Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт космических и информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. С. Кузнецов
подпись
«_____» _____ 2022г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 – Программная инженерия

Разработка и реализация алгоритмов конвертации PDF файлов

Руководитель _____ доцент, канд. техн. наук А.В. Хныкин
подпись, дата

Выпускник _____ А.А. Газзаев
подпись, дата

Нормоконтролер _____ ст. преподаватель Н.Б. Позолотина
подпись, дата

Красноярск 2022

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт космических и информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ

Заведующий кафедрой

_____ А. С. Кузнецов
подпись

«_____» _____ 2022г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Газзаеву Андрею Александровичу

Группа КИ18-16б Направление (специальность) 09.03.04 Программная инженерия

Тема выпускной квалификационной работы разработка и реализация алгоритмов конвертации PDF файлов

Утвержденная приказом по университету № 6060/с от 21.04.2022

Руководитель ВКР А.В. Хныкин, кандидата технических наук, доцента базовой кафедры интеллектуальных систем управления, Института космических и информационных технологий.

Исходные данные для ВКР описание предметной области, технические требования.

Перечень разделов ВКР введение, анализ предметной области, управление и проектирование, разработка серверной части веб-сервиса, заключение.

Перечень графического материала презентация

Руководитель ВКР _____ А.В. Хныкин
подпись

Задание принял к исполнению _____ А. А. Газзаев
подпись

«_____»____20___г.

РЕФЕРАТ

Выпускная квалификационная работа на тему «Разработка и реализация алгоритмов конвертации PDF файлов» содержит 47 страницы текстового документа, 51 рисунок и 26 использованных источников.

СЕРВЕРНАЯ ЧАСТЬ, СЕРВИС ПО КОНВЕРТАЦИИ И ХРАНЕНИЮ PDF-ФАЙЛОВ, ВЕБ-СЕРВИС, NODE.JS, TYPESCRIPT

Целью выпускной квалификационной работы является разработка и реализация алгоритмов конвертации PDF файлов, путем разработки серверной части веб-сервиса для хранения и конвертации pdf файлов.

Для достижения поставленной цели были решены следующие задачи:

- провести анализ существующих аналогов;
- определить стек технологий разрабатываемого продукта;
- разработать серверную часть части веб-сервиса для хранения и конвертации pdf-файлов.

В результате бакалаврской работы были разработаны и реализованы алгоритмы конвертации PDF файлов, путем создания серверная часть веб-сервиса для хранения и конвертации pdf файлов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На данный момент времени формат PDF [1] является самым надежным средством представления информации. Формат PDF гарантирует, что шрифты, изображения, графики и форматирование исходного файла останутся неизменными даже со сменной версии PDF.

Из плюсов PDF вытекают и минусы, такие как: не возможность изменить PDF-файл стандартными средствами или восстановить исходный файл. Поэтому приходится постоянно хранить два экземпляра файла, что в свою очередь приводит к путанице и риску потери исходного файла.

Конечно, существует множество сервисов предоставляющих доступ к конвертации pdf-файлов, таких как ilovepdf [2], pdf2go [3], smallpdf [4] и т.д. Данные сервисы могут решить одну из проблем, но и добавить другую, а именно конфиденциальность файлов. Дело в том, что серверы данных сервисов находятся в других странах (не в России). К тому же для хранения файлов приходится использовать другие сервисы. Что так же приводит к риску компрометации файлов.

Таким образом, появляется цель разработать серверную часть веб-сервис для хранения и конвертации pdf-файлов, который будет размещен на серверах, находящихся в России.

Интерфейс веб-сервиса будет выполнен с учетом UX/UI (UX — это функционал интерфейса, UI — его внешний вид) дизайна. Также интерфейс должен быть адаптивным и доступным на большинстве устройств.

Конвертация будет представлять собой набор функций, таких как: конвертация pdf-файла в jpg-файлы и docx-файл [5], создание pdf-файла из jpg-файлов, объединение pdf-файлов в один, разделение pdf-файла на несколько, хранение pdf-файлов (соответственно просмотр, скачивание и загрузка).

Архитектура сервиса будет client side rendering с использованием API, предоставленным сервером. То есть проект будет представлять собой две части: Front-end и Back-end.

Для разработки веб-сервиса будут использоваться новейшие технологии. Для Front-end используется Angular.js [6]. Для Back-end используется node.js [7] в связке с Express.js [8].

В данной работе будет представленная подробное описание работы относительно back-end.

1 Анализ предметной области

Целевая область проекта – конвертация и хранение pdf-файлов. Платформой для взаимодействия с программой является браузер, так как веб-сервисы считаются наиболее удобными и доступными средствами получения услуг (в дальнейшем программа будет называться веб-сервисом). Основные пользователи данного продукта являются все те, кто, так или иначе, работают с документами (к примеру: студенты, преподаватели, учителя и др.).

1.1 Актуальность и цель работы

Существует множество веб-сервисов по конвертации pdf-файлов, но сервера данных сервисов находятся в других странах, что может сказаться на конфиденциальности файлов в этих сервисах. Также в связи с уходом многих поставщиков программного обеспечения (ПО) с рынка, появляется риск отсутствия доступности к этим сервисам. Также ни в одном из сервисов-конкурентов, представленных ниже, нет возможности хранения pdf-файлов, что ставит под вопрос удобство этих сервисов.

Таким образом, необходимо разработать веб-сервис по конвертации и хранению pdf-файлов.

1.2 Анализ существующих решений

Перед подробным анализом конкурентов стоит заметить, что большая часть возможностей у них совпадает.

1.2.1 Сервис по конвертации pdf-файлов “I love pdf”

На данный момент www.ilovepdf.com является самым популярным сервисом по версии google [9]. Он предоставляет множество возможностей по

работе с pdf-файлами, таких как организация, оптимизация, конвертация из, конвертация в, редактирование и защита pdf-файлов (рисунок 1.1).

Рисунок 1.1 – Все бесплатные возможности сервиса

Также у сервиса есть desktop версия для работы offline (рисунок 1.2).

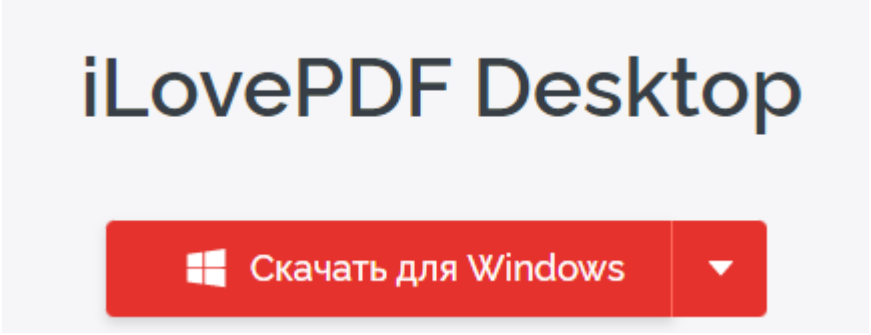


Рисунок 1.2 – Версия сервиса для ПК

Сервис имеет поддержку множества языков (рисунок 1.3).

Рисунок 1.3 – Виды выбора языка

И в сервисе присутствуют платные возможности такие, как конвертация в нестандартные или не популярные форматы, работа с offline версией продукта, отсутствие рекламы и т.д. (рисунок 1.4).

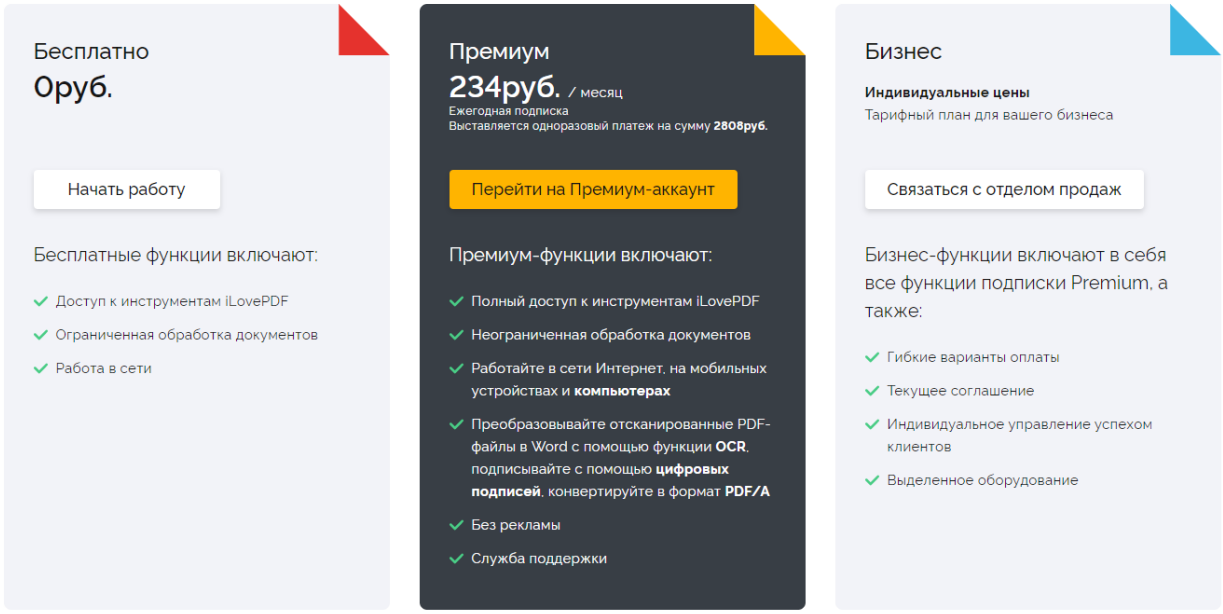


Рисунок 1.4 – Платная подписка на сервис

С помощью сервиса по анализу ip 2ip.ru [10], было определено расположение сервера. Из представленных данных (рисунок 1.5) можно однозначно определить, где располагается сервер данного сервиса.

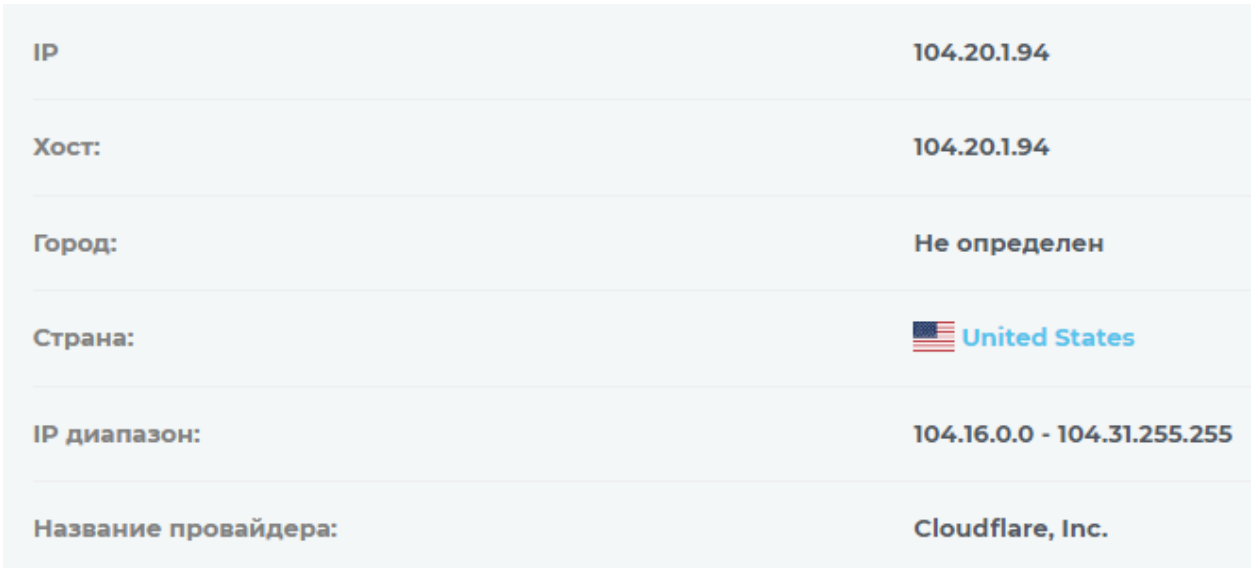


Рисунок 1.5 – Расположение сервиса

1.2.2 Сервис по конвертации pdf-файлов “pdf 2 go”

На данный момент www.pdf2go.com выдается второй ссылкой в google и поэтому считается одним из популярных сервисов. Исключительно особенностью данного сервиса является возможность редактирования pdf-файлов. В остальном в нем повторяются все возможности, как и в предыдущем конкуренте (рисунок 1.6).

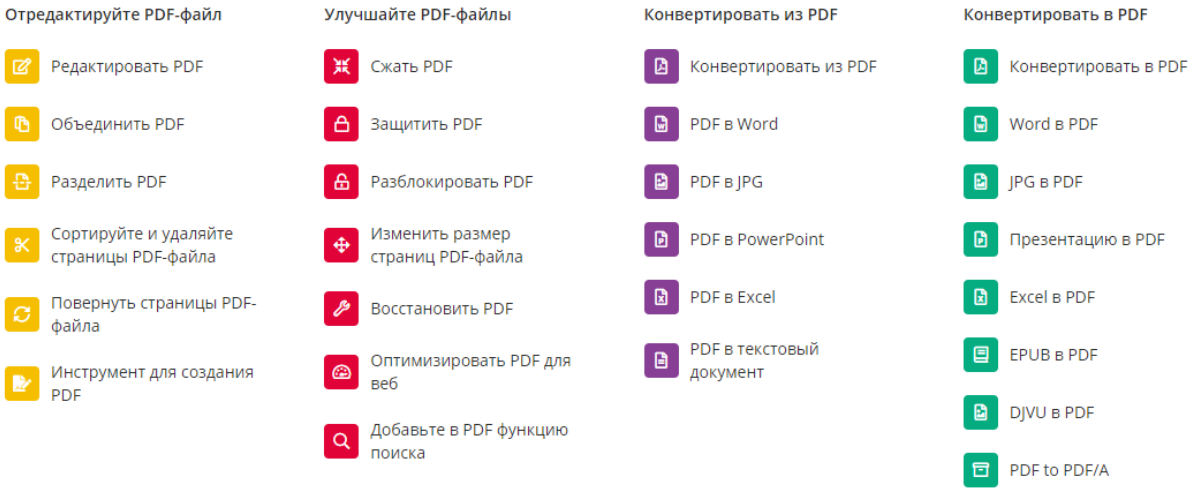


Рисунок 1.6 – Бесплатные возможности сервиса

У него также присутствует desktop-версия продукт и платная подписка (рисунок 1.7, 1.8).

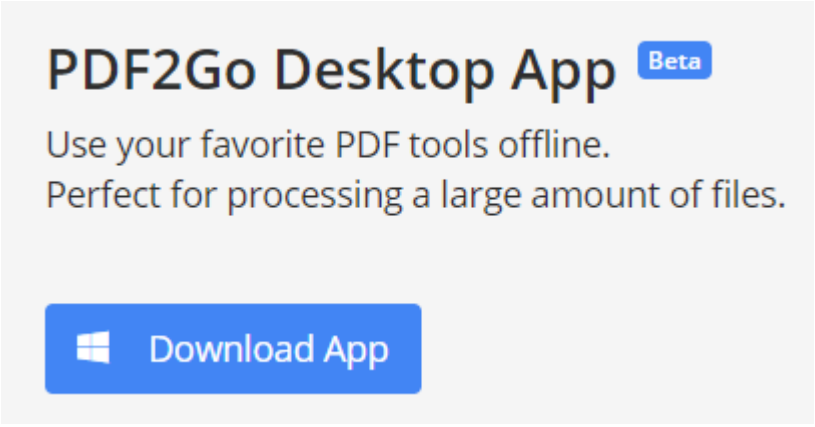


Рисунок 1.7 – Desktop-версия

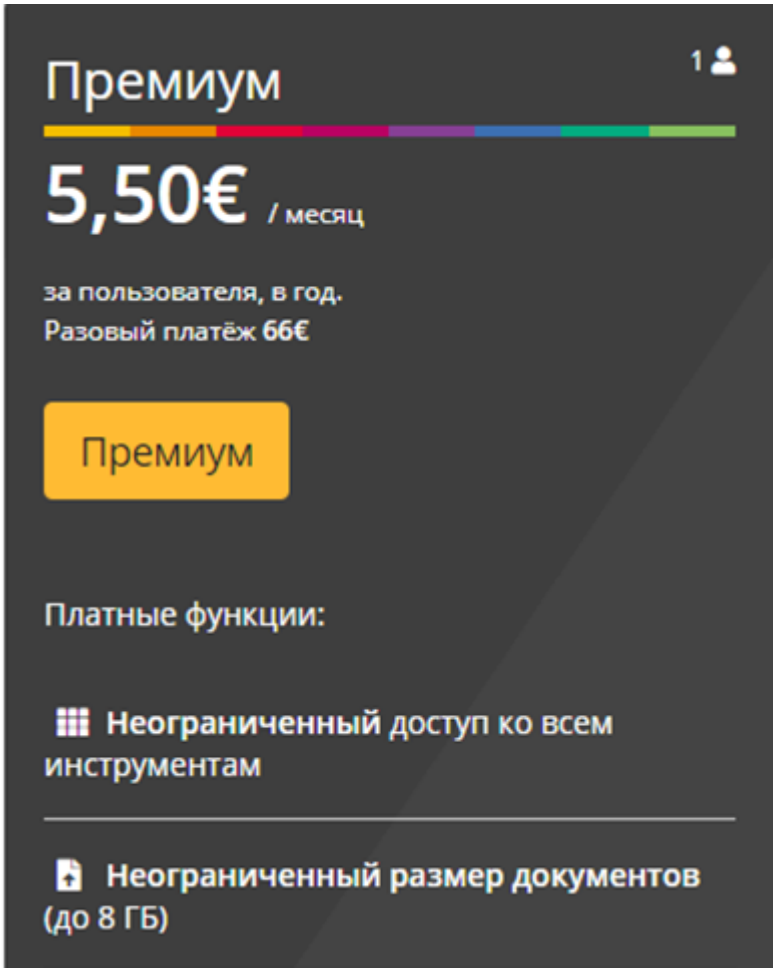


Рисунок 1.8 – Платная подписка

По данным сервиса 2ip.ru можно сделать вывод, что сервера находятся за границей (рисунок 1.9).

IP	104.22.78.171
Хост:	104.22.78.171
Город:	Не определен
Страна:	United States
IP диапазон:	172.64.0.0 - 172.71.255.255
Название провайдера:	Cloudflare, Inc.

Рисунок 1.9 – Расположение сервиса

1.2.3 Сервис по конвертации pdf-файлов “small pdf ”

В smallpdf.com повторяются основные возможности сервисов по конвертации. Но этот сервис ещё имеет IOS и Android версии (рисунок 1.10, 1.11).

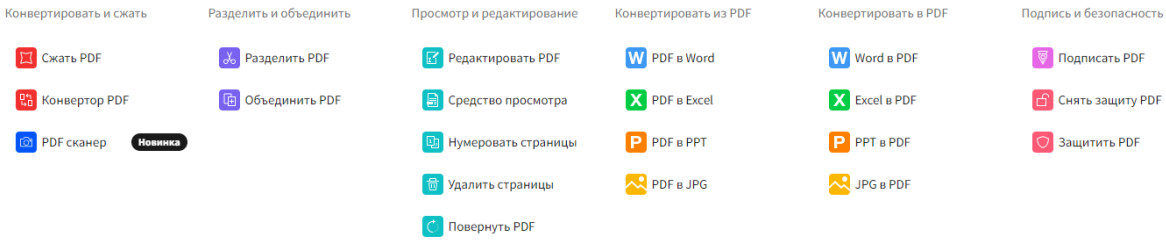


Рисунок 1.10 – Бесплатные возможности сервиса

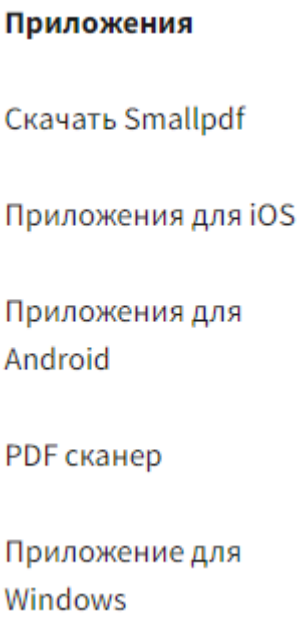


Рисунок 1.11 – Версии сервиса для скачивания

Так же сервис, по сравнению с предыдущими конкурентами, имеет расширенный функционал, относящийся к коммерциализации (рисунок 1.12).

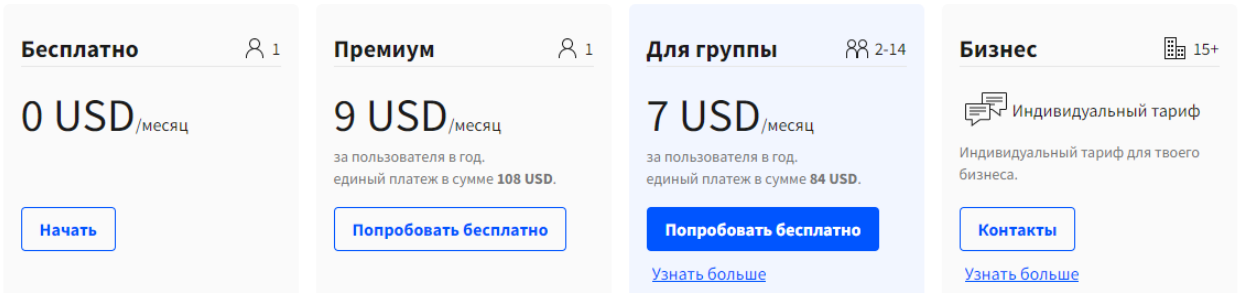


Рисунок 1.12 – Виды подписок

Все также по данным сервиса 2ip.ru, делается вывод, что сервера находятся за границей (рисунок 1.13).

Рисунок 1.13 – Расположение сервиса

1.3 Определение требований к системе

Проанализировав конкурентов в подразделе 1.2, были составлены следующие ключевые функциональные требования к системе:

– просмотр конкретного файла;

– конвертация pdf-файла в jpg-файлы;

– создание pdf-файла из jpg-файлов;

– объединение pdf-файлов;

– разделение pdf-файла;

– конвертация pdf-файла в word;

– загрузка файлов;

– скачивание файлов;

– удаление файлов;

– просмотр всех файлов.

В связи с ограниченностью времени были выбраны для реализации самые основные функции по конвертации.

1.4 Выбор инструментов разработки

В данный момент есть множество способов и инструментов реализации данного проекта. Проект представляет собой классическое веб-приложение, где backend – серверная часть, а frontend – клиентская часть.

Так как проект реализуют 2 человека, то и реализация была поделена на две части backend и frontend. В данном случаи в отчете будет представлена только реализация backend-части.

1.4.1 Проектирование архитектуры сервиса

Для проектирования архитектуры баз данных использовался веб-редактор draw.io и UML.

Draw.io [11] — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

UML [12] (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования

в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

1.4.2 Разработка серверной части

Серверная часть написана на node.js. Эта технология позволяет запускать javascript [13] код отдельно от браузера. Это значит, что node.js позволяет реализовывать серверную часть программы, предоставляющую API клиентской части. Node.js был выбран из-за развитых навыков в javascript.

Также использовался framework для node.js под название express.js. Он упрощает и ускоряет разработку серверной части. Так как имеет множество возможностей “из коробки”.

В качестве базы данных был выбран PostgreSQL [14] – это реляционная база данных. Данная технология позволяет хранить и организовывать сложные структуры данных. Для взаимодействия с базой данных был выбран ORM [15] (Object-Relational Mapping) такой как sequelize [16]. Объектная модель представления данных позволяет работать с базой данных, не писать вручную SQL-запросы, а пользоваться уже заранее реализованными функциями. Да и в коде, работать с заранее готовыми объектами гораздо удобней.

Для реализации ряда функций, в системе нужна регистрация, авторизация и аутентификация. Для этого используется две технологии:

– JWT [17] (JSON Web Tokens) – за счет этого токена происходит авторизация пользователя в системе;

– OAuth2.0 [18] – протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе. Протокол избавляет от необходимости доверять приложению логин и пароль, а также позволяет выдавать ограниченный набор прав, а не все сразу.

Для работы с файлами был выбран Yandex Object Cloud [19], документы в хранилище хранятся в файловой системе, то есть, папка и в не хранятся файлы пользователя.

1.5 Методология разработки

Гибкая методология разработки (англ. agile software development) является самым популярным подходом к разработке ПО. Agile – методология, основывающаяся на цикличном методе, в котором условия и решения развиваются с помощью взаимной работы самоорганизующихся кросс-функциональных команд между собой. К гибким методологиям, в частности, относят Scrum. Данный метод использовался при разработке данного проекта.

SCRUM — минимально необходимый набор мероприятий, артефактов, ролей, на которых строится процесс SCRUM-разработки, позволяющий за фиксированные небольшие промежутки времени, называемые спринтами (sprints), предоставлять конечному пользователю работающий продукт с новыми бизнес-возможностями, для которых определён наибольший приоритет. В конце спринта Scrum-команда встречается на обзорном совещании результатов спринта с заказчиком, и представляет ему инкремент бизнес-продукта, который она успела сделать за спринт.

Так же для отслеживания задач и управления проектом использовалась kanban доска JIRA [20] (рисунок 1.14).

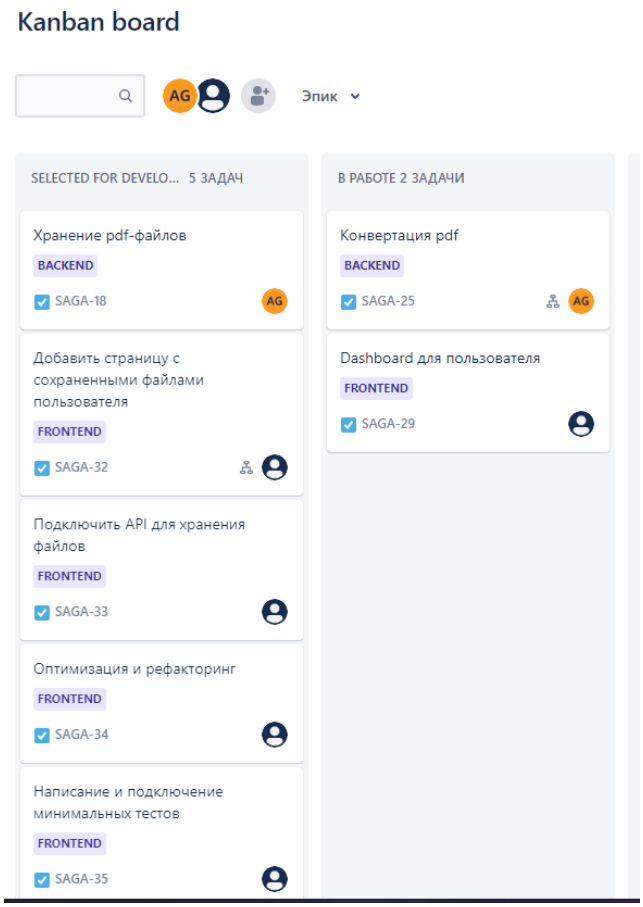


Рисунок 1.14 – kanban доска

При разработке веб-приложения использовалась система контроля версий Git [21], репозиторий размещался на хостинге GitHub [22]. Git позволил соблюдать версионность и отслеживать изменение в коде сервиса.

Git — абсолютный лидер по популярности среди современных систем управления версиями с распределенной архитектурой.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

1.6 Выводы по разделу

Целью раздела являлось изучение предметной области, в результате которого были проанализированы конкуренты разрабатываемого веб-сервиса.

Был полностью определен технологически стек для разработки и проектирования серверной части приложения: Node.js – это программная платформа, которая транслирует JavaScript в машинный код, исполняемый на стороне сервера. Таким образом, JavaScript можно использовать для создания серверной части. Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. PostgreSQL – свободная объектно-реляционная система управления базами данных (СУБД). Sequelize — это современная ORM TypeScript и Node.js для Postgres, MySQL, MariaDB, SQLite и SQL Server и других. Благодаря надежной поддержке транзакций, отношениям, нетерпеливой и ленивой загрузке, репликации чтения и многому другому. Draw.io — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого.

Также в качестве методологии разработки был выбран Scrum. В качестве системы контроля версий было решение использовать Git и разместить репозиторий на платформе GitHub.

2 Управление и проектирование

В каждом проекте, где участвует больше одного человека, необходимо организовать связь и отслеживание задач между участниками команды. Для работы с кодом также используются специальные программы (VS code [23], Git, GitHub).

2.1 Управление задачами и коммуникация

Управление задачами участников осуществлялось через платформу Jira. Эта платформа обладает широким функционалом представления задач. Jira — коммерческая система отслеживания ошибок, предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для управления проектами. Она позволяет не только отслеживать задачи, но и назначать их на конкретного члена команды, отслеживать сроки выполнения и организация жизненного цикл задачи (рисунок 2.1).

Рисунок 2.1 – Жизненный цикл задач

К тому же JIRA позволяет реализовать методологию agile в полной мере. JIRA имеет гибкий функционал создания карточек заданий. После создания карточки в неё можно добавить описание, прикрепить рисунок, разделить задание на под задачи, назначить время, человека и метку, оставить комментарий (Рисунки 2.2 – 2.6).

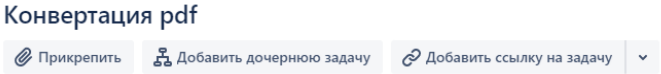


Рисунок 2.2 – Возможность добавить изображение или под задачи

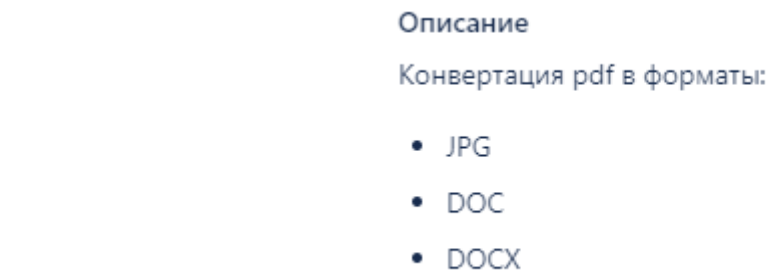


Рисунок 2.3 – Описание карточки

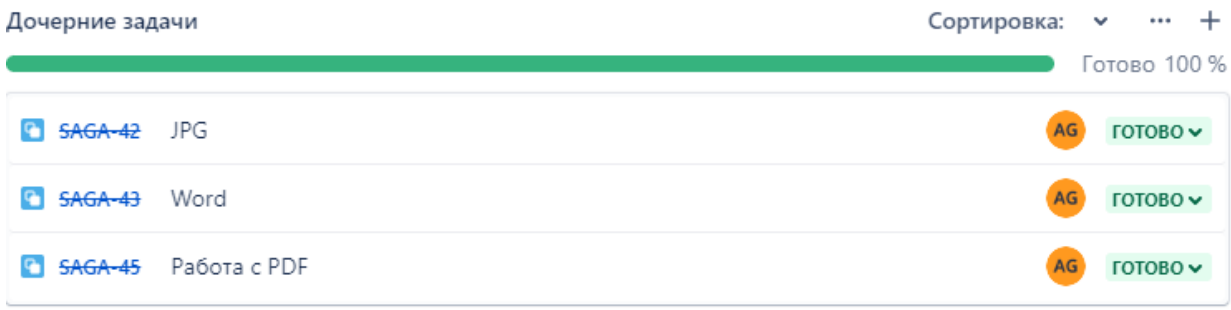


Рисунок 2.4 – Прогресс карточки

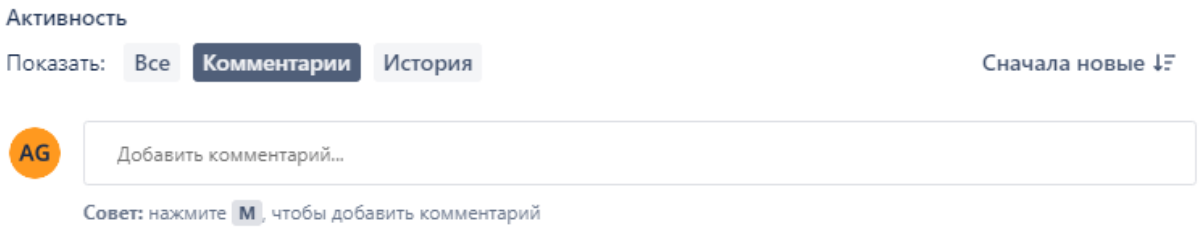


Рисунок 2.5 – Комментарии карточки

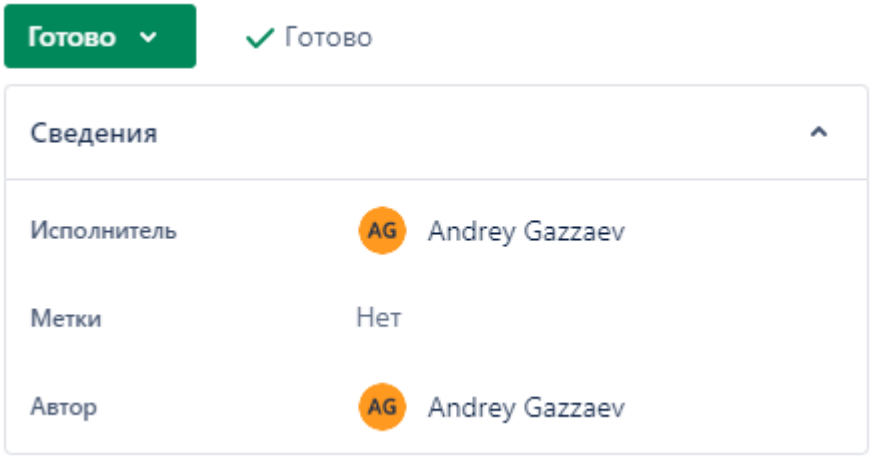


Рисунок 2.6 – Сведения карточки

Перед началом работы была создана доска. Данная доска содержит четыре колонки, которые содержат определенные задания или информацию.

Первая колонка содержит все задания, которые нужно реализовать. Разработчик сам определяет приоритет задачи и выбирает самую приоритетную (рисунок 2.7).

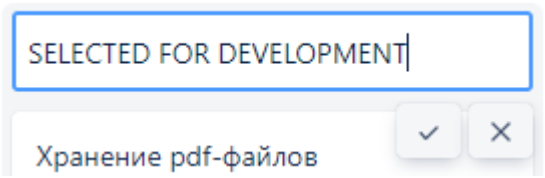


Рисунок 2.7 – Колонка “SELECTED FOR DEVELOPMENT”

Во второй колонке хранятся задачи, которые находятся в работе (рисунок 2.8). Разработчик не может выполнять больше одной работы за раз. То есть каждый разработчик выполняет одну работу.

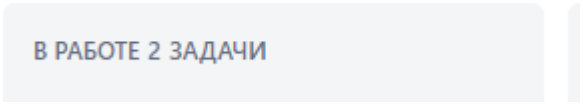


Рисунок 2.8 – Колонка “в работе”

Третья колонка предназначена для тестирования выполненных задач (рисунок 2.9). Каждый разработчик тестирует только те задачи, которые относятся к его специальности. Если тест провалился то задача возвращается в колонку “SELECTED FOR DEVELOPMENT” с пометкой возобновлено.

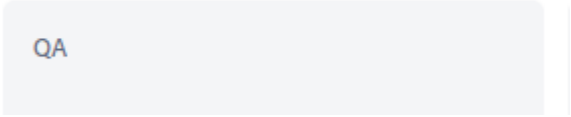


Рисунок 2.9 – Колонка “QA”

В четвертой колонки находятся полностью выполненные и протестированные задачи (рисунок 2.10). После завершения одного из этапов разработки все карточки с заданиями отправляются в архив.

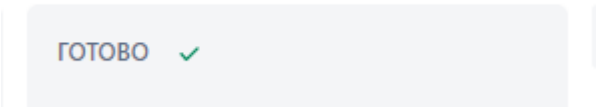


Рисунок 2.10 – Колонка “Готово”

Вся коммуникация можно разделить на две части online и offline.

К online коммуникации относятся такие программы как Skype, Discord, VK. Данные сервисы позволяют обмениваться текстовыми и голосовыми сообщениями, а также проводить аудио- и видеозвонки. К offline относятся все очные встречи в институте.

2.2 Система контроля версий

Для отслеживания версионности сервиса использовалась система контроля версий Git, а также создан удаленный репозиторий на веб-сервисе для хранения проекта GitHub (рисунок 2.11).

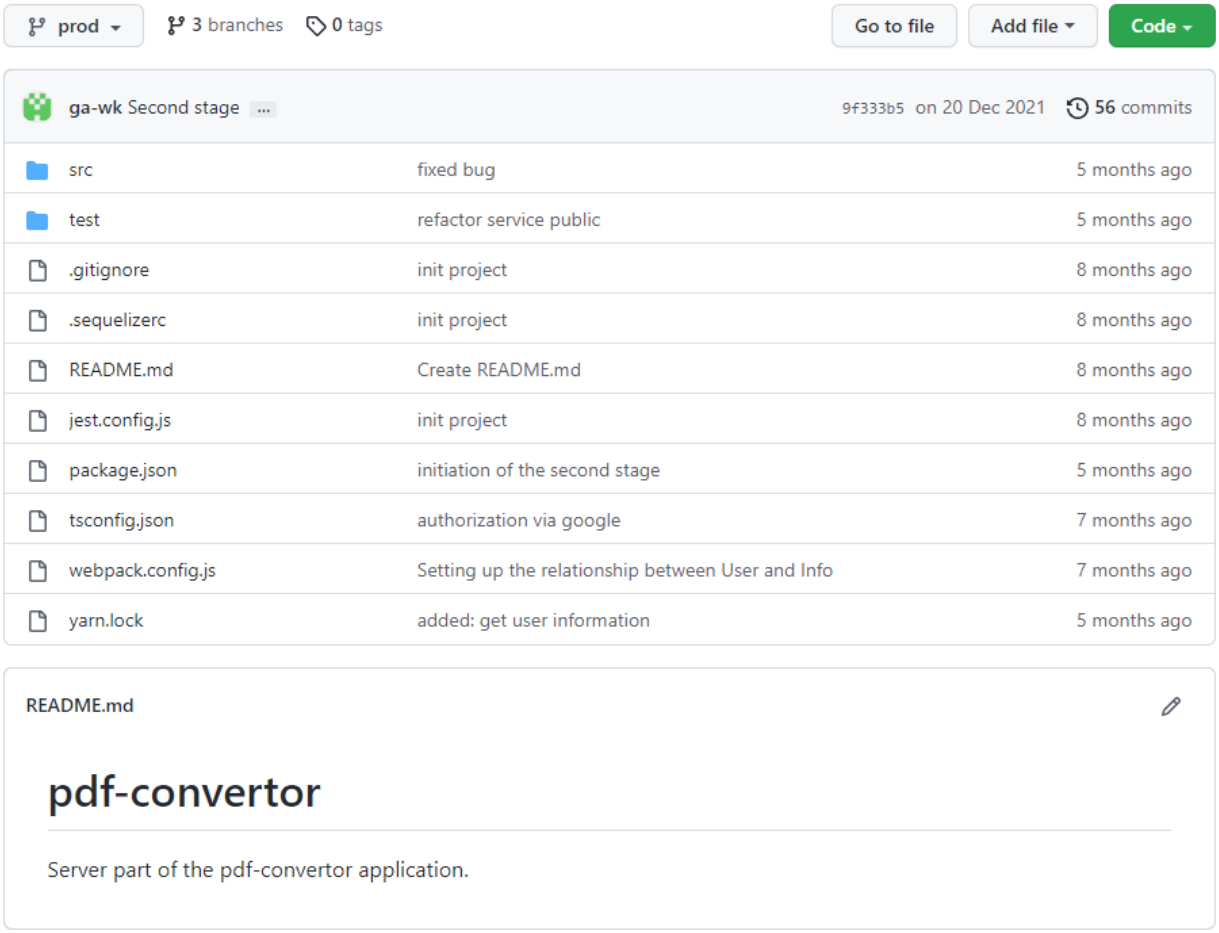


Рисунок 2.11 – Удаленный репозиторий GitHub

Структура репозитория состоит из папок и файлов:

– src – содержит весь исходный код проекта;

– test – содержит тесты;

– .gitignore, .sequelizerc, jest.config, tsconfig, webpack.config – конфигурационные файлы;

– README.md – информационный файл.

Данный репозиторий позволяет не только хранить проект, но и делить его на ветки разработки (рисунок 2.12).

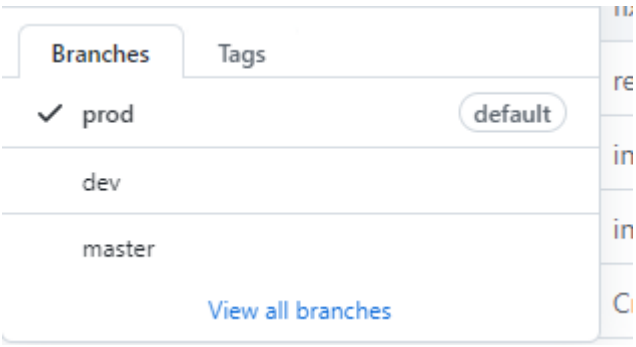


Рисунок 2.12 – Ветки проекта

Для работы использовались две основных ветки – это dev и prod. На dev ветке шла основная разработка проекта. На prod выходили только релизы сервиса. По окончании работы на ветке dev создается pull-request или запрос на слияние dev ветки с prod. После создания запроса на слияние запрашивается code review от остальных участников, которые оставляют свои замечания или одобряют решение. После одобрения от всех участников, тим-лид делает merge или сливает ветку, что означает успешное выполнение текущего этапа разработки (Рисунки 2.13 – 2.16).

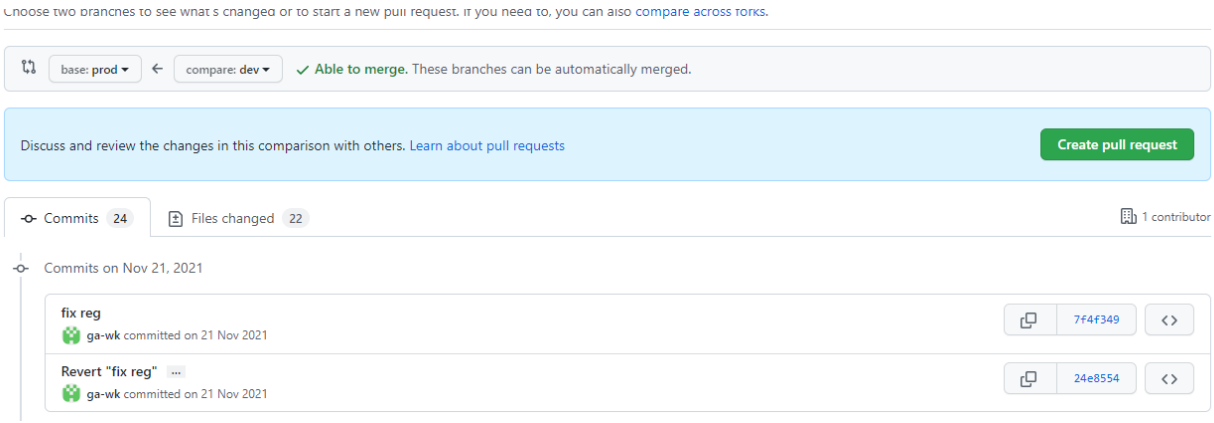


Рисунок 2.13 – Создание pull-request

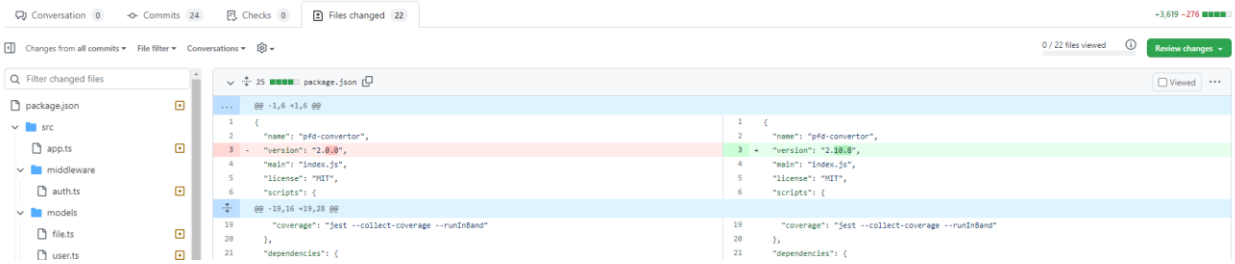


Рисунок 2.14 – Провидение code review

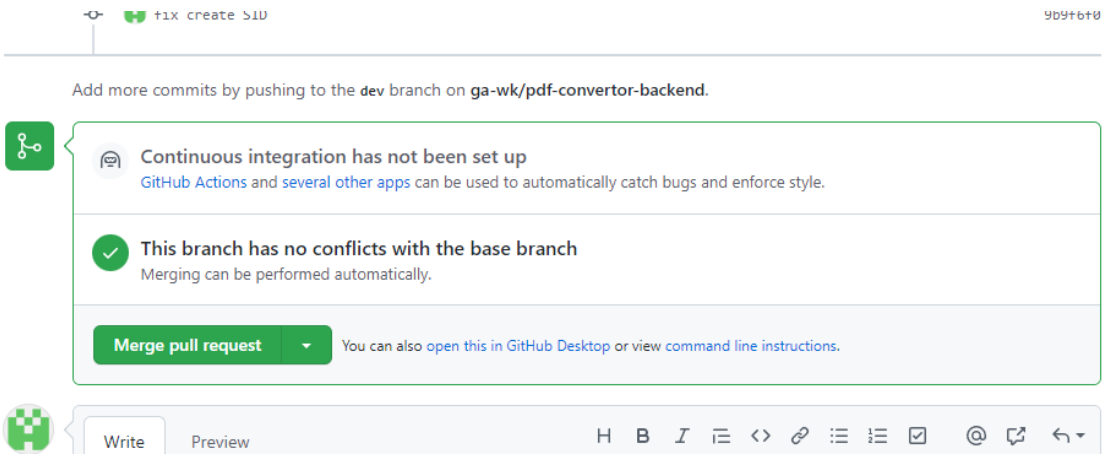


Рисунок 2.15 – Слияние веток

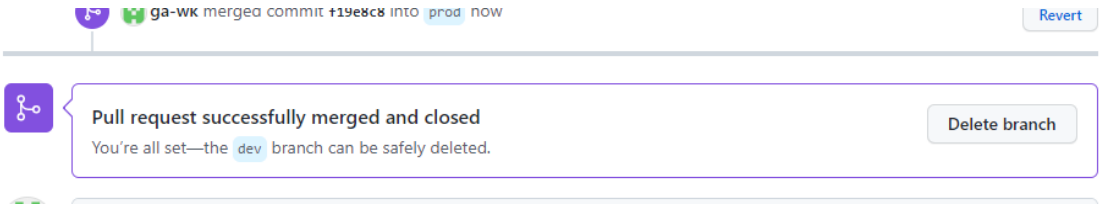


Рисунок 2.16 – Успешное слияние

2.3 Архитектура веб-сервиса

Сервис представляет из себя client-side проект, это значит, frontend обращается к backend для того что бы получить какие либо данные через предоставляемый api. Для предоставления api используется rest прослойка. REST [24] — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.

Rest также позволяет сделать запросы к серверу более предсказуемыми. К примеру, если клиенту нужен общий список файлов, он обращается по url /files/, а если клиенту нужен конкретный файл, то url выглядит уже так /files/1.

Также rest позволяет использовать один и тот же url для получения, изменения и удаления файла.

Данные с сервера чаще всего приходят в формате JSON [25]. JSON — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Но иногда сервер присылает файл и в зависимости от реализации клиента файл может начать скачивается автоматически или скачен по нажатию на кнопку.

Сервер же может так же как клиент принимать либо json, либо файл. Если серверу отправляется файл, то он должен быть отправлен с заранее правильно настроенной формы.

С помощью PostgreSQL была разработана структура данных в БД. Таблица OAuthClients хранит провайдеров, с помощью которых авторизоваться по протоколу OAuth 2.0. Таблицы Users, Info, Files являются основными. В Users хранятся служебная информация о пользователе, в Info хранятся пользовательская информация, В Files хранится информация о файлах пользователя, которые хранятся в облаке.

2.4 API веб-сервиса

По окончанию разработки данного проекта образовался список возможностей сервиса:

– получить все доступные urls для авторизации – get http://{domain}/auth/urls;

– авторизация в через google – get http://{domain}/auth/google?code=код;

– авторизация через yandex – get http://{domain}/auth/yandex?code=код;

– авторизация через vk – get http://{domain}/auth/vk?code=код;

– регистрация – post http://{domain}/auth/registration;

– вход – post http://{domain}/auth/login;

– информация о пользователе – get http://{domain}/info/user;

– конвертация pdf в word – post http://{domain}/pdfto/word;

– конвертация pdf в jpg – post http://{domain}/pdfto/jpg;

– получение из pdf всех картинок в формате jpg – post http://{domain}/pdfto/extractjpg;

– конвертация jpgs в pdf – post http://{domain}/jpgto/pdf;

– объединить pdf файлы – post http://{domain}/pdfprocessing/mergepdfs;

– разделить pdf файл – post http://{domain}/pdfprocessing/splitpdf/:range;

– сохранить pdf файл – post http://{domain}/pdfcloud/:key;

– удалить pdf файл – delete http://{domain}/pdfcloud/:key;

– скачать pdf файл – get http://{domain}/pdfcloud/:key;

– получить список pdf файлов get http://{domain}/pdfcloud/list.

2.5 Диаграммы вариантов использования

Рассмотрим все варианты использования сервиса всеми классами пользователей (рисунки 2.17 – 2.20). Для этого будет использоваться диаграмма вариантов использования.

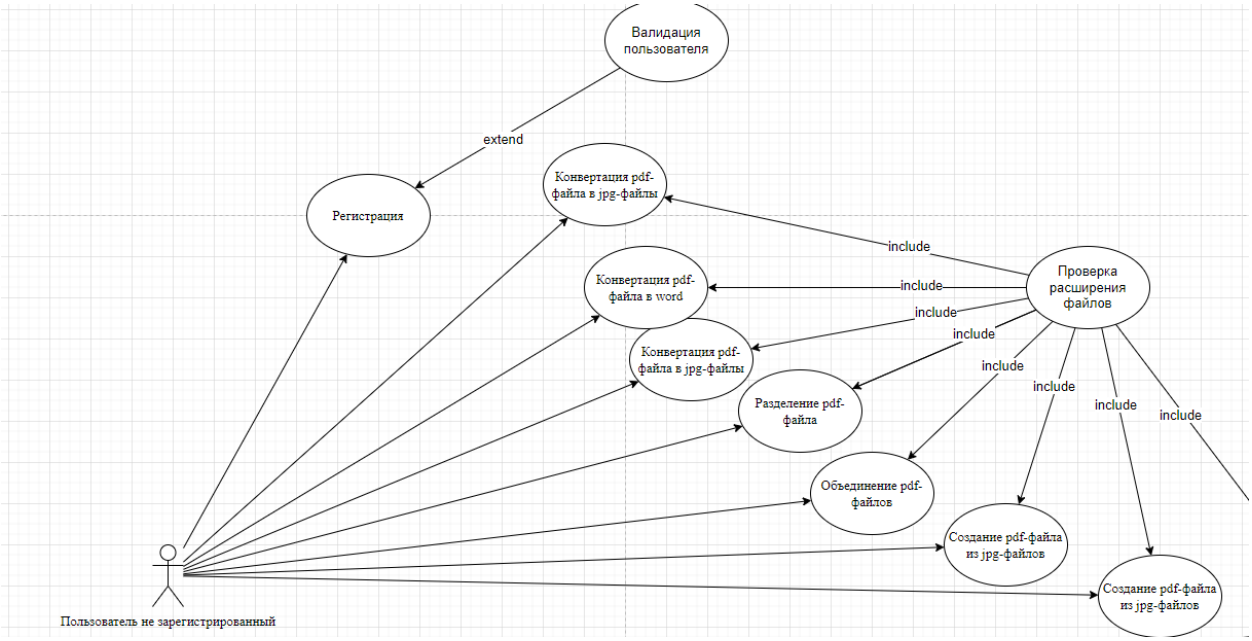


Рисунок 2.17 – Use Case Diagram для не зарегистрированных пользователей

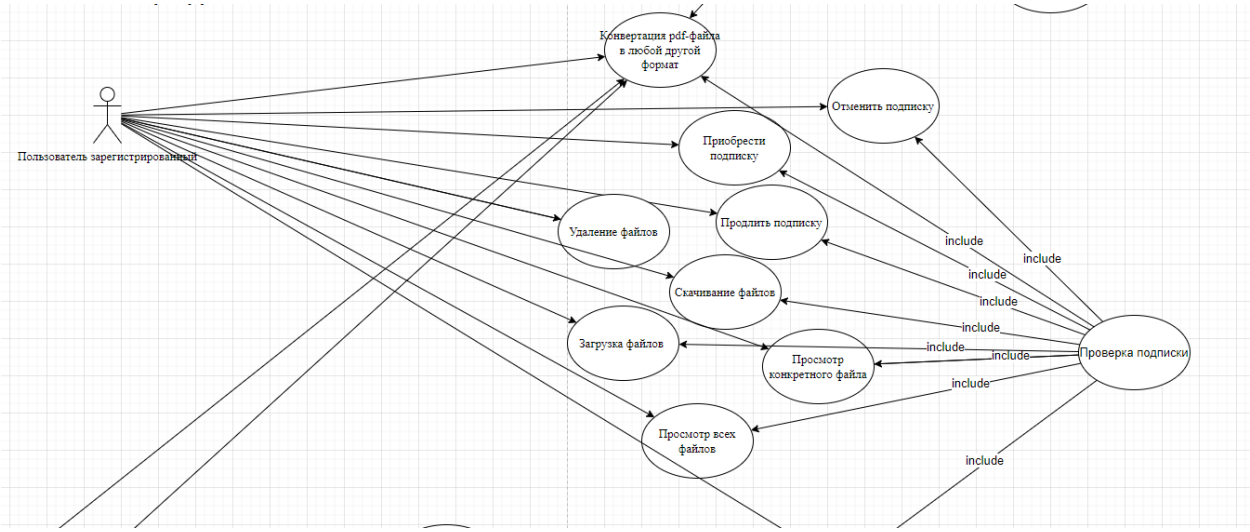


Рисунок 2.18 – Use Case Diagram для зарегистрированных пользователей

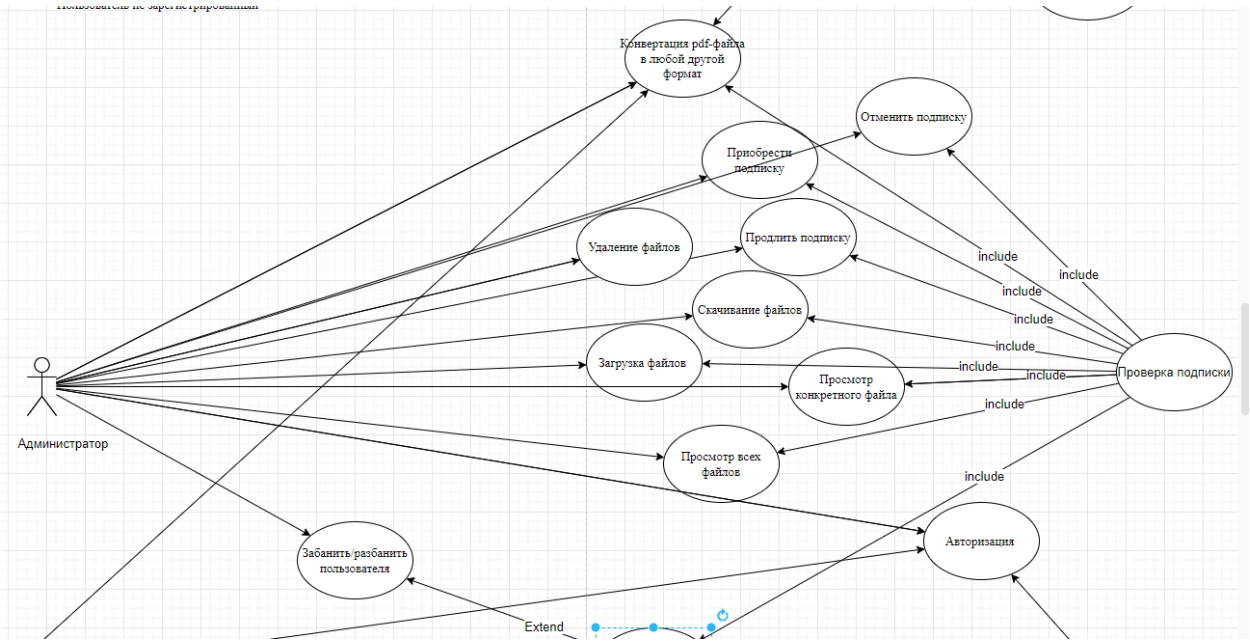


Рисунок 2.19 – Use Case Diagram для администратора

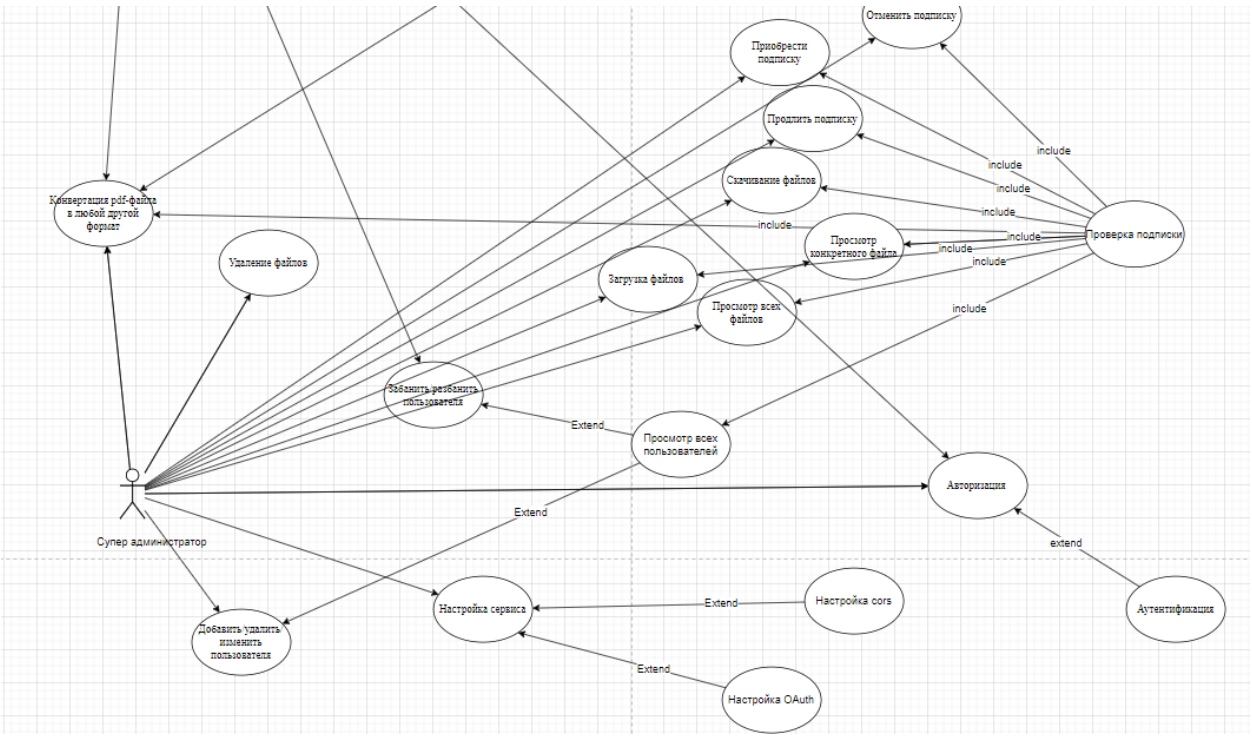


Рисунок 2.20 – Use Case Diagram для супер администратора

Можно заметить, что минимальное количество функция у не зарегистрированных пользователей. Зарегистрированные пользователи могут использовать, помимо функций конвертации, функции работы с файлами, так же функции работы с подпиской. У администратора есть возможность блокировать и разблокировать пользователя. Супер администратор обладает абсолютной всеми функциями сервиса, за исключением регистрации, как и все зарегистрированные пользователи.

2.6 Выводы по разделу

Целью раздела являлось описание проектирования разрабатываемого веб-сервиса. Были описаны архитектура, все возможности API, модели вариантов использования для не зарегистрированных и зарегистрированных пользователей, администратора и супер администратора.

Кроме того, были определены инструменты коммуникаций и управления проекта. Каждый инструмент важен для понимания задачи и верной коммуникации между другими участниками команды.

3 Разработка серверной части веб-сервиса

3.1 Структура проекта и зависимости

Создание проекта началось со структуры папок (рисунок 3.1) и настройки зависимостей.

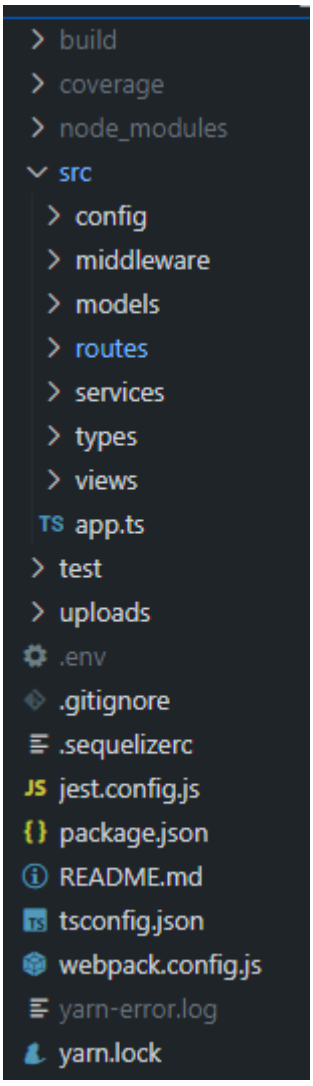


Рисунок 3.1 – Структура проекта

Устройство папок стандартное, но стоит отметить основные папки и файлы, такие как:

– папка «src» – содержит весь исходный код;

– папка «test» – содержит тесты для функций;

– папка «config» – содержит конфигурационные файлы для базы данных;

– папка «middleware» – содержит промежуточные функции для проверки безопасности;

– папка «models» – содержит объектное представление таблиц базы данных;

– папка «routes» – содержит все возможные точки входа для работы с API сервера;

– папка «services» – содержит основные функции API;

– файл package.json – содержит все зависимости и скрипты (рисунок 3.2). Зависимости можно разделить на два типа: для “production” и “development” (рисунок 3.3, 3.4).. Скрипты могут быть самыми разнообразными, в данном проекте используется для запуска проекта «start:dev:win» и сборки «build:dev:win».

```
"scripts": {
  "build:prod:unix": "NODE_ENV=production webpack",
  "start:prod:unix": "yarn build:prod:unix && node build/index.js",
  "build:dev:unix": "NODE_ENV=development webpack --config webpack.config.js",
  "start:dev:unix": "yarn build:dev:unix && node build/index.js",
  "build:dev:win": "SET NODE_ENV=development && webpack --mode=development",
  "start:dev:win": "yarn build:dev:win && SET NODE_ENV=development && node build/index.js",
  "build:prod:win": "SET NODE_ENV=production && webpack --mode=production",
  "start:prod:win": "yarn build:prod:win && SET NODE_ENV=production && node build/index.js",
  "build": "set NODE_ENV=production && webpack --mode=production",
  "start": "node build/index.js",
  "test": "jest --runInBand --detectOpenHandles",
  "test.a": "jest info --detectOpenHandles",
  "coverage": "jest --collect-coverage --runInBand"
},
```

Рисунок 3.2 – Все возможные в проекте скрипты

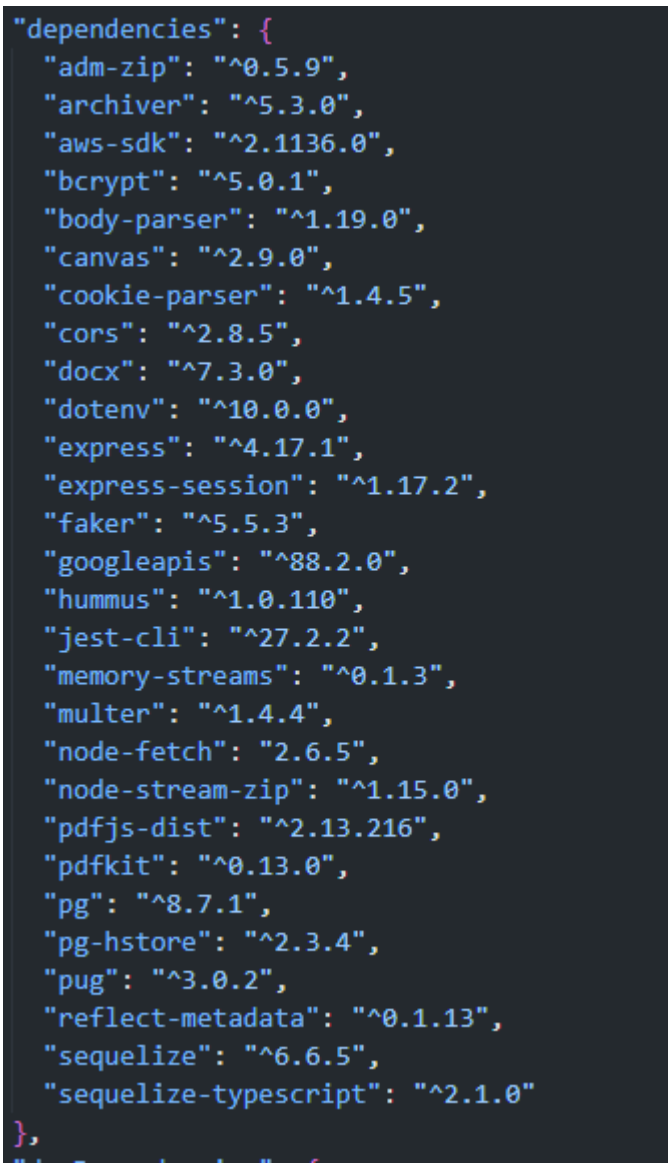


Рисунок 3.3 – Все зависимости для «production»

Рисунок 3.4 – Все зависимости для «development»

3.2 Представление данных

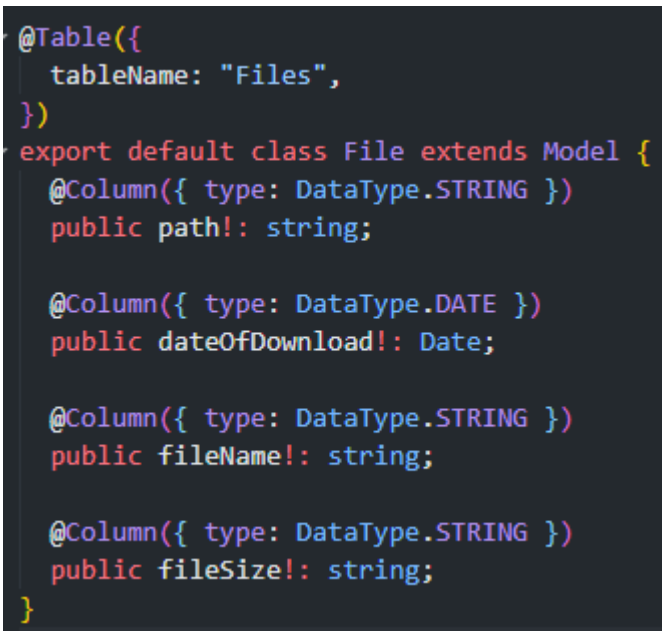
Вся информация о пользователях и их файлах хранится в реляционной базе данных PostgreSQL. Для облегчения работы с БД был использован Sequelize. Sequelize – это современная ORM TypeScript и Node.js для Postgres. ORM – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Таблицы были представлены в виде моделей (рисунок 3.5 – 3.8).

```
@Table({
  tableName: "Users",
})
export default class User extends Model {
  @Column({ type: DataType.STRING, unique: true, primaryKey: true })
  public userId!: string;

  @Column({ type: DataType.STRING })
  public accessToken!: string;

  @Column({ type: DataType.DATE })
  public createAccessToken!: Date;

  @Column({ type: DataType.DATE })
  public createSID!: Date;

  @Column({ type: DataType.STRING })
  public SID!: string;

  @Column({ type: DataType.STRING })
  public password!: string;

  @Column({ type: DataType.STRING })
  public role!: string;

  @Column({ type: DataType.BOOLEAN })
  public active!: boolean;

  @HasOne(() => Info, { foreignKey: "userId", onDelete: "CASCADE" })
  public userInfo!: Info;

  @HasMany(()=> File,  { foreignKey: "userId", onDelete: "CASCADE" })
  public userFiles?: File[];
}
```

Рисунок 3.5 – модель таблицы «User»

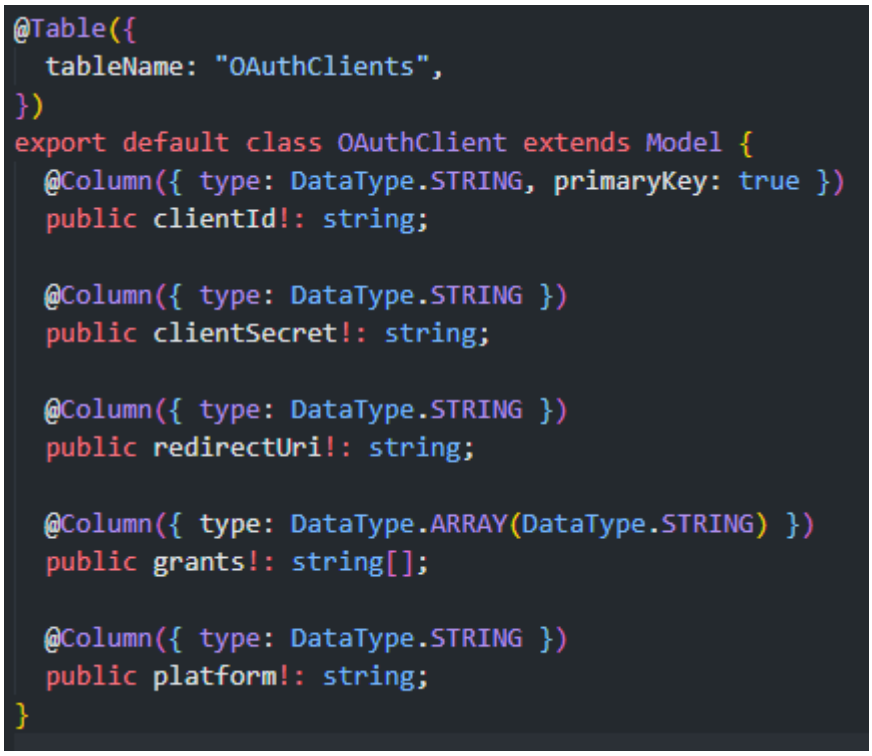
```
@Table({
  tableName: "Info",
})
export default class Info extends Model {
  @Column({ type: DataType.STRING })
  public email!: string;

  @Column({ type: DataType.STRING })
  public fullName!: string;
}
```

Рисунок 3.6 – модель таблицы «Info»



```
@Table({
  tableName: "Files",
})
export default class File extends Model {
  @Column({ type: DataType.STRING })
  public path!: string;

  @Column({ type: DataType.DATE })
  public dateOfDownload!: Date;

  @Column({ type: DataType.STRING })
  public fileName!: string;

  @Column({ type: DataType.STRING })
  public fileSize!: string;
}
```

Рисунок 3.7 – модель таблицы «File»



```
@Table({
  tableName: "OAuthClients",
})
export default class OAuthClient extends Model {
  @Column({ type: DataType.STRING, primaryKey: true })
  public clientId!: string;

  @Column({ type: DataType.STRING })
  public clientSecret!: string;

  @Column({ type: DataType.STRING })
  public redirectUri!: string;

  @Column({ type: DataType.ARRAY(DataType.STRING) })
  public grants!: string[];

  @Column({ type: DataType.STRING })
  public platform!: string;
}
```

Рисунок 3.8 – модель таблицы «OAuthClient»

Эта концепция позволяет пользоваться БД не напрямую, а через прослойку. Тем самым повышая безопасность приложения.

В случаи, когда нужно отправить данные клиенту, то используется формат JSON.

3.3 Разработка

Ещё до начала разработки проект был поделен на 3 этапа:

– авторизация, аутентификация, регистрация;

– работа с pdf-файлами;

– конвертация файлов.

3.3.1 Регистрация, авторизация, аутентификация

Для решения данной задачи были выбраны такие инструменты как:

– PostgreSQL – Данная база данных является реляционной. Она позволяет организовывать сложные структуры данных без повышения сложности работы. Информация о пользователях будет храниться в данной БД после регистрации;

– JSON Web Token (JWT) – открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Стандарт JWT позволит аутентифицировать пользователя;

– OAuth 2.0 – открытый протокол авторизации, обеспечивающий предоставление третьей стороне ограниченный доступ к защищённым ресурсам пользователя без передачи ей логина и пароля. Также является упрощенным способом регистрации пользователя.

С помощью PostgreSQL была разработана структура данных в БД (рисунок 3.9). Таблица OAuthClients хранит провайдеров, с помощью которых авторизоваться по протоколу OAuth 2.0. Таблицы Users, Info, Files являются основными. В Users хранятся служебная информация о пользователе, в Info хранятся пользовательская информация, в Files хранится информация о файлах пользователя, которые хранятся в облаке.

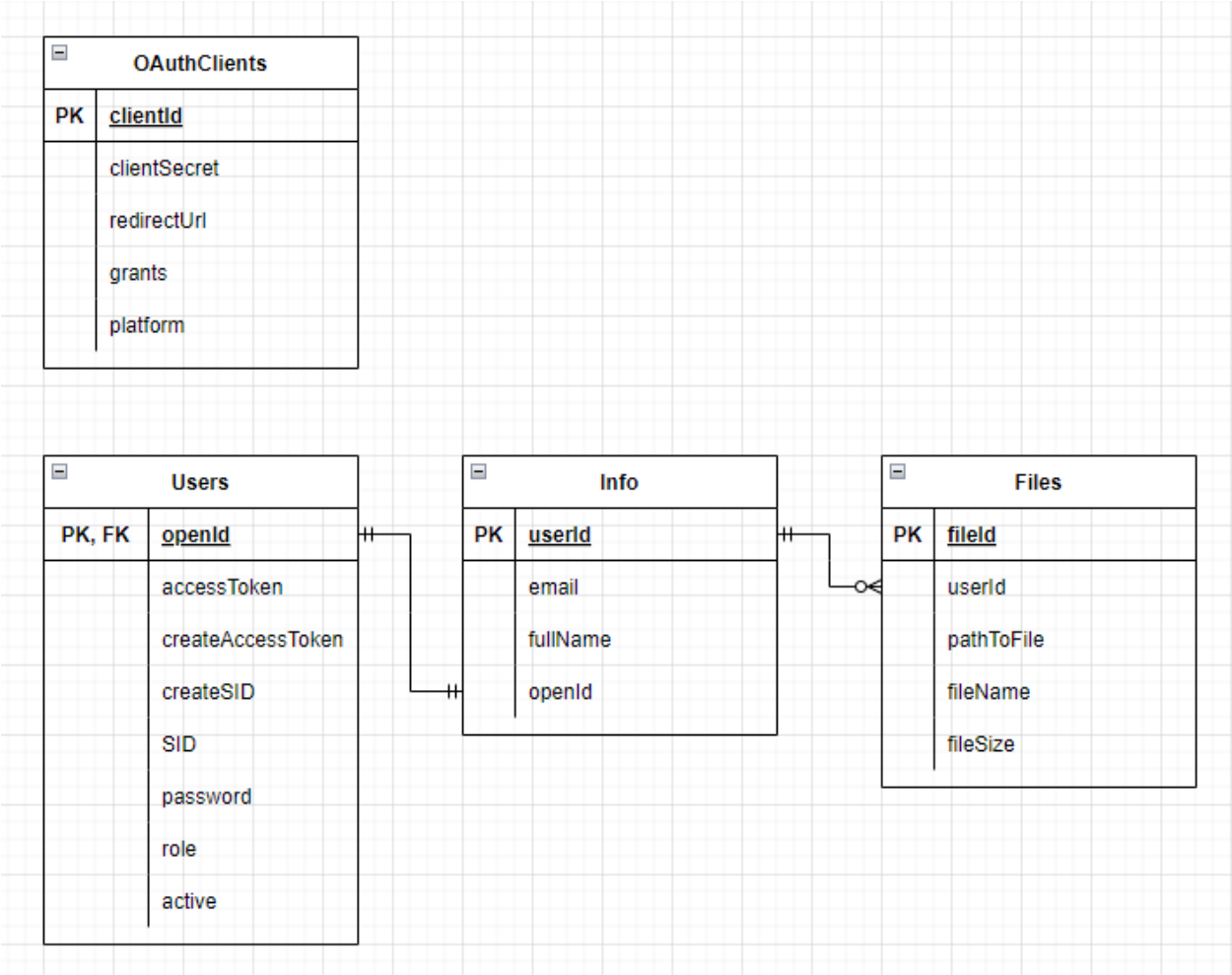


Рисунок 3.9 – Структура БД

Пользователь может войти в приложение, либо зарегистрироваться, если до этого процесс регистрации не проходил.

При нажатии на кнопку «вход» или «зарегистрироваться» появляется всплывающее модальное окно, где пользователь может выбрать, как именно ему авторизоваться в приложении: с помощью почты или использовать популярные сервисы, такие как Google, VK, Yandex.

Как сказано выше, JWT – это токен который позволяет аутентифицировать пользователя. Токен состоит из трех частей (рисунок 3.10):

– заголовок, в нем описывается, какой алгоритм используется для шифрования и тип токена;

– тело, в нем находится вся не обходимая информация о пользователе. к примеру, id пользователя;

– сигнатура, единственная часть токена которую нельзя расшифровать. за счет неё и происходит аутентификация пользователя, так как без ключа шифрования её трудно подделать.

Рисунок 3.10 – Структура JWT до шифрования

После шифрования JWT будет состоять из трех частей разделенными точками (рисунок 3.11).

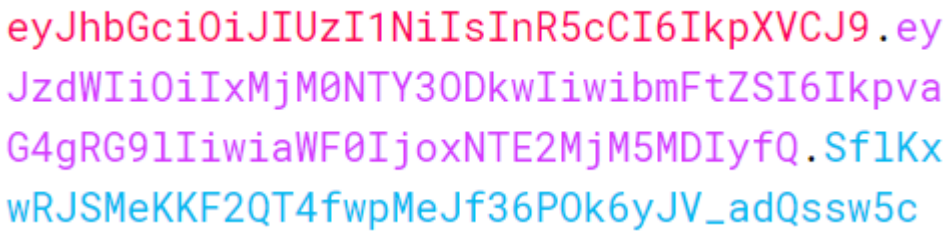


Рисунок 3.11 – Готовый JWT

Упрощённая схема работы протокола OAuth 2.0 представлена на рисунке 3.12.

Протокол работает следующим образом:

– пользователь переходит по url провайдера;

– авторизуется у провайдера;

– после авторизации провайдер генерирует пользователю code;

– после пользователь переходит обратно на страницу сервиса;

– следом передает серверу code;

– далее сервер обменивает code на access и refresh ключи у провайдера;

– с помощью access ключа сервер получает всю необходимую информацию о пользователе.

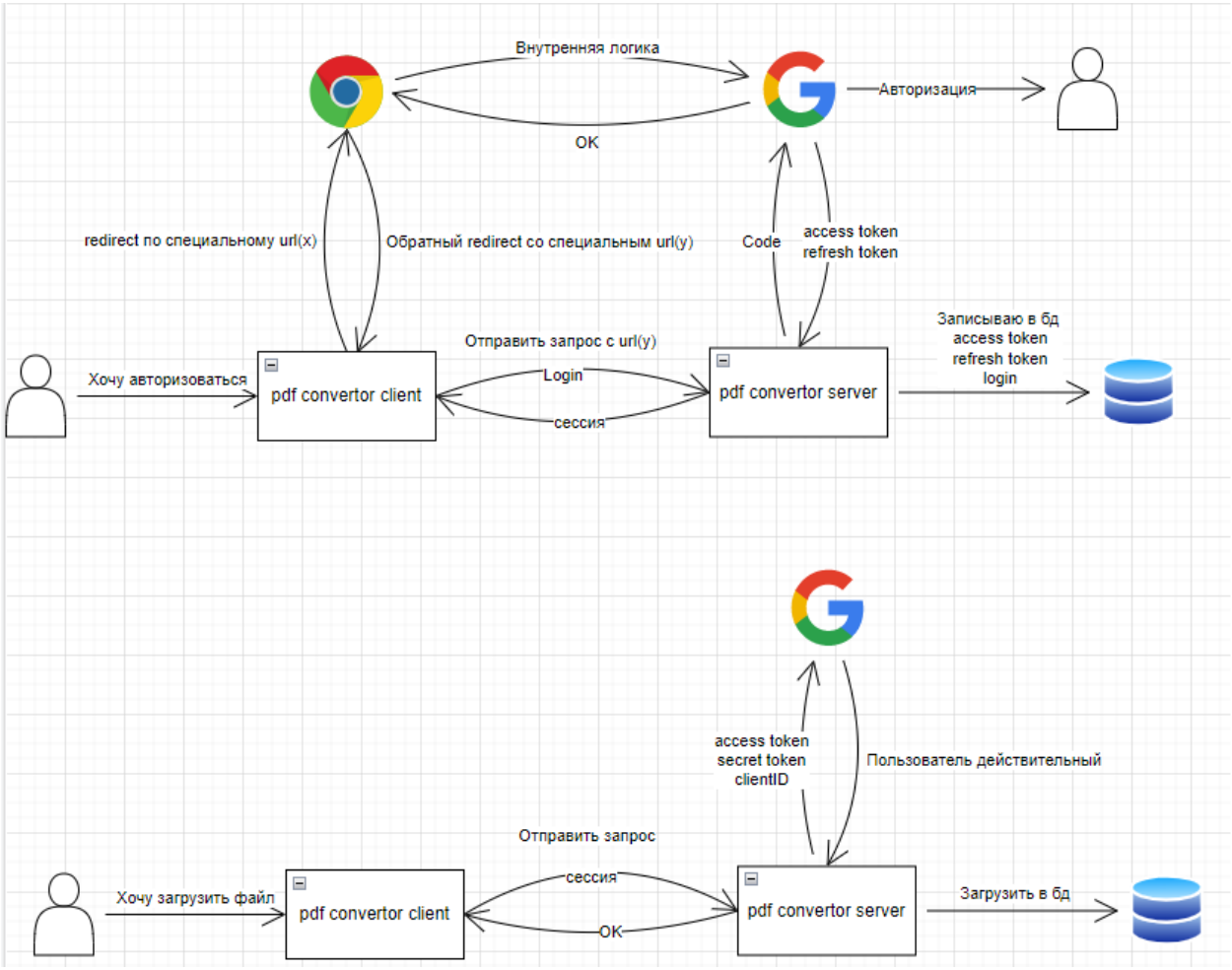


Рисунок 3.12 – работа протокола OAuth 2.0

3.3.2 Работа с pdf-файлами

Для хранения необходимо было найти провайдера, который предоставляет облачное хранилище с серверами в России. Выбор пал на YandexCloud.

Yandex предоставляет доступ к Yandex Object Storage. Yandex Object Storage – это универсальное масштабируемое решение для хранения данных. Оно подходит как для высоконагруженных сервисов, которым требуется надежный и быстрый доступ к данным, так и для проектов с невысокими требованиями к инфраструктуре хранения.

Хранение файлов строится по определенной иерархии папок и файлов. Фактически название папки, где хранятся файлы – это id пользователя. И в этой

папке хранятся файлы, относящиеся только к этому конкретному пользователю (рисунок 3.13, 3.14).

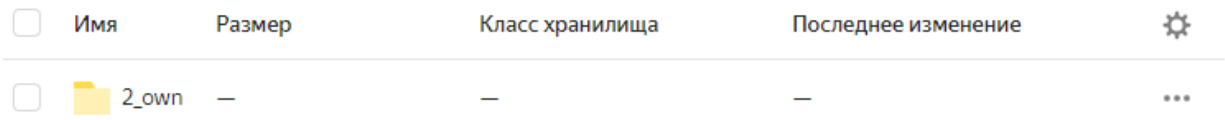


Рисунок 3.13 – папка пользователя

«Own» в id пользователя означает, что пользователь прошёл процедуру регистрации через сервис. Иначе если бы пользователь зарегистрировался через google, в id находилось бы слово «google».

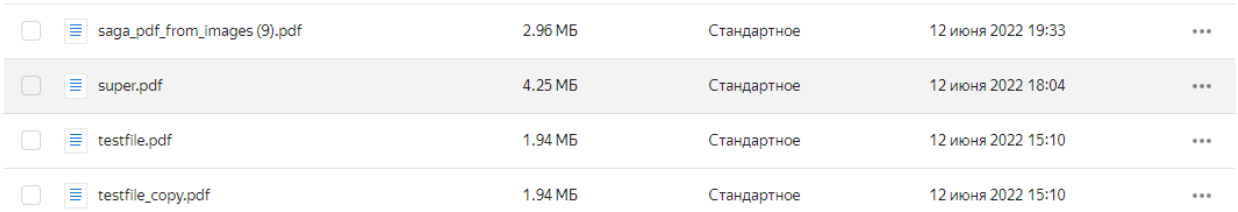


Рисунок 3.14 – файлы пользователя

Если есть необходимость добавить в хранилище файл, с название которое уже присутствует в хранилище, то наш сервис автоматически подпишет слово «_copy» к файлу.

Работа с Yandex Object Storage вилась через сервис AWS. Amazon Web Services (AWS) – это самая распространенная в мире облачная платформа с широчайшими возможностями, предоставляющая более 200 полнофункциональных сервисов для центров обработки данных по всей планете.

Yandex Object Storage имеет интеграцию через AWS. Но и без AWS существует возможность обращения к Object Storage. Для этого необходимо вручную разработать функционал, но так как была нехватка времени, был использован AWS.

Ключевые функции AWS – это upload (для загрузки файла), getObject (для скачивания файла), deleteObject (для удаления файла).

Работа с данными функциями представлена на рисунках 3.15 – 3.17

```
const result = await new Promise(function (resolve, reject) {
  aws.upload(params, function (err, data) {
    if (err) return reject(err);
    return resolve(data);
  });
});
```

Рисунок 3.15 – функция upload

```
const result: AWS.S3.GetObjectOutput = await new Promise(function (
  resolve,
  reject
) {
  aws.getObject(params, function (err, data) {
    if (err) return reject(err);
    return resolve(data);
  });
```

Рисунок 3.16 – функция getObject

```
const result = await new Promise(function (resolve, reject) {
  aws.deleteObject(params, function (err, data) {
    if (err) return reject(err);
    return resolve(data);
  });
});
```

Рисунок 3.17 – функция deleteObject

Функции, которые поддерживает наш сервис при работе с файлами:

– сохранение pdf файла;

– сохранение нескольких pdf файлов;

– удаление pdf файла;

– удаление нескольких pdf файлов;

– скачивание pdf файла;

– скачивание нескольких pdf файлов;

– получение списка pdf файлов.

3.3.3 Конвертация файлов

Для того что бы считать данных этап завершенным необходимо было реализовать следующие функции:

– конвертация PDF в JPG;

– экспорт всех картинок из pdf в формате jpg;

– конвертация jpgs в pdf;

– объедение pdf-файлов;

– разделение pdf-файлов;

– конвертация pdf в word.

В процесс разработки была придумана особенность нашего сервиса, а именно «Цепочка конвертаций».

Подробности о реализации данных функций и использованных инструментов представлены ниже.

3.3.3.1 Конвертация PDF в JPG

Для конвертации PDF в JPG использовалась библиотека PDF.JS. С помощью этой библиотеки удалось получить структуру PDF-страницы. Для отрисовки документа использовалась библиотека node-canvas. С её помощью получилось реализовать отрисовку и преобразование в JGP файл.

3.3.3.2 Экспорт всех картинок из PDF в формате JPG

Для экспорта картинок из PDF-файла использовалась все та же библиотека PDF.JS, но в этот раз необходимо было определить не структуру

файла, а информацию о картинках. После чего нужно было отрисовать их. Для этого использовалась библиотека node-canvas. Далее все экспортированные картинки архивировались с помощью библиотеки adm-zip.

3.3.3.3 Конвертация JPGs в PDF

Для интерпретации JPG-файла в PDF-файл использовалась библиотека node-canvas. Каждая картинка была отрисована как новая страница. Далее все страницы были преобразованы в PDF-файл.

3.3.3.4 Объедение PDF-файлов

Для объединения PDFs использовались такие библиотеки как hummus и memoryStreams. MemoryStreams позволил создать локальный поток. Hummus позволил считывать страницы PDF и добавлять их в поток созданным memoryStreams. Далее весь поток преобразовывался в PDF-файл.

3.3.3.5 Разделение PDF-файлов

Для разделения PDF использовались такие библиотеки как PDF.js, hummus и memoryStreams. С помощью PDF.js было найдено количество страниц, после чего было найдено количество страниц в разделенных PDFs. Следом, как и в *Объедение PDF-файлов* создавались локальные потоки и в них добавлялись PDF-страницы. Далее все разделенные PDF-файлы были архивированы с помощью библиотеки adm-zip.

3.3.3.6 Конвертация PDF в WORD

Задача по конвертации PDF в WORD крайне объёмная и трудная в реализации. На данном этапе разработки удалось достичь почти полного

соответствия PDF к WORD, для PDF версии 1.5. Это связано с тем что разные приложения по конвертации в PDF могут формировать из одного и того же исходного файла, PDF с разной структурой.

Для реализации этой задачи был создан алгоритм, который способен анализировать данные, которые были получены из PDF.js. Эти данные содержат в себе информацию и шрифтах, расположение объектов на страницы, содержимое этих страниц. Далее эти данные были интерпретированы в пригодный для дальнейшей работы вид. После чего с помощью библиотеки DOCX был создан WORD-файл.

3.3.3.7 Цепочка конвертаций

Особенность нашего сервиса является цепочка конвертаций. Пользователь может самостоятельно определить последовательность конвертаций.

Последовательность зависит от параметров в URL. URL, в нашем случаи, может вместить в себя до пяти параметров. Параметры задаются как ключ-значение (рисунок 3.18).

```
/conversion/chain?step1=mergepdf&step2=extractJPGs&step3=jpgs2pdf&step4=splitpdf,5
```

Рисунок 3.18 – пример цепочки конвертаций

3.4 Вывод по разделу

В этом разделе были описаны все этапы разработки серверной части веб-сервиса по конвертации и хранению pdf-файлов и инструментов используемых в их реализации.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы был проведен анализ предметной области и существующих аналогов разрабатываемого продукта, а также определен технологический стек. После этого были разработаны и реализованы алгоритмы конвертации PDF файлов, путем создания серверной части веб-сервиса для хранения и конвертации pdf-файлов.

В процессе разработки применялись навыки взаимодействия с фреймворком Express.js, а также с системой контроля версий Git и сервисом GitHub. Хостинг сервера в целях разработки и тестирования был выбран heroku [26].

В результате проведенной работы была разработана серверная часть веб-сервиса по конвертации и хранению PDF-фалов, включающая следующий функционал:

– регистрация;

– авторизация и аутентификация;

– загрузка файлов;

– скачивание файлов;

– удаление файлов;

– просмотр всех файлов;

– конвертация pdf-файла в jpg-файлы;

– создание pdf-файла из jpg-файлов;

– объединение pdf-файлов;

– разделение pdf-файла;

– конвертация pdf-файла в word;

– цепочка конвертаций.

Дальнейшее развитие данного сервиса предполагает добавления новых функций для работы с pdf-фалами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. PDF [Электронный ресурс] // PDF – Режим доступа: http://product.corel.com/help/CorelDRAW/540223850/Main/RU/Documentation/wwhelp/wwhimpl/common/html/wwhelp.htm#href=CorelDRAW-Portable-Document-Format-PDF.html&single=true

2. Ilovepdf [Электронный ресурс] // Ilovepdf – Режим доступа: https://www.ilovepdf.com/ru

3. pdf2go [Электронный ресурс] // pdf2go – Режим доступа: https://www.pdf2go.com/ru

4. smallpdf [Электронный ресурс] // smallpdf – Режим доступа: https://smallpdf.com/ru

5. Docx [Электронный ресурс] // Docx – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Word

6. Angular [Электронный ресурс] // Angular – Режим доступа: https://angular.io/

7. Node.js [Электронный ресурс] // Node.js – Режим доступа: https://nodejs.org/en/

8. Express.js [Электронный ресурс] // Express.js – Режим доступа: https://expressjs.com/ru/

9. Google [Электронный ресурс] // Google – Режим доступа: https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)

10. 2ip.ru [Электронный ресурс] // 2ip.ru – Режим доступа: https://2ip.ru/

11. Draw.io [Электронный ресурс] // Draw.io – Режим доступа: https://app.diagrams.net/

12. UML [Электронный ресурс] // UML – Режим доступа: https://ru.wikipedia.org/wiki/UML

13. Javascript [Электронный ресурс] // Javascript – Режим доступа: https://ru.wikipedia.org/wiki/JavaScript

14. PostgreSQL [Электронный ресурс] // PostgreSQL – Режим доступа: https://www.postgresql.org/about/news/postgresql-14-released-2318/

15. ORM [Электронный ресурс] // ORM – Режим доступа: https://ru.wikipedia.org/wiki/ORM

16. Sequelize [Электронный ресурс] // Sequelize – Режим доступа: https://sequelize.org/

17. JWT [Электронный ресурс] // JWT – Режим доступа: https://jwt.io/

18. OAuth2.0 [Электронный ресурс] // OAuth2.0 – Режим доступа: https://oauth.net/2/

19. Yandex Object Cloud [Электронный ресурс] // Yandex Object Cloud – Режим доступа: https://cloud.yandex.ru/services/storage

20. JIRA [Электронный ресурс] // JIRA – Режим доступа: https://www.atlassian.com/ru/software/jira

21. Git [Электронный ресурс] // Git – Режим доступа: https://git-scm.com/

22. GitHub [Электронный ресурс] // GitHub – Режим доступа: https://github.com/

23. VS Code [Электронный ресурс] // VS Code – Режим доступа: https://code.visualstudio.com/

24. REST [Электронный ресурс] // REST – Режим доступа: https://ru.wikipedia.org/wiki/REST

25. JSON [Электронный ресурс] // JSON – Режим доступа: https://ru.wikipedia.org/wiki/JSON

26. Heroku [Электронный ресурс] // Heroku – Режим доступа: https://dashboard.heroku.com/

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. С. Кузнецов
подпись
«16» 06 2022г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 – Программная инженерия

Разработка и реализация алгоритмов конвертации PDF файлов

Руководитель _____ 16.06.22 доцент, канд. техн. наук А.В. Хныкин
подпись, дата

Выпускник _____ 16.06.22 А.А. Газзаев
подпись, дата

Нормоконтролер _____ 16.06.22 ст. преподаватель Н.Б. Позолотина
подпись, дата

Красноярск 2022