.

Таким образом приложение должно выполнять функции:

* биометрическая аутентификация;
* создание пароля для входа в приложение;
* аутентификация по паролю;
* предоставление списка аккаунтов;
* хранение аккаунтов;
* просмотр аккаунта;
* добавление аккаунта;
* редактирование аккаунта;
* удаление аккаунта;
* генерация надежного пароля.

# **2.2 Проектирование базы данных**

В базе данных данного приложения должны быть реализованы 2 таблицы.

В таблице «Accounts» содержится информация о аккаунтах. Столбцы данной таблицы:

* Id – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* Name – строковый тип данных, максимальной длинной 100 символов;
* Login – строковый тип данных, максимальной длинной 100 символов;
* Password – строковый тип данных, максимальной длинной 100 символов;
* Description – строковый тип данных, максимальной длинной 100 символов.

В таблице «UserPassword» хранится заданный пользователем при первом запуске приложения пароль. Имеет один столбец:

* Password – строковый тип данных, максимальной длинной 50 символов, является первичным ключом;

**2.4 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование разрабатываемого приложения.

Был определен основной функционал приложения и спроектирована база данных с её таблицами, определены столбцы, их типы данных, длина, ключи и связи.

# **3 Программная реализация приложений**

В данной главе будут рассмотрены средства, необходимые для разработки приложения и базы данных. Описаны пакеты и содержащиеся в них классы, методы, их назначение, а также важнейшие свойства классов, где это необходимо.

# **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Java – строго типизированный объектно-ориентированный язык программирования.

SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.

# **3.2 Разработка базы данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite, а для обеспечения безопасности будет использована библиотека SQLCipher.

Код создания мобильной базы данных представлен в Приложении Б.

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio 3.0. Для написания приложения был использован язык программирования Java.

При разработке использовалась трехуровневая архитектура «Репозиторий-Сервис-Представление». Данный подход позволяет инкапсулировать обращение к базе данных, произвести необходимые вызовы для шифрования базы данных только в тех местах, где это нужно, а также сделать приложение наиболее гибким и возможным к расширениям разного рода.

Часть приложения «репозиторий» отвечает за работу с базой данных, описывая методы создания базы данных, таблиц, столбцов. Другая часть приложения «серсис» предоставляет методы, реализующие бизнес-логику, которая, в некоторых случаях, обращается к «репозиторию», когда необходима работа с данными и их хранением. Часть приложения «представление» отвечает за вывод и корректную обработку данных на экране пользователя. «Представление» никак не связано с «репозиторием», что позволит, при необходимости, заменять эти элементы другими, реализующими такой же функционал.

Графическая структура проекта представлена на рисунке 3.3.1.

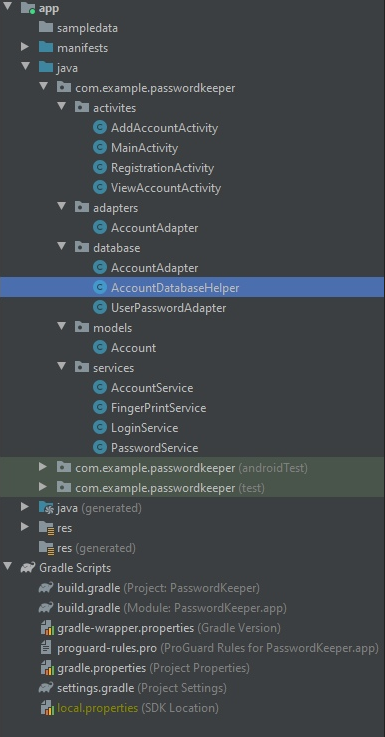


Рисунок 3.3.1 – Графическая структура проекта

Пакет activity хранит в себе классы активностей, которые позволяют добавить новых объекты в базу данных. вариантом (см. таблицу 3.3.1).

**Таблица 3.3.1 – Содержание пакета activity**

|  |  |
| --- | --- |
| Класс AddAccountActivity – добавление нового акканта | |
| GeneratePasswordClick | Метод, привязанный к событию нажатия на кнопку «Generate Password». |
| AddAccount | Метод, привязанный к событию нажатия на кнопку «Save». |
| Класс MainActivity – просмотр списка аккаунтов | |
| addButtonClick | Метод, привязанный к событию нажатия на кнопку «Add». |
| ShowAccountList | Метод заполнения списка имеющимися аккаунтами. |

Пакет Adapters хранит в себе класс, определяющий адаптер для отображения списка аккаунтов (см. таблицу 3.3.2).

**Таблица 3.3.2 – Содержание пакета AccountAdapter**

|  |  |
| --- | --- |
| Класс AccountAdapter – адаптер для отображения списка аккаунтов | |
| onClick | Метод, обрабатывающий нажатие на элемент списка аккаунтов. |
| getView | Метод для отображения списка аккаунтов. |

Пакет database хранит в себе классы для работы приложения с базой данных (см. таблицу 3.3.3).

**Таблица 3.3.3 – Содержание пакета database**

|  |  |
| --- | --- |
| Класс AccountDatabaseHelper – управления базой данных | |
| onCreate | Метод создания базы данных |
| onUpgrade | Метод вызываемый при смени версии базы данных |
| onConfigure | Метод вызываемый для конфигурации базы данных |
| getReadableDatabase | Метод, открывающий соединение с базой данных с помощью пароля, только для чтения |
| getWritableDatabase | Метод, открывающий соединение с базой данных с помощью пароля, только для чтения и записи |
| Класс AccountAdapter – предоставляет методы для обращения к таблице аккаунтов | |
| open | Метод, открывающий соединение с базой данных. |
| close | Метод, закрывающий соединение с базой данных. |
| insert | Метод, добавляющий аккаунт в базу данных. |
| delete | Метод, удаляющий аккаунт из базы данных. |
| update | Метод, добавляющий отредактированный аккаунт в базу данных. |
| getAllEntities | Метод для получения всех аккаунтов. |
| getAccount | Метод для получения конкретного аккаунта. |
| Класс UserPasswordAdapter – предоставляет методы для обращения к таблице пароля | |
| open | Метод, открывающий соединение с базой данных. |
| close | Метод, закрывающий соединение с базой данных. |
| get | Метод, получающий пароль. |
| passwordExist | Метод, проверяющий наличие пароля в базе данных. |
| checkPass | Метод, проверяющий соответствие введенного пароля и пароля в базе данных. |

Пакет Models хранит в себе класс для описания структуры аккаунта (см. таблицу 3.3.4).

**Таблица 3.3.4 – Содержание пакета Models**

|  |  |
| --- | --- |
| Класс Account – описание структуры аккаунта. | |
| Id | Свойство, указывающее идентификатор аккаунта. |
| Name | Свойство, указывающее название аккаунта. |
| Login | Свойство, указывающее логин аккаунта. |
| Password | Свойство, указывающее пароль от аккаунта. |
| Description | Свойство, указывающее описание к аккаунту. |
| getId | Метод, возвращающий идентификатор аккаунта. |
| getName | Метод, возвращающий логин аккаунта. |
| getLogin | Метод, возвращающий логин аккаунта. |
| getPassword | Метод, возвращающий пароль от аккаунта. |
| getDescription | Метод, возвращающий описание к аккаунту. |

Пакет Services хранит в себе классы, реализующие бизнес-логику (см. таблицу 3.3.5).

**Таблица 3.3.5 – Содержание пакета services**

|  |  |
| --- | --- |
| Класс AccountService – логика работы с аккаунтом. | |
| getAccounts | Метод получения аккаунтов. |
| createAccount | Метод создания аккаунта. |
| updateAccount | Метод редактирования аккаунта. |
| deleteAccount | Метод удаления аккаунта |
| Класс FingerPrintMaster – логика аутентификации по отпечатку пальца | |
| auth | Метод, проверяющий отпечаток пальца и дающий возможность захода в приложении при положительной проверке. При отрицательной показывает ошибку. |
| Класс LoginService – логика аутентификации по паролю | |
| addPassword | Метод добавления пароля. |
| checkPass | Метод сравнения пароля с хранящимся в базе данных |
| isPasswordExist | Метод проверки существования пароля в базе данных. |
| Класс PasswordService – логика генерации пароля | |
| generatePassword | Метод генерации пароля |
| getRandomChar | Метод получения случайного символа. Необходим для генерации пароля. |

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

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

# **4 Обеспечение безопасности приложения**

Данное приложение направлено на хранение данных, которые не должны быть получены иными лицами, кроме владельца, поэтому необходимо тщательно подойти к обеспечению безопасности приложения.

Также приложение обеспечивает безопасность аккаунтов путем генерации надежного пароля для последующего использования.

# **4.1 Генерация надежных паролей**

Класс «PasswordService» в пакете «Services» обеспечивает генерацию надежного пароля. Надежность пароля определяется его длинной, разностью символов в нем и использовании специальных символов. Для генерации пароля был разработан алгоритм, продемонстрированный в приложении В.

Описание разработанного алгоритма по шагам:

1. Генерируется массив символьного типа случайной длины от 10 до 15 элементов.
2. В цикле массив заполняется случайными символами из таблицы ASCII с номерами элементов от 93 до 126 путем генерации случайного целочисленного числа в этом диапазоне и приведение его к символьному типу (что в языке Java возвращает символ из таблицы ASCII с соответствующим символом).
3. В цикле, который совершает 4 итерации с каждой итерацией берется случайное число от 0 до 10 (позиции элементов в массиве, которые точно будут присутствовать, вне зависимости от случайно сгенерированной длинны), после чего данная позиция в массиве перезаписывается на символ из функции «GetRandomChar», в которую передается число (номер итерации) и в зависимости от числа с помощью конструкции «switch-case» возвращается символ из одного из наборов символов. Наборы основаны на таблице ASCII и хранят в себе значения индексов символов, например специальных или чисел. В каждой итерации проверяется, не была ли использована прошлая позиция для подстановки символа. Если была, то генерируется новая, пока не будет получена незатронутая этим циклом позиция.

После выполнения всех шагов алгоритма получается строка длиной от 10 до 15 случайных символов и имеющая в первых 10 позициях четыре символа, из которых одно является числом, одно большой буквой и два специальными символами. Второй цикл сделан для того, чтобы если (хоть и с низким шансом) в первом цикле сгенерируется простой пароль, то есть без специальных символов, разных регистров и цифр, усложнить начальный пароль и точно сделать его сложным для взлома и перебора.

# **4.2 Защита базы данных**

Для защиты базы данных была использована библиотека SQLCipher, которая используется для шифрования баз данных SQLite. Для того, чтобы воспользоваться данной библиотекой были проведены следующие шаги:

1. Добавление в файл «build.gragle» проекта зависимости «net.zetetic:android-database-sqlcipher:4.4.2@aar».
2. Замена импортированных классов в местах обращения к базе данных на классы, предоставляемые библиотекой.
3. В начало каждого метода обращения к базе данных добавлена строка «SQliteDatabase.loadlibs(context)», необходимая для работы библиотеки.

После выполнения описанных шагов база данных начинает шифроваться с помощью библиотеки SQLCipher,

# **4.3 Запрет на запись и снимки экрана**

Для предотвращения возможных ошибок самого пользователя, а именно записи или снимков экранов приложения, содержащие конфиденциальные данные был реализован запрет на эти действия. Листинг реализации на одной из активити приложения представлен в приложении Г.

# **4.4 Биометрическая аутентификация**

Для удобства пользователя реализована возможность входа в приложение при помощи отпечатка пальца. Код для реализации биометрической аутентификации был предоставлен документацией по платформе android [1]. Для работы биометрической аутентификации необходимо иметь версию Android API не ниже 16.

# **4.4 Выводы по главе «Обеспечение безопасности приложения»**

В приложении были реализованы генерация паролей, шифрование базы данных, запрет на снимок и запись экрана, биометрическая аутентификация.

Этих мер достаточно для защиты данных пользователя и если он не будет совершать ошибок, основанных на человеческом факторе, то его данные, включая пароли, будут в безопасности.

# **5 Руководство пользователя**

При входе в приложение пользователь должен ввести пароль(при первом входе его необходимо установить) или пройти аутентификацию по отпечатку пальца (Рисунок 5.1.1).

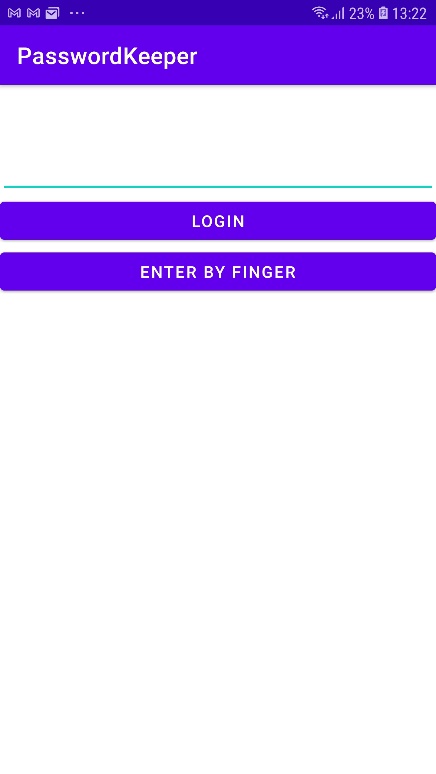


Рисунок 5.1.1 – Начальный экран

При неудачном входе по отпечатку пальца пользователь получает сообщение (Рисунок 5.1.2).

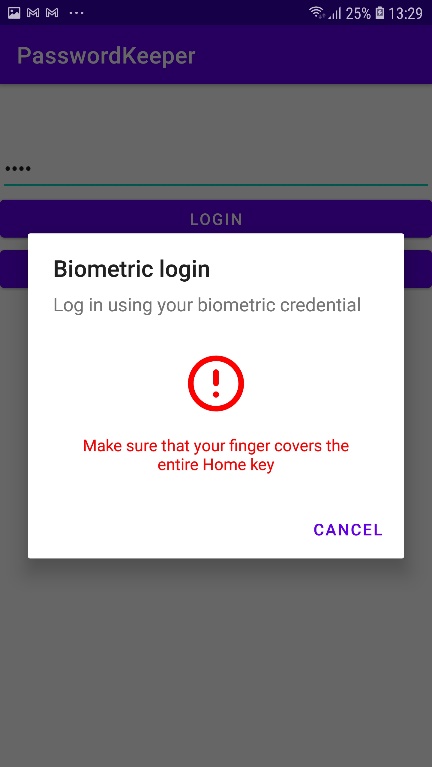


Рисунок 5.1.2 – Неудачная аутентификация.

После входа в приложение пользователь видит список аккаунтов, имеет возможность тапнуть по одному из них для просмотра деталей аккаунта. Также имеется кнопка для добавления нового аккаунта (Рисунок 5.1.3).

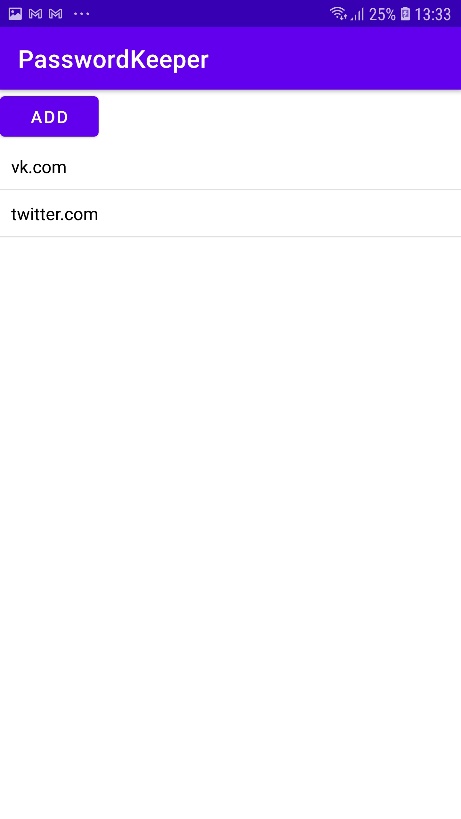


Рисунок 5.1.3 – Список аккаунтов.

При нажатии на один из аккаунтов откроются его детали, где пользователь может скопировать значение любого поля путем тапа по нему, редактировать или удалить аккаунт (Рисунок 5.1.4).

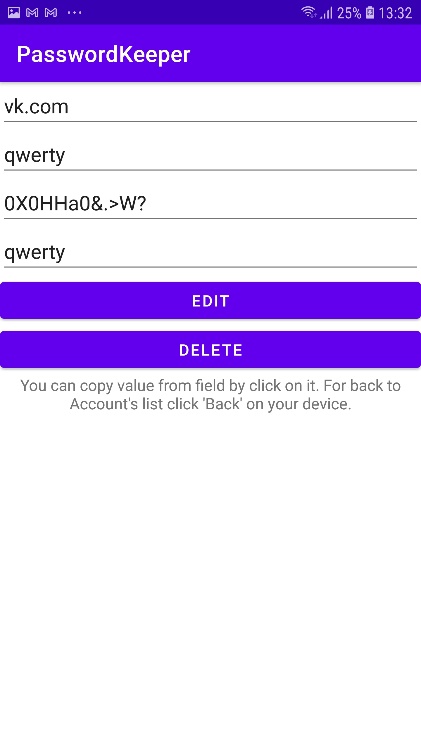


Рисунок 5.1.4 – Детали аккаунта.

При нажатии на кнопку редактирования появляется возможность изменять поля, сгенерировать новый пароль, после чего сохранить отредактированный аккаунт (Рисунок 5.1.5).

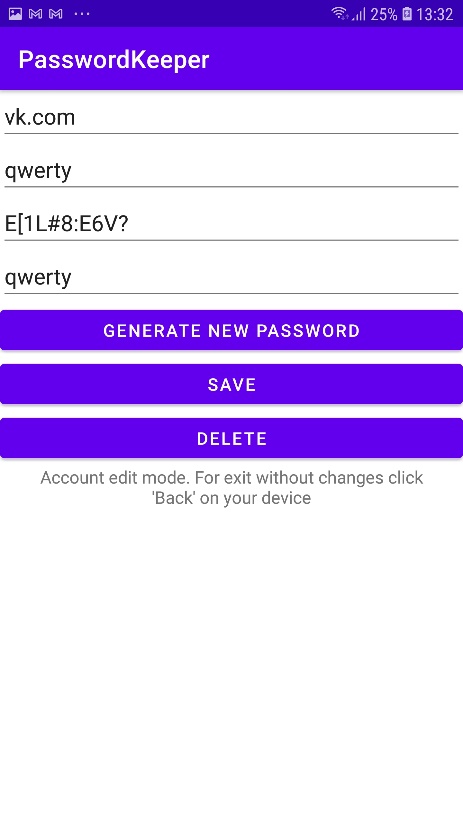


Рисунок 5.1.5 – Редактирование аккаунта.

При удалении аккаунта пользователь получает соответствующее уведомление (Рисунок 5.1.6).

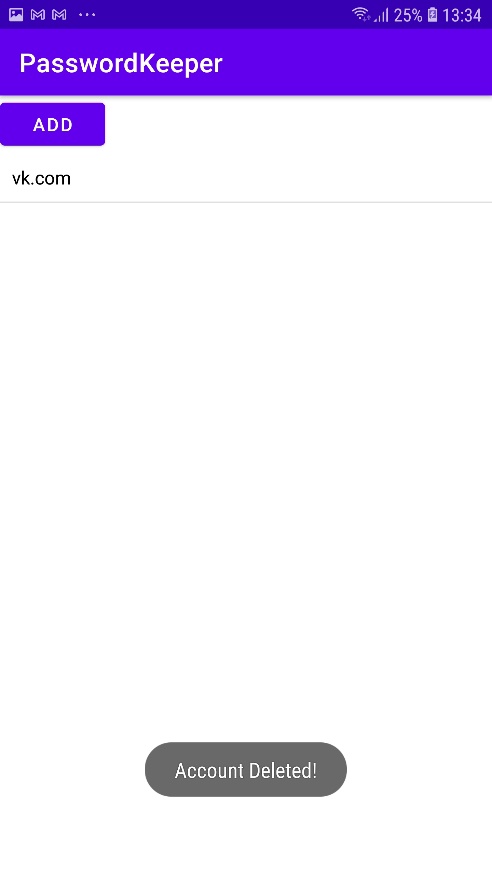


Рисунок 5.1.6 – Удаление аккаунта

При нажатии на экране со списком аккаунтов кнопки добавления аккаунта появляется экран с полями, необходимыми для заполнения и возможностью генерации пароля (Рисунок 5.1.7).

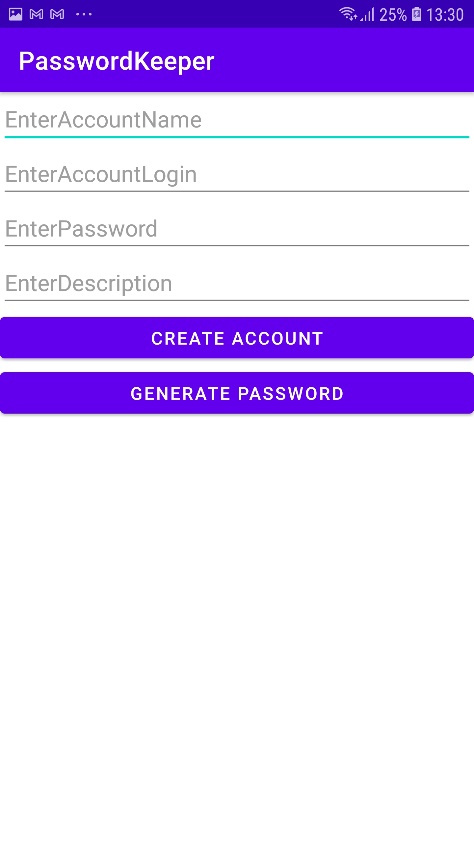


Рисунок 5.1.7 – Создание аккаунта.

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

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью курсового проекта была разработка приложения, которое позволит пользователю безопасно хранить данные своих аккаунтов, а также удобно ими манипулировать.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В мобильном приложении был определен основной функционал, также были разработаны необходимые диаграммы. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQLite, сред разработки Android Studio, языка программирования java, библиотеки SQLCipher.

Была реализована база данных.

При решении данной задачи была достигнута поставленная цель, и был создан проект «PassManager».

* безопасность приложения.

Для реализации данной задачи разработан алгоритм генерации пароля, реализована биометрическая аутентификация, шифрование базы данных и запрет на съемку и запись экрана.

* разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

– аутентификация PrintFinger;

* добавление новых аккаунтов;
* просмотр и удобное копирование полей из аккаунта;
* изменение существующих аккаунтов;
* удаление аккаунта;
* генерация безопасного пароля.

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что задачи были выполнены в полном объеме.

# **Список использованных источников**

1 Архитектура REST [Электронный ресурс]. / habrahabr.ru – Режим доступа: https://habrahabr.ru/post/38730 – Дата доступа: 29.09.2020.

1. Программирование под ОС Андроид [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/java/android. Дата доступа: 06.11.2020
2. Android Adapters Tutorial [Электронный ресурс] – codinginflow.com – Режим доступа: https://codinginflow.com/tutorials/android/countdowntimer/part-4-time-input. – Дата доступа: 23.10.2020
3. Руководство по SQLiteCipher [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sql/cipher. Дата доступа: 23.11.2020
4. Руководство по SQLlite [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sql/sqlite. Дата доступа: 21.11.2020

**ПРИЛОЖЕНИЕ А**

Код AddAccountActivity

@Override

protected void onCreate(Bundle savedInstanceState) {

SQLiteDatabase.loadLibs(this);

super.onCreate(savedInstanceState);

accountService = new AccountService(getApplicationContext());

setContentView(R.layout.activity\_add\_account);

nameTextView = findViewById(R.id.newAccountNameTextView);

loginTextView = findViewById(R.id.newAccountLoginTextView);

passTextView = findViewById(R.id.newAccountPasswordEditText);

descriptionTextView = findViewById(R.id.newAccountDescriptionEditText);

}

public void GeneratePasswordClick(View view) {

String pass = PasswordService.GetGeneratedPassword();

passTextView.setText(pass);

}

public void AddAccount(View view) {

String name = nameTextView.getText().toString();

String login = loginTextView.getText().toString();

String pass = passTextView.getText().toString();

String description = descriptionTextView.getText().toString();

Account newAccount = new Account(name, login, pass, description);

accountService.CreateAccount(newAccount);

Toast.makeText(this, "Account Successfully Created", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(this, MainActivity.class);

}

}

**ПРИЛОЖЕНИЕ Б**

Код создания структуры мобильной базы данных

import net.sqlcipher.database.SQLiteDatabase;

public class AccountDatabaseHelper extends SQLiteOpenHelper {

private static final String DATABASE\_NAME = "main.db";

private static final int SCHEMA = 1;

static final String TABLE\_ACCOUNTS = "accounts";

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

public static final String COLUMN\_NAME = "name";

public static final String COLUMN\_LOGIN = "login";

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

public static final String COLUMN\_DESCRIPTION = "description";

static final String TABLE\_USER\_PASSWORD = "main";

public static final String COLUMN\_USER\_PASSWORD = "password";

public AccountDatabaseHelper(Context context) {

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

}

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL("CREATE TABLE " + TABLE\_ACCOUNTS + "(" + COLUMN\_ID

+ " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN\_NAME

+ " TEXT, " + COLUMN\_LOGIN + " TEXT, " + COLUMN\_USER\_PASSWORD + " TEXT," + COLUMN\_DESCRIPTION + " TEXT);");

db.execSQL("CREATE TABLE " + TABLE\_USER\_PASSWORD + " (" + COLUMN\_USER\_PASSWORD

+ " TEXT);");

}

@Override

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

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

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

onCreate(db);

**ПРИЛОЖЕНИЕ В**

Генерация пароля

public class PasswordService {

public static String GetGeneratedPassword() {

return GeneratePassword();

}

private static String GeneratePassword() {

Random random = new Random();

int utfCount = random.nextInt(5) + 10;

char[] utfNumbers = new char[utfCount];

for (int i = 0; i < utfCount; i++) {

utfNumbers[i] = (char) (random.nextInt(93) + 33);

}

int temp = 0;

for (int i = 0; i < 4; i++) {

int randomPosition = random.nextInt(10);

while (temp == randomPosition) {

randomPosition = random.nextInt(10);}

temp = randomPosition;

utfNumbers[randomPosition] = (char) GetRandomChar(i);}

return new String(utfNumbers);}

private static int GetRandomChar(int a) {

Random random = new Random();

switch (a) {

case 0:

return random.nextInt(14) + 33;

case 1:

return random.nextInt(9) + 48;

case 2:

return random.nextInt(6) + 58;

case 3:

return random.nextInt(25) + 65;

}

return 0;

}

}