МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева»
(Самарский университет)

Институт естественнонаучный
Факультет механико-математический
Кафедра безопасности информационных систем

КУРСОВАЯ РАБОТА

ПРОЕКТИРОВАНИЕ *ANDROID*-ПРИЛОЖЕНИЯ ДЛЯ ШИФРОВАНИЯ ДАННЫХ

по специальности 10.05.01 Компьютерная безопасность
(уровень специалитета)

Обучающийся 6442-100501*D* гр. _____ Н. С. Дурасов

Руководитель КР
к.ф.-м.н., _____А. Н. Крутов

Нормоконтролер _____/_____/

Самара 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мобильные устройства в современном мире являются не просто средством связи. Рынок мобильных устройств растет очень быстро и уже давно обогнал рынок персональных компьютеров. Каждое новое мобильное устройство улучшает свои возможности и вычислительные мощи. Исходя из этого, люди всё чаще задумываются об обеспечении информационной безопасности.

Это устройства, которые хранят уязвимые персональные данные, несанкционированный доступ к которым может привести к непредсказуемым последствиям.

Актуальность данной темы можно обосновать тем, что даже самые новейшие модели мобильных телефонов и популярные приложения, направленные на защиту данных владельца устройства, не способны обеспечить безопасность данных владельца в должном виде. Появляется необходимость в проектировании совершенного приложения для защиты данных.

Цель данной работы – проектирование мобильного приложения на базе операционной системы *Android*, которое предоставит пользователю устройства гарантию, что все персональные данные будут под защитой этого приложения.

Для достижения поставленной цели были поставлены следующие задачи:

– дать анализ существующим аналогам приложений;

– изучить особенности проектирования и разработки мобильного приложения для *ОС Android*;

– спроектировать архитектуру мобильного приложения и его компоненты;

– изучить как мобильное приложение будет защищать пользовательские данные.

1 Анализ предметной области

1.1 Анализ аналогичных проектов

Как было описано выше, цель данной работы является проектирование мобильного приложения, которые позволит защитить данные владельца мобильного устройства. В рамках этой работы предлагается решение поставленной задачи: приложение, которое предназначено для шифрования и дешифрования данных пользователя.

В работе рассматривается аналоги трёх популярных приложений, сделаем выводы о преимуществах и недостатков и распишем чем проектируемое приложение будет отличаться от конкурентов.

LUKS Manager (рисунок 1) – старейшая программа для шифрования дынных на *Android*. До *LUKS Manager* программы шифрования на самом деле этим и не занимались, а делали различные махинации, по типу присваивания файлу скрытого атрибута, которые вообще не гарантировали безопасность.

Рисунок 1 – Приложение *LUKS Manager*

Достоинства:

– программа бесплатная*;*

– шифрование осуществляется достаточно быстро*;*

Недостатки*:*

– приложение имеет довольно ограниченную функциональность (нет облачного шифрования или поддержки монтирования сетевых директорий);

– файловый менеджер, который встроек в приложение неудобен. Возможность лишь просматривать директории и файлы;

– отсутствует возможность воспользоваться или выбрать внешний файловый менеджер;

– нет поддержки русского языка.

– программа требует *root*-доступ от пользователя.

Cryptonite (рисунок 2) – программа, которая довольно молода и сейчас находится на стадии тестирования. Исходя и этих фактов, можно сделать вывод, что использовать её в качестве шифрования особо важных файлов – не разумно.

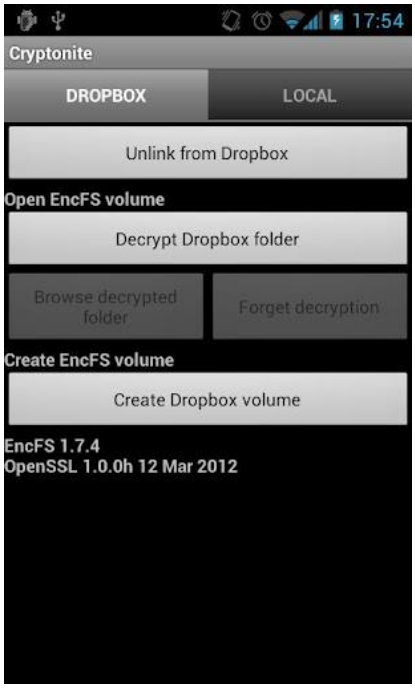


Рисунок 2 – Приложение *Cryptonite*

Преимущества:

– возможность пользования без *root*-доступа;

– шифрование осуществляется довольно быстро;

– существует встроенный файловый менеджер;

– есть поддержка русского языка;

– существует монтирования сетевых папок.

Недостатки:

– программа не бесплатная, но существует *lite*-версия. *lite*-версия не поддерживает монтирование и сетевые папки, что является огромным недостатком;

– чтобы пользоваться приложением полноценно нужна поддержка *fuse* (файловая система на основе облаков), которая существует не на всех устройствах.

EDS-Lite (рисунок 3) – молодая, но перспективная программа. Осуществляет хранение файлов в зашифрованном контейнере для предотвращения доступа от несанкционированного пользователя.

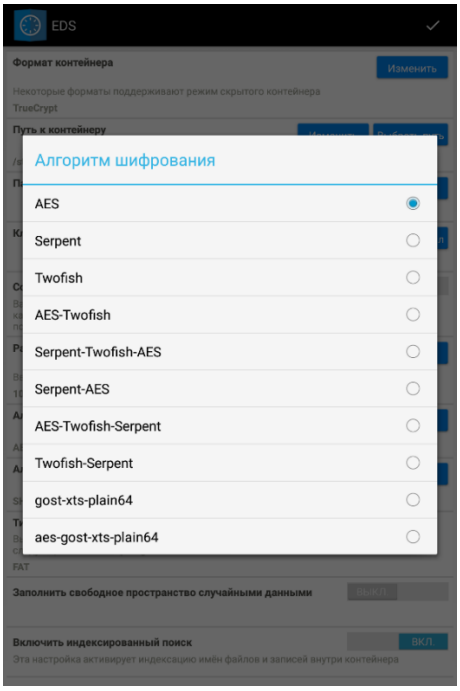


Рисунок 3 – Приложение *EDS-Lite*

Преимущества:

– для работы не нужны *root*-права;

– существует поддержка *TrueCrypt* контейнеров*;*

– существует свой файловый менеджер.

Недостатки*:*

– шифрование осуществляется не самым быстром образом;

– нет возможности работать с зашифрованным контейнером как с обычной директорией.

Рассмотрев все варианты приложений, которые существуют на данный момент можно сделать вывод о том, насколько актуально проектировать собственное приложение. Некоторые приложения действительно могут обеспечить безопасность мобильного устройства, но за это придётся отдавать деньги, так как приложения платные. Есть вариант скачать *Lite* версии приложений, но тогда большинство рабочего функционала, которое защищает устройство, будет недоступно потребителю. Помимо этого, различные мелкие недочёты, по типу недоступности русского языка или неудобный интерфейс, будут негативно влиять на владельца приложения. По этим причинам существует необходимость проектирования собственного приложения.

Ниже представлены особенности, которыми будет обладать проектируемое приложение и какими будут отличия и преимущества от других приложений.

Особенности разрабатываемого приложения:

– шифрование таких данных пользователя, как видеофайлы, фото, документации и другие данные;

– файлы, которые зашифровались и расшифровались хранятся в директориях, где находились их исходные файлы;

– удаление файлов, для которых существуют зашифрованные аналоги;

– для шифрования файлов будет использоваться 128-битный алгоритм шифрования *AES*;

– пароль, который будет хэшироваться алгоритмом *MD5*. Это позволит паролю не хранится ни на мобильном устройстве, ни на другом ресурсе;

– в целях безопасности принято решение о недоступности выхода в интернет при использовании приложения;

– у приложения будет удобный и понятный интерфейс;

– приложение будет абсолютно бесплатным и не содержать рекламы. Данная позиция предоставит возможность устанавливать приложение для всех людей, обладающих мобильным устройством;

– поддержка русского и английского языка;

– для полноценного пользования приложением не требуются *root*-права.

– присутствие своего файлового менеджера.

1.2 Обзор средств разработки для платформы *Android*

На сегодняшний день существует несколько платформ для создания и проектирования *Android* приложений:

1) *Android Studio*, которая основывается на языке программирования *Java* и *Android SDK*;

2) среда разработки *Eclipse*, которая поддерживает язык *Java* и *Android Development Tools*;

3) платформа *Xamarin*, которая основывается на языке программирования *C#* и платформе *.NET*.

Android SDK – средство мобильной разработки, которая включает в себя различные библиотеки, инструменты, которые помогают разработчику создавать мобильные приложения на платформе *Android*.

1) *API Android SDK* – *API* библиотека *Android*, которая предоставляется разработчику для создания приложений;

2) *SDK* документация – составляет из себя большую справочную информацию, которая перечисляет, что включено в каждый класс или пакет и как это использовать при разработке собственного мобильного приложения;

3) *Android Virtual Device* (*AVD*) – интерактивный эмулятор мобильного устройства на базе операционной системы *Android*. Эмулятор предоставляет возможность запускать и тестировать разрабатываемое приложение без использования реального устройства на *Android*.

4) *Development Tools* – *SDK* состоит из серии инструментальных средств, которые в процессе разработки позволяют компилировать и отлаживать приложения;

5) *Simple Code* – *Android SDK* обеспечивает стандартными приложениями, которые показывают возможности *Android*. Также предоставляет программы, которые могут показать, как используются индивидуальные возможности *API* в коде разрабатываемого приложения.

Android Studio – интегрированная среда разработки (*IDE*), которая позволяет работать с платформой *Android*. Была анонсирована 16 мая 2013 года на конференции *Google I/O* в городе Сан-Франциско. *Android Studio* является официальной средой разработки приложений для платформы *Android* [1].

Особенности*:*

– отличительная особенность эмулятора – способность просмотра показателей производительности после запуска создаваемого приложения на самых популярных устройствах;

– среда разработки *Android Studio* последних версий стала удобной даже для начинающих разработчиков. Реализация современного средства упаковки кода или его маркировки. Для облегчения переноса компонентов в среду разработки была реализована функция *Drag-n-Drop*, которой пользуются все создатели программного обеспечения.

– *Android Studio* предоставляет шаблоны основных компонентов и макетов *Android*;

– *Google Cloud Messaging* – позволяет связаться с целевой аудиторией приложения, после того, как вы пустили его в стадию релиза.

Eclipse – свободная интегрированная среда разработки кроссплатформенных приложений [2]. *ADT* (*Android Development Tools*) – плагин для *Eclipse IDE*, который позволяет создавать приложения на платформу *Android*. *Eclipse* так же позволяет создавать и приложения на *Android*, но *Android Studio* является инструментом, который был специально создан для разработки под мобильные устройства. Это приводит к тому, что на *Android Studio* намного проще и быстрее создавать приложения, чем с *Eclipse*.

Xamarin – фреймворк для кроссплатформенной разработки приложений на мобильные устройства, в котором используется язык программирования *C#* [3]. У *Xamarin* существуют следующие преимущества и недостатки:

1) Мобильные приложения создаются под разные платформы (*iOS*, *Android*, *Windows Phone*);

2) не обязательно знать базовые средства реализации приложений на *android* (основан на *C#* и платформе *.NET*);

3) лицензия на платной основе*;*

4) производительность на низком уровне, по сравнению с конкурентами;

5) неполная поддержка стандартных *API* платформы.

Учитывая преимущества и недостатки инструментов, рассмотренных выше, для проектирования приложения была выбрана среда разработки *Android Studio*. В некоторых моментах, для удобства, будет также использоваться *Eclipse IDE*.

2 Проектирование приложения

2.1 Предпроектный анализ

Диаграмма прецедентов (вариантов использования), которая показывает отношения между актерами и прецедентами. Диаграмма прецедентов нужна для того, чтобы описать моделируемую систему на концептуальном уровне и обычно применяется для спецификации требований к системе. В ходе работы в качестве актера будет пользователь. В данной работе диаграмма прецедентов (рисунок 4).

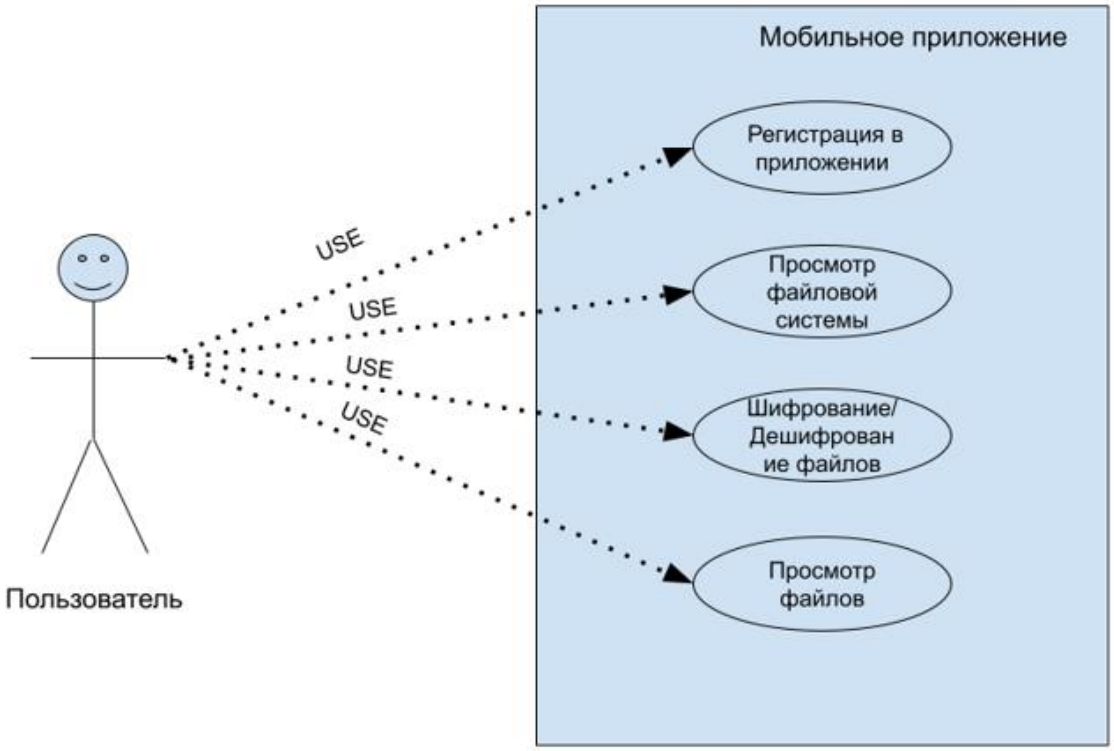


Рисунок 4 – Диаграмма прецедентов

Помимо диаграммы прецедентов следует также показать, как будет работать приложение от начала и до конца. Для это служит диаграмма

деятельности. Диаграмма деятельности (рисунок 5), которая обеспечивает графическое представления потока системы, представлена ниже:

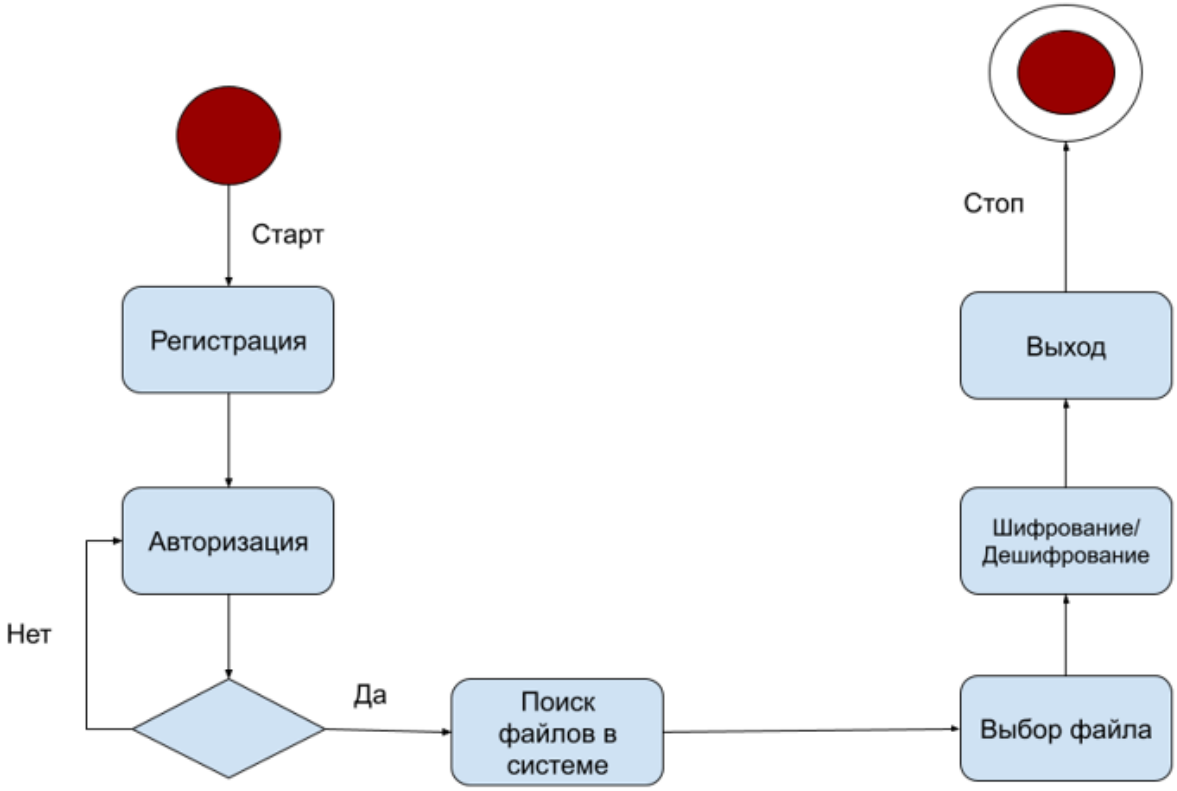


Рисунок 5 – Диаграмма деятельности приложения

Ниже будут представлены описания окон, которые будут использоваться для продуктивного использования приложения:

Экран приложения №1 будет появляться на долю секунду, практически не заметен пользователю приложения. Служит для того, чтобы проверять первый вход в приложение.

Экран приложения №2 будет открываться единожды при первом запуске, либо если мастер-пароль пользователя ещё не установлен. Помимо этого, экран будет содержать элементы взаимодействия:

– поле для того, чтобы ввести пароль и поле для подтверждения пароля (*edit text*);

– текстовое поле, пустое (*TextView*), которое предназначено для вывода сообщения об ошибках. примеры ошибок: пароли не совпадают, некорректная длина пароля (слишком короткий пароль или слишком длинный) или нельзя записать в файл хэш пароля;

– кнопка с текстом «*Ok*», чтобы проверять пароли на совпадение, записи хэша пароля в файл и переход на следующий экран (*ImageButton*);

– кнопка с текстом «*Cancel*» для того, чтобы очистить поля для ввода паролей (*ImageButton*).

Экран приложения №3 предназначен для входа в приложение. Будет появляться в том случае, если пароль установлен и приложение запущено не в первый раз. Экран будет содержать элементы взаимодействия*:*

– поле для того, чтобы ввести пароль (E*ditText*);

– текстовое поле, пустое (*TextView*), которое предназначено для вывода сообщения об ошибках. примеры ошибок: пароли не совпадают, некорректная длина пароля (слишком короткий пароль или слишком длинный) или нельзя записать в файл хэш пароля;

– кнопка с текстом «*Ok*», чтобы проверять пароли на совпадение, записи хэша пароля в файл и переход на следующий экран (*ImageButton*);

– кнопка с текстом «*Cancel*» для того, чтобы очистить поля для ввода пароля (*ImageButton*);

Экран приложения №4 будет содержать элементы взаимодействия:

– кнопка *с* текстом «*Change password*» для того, чтобы сменить пароль (*Button*);

– кнопка с текстом «*Encrypt or Decrypt*» для шифрования и расшифрования объекта (*Button*).

Экран приложения №5 будет предназначен для того, чтобы сменить мастер-пароль. Будет содержать такие элементы взаимодействия*:*

– поле с текстом «*Your Password*» для того, чтобы установленный пароль (*EditText*);

– поле с текстом «*New Password*» для того, чтобы ввести новый пароль (*EditText*);

– поле с текстом «*Confirm New Password*» для того, чтобы подтвердить новый пароль (*EditText*);

– кнопка с текстом «*Ok*» для того, чтобы проверить установленный пароль, совпадение полей «*New Password*» и «*Confirm New Password*» и для записи нового пароля в файл (*Button*);

– кнопка с текстом «*Ok*» для того, чтобы очищать поля для ввода паролей.

Экран приложения №6 будет предназначен для обзора, шифрования и дешифрования файлов. Будет содержать такие элементы взаимодействия*:*

– таблица файлов и директорий (*ListView*);

– диалог (*AlertDialog*), который будет содержать такие элементы, как кнопку с текстом «*Open*» (*Button*) для того, чтобы открыть файл, кнопку с текстом «*Decrypt*» (*Button*) для того, чтобы шифровать файлы и кнопку с текстом «*Decrypt*» (*Button*) для того, чтобы дешифровать файл.

Диаграмма классов [4] (*class diagram*) – структурная диаграмма языка моделирования *UML*. С помощью диаграммы классов можно показать, какую общую структуру иерархии классов будет иметь приложение, и как эти иерархии будут кооперировать. Диаграмма классов (рисунок 6) моделируемого приложения*:*

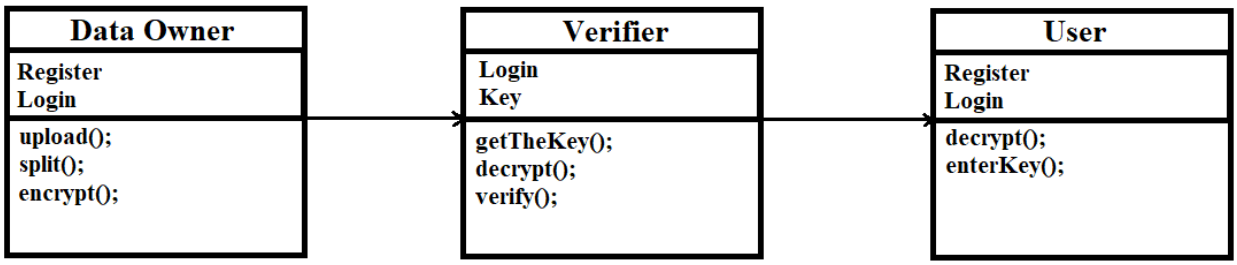


Рисунок 6 – Диаграмма классов приложения

2.2 Проектирование интерфейса

Данный этап служит для проектирования схем страниц приложения. На этих страницах будет расположена информация и элементы управления, которые будут составлять экраны приложения. Представим результаты в виде таблицы на рисунке (рисунок 6)*:*

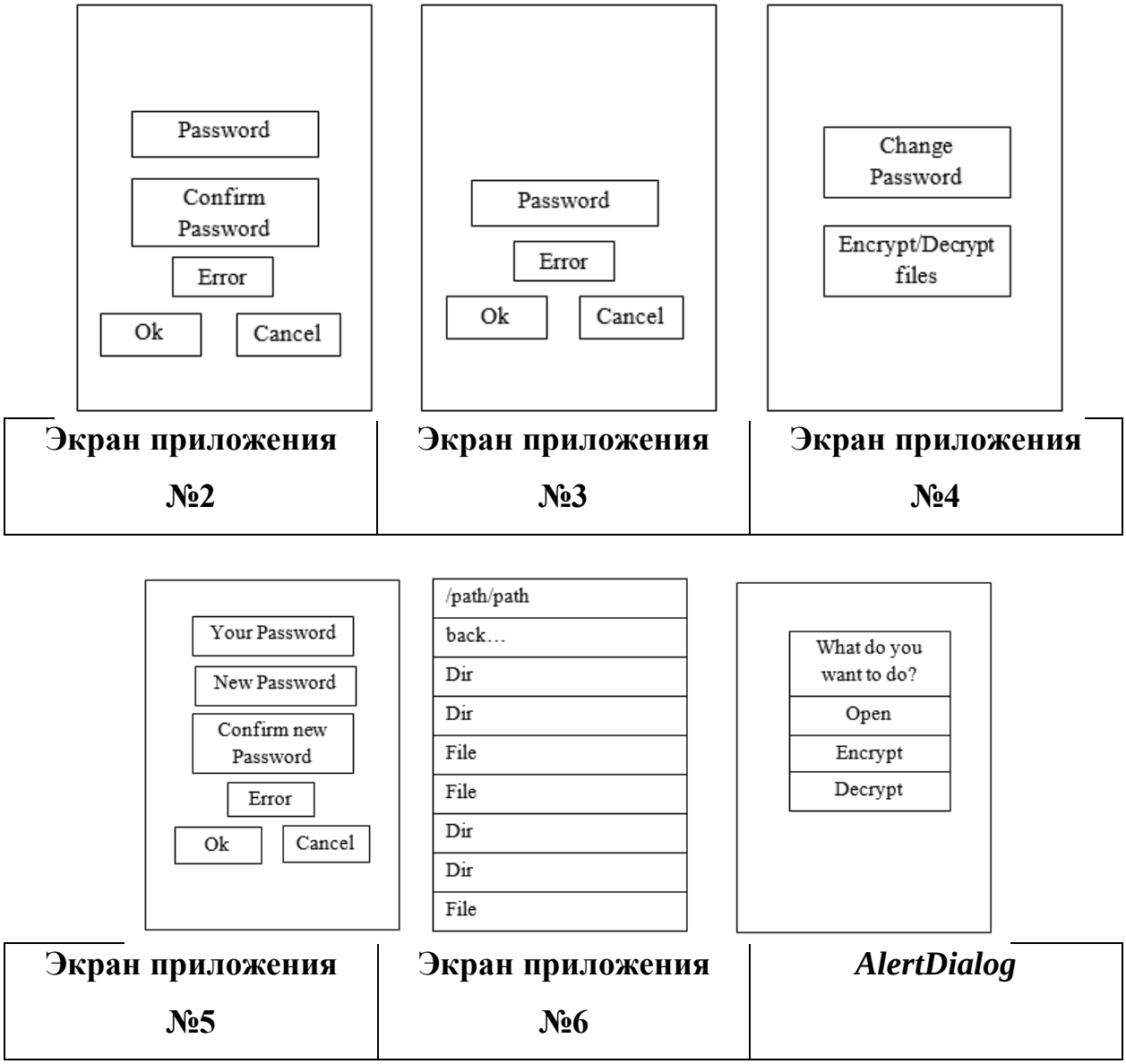


Рисунок 6– Структурные схемы интерфейса

Первый экран не указывается, так как он является вспомогательным.

2.3 Сбор требований

Данный этап служит для того, чтобы составить перечень того, что может сделать обычный пользователь в приложении. Это поможет принять во внимание все требования для проектируемого приложения. Итоговый перечень будет представлен в виде таблицы (таблица 1), в которой описано за что отвечает каждый элемент приложения:

Таблица 1 – Действия пользователя приложения при каждом элементе

Роль	**Событие**	**Наименование объекта**	**Объект**	**Дополнительная возможность**
Экран приложения №1				
Экран приложения №2				
Пользователь	Нажимает	*Password*	*EditText*	Вводить пароль, редактировать пароль
Пользователь	Нажимает	*Confirm Password*	*EditText*	Вводить пароль, редактировать пароль
Пользователь	Нажимает	*Ok*	*Button*	Сравнить пароль, вычислить хэш и запись хэш в файл, переход на экран приложения №4
Пользователь	Нажимает	*Cancel*	*Button*	Очистить ввода пароля в поле
Экран приложения №3				
Пользователь	Нажимает	*Password*	*EditText*	Ввести пароль, отредактировать пароль

Продолжение таблицы 1

Пользователь	Нажимает	*Ok*	*Button*	Сравнить пароли, вычислить хэш, перейти на экран приложения №4
Пользователь	Нажимает	*Cancel*	*Button*	Очистить поле для ввода пароля
Экран приложения №4				
Пользователь	Нажимает	*Change Password*	*Button*	Перейти на экран приложения №5
Пользователь	Нажимает	*Encrypt or decrypt*	*Button*	Перейти на экран приложения №6
Экран приложения №5				
Пользователь	Нажимает	*Your Password*	*EditText*	Ввести пароль, отредактировать пароль
Пользователь	Нажимает	*New Password*	*EditText*	Ввести пароль, отредактировать пароль
Пользователь	Нажимает	*Confirm New Password*	*EditText*	Ввести пароль, отредактировать пароль
Пользователь	Нажимает	*Ok*	*Button*	Сравнить пароли, вычислить хэш, записать хэш в файл, перейти на экран приложения №4
Пользователь	Нажимает	*Cancel*	*Button*	Очистить ввода пароля в поле

Продолжение таблицы 1

Экран приложения №6				
Пользователь	Нажимает	Список директорий и файлов	*ListView*	Навигация в файловой системе, вызвать *AlertDialog*
AlertDialog				
Пользователь	Нажимает	*Open*	*Button*	Открыть файл
Пользователь	Нажимает	*Encrypt*	*Button*	Шифровать файл
Пользователь	Нажимает	*Decrypt*	*Button*	Дешифровать файл

Таблица 1 – Действия пользователя приложения при каждом элементе

Основываясь на данных в таблице, можно составить навигационную схему. На ней будет расположена архитектура приложения, которая будет состоять из всех элементов приложения. Данная навигационная схема (рисунок 7) сгруппирует всю информацию, чтобы минимизировать количество действий, которые будут необходимы пользователю приложения.

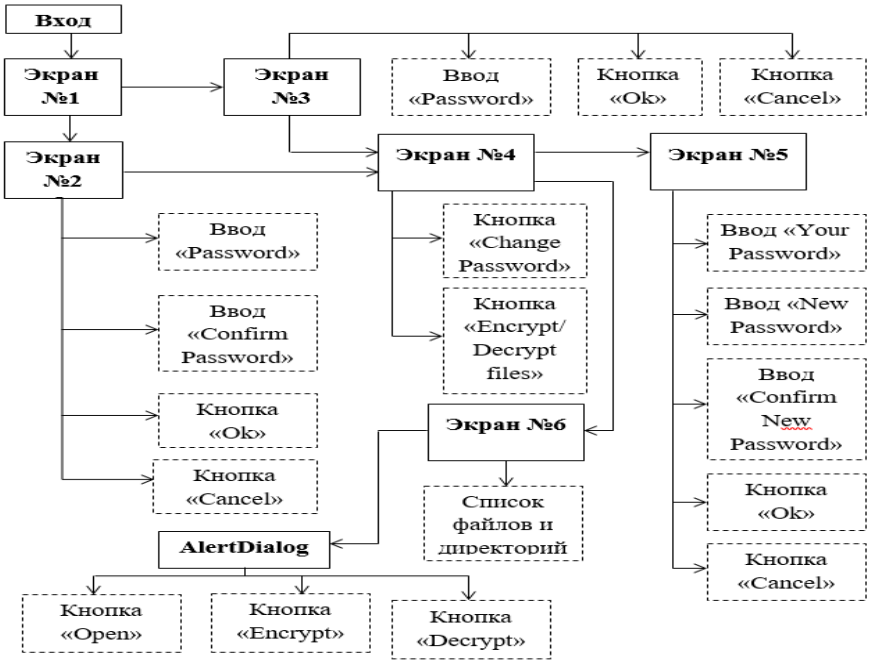
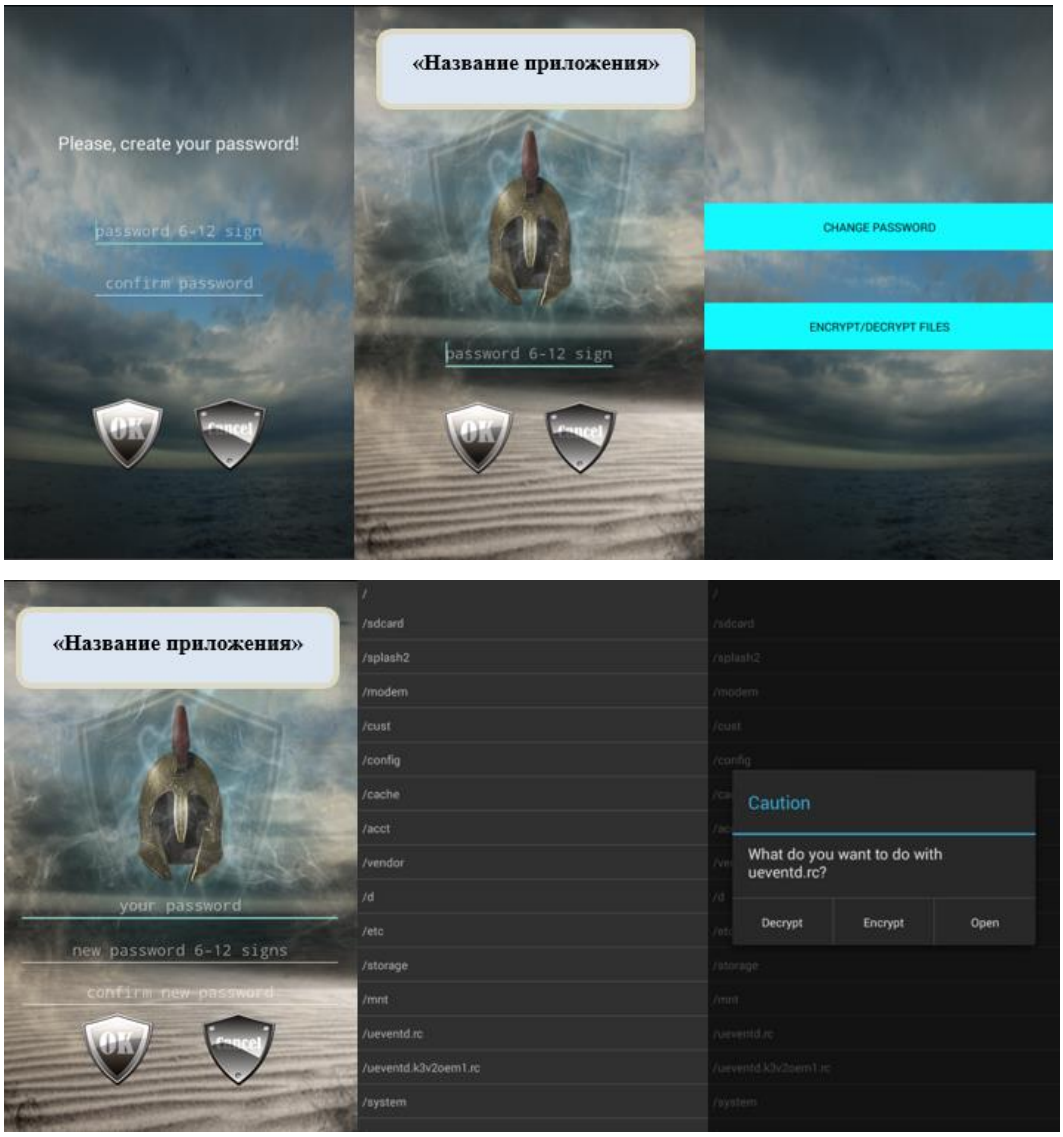


Рисунок 7 – Навигационная схема

2.4 Дизайн интерфейса

После того, как были спроектированы структурные схемы интерфейса можно уже представить, как будет выглядеть интерфейс будущего приложения (рисунок 8):

Экран приложения №2	**Экран приложения №3**	**Экран приложения №4**



Экран приложения №5	**Экран приложения №6**	***AlertDialog***

Рисунок 8 – Дизайн интерфейса

3 Защита данных приложения

3.1 Алгоритм шифрования *AES*

AES [5] – концепция, которая сводится к подстановкам и перестановкам цифр в блоках данных, которые обрабатываются. Шифр подстановки – это метод шифрования, где элементы изначального текста будут заменяться на зашифрованный текст, который будет строиться по определенному правилу. Причем элементами текста могут быть отдельными символами, парами букв, тройками букв и так далее.

Алгоритм *AES* будет работать с отдельными блоками данных, у которых фиксированная длина (128 бит). Вся шифруемая информация будет разбиваться на эти блоки. Если выяснилось, что длина сообщения меньше длины блока, то сообщение будет дополняться до размеров блока.

Алгоритм основан на принципах перестановок и подстановок, у которого архитектура будет иметь следующие условия:

– блок, который шифруется, будет представляться в виде двумерного байтового массива;

– за один раунд будет шифроваться весь блок данных;

– криптографические преобразования будут выполняться как над отдельными байтами массива, так и над строками и столбцами.

Для того, чтобы осуществить надежность и криптоустойчивость приложения было выбрано 128-битное шифрование. Это означает, что данные будут разделены на массив четыре на четыре, который будет содержать 16 байт. Каждый байт содержит восемь бит. Следовательно, умножение 16 байтов на 8 бит даёт всего 128 бит в каждом блоке.

Ниже приведена блок-схема (рисунок 9), на которой показано, как будет происходить шифрование и дешифрование файлов приложении.

Рисунок 9 – Алгоритм шифрования и дешифрования.

3.2 Другие способы защиты данных в приложении

Помимо того, что приложение будет шифровать и расшифровывать данные пользователя, оно должно максимально обеспечить защиту данных от злоумышленников внутри приложения. Для этого могут быть выполнены следующие условия:

– Приложения невозможно сворачивать. То есть при сворачивании приложение выходило на экран авторизации. Это не позволит злоумышленнику делать различные махинации с файлами после того, как человек свернул приложение и оставил мобильное устройство на видном месте;

– Временная проверка на использование приложение. После определенного периода времени запускается проверка, когда будет запрашивать у пользователя ответ о том, пользуется ли он ещё приложением при бездействии. Чтобы отправить ответ приложению у пользователя будет от 10 до 20 секунд.

– При регистрации или авторизации будет находиться проверка на капчу. Капча – компьютерный тест, который используется для того, чтобы определить, кем является пользователь системы: компьютер или пользователь. Данная проверка может спасти от такой махинации, как полный перебор (*brute force*).

– Помимо капчи, при входе в приложение у пользователя будет 3 попытки для ввода корректного пароля. Данное действие позволит защитить данные от перебора пароля от конкретного злоумышленника.

– Когда владелец устройства заходит в приложение, отключается возможность использовать интернет. Это позволит защитить данные в приложении от различных угроз, связанных с сетью. Например, различные расширения в браузере могут забирать информацию об использовании человеком конкретных приложений.

ЗАКЛЮЧЕНИЕ

Мобильные устройства всегда выполняют функцию «мобильного секретаря». Они освобождают голову владельца устройства от хранения большого количества информации. Кроме того, в этой информации иногда хранится очень важные сведения: номера кредитных карт, пароли от социальных сетей, электронная почта.

Потерять мобильное устройство – настоящая беда для владельца. Иногда его крадут специально для того, чтобы узнать о личной жизни или просто завладеть важной информацией. В разных ситуациях его не крадут, а пользуются на небольшом промежутке времени, но для опытного злоумышленника такое время будет достаточно, чтобы совершить злодеяние.

Утеря личной конфиденциальной информации обернётся пользователю серьёзными проблемами, если попадет не в нужные руки. Поэтому каждому владельцу мобильного устройства стоит беспокоиться о конфиденциальности своих данных. Для этого существуют специальные средства защиты телефона от несанкционированного доступа.

В данной курсовой работе была проведена работа по проектированию приложения для мобильного устройства для платформы *Android*, которое сможет защитить данные пользователя, путём шифрования и дешифрования.

Были выполнены следующие задачи при проектировании приложения:

– дан анализ существующим аналогам приложения;

– были изучены особенности проектирования и разработки мобильного приложения для ОС *Android*;

– спроектирована архитектура мобильного приложения и его компоненты.

СПИСОК СОКРАЩЕННЫХ ОБОЗНАЧЕНИЙ

ОС - Операционная система.

AES - Advanced Encryption Standart, расширенный стандарт шифрования.

SDK - Software Development Kit, комплект разработки программного обеспечения.

API - Application Programming Interface, программный интерфейс приложения.

AVD - Android Virtual Device, виртуальное устройство *Android*.

IDE - Integrated Drive Electronics, интегрированная среда разработки.

ADT - Android Development Tools, инструменты разработки *Android*.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Android Studio* [Электронный ресурс]. Режим доступа: *URL* - https://ru.wikipedia.org/wiki/Android_Studio (Дата обращения: 25.05.2022).
2. *Eclipse* [Электронный ресурс]. Режим доступа: *URL* - https://ru.wikipedia.org/wiki/Eclipse (Дата обращения: 25.05.2022).
3. *Xamarin* [Электронный ресурс]. Режим доступа: *URL* - https://ru.wikipedia.org/wiki/Xamarin (Дата обращения: 25.05.2022).
4. Диаграмма классов [Эектронный ресурс]. Режим доступа: *URL* - https://ru.wikipedia.org/wiki/Диаграмма_классов (Дата обращения: 25.05.2022).
5. AES [Электронный ресурс]. Режим доступа: *URL* - https://ru.wikipedia.org/wiki/AES_(стандарт _шифрования) (Дата обращения: 25.05.2022).
6. Баричев С.Г. Основы современной криптографии [Текст] / Баричев С.Г., Гончаров В.В., Серов Р.Е. // М.: Горячая линия – Телеком, 2002. 152 с.
7. Голощапов А.Л. *Google Android*: программирование для мобильных устройств [Текст] / Голощапов А.Л // СПб: БХВ-Петербург, 2011. 448 с.
8. Делессио К. Создание приложений для Android за 24 часа [Текст] / Делессио К., Дарси Л., Кондер Ш. М. // Эксмо, 2015. 528с.
9. Дейтел П. Android для программистов: создаем приложения [Текст] / Дейтел П., Дейтел Х., Дейтел Э., Моргано М. // СПб.: Питер, 2013. 560 с.
10. Колисниченко Д., Программирование для *Android.* Самоучитель [Текст] / Колисниченко Д. // СПб.: БХВ-Петербург, 2012. 272с.
11. Лафоре Р., Структуры данных и алгоритмы в *Java*. СПб.: Питер, 2015. 704 с.
12. Майер Р. Android 2. Программирование приложений для планшетных компьютеров и смартфонов [Текст] / Майер Р. // М.: Эксмо, 2011. 672 с.

13. Пасенко С. Алгоритмы шифрования. Специальный справочник [Текст] / Пасенко С. // СПб.: БХВ-Петербург, 2009. 576 с.

14. Освой программирование играючи. Разработка под *Android* [Электронный ресурс]. Режим доступа: *URL* - http://developer.alexanderklimov.ru/android/ (Дата обращения: 26.05.2022)

15. Роджерс Р. Android. Разработка приложений [Текст] / Роджекрс Р. // М.: ЭКОМПаблишерз, 2010. 400 с.

16. Флэнаган Д., *Java* в примерах. Справочник, 2-е издание [Текст] / Флэнгана Д. // М.: Символ-Плюс, 2003. 664 с.

17. Хашими С. Разработка приложений для *Android* [Текст] / Хашими С., Коматинени С., Маклин Д. // СПб.: Питер, 2011. 400 с.

18. Цехнер М., Программирование игр под *Android* [Текст] / Цехнер М. // СПб.: Питер, 2013. 688с.

19. Эккель Б., Философия *Java*, 4-е издание [Текст] / Эккель Б. // СПб.: Питер, 2014. 640с.

20. *UI Modeling Company*. Проектирование и дизайн интерфейсов [Электронный ресурс]. Режим доступа: *URL*: - http://www.uimodeling.ru/process/user-interface-design.html (Дата обращения: 28.05.2022).