Министерство просвещения РФ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»

Институт математики, физики, информатики и технологий

Кафедра информатики, информационных технологий
и методики обучения информатике

ВЕБ-СЕРВИС ДЛЯ ЭЛЕКТРОННОГО ПОДПИСАНИЯ ДОКУМЕНТОВ

Выпускная квалификационная работа
бакалавра по направлению подготовки
09.03.02 – Информационные системы и технологии

Исполнитель: Бабре Антон Иванович
Обучающийся группы ИСиТ
1931z
_____

Руководитель: Арбузов С.С.
к.п.н., доцент кафедры
ИИТиМОИ
_____

Допущено к защите
Зав. кафедрой
«__»_____ 2024 г. _____

Екатеринбург – 2024

Реферат

Бабре А.И. ВЕБ-СЕРВИС ДЛЯ ЭЛЕКТРОННОГО ПОДПИСАНИЯ ДОКУМЕНТОВ, выпускная квалификационная работа: 57 стр., рис 47, табл. 1, библ. 56 назв., приложений 3.

Ключевые слова: веб, сервис, подписание, python, redis, nginx.

Предмет разработки – веб-сервис электронного подписания документов.

Цель работы – спроектировать и разработать веб-сервис, позволяющий подписывать электронные или отсканированные документы в формате pdf факсимильной подписью через веб-браузер.

В работе описаны результаты проектирования и разработки веб-сервиса электронного подписания документов, схемы работы сервиса и результаты работы с сервисом.

Система выполнена в среде разработки Microsoft Visual Studio Code с использованием языков программирования Python и JavaScript, базы данных Redis.

Система может быть использована для подписания существующих документов в электронном виде факсимильной подписью. Обмен документами возможен между двумя и более зарегистрированными пользователями.

Оглавление

Введение

В современном мире доля электронного документооборота очень стремительно растет. С каждым месяцем появляются новые сервисы для работы с такими документами, улучшается и дорабатывается функционал существующих сервисов. Но несмотря на столь стремительный рост популярности электронных документов большинство людей продолжают использовать бумажный документооборот.

Актуальность разработки веб-сервиса электронного подписания документов заключается в нескольких аспектах:

1. Ускорение процесса подписания документов. Вместо того чтобы высылать документы на подписание по почте или курьером.

2. Увеличение комфорта пользователя. Простота использования веб-сервиса позволяет упростить процесс подписания документов для пользователя. Ему не нужно стоять в очередях и тратить время на ожидание визы или подписи документов.

3. Улучшение безопасности. Электронная подпись позволяет установить уверенность в том, что документ был подписан соответствующим лицом и не был изменен после подписания.

4. Экологический эффект. Сокращение использования бумажных документов и очередей уменьшает проблему экологического загрязнения и способствует экономическому росту.

Разработка веб-сервиса электронного подписания документов необходима для упрощения и ускорения процесса подписания документов. Это позволит устранить необходимость в бумажных документах, которые могут быть легко утеряны или подделаны, а также сократить время на передачу и обработку документов. Веб-сервис электронного подписания документов позволит сторонам

быстро и безопасно подписать документы и обеспечить сохранность документов в цифровом формате.

Предмет разработки – веб-сервис для электронного подписания документов.

Цель работы – спроектировать и разработать веб-сервис, позволяющий подписывать электронные или отсканированные документы в формате pdf факсимильной подписью через веб-браузер.

Цель создания веб-сервиса электронного подписания документов заключается в облегчении процесса подписания документов, повышении его эффективности, ускорении перевода бумажных документов в электронный формат, а также в обеспечении высокой степени безопасности и надежности подписания документов. Веб-сервисы электронного подписания документов позволяют значительно ускорить обработку документов и снизить ошибки при их подписании, что существенно повышает производительность работы компаний и множество других сфер деятельности.

Для достижения цели работы, были поставлены следующие задачи:

1. Произвести анализ информационных источников, посвящённых основами электронного документооборота, выделить их достоинства и недостатки существующих систем для подписания электронных документов.
2. Проанализировать методы и средства разработки систем электронного документооборота.
3. Подготовить техническое задание на разработку веб-сервиса электронного подписания документов.
4. Спроектировать и реализовать веб-сервис электронного подписания документов.
5. Подготовить техническую документацию в виде руководства пользователя.

Глава 1. Теоретико-аналитическая часть

1.1 Теоретические основы электронного документооборота

С очень стремительным развитием информационных технологий в современном мире возрос и спрос на системы электронного документооборота. На текущий момент даже в самых маленьких организациях внедряется электронный документооборот и ведётся параллельно с классическим, бумажным документооборотом.

В рамках данной работы на основе анализа информационных источников [46, 47, 48] под электронным документооборотом будем понимать систему обмена и управления электронными документами в рамках организации, несколькими организациями, а последнее время даже физическими лицами.

Электронный документооборот основан на целом ряде принципов и концепций, которые в свою очередь определяют его основные характеристики и преимущества:

1. Электронная подпись. Одним из ключевых аспектов электронного документооборота является использование электронной подписи. Она обеспечивает юридическую значимость документов, подписанных электронным способом, и обеспечивает их надежность и целостность.

2. Электронное хранение. В электронном документообороте предполагается возможность электронного хранения и обмена документами без необходимости использования бумажных носителей. Это позволяет сократить расходы на хранение документов и повысить доступность информации.

3. Интеграция с бизнес-процессами. Одной из целей электронного документооборота является автоматизация и оптимизация бизнес-процессов, связанных с обменом документами. В электронном документообороте предполагается осуществление интеграции систем управления документами с другими информационными системами предприятия.

4. Безопасность. В электронном документообороте уделяется внимание вопросам защиты информации от несанкционированного доступа, изменений и утечек. В этом контексте особенно важным является обеспечение безопасности при обмене документами между различными участниками процесса.

5. Стандартизация. Для эффективной реализации электронного документооборота в организации необходимо использование стандартизированных форматов документов и протоколов обмена информацией. Теория электронного документооборота предполагает использование таких стандартов как XML, PDF, EDI и др.

В целом, теоретические основы электронного документооборота включают в себя принципы безопасности, эффективности, автоматизации бизнес-процессов и стандартизации обмена информацией. Эти основы позволяют организациям совершенствовать свои процессы управления документами и повышать эффективность их деятельности.

Системы электронного документооборота предназначаются для решения определённых задач. Поэтому такие системы можно разделить на следующие типы:

1. Системы регистрационного типа. Такие системы позволяют учитывать, вести реестр электронных документов.

2. Системы формирующие документы в электронном виде. Такие системы позволяют создавать документы, которые впоследствии будут использоваться в электронном документообороте.

3. Электронные архивные системы. Такие системы специально разработаны для хранения и управления архивными документами. Они обеспечивают организацию и быстрый доступ к документам, а также возможности автоматизации процессов архивации и управления жизненным циклом документов.

4. Системы обрабатывающие документы в электронном виде. Такие системы позволяют выполнять комплекс действий с электронными документами, например создание и регистрация электронного документа.

Основной элемент электронного документооборота — это файл, который создается с использованием компьютерной техники, определенных программ и подписывается электронной подписью. Он оформляется согласно всем правилам и с точки зрения юридической эквивалентен своему бумажному варианту. Только электронные формы актов, счетов-фактур или договоров хранятся не в громоздких папках на пыльных стеллажах, а на компьютерных жестких дисках, CD или флешках, в локальной сети или в облачных хранилищах.

Внутри компании электронный документооборот может осуществляться двумя способами:

1. Первый вариант – обмен документами внутри компании или между контрагентами происходит через электронную почту, что не требует дополнительного оборудования или значительных финансовых затрат на специализированное программное обеспечение. Однако передача конфиденциальных документов через незащищенные каналы электронной почты может быть небезопасной и в некоторых случаях невозможной, например, государственные учреждения не принимают документы, отправленные по электронной почте.

2. Второй вариант предполагает подключение к системе электронного документооборота, которая обеспечивает обмен всеми типами документов, формализованными и неформализованными. В рамках этой системы документ проходит все этапы от создания и обработки до хранения и контроля исполнения, что помогает упорядочить и ускорить процессы, повысить эффективность работы. Подписание пакета документов с использованием системы электронного документооборота занимает всего несколько минут, и это можно сделать, не покидая рабочего места. Все действия контрагентов соответствуют регламенту операторов систем электронного документооборота, а передача документов осуществляется по защищенным каналам связи, что обеспечивает конфиденциальность и защиту от посторонних вмешательств.

На сегодняшний день разработка сервиса электронного подписания документов является особо значимой, перспективной и востребованной. Связано это

с постоянным увеличением популярности электронного документооборота у людей и электронный документооборот постепенно вытесняет обыкновенный, бумажный документооборот. С постепенным ростом интереса к электронным документам возрастает и потребность в сервисах и программах способных обеспечить быструю, качественную и простую работу с электронными документами.

На сегодняшний день можно выделить несколько главных сервисов, которые позволяют людям работать с электронными документами.

Контур Диадок — юридически значимый онлайн-документооборот между организациями. Данный сервис позволяет оперативно формировать и обмениваться документами с контрагентами.

К достоинствам Контур Диадок можно отнести:

1. Удобство использования. Сервис представляет широкий выбор инструментов для работы с электронными документами.

2. Возможность интеграции с другими сервисами. Сервис предоставляет возможность интегрироваться в различные программы для работы с документами, такими как 1С.

3. Конструктор документов. Сервис позволяет сформировать документ заполняя необходимые поля напрямую с сайта сервиса. При этом есть возможность загрузить документ, сформированный в другой программе или сервисе.

К недостаткам Контур Диадок можно отнести:

1. Стоимость. Использование сервиса может потребовать определенных затрат, особенно для малых и средних предприятий.

2. Зависимость от интернет-соединения. Для работы с сервисом требуется постоянный доступ к сети интернет.

DocuSign — это сервис, позволяющий загружать, отправлять на подписание, просматривать, подписывать и отслеживать статус документов. Делать это

можно как из интерфейса сайта, так и с помощью собственного приложения для мобильных устройств.

К достоинствам DocuSign можно отнести:

1. Удобство использования. Сервис имеет интуитивно понятный интерфейс для работы с документами.

2. Возможность работы на разных устройствах. Сервис предоставляет возможность пользователю скачать приложение на мобильное устройство и подписывать документы с него.

К Недостаткам DocuSign можно отнести:

1. Стоимость. Для некоторых пользователей стоимость подписки может быть неоправданно высокой.

2. Зависимость от интернет-соединения. Для работы с сервисом требуется постоянный доступ к сети интернет.

Таким образом, проанализировав основы электронного документооборота, можно сделать вывод о том, что это современный эффективный способ обработки документов и информации доступный не только организациям, но и физическим лицам. Электронный документооборот позволяет сократить время на обработку и передачу документов, уменьшить вероятность ошибок, повысить уровень безопасности и сохранности информации. Кроме того, он также способствует повышению производительности и улучшению взаимодействия между сотрудниками и партнерами.

1.2 Методы и средства разработки систем электронного документооборота

Для достижения результата разработки существуют два метода: каскадный (Waterfall) и гибкий (Agile). Неправильный выбор метода разработки, может отрицательно сказаться на продукте разработки.

Waterfall – запланированный и детализированный подход, где исполнитель продукта строго придерживается разработанному ранее плану. Agile – полная противоположность предыдущего метода, которая предполагает гибкость разработки с возможностью внесения изменений на каждом этапе разработки.

Достоинства каскадного метода:

1. Простота управления проектом. Каждый этап разработки проходит последовательно.

2. Прогнозируемость. Этапы разработки определены заранее, что облегчает планирование и контроль разработки.

3. Документация. Каждый этап разработки имеет свою документацию, что способствует надежному управлению процессом разработки.

Недостатки каскадного метода:

1. Отсутствие гибкости. Изменение требований могут замедлить процесс разработки.

2. Длительный срок разработки. Разработка зависит от завершения предыдущего этапа, что как правило приводит к задержкам в разработке.

3. Масштабные изменения. Изменения в требованиях могут потребовать полного пересмотра проекта.

Достоинства гибкого метода:

1. Гибкость. Легкость внесения изменений в ход проекта без значительной переработки.

2. Ориентация на клиента. Этот метод позволяет более тесное взаимодействие с заказчиком и быструю реакцию на его потребности.

3. Промежуточные результаты. Этот метод позволяет возможность получения рабочего продукта на ранний этапах разработки.

Недостатки гибкого метода:

1. Сложность управления проектами большого объёма.

2. Риск недостаточного контроля. Метод может привести к недостаточному контролю над процессом разработки.

3. Неполная документация. Метод предполагает минимальное количество документации, что может затруднить понимания проекта другими участниками.

Разобрав рассмотренные методы разработки, выделив их достоинства и недостатки можно сделать вывод, что для разработки небольшого проекта и небольшой командой наилучшим методом для разработки будет гибкий метод (Agile). При разработке веб-сервиса электронного подписания документов будет использован метод Agile. Сделан такой выбор был неспроста. Выбор этого метода обусловлен возможностью внесения необходимых изменений на любом этапе разработки и возможностью получения обратной связи от заказчика.

В современном мире информационных технологий существует весьма обширное разнообразие инструментов и языков программирования для разработки веб-сервиса. Выбор инструмента или языка программирования достаточно трудоёмкий процесс, который требует очень много времени. Неправильный выбор инструмента или языка программирования при использовании гибкого (Agile) метода разработки может оказаться причиной для написания проекта с нуля, даже если проект почти готов, а для реализации необходимой задумки в выбранном языке программирования или инструменте возможностей нет.

Разберём некоторые языки программирования и инструменты, которые используются для разработки веб-сервисов.

Для работы разработанного веб-сервиса потребуется веб-сервер, который будет отображать информацию в окне браузера пользователю. Apache и Nginx – это два популярных веб-сервера. Они способны обрабатывать большие объёмы трафика и обеспечивать высокую производительность, но у них есть различия:

1. Производительность: Nginx обычно считается более производительным сервером, чем Apache, особенно для обработки статических файлов и высоконагруженных сценариев.

2. Ресурсы: Nginx обычно потребляет меньше оперативной памяти и ресурсов процессора, чем Apache.

3. Конфигурация: Apache имеет более гибкую и понятную конфигурацию, в то время как Nginx использует более компактный и читаемый синтаксис конфигурации.

Для хранения таких данных как имена, пароли и другие учетные данные пользователей потребуется база данных, в которой эти данные будут храниться. В настоящее время существуют такие популярные базы данных как MySQL, PostgreSQL и Redis.

MySQL – это одна из самых популярных баз данных, которая хорошо подходит для средних и крупных проектов. Она поддерживает широкий набор функций и имеет отличную производительность при работе с небольшими и средними объемами данных. Однако, при работе с большими объемами данных, MySQL может столкнуться с проблемами производительности из-за своей структуры, что делает ее менее подходящей для аналитики и отчетности. В таких случаях, могут быть более подходящими альтернативами базы данных, специально оптимизированные для работы с большими объемами данных,

PostgreSQL – более мощная и расширяемая база данных, чем MySQL. Она имеет богатый набор функций и поддерживает хранилище процедур, что делает ее хорошим выбором для больших и сложных проектов. Благодаря своей открытой и расширяемой архитектуре, PostgreSQL позволяет разработчикам создавать собственные типы данных, индексы и функции, что делает ее идеальным выбором для проектов с уникальными требованиями.

Redis – это высокопроизводительная база данных, которая обычно используется для кэширования данных и обработки сообщений. Она быстро и просто масштабируется, что делает ее идеальным выбором для приложений, требующих

быстрого доступа к данным. От вышеперечисленных баз данных Redis отличается тем, что хранит данные в оперативной памяти, что с одной стороны уменьшает время обработки запросов, но с другой способно нарушить целостность данных при внезапном отключении питания сервера, на котором храниться эта база.

Таким образом, выбор между MySQL, PostgreSQL и Redis зависит от конкретных потребностей проекта. Если требуется простая и распространенная база данных, то MySQL подойдет. Если нужны сложные функции и хранилище процедур, то PostgreSQL будет лучшим выбором. В случае если необходима быстрая и простая масштабируемость, то Redis идеально подойдет.

Язык программирования при разработке проекта – является одним из наиболее важных элементов проекта. При выборе языка программирования следует чётко понимать его возможности. Для разработки веб-сайтов и веб-сервисов используются следующие популярные языки программирования: Python, Java, JavaScript и PHP.

Python – это универсальный язык программирования, который часто используется для разработки веб-приложений, анализа данных, научных вычислений и автоматизации задач. Он отличается простым и чистым синтаксисом, что делает его популярным среди начинающих программистов. Кроме того, этот язык программирования обладает большим количеством фреймворков. Фреймворк – это набор инструментов, библиотек и стандартов, которые используются для разработки программного обеспечения. Фреймворк предоставляет разработчикам готовые компоненты и функциональность, которые могут быть использованы для создания приложений, веб-сайтов и других программных проектов. Он облегчает работу разработчиков, ускоряет процесс создания приложений и обеспечивает стандартизацию кода [49].

Java – это язык программирования, который чаще всего используется для создания больших корпоративных приложений и веб-серверов. Он имеет богатую библиотеку классов и обеспечивает высокую стабильность и производительность. Java также известен своей платформонезависимостью, что означает, что

программы, написанные на Java, могут работать на любой платформе, которая поддерживает виртуальную машину Java. Это делает Java очень гибким и удобным выбором для разработчиков, которые хотят создавать приложения, которые могут работать на различных устройствах и операционных системах. Кроме того, Java поставляется с широким набором инструментов для управления памятью, безопасностью и многими другими аспектами программирования, что делает его отличным выбором для построения надежных и безопасных приложений [50].

JavaScript – это язык программирования, который широко используется для создания интерактивных веб-сайтов и веб-приложений. Он выполняется в браузере пользователя и обеспечивает динамическое изменение содержимого веб-страниц. JavaScript обеспечивает возможность создания различных функций, анимаций, обработчиков событий и многих других возможностей, которые делают веб-сайты более интерактивными и привлекательными для пользователей. Он также может использоваться для разработки серверных приложений с использованием платформы Node.js. JavaScript является одним из основных языков веб-разработки и часто используется в сочетании с HTML и CSS для создания полноценных веб-приложений. Кроме того, с появлением новых фреймворков и библиотек, JavaScript стал еще более мощным инструментом для разработки современных веб-приложений [51].

PHP – это язык программирования, который используется в основном для создания динамических веб-страниц. Он легко встраивается в HTML код и может быть использован для создания различных веб-приложений, таких как формы обратной связи, системы управления контентом, интернет-магазины, форумы и многое другое. Он работает на стороне сервера и взаимодействует с базами данных и другими технологиями, чтобы обеспечить создание динамических и интерактивных веб-сайтов. PHP широко поддерживается большинством хостинг-провайдеров и является одним из основных языков, используемых для создания динамических веб-сайтов. Он обладает обширной библиотекой функций для работы с различными типами данных, файлами, изображениями, базами данных и

другими аспектами веб-разработки. Кроме того, PHP является открытым исходным кодом, что позволяет разработчикам создавать собственные расширения и улучшать язык программирования в целом. Благодаря этому сообщество PHP постоянно растет и развивается, что делает его мощным инструментом для создания динамических веб-приложений [52].

Flask – это легковесный фреймворк для создания веб-приложений на языке программирования Python. Он предоставляет минимальный набор инструментов для разработки веб-приложений, оставляя разработчику большую свободу в выборе инструментов и библиотек для конкретной задачи. Flask широко используется для создания прототипов, небольших и средних веб-приложений, а также в качестве основы для API-серверов. Основные особенности Flask включают простоту использования, гибкость, расширяемость и обширное сообщество разработчиков и библиотек [53].

Фреймворк Bcrypt в Python предоставляет возможность хеширования паролей с использованием алгоритма Bcrypt, который является сильным и безопасным способом хранения паролей. Этот фреймворк предоставляет удобный API для хеширования паролей и их сравнения, что позволяет разработчикам легко интегрировать его в свои приложения.

React – это библиотека JavaScript для создания пользовательских интерфейсов, разработанная компанией Facebook. Она позволяет создавать интерактивные веб-приложения, обеспечивая удобное управление отображением данных на странице. React использует компонентный подход к построению интерфейса, разделяя приложение на множество маленьких, переиспользуемых компонентов, каждый из которых отвечает за свою часть пользовательского интерфейса. Это делает код более чистым, организированным и легким для поддержки. React стал одним из самых популярных фреймворков для разработки фронтенд-приложений благодаря своей гибкости, производительности и активному сообществу разработчиков [54].

Bootstrap – это популярный фреймворк для разработки веб-приложений и сайтов, он включает в себя готовые компоненты HTML, CSS и JavaScript для создания адаптивного и стильного пользовательского интерфейса. В частности, Bootstrap содержит множество готовых JavaScript компонентов, таких как модальные окна, всплывающие подсказки, аккордеоны, карусели, вкладки и многое другое. Он также включает в себя JavaScript плагины, такие как jQuery и Popper.js, которые облегчают работу с различными элементами пользовательского интерфейса. Использование Bootstrap JavaScript позволяет быстро и удобно создавать интерактивные элементы веб-приложений, что делает его популярным среди веб-разработчиков [55].

Redis – это открытая хранимая в памяти база данных, которая используется для хранения данных в формате ключ-значение. Он широко используется для кэширования данных и ускорения производительности приложений. Redis поддерживает различные типы данных, такие как строки, списки, множества, хэши и многое другое. Он также предоставляет различные операции для работы с этими типами данных, такие как добавление, удаление, обновление и поиск. Redis также поддерживает репликацию данных и кластеризацию для обеспечения высокой доступности и масштабируемости [56].

Большинство современных сайтов и сервисов состоят из двух основных частей: пользовательского интерфейса (фронтенда), предназначенного для взаимодействия с обычными пользователями, и внутренней технической части (бэкенда), которая обрабатывает запросы от фронтенда.

Фронтенд – это часть веб-приложения, которую видит и с которой взаимодействует пользователь. Она отвечает за отображение контента и интерактивность пользовательского интерфейса. Фронтенд обычно включает в себя HTML, CSS и JavaScript.

Бэкенд – это часть веб-приложения, которая отвечает за обработку данных, бизнес-логику и взаимодействие с базой данных. Бэкенд обычно включает в себя

серверное программное обеспечение, базы данных и другие серверные технологии.

Разработка фронтенда и бэкенда веб-сервиса очень тесно связаны и неотделимы друг от друга. Как правило они связаны по следующей схеме (см. рисунок 1):

1. Пользователь выполняет какие-либо действия во фронтенде. Например, нажимает кнопку.

2. Фронтенд отправляет информацию об совершённом действии в бэкенд.

3. Бэкенд обрабатывает полученную информацию. Например, формирует список пользователей в веб-сервисе при открытии пользователем выпадающего меню.

4. Бэкенд отправляет сформированную информацию в бэкенд.

5. Фронтенд отображает пользователю список пользователей веб-сервиса.

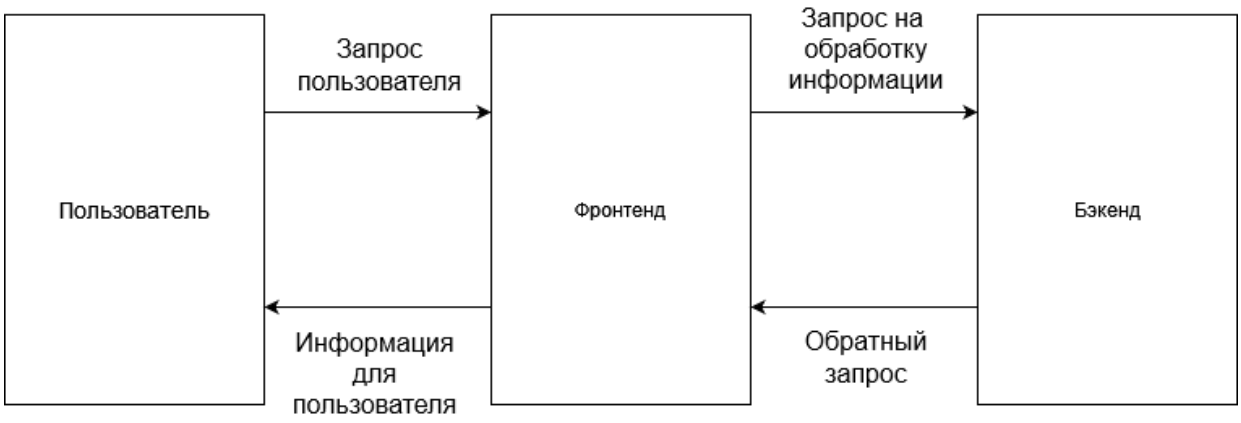


Рисунок 1. Схема взаимодействия фронтенда с бэкендом

Таким образом, выбор метода разработки и языка программирования являются одними из важных этапов перед разработкой сервиса. Эти решения могут существенно влиять на процесс разработки, стоимость, производительность и качество программного обеспечения. При выборе метода разработки и языка программирования следует учитывать требования проекта, квалификацию команды разработчиков, возможности интеграции с другими системами, поддержку и обновление языка программирования, а также экосистему инструментов и библиотек. В конечном итоге, правильный выбор метода разработки и языка программирования поможет обеспечить успешное выполнение проекта и достижение его целей.

1.3 Формализованное описание технического задания

Разработка веб-сервиса электронного подписания документов была составлена на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

1. Общие сведения.

1.1. Название организации-заказчика.

ФГБОУ ВО «Уральский государственный педагогический университет», Институт математики, физики, информатики и технологий, кафедра информационно-коммуникационных технологий в образовании.

1.2. Название продукта разработки.

Веб-сервис электронного подписания документов

1.3. Назначение продукта.

Для использования физическими лицами и организациями для подписания электронных документов.

1.4. Плановые сроки начала и окончания работ.

В соответствии с планом выполнения ВКР (01.09.2023 – 15.02.2024).

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки.

Данный продукт может использоваться в любой отрасли или секторе, где требуется подпись документа.

2.2. Характеристика персонала (количество, квалификация, степень готовности).

Специальных навыков для использования данного продукта не требуется.

3. Требования к продукту разработки.

3.1. Требования к продукту в целом.

Разрабатываемый продукт должен соответствовать следующим требованиям:

- Безопасность – продукт должен обеспечивать достаточный уровень безопасности для электронных документов.
- Доступность – продукт должен быть доступен 24/7.
- Масштабируемость – продукт должен обладать возможностью масштабирования в зависимости от объёмов.
- Конфиденциальность – продукт должен гарантировать конфиденциальность информации и защиту личных данных пользователей.
- Аутентификация – продукт должен обеспечивать возможность аутентификации пользователей, чтобы предотвратить несанкционированный доступ к документам.
- Лёгкость использования – продукт должен быть интуитивно понятным и простым для использования базовым пользователем продукта.
- Совместимость – продукт должен быть совместим с различными операционными системами и программным обеспечением.

Важным требованием к разрабатываемому продукту является процесс подписания документа. После загрузки документа в сервис, который требуется подписать и выбора людей, которые должны это сделать – документ должен получать отметку «На подписании» и помещаться в соответствующий раздел. Процесс подписания документа, имеющего статус «На подписании», должен заключаться в наложении в нужном месте документа ранее загруженной в сервис факсимильной электронной подписи.

3.2. Аппаратные требования.

Для работы с данным продуктом необходимо следующее техническое обеспечение со следующими минимальными характеристиками:

- Процессор с тактовой частотой не менее 1 ГГц.
- Оперативная память объёмом не менее 2 ГБ.

- Жёсткий диск объёмом от 60 ГБ.
- Постоянный доступ к сети интернет.

3.3. Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).

Для работы с данным продуктом необходимо следующее программное обеспечение:

- Операционная система семейства Windows, Linux или macOS.
- Браузер Google Chrome, Mozilla Firefox или Opera с включенной поддержкой HTML5 и JavaScript.
- Антивирусное программное обеспечение.

3.4. Указание программного обеспечения, используемого для реализации.

Для реализации данного продукта должно быть использовано следующее программное обеспечение:

- Веб-сервер для отображения веб-сервиса пользователю – Apache или Nginx как в связке, так и отдельно.
- База данных для хранения информации о пользователях и документах на выбор – MySQL или Redis.
- Среда разработки – Microsoft Visual Studio Code.
- Языки программирования – HTML, CSS, JavaScript, Python.

3.5. Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.

В данном продукте источником данных является пользователь, зарегистрировавшийся в системе. Порядок ввода данных в систему происходит следующим образом:

- Пользователь загружает заранее подготовленный документ, который требуется подписать.

- Ввод контактной информации, такой как ФИО и электронная почта адресованных лиц, которым необходимо отправить на подписание документ.
- Сохранение и последующая отправка документа адресатам.
- Ожидание подписания документа адресатами.

3.6. Порядок взаимодействия с другими системами, возможности обмена информацией.

Для работы данного продукта взаимодействие с другими системами не требуется. Продукт автономен и не зависит от других систем.

3.7. Меры защиты информации.

Для защиты информации в данном продукте должны использоваться следующие меры:

- Шифрование данных – входящий и исходящий трафик должен шифроваться с использованием протоколов защиты транспортного уровня, таких как SSL или TLS.
- Аутентификация – пользователи должны аутентифицироваться при входе в систему с использованием надёжных методов аутентификации, таких как логин и пароль.
- Физическая безопасность – все серверы и хранилища данных должны находится в зоне физической безопасности и могут быть доступны только авторизованному персоналу.
- Безопасность данных – все данные должны храниться в безопасном месте, защищённом от несанкционированного доступа.
- Регулярное обновление – операционная система сервера с установленным продуктом должна обновляться регулярно с учётом последних обновлений безопасности. Это может включать в себя обновление программного обеспечения и обновления операционной системы.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

Интерфейс продукта должен быть минималистичным, элементы интерфейса должны быть в едином стиле, авторизация пользователя в системе должна производиться по логину и паролю. Интерфейс должен быть интуитивно понятным и лёгким в использовании, даже для пользователей без специальных знаний.

4.2. Размещение информации на экране, дизайн экрана.

Схема интерфейса до входа пользователя в систему(см. рисунок 2). У пользователя есть возможность пройти регистрацию в системе, а в случае уже наличия логина и пароля – войти в систему.

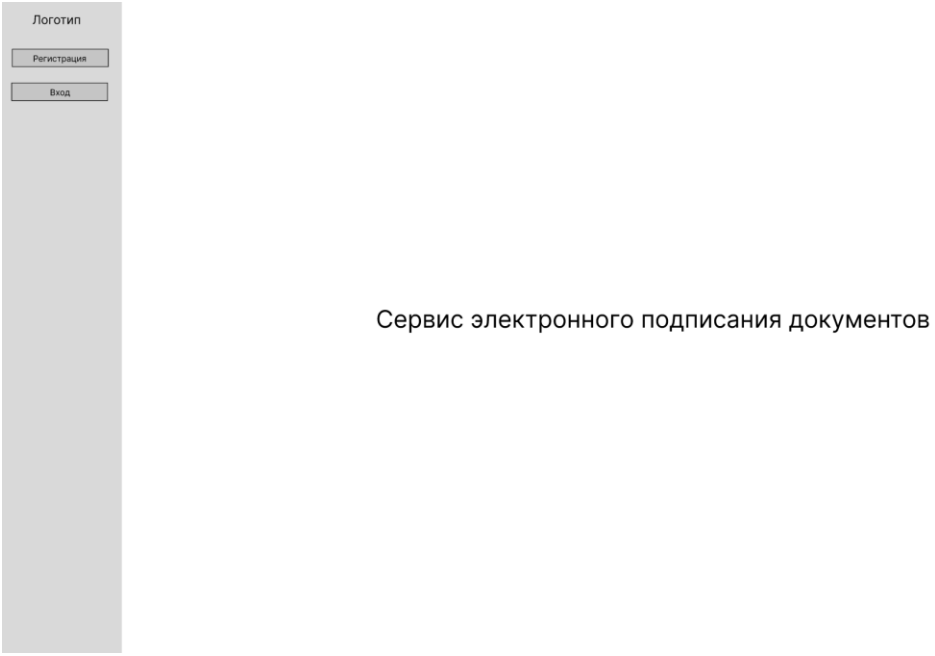


Рисунок 2. Схема интерфейса до входа пользователя в систему

Схема интерфейса после входа пользователя в систему (см. рисунок 3). У пользователя есть возможность посмотреть список своих документов и создать новый документ.

Рисунок 3. Схема интерфейса после входа пользователя в систему

Схема интерфейса после формирования документа (см. рисунок 4). У пользователя после составления документа есть возможность выбрать адресатов, заполнить их данные и отправить документ им на подпись.

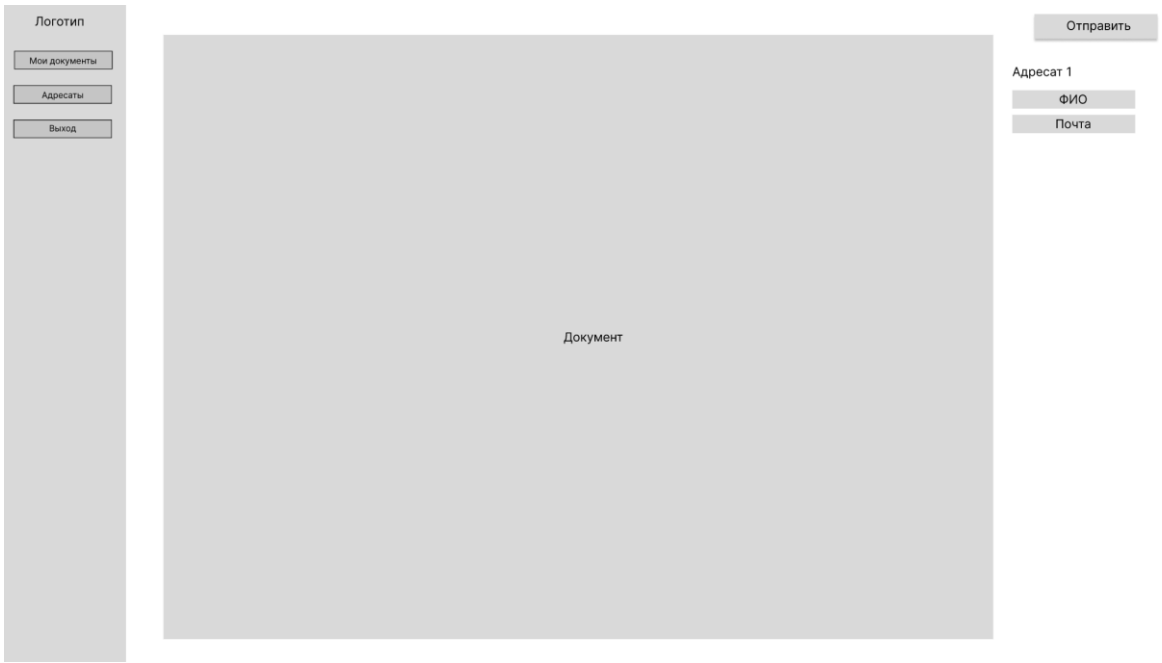


Рисунок 4. Схема интерфейса после формирования документа

4.3. Особенности ввода информации пользователем, представление выходных данных.

Ввод информации должен осуществляться пользователем. В настройках пользователь должен добавить подпись с возможностью изменения подписи. Пользователь через специальную форму должен загружать документ, и выбирать согласующих. После отправки документа на подписание – у автора в специальном разделе должен отслеживаться статус документа, а у согласующих лиц в разделе документов на подписание – должен появиться документ для подписи. После подписания документа всеми участниками процесса – документ должен становиться доступным для подписи у всех участников. При отклонении документа – документ должен быть занесен в соответствующий раздел.

5. Требования к документированию

5.1. Перечень сопроводительной документации.

Для данного продукта необходим следующий перечень сопроводительной документации:

- Инструкция пользователя данного продукта.
- Системные требования к серверу размещения данного продукта.
- Инструкция администратора данного продукта.

5.2. Требования к содержанию отдельных документов.

Требований к содержанию отдельных документов не предусмотрено.

6. Порядок сдачи-приемки продукта

Сдача-приёмка продукта производится в соответствии со сроками выполнения ВКР.

Глава 2. Практическая (оригинальная) часть

2.1 Модельные представления объекта разработки

Главной целью данной работы является разработка веб-сервиса электронного подписания документов. Данный веб-сервис будет разработан основываясь на техническом задании подготовленным в разделе 1.3. Структура страниц и пунктов меню веб-сервиса изображена на структурной схеме (см. рисунок 5).

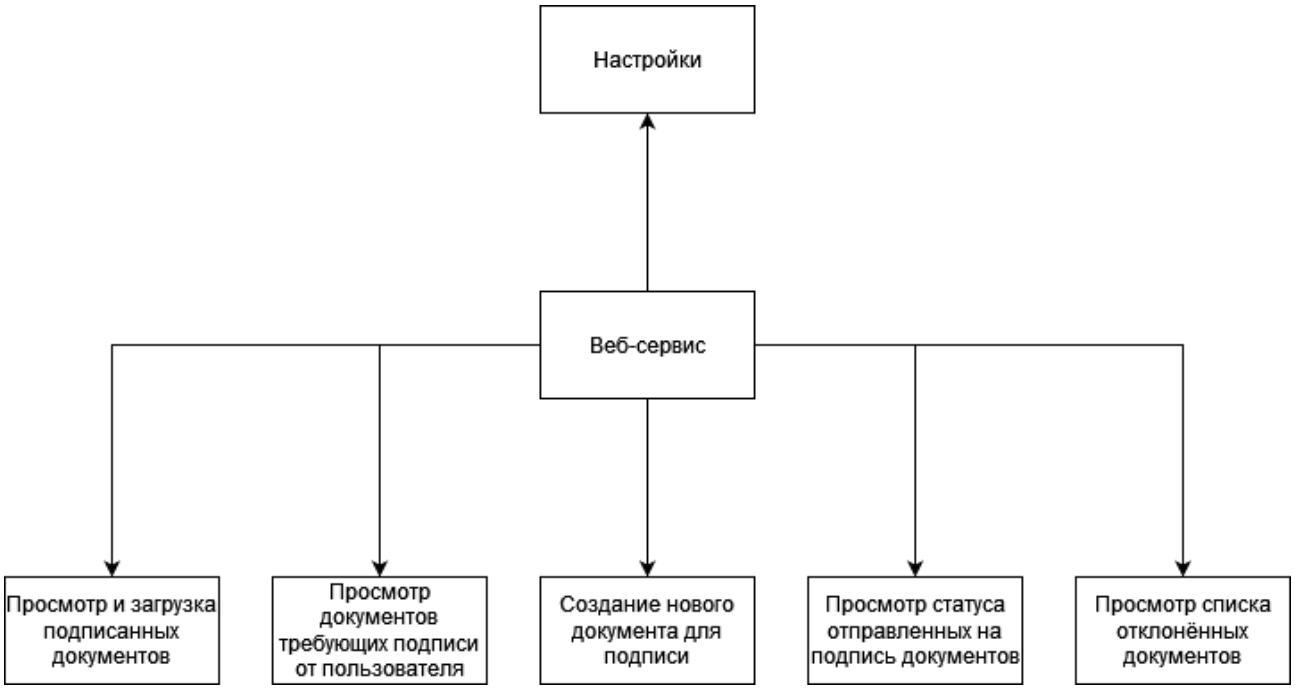


Рисунок 5. Структурная схема веб-сервиса электронного подписании документов

Доступ пользователя к веб-сервису электронного подписания документов будет осуществляться при помощи браузера. Начальным этапом работы пользователя с веб-сервисом будет регистрация пользователя в системе и последующий вход в веб-сервис. После входа в веб-сервис пользователю будут представлены следующие функции веб-сервиса:

1. Создание нового документа для подписи – в данном разделе у пользователя будет возможность загрузить ранее созданный документ в веб-сервис, выбрать лицо, от которого необходимо получить подпись и отправить документ для подписи.

2. Просмотр документов, требующих подписи от пользователя – в данном разделе у пользователя будет возможность ознакомиться со списком документов и их текстом после чего принять для себя решение подписать документ или отклонить его.

3. Просмотр статуса отправленных на подпись документов – в данном разделе у пользователя будет возможность отследить статусы отправленных ранее документов на подпись.

4. Просмотр списка отклонённых документов – в данном разделе у пользователя будет возможность ознакомится с причинами отклонения ранее отправленных документов.

5. Просмотр и загрузка подписанных документов – в данном разделе у пользователя будет возможность ознакомиться со списком и содержимым документов, которые были подписаны, а после чего загрузить их на устройство для дальнейшей работы с документом.

Процесс подписания документа описан в блок-схеме (см. рисунок 6). В данном процесс у пользователя есть возможность подписать документ разместив подпись в необходимой части, предварительно ознакомившись с содержимым документа. Помимо возможности подписать документ, есть функция отклонить документ указав при этом причину отклонения, если он по определённым причинам не может быть подписан.

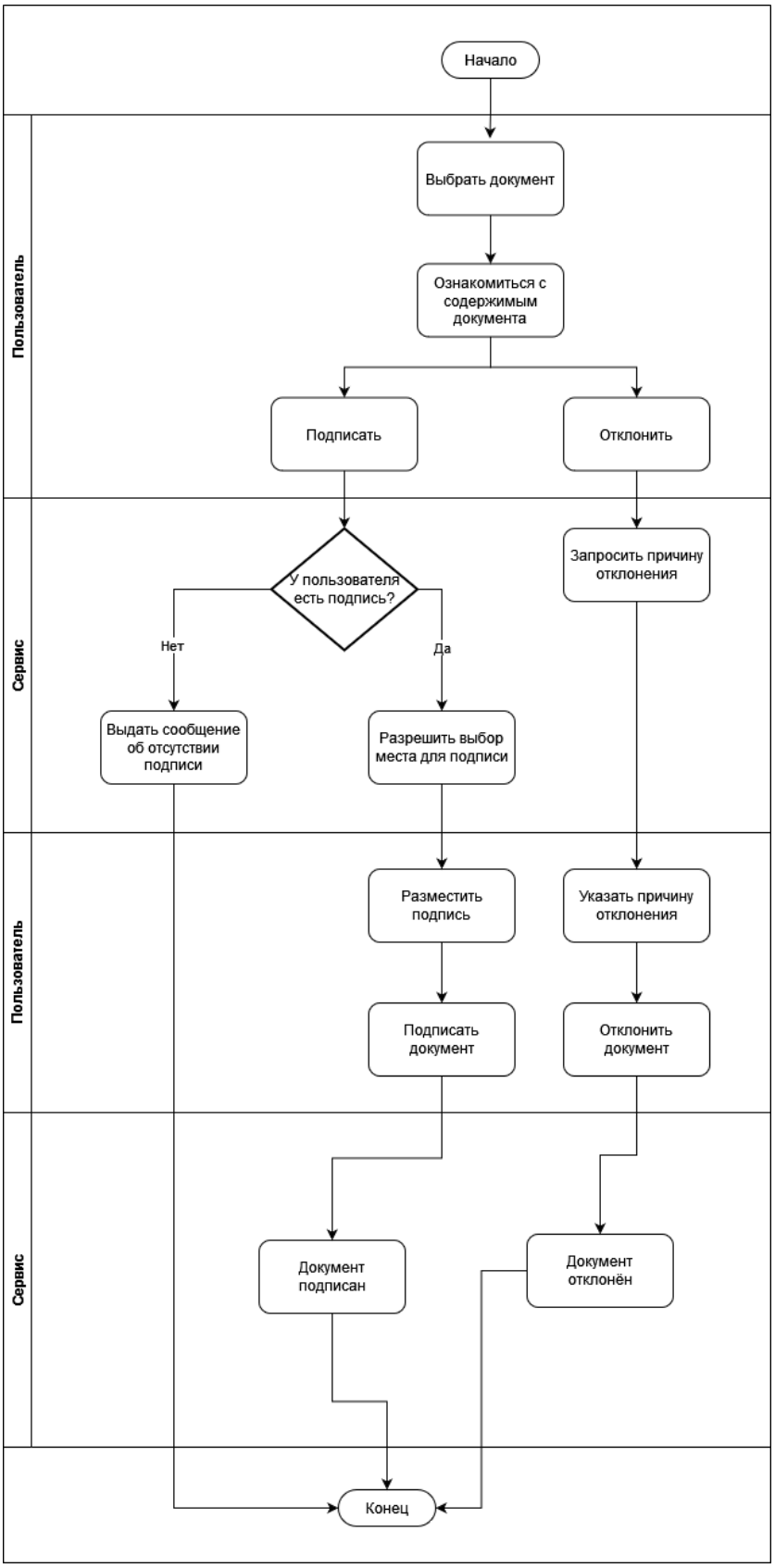


Рисунок 6. Процесс подписания документа

Веб-сервис электронного подписания документов будет разработан по принципу разделения на фронтенд и бэкенд. Фронтенд будет отвечать за пользовательский интерфейс, взаимодействие с пользователями и отображение данных, а бэкенд будет отвечать за обработку запросов, хранение данных, безопасность и логику приложения. Клиентский интерфейс и серверная часть будут разработаны отдельно, чтобы обеспечить масштабируемость, гибкость и удобное сопровождение приложения. Такой подход позволит лучше организовать разработку, облегчит интеграцию с различными платформами и обеспечит удобство использования для конечных пользователей.

Веб-сервис электронного подписания документов будет разработан по следующей схеме (см. рисунок 7). Принцип работы фронтенда с бэкендом описан на схеме взаимодействия фронтенда с бэкендом (см. рисунок 1).

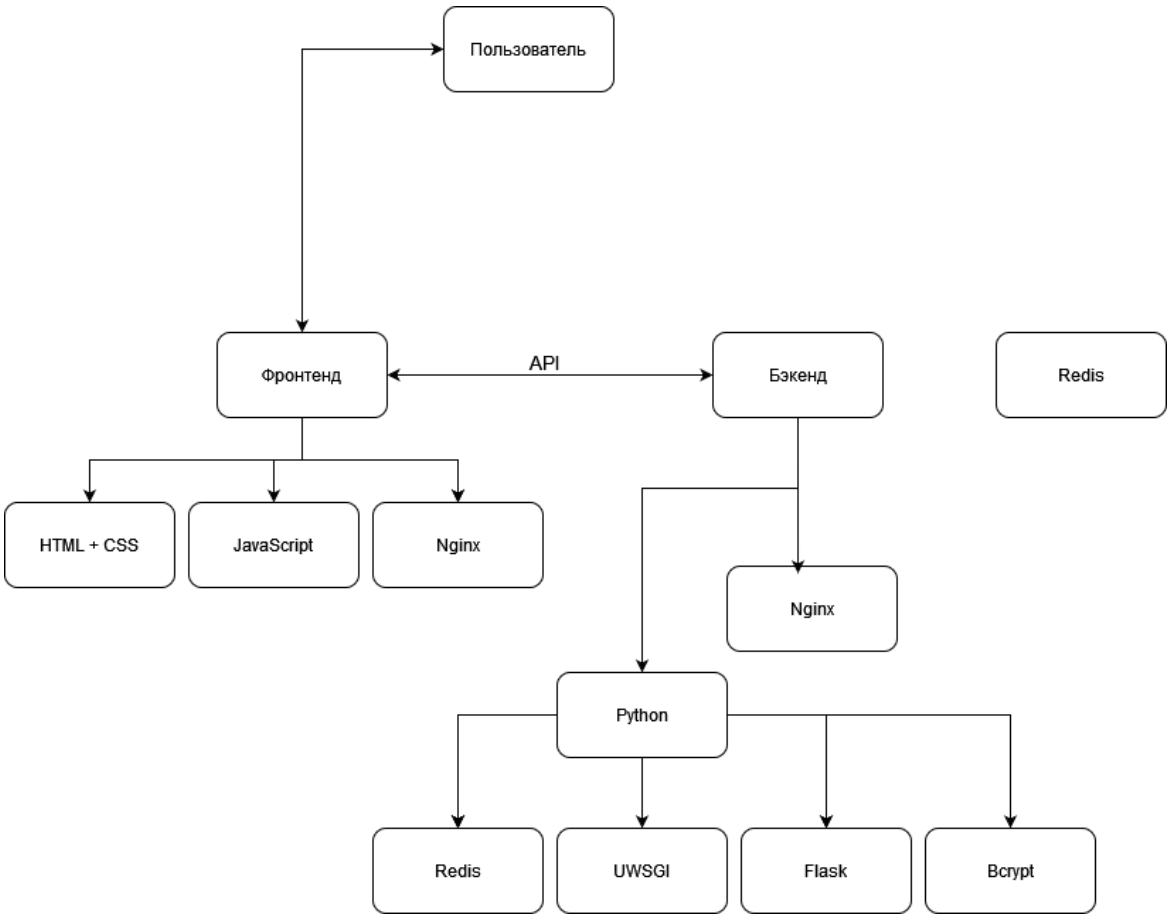


Рисунок 7. Схема веб-сервиса электронного подписания документов

Таким образом, разрабатываемый веб-сервис электронного подписания будет основываться на архитектуре фронтенд бэкенд. Фронтенд будет обеспечивать отрисовку элементов веб-сервиса пользователю, Бэкенд в свою очередь благодаря использованию языка Python с фреймворками будет обеспечивать обработку и хранение данных. Такая архитектора позволит создать надежный и масштабируемый веб-сервис, способный эффективно обрабатывать запросы пользователей и передачу данных.

2.2 Описание продукта (результата разработки)

В результате был разработан веб-сервис электронного подписания документов, позволяющий двум и более пользователям подписать заранее подготовленный документ в формате pdf в любой из существующих программ для создания документов. Результат разработки полностью основан на схеме (см. рисунок 6) и состоит из модулей фронтенда и бэкенда.

Сам веб-сервис электронного подписания документов представляет из себя виртуальную машину под управлением операционной системы семейства Linux с установленным на неё Python, базой данных Redis, сервером веб-приложений Nginx. Внутри виртуальной машины создан сервис (см. рисунок 8), запускающий бэкенд.

```
[Unit]
Description=uWSGI instance to serve sign backend
After=network.target

[Service]
User=webuser
Group=webuser
WorkingDirectory=/opt/sign/backend
Environment="PATH=/opt/sign/backend/bin"
ExecStart=/opt/sign/backend/bin/uwsgi --ini backend.ini

[Install]
WantedBy=multi-user.target
```

Рисунок 8. Код сервиса, запускающего бэкенд

Для отображения пользователю информации на экране и работы фронтенда с бэкендом используется сервер веб-приложений Nginx. Пользователь обращается на 80 порт при помощи браузера и Nginx ему в ответ рисует содержимое фронтенда, а именно файл index.html. В тоже самое время фронтенд общается с бэкендом при помощи того же Nginx, но уже через порт 9200 (см. рисунок 9).

```
server {
    listen 80;
    server_name 192.168.88.16;
    root /opt/sign/frontend;
    index index.html;
    location / {
        try_files $uri /index.html;
    }
}

server {
    listen 9200;
    server_name 192.168.88.16;
    location / {
        include uwsgi_params;
        uwsgi_pass unix:/opt/sign/backend/backend.sock;
    }
}
```

Рисунок 9. Содержимое файла конфигурации Nginx

Взаимодействие пользователя с веб-сервисом осуществляется через фронтенд. Во фронтенде используются HTML, CSS и JavaScript. HTML и CSS отвечают визуальное оформление веб-сервиса и размещение элементов, а JavaScript за получение запросов от пользователя и соединение с бэкендом. Фронтенд представляет из себя html файл (см. рисунок 10), в который подключены файлы с стилями CSS и файлы JavaScript. Для работы фронтенда у пользователя в браузере обязательно должна быть включена поддержка JavaScript иначе, сайт отображаться не будет, и пользователь получит сообщение о необходимости включить JavaScript у себя в настройках браузера.

```html
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8"/>
        <link rel="icon" href="/favicon.ico"/>
        <meta name="viewport" content="width=device-width,initial-scale=1"/>
        <meta name="theme-color" content="#000000"/>
        <meta name="description" content="Web site for sign documents"/>
        <title>Sign Document</title>
        <script defer="defer" src="/static/js/main.87dbf9d8.js"></script>
        <link href="/static/css/main.ec956d66.css" rel="stylesheet">
    </head>
    <body>
        <noscript>You need to enable JavaScript to run this app.</noscript>
        <div id="root"></div>
    </body>
</html>
```

Рисунок 10. Код главной страницы фронтенда

При первом входе на страницу сервиса – пользователя встречает страница ввода регистрационных данных пользователя (см. рисунок 11).

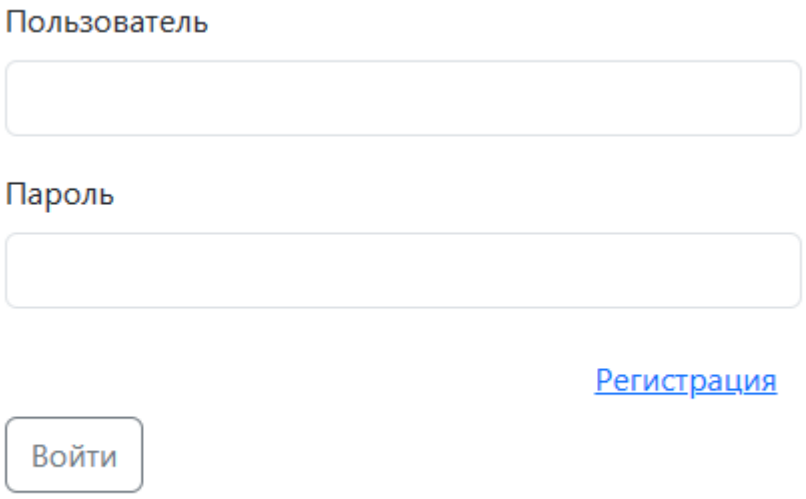


Рисунок 11. Страница входа в веб-сервис

Данная страница ввода логина и пароля отображается пользователю только в том случае, если у пользователя в браузере отсутствует куки под названием token (см. рисунок 12). Куки – это некий фрагмент данных, который сервер отправляет клиенту, то есть браузеру. И браузер сохраняет этот фрагмент данных

для дальнейшей работы с веб-сервисом. Проверка наличия этой куки реализована функцией check (см. рисунок 13).

Рисунок 12. Пример данных в Куки token

```
def check(data):
    if data is None:
        return False
    try:
        user_info = json.loads(data.replace("'", '"'))
    except json.JSONDecodeError:
        return False
    user = user_info['user']
    token = user_info['token']

    config = configparser.ConfigParser()
    config.read('config')
    redis_host = config['Path']['redis']

    pool = redis.ConnectionPool(host=redis_host, port=6379, db=0, decode_responses=True)
    db = redis.Redis(connection_pool=pool)

    if not db.exists(user):
        return False

    user_db = db.hgetall(user)
    if token != user_db['token']:
        return False

    timestamp = datetime.datetime.now().timestamp()
    if int(timestamp) - int(user_db['timestamp']) > 3600:
        return False
    else:
        return True
```

Рисунок 13. Функция проверки check

В случае если у пользователя отсутствуют регистрационные данные – на странице входа есть кнопка, позволяющая пользователю зарегистрироваться в сервисе. На странице регистрации от пользователя сервиса требуется ввести: логин латинскими буквами, фамилию, имя, отчество и пароль два раза для исключения возможности ввести некорректный пароль (см. рисунок 14).

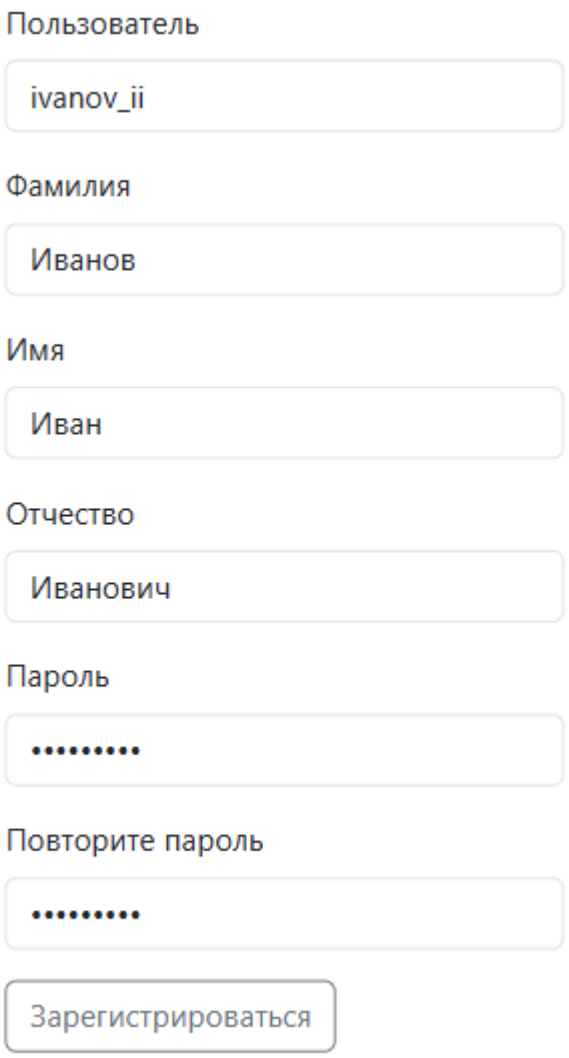


Рисунок 14. Страница регистрации пользователя

После нажатия пользователем кнопки зарегистрироваться – из фронтенда сформируется запрос к бэкенду и в бэкенде будет задействована функция put (см. рисунок 15) в классе авторизации и регистрации Auth. Функция put осуществляет сбор информации о заполненных полях в форме регистрации пользователем, проверяет их наличие в базе данных Redis и в случае если данные в базе отсутствуют – добавляет их в базу тем самым регистрируя пользователя в веб-сервисе. В случае наличия в базе регистрационных данных – пользователь получит сообщение о существовании такой учётной записи.

```python
def put(self):
    user = request.json['user']
    firstname = request.json['firstname']
    lastname = request.json['lastname']
    surname = request.json['surname']
    b64_password = request.json['password']

    if self.rd.exists(user):
        response = headers.create('User already exist')
        return response, 409

    salt = bcrypt.gensalt(12)
    password = bcrypt.hashpw(b64_password.encode('utf-8'), salt)

    user_info = {
        'firstname': firstname,
        'lastname': lastname,
        'surname': surname,
        'hash': password,
        'token': '',
        'timestamp': ''
    }
    self.rd.hset(user, mapping=user_info)

    response = headers.create({'status': 'success'})
    return response, 201
```

Рисунок 15. Функция регистрации пользователя

Если пользователь попытается зайти в сервис при помощи некорректных регистрационных данных – появится ошибка, которая сообщит пользователю сервиса о некорректности введенных регистрационных данных (см. рисунок 16).

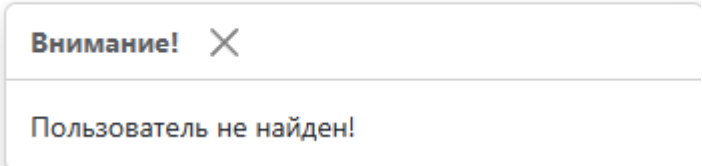


Рисунок 16. Всплывающее уведомление при попытке входа с некорректными данными

Функция, которая обрабатывает вход пользователя в веб-сервис также принадлежит классу Auth и имеет название post (см. рисунок 17). Она получает от фронтенда введённые на странице входа логин и пароль пользователя и

обрабатывает их. Проверяя наличие пользователя в базе данных и корректность ввода пароля в случае совпадения логина. В случае ввода корректных данных в браузер пользователя будет записана куки token (см. рисунок 12).

```
def post(self):
    user = request.json['user']
    password = request.json['password']

    if not self.rd.exists(user):
        response = headers.create({'error': 'Пользователь не найден!'})
        return response, 404

    user_info = self.rd.hgetall(user)

    if not bcrypt.checkpw(password.encode('utf-8'), user_info['hash'].encode('utf-8')):
        response = headers.create({'error': 'Некорректный логин или пароль!'})
        return response, 401

    token = token_hex(10)
    timestamp = datetime.datetime.now().timestamp()
    user_info['token'] = token
    user_info['timestamp'] = int(timestamp)
    self.rd.hset(user, mapping=user_info)
    cookies = {'user': user, 'token': token}
    response = headers.create_with_cookie('token', str(cookies))
    return response, 200
```

Рисунок 17. Функция входа пользователя в веб-сервис

После получения пользователем регистрационных данных и последующего входа в сервис – пользователь попадает на главную рабочую область сервиса (см. рисунок 18). Которая состоит из бокового меню сервиса и зоны работы с документами.

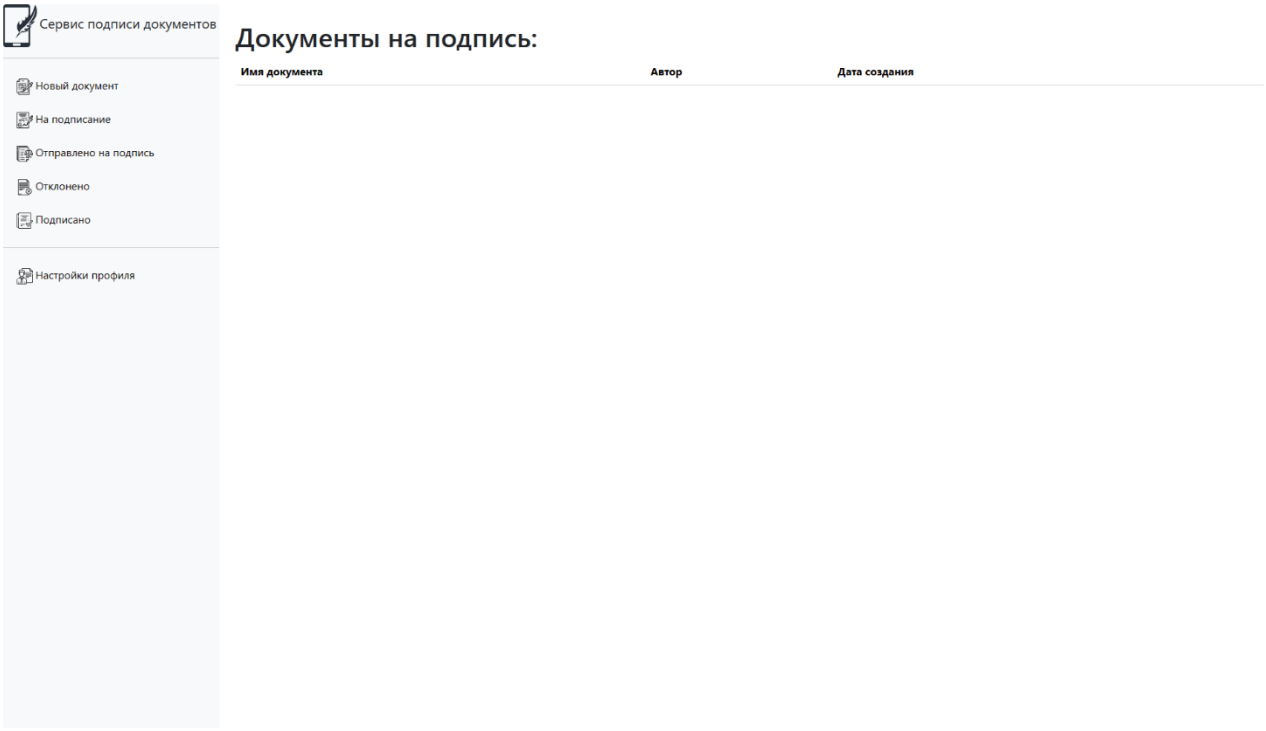


Рисунок 18. Главная рабочая область веб-сервиса

Боковое меню позволяет пользователю переключаться между разделами для работы с документами (см. рисунок 19).

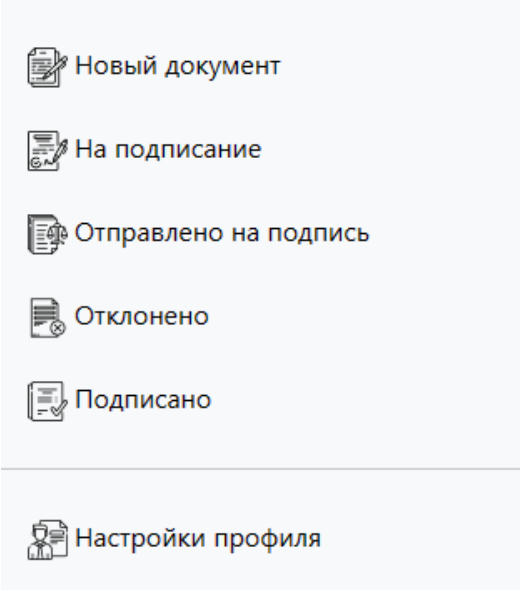


Рисунок 19. Боковое меню веб-сервиса

Раздел «Новый документ». В данном разделе у пользователя есть возможность загрузить в сервис ранее созданный документ в формате pdf выбрав его на компьютере, либо перетащив в специальную форму (см. рисунок 20).

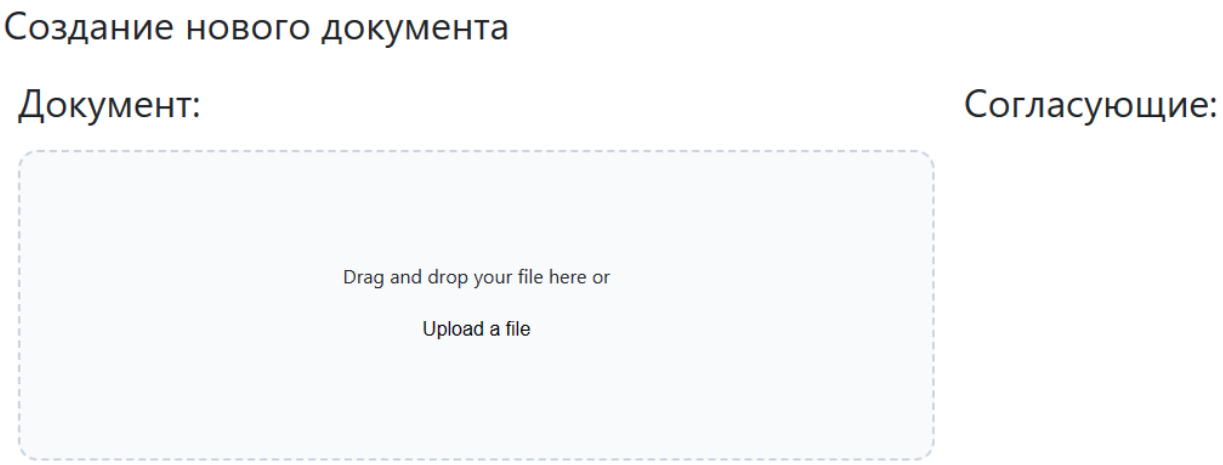


Рисунок 20. Внешний вид раздела «Новый документ»

После загрузки документа – пользователю отображается содержимое загруженного документа, если у документа несколько страниц есть возможность пролистать документ и ознакомиться с ним. Также после загрузки документа появляется возможность выбрать список лиц от кого требуется подпись документа. Если их несколько, то достаточно выбрать двух пользователей из выпадающего списка. После выбора пользователей достаточно нажать на кнопку «Отправить на согласование» и документ отправится всем ранее указанным пользователям (см. рисунок 21).

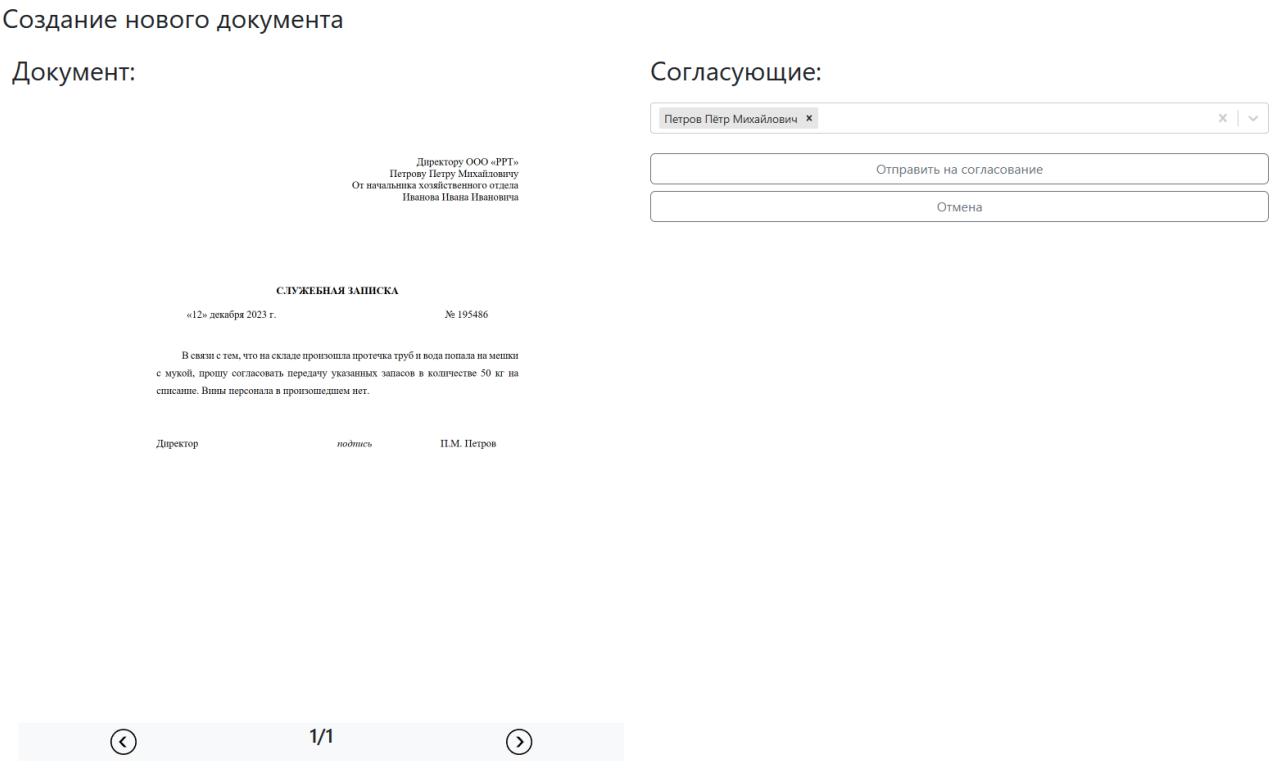


Рисунок 21. Внешний вид раздела «Новый документ» после загрузки документа и выбора согласющих

В бэкенде загрузка документа на сервер обрабатывается функцией put (см. рисунок 22) класса document. При выполнении этой функции выполняется проверка на наличие в файловой системе веб-сервиса документов с тем же именем с каким документ был загружен пользователем. Кроме проверки наличия документа на сервере происходит утсановка документу статуса ForSign для лиц указанных в списке согласующих. После занесения всех данных в базу данных Redis загруженный пользователем файл сохраняется в файловую систему веб-сервиса.

```python
def put(self):

    if self.status_code != 200:
        return self.response, 401

    signers_form = request.form.get('signers')
    file = request.files['file']

    if self.rd.exists(file.filename):
        response = headers.create({'error': 'Такой документ уже есть в системе!'})
        return response, 409

    signers = signers_form.split(',')
    signers_db = {}

    for signer in signers:
        signers_db[signer] = {
            'status': 'ForSign',
            'comment': ''
        }
    create_date = datetime.datetime.now().timestamp()
    doc_info = {
        'author': self.user,
        'status': 'ForSign',
        'signers': str(signers_db),
        'createDate': int(create_date)
    }

    self.rd.hset(file.filename, mapping=doc_info)

    file_path = 'files/documents/' + file.filename
    file.save(file_path)

    response = headers.create({'status': 'success'})
    return response, 201
```

Рисунок 22. Функция загрузки файла для подписи на сервер

Получения списка пользователей для подписания документов со стороны бэкенда выполняется функцией get (см. рисунок 23) класса users. Итоговые данные, полученные после выполнения данной функции, представляют из себя список пользователей с логинами и ФИО (см. рисунок 24).

```python
def get(self):
    if self.status_code != 200:
        return self.response, 401

    data = []
    token = request.cookies.get('token')
    user_info = json.loads(token.replace("'", '"'))
    keys = self.rd.keys()
    for key in keys:
        if user_info['user'] == key:
            continue
        user = self.rd.hgetall(key)
        data.append({
            'value': key,
            'label': user['lastname'] + ' ' + user['firstname'] + ' ' + user['surname']
        })

    response = headers.create(data)
    return response, 200
```

Рисунок 23. Функция вывода списка пользователей

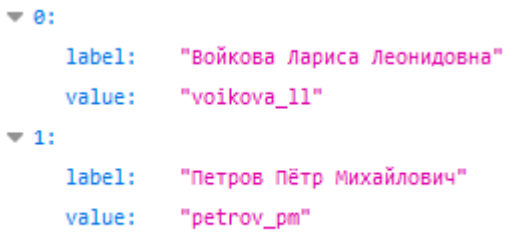


Рисунок 24. Результат выполнения функции вывода пользователей

Раздел «На подписание». В данный раздел помещаются все документы, для которых согласующим указан текущий пользователь (см. рисунок 25). В бэкенде данный раздел обрабатывается функцией for_sign (см. рисунок 26) класса getdocuments. Данная функция в результате запроса к базе данных выводит список (см. рисунок 27), состоящий из имени документа, даты его загрузки и ФИО автора документа.

Рисунок 25. Внешний вид раздела «На подписание»

```python
def for_sign(self):
    keys = self.db_docs.keys()
    docs = []
    for doc in keys:
        doc_info = self.db_docs.hgetall(doc)
        signers = json.loads(doc_info['signers'].replace("'", '"'))
        if self.user in signers:
            if signers[self.user]['status'] != 'ForSign':
                continue
            author = self.db_users.hgetall(doc_info['author'])
            full_name = author['lastname'] + ' ' + author['firstname'] + ' ' + author['surname']
            create_date = datetime.datetime.fromtimestamp(int(doc_info['createDate']))
            docs.append({'name': doc, 'author': full_name, 'createDate': create_date})
    self.db_docs.close()
    self.db_users.close()
    return docs
```

Рисунок 26. Функция вывода списка документов на подпись

```
' 0:
    author:       "Петров Пётр Михайлович"
    createDate:   "Sun, 14 Jan 2024 19:55:07 GMT"
    name:         "служебная записка3.pdf"
```

Рисунок 27. Результат работы функции вывода списка документов на подпись

По нажатию кнопки «Detail» пользователь увидит содержимое документа, с возможностью пролистать документ, в случае если документ содержит более одного листа. Кроме того, у пользователя есть возможность подписать документ либо отклонить (см. рисунок 28). С стороны бэкенда в этот момент выполняется функция get (см. рисунок 29) класса document которая выгружает пользователю необходимый документ.

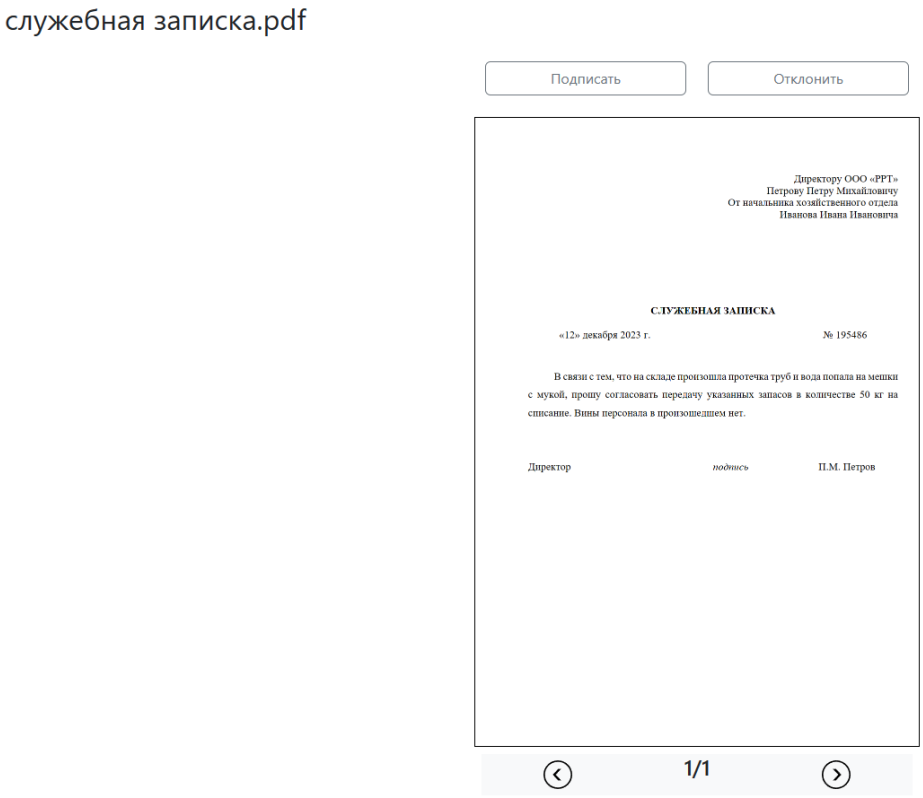


Рисунок 28. Внешний вид страницы с подробностями подписываемого документа

```
def get(self):

    if self.status_code != 200:
        return self.response, 401

    document = request.args.get('id')
    document_path = 'files/documents/' + document

    config = configparser.ConfigParser()
    config.read('config')
    url = config['Path']['url']
    response = make_response(send_file(document_path))
    response.headers.add('Access-Control-Allow-Headers', 'content-type')
    response.headers.add('Access-Control-Allow-Origin', url)
    response.headers.add('Access-Control-Allow-Credentials', 'true')
    response.headers.add('Access-Control-Allow-Methods', 'GET, POST, PUT')
    return response, 200
```

Рисунок 29. Функция выгрузки необходимого документа

По нажатию кнопки «Подписать» курсор меняет свой вид и пользователю необходимо нажать в документе на поле для подписи, после чего в предпросмотре документа отобразится подпись пользователя, и кнопка «Подписать» сменится на кнопку «Согласовать» (см. рисунок 30). По нажатию которой документ получит статус подписанного и станет доступен для скачивания автором документа и тем, кто его подписал. Наложение подписи происходит на фронтенде, но сохранение документа всё равно происходит на бэкенде. Происходит это следующим образом: вызывается функция delete класса document (см. рисунок 31) которая удаляет документ, а затем вызывается фнукция post (см. рисунок 32) того же класса которая сохраняет документ в файловой системе веб-сервиса и меняет ему статус на Signed.

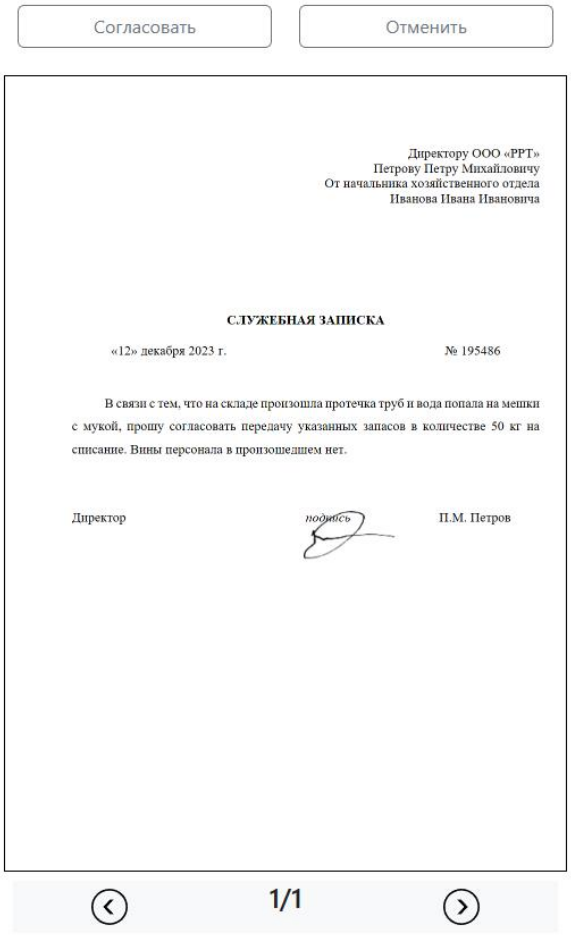


Рисунок 30. Внешний вид документа после выбора места для подписи

```
def delete(self):
    doc = request.json['id']
    self.rd.delete(doc)
    file_path = 'files/documents/' + doc
    os.remove(file_path)
    response = headers.create({'status': 'succcess'})
    return response, 200
```

Рисунок 31. Функция удаления файла документа

```
def post(self):
    if self.status_code != 200:
        return self.response, 401

    status = request.json['status']

    if status == 'signed':
        file = request.json['file']
        filename = request.json['id']
        binary = a2b_base64(file)
        file_path = 'files/documents/' + filename
        f = open(file_path, 'wb')
        f.write(binary)
        f.close()

        doc = self.rd.hgetall(filename)
        signers = json.loads(doc['signers'].replace("'", '"'))
        signers[self.user]['status'] = 'Signed'
        doc['signers'] = str(signers)

        statuses = []
        for signer in signers:
            statuses.append(signers[signer]['status'])
        if all(elem == 'Signed' for elem in statuses):
            doc['status'] = 'Signed'

        self.rd.hset(filename, mapping=doc)

    else:
        filename = request.json['id']
        reason = request.json['reason']
        doc = self.rd.hgetall(filename)
        signers = json.loads(doc['signers'].replace("'", '"'))
        signers[self.user] = {
            'status': 'Declined',
            'comment': reason
        }
        doc['signers'] = str(signers)
        doc['status'] = 'Declined'
        self.rd.hset(filename, mapping=doc)

        response = headers.create({'status': 'success'})
        return response, 200

    response = headers.create({'status': 'success'})
    return response, 200
```

Рисунок 32. Функция подписания или отклонения подписи

При отклонении документа сервис обязательно запросит причину отклонения и в последующем указанная причина будет отображена у автора документа (см. рисунок 32). В бэкенде при отклонении файла вызывается функция post (см. рисунок 31) класса document, но в этом случае вместо перезаписи документа в базу данных вносится комментарий указанный пользователем при отклонении и документу проставляется статус Declined.

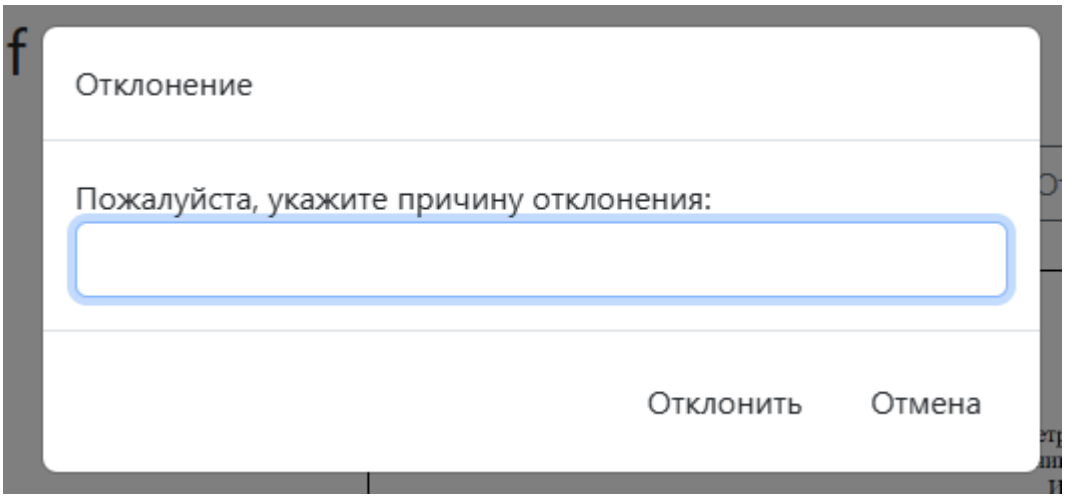


Рисунок 32. Всплывающее окно с полем ввода причины отклонения

Раздел «Отправлено на подпись». В данном разделе отображается список документов которвые были отправлены на подпись. Этот раздел содержит в себе информацию о имени документа, дате загрузки документа в сервис, списке согласущих лиц и статусе документа (см. рисунок 33). В бекенде список документов ожидающих подписи формируется функцией on_sign (см. рисунок 34) класса getdocuments. Данная функция в результате запроса к базе данных выводит список (см. рисунок 35), состоящий из имени документа, даты его загрузки и ФИО людей, от которых требуется подпись документа.

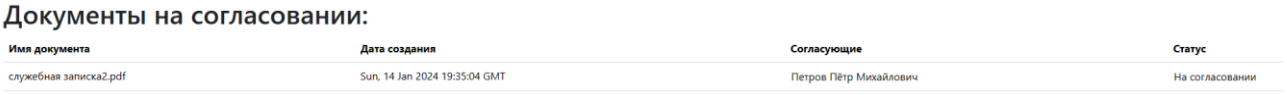


Рисунок 33. Раздел «Отрправлено на подпись»

```
def on_sign(self):
    keys = self.db_docs.keys()
    docs = []
    for doc in keys:
        doc_info = self.db_docs.hgetall(doc)
        if doc_info['status'] != 'ForSign':
            continue
        if doc_info['author'] != self.user:
            continue
        users = []
        signers = json.loads(doc_info['signers'].replace("'", '"'))
        for signer in signers:
            if signers[signer]['status'] == 'ForSign':
                status = 'На согласовании'
            else:
                status = 'Согласовано'
            user_info = self.db_users.hgetall(signer)
            full_name = user_info['lastname'] + ' ' + user_info['firstname'] + ' ' + user_info['surname']
            users.append({'name': full_name, 'status': status})
        create_date = datetime.datetime.fromtimestamp(int(doc_info['createDate']))
        data = {
            'name': doc,
            'createDate': create_date,
            'signers': users
        }
        docs.append(data)
    return docs
```

Рисунок 34. Функция вывода документов ожидаюих подписи

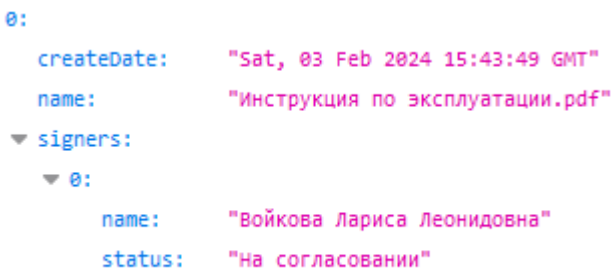


Рисунок 35. Результат работы функции вывода списка документов ожидающих подписи

Раздел «Отклонено». Данный раздел содержит список документов, которые были отклонены согласующими и содержит в себе информацию о имени документа, дате его загрузки в сервис, именах согласующих и причине отклонения, которую прописали согласующие (см. рисунок 36). Информация об отклоненных документах отображается только у авторов документов. В бэкенде данный раздел обрабатывается функцией declined (см. рисунок 37) класса getdocuments. Данная функция выводит список документов, у которых проставлен статус declined и результатом работы функции является список документов с причиной отклонения (см. рисунок 38).

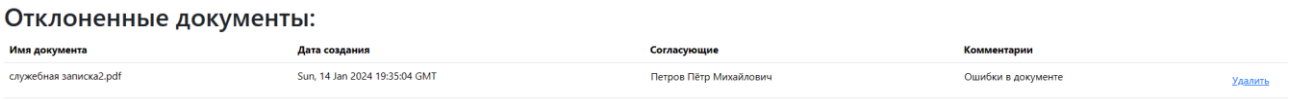


Рисунок 36. Раздел «Отклонено»

```
def declined(self):
    keys = self.db_docs.keys()
    docs = []
    for doc in keys:
        doc_info = self.db_docs.hgetall(doc)
        if doc_info['status'] == 'Declined':
            if self.user != doc_info['author']:
                continue
            signers = json.loads(doc_info['signers'].replace("'", '"'))
            users = []
            for signer in signers:
                if signers[signer]['status'] == 'Declined':
                    user_info = self.db_users.hgetall(signer)
                    full_name = user_info['lastname'] + ' ' + user_info['firstname'] + ' ' + user_info['surname']
                    users.append({'name': full_name, 'reason': signers[signer]['comment']})

            create_date = datetime.datetime.fromtimestamp(int(doc_info['createDate']))
            data = {
                'name': doc,
                'createDate': create_date,
                'signers': users
            }
            docs.append(data)
    return docs
```

Рисунок 37. Функция вывода отклонённых документов

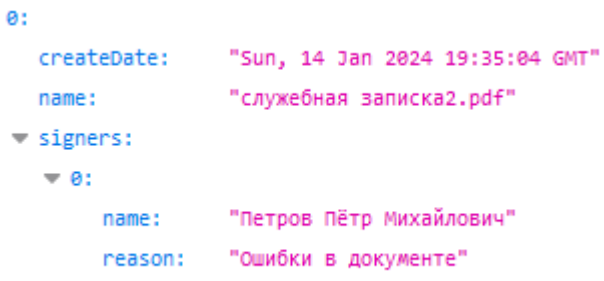


Рисунок 38. Результат работы функции вывода отклонённых документов

Раздел «Подписано». Данный раздел содержит список документов, которые были подписаны и содержит в себе информацию о имени документа, дате его загрузки в сервис, именах согласующих и кнопку, которая позволяет скачать документ (см. рисунок 39). Информация о подписанных документах доступна для всех участников подписания: автора документа и согласующих лиц. В бэкенде данный раздел обрабатывается функцией signed (см. рисунок 40) класса get-documents. Результатом работы данной функции является список (см. рисунок 41), содержащий ФИО автора, ФИО согласующих, дату создания документа и имя документа.

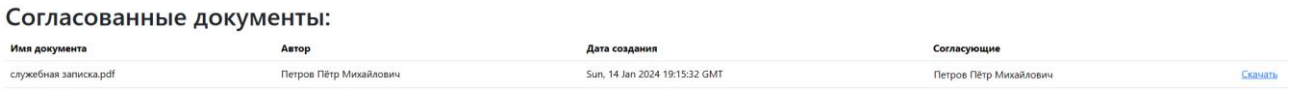


Рисунок 39. Раздел «Подписано»

```python
def signed(self):
    keys = self.db_docs.keys()
    docs = []
    for doc in keys:
        doc_info = self.db_docs.hgetall(doc)
        if doc_info['status'] == 'Signed':
            author = self.db_users.hgetall(doc_info['author'])
            full_name = author['lastname'] + ' ' + author['firstname'] + ' ' + author['surname']
            signers = json.loads(doc_info['signers'].replace("'", '"'))
            users = []
            for signer in signers:
                user_info = self.db_users.hgetall(signer)
                full_name = user_info['lastname'] + ' ' + user_info['firstname'] + ' ' + user_info['surname']
                users.append({'name': full_name})
            create_date = datetime.datetime.fromtimestamp(int(doc_info['createDate']))
            data = {
                'name': doc,
                'author': full_name,
                'createDate': create_date,
                'signers': users
            }
            docs.append(data)
    return docs
```

Рисунок 40. Функция отображения подписанных документов

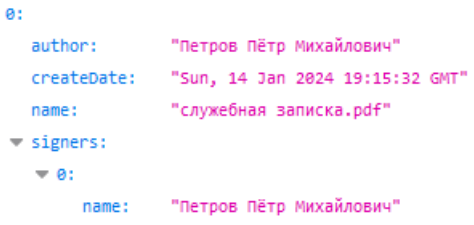
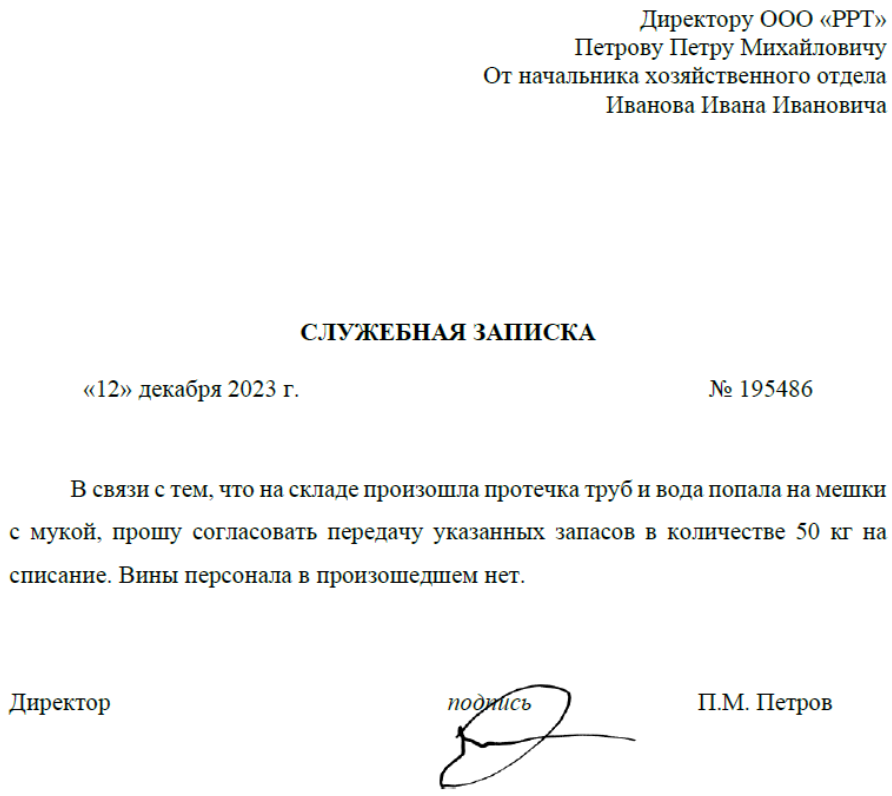


Рисунок 41. Результат работы функции вывода списка подписанных документов

По нажатию кнопки загрузки – документ скачивается на компьютер пользователя в формате pdf и содержит в себе ранее наложенную подпись согласующего (см. рисунок 42). В бэкенде скачивание документа реализовано функцией get класса document (см. рисунок 29).



Директору ООО «РРТ»
Петрову Петру Михайловичу
От начальника хозяйственного отдела
Иванова Ивана Ивановича

СЛУЖЕБНАЯ ЗАПИСКА

«12» декабря 2023 г. № 195486

В связи с тем, что на складе произошла протечка труб и вода попала на мешки с мукой, прошу согласовать передачу указанных запасов в количестве 50 кг на списание. Вины персонала в произошедшем нет.

Директор *подпись* П.М. Петров

Рисунок 42. Итоговый подписанный документ

Раздел «Настройки профиля». Данный раздел позволяет отредактировать пользователю свой профиль: изменить ФИО, пароль (для смены пароля необходимо обязательно ввести старый и новый пароли) (см. рисунок 43). В бэкенде изменение ФИО и пароля пользователей реализовано несколькими функциями. Для отображения ФИО используется функция get (см. рисунок 44) класса user, она выводит только ФИО. Для изменения ФИО и пароля используется функция post класса user, эта функция получается в себя введённые данные пользователем, а затем записывает их в базу данных. Добавление подписи также реализовано через этот раздел. Если подписи нет – пользователю выдаётся информация об отсутствии подписи и кнопка для создания подписи, по нажатию которой появляется всплывающее окно с возможностью нарисовать подпись курсором (см. рисунок 45). Если у пользователя есть подпись – сервис позволяет изменить её. В бэкенде обработкой подписи занимаются функции get и put класса sign (см. рисунок 46). Функция get выводит пользователю изображение из файла подписи, которая расположена в файловой системе веб-сервиса. Форма для рисования подписи реализована средствами JavaScript в фронтенде веб-сервиса. После ввода новой подписи в работу вступает функция put, которая заменяет файл подписи на новый либо создаёт новый файл в случае его отсутствия.

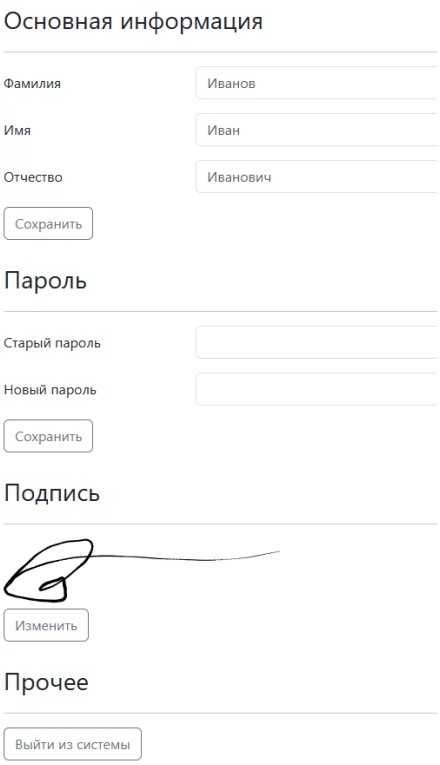


Рисунок 43. Раздел «Настройки профиля»

```python
def get(self):
    if self.status_code != 200:
        return self.response, 401

    user = self.user_info['user']
    user_info = self.rd.hgetall(user)
    user_data = {
        'firstname': user_info['firstname'],
        'lastname': user_info['lastname'],
        'surname': user_info['surname']
    }
    response = headers.create(user_data)
    return response, 200

def post(self):
    if self.status_code != 200:
        return self.response, 401
    user = self.user_info['user']
    request_type = request.json['type']
    if request_type == 'user':
        mapping = {
            'firstname': request.json['firstname'],
            'lastname': request.json['lastname'],
            'surname': request.json['surname']
        }

        self.rd.hset(user, mapping=mapping)

        response = headers.create({'status': 'success'})
        return response, 200
    else:
        user_db = self.rd.hgetall(user)
        old_pass = request.json['old']
        new_pass = request.json['new']

        if not bcrypt.checkpw(old_pass, user_db['hash']):
            response = headers.create({'error': 'Неверный пароль!'})
            return response, 403

        password = bcrypt.hashpw(new_pass, bcrypt.gensalt(12))
        user_db['hash'] = password

        self.rd.hset(user, mapping=user_db)
        response = headers.create({'status': 'success'})
        return response, 200
```

Рисунок 44. Функции вывода и изменения ФИО и пароля

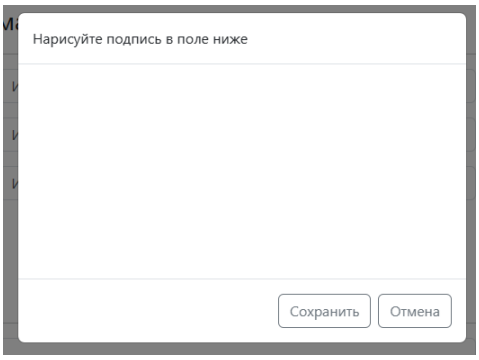


Рисунок 45. Всплывающее окно с формой для создания подписи

```
def get(self):

    if self.status_code != 200:
        return self.response, 401

    user_file = 'files/signs/' + self.user

    if not os.path.isfile(user_file):
        response = headers.create({'error': 'Подпись не найдена!'})
        return response, 404

    config = configparser.ConfigParser()
    config.read('config')
    url = config['Path']['url']
    response = make_response(send_file(user_file))
    response.headers.add('Access-Control-Allow-Headers', 'content-type')
    response.headers.add('Access-Control-Allow-Origin', url)
    response.headers.add('Access-Control-Allow-Credentials', 'true')
    response.headers.add('Access-Control-Allow-Methods', 'GET, POST, PUT')
    return response, 200

def put(self):
    if self.status_code != 200:
        return self.response, 401

    user_file = 'files/signs/' + self.user

    file = request.form['file']
    f = open(user_file, 'w')
    f.write(file)
    f.close()

    response = headers.create({'status': 'success'})
    return response, 200
```

Рисунок 46. Функции вывода и загрузки подписи

Выход пользователя из сервиса осуществляется кнопкой «Выйти из системы» (см. рисунок 43) которая расположена в разделе настроек профиля. В бэкенде это реализовано функцией delete (см. рисунок 47) класса auth. Данная функция работает очень просто и удаляет из данных браузера куку token.

```
def delete(self):
    token = request.cookies.get('token')
    user_json = json.loads(token.replace("'", '"'))
    user = user_json['user']
    self.rd.hset(user, mapping={'token': ''})
    response = headers.create({'status': 'success'})
    return response, 200
```

Рисунок 47. Функция выхода пользователя из системы

2.3 Результаты апробации, техническая документация

После разработки веб-сервиса электронного подписания документов – данный веб-сервис был установлен на локальную виртуальную машину внутри компании, затем часть сотрудников компании внутри которой был развёрнут веб-сервис получила ссылку для входа в веб-сервис, руководство пользователя и ссылку на лист опроса с рядом критериев, каждый из которых можно оценить по 5-бальной шкале, где 0 – «очень плохо», а 5 – «очень хорошо».

Критерии, представленные в опросном листе, были следующие:

1. Удобство пользовательского интерфейса
2. Привлекательность пользовательского интерфейса
3. Корректность работы
4. Удовлетворённость веб-сервисом

Результаты опроса представлены в Таблице 1.

Таблица 1
Экспертные оценки, уровень согласованности экспертов

		Эксперт, оценка							
		1	2	3	4	5	6	Среднее	Степень согласованности
Критерий	Удобство пользовательского интерфейса	5	5	5	5	5	5	5	Высокая
	Привлекательность пользовательского интерфейса	4	5	5	4	5	5	4,7	Высокая
	Корректность работы	5	5	5	5	5	5	5	Высокая
	Удовлетворённость веб-сервисом	5	5	5	5	5	5	5	Высокая

Среди замечаний и рекомендаций по совершенствованию веб-сервиса электронного подписания документов были следующие: «Скучный и очень

простой интерфейс», «Отсутствуют всплывающие подсказки на всех этапах работы в веб-сервисе», «В некоторых местах без руководства пользователя может быть сложно разобраться».

Таким образом, по результатам проведённого опроса, веб-сервис электронного подписания документов, готов к работе и дальнейшему распространению.

Руководство пользователя по работе с веб-сервисом электронного подписания документов подготовлено в программе «Dr. Explain» в формате .pdf. Структура руководства пользователя полностью соответствует ГОСТ. Руководство пользователя содержит в себе картинки с детальным описанием полей ввода и элементов веб-сервиса.

Заключение

При выполнении данной выпускной работы было разработано задание на разработку веб-сервиса электронного документооборота и разработан сам веб-сервис. Разработанный веб-сервис соответствует всем требованиям, указанным в техническом задании.

Для достижения поставленной цели были решены следующие задачи:

1. Выполнен анализ основных принципов электронного документооборота, выделены различные типы систем электронного документооборота, выделены их достоинства и недостатки существующих систем электронного подписания документов.

2. Выполнен анализ методов и средств разработки систем электронного документооборота. Выполнен анализ различных языков программирования и их фреймворков, выделены основные методы разработки.

3. Подготовлено техническое задание на разработку веб-сервиса электронного подписания документов.

4. Разработан веб-сервис электронного подписания документов на основании технического задания. Проведён опрос о функциональности веб-сервиса среди потенциальных пользователей.

5. Подготовлена техническая документация в виде руководства пользователя и руководства по установке на виртуальную машину в .pdf формате.

Таким образом, следует считать, что результаты разработки соответствуют всем требованиям, указанным в техническом задании, работа носит законченный характер, поставленные задачи выполнены, а итоговая цель работы достигнута.

Список информационных источников

1. ГОСТ 34.602-89. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы : [сайт]. – URL : https://docs.cntd.ru/document/1200006924 (дата обращения: 29.09.2023).
2. ГОСТ 34.601-90. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания : [сайт]. – URL : https://docs.cntd.ru/document/1200006921 (дата обращения: 29.09.2023).
3. ГОСТ 34.201-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем : [сайт]. – URL : https://docs.cntd.ru/document/1200006974 (дата обращения: 29.09.2023).
4. РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов : [сайт]. – URL : https://docs.cntd.ru/document/1200006978 (дата обращения: 29.09.2023).
5. ГОСТ Р 7.0.100 – 2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления : [сайт]. – URL : https://docs.cntd.ru/document/1200161674 (дата обращения 07.01.2024).
6. Руководство по языку программирования Python : [сайт]. – URL : https://metanit.com/python/tutorial/ (дата обращения: 05.01.2024).
7. Руководство по JavaScript : [сайт]. – URL : https://metanit.com/web/javascript/ (дата обращения: 05.01.2024).
8. Руководство по React : [сайт]. – URL : https://metanit.com/web/react/ (дата обращения: 05.01.2024).
9. Документация Bootstrap на русском языке : [сайт]. – URL : https://bootstrap-4.ru/docs/5.3/getting-started/introduction/ (дата обращения: 05.01.2024).
10. Python : [сайт]. – URL : https://python.org/ (дата обращения: 05.01.2024).
11. Redis : [сайт]. – URL : https://redis.io/docs/ (дата обращения: 05.01.2024).

12. Flask : [сайт]. – URL : https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 05.01.2024).
13. Bcrypt : [сайт]. – URL : https://github.com/pyca/bcrypt/ (дата обращения: 05.01.2024).
14. Начало работы с Python в Windows для начинающих. – Microsoft : [сайт]. – URL : https://learn.microsoft.com/ru-ru/windows/python/beginners (дата обращения: 05.01.2024).
15. Создание веб-приложения с помощью Flask в Python 3 : [сайт]. – URL : https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3-ru (дата обращения: 05.01.2024).
16. Microsoft Visual Studio Code : [среда разработки]. – URL : https://code.visualstudio.com/download (дата обращения: 05.01.2024).
17. Redis в Python — Полная документация на примерах : [сайт]. – URL : https://python-scripts.com/redis (дата обращения: 05.01.2024).
18. Flask и React — от нуля до Full Stack проекта (с примерами) : [сайт]. – URL : https://python-scripts.com/flask-react-from-zero (дата обращения: 06.01.2024).
19. Flask на примерах — Настройка проекта : [сайт]. – URL : https://python-scripts.com/flask-project-setup (дата обращения: 06.01.2024).
20. Nginx Documentation : [сайт]. – URL : https://nginx.org/en/docs/ (дата обращения: 10.01.2024).
21. Draw.io : [сайт]. – URL : https://www.drawio.com/ (дата обращения: 10.01.2024).
22. Управление строками в Redis : [сайт]. – URL : https://www.8host.com/blog/upravlenie-strokami-v-redis/ (дата обращения: 18.01.2024).
23. Хеширование паролей в Python: туториал по Bcrypt с примерами Хеширование паролей в Python: туториал по Bcrypt с примерами : [сайи]. – URL : https://pythonist.ru/heshirovanie-parolej-v-python-tutorial-po-bcrypt-s-primerami/ (дата обращения: 18.01.2024).

24. Хэш-пароли с использованием библиотеки bcrypt в Python : [сайт]. – URL : https://dev-gang.ru/article/heshparoli-s-ispolzovaniem-biblioteki-bcrypt-v-python-u8pq4bpzyb/ (дата обращения: 18.01.2024).
25. Flask-Bcrypt : [сайт]. – URL : https://flask-bcrypt.readthedocs.io/en/1.0.1/ (дата обращения: 18.01.2024).
26. Мега-Учебник Flask, Часть 1: «Привет, Мир!» : [сайт]. – URL : https://habr.com/ru/articles/193242/ (дата обращения: 18.01.2024).
27. Мега-Учебник Flask, Часть 2: Шаблоны : [сайт]. – URL : https://habr.com/ru/articles/193260/ (дата обращения: 18.01.2024).
28. Мега-Учебник Flask, Часть 3: Формы : [сайт]. – URL : https://habr.com/ru/articles/194062/ (дата обращения: 18.01.2024).
29. Мега-Учебник Flask, Часть 4: База данных : [сайт]. – URL : https://habr.com/ru/articles/196810/ (дата обращения: 18.01.2024).
30. Redis для начинающих : [сайт]. – URL : https://webdevblog.ru/redis-dlya-nachinajushhij/ (дата обращения: 18.01.2024).
31. Redis: начало работы и основные команды : [сайт]. – URL : https://timeweb.cloud/tutorials/redis/nachalo-raboty-i-osnovnye-komandy (дата обращения: 18.01.2024).
32. Простой веб-сервер с использованием Python и Flask : [сайт]. – URL : https://codex.so/python-flask (дата обращения: 18.01.2024).
33. Краткое руководство. Создание первого веб-приложения Python с помощью Visual Studio : [сайт]. – URL : https://learn.microsoft.com/ru-ru/visualstudio/ide/quickstart-python?view=vs-2022 (дата обращения: 18.01.2024).
34. Visual Studio | Документация по Python : [сайт]. – URL : https://learn.microsoft.com/ru-ru/visualstudio/python/?view=vs-2022 (дата обращения: 18.01.2024).
35. Настройка NGINX + uWSGI для Python : [сайт]. – URL : https://www.dmosk.ru/miniinstruktions.php?mini=uwsgi-nginx (дата обращения: 18.01.2024).

36. Запуск Python 3 на веб-сервере nginx в Debian 11 и Ubuntu Server 21.10 : [сайт]. – URL : https://dondub.com/2022/03/zapusk-python-3-na-veb-servere-nginx-v-debian-11-i-ubuntu-server-21-10/ (дата обращения: 18.01.2024).
37. How To Serve Flask Applications with uWSGI and Nginx on Ubuntu 18.04 : [сайт]. – URL : https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-uswgi-and-nginx-on-ubuntu-18-04 (дата обращения: 18.01.2024).
38. Debian : [сайт]. – URL : https://www.debian.org/ (дата обращения: 18.01.2024).
39. Веб-сервисы в теории и на практике для начинающих : [сайт]. – URL : https://habr.com/ru/articles/46374/ (дата обращения: 18.01.2024).
40. Создание веб-приложения с использованием Python Flask : [сайт]. – URL : https://timeweb.cloud/tutorials/python/sozdanie-veb-prilozheniya-s-ispolzovaniem-python-flask (дата обращения: 18.01.2024).
41. Примеры работы с NoSQL базой данных Redis из Python : [сайт]. – URL : https://900913.ru/2021/01/05/nosql-database-redis-with-python-examples/ (дата обращения: 18.01.2024).
42. Самоучитель по Python для начинающих. Часть 23: Основы веб-разработки на Flask : [сайт]. – URL : https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-23-osnovy-veb-razrabotki-na-flask-2023-06-27 (дата обращения: 18.01.2024).
43. Уроки по Flask на русском : [сайт]. – URL : https://pythonru.com/tag/uroki-po-flask-na-russkom (дата обращения: 18.01.2024).
44. Python цикл for — for i in range : [сайт]. – URL : https://pythonru.com/osnovy/python-cikl-for-for-i-in-range (дата обращения: 18.01.2024).
45. Signature Pad : [сайт]. – URL : https://github.com/szimek/signature_pad (дата обращения: 18.01.2024).
46. Что такое электронный документооборот? : [сайт]. – URL : https://www.diadoc.ru/docs/faq/faq-166 (дата обращения: 18.01.2024).

47. Что такое электронный документооборот и зачем он нужен : [сайт]. – URL : https://ofd.ru/blog/cases/chto-takoe-elektronnyy-dokumentooborot-i-zachem-nuzhen (дата обращения: 18.01.2024).
48. Калугина Е.А. СИСТЕМА ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА, ЕЕ ПРЕИМУЩЕСТВА И ПЕРЕХОД НА ЭЛЕКТРОННЫЙ ДОКУМЕНТООБОРОТ // Вестник Национального Института Бизнеса. 2019. № 37. С. 110-113.
49. Ипполитов К.Г. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА PYTHON // Учебное пособие / Казань, 2023.Издательство: Общество с ограниченной ответственностью "Редакционно-издательский центр "Школа"
50. Парамонов И.В. ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA И JAVA-ТЕХНОЛОГИИ // учебное пособие / Ярославль, 2006.
51. Дунаев В.В. САМОУЧИТЕЛЬ JAVASCRIPT // (3-е изд) Санкт-Петербург, 2010.
52. Скачков А.Е. ОБЗОР ИСПОЛЬЗОВАНИЯ PHP И MYSQL В ПРОГРАММИРОВАНИИ // В сборнике: Информационные системы и технологии: теория и практика. Сборник научных трудов. Отв. редактор М.Р. Вагизов. Санкт-Петербург, 2023. С. 141-146.
53. Грузин Н.А. СРАВНЕНИЕ ФРЕЙМВОРКОВ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ: LARAVEL, DJANGO И FLASK // Modern Science. 2020. № 2-1. С. 359-364.
54. Насиров Э.Ф. ФРЕЙМВОРК JAVASCRIPT REACT JS // В сборнике: НАУКА И ОБРАЗОВАНИЕ: СОХРАНЯЯ ПРОШЛОЕ, СОЗДАЁМ БУДУЩЕЕ. сборник статей XXV Международной научно-практической конференции : в 2 ч.. 2019. С. 62-64.
55. Дунаевский А.С. ФРЕЙМВОРК BOOTSTRAP – ИНСТРУМЕНТ РАЗРАБОТКА СОВРЕМЕННОГО ВЕБ-САЙТА // В книге: Дни науки КФУ им. В.И.Вернадского. Материалы III научно-практической конференции профессорско-преподавательского состава, аспирантов, студентов и молодых ученых. Гуманитарно-педагогическая академия. 2017. С. 177-179.

56. Костенко И.П. О ПРЕИМУЩЕСТВАХ СУБД REDIS // В книге: Актуальные проблемы науки и техники. 2022. Материалы Всероссийской (национальной) научно-практической конференции. Отв. редактор Н.А. Шевченко. Ростов-на-Дону, 2022. С. 344-345.

Приложения

Приложение 1.

Электронный носитель, содержащий:

- Архив «sign.tar» с веб-сервисом электронного подписания документов.
- Руководство пользователя «руководство пользователя.pdf» с подробным руководством пользователя веб-сервиса для электронного подписания документов.

Название	Ссылка	QR-код
Код веб-сервиса	https://drive.google.com/file/d/1T-hCGpiWkbEWEMM2o17RZcifP-9enyKA/view?usp=sharing	
Руководство пользователя	https://drive.google.com/file/d/1GXzPiw9N7YxWSwbRtxTqzTzTjb891k6Q/view?usp=sharing	

Приложение 2.

Руководство по установке веб-сервиса для электронного подписания документов на сервер под управлением ОС Debian.

Установка веб-сервиса осуществляется по шагам, описанным ниже:

1. Установить необходимые приложения выполнив команду: **apt install redis nginx python3-pip python3-venv uwsgi**
2. Создать пользователя выполнив команду: **useradd webuser**
3. Загрузить архив с веб-сервисом на сервер и разархивировать его выполнив команду: **tar -xvf sign.tar -C /**
4. Перейти в каталог выполнив команду: **cd /opt/sign/backend/**
5. Выдать права на каталог с веб-сервисов ранее созданному пользователю выполнив команду: **chown -R webuser: /opt/sign/backend**
6. Сделать замену адреса бэкенда в фронденте выполнив команду предварительно заменив ip адрес 192.168.88.16 на текущий адрес сервера: **sed -i 's/localhost/192.168.88.16/g' /opt/sign/frontend/static/js/main.87dbf9d8.js**
7. Отредактировать файл конфигурации Nginx выполнив команду: **vim /etc/nginx/nginx.conf**
8. В строке user прописать имя ранее созданного пользователя webuser
9. После **types_hash_max_size 2048;** прописать **client_max_body_size 100M;**
10. Создать сервис бэкенда выполнив команду: **vim /etc/systemd/system/sign-backend.service** и вписать туда следующие строки:

```
[Unit]
Description=uWSGI instance to serve sign backend
After=network.target
[Service]
User=webuser
Group=webuser
WorkingDirectory=/opt/sign/backend
Environment="PATH=/opt/sign/backend/bin"
ExecStart=/opt/sign/backend/bin/uwsgi --ini backend.ini
[Install]
WantedBy=multi-user.target
```

11. Перезапустить сервисы выполнив команду: **systemctl daemon-reload**
12. Запустить службу бэкенда выполнив команду: **systemctl start sign-backend**

13. Заменить конфигурацию веб-сервера Nginx выполнив команду: **vim /etc/nginx/sites-available/default** на следующие строки, заменив ip адрес 192.168.88.16 на текущий адрес сервера:

```
server {
    listen 80;
    server_name 192.168.88.16;
    root /opt/sign/frontend;
    index index.html;
    location / {
        try_files $uri /index.html;
    }
}
server {
    listen 9200;
    server_name 192.168.88.16;
    location / {
        include uwsgi_params;
        uwsgi_pass unix:/opt/sign/backend/backend.sock;
    }
}
```

14. Веб-сервис электронного подписания документов готов к работе

Приложение 3.

Руководство администратора веб-сервиса для электронного подписания документов

Администратору веб-сервиса для электронного подписания документов доступно управление пользователями (удаление ненужных пользователей) через консоль базы данных Redis. Для подключения к консоли Redis администратору необходимо подключиться к серверу, на котором расположен веб-сервис и выполнить команду redis-cli после чего, откроется консоль Redis. Для отображения списка всех зарегистрированных пользователей необходимо ввести команду keys * которая отобразит список логинов зарегистрированных пользователей. Для удаления пользователя необходимо ввести команду DEL user_u, где вместо user_u необходимо указать логин пользователя, которого необходимо удалить.