

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)


Институт компьютерных систем и информационной безопасности
Кафедра компьютерных технологий и информационной безопасности


ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению: 10.05.03 – Информационная безопасность
автоматизированных систем

на тему «Разработка защищенной веб-платформы передачи сообщений.
Разработка серверной части и веб-интерфейса»
(наименование темы)


(КТИБ.100503.017.ПП)
(обозначение документа)

Автор  16.06.2022 Семенов В.Р.
(подпись, дата, расшифровка подписи)

Руководитель  16.06.22 Макарян А.С.
(подпись, дата, расшифровка подписи)


Консультанты:

Безопасность и экологичность работы  16.06.22 Александрова А.В.
(подпись, дата, расшифровка подписи)

Технико-экономическое обоснование  16.06.22 Дьяченко Р.А.
(подпись, дата, расшифровка подписи)

Нормоконтролер  16.06.22 Федоров С.Ю.
(подпись, дата, расшифровка подписи)

Выпускная квалификационная работа допущена к защите 22.6 2022 г.

Зав. кафедрой  Дьяченко Р.А.
(подпись, дата, расшифровка подписи)

Краснодар
2022

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности
Кафедра компьютерных технологий и информационной безопасности

УТВЕРЖДАЮ
Зав. кафедрой
компьютерных технологий
и информационной безопасности
д.т.н., проф. Р Дьяченко Р.А.
«14» марта 2022 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

По специальности 10.05.03 – Информационная безопасность
автоматизированных систем

студенту Семенову Владиславу Романовичу
(фамилия, имя, отчество)

Тема выпускной квалификационной работы «Разработка защищенной веб-платформы передачи сообщений. Разработка серверной части и веб-интерфейса»

Утверждена приказом ректора университета № 348-Ст от 14.03.2022

Руководитель к.т.н, доцент, Макарян Александр Самвелович
(должность, фамилия, инициалы)

Консультанты (с указанием относящихся к ним разделам):

- 1 Доцент, к.т.н. Александрова А. В. – раздел безопасности и экологичности
- 2 Доцент, д.т.н. Дьяченко Р. А. – раздел экономического обоснования

Срок сдачи законченного проекта на кафедру 20.06.2022

Краснодар
2022 г.

Содержание выпускной квалификационной работы

Введение

1 Нормативные ссылки

2 Термины и определения

3 Сокращения

4 Анализ предметной области

4.1 Мессенджер как основное средство коммуникации

4.2 Актуальность разработки защищенной веб-платформы передачи сообщений

4.3 Угрозы безопасности веб-приложений

5 Определение требований к защищенной веб-платформе передачи сообщений

5.1 Требования к функциональной структуре

5.2 Требования к составу выполняемых функций

5.3 Требования к безопасности

5.4 Системные требования

6 Проектирование веб-платформы

6.1 Декомпозиция системы

6.2 Клиент-серверная архитектура

6.3 Модель базы данных

6.4 Обзор способов аутентификации пользователей

6.5 Выбор оптимального способа аутентификации для защищенной веб-платформы передачи сообщений

7 Разработка защищенной веб-платформы

7.1 Аутентификация на основе JWT

7.2 Аутентификация по одноразовому паролю на почту

7.3 Хеширование паролей

7.4 Защита от атаки полным перебором

7.5 Требования к сложности пароля и валидация данных

7.6 Основные алгоритмы подсистемы аутентификации

7.7 Подсистема чатов

8 Безопасность системы

8.1 Использование переменных окружения для безопасного хранения конфиденциальной информации о системе

8.2 Защита от потери данных

8.3 Защита от SQL-инъекций

8.4 Безопасность сервера БД и СУБД

8.5 Безопасность веб-интерфейса

8.6 Использование протокола HTTPS

9 Безопасность и экологичность внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime»

10 Технико-экономическое обоснование эффективности внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime»

Заключение

Список использованной литературы

Общее количество листов ПЗ 109

Объём иллюстративной части

Мультимедийная презентация 15 кадров (диск CD-R 700 Mb)

Общее количество иллюстративной части _____

Календарный план выполнения выпускной квалификационной работы

| Число | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------|--------------------------------------|-------------------------|------------------------|--------------------------------------|---|---|--|--------------------------------|-------------------------|----|----|--|--------------------------------|---|----|----|--|--------------------------------|------------|----|----|----------|----|----|----|----|-------------------|----|----|----|----|--|
| Месяц | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | |
| Март | Экзаменационная сессия | | | | | | | | | | | | | Государственные экзамены | | | | | | | | | | | | | I Этап | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | Постановка задачи | | | | | |
| Апрель | I Этап | В | I Этап | | | | В | II Этап | | | | В | II Этап | | | | В | II Этап | | | | X | | | | | | | | | | |
| | Постан. задачи | | НИР. Постановка задачи | | | | | НИР. Анализ предметной области | | | | | НИР. Анализ предметной области | | | | | НИР. Анализ предметной области | | | | | | | | | | | | | | |
| Май | В | III Этап | | | | В | III Этап | | | | В | III Этап | | | | В | III Этап | | | | В | IV Этап | | | | | | | | | | |
| | | НИР. Основная часть ВКР | | | | | Преддипломная практика. Основная часть ВКР | | | | | Преддипломная практика. Основная часть ВКР | | | | | Преддипломная практика. Основная часть ВКР | | | | | Текст ПЗ | | | | | | | | | | |
| Июнь | IV Этап | | В | IV Этап | | | | В | V Этап | | | | В | VI Этап | | | | В | VII Этап | | | | X | | | | | | | | | |
| | Рабочий проект. Подготовка текста ПЗ | | | Рабочий проект. Подготовка текста ПЗ | | | | | Раздел БЖД. Раздел ТЭО. | | | | | Нормоконтроль ВКР. Рецензирование. Подготовка доклада | | | | | Защита ВКР | | | | | | | | | | | | | |

В – выходной день

Х – отсутствующий день в месяце

Студент Семенов В.Р. 14.08.2022
(подпись, дата)

Руководитель Макарян А.С.
(подпись, дата)

Список основной и рекомендованной литературы

1 Афанасьев А.А. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам / А.А. Афанасьев, Л.Т. Веденьев. — Москва: Горячая Линия - Телеком, 2009. — 552 с.

2 ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования. [Текст] / М.: Стандартиформ, 2018. URL: <https://docs.cntd.ru/document/1200135525>

3 ГОСТ Р 58412-2019 Защита информации. Разработка безопасного программного обеспечения. Угрозы безопасности информации при разработке программного обеспечения. [Текст] / М.: Стандартиформ, 2019. URL: <https://docs.cntd.ru/document/1200164529>

4 ГОСТ Р 34.11-2012 Информационная технология. Криптографическая защита информации. Функция хэширования. [Текст] / М.: Стандартиформ, 2013. URL: <https://docs.cntd.ru/document/1200095035>, (дата обращения: 21.05.2022)

5 ГОСТ 34.11-2018 Информационная технология. Криптографическая защита информации. Функция хэширования. [Текст] / М.: Стандартиформ, 2018. URL: <https://docs.cntd.ru/document/1200161707>

Реферат

Пояснительная записка выпускной квалификационной работы содержит: 109 страниц, 9 таблиц, 26 рисунков, 23 источника, 1 приложение.

Иллюстративная часть выпускной квалификационной работы: мультимедийная презентация 15 кадров.

БЕЗОПАСНОСТЬ, МЕССЕНДЖЕР, СЕРВЕР, ВЕБ-ИНТЕРФЕЙС, РАЗРАБОТКА, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, БАЗА ДАННЫХ (БД), ДВУХФАКТОРНАЯ АУТЕНТИФИКАЦИЯ, ОДНОРАЗОВЫЙ ПАРОЛЬ, ХЕШИРОВАНИЕ.

Объектом исследования является разработанный корпоративный мессенджер, в котором реализованы функции для безопасного общения между пользователями.

Цель выпускной квалификационной работы: разработка защищенной веб-платформы передачи сообщений, для использования в организациях, в которых существует потребность в передачи конфиденциальной информации между сотрудниками.

Основные полученные результаты:

- исследованы основные угрозы безопасности мессенджеров;
- разработана защищенная веб-платформа передачи сообщений.

Разработанное ПО прошло стадии определения требований, анализа требований, проектирования системы, кодирования и тестирования.

Основные траты приходятся на покупку и техническую поддержку серверов и специалиста по сопровождению защищенной веб-платформы передачи сообщений в организации. Период окупаемости капитальных вложений составляет один месяц, при условии, что годовая прибыль компании составляет более 40 000 000 рублей.

| | | | | | | | | |
|-----------|---------------|-----------------|----------|------|---|--------|------|--------|
| | | | | | КТИБ 100503.017.ПП | | | |
| Изм. | Лист | № докум. | Подп. | Дата | Пояснительная записка выпускной квалификационной работы | Стадия | Лист | Листов |
| Разраб. | Семенов В.Р. | <i>Семенов</i> | 16.06.22 | П | | 8 | 109 | |
| Пров. | Макарян А.С. | <i>Макарян</i> | 16.06.22 | | | | | |
| Н. контр | Федоров С.Ю. | <i>Федоров</i> | 16.06.22 | | | | | |
| Консульт. | Дьяченко Р.А. | <i>Дьяченко</i> | 16.06.22 | | | | | |
| Утв. | Дьяченко Р.А. | <i>Дьяченко</i> | 16.06.22 | | | КубГТУ | | |

Содержание

| | |
|---|----|
| Введение..... | 11 |
| 1 Нормативные ссылки..... | 13 |
| 2 Термины и определения | 14 |
| 3 Сокращения..... | 17 |
| 4 Анализ предметной области | 18 |
| 4.1 Мессенджер как основное средство коммуникации | 18 |
| 4.2 Актуальность разработки защищенной веб-платформы передачи сообщений..... | 19 |
| 4.3 Угрозы безопасности веб-приложений..... | 21 |
| 5 Определение требований к защищенной веб-платформе передачи сообщений..... | 28 |
| 5.1 Требования к функциональной структуре..... | 29 |
| 5.2 Требования к составу выполняемых функций..... | 29 |
| 5.3 Требования к безопасности..... | 30 |
| 5.4 Системные требования | 30 |
| 6 Проектирование веб-платформы | 32 |
| 6.1 Декомпозиция системы | 32 |
| 6.2 Клиент-серверная архитектура | 33 |
| 6.3 Модель базы данных..... | 34 |
| 6.4 Обзор способов аутентификации пользователей | 37 |
| 6.5 Выбор оптимального способа аутентификации для защищенной веб-платформы передачи сообщений | 44 |
| 7 Разработка защищенной веб-платформы | 46 |
| 7.1 Аутентификация на основе JWT | 47 |
| 7.2 Аутентификация по одноразовому паролю на почту..... | 48 |
| 7.3 Хеширование паролей | 49 |
| 7.4 Защита от атаки полным перебором | 51 |
| 7.5 Требования к сложности пароля и валидация данных..... | 52 |
| 7.6 Основные алгоритмы подсистемы аутентификации..... | 55 |
| 7.7 Подсистема чатов..... | 58 |

| | |
|--|----|
| 8 Безопасность системы..... | 60 |
| 8.1 Использование переменных окружения для безопасного хранения конфиденциальной информации о системе | 60 |
| 8.2 Защита от потери данных..... | 63 |
| 8.3 Защита от SQL-инъекций..... | 64 |
| 8.4 Безопасность сервера БД и СУБД..... | 65 |
| 8.5 Безопасность веб-интерфейса..... | 68 |
| 8.6 Использование протокола HTTPS..... | 68 |
| 9 Безопасность и экологичность внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime» | 71 |
| 10 Технико-экономическое обоснование эффективности внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime»..... | 81 |
| Заключение | 88 |
| Список использованных источников | 89 |
| Приложение А. Листинг программы..... | 92 |

Иллюстративная часть ВКР

Мультимедийная презентация 15 кадров (CD диск 700Мб)

Введение

В современном мире интернет охватил все сферы жизнедеятельности людей. Обмен информацией и общение между людьми сегодня переходят в виртуальное пространство. В месте с тем, мессенджеры позволяют закрыть обе эти потребности.

Мессенджеры – являются новым способом взаимодействия между людьми, при помощи обмена сообщениями в реальном времени. При возможности доступа к различной персональной информации у пользователей возникает потребность в осознанном потреблении и приватном взаимодействии. Потому и растет актуальность среди приложений, предоставляемых подобные возможности.

Недавно мир столкнулся с большим вызовом – пандемией. В новых условиях многие были вынуждены перейти на дистанционный формат работы, но как оказалось несмотря на то, что существует огромное количество приложений для связи, многие из них оказались непригодными, для корпоративного использования. Это верно, так как каждое приложение, так или иначе, разрабатывается для определенной категории людей, а также для использования в определенных условиях. Большинство имеющихся приложения не предназначались как средства передачи корпоративной информации.

Таким образом, на сегодняшний день, как никогда актуальна разработка защищенной веб-платформы передачи сообщений, которая позволила бы организациям снизить риски утечки информации ограниченного распространения, связанные с обменом конфиденциальной информацией между сотрудниками.

К сожалению, с каждым днем количество возможных атак увеличивается, как и сложность их выявления, а также поиска защиты от них. Естественно, что невозможно разработать идеально защищенную систему, устойчивую сразу ко всем атакам, которая позволит раз и на всегда закрыть

вопрос тайны переписки. Во-первых, потому что нет конечного количества уязвимостей, закрыв которые можно не переживать о безопасности. А во-вторых, потому что невозможно не допустить ни единой ошибки при разработке программного продукта.

В связи с этим, следует сосредоточиться на разработке максимально гибкой структуры, которая позволит постепенно усложняться и совершенствоваться путем внедрения новых функций для защиты передаваемой информации и самого приложения.

Проектирование системы и ее кодирование являются наиболее важными этапами в жизненном цикле разработки программ. Именно на данном этапе закладываются возможности системы, определяются ее сильные и слабые стороны, а также определяются ее возможности и целевое назначение.

При реализации работы стоит уделить особое внимание разработке системы аутентификации, так как, по данным многих исследований, в том числе проекта «Топ 10 OWASP» за последние годы, совершенствование систем аутентификации не теряет своей актуальности. Одной из причин этого являются атаки на пароли пользователей. Данный вид атак своим успехом во многом обязан пользователям, использующим «слабые» пароли.

1 Нормативные ссылки

Федеральный закон РФ N 149-ФЗ от 27.07.2006 «Об информации, информационных технологиях и о защите информации».

ГОСТ 2.301-68 ЕСКД. Форматы.

ГОСТ Р 2.105-2019 ЕСКД. Общие требования к текстовым документам.

ГОСТ Р 7.0.99-2018 СИБИД. Реферат и аннотация. Общие требования.

ГОСТ Р 7.0.100-2018 СИБИД. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

ГОСТ Р 50922-2006 Защита информации. Основные термины и определения.

ГОСТ Р 51904-2002 Программное обеспечение встроенных систем. Общие требования к разработке и документированию.

ГОСТ Р 56546-2015 Защита информации. Уязвимости информационных систем. Классификация уязвимостей информационных систем.

ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования.

ГОСТ Р 58833-2020 Защита информации. Идентификация и аутентификация. Общие положения.

ГОСТ Р ИСО 9241-110-2016 Эргономика взаимодействия человек-система. Часть 110. Принципы организации диалога.

2 Термины и определения

Авторизация – предоставление субъекту доступа прав доступа, а также предоставление доступа в соответствии с установленными правилами управления доступом.

Аутентификация – действия по проверке подлинности субъекта доступа и/или объекта доступа, а также по проверке принадлежности субъекту доступа и/или объекту доступа предъявленного идентификатора доступа и аутентификационной информации.

Аутентификационная информация – информация, используемая при аутентификации субъекта доступа или объекта доступа.

Алгоритм – конечное множество четко определенных правил, которые задают последовательность действий для выполнения конкретной задачи.

Архитектура – организационная структура системы, в которой идентифицированы компоненты, их интерфейсы и концепция взаимодействия между ними.

База данных – совокупность взаимосвязанных данных, сохраненных в одном или более компьютерных файлах в виде, позволяющем обращаться к ним пользователям или компьютерным программам с помощью системы управления базой данных.

Веб-сокеты – протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Идентификация – действия по присвоению субъектам и объектам доступа идентификаторов и/или по сравнению предъявляемого идентификатора с перечнем присвоенных идентификаторов.

Интерфейс – форма взаимосвязи между двумя или более объектами, которые совместно используют и обеспечивают данные или обмениваются ими.

Информация – сведения (сообщения, данные) независимо от формы их представления;

Компонент – замкнутая часть, комбинация частей или элемент, которые выполняют в системе отдельную функцию.

Контроль доступа – это ограничение доступа пользователей к некоторым функциям приложения, то есть применяется политика разграничения прав доступа.

Конфиденциальная информация – доверительная информация, не подлежащая огласке.

Мессенджер – это программа для смартфона или персонального компьютера для мгновенного обмена сообщения.

Несанкционированный доступ – доступ к информации, хранящейся на различных типах носителей в компьютерных базах данных, файловых хранилищах, архивах, секретных частях различных организаций путём изменения (повышения, фальсификации) своих прав доступа.

Объект доступа – единица информационного ресурса автоматизированной системы доступа, к которой регламентируется правилами разграничения доступа.

Объектно-реляционные отображения – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Операционная система – Совокупность системных программ, предназначенная для обеспечения определенного уровня эффективности системы обработки информации за счет автоматизированного управления ее работой и предоставляемого пользователю определенного набора услуг.

Переменные окружения – именованные переменные, содержащие текстовую информацию, которую могут использовать запускаемые программы.

Полный перебор или метод «грубой силы» – метод решения задачи путем перебора всех возможных вариантов.

Привилегия – метка разрешения для субъекта на доступ к ресурсу.

Программное обеспечение – совокупность компьютерных программ и программных документов, необходимых для эксплуатации этих программ.

Протокол – набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.

Разграничение доступа – разрешение или отказ в осуществлении операции с ресурсом.

Репозиторий – место, где хранятся и поддерживаются какие-либо данные.

Система контроля версий – программное обеспечение для облегчения работы с изменяющейся информацией.

Система – множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность.

Система управления базами данных – комплекс программных средств для создания баз данных, хранения и поиска в них необходимой технической информации.

Соль – это уникальная случайно сгенерированная строка, которая добавляется к каждому паролю в процессе хеширования.

Субъект доступа – лицо или процесс, действия которого регламентируются правилами разграничения доступа.

Токен доступа – доверенный объект, инкапсулирующий полномочия субъекта для получения доступа к ресурсу.

Фреймворк – программная платформа, определяющая структуру программной системы и облегчающая разработку.

Электробезопасность – система организационных и технических мероприятий и средств, обеспечивающих защиту людей от вредного и опасного воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

3 Сокращения

API – Application programming Interface – «программный интерфейс приложения».

HTTP – Hyper Text Transfer Protocol – «протокол передачи гипертекстовой разметки».

HTTPS – Hyper Text Transfer Protocol Secure, расширенная версия HTTP с шифрованием, для повышения безопасности.

JSON – JavaScript Object Notation.

JWT – JSON Web Token.

ORM – Object-Relational Mapping – «объектно-реляционные отображения».

SQL – Structured Query Language – «структурированный язык запросов».

ОС – операционная система.

БД – база данных.

СУБД – система управления базами данных.

НСД – несанкционированный доступ.

ЗВПС – защищенная веб-платформа передачи сообщений.

НПА – нормативный правовой акт

4 Анализ предметной области

4.1 Мессенджер как основное средство коммуникации

В наши дни мессенджер – это многофункциональная телефонная книга, с контактами которой пользователь может не только совершать звонки и обмениваться sms-сообщениями, но также передавать визуальную, аудио, аудиовизуальную информацию, создавать чаты и получать большой объем желаемой информации с помощью ботов, каналов посредством интернета. Явными преимуществами здесь выступают простота интерфейса, пример которого представлен на рисунке 4.1. Данный интерфейс внешне напоминает телефонную книгу, защищенность персональных данных, ощущение общения в формате «тет-а-тет» и закрытость от рекламодателей. Мессенджеры являются столь успешными в большей степени благодаря тому, что они очень удобны для пользователя, а также из-за того, что имеют широкий функционал для онлайн общения.

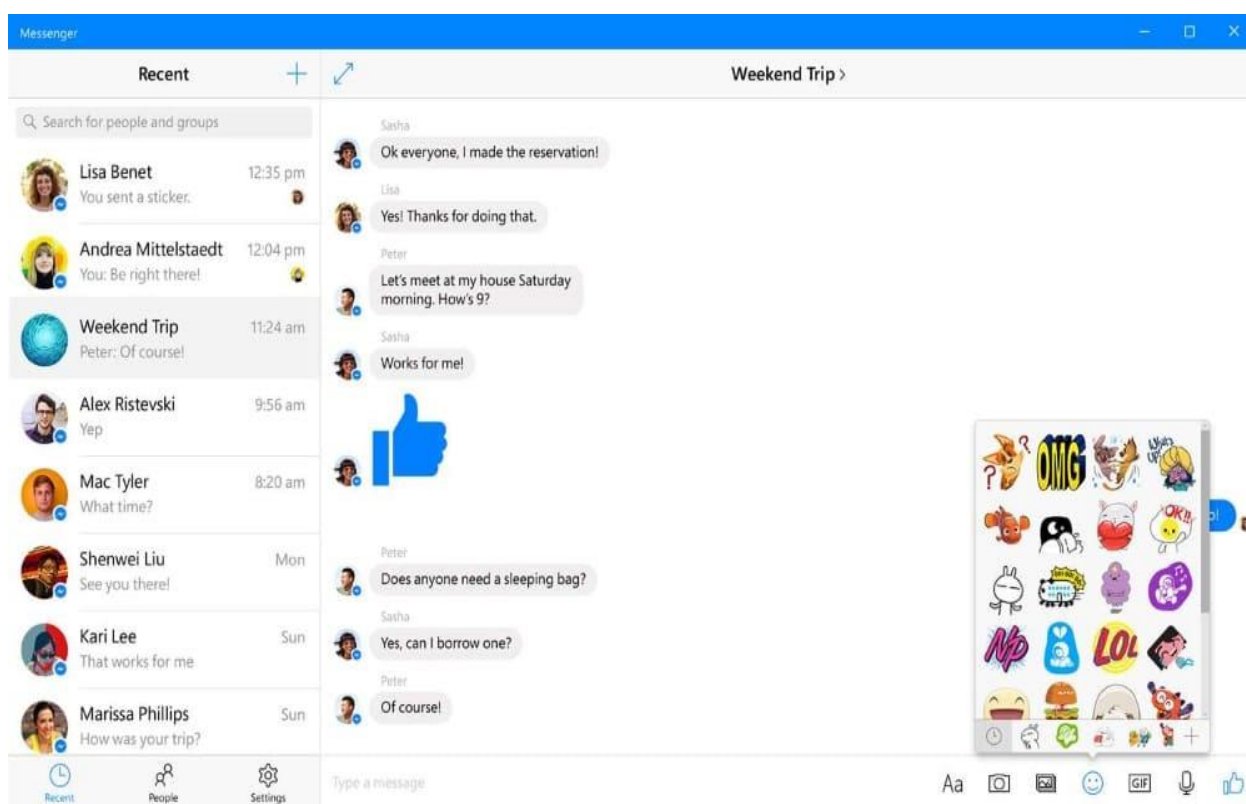


Рисунок 4.1 – Интерфейс мессенджера

Мессенджеры являются столь успешными в основном благодаря тому, что они очень удобны для пользователя, а также из-за того, что имеют широкий функционал для онлайн общения. Основными функциями мессенджеров выступают:

- бесплатный чат (текстовый, голосовой и видеочат);
- передача файлов;
- инструменты для совместной работы в режиме реального времени;
- оповещения и напоминания;
- хранение истории сообщений;
- индикация состояния собеседника (онлайн или оффлайн), занесенных

в список контактов.

4.2 Актуальность разработки защищенной веб-платформы передачи сообщений

Актуальность разработки защищенной веб-платформы передачи сообщений определяют несколько основных факторов:

1 Хранение конфиденциальной информации на серверах больших компаний несет в себе определенные риски, связанные с ее распространением. В недавнем прошлом достаточно событий, рассмотрев которые это становится очевидным. Ниже приведены две новости, хорошо иллюстрирующие данное утверждение:

1.1 В октябре 2018 года, ИБ-специалист Мэтью Сюиш (Matt Suiche) обнаружил, что защищенный мессенджер Signal некорректно осуществляет апгрейд от расширения для Chrome до полноценного десктопного клиента. Дело в том, что во время этой процедуры сообщения пользователя экспортируются в незащищенные текстовые файлы.[5]

Сообщение Мэтью представлено на рисунке 4.2



Рисунок 4.2 – Небезопасное обновление мессенджера

1.2 Первого марта 2022 года служба безопасности компании «Яндекс» объявила об утечке информации из БД. Часть новости представлена на рисунке 4.3.

[Новости Яндекса](#) / [2022](#) /

Служба безопасности Яндекс Еды сообщила об утечке информации

Служба информационной безопасности Яндекс Еды выявила утечку информации. В результате недобросовестных действий одного из сотрудников в интернете были опубликованы телефоны клиентов и информация об их заказах: состав, время доставки и так далее. Утечка не коснулась банковских, платёжных и регистрационных данных пользователей, то есть логинов и паролей. Эти данные в безопасности. Команда сервиса извиняется перед пользователями и отправит всем, кого коснулась утечка, письмо с подробностями.

Рисунок 4.3 – Новость об утечке из БД компании «Яндекс»

2 Недавняя пандемия во многом изменила уклад жизни современного общества. В нынешних условиях, когда острая фаза пандемии закончилась, стал вполне очевиден тот факт, что удаленный формат работы никуда не денется. Однако, хоть и существует огромное количество приложений для связи, многие из них непригодны, для корпоративного использования. Так как

каждое решение, так или иначе, разрабатывается для определенной категории пользователей, а также для использования в определенных условиях. А Большинство имеющихся платформ для обмена сообщениями, изначально не предназначались как средства передачи корпоративной информации, что накладывает определенные ограничения.

3 Использование иностранных мессенджеров, как показали события после начала военной спецоперации 2022 года, несут в себе большие риски для пользователей РФ. Конечно, угрозы существовали изначально, но триггер в виде операции сработал и показал, что использование иностранного ПО опасно и неэффективно, по причине, описанной далее. Актуальна угроза отключения РФ от пользования иностранными мессенджерами, что негативно повлияет на все сферы деятельности, так же агенты недружественных стран ведут информационные войны посредством использования как не декларированных программных уязвимостей, так и возможностей системы.

Из описанных выше проблем следует, что для получения, хоть уверенности в сохранности конфиденциальной информации, организациям следует иметь свой собственный аналог популярных платформ. Да, данное приложение будет иметь урезанный функционал, но также будет приносить куда больше уверенности в информационной безопасности сведений, не предназначенных для широкого распространения.

4.3 Угрозы безопасности веб-приложений

Одной из угроз безопасности информации является угроза, связанная с наличием уязвимостей программ. Существуют сотни угроз, которые могут повлиять на безопасность веб-приложений. Этой теме посвящены многие статьи и нормативные документы. Данные тексты говорят о том, что для обеспечения необходимого уровня защищенности информации требуется реализация мер, предотвращающих возникновение и устраняющих уязвимости, которые могут привести к реализации данных угроз.

Уязвимости ИС по области происхождения подразделяются на следующие классы:

- уязвимости кода;
- уязвимости конфигурации;
- уязвимости архитектуры;
- организационные уязвимости;
- многофакторные уязвимости. [1]

В соответствии с государственным стандартом по разработке безопасного программного обеспечения [2] на каждом этапе разработки есть свои меры и требования для реализации разработки безопасного программного обеспечения.

Так, при анализе требований к программному обеспечению, разработчик программного обеспечения должен определить требования по безопасности, предъявляемые к разрабатываемому программному обеспечению.

После анализа требований к программному обеспечению обычно выделяют следующие требования к безопасности программ:

- реализация системы идентификации и аутентификации;
- обеспечение защиты от несанкционированного доступа к информации, то есть ограничить возможность доступа к ресурсам информационной системы, осуществляемого с нарушением установленных прав и правил доступа к ресурсам информационной системы с применением штатных средств информационной системы или средств, аналогичных им по своему функциональному назначению и техническим характеристикам.
- регистрация событий (ведение учета действий пользователей, попыток неправомерного получения доступа, ошибок в работе программного обеспечения);
- контроль точности, полноты и правильности данных, поступающих в программу;
- обработка программных ошибок и исключительных ситуаций.

В свою очередь, при проектировании архитектуры программного обеспечения необходимо провести моделирование актуальных угроз

безопасности информации. После чего уточнить архитектуру программы с учетом результатов данного моделирования.

При выполнении тестирования программного обеспечения необходимо достичь двух целей. Во-первых, подтвердить, что программа соответствует требованиям по безопасности, предъявленным при анализе. Во-вторых, устранить уязвимости в программном коде. Эти цели могут быть достигнуты при комплексной реализации следующих видов:

- функциональное тестирование программы (задача данного вида тестирования установить соответствие разработанного программного обеспечения исходным функциональным требованиям компании клиента);
- тестирование на проникновение (цель подобного тестирования – выявить уязвимости);
- динамический анализ кода программы;
- фаззинг-тестирование программы.

После разработки безопасного программного обеспечения угроза безопасности информации все еще актуальна, поэтому существуют меры, при установке программного обеспечения и поддержки оно. Основной угрозой в данном случае является нарушение целостности программного обеспечения. То есть, отсутствует любое ее изменение либо изменение осуществляется только преднамеренно субъектами, имеющими на него право.

Для достижения подобного результата необходимо обеспечить защиту целостности программного обеспечения в процессе передачи, а также представить пользователю эксплуатационные документы. В таком случае пользователь получит возможность обнаружить любое расхождение между оригиналом программного обеспечения и полученной им копией.

Не стоит забывать, что даже после установки правильного программного обеспечения в процессе эксплуатации могут возникнуть ошибки. В данном случае разработчик программного обеспечения должен реализовать отслеживание, поиск и устранение ошибок и уязвимостей программы. Поиск уязвимостей следует проводить с использованием:

- моделирования угроз безопасности информации;
- статического анализа кода программы;
- экспертизы исходного кода программы;
- функционального тестирования программы;
- тестирования на проникновение;
- динамического анализа кода программы;
- фаззинг-тестирования программы.

Классификацией векторов атак и уязвимостей занимается сообщество OWASP (Open Web Application Security Project). Это международная некоммерческая организация, сосредоточенная на анализе и улучшении безопасности программного обеспечения.[7]

OWASP создал список из десяти самых опасных векторов атак на Web-приложения, этот список получил название OWASP TOP-10 и в нем сосредоточены самые опасные уязвимости и его обновляют каждые 3-4 года. А также предоставляются меры по предотвращению перечисленных в отчете уязвимостей. Последняя редакция списка вышла в 2021 году.[8]

На рисунке 4.4. представлено сравнение рейтингов OWASP 2017 и 2021 годов.

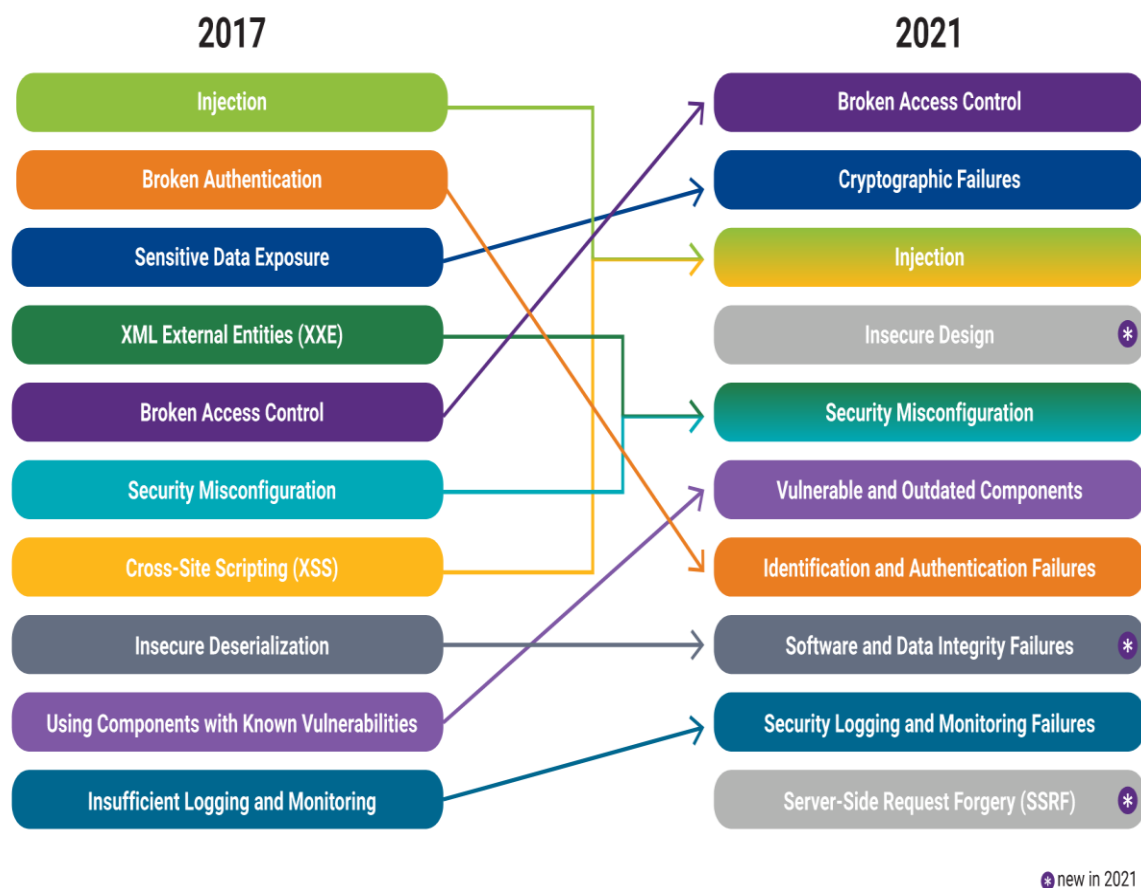


Рисунок 4.4 – Топы OWASP

Угроза нарушения контроля доступа актуальна для любой системы, в которой есть разграничение прав доступа. Данный вид угрозы подразумевает получение субъектом доступа, несоблюдающем соответствующей привилегией для получения доступа, к объекту доступа. Проблемы с доступом возникают из-за ненадежной системы аутентификации и/или из-за ошибок в программном коде, выполняющемся при обработке запроса на получение доступа.

Криптографические ошибки, ранее именуемые как «Раскрытие конфиденциальной информации», это то, что может раскрыть конфиденциальные данные. Данный вид угрозы актуален для любой не открытой системы, в которой хранится информация, не подлежащая распространению, к примеру, пароли.

В случае с паролями, в качестве криптографической ошибки может выступать использование хеш-функции, которая не имеет высокого уровня

криптостойкости или же устаревшей. А также, не использование «соли», для обеспечения большей надёжности хранимых в БД хешей паролей.

Угроза инъекций представляет собой внедрение кода злоумышленника в систему, для его дальнейшего выполнения. Данный вид угрозы актуален в том случае, если нет фильтрации вводимых пользователем данных. Например, когда разработчики отказываются от систем ORM и переходят на чистый SQL, для улучшения производительности системы, но на одном из интерфейсов, предназначенных для пользователей, забывают переработать ввод данных пользователем.

Небезопасный дизайн является новой категорией угроз, в которой основное внимание уделяется рискам, связанным с недостатками проектирования. В качестве примера можно привести возможность бронирования мест в ресторане. Если разработчик заранее не озаботится, и не внесет ограничение на количество бронируемых мест одним пользователем, то злоумышленник может воспользоваться данной уязвимостью, заблокировав возможность бронирования столика реальным клиентам, что приведет к значительным потерям для бизнеса.

Угроза небезопасной настройки актуальна для любой системы. В эту категорию угроз входит исчерпывающие сведения об ошибке. Например, в случае ошибки при попытке авторизации пользователя, нет необходимости в том, чтобы сервер выдавал ошибку по конкретному полю учетной записи (имя пользователя, пароль). Так как, это упростит злоумышленнику задачу подбора пароля, ведь он будет точно знать, что учетная запись с конкретным именем пользователя точно есть в БД.

Угроза уязвимых и устаревших компонентов всегда актуальна. Данная категория угроз связана с использованием стороннего ПО, коим являются ОС, языки программирования, а также фреймворки и библиотеки. Не так уж редко происходят случаи, когда в каком-нибудь крупном решении находят уязвимость. В частности, можно привести следующий случай:

9 декабря сообщение о недавно обнаруженной уязвимости в чрезвычайно популярном фрагменте компьютерного кода начало распространяться по сообществу кибербезопасности. На следующий день почти все крупные компании, использовавшие данный код, находились в состоянии кризиса, пытаясь понять, как пострадали их продукты и как они могли бы залатать дыру. [9]

Ошибки идентификации и аутентификации.

Подтверждение личности пользователя, аутентификация и управление сеансами имеет решающее значение для защиты от атак, связанных с аутентификацией. Тем не менее, иногда приложение не может должным образом аутентифицировать пользователей, открывая возможности для атак злоумышленников. Для закрытия подобной уязвимости необходимо использовать двухфакторную аутентификацию, ограничивать количество попыток входа, а также устанавливать сложность пароля.

5 Определение требований к защищенной веб-платформе передачи сообщений

Разработка любого программного продукта, в том числе и веб-платформы, разделяется на несколько подпроцессов, в ходе выполнения которых создается программное средство:

- процесс определения требований к ПО;
- процесс проектирования ПО;
- процесс кодирования ПО;
- процесс интеграции.

Более полный список задач решаемых, в процессе разработки и жизненного цикла ПО представлен на рисунке 5.1.

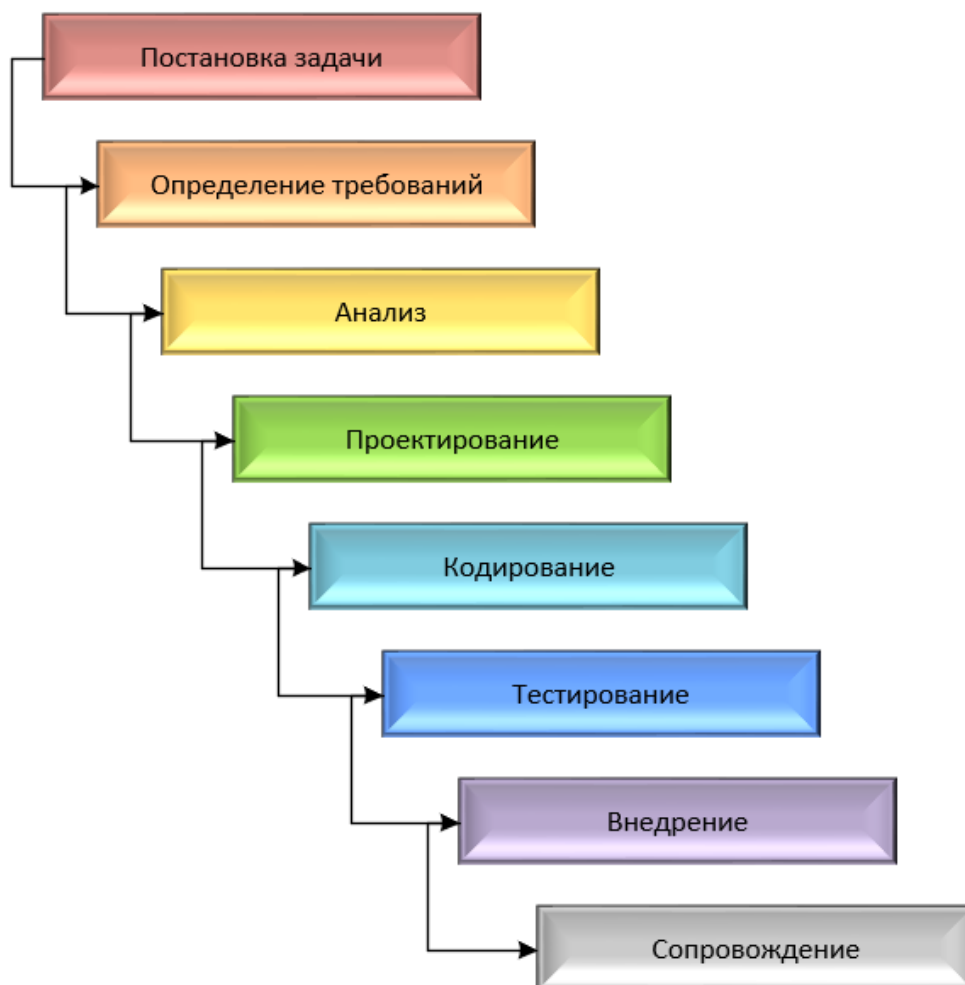


Рисунок 5.1 – Жизненный цикл и процессы разработки ПО

5.1 Требования к функциональной структуре

Разрабатываемое приложение должно включать в себя следующие компоненты:

- веб-сервер, для обработки запросов с клиентского приложения, и взаимодействия с БД;
- веб-приложение – в качестве интерфейса взаимодействия пользователя с разрабатываемой платформой;
- мобильное приложения, также для взаимодействия пользователя с разрабатываемой платформой;
- сервер БД и сама БД, для хранения информации накапливаемой в процессе эксплуатации разрабатываемого программного продукта, а также независимой работы отдельных компонентов.

Из предъявленных требований уже видно, что необходимо реализовать клиент-серверной взаимодействие между веб-приложением или мобильным приложением и веб-сервером. Также стоит заметить, что имеется 2 отдельных клиентских компонента, реализующие роль интерфейса, что в свою очередь делает целесообразным использование клиент-серверной REST API архитектуры для минимизации кода разрабатываемого приложения. Это, в свою очередь, приведет к появлению меньшему количеству ошибок и уязвимостей, так как веб-приложение и мобильное приложение будут взаимодействовать с сервером по единому интерфейсу.

Разрабатываемое приложение также должно иметь следующие подсистемы:

- подсистема аутентификации пользователей;
- подсистема чатов.

5.2 Требования к составу выполняемых функций

Подсистема аутентификации должна выполнять следующие функции:

- регистрация новых пользователей администратором;
- авторизация зарегистрированных пользователей при предоставлении корректной информации;

- отправка одноразового пароля на почту;
- аутентификация запросов к защищенным ресурсам;
- выход из системы.

Подсистема чатов должна выполнять следующие функции:

- создание нового чата;
- создание группового чата;
- отправка сообщения;
- отображение новых сообщений в реальном времени;
- выбор чата;
- получение сообщений выбранного чата;
- присоединение к чату;
- выход из чата;
- хранения информации о сообщении;

5.3 Требования к безопасности

– минимизация ошибок в подсистеме аутентификации, путем создания единого способа аутентификации как для веб-приложения, так и для мобильного приложения;

- реализация двухфакторной аутентификации;
- хеширование паролей пользователей;
- проверка корректности данных, полученных сервером;
- установка требований к сложности пароля;
- защита от получения НСД путем атаки полным перебором;
- ограничение прав доступа к БД;
- защита от инъекций;
- журналирование операций с БД;
- защита от потери данных.

5.4 Системные требования

В качестве минимальных системных требований к веб-серверу следует использовать следующие требования:

- ОС Windows Server 2008 и выше; Debian; Ubuntu; CentOS; FreeBSD;

- 64-разрядный процессор с частотой 1,4 ГГц;
- 512 Мб оперативной памяти;
- 16 Гб свободного места на жёстком диске.

Для работы веб-приложения достаточно наличия предустановленного веб-браузера на одной из ОС с графическим интерфейсом.

Также, для корректной работы базы данных сервер должен отвечать следующим минимальным системным требованиям:

- ОС Windows Server 2008 и выше; Debian; Ubuntu; CentOS; FreeBSD;
- процессор аналогичный Pentium III с тактовой частотой 600 МГц или выше (рекомендуется 1 ГГц или выше)
- минимальный объем оперативной памяти 512 Мб, рекомендуемый 1024 Мб или выше;
- объём дискового пространства от 6 Гб;

6 Проектирование веб-платформы

6.1 Декомпозиция системы

Для оптимизации работы по реализации проекта использован метод анализа. Основной операцией анализа является разделение целого на части. Задача распадается на подзадачи, система — на подсистемы, цели — на подцели и т.д. При необходимости этот процесс повторяется, что приводит к иерархическим древовидным структурам.

Успех и значение аналитического метода состоит не только и не столько в том, что сложное целое расчленяется в конечном счете на простые части, а в том, что, будучи соединены надлежащим образом, эти части снова образуют единое целое. Этот момент агрегирования частей в целое является конечным этапом анализа, поскольку лишь только после этого можно объяснить целое через его части — в виде структуры целого.[10]

С помощью метода анализа веб-платформа была разбита на 4 подсистемы — веб-сервер, веб-приложение, мобильное приложения и БД, а также несколько основных компонентов этих подсистем. Схема декомпозиции системы представлена на рисунке 6.1.



Рисунок 6.1 – Декомпозиция системы

6.2 Клиент-серверная архитектура

Одним из основных требований к приложению является реализация клиент-серверной REST API архитектуры. При реализации данного подхода как веб-приложение, так и мобильное приложение будут взаимодействовать сервером через единый интерфейс. Это позволит упростить реализацию самого приложения, а также тестирование его модулей.

Сама концепция клиент-серверной архитектуры заключается в разделении некоторых зон ответственности в системе потребности интерфейса на стороне клиента отделяются от серверных потребностей, где хранится информация. Соответствие данному требованию позволяет повысить возможность переносить клиентские коды на другие платформы. Кроме того, это приводит к значительному упрощению сервера, что позитивно сказывается на масштабируемости. Разграничение на «клиента» и «сервер» также дает возможность развиваться частям единой системы независимо.

Архитектурный стиль взаимодействия компонентов распределенного приложения REST API позволяет придать проектируемой системе следующие свойства:

- производительность;
- масштабируемость;
- гибкость к изменениям;
- отказоустойчивость;
- простота поддержки.

В действительности их намного больше. Если внимательно посмотреть на эти свойства, то можно заметить, что это так называемые нефункциональные требования к системе.

Для достижения этих не функциональных требований, при использовании REST API, соблюдается такие принципы как:

- клиент-серверная архитектура;
- отсутствие информации о состоянии клиента на сервере;
- единый интерфейс;

Для обращения к серверу используется протокол HTTP. Для данного протокола действия над данным задаются с помощью методов, основные из них это: GET – получить, POST – отправить/добавить, PUT – заменить, PATCH – изменить, DELETE – удалить.

Использование REST API архитектуры подразумевает наличие двух основных компонентов при взаимодействии с сервером: рассмотренные выше методы HTTP протокола, и ресурс. Под ресурсом понимается часть адресной строки веб-сайта, идущей после доменного имени, например: «chats/», «users/».

На рисунке 6.2 отображена взаимосвязь разных компонентов системы при использовании клиент-серверной REST API архитектуры.



Рисунок 6.2 – Клиент-серверная архитектура

6.3 Модель базы данных

База данных – один из основных компонентов любой информационной системы, она хранит информацию в структурированном виде с помощью таблиц и атрибутов.

Рассматриваемая далее реляционная модель позволяет хранить задавать различные связи между таблицами, объединяя их в единую базу, что позволяет свести к минимуму избыточность хранимых данных, упростить их ввод и организацию запросов.

Данная технология позволяет эффективно хранить большие объемы данных, не перегружая систему избыточной информацией, и в то же время позволяет сократить время на поиск нужной информации, или вовсе запрограммировать поиск необходимой информации, и ее дальнейшего

представления в виде отчета, не затрачивая лишнего времени пользователя системы.

Реляционные базы данных имеют следующий ряд свойств:

- уникальное имя для каждой таблицы;
- фиксированное число полей;
- на пересечении строки и столбца всегда есть только одно значение;
- записи отличаются друг от друга хотя бы одним значением элемента;
- полям присваиваются индивидуальные имена;
- в каждый из столбцов необходимо вставлять данные одного типа.

Разработанная модель базы данных состоит из 3 объектов: пользователь, сообщение и чат

Ниже также приведены атрибуты объектов. В таблице 6.1 содержится информация о атрибутах объекта «пользователь» и их описание.

Таблица 6.1 – Пользователь

| № | Название атрибута | Тип | Длина | Описание |
|---|-------------------|---------|-------|--------------------------------|
| 1 | username | varchar | 20 | Идентификатор пользователя |
| 2 | email | varchar | 50 | Электронная почта пользователя |
| 3 | password_hash | varchar | 512 | Хэш пароля пользователя |
| 4 | password_salt | varchar | 50 | Соль для хэша пароля |
| 5 | otp_number | int | 6 | Код подтверждения |
| 6 | otp_try | int | 1 | Количество попыток входа |
| 7 | created | date | - | Дата создания пользователя |

В таблице 6.2 содержится информация о атрибутах объекта «чат» и их описание.

Таблица 6.2 – Чат

| № | Название атрибута | Тип | Длина | Описание |
|---|-------------------|----------|-------|------------------------|
| 1 | id | int | 10 | Идентификатор чата |
| 2 | title | nvarchar | 50 | Название чата |
| 3 | is_private | boolean | - | Метка приватности чата |

В таблице 6.3 содержится информация о атрибутах объекта «сообщение» и их описание.

Таблица 6.3 – Сообщение

| № | Название атрибута | Тип | Длина | Описание |
|---|-------------------|----------|-------|--------------------------------|
| 1 | id | int | 10 | Идентификатор сообщения |
| 2 | user_username | varchar | 20 | Идентификатор автора сообщения |
| 3 | chat_id | int | 10 | Идентификатор чата сообщения |
| 4 | text | nvarchar | 2500 | Текст сообщения |
| 5 | Created_datetime | date | - | Дата отправки сообщения |

Также была создана вспомогательная таблица, которая хранит информацию о пользователя в чатах. Данная таблица необходима для реализации связи «многие ко многим» между объектами базы данных «Пользователь» и «Чат».

Реляционная модель базы данных представлена на рисунке 6.3.

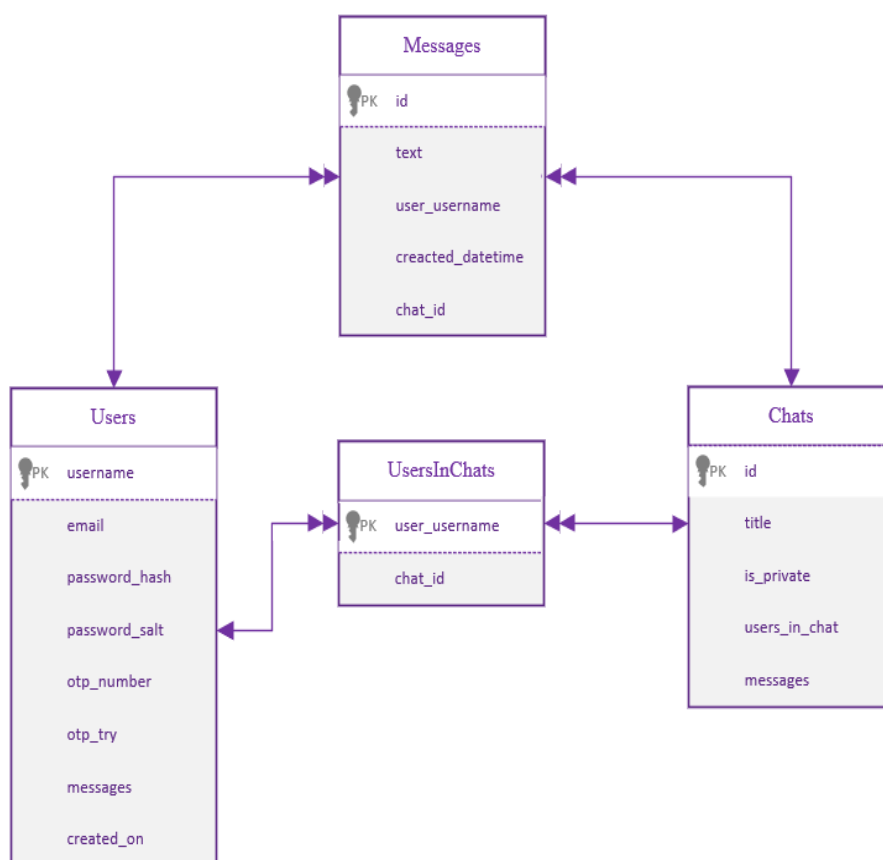


Рисунок 6.3 – Реляционная модель базы данных

6.4 Обзор способов аутентификации пользователей

В настоящее время, в зависимости от различных требований к веб-приложениям используют большое множество способов аутентификации. Высокой актуальностью обладает вопрос поиска оптимального протокола аутентификации для конкретного веб-приложения, по причине чего возникает необходимость анализа существующих способов реализации системы аутентификации.[11]

Аутентификация является одним из наиболее распространенных требований для веб-приложений. Возможность быстро и легко войти на сайт огромную роль для пользователя. В тоже время, согласно рейтингу OWASP (Open Web Application Security Project), недостатки в системе аутентификации являются одной из наиболее распространенных уязвимостей веб-приложений наряду с SQL-инекциями и межсайтовым скриптингом (XSS).

В контексте веб-приложений идентификация позволяет определить субъекта, который пытается получить доступ к ресурсу, аутентификация позволяет подтвердить подлинность субъекта, запрашивающего доступ, а авторизация предоставляет доступ к запрашиваемому ресурсу в соответствии с правами, которые установлены для субъекта.

Аутентификация по паролю.

Этот метод основывается на том, что пользователь должен предоставить имя пользователя и пароль для успешной идентификации и аутентификации в системе. Пара из имени пользователя и пароля задается пользователем при его регистрации в системе, при этом в качестве имени пользователя может выступать адрес электронной почты пользователя.

При использовании аутентификации по паролю при каждом запросе данные пользователя будут передаваться в заголовке Authorization в незашифрованном виде. Однако при использовании HTTPS протокола, является относительно безопасной.

Схема взаимодействия при аутентификации по паролю представлена на рисунке 6.4.

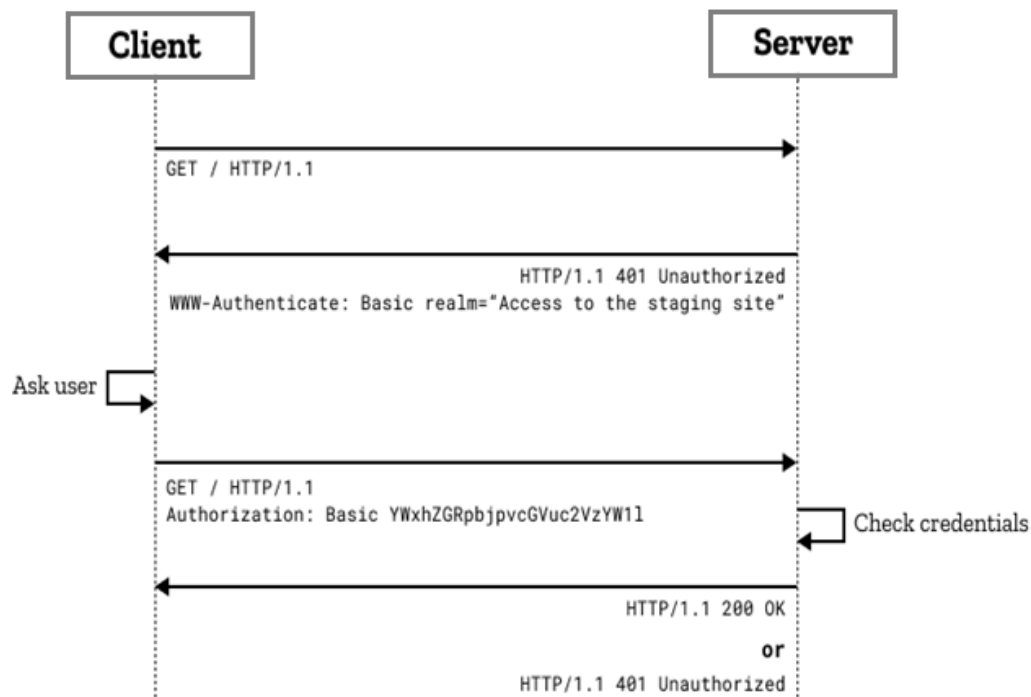


Рисунок 6.4 – Аутентификация по паролю

При данном способе у пользователя нет стандартной возможности выйти из веб-приложения, кроме как закрыть все окна браузера, так как протокол HTTP не отслеживает состояния, и, если сервер аутентифицирует пользователя с помощью имени и пароля, приложение не будет знать, тот ли это человек, что и в предыдущем запросе, и пользователю необходимо аутентифицироваться снова. Чтобы избавиться от этого неудобства, придумали аутентификацию на основе сессий, с помощью которых реализовали отслеживание состояний.[12]

Аутентификация на основе сессий.

Как уже было сказано выше, аутентификация на основе сессий позволяет хранить состояние. Это реализуется за счет того, что аутентификационная запись или сессия хранятся и на сервере, и на клиенте. Сервер должен отслеживать активные сессии в базе данных, а в браузере или приложении создаётся cookie файл, в котором хранится идентификатор сессии. Эта аутентификация является самым распространённым и широко известным методом аутентификации.

Процедура аутентификации на основе сессий выглядит следующим образом:

- пользователь вводит в браузере своё имя и пароль, после чего клиентское приложение отправляет на сервер запрос;
- сервер проверяет пользователя, аутентифицирует его, шлёт приложению уникальный пользовательский токен (сохранив его в памяти или базе данных);
- клиентское приложение сохраняет токены в куках и отправляет их при каждом последующем запросе;
- сервер получает каждый запрос, требующий аутентификации, с помощью токена аутентифицирует пользователя и возвращает запрошенные данные клиентскому приложению;
- когда пользователь выходит, клиентское приложение удаляет его токен, поэтому все последующие запросы от этого клиента становятся не аутентифицированными.[13]

Схема взаимодействия при аутентификации на основе сессий представлена на рисунке 6.5.

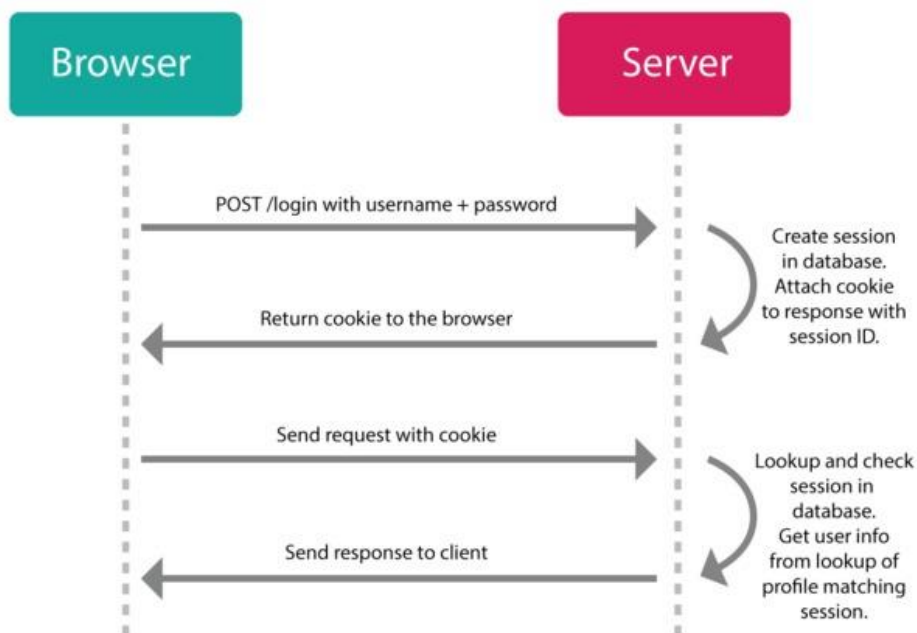


Рисунок 6.5 – Аутентификация на основе сессий

Основные недостатки данного способа аутентификации:

- использование памяти. При каждой аутентификации пользователя сервер должен создавать у себя запись. Обычно она хранится в памяти, и при большом количестве пользователей есть вероятность слишком высокой нагрузки на сервер;
- сложность масштабирования. При использовании нескольких серверов придётся реплицировать пользовательские сессии;
- недостаточная идентификация. Если на компьютере используется более одного браузера, то, каждый имеет отдельное хранилище для cookie. Поэтому cookie идентифицируют не человека, а сочетание учётной записи, компьютера, и браузера. Таким образом, любой человек, который использует несколько учётных записей, компьютеров или браузеров, имеет несколько наборов cookie;
- кража cookie. Во время эксплуатации сервер веб-приложения и браузер пользователя постоянно обмениваются cookie. Поскольку cookie могут содержать конфиденциальную, их содержимое не должно быть доступно другим, так как файлы-cookie могут быть украдены с помощью анализа трафика.

Аутентификация по ключам доступа.

Аутентификация по ключам доступа чаще всего используется для аутентификации при обращении к веб-сервису из другого веб-сервиса или приложения. Ключ доступа (access key, API key) представляет собой уникальную длинную последовательность, состоящую из произвольного набора символов.

Популярные почтовые сервисы, такие как: «Почта mail.ru», «Яндекс.Почта», «Gmail» предоставляют доступ через «пароли для внешних приложений». Пользователем запрашивается пароль, который далее сохраняется в клиентском приложении.

Использование данного способа аутентификации подразумевает ограниченные возможности при взаимодействии клиентского приложения с

веб-сервисом. Иными словами, при аутентификации по «паролю для внешних приложений» можно использовать лишь сам почтовый сервис, а не всю экосистему, предоставляемую веб-сервисом.

В некоторых случаях веб-сервисы могут предоставлять возможность настраивать права клиентских приложений и время действия ключа доступа.

Использование ключей позволяет избежать передачи пароля пользователя сторонним приложениям (в примере выше пользователь сохранил в веб-приложении не свой пароль, а ключ доступа). Ключи обладают значительно большей энтропией по сравнению с паролями, поэтому их практически невозможно подобрать. Кроме того, если ключ был раскрыт, это не приводит к компрометации основной учетной записи пользователя - достаточно лишь аннулировать этот ключ и создать новый. [12]

Аутентификация на основе токенов

Аутентификация на основе токенов работает схожим с аутентификацией на основе сессий образом, но имеет некоторые преимущества.

JSON Web Token (JWT) – это компактный формат для представления аутентифицирующей информации, предназначенный для сред с ограниченным пространством, таких как заголовки авторизации HTTP и параметры запроса URI.[14]

JWT содержит три блока: заголовок, полезную нагрузку и подпись. Первые два блока представлены в JSON-формате и дополнительно закодированы в формат base64. Обычно заголовок состоит из двух полей – типа токена и алгоритма хэширования подписи. Полезная нагрузка содержит произвольные поля в формате JSON – пары имя/значения. А подпись является результатом выполнения хэш-функции, примененной к заголовку и полезной нагрузке с добавлением закрытого ключа.

Таким образом, при попытке злоумышленником подменить данные, токен не пройдет процедуру аутентификации, поскольку подпись не будет соответствовать заявленным значениям. Сгенерировать же новую без закрытого ключа не представляется возможным.

Преимущества аутентификации на основе JWT:

- для использования не требуется хранить данные о сессиях;
- позволяет хранить дополнительную полезную информацию о пользователе, например, имя пользователя для отображения на странице приложения, что позволяет сократить количество запросов к базе данных и увеличить производительность;
- делает возможным получение доступа к нескольким доменам и сервисам с помощью одного токена;[15]
- обеспечивает большую безопасность при краже токена, чем при использовании сессий, так как время токена доступа сильно ограничено, а токен для обновления доступа отправляется только при необходимости обновить токен доступа, что снижает вероятность самой кражи.

При реализации данного вида аутентификации, обычно, создаются сразу два токен:

- токен доступа. Данный тип токена используется используется для аутентификации пользователя и имеет короткий срок жизни.
- токен обновления доступа. Имеет более длительный срок и используется только для запроса нового токена доступа по истечению его срока жизни.

На рисунке 6.6 представлена схема аутентификации на основе JWT.

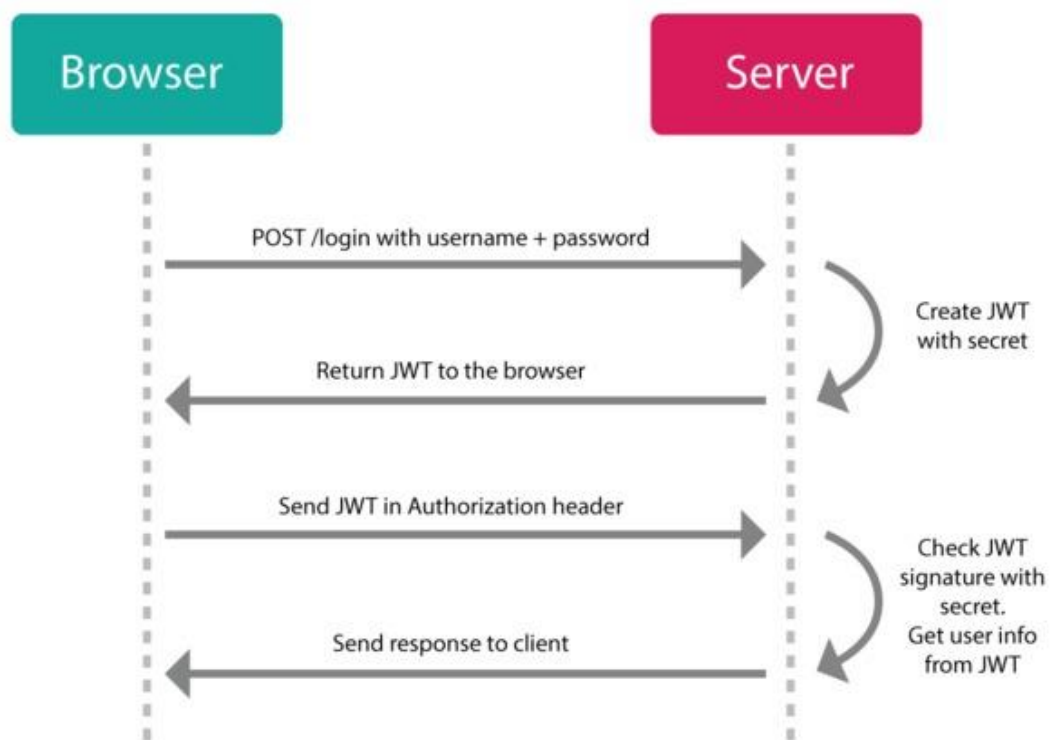


Рисунок 6.6 – Аутентификация на основе JWT

На рисунке 6.7 представлен пример токена доступа.

Encoded
PASTE A TOKEN HERE

Decoded
EDIT THE PAYLOAD AND SECRET

```
eyJhbGciOiJIUzI1NiIsInR5cCI6ImlmFjY2VzcyJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.CjodYGEkdIA3_q1E1YPZIBNyJxcz6yoztGIpTkPBo
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "access"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Kubstu-10-05-03-2022
) ☐ secret ☒ base64 ☐ encoded
```

Рисунок 6.7 – Пример токена доступа

Аутентификация по одноразовым паролям

Аутентификация по одноразовым паролям, или иначе One-Time Password (OTP) обычно применяется дополнительно к аутентификации по паролям для реализации двухфакторной аутентификации, или иначе two-factor authenticate (2FA). В этой концепции пользователю необходимо предоставить данные двух типов для входа в систему: что-то, что он знает (например, пароль), и что-то, чем он владеет (например, номер телефона или доступ к почтовому ящику). Наличие двух факторов позволяет в значительной степени увеличить уровень безопасности.

Преимущество одноразового пароля по сравнению со статическим состоит в том, что пароль невозможно использовать повторно. Таким образом, злоумышленник, перехвативший данные из успешной сессии аутентификации, не может использовать скопированный пароль для получения доступа к защищаемой информационной системе. Использование одноразовых паролей само по себе не защищает от атак, основанных на активном вмешательстве в канал связи, используемый для аутентификации.

Чаще всего, для авторизации в веб-приложения пользователю отправляется пароль на электронную почту или SMS на номер телефона.

Данный способ аутентификации подвержен следующим угрозам:

- для аутентификации по SMS – это возможность перехвата SMS злоумышленником при подключении к линии сотовой связи, вредоносное ПО на телефоне, утеря номера;

- для аутентификации по электронной почте – получение НСД к почтовому ящику пользователя через фишинг, вредоносное ПО, утечку личных данных пользователей веб-сервисов;

6.5 Выбор оптимального способа аутентификации для защищенной веб-платформы передачи сообщений

На основании изложенных выше требований и обзоре способов аутентификации были выбраны следующие методы аутентификации:

- аутентификация на основе JSON Web Token;

- аутентификация по одноразовому паролю.

Данный способ аутентификации является более предпочтительным так как:

- не требует выделения дополнительной памяти для хранения информации о текущих сессиях;

- обеспечивает лучшую защиту, чем иные рассмотренные способы реализации;

- недостатки технологии являются не существенными в рамках разрабатываемого приложения;

- позволяет реализовать REST API архитектуру, заявленную в качестве требования к разрабатываемому приложению, что в свою очередь позволит использовать единый интерфейс для аутентификации пользователей веб-приложения и мобильного приложения.

Аутентификация по одноразовому паролю выбрана в качестве второго фактора в подсистеме аутентификации, что послужит дополнительным рубежом защиты для конфиденциальных данных пользователей.

7 Разработка защищенной веб-платформы

При разработке серверной части проекта был использован высокоуровневый язык программирования «Python». Данный язык получил широкое распространение в сфере веб-разработки в следствии чего имеется множество библиотек и фреймворков, позволяющих в короткие сроки реализовать проект.

Для реализации серверной части был выбран веб-фреймворк Flask. Данный фреймворк изначально предоставляет лишь самые базовые возможности, не нагружая приложение. Такой подход идеально подходит для создания небольших приложений.

Помимо базового функционала, предоставляемого Flask, были также подключены следующие пакеты:

- Flask-SQLAlchemy и Flask-Migrate для взаимодействия с базой данных;
- Flask-Mail для отправки одноразового пароля на почту пользователя;
- SocketIO для отображения изменения в чатах в режиме реального времени.

Список ПО, используемого при разработке серверной части веб-приложения указано в таблице 7.1.

Таблица 7.1 – Программное обеспечение для разработки серверной части

| № | Тип программного обеспечения | Название | Версия |
|---|------------------------------|------------------|--------|
| 1 | Язык программирования | Python | 3.10 |
| 2 | Фреймворк | Flask | 1.1.4 |
| 3 | Пакет | Flask-SQLAlchemy | 2.5.1 |
| 4 | Пакет | Flask-Migrate | 3.1.0 |
| 5 | Пакет | Flask-Mail | 0.9.1 |
| 6 | Пакет | Flask-SocketIO | 5.2.0 |
| 7 | Библиотека | PyJWT | 2.3.0 |

7.1 Аутентификация на основе JWT

В рамках разрабатываемой системы большинство функций системы должны быть доступны только авторизованным пользователям. Для решения данной задачи была разработана подсистема аутентификации пользователей. Данная подсистема позволяет определить нужно ли обрабатывать запрос, поступивший на сервер, или же следует выдать ответ с сообщением – «Пользователь не авторизован» и соответствующей данному ответу кодом 401.

Для создания аутентификации на основе JSON Web Token (JWT) использовалась библиотека «PyJWT», которая позволяет кодировать и декодировать токены. JWT — это открытый отраслевой стандарт (RFC 7519) для безопасной аутентификации между двумя сторонами.

Также, был реализован класс MyJWT являющейся, своего рода оберткой над библиотекой «PyJWT». Данный класс позволяет осуществлять основные операции с JWT, а именно:

- кодирование токена доступа и токена обновления доступа с параметрами, задаваемыми в настройках приложения;
- декодирование токена доступа и токена обновления доступа с параметрами, задаваемыми в настройках приложения;
- получение токена из запроса;
- декоратор, для проверки наличия токена доступа в разных частях запроса: файле-cookie, заголовке и теле запроса;
- получения имени пользователя по токену;
- получение пользователя по токену.

Для кодирования токена была реализована функция «encode_token». Данная функция позволяет сгенерировать JWT. В качестве параметров данная функция принимает: username - имя пользователя, которое будет помещено в полезную нагрузку токена и token_type - тип токена, помещаемый в заголовок. При работе алгоритма из настроек приложения извлекается информация о времени действия токена в зависимости от его типа, а также информацию о

сервере аутентификации и серверах, для которых этот токен предназначен. Далее информация кодируется и дополняется подписью, секретный ключ для которой, также извлекается из настроек приложения.

Помимо функции кодирования токена была реализована функция «decode_token» позволяющая проверить корректность JWT. В качестве параметров данная функция принимает: token - сам токен и token_type - тип токена. В случае, если токен оказывается корректным и срок его действия не закончился возвращается полезная нагрузка.

Иметь возможность только создавать и проверять корректность токена недостаточно. Помимо этого, также необходимо правильно обрабатывать запросы с корректными и не корректными JWT. Для этого реализована функция «get_token_from_request» - для получения токена из запроса и декоратор «jwt_required» - для проверки наличия и корректности токена в запросе. Данный декоратор устанавливается для всех функций, предназначенных для выполнения только авторизованными пользователями. Работа этого декоратора заключается в том, чтобы проверить токен перед непосредственным выполнением функции и в случае, если токен окажется не корректным – завершить выполнение алгоритма с одним из следующих ответов сервера:

- токен отсутствует;
- срок действия токена истек;
- не корректный токен.

Кроме этого, была написана «get_user_from_jwt» - для получения имени пользователя, которое необходимо для создания токена, непосредственно из полезной нагрузки JWT, а также функция «get_user_from_jwt», которая позволяет получить о пользователе всю информацию, хранимой в БД приложения.

7.2 Аутентификация по одноразовому паролю на почту

В дополнение к аутентификации по JWT, была реализована аутентификация по одноразовому паролю, используемая в качестве второго

фактора аутентификации. Для реализации отправки одноразового пароля на электронную почту пользователя был использован пакет «Flask-Mail», предоставляющий удобный интерфейс для взаимодействия с популярными почтовыми сервисами.

Для генерации самого одноразового пароля была использована стандартная библиотека языка python, а именно ее модуль «secrets». Данный модуль предназначен для генерации криптографически стойких псевдослучайных чисел. В качестве одноразового пароля выступает строка из шести случайно сгенерированных чисел. Данная строка в последствии будет отправлена на почту пользователя, которому будет необходимо ввести ее в соответствующую форму для завершения авторизации. В случае получения сервером корректной строки код становится недействительным. Пример отправляемого письма представлен на рисунке 6.1.

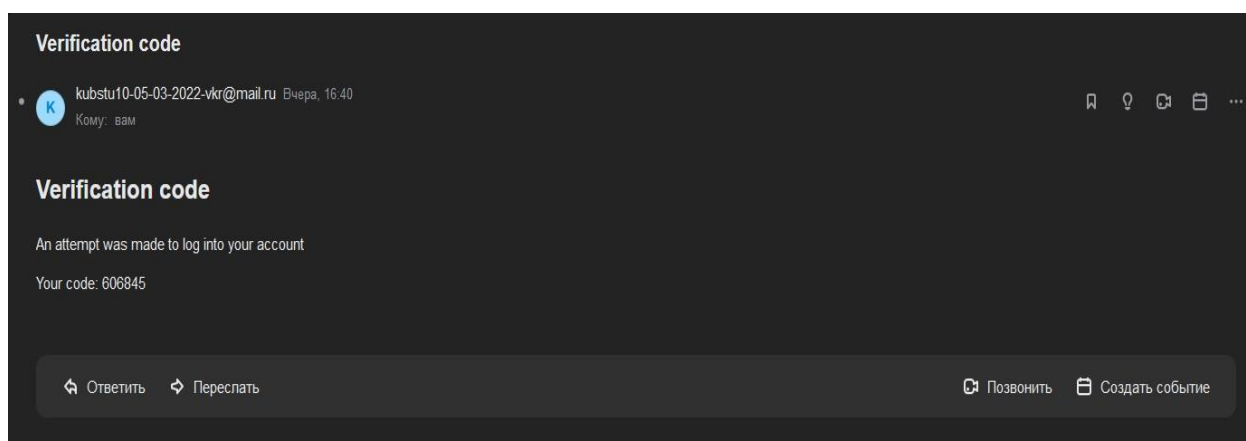


Рисунок 7.1 – Письмо с кодом подтверждения

7.3 Хеширование паролей

Самый важный аспект системы учетных записей — защита паролей пользователей. Взлом баз данных — не самое редкое явление, поэтому нужно обеспечить защиту личных данных пользователей, в том числе паролей, в подобных случаях.

Для защиты данных аутентификации в самой БД к паролю пользователей необходимо применять один из алгоритмов хеширования, чтобы не хранить пароли в открытом виде. Таким образом, даже если

злоумышленник получит доступ к БД, он не сможет получить доступ к аккаунтам пользователей. Но даже в таком случае есть свои нюансы.

Хешированные пароли не уникальны сами по себе из-за детерминированной природы хеш-функции: при вводе одного и того же пароля всегда получается один и тот же вывод.[16]

Помимо детерминированности хеш-функции также обладают следующим набором свойств:

- высокая скорость вычисления результата хеш-функции для любого заданного значения;
- невозможность получить исходное значение из результата выполнения хеш-функции, за исключением полного перебора всех возможных значений;
- лавинный эффект – изменение в одном символе исходного значения ведет к полному изменению результирующей строки;
- крайне низкая вероятность получения одной результирующей строки с разными исходными значениями.

Таким образом, чтобы избежать ситуации, когда в базе данных хранится два одинаковых хэша, имеющих в качестве исходных данных один и тот же пароль, для модели пользователя было добавлено поле для хранения «соли» – уникальной случайно сгенерированной строки, которая будет добавляться к паролю пользователя перед процессом хеширования. При таком способе в случае получения злоумышленником НСД к БД он не сможет узнать, у каких пользователей стоит одинаковый пароль от учетных записей.

Для реализации задачи хеширования пароля был написан класс «HashPassword». Данный класс позволяет выполнить следующие действия:

- получать случайную строку, сгенерированную с помощью криптографически стойкого генератора псевдослучайных чисел, для записи в таблицу пользователей в качестве «соли» для конкретного пользователя при установке пароля.

- получения хеша из передаваемой строки. Для хеширования пароля используется хэш-функция SHA256;
- сравнения передаваемого хеша с информацией, хранимой в БД.

Пример хеша пароля приведен на рисунке 6.2.

```
>>> user.password_hash  
'ff579a88791cc22eceab1b829810019117973fd6c59bb14b9ac936ffe543cceb'
```

Рисунок 7.2 – Хеш пароля пользователя

7.4 Защита от атаки полным перебором

Разрабатываемая система защищена от атаки полным перебором или иначе метода грубой силы. Данный метод подразумевает перебор всех возможных значений, для нахождения такого значения, которое бы удовлетворяла требованию системы. Основным вектором подобной атаки является подбор пароля. Подобная атака в разработанной системе нейтрализуется с помощью ограничения попыток входа.

Как уже было сказано выше, для успешной авторизации пользователю необходимо пройти двухфакторную аутентификацию. Для отправки кода подтверждения на почту пользователя была реализована отдельная функция, которая представлена на рисунке 7.3.

```
@staticmethod  
def authenticate_with_email(user):  
    try:  
        user.set_otp_number(TwoFactorAuthentication.get_OTP_number())  
        db.session.add(user)  
        TwoFactorAuthentication.send_OTP_number(user.get_email(), user.get_otp_number)  
        user.clear_otp_try()  
        db.session.commit()  
    except Exception as e:  
        raise Exception(f'{e}, \n error authenticate_with_email')
```

Рисунок 7.3 – Функция двухфакторной аутентификации

В конце выполнения данная функция сбрасывает значения количества попыток авторизации пользователя. При каждой попытке входа сервер обращается к базе данных посредством метода «get_otp_number» модели пользователя, который в свою очередь вернет результат выполнения функции «otp_try», представленной на рисунке 7.4. Данная функция при каждом вызове увеличивает число использованных попыток входа на 1. В случае достижения допустимого количества пользователь уже получить доступ к веб-платформе и ему потребуется запросить новый код подтверждения.

```
def otp_try(user):  
    if user.otp_try < app.config.get("OTP_TRY_COUNT", 3):  
        user.otp_try += 1  
        db.session.commit()  
        return user.otp_number  
    else:  
        raise OTPTryError
```

Рисунок 7.4 – Доступ к полю кода подтверждения

7.5 Требования к сложности пароля и валидация данных

Для большей защищенности системы от НСД также необходимо заранее определять допустимую сложность устанавливаемых паролей. В настройках разрабатываемой системы могут быть указаны следующие параметры, непосредственно относящиеся к сложности пароля:

- наличие в строке пароля символов в верхнем регистре;
- наличие в строке пароля символов в нижнем регистре;
- наличие в строке пароля цифр;
- наличие в строке пароля спецсимволов;
- длина пароля.

Проверка сложности пароля проводится в следующих случаях:

- при установке нового пароля пользователю введенная строка вначале проходит проверку и лишь в случае успешного завершения оной происходит генерация случайно строки «соли» и хеширование самого пароля;

– при обращении клиента к серверу для авторизации, введенный пользователем пароль вначале проверяется на соответствие указанными выше настройками, и лишь в том случае, если введенный пароль проходит проверку происходит обращение к БД;

– при вводе пароля в форме клиентского приложения.

Данные проверки позволяют решить следующий ряд задач:

– ограничить возможность установления легко подбираемых паролей, что увеличивает защищенность от НСД;

– защитить систему от НСД, при попытке получить доступ не через клиентское приложение;

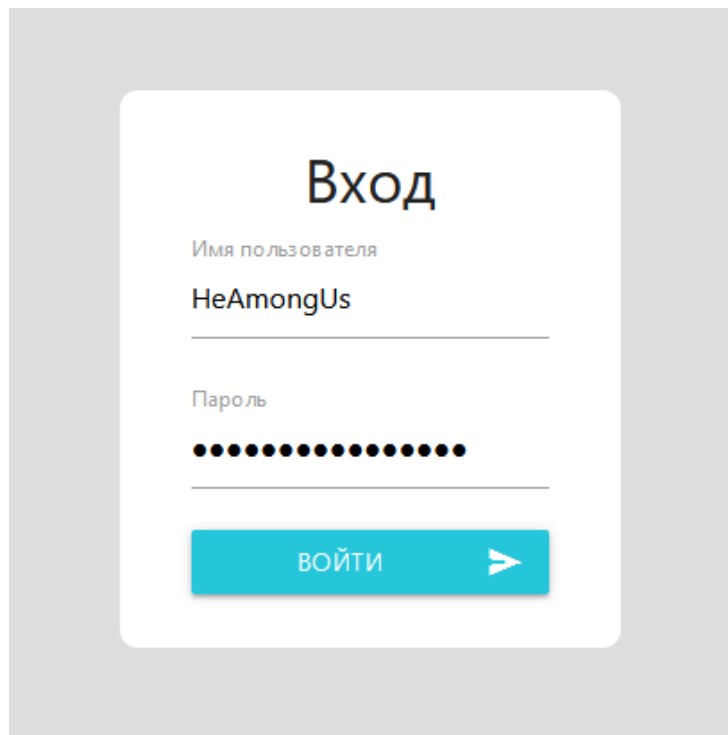
– ограничить количество запросов, которые не пройдут проверку на сервере, в следствии чего снизить нагрузку на сам сервер;

– обеспечить обратную связь пользователю системы, в случае ввода некорректной строки пароля.

В данном случае происходит некоторое дублирование кода, необходимое, в том числе, для защиты от злоумышленников, которые не будут пользоваться клиентским приложением, а используют иное ПО, предназначенное для отправки запросов на веб-сервер.

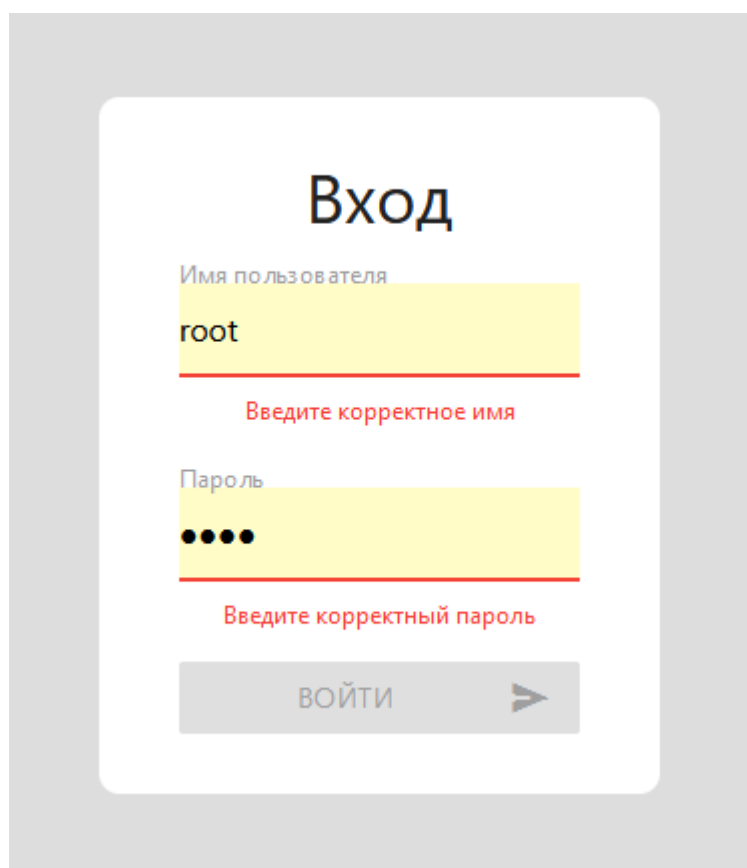
Для обеспечения проверки вводимых значений был использован пакет «Vuelidate» для фреймворка Vue.js третьей версии. С помощью данных технологий было реализовано динамическое отображение информации о корректности введенных значений.

На рисунках 7.5 и 7.6 представлена реализованная форма авторизации при ввод допустимых и недопустимых значения пароля.



The image shows a login form titled "Вход" (Login) on a light gray background. The form is a white rounded rectangle. It contains two input fields: "Имя пользователя" (Username) with the value "HeAmongUs" and "Пароль" (Password) with masked characters. Below the fields is a blue button labeled "ВОЙТИ" (Login) with a right-pointing arrow. The form is centered and has a subtle drop shadow.

Рисунок 7.5 – Форма авторизации с допустимыми значениями



The image shows the same login form as in Figure 7.5, but with invalid values. The "Имя пользователя" field contains "root" and the "Пароль" field contains four dots. Both fields are highlighted with a yellow background and a red border. Red error messages are displayed below each field: "Введите корректное имя" (Enter correct name) and "Введите корректный пароль" (Enter correct password). The "ВОЙТИ" button is now gray and disabled.

Рисунок 7.6 – Форма авторизации с не допустимыми значениями

7.6 Основные алгоритмы подсистемы аутентификации

Одним из основных алгоритмов является алгоритм обновления токена доступа. Этот алгоритм запускается в случае, если один из отправляемых на сервер запросов вернул ответ с кодом ошибки 401 и сообщением «Истек срок действия токена доступа». Данная ошибка говорит о том, что запрос не был авторизован сервером, и необходимо получить новый токен доступа. Поэтому, в случае получения подобного ответа от сервера, клиентское приложение совершает попытку обновить токен доступа, и в случае успеха повторяет запрос, который не был авторизован сервером.

Также был реализован алгоритм выхода из системы. Так как, технологи JWT позволяет не хранить данные о активных сеансах на сервере, то основная часть алгоритма выполняется в веб-приложении. В результате выполнения алгоритма из браузера удаляется вся информация, относящаяся к веб-приложению. Таким образом, при следующем запросе на сервер, пользователь будет не авторизован, поэтому при выходе из системы пользователь попадает на страницу авторизации.

Другим основным алгоритмов разрабатываемой веб-платформы является алгоритм авторизации пользователей. Данный алгоритм выполняется на веб-сервере в случае получения HTTP запроса с методом POST на ресурс «api/v1/login/».

В начале функции из данных запроса извлекается имя пользователя и пароль. Далее проверяется корректность введенных данных. Если полученные значения имени пользователя и пароля не проходят проверку, то сервер возвращает ответ с сообщением «Некорректное имя пользователя или пароль» с кодом 403.

На следующем шаге происходит попытка сопоставления полученных данных с данным из БД. Если пользователь с заданным именем пользователя зарегистрирован, то полученный пароль соединяется с «солью» и проходит через функцию хеширования, после чего сравнивается с хешем, который

храниться в БД. Если данные не совпали, то сервер возвращает ответ с сообщением «Некорректное имя пользователя или пароль» с кодом 403.

Далее происходит попытка извлечь из данных запроса код подтверждения. Если кода подтверждения нет в запросе, то на электронную почту пользователя отправляется письмо с кодом подтверждения, а также сервер отправляет ответ «На вашу почту отправлено письмо с кодом подтверждения» с кодом 200.

В случае если вместе с именем пользователя и паролем был отправлен и код подтверждения, то будет произведена проверка данного кода на корректность. В случае если код не пройдет проверку сервер отправит ответ «Не корректный код подтверждения» с кодом 403.

Если же код пройдет проверку на допустимость значения, то он будет сравнен с кодом, отправленным на почту пользователя. При неудачном сравнении серверу отправит аналогичный ответ. В случае с успешным прохождением сравнения будет сгенерирован токен доступа и токен обновления доступа. После чего сервер отправит ответ с сообщением «Вход выполнен успешно» и токенами в качестве полезной нагрузки ответа для дальнейшей обработки клиентским приложением.

На рисунке 7.7 представлена блок-схема алгоритма авторизации.

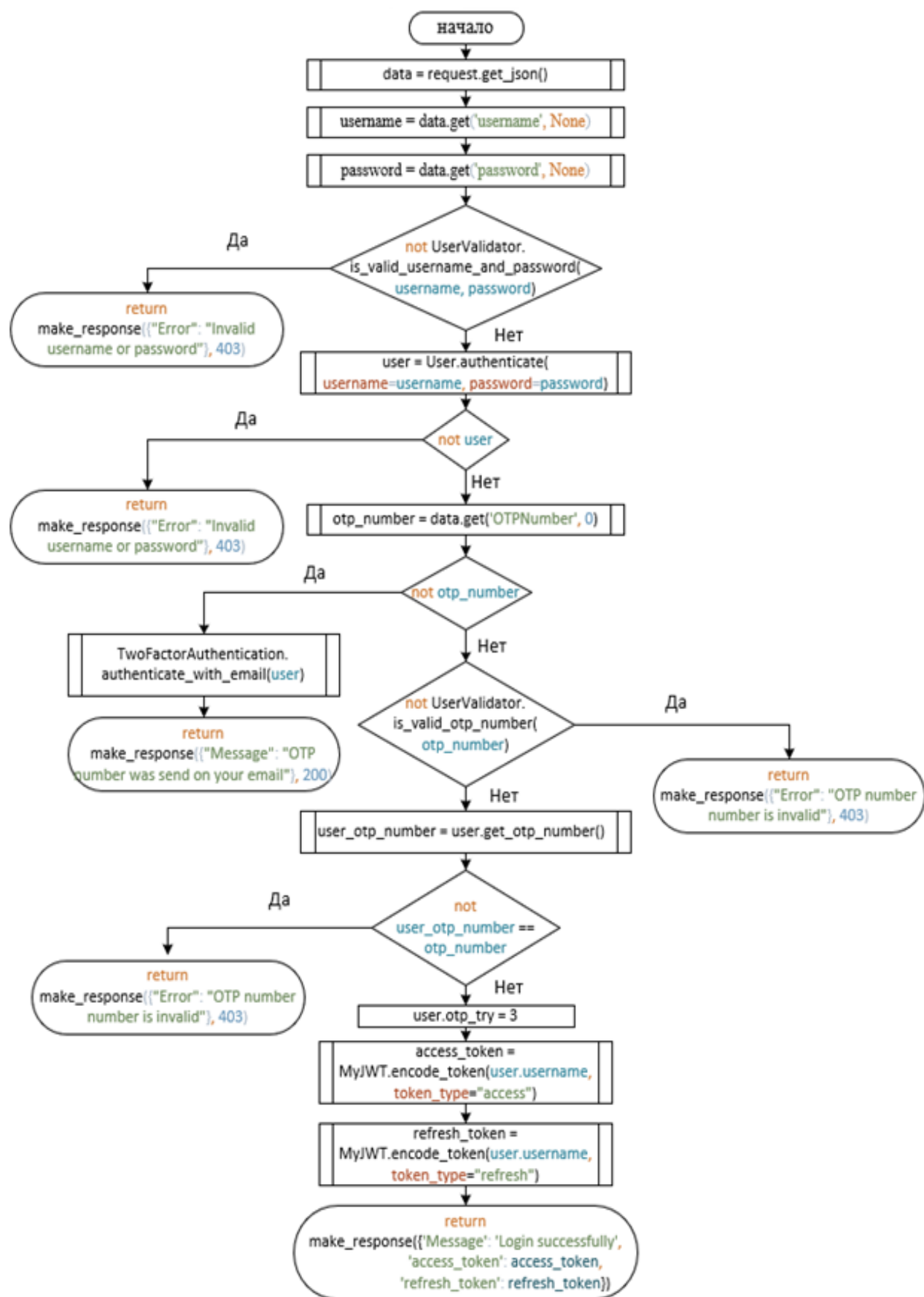


Рисунок 7.7 – Блок-схема алгоритма авторизации

После успешной авторизации клиентское веб-приложение перенаправляет пользователя на страницу чатов.

Итоговая схема авторизации в защищённой веб-платформе передачи сообщений представлена на рисунке 7.8.

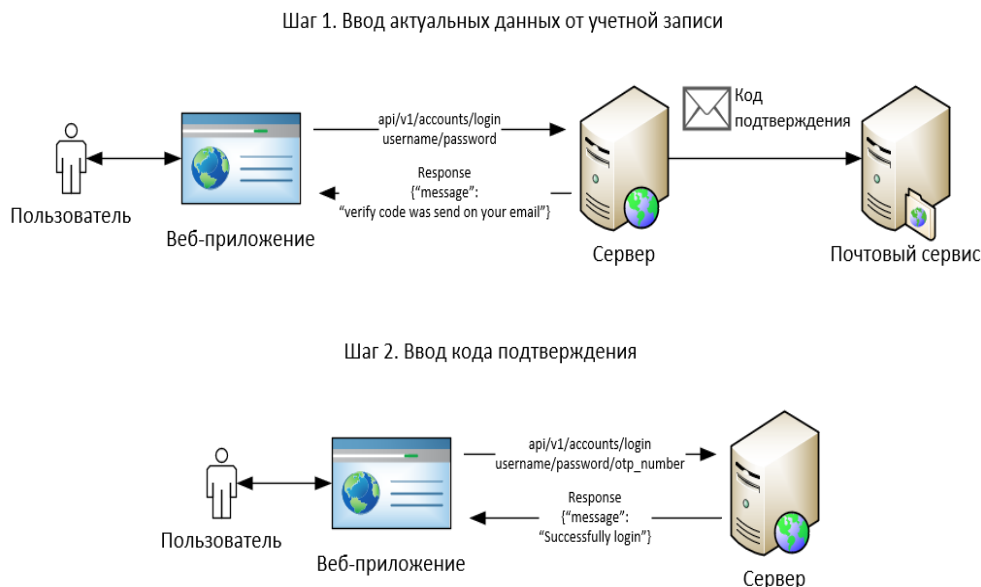


Рисунок 7.8 – Схема авторизации

7.7 Подсистема чатов

Как не трудно догадаться, основной функцией защищенной веб-платформы передачи сообщений является обмен сообщениями. Для этого была разработана подсистема чатов, все функции которой доступны лишь авторизованным пользователям.

Данная подсистема реализует следующие функции:

- получение списка чатов, в которых состоит пользователь;
- получение сообщений чата;
- поиск по всем чатам;
- подключение к чату;
- выход из чата;
- отправка сообщений;
- отображение отправленных сообщений в реальном времени.

Получение списка чатов пользователя происходит сразу после авторизации. Для этого клиентское приложение отправляет запрос серверу вместе с токеном доступа. При обработке запроса, сервер, извлекает JWT, который к этому моменту прошел проверку корректности, уже из самого токена извлекается имя пользователя. По этому имени пользователя делается запрос в БД, для получения списка чатов, в которых состоит пользователь. После чего, полученный список преобразовывается в формат JSON и отправляется в качестве ответа клиентскому приложению, где эти сообщения будут отображены в пользовательском интерфейсе.

Запрос на получение сообщений из чата отправляется в момент выбора конкретного чата. Клиентское приложение также отправляет запрос серверу, который обрабатывается только в случае, если в запросе также был отправлен корректный токен доступа. В параметрах данного запроса также должен быть передан идентификатор чата, для обращения к конкретной записи из таблицы в БД. В начале обработки запроса, аналогично примеру выше, отправляется запрос в БД для получения дополнительной информации о текущем пользователе. А также информацию о чате, в том числе и список пользователей, которые состоят в чате. Далее проверяется находится ли пользователь, совершивший запрос в текущем чате, после чего в качестве ответа будет возвращен JSON, содержащий информацию о сообщениях в запрошенном чате.

Для отправки и отображения сообщений используется технология WebSocket, данный протокол работает поверх HTTP позволяет отобразить сообщение на клиенте в ту же секунду, в которую оно было отправлено самим пользователем, или другим пользователем состоящем в чате. Подобное быстроедействие достигается за счет того, что клиент находится в состоянии прослушивания заранее определенных событий с сервера. И в случае наступления подобного события, будет выполнен предназначенный для этого события алгоритм. В данном случае, при отправке сообщения в чат, клиентское приложение мгновенно выводит его на экран.

8 Безопасность системы

Для каждой системы важна безопасность, особенно если это связано с крупными корпорациями и бизнес-проектами, в которых крайне нежелательна утечка данных или потеря их целостности. Поэтому корпоративный чат должен быть конфиденциальным, чтобы неавторизованные пользователи не имели возможности просмотреть сообщения и данные пользователей, а также изменять их. В случае если возникла ситуация, когда злоумышленники все-таки получили доступ в систему, необходимо сократить риски модификации и повреждения данных.

8.1 Использование переменных окружения для безопасного хранения конфиденциальной информации о системе

При разработки периодически возникает необходимость работы с какой-либо конфиденциальной информацией, в том числе:

- различные токены, ключи и иная аутентификационная информация от внешних веб-сервисов;
- учетные данные пользователя БД;
- настройки приложения, которые могут упростить получение НСД.

Вносить такую информацию в код программы, а тем более хранить в публично доступной системе контроля версий не безопасно. Но даже если

У данной проблемы есть следующие пути решения:

1. Хранение конфиденциальной информации в отдельных файлах, которые не подлежат хранению в публичных системах контроля версий;
2. Использование переменных командной оболочки для хранения конфиденциальной информации;
3. Использование переменных среды для хранения конфиденциальной информации.

Самый простой путь решения данной проблемы, это создание отдельных конфигурационных файлов со всей чувствительной информацией и

добавление его в файле .gitignore, для того, чтобы избежать попадания конфиденциальной информации в репозиторий.

У данного подхода есть два недостатка: добавление в .gitignore не гарантирует того, что файл не попадет в репозиторий, вся конфиденциальная информация чаще всего хранится в директории проекта, соответственно, все, кто получил доступ к проекту получает и доступ и ко всей защищаемой информации.

Другие два подхода очень похожи между собой, что при работе с переменными командной оболочки, что при работе с переменными среды используются так называемые переменные окружения, позволяющие получать запущенному приложению необходимые предопределенные значения по некоторому ключу. Основное различие двух других подходов заключается в месте хранения информации. В случае с переменными командной оболочки конфиденциальной информация будет храниться в файле с настройками конкретной командной оболочки, что соответственно ограничит их использование одним интерфейсом.

Третий подход позволяет обойти данные ограничения в соответствии с уровнем, на котором была установлена та или иная переменная.

Существуют переменные окружения следующих уровней:

- системные переменные;
- пользовательские переменные;
- временные или переменные сеанса.

Системные переменные доступны всем пользователям и запускаемым программам. Следовательно, для доступа к таким переменным необходимо войти в систему.

Пользовательские переменные доступны конкретному пользователю системы, соответственно, для их получения необходимо войти в систему под конкретным пользователем.

Временные переменные, также именуемые переменными сеанса, хранят информацию в рамках одного сеанса (сессии) пользователя или же экземпляра

приложения командной оболочки. При завершении сеанса хранимая информация будет удалена.

При разработке проекта использовалась ОС Windows 10 и один пользователь системы. В данном случае для хранения конфиденциальной информации, используемой веб-приложением, наиболее целесообразно будет воспользоваться интерфейсом ОС для добавления новых переменных окружения на уровне пользователя.

Интерфейс для использования переменных окружения в ОС Windows 10 представлен на рисунке 8.1.

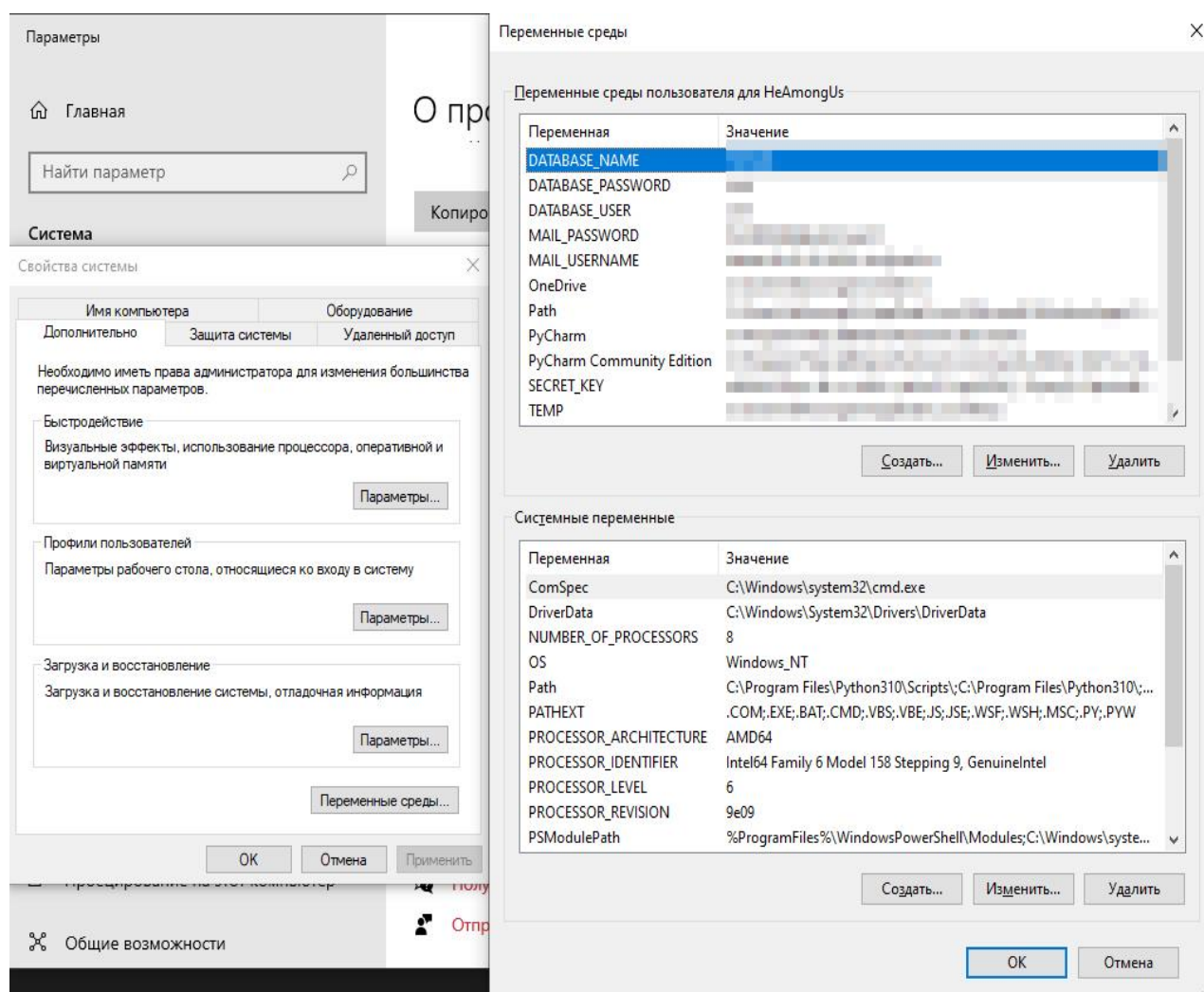


Рисунок 8.1 – Интерфейс переменных окружения ОС Windows

На рисунке 8.2 представлены настройки веб-приложения до использования переменных окружения.

```

class BaseConfig:
    SECRET_KEY = 'flask-insecure-&-x+uid2c+ynfwd11v!py&7&3_-35yacq7=2pkvim@gcfc@124'
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    ALLOWED_HOSTS = [
        'http://127.0.0.1:8080',
        'http://192.168.0.108:8080',
        'https://192.168.43.108:8080',
    ]
    CORS_HEADERS = 'Content-Type'

    # Config Flask-Mail
    MAIL_USERNAME = "kubstvu10-05-03-2022-vkr@mail.ru"
    MAIL_PASSWORD = 'TmNZJMEBi8jmE2CazaFY'

```

Рисунок 8.2 – Настройки в открытом виде

На рисунке 8.3 представлены настройки веб-приложения после добавления переменных окружения на уровне пользователя.

```

class BaseConfig:
    SECRET_KEY = os.environ.get('SECRET_KEY')
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    ALLOWED_HOSTS = [
        'http://127.0.0.1:8080',
        'http://192.168.0.108:8080',
        'https://192.168.43.108:8080',
    ]
    CORS_HEADERS = 'Content-Type'

    # Config Flask-Mail
    MAIL_USERNAME = os.environ.get("MAIL_USERNAME")
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD')

```

Рисунок 8.3 – Настройки приложения с переменными окружения

8.2 Защита от потери данных

Одним из значимых способов повышения безопасности системы является резервное копирование. Уже на стадии проектирования системы стоит задуматься о том, что разрабатываемая защищенная веб-платформа передачи сообщений может пострадать от действий злоумышленников, даже

не смотря на все реализованные методы защиты. а именно от незаконного добавления, изменения, удаления или копирования информации, сохраняющейся в базе данных.

Для защиты от подобных манипуляций с БД необходимо обеспечить резервное копирование. Резервное копирование подразумевает создание копии данных, предназначенной для восстановления в оригинальном или новом месте расположения в случае, если злоумышленник получил НСД к БД и модифицировал или повредил исходную базу.

База данных может пострадать не только от действий злоумышленников, но и при внезапном выходе из строя сервера или накопителя, на котором хранится БД. Без внедрения системы резервного копирования в случае возникновения подобных ситуаций, система оказывается не готова к последствиям, а компания может понести финансовые и репутационные издержки.

Чаще всего компании начинают задумываться о подобных мерах защиты только после возникновения экстренных ситуаций, когда уже нанесен непоправимый ущерб, что является в корне неверным подходом. Для обеспечения защиты от потери данных использовалась технология избыточного расположения жестких дисков. Суть данной технологии состоит в хранении одной информации сразу на нескольких дисках, что также позволяет ускорить обработку запросов пользователей.

Издержки от реализации подобной защите учтены в разделе 10 данной работы, определяющем экономический эффект внедрения защищенной веб-платформы передачи сообщений.

8.3 Защита от SQL-инъекций

SQL инъекция — это один из самых доступных способов взлома сайта. Его суть заключается в том, что злоумышленник внедряет различные операторы в SQL-запрос к базе данных. После этого результат выполнения запроса может полностью измениться. С помощью таких инъекций можно получить или полностью изменить данные в базе. Так же это может повлечь

за собой обхождение мер безопасности системы, так как данные с сервера будут получены с помощью SQL-инъекций. Таким образом злоумышленник может полностью перевернуть всю базу данных: украсть все данные, поменять данные в системе и обойти авторизацию. Самую главную опасность влечет за собой такой оператор, как DROP TABLE. С его помощью можно полностью удалить таблицы из базы данных, что может привести к полному уничтожению всей базы.[17]

Лучший способ предотвратить появление уязвимостей для SQL-инъекций – это использовать фреймворк, который позволяет безопасно создавать и настраивать запросы. Для выполнения данной задачи был использован пакет «Flask_SQLAlchemy», который является оберткой над фреймворком для работы с БД на языке python «SQLAlchemy».

8.4 Безопасность сервера БД и СУБД

Правильно структурированная стратегия безопасности БД должна включать средства управления для нейтрализации разных векторов угроз. Лучший подход — это встроенная структура средств управления безопасностью, которую можно легко развернуть для применения соответствующих уровней безопасности.

Разграничение прав доступа

Основным шагом в обеспечении безопасности СУБД является проверка идентификационных данных пользователя, который обращается к базе данных (аутентификация), и контроль над тем, какие операции он может выполнять (авторизация). Надежные средства аутентификации и авторизации помогают защитить данные от злоумышленников. Кроме того, внедрение разделения обязанностей помогает предотвратить злоупотребление привилегированными пользователями своими системными привилегиями для доступа к конфиденциальным данным, а также помогает предотвратить случайные или злонамеренные изменения в базе данных.[18]

В качестве СУБД при разработке была использована PostgreSQL версии 14.2. Данная СУБД использует концепцию ролей (roles) для управления

разрешениями на доступ к базе данных. Роль можно рассматривать как пользователя базы данных или как группу пользователей, в зависимости от того, как роль настроена. Роли могут владеть объектами базы данных (например, таблицами и функциями) и выдавать другим ролям разрешения на доступ к этим объектам, управляя тем, кто имеет доступ и к каким объектам. Кроме того, можно предоставить одной роли членство в другой роли, таким образом одна роль может использовать права других ролей.

Концепция ролей включает в себя концепцию пользователей («users») и групп («groups»). До версии 8.1 в PostgreSQL пользователи и группы были отдельными сущностями, но теперь есть только роли. Любая роль может использоваться в качестве пользователя, группы, и того и другого.

Роли базы данных концептуально полностью отличаются от пользователей операционной системы. На практике поддержание соответствия между ними может быть удобным, но не является обязательным.

При подключении к серверу БД, клиентское приложение указывает имя пользователя PostgreSQL, так же, как и при обычном входе пользователя на компьютер с ОС Windows или Unix. При работе в среде SQL по имени пользователя определяется, какие у него есть права доступа к объектам базы данных.[19]

Для большей безопасности при работе с БД необходимо ограничить права пользователя, который используется для подключения к СУБД. Данная манипуляция не позволит злоумышленнику получить доступ к информации, хранимой в таблицах, которые не используются при взаимодействии клиента с сервером. А также позволит избежать случаев повреждения, модификации и удаления БД.

Для этого доступ к БД был разделен между двумя пользователями. Первый используется сервером для сохранения и доступа к новым сообщениям, а также получения доступа к информации для успешной аутентификации пользователя. Второй же предназначен для администратора системы.

В данном случае разрабатываемое приложение также должно получить некоторую конфиденциальную информацию. Для безопасного хранения данной информации был использован рассмотренные выше способ с переменными окружения на уровне пользователя.

На рисунке 7.4 представлены настройки веб-приложения после разделения доступа к БД и добавления переменных окружения на уровне пользователя.

```
class Config(BaseConfig):
    DEBUG = True
    DATABASE_NAME = os.environ.get("DATABASE_NAME")
    DATABASE_USER = os.environ.get("DATABASE_USER")
    DATABASE_USER_PASSWORD = os.environ.get("DATABASE_USER_PASSWORD")
    DATABASE_ROOT = os.environ.get("DATABASE_ROOT")
    DATABASE_ROOT_PASSWORD = os.environ.get("DATABASE_ROOT_PASSWORD")
    SQLALCHEMY_DATABASE_URI = f"postgresql://{DATABASE_USER}:{DATABASE_USER_PASSWORD}@localhost:5432/{DATABASE_NAME}"
```

Рисунок 8.4 – Настройки приложения БД

Аудит и мониторинг

Все действия с базой данных должны регистрироваться для целей аудита, включая действия, происходящие в сети, а также действия, инициированные в базе данных (обычно посредством прямого входа в систему), которые обходят любой мониторинг сети. Аудит должен выполняться, даже если сеть зашифрована. Базы данных должны обеспечивать надежный и всесторонний аудит, который включает информацию о данных, клиенте, с которого делается запрос, подробности операции и сам SQL-оператор.[18]

Ведение подробного контрольного журнала — одна из опций безопасности системы, которое часто упускается из виду. Для улучшения видимости того, что происходит в базе данных необходимо включить подробное ведение журнала. Это позволит включить ведение журнала всех попыток подключения и всех выполненных операторов SQL.[20]

Прослушивание адресов

Хорошей практикой является ограничение портов и адресов, которые прослушивает сервер для клиентских подключений, с помощью параметра `listen_addresses` файла конфигурации. Если узел, на котором работает PostgreSQL, имеет несколько сетевых интерфейсов, можно использовать этот параметр, чтобы убедиться, что сервер прослушивает только те интерфейсы, через который клиенты будут подключаться к нему.[20]

8.5 Безопасность веб-интерфейса

Как уже было сказано выше, для разработки веб-интерфейса использовался фреймворк «VUE.js» для языка программирования JavaScript. Данный фреймворк предоставляет защиту от внедрения HTML, CSS, и JavaScript.

Для защиты от внедрения HTML-кода VUE.js автоматически экранирует HTML-содержимое, предотвращая случайное внедрение HTML в приложение. Однако, данная атака все еще может быть актуальна, если разработчик предоставит пользователю возможность отрисовать введенный HTML-код. В рамках разрабатываемой платформы у пользователя, при взаимодействии с приложением, нет необходимости в написании и отрисовки HTML-код. Таким образом, данная атака не является актуальной.

Аналогичная ситуация обстоит и с внедрением CSS и JavaScript-кода. Если на уровне программного кода не предоставлять пользователю возможности для написания JavaScript-кода или CSS стилей, то фреймворк автоматически заблокирует исполнение кода.

8.6 Использование протокола HTTPS

HTTPS – это протокол HTTP, но который работает через криптографический протокол SSL. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений

Протокол SSL

SSL (Secure Sockets Layer) протокол позволяет безопасно общаться пользователю и серверу и вся информация, передающаяся между ними, является конфиденциальной. Все данные отправляются в закодированном виде и их расшифровка возможна только с помощью специального сгенерированного сеансового ключа.

Чтобы подключить данный протокол необходимо зарегистрировать сертификат SSL сервера в центре сертификации. SSL сертификат – выполняет роль подписи сервера, подтверждающей, что запрос направлен от реального сервера и является настоящим без подмен.

В сертификате есть следующие обязательные поля:

- адрес сервера, где оформлен сертификат;
- наименование организации, на которую оформлен сертификат;
- адрес организации;
- срок действия сертификата.

Центр сертификации или удостоверяющий центр — сторона (отдел, организация), чья честность неоспорима, а открытый ключ широко известен. Задача центра сертификации — подтверждать подлинность ключей шифрования с помощью сертификатов электронной подписи.[21]

После выполнения регистрации сертификата, SSL становится доступен серверу для общения между клиентами. Схема образования соединения по SSL у клиент-сервера следующая:

- клиент подключается к серверу и посылает запрос на безопасное подключение по протоколу SSL;
- клиент показывает все алгоритмы для шифрования и хеширования, которые может поддерживать;
- сервер сравнивает свой список алгоритмов с алгоритмами клиента и выбирает самый надежный;
- сервер отправляет клиенту свой сертификат для аутентификации;
- клиент удостоверяется в подлинности сертификата, который отправил сервер, используя корневые сертификаты удостоверяющего центра;

– происходит генерация сеансового ключа с выбранным алгоритмом шифрования.

После всех проделанных шагов между клиентом и сервером устанавливается безопасное соединение, и они общаются с использованием сгенерированных ключей. Сервер и клиент создают по два ключа: открытый и закрытый. Они передают друг другу только открытые ключи, с их помощью можно только закодировать сообщение. А расшифровать такие данные можно только при помощи закрытого ключа. Клиент отправляет данные на сервер, сервер кодирует их с помощью своего открытого ключа и расшифровать их может только он сам. Точно так же и с клиентом.

9 Безопасность и экологичность внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime»

9.1 Значение и задачи безопасности жизнедеятельности

Трудовая деятельность разработчика подразумевает использование вычислительной техники (компьютера) на всех стадиях разработки программного продукта. Вследствие чего увеличивается нагрузка на психофизиологическое состояние человека, что может привести не только к снижению работоспособности, но и к проблемам со здоровьем. При недолгой работе с компьютером обычно сталкиваются с:

- болями в спине, запястьях, рук;
- жжением в глазах;
- снижением четкости зрения.

Учитывая данное обстоятельство, вопрос защиты разработчика от вредного влияния компьютера во время работы становится актуальным.

Задачами безопасности жизнедеятельности в обеспечении комфортной и безопасной работы являются:

- анализ и оценка негативных воздействий при работе с компьютером;
- определение мер защиты для тестирующего от возможных вредных последствий во время работы с ВТ;
- по возможности максимальная минимизация всех негативных факторов, влияющих на тестирующего во время работы;
- создание нормальной и безопасной среды для работы разработчика.
- приспособление ПО для его безопасного и эффективного использования людьми, с учетом психофизиологических особенностей человеческого организма.

9.2 Анализ условий труда и вредных производственных факторов при работе с компьютером на рабочем месте

Анализ условий труда для оценки их напряженности основан на выборке группы сотрудников, за которыми ведутся наблюдения с фиксированием

результатов в течении всего рабочего дня, на протяжении не менее пяти рабочих дней. В процессе анализа фиксируется как время, затраченное на выполнение той или иной задачи, так и факторы, влияющие на работоспособность каждого сотрудника.

Оснащение рабочего места разработчика:

- ноутбук или стационарное АРМ;
- компьютерная периферия (компьютерная мышь, клавиатура, принтер);
- компьютерный стол и стул;
- телефон;
- зарядные устройства.

Используемое ПО разработчиком во время работы:

- среды разработки: «PyCharm» и/или «WebStorm»;
- язык программирования «Python» и/или «Javascript»;
- ПО для отладки и контроля версий;
- Chrome Web Tools.

Организация трудового распорядка

Рабочий день разработчика составляет 9 часов, 8 из которых он занят выполнением своих трудовых обязанностей/ Как и другие категории сотрудников, офисные работники имеют право на предоставление им перерыва на обед, продолжительность которого регламентирована положениями статьи 108 ТК РФ. Согласно данной статьи его длительность может составлять от 30 минут до 2 часов, причем этот перерыв не включается в состав рабочего времени. Конкретная длительность перерыва и время его предоставления определяются правилами трудового распорядка внутри компании либо соглашением между работниками и работодателем. Диаграмм распределение активности разработчика в течении трудового дня представлена на рисунке 9.1

Распределение активности разработчика в течении рабочего дня



Рисунок 9.1 – Диаграмма распределения активности разработчика

Кроме того, перерывы в работе для отдыха от компьютера предоставляются отдельно от перерыва на обед. А с 2021 года установление перерывов во время работы за компьютерами не урегулировано НПА, то есть перерывы устанавливаются самостоятельно руководителем.

Для сотрудников организации трудовые обязанности предполагают постоянную работу за компьютером, для них предусмотрены дополнительные перерывы. Общая продолжительность перерывов в работе за монитором определяется в зависимости от вида деятельности и пользовательской активности, которая рассчитывается на основании количества обрабатываемых знаков за минуту. На основании этих критериев регламентированная продолжительность отдыха от такой работы при условии 8-часовой смены должна составлять от 50 до 90 минут. Во время такого отдыха рекомендуется выполнять физические упражнения.

Микроклимат в офисе

Основные требования к параметрам микроклимата и охране труда в офисе должны характеризоваться параметрами в таблице 9.1.

Таблица 9.1 – Параметры микроклимата

| № | Наименование параметра | Допустимые значения |
|---|--------------------------------------|---------------------|
| 1 | Уровень влажности | 40-60% |
| 2 | Скорость движения воздуха | 0,1-0,3 м/с |
| 3 | Уровень шума | Не более 80 дБА |
| 4 | Уровень освещенности | Не менее 400 Лк |
| 5 | Температура воздуха в холодный сезон | От 22° до 24° |
| 6 | Температура воздуха в теплый сезон | От 23° до 25° |

Требования к параметрам освещения внутри офиса определены дополнительным документом – национальным стандартом РФ ГОСТ Р 55710-2013.

9.3 Оценка напряженности трудового процесса и расчет трудоемкости работ

Для оценки напряженности трудового процесса разработчика, учитывая специфику его деятельности, которая связана с большой длительностью сосредоточенной работы и ответственностью, был использован метод проведения измерений, на основании Руководства Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда». Анализ напряженности труда представлен в таблице 9.2.

Таблица 9.2 – Анализ напряженности трудового процесса разработчика

| Показатели напряженности трудового процесса | Фактическое значение напряженности до внедрения методики, Класс условий труда |
|--|---|
| 1. Интеллектуальные нагрузки | |
| 1.1 Содержание работы | Решение сложных задач с выбором по известным алгоритмам (работа по серии инструкций), 3.1 |
| 1.2 Восприятие сигналов (информации) и их оценка | Восприятие сигналов с последующим сопоставлением фактических значений параметров с их номинальными значениями. Заключительная оценка фактических значений параметров, 3.1 |
| 1.3 Распределение функций по степени сложности задания | Обработка, проверка и контроль за выполнением задания, 3.1 |
| 1.4 Характер выполняемой работы | Работа по индивидуальному плану, 1 |

Продолжение таблицы 9.2

| | |
|---|--|
| 2 Сенсорные нагрузки | |
| 2.1. Длительность сосредоточенного наблюдения (в % от времени смены) | Более 75, 3.2 |
| 2.2 Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 ч работы | До 75, 1 |
| 2.3 Число производственных объектов одновременного наблюдения | До 5, 1 |
| 2.4 Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0.5 м) в мм при длительности сосредоточенного наблюдения (% времени смены) | Более 5 мм – 100%, 1 |
| 2.5 Работа с оптическими приборами (микроскопы, лупы и т.п.) при длительности сосредоточенного наблюдения (% времени смены) | До 25, 1 |
| 2.6 Наблюдение за экранами видеотерминалов (часов в смену): | |
| при буквенно-цифровом типе отображения информации | Более 4, 3.2 |
| при графическом типе отображения информации | До 5, 2 |
| 2.7 Нагрузка на слуховой анализатор (при производственной необходимости восприятия речи или дифференцированных сигналов) | Разборчивость слов и сигналов от 100 до 90%. Помехи отсутствуют, 1 |
| 2.8 Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемое в неделю) | до 16, 1 |
| 3 Эмоциональные нагрузки | |
| 3.1 Степень ответственности за результат собственной деятельности. Значимость ошибки | Несет ответственность за функциональное качество вспомогательных работ (заданий). Влечет за собой дополнительные усилия со стороны вышестоящего руководства, 2 |
| 3.2 Степень риска для собственной жизни | Исключена, 1 |
| 3.3 Степень ответственности за безопасность других лиц | Исключена, 1 |
| 3.4 Количество конфликтных ситуаций, обусловленных профессиональной деятельностью, за смену | Отсутствуют, 1 |

Окончание таблицы 9.2

| | |
|--|--|
| 4 Монотонность нагрузок | |
| 4.1 Число элементов (приемов), необходимых для реализации простого задания или в многократно повторяющихся операциях | 9-6, 2 |
| 4.2 Продолжительность (в с.) выполнения простых производственных заданий или повторяющихся операций | 100-25, 2 |
| 4.3 Время активных действий (в % к продолжительности смены). В остальное время наблюдение за ходом производственного процесса. | 20 и более, 1 |
| 4.4 Монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса в % от времени смены) | 76-80, 2 |
| 5 Режим работы | |
| 5.1 Фактическая продолжительность рабочего дня | 8-9 ч, 2 |
| 5.2 Сменность работы | Односменная работа (без ночной смены), 1 |
| 5.3 Наличие регламентированных перерывов и их продолжительность | Перерывы регламентированы, достаточной продолжительности: 7% и более рабочего времени, 1 |

Так как 4 показателя относятся к классу 3.1, а 2 показателя к классу 3.2, то класс условий труда по показателям напряженности – 3.1 (вредный). Учитывая полученные данные, необходимо снизить нагрузку на сотрудника, которую он испытывает в ходе работы.

Необходимо посчитать трудоемкость работ. Чтобы найти коэффициент трудоемкости, необходимо взять время, потраченное на решение задачи и разделить на количество выполненных задач в день:

$$T_p = \frac{T}{Q}$$

где T_p – трудоемкость;

T – время, затраченное на создание тестов (в часах);

Q – количество решенных задач за рабочий день.

Подставляя, получаем следующее значение:

$$T_p = \frac{8}{6} = 1,33 \text{ час/задача}$$

Для снижения вредных факторов во время работы, а именно длительная и напряженная работа за ВТ и улучшения условий труда для разработчика, одним из вариантов решения является внедрение еще одного специалиста по обеспечению качества программного продукта. Данное направленно на повышение эффективности, подробнее можно ознакомиться в разделе 10 данной работы.

9.4 Электробезопасность

Рекомендации при использовании электроприборов определены в документе «Правила устройства электроустановок». Для разработчиков характерен следующий набор правил рекомендуемых к соблюдению при выполнении трудовых обязанностей:

- эксплуатация офисной и бытовой техники, используемой в организации в соответствии с инструкциями производителей;
- соблюдение ограничений по уровню нагрузки на электрическую сеть;
- отключение электроприборов на время отсутствия в офисе.

Для обеспечения электробезопасности проводиться следующие мероприятия:

- контроль и профилактика повреждений изоляция - производится квалифицированным специалистами по электромонтажу;
- контроль защищенности заземления электропроводки;
- установка защиты от короткого замыкания, автоматическое отключение питания.

Во время работы оператору АРМ запрещается:

Переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;

- попадание влаги на поверхность системного блока, монитора, рабочего стола, клавиатуры, дисковод, принтера и других приборы;
- самостоятельное вскрытие корпуса техники и дальнейший ремонт.

9.5 Пожарная безопасность

Меры пожарной безопасности помещений, которые по своему назначению можно отнести к офисам изложены в следующих документах:

- Федеральный закон РФ № 69-ФЗ «О пожарной безопасности» от 21.12.1994 г.;
- Федеральный закон РФ № 123-ФЗ «Технический регламент о требованиях ПБ» от 22.07.2008 г.;
- Постановление правительства РФ от 16 сентября 2020 года N 1479 «Об утверждении Правил противопожарного режима в Российской Федерации».

Согласно федеральному закону №69 ответственным за пожарную безопасность является ее руководитель.

В помещении, которое отводится под офис разработки, горючими компонентами являются: различные перегородки, изоляции кабелей, мебель и прочее. Высокая вероятность возникновения пожара напрямую связана с ВТ, так как источником зажигания могут являться электронные схемы, стабилизаторы электропитания, приборы технического обслуживания и прочие электроприборы.

Таким образом, в данном помещении должны быть установлены: пожарные извещатели автоматической сигнализации, передающие сигнал в помещение пожарного поста, охраны или диспетчерской здания, а также иметься воздушно-пенные, воздушно-эмульсионные, порошковые огнетушители.

9.6 Охрана окружающей среды

В ходе работы с компьютерной техникой, некоторые компоненты могут ломаться и при этом не подлежать восстановлению. Также с течением времени происходит износ и моральное устаревание техники. Проще говоря, любая техника в конце концов приходит в негодность, а согласно приказу Минприроды от 11.06.2021 №399 «Об утверждении требований при обращении с группами однородных отходов I-V классов опасности» в мусор

нельзя выбрасывать оборудование компьютерное, электронное, оптическое, утратившее потребительские свойства. Иными словами, является обязательным утилизация жестких дисков, процессоров, оперативной и видео памяти, приводов дисков принтеров, картриджей, компьютерных мышек, клавиатур и иной вышедшей из строя компьютерной техники.

9.7 Эргономика приложения

В разработке программного обеспечения важную роль составляет эргономическая составляющая проекта, позволяющая повысить эффективность труда работника, и снизить психофизиологическую нагрузку.

Для разработки эргономичного интерфейса использовался стандарт ГОСТ Р ИСО 9241-110-2016. В данном стандарте рассмотрены принципы внутрисистемного диалога, которые в большинстве случаев не зависят от способов ведения диалога и применимы при анализе, проектировании и оценке интерактивных систем.

Стандарт нацелен на предотвращения ошибок, в части:

- выполнения действий, не предусмотренных производственным заданием;
- наличия в системе информации, вводящей пользователя в заблуждение;
- наличия недостаточной и неполной информации об интерфейсе пользователя;
- появления неожиданной ответной реакции интерактивной системы;

На основе вышеописанных стандартов был разработан адаптивный веб-дизайн, обеспечивающий правильное отображение сайта на различных устройствах, динамически подстраивающийся под разрешение и соотношение сторон экрана пользователя. Сайт может работать на смартфоне, планшете, ноутбуке, то есть на всем спектре устройств, при этом не урезая функционал или удобство его использования, сохраняется привычный интерфейс. Адаптивный дизайн сайта представлен на рисунке 9.2.



Рисунок 9.2 – Адаптивный дизайн

Также стоит отметить, что разрабатываемая веб-платформа предназначена для использования в компаниях, а не для общего пользования. Офисный сотрудник нередко сталкивается со стрессом, что было учтено в дизайне приложения. Были изучены различные цветовые решения и их влияние на психику человека. Таким образом, приложение выполнено в синих оттенках. Известно, что синие тона оказывают успокаивающее влияние на организм человека, помогают справиться с высоким давлением или температурой.

Синий цвет также стимулирует логическое мышление, в том числе и анализ действий, человек с помощью него становится более дисциплинированным.

9.8 Вывод

В данной главе рассмотрено применение безопасности жизнедеятельности для улучшения условий труда разработчика. Целью разработки защищенной веб-платформы передачи сообщений в рамках раздела безопасности и экологичности является снижение психофизиологической нагрузки, а также снижение времени затрачиваемой на активность в течении рабочего дня, связанной с проектированием и обсуждением задач с другими разработчиками, занимаемой до 40% рабочего времени.

10 Технико-экономическое обоснование эффективности внедрения защищенной веб-платформы передачи сообщений в ПАО «GameTime»

10.1 Определение и оценка капитальных затрат при внедрении и переводе сотрудников на использование защищенной веб-платформы передачи сообщений (ЗВПС) «АУУА» ПАО «GameTime»

Требования к ЗВПС от организации:

- безопасность чатов;
- русский язык;
- установка на локальный сервер организации;
- кроссплатформенность;
- бесплатный доступ;
- открытый исходный код, позволяющий модернизировать ЗВПС под нужды организации;
- синхронизация уведомлений;
- администрирование чатов;
- чат, ориентированный на общение в компаниях.

В ходе ВКР была разработана ЗВПС «АУУА» для общения внутри организации, удовлетворяющая всем требованиям выше. Программный продукт является кроссплатформенным, может запускаться WEB версия на компьютерах с OS Windows, Mac OS, и Linux. Для удобства нестационарного использования ЗВПС доступно мобильное приложение на OS Android. APK файл идет в наборе ЗВПС «АУУА».

Основная часть затрат – локальные серверы организации, которые гарантируют отсутствие блокировок и потерю данных в следствие санкций недружественных государств, что соответствует основным принципам информационной безопасности: конфиденциальности, целостности и доступности. Для запуска на мобильных устройствах ЗВПС необходимы устройства на OS андроид API level не меньше 29 (Android 4.4 Kitkat),

максимальный поддерживаемый API level 32 (Android 12 Snow Cone). Информация по стоимости технического оснащения бралась с интернет-магазина «Ситилинк». Оценка капитальных затрат при внедрении ЗВПС «АУУА» в ПАО «GameTime» представлена в таблице 10.1.

Таблица 10.1 – Оценка капитальных затрат при внедрении ЗВПС «АУУА» в ПАО «GameTime»

| № п/п | Статья затрат | Краткое описание | Оценка затрат, руб. |
|----------|--|---|------------------------|
| 1 | Техническое оснащение | | |
| 1.1 | Сервер Веб приложения | Сервер Dell PowerEdge T40 1xE-2224G 1x8GbUD x3 1x1Tb 7.2K 3.5" SATA RW 1G 1P 1x290W 1Y NBD Cabled | 163000 |
| 1.2 | Сервер БД в отказоустойчивом исполнении | Сервер Dell PowerEdge T40 1xE-2224G 1x8GbUD x3 1x1Tb 7.2K 3.5" SATA RW 1G 1P 1x290W 1Y NBD Cabled | 185000 |
| 2 | Работы на старте | | |
| 2.1 | Затраты на внедрение тех. средств (стоимость работ). | Внедрение и настройка системы, пусконаладочные работы и подключение к серверу веб-приложения; Настройка и подключение основных модулей. | 20000 |
| 2.2 | Затраты на процессное обеспечение (стоимость работ) | Определение ключевых ролей, сроков подключения, типовых настроек; Разработка регламента расследования и реагирования на инциденты ИБ; Разработка инструкций оператора/аналитика/пользователя. | 35000 |
| Итого | | | 403 000 |

10.2 Определение и оценка операционных затрат при внедрении ЗВПС в ПАО «GameTime»

Определим требуемое количество специалистов для группы обслуживания ЗВПС в ПАО «GameTime». Для этого определим плановую трудоемкость.

Плановая трудоемкость T , чел. дн. определяется по формуле:

$$T = t_i \cdot N_i, \quad (10.1)$$

где t_i – среднее время обработки одного события, ч.;

N_i – средний поток событий.

Ожидаемый поток событий примем равным 1500. Среднее время обработки одного события около 45 мин., что составляет 0,75 ч. или 0,09375 чел. дн., тогда согласно формуле (10.1):

$$T = 0,09375 \cdot 1500 = 140,625 \text{ чел. дн.}$$

Количество специалистов C определяется как:

$$C = \frac{T}{247 \cdot K_{ПЗ}}, \quad (10.2)$$

где T – плановая трудоемкость группы, чел. дн.;

247 – количество рабочих дней в году 1 специалиста;

$K_{ПЗ}$ – коэффициент продуктивной загрузки.

При коэффициенте продуктивной загрузки в 0,7 согласно формуле (10.2) получим:

$$C = \frac{140,625}{247 \cdot 0,7} = 1 \text{ чел.}$$

Для пятидневного режима одного специалиста вполне достаточно. Однако для эффективной работы организации необходим запасной сотрудник на непредвиденные случаи, по типу: болезнь, смерть, неотложные обстоятельства, в связи с чем предлагается к найму второй сотрудник.

Определим основные и дополнительные статьи затрат для группы обслуживания ЗВПС. Данные по заработной плате системного администратора брались с сайта компании интернет-рекрутинга «HeadHunter». Оценка операционных затрат группы обслуживания ЗВПС представлена в таблице 10.2.

**Таблица 10.2 – Оценка операционных затрат группы обслуживания
ЗВПС**

| № п/п | Основная статья затрат | Системный администратор |
|----------|---|----------------------------|
| 1 | Системный администратор, обслуживающий ЗПВС (з/п одного сотрудника в месяц) | 50 000 |
| 2 | НДФЛ, 13 % от з/п | 6 500 |
| 3 | Взносы в пенсионный фонд, соц. страх., ФМС, страх. от НС и ПЗ (22%, 2,9%, 5,1%, 0,5%) от з/п | 15 250 |
| 4 | Стоимость обучения (из расчета 10% от з/п) | 5 000 |
| 5 | Затраты на поиск персонала HR (из расчета 5% от з/п) | 2 500 |
| 6 | Организация и обслуживание рабочих мест: аренда помещения, компьютерная техника, электричество, бэкофис и пр. (из расчета 25% от з/п) | 12 500 |
| 7 | Премии, бонусы, оплата переработок (из расчета 20% от з/п) | 10 000 |
| 8 | Социальный пакет, ДМС и пр. (стоимость в год) | 10 000 |
| 9 | Итого стоимость сотрудника для компании в месяц: | 111 750 |
| 10 | Итого стоимость сотрудника для компании в год | 1 341 000 |

Оценка операционных затрат при сопровождении ЗВПС «АУУА» в
ПАО «GameTime» представлена в таблице 10.3.

**Таблица 10.3 – Оценка операционных затрат при сопровождении
ЗВПС «АУУА» в ПАО «GameTime»**

| № п/п | Статья затрат | Количество специалистов | Годовая оценка затрат, руб. |
|----------|--|----------------------------|-----------------------------------|
| 1 | Техническая поддержка ЗПВС: базовые консультации, восстановление работоспособности (стоимость работ) | 1 | 50 000 |
| | Команда обслуживания ЗПВС | | |
| 2 | Системный администратор | 2 специалиста | 2 682 000 |
| 3 | Прочие косвенные расходы | | 348 000 |
| 4 | Итого | | 3 080 000 |

Таким образом, общая сумма операционных затрат при сопровождении
ЗВПС «АУУА» в ПАО «GameTime» составила 3 080 000 рублей.

10.3 Расчет совокупной стоимости внедрения ЗВПС «АУУА» в ПАО «GameTime»

Совокупная стоимость внедрения ЗВПС в ПАО «GameTime» определяется по формуле:

$$ССВ = CAPEX + OPEX \cdot n, \quad (10.3)$$

где CAPEX – капитальные затраты, руб.;

OPEX – операционные затраты, руб.;

n – срок планирования, г.

Тогда совокупная стоимость внедрения ЗВПС на один год согласно формуле (10.3) составит:

$$ССВ = 403\,000 + 3\,080\,000 \cdot 1 = 3\,483\,000 \text{ руб.}$$

10.4 Расчет экономической эффективности и оценка целесообразности внедрения ЗВПС в ПАО «GameTime»

Капитальные затраты на внедрение ЗВПС составляют 403 000 рублей. Операционные в несколько раз больше и составляют 3 080 000 рублей.

Капитальные затраты проекта составляют 1% от годовой прибыли.

Вычислим экономическую эффективность защиты информации $\mathcal{E}_{\text{зи}}$, руб, за год:

$$\mathcal{E}_{\text{зи}} = \frac{P_{\text{сум}}}{\mathcal{Z}_{\text{зи}}}, \quad (10.4)$$

где $P_{\text{сум}}$ – сумма предотвращенного ущерба, руб.;

$\mathcal{Z}_{\text{зи}}$ – суммарные затраты на ЗИ, руб.

Суммарные затраты на ЗИ за год составили 3 483 000 руб. Учитывая среднюю годовую прибыль, сумма предотвращенного ущерба составляет 40 000 000 руб., тогда согласно (10.4):

$$\mathcal{E}_{\text{зи}} = \frac{40\,000\,000}{3\,483\,000} = 11,5 \text{ руб.}$$

Это означает, что каждый рубль, потраченный на ЗИ, позволил бы сохранить владельцу 11,5 рублей.

Расчетный срок окупаемости капитальных вложений T_p , дн. можно найти по формуле:

$$T_p = \frac{365}{\Delta_{3и}}, \quad (10.5)$$

Тогда согласно (10.5) имеем:

$$T_p = \frac{365}{11,5} = 31 \text{ дн.}$$

Следовательно, капитальные вложения окупятся в течение одного месяца.

10.5 Целесообразность внедрения ЗВПС «АУУА» в ПАО «GameTime»

С учетом сложившейся геополитической обстановки в мире, против России ведется информационная война со стороны недружественных стран. Почти все социальные сети и мессенджеры, которыми пользуются в Российской Федерации, являются зарубежными. Учитывая этот факт, все данные пользователей с России, проходящие по этим приложениям, хранятся и обрабатываются за рубежом. В связи с санкциями, почти все социальные сети заблокированы в России.

Мессенджеры пока доступны в использовании гражданами Российской Федерации, однако нет гарантий, что их не заблокируют, а также стоит учитывать, что некоторые из них достаточно небезопасны и собирают много пользовательских данных.

В связи с перечисленными выше предположениями, необходима разработка отечественной защищенной веб-платформы передачи сообщений, специализированный мессенджер для коммерческих организаций. Который обеспечит КЦД информации.

Стоимость покупки и внедрения ЗВПС является достаточно невысокой. Так как лицензия на ПО «АУУА» свободная, основные затраты составляют облуживание и разворачивание ПО, приобретение серверов, с чем столкнется кампания как при разработке ЗВПС самостоятельно, так и используя готовое решение, которое отвечает всем требованиям и имеет открытый исходный код, позволяющий модернизировать ПО. Последний вариант выгоднее так как нет необходимости в найме команды разработчиков.

Разработка собственной ЗВПС имеет значение только при необходимости использования других технологий разработки: другие технологии шифрования и языки программирования. Однако «АУУА» обеспечивает необходимый функционал, а также защищенность данных.

Заключение

В ходе выпускной квалификационной работе были проанализированные угрозы безопасности веб-приложений. На основе данных материалов были определены основные угрозы и определены требования к разрабатываемому приложению.

Также, в процессе выполнения данной работы были выполнены следующие задачи:

- анализ способов аутентификации;
- выбор оптимального способа аутентификации для разрабатываемой системы;
- проектирование и создание БД;
- реализация системы аутентификации;
- реализация второго фактора аутентификации по одноразовому паролю на почту;
- хеширование паролей;
- реализация защиты от НСД при атаке полным перебором;
- были определены требования к сложности пароля, которые можно менять в настройках приложения;
- и иные сопутствующие задачи.

И хотя данная разработанная система не идеальна и имеет свои недостатки, в плане защиты данных она не уступает проектам, над которыми трудятся команды крупных IT-компаний. Однако, благодаря направленности разработанного решения, вполне может занять свою нишу.

Список использованных источников

- 1 ГОСТ Р 51241-2008 Защита информации. Уязвимости информационных систем. Классификация уязвимостей информационных систем;
- 2 ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования;
- 3 ГОСТ Р 58833-2020 Защита информации. Идентификация и аутентификация. Общие положения;
- 4 ГОСТ Р 51904-2002 Программное обеспечение встроенных систем. Общие требования к разработке и документированию;
- 5 У мессенджера Signal проблемы: незашифрованные сообщения на диске и ключи шифрования в открытом виде [Электронный ресурс] // Хакер: журнал. URL: <https://xakep.ru/2018/10/24/signal-flaws-2/> (дата обращения: 12.06.2022 г.);
- 6 Служба безопасности Яндекс Еды сообщила об утечке информации [Электронный ресурс] // новости Яндекса: сайт. URL: https://yandex.ru/company/services_news/2022/01-03-2022 (дата обращения: 12.06.2022 г.);
- 7 OWASP TOP-10: практический взгляд на безопасность веб-приложений [Электронный ресурс] // Habr: сайт URL: <https://habr.com/ru/company/simplepay/blog/258499/> (дата обращения: 17.05.2022 г);
- 8 OWASP Топ 10 – 2021 [Электронный ресурс] // Double#: сайт URL: <https://2sharp.pro/infosec/92-owasp-top-10-2021.html> (дата обращения: 17.05.2022 г);
- 9 The ‘most serious’ security breach ever is unfolding right now. Here’s what you need to know. [Электронный ресурс] // The Washington Post: газета. URL: <https://www.washingtonpost.com/technology/2021/12/20/log4j-hack-vulnerability-java/> (дата обращения: 13.06.2022 г.);

10 Живицкая, Е. Н. Системный анализ и проектирование информационных систем: учебно-метод. пособие для студентов специальности I-40 01 02-02 «Информ. системы и технологии в экономике» / Е. Н. Живицкая, О. П. Едемская. – Минск: БГУИР, 2005. – 59 с.: ил.;

11 Иванов Вадим Вадимович, Лубова Елена Сергеевна, Черкасов Денис Юрьевич Аутентификация и авторизация // Проблемы Науки. 2017. №2 (84);

12 Блог компании DataArt. Обзор способов и протоколов аутентификации в веб-приложениях - [Электронный ресурс] // Habr: сайт– URL: <https://habr.com/ru/company/dataart/blog/262817/> (дата обращения 03.05.2022);

13 Блог компании VK. Как ты реализуешь аутентификацию, приятель? - [Электронный ресурс] // Habr: сайт – URL: <https://habr.com/ru/company/vk/blog/343288/> (дата обращения 03.05.2022);

14 M. Jones, J. Bradley, N. Sakimura, «Request for Comments: 7519» // Internet Engineering Task Force, 2015;

15 Justin Richer, Antonio Sanso. OAuth 2 in Action, 2017;

16 Dan Arias. Adding Salt to Hashing: A Better Way to Store Passwords – [Электронный ресурс] // auth0: сайт – URL: <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (дата обращения: 02.05.2022 г.);

17 SQL инъекции. Проверка, взлом, защита [Электронный ресурс] // Habr: сайт– URL: <https://habr.com/ru/post/130826/> (дата обращения: 09.06.2022 г.);

18 Определение безопасности данных [Электронный ресурс] // Oracle: сайт – URL: <https://www.oracle.com/cis/security/database-security/what-is-data-security/> (дата обращения: 09.06.2022 г.);

19 Обеспечение безопасности базы данных PostgreSQL [Электронный ресурс] // Habr: сайт– URL: <https://habr.com/ru/post/550882/> (дата обращения 03.05.2022);

20 Документация PostgreSQL и Postgres Pro [Электронный ресурс] // PostgresPro: сайт <https://postgrespro.ru/docs/> (дата обращения: 09.06.2022 г.);

21 Что такое центр сертификации? [Электронный ресурс] // SQLcertificate: сайт – 2017. – №1. – URL: <https://sslcertificate.ru/faq/what-is-certificate-authority.html> (дата обращения: 16.06.2022)

22 Сервер Dell PowerEdge T40 [Электронный ресурс] // Ситилинк: российская сеть магазинов URL: <https://www.citilink.ru/product/server-dell-poweredge-t40-1xe-2224g-1x8gbud-x3-1x1tb-7-2k-3-5-sata-rw-1445064/> (дата обращения: 11.06.2022 г.)

23 Вакансии на работу системным администратором [Электронный ресурс]: HeadHunter: компания интернет-рекрутинга. URL: <https://krasnodar.hh.ru/catalog/informacionnye-tehnologii-internet-telekom/sistemnyj-administrator> (дата обращения: 11.06.2022 г.)

Приложение А.

Листинг программы

Листинг 1. Файл конфигурации «config.py»:

```
import os
import secret
from datetime import timedelta

app_dir = os.path.abspath(os.path.dirname(__file__))

class BaseConfig:
    SECRET_KEY = os.environ.get('SECRET_KEY') or secret.SECRET_KEY
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    ALLOWED_HOSTS = ['https://localhost:8080/', 'https://127.0.0.1:8080/']
    CORS_HEADERS = 'Content-Type'

    # Config Flask-Mail
    MAIL_USERNAME = os.environ.get("MAIL_USERNAME") or secret.MAIL_USERNAME
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD') or secret.MAIL_PASSWORD
    MAIL_SERVER = 'smtp.mail.ru'
    MAIL_PORT = 2525
    MAIL_USE_TLS = True
    MAIL_DEFAULT_SENDER = MAIL_USERNAME

    # settings of fields accounts.models.User
    NAME_MIN_LENGTH = 6
    OTP_TRY_COUNT = 3
    USERNAME_MIN_LENGTH = 6

    # settings of field 'password' in accounts.models.User
    PASSWORD_MIN_LENGTH = 6
    PASSWORD_INCLUDES_SPECIAL_SYMBOLS = True
    PASSWORD_INCLUDES_LOWERCASE = True
    PASSWORD_INCLUDES_UPPERCASE = True
```

```

PASSWORD_INCLUDES_DIGITS = True
ALLOWED_SPECIAL_SYMBOLS = '!#%+~*/<=>?@[^_|\~'

# JWT
JWT_ISSUER_NAME = 'backend_server'
JWT_AUDIENCE_LIST = ['frontend_client', 'mobile_client']
JWT_IDENTITY_CLAIM = "username"
JWT_ACCESS_TOKEN_EXPIRES = timedelta(minutes=30)
JWT_REFRESH_TOKEN_EXPIRES = timedelta(days=30)

class Config(BaseConfig):
    DEBUG = True
    DATABASE_NAME = os.environ.get("DATABASE_NAME")
    DATABASE_USER = os.environ.get("DATABASE_USER")
    DATABASE_USER_PASSWORD = os.environ.get("DATABASE_USER_PASSWORD")
    DATABASE_ROOT = os.environ.get("DATABASE_ROOT")
    DATABASE_ROOT_PASSWORD = os.environ.get("DATABASE_ROOT_PASSWORD")
    SQLALCHEMY_DATABASE_URI = f"postgresql://{DATABASE_USER}:{DATABASE_
USER_PASSWORD}@localhost:5432/{DATABASE_NAME}"

```

Листинг 2. Файл инициализации «__init__.py»:

```

from flask import Flask
from flask_cors import CORS
from flask_migrate import Migrate
from flask_socketio import emit

from app.accounts import accounts
from app.accounts.my_jwt import MyJWT
from app.accounts.models import User
from app.chats.models import Chat, Message
from app.common import db, mail, socketio

from app.chats import chats

def create_app():

```

```

app = Flask(__name__)
app.config.from_object('config.Config')
app.debug = app.config.get("DEBUG")
cors = CORS(
    app,
    supports_credentials=True,
    resources={
        r'/': {'origins': app.config.get("ALLOWED_HOST", '*')},
        r'/api/v1/*': {'origins': app.config.get("ALLOWED_HOST", '*')},
    })
app.register_blueprint(accounts)
app.register_blueprint(chats)
db.init_app(app)
migrate = Migrate(app, db)
mail.init_app(app)
socketio.init_app(app,
    cors_allowed_origins=app.config.get("ALLOWED_HOST", '*'),
    async_mode='threading')
return socketio, app

```

Листинг 3. Модель пользователя «accounts/models.py»:

```

from datetime import datetime
from ..common import db
from .utils import HashPassword, otp_try
from .validator import UserValidator

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=False)
    messages = db.relationship('Message', backref='user', lazy='dynamic')
    otp_number = db.Column(db.Integer)
    otp_try = db.Column(db.Integer, default=0)
    password_hash = db.Column(db.String(100), nullable=False)
    password_salt = db.Column(db.String(100))

```

```

created_on = db.Column(db.DateTime(), default=datetime.utcnow)
updated_on = db.Column(
    db.DateTime(),
    default=datetime.utcnow,
    onupdate=datetime.utcnow
)

def __init__(self, username, email, password):
    self.set_username(username)
    self.set_email(email)
    self.set_password(password)

@classmethod
def authenticate(cls, **kwargs):
    username = kwargs.get('username')
    password = kwargs.get('password')
    if not username or not password:
        return None

    user = cls.query.filter_by(username=username).first()
    if not user or not HashPassword.check_hash(user.password_hash, password,
user.password_salt):
        return None
    return user

def __repr__(self):
    return "<{}: {}>".format(self.id, self.username)

def to_dict(self):
    return dict(id=self.id, name=self.name, username=self.username, email=self.email)

def set_username(self, username):
    if UserValidator.is_valid_username(username):
        self.username = username

```

```

def set_email(self, email):
    if UserValidator.is_valid_email(email):
        self.email = email

def set_password(self, password):
    if UserValidator.is_valid_password(password):
        self.password_salt = HashPassword.get_random_salt()
        self.password_hash = HashPassword.get_hash(password, self.password_salt)

def set_otp_number(self, otp_number):
    if UserValidator.is_valid_otp_number(otp_number):
        self.otp_number = otp_number

def get_otp_number(self):
    return otp_try(self)

def clear_otp_try(self):
    self.otp_try = 0

def get_email(self):
    return self.email

```

Листинг 4. Класс токенов «my_jwt.py»:

```

import datetime
import jwt
from functools import wraps
from flask import make_response, request
from flask import current_app as app
from flask.wrappers import Response
from app.accounts.models import User

class MyJWT:
    @staticmethod
    def encode_token(username: str, token_type: str = "access", ) -> str:
        """Create token by username"""

```



```

if token_type not in ['access', 'refresh']:
    token_type = 'access'
try:
    payload = {
        'iat': datetime.datetime.utcnow(),
        'exp':
            datetime.datetime.utcnow() +
            app.config.get(f'JWT_{token_type.upper()}_TOKEN_EXPIRES',
datetime.timedelta(minutes=5)),
        'username': username,
    }
    if app.config.get('JWT_ISSUER_NAME', None):
        payload['iss'] = app.config.get('JWT_ISSUER_NAME')
    if app.config.get('JWT_AUDIENCE_LIST', None):
        payload['aud'] = app.config.get('JWT_AUDIENCE_LIST')

    return jwt.encode(
        payload,
        app.config.get('SECRET_KEY', ''),
        headers={"typ": token_type, },
        algorithm='HS256'
    )
except jwt.InvalidAlgorithmError:
    print("InvalidAlgorithmError")
    raise jwt.InvalidAlgorithmError

@staticmethod
def decode_token(token: str, token_type: str = "access", ) -> dict:
    """Check token"""
    if jwt.get_unverified_header(token).get("typ", "") == token_type:
        kwargs = {}
        if app.config.get('JWT_AUDIENCE_LIST', None):
            kwargs['audience'] = app.config.get('JWT_AUDIENCE_LIST')
        if app.config.get('JWT_ISSUER_NAME', None):
            kwargs['issuer'] = app.config.get('JWT_ISSUER_NAME')

```

```

        payload = jwt.decode(token, app.config.get('SECRET_KEY', ''), algorithms="HS256",
**kwargs)
        if app.config.get(
            "JWT_IDENTITY_CLAIM",
            None
        ) and app.config.get("JWT_IDENTITY_CLAIM") in payload:
            return payload
        else:
            raise jwt.InvalidTokenError

```

@staticmethod

```

def jwt_required(token_type="access"):
    """authenticate request"""
    def wrapper(func):
        @wraps(func)
        def inner(*args, **kwargs):
            try:
                token = MyJWT.get_token_from_request(token_type)
                if token:
                    MyJWT.decode_token(token)
                    return func(*args, **kwargs)
                else:
                    return make_response({'message': 'token is missing'}, 401)
            except jwt.ExpiredSignatureError:
                return make_response({'message': 'token is expired'}, 401)
            except (jwt.InvalidTokenError, jwt.DecodeError) as e:
                return make_response({'message': 'token is invalid'}, 401)
            return inner
        return wrapper

```

@staticmethod

```

def get_username_from_jwt(token: str, token_type: str = 'access') -> str:
    return MyJWT.decode_token(token, token_type=token_type).get('username')

```

@staticmethod

```

def get_current_user(token_type: str = 'access') -> User:
    token = MyJWT.get_token_from_request(token_type)
    username = MyJWT.get_username_from_jwt(token, token_type)
    return User.query.filter_by(username=username).first()

@staticmethod
def get_token_from_request(token_type='access'):
    # From header
    token = request.headers.get('Authorization', None)
    if token:
        return token
    # From json
    data = request.get_json()
    if data:
        token = data.get(app.config.get(f'JWT_{token_type.upper()}_TOKEN_COOKIE_NAME',
None))
    if token:
        return token

```

Листинг 5. Дополнительные функции «utils.py»:

```

import secrets
from hashlib import sha256
from flask_mail import Message
from flask import current_app as app
from .custom_error import OTPTryError
from app.common import db, mail

class HashPassword:
    @staticmethod
    def get_random_salt() -> str:
        return secrets.token_hex(8)

    @staticmethod
    def get_hash(password: str, salt: str) -> str:

```

```

        password_sting = password + salt
        return sha256(password_sting.encode()).hexdigest()

    @staticmethod
    def check_hash(password_hash: str, password: str, salt: str) -> bool:
        return HashPassword.get_hash(password, salt) == password_hash

class TwoFactorAuthentication:
    @staticmethod
    def get_OTP_number():
        return int("".join(secrets.choice("0123456789") for i in range(6)))

    @staticmethod
    def send_OTP_number(email, otp_number):
        msg = Message("Verification code", recipients=[f'{email}'])
        msg.html = \
            f"<h1>Verification code</h1>" \
            f"<h5>An attempt was made to log into your account</h5>" \
            f"<p>Your code: <strong>{otp_number}</strong> </p>"
        if not app.config.get("DEBUG"):
            mail.send(msg)

    @staticmethod
    def authenticate_with_email(user):
        try:
            user.set_otp_number(TwoFactorAuthentication.get_OTP_number())
            TwoFactorAuthentication.send_OTP_number(user.get_email(), user.otp_number)
            user.clear_otp_try()
            db.session.add(user)
            db.session.commit()
        except Exception as e:
            raise Exception(f'{e}, \n error authenticate_with_email')

    def otp_try(user):
        if user.otp_try < app.config.get("OTP_TRY_COUNT", 3):

```

```

        user.otp_try += 1
        db.session.add(user)
        db.session.commit()
        return user.otp_number
    else:
        raise OTPTryError

```

Листинг 6. Класс проверки полей модели пользователя «validator.py»:

```

from re import match as re_match
from string import ascii_lowercase, ascii_uppercase, digits
from flask import current_app as app

class UserValidator:
    @staticmethod
    def is_valid_username(username: str) -> bool:
        if len(username) >= app.config.get("USERNAME_MIN_LENGTH", 0):
            return True
        else:
            return False

    @staticmethod
    def is_valid_email(email: str) -> bool:
        if re_match(r"^[A-Za-z0-9\.\_+]+\@[A-Za-z0-9\.\_]+\.[a-zA-Z]*$", email):
            return True
        else:
            return False

    @staticmethod
    def __is_valid_password_length(password: str) -> bool:
        is_valid_password_length = len(password) >=
app.config.get('PASSWORD_MIN_LENGTH', 0)
        if is_valid_password_length:
            return True
        else:
            return False

```

```
@staticmethod
def __is_valid_password_lowercase(password) -> bool:
    if app.config.get("PASSWORD_INCLUDES_LOWERCASE", False):
        for char in password:
            if char in ascii_lowercase:
                return True
        return False
    else:
        return True
```

```
@staticmethod
def __is_valid_password_uppercase(password) -> bool:
    if app.config.get("PASSWORD_INCLUDES_UPPERCASE", False):
        for char in password:
            if char in ascii_uppercase:
                return True
        return False
    else:
        return True
```

```
@staticmethod
def __is_valid_password_digit(password) -> bool:
    if app.config.get("PASSWORD_INCLUDES_DIGITS", False):
        for char in password:
            if char in digits:
                return True
        return False
    else:
        return True
```

```
@staticmethod
def __is_valid_password_special_symbols(password) -> bool:
    if app.config.get("PASSWORD_INCLUDES_SPECIAL_SYMBOLS", False):
        for char in password:
```

```

        if char in app.config.get("ALLOWED_SPECIAL_SYMBOLS"):
            return True
        return False
    else:
        return True

    @classmethod
    def is_valid_password(cls, password) -> bool:
        return all([
            cls.__is_valid_password_length(password),
            cls.__is_valid_password_lowercase(password),
            cls.__is_valid_password_uppercase(password),
            cls.__is_valid_password_digit(password),
            cls.__is_valid_password_special_symbols(password),
        ])

    @staticmethod
    def is_valid_otp_number(otp_number) -> bool:
        return 100000 <= int(otp_number) < 1000000

    @staticmethod
    def is_valid_username_and_password(username, password) -> bool:
        return all([
            UserValidator.is_valid_username(username),
            UserValidator.is_valid_password(password),
        ])

```

Листинг 7. Обработка запросов аутентификации «accounts/views.py»:

```

import json
from flask import make_response, request
from flask import current_app as app
from app.accounts import accounts
from app.accounts.models import User
from app.accounts.utils import TwoFactorAuthentication
from app.accounts.my_jwt import MyJWT

```

```

from app.accounts.custom_error import *
from app.accounts.validator import UserValidator
from ..common import db

@accounts.route('/login', methods=['post'])
def login():
    data = request.get_json()
    username = data.get('username', None)
    password = data.get('password', None)

    if not UserValidator.is_valid_username_and_password(username, password):
        return make_response({"Error": "Invalid username or password"}, 403)

    user = User.authenticate(username=username, password=password)

    if not user:
        return make_response({"Error": "Invalid username or password"}, 403)

    otp_number = data.get('OTPNumber', 0)

    if not otp_number:
        TwoFactorAuthentication.authenticate_with_email(user)
        return make_response({"Message": "OTP number was send on your email"}, 200)

    if not UserValidator.is_valid_otp_number(otp_number):
        return make_response({"Message": "OTP number number is invalid"}, 403)

    try:
        user_otp_number = user.get_otp_number()
    except OTPTryError as e:
        return make_response({"Error": f"OTP try error. New code was send on your email"}, 403)

    if not (user_otp_number == otp_number or app.config.get("DEBUG")):
        return make_response({"Error": "OTP number number is invalid"}, 403)

```



```

user.otp_try = 3
db.session.commit()

access_token = MyJWT.encode_token(user.username, token_type="access")
refresh_token = MyJWT.encode_token(user.username, token_type="refresh")

return make_response({'Message': 'Login successfully',
                      'access_token': access_token,
                      'refresh_token': refresh_token}, 200)

@accounts.route("/logout", methods=["DELETE"])
@MyJWT.jwt_required()
def logout():
    response = make_response({"Message": "logout successful"})
    return response

@accounts.route("/refresh", methods=['POST'])
@MyJWT.jwt_required(token_type="refresh")
def refresh():
    refresh_token = MyJWT.get_token_from_request(token_type="refresh")
    username = MyJWT.get_username_from_jwt(refresh_token, token_type='refresh')
    access_token = MyJWT.encode_token(username=username, token_type='access')
    return json.dumps({"Message": "token was refreshed",
                      'access_token': access_token,
                      'refresh_token': refresh_token})

@accounts.route("/user")
@MyJWT.jwt_required()
def get_user_info():
    return make_response(MyJWT.get_current_user(token_type='access').to_dict())

```

Листинг 8. Модели подсистемы чатов «chats/models.py»:

```

from datetime import datetime
from ..common import db

```

```

from ..accounts.models import User

users = db.Table('users',
    db.Column('user_username', db.String(20), db.ForeignKey('user.username')),
    db.Column('chat_id', db.Integer, db.ForeignKey('chat.id'))
)

class Chat(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(20), unique=True, nullable=False)
    # one Chat to many messages
    messages = db.relationship('Message', backref='chat', lazy='dynamic')
    # many USERS to many CHATS
    users_in_chat = db.relationship('User', secondary=users,
                                    backref=db.backref('chats', lazy='dynamic'))

    def __repr__(self):
        return f'Chat: {self.id}: {self.title}'

    @property
    def serialize(self):
        return {
            "id": self.id,
            "title": self.title
        }

class Message(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String(2500), nullable=False)
    created_on = db.Column(db.DateTime(), default=datetime.utcnow)
    # Foreign keys
    user_username = db.Column(db.String(20), db.ForeignKey('user.username'))
    chat_id = db.Column(db.Integer, db.ForeignKey('chat.id'))

    def __repr__(self):

```

```

        return f'MSG: {self.id}: {self.text}'

    @property
    def serialize(self):
        return {
            "id": self.id,
            "username": self.user_username,
            "chatId": self.chat_id,
            "text": self.text,
            "created": self.created_on.strftime("%H:%M:%S")
        }

```

Листинг 9. Обработка запросов подсистемы чатов «chats/models.py»:

```

from flask import jsonify
from flask_socketio import emit
from .models import Chat, Message
from ..accounts import MyJWT
from ..common import socketio, db
from app.chats import chats

@chats.route("/list", methods=['GET'])
@MyJWT.jwt_required()
def get_chats():
    user_chats = MyJWT.get_current_user().chats.all()
    return jsonify([chat.serialize for chat in user_chats])

@chats.route("/list/<title>", methods=['GET'])
@MyJWT.jwt_required()
def get_searched_chats(title: str):
    searched_chats = Chat.query.filter(Chat.title.contains(title)).all()
    return jsonify([chat.serialize for chat in searched_chats])

@chats.route("/<chat_id>", methods=['GET'])
@MyJWT.jwt_required()
def get_messages(chat_id: int):

```

```

chat = Chat.query.filter_by(id=chat_id).first()
if MyJWT.get_current_user() in chat.users_in_chat:
    return jsonify([msg.serialize for msg in chat.messages])
return jsonify({})

@chats.route("/<chat_id>", methods=['POST'])
@MyJWT.jwt_required()
def enter_in_chat(chat_id: int):
    if chat_id:
        chat = Chat.query.filter_by(id=chat_id).first()
        current_user = MyJWT.get_current_user()
        chat.users_in_chat.append(current_user)
        db.session.commit()

@chats.route("/<chat_id>", methods=['DELETE'])
@MyJWT.jwt_required()
def leave_from_chat(chat_id):
    if chat_id:
        chat = Chat.query.filter_by(id=chat_id).first()
        current_user = MyJWT.get_current_user()
        chat.users_in_chat.remove(current_user)
        db.session.commit()

@socketio.on("send message")
@MyJWT.jwt_required()
def send_message(payload: dict):
    chat_id = payload.get("chatId")
    current_user = MyJWT.get_current_user()
    chat = Chat.query.filter_by(id=chat_id).first()
    if current_user in chat.users_in_chat:
        text = payload.get("messageText")
        new_msg = Message(text=text, user_username=current_user.username, chat_id=chat_id)
        db.session.add(new_msg)
        db.session.commit()
        new_msg = Message.query.filter_by(

```

```
chat_id=chat_id,  
user_username=current_user.username,  
text=text  
)order_by(Message.id.desc()).first()  
emit("display new message", new_msg.serialize, broadcast=True)
```

Отчет о проверке на заимствования №1



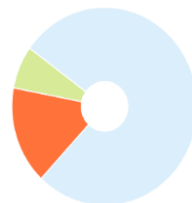
Автор: Семенов Владислав Романович
Проверяющий: Макарян Александр Самвелович
Организация: Кубанский государственный технологический университет
Отчет предоставлен сервисом «Антиплагиат» - <http://kubstu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 815
Начало загрузки: 20.06.2022 05:27:21
Длительность загрузки: 00:00:15
Имя исходного файла: 17-K-AC1 Семенов В.Р.
ВКР с титулами.pdf
Название документа: 17-K-AC1 Семенов В.Р.
ВКР с титулами.pdf
Размер текста: 130 кБ
Тип документа: Выпускная квалификационная работа
Символов в тексте: 133011
Слов в тексте: 16161
Число предложений: 1090

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 20.06.2022 09:37:06
Длительность проверки: 00:00:48
Комментарии: не указано
Поиск с учетом редактирования: да
Модули поиска: Перефразирования по коллекции издательства Wiley, Переводные заимствования издательства Wiley (RuEn), Издательство Wiley, Переводные заимствования по eLIBRARY.RU (EnRu), Переводные заимствования по Интернету (EnRu), Медицина, Интернет Плюс, Сводная коллекция ЭБС, Перефразирования по eLIBRARY.RU, Перефразирования по Интернету, Шаблонные фразы, Библиография, ИПС Адилет, Сводная коллекция РГБ, Переводные заимствования (RuEn), Диссертации НББ, СПС ГАРАНТ, Кольцо вузов, Патенты СССР, РФ, СНГ, eLIBRARY.RU, Цитирование, Переводные заимствования, СМИ России и СНГ, Модуль поиска "КубГТУ"



ЗАИМСТВОВАНИЯ

15,9%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

7,45%

ОРИГИНАЛЬНОСТЬ

76,65%