

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

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. Web-приложение

Студент

Кор. 13.06.18 Королев Б.Д.
(подпись, дата)

Руководитель

Трокоз 13.06.18 Трокоз Д.А.
(подпись, дата)

Нормоконтролёр

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

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

Заведующий кафедрой

Пашенко Д.В.
(подпись) Пашенко Д.В.

Работа защищена с отметкой отм. (протокол заседания ГАК от 19.06.18 № 8)

Секретарь ГАК

Гурин Е.И.
(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

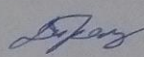
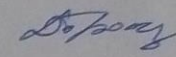
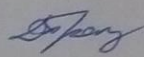
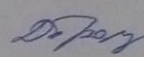
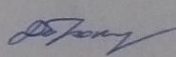
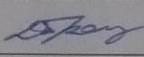
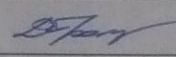
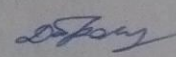
Д.В. Пашенко

2018г

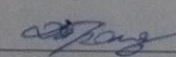
ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

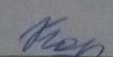
1. Студент Королев Богдан Дмитриевич гр. 14ВВ1 факультета вычислительной техники.
2. Тема работы «Единый портал учёта документов с использованием технологии Blockchain. Web-приложение».
Тема утверждена приказом ПГУ № 398а/о от "03" мая 2018 г.
3. Руководитель работы доцент кафедры ВТ ПГУ кандидат технических наук Трокоз Д.А.
4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение: система для организации верификации документов, предназначенная для проверки документов на неизменность и подтверждение авторства документов с помощью цифровой подписи.
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 структуры блокчейна, особенностей его работы и свойств для применения в различных сферах человеческой деятельности, обзор основных информационных систем, работающих на технологии блокчейн, составление общей схемы работы системы для организации портала учета документов, а также разработка веб-приложения.

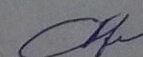
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области и составление общей схемы работы системы	10%	12.03.18	
2.	Изучение технологии блокчейн и организации клиент - серверной архитектуры с помощью протокола HTTP	15%	27.03.18	
3.	Изучение современных серверных информационных систем, работающих на технологии блокчейн	15%	11.04.18	
4.	Разработка серверного приложения	15%	26.04.18	
5.	Разработка клиентского приложения	20%	7.05.18	
6.	Составление выводов	5%	14.05.18	
7.	Отладка и тестирование	10%	21.05.18	
8.	Оформление выпускной работы	10%	28.05.18	

Дата выдачи задания " 26 " февраля 2018 г.

Руководитель бакалаврской работы  13.06.18 Трокоз Д.А.
(подпись, дата)

Задание к исполнению принял студент  13.06.18 Королев Б.Д.
(подпись, дата)

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

РЕФЕРАТ

Бакалаврская работа содержит 89 листов, 34 рисунка, 2 таблицы, 4 источника, 3 приложения.

БД, БЛОКЧЕЙН, ВЕБ-ПРИЛОЖЕНИЕ, ВЕРИФИКАЦИЯ, ИС, ПОДПИСЬ, ХЕШ, ЦЕЛОСТНОСТЬ.

Цель работы – разработка клиент-серверного приложения, осуществляющего добавление информации в базу данных (блокчейн), проверку добавленной информации на неизменность, а также целостность структуры базы данных.

Исходя из поставленной цели в процессе разработки необходимо решить следующие задачи:

1. Определить состав и принципы функционирования технологии блокчейн;
2. Проанализировать характеристики блокчейна, а также методы поддержания целостности структуры базы данных при использовании данной технологии;
3. Изучить алгоритмы добавления транзакций в БД и проверки её на неизменяемость.

Объектом разработки является комплекс приложений (клиент и сервер), которые должны, взаимодействуя между собой, осуществлять добавление хешей файлов в базу данных, верификацию и проверку подлинности хешей с помощью ассиметричного шифрования и цифровой подписи, а также проверку базы данных на внесенные изменения.

					ПГУ 1.090301.12.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Единый портал учёта документов с использованием технологии Blockchain. Web-приложение Пояснительная записка	Лит.	Лист	Листов
Разраб.		Королев Б.Д.		13.06.18				
Провер.		Трокоз Д.А.		13.06.18			4	89
Н. Контр.		Кучин А.В.		13.06.18		ФВТ, каф. ВТ		

В результате проведенной работы создан проект, удовлетворяющий поставленной задаче, а именно, создан сервер, принимающий POST и GET запросы от клиентов через протокол HTTP, взаимодействующий с базой данных с помощью SQL-запросов, а также клиентское приложение, реализуемое страницами, отправляющимися с сервера. Страницы имеют в своем составе все необходимые ресурсы для шифрования, подписи и проверки документов на стороне клиента средствами браузера.

Данный проект рекомендуется использовать при необходимости контроля целостности и неизменности документов, созданных клиентами. Приложение может быть внедрено в областях, где требуется невозможность незаметного внесения искажений в базу данных с целью извлечения из этого выгоды.

					ПГУ 1.090301.12.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Содержание

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ	7
ВВЕДЕНИЕ	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	9
1.1 Развитие веб-технологий в современном мире	9
1.1.1 Сферы применения веб-технологий.....	12
1.2 Сетевая модель передачи данных HTTP	13
1.2.1 Структура HTTP-запросов и ответов	14
1.3. Технология блокчейн	18
1.3.1 Характеристики блокчейна	20
1.3.2 Применение технологии блокчейн в различных сферах человеческой деятельности	21
1.4 Основные принципы работы блокчейна Ethereum	23
1.4.1 Правила консенсуса – протокол GHOST	24
1.4.2 Компоненты платформы Ethereum	25
2 РАЗРАБОТКА ПРИЛОЖЕНИЯ	29
2.1 Постановка задачи	29
2.2 Структура данных приложений	29
2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса.....	31
2.4 Описание разработки программы	34
2.5 Отладка и тестирование	37
3 ОПИСАНИЕ ПРОГРАММЫ	38
3.1 Описание клиентского приложения	38
3.3 Описание серверного приложения	41
3.4 Руководство пользователя	42
ЗАКЛЮЧЕНИЕ	55
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	56
ПРИЛОЖЕНИЕ А Листинги программы сервера	57
Приложение А.1 Файл сервера main.py.....	57
Приложение А.2_Файл сервера block.py.....	59
ПРИЛОЖЕНИЕ В Листинги программы клиента	63
Приложение В.1 Файл клиента base.html.....	63
Приложение В.2 Файл клиента index.html	79
ПРИЛОЖЕНИЕ С Презентация	83

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД – База данных

TCP – Transmission Control Protocol

IP – Internet Protocol

HTTP – HyperText Transfer Protocol

ИС – Информационная система

WWW – World Wide Web

URL – Uniform Resource Locator

ID – Identifier

PHP – PHP: Hypertext Preprocessor

HTML – HyperText Markup Language

URI – Uniform Resource Identifier

GHOST - Greedy Heaviest Observed Subtree

JSON – JavaScript Object Notation

ИС – Информационная система

					ПГУ 1.090301.12.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

ВВЕДЕНИЕ

Неоспоримым фактом является то, что безопасность занимает высокую позицию в нашей жизни, что является причиной того, что развитие технологий, обеспечивающих безопасность, является неотъемлемой частью научно-технического прогресса.

Необходимость создания протоколов для верификации и подтверждения авторства документов заставляет придумывать и использовать все более и более новые и совершенные алгоритмы подписи и верификации файлов. Однако неотъемлемой частью алгоритмов, позволяющих предотвратить незаметное изменение данных, а также проверить их подлинность являются специальные структуры данных и технологии, которые в последнее время стали всё активнее проникать в различные сферы человеческой деятельности. Одной из таких технологий является Blockchain.

В основе данной технологии лежит хеширование блоков информации и связывание путем полученных хешей этих блоков в цепочку, в результате чего каждый последующий блок становится зависим от предыдущего и, как следствие, изменение информации в блоках становится явно заметным.

Однако основным недостатком существующих систем на основе этой технологии является техническая сложность реализации и использования.

В рамках данной выпускной квалификационной работы необходимо разработать систему регистрации документов на основе технологии блокчейн, которая бы обладала простым и понятным пользовательским интерфейсом в сочетании с функциональностью и надежностью и позволяла бы устранить проблему проверки важных документов на неизменность с течением времени.

					ПГУ 1.090301.12.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Развитие веб-технологий в современном мире

Рост интернета и количества изданий, посвященных Сети, предвещает широкое ее распространение даже в далеких от техники областях. Internet превращается из большой игрушки для интеллектуалов в полноценный источник разнообразной полезной информации для любой категории пользователей.

Специалисты прогнозируют, что через 10 лет около 50 процентов семей (безусловно, пока лишь в развитых странах) будут использовать интернет ежедневно. Использование телевидения и радиоприемников не будет заменено компьютерами, но они будут иметь мощный процессор, память и фактически сами станут в некотором отношении компьютерами.

Настанет время, когда любой человек, владеющий компьютером, получит возможность "скачать" с его помощью статьи, иллюстрации, видео- или аудиоинформацию по любой интересующей его теме. Он сможет получить эту информацию когда и где он этого пожелает. Через некоторое время системы искусственного интеллекта, осуществляющие параллельный машинный перевод, а также идентификацию и распознавание голоса сломают национальные и языковые барьеры и сделают возможным свободный транснациональный обмен информацией.

Всемирная паутина (www), или просто Web, существует уже несколько десятков лет. Изменения, происходившие в технологиях представления информации и взаимодействия с пользователем, оказали на него существенное влияние. Можно сказать, что Веб построен на протоколах, контенте (информации и данных), серверных и клиентских скриптах. Можно выделить

					ПГУ 1.090301.12.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

несколько этапов развития интернета, отражающих путь его эволюции. Этапы не приходят на смену друг друга, накладываясь друг на друга, принося новые возможности и частично заменяя старые технологии.

Передовым идеям позволяет легко проявляться свобода и демократичность сети, отсутствие социальных сдерживающих факторов. Интернет позволяет заинтересованным и способным людям объединяться и создавать более крупные проекты.

Важнейшим компонентом web-технологий являются базы данных, которые хранят содержимое сайта. Происходит динамическое формирование страниц с помощью языка программирования на сервере в зависимости от приходящих запросов клиента. После чего на клиентский компьютер отправляется готовый HTML-документ.

На сегодняшний день помимо PHP используется множество других языков программирования (Ruby, Python). В конце первого десятилетия XX века событийно-ориентированное программирование начало активно использоваться в Интернете. Доставка в браузер клиента страниц не в готовом виде, а посредством управляемых событиями приложений для обмена информацией повысило удобство использования сайтов. Для этих целей служит язык JavaScript а также его библиотеки. Преимущества JavaScript заключается в том, что не приходится отправлять большой объем HTML-разметки и данных в ответ на каждое движение пользователя. Вместо этого происходит обновление небольших фрагментов содержимого. У Пользователя нет необходимости в перезагрузке страницы для изменения её содержимого. Изменения контента происходят динамически, в зависимости от произошедших событий. При этом HTML и JavaScript обычно работают вместе, обеспечивая наилучшую функциональность сайта.

					ПГУ 1.090301.12.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Для работы Web необходимо наличие TCP/IP сети, что заставляет отказываться от других сетевых технологий не только в глобальных сетях, но и в сетях локальных. С другой стороны, наличие TCP/IP сети дает возможность легко интегрировать информационную систему с другими информационными сервисами, давая очень хороший шанс для внедрения в качестве базы информационной системы Web. При этом решения автоматически становятся масштабируемыми, т.к. для технологий TCP/IP не имеет значения, где расположены ресурсы и интерфейс пользователя: локально, в рамках корпоративной сети или предприятия, или распределены по глобальной сети. Очевидным становится факт того, что основная область применения web-технологий - распределенные информационные системы. При этом необходимо унифицировать форматы подготовки, обмена и отображения информации. Рассмотрим процесс миграции ИС из традиционной технологической схемы локальной системы в web-технологию. Традиционная схема представляет из себя:

1. интерфейс пользователя
2. ядро системы
3. информационный массив
4. интерфейс администратора
5. утилиты администратора

С точки зрения web-технологии интерфейс пользователя - это браузер, который взаимодействует с ядром через http-сервер. Таким образом происходит первый этап декомпозиции традиционной информационной системы в Web. Второй шаг - это возможность использования браузера в качестве интерфейса администратора. Здесь возникают вопросы разграничения доступа и актуализации информации в базах данных системы. Следующий шаг - распределение нагрузки по нескольким серверам, а также использование кэширования на серверах-посредниках. Пока декомпозиции подвергалась

связка "конечный пользователь-ядро". Можно провести декомпозицию и на стороне сервера. Первым таким шагом является применение CGI при доступе к ресурсам. Сервер становится посредником между браузером и сервером ресурса. Более эффективно это решается за счет API, когда сам http-сервер и имеет модуль доступа к backend процессу. Другим важным моментом является внедрение результатов обращения к внешнему ресурсу в готовые шаблоны страниц.

1.1.1 Сферы применения веб-технологий

Основными сферами применения веб-технологий являются:

- Применение интернет и Web-технологии в учебной деятельности. Интернет является перспективным направлением развития современного образования, давая возможность создания комфортных условий для обеспечения организации учебной деятельности за счет создания информационной образовательной среды.
- Использование web – технологий в сфере маркетинга. Проникновение Интернета во все стороны жизни общества является глобальным трендом современной эпохи. Внедрение интернет-технологий в субъекты экономики становится повышает их конкурентноспособность на потребительском рынке.
- Интернет технологии в рекламе. Ни один маркетинговый план, направленный на увеличение сбыта продукции или раскрутку компании, не может обойтись без эффективной рекламы. Интернет технологии в рекламе – это один из наиболее действенных способов добиться задуманного результата.
- Использование интернета в торговле. Грамотно построенный способ ведения бизнеса гарантирует быстрое развитие компании и увеличение доходов. Торговля онлайн практически уравнила шансы крупных оптовых поставщиков и мелких игроков в любой сфере рынка.

1.2 Сетевая модель передачи данных HTTP

HTTP («протокол передачи гипертекста») — протокол прикладного уровня передачи данных в виде гипертекстовых документов в формате «HTML», в настоящий момент может использоваться также и для передачи произвольных данных. Основой протокола является клиент-серверная технология, предполагая существование:

- Потребителей (клиентов), которые иницируют соединение и посылают запрос для получения некоего ресурса;
- Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают клиенту сообщение с результатом.

• Основным объектом манипуляции в HTTP является ресурс, на который указывает URI в запросе клиента. Ресурсами обычно являются хранящиеся на сервере файлы, HTML страницы, логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (в частности, для этого используется HTTP-заголовок). Возможность указания способа кодирования сообщения позволяет клиенту и серверу обмениваться двоичными данными, несмотря на то, что данный протокол является текстовым.

• HTTP — протокол прикладного уровня; аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов в протоколе используются глобальные URI. Особенностью протокола HTTP является то, что он не сохраняет своего состояния, что влечет отсутствие сохранения промежуточного состояния между парами «запрос-ответ». HTTP-компоненты имеют возможность самостоятельного осуществления сохранения информации о

состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера).Посылающий запросы браузер имеет возможность отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

1.2.1 Структура HTTP-запросов и ответов

Коммуникационная транзакция между клиентом и сервером через протокол HTTP состоит из того, что мы называем HTTP-сообщениями. HTTP-сообщение состоит из HTTP-запроса, посылаемого клиентом серверу, и HTTP-ответа, возвращаемого сервером клиенту на основе полученного запроса.

1 GET / HTTP/1.1	
2 Host: www.example.com	
Line 1	
GET	HTTP method
/	Path of resource
HTTP/1.1	HTTP version

Рисунок 1 – элементарный HTTP-запрос

Элементарный HTTP запрос состоит из названия метода (GET), пути к ресурсу (/), и версии HTTP протокола (HTTP/1.1)

В протоколе HTTP поддерживается несколько методов. В самой первой версии HTTP/1.0 было три метода: GET, POST и HEAD. В HTTP/1.1 появилось несколько новых методов: OPTIONS, CONNECT, TRACE, PUT и DELETE. В RFC 5789, появившегося в 2010 году, добавился метод PATCH

<i>HTTP/1.1 Methods</i>	
GET	The GET method is used to simply request a resource from a web server. Parameters within this method are passed over the URL.
POST	This method is used to send data to a web server via the body of the HTTP request.
HEAD	This method is like the GET method but outputs only the HTTP response headers in the response returned by the server and not the body.
OPTIONS	This method is used to retrieve a list of methods the web server accepts via the 'Allow' HTTP response header.
PUT	This method is used to replace an existing resource or create a new resource on a web server.
DELETE	This method is used to delete a resource on a web server.
TRACE	The TRACE method is used for testing purposes and reflects the entire message received by the web server back to the client. This allows the client to see exactly what the web server received.
CONNECT	The CONNECT method is hardly ever used and is for use with a proxy that can dynamically switch to being a tunnel.
PATCH	This method is like the PUT method. It differs because it allows for partial resource modification as to where the PUT method only allows for complete resource replacement.

Рисунок 2 – перечень методов, поддерживаемых протоколом HTTP/1.1

GET необходим для простого запроса ресурсов с веб-сервера. Отличительной особенностью метода является то, что параметры передаются через URL.

POST используется для отсылки данных на веб-сервер через тело HTTP-запроса.

HEAD схож с методом GET, однако выводит только заголовки HTTP-ответа, который возвращает сервер.

OPTIONS используется для получения списка методов, принимаемых веб-сервером, которые хранятся в заголовке 'Allow' в HTTP-ответе.

PUT предназначен для замены существующего или создания нового ресурса на веб-сервере.

TRACE используется при тестировании и отправляет полное сообщение, полученное веб-сервером, обратно клиенту, что позволяет удостовериться в конкретном содержимом, полученном веб-сервером. CONNECT используется редко и совместно с прокси-сервером, для динамического переключения в режим туннеля.

PATCH схож с методом PUT. Отличие заключается в этом, что PATCH поддерживает частичную модификацию, в отличие от метода PUT, поддерживающего лишь полную замену ресурса.

URL является подмножеством унифицированных идентификаторов ресурсов (URI). Типичная структура URL выглядит так:

[Protocol]://[host]/[resource path]?[parameter1]&[parameter2]

По сути, URL обозначает местонахождение сетевого ресурса а также метод передачи данных целевому ресурсу.

<i>Other Common HTTP Request Headers</i>	
User-Agent	Contains browser specific information.
Accept	Specifies what type of content the client can accept.
Accept-Encoding	Specifies the type of encoding the client can accept.
Accept-Language	Specifies the language of the client.
Content-Length	This is the length of the request body in octets. This actual value doesn't matter much but some HTTP methods require this header such as PUT but it can simply be set to 0 if its required.
Referer	Contains the URL where the request originated.
Cookie	Submits cookies to the server for session management.
Connection	Used to tell the server whether to close the connection or leave the connection open for subsequent requests.
Authorization	A header that contains platform authentication related information.

Рисунок 3 – наиболее распространенные заголовки HTTP-запроса

User-Agent содержит информацию о браузере.

Accept определяет тип содержимого, принимаемого клиентом.

Accept-Encoding определяет тип кодировки, принимаемой клиентом.

Content-Length определяет длину тела запроса в октетах. Это значение не имеет особой важности, но для некоторых HTTP-методов (например, PUT) требуются этот заголовок. Если необходимо, в значение этого заголовка можно установить 0.

Referer содержит URL-источник запроса.

Cookie используется для отправки cookies на сервер для управления сессией.

Connection используется для сообщения серверу о закрытии соединения или оставления его открытым для последующих запросов.

Authorization содержит информацию, которая имеет отношение к аутентификации на конкретной платформе.

После обработки сервером клиенту отправляется ответ, похожий по структуре на HTTP-запрос.

```

1 HTTP/1.1 200 OK
2 Date: Mon, 24 Jul 2017 04:11:15 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 X-Powered-By: PHP/5.5.9-1ubuntu4.21
5 Vary: Accept-Encoding
6 Content-Length: 2548
7 Content-Type: text/html

```

Рисунок 4 – элементарный HTTP-ответ

Код статуса является важной частью HTTP-сообщения, поскольку свидетельствует о том, как приложение обработало запрос.

HTTP Response Status Codes	
1xx	Informational
2xx	OK, the request was successful
3xx	The client is to be redirected.
4xx	The request cannot be processed for some reason such as its unauthorized or the request has an error.
5xx	The server encountered an error.

Рисунок 5 – коды статусов HTTP-ответа

Код ответа «100 Continue» используется редко. Наиболее распространенным является код «200 OK», сигнализирующий о том, что запрос корректен, ресурс существует, сервер обработал запрос и вернул ресурс в теле ответа. Код 201 означает, что ресурс, который запрашивается в запросе, был успешно создан. Данный код обычно является результатом запроса с использованием метода PUT. Коды статуса 3xx обычно возвращаются приложениями со ссылкой на ресурс для перенаправления клиента. Ссылка находится либо в теле ответа, либо в заголовке «location». Коды статуса 4xx отсылаются в случае, если запрашиваемого ресурса не существует на сервере или пользователь не авторизован для получения содержимого или при возникновении ошибки в запросе, отправленном серверу. Коды статуса 5xx возвращаются в случае возникновения ошибки на сервере или невозможности обработать запрос.

Common HTTP Response Headers	
Date	Returns the date in which the request was retrieved.
Server	Information associated with the web server (example: IIS/Apache)
X-Powered-By	Information associated with the scripting technology and version in use (example: PHP or Asp.net)
Content-Length	Like its presence in a request header, it's the length of the response body in octets.
Content-Type	Indicates the content type of the resource returned.
Set-Cookie	Cookies to be used by the client in proceeding requests to manage the session.
Expires	A time that the server will no longer consider the response returned valid.
Cache-Control	Directs the client as to whether to cache the responses returned.

Рисунок 6 – HTTP-заголовки ответов

Date содержит дату получения запроса.

Server содержит информацию о веб-сервере (например, IIS/Apache).

X-Powered-By содержит информацию, которая касается технологии, используемой в скриптах, и текущую версию (например, PHP или Asp.net).

Content-Length схож с аналогичным заголовком в запросе и содержит длину тела ответа в октетах.

Expires содержит, время, по истечении которого сервер не будет рассматривать ответ как корректный.

Set-Cookie содержит cookie, используемые клиентом при формировании запроса для управления сессией.

Cache-Control указывает клиенту, нужно ли кэшировать возвращаемые запросы.

1.3. Технология блокчейн

Прямой перевод этого словосочетания — цепочка блоков — лучше всего объясняет его суть. Блокчейн — это способ хранения данных или, как его еще называют, цифровой реестр транзакций. В него можно добавлять данные практически обо всем: о выданных кредитах, правах на собственность, результатах голосования, фактах нарушения правил дорожного движения, бракосочетаниях и так далее.

Записи в блокчейне — это данные о транзакциях, произведенных внутри сети. Они объединяются друг с другом в специальную структуру — блок. Информация, содержащаяся в них, открыта, она не шифруется и ее можно

быстро перепроверить. Каждый блок помимо транзакций также содержит свой порядковый номер и так называемую контрольную сумму предыдущего блока — результат обработки содержащихся в предыдущем блоке данных особой криптографической функцией (хеш-функцией). Они используются для проверки целостности каких-либо данных, так как получающееся в результате вычислений решение, или, как его еще называют, контрольная сумма (хеш-сумма), будет меняться до неузнаваемости при внесении даже малейших изменений в исходные данные. Это свойство позволяет легко отслеживать любые несанкционированные попытки изменений.

Все записи в блокчейне непрерывно синхронизируются между всеми участниками сети, сохраняясь на каждом устройстве. Это позволяет отслеживать попытки изменения и избавляться от них, обеспечивая безопасность хранящихся данных. Представьте себе обычный пазл. Если вы извлечете оттуда одну деталь и замените ее на похожую, но из другого набора, то картинка не сложится. Так и блокчейн препятствует изменению информации в каждом из блоков.

Как только цепочка будет обновлена и в нее вставят блок, он уже больше не может быть изменен, возможно только добавление новых записей, причем данные будут обновляться на всех компьютерах в сети одновременно. Таким образом, в блокчейне практически невозможно подделать какие-либо данные, так как они сразу же будут обнаружены другими устройствами, входящими в цепочку. И даже больше — измененная информация не сохранится, цепочка вернет старые данные, которые "записаны" на большинстве компьютеров.

Конечно, можно попытаться захватить более 51% от числа всех включенных в сеть компьютеров и попытаться внести изменения сразу на всех, но такой сценарий будет крайне затратным в реализации и, скорее всего, совсем не выгодным.

1.3.1 Характеристики блокчейна

Любая система блокчейн имеет свой протокол, но фактически этот протокол включает в себя стандартный набор характеристик:

- Прозрачность и публичность.
- Равноправность участников.
- Неизменяемость.
- Децентрализованность и распределенность.
- Предоставление анонимности.

Прозрачность и публичность сейчас перекликаются с принципом анонимности. Вопрос создания публичных и частных блокчейнов сейчас очень важен для представителей биткоин-индустрии, так как он является важным в определении дальнейшей политики создания новых приложений на основе распределенного реестра.

Принцип неизменяемости является, пожалуй, самым устойчивым и независимым ни от каких-либо разработок и решений. Какие бы данные вы не вносили в свою учетную запись, они не сотрутся уже никогда. На это никто не сможет повлиять, так как блоки постоянно добавляются и представляют собой единую последовательную цепь. Пролистать все обратно уже невозможно. На этом и построен принцип децентрализованной системы.

Децентрализованность и распределенность несут в себе то преимущество и те особенности функционирования, которые вы не найдете ни в одном программном обеспечении и ни в одной даже самой налаженной деятельности. Система сама себя контролирует и дает себе команды о движении транзакций.

В 2015 году на Виталик Бутерин разделил блокчейны на три типа: частная блокочная цепь (fully private blockchains); блокочная цепь, принадлежащая

консорциуму (consortium blockchains); и публичная блокочная цепь (public blockchains).

- Частная блокочная цепь - все транзакции отслеживаются и контролируются центральным органом.
- Блокочная цепь консорциума - согласование транзакций происходит среди отобранных узлов-представителей консорциума.

Публичная блокочная цепь - отбор и перемещение транзакций никем не контролируется и происходит в свободном порядке.

1.3.2 Применение технологии блокчейн в различных сферах человеческой деятельности

Существует множество сфер применений технологии в финансовой системе, кроме Bitcoin, поэтому интерес проснулся и у банков, и у других финансовых учреждений.

По данным Всемирного Экономического Форума, в ближайшие годы мы будем свидетелями значительных изменений, и blockchain конечном итоге станет “сердцем” будущей мировой финансовой системы.

Вместо того чтобы полагаться на услуги хранения данных в централизованном облаке, таком как Dropbox, Amazon или Google Drive, технология *blockchain* предлагает возможность хранения данных и файлов в P2P-сети (*peer-to-peer*), то есть, хранятся несколькими членами сети.

В существующих централизованных системах вся информация или файлы хранятся в одном месте и поэтому может быть подвержена нападению со стороны *хакеров* или потере информации из-за технической проблемы или природных катастроф таких, как пожары или наводнения.

В распределительном хранении, файлы и данные хранятся в форме аналогичной системе P2P, таких, как Torrent или Emule, то есть, одна из копий хранится в каждом узле сети.

На сегодняшний день это обычное дело, что каждая часть конечного продукта поставляется из различных мест и компаний. Это создает целую цепочку поставок, которая собирает, производит и продает конечный продукт.

Иногда цепочки настолько велики, что трудно полностью отследить все движение.

Блокчейн предлагает решения против подделок и упрощения отслеживания продукции для получения сертификата.

Использовать *blockchain* в качестве услуги нотариуса — это легко и дешево.

Позволяет создавать неизменные записи и отследить документ и цепь событий. Блокчейн позволяет, например, проверить подлинность любого документа, который был в нем зарегистрирован, что исключает необходимость в централизованной власти, или в третьем лице для подтверждения.

Поскольку *blockchain* невозможно «переписать» и запись может быть подтверждена независимыми третьими лицами, эти услуги являются юридически обязывающими.

Среди его основных преимуществ:

- Является более безопасным, поскольку, в отличие от человека, который может быть подвержен коррупции, не может быть изменен.
- Гораздо дешевле, учитывая высокие тарифы нотариусов.

- Благодаря возможностям записывать в ней все что угодно анонимно, гарантирует конфиденциальность зарегистрированного документа и тех, кто обращается за услугами к нотариусу.

1.4 Основные принципы работы блокчейна

Ethereum

Блокчейн Эфириум является, по сути, системой состояния транзакций. В информатике такое понятие, как «система состояний» или «машина состояний» – это система, которая обрабатывает вводимую информацию и на основании последней преобразуется в новое состояние. В машине состояний Эфириума все процессы начинаются с «первоначального состояния». Такое состояние представляет собой аналог нулевого состояния, в котором находится машина до того момента, как в ее сети начнут происходить какие-либо действия, связанные с транзакциями. Когда такие действия начнут происходить, первоначальное состояние заменяется на конечное, при этом в любой момент времени конечное состояние отображает текущее состояние Эфириума.

Состояние Эфириума имеет миллионы транзакций. Эти транзакции сгруппированы в «блоки». Блок содержит ряд транзакций, при этом каждый последующий блок соединен с предыдущим, благодаря чему обеспечивается своеобразная цепочка блоков. На рисунке 1 показана цепочка блоков.

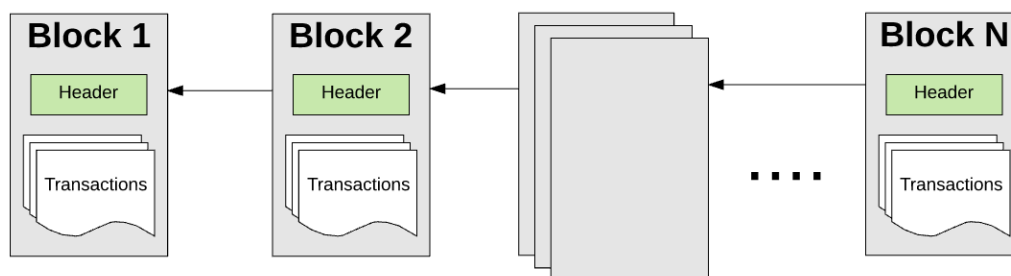


Рисунок 7 – цепочка блоков

Транзакция должна быть корректной, для того чтобы вызвать ее переход из одного состояния в другое. Транзакция считается корректной только тогда, когда она прошла процесс проверки – так называемый «майнинг». Майнинг — это когда группа узлов (компьютеров) расходует свои вычислительные ресурсы для создания блока корректных транзакций.

Любой узел в сети, объявляющий себя майнером, может попытаться создать и проверить блок транзакций. Распространенным опытом является попытки множества майнеров одновременного создания и проверки блока транзакций. Каждый майнер предоставляет свое математическое «доказательство» при отправке блока в блокчейн, и это доказательство выступает в роли своеобразной гарантии: в случае если доказательство существует, транзакции в блоке считаются корректными. Майнер должен предоставить свое математическое доказательство быстрее, чем это сделает любой другой конкурент, для того чтобы его блок был добавлен в основной блокчейн. Процесс проверки каждого блока, который заключается в предоставлении майнером своего математического доказательства, называется «доказательством работы».

1.4.1 Правила консенсуса – протокол GHOST

Всякий раз, когда генерируются несколько путей, возникает «разветвление». Зачастую, разветвления очень нежелательны, поскольку они нарушают целостность системы, а пользователям приходится выбирать одну из возможных цепочек. Чтобы определить, какой из возможных путей является корректным, и предотвратить образования множества цепей, в Эфириуме применяется метод, называемый «протокол GHOST».

Протокол GHOST объявляет, что мы должны выбрать только тот путь, на котором было выполнено наибольшее число вычислений. Для определения

					ПГУ 1.090301.12.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

такого пути можно использовать номер того блока, который был определен последним («листовой блок»). Благодаря такому подходу можно определить общее число блоков, находящихся в текущем пути (без учета блока первоначального состояния). Чем выше находится блок, тем длиннее путь и тем больше обоснований должны предоставить майнеры. Исходя из таких соображений, принимается единственно верная версия для текущего состояния.

1.4.2 Компоненты платформы Ethereum

Компонентами Ethereum являются:

- учетные записи
- состояние
- горючее о вознаграждение
- транзакции
- блоки
- выполнение транзакций
- майнинг
- обоснование

Также, при хешировании в блокчейне Ethereum используется алгоритм КЕССАК-256.

Глобальное общее состояние платформы Эфириум состоит из множества небольших объектов – учетных записей, которые взаимодействуют между собой за счет парадигмы обмена сообщениями. У каждой учетной записи есть определенное состояние и 20-байтовый адрес. Адресом в Эфириум является 160-битный идентификатор, используемый для идентификации любой из учетных записей.

Глобальное состояние Эфириума – это сопоставление адресам учетной записи состояний счета. Это сопоставление хранится в структуре данных – префиксного дерева Меркла.

Любой узел, для которого требуется проверка какой-либо части данных, использует так называемое «доказательством Меркла». Последнее состоит из:

- Фрагмента данных, который должен быть проверен
- Корневого хеша дерева
- Так называемой «ветви» – всех хешей, от проверяемого фрагмента данных до корня.

Каждый пользователь, считывающий такое доказательство, может проверить, является ли хеширование для определенной ветви соответствующим на всем участке дерева, а также занимает ли данный фрагмент соответствующее положение в этом дереве.

Горючее – это единица измерения, которая используется для определения размера оплаты по конкретному вычислению. Цена на горючее – это количество «эфира», которое вы способны потратить на каждую единицу горючего.

Для проведения любой транзакции отправитель должен установить лимит горючего, а также цену на горючее. Все деньги, которые были потрачены на приобретение отправителем горючего, отправляются на адрес бенефициария, который в большинстве случаев является адресом майнера.

Транзакция является криптографически подписанной частью инструкции, которая сначала задается внешней учетной записью, а затем упорядочивается и передается в блокчейн.

Всего существует два типа транзакций: отправка сообщений и создание

					ПГУ 1.090301.12.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

контракта (другими словами, такие транзакции создают новые контракты в сети Эфириума).

Все транзакции так или иначе сгруппированы в «блоки». Блокчейн содержит несколько таких блоков, соединенных между собой. Такие блоки состоят из:

- заголовка блока
- информации о серии транзакций, включенных в данный блок
- серии других заголовков блоков для текущих оммеров

В первую очередь, значение авансовой стоимости горючего вычитается со счета отправителя, а показатель nonce отправителя увеличивается на 1. После чего мы можем подсчитать оставшееся количество горючего по следующей формуле: от общего количества горючего, необходимого для проведения транзакции, отнимаем заданное количество горючего. После этого начинается выполнение транзакции. В течение проведения текущей транзакции в Эфириуме происходит отслеживание «подсостояния». Подсостояние необходимо для записи информации, которая была собрана во время проведения текущей транзакции. Такая информация будет необходима сразу по завершении выполнения транзакции и содержит:

- **Self-destruct set:** набор учетных записей, которые будут удалены по завершении проведения транзакции
- **Log series:** заархивированные и проиндексированные контрольные точки, необходимые для выполнения кода виртуальной машины.
- **Refund balance:** сумму, которая должна быть возвращена отправителю по завершении транзакции.

После этого происходит выполнение различных вычислений, необходимых для проведения транзакции.

					ПГУ 1.090301.12.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

После того как все шаги, необходимые для проведения транзакции, были завершены (при условии, что все указанные выше требования также были выполнены), состояние транзакции завершается, при этом происходит подсчет количества неиспользованного горючего, которое должно быть возвращено отправителю.

					ПГУ 1.090301.12.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

2 РАЗРАБОТКА ПРИЛОЖЕНИЯ

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.

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.

Устройства ввода-клавиатура и мышь.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов. Необходимо определиться с выбором доступного и интуитивно понятного меню для возможности эффективной работы с программой.

Операционная система - MS Windows

Среда программирования –Notepad++

Язык программирования – Python, JavaScript, HTML

2.2 Структура данных приложений

Для реализации клиентской части программы были выбраны язык разметки страниц HTML и язык для построения логики JavaScript, которые поддерживаются текстовым редактором Notepad++.

					ПГУ 1.090301.12.001 ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

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

На рисунках 8 и 9 показаны UML-диаграммы вариантов использования клиентской программы.

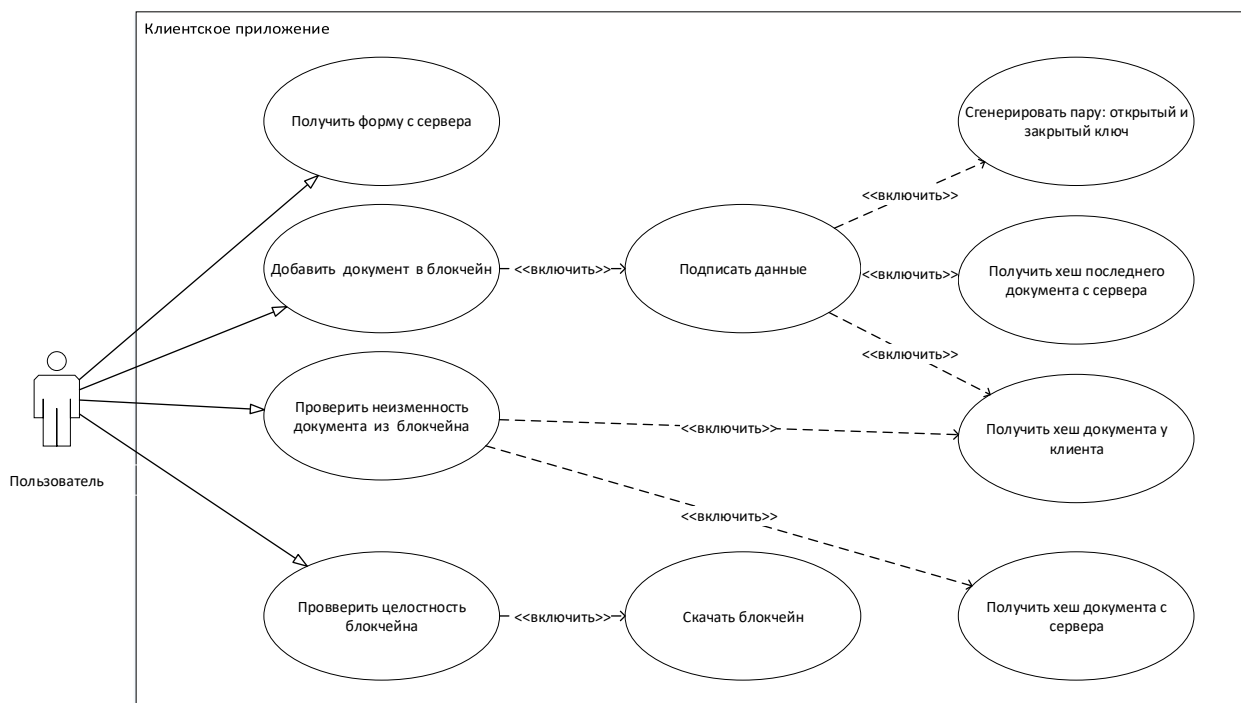


Рисунок 8 - UML-диаграмма вариантов использования клиентской программы

Программа-клиент представляет собой форму, отображаемую в браузере, и имеет возможность выбора файла для добавления в блокчейн, поля и кнопки для генерации открытого и закрытого ключей, получения хеша последней записи, а также верификации документа и проверки целостности БД.

На рисунке 13 показана UML-диаграмма вариантов использования серверной программы.

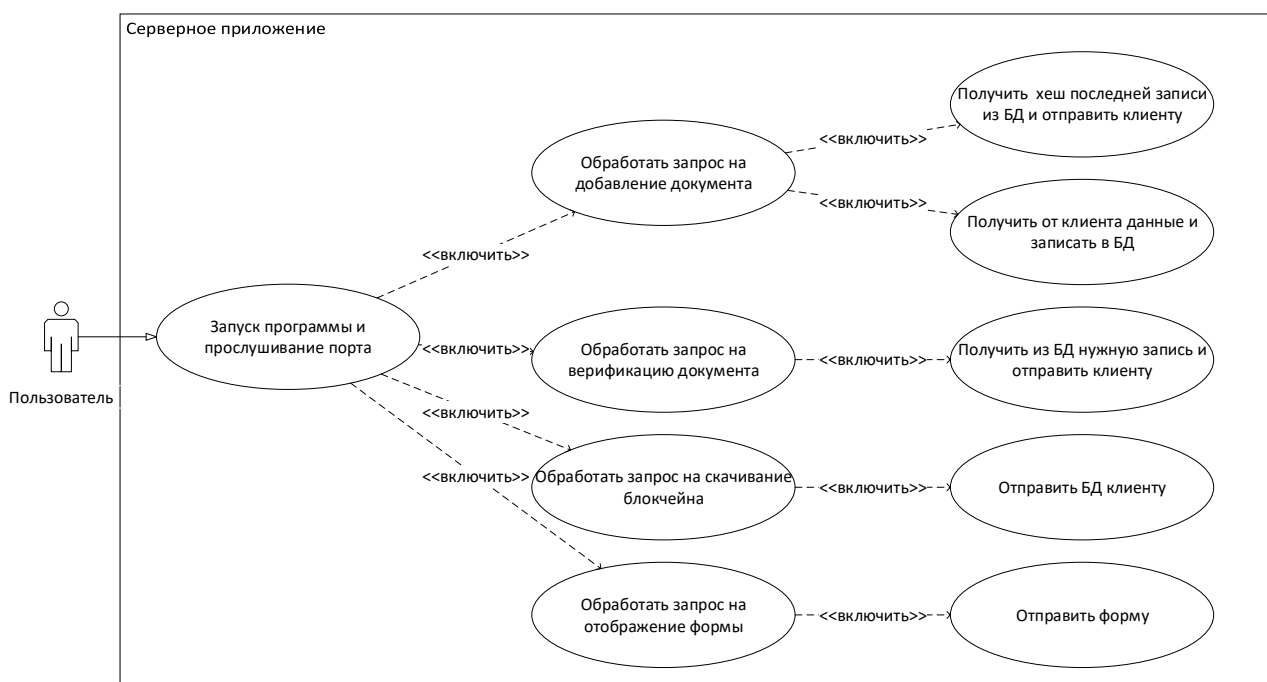


Рисунок 1 - UML-диаграмма вариантов использования серверной программы

Программа-сервер прослушивает 5000-й порт, принимает запросы от клиентов на добавление, и верификацию документов, после чего формирует запросы к БД и отправляет результат обратно пользователям. Так же имеется возможность скачивания БД в виде файла на сторону клиента для проверки её целостности.

2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса

Программа-клиент представлена в виде HTML – страницы, в которой содержится JavaScript код, интерпретируемый браузером для выполнения логики приложения. Пользователь, при вводе интернет адреса, получает эту страницу, на которой расположена форма, включающая в себе элементы пользовательского интерфейса: кнопки, текстовые поля. Основные функции программы помещены в отдельные группы, для достижения интуитивно понятного интерфейса.

На рисунке 10 изображена UML-диаграмма состояний программы-клиента.



Рисунок 2 - UML-диаграмма состояний программы-клиента в виде HTML страницы

На рисунке 11 изображена UML-диаграмма состояний программы-сервера.

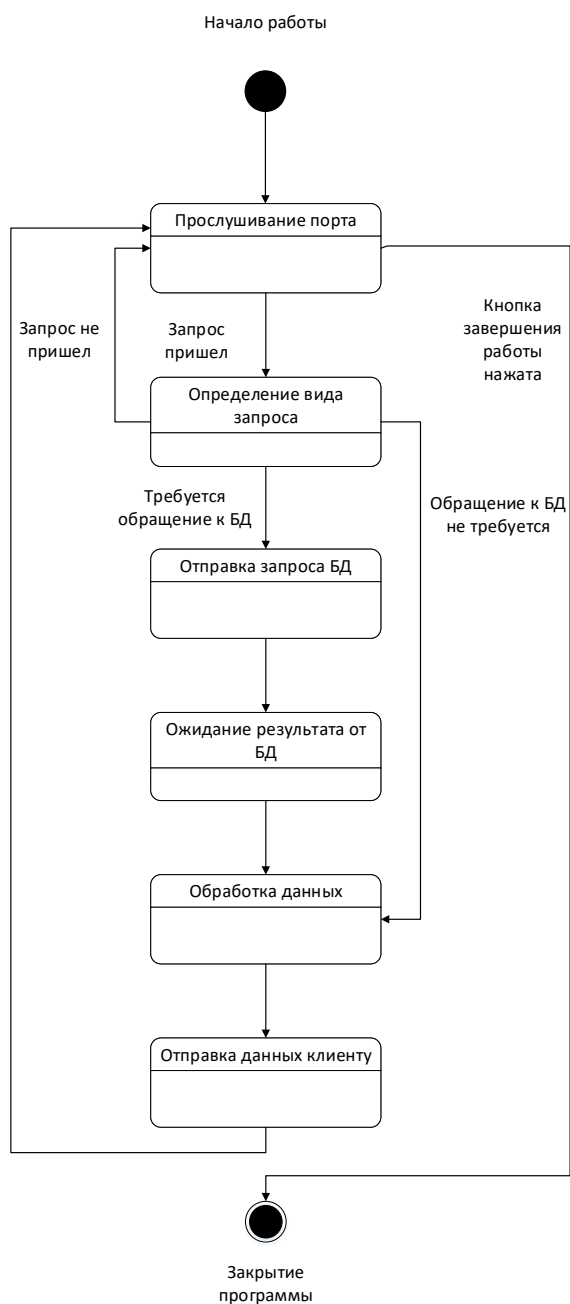


Рисунок 11 - UML-диаграмма состояний программы-сервера

2.4 Описание разработки программы

Для написания программы-клиента были использованы языки программирования HTML, JavaScript. Для разработки программы-сервера использовался язык программирования Python.

В качестве среды программирования был выбрана программа для редактирования кода Notepad++.

Интерфейс программы клиента представлен в виде формы 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 (загрузке HTML страницы) на экране появляется форма с кнопками выбора файлов и отправки запросов, текстовыми полями для вставки ключей, полями вывода для данных, полученных от сервера. После загрузки формы пользователь может произвести различные действия: добавить файл в БД, проверить файл в БД на неизменность и проверить саму БД на целостность.

При необходимости добавления файла в блокчейн необходимо перейти в область “Добавление транзакции в блокчейн”, нажать на кнопку “Выбрать файл”. После нажатия на кнопку выбора файла будет открыто окно выбора файла. При выборе файла текстовое поле “Хеш файла” заполнится автоматически хешем выбранного файла. Далее следует сгенерировать открытый и закрытый ключи, нажав на кнопку “Сгенерировать открытый и закрытый ключ” или, если они уже имеются, вставить их в соответствующие поля формы. Следом стоит нажать на кнопку “Получить хеш последней транзакции”, после чего на форме появится строка с хешем 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

					ПГУ 1.090301.12.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

сообщение. После нажатия на кнопку “Добавить транзакцию в блокчейн” 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 хеша 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 в область “Проверка подлинности”, нажать на кнопку выбора файла, после чего хеш выбранного файла так же автоматически рассчитается и заполнит текстовое поле “Хеш файла”. После этого необходимо ввести в текстовое поле “ID файла”, ID транзакции, которое было возвращено сервером при добавлении файла в блокчейн и нажать на кнопку “Получить хеш файла по ID”. В результате сервер пришлет хеш 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 хеша файла необходимо нажать на кнопку “Сравнить хеши”. Если файл не был изменен с момента загрузки его в блокчейн, то в поле “Результат проверки” будет написано значение “документ не был изменен”, иначе будет показано “документ был изменен”.

Для проверки целостности БД следует перейти в область “Проверка целостности блокчейна”, нажать на кнопку “Загрузить блокчейн” и выбрать место сохранения БД. После загрузки БД следует нажать на кнопку выбора файла и выбрать загруженный блокчейн, а затем нажать на кнопку “Проверить целостность”. Если в БД не были попытки изменить уже внесенные данные, то поле “Результат проверки” отобразит значение “Блокчейн не поврежден”, в противном случае отобразится “Блокчейн поврежден”, что свидетельствует о попытке изменения данных в БД.

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 12 показана UML-диаграмма классов программы-клиента.

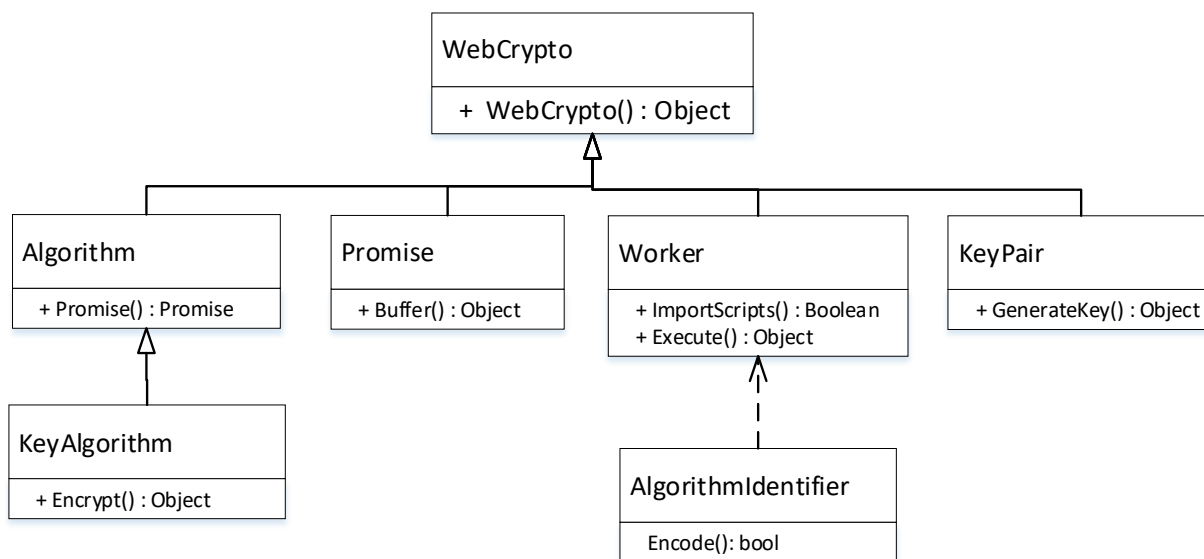


Рисунок 12 - UML-диаграмма классов программы-клиента

На рисунке 13 показана UML-диаграмма классов программы-сервера.

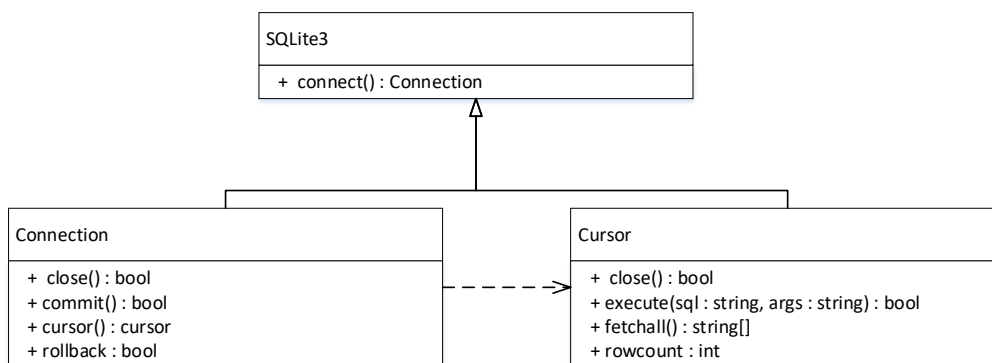


Рисунок 13 - UML-диаграмма классов программы-сервера

Для написания кода внутри HTML страницы программы клиента были использованы библиотеки для хеширования Crypto.js, MD5.js, а также GostCrypto.js, которая реализует функционал асимметричного шифрования с открытым ключом. Класс WebCrypto является родителем для классов Algorithm, Promise, Worker, KeyPair в библиотеке и содержит необходимые конструкторы для создания криптографических объектов. Класс KeyPair

реализует генерацию пары открытого и закрытого ключей, которые используются в алгоритме подписи для функций класса KeyAlgorithm.

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 была выбран редактор кода Notepad++, содержащий в себе различные средства для разработки и отладки модулей и программ. 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.

					ПГУ 1.090301.12.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Описание клиентского приложения

Для запуска клиентского приложения пользователь входит в браузер и набирает в адресной строке IP и порт сервера, после чего на сервер посылается GET запрос по протоколу HTTP и в ответ клиенту отсылается HTML страница с формой, с которой пользователь в последствие и работает.

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 файла	Нажата кнопка “Выбор файла”	Выбор файла и хеширование его содержимого
Генерация открытого и закрытого ключей	Нажатие на кнопку “Сгенерировать пару ключей”	Генерация ключей
Получение хеша последней транзакции	Нажата кнопка “Получить хеш последней транзакции”	Отправка запроса к серверу, ожидание ответа, и показ результата в текстовом поле “Хеш последней транзакции в блокчейне”
Подпись конкатенированной строки, состоящей из последнего хеша из БД и хеша файла	Нажата кнопка “Подписать”	Шифрование закрытым ключом информации

Продолжение таблицы 1

Отправка данных на сервер для добавления транзакции в БД	Нажата кнопка “Добавить транзакцию в блокчейн”	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 хеша файла по ID	Нажата кнопка “Получить хеш файла по ID”	Отправка запроса на сервер и ожидание результата
Показ сообщения о неправильно введенном ID	Неправильно введено ID документа в поле “ID файла”	Показ соответствующего сообщения пользователю
Сравнение хешей и выдача соответствующего сообщения	Нажата кнопка “Сравнить хеши”	Сравнение хешей в полях “Хеш файла из блокчейна” и “Хеш 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 целостность”	Проверка целостности БД и вывод соответствующего результата пользователю

Диаграмма активности программы-клиента приведена на рисунке 14.

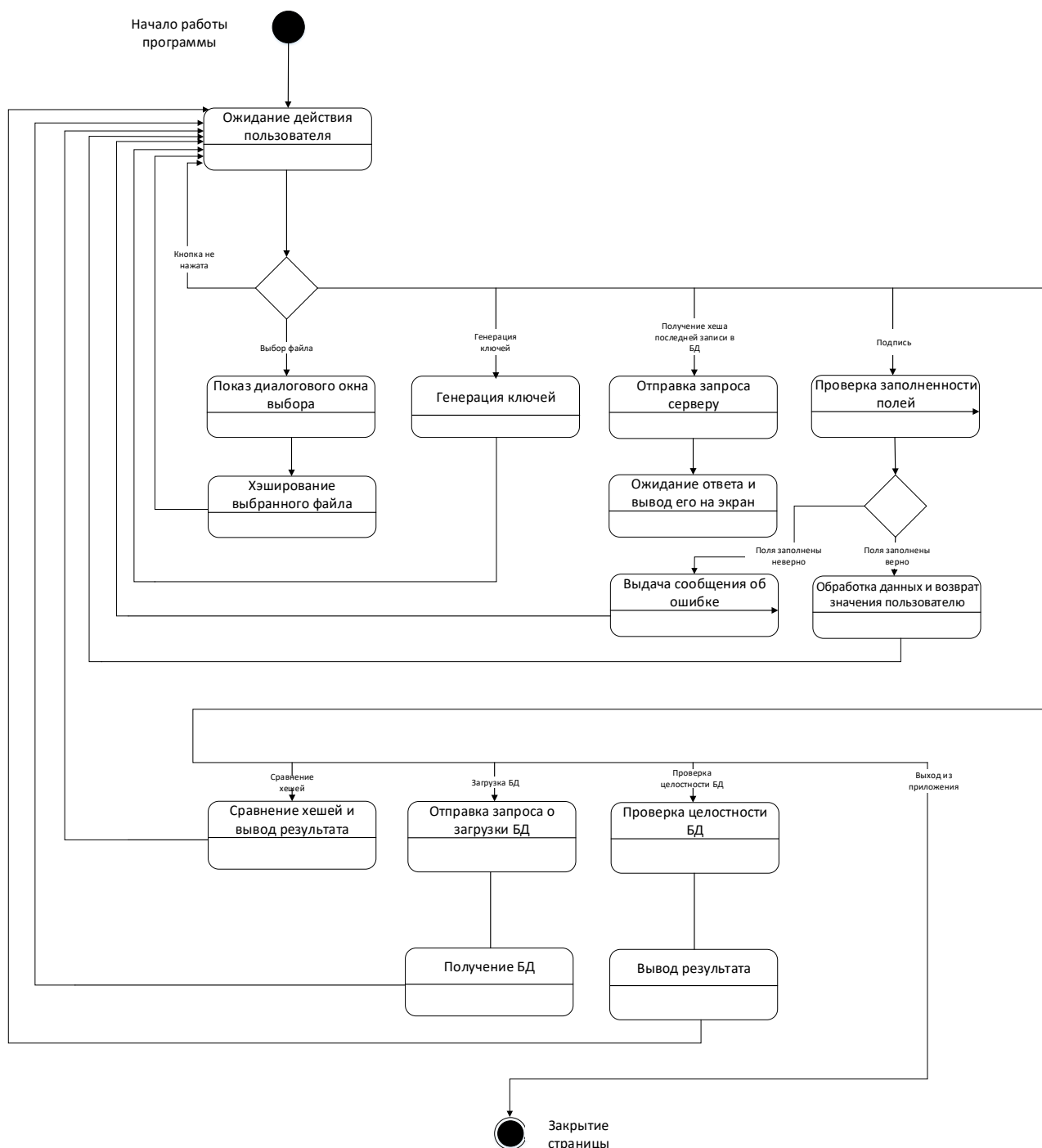


Рисунок 14 – UML- диаграмма активности программы-клиент

3.3 Описание серверного приложения

После запуска сервера, программа начинает прослушивать 5000й 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 прослушивания порта	Запуск программы	Запуск сервера и прослушивание 5000-го порта
Получение запроса на добавление записи в БД	-	Получение хеша последней записи, расшифровка подписи публичным ключом и получение из подписи хеша последней записи. Если хеши совпадают записать транзакцию в БД, иначе послать пользователю хеш 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 15.

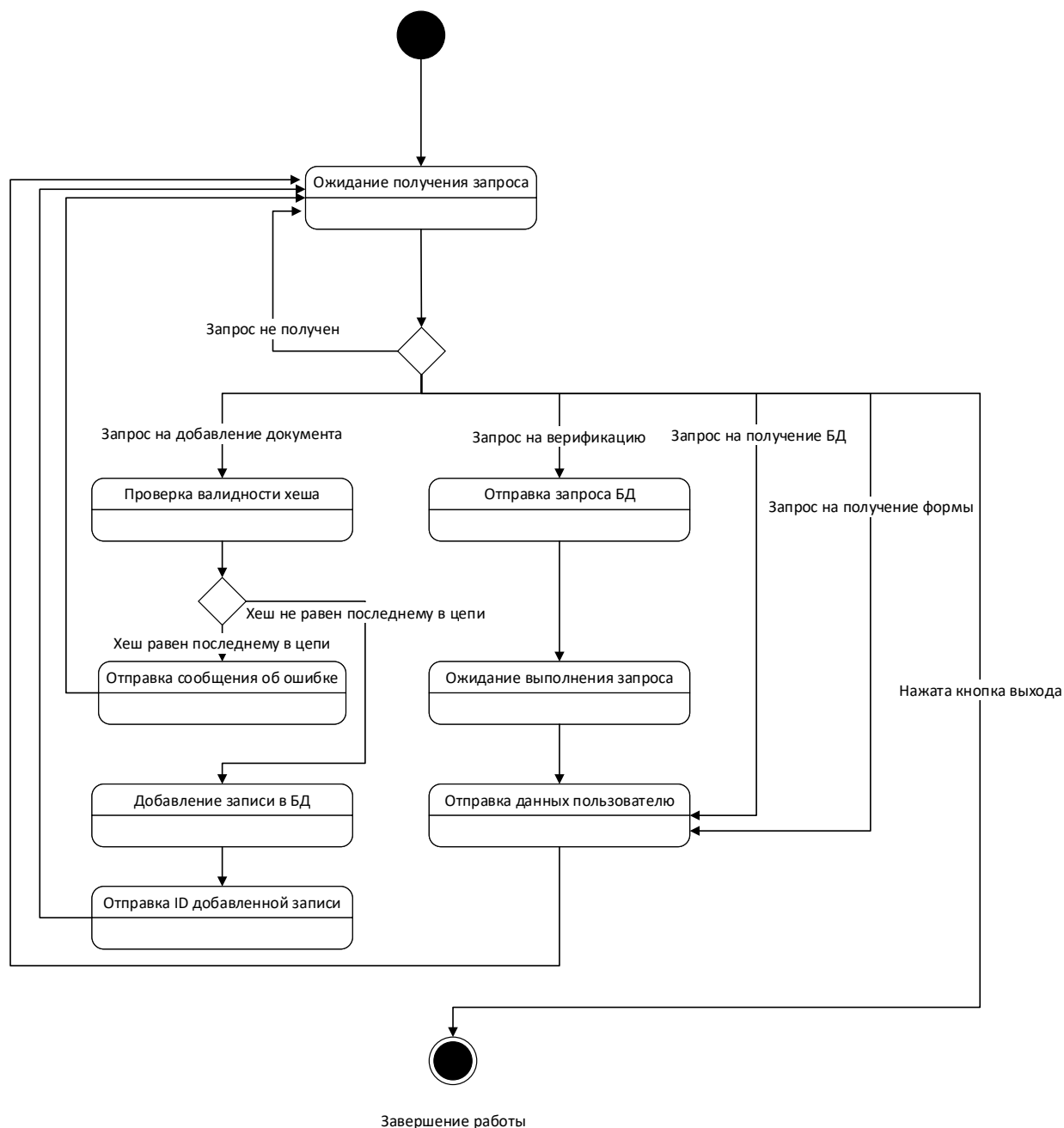


Рисунок 15 – UML- диаграмма активности программы-сервера

3.4 Руководство пользователя

При открытии клиентского приложения через браузер появляется форма, которое включает группы “Добавление транзакции”, “Проверка подлинности” и “Проверка блокчейна”, содержащие кнопки и поля ввода.

Добавление транзакции

Выберите файл

Файл не выбран

Хэш файла

Хэш файла

Закрытый ключ

Закрытый ключ

Открытый ключ

Открытый ключ

Сгенерировать пару ключей

Хэш последней транзакции в блокчейне:

Получить хэш последней транзакции

Подписать

Подпись

Подпись

Добавить транзакцию в блокчейн

Рисунок 16 – Область “Добавление транзакции” в клиентской программе

Для добавления нового файла в БД следует нажать на кнопку “Выберите файл” и в диалоговом окне выбрать файл, который будет добавлен в блокчейн. После указанных действий браузер с помощью встроенных скриптов в форму автоматически рассчитает хеш файла и вставит в текстовое поле “Хеш файла”.

Добавление транзакции

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

Закрытый ключ

Закрытый ключ

Открытый ключ

Открытый ключ

Сгенерировать пару ключей

Хэш последней транзакции в блокчейне:

Получить хэш последней транзакции Подписать

Подпись

Подпись

Добавить транзакцию в блокчейн

Рисунок 17 – автоматический расчет хеша файла

Затем следует ввести открытый и закрытый ключи в соответствующие поля “Открытый ключ” и “Закрытый ключ” или сгенерировать новые, нажав на кнопку “Сгенерировать пару ключей”.

Добавление транзакции

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

Закрытый ключ

b4c4a5e31e9b32c8165ff2a06ba407efb4afa49e396bbf1d068bdb8c0caf0a:

Открытый ключ

a14cec066404627b1675e285f6cd139b9e74037f92799be2a6f24de27c117

Сгенерировать пару ключей

Хэш последней транзакции в блокчейне:

Получить хэш последней транзакции Подписать

Подпись

Подпись

Добавить транзакцию в блокчейн

Рисунок 18 – генерация открытого и закрытого ключей

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

Добавление транзакции

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

Закрытый ключ

90d17ca48c14f0d023501a050fb80e4f63e6bf32322fc48788ec3d3694d358

Открытый ключ

21c6383ef7ca83f153559318dde6bbe8851677ed009f8fd5b469cabed5b2e

Сгенерировать пару ключей

Хэш последней транзакции в блокчейне:

8f02285ce4502b44202b75f33042ba3e

Получить хэш последней транзакции Подписать

Подпись

Подпись

Добавить транзакцию в блокчейн

Рисунок 19 – получение хеша последней транзакции в БД.

После этого можно подписать хэш последней транзакции и файла закрытым ключом, нажав на кнопку “Подписать”.

Добавление транзакции

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

Закрытый ключ

90d17ca48c14f0d023501a050fb80e4f63e6bf32322fc48788ec3d3694d358

Открытый ключ

21c6383ef7ca83f153559318dde6bbe8851677ed009f8fd5b469cabed5b2e

Сгенерировать пару ключей

Хэш последней транзакции в блокчейне:

8f02285ce4502b44202b75f33042ba3e

Получить хэш последней транзакции Подписать

Подпись

d5d94b3af2c8b5a95d4270c3e7123f0d46c21047b3657f151b28953153db4

Добавить транзакцию в блокчейн

Рисунок 20 – подпись данных

Следом нужно нажать на кнопку “Добавить транзакцию в блокчейн” 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 транзакции и хэш из расшифрованной подписи. Если хеши совпадают, то запись добавляется в БД и пользователю приходит соответствующее сообщение, в противном случае пользователя просят о повторе получения хеша и подписи документа.

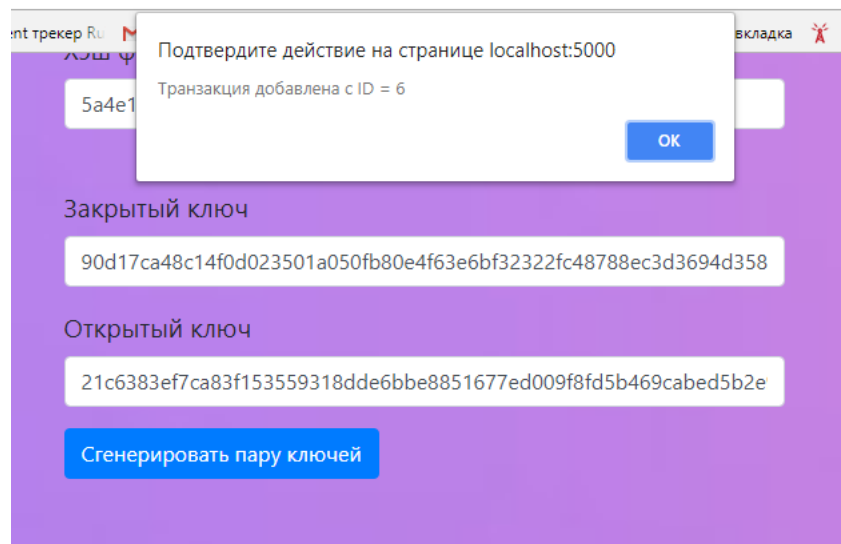


Рисунок 21 – успешное добавление транзакции в блокчейн

В случае успешного добавления файла появится сообщение с указанием ID добавленного файла, его следует сохранить для возможности последующей верификации документа.

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

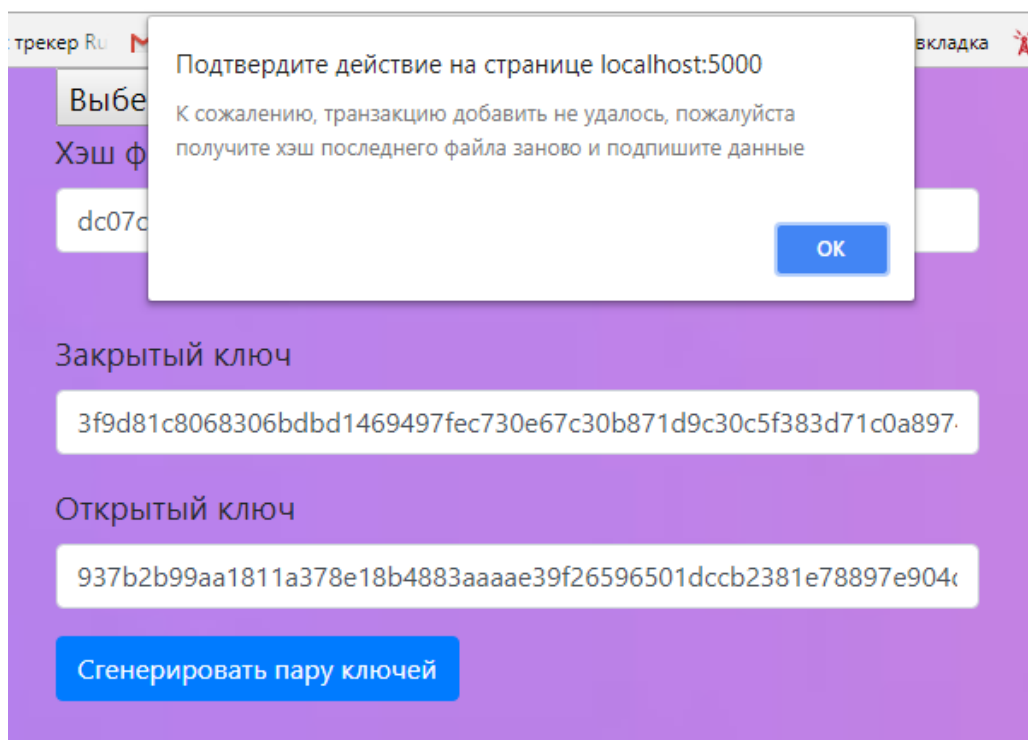


Рисунок 22 – неудачное добавление записи в БД

Для необходимости верификации документа необходимо перейти в раздел “Проверка подлинности”.

Рисунок 23 – раздел “Проверка подлинности”

После вышеуказанных действий потребуется выбрать файл с помощью кнопки “Выберите файл”, хеширование содержимого произойдет автоматически и отобразится в поле “Хэш файла”.

Следом нужно нажать кнопку “Хэш файла по ID”, после чего к серверу отправится запрос на получение хеша файла в записи с введенным ID. В результате в поле “Хэш файла из блокчейна” отобразится нужное значение.

Рисунок 24 – получение хеша файла по ID из БД

Если ID файла введено неверно – появится соответствующее сообщение.

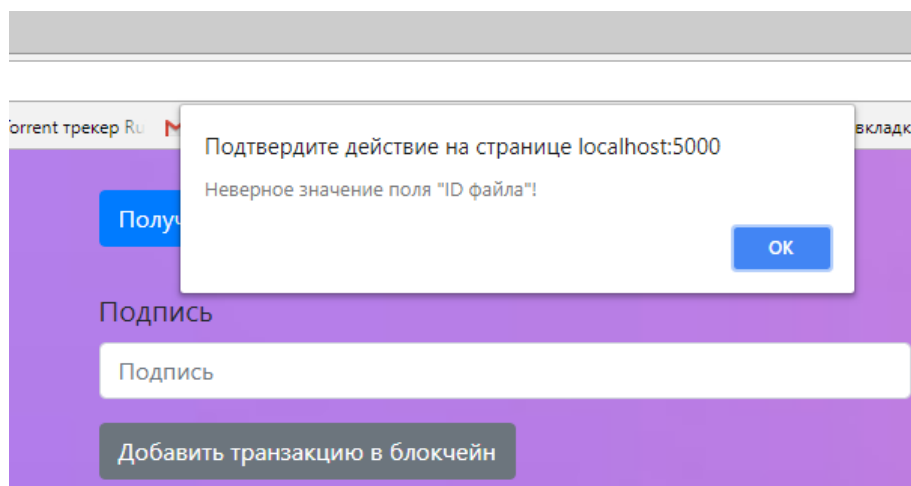


Рисунок 25 – неверный ввод ID

После получения хеша можно нажать на кнопку “Сравнить хеши” и получить вердикт об измененности выбранного документа и документа из блокчейна.

Если документ не был изменен с момента записи его в блокчейн, хеш сохранится и в поле результат проверки выведется значение “Документ не был изменен”.

Проверка подлинности

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

ID файла

14

Получить хэш файла по ID

Хэш файла из блокчейна:

5a4e1193a0185d206b62f26331499cc5

Сравнить хэши

Результат проверки: документ не был изменен

Рисунок 26 – вердикт об отсутствии изменений в документе

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

Проверка подлинности

Выберите файл Otchet_stu...e3333.doc

Хэш файла

5a4e1193a0185d206b62f26331499cc5

ID файла

ID файла

Получить хэш файла по ID

Хэш файла из блокчейна:

4d8cd9813b22ef6420e1066fb7702a71

Сравнить хэши

Результат проверки: документ был изменен

Рисунок 27 – вердикт об изменении документа

Для проверки целостности самого блокчейна и обнаружения факта возможных попыток его изменения, пользователю необходимо перейти в раздел проверка целостности блокчейна и нажать на кнопку “Загрузить блокчейн”.

Проверка целостности блокчейна

Загрузить блокчейн

Выберите файл Файл не выбран

Проверить целостность

Результат проверки

Рисунок 28 – раздел “Проверка целостности блокчейна”

После загрузки БД появится окно с выбором местоположения файла, после чего скачанный блокчейн можно проверить на целостность.

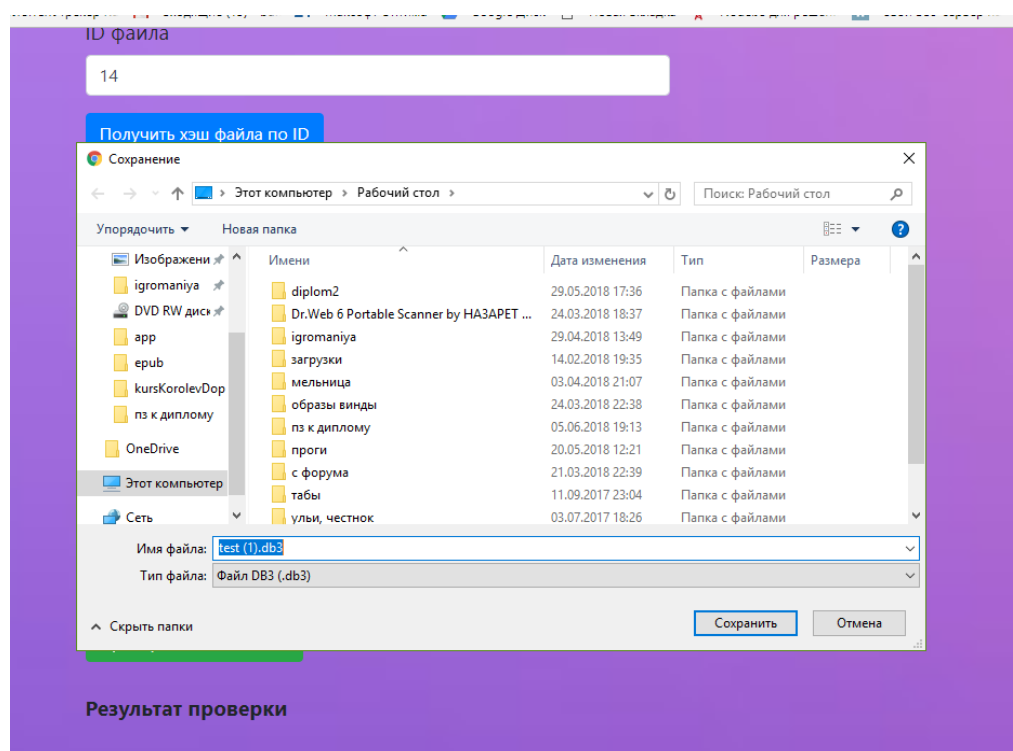


Рисунок 29 – скачивание БД.

Следом нужно нажать на кнопку “Выбрать файл” и выбрать соответствующую БД, программа начнет считывать из нее данные и по окончании выдаст сообщение о прочтении.

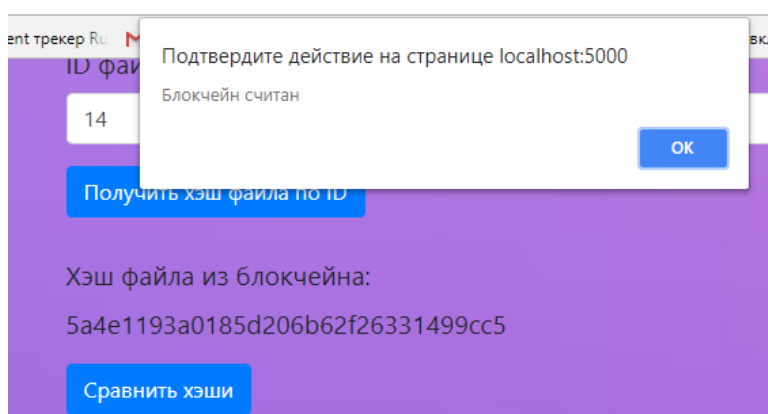


Рисунок 30 – сообщение об успешном чтении БД

Для проверки БД на целостность остается нажать на кнопку “Проверить целостность” и подождать, пока программа проверяет блокчейн. В основе проверки лежит процедура последовательного вычисления хеша предыдущей

строки, расшифровки открытым ключом подписи следующей строки и извлечения хеша предыдущей строки. Следом идет сравнения полученных хешей, в результате которого делается вердикт о неизменности записи.

Если были попытки изменить какую-либо транзакцию, в поле “Результат проверки” выдастся сообщение с ID измененного файла.

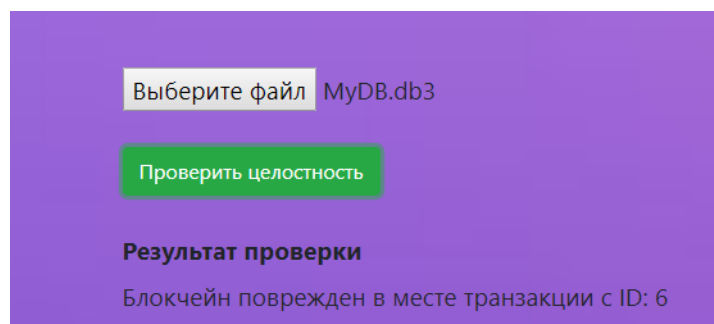


Рисунок 31 – повреждение блокчейна

В противном случае выведется сообщения о целостности БД.

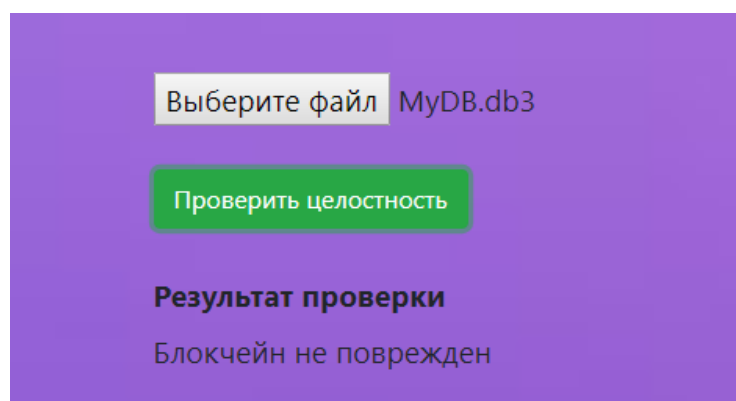


Рисунок 32 – сообщение о целостности блокчейна

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

```
C:\WINDOWS\system32\cmd.exe - python main.py
Microsoft Windows [Version 10.0.16299.461]
(c) Корпорация Майкрософт (Microsoft Corporation), 2017. Все права защищены.

C:\Users\User>cd diplom2/flask/app

C:\Users\User\diplom2\flask\app>python main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 692-463-629
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Рисунок 33 – запуск серверной программы

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

При получении различных запросов в консоль будет выводиться IP-адрес источника запроса, и тип запроса.

```
C:\Users\User\diplom2\flask\app>python main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environmen
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 692-463-629
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [05/Jun/2018 18:42:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2018 18:42:16] "GET /favicon.ico HTTP/1.1" 404 -
```

Рисунок 34 – принятие запросов сервером

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

ЗАКЛЮЧЕНИЕ

Результатом работы является веб-приложение, имеющее интуитивно понятный пользовательский интерфейс, 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.

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 веб-приложений, работы с базами данных.

Программа может быть улучшена добавлением личного кабинета с возможностью авторизации, сохранением истории добавленных файлов, а также возможностью регистрации и проверки папок с документами.

					ПГУ 1.090301.12.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Васильев А. Н. «JavaScript в примерах и задачах» Москва: Эксмо, 2017. – 720 с.
2. Марк Лутц «Learning Python» Москва: Символ-Плюс, 2011. – 848 с.
3. Дэвид Сойер Макфарланд «JavaScript и JQuery. Исчерпывающее руководство» Москва: Эксмо, 2017. – 880 с.
4. Эдди Османи «Паттерны для масштабируемых JavaScript-приложений» URL: <http://largescalejs.ru/what-exactly-is-a-large-javascript-application/>

ПРИЛОЖЕНИЕ А

Листинги программы сервера

Приложение А.1

Файл сервера 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(idOfFileParam)
    return json.dumps({'crypted_hash': result})
```

					ПГУ 1.090301.12.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

```

@app.route('/send_crypted_hash',methods=['POST','GET'])
def WriteBlockToBlockchain():
    CryptedHashParam = request.form['cryptedException']
    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 сервера 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):
```

					ПГУ 1.090301.12.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

```

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

    #получаем результат запроса

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
61

```

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()

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
62

ПРИЛОЖЕНИЕ В

Листинги программы клиента

Приложение В.1

Файл клиента base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset= "UTF-8">
  <title>Blockchain test</title>
  <script      src="https://cdnjs.cloudflare.com/ajax/libs/crypto-
js/3.1.2/rollups/md5.js" charset="utf-8"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/cryptico/0.0.134352294
0/cryptico.js" charset="utf-8"></script>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min
.js"></script>
  <script
src="http://gostcrypto.com/scripts/gost/gostCrypto.js"></script>
  <script
src="http://gostcrypto.com/scripts/gost/gostCoding.js"></script>
  <!--                                <script
src="http://kripken.github.io/sql.js/GUI/gui.js"></script>
  <script
src="https://raw.githubusercontent.com/kripken/sql.js/master/js/sq
l.js"></script>
  <script
src="http://kripken.github.io/sql.js/GUI/codemirror/mode/sql/sql.j
s"></script>
```

					ПГУ 1.090301.12.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

```

<script
src="https://raw.githubusercontent.com/kripken/sql.js/master/js/sql-
debug.js"></script>

<script
src="https://raw.githubusercontent.com/kripken/sql.js/master/js/work-
er.sql.js"></script>

<script
src="http://kripken.github.io/sql.js/GUI/codemirror/lib/codemirror
.js"></script>

<script
src='C://Users/User/diplom2/flask/app/js/sql.js'></script>-->

```

```

<script src="https://rawgit.com/kripken/sql.js/master/js/sql.js"
charset="utf-8"></script>

```

```

<script type="text/javascript">

```

```

    var db;

    var Signature;

    var PublicKey;

    var HashOfFile;

    var Message;

    var Id = 1;

    var HashOfPreviousTransaction;

    var locked;

    var resultOfVer;

    var stmt;

```

```

function readFile() {

```

```

    document.getElementById('exampleInputHashOfFile').value = "";

    var object = document.getElementById('fileInput');

    var file = object.files[0]

```

					ПГУ 1.090301.12.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

```

var reader = new FileReader()

// считываем бинарные данные файла в виде строки
reader.readAsBinaryString(file);

reader.onload = function() {

    //вставляем его хеш в поле
    //alert(reader.result);

    document.getElementById('exampleInputHashOfFile').value =
CryptoJS.MD5(reader.result).toString();

    }

}

function GenerateRSAKeys() {

    // Generate key pair
    gostCrypto.subtle.generateKey('GOST R 34.10', true,
['sign', 'verify']).then(function(keyPair) {

        // Store key in secluded place
        return      gostCrypto.subtle.exportKey('raw',
keyPair.privateKey).then(function(result) {

            document.getElementById('PrivateKey').value
= gostCrypto.coding.Hex.encode(result);

            // Provide the public key to receipient
            return      gostCrypto.subtle.exportKey('raw',
keyPair.publicKey).then(function(result) {

                document.getElementById('PublicKey').value =
gostCrypto.coding.Hex.encode(result);

                });

            });
}

```

```

        }).catch(function(error) {
            alert(error.message);
        });
    }

    function crypt(){

        if(document.getElementById('lastHash').innerHTML.length == 0)
        {
            alert('Поле "Хеш последней транзакции" не
заполнено!');

            return 0;
        }

        if(document.getElementById('exampleInputHashOfFile').value.len
gth == 0)
        {
            alert('Поле "Хеш файла" не заполнено!');
            return 0;
        }

        // Get private key from secluded place
        gostCrypto.subtle.importKey('raw',
gostCrypto.coding.Hex.decode(document.getElementById('PrivateKey')
.value),

                                'GOST            R            34.10',            true,
['sign']).then(function(key) {

                                // Use private key for signing message
                                return            gostCrypto.subtle.sign('GOST            R
34.10/GOST R 34.11', key,

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
66

```

        gostCrypto.coding.Chars.decode(document.getElementById('lastHash').innerHTML + ' ' + document.getElementById('exampleInputHashOfFile').value));

        }).then(function(result) {

            // Send signature with message

            document.getElementById('cryptedException').value =
gostCrypto.coding.Hex.encode(result);

        }).catch(function(error) {

            alert(error.message);

        });

    }

    function verify_hashes(){

        document.getElementById('ResultOfVerification').innerHTML =
"<b>Результат проверки:</b> документ был изменен";

        if(document.getElementById('HashOfFileVerification').value.length == 0)

        {

            alert('Поле "Хеш файла" не заполнено!');

            return 0;

        }

        if(document.getElementById('CryptedExceptionFromBlockchain').innerHTML.length == 0)

        {

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
67

```

        alert('Поле "Хеш файла из блокчейна" не
заполнено!');

        return 0;
    }

    if (document.getElementById('HashOfFileVerification').value ===
document.getElementById('CryptedHashFromBlockchain').innerHTML) {

        document.getElementById('ResultOfVerification').innerHTML
="<b>Результат проверки:</b> документ не был изменен";

        return 0;
    }

    else{

document.getElementById('ResultOfVerification').innerHTML      =
"<b>Результат проверки:</b> документ был изменен";

        return 0;
    }

}

function readFileVerification(){
    document.getElementById('HashOfFileVerification').value = "";

    var                                object                                =
document.getElementById('fileInputVerification');

    var file = object.files[0]
    var reader = new FileReader()

    // считываем бинарные данные файла в виде строки
    reader.readAsBinaryString(file);

    reader.onload = function() {

        //вставляем его хеш в поле

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
68

```

        document.getElementById('HashOfFileVerification').value =
CryptoJS.MD5(reader.result).toString();

    }

}

function check_integrity()

{

        window.stmt = db.prepare("SELECT
id,Signature,PublicKey,HashOfFile FROM transactions order by id");

        if(window.stmt.step())
        {
                var row = window.stmt.getAsObject();

                window.HashOfPreviousTransaction =
CryptoJS.MD5(String(row.Id) + row.Signature + row.PublicKey +
row.HashOfFile).toString();

        }
        else
        {
                document.getElementById('ResultOfCheckingIndex').innerHTML =
'База пуста';

                return 0;

        }

        check();

}

function check()

{

        if(window.stmt.step())

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.12.001 ПЗ

Лист
69

```

{
    var row = window.stmt.getAsObject();
    window.Id = row.Id;
    console.log('test id ' + window.Id )
    window.PublicKey = row.PublicKey;
    window.Signature = row.Signature;
    window.HashOfFile = row.HashOfFile;
    window.Message =
window.HashOfPreviousTransaction + ' ' + row.HashOfFile;

    // Get public key from trusted source
    gostCrypto.subtle.importKey('raw',
gostCrypto.coding.Hex.decode(window.PublicKey),
                                'GOST R 34.10', true,
['verify']).then(function(key) {

        // Use public key for verify message
signature
        return gostCrypto.subtle.verify('GOST R
34.10/GOST R 34.11', key,

        gostCrypto.coding.Hex.decode(window.Signature),
gostCrypto.coding.Chars.decode(window.Message));

    }).then(function(result) {
        // Check result
        if (result)
        {
            window.HashOfPreviousTransaction =
CryptoJS.MD5(String(window.Id) + window.Signature +
window.PublicKey + window.HashOfFile).toString();

```

					ПГУ 1.090301.12.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

        check();

    }

    else

    {

        document.getElementById('ResultOfCheckingIndex').innerHTML =
        'Блокчейн поврежден в месте транзакции с ID: ' +
        (Number(window.Id)-1);

    }

    }).catch(function(error) {

        alert(error.message);

    });

}

else if(window.Id > 1)

{

    document.getElementById('ResultOfCheckingIndex').innerHTML =
    'Блокчейн не поврежден';

    }

    else

    {

        document.getElementById('ResultOfCheckingIndex').innerHTML =
        'Блокчейн пуст';

        return 0;

    }

}

function loadDBFromFile()

{

```

```

var dbFileElm =
document.getElementById('fileInputCheckIntegrity');

var f = dbFileElm.files[0];
var r = new FileReader();
r.onload = function() {
    var Uints = new Uint8Array(r.result);
    window.db = new SQL.Database(Uints);
    alert("Блокчейн считан");
}
r.readAsArrayBuffer(f);
}

```

</script>

<script>

```

function get_last_hash() {
//получение хеша последней записи в блокчейне
$.ajax({
    type: "POST",
    url: "/get_last_hash",
    data: $('form').serialize(),
    success: function(response) {
        var json = jQuery.parseJSON(response)
        console.log(json.Id);
        console.log(json.Signature);
        console.log(json.PublicKey);
        console.log(json.HashOfFile);
    }
}

```

```

        lastHash = CryptoJS.MD5(json.Id +
json.Signature + json.PublicKey + json.HashOfFile).toString();
        $('#lastHash').html(lastHash)

        console.log(response);
    },
    error: function(error) {
        console.log(error);
    }
});
}

function get_crypted_hash_by_id(){
    //проверка id на целое число

    if(isNaN(parseInt(document.getElementById('IdOfFile').value,10
))) {

        alert('Неверное значение поля "ID файла"!');

        document.getElementById('CryptedHashFromBlockchain').value =
        "";

        return 0;
    }

    document.getElementById('IdOfFile').value =
    parseInt(document.getElementById('IdOfFile').value,10);

    document.getElementById('CryptedHashFromBlockchain').value =
    "";

    $.ajax({
        type: "POST",
        url: "/get_crypted_hash_by_id",

```

```

        data: $('form').serialize(),
        success: function(response) {
            var json = jQuery.parseJSON(response)

document.getElementById('CryptedHashFromBlockchain').innerHTML    =
json.crypted_hash;

            console.log(response);
        },
        error: function(error) {
            console.log(error);
        }
    });
}

function send_crypted_hash(){
    //если у нас нет хеша последней строчки блокчейна -
    предупредить
    if(document.getElementById('lastHash').innerHTML.length
== 0)
    {
        alert('Поле "Хеш последней транзакции в блокчейне"
не заполнено!');
        return 0;
    }
    if(document.getElementById('PublicKey').value.length ==
0)
    {
        alert('"Public key is empty!');
        return 0;
    }
}

```

```

        if (document.getElementById('exampleInputHashOfFile').value.length == 0)
        {
            alert('Поле "Хеш файла" не заполнено!');
            return 0;
        }
        if (document.getElementById('cryptedException').value.length == 0)
        {
            alert('Поле "Подпись" не заполнено!');
            return 0;
        }
        $.ajax({
            type: "POST",
            url: "/send_cryptedException",
            data: $('form').serialize(),
            success: function(response) {
                var json = jQuery.parseJSON(response)
                alert(json.response);
            },
            error: function(error) {
                console.log(error);
            }
        });

    }

    function load_database()
    {

```

| | | | | | | |
|------|------|----------|---------|------|------------------------|------|
| | | | | | ПГУ 1.090301.12.001 ПЗ | Лист |
| | | | | | | 75 |
| Изм. | Лист | № докум. | Подпись | Дата | | |

```

$.ajax({
    type: "POST",
    url: "/load_database",
    success: function(response) {
        resp = btoa(response);
        download(response, "test.db3", "");
    },
    error: function(error) {
        console.log(error);
    }
});
}

// Function to download data to a file
function download(data, filename, type) {
    var file = new Blob([data], {type: type});

    if (window.navigator.msSaveOrOpenBlob) // IE10+
        window.navigator.msSaveOrOpenBlob(file, filename);
    else { // Others
        var a = document.createElement("a"),
            url = URL.createObjectURL(file);
        a.href = url;
        a.download = filename;
        document.body.appendChild(a);
        a.click();
        setTimeout(function() {
            document.body.removeChild(a);
            window.URL.revokeObjectURL(url);
        }, 0);
    }
}

```

```

    }

}

</script>

<link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstra
p.min.css"                                integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

</head>
<body>
    <style>
        body {
            background-image:
url(https://avatanplus.com/files/resources/mid/577adaf576796155b7e
74eed.jpg);
            background-repeat: no-repeat;
            background-size: 200%;
            font-size: 120%;
        }
    </style>
<div class = "container">
    <div class ="row">
        <div class = "col-6 mx-auto mt-5">
            {% block content %}
            {% endblock %}
        </div>
    </div>

```

| | | | | | | |
|------|------|----------|---------|------|------------------------|------|
| | | | | | ПГУ 1.090301.12.001 ПЗ | Лист |
| | | | | | | 77 |
| Изм. | Лист | № докум. | Подпись | Дата | | |

```

</div>

<div class ="container">
    <div class ="row">
        <div class = "col-6 mx-auto mt-5 text-center">

            <!--<form action = "{{url_for('check')}}" method = "GET">
                <button    type="submit"    class="btn    btn-success">check
integrity</button>
            </form>-->
        </div >
    </div>

    <div class ="row">
        <div class = "col-6 mx-auto mt-5">
            {% block checking %}

            {% endblock %}
        </div>
    </div>
</div>
</body>
<script type="text/javascript">
</script>
</html>

```

Приложение В.2 Файл клиента index.html

```
{% extends 'base.html' %}

{% block content %}

    <form action="{{ url_for('index')}}" method="POST">

    <div class="form-group">

        <label for="AddingTransactionLabel" ><b> <ins>      Добавление
транзакции </ins></b></label>

    </div>

    <input          type="file"          name="img"          id='fileInput'
onchange="readFile()">

    <br>

    <div class="form-group">

        <label for="exampleInputHashOfFileLabel" >Хеш файла</label>

        <input          type="text"          class="form-control"
id="exampleInputHashOfFile"      placeholder="Хеш      файла"      name
="HashOfFile">

    </div>

    <br>

    <div class="form-group">

        <label for="exampleInputPrivateKeyLabel">Закрытый ключ</label>

        <input          class="form-control"          id="PrivateKey"
placeholder="Закрытый ключ" name = "PrivateKey">

    </div>

    <div class="form-group">

        <label for="exampleInputPublicKeyLabel">Открытый ключ</label>
```

| | | | | | | |
|------|------|----------|---------|------|------------------------|------|
| | | | | | ПГУ 1.090301.12.001 ПЗ | Лист |
| | | | | | | 79 |
| Изм. | Лист | № докум. | Подпись | Дата | | |

```

        <input      type="text"      class="form-control"      id="PublicKey"
placeholder="Открытый ключ" name = "PublicKey">
    </div>

    <button type="button" onclick = "GenerateRSAKeys()" class="btn
btn-primary">Сгенерировать пару ключей</button>

    <br>

    <br>

    <br>

    <div class="form-group">
        <label          for="exampleInputLastHashInBlockchainLabel">Хеш
последней транзакции в блокчейне:</label>
    </div>

    <div id="lastHash"></div>

    <br>

    <button type="button" onclick = "get_last_hash();" class="btn
btn-primary">Получить хеш последней транзакции</button>

    <button type="button"      onclick = "crypt();" class="btn btn-
primary">Подписать</button>

    <br>

    <br>

    <div class="form-group">
        <label for="cryptedExceptionLabel">Подпись</label>
        <input type="text"      class="form-control"      id="cryptedException"
placeholder="Подпись" name = "cryptedException">
    </div>

    <button type="button"      onclick = "send_cryptedException();"
class="btn btn-secondary">Добавить транзакцию в блокчейн</button>

    <br>

    <br>

    <div class="form-group">

```

```

        <label for="VerificationLabel"><b> <ins> Проверка подлинности
</ins></b></label>

    </div>

    <input type="file" name="img" id='fileInputVerification'
onchange="readFileVerification();">

    <br>

    <div class="form-group">

        <label for="HashOfFileVerificationLabel" >Хеш файла</label>

        <input type="text" class="form-control"
id="HashOfFileVerification" placeholder="Хеш файла" name
="HashOfFileVerification">

    </div>

    <div class="form-group">

        <label for="exampleInputIdOfFileLabel">ID файла</label>

        <input type="text" class="form-control" id="IdOfFile"
placeholder="ID файла" name = "IdOfFile">

    </div>

    <button type="button" onclick = "get_crypted_hash_by_id();"
class="btn btn-primary">Получить хеш файла по ID</button>

    <br>

    <br>

    <div class="form-group">

        <label for="exampleInputCryptedHashOfFile">Хеш файла из
блокчейна: </label>

        <div id="CryptedHashFromBlockchain"></div>

    </div>

    <button type="button" onclick = "verify_hashes();" class="btn
btn-primary">Сравнить хеши</button>

    <div class="form-group">

        <br>

        <label id="ResultOfVerification"
for="exampleInputResultLabel"></label>

```

```

</div>

<div class="form-group">
    <label      for="CheckIntegrityLabel"><b>      <ins>      Проверка
целостности блокчейна </ins></b></label>
</div>

<div><button  type="button"  onclick  =  "load_database();"
class="btn btn-warning">Загрузить блокчейн</button></div>

<br>

<br>

<input      type="file"      name="img2"      accept=".db3"
onchange="loadDBFromFile()" id='fileInputCheckIntegrity'>

<br>

<br>

<button type="button" onclick = "check_integrity();" class="btn
btn-success">Проверить целостность</button>

<div class="form-group">

<br>

<label      for="ResultOfCheckingLabel"><b>Результат      проверки</b>
</label>

<div id="ResultOfCheckingIndex"></div>

</div>

</form>

{% endblock %}

{% block checking %}

    {% for res in results %}

        <div class = "result">

            block {{res['block']}}: {{res['result']}}

        </div>

    {% endfor %}

{% endblock %}

```

ПРИЛОЖЕНИЕ С

Презентация



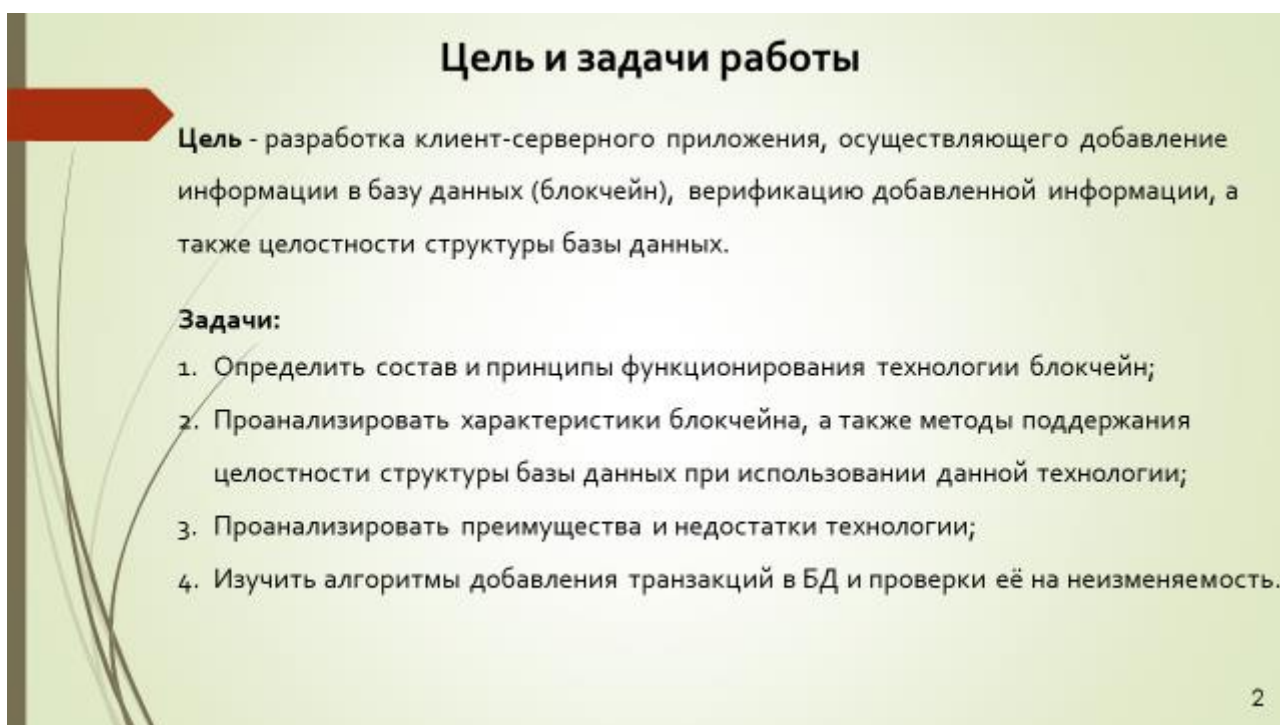
Тема работы:

«Единый портал учёта документов с использованием технологии Blockchain. Web-приложение»

Выполнил:
студент группы 14ВВ1 Королев Б.Д.

Научный руководитель:
к.т.н., доцент Трокоз Д.А.

Пенза, 2018



Цель и задачи работы

Цель - разработка клиент-серверного приложения, осуществляющего добавление информации в базу данных (блокчейн), верификацию добавленной информации, а также целостности структуры базы данных.

Задачи:

1. Определить состав и принципы функционирования технологии блокчейн;
2. Проанализировать характеристики блокчейна, а также методы поддержания целостности структуры базы данных при использовании данной технологии;
3. Проанализировать преимущества и недостатки технологии;
4. Изучить алгоритмы добавления транзакций в БД и проверки её на неизменяемость.

2

| | | | | | | |
|------|------|----------|---------|------|------------------------|------|
| | | | | | ПГУ 1.090301.12.001 ПЗ | Лист |
| | | | | | | 83 |
| Изм. | Лист | № докум. | Подпись | Дата | | |

Определение технологии «Блокчейн»



Блокчейн — это способ хранения данных (цифровой реестр транзакций). В него можно добавлять данные практически обо всем: о выданных кредитах, правах на собственность и т.д.

3

Виды блочных цепей

Публичная БЦ



БЦ, принадлежащая консорциуму



4

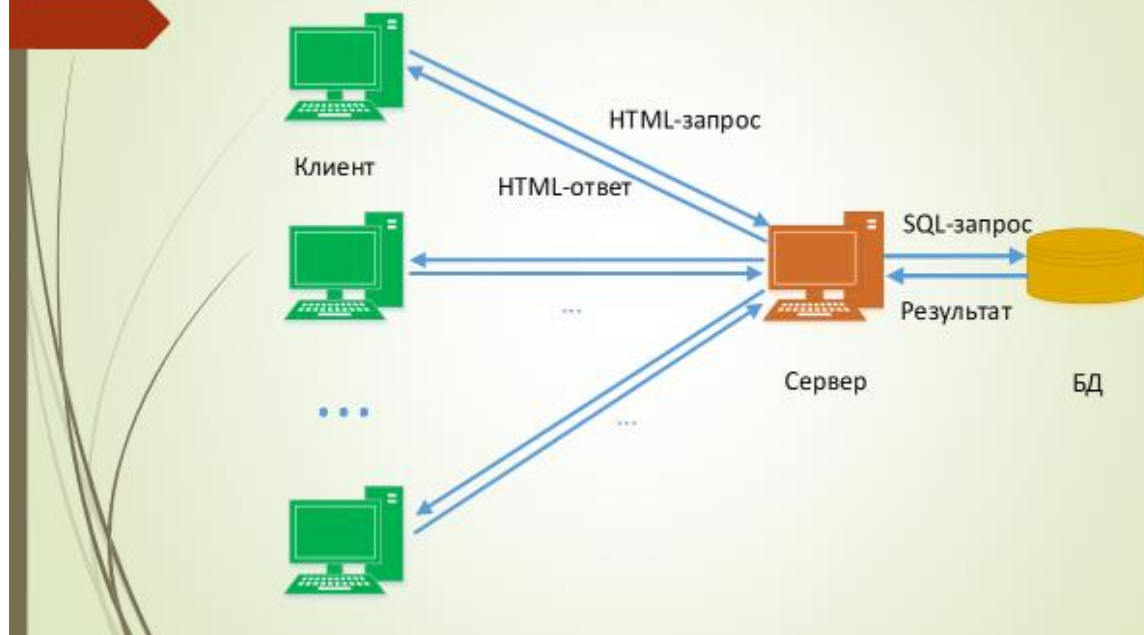
| | | | | |
|------|------|----------|---------|------|
| | | | | |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.12.001 ПЗ

Лист
84

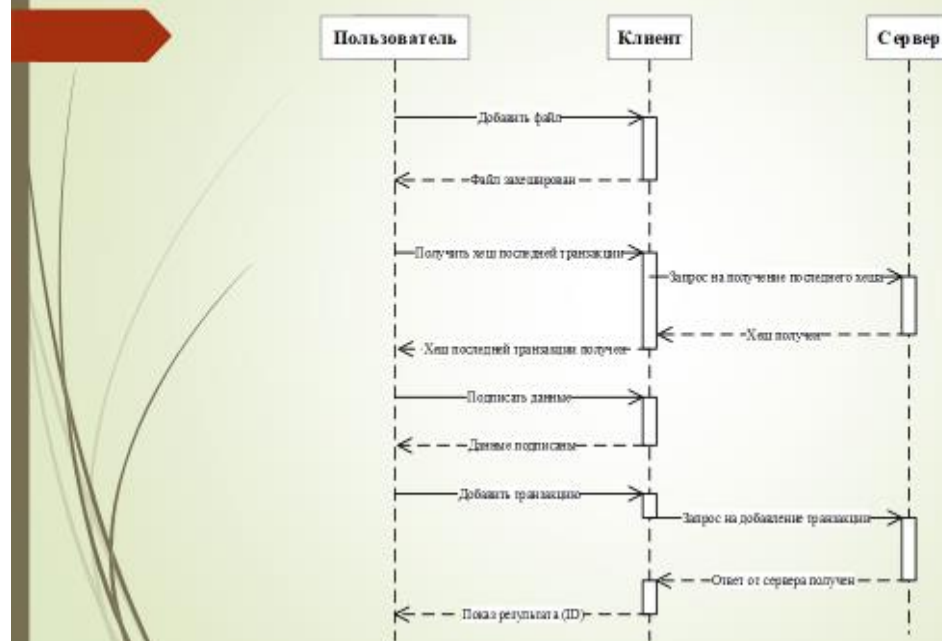


Схема системы



8

Диаграмма последовательности (добавление файла)



9

Добавление транзакции в блокчейн

| Id | Signature | PublicKey | HashOfFile |
|----|-----------------|--------------------|---------------|
| 1 | 0000 | 0000 | 0000 |
| 2 | 0dbd528771d8... | 3ea5cb9201d732... | 4d8cd9813b... |
| 3 | c85e2a1af276... | 7cb954f49f27b11... | 83f75adaa1... |
| 4 | 91f70d16ac44... | 6a5a1261e9cceb... | 9aa6a58e8c... |
| 5 | 47cbb599fb1a... | df881cefb8448a0... | b867200a60... |

→ Hash + HashOfFile

Crypt(Hash + HashOfFile)

10

Область формы для добавления транзакции

Выберите файл:

Хэш файла:

Закрывать ключ:

Открытый ключ:

Хэш последней транзакции в блокчейне:

Подпись:

11

Диаграмма последовательности (верификация файла)



12

Область формы для верификации файла

Проверка подлинности

Выберите файл

Хэш файла:

ID файла:

Хэш файла из блокчейна:

4d8cd9813b22ef6420e1066fb7702a71

Результат проверки: документ был изменен

13

| | | | | |
|------|------|----------|---------|------|
| | | | | |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ 1.090301.12.001 ПЗ

Лист
88

Проверка блокчейна на целостность

| Id | Signature | PublicKey | HashOfFile |
|----|-----------------|--------------------|---------------|
| 1 | 0000 | 0000 | 0000 |
| 2 | 0dbd528771d8... | 3ea5cb9201d732... | 4d8cd9813b... |
| 3 | c85e2a1af276... | 7cb954f49f27b11... | 83f75adaa1... |
| 4 | 91f70d16ac44... | 6a5a1261e9cceb... | 9aa6a58e8c... |
| 5 | 47cbb599fb1a... | df881cefb8448a0... | b867200a60... |

Decrypt(Signature, PublicKey) → Hash + HashOfFile

Hash

= → Результат

14

Область формы для проверки целостности блокчейна

Проверка целостности блокчейна

Загрузить блокчейн

Выберите файл MyDB.db3

Проверить целостность

Результат проверки

Блокчейн не поврежден

Спасибо за внимание

Выступал: Королев Богдан –
студент группы 14BB1