МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Единый портал учёта документов с использованием технологии Blockchain. Мобильное приложение

Студент _____ (подпись, дата) Шаповалов Игорь Сергеевич

Руководитель _____ (подпись, дата) Трокоз Д.А.

Нормоконтролёр 17.06.18 (подпись, дата) Кучин А.В.

Работа допущена к защите (протокол заседания кафедры от 25.05.2018 № 38-09-12)

Заведующий кафедрой _____ (подпись) Пащенко Д.В.

Работа защищена с отметкой отл. (протокол заседания ГЭК от 20.06.18 № 11)

Секретарь ГЭК _____ (подпись) Гурин Е.И.

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет Вычислительной техники

Кафедра Вычислительная техника

«Утверждаю»
Заведующий кафедрой
_____ Д.В. Пащенко
«__»_____ 2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Шаповалов Игорь Сергеевичгр. 14ВВ1 факультета вычислительной техники.

2.Тема работы: «Единый портал учёта документов с использованием технологии Blockchain. Мобильное приложение».

Тема утверждена приказом ПГУ № 398а/о от 03.05.2018 г.

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Трокоз Д.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение разработки – клиент-серверное приложение, предназначенноедля добавления документов в базу данных и последующей проверки того, что документ не был изменен.
Возможности системы: добавление документа в базу данных, проверка документа на неизменность после добавления в базу данных, подтверждение авторства документа с помощью цифровой подписи, проверка целостности базы данных.

5. Перечень подлежащих разработке вопросов.
Изучение сфер применения мобильных приложений, выборархитектуры приложения и среды разработки, выбор языков программирования для клиентского и серверного приложения, определение способа взаимодействия клиента и сервера.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	13.03.18	
2.	Выбор архитектуры приложения, языков программирования, среды разработки и способа взаимодействия	10	25.03.18	
3.	Определение способа организации пользовательского интерфейса	10	03.04.18	
4.	Разработка серверного приложения	20	11.04.18	
5.	Разработка клиентского приложения	20	25.04.18	
6.	Тестирование и отладка	10	13.05.18	
7.	Оформление выпускной работы	20	26.05.18	

Дата выдачи задания "5" марта 2018 г.

Руководитель бакалаврской работы _____ Трокоз Д.А.
(подпись, дата)

Задание к исполнению принял студент _____ Шаповалов И.С.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 15.06.2018 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 117 листов, 27 рисунков, 2 таблицы, 9 источников, 3 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, АНДРОИД, БЛОКЧЕЙН, ХЭШ, БД.

Цель работы – разработать клиент-серверное приложение, осуществляющее добавление подписанного хэша документа в базу данных, построенной по типу технологии блокчейн, проверку ранее добавленного документа на неизменность и проверку отсутствия повреждения базы данных.

Объектом разработки является клиент-серверное приложение, состоящее из клиентского мобильного приложения на ОС Андроид с графическим интерфейсом, взаимодействующее с сервером, который управляет данными, построенными по типу блокчейна. Они осуществляют добавление подписанного хэша документа с целью дальнейшей проверки неизменности документа.

В результате проведенной работы реализован проект, удовлетворяющий поставленной задаче, а именно, создано серверного приложение, принимающее запросы от клиентов и взаимодействующее с базой данных, а также клиентское мобильное приложение, имеющее возможность хэшировать, подписывать документ и проверять его на неизменность.

Данное приложение рекомендуется использовать в тех областях, где важен контроль за неизменностью документов, например, бизнес или государственные учреждения. Проект может быть применен в данных сферах с целью пресечения подмены информации для собственной выгоды.

Содержание

РЕФЕРАТ 4

Содержание 5

Введение 8

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9

1.1 Сферы применения мобильных приложений 9

1.2 Платформа Android среди других мобильных операционных систем 11

1.3 Архитектура программного комплекса 13

1.4 Обоснование языка программирования 15

1.5 Выбор программной среды разработки 21

1.6 Структуры данных JSON 24

1.7 Технология блокчейн и её защищенность 25

1.7.1 Публичные блокчейны 27

1.7.2 Приватные блокчейны 28

1.7.3 Плюсы и минусы технологии блокчейн и её перспективы 31

2 РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ 34

2.1 Постановка задачи 34

2.2 Структура данных приложений 35

2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса 36

2.4 Описание разработки программы 38

2.5 Отладка и тестирование 42

3 ОПИСАНИЕ ПРОГРАММЫ 43

3.1 Описание клиентского приложения 43

3.3 Описание серверного приложения 46

3.4 Руководство пользователя 47

ЗАКЛЮЧЕНИЕ 61

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 62
Приложение А Листинги программы сервера 63
Приложение А.1 Файл «main.py» 63
Приложение А.2 Файл «block.py» 65
Приложение В Листинги программы клиента 69
Приложение В.1 Файл «MainActivity.java» 69
Приложение В.2 Файл «LoadFile.java» 72
Приложение В.3 Файл «CheckFile.java» 85
Приложение В.4 Файл «CheckBD.java» 92
Приложение В.5 Файл «FileUtils.java» 99
Приложение В.6 Файл «Request.java» 100
Приложение В.7 Файл «RequestGetHashById.java» 100
Приложение В.8 Файл «RequestSaveBD.java» 101
Приложение В.9 Файл «RequestSendFile.java» 101
Приложение В.10 Файл «activity_main.xml» 102
Приложение В.11 Файл «load_file.xml» 103
Приложение В.12 Файл «check_file.xml» 106
Приложение В.13 Файл «check_bd.xml» 109
Приложение С Презентация 111

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

ОС – Операционная система

БД – База анных

APK - Android Package Kit

SDK - Software Development Dit

TCP – Transmission Control Protocol

IP – Internet Protocol

HTTP – HyperText Transfer Protocol

ИС – Информационная система

URL – Uniform Resource Locator

ID – Identifier

URI – Uniform Resource Identifier

JSON – JavaScript Object Notation

FTP - File Transfer Protocol

IDE - Integrated Development Environment

ПО - Программное обеспечение

Введение

С развитием информационных технологий всё больше данных переносится в электронный вид. Однако вместе с этим удобством возникают и трудности, связанные с безопасностью и целостностью информации.

Везде, где есть недоверие между участниками может быть использована технология Blockchain. В частности, в совершении сделок купли-продажи или проверки достоверности услуг или документов.

Особенность технологии в том, что все данные объединяются с помощью математических алгоритмов в блоки, а последние образуют цепочку. Каждый блок содержит хэш предыдущего. Эта цепочка начинается от первого блока в системе и действует до последнего. Редактировать информацию, которая находится в сети длительное время, непрактично. Нужно переписывать все блоки. Длина цепи определяется по сложности, а не по количеству элементов. Метод хэширования не оставляет детальной информации о транзакции, а только подтверждает сам факт ее существования. После обновления данных их нельзя изменить. Можно только добавить новую информацию. А в случае подмены информации на каком-то этапе, она не будет соответствовать цифровой подписи.

Используя Blockchain, можно также сэкономить на оплате услуг посредников и гарантов. Однако наряду со всеми плюсами есть и минусы, связанные с требованием больших мощностей, что существенно увеличивает затраты на создание и внедрение в какую-либо сферу.

В рамках данной выпускной квалификационной работы требуется разработать систему на основе технологии блокчейн, которая бы имела интуитивно понятный пользовательский интерфейс на клиентской стороне и возможность проверки документа на неизменность.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Сферы применения мобильных приложений

Сегодняшний день является временем научно-технического прогресса, очень сложно представить себе жизнь и быт современного общества без использования мобильных устройств. Ускоряется ритм жизни, вместо с ней ускоряется процесс создания обществом технических новинок для своего удобства. Взять, к примеру, мобильные телефоны. Мы пользуемся ими повсюду - дома, в поездке, на работе, на отдыхе. Это удобно можно даже сказать комфортно. Мобильный телефон есть у каждого, и у бизнесмена, и школьника, он неожиданно ворвался, но прочно осели закрепился в нашей жизни.

Очевидно, что мобильный телефон уже давно перестал быть просто средством общения. Обыкновенный разговор по телефону постепенно становится второстепенной функцией, пропадая в огромном наборе функций, реализуемых мобильным телефоном. Слушать музыку, фотографировать, играть. Этот список можно продолжать бесконечно. Таким образом, мобильный телефон стал многофункциональным устройством, позволяющим человеку пользоваться практически всеми современными технологиями.

Изменилось и отношение людей к мобильным телефонам. Сегодня, забыв сотовый дома, люди ощущают себя не комфортно. Отсутствие возможности позвонить родственникам, друзьям или знакомым придает ощущение оторванности от общества. Хотя, всего несколько лет назад они спокойно обходились без мобильного телефона и считали его атрибутом роскоши.

Всего пару десятилетий назад были выпущены первые мобильные телефоны. Они были большими, неудобными в использовании, не имели дополнительных функций, и к тому же, были не по карману человеку со средним доходом. Но со временем все меняется, и мобильники не исключение. Даже напротив, их развитие идет колоссальными темпами. Если на заре своего

развития телефоны старались сделать компактными и дешевыми, то сегодня, достигнув в этом приличных показателей, мобильные телефоны делают универсальным устройством, которое может заменить диктофон, фотоаппарат, и даже компьютер.

Эта универсальность достигается благодаря новым возможностям операционных систем, которые развиваются вместе с самими аппаратными возможностями. Успех мобильного устройства еще и в большей степени зависит от того, какая на нем установлена операционная система. Их на рынке огромное количество: iPhone OS, MeeGo, Windows Mobile, Samsung Bado OS и много других. Но несомненным лидером по популярности, на сегодняшний день выступает, несомненно ОС Android. Она нацелена на массовую аудиторию и является весьма доступной и удобной программной разработкой. Секрет ее успеха очень прост - она открыта и бесплатна, предоставляет программисту удобные средства разработки и возможность реализации своих продуктов, число которых на сегодняшний день насчитывает несколько сотен тысяч.

Количество, проданных смартфонов в России в предыдущем 2017 году не только заметно выросло по сравнению с предыдущим годом, но и вовсе побило рекорд 2014 года. Динамику последних лет можно посмотреть на следующей диаграмме:

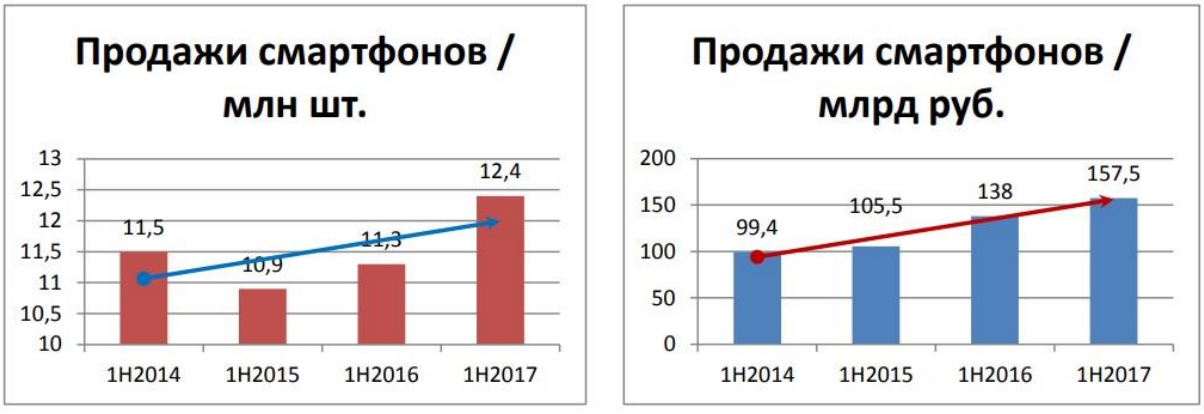


Рисунок 1 – обзор продаж смартфонов

С появлением мобильных телефонов, планшетов, посещать сайт и компьютерное приложение стало менее востребовано. Поэтому для работы

можно с легкостью использовать мобильное приложение. У них есть свои преимущества перед сайтом:

– увеличено быстродействие;

– загружается только необходимая информация, т.е. экономит интернет трафик;

– никакой рекламы;

– более удобный интерфейс для просмотра;

– более удобный функционал.

1.2 Платформа Android среди других мобильных операционных систем

Наличие операционной системы (ОС) - главная особенность, которая отличает смартфон от обычного мобильного телефона. При выборе конкретной 8 модели телефона или устройства, операционная система часто является определяющим фактором. Наиболее распространенные операционные системы для смартфонов и платформ: Symbian OS, BlackBerry OS (RIM), Windows Mobile и Windows CE, Windows Phone 7, Palm OS, Linux, Bada, Android, ОС IOS, Windows Phone 8

В настоящее время Android развивается в геометрической прогрессии: каждый год число пользователей этой операционной системы постоянно растет. Согласно последнему отчету компании Canalys, ведущего аналитика индустрии высоких технологий, операционная система Android занимает 69,2% мирового рынка мобильных устройств. Конечно, этот факт привлекает внимание многих разработчиков создавать мобильные приложения, специально для Android. Возможно, на сегодняшний день она является самой популярной и интересной системой. Разработчики дают пользователям уникальную возможность – установив набор свободного программного обеспечения, можно создать программы для системы и продавать их в специализированном интернет-магазине.

В наше время все больше и больше смартфонов, коммуникаторов, планшетных ПК и других видов устройств, удобных для использования как в повседневной жизни, так и в заграничных поездках в частности, выпускаются на базе ОС Android. Но в чем же причины популярности данной операционной системы?

- Android поддерживает большое количество устройств разных производителей.
- Android характеризуется высокой доступностью средств разработки. Средства разработки для платформы Android бесплатны, в то время как разработка, к примеру, под iPhone (от компании Apple) требует немалых начальных финансовых вложений.
- Наличие бесплатных библиотек для работы со сторонними ресурсами (Yandex MapKit, Google Map API, др.), в то время как для Windows Phone Mobile такие библиотеки не распространены.
- Некоторые обозреватели отмечают, что Android проявляет себя лучше одной из своих конкуренток, Apple iOS, в ряде особенностей, таких как: веб-сёрфинг, интеграция с сервисами Google Inc. и прочих. Android, в отличие от iOS, является открытой платформой, что позволяет реализовать на ней больше функций.
- В отличие от iOS и Windows Phone 7, в Android заявлена полноценная реализация Bluetooth-стека, позволяющая в том числе передачу и приём файлов. Присутствует реализация FTP-сервера, режима точки доступа к сети (службы PAN) и групповой одноранговой сети через Bluetooth (службы GN).
- В Android-аппаратах, как правило, присутствует MicroSD-кардридер, позволяющий быстрый перенос файлов с компьютера на телефон. Кроме того, в iOS и Windows Phone 7 невозможна прямая передача каких-либо файлов с телефона и на телефон, кроме как через программы синхронизации (iTunes и Zune), в то время как телефоны на Android экспортируют файловую систему карты памяти как USB mass storage device (в версиях старше 3.0 на усмотрение производителя) или с помощью Media Transfer Protocol.

– Несмотря на изначальный запрет на установку программ из «непроверенных источников» (например, с карты памяти), это ограничение отключается штатными средствами в настройках аппарата, что позволяет устанавливать программы на телефоны и планшеты без интернет-подключения (например, пользователям, не имеющим Wi-Fi-точки доступа и не желающим тратить деньги на мобильный интернет, который обычно стоит дорого), а также позволяет всем желающим бесплатно писать приложения для Android и тестировать на своём аппарате.

– Android доступен для различных аппаратных платформ, таких как ARM, MIPS, x86.

– Существуют альтернативные Google Play магазины приложений: Amazon Appstore, Opera Mobile Store, Yandex.Store, GetUpps.

1.3 Архитектура программного комплекса

Разработка приложений под ОС Android не требует от разработчика использования специфической аппаратной и/или программной платформы, что упрощает процесс разработки ПО. Для проектирования используется интегрированная среда разработки (IDE) AndroidStudio от компании Google, которая включает в себя все необходимые инструменты для создания, отладки и тестирования программных приложений. Для работы с данной IDE нужен компьютер с одной из нижеперечисленных ОС и следующими минимальными аппаратными характеристиками:

ОС Windows:

– Microsoft Windows 7/8/10 (32- или 64-bit);

– 3GB RAM (плюс 1GB для Android Emulator);

– 2GB дискового пространства;

– 1280x800 разрешение экрана.

ОС Mac:

– Mac OS X 10.10 (Yosemite) илиновее;

– 3GB RAM (плюс 1GB для Android Emulator);

– 2GB дискового пространства;

– 1280x800 разрешение экрана.

ОС Linux:

– GNOME или KDE desktop;

– Ubuntu 14.04 LTS или новее (32- или 64-bit);

– GNU C Library (glibc) 2.19 илиновее;

– 3GB RAM (плюс 1GB для Android Emulator);

– 2GB дискового пространства;

– 1280x800 разрешение экрана.

Так же для работы с IDE и компиляции приложений нужна минимальная версия Java 6, хотя рекомендуется использование Java 7 и выше.

IDEAndroidStudio располагает богатым функционалом, который помогает ускорить разработку приложений, а именно:

– расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи технологии Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана;

– сборка приложений, основанная на Gradle;

– различные виды сборок и генерация нескольких .apk файлов;

– рефакторинг кода;

– статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и др.;

– встроенный ProGuard и утилита для подписи приложений;

– шаблоны основных макетов и компонентов Android;

– поддержка разработки приложений для AndroidWear и Android TV;

– встроенная поддержка GoogleCloudPlatform, которая включает в себя интеграцию с сервисами GoogleCloudMessaging и AppEngine;

– работа с обновленным компилятором Jack;

– функция Instant Run.

Также данная IDE включает в себя ряд необходимых средств для разработки приложений:

– SDK Tools - компонент для Android SDK. Он включает в себя полный набор средств разработки и отладки для ОС Android;

– Debugger – утилита позволяющая производить отладку приложений, запущенных на AndroidEmulator или подключенном устройстве Android. С помощью отладчика можно сделать следующее: выбрать устройство для отладки приложения, установить контрольные точки в коде Java и C/C++, изучить переменные и оценить выражения во время выполнения, и т.д.;

– Testingtools&frameworks – набор инструментов для создания и тестирования приложения;

– APK Analyzer – утилита для проведения проверки содержимого APK. Показывает размер каждого компонента, чтобы можно было определить способы уменьшения общего размера APK. Также позволяет просматривать упакованные assets, проверять файлы DEX для устранения проблем с быстрым запуском и сравнивать различия между двумя APK;

– AndroidVirtualDevice (AVD) - позволяет определить характеристики Android-телефона, планшета, AndroidWear или устройства Android TV, которые необходимо моделировать в AndroidEmulator. AVD Manager помогает легко создавать AVD и управлять ими;

– и т.д.

1.4 Обоснование языка программирования

Java — объектно-ориентированный язык программирования, разрабатываемый компанией SunMicrosystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (JamesGosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения [22].

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной Java -машине (JVM) — программе, обрабатывающей байтовый код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байтовый код в отличие от текста обрабатывается значительно быстрее.

Достоинство подобного способа выполнения программ — в полной независимости байт-кода от ОС и оборудования, что позволяет выполнять Java приложения на любом устройстве, которое поддерживает виртуальную машину. Другой важной особенностью технологии Java является гибкая система безопасности, благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание. Это позволяет пользователям загружать программы, написанные на Java, на их компьютеры (или другие устройства, например, мобильные телефоны) из неизвестных источников, при этом не опасаясь заражения вирусами, пропажи ценной информации и т. п.

Внутри Java существуют 3 основных семейства технологий:

J2EE — JavaEnterpriseEditon, для создания программного обеспечения промышленного уровня. На рисунке 2 показана структура J2EE;

J2SE — JavaStandardEditon, для создания пользовательских приложений, в первую очередь — для настольных систем;

J2ME — JavaMicroEdition, для использования в устройствах, ограниченных по вычислительной мощности, в том числе мобильных телефонах, PDA, встроенных системах.

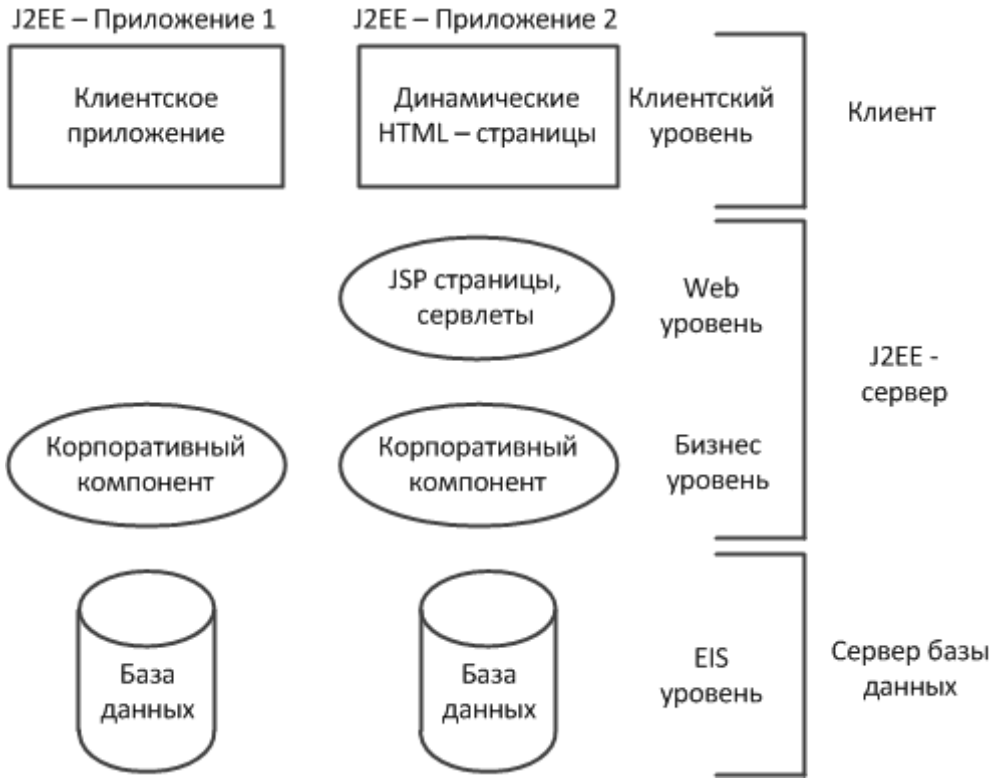


Рисунок 2 – структура J2EE

Синтаксис Java близок к синтаксису языка C++. Унаследовав самое лучшее от языка программирования C++, язык Java при этом избавился от некоторых недостатков C++, в результате чего на нем стало проще программировать. В этом языке нет, например, указателей, которые сложны в использовании и потенциально могут послужить причиной доступа программы к не принадлежащей ей области памяти. Нет множественного наследования и шаблонов, хотя функциональные возможности языка Java от этого не пострадали.

Огромное преимущество Java заключается в том, что на этом языке можно создавать приложения, способные работать на различных платформах. К сети Internet подключены компьютеры самых разных типов - Pentium PC, Macintosh, рабочие станции Sun и так далее. Даже в рамках компьютеров, созданных на базе процессоров Intel, существует несколько платформ, например, MicrosoftWindows версии 3.1, Windows 95, Windows NT, OS/2, Solaris, различные разновидности операционной системы UNIX с графической оболочкой XWindows.

Кроме того, Java — полностью объектно-ориентированный язык, даже в большей степени, чем C++. Все сущности в языке Java являются объектами, за исключением немногих основных типов (primitivetypes), например, чисел (int).

Исключена возможность явного освобождения памяти. Память в языке Java освобождается автоматически с помощью механизма сборки мусора. Программист защищен от ошибок, связанных с неправильным использованием памяти.

Если предоставить в распоряжение программиста только язык программирования и не снабдить его набором готовых модулей, предназначенных для решения самых распространенных задач, ему придется отвлекаться на множество мелких деталей.

Обычно все профессиональные системы разработки приложений на языке программирования C++ содержат в своем составе набор стандартных библиотечных функций или библиотеки классов. В комплекте со всеми средствами разработки Java поставляются достаточно развитые библиотеки классов, значительно упрощающие программирование. Далее будет кратко описан состав и назначение библиотек классов Java.

1. Встроенные классы Java

В языке Java все классы происходят от базового класса Object, и, соответственно, наследуют методы этого класса. Некоторые библиотеки классов подключаются автоматически, и их называют встроенными. К ним относятся, в частности, классы из библиотеки java.lang.* Другие библиотеки классов необходимо подключать в исходном тексте приложения Java явным образом с помощью оператора import.

2. Встроенный класс String

Класс String предназначен для работы с такими часто встречающимися объектами, как текстовые строки. Методы этого класса позволяют выполнять над строками практически все операции, которые выполнялись раньше при помощи библиотечных функций C. Это преобразование строки в число и обратно

с любым заданным основанием, определение длины строки, сравнение строк, извлечение подстроки и так далее.

Хотя в языке Java не допускается перезагрузка (переопределение) операторов, для объектов класса Stirng и объектов всех произошедших от него классов сделана встроенная перезагрузка операторов "+" и "+=". С помощью этих операторов можно выполнять слияние (конкатенацию) текстовых строк, например: System.out.println("x = " + x + '\n');

Здесь в качестве параметра функции println передается текстовая строка, составленная из трех компонент: строки "x = ", числа x и символа перехода на следующую строку '\n'. Значение переменной x автоматически преобразуется в текстовую строку (что выполняется только для текстовых строк) и полученная таким образом текстовая строка сливается со строкой "x = ".

3. Другие встроенные классы

Среди других встроенных классов отметим класс Math, предназначенный для выполнения математических действий, таких как вычисление синуса, косинуса и тангенса.

Предусмотрены также классы для выполнения запуска процессов и потоков, управления системой безопасности, а также для решения прочих системных задач.

Библиотека встроенных классов содержит очень важные классы для работы с исключениями. Эти классы нужны для обработки ошибочных ситуаций, которые могут возникнуть при работе приложений Java.

4. Замещающие классы

Очень часто в приложениях вместо базовых типов переменных используются объекты встроенных классов, которые называются замещающими классами (wrapperclasses). Заметим, что для преобразования базовых типов данных в объекты замещающего класса и обратно нельзя применять оператор присваивания. Вместо этого необходимо использовать соответствующие конструкторы и методы замещающих классов.

Ниже будут кратко перечислены подключаемые библиотеки классов Java для того чтобы можно было оценить их возможности. Подробное описание этих классов есть в справочной системе JavaWorkShop и в различной литературе, посвященной Java. Так что ограничимся описанием тех классов, которые будут использоваться в данном приложении.

1. Библиотека классов java.util

Полезна при составлении приложений, потому что в ней имеются классы для создания таких структур, как динамические массивы, стеки и словари. Есть классы для работы с генератором псевдослучайных чисел, для разбора строк на составляющие элементы (токены), для работы с календарной датой и временем.

2. Библиотека классов java.io

В библиотеке классов java.io собраны классы, имеющие отношение к вводу и выводу данных через потоки. Заметим, что с использованием этих классов можно работать не только с потоками байт, но также и с потоками данных других типов, например, числами int или текстовыми строками.

3. Библиотека Android

Необходима для создания приложений под ОС Android. Она предоставляет широкий набор классов для работы с ОС, базами данных, аудио, видео и т.д. Основным классом является Activity – это некое подобие экранной формы в Windows.

Базовым языком программирования под ОС Android считается Java и набор библиотек API. В качестве основного сайта по вопросам разработки под Android является сайт AndroidDevelopers. В качестве IDE для разработки Google разработала ряд дополнительных возможностей для IDEIntelliJ IDEA и в итоге выпустила IDE под названием AndroidStudio. Кроме того, потребуется JavaDevelopmentKit (JDK) версии 5 или выше.

1.5 Выбор программной среды разработки

AndroidStudio - интегрированная среда разработки для работы с платформой Android, анонсированная 16 мая 2013 года на конференции GoogleI/O.

IDE находилась в свободном доступе начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина AndroidDevelopmentTools (ADT) для Eclipse.

AndroidStudio основана на программном обеспечении IntelliJIDEA от компании JetBrains и является официальным средством разработки Android-приложений. Данная среда разработки доступна для Windows, OS X и Linux.

AndroidStudio предлагает ряд функций, повышающих производительность при создании приложений для Android, таких как:

– гибкая система сборки на основе Gradle;

– быстрый и многофункциональный эмулятор Android-устройств;

– единая среда, в которой можно разрабатывать ПО для всех устройств Android;

– InstantRun для внесения изменений в запущенное приложение без создания нового APK;

– обширные инструменты и платформы тестирования;

– инструменты Lint для отслеживания производительности, удобства использования, совместимости версий и других проблем;

– поддержка C ++ и NDK;

– встроенная поддержка облачной платформы Google, позволяющая легко интегрировать облачные системы обмена сообщениями Google и AppEngine;

– и т.д.

Каждый проект в AndroidStudio содержит один или несколько модулей с файлами исходного кода и файлами ресурсов. Типы модулей включают:

– модули приложений Android;

– библиотечные модули;

– модули GoogleAppEngine.

По умолчанию AndroidStudio отображает файлы проекта в представлении проекта Android (рисунок 3). Это представление организовано модулями, чтобы обеспечить быстрый доступ к исходным файлам проекта.

Все файлы сборки видны на верхнем уровне под GradleScripts, и каждый модуль приложения содержит следующие папки:

– manifests - содержит файл AndroidManifest.xml;

– java – содержит файлы исходного кода Java, включая тестовый код JUnit;

– res - содержит все не-кодовые ресурсы, такие как XML-макеты, строки пользовательского интерфейса и растровые изображения.

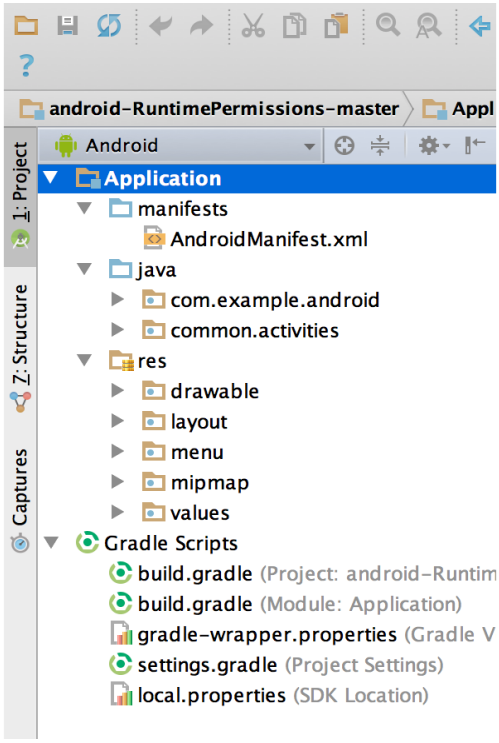


Рисунок 3 — отображение файлов в проекте AndroidStudio

Главное окно AndroidStudio состоит из нескольких логических областей (рисунок 4). Перечислим их.

1. Панель инструментов позволяет выполнять широкий спектр действий, включая запуск приложения и запуск инструментов Android.

2. Навигационная панель помогает перемещаться по проекту и открывать файлы для редактирования. Он обеспечивает более компактный вид структуры, видимой в окне «Проект».

3. В окне редактора можно создать и изменить код. В зависимости от типа текущего файла тип редактора может изменяться. Например, при просмотре файла макета в редакторе отображается редактор макетов.

4. Панель инструментов работает вокруг области окна IDE и содержит кнопки, которые позволяют разворачивать или сворачивать отдельные окна инструментов.

5. Окна инструментов предоставляют доступ к определенным задачам, таким как управление проектами, поиск, контроль версий и многое другое.

6. Строка состояния отображает статус проекта и самой IDE, а также любые предупреждения или сообщения.

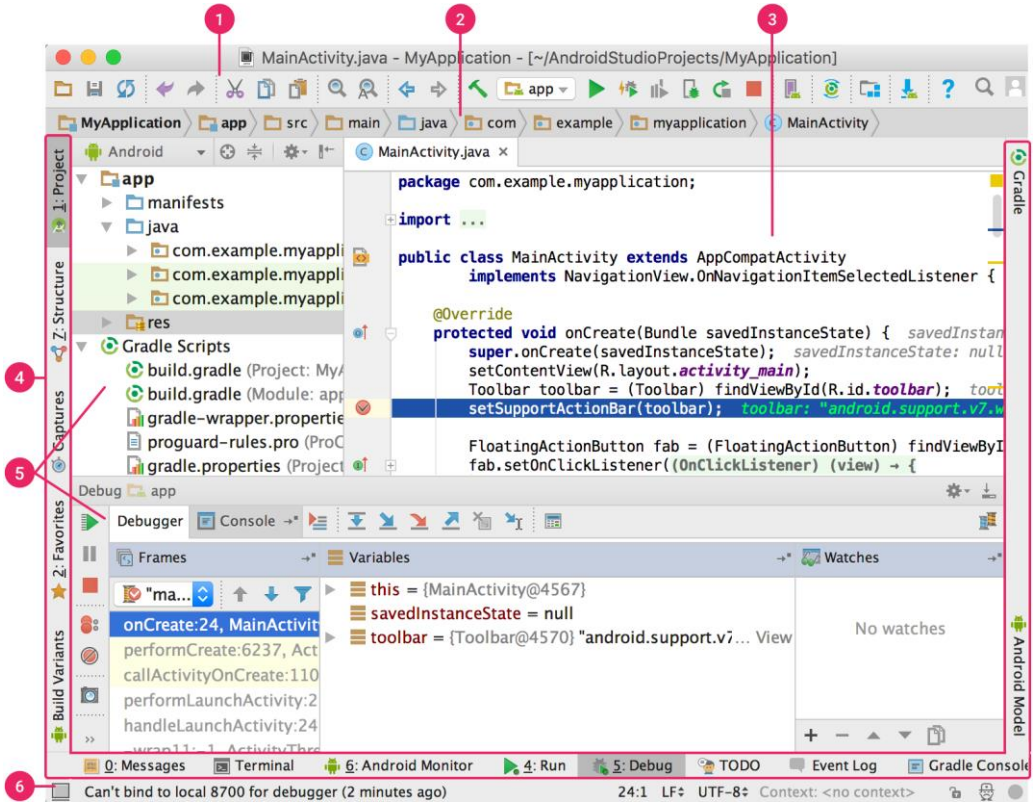


Рисунок 4 — главное окно IDEAndroidStudio

Можно организовать элементы главного окна таким образом, чтобы получить больше места на экране, скрывая или перемещая панели инструментов и окна инструментов. Также можно использовать быстрые клавиши для доступа к большинству функций IDE.

В любое время можно выполнять поиск по исходному коду, базам данных, действиям, элементам пользовательского интерфейса и т.д.

1.6 Структуры данных JSON

Разработка структуры данных является одной из важнейших задач на начальном этапе разработки, зачастую именно данные определяют возможный функционал разрабатываемого приложения и принципы его работы.

Используемый в ВКР архитектурный шаблон проектирования MVP дает возможность отделить данные от их представления, что дает возможность хранить данные в любом удобном виде и изменять их без ущерба для основной логики программы.

Входные данные, поступающие от сервера, представляют собой набор данных в формате Json (рисунок 5). JSON – это простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999. JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам С-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других.

Рисунок 5 – пример данных в формате json

После получения данные представляются в виде потока данных, которые внутри себя содержат структуру классов.

1.7 Технология блокчейн и её защищенность

Технология блокчейн была создана, в частности, чтобы позволить людям, которые не доверяют друг другу, делиться ценными данными безопасным и защищенным способом. Это связано с тем, что блочная цепочка хранит данные, используя сложные математические и инновационные правила программного обеспечения, которые чрезвычайно сложно преодолеть посредством хакерской атаки для последующих манипуляций.

Но безопасность блокчейна — даже самых совершенных систем — может потерпеть неудачу в тех местах, где причудливые математические и программные правила вступают в контакт с людьми, которые являются опытными читерами. В реальном мире все намного легче может лишиться упорядоченности.

Чтобы понять, почему, стоит начать с попытки выяснить, что же делает блочную технологию «безопасной» в принципе. Хорошим примером в этом случае является Биткоин. В его блокчейне хранятся общие данные — это история каждой транзакции монет. Его базу данных также называют распределенной книгой. Она хранится в большом количестве экземплярах в сети пользователей

криптовалютой, называемых «узлами». Каждый раз, когда кто-то отправляет транзакцию в регистр, узлы проверяют, чтобы убедиться, что она действительна.

Большое количество таких пользователей конкурируют за размещение действительных транзакций в блоки и добавление их в цепочку из предыдущих. Владельцы таких сетевых узлов называются майнерами. В качестве вознаграждения за участие в поддержании работы сети они получают выплаты в Биткоине.

Систему делают теоретически защищенной от несанкционированного доступа две вещи: криптографический код, уникальный для каждого блока, и «консенсусный протокол». Это процесс, посредством которого узлы в сети согласуются с общей историей транзакций.

Уникальный код (хеш) требует много времени и энергии для генерации. Он служит доказательством того, что майнер, добавивший блок в цепочку, проделал определенный объем вычислительной работы, чтобы заработать вознаграждение. По этой причине алгоритм Биткоина носит название «доказательство выполнения работы». Он также служит своего рода печатью, поскольку для изменения блока потребуется генерация нового хеша. Проверка хеша на соответствие определенному блоку очень проста, и как только узлы завершают этот процесс, они обновляют свои соответствующие копии блочной цепочки, после присоединения нового блока. Этот алгоритм носит название «консенсусный протокол».

Последний элемент безопасности состоит в том, что все хеши также служат связями в блок-цепочке. Каждый блок включает в себя уникальный хеш предыдущего блока. Поэтому, для внесения изменений в сеть потребуется вычислить новый хеш не только для блока, в котором он находится, но и для каждого последующего блока. И сделать это нужно до того, как другие узлы добавят следующий блок в цепочку. Поэтому, такой процесс требует огромной вычислительной мощности, которой современные компьютеры пока не обладают. Но даже в этом случае гарантировать успех невозможно. Добавляемые блоки будут конфликтовать с существующими, а другие узлы автоматически

отклонят вносимые изменения. Это делает blockchain защищенным от несанкционированного доступа и любых попыток изменить хранящуюся в нем информацию.

1.7.1 Публичные блокчейны

Самое основное условие — в публичном блокчейне нет контролирующего органа. Здесь надзирает за протекающими в системе процессами все сообщество — от майнеров, и разработчиков, до поставщиков услуг и простых пользователей. Уникальное свойство такой сети — возможность создавать децентрализованные приложения, которым на техобслуживание требуется минимум затрат. Доступ к публичному блокчейну имеет каждый желающий.

Публичный blockchain может просмотреть кто-угодно из любого уголка мира. Также каждый имеет возможность создать в нем транзакцию. Более того такая система позволяет любому пользователю принимать участие в процессе консенсуса и определять какие блоки будут добавлены в сеть, а какие отклонены. Защищенность таких систем обеспечивается криптографическими вычислениями. Наиболее распространенные алгоритмы — proof of work или proof of stake.

Другие преимущества публичных или открытых блокчейнов:

- Защита от вмешательства разработчиков. В отдельных моментах создатели системы никак не могут повлиять на нее и что-либо поменять в коде или данных на свое усмотрение. А это устраняет возможность цензуры. Однако недаром Томас Шеллинг как-то сказал: «Слабость иногда может обернуться преимуществом». Ведь в результате уровень доверия простых пользователей к системе повышается на порядок.

- Мощный сетевой эффект. В подобной среде разработчику легко собрать вокруг своего приложения обширную пользовательскую базу, поскольку пользователи одних приложений системы быстро узнают о других только что

созданных. Для этого достаточно разрешить своей программе, например, мобильному кошельку, взаимодействовать с уже работающим софтом.

– Безопасность, для поддержания которой необходимо небольшое количество средств. А вот чтобы атаковать такую систему потребуются слишком мощные вычислительные ресурсы, что делает атаку попросту невыгодной для злоумышленников. На сегодня самой безопасной из подобных систем считается биткоин.

Кроме того, как утверждает основатель Bitcoin Foundation Йон Матонис, существование публичных blockchain способствует более быстрому развитию инновационных технологий. Значительным прорывом стало использование смарт-контрактов, что дает возможность устранить посредников при проведении сделки.

Возьмем к примеру, две компании. Компания А для компании В разрабатывает дорогостоящий софт. Если А перешлет свою разработку первой, есть вероятность, что денег так и не увидит за выполненную работу. Если В отправит первым оплату, рискует остаться и без денег, и без софта. Раньше в такой ситуации требовались посредники, которые забирали свой процент от сделки. Сегодня можно полностью обойтись без посреднических услуг, доверившись смарт-контрактам. Они автоматически произведут обмен, предварительно проверив выполнение всех условий.

1.7.2 Приватные блокчейны

Приватные блокчейны характеризуются ограниченным уровнем допуска. Подтверждение транзакций в таких сетях, проведение аудита, управление базами доступно четко определенному кругу лиц. Если говорить о чтении, то такое право могут сделать как общедоступным, так и жестко ограничить. Таким образом это уже централизованная система, в которой речи уже не идет о равноправии и полной открытости, что присуще классическим сетям.

Преимущества приватных блокчейнов:

– Проверенные валидаторы, которые не смогу спровоцировать риск атаки 51%.

– Удешевление транзакций, поскольку проверяются они всего лишь небольшим количеством высокопроизводительных узлов.

– В приватной системе параметр TPS (transactions per second) можно сделать на порядок большим чем в открытых сетях. Единственным ограничивающим фактором в данной ситуации будет пропускная способность наиболее слабого узла платформы.

– Высокая скорость подтверждения транзакций.

– Откат записей. Компания, которой принадлежит приватная сеть, сможет при необходимости отменить транзакцию, внести изменения в баланс. Например, при фиксации прав на земельные участки эта возможность просто необходима, поскольку иначе земля может оказаться в недобросовестных руках.

Контроль сети единым центром удобен и в том плане, что дает возможность оперативно обновлять и улучшать функциональность системы, что особенно привлекательно для организаций, имеющих дело с системами учета — так они могут прогнозировать дальнейшие действия.

Еще один нюанс закрытых систем — для их эффективного функционирования вовсе не требуется алгоритм Доказательства работы (proof-of-work). Его могут подключить только по желанию, чтобы упростить аудит и повысить защищенность сети. В этом случае доверие пользователей уже не базируется исключительно на доверии к единому органу в виде организации, а проистекает из строгих математических законов.

Возьмем BitShares. В этой сети существует строго очерченное количество валидаторов, и каждый из них имеет свою пару ключей — один секретный, второй открытый. При формировании блоков в сети соблюдается очередность. Если кто-то из узлов не вложился в отведенный промежуток времени, он данный

цикл пропускает. Таким образом определить кто создал тот или иной блок можно по цифровой подписи в заголовке.

Сторонники закрытых систем считают, что эта технология со всеми ее возможностями способна уже в ближайшем будущем если не заменить, то полностью интегрироваться в ряд централизованных бизнесов. В первую очередь это касается учреждений, которые используют всевозможные реестры или системы учета финансов. Внедрение приватного blockchain в отличие от биткоина дает возможность подобным организациям, по их мнению, придерживаться требований в вопросах отмывания денег, соблюдать нормы закона «О перемещаемости и подотчётности страхования жизни».

Руководитель Lisk Макс Кордек утверждает, что «традиционные институты сразу переключиться на публичную систему блокчейна не в состоянии. И приватный blockchain — это логичный шаг на пути в криптографическое будущее». Он позволяет отслеживать ошибки и при этом исключает возможность подделки данных. К тому же закрытый блокчейн намного быстрее и дешевле открытого.

Противники приватных блокчейнов считают, что само их существование противоречит основным преимуществам технологии, таким как открытость, свобода от ограничений и безопасность. Некоторые даже считают, что это и вовсе не блокчейн, а просто технология распределенного реестра.

Они уверены, что общедоступный blockchain в любом случае будет на шаг впереди по развитию, чем закрытые системы хотя бы потому, что к их разработке и поддержанию может присоединиться любой желающий.

Популярно и мнение, что закрытый блокчейн предназначен лишь для удешевления расходов компаний, тогда как публичный способен совершить революцию в современном мире.

Эксперты не менее категоричны также в том, что приватным блокчейнам придется коммуницировать с внешним миром, выходить за границы своего четко

очерченного пространства. Впрочем, в данный момент большая часть приватных блокчейнов находится на экспериментальной стадии и когда они будут внедрены — вопрос открытый.

1.7.3 Плюсы и минусы технологии блокчейн и её перспективы

Чтобы в полной мере понять, что такое технология блокчейн и стоит ли становиться частью этой системы, необходимо разобрать существующие преимущества и недостатки, которые были подтверждены путем многочисленных исследований. Система блокчейн постоянно развивается и занимает все больше областей, вовлекая в свою цепь новых участников. Многие предприниматели считают, что если их компания не станет частью блокчейна, то можно остаться в стороне от мировых трендов.

Эксперты уверяют, что внедрение блокчейна по своему возможному эффекту не уступает открытию интернета, просто нужно еще немного времени для осознания этого. К плюсам блокчейна можно отнести:

1. Децентрализация. Участники сети имеют равные права и способны обмениваться сведениями непосредственно друг с другом, без участия посредников.

2. Надёжность. Вероятность взлома злоумышленниками или замена информации в блоках исключена, что обусловлено применением шифрованных ключей.

3. Прозрачность. Блоки цепочки блокчейн открыты для пользователей, а транзакции легко проверить.

4. Универсальность. Блокчейн — уникальная технология, которая может применяться не только в финансовой сфере, но и в других областях.

5. Высокая скорость. Возможности блокчейн-технологии позволяют снизить время транзакций до одной минуты.

6. Низкие комиссии. Благодаря отсутствию комиссионных платежей, транзакционные сборы минимальны.

Поскольку система только развивается, минусов не избежать:

1. Невозвратность. Отменить операцию, если она проведена по ошибке, не получится.

2. Риск атаки. Если 51 процентом цепи будет владеть один пользователь, целостность сети нарушается.

3. Масштабируемость. При текущем размере блока система обрабатывает до семи операций за 1 секунду. С ростом числа пользователей этого показателя становится мало.

4. Неопределенность статуса. Многие страны никак не определяться с применением технологии.

5. Новизна. Пока еще сложно найти разработчиков, которые бы быстро и без ошибок справились с работой. К тому же для поддержания системы необходимы специалисты, которых также мало.

Благодаря своим многообещающим возможностям, в 2017 году блокчейн продолжает оставаться в центре внимания и по-прежнему вызывает бурный ажиотаж у сообщества.

В развитии блокчейна значительную лепту внес Эфириум, позволяющий создавать децентрализованные приложения и управлять цифровыми активами без участия третьей стороны в передаче кода, в том числе при программировании логики работы обменных сервисов, как это и характерно для умных контрактов. Наряду с цифровыми активами становится возможным существование множества других типов записей, включая подписи, доказательства существования и тому подобное.

Приобретая фантастические возможности, технология вселяет ещё более смелые надежды, но одновременно и пугает своим могуществом, становясь потенциальной угрозой для «патриархальных» социальных структур.

Очередным шагом развития блокчейнов станет разработка механизма юридического признания субъектов, созданных на их платформах. Отчего могут возникнуть дополнительные проблемы.

2 РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

В данной главе описана реализация клиентской и серверной частей приложения. Особое внимание уделяется выделению и описанию различных сущностей, и их взаимодействию между собой. При этом освещены основные особенности реализации, связанные с программированием на выбранных языках. Целью данного раздела является объяснить принцип работы и внутреннее устройство системы в целом.

2.1 Постановка задачи

Необходимо разработать клиентское и серверное приложения, работающие по протоколу HTTP. Серверное приложение должно работать с базой данных. Доступ к базе данных должен осуществляться посредством добавления в нее новых записей и получения необходимых записей. Каждая следующая запись должна зависеть от предыдущей. Все действия проверки подлинности и шифрации записей должны осуществляться на клиенте, которое представлено в виде мобильного приложение на ОС Android.

Многомодульность программы. Необходимо разбить программу на модули. Это повысит наглядность и легкость восприятия кода программы, а также облегчит поиск и исправление возможных ошибок.

Режим работы видеосистемы-графический.

Пользовательский интерфейс. Необходимо определиться с выбором доступного и интуитивно понятного меню для возможности эффективной работы с программой.

Операционная система - MS Windows(сервер), OS Android(клиент)

Среда программирования – Notepad++ (сервер), Android Studio(клиент)

Язык программирования – Python(сервер), Java(клиент)

2.2 Структура данных приложений

Для реализации клиентской части программы был выбран язык программирования Java, который поддерживается средой разработки Android Studio.

Взаимодействие с пользователем построено на использовании кнопок, всплывающих окон и текстовых полей для ввода.

На рисунке 6 показаны UML-диаграммы вариантов использования клиентской программы.

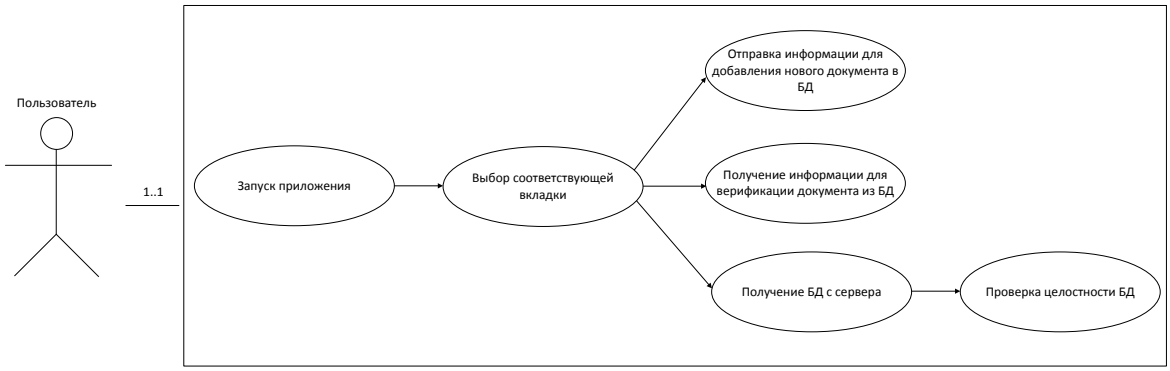


Рисунок 6 - UML-диаграмма вариантов использования клиентской программы

Программа-клиент представляет собой несколько Activity, для перехода к которым можно использовать вкладки сверху. Каждая Activity имеет свой функционал. В частности, первая предназначена для добавления хэша документа в БД, вторая - для верификации документа, а третья – для проверки целостности БД.

На рисунке 7 показана UML-диаграмма вариантов использования серверной программы.

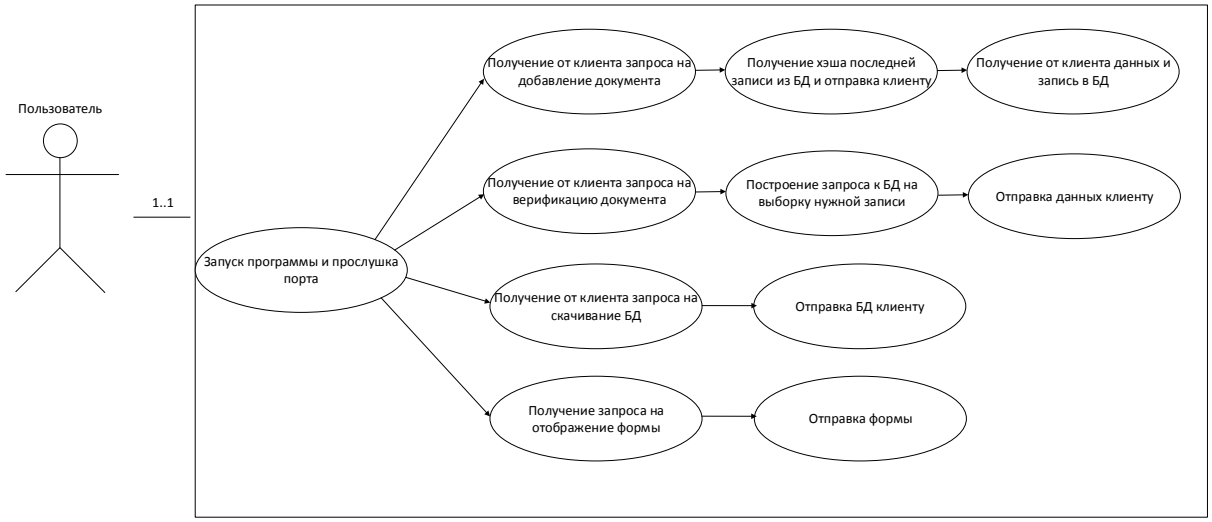


Рисунок 7 - UML-диаграмма вариантов использования серверной программы

Программа-сервер по запросам от клиентов на добавление, и верификацию документов формирует запросы к БД и отправляет результат обратно пользователям. Для проверки её целостности клиент может сказать всю базу данных в виде файла.

2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса

Программа-клиент представлена в виде форм Activity, которая связана с кодом на Java, осуществляющим функционал. При входе в приложение пользователь видит первую Activity с кнопками и полями для добавления документа в БД, а с помощью вкладок сверху быстро и удобно может переключиться на другую задачу.

На рисунке 8 изображена UML-диаграмма состояний программы-клиента.

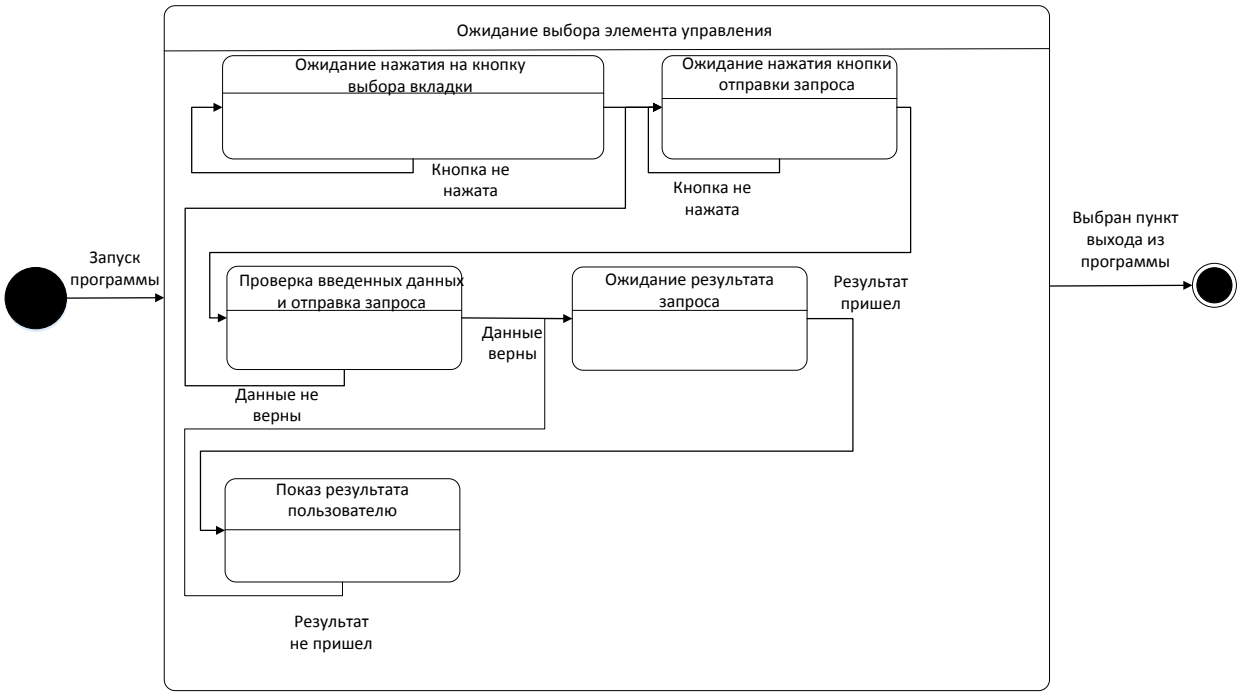


Рисунок 8 - UML-диаграмма состояний программы-клиента

На рисунке 9 изображена UML-диаграмма состояний программы-сервера.

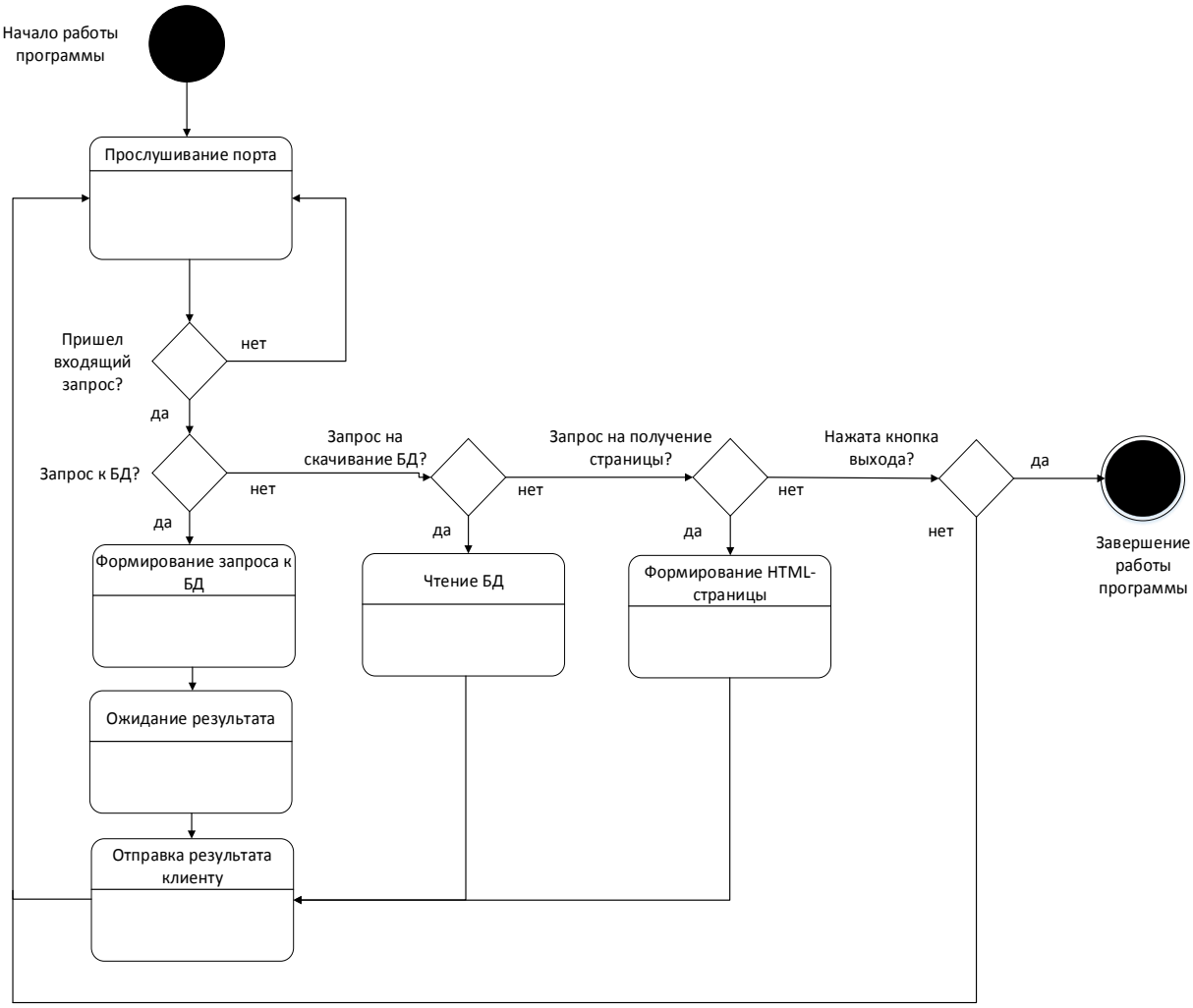


Рисунок 9 - UML-диаграмма состояний программы-сервера

2.4 Описание разработки программы

Для написания программы-клиента был использован язык программирования Java. Для разработки программы-сервера использовался язык программирования Python.

В качестве среды программирования сервера была выбрана программа для редактирования кода Notepad++, а для разработки клиента – среда разработки Android Studio.

Интерфейс программы клиента представлен в виде экранной формы Activity с элементами управления, так как этот способ представления функций наиболее удобен для восприятия и может быть легко освоен даже не подготовленным пользователем.

При запуске программы-клиента (запуск приложения) на экране появляется первая экранная форма Activity с кнопками выбора файлов, отправки запросов и генерации ключей, текстовыми полями для отображения ключей, полями вывода для данных, полученных от сервера. После запуска мобильного приложения пользователь имеет возможность выбрать нужную ему вкладку: «Добавить документ», «Проверить документ», «Проверить БД», а затем на ней уже выполнить необходимые ему действия в соответствии с возможностями вкладки.

Для добавления документа в блокчейн необходимо перейти в первую вкладку и нажать на кнопку “Выбрать файл”. После нажатия на кнопку выбора файла будет запущен менеджер выбора файла и открыто соответствующее окно. После выбора файла текстовое поле “Хэш файла” заполнится автоматически рассчитанным хэшем выбранного файла. Далее следует сгенерировать открытый и закрытый ключи, нажав на кнопку “Сгенерировать ключи” или, если они уже имеются, вставить их в соответствующие поля. Следом стоит нажать на кнопку “Получить хэш последней строки”, после чего на форме появится строка с хэшем транзакции в БД, которая будет входить в состав формируемой транзакции для реализации структуры цепочки блоков. Следом нужно нажать на кнопку “Подписать”, и в текстовом поле “Подпись” появится соответствующее зашифрованное сообщение. После нажатия на кнопку “Добавить транзакцию” данные будут отправлены на сервер, после чего от сервера придет ответ о том, что транзакция добавлена с указанием её ID или о необходимости повторного формирования хэша последней транзакции и подписи. ID транзакции документа как и закрытый ключ необходимо сохранить, так как он понадобится для последующей верификации документа.

Для верификации документа необходимо перейти во вкладку “Проверка документа”, нажать на кнопку выбора файла, после чего хэш выбранного файла так же автоматически рассчитается и заполнит текстовое поле “Хэш файла”. После этого необходимо ввести в текстовое поле “ID” ID транзакции, которое было возвращено сервером при добавлении файла в блокчейн и нажать на кнопку

“Получить хэш по ID”. В результате сервер пришлет хэш нужного файла или, если транзакции с таким ID в БД не будет, вернет соответствующее сообщение. В этом случае стоит проверить правильность введенного ID и повторить вышеперечисленные действия. После получения хэша файла необходимо нажать на кнопку “Сравнить хэши”. Если файл не был изменен с момента загрузки его в блокчейн, то будет показано диалоговое окно с текстом “документ не был изменен”, иначе будет показано “документ был изменен”.

Для проверки целостности БД следует перейти в область “Проверить БД”, нажать на кнопку “Загрузить БД” и база данных автоматически скачается. После загрузки БД следует нажать на кнопку выбора файла и выбрать загруженный блокчейн, а затем нажать на кнопку “Проверить БД”. Если БД не была повреждена, то появится диалоговое окно с текстом “Блокчейн не поврежден!”, в противном случае появится текст “Блокчейн поврежден!”

Программа-сервер принимает необходимые запросы от клиентов и, в зависимости от типа запроса, передает страницу клиенту, отправляет SQL-запрос в БД и возвращает результат, или отправляет файл базы на сторону клиента.

На рисунке 10 показана UML-диаграмма классов программы-клиента.

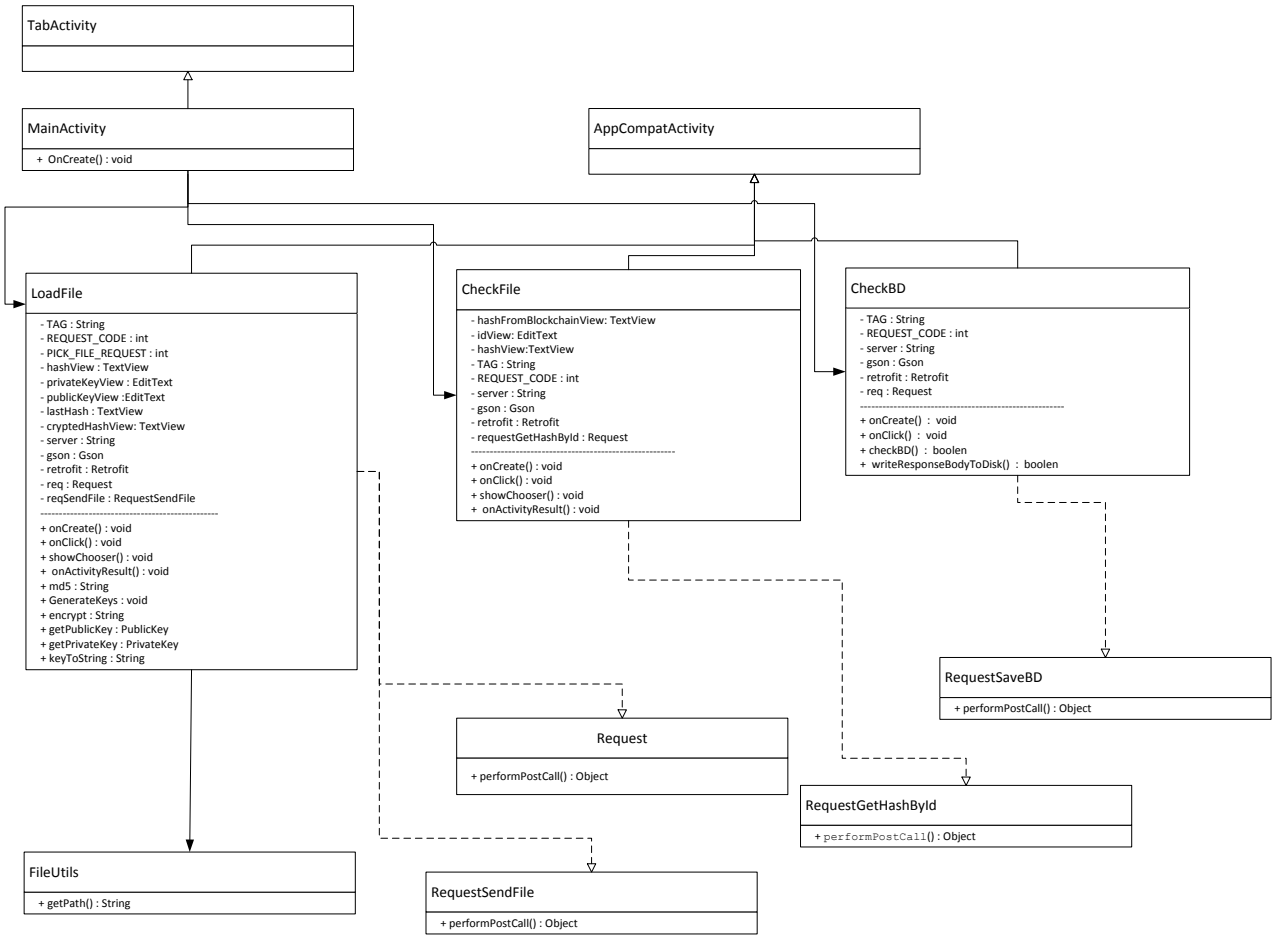


Рисунок 10 - UML-диаграмма классов программы-клиента

На рисунке 11 показана UML-диаграмма классов программы-сервера.

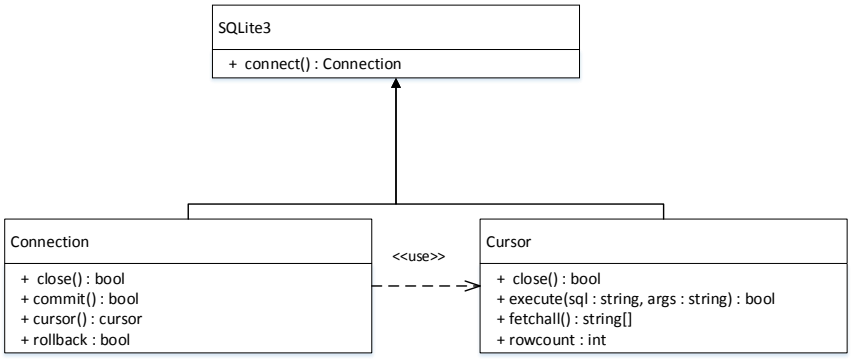


Рисунок 11 - UML-диаграмма классов программы-сервера

Для реалиации перехода на нужную вкладку класс MainActivity использует intent(намерение) на соответсвующие классы (LoadFile, CheckFile и CheckBD). Каждый из этих классов содержит свою локигу. В частнгости, для реализации открытия окна выбора файла был использован метод showChooser. Для

реализации подписи и хэширования методы encrypt, открытый и закрытый ключи для которого генерирует метод GenerateKeys, и md5 соотвественно. Отправка запроса на сервер происходит при помощи библиотеки Retrofit 2, которая удобна при формировании пост-запросов.

Серверное приложение использует библиотеки Json и SQLite3 для пересылки сообщений клиенту в удобном виде и взаимодействия с БД соответственно. В библиотеке SQLite3 используется одноименный класс SQLite3, реализующий функцию соединения с БД, а также вспомогательные классы Connection и Cursor, необходимые для удобного перебора возвращаемых запросом записей.

Взаимодействие объектов классов происходит посредством передачи различных переменных других классов в качестве параметров функций, а также использования объектов классов внутри собственных классов для возврата результата.

2.5 Отладка и тестирование

В качестве среды разработки программы-сервера был выбран редактор кода Notepad++, содержащий в себе различные средства для разработки и отладки модулей и программ. А для разработки программы-клиента была выбрана удобная и многофункциональная среда разработки Android Studio. Для отладки программы использовались различные инструменты: точка останова в браузере, пошаговое выполнение кода программы, анализ содержимого глобальных и локальных переменных, результаты работы некоторых функций, трассировка, вывод переменных в консоль.

Тестирование проводилось в рабочем порядке, в процессе разработки и после завершения написания программы, все выявленные ошибки были устранены.

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Описание клиентского приложения

Для запуска клиентского приложения пользователь по соответсвующей иконке и названию запускает приложение со своего рабочего стола телефона, после чего открывается стартовое окно.

Приложение может выполнять три основные функции: добавление документа в БД, верификация документа и проверка целостности блокчейна. Для выполнения каждой из функций от пользователя требуется определенный алгоритм действий.

Описание всех событий клиентского приложения показано в таблице 1.

Таблица 1 – описание состояний программы-клиента

Событие	Действие Пользователя	Действие программы
Выбор вкладки	Нажата вкладка	Переход на соответсвующую вкладку
Выбор файла	Нажата кнопка “Выбрать файл”	Выбор файла и хэширование его содержимого
Генерация открытого и закрытого ключей	Нажатие на кнопку “Сгенерировать ключи”	Генерация ключей
Получение хэша последней строки	Нажата кнопка “Получить хэш последней строки”	Отправка запроса к серверу, ожидание ответа, и показ результата в текстовом после “Хэш последней строки”
Подпись конкетенированной строки, состоящей из последнего хэша БД и хэша файла	Нажата кнопка “Подписать”	Шифрование закрытым ключом информации, результат в поле “Подписанный хэш”

Продолжение таблицы 1

Отправка данных на сервер для добавления транзакции в БД	Нажата кнопка “Отправить строку”	Отправка запроса серверу, ожидание ответа
Показ сообщения об успешном добавлении транзакции	-	Получение ответа от сервера об успешном добавлении транзакции и показ пользователю
Показ сообщения с просьбой повторить действия	-	Получение ответа от сервера о неуспешном добавлении транзакции и показ пользователю
Получение хэша файла по ID	Нажата кнопка “Получить хэш по ID”	Отправка запроса на сервер и ожидание результата
Показ сообщения о неправильно введенном ID	Неправильно введено ID документа в поле “ID”	Показ соответствующего сообщения пользователю
Сравнение хэшей и выдача соответствующего сообщения	Нажата кнопка “Сравнить хэши”	Сравнение хэшей в полях “Хэш файла из блокчейна” и “Хэш файла” и выдача соответствующего сообщения
Загрузка БД	Нажата кнопка “Загрузить БД”	Отправка запроса о загрузке БД на сервер и получение данных и вывод сообщения пользователю
Проверка целостности БД	Нажата кнопка “Проверить БД”	Проверка целостности БД и вывод соответствующего результата пользователю

Диаграмма активности программы-клиента приведена на рисунке 12.

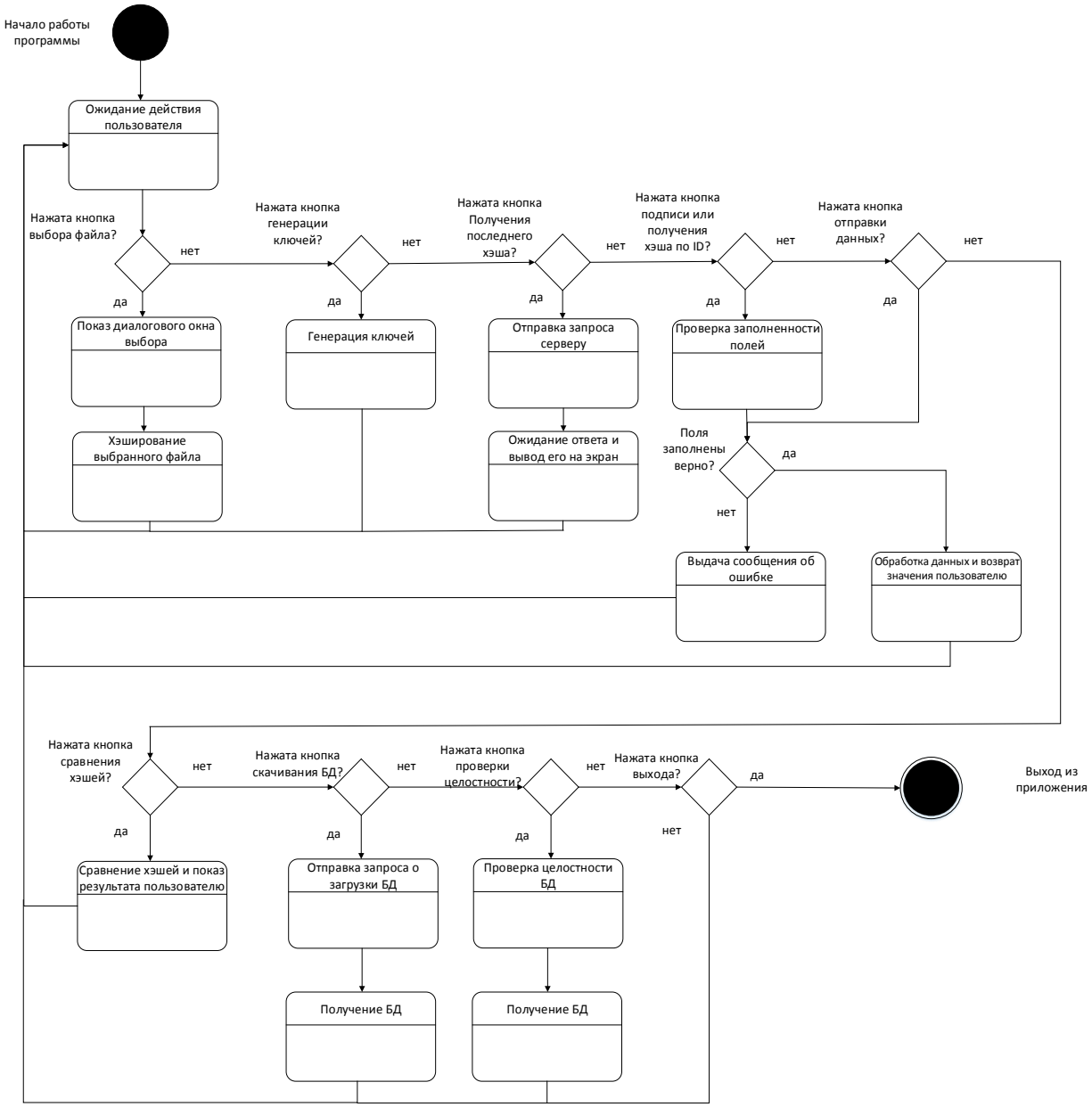


Рисунок 12 – UML- диаграмма активности программы-клиент

3.3 Описание серверного приложения

После запуска сервера, программа начинает прослушивать 5000-й порт и в зависимости от различных типов запросов от клиента, присылаемых на разные url, выполняет ту или иную функцию, а затем возвращает результат обратно клиенту.

Описание всех событий сервера выполнено в таблице 2.

Таблица 2 - описание состояний программы-сервера

Событие	Действие Пользователя	Действие программы
Начало прослушки порта	Запуск программы	Запуск сервера и прослушка 5000го порта
Получение запроса на добавление записи в БД	-	Получение хэша последней записи, расшифровка подписи публичным ключом и получение из подписи хэша последней записи. Если хэши совпадают записать транзакцию в БД, иначе послать пользователю хэш последней транзакции в БД и попросить подписать запись еще раз.
Получение запроса на верификацию записи в БД	-	Формирование запроса к БД по полученному ключу и отправка результата обратно.
Получение запроса на отправку формы	-	Отправка HTML- страницы.
Получение запроса на скачивание БД	-	Загрузка БД и отправка пользователю
Выход	Нажата кнопка "Выход"	Выход из приложения

Диаграмма активности программы-сервера приведена на рисунке 13.

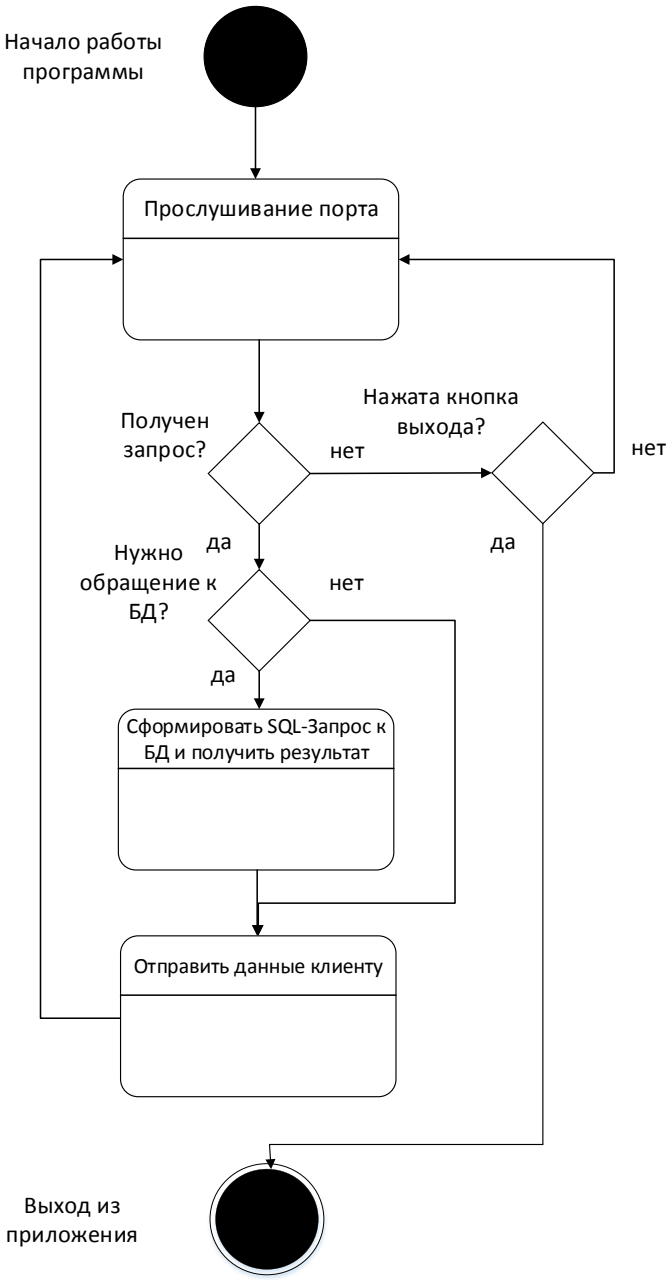


Рисунок 13 – UML- диаграмма активности программы-сервера

3.4 Руководство пользователя

При открытии клиентского приложения с телефона открывается стартовое окно (первая вкладка), содержащее кнопки и поля ввода.

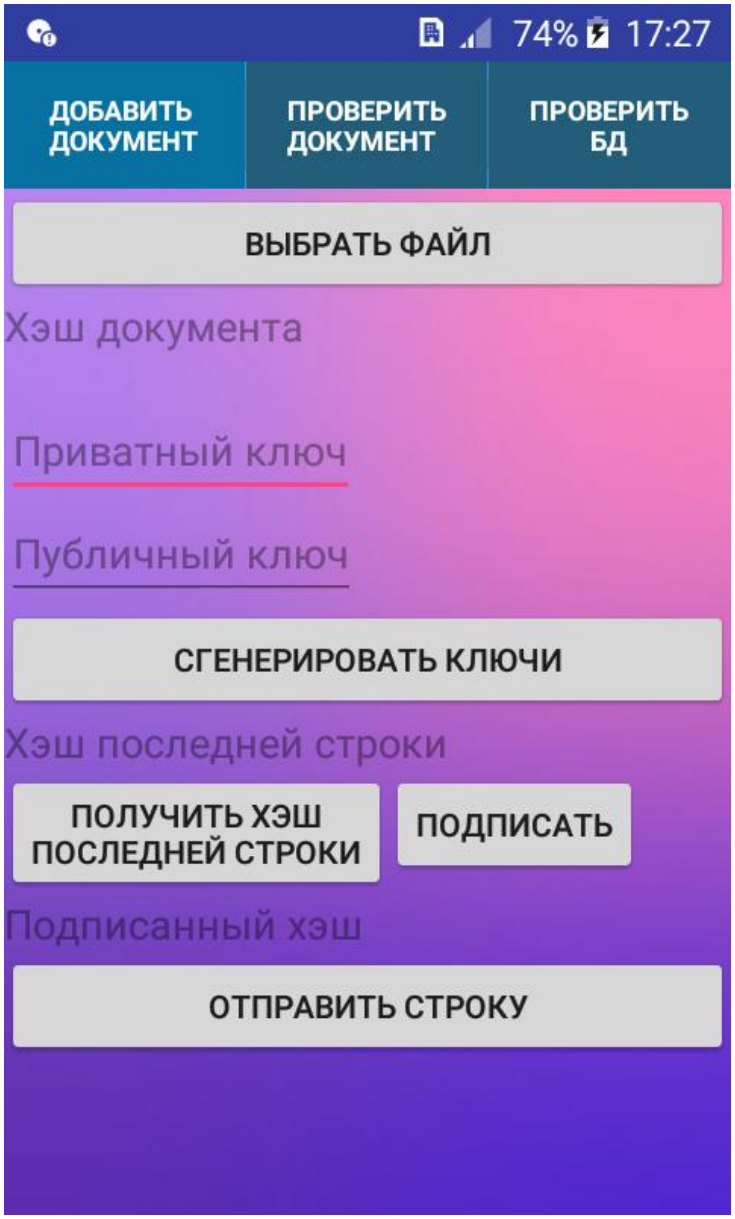


Рисунок 14 – Вкладка “Добавить документ” в клиентской программе

Для добавления нового файла в БД следует нажать на кнопку “Выбрать файл” и в диалоговом окне выбрать файл, который будет добавлен в блокчейн.

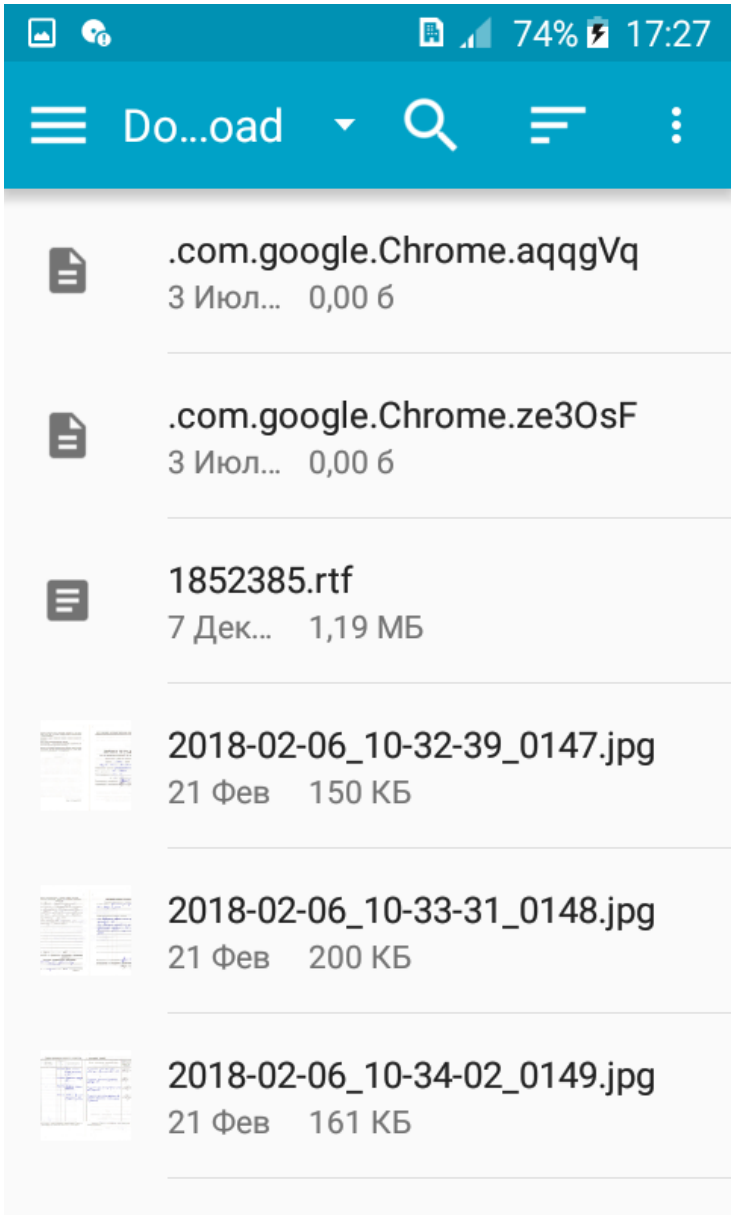


Рисунок 15 – Окно выбора файла

После указанных действий приложение с помощью встроенного кода автоматически рассчитает хэш файла и вставит в текстовое поле “Хэш документа”.

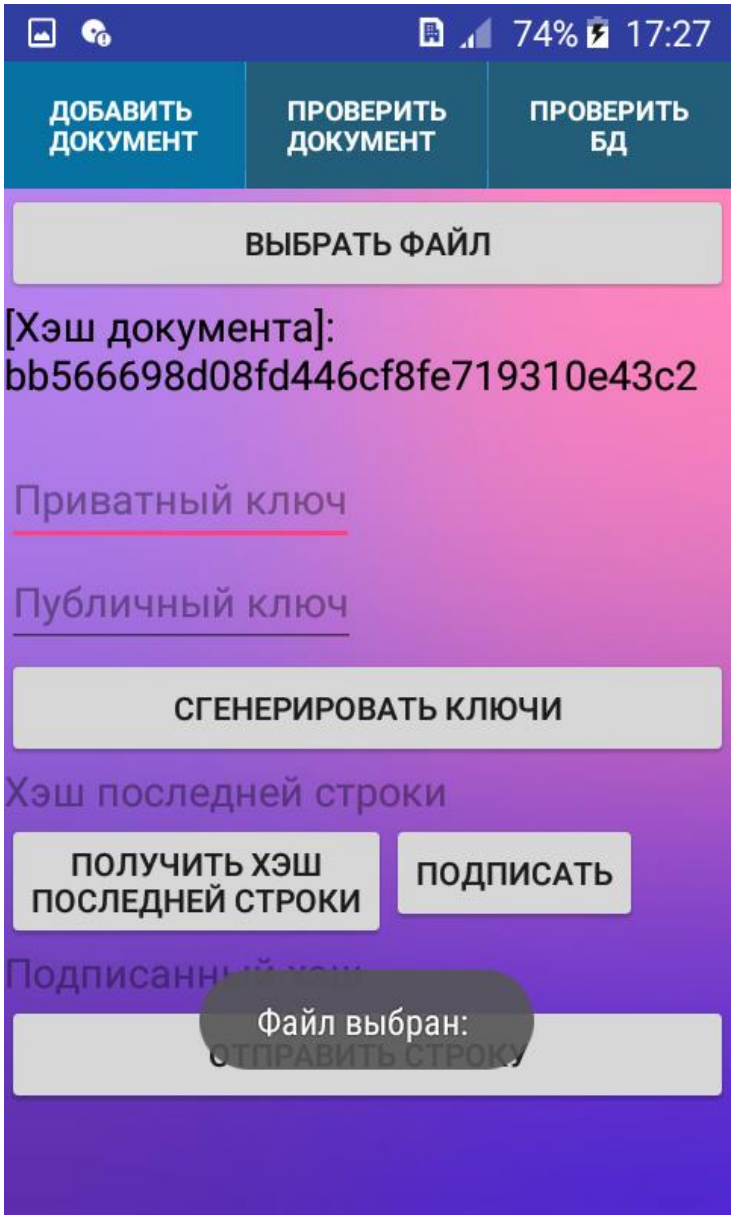


Рисунок 16 – автоматический расчет хэша файла

Затем следует ввести открытый и закрытый ключи в соответствующие поля “Приватный ключ” и “Публичный ключ ключ” или сгенерировать новые, нажав на кнопку “Сгенерировать ключи”.

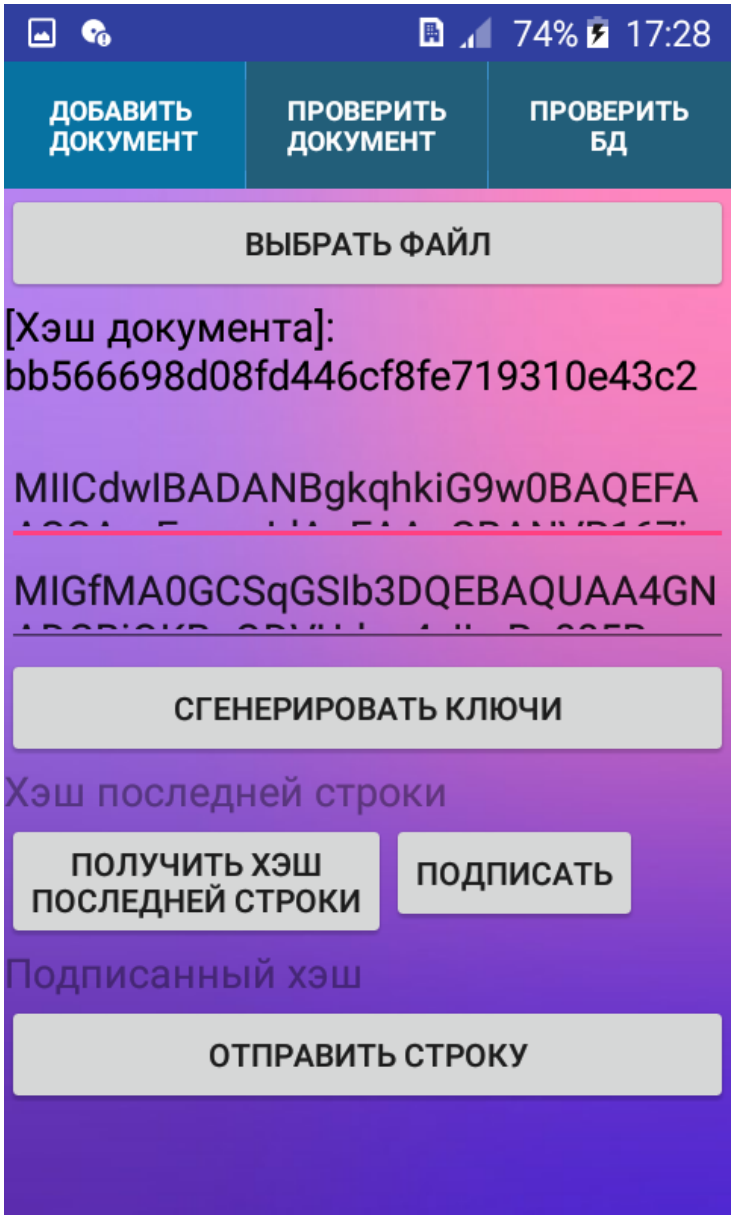


Рисунок 17 – генерация открытого и закрытого ключей

После генерации ключей необходимо получить хэш последней транзакции в БД, нажав на кнопку “Получить хэш последней строки”.

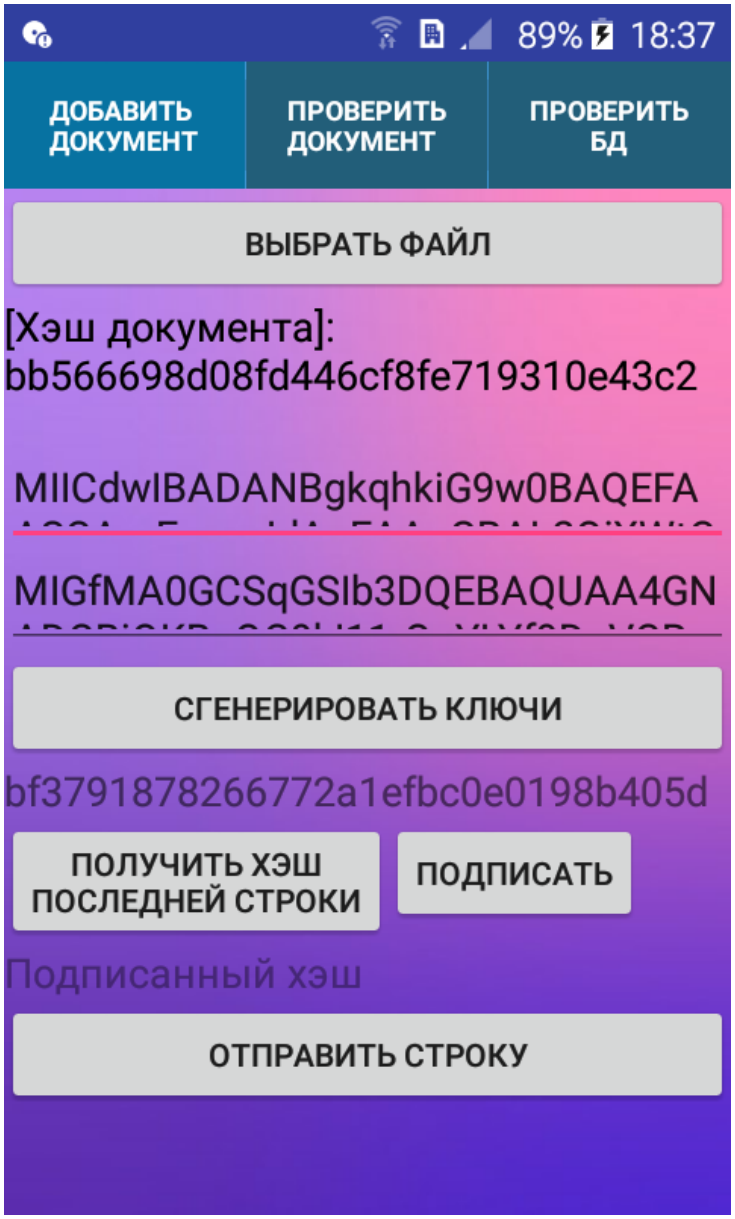


Рисунок 18 – получение хэша последней строки из БД.

После этого можно подписать хэш последней транзакции и файла закрытым ключом, нажав на кнопку “Подписать”. Если какое-либо из предыдущих полей останется незаполненным, то подписать не получится и появится сообщение о проблеме.

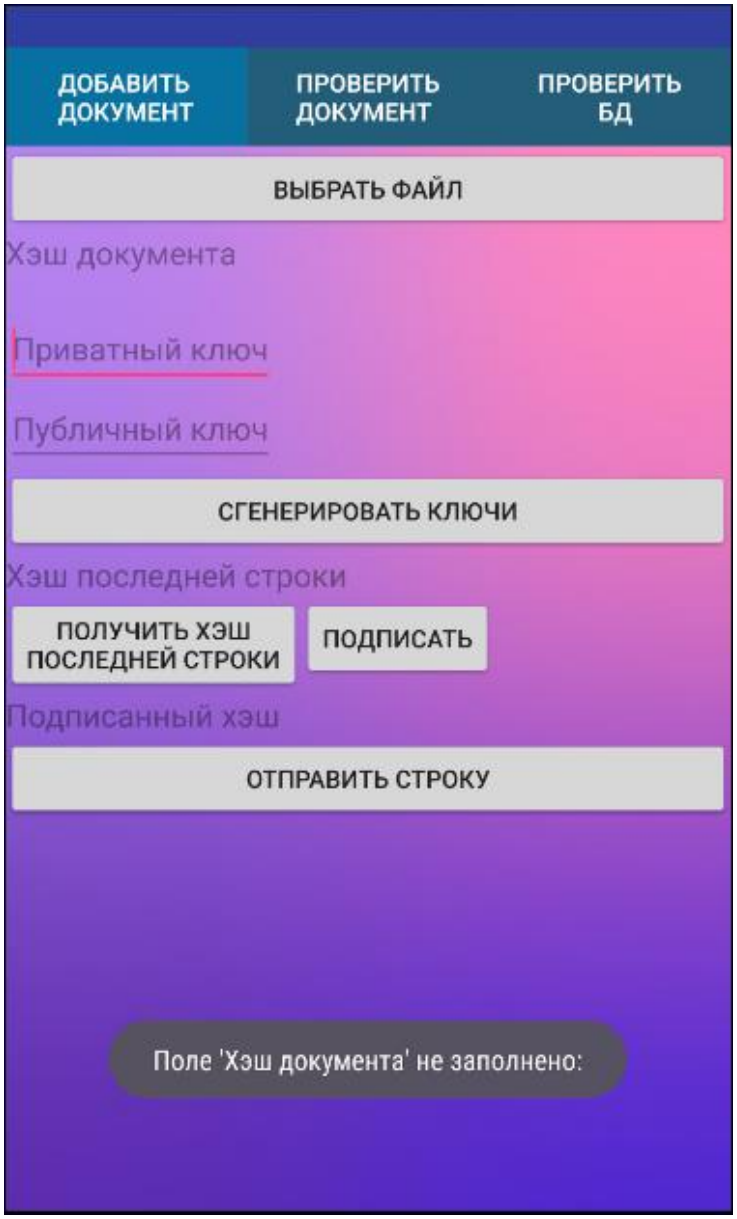


Рисунок 19 – попытка подписать данные пустым полем хэша.

Если все данные были заполнены, то подпись пройдет успешно и в поле “Подписанный хэш” появится соответсвующая информация.

Рисунок 20 – подпись данных

Следом нужно нажать на кнопку “Отправить строку” и получить ответ от сервера. В случае успешного добавления транзакции появится сообщение об успешном добавлении новой записи в БД. Если кроме текущего пользователя одновременно с ним какой-то еще пользователь пытается добавить файл в блокчейн, то срабатывает механизм синхронизации, в результате которого выбирается только один пользователь который добавляет свои данные в блокчейн, при этом с помощью публичного ключа расшифровывается подпись и проверяется хэш последней записи и хэш из расшифрованной подписи. Если хэши совпадают, то запись добавляется в БД и пользователю приходит

соответствующее сообщение, в противном случае пользователя просят о повторе получения хэша и подписи документа.

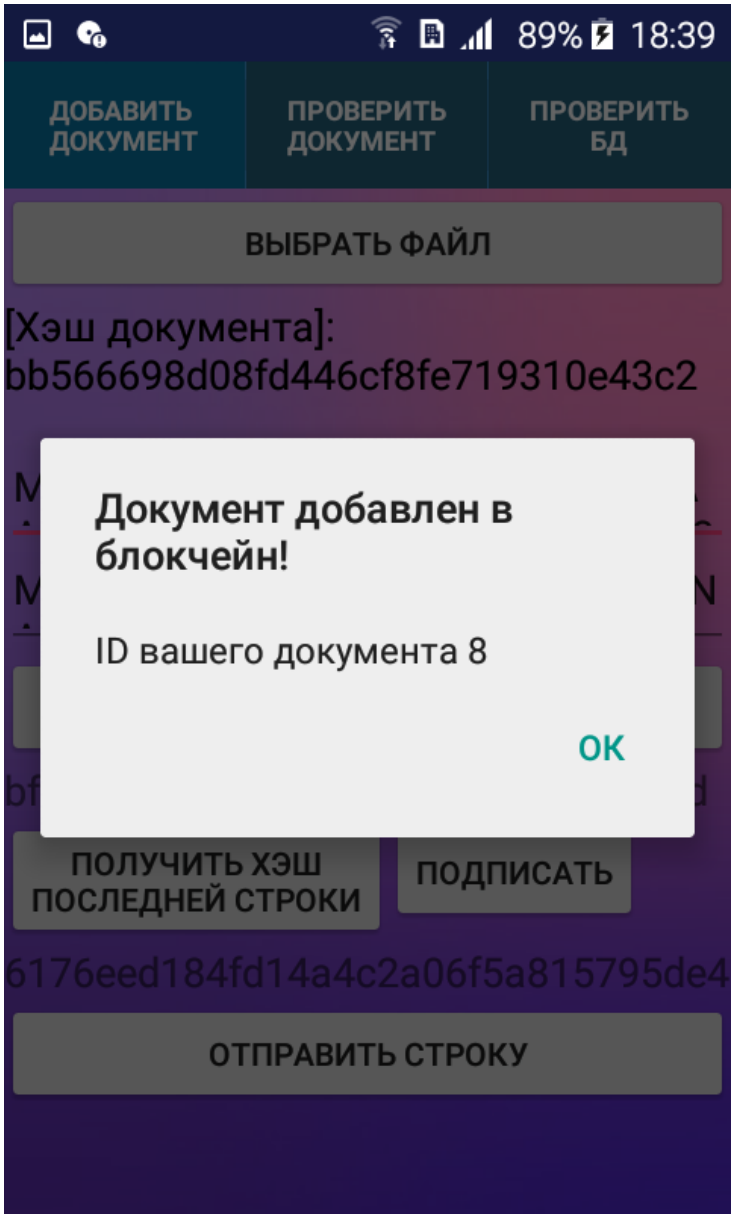


Рисунок 21 – успешное добавление транзакции в блокчейн

При успешном добавления файла пользователю следует сохранить полученное в сообщении ID, так оно пригодится для дальнейшей проверки документа на неизменность.

В случае неудачного добавления в блокчейн появится сообщение с просьбой повторить шаги получения последнего хэша файла, подписи, и отправки данных на сервер.

При необходимости проверки документа на неизменность необходимо перейти во вкладку “Проверить документ”.

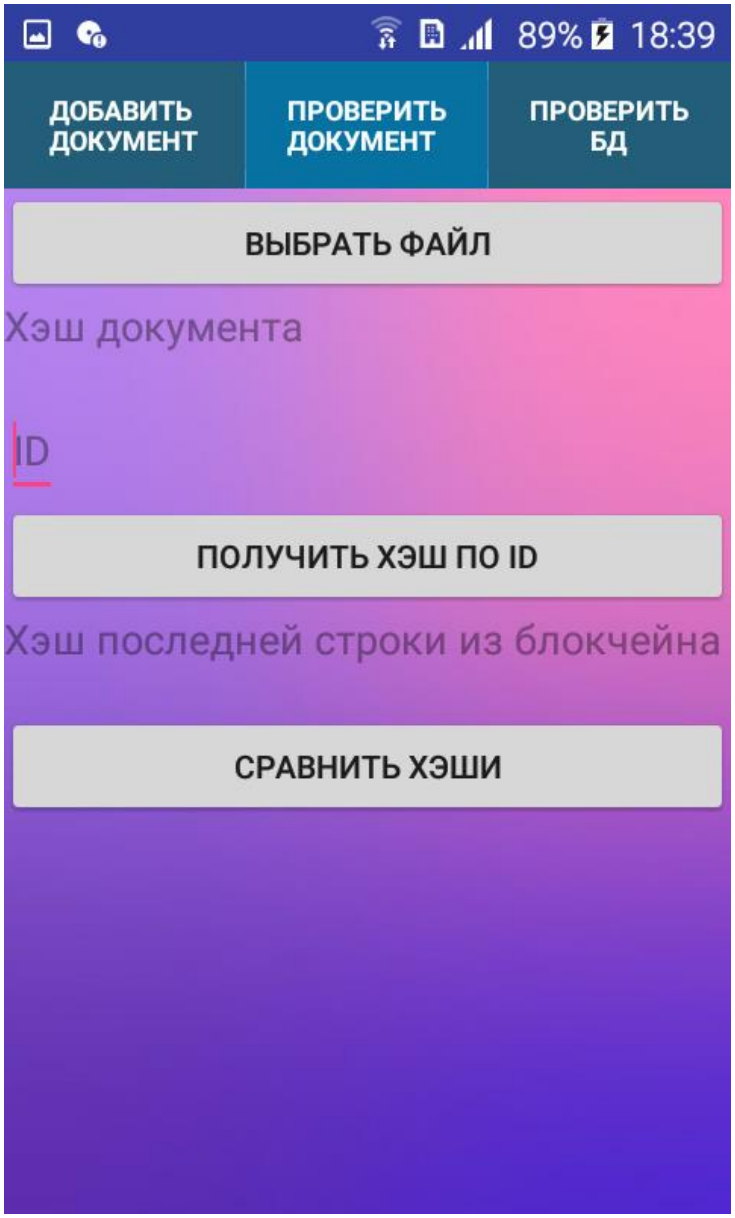


Рисунок 22 – раздел “Проверить документ”

Затем с помощью кнопки “Выбрать файл” нужно выбрать файл, который пользователь желает проверить. Хэширование содержимого произойдет автоматически и отобразится ниже.

Следом нужно нажать кнопку “Получить хэш по ID”, по запрошенному ID сервер выдаст хэш записи. В результате в поле “Хэш последней строки из блокчейна” отобразится нужное значение.

Рисунок 23 – получение хэша файла по ID из БД

После получения хэша можно нажать на кнопку “Сравнить хэши” и получить сообщение о неизменности документа.

Если документ не был изменен с момента записи его в блокчейн, хэш сохранится и появится диалоговое окно с сообщением “Документ не был изменен”.

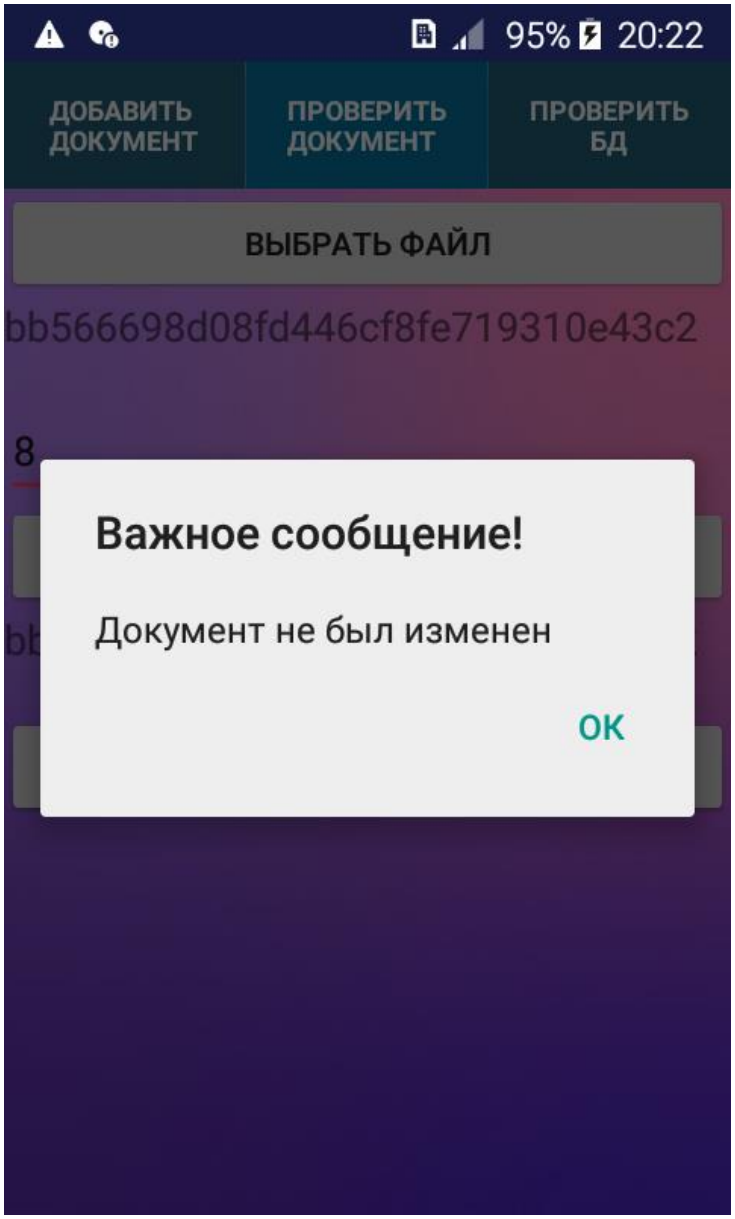


Рисунок 24 – вердикт об отсутствии изменений в документе

Если документ с момента записи его в БД по каким-то причинам изменялся, хэш изменится и в поле результат проверки выведется значение “Документ был изменен”.

Для проверки целостности самого блокчейна и обнаружения факта возможных попыток его изменения, пользователю необходимо перейти во вкладку “Проверить БД” и нажать на кнопку “Загрузить блокчейн”.

Следом нужно нажать на кнопку “Выбрать файл” и выбрать соответствующую скачаенный файл. Затем после нажатия на кнопку “Проверить БД” и дождаться результат проверки. В основе проверки лежит процедура последовательного вычисления хэша предыдущей строки, расшифровки

открытым ключом подписи следущей строки и извлечения хэша предыдущей строки. Следом идет сравнения полученных хэшей, в результате которого делается вердикт о неизменности записи.

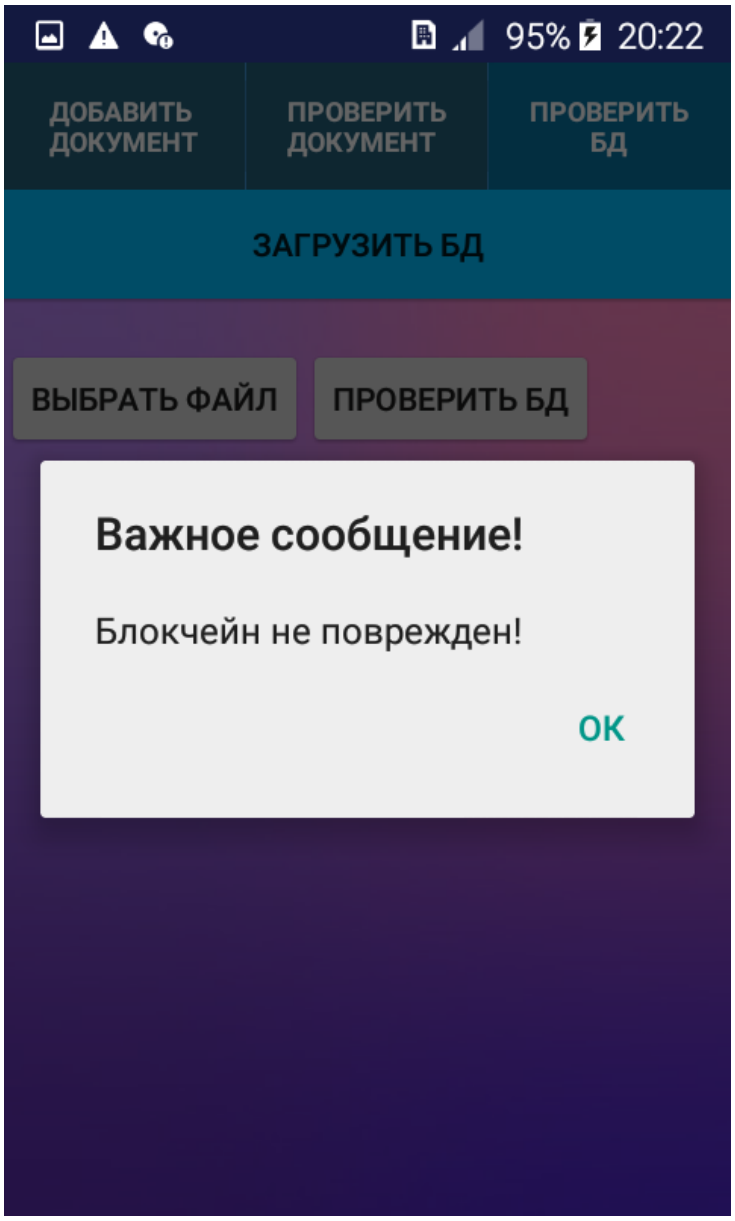


Рисунок 25 – сообщение о неизменности блокчейна

Для открытия серверного приложения необходимо через консоль дойти до папки программы и запустить Python скрипт на компьютере сервера.

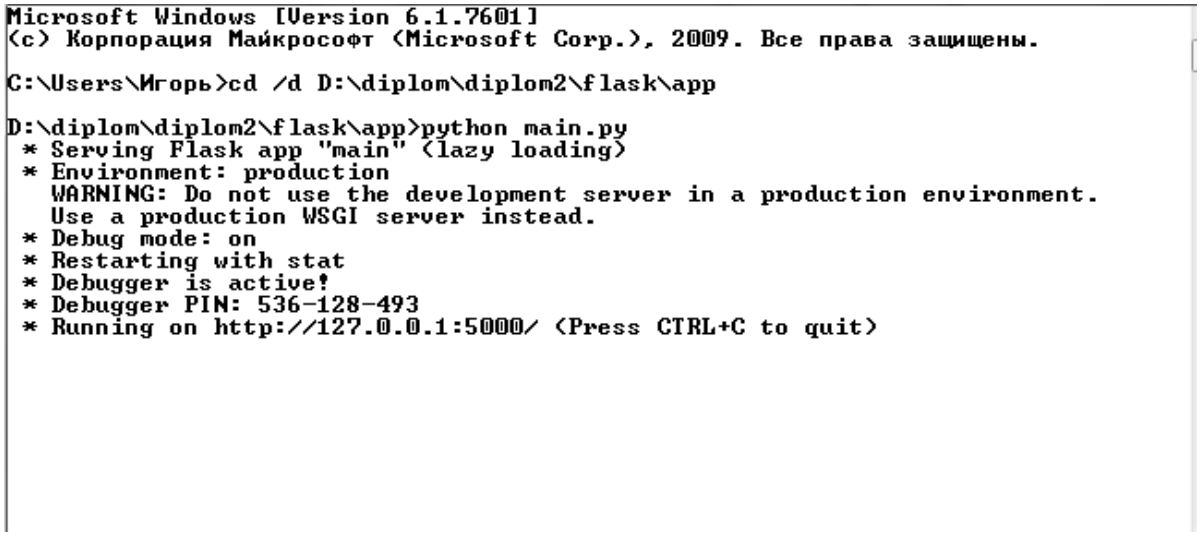


Рисунок 26 – запуск серверной программы

После этого в консоль выведется сообщение об успешном запуске программы и начнется прослушивание порта.

При получении различных запросов в консоль будет выводится ip-адрес источника запроса, и тип запроса.

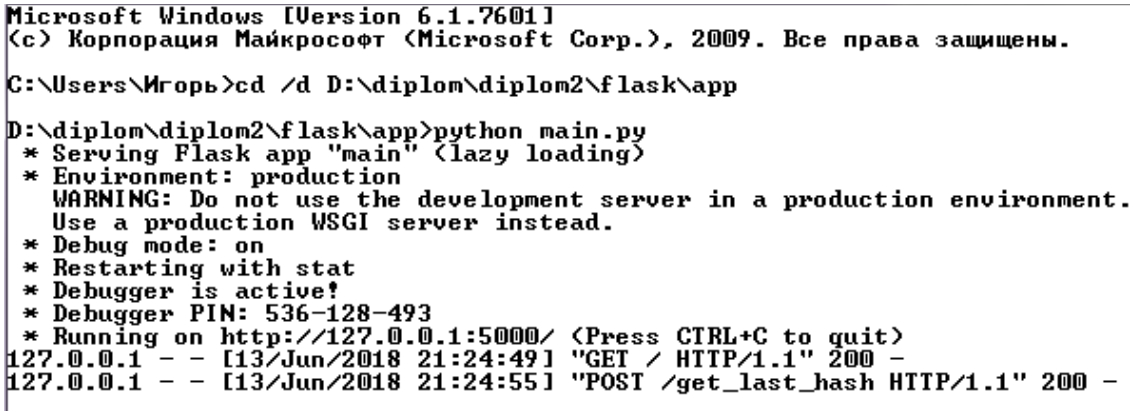


Рисунок 27 – принятие запросов сервером

Если возникла необходимость прекратить работу сервера, нужно нажать комбинацию клавиш Ctrl+C.

ЗАКЛЮЧЕНИЕ

Результатом работы является мобильное приложение с интуитивно понятным пользовательским интерфейсом, которое позволяет регистрировать файлы в БД и проверять их неизменность после добавления. Приложение обеспечивает невозможность внесения злоумышленником незаметных изменений в базу данных после добавления транзакции при отсутствии у злоумышленника технического превосходства. Данный проект полностью удовлетворяет поставленной цели.

К особенностям системы можно отнести установленную на мобильном устройстве ОС Android. Не каждый мобильный телефон работает под управлением данной ОС, но именно эта платформа в данный момент является самой популярной. Достоинством применения данной ОС является и то, что выпускаются телефоны разной ценовой категории под ее управлением. Распространение данного программного обеспечения и его внедрение также не составляет труда, так как разработчики Android предусмотрели специальный Internet-ресурс, для распространения как бесплатного, так и платного программного обеспечения.

Отладка и тестирование, проведенные на созданной БД, а также множество добавленных и проверенных файлов, позволили исправить ошибки в программе и добиться получения корректных результатов ее работы.

Работа над проектом позволила расширить знания и улучшить навыки создания мобильных приложений, работы с базами данных.

В качестве расширения и доработки данного программного продукта можно рассматривать разработку ПО под другие мобильные операционные системы (например, iOS и WindowsPhone), а также улучшение добавлением личного кабинета с возможностью авторизации, сохранением истории добавленных файлов и автоматическим добавлением файлов из установленных папок.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1) Statcounter [Электронный ресурс] – Статья с Statcounter. URL: http://gs.statcounter.com/press/android-overtakes-windows-for-first-time
2) Android [Электронный ресурс] – Official Android WebSite. URL: https://www.android.com/intl/ru_ru/
3) Голощапов А. Л. GoogleAndroid: программирование для мобильных устройств. БХВ-Петербург, 2011. – 804 с.
4) Android Developer [Электронныйресурс] – Official Android Developer WebSite. URL: http://developer.android.com/
5) Брайн Харди, Билл Филлипс - Программирование под Android (Для профессионалов). Питер, 2014. - 592 с
6) Wikipedia [Электронный ресурс] – Статья из Wikipedia. URL: http://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений
7) Wikipedia [Электронный ресурс] – Статья из Wikipedia. URL: https://ru.wikipedia.org/wiki/Material_Design
8) Habrahabr [Электронный ресурс] – Статья с сайта Habrahabr. URL: https://habrahabr.ru/post/314028/
9) Habrahabr [Электронный ресурс] – Статья с сайта Habrahabr. URL: https://habrahabr.ru/post/299806/

Приложение А

Листинги программы сервера

Приложение А.1

Файл «main.py»

```
from flask import Flask
from flask import render_template, redirect,url_for
from flask import request
from block import *

app = Flask(__name__)

@app.route('/',methods=['POST','GET'])
def index():
  return render_template('index.html');

@app.route('/cheching', methods =['GET'])
def check():
  results = check_integrity()
  return render_template('index.html',results = results)

@app.route('/get_crypted_hash_by_id',methods=['POST','GET'])
def getCryptedHashAndPublicKeyByIdServer():
     idOfFileParam = request.form['IdOfFile']
     print(idOfFileParam)
     result     =     getCryptedHashAndPublicKeyById(IdOfFile     =
idOfFileParam);
     return json.dumps({'crypted_hash': result})

@app.route('/send_crypted_hash',methods=['POST','GET'])
```

```
def WriteBlockToBlockchain():
     CryptedHashParam = request.form['cryptedHash']
     PublicKeyParam = request.form['PublicKey']
     HashOfFileParam = request.form['HashOfFile']
     result = write_block(hashOfFile = CryptedHashParam, publicKey
= PublicKeyParam, hash = HashOfFileParam)
     return json.dumps({'response': result})

@app.route('/get_last_hash', methods=['POST','GET'])
def getLastHashInBlockchain():
     res = GetLastHash()
     return    json.dumps({'Id':    res[0],'Signature':    res[1],
'PublicKey': res[2], 'HashOfFile': res[3]})

@app.route('/load_database', methods=['POST','GET'])
def LoadDatabaseServer():
     data = LoadDatabase()
     #return json.dumps({'data': data})
     return data

if __name__ == '__main__':
  app.run(debug=True)
```

Приложение А.2

Файл «block.py»

```
import json
import os
import hashlib
import rsa
import sqlite3
import base64
blockchain_dir = os.curdir + '/blockchain/' #./blockchain/test

def get_hash(filename):
  file = open(blockchain_dir + filename,'rb').read()
  return hashlib.md5(file).hexdigest()

def getHashOfString(Message):
     Message = Message.encode('utf-8')
     #получаем хеш строки
     hash = hashlib.md5(Message)
     return hash

def LoadDatabase():
     file = open('MyDB.db3','rb')
     data = base64.b64encode(file.read())
     file.close();
     return data

def write_block(hashOfFile, publicKey, hash):
  #создаем соединение с бд
  conn = sqlite3.connect('MyDB.db3')
```

```
  # Создаем курсор - это специальный объект который делает запросы
и получает их результаты
  cursor = conn.cursor()
  #выполняем запрос к бд
  cursor.execute("insert into transactions
(Signature,PublicKey,HashOfFile) values('" + hashOfFile + "','" +
publicKey+ "','" + hash +"')")
  #сохраняем транзакцию т.к. вносили изменения в базу
  conn.commit()
  #получаем id только что внесенной записи
  cursor.execute("select id from transactions order by id desc limit
1");

  #получаем результат запроса
  result = cursor.fetchone()
  #закрываем соединение с базой
  conn.close()
  if result:
   id = result[0]
   testString = "Транзакция добавлена с ID = " + str(id)
   return testString
  else:
   return "could not add transaction"

def getCryptedHashAndPublicKeyById(IdOfFile):
 res = ['','','']
  #создаем соединение с бд
  conn = sqlite3.connect('MyDB.db3')
  # Создаем курсор - это специальный объект который делает запросы
и получает их результаты
  cursor = conn.cursor()
```

```
  #запрашиваем шифрованный хеш документа и публичный ключ по id
  cursor.execute("select HashOfFile from transactions where id = "
+ IdOfFile);

  #получаем результат запроса
  results = cursor.fetchall()

  if results:
   temp = results.pop(0)
   res = temp[0]
   return res
  else:
   return 'Хеш с данным ID транзакции не найден!'

def GetLastHash():
  #создаем соединение с бд
  conn = sqlite3.connect('MyDB.db3')
  # Создаем курсор - это специальный объект который делает запросы
и получает их результаты
 cursor = conn.cursor()

  #получаем последний шифрованный хеш из бд
  cursor.execute("select * from transactions order by id desc limit
1");

  #получаем результат запроса
  results = cursor.fetchall()

  #закрываем соединение с базой
  conn.close()

  #если результат не пустой - получить хеш строки
```

```
  if results:
   res                                                              =
[str(results[0][0]),results[0][1],results[0][2],results[0][3]]
   return res
  #иначе вернуть пустую строку
  else:
   return ['','','','']
def main():
 write_block(name='ivan',amount=7,to_whom='petya')
  #print(check_integrity())
if __name__ == '__main__':
 main()
```

Приложение В

Листинги программы клиента

Приложение В.1

Файл «MainActivity.java»

```
package com.shap.diplom;
import android.app.TabActivity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.TextView;

public class MainActivity extends TabActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // получаем TabHost
        final TabHost tabHost = getTabHost();
        // инициализация была выполнена в getTabHost

        TabHost.TabSpec tabSpec;

        tabSpec = tabHost.newTabSpec("tag1");
        tabSpec.setIndicator("Добавить\nдокумент");
        tabSpec.setContent(new Intent(this, LoadFile.class));
        tabHost.addTab(tabSpec);
```

```
tabHost.getTabWidget().getChildAt(tabHost.getCurrentTab()).setBack
groundColor(Color.parseColor("#0772A1")); // selected

        tabSpec = tabHost.newTabSpec("tag2");
        tabSpec.setIndicator("Проверить\nдокумент");
        tabSpec.setContent(new Intent(this, CheckFile.class));
        tabHost.addTab(tabSpec);

        tabSpec = tabHost.newTabSpec("tag3");
        tabSpec.setIndicator("Проверить\n          БД");
        tabSpec.setContent(new Intent(this, CheckBD.class));
        tabHost.addTab(tabSpec);

        for(int i=0;i<tabHost.getTabWidget().getChildCount();i++) {
            if (i == 0) {
                TextView          tv          =          (TextView)
tabHost.getTabWidget().getChildAt(i).findViewById(android.R.id.tit
le);

tabHost.getTabWidget().getChildAt(i).setBackgroundColor(Color.pars
eColor("#0772A1")); // unselected
                tv.setTextColor(Color.parseColor("#ffffff"));

            } else {

                TextView          tv          =          (TextView)
tabHost.getTabWidget().getChildAt(i).findViewById(android.R.id.tit
le);

tabHost.getTabWidget().getChildAt(i).setBackgroundColor(Color.pars
eColor("#225E79")); // unselected
```

```
                tv.setTextColor(Color.parseColor("#ffffff"));
            }
        }

        tabHost.setOnTabChangedListener(new
TabHost.OnTabChangeListener() {

            @Override
            public void onTabChanged(String tabId) {

                for (int i = 0; i <
tabHost.getTabWidget().getChildCount(); i++) {

tabHost.getTabWidget().getChildAt(i).setBackgroundColor(Color.pars
eColor("#225E79")); // unselected
                    TextView tv = (TextView)
tabHost.getTabWidget().getChildAt(i).findViewById(android.R.id.tit
le); //Unselected Tabs
                    tv.setTextColor(Color.parseColor("#ffffff"));
                }


tabHost.getTabWidget().getChildAt(tabHost.getCurrentTab()).setBack
groundColor(Color.parseColor("#0772A1")); // selected
                TextView tv = (TextView)
tabHost.getCurrentTabView().findViewById(android.R.id.title); //for
Selected Tab
                tv.setTextColor(Color.parseColor("#ffffff"));

            }
        });
    }
}
```

Приложение В.2

Файл «LoadFile.java»

```
package com.shap.diplom;


import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.ActivityNotFoundException;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.File
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.security.Key;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.HashMap;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class LoadFile extends AppCompatActivity implements
OnClickListener {

    private static final String TAG = "FileChooserExampleActivity";

    private static final int REQUEST_CODE = 6384; //
onActivityResult request
    private static final int PICK_FILE_REQUEST = 0;
    TextView hashView;
    EditText privateKeyView;
    EditText publicKeyView;
```

```
    TextView lastHash;
    TextView lastHashView;
    TextView cryptedHashView;
    private final String server = "http://127.0.0.1:5000";

    private Gson gson = new GsonBuilder().create();

    private Retrofit retrofit = new Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create(gson))
            .baseUrl(server)
            .build();
    private Request req = retrofit.create(Request.class);
    private          RequestSendFile          reqSendFile          =
retrofit.create(RequestSendFile.class);
    // code

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.load_file);

        // Create a simple button to start the file chooser process
        Button buttonRead = (Button) findViewById(R.id.buttonRead);
        Button           buttonGenKey           =           (Button)
findViewById(R.id.buttonGenerateKey);
        Button           buttonLastHash           =           (Button)
findViewById(R.id.getLastHash);
        Button           buttonSendFile           =           (Button)
findViewById(R.id.sendFile);
        Button buttonCrypt = (Button) findViewById(R.id.crypt);
```

```
        buttonRead.setOnClickListener(this);
        buttonGenKey.setOnClickListener(this);
        buttonLastHash.setOnClickListener(this);
        buttonSendFile.setOnClickListener(this);
        buttonCrypt.setOnClickListener(this);

        hashView = (TextView) findViewById(R.id.hashView);
        privateKeyView = (EditText) findViewById(R.id.privateKey);
        publicKeyView = (EditText) findViewById(R.id.publicKey);
        lastHash = (TextView) findViewById(R.id.lastHash);
        cryptedHashView = (TextView)
findViewById(R.id.cryptedHash);
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.buttonRead:
                showChooser();
                break;
            case R.id.buttonGenerateKey:
                GenerateKeys();
                break;
            case R.id.getLastHash:
                Call<Object> call = req.performPostCall();
                call.enqueue(new Callback<Object>() {
                    @Override
                    public void onResponse(Call<Object> call,
Response<Object> response) {
                        if (response.isSuccessful()) {
                            HashMap<String, String> map =
```

```
gson.fromJson(response.body().toString(), HashMap.class);


lastHash.setText(md5((map.get("Id")+map.get("Signature")+map.get("
PublicKey")+map.get("HashOfFile")).toString()));
                        }
                    }

                    @Override
                    public   void   onFailure(Call<Object>   call,
Throwable t) {
                        lastHash.setText("Request error");
                    }
                });

                break;
            case R.id.crypt:
                if (hashView.getText().toString().equals("")) {
                    Toast.makeText(LoadFile.this,
                            "Поле 'Хэш документа' не заполнено:",
Toast.LENGTH_LONG).show();
                    break;
                }
                if (privateKeyView.getText().toString().equals("")
                        ||
privateKeyView.getText().toString().equals("Приватный ключ")) {
                    Toast.makeText(LoadFile.this,
                            "Поле     для     приватного     ключа
некорректно:", Toast.LENGTH_LONG).show();
                    break;
                }
```

```
                if (publicKeyView.getText().toString().equals("")
                        ||
publicKeyView.getText().toString().equals("Публичный ключ")){
                    Toast.makeText(LoadFile.this,
                            "Поле для публичного ключа
некорректно:", Toast.LENGTH_LONG).show();
                    break;
                }
                String text = hashView.getText().toString();
                String PRIVATE_KEY =
privateKeyView.getText().toString();
                //encrypt
                String result = encrypt(text,PRIVATE_KEY );
                 break;

            case R.id.sendFile:
                if (hashView.getText().toString().equals("")) {
                    Toast.makeText(LoadFile.this,
                            "Поле 'Хэш документа' не заполнено!",
Toast.LENGTH_LONG).show();
                    break;
                }
                if (privateKeyView.getText().toString().equals("")
                        ||
privateKeyView.getText().toString().equals("Приватный ключ")) {
                    Toast.makeText(LoadFile.this,
                            "Поле для приватного ключа
некорректно!", Toast.LENGTH_LONG).show();
                    break;
                }
                if (publicKeyView.getText().toString().equals("")
```

```
                        ||
publicKeyView.getText().toString().equals("Публичный ключ")){
                    Toast.makeText(LoadFile.this,
                            "Поле для публичного ключа
некорректно!", Toast.LENGTH_LONG).show();
                    break;
                }
                if (lastHash.getText().toString().equals("")){
                    Toast.makeText(LoadFile.this,
                            "Поле последнего хэша пусто!",
Toast.LENGTH_LONG).show();
                    break;
                }
                if
(cryptedHashView.getText().toString().equals("")){
                    Toast.makeText(LoadFile.this,
                            "Поле подписи пусто!",
Toast.LENGTH_LONG).show();
                    break;
                }
                Call<Object> callSendFile =
reqSendFile.performPostCall(hashView.getText().toString());
                callSendFile.enqueue(new Callback<Object>() {
                    @Override
                    public void onResponse(Call<Object> call,
Response<Object> response) {
                        if (response.isSuccessful()) {
                            HashMap<String, String> map =

gson.fromJson(response.body().toString(), HashMap.class);
                            String id= map.get("id");
```

```
                            AlertDialog.Builder builder = new
AlertDialog.Builder(LoadFile.this);
                            builder.setTitle("Документ добавлен в
блокчейн! ")
                                    .setMessage("ID вашего
документа:"+id)
                                    .setCancelable(false)
                                    .setNegativeButton("OK",
                                            new
DialogInterface.OnClickListener() {
                                                public void
onClick(DialogInterface dialog, int id) {

dialog.cancel();
                                                }
                                            });
                            AlertDialog alert = builder.create();
                            alert.show();
                            }
                    }
                    @Override
                    public void onFailure(Call<Object> call,
Throwable t) {
                    }
                });

                break;
        }
    }
    public void showChooser() {
        //Выбор файла
        Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
```

```
        intent.setType("*/*");
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        try {
            startActivityForResult(intent, REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            Log.e("Err", "Choser err");
        }
    }

    @SuppressLint("LongLogTag")
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        switch (requestCode) {
            case REQUEST_CODE:
                // If the file selection was successful
                if (resultCode == RESULT_OK) {
                    if (data != null) {
                        // Get the URI of the selected file
                        final Uri uri = data.getData();
                        Log.i(TAG, "Uri = " + uri.toString());
                        try {
                            // Get the file path from the URI
                            final        String        path        =
FileUtils.getPath(this, uri);
                            Toast.makeText(LoadFile.this,
                                    "Файл        выбран:        ",
Toast.LENGTH_LONG).show();
                        } catch (Exception e) {
                            Log.e("FileSelectorTestActivity", "File
select error", e);
                        }
```

```
                        File file = new File(uri.getPath());
                        try {
                            InputStream inputStream =
getContentResolver().openInputStream(uri);
                            BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream));
                            StringBuilder stringBuilder = new
StringBuilder();
                            String line;
                            try {
                                while ((line =
bufferedReader.readLine()) != null) {
                                    stringBuilder.append(line);
                                }
                                hashView.setText("[Хэш
документа]:\n" + md5(stringBuilder.toString()));
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        } catch (FileNotFoundException e) {
                            e.printStackTrace();
                        }
                    }
                }
                break;
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
    String md5(String in) {
        MessageDigest digest;
        try {
            digest = MessageDigest.getInstance("MD5");
```

```
            digest.reset();
            digest.update(in.getBytes());
            byte[] a = digest.digest();
            int len = a.length;
            StringBuilder sb = new StringBuilder(len << 1);
            for (int i = 0; i < len; i++) {
                sb.append(Character.forDigit((a[i] & 0xf0) >> 4,
16));
                sb.append(Character.forDigit(a[i] & 0x0f, 16));
            }
            return sb.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return null;
    }

    public void GenerateKeys(){
        Key publicKey = null;
        Key privateKey = null;
        try {
            KeyPairGenerator                kpg                =
KeyPairGenerator.getInstance("RSA");
            kpg.initialize(1024);
            KeyPair kp = kpg.genKeyPair();
            publicKey = kp.getPublic();
            privateKey = kp.getPrivate();
        } catch (Exception e) {
            Log.e("Crypto", "RSA key pair error");
        }
        privateKeyView.setText(keyToString(privateKey));
        publicKeyView.setText(keyToString(publicKey));
```

```
    }

    public static String encrypt(String text, String pri_key) {
        try {
            byte[] data = text.getBytes("utf-8");
            PrivateKey privateKey =
getPrivateKey(Base64.decode(pri_key.getBytes("utf-8"),
Base64.DEFAULT));
            Cipher cipher =
Cipher.getInstance("RSA/ECB/OAEPWithSHA-256AndMGF1Padding");
            cipher.init(Cipher.ENCRYPT_MODE, privateKey);
            return
Base64.encodeToString(cipher.doFinal(data),Base64.DEFAULT);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
    public static String decrypt(String text, String pub_key) {
        try {
            byte[] data = Base64.decode(text,Base64.DEFAULT);
            PublicKey publicKey =
getPublicKey(Base64.decode(pub_key.getBytes("utf-
8"),Base64.DEFAULT));

            Cipher cipher =
Cipher.getInstance("RSA/ECB/OAEPWithSHA-256AndMGF1Padding");
            cipher.init(Cipher.DECRYPT_MODE, publicKey);
            return new String(cipher.doFinal(data),"utf-8");
        } catch (Exception e) {
            return null;
        }
```

```
    }

    public static PublicKey getPublicKey(byte[] keyBytes) throws
NoSuchAlgorithmException, InvalidKeySpecException {
        X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(keyBytes);
        KeyFactory keyFactory =
KeyFactory.getInstance("RSA/ECB/OAEPWithSHA-256AndMGF1Padding");
        return keyFactory.generatePublic(keySpec);
    }

    /**
     * @param keyBytes
     * @return
     * @throws NoSuchAlgorithmException
     * @throws InvalidKeySpecException
     */
    public static PrivateKey getPrivateKey(byte[] keyBytes) throws
NoSuchAlgorithmException,
            InvalidKeySpecException {
        PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(keyBytes);
        KeyFactory keyFactory =
KeyFactory.getInstance("RSA/ECB/OAEPWithSHA-256AndMGF1Padding");
        return keyFactory.generatePrivate(keySpec);
    }


    public static String keyToString(Key secretKey) {
        return Base64.encodeToString(secretKey.getEncoded(),
Base64.DEFAULT);
    }
```

```
    public static Key stringToKey(String stringKey) {
        byte[]    encodedKey    =    Base64.decode(stringKey.trim(),
Base64.DEFAULT);
        return new SecretKeySpec(encodedKey, 0, encodedKey.length,
"AES");
    }
}
```

Приложение В.3

Файл «CheckFile.java»

```
package com.shap.diplom;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.ActivityNotFoundException;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class CheckFile extends AppCompatActivity implements
OnClickListener {
    TextView hashView;
    EditText idView;
    TextView hashFromBlockchainView;
    private static final String TAG = "FileChooserExampleActivity";
    private static final int REQUEST_CODE = 6384; //
onActivityResult request

    private final String server = "http://127.0.0.1:5000";


    private Gson gson = new GsonBuilder().create();


    private Retrofit retrofit = new Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create(gson))
```

```
            .baseUrl(server)
            .build();
    private RequestGetHashById requestGetHashById =
retrofit.create(RequestGetHashById.class);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.check_file);

        Button buttonRead = (Button)
findViewById(R.id.buttonRead1);
        Button buttonGenKey = (Button)
findViewById(R.id.buttonGetHashById);
        Button buttonLastHash = (Button)
findViewById(R.id.buttonCheckHashes);

        buttonRead.setOnClickListener(this);
        buttonGenKey.setOnClickListener(this);
        buttonLastHash.setOnClickListener(this);

        hashView = (TextView) findViewById(R.id.hashView1);
        idView = (EditText) findViewById(R.id.id);
        hashFromBlockchainView = (TextView)
findViewById(R.id.hashFromBlockchain);
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.buttonRead1:
```

```
                showChooser();
                break;
            case R.id.buttonGetHashById:
                Call<Object> call =
requestGetHashById.performPostCall(idView.getText().toString());
                call.enqueue(new Callback<Object>() {
                    @Override
                    public void onResponse(Call<Object> call,
Response<Object> response) {
                        if (response.isSuccessful()) {
                            HashMap<String, String> map =

gson.fromJson(response.body().toString(), HashMap.class);
                            String hash = map.get("crypted_hash");
                            hashFromBlockchainView.setText(hash);
                        }
                    }

                    @Override
                    public void onFailure(Call<Object> call,
Throwable t) {
                        hashFromBlockchainView.setText("Request
error");
                    }
                });

hashFromBlockchainView.setText("bb566698d08fd446cf8fe719310e43c2")
;
                break;
            case R.id.buttonCheckHashes:
```

```
                if
(hashView.getText().toString().equals(hashFromBlockchainView.getTe
xt().toString())) {
                    AlertDialog.Builder     builder     =     new
AlertDialog.Builder(CheckFile.this);
                    builder.setTitle("Важное сообщение!")
                            .setMessage("Документ не был изменен!")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new
DialogInterface.OnClickListener() {
                                        public                void
onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
                    AlertDialog alert = builder.create();
                    alert.show();
                } else {
                    AlertDialog.Builder     builder     =     new
AlertDialog.Builder(CheckFile.this);
                    builder.setTitle("Важное сообщение!")
                            .setMessage("Документ не был изменен")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new
DialogInterface.OnClickListener() {
                                        public                void
onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
```

```
                    AlertDialog alert = builder.create();
                    alert.show();

                    break;
                }
        }
    }
    public void showChooser() {
        //Выбор файла
        Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
        intent.setType("*/*");
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        try {
            startActivityForResult(intent, REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            Log.e("Err", "Choser err");
        }
    }

    @SuppressLint("LongLogTag")
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        switch (requestCode) {
            case REQUEST_CODE:
                // If the file selection was successful
                if (resultCode == RESULT_OK) {
                    if (data != null) {
                        // Get the URI of the selected file
                        final Uri uri = data.getData();
                        Log.i(TAG, "Uri = " + uri.toString());
```

```
                        try {
                            // Get the file path from the URI
                            final String path =
FileUtils.getPath(this, uri);
                            Toast.makeText(CheckFile.this,
                                    "Файл выбран: ",
Toast.LENGTH_LONG).show();
                        } catch (Exception e) {
                            Log.e("FileSelectorTestActivity", "File
select error", e);
                        }
                        File file = new File(uri.getPath());
                        try {
                            InputStream inputStream =
getContentResolver().openInputStream(uri);
                            BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream));
                            StringBuilder stringBuilder = new
StringBuilder();
                            String line;
                            try {
                                while ((line =
bufferedReader.readLine()) != null) {
                                    stringBuilder.append(line);
                                }
                                LoadFile lf = new LoadFile();

                                hashView.setText("[Хэш
документа]:\n" + lf.md5(stringBuilder.toString()));
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
```

```
                        } catch (FileNotFoundException e) {
                            e.printStackTrace();
                        }
                    }
                }
                break;
            }
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
```

Приложение В.4

Файл «CheckBD.java»

```
package com.shap.diplom;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.view.View.OnClickListener;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
```

```
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import okhttp3.ResponseBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class CheckBD extends AppCompatActivity implements
OnClickListener {
    private static final String TAG = "1234";
    private final String server = "http://127.0.0.1:5000";
    String line;

    private Gson gson = new GsonBuilder().create();


    private Retrofit retrofit = new Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create(gson))
            .baseUrl(server)
            .build();
    private Request req = retrofit.create(Request.class);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.check_bd);

        Button buttonRead = (Button) findViewById(R.id.buttonRead2);
        Button buttonLoadBD = (Button) findViewById(R.id.buttonLoadBD);
        Button buttonCheckBD = (Button) findViewById(R.id.checkBD);

        buttonRead.setOnClickListener(this);
        buttonLoadBD.setOnClickListener(this);
        buttonCheckBD.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.buttonRead2:
                LoadFile lf = new LoadFile();
                lf.showChooser();
                try {
                    File file = new File("/Download//Db3.sql");
                    //создаем объект FileReader для объекта File
                    FileReader fr = new FileReader(file);
                    //создаем BufferedReader с существующего FileReader для построчного считывания
                    BufferedReader reader = new BufferedReader(fr);
                    // считаем сначала первую строку
                    line = reader.readLine();
                    while (line != null) {

                        // считываем остальные строки в цикле
                        line = reader.readLine();
```

```
                    }
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
            case R.id.buttonLoadBD:
                Call<Object> call = req.performPostCall();
                call.enqueue(new Callback<Object>() {
                    @Override
                    public void onResponse(Call<Object> call,
Response<Object> response) {
                        if (response.isSuccessful()) {
                            boolean writtenToDisk =
writeResponseBodyToDisk((ResponseBody) response.body());
                        }
                    }

                    @Override
                    public void onFailure(Call<Object> call,
Throwable t) {

                    }
                });
                break;
            case R.id.checkBD:

                if (checkBD()) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(CheckBD.this);
                    builder.setTitle("Важное сообщение!")
```

```
                            .setMessage("Блокчейн не поврежден!")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new
DialogInterface.OnClickListener() {
                                        public                  void
onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
                    AlertDialog alert = builder.create();
                    alert.show();
                } else {
                    AlertDialog.Builder     builder     =     new
AlertDialog.Builder(CheckBD.this);
                    builder.setTitle("Важное сообщение!")
                            .setMessage("Блокчейн не поврежден!")
                            .setCancelable(false)
                            .setNegativeButton("OK",
                                    new
DialogInterface.OnClickListener() {
                                        public                  void
onClick(DialogInterface dialog, int id) {
                                            dialog.cancel();
                                        }
                                    });
                    AlertDialog alert = builder.create();
                    alert.show();



                    break;
                }
```

```
        }

    }

    private boolean checkBD() {
        while (line!=null){
        }
            return true;
        };

    private boolean writeResponseBodyToDisk(ResponseBody body) {
        try {
            File futureStudioIconFile = new
File(getExternalFilesDir(null) + File.separator + "Future Studio
Icon.png");

            InputStream inputStream = null;
            OutputStream outputStream = null;

            try {
                byte[] fileReader = new byte[4096];

                long fileSize = body.contentLength();
                long fileSizeDownloaded = 0;

                inputStream = body.byteStream();
                outputStream = new
FileOutputStream(futureStudioIconFile);

                while (true) {
                    int read = inputStream.read(fileReader);
```

```
                    if (read == -1) {
                        break;
                    }

                    outputStream.write(fileReader, 0, read);

                    fileSizeDownloaded += read;

                    Log.d(TAG, "file download: " +
fileSizeDownloaded + " of " + fileSize);
                }

                outputStream.flush();

                return true;
            } catch (IOException e) {
                return false;
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }

                if (outputStream != null) {
                    outputStream.close();
                }
            }
        } catch (IOException e) {
            return false;
        }
    }
}
```

Приложение В.5

Файл «FileUtils.java»

```
package com.shap.diplom;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;

import java.net.URISyntaxException;

public class FileUtils {
    public static String getPath(Context context, Uri uri) throws
URISyntaxException {
        if ("content".equalsIgnoreCase(uri.getScheme())) {
            String[] projection = { "_data" };
            Cursor cursor = null;

            try {
                cursor   =   context.getContentResolver().query(uri,
projection, null, null, null);
                int                 column_index                =
cursor.getColumnIndexOrThrow("_data");
                if (cursor.moveToFirst()) {
                    return cursor.getString(column_index);
                }
            } catch (Exception e) {
                // Eat it
            }
        }
        else if ("file".equalsIgnoreCase(uri.getScheme())) {
            return uri.getPath();
```

```
        }
        return null;
    }
}
```

Приложение В.6

Файл «Request.java»

```
package com.shap.diplom;
import java.util.HashMap;

import retrofit2.Call;
import retrofit2.http.FieldMap;
import retrofit2.http.FormUrlEncoded;
import retrofit2.http.POST;

public interface Request  {
    //@FormUrlEncoded
    @POST("/get_last_hash")
    Call   <Object>   performPostCall(/*@FieldMap   HashMap<String,
String> postDataParams*/);
}
```

Приложение В.7

Файл «RequestGetHashById.java»

```
package com.shap.diplom;

import java.util.HashMap;

import retrofit2.Call;
import retrofit2.http.Field;
```

```
import retrofit2.http.FieldMap;
import retrofit2.http.POST;

public interface RequestGetHashById {
    @POST("/get_crypted_hash_by_id")
    Call<Object> performPostCall(@Field("id") String id);
}
```

Приложение В.8

Файл «RequestSaveBD.java»

```
package com.shap.diplom;

import retrofit2.Call;
import retrofit2.http.Field;
import retrofit2.http.POST;
import retrofit2.http.Streaming;

public interface RequestSaveBD {
    @Streaming
    @POST("/load_database")
    Call<Object> performPostCall();
}
```

Приложение В.9

Файл «RequestSendFile.java»

```
package com.shap.diplom;

import java.util.HashMap;

import retrofit2.Call;
```

```
import retrofit2.http.Field;
import retrofit2.http.FieldMap;
import retrofit2.http.POST;

public interface RequestSendFile {
    @POST("/send_crypted_hash")
    Call<Object>    performPostCall(@Field("hashOfFile")    String
hashOfFile);
}
```

Приложение В.10

Файл «activity_main.xml»

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TabHost
        android:id="@android:id/tabhost"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TabWidget
                android:id="@android:id/tabs"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
                style="@style/MyTabs"
                android:background="#3E94D1"
                >
            </TabWidget>

            <FrameLayout
                android:id="@android:id/tabcontent"
                android:layout_width="match_parent"
                android:layout_height="match_parent">
            </FrameLayout>

        </LinearLayout>
    </TabHost>
</android.support.constraint.ConstraintLayout>
```

Приложение В.11

Файл «load_file.xml»

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".LoadFile">

    <Button
        android:id="@+id/buttonRead"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_gravity="center_horizontal"
        android:text="@string/read_file" />

    <TextView
        android:id="@+id/hashView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textColor="#000000"
        android:hint="@string/hash"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@+id/buttonRead"
        />

    <EditText
        android:id="@+id/privateKey"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"
        android:maxLines="1"
        android:hint="@string/private_key"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@id/hashView" />

    <EditText
        android:id="@+id/publicKey"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:maxLines="1"
```

```
        android:hint="@string/public_key"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@id/privateKey" />

    <Button
        android:id="@+id/buttonGenerateKey"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/generate_key"
        app:layout_constraintTop_toBottomOf="@id/publicKey" />

    <TextView
        android:id="@+id/lastHash"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:hint="@string/last_hash_view"
        android:textSize="18sp"

app:layout_constraintTop_toBottomOf="@id/buttonGenerateKey" />

    <Button
        android:id="@+id/getLastHash"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/last_hash"
        app:layout_constraintTop_toBottomOf="@id/lastHash" />

    <Button
```

```
        android:id="@+id/crypt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/crypt"
        app:layout_constraintLeft_toRightOf="@id/getLastHash"
        app:layout_constraintTop_toBottomOf="@id/lastHash" />

    <TextView
        android:id="@+id/cryptedHash"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:maxLines="1"
        android:hint="@string/crypted_hash"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@id/getLastHash" />

    <Button
        android:id="@+id/sendFile"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/send_file"
        app:layout_constraintTop_toBottomOf="@id/cryptedHash" />
</android.support.constraint.ConstraintLayout>
```

Приложение В.12

Файл «check_file.xml»

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.constraint.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".CheckFile">

    <Button
        android:id="@+id/buttonRead1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/read_file"
        />
    <TextView
        android:id="@+id/hashView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"

        android:hint="@string/hash"
        android:textSize="18sp"
        app:layout_constraintTop_toBottomOf="@+id/buttonRead1"
        />



    <EditText
        android:id="@+id/id"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"
        app:layout_constraintTop_toBottomOf="@id/hashView1"
        android:textSize="18sp"
        android:hint="@string/id"
        />

    <Button
        android:id="@+id/buttonGetHashById"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/id"
        android:text="@string/get_hash_by_id"
        android:layout_gravity="center_horizontal" />
    <TextView
        android:id="@+id/hashFromBlockchain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"

app:layout_constraintTop_toBottomOf="@id/buttonGetHashById"
        android:textSize="18sp"
        android:hint="Хэш последней строки из блокчейна"/>
    <Button
        android:id="@+id/buttonCheckHashes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"

app:layout_constraintTop_toBottomOf="@id/hashFromBlockchain"
        android:text="@string/check_hash"
```

```
        android:layout_gravity="center_horizontal"
        />
</android.support.constraint.ConstraintLayout>
```

Приложение В.13

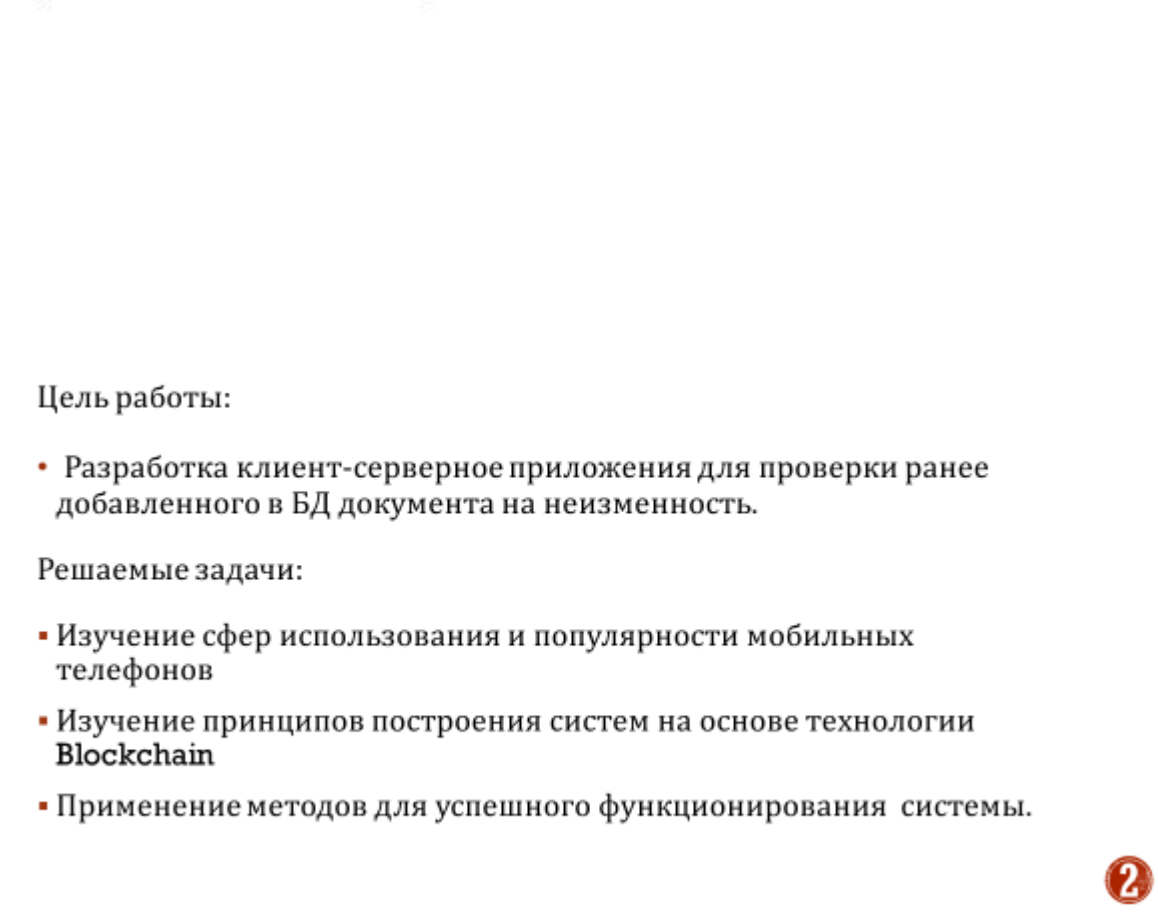
Файл «check_bd.xml»

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".CheckBD">
    <Button
        android:id="@+id/buttonLoadBD"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="#00BFFF"
        android:text="@string/load_bd"
        />

    <Button
        android:id="@+id/buttonRead2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/buttonLoadBD"
        android:layout_marginTop="20dp"
        android:text="@string/read_file"
```

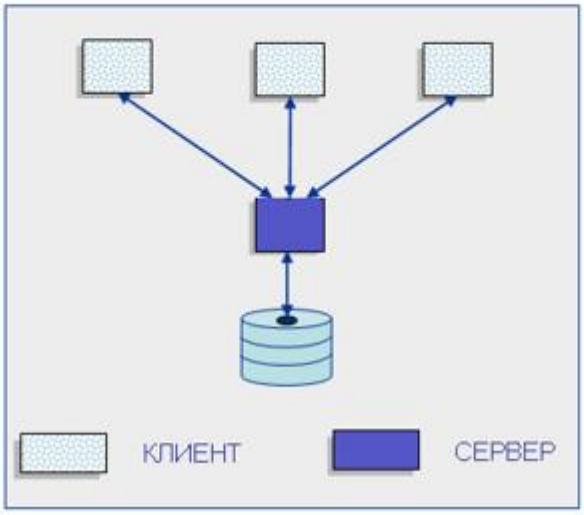
```
        android:layout_gravity="center_horizontal" />

    <Button
        android:id="@+id/checkBD"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_gravity="center_horizontal"
        android:text="@string/check_bd"
        app:layout_constraintLeft_toRightOf="@id/buttonRead2"
        app:layout_constraintTop_toBottomOf="@id/buttonLoadBD" />

</android.support.constraint.ConstraintLayout>
```

Приложение С

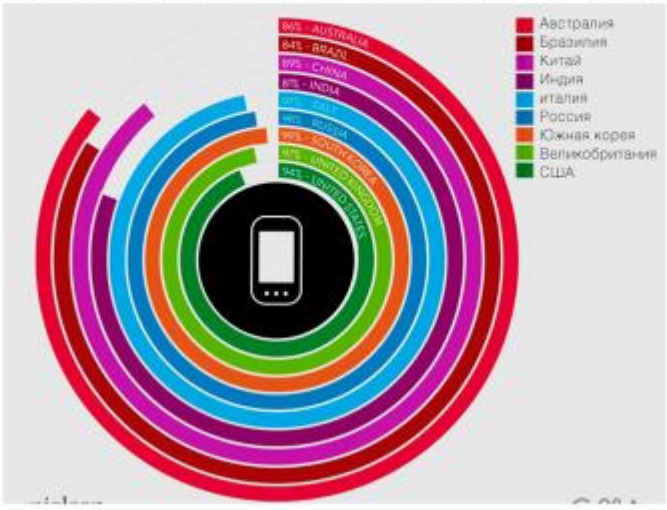
Презентация



АРХИТЕКТУРА СИСТЕМЫ

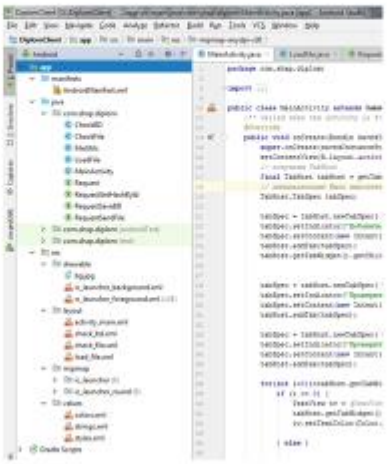


УРОВЕНЬ ПРОНИКНОВЕНИЯ МОБИЛЬНЫХ ТЕЛЕФОНОВ



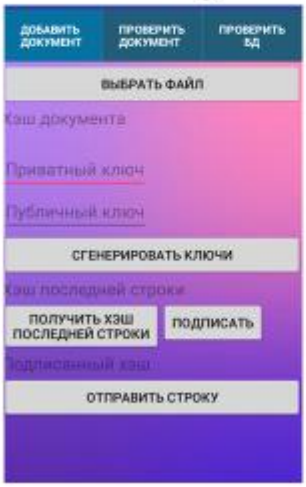
5

СТРУКТУРА ПРОЕКТА

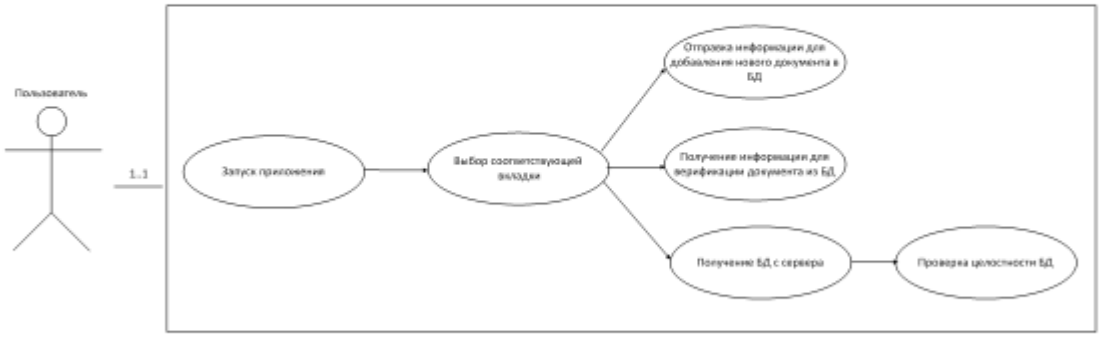


6

ЭКРАН ПРИЛОЖЕНИЯ(ДОБАВЛЕНИЕ ДОКУМЕНТА)



ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С КЛИЕНТСКОЙ ПРОГРАММОЙ



АЛГОРИТМ ХЕШИРОВАНИЯ MD5

АЛГОРИТМ RSA

Сценарий: Боб посылает Алисе сообщение m

11

12

Спасибо за внимание!

Выступал: Шаповалов Игорь

Студент группы 14ВВ1