МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Системное программирование и компьютерные технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Реализация аутентификации пользователя в соответствии с законом «О персональных данных» на стороне сервера для информационной системы «Медик»

Студент А.И. Белоусов
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н., С.В. Баумгертнер
(ученая степень, звание, И.О. Фамилия)

Консультант К.А. Селиверстова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Данная дипломная работа посвящена теме аутентификации пользователя на стороне сервера. Данная работа предназначена для решения одной из актуальных на сегодняшний день проблем – проблемы определения того, что под данным логином и паролем зашел именно тот человек, который владеет данной учетной записью. Для этого, в данной работе разработан алгоритм аутентификации пользователя на стороне сервера в соответствии с законом «О персональных данных» и ГОСТ Р ИСО/МЭК 9594-8-98 «Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации». Разработанный алгоритм предусматривает введение кода подтверждения, который посылается на подтвержденную почту клиенту, при введении верного логина и пароля. Также, для большей защиты пользователей, в данной работе применяется особый способ хранения паролей на стороне сервера, таким методом является «соленое» хэширование с применением алгоритма хэширования «Стрибог».

В данной работе применяется не просто один из алгоритмов хэширования для хранения паролей, к этому алгоритму еще добавлен дополнительный слой защиты в виде метода «соленого» хэширования.

Результатом данной работы является спроектированная и реализованная система аутентификации пользователя, которая соответствует закону «О персональных данных» и ГОСТ к аутентификации пользователя в информационных системах, где хранятся конфиденциальные данные, с применением функции хэширования, которая является ГОСТ Р 34.11-2012, а также добавлением к ней дополнительной защиты от различного вида атак перебором паролей, так как в данной системе будут храниться и обрабатываться конфиденциальные данные о пациентах, а конфиденциальность таких данных очень важна.

Дипломная работа включает: 45 страниц, 12 иллюстрации, 2 таблицы, три части и 20 источников.

Ключевые слова: аутентификация, пользователь, сервер, хэширование.

Объект исследования: аутентификация пользователя.

Цель работы: реализация системы аутентификации пользователя на стороне сервера.

Задачи: ознакомится с научными работами на схожую или смеженную тематику, изучить необходимую литературу, спроектировать и реализовать программный модуль аутентификации пользователя в соответствии с законом «О персональных данных», протестировать реализованный модуль, сделать выводы, основываясь на результатах тестирования.

В ходе исследования был применен язык программирования Java с его различными фреймфорками и интерфейсами.

Полученные результаты: спроектирован и реализован программный модуль аутентификации пользователя. Был произведен анализ данных, которые были получены в результате тестирования модуля.

Область применения и значимость работы: результаты работы могут быть использованы при дальнейшем развитии данного программного модуля для аутентификации пользователей в информационной системе «Медик», а также в других системах.

Abstract

The present graduation work is devoted to “Implementation of user authentication according to 'About protection of personal data' law on the server side for 'Medic' system”. This graduation work must resolve the actual problem – user authentication.

The graduation work contains of an explanatory note on 45 pages, introduction, including 12 figures, 2 tables, the list of 20 references including 5 foreign sources and 1 appendix.

The aim of graduation work is the program realization of user authentication system working on server side.

The object of the graduation work is a user authentication.

The subject of the graduation work is algorithm of authentication users on server side.

The program is written in Java language with various frameworks. We use software platform IntelliJ IDEA.

During this graduation work we implemented algorithm of a user authentication on server side. This algorithm uses “Stribog” hash algorithm and some features. This feature includes using salt to take a hash for a password. Using a hash method, we need to safe storage of users’ passwords. Also, for advanced security during authentication we send verify code to user via email.

In conclusion, we would like to stress that this topic is relevant today, because to access sensitive data is very important thing. This access should be granted to a limited circle of people, especially when it comes to patient data.

Содержание

Введение 7

1 Анализ задачи аутентификации пользователя в соответствии с законом «О персональных данных» для информационной системы «Медик» 10

1.1 Анализ требований к аутентификации в информационной системе «Медик» 10

1.2 Анализ существующих технологий аутентификации 13

1.3 Постановка задачи на разработку системы аутентификации пользователя для информационной системы «Медик» 15

2 Проектирование и разработка алгоритма аутентификации пользователя в информационной системе «Медик» 17

2.1 Проектирование технологии аутентификации пользователя для информационной системы «Медик» 17

2.2 Разработка архитектуры модуля аутентификации пользователя на стороне сервера в соответствии с законом «О персональных данных» 23

2.3 Реализация алгоритма аутентификации пользователя в соответствии с требованиями закона «О персональных данных» 30

2.4 Разработка диаграммы развертывания системы аутентификации пользователя на стороне сервера 32

3 Оценка эффективности работы алгоритма аутентификации пользователя в соответствии с требованиями закона «О персональных данных» 34

3.1 Проведение статистических исследований возможности несанкционированного доступа к персональным данным пациентов .. 34

3.2 Проверка надежности процесса аутентификации пользователя в информационной системе «Медик» 37

Заключение 40

Список используемой литературы и используемых источников 42

Приложение А Файловые ресурсы разработанного модуля аутентификации для информационной системы «Медик» 45

Введение

В выпускной квалификационной работе (ВКР) будет рассмотрено решение задачи аутентификации пользователя на стороне сервера.

Сегодня аутентификация пользователя имеет большое значение, так как есть очень много сведений, для которых необходимо сохранить конфиденциальность, а значит и доступ к таким данным должен быть под контролем. То есть, к определенной информации должен иметь доступ определенный круг лиц.

В соответствии с настоящим законодательством РФ, для передачи и хранения персональных данных Федеральный закон «О персональных данных» устанавливает требование, в соответствии с которым хранение данных нужно осуществлять в форме, которая позволяет определить субъекта персональных данных не дольше, чем того требует цель обработки, данные подлежат уничтожению по достижении цели обработки или в случае утраты необходимости в их достижении.

На сегодняшний день, в информационной системе «Медик» существует возможность получить защищенный доступ к данным с помощью стационарных компьютеров. Также передача данных в информационной системе «Медик» осуществляется с использованием беспроводной технологии Wi-Fi. Но нет никакой гарантии, что вошедший с устройства с ОС Android является тем, за кого себя выдает. Поэтому, для большей безопасности пользования системой необходимо, чтобы можно было осуществлять защищенный вход в систему с любого устройства, для этого на стороне сервера нужно определять, что с данного устройства зашел именно тот человек, который является владельцем данной учетной записи. Данная работа как раз-таки и будет посвящена теме аутентификации пользователя на стороне сервера.

Проблема аутентификации пользователя рассматривается в таких работах, как «Аутентификация и авторизация» [10], «Современные методы

аутентификации: токен и это все о нем» [12], «Обеспечение безопасности протокола ssh: шифрование, аутентификация сервера, аутентификация клиента» [14]. В работе «Аутентификация и авторизация» говорится о различиях аутентификации и авторизации, о применении токенов для того, чтобы каждое действие пользователя было подтверждено, что именно этот пользователь производит те или иные действия в системе, но в этой работе не рассматривается, каким образом передаются данные между клиентом и сервером и самое главное, подробно не рассматривается, как определить на стороне сервера, что в систему пытается войти именно тот пользователь, за которого он себя выдает. В работе «Обеспечение безопасности протокола ssh: шифрование, аутентификация сервера, аутентификация клиента» рассматривается как раз защищенная передача данных при аутентификации пользователя и сервера, а также различные способы аутентификации, в работе «Современные методы аутентификации: токен и это все о нем» рассматриваются способы аутентификации пользователя, а также шифрование данных во время их передачи, помимо того говорится о портативных смарт-картах. В этих работах также рассматривается как определить на стороне сервера, что в систему пытается войти именно тот пользователь, за которого он себя выдает, но раскрывается эта тема не полностью, больший упор идет на способы защиты данных, при их передаче, а также на аппаратный способ аутентификации, что не подходит для информационной системы «Медик», так как, во-первых, в смартфон или планшет не вставить физический токен, во-вторых, это все стоит не малых средств. Исходя из рассмотренных источников, можно сделать вывод, что тема аутентификации пользователя раскрыта, но не в том аспекте, в котором она рассматривается в данной работе, поэтому тема аутентификации пользователя нуждается в дальнейшей разработке.

Объектом исследования является аутентификация пользователя.

Предметом исследования является алгоритм аутентификации пользователя на стороне сервера.

Целью работы является реализация алгоритма аутентификации пользователя на стороне сервера.

Для достижения цели необходимо решить ряд задач: ознакомится с научными работами на схожую или смеженную тематику, изучить необходимую литературу, спроектировать и реализовать программный модуль аутентификации пользователя в соответствии с законом «О персональных данных», протестировать реализованный модуль, сделать выводы, основываясь на результатах тестирования.

В первой главе работы производится анализ задачи аутентификации пользователя в соответствии с законом «О персональных данных» для информационной системы «Медик», а именно производится анализ требований к аутентификации в информационной системе «Медик», описывается сама информационная система, также производится анализ различных способов аутентификации и производится выбор способа аутентификации, который будет реализован, производится постановка задачи на разработку системы аутентификации пользователя в информационной системе «Медик» в соответствии с законом «О персональных данных» и ГОСТ для аутентификации в системах, где хранится конфиденциальная информация.

Во второй главе проектируется алгоритм функционирования системы аутентификации пользователя на стороне сервера, а также описывается математическая часть решаемой задачи.

В третьей главе тестируется разработанный программный модуль и анализируются результаты тестирования. Исходный код разработанного программного модуля находится на компакт-диске, файловая структура которого приведена в Приложении А.

1 Анализ задачи аутентификации пользователя в соответствии с законом «О персональных данных» для информационной системы «Медик»

1.1 Анализ требований к аутентификации в информационной системе «Медик»

«Медик» – это информационная система, которая разрабатывается для ГБУЗ СО ТГКБ №5 – Медгородок.

Разрабатываемая информационная система будет использоваться для передачи, хранения, поиска, а также обработки информации, о пациентах ГБУЗ СО ТГКБ №5.

Целью информационной системы «Медик» является защищенный и подтвержденный доступ с мобильных устройств к информации о пациентах.

Информационная система «Медик» спроектирована с использованием клиент-серверной архитектуры. Передача данных в информационной системе «Медик» осуществляется от сервера на операционной системе Linux к клиентам – мобильным устройствам на операционной системе Android.

Передача данных происходит с использованием беспроводной технологии Wi-Fi. При такой архитектуре системы встает вопрос о том, как быть убежденным в том, что тот человек, который заходит с устройства является тем, за кого себя выдает. Вопрос этот возникает так, как в соответствии с пунктом 1 статьи 19 ФЗ «О персональных данных» оператор (в данном случае ГБУЗ СО ТГКБ №5) обязан принимать необходимые правовые, организационные, технические меры или обеспечивать их принятие для защиты персональных данных от неправомерного или случайного доступа к ним, уничтожения, изменения, блокирования, копирования, предоставления, распространения персональных данных, а

также от иных неправомерных действий в отношении персональных данных [18].

Исходя из пункта 1 статьи 19 ФЗ «О персональных данных», сейчас есть потребность в разработке системы аутентификации пользователей для доступа к данным пациентов в информационной системе «Медик», которая будет применяться в ГБУЗ СО ТГКБ № 5.

В соответствии с законом, а в частности со статьей 7 и подпунктом 4 пункта 2 статьи 10 федерального закона № 152-ФЗ «О персональных данных» при аутентификации пользователя в системе необходимо, чтобы доступ к информации о пациентах получал только человек (врач) или круг лиц (врачи), которые имеют доступ к этой информации.

Согласно федеральному закону, который рассмотрен выше, и необходимости обеспечения защищенного и подтвержденного доступа к данным пациентов в ГБУЗ СО ТГКБ №5, требуется с помощью системы аутентификации пользователя организовать подтвержденный доступ к конфиденциальной информации о пациентах.

Информационная система «Медик» это система, которая предназначена для обработки данных о пациентах, которые относятся к специальной категории персональных данных, так как они содержат данные о состоянии здоровья пациентов, а такие данные согласно закону № 152-ФЗ «О персональных данных» относятся к специальной категории персональных данных.

Согласно статье 18.1 и статье 19 федерального закона № 152-ФЗ «О персональных данных» за оператором остается выбор того, как будет реализована защита конфиденциальных данных. Было принято решение реализовать аутентификацию пользователя в соответствии с ГОСТ Р ИСО/МЭК 9594-8-98 «Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации», так как данный ГОСТ рекомендован для использования в системах аутентификации пользователя [9].

Также следует отметить, что в соответствии с ГОСТ Р ИСО/МЭК 9594-8-98 «Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации» существует два вида аутентификации пользователей: простая и строгая. При простой аутентификации требуется лишь логин и пароль пользователя для входа в систему, что недостаточно для уверенности в том, что под данным аккаунтом вошел именно тот человек, который ожидался. Выбор варианта аутентификации пользователя осуществляется в соответствии с соответствующими нормативными правовыми актами.

В статье 7 и подпункте 4 пункта 2 статьи 10 федерального закона № 152-ФЗ «О персональных данных» говорится о том, что доступ к информации о пациентах должны получать только определенные лица. Кроме того, в вышеописанном ГОСТ рекомендуется использовать строгую аутентификацию пользователя в системе, для большей надежности, так как в качестве основы услуг, обеспечивающих защиту, должна использоваться только строгая аутентификация.

Для обеспечения выполнения данных требований будем использовать строгую аутентификацию, которая включает в себя использование блока асимметричного шифрования данных, использование электронной подписи для подписи сертификата пользователя, который ему выдается для взаимодействия с системой. Этот сертификат должен выдаваться уполномоченным по сертификации.

Также для большей защищенности персональных данных, для входа пользователя в систему будем использовать хэширование паролей, чтобы не хранить их на сервере в открытом виде, так как, если данные с сервера будут скомпрометированы, то учетные записи будут взломаны и доступ к информации о пациентах будет предоставлен третьим лицам без согласия пациентов, что нарушает закон «О персональных данных». Также в соответствии с ГОСТ необходимо использовать хэш-функцию для формирования удостоверения личности и маркеров аутентификации.

Исходя из вышеизложенного можно составить требования к аутентификации пользователя в информационной системе «Медик»:

- требуется использовать строгую аутентификацию пользователей;
- разработанная аутентификация должна соответствовать законодательным требованиям;
- необходимо использовать ГОСТ Р 34.11-2012 для хэширования паролей пользователей.

На данный момент ГОСТ Р 34.11-2012 является действующим стандартом хэширования, а также ГОСТ Р ИСО/МЭК 9594-8-98 является действующим стандартом аутентификации пользователей, поэтому мы будем их использовать при реализации аутентификации пользователя на стороне сервера в информационной системе «Медик».

1.2 Анализ существующих технологий аутентификации

В настоящее время введен ГОСТ Р ИСО/МЭК 9594-8-98 на реализацию аутентификации пользователя. Он введен впервые, поэтому у него нет официальных альтернатив для рассмотрения [9]. Но существует ряд походов, которые применяются для аутентификации пользователя параллельно этому ГОСТ, если не требуется соблюдение такового [3], [4], [11], [13].

Рассмотрим кратко данные подходы [15].

- аутентификация пользователя происходит по логину и паролю без применения хэширования паролей;
- аутентификация пользователя производится по логину и паролю с применением хэширования паролей;
- аутентификация пользователя происходит по логину и паролю с применением «соленого» хэширования;
- аутентификация пользователя происходит с применением сертификации ключа при ассиметричном шифровании.

Сравним способы аутентификации пользователя, которые можно использовать в информационной системе «Медик» и представим результаты сравнения в таблице 1 [20].

Таблица 1 – Сравнение способов аутентификации

Способ аутентификации	Наличие хэширования пароля	Наличие «соленого» хэширования	Наличие применения сертификата пользователя
По логину и паролю без хэширования паролей	Нет	Нет	Нет
По логину и паролю с хэшированием паролей	Есть	Нет	Нет
По логину и паролю с применением «соленого» хэширования	Есть	Есть	Нет
По логину и паролю с применением сертификации ключа	Есть	Нет	Есть

Практически все перечисленные подходы будут относиться к простой аутентификации пользователя в соответствии с ГОСТ, что не подходит для реализации аутентификации в соответствии с поставленными требованиями.

Исходя из сравнения существующих технологий аутентификации пользователя, можно прийти к выводу, что требуется разработать собственную реализацию алгоритма аутентификации пользователя в соответствии с законом «О персональных данных» и ГОСТ для аутентификации пользователей в информационных системах. Такой вывод был сделан, так как в некоторых решениях нет хэширования вообще, где-то нет только «соленого» хэширования, и почти нигде нет наличия сертификата пользователя, который нужен в соответствии с ГОСТ.

Исходя из вышеперечисленного можно сказать, что существующие подходы к аутентификации пользователя не удовлетворяют всем поставленным требованиям в ГОСТ, или соответствуют частично действующему ГОСТ, и, следовательно, не рекомендуются к использованию в информационной системе «Медик», так как в ней будут обрабатываться конфиденциальные данные пациентов, доступ к которым должен быть предоставлен только тем лицам, которые имеют право доступа к личным данным пациентов, в соответствии со статьей 7 и подпунктом 4 пункта 2 статьи 10 федерального закона № 152-ФЗ «О персональных данных». Поэтому необходимо реализовать собственный алгоритм аутентификации пользователя в системе для информационной системы «Медик».

1.3 Постановка задачи на разработку системы аутентификации пользователя для информационной системы «Медик»

После проведения анализа требований аутентификации пользователя в информационной системе «Медик», а также после анализа существующих технологий аутентификации, сделаем постановку задачи на разработку системы аутентификации пользователя для информационной системы «Медик».

Постановка задачи на разработку системы аутентификации пользователя была произведена, основываясь на требованиях законодательства Российской Федерации, а в частности, основываясь на таких нормативных правовых актах, как:

- Федеральный закон № 152-ФЗ «О персональных данных»;
- ГОСТ Р ИСО/МЭК 9594-8-98 «Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации»;

– ГОСТ Р 34.11-2012 «Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования (с Поправкой)».

Основываясь на требованиях, которые установлены настоящим законодательством, а также основываясь на функциях, которые необходимы ГБУЗ СО ТГКБ №5, были сформулированы следующие требования к системе аутентификации пользователя в информационной системе «Медик»:

– требуется, чтобы при аутентификации пользователя можно было быть убежденным, что это тот самый человек, который ожидается, чтобы конфиденциальность данных не была нарушена;
– алгоритм аутентификации должен соответствовать требованиям ГОСТ и требованиям закона «О персональных данных»;
– требуется использовать алгоритм хэширования, который описан в ГОСТ Р 34.11-2012 «Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования (с Поправкой)», чтобы уменьшить возможность взлома аккаунтов пользователей и доступа третьих лиц к конфиденциальной информации.

2 Проектирование и разработка алгоритма аутентификации пользователя в информационной системе «Медик»

2.1 Проектирование технологии аутентификации пользователя для информационной системы «Медик»

После анализа требований к аутентификации пользователя в информационной системе «Медик», а также постановки задачи на разработку системы аутентификации пользователя для информационной системы «Медик», займемся проектированием технологии аутентификации пользователя для информационной системы «Медик».

В государственном стандарте для аутентификации пользователя даются рекомендации к реализации аутентификации пользователя. Будем придерживаться данных рекомендаций.

Исходя из требований было принято решение спроектировать собственный алгоритм аутентификации пользователя, который соответствует закону «О персональных данных» и ГОСТ Р ИСО/МЭК 9594-8-98 «Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации».

Данный алгоритм описан блок-схемой, которая продемонстрирована на рисунке 1.

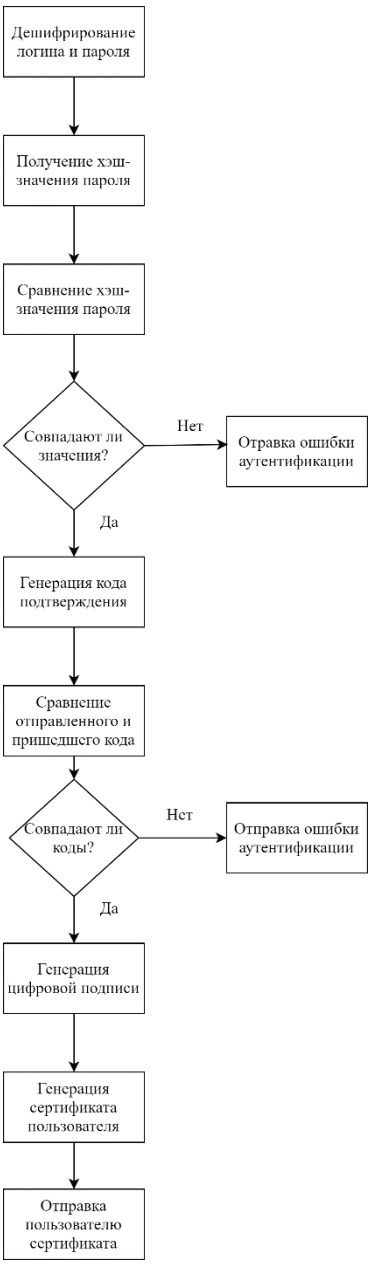


Рисунок 1 – Общая схема работы алгоритма аутентификации пользователя

Распишем некоторые этапы алгоритма.

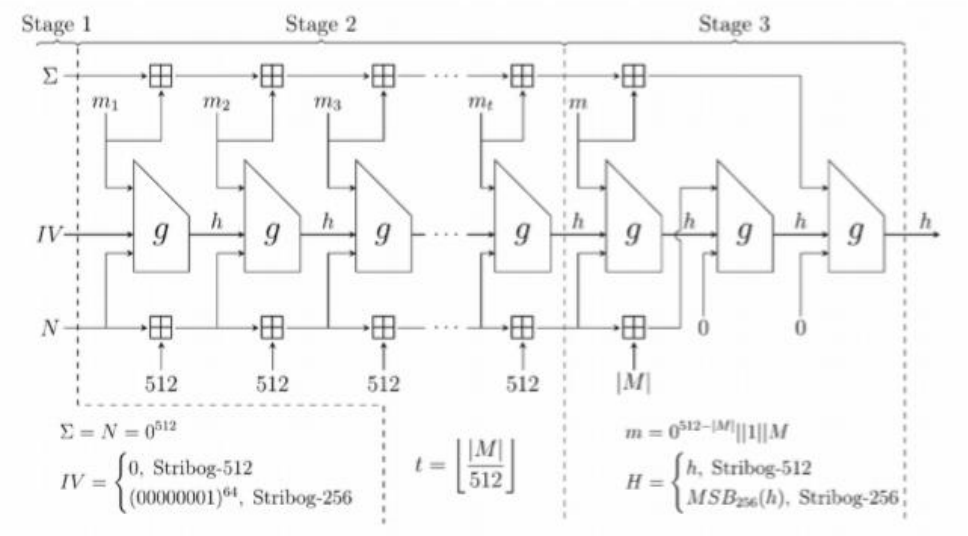
На вход в модуль получения хэш-значения пароля приходит дешифрированный пароль из модуля дешифрирования. Модуль получения хэш-суммы от пришедшего дешифрированного пароля должен производить вычисление хэш-суммы пришедшего пароля с применением алгоритма хэширования «Стрибог» (ГОСТ Р 34.11-2012) [8].

Для большей защищенности пользовательских аккаунтов от несанкционированного доступа будем использовать «соленое» хэширование

паролей [6]. Этот метод хэширования паролей предотвратит подбор паролей с помощью таблиц поиска, обратных таблиц поиска и радужных таблиц, потому что все они становятся неэффективными, так как злоумышленник не узнает заранее, какая будет «соль», и поэтому не сможет вычислить заранее эти таблицы [2].

Для каждого пользователя нужно генерировать свою «соль», будь то создание новой учетной записи или смена пароля в уже существующей учетной записи. «Соль» нужно генерировать с помощью криптографически стойкого генератора псевдослучайных чисел [7].

Рассмотрим более подробно применяемый алгоритм хэширования. Для хэширования паролей будем использовать алгоритм «Стрибог», блок-схема которого приведена на рисунке 2 [17].



stage1 – этап инициализации, stage2 – этап итерационного применения функции сжатия, stage3 – завершающий этап, IV – вектор инициализации, ∑ - контрольная сумма, N – номер итерации, MSB_{256} – старшие 256 бит

Рисунок 2 – Алгоритм вычисления хэш-функции по стандарту ГОСТ Р 34.11-2012

На первом шаге алгоритма необходимо присвоить начальные значения текущим величинам. $h := IV; N := 0^{512} \in V_{512}; \sum := 0^{512} \in V_{512}$.

Далее нам необходимо проверить условие $|M| < 512$. Если условие выполняется, то переходим к следующему шагу. Если это условие не выполняется, то необходимо произвести следующие вычисления. Пункт первый. Вычислить подвектор $m \in V_{512}$ сообщения M: $M = M' \parallel m$.

Далее нам необходимо будет вычислить новые значения для $h; N; \sum$.

После чего необходимо задать значение $M \coloneqq M'$. Далее опять проверяем условие из второго шага, если оно выполняется, то переходим к третьему шагу.

На третьем шаге необходимо вычислить следующие значения $m \coloneqq 0^{511-|M|} \parallel 1 \parallel M$, $h \coloneqq g_N(h, m)$, $N \coloneqq Vec_{512}(t_{512}(N) \boxplus |M|)$, $\sum \coloneqq Vec_{512}\left(t_{512}(\sum) \boxplus \in t_{512}(m)\right)$, $h \coloneqq g_0(h, N)$.

На выходе получаем значение хэш-функции.

За операцию нахождения хэш-значения будет отвечать класс нахождения хэш-значения пароля, который будет находиться в maven модуле проекта – модуле impl. Этот класс будет обращаться к репозиторию для получения значения «соли» для данного пользователя. Репозиторий будет находиться в maven модуле проекта – модуле impl. После получения ответа от базы данных, класс, который отвечает за получения хэш-значения пароля будет вычислять это значение с использованием алгоритма хэширования «Стрибог», применяя при этом «соленое» хэширование.

Следующим шагом алгоритма необходимо сравнить полученное хэш-значение, с тем, что хранится в базе данных. Для этого задействуется модуль сравнения хэш-значения пришедшего пароля с хэш-значением пароля, которое хранится в базе данных для данного логина.

За эту операцию будет отвечать класс сравнения хэш-значений. Этот класс будет находиться в maven модуле проекта – модуле impl. Для того, чтобы сравнить значение пришедшего хэш-значения с тем, что хранится в базе данных, данный модуль обращается к репозиторию для получения хэш-значения пароля для данного пользователя, которое хранится в базе данных. После получения ответа от базы данных, этот модуль сравнивает хэш-

значение пришедшего пароля со значением из базы данных. Если значения совпадают, то работа алгоритма продолжается, иначе пользователю посылается сообщение, что логин или пароль введен не верно.

Следующим шагом алгоритма является генерация кода подтверждения и его отправка пользователю на корпоративную почту по защищенному соединению [1]. Отправление кода подтверждения на корпоративную почту пользователя будет происходить по защищенному http соединению с помощью сервлетов и реализаций их методов [19]. За эту часть будет отвечать класс, который будет находиться в maven модуле проекта – модуле api. Для того, чтобы сгенерировать код подтверждения, будет использоваться генератор UUID randomUUID, который находится в пакете java.uti в классе UUID. Для отправки кода подтверждения будет использоваться сервлеты.

Следующим шагом алгоритма является сравнение пришедшего кода подтверждения с отправленным пользователю. Для получения запроса также будут использоваться сервлеты. Сравнение будет осуществляться с помощью операции «исключающее или», потому что тогда будет стопроцентная гарантия, что числа действительно равны, так как результат операции XOR всегда выдает ноль только в том случае, если числа одинаковые [5]. Эту часть работы будет выполнять класс, который будет находиться в maven модуле проекта – модуле api.

Далее происходит генерация цифровой подписи, с помощью которой подписываются маркеры в справочнике. Генерация осуществляется в классе Session пакета util. Цифровой подписью подписывается информация об пользователе, которая записана в справочнике пользователя. Информация подписывается путем добавления к ней зашифрованной сводки информации. Эта сводка генерируется с использованием однонаправленной хэш-функции, а шифрование выполняется с использованием личного ключа подписывающего лица. Блок-схема представлена на рисунке 3.

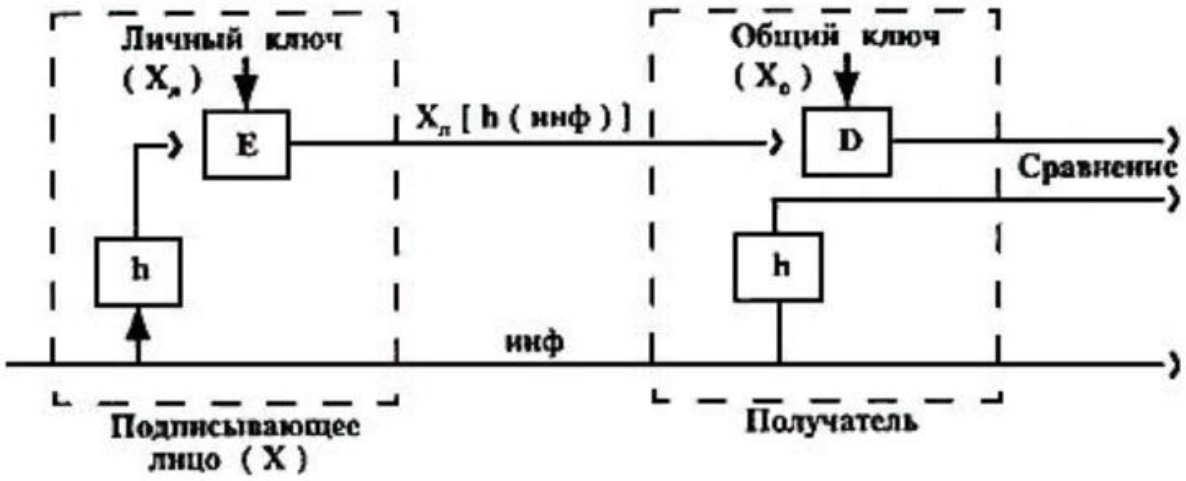


Рисунок 3 – Алгоритм работы цифровой подписи

Данный алгоритм включает в себя генерацию и отправку сертификата, в который входит также общий ключ, пользователя. Это необходимо, чтобы можно было выполнить аутентификацию, и сервер мог быть убежден, что данный пользователь является тем, что прошел авторизацию в системе и мог ему предоставлять доступ к необходимым данным. Общий ключ пользователя генерируется на стороне сервера и передается пользователю вместе с сертификатом, также, как и личный ключ пользователя. Эти ключи необходимы для шифрования и дешифрирования данных, которыми обмениваются клиент и сервер. Тут используется ассимметричное шифрование. То есть, если информация была зашифрована пользователем с применением секретного ключа, то серверу для расшифровки необходим общий ключ, и наоборот. Для обеспечения подлинности пользователя сертификат обновляется каждые 24 часа.

После проектирования алгоритма, можно перейти к разработке архитектуры разрабатываемого модуля.

2.2 Разработка архитектуры модуля аутентификации пользователя на стороне сервера в соответствии с законом «О персональных данных»

Для дальнейшей работы необходимо определить какие модули, пакеты и классы необходимо разработать для реализации аутентификации пользователя. Рассмотрим последовательность выполняемых операций для решения поставленной задачи.

За операцию нахождения хэш-значения пароля будет отвечать класс, который будет находиться в maven модуле проекта – модуле impl. Этот класс будет обращаться к репозиторию для получения значения «соли» для данного пользователя. Репозиторий будет находиться в maven модуле проекта – модуле impl. В репозитории будет находиться класс, взаимодействующий с базой данных по средством JPQL запросов. После получения ответа от базы данных, класс, который отвечает за получения хэш-значения пароля будет вычислять это значение с использованием алгоритма хэширования «Стрибог», применяя при этом «соленое» хэширование.

За операцию сравнение полученного хэш-значения пароля со значением, хранящимся в базе данных, будет отвечать класс сравнения хэш-значений. Этот класс будет находиться в maven модуле проекта – модуле impl. Этот класс будет взаимодействовать с репозиторием для получения хэш-значения пароля из базы данных postgeSQL. После получения ответа от базы данных, этот модуль сравнивает хэш-значение пришедшего пароля со значением из базы данных. Если хэш-занчения не совпадают, то пользователю отправляется ошибка аутентификации, иначе работа продолжается и начинается генерация кода подтверждения.

За операцию генерации случайного значения, которое будет отправлено пользователю для подтверждения его подлинности, будет отвечать класс, который будет находиться в maven модуле проекта – модуле api. Для отправки кода подтверждения будет использоваться сервлеты.

За операцию сравнения отправленного пользователю и пришедшего от пользователя кода подтверждения, будет отвечать класс, который будет находиться в maven модуле проекта – модуле api. Пришедший на вход по защищенному соединению код подтверждения будет сравниваться с отправленным с помощью операции «исключающее или». Если отправленный пользователю код совпадает с полученным от пользователя кодом, то работа алгоритма продолжается, и начинается генерация цифровой подписи, иначе пользователю отправляется сообщение, что введен не верный код.

За операцию генерации цифровой подписи будет отвечать класс, который находится в maven модуле проекта – модуле api. Он будет взаимодействовать с классом, отвечающим за генерацию сертификата пользователя, который также будет находиться в модуле api. Чтобы можно было подтвердить сертификат подписью.

Чтобы сгенерировать сертификат пользователя будет использоваться класс, который отвечает за генерацию сертификата, в который входят ключи общего пользования, и некоторая другая информация, к который применено шифрование личным ключом уполномоченного по сертификации, в нашем случае уполномоченным по сертификации будет сервер. В данный класс также будет входить метод генерации личного ключа уполномоченного по сертификации. После генерации сертификата, пользователю отправляется сертификат и предоставляется вход в систему. Это класс будет находиться в модуле api.

После того, как мы определились с тем, какие модули нам необходимы для решения задачи аутентификации пользователя, приведем диаграмму модулей проектируемого программного модуля. Данная диаграмма представлена на рисунке 4 в виде UML-диаграммы. Эта диаграмма показывает взаимодействие модулей программы.

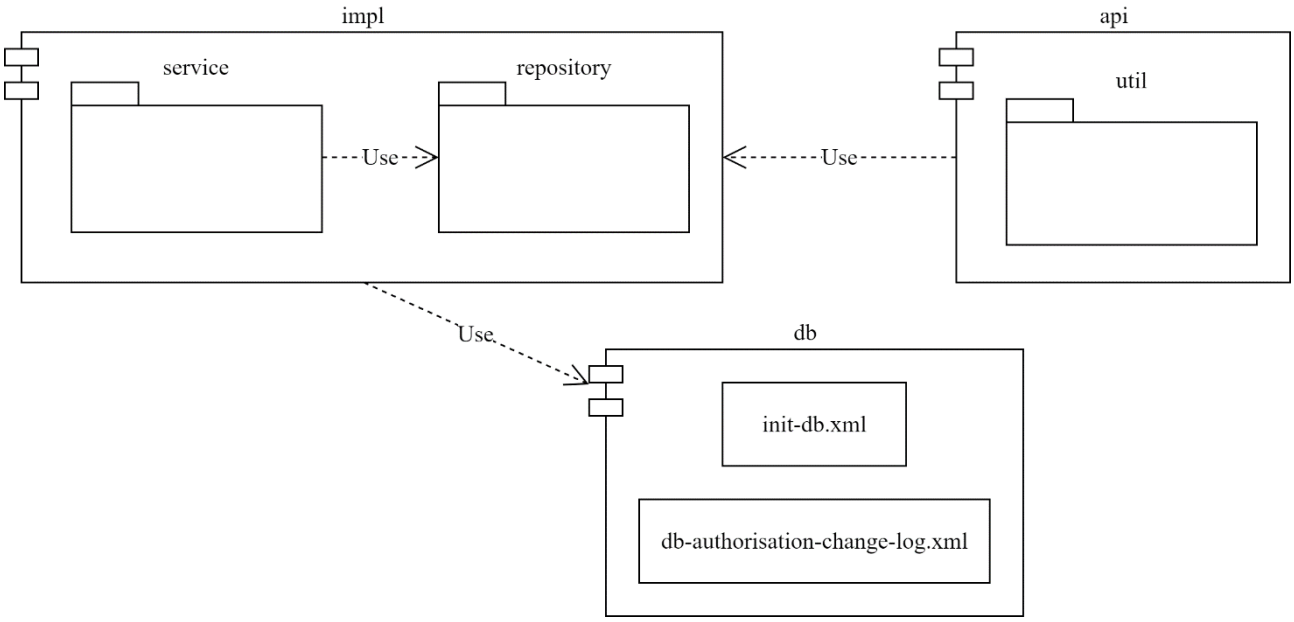


Рисунок 4 – Диаграмма классов модульный уровень

Детализируем диаграмму классов на уровне модулей до пакетного уровня взаимодействия, и так углубляясь дойдем до уровня взаимодействия классов внутри пакетов. Пакетный уровень взаимодействия представлен на рисунке 5.

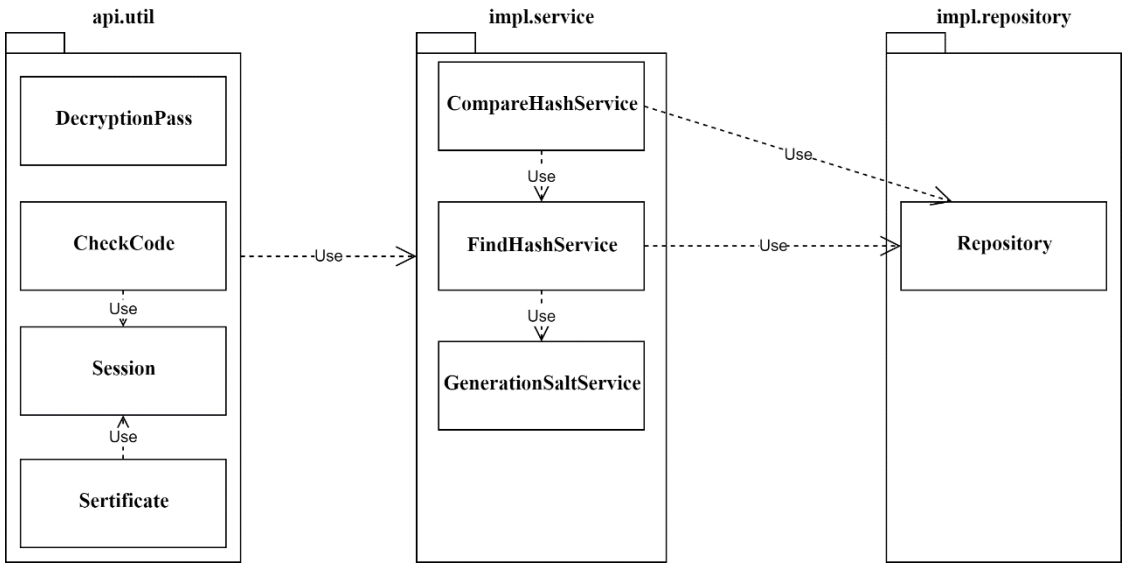


Рисунок 5 – Диаграмма классов пакетный уровень

На рисунке 5 мы можем увидеть, что для выполнения поставленных задач есть модули, которые общаются между собой. Для взаимодействия нашей программы с внешним миром используется модуль api, где происходит дешифрирование полученных данных, далее модуль обращается к модулю impl, где происходит нахождения хэша пришедшего пароля, генерация «соли», если аккаунт с таким логином и паролем только создается, или если пользователь меняет свой пароль. Также в модуле impl происходит сравнение хэш-значения пароля, который пришел с хэш значением, которое хранится в базе данных для данного пользователя. Работа с базой данных осуществляется через класс Repository. Если хэш-значение пришедшего пароля совпадает с его значением в базе данных, то модуль api генерирует и отправляет одноразовый код подтверждения пользователю, иначе пользователю посылается сообщение, что логин или пароль не верны. Далее, когда пользователь вводит код подтверждения, то модуль api производит сравнение отправленного и полученного от пользователя кода подтверждения. При совпадении этих значений, генерируется цифровая подпись, потом генерируется и отправляется пользователю сертификат и одновременно устанавливается новая сессия для пользователя, иначе отправляется сообщение, что код подтверждения введен не верно.

Далее детализируем нашу UML-диаграмму до уровня пакета util, чтобы увидеть взаимодействие классов внутри пакета. Данная диаграмма представлена на рисунке 6. Программный код пакета util находится в приложении А.

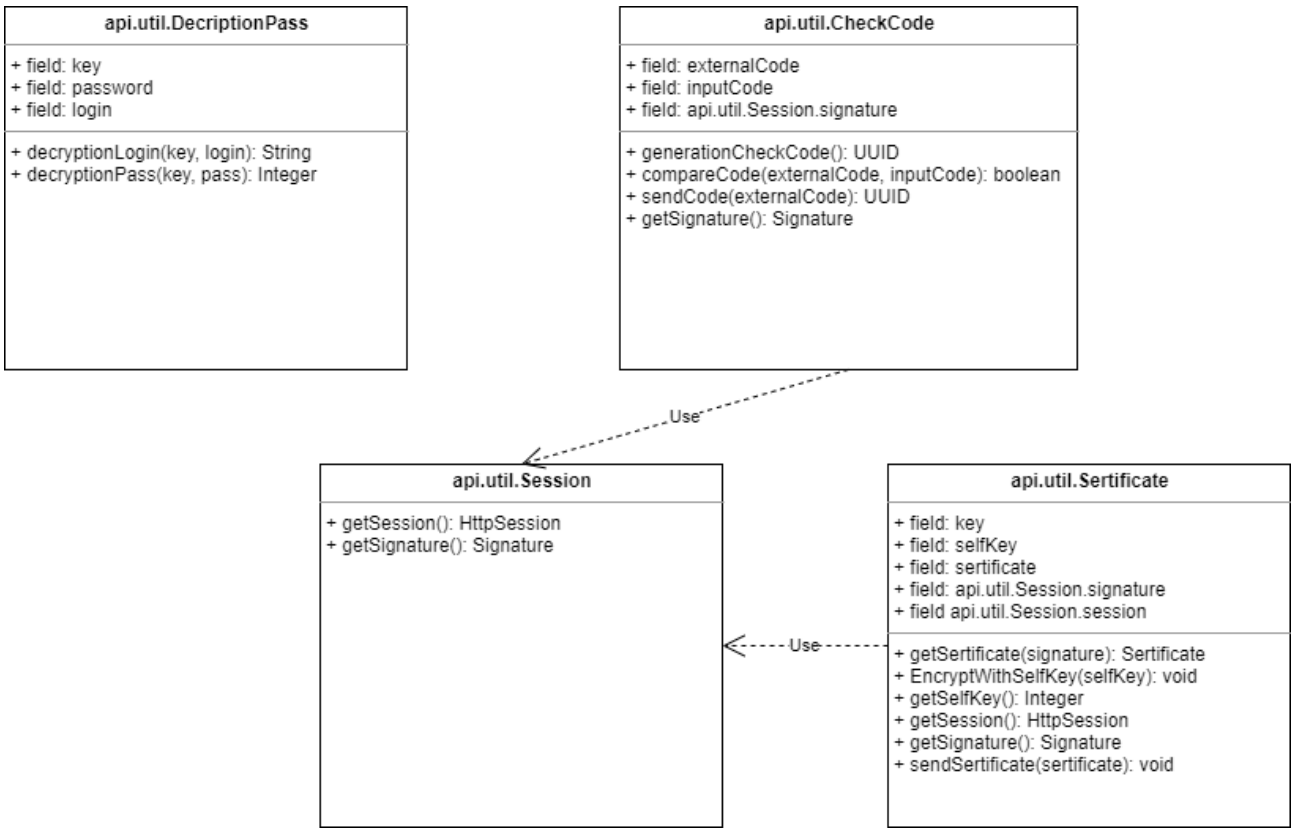


Рисунок 6 – Диаграмма классов пакета util

Класс DecriptionPass отравляет в модуль impl дешифрированный логин и пароль. Метод generationCheckCode класса CheckCode используется классом CompareHashService. Если метод generationCheckCode был вызван, то пользователю отправляется сгенерированный код подтверждения. Далее происходит выполнение сравнения, отправленного и полученного кодов подтверждения, если они совпадают, то вызывается метод getSignature класса Session и вызывается метод sendSertificate, иначе отправляется сообщение, что введен не верный код.

Далее рассмотрим UML-диаграмму пакета service, чтобы увидеть взаимодействие классов в пакете. Данная диаграмма представлена на рисунке 7. Полный программный код пакета service находится в приложении А.

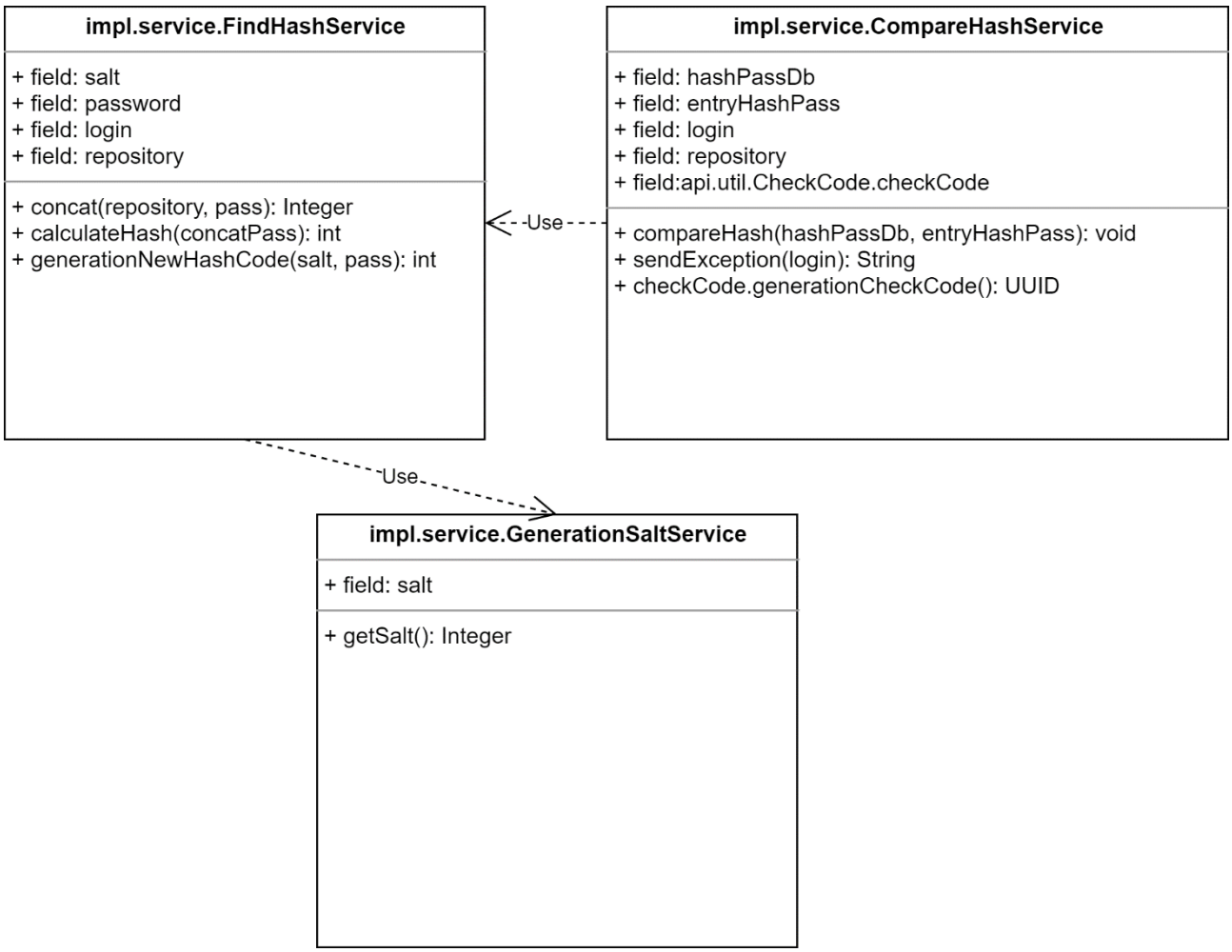
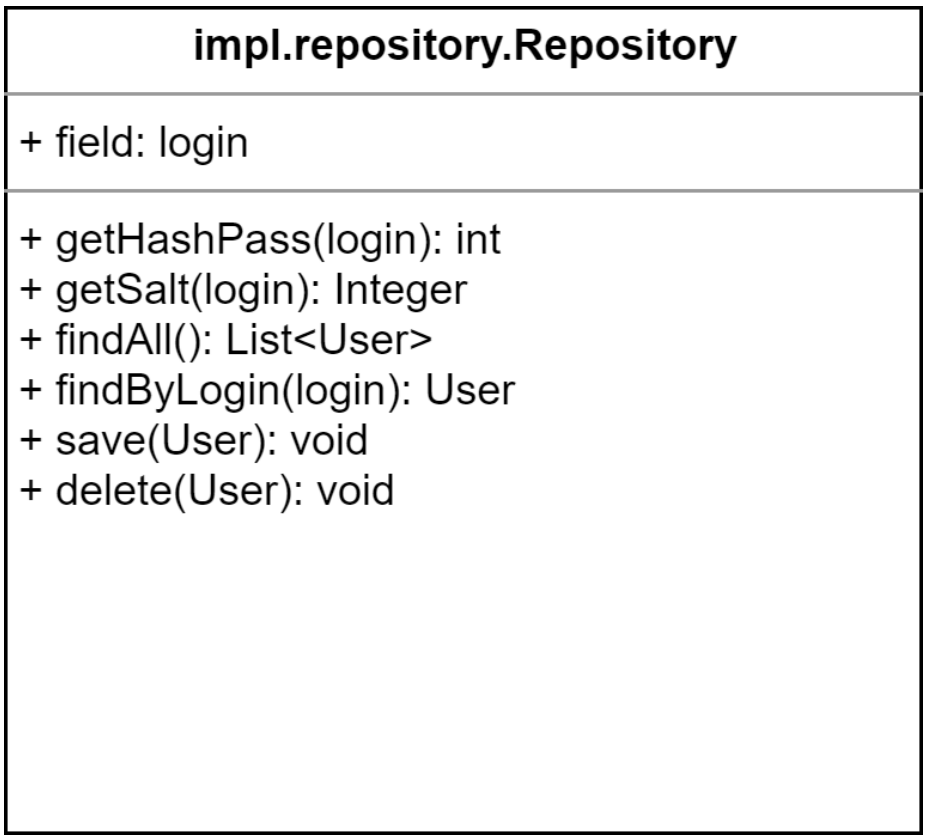


Рисунок 7 – Диаграмма классов пакета service

Первым делом в этом модуле вызывается класс FindHashService, в котором происходит вычисление хэш-значения пришедшего пароля. Далее вызывается класс CompareHashService, в этом классе происходит сравнение хэш-кода пришедшего пароля с хэш-кодом в базе данных, если они не совпадают, то вызывается метод sendException, который отправляет сообщение о том, что логин или пароль не верны, если совпадают, то вызывается метод generationCheckCode из класса CheckCode. Если происходит регистрация нового пользователя или смена пароля уже существующего пользователя, то в классе FindHashService происходит вызов метода generationNewHashCode и метода getSalt из класса GenerationSaltService.

И рассмотрим последний пакет в реализуемом программном модуле – диаграмму классов пакета repository. Диаграмма классов пакета repository представлена на рисунке 8. Полный код пакета repository находится в



приложении А.

Рисунок 8 – Диаграмма классов пакета repository

Класс Repository и его методы вызываются классами FindHashService и CompareHashService. В классе Repository происходит взаимодействие с базой данных посредством JPQL запросов. Методы этого класса используются в других классах для реализации бизнес-логики.

Для взаимосвязи модуля impl с базой данных используется конфигурационный файл hibernate.properties, для настройки hibernate используется файл persistence.xml, а для описания базы данных используются скрипты миграции, которые находятся в файлах init-db.xml и db-authorisation-change-log.xml.

После разработки архитектуры проектируемого программного модуля, оценим возможность внедрения проектируемого программного модуля аутентификации пользователя на аналогичные объекты управления.

Данный модуль будет довольно сложно внедрить на какие-либо другие объекты, так как, скорее всего, придется исправлять некоторые части программы, в которых используются специфичные форматы данных.

Проектируемый программный модуль должен быть достаточно гибким и настраиваемым на различные окружения, так как используется язык программирования Java. Приложение, написанное на этом языке программирования, будет запускаться на любой платформе, где есть JVM и JRE. Для настройки приложения будет достаточно его развернуть на сервере Tomcat и оно будет готово к работе.

2.3 Реализация алгоритма аутентификации пользователя в соответствии с требованиями закона «О персональных данных»

После проектирования алгоритма и разработки архитектуры программного модуля, можно перейти к непосредственной реализации. Для этого был использован язык программирования Java.

Для реализации аутентификации пользователя необходимо получить хэш-значение пароля. Реализация данного метода представлена на рисунке 9.

```
@Override
    public byte[] getHash(File file, boolean outputMode)
    {
        init(outputMode);

        byte[] buffer = new byte[64];
        try(RandomAccessFile IStream = new RandomAccessFile(file, "r");)
        {
            long length = IStream.length();
            for(;length >= 64; length -=64)
            {
                IStream.seek(length - 64);
                IStream.read(buffer, 0, 64);

                hashPart(buffer);
            }

            IStream.seek(0);
            buffer = new byte[(int)length];
            IStream.read(buffer, 0, (int)length);
            hashPart(buffer);
        }
        catch(IOException exception)
        {
            System.err.println(exception.toString());
        }

        if(outputMode)
            return h;
        return Arrays.copyOf(h, 32);
    }
```

Рисунок 9 – Хэш-функция «Стрибог»

Также в соответствии с алгоритмом необходимо реализовать генерацию цифровой подписи. Метод, реализующий заполнение электронной подписи представлен на рисунке 10.

```
public AbstractSignatureParameters createParameter(DSSPrivateKeyEntry signer) {
        PAdESSignatureParameters parameters = new PAdESSignatureParameters();

        parameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_B);
        parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);
        parameters.setSigningCertificate(signer.getCertificate());
        boolean useField = selectSignatureField(parameters);
        if (prop.useVisibleSign()) {
            CertificateToken ct = signer.getCertificate();
            String humanReadableSigner = DSSASN1Utils.getHumanReadableName(ct);
            humanReadableSigner = humanReadableSigner + "\n" + getDateAndTimeString();
            parameters.setSignatureImageParameters(addVisibleSignatureParameters(humanReadableSigner, useField));
        }

        return parameters;
    }
```

Рисунок 10 – Заполнение электронной подписи

Также в соответствии с алгоритмом необходимо реализовать создание сертификата пользователя. Метод, который создает сертификат пользователя представлен на рисунке 11.

```
@Override
public CsrWithPrivateKey generateResponse(final DistinguishedName dn) {
    final KeyPair pair = KeysUtil.generateKeyPair();
    try {
        final PrivateKey privateKey = pair.getPrivate();
        final PublicKey publicKey = pair.getPublic();
        final X500Name x500Name = dn.getX500Name();
        final ContentSigner signGen = new JcaContentSignerBuilder(SIGNATURE_ALGORITHM)
            .build(privateKey);
        final PKCS10CertificationResponseBuilder builder = new JcaPKCS10CertificationResponseBuilder(
            x500Name, publicKey);
        final PKCS10CertificationResponse csr = builder.build(signGen);
        return new CsrWithPrivateKeyImpl(csr, privateKey);
    } catch (final OperatorCreationException e) {
        throw new CaException(e);
    }
}
```

Рисунок 11 – Создание сертификата пользователя

После проектирования и реализации архитектуры программного модуля необходимо перейти к разработке диаграммы развертывания.

2.4 Разработка диаграммы развертывания системы аутентификации пользователя на стороне сервера

Составим диаграмму развертывания для того, чтобы понимать, как будет проецироваться разработанный программный модуль на аппаратную архитектуру. Составленная диаграмма развертывания представлена на рисунке 12.

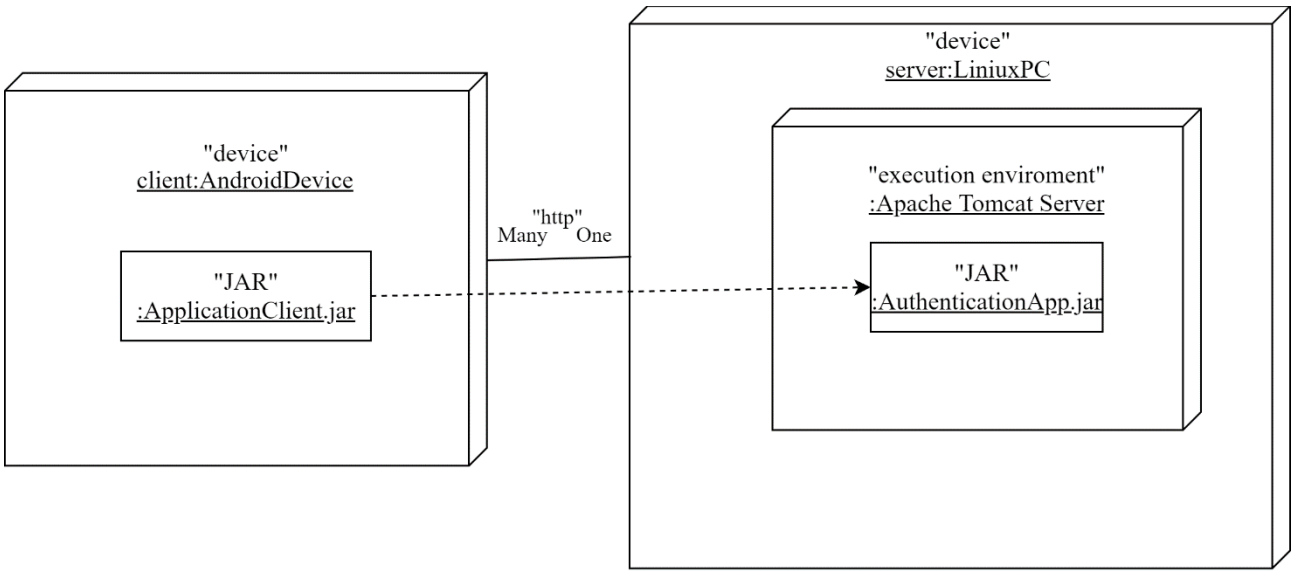


Рисунок 12 – Диаграмма развертывания

На данной диаграмме развертывания представлено, как будет проецироваться программная архитектура на аппаратную архитектуру. С помощью сборщика проектов Maven будет собран jar файл с приложением. Этот jar файл будет развернут на сервере Apache Tomcat Server. Физически данный сервер будет представлять машину на ОС Linux, на которой запущен Tomcat. Разработанному программному модулю будет посылать запросы клиентское приложение, установленное на какой-либо гаджет с ОС Android, посредством http запросов.

После проектирования, разработки программного модуля, а также разработки его диаграммы развертывания, можно перейти к следующему шагу – тестированию программного решения.

3 Оценка эффективности работы алгоритма аутентификации пользователя в соответствии с требованиями закона «О персональных данных»

3.1 Проведение статистических исследований возможности несанкционированного доступа к персональным данным пациентов

После реализации программного модуля аутентификации пользователя необходимо провести тестирование основных компонентов программного модуля. Чтобы оценить риски и исследовать надежность основных этапов аутентификации, нужно создать математическую модель.

В процессе моделирования надо учитывать роль в процедурах подготовки и проведения аутентификации пользователей информационной системы «Медик». Аутентификация включает в себя две основные составляющие – подтверждение подлинности предъявляемой уникальной информации, которой владеет только пользователь, и проверки принадлежности пользователю этой информации.

Любую аутентификацию можно рассматривать, как цепь последовательных событий [16]. Такая процедура может быть однократной, например регистрация нового пользователя, либо может быть длительной по времени, например хранение данных, либо может быть часто повторяющейся, к таким процедурам можно отнести: предъявление аутентификатора, протоколы обмена данными между клиентом и сервером, валидация данных, авторизация. Также стоит понимать, что при большом числе зарегистрированных пользователей, например более пятисот, система аутентификации пользователя должна подчиняться законам систем массового обслуживания (СМО) [16]. Поэтому надо предусмотреть возможность исследования поведения системы, при случайном потоке заявок

на аутентификацию, этот поток зависти от времени. Для корпоративных систем пик активности запросов приходится на утро, а в системах общего пользования максимальная нагрузка носит случайный характер. Общепринятого подхода к исследованию безопасности и надежности систем аутентификации пока что не выработано [16].

Для более простого моделирования аутентификации разделим этот процесс на однородные составляющие по функциональным и вероятностно-статистическим характеристикам. Предпосылкой для такого разделения является тот факт, что разные составляющие имеют разные, сильно отличающиеся временные характеристики (характеристики по времени). Так регистрация пользователя происходит только один раз и занимает относительно небольшой промежуток времени. Тогда, как хранение данных о пользователе – занимает много времени, поэтому к ней можно применить вероятностные и статистические методы. Оставшиеся процедуры, такие как проверка подлинности, валидация данных, связаны с временем выполнения процедур, а также они многократно повторяются.

После всех рассуждений, сформулируем такие составляющие:

- процедура регистрации (стационарный процесс), потому что она не связана со временем;
- процедура хранения, которая связна со временем (при моделировании можно применять распределение Пуассона);
- протоколы обмена информацией происходят очень быстро, порядка секунды. В этой составляющей необходимо учитывать поломки аппаратного или программного компонентов, ошибки пользователей и атаки (при моделировании можно применять экспоненциальное распределение);
- при валидации, которая тоже происходит по времени порядка секунды, вероятность поломки для корпоративных систем мала;
- процесс принятия решения о прохождении аутентификации также производится порядка секунды. Нужно учитывать вероятность

отказа (опасного отказа – вероятность ошибочного предоставления доступа в систему).

После сохранения секрета и электронного удостоверения, происходит процедура предъявления удостоверения, чтобы можно было запустить протокол аутентификации.

Под определением надежности и безопасности системы аутентификации будем понимать функциональную надежность и функциональную безопасность. Функциональная надежность – выполнение системой предусмотренных функций с приемлемым уровнем безошибочности во время эксплуатации. А функциональная безопасность – это способность системы выполнять предусмотренные задачи с заданным уровнем доступности, целостности и конфиденциальности.

Сформулируем вероятностную модель системы аутентификации пользователей и оценим ее стационарные характеристики. Примем некоторые допущения:

- процесс регистрации пользователя будем считать процессом однократного срабатывания;
- объединим блоки хранения данных и протокол аутентификации в один блок. Этот блок многократного хранения и предъявления аутентификатора представим, как пуассоновский, также его можно отнести к классу марковских процессов;
- блок валидации объединим с блоком принятия решений.

Сформулируем критерии отказа и опасного отказа. В модуле регистрации отказ – недопуск легального пользователя, а опасный отказ – доступ нелегального пользователя. Отказ в модуле подтверждения подлинности и в модуле принятия решения об авторизации пользователя не влияют на вероятностную модель работы системы в целом. Опасный отказ работы модуля принятия решения – прохождение аутентификации злоумышленником под видом легального пользователя.

В качестве критериев функциональных отказов можно принимать ошибки, которые не должны превышать определенного порога, который устанавливает границы, после которых система перестает выполнять заданные функции.

Сумма вероятностей выхода из каждого состояния – полная группа событий $\sum_{i=1}^{n} P_i = 1$, где n – число состояний системы.

После рассмотрения теоретической части тестирования, перейдем к тестированию разработанной системы.

3.2 Проверка надежности процесса аутентификации пользователя в информационной системе «Медик»

После описания метода статистических исследований, перейдем к статистическому исследованию надежности разработанной системы аутентификации для информационной системы «Медик».

Если в системе количество пользователей превышает определенное количество, то система аутентификации может рассматриваться, как СМО с интенсивностью входящего пуассоновского потока λ. Интенсивность обработки заявок обозначим μ, тогда определяющим работу системы параметром будет $\rho = \frac{\lambda}{\mu}$. При условии, что $\rho < 1$ – процесс может считаться стационарным, то переход системы из одного состояния в другое будем моделировать при помощи цепей Маркова [16]. Распишем состояния системы: 1 – отправлен запрос регистрации на сервер; 2 – данные имеются для проверки подлинности пользователя; 3 – имеются данные для проверки валидности; 4 – имеются все данные для процесса авторизации. p_{ij} – вероятность перехода из состояния i в состояние j.

Вектор вероятности P можно определить следующим образом, как показано в формуле (1).

$$\begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{Bmatrix} = (P_1, P_2, P_3, P_4, P_5, P_6) \times \begin{pmatrix} 0 & p_{12} & 0 & 0 & p_{15} & p_{16} \\ 0 & 0 & p_{23} & 0 & p_{25} & 0 \\ 0 & 0 & 0 & p_{34} & 0 & p_{36} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

Из формулы (1) можно найти выражения стационарных вероятностей P_i того, что система находится в состоянии i $(i = 1, 2, 3, 4, 5, 6)$:

$$P_2 = P_1 \times p_{12}, \quad (2)$$

$$P_3 = P_2 \times p_{23}, \quad (3)$$

$$P_4 = P_3 \times p_{34}, \quad (4)$$

$$P_5 = P_1 \times p_{15} + P_2 \times p_{25}, \quad (5)$$

$$P_6 = P_1 \times p_{16} + P_3 \times p_{36}. \quad (6)$$

Все вероятности можно выразить через P_1. Из условия нормировки $\sum_{i=1}^{6} p_i = 1$ получим $P_1 \times (p_{12} + p_{12}p_{23}p_{34} + p_{15} + p_{12}p_{25} + p_{16} + + p_{12}p_{23}p_{36}) = 1$. Откуда получим (7) – (12).

$$P_1 = \frac{1}{p_{12} \times (1 + p_{12} + p_{23}p_{34} + p_{25} + p_{12}p_{23}p_{16}) + p_{15}} = \frac{1}{A}, \quad (7)$$

$$P_2 = \frac{p_{12}}{A}, \quad (8)$$

$$P_3 = \frac{p_{12}p_{12}}{A}, \quad (9)$$

$$P_4 = \frac{p_{12}p_{23}}{A}, \quad (10)$$

$$P_5 = \frac{p_{15} + p_{12}p_{25}}{A}, \quad (11)$$

$$P_6 = \frac{p_{16} + p_{12}p_{23}p_{36}}{A}. \quad (12)$$

Величина вероятностей переходов p_{12}, p_{23}, p_{34} может меняться в диапазоне от 0,8 до 1, а величина вероятностей переходов p_{15}, p_{36} в среднем на два порядка ниже.

В реальности значения вероятности P_i лежит в пределах от 0,8 до 1. Это значит, что ошибки, которые случаются при вводе данных аутентификации, а также сбои программного и аппаратного обеспечения могут привести к отказам или задержкам по времени, а также могут привести к опасным отказам с малой вероятностью [16].

Поскольку тестовые данные, которые были получены в ходе выполнения данной работы совпали с теоретическими данными, которые были получены в работе «Концепция моделирования процессов аутентификации» [16], то можно сделать вывод, что разработанная система аутентификации пользователя для информационной системы «Медик» работает на приемлемом уровне правильности функционирования, при аутентификации пользователя в информационной системе.

Заключение

В ходе выполнения данной ВКР были изучены работы на смеженные темы, исследована литература по теме аутентификации, хэширования данных, а также математический аппарат функции хэширования «Стрибог». Был произведен анализ задачи аутентификации в соответствии с законом «О персональных данных», был произведен анализ требований к аутентификации пользователя в информационной системе «Медик», рассмотрены и проанализированы различные способы аутентификации, сделан выбор реализуемого способа аутентификации.

Также была произведена постановка задачи на разработку системы аутентификации для информационной системы «Медик» в соответствии с законом «О персональных данных» и ГОСТ для аутентификации в системах, где хранится конфиденциальная информация. Был спроектирован по средством диаграмм классов различной детализации и реализован сам программный модуль аутентификации пользователя на стороне сервера с применением различных фреймворков языка программирования Java.

Этот программный модуль для большей безопасности использует алгоритм хэширования «Стрибог», который дополнен методом «соленого» хэширования паролей пользователей, что можно рассматривать как некую научную и практическую значимость. Также было произведено исследование надежности функционирования разработанной системы, посредством математического моделирования стационарного потока заявок на аутентификацию. Исходя из результатов исследования, можно сказать, что разработанный модуль работает в достаточной степени надежно.

В результате проделанной работы была спроектирована и реализована система аутентификации пользователя на стороне сервера, с применением которой можно будет быть больше уверенным в том, что вошедший пользователь с любого устройства является тем, за кого себя выдает. В ходе исследования было выявлено, что алгоритм хэширования ГОСТ Р 34.11-2012

является очень надежным алгоритмом хэширования, который стоит применять в системах, где необходимо обезопасить какие-либо данные посредством их хэширования. В данной работе этот алгоритм применялся для хэширования паролей пользователей, чтобы их можно было с достаточной долей безопасности хранить в базе данных.

Результаты данной работы можно использовать для дальнейшей доработки спроектированного программного модуля, усовершенствования его для применения к различным задачам в смеженных областях деятельности. Также результаты данной работы можно использовать, при проектировании новых информационных систем, в которых, разработанный модуль может быть частью большей системы. Нет смысла развивать данную работу в ключе кроссплатформенности, так как данное программное решение будет функционировать на любой операционной системе, где есть JVM и установлена JRE.

На данный момент эта работа является актуальной и значимой, так как сейчас очень много систем, где необходимо работать с конфиденциальными данными, доступ к которым должен быть под особым контролем.

Цель данной работы была достигнута и сопутствующие задачи были решены в полной мере.

Список используемой литературы и используемых источников

1. Agievich S.V. Ehe: nonce misuse-resistant message authentication // ПДМ. 2018. №39. URL: https://cyberleninka.ru/article/n/ehe-nonce-misuse-resistant-message-authentication (дата обращения: 22.06.2020).

2. Boychenko Oleg, Gavrikov Ilya Using hashing algorithms for user authentication in an online testing and education platform // International scientific review. 2016. №9 (19). URL: https://cyberleninka.ru/article/n/using-hashing-algorithms-for-user-authentication-in-an-online-testing-and-education-platform (дата обращения: 22.06.2020).

3. Evseev S. P., Tomashevskyy B. P. Two-factor authentication methods threats analysis // Радіоелектроніка, інформатика, управління. 2015. №1 (32). URL: https://cyberleninka.ru/article/n/two-factor-authentication-methods-threats-analysis (дата обращения: 22.06.2020).

4. Gavrikov Ilya Using two-factor authentication for securing user accounts in an online testing and education system // European research. 2016. №6 (17). URL: https://cyberleninka.ru/article/n/using-two-factor-authentication-for-securing-user-accounts-in-an-online-testing-and-education-system (дата обращения: 22.06.2020).

5. Oganesyants Lev, Vafin Ramil, Galstyan Aram, Semipyatniy Vladislav, Khurshudyan Sergey, Ryabova Anastasia Prospects for DNA authentication in wine production monitoring // Foods and Raw materials. 2018. №2. URL: https://cyberleninka.ru/article/n/prospects-for-dna-authentication-in-wine-production-monitoring (дата обращения: 22.06.2020).

6. Богданов, Д.С., Дали, Ф.А., Миронкин, В.О. Об универсальном древовидном режиме выработки хэш-кода // Современные информационные технологии и ИТ-образование. 2018. №2. URL: https://cyberleninka.ru/article/n/ob-universalnom-drevovidnom-rezhime-vyrabotki-hesh-koda (дата обращения: 01.06.2020).

7. Будько, М.Б., Будько, М.Ю., Гирик, А.В., Грозов, В.А. Метод оценки качества криптостойких генераторов псевдослучайных последовательностей // Вопросы кибербезопасности. 2018. №4 (28). URL: https://cyberleninka.ru/article/n/metod-otsenki-kachestva-kriptostoykih-generatorov-psevdosluchaynyh-posledovatelnostey (дата обращения: 01.06.2020).

8. ГОСТ Р 34.11-2012 Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования (с Поправкой).

9. ГОСТ Р ИСО/МЭК 9594-8-98 Информационная технология (ИТ). Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации.

10. Иванов, В.В, Лубова, Е.С., Черкасов, Д.Ю. Аутентификация и авторизация // Проблемы Науки. 2017. №2 (84). URL: https://cyberleninka.ru/article/n/autentifikatsiya-i-avtorizatsiya (дата обращения: 01.06.2020).

11. Иванов, Р.В. Разработка криптографического протокола двухфакторной аутентификации // Вестник науки и образования. 2015. №5 (7). URL: https://cyberleninka.ru/article/n/razrabotka-kriptograficheskogo-protokola-dvuhfaktornoy-autentifikatsii (дата обращения: 01.06.2020).

12. Комаров, А. Современные методы аутентификации: токен и это все о нем…! // T-Comm. 2008. №6. URL: https://cyberleninka.ru/article/n/sovremennye-metody-autentifikatsii-token-i-eto-vse-o-nem (дата обращения: 15.06.2020).

13. Македонский, С.А., Сухаревская, Е.В. Разработка формальной модели исследования систем аутентификации // NBI-technologies. 2017. №3. URL: https://cyberleninka.ru/article/n/razrabotka-formalnoy-modeli-issledovaniya-sistem-autentifikatsii (дата обращения: 01.06.2020).

14. Мешкова, Е.В., Митрошина, Е.В. Обеспечение безопасности протокола ssh: шифрование, аутентификация сервера, аутентификация

клиента // E-Scio. 2017. №1 (4). URL: https://cyberleninka.ru/article/n/obespechenie-bezopasnosti-protokola-ssh-shifrovanie-autentifikatsiya-servera-autentifikatsiya-klienta (дата обращения: 01.06.2020).

15. Рацеев, С.М., Ростов, М.А. О протоколах аутентификации с нулевым разглашением знания // Изв. Сарат. ун-та Нов. сер. Сер. Математика. Механика. Информатика; Izv. Saratov Univ. (N.S.), Ser. Math. Mech. Inform.. 2019. №1. URL: https://cyberleninka.ru/article/n/o-protokolah-autentifikatsii-s-nulevym-razglasheniem-znaniya (дата обращения: 16.06.2020).

16. Сабанов, А.Г. Концепция моделирования процессов аутентификации // Доклады ТУСУР. 2013. №3 (29). URL: https://cyberleninka.ru/article/n/kontseptsiya-modelirovaniya-protsessov-autentifikatsii (дата обращения: 22.06.2020).

17. Санчес, Р.Х. Агустин. Сравнительный анализ старого и нового стандартов РФ на криптографическую функцию хэширования // МНИЖ. 2016. №3-2 (45). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-starogo-i-novogo-standartov-rf-na-kriptograficheskuyu-funktsiyu-heshirovaniya (дата обращения: 01.06.2020).

18. Федеральный закон от 27.07.2006 N 152-ФЗ (ред. от 31.12.2017) «О персональных данных».

19. Филимошин, В.Ю., Давлеткиреева, Л.З. Безопасная аутентификация без использования https // International Journal of Open Information Technologies. 2017. №7. URL: https://cyberleninka.ru/article/n/bezopasnaya-autentifikatsiya-bez-ispolzovaniya-https (дата обращения: 01.06.2020).

20. Юркин, Д.В., Уткина, А.А., Первушин, А.О. Формализованный анализ протоколов аутентификации // Информационно-управляющие системы. 2018. №2 (93). URL: https://cyberleninka.ru/article/n/formalizovannyy-analiz-protokolov-autentifikatsii (дата обращения: 16.06.2020).

Приложение А

Файловые ресурсы разработанного модуля аутентификации для информационной системы «Медик»

На приложенном компакт-диске находятся следующие файловые ресурсы, описанные в таблице А.1.

Таблица А.1 – Файловые ресурсы модуля аутентификации пользователя, представленные на компакт-диске

Название пакета	Название класса	Описание класса
util	CheckCode	Генерирует код подтверждения для входа в систему, сравнивает отправленный пользователю на почту и пришедший от пользователя код подтверждения
util	DecryptionPass	Дешифрирует пришедший логин и пароль от пользователя
util	Sertificate	Генерирует сертификат пользователя, генерирует личный ключ уполномоченного по сертификации, шифрует информацию личным ключом уполномоченного по сертификации, отправляет сертификат пользователю
util	Session	Генерирует цифровую подпись, устанавливает сессию для пользователя
repository	Repository	Взаимодействует с базой данных посредством JPQL запросов
service	CompareHashService	Сравнивает полученное хэш-значения пароля с хэш-значением пароля из базы данных для конкретного пользователя. При несовпадении значений, отправляется сообщение о том, что введенный логин или пароль не верны
service	FindHashService	Находит хэш-значение пароля для пришедшего пароля, для нового пользователя или при смене пароля генерирует новое хэш-значение пароля
service	GenerationSaltService	Генерирует значение соли для нового пароля