|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: Разработка серверных частей интернет-ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть интернет-ресурса “Совместный редактор документов”

Студент: Чернов Тимур Михайлович

Группа: ИКБО-24-20

Работа представлена к защите \_\_\_\_\_\_\_\_\_\_/ /Чернов Т.М./

(дата) (подпись и ф.и.о. студента)

Руководитель: Синицын Анатолий Васильевич, старший преподаватель

Работа допущена к защите \_\_\_\_\_\_\_\_\_\_/ / Синицын А.В./

(дата) (подпись и ф.и.о. рук-ля)

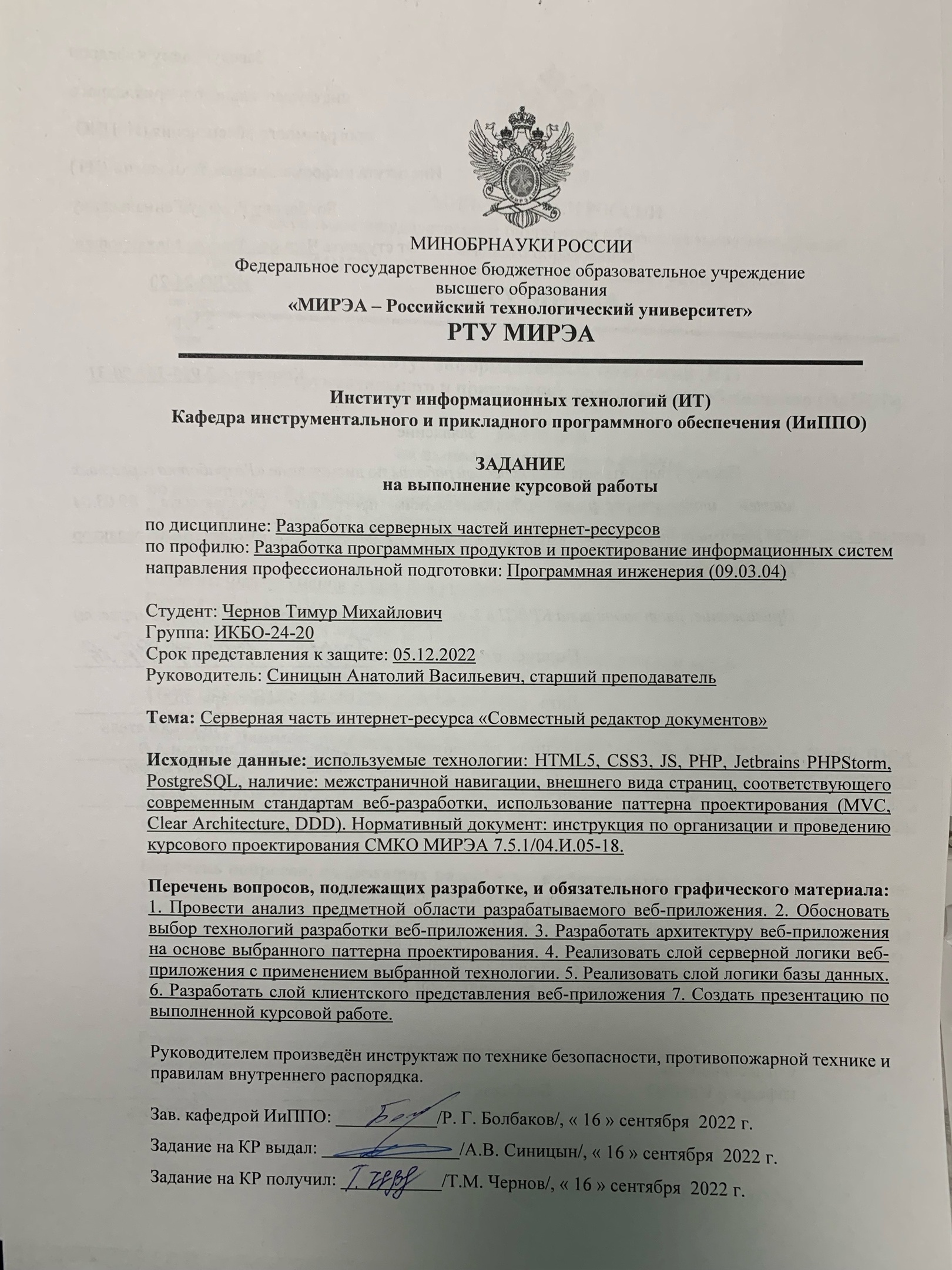
Оценка по итогам защиты:

/ , Синицын А.В., старший преподаватель/

/ , /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА. 2022 г.



**АННОТАЦИЯ**

Курсовая работа на тему: “Совместный редактор документов”, содержащая 36 страниц отчёта, 16 рисунков и 15 использованных источников.

Разработанное приложение же содержит из 36 PHP-файлов, 6 HTML-страниц, 3 CSS-файлов и 30 JavaScript-файлов.

В ведении обосновывается актуальность выбранной темы, к тому же определяются цель работы и задачи, которые необходимо решить для достижения цели, описываются объект и предмет исследования, используемые методы и информационная база исследования, помимо этого, кратко характеризуется структура курсовой работы по разделам.

В основной части всего 4 раздела: «Сбор и анализ требований к программному продукту», «Выбор и обоснование технологий», «Разработка архитектуры приложения на основе выбранного паттерна» и «Разработка серверной части интернет-ресурса». В них содержится материал, необходимый для того, чтобы достигнуть цели.

В заключенииописываются теоретические выводы и достигнутые результаты в ходе разработки приложения на тему “Совместный редактор документов”.

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc122294654)

[ВВЕДЕНИЕ 6](#_Toc122294655)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc122294656)

[1 СБОР И АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ 7](#_Toc122294657)

[1.1 Описание и анализ предметной области 7](#_Toc122294658)

[1.2 Формирование основных требований к приложению 10](#_Toc122294659)

[1.3 Выводы к разделу 1 10](#_Toc122294660)

[2 ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ 11](#_Toc122294661)

[2.1 Среда разработки 11](#_Toc122294662)

[2.2 Пакетный менеджер 12](#_Toc122294663)

[2.3 База данных 12](#_Toc122294664)

[2.5 Средство тестирования 13](#_Toc122294665)

[2.6 Выводы к разделу 2 13](#_Toc122294666)

[3 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА 14](#_Toc122294667)

[3.1 Логическая структура 14](#_Toc122294668)

[3.2 Выводы к разделу 3 18](#_Toc122294669)

[4 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА 19](#_Toc122294670)

[4.1 Создание интернет-ресурса 19](#_Toc122294671)

[4.2 Создание программно-аппаратной части веб-приложения 22](#_Toc122294672)

[4.3 Выводы к разделу 4 33](#_Toc122294673)

[ЗАКЛЮЧЕНИЕ 34](#_Toc122294674)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc122294675)

[ПРИЛОЖЕНИЕ А (справочное) Исходный код веб-приложения 38](#_Toc122294676)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

|  |  |
| --- | --- |
| PHP | – Hypertext Preprocessor |
| MVC | – Model-View-Controller |
| HTTP  СУБД | – Hyper Text Transfer Protocol  – Система управления базами данных |
| UML  БД  ФИО  PDO | – Unified Modeling Language  – База данных  – Фамилия, имя, отчетство  – PHP Data Objects |

# ВВЕДЕНИЕ

Сейчас мы живем в век информационных технологий, нашу жизнь нельзя представить без телефонов, компьютеров, планшетов. Часто в рабочих целях люди использую документы для обмена или хранения информации, для подготовления отчётов и т.д.

Совместный редактор документов - это инструмент, который позволяет пользователям создавать, редактировать и совместно использовать документы в режиме реального времени через интернет. Совместный редакторы документов стали очень популярными в последнее время, так как они позволяют пользователям работать с документами из любой точки мира, где есть доступ к интернету. Это особенно удобно для командной работы, когда несколько людей должны работать над одним документом одновременно. Кроме того, редакторы документов обычно имеют множество функций, таких как форматирование текста, вставка изображений и таблиц, проверка орфографии и многое другое.

Поскольку редакторы документов пользуются большой популярностью, моя цель – разработать приложение на тему “Совместный редактор документов”, который будет достаточно прост и удобен в использовании.

Для того, чтобы достигнуть этой цели, необходимо:

1. провести анализ предметной области и сформировать основные требования к приложению,
2. обосновать выбор средств ведения разработки,
3. разработать архитектуру веб-приложения на основе выбранного паттерна проектирования,
4. разработать веб-приложение с использованием PHP, выбранной технологии и инструментария,
5. реализовать слой серверной логики, базы данных и клиентской логики веб-приложения.

# ОСНОВНАЯ ЧАСТЬ

# СБОР И АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ

## Описание и анализ предметной области

Темой данного веб-приложения является «Совместный редактор документов». Наименование разработанного приложения – «Совместные документы».

Предметной областью для данной курсовой работы является исследование веб-сервисов, предоставляющих пользователям возможность создавать, редактировать и совместно использовать документы в режиме реального времени через интернет.

Удобный совместный редактор должен соответствовать следующим требованиям:

1. Возможность создания новых документов и открытия существующих;
2. Возможность редактирования документов несколькими пользователями одновременно;
3. Возможность просмотра истории изменений документа;
4. Возможность комментирования документа;
5. Возможность управления правами доступа к документу;
6. Возможность экспорта документа в различные форматы;
7. Интуитивно понятный и простой интерфейс;
8. Возможность быстрого доступа к основным функциям;
9. Защита данных пользователей;
10. Защита от несанкционированного доступа к документам;
11. Быстрый доступ к документам;
12. Быстрое редактирование документов;
13. Высокая скорость загрузки и сохранения документов.

Существует достаточно много аналогов совместных редакторов. Поэтому для разработки данного веб-приложения были рассмотрены и выявлены общие принципы построения уже существующих приложений  с такой же направленностью, а также было проведено исследование контента этих страниц.

Были рассмотрены такие сайты, как Google Docs [1] (Рисунок 1.1.1), Яндекс. Документы [2] (Рисунок 1.1.2), Draft [3] (Рисунок 1.1.3).

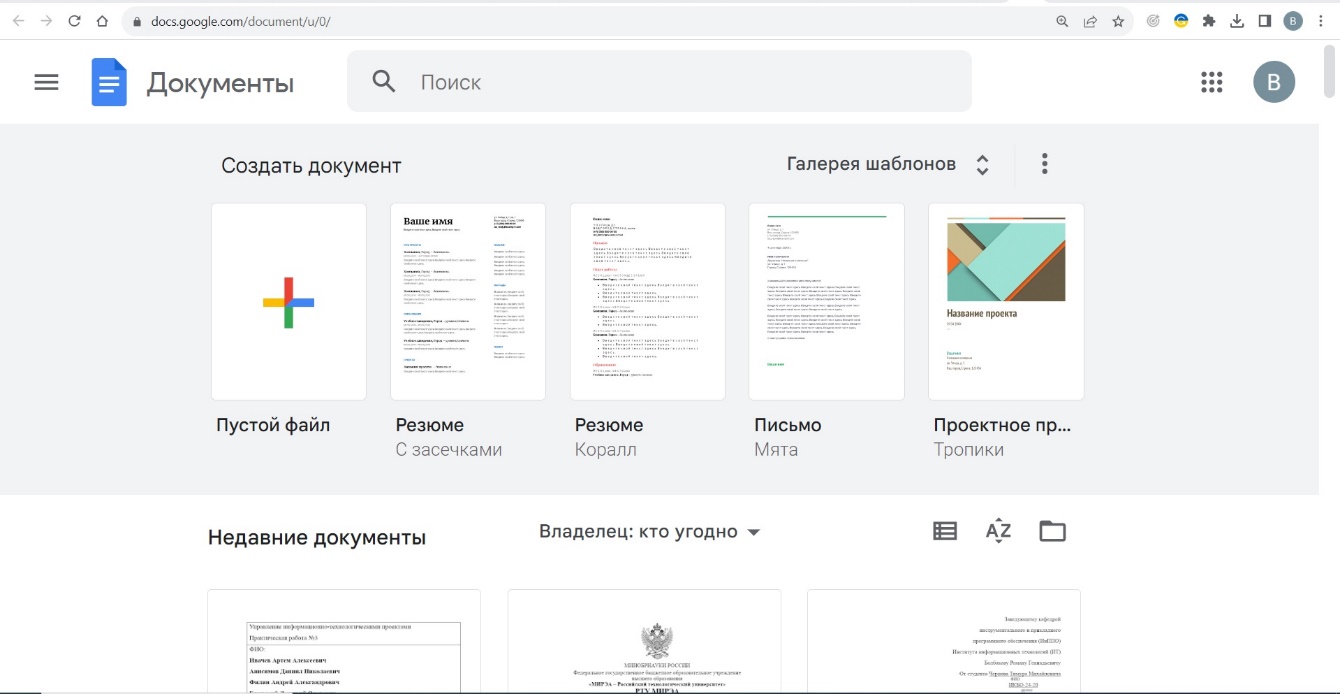


Рисунок 1.1.1 – Веб-сайт Google Docs

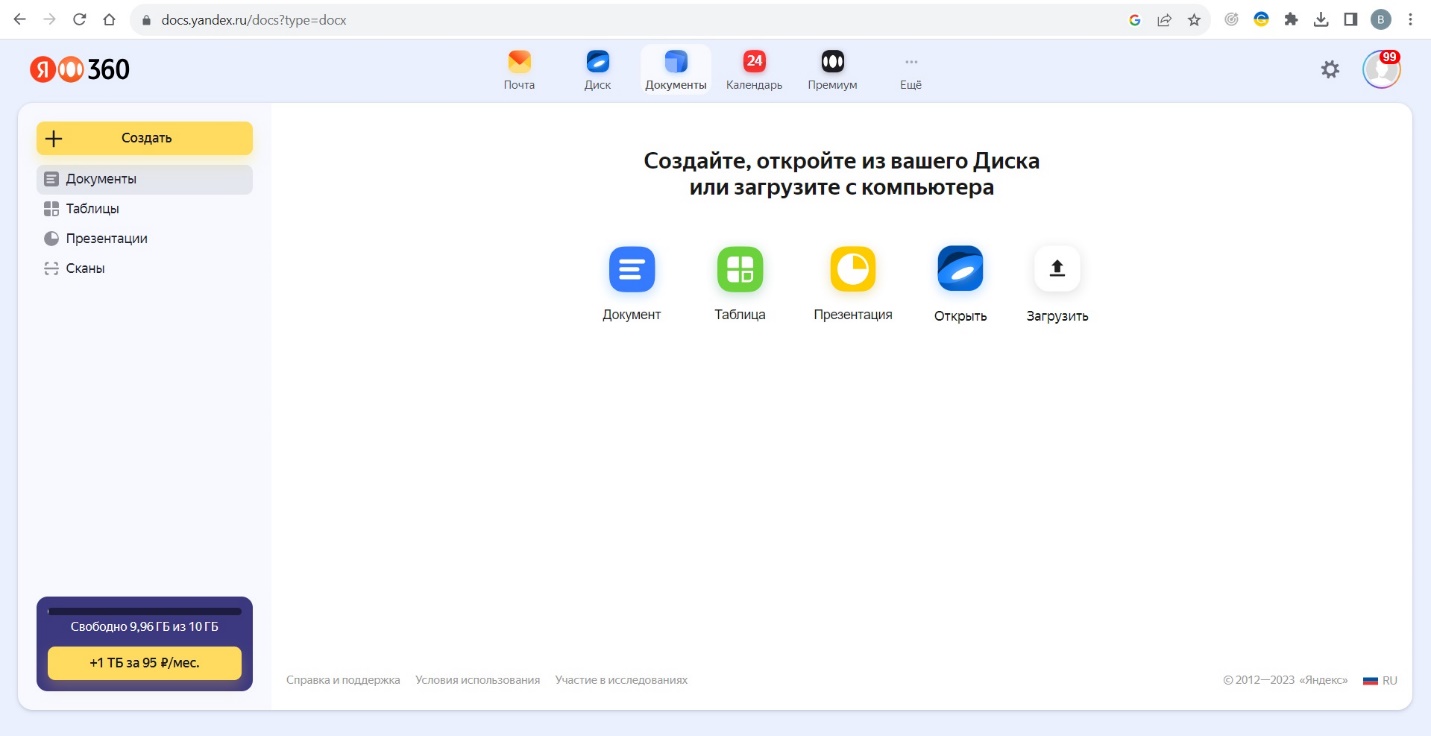


Рисунок 1.1.2 – Веб-сайт Яндекс. Документы

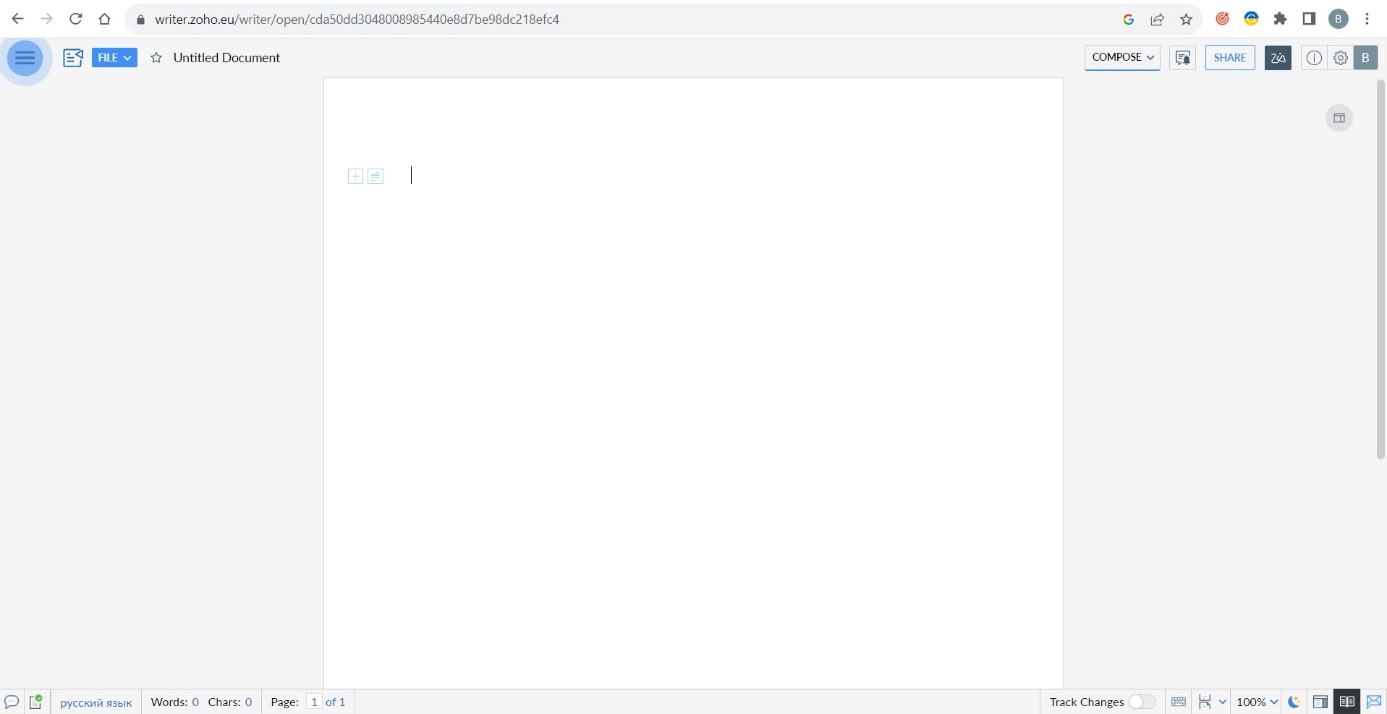


Рисунок 1.1.3 – Веб-сайт Zoho Writer

Основные функции данных сайтов включают создание и редактирование документов: все сервисы позволяют пользователям создавать новые документы и редактировать существующие. Пользователи могут добавлять текст, изображения, таблицы, графики и другие элементы в свои документы. Сайты обеспечивают возможность одновременного редактирования документов несколькими пользователями. Это позволяет пользователям работать над документом в режиме реального времени, видеть изменения других участников и вносить собственные правки. Сервиса предлагают хранилище для сохранения документов в облаке. Пользователи могут сохранять свои документы в своем аккаунте и получать к ним доступ с любого устройства с подключением к интернету. Они также могут легко делиться документами с другими пользователями, предоставлять права доступа и управлять ими.

Несмотря на множество преимуществ совместных редакторов документов, есть и некоторые потенциальные недостатки, так для работы в совместных редакторах документов требуется стабильное интернет-соединение. Если соединение прерывается или медленное, это может привести к задержкам при редактировании и совместной работе. Некоторые передовые функции и возможности, доступные в офисных приложениях с полной установкой на компьютер, могут быть ограничены или отсутствовать в совместных редакторах. Некоторые пользователи могут испытывать нехватку функций, которые они привыкли использовать в других программных средствах. Также в совместных редакторах документов данные хранятся в облаке, что может вызывать опасения относительно конфиденциальности и безопасности. Важно обеспечить надлежащую защиту доступа к документам и убедиться в надежности системы хранения данных.

## Формирование основных требований к приложению

Согласно заданию на курсовую работу, необходимо использовать определенный набор технологий. Во-первых, для каждой из этих технологий нужно применить следующие документы: документация по языку PHP и, конечно, инструменты и технологии HTML5, CSS3 и MySQL.

Во-вторых, необходимо создать MVC приложение для объединения свойств классов и среды разработки Jetbrains PHPStorm.

Проведя анализ предметной области и технической литературы, можно сделать вывод, что приложение должно ориентироваться на взаимодействие с клиентами.

К тому же, приложение должно предоставлять пользователю возможность иметь собственный аккаунт для безопасности данных, а также возможность создания и хранения нескольких документов, которые будут записываться в базу данных. Для этого необходимо использовать MySQL.

Переходя теперь к тому, что должно быть внутри приложения и проанализировав эти веб-приложения, были учтены некоторые моменты, которые необходимы к выполняемому проекту, то есть:

1. Сайт должен быть стабильным, то есть взаимодействие с любыми процессами не должно приводить к возникновению ошибок;
2. Необходимо наличия простого и удобного интерфейса;
3. Возможность зарегистрироваться на сайте и создавать новые документы;
4. Возможность работы с большим количеством пользователей;
5. Поддержка множества форматов документов;

## Выводы к разделу 1

На основе предметной области, которая была рассмотрена и проанализирована, были сформированы основные требования для приложения и его функционал.

# ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ

## Среда разработки

В качестве основной среды разработки была выбрана Jetbrains PHPStorm.

Что же такое Jetbrains PHPStorm? PHPStorm [4] – это среда разработки, которая по-настоящему понимает ваш код. Если точнее говорить, то IDE обеспечивает поддержку PHP, также моментально предотвращает ошибки, предоставляет умное автодополнение, подсветку синтаксиса и безопасные рефакторинги, помимо этого есть возможность редактирования кода на HTML, CSS и JavaScript.

На самом деле эта среда разработки достаточна удобна и выигрывает у большинства редакторов по функционалу.

К примеру, отладка PHP-приложений не нуждается в конфигурации. То есть можно использовать визуальный отладчик, чтобы проверить переменные, устанавливать точки останова и отлаживать удаленные приложения.

К слову говоря, поскольку PHPStorm построена на базе платформы IntelliJ, которую совершенствуют уже более 15 лет, то, учитывая весь спектр возможностей, она подойдет для всех разработчиков.

К тому же, PhpStorm предоставляет инструменты для работы с базами данных и SQL-кодом в ваших проектах. IDE помогает подключаться к базам данных, редактировать таблицы, выполнять запросы, а также анализировать схемы с помощью UML-диаграмм.

Помимо этого, позволяет связать свой проект с git-репозиторием и осуществлять коммиты, пуши и другое непосредственно из среды разработки, предоставляет различные подсказки, что помогает работать быстрее и эффективнее.

## Пакетный менеджер

Сам же проект использует Composer [5] – пакетный менеджер, позволяющий немного упростить написание кода, поскольку он ищет и устанавливает, и распаковывает библиотеки вместе с зависимостями.

Собственно говоря, это удобный инструмент, благодаря которому не будешь испытывать проблемы при поиске и разрешении зависимостей у библиотек. К тому же, Composer упрощает не только установку библиотек, но и использование их. По сути, он сам подключает все необходимые файлы классов библиотеки. За это отвечает специальный сценарий autoload.php.

## База данных

Для этого веб-приложения была создана база данных, работа с которой была произведена через СУБД MySQL [6]. Это свободная и открытая система управления базами данных, основанная на модели клиент-сервер. Она предоставляет эффективные средства для хранения, управления и извлечения данных.

MySQL обладает простым и понятным синтаксисом SQL, что облегчает разработку, создание запросов и управление базами данных. Он также имеет множество инструментов и документации, которые облегчают работу с ним.

## Сервер

Apache HTTP Server (обычно известный как Apache) – это свободный и открытый веб-сервер, который является одним из самых популярных и широко используемых веб-серверов в мире.

Apache распространяется по лицензии Apache, что означает, что он доступен бесплатно для использования, модификации и распространения. Открытый исходный код позволяет разработчикам экспериментировать, адаптировать и улучшать его под уникальные потребности.

Apache славится своей надежностью и стабильностью. Он широко применяется в критически важных окружениях. Apache имеет хорошо разработанную архитектуру, которая позволяет обрабатывать большое количество запросов и обеспечивает стабильную работу сервера.

Apache предлагает множество модулей и расширений, которые позволяют настраивать и расширять функциональность сервера. Это позволяет администраторам настроить Apache под свои потребности и добавить дополнительные возможности, такие как сжатие данных, кэширование, аутентификация и шифрование.

Apache может быть установлен и работать на различных операционных системах, таких как Windows, Linux, macOS и другие. Это позволяет выбирать наиболее подходящую платформу для разворачивания сервера, с учетом требований проекта.

Apache имеет огромное сообщество разработчиков и пользователей, что означает наличие обширной документации, форумов, ресурсов и инструментов для поддержки и помощи. Это облегчает разработку, настройку и устранение проблем сервера.

Apache остается одним из самых популярных выборов для веб-серверов благодаря своей надежности, гибкости и открытому исходному коду, а также поддержке сообщества и расширенным возможностям настройки.

* 1. **Паттерн**

В качестве выбранного паттерна был выбран MVC-паттерн [7], с помощью которого будет строиться фундамент и реализовываться программа. Почему был выбран MVC? Собственно, поскольку шаблон MVC описывает просто способ построения структуры приложения, главной целью которого как раз является отделение бизнес-логики от пользовательского интерфейса.

Именно поэтому приложение можно проще масштабировать, тестировать и реализовывать.

## Средство тестирования

В качестве средства тестирования у нас выступает локальный HTTP сервер localhost. Localhost [8] – это сервер, с помощью которого можно тестировать свои приложения. К тому же, тестирование на localhost удобно тем, что операционная система становится симулированным сервером после запуска обратной связи. То есть, можно загружать все необходимые файлы программы на сервер и проверять её работоспособность.

То есть если в браузере Firefox или Yandex Browser вбить в командную строку localhost и нужный порт можно будет перейти на веб-приложение.

## Выводы к разделу 2

Подводя итоги по второму разделу, был осуществлен краткий обзор на технологии и средства, которые используются для разработки серверной части интернет-ресурса.

# РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА

## Логическая структура

Для того, чтобы приложение удовлетворяло техническим требованиям, оно, во-первых, должно обладать необходимым, присущим данному типу приложений и предметной области, функционалом.

Именно поэтому для разработки архитектуры веб-приложения следует выявить бизнес-правила, на основе которых будет строиться информационная система пользователя и администратора.

В связи с тем, что было рассмотрено, на рисунке 3.1.1 представлена диаграмма вариантов использования UML для проектируемой информационной системы пользователя:

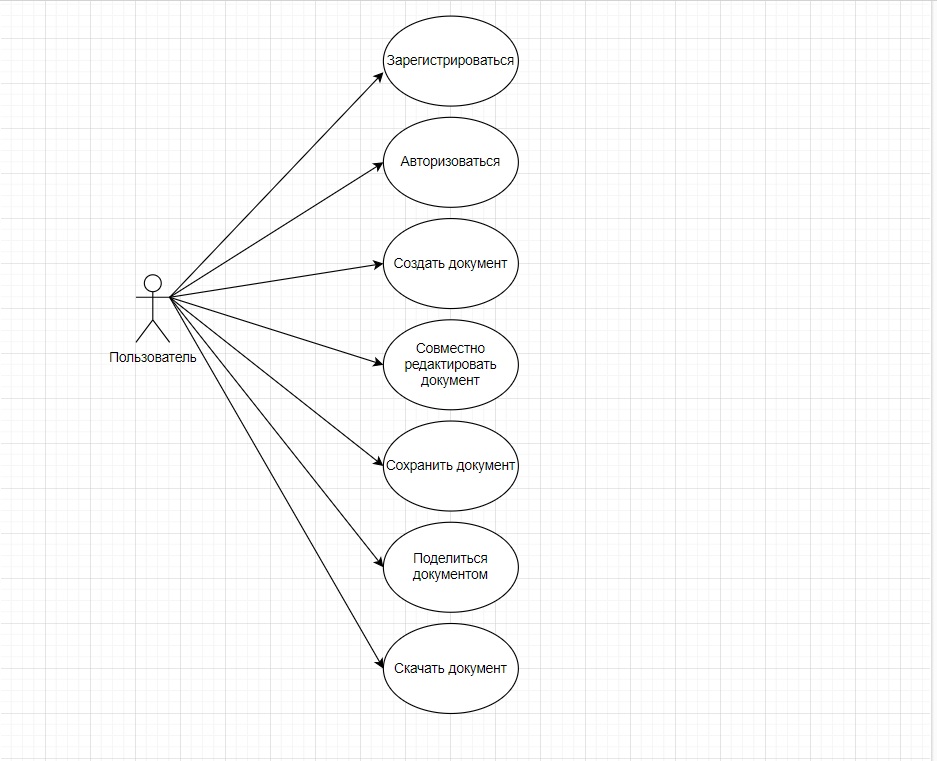


Рисунок 3.1.1 – Диаграмма вариантов использования информационной системы пользователя

Если говорить об информационной системе администратора, то необходимо, чтобы у него был удобный и простой интерфейс, позволяющий вносить все необходимые правки. В связи с тем, что было рассмотрено, на рисунке 3.1.2 представлена диаграмма вариантов использования UML для проектируемой информационной системы администратора:

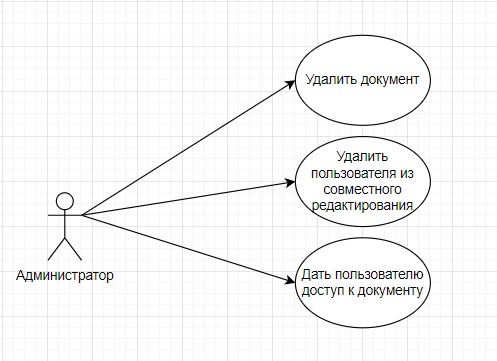


Рисунок 3.1.2 – Диаграмма вариантов использования информационной системы администратора

Для разработки веб-приложения в предыдущей главе был выбран паттерн MVC [9]. Этот паттерн предполагает отделение бизнес-логики от пользовательского интерфейса. Сам шаблон состоит из модели, контроллера и вида. То есть модель предоставляет данные и правила бизнес-логики, а также взаимодействует с базой данных, вид или представление отвечает за пользовательский интерфейс, а контроллер как раз обеспечивает взаимодействие между моделью и представлением. На рисунке 3.1.3 представлен шаблон MVC-архитектуры:

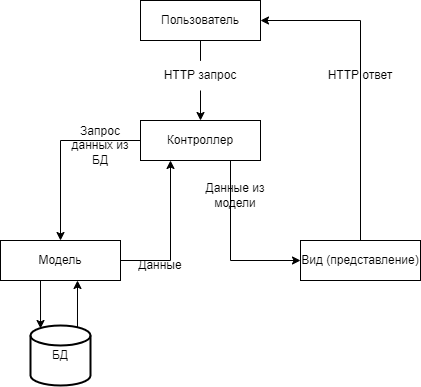


Рисунок 3.1.3 – Шаблон MVC-архитектуры

Далее, рассмотрим сущности базы данных. Для того, чтобы разработать ее архитектуру, необходимо сначала понять, что это. Сущности БД [10] – это объект БД, в котором хранятся определенные данные в соответствие с учебным пособием о “Базах данных”.

Если говорить о главном слое Инфраструктуры приложения, то можно заметить, что участвует в процессе Клиент, Веб-сервер и обработчик на веб-сервере, обрабатывающий поступающие запросы. То есть пользователь через интерфейс обращается у нас к Контролеру, который уже затем взаимодействует с Моделью и базой данных и выдает результат в виде новой страницы или новых данных. Для наглядности взаимодействия с системой из Сущностей и Контролеров были убраны данные. Диаграмма данного слоя изображена ниже на рисунке 3.1.4:

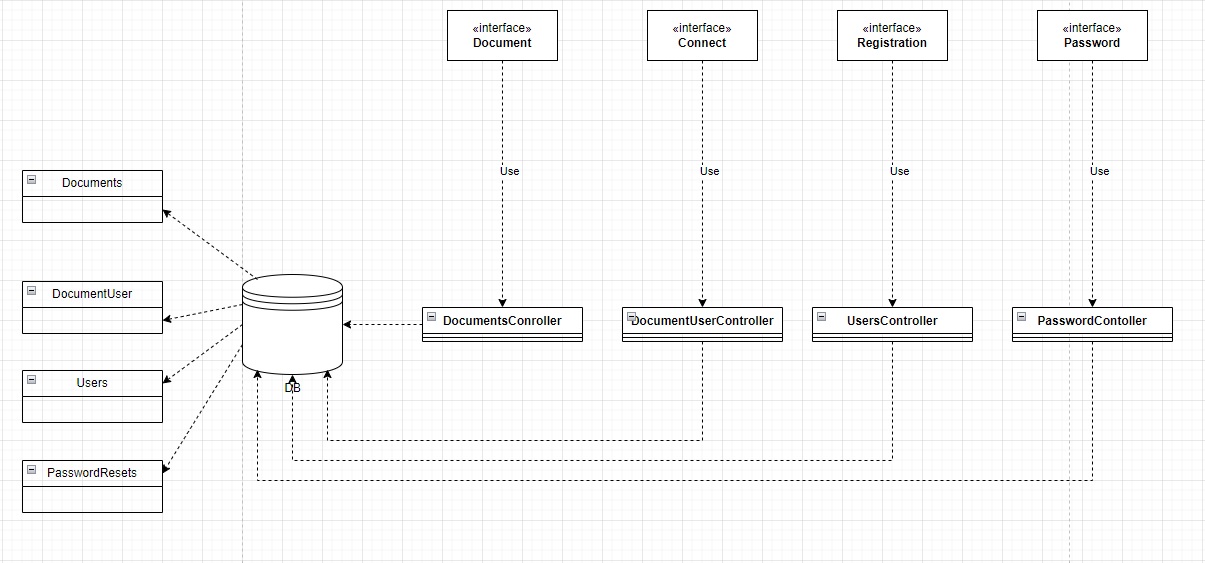


Рисунок 3.1.4 – Диаграмма слоя Инфраструктуры

Благодаря разработке и рассмотрению архитектуры MVC-приложения, время для написания кода и разработке серверной части значительно уменьшится и облегчится.

## Выводы к разделу 3

На основе требований, описанных в разделе 2, были сформированы более детальные задачи и функции, благодаря чему была разработана архитектура приложения «Совместный редактор документов» на основе MVC-паттерна.

## РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА

## Создание интернет-ресурса

Во-первых, наш веб-ресурс разработан с применением PHP, HTML, CSS3, JavaScript и MySQL.

Чтобы убедиться в работе приложения по серверу можно просмотреть рисунки 4.1.1 – 4.1.7.

Сначала рассмотрим страницу авторизации и регистрации (Рисунки 4.1.1 – 4.1.2).

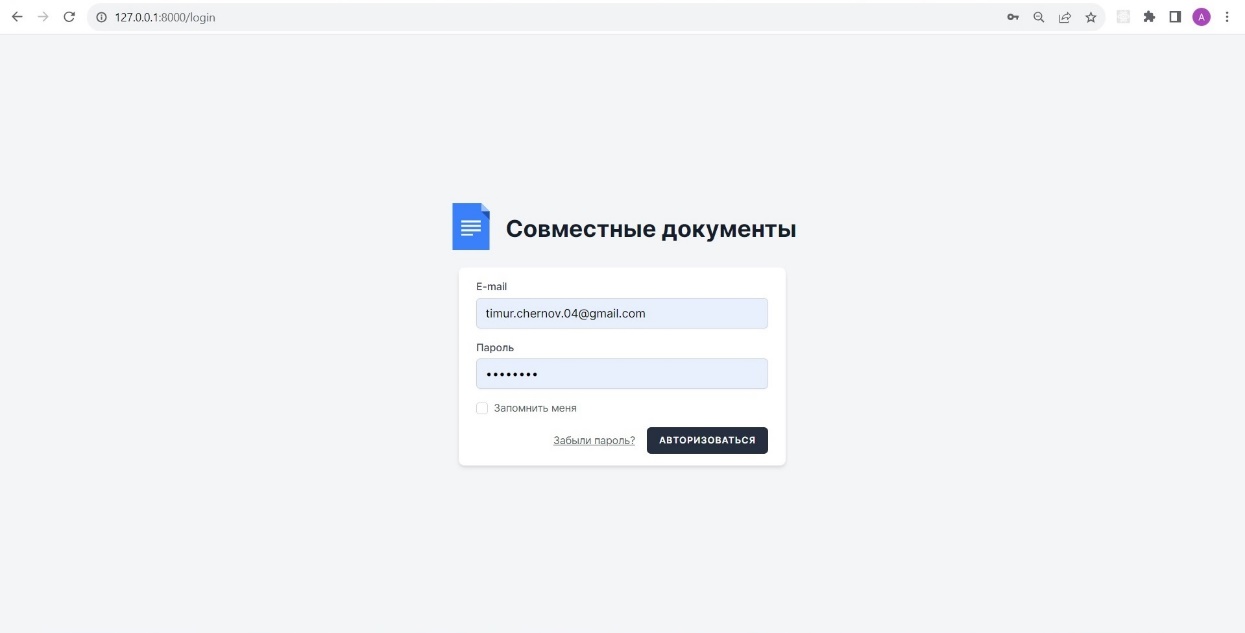


Рисунок 4.1.1 – Страница авторизации на сайт

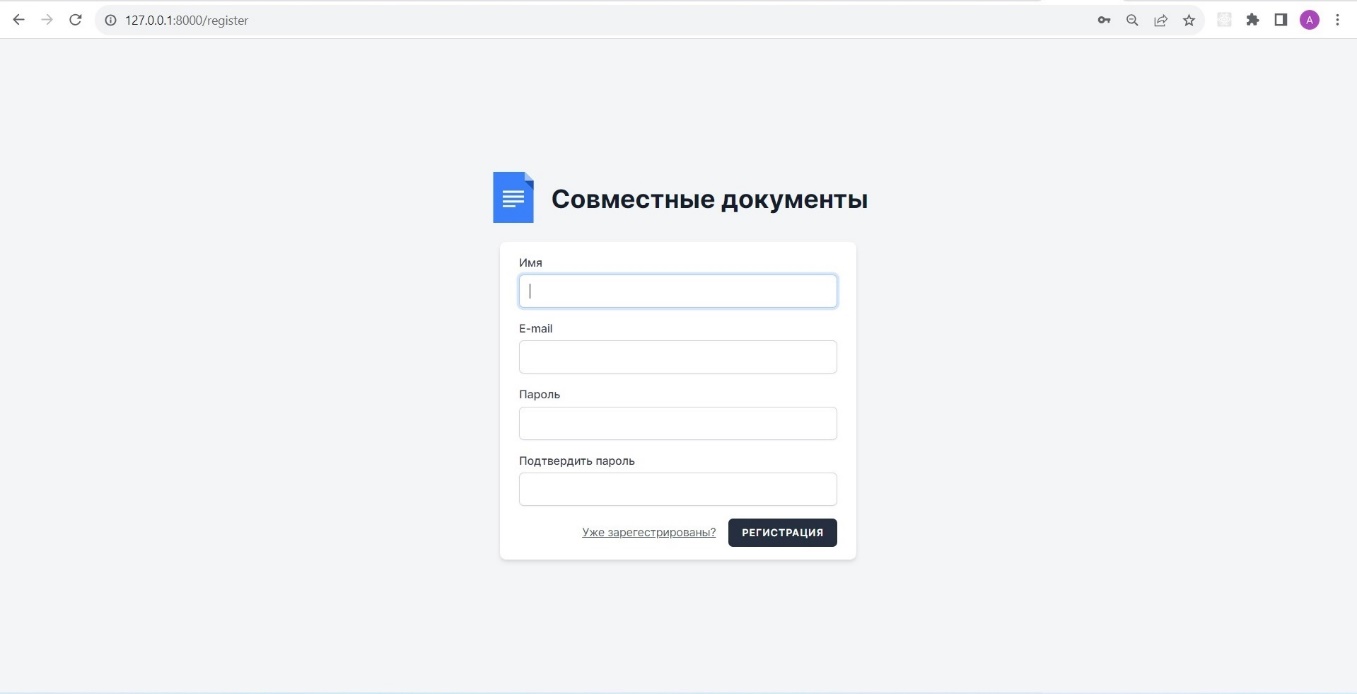


Рисунок 4.1.2 – Страница регистрации пользователя

При регистрации пользователю необходимо указать своё имя, почту и придумать пароль.

Теперь же рассмотрим со стороны пользователя, как будет выглядеть страница, когда он войдет в аккаунт. Ниже на рисунке 4.1.3 можно заметить, что пользователь может создать новый документ или войти в уже существующий, а также воспользоваться поиском для того, чтобы найти нужный документ.

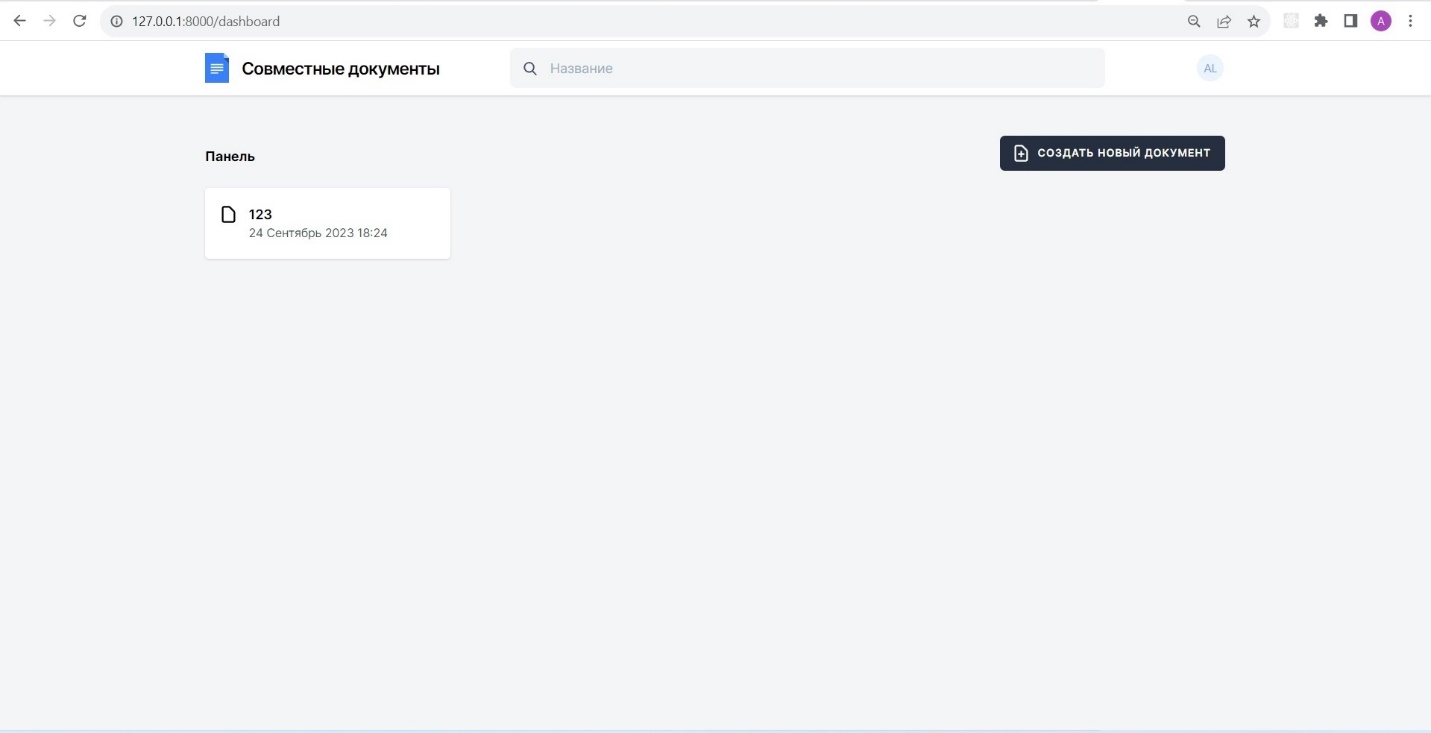


Рисунок 4.1.3 – Главная страница при входе пользователя

Помимо этого пользователь может зайти в личный кабинет (Рисунки 4.1.4-4.1.5).

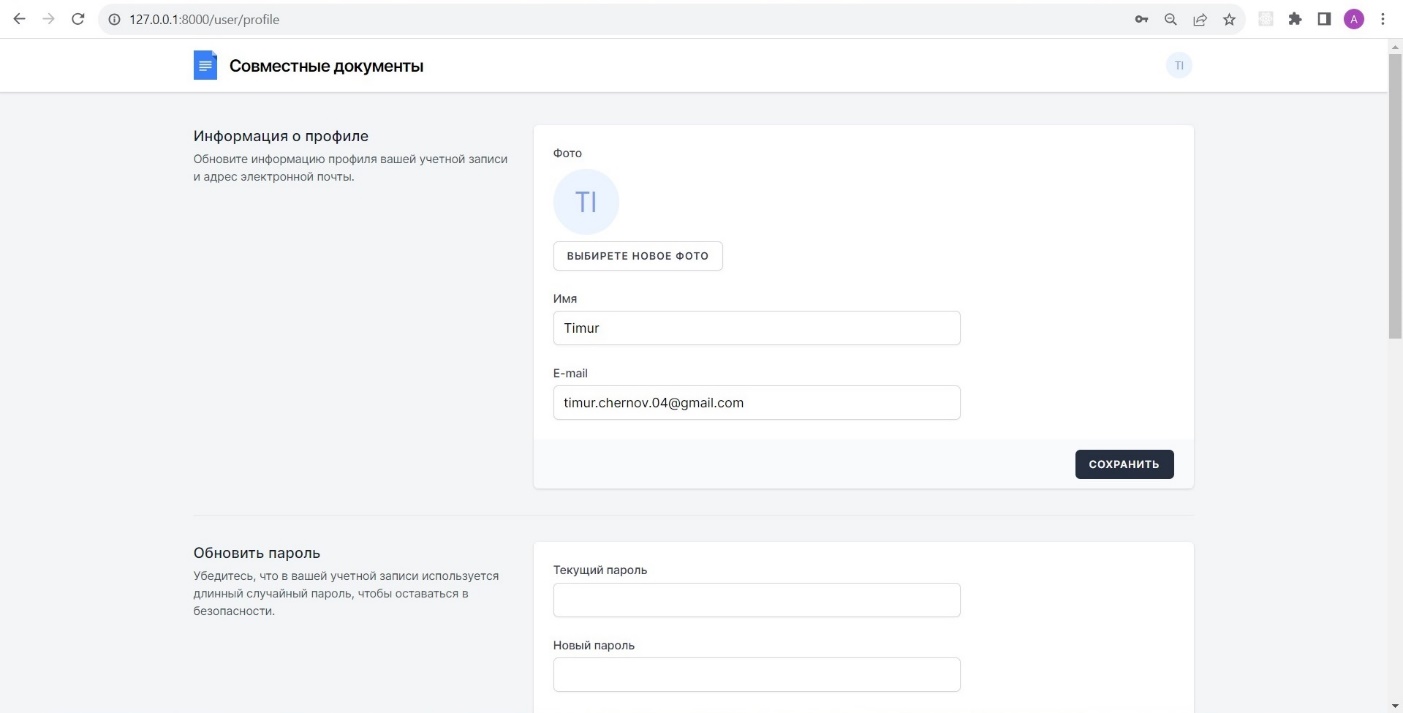


Рисунок 4.1.4 – Личный кабинет пользователя

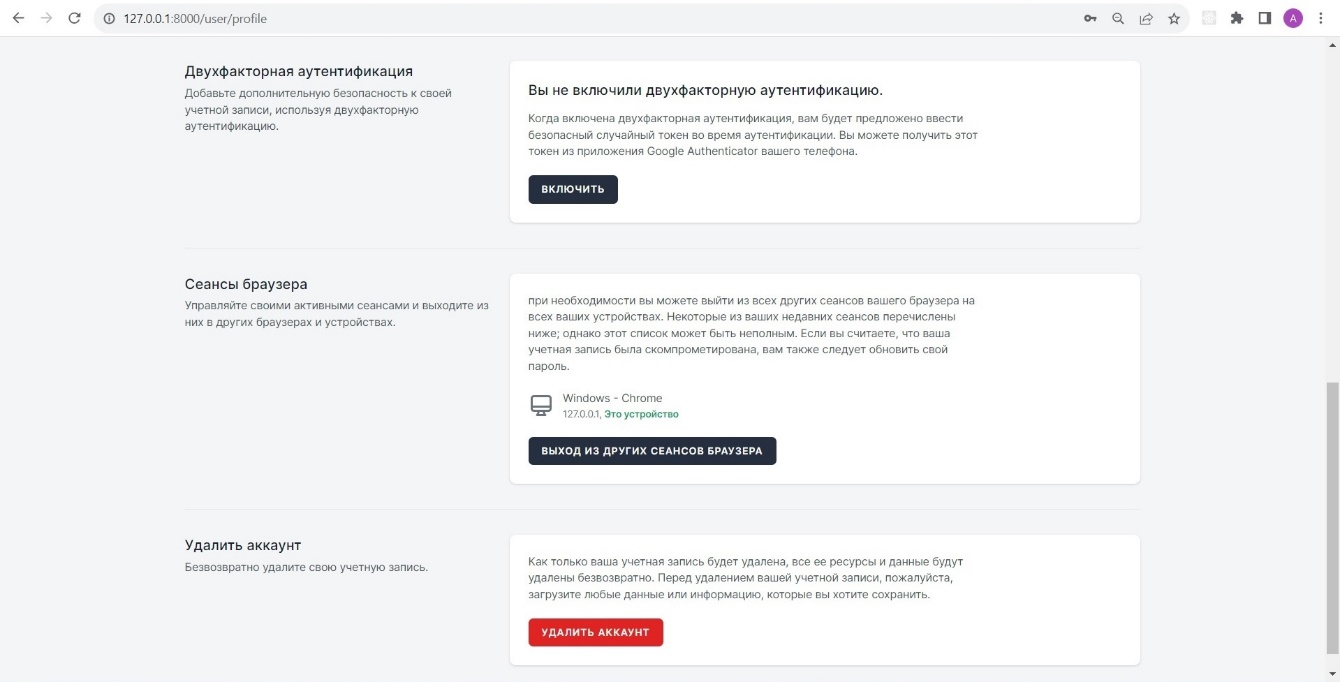


Рисунок 4.1.5 – Личный кабинет пользователя

Также пользователь может редактировать документ (Рисунок 4.1.6)

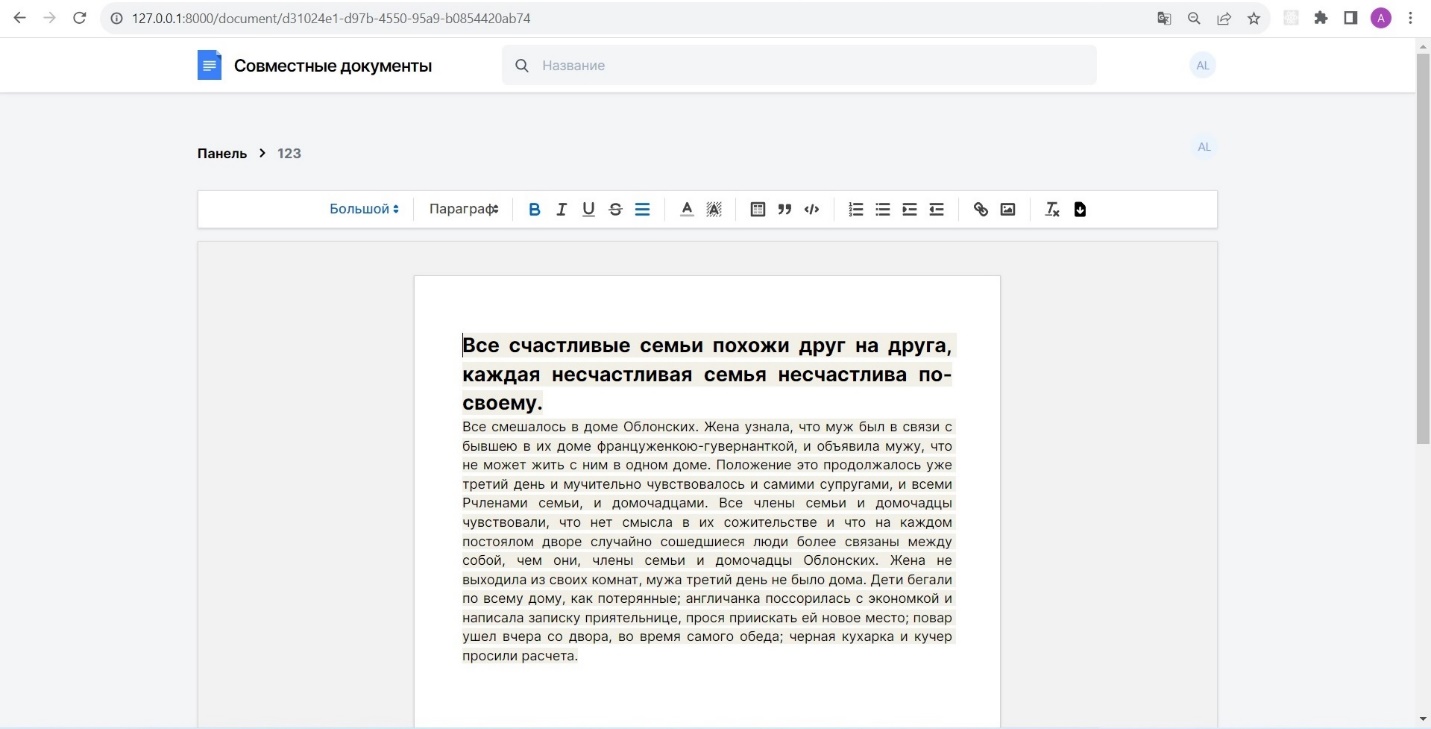


Рисунок 4.1.6 – Страница редактирования документа

Теперь же на рисунке 4.1.7 рассмотрим веб-приложение со стороны администратора. У администратора есть возможность дать пользователю доступ к документу и др.

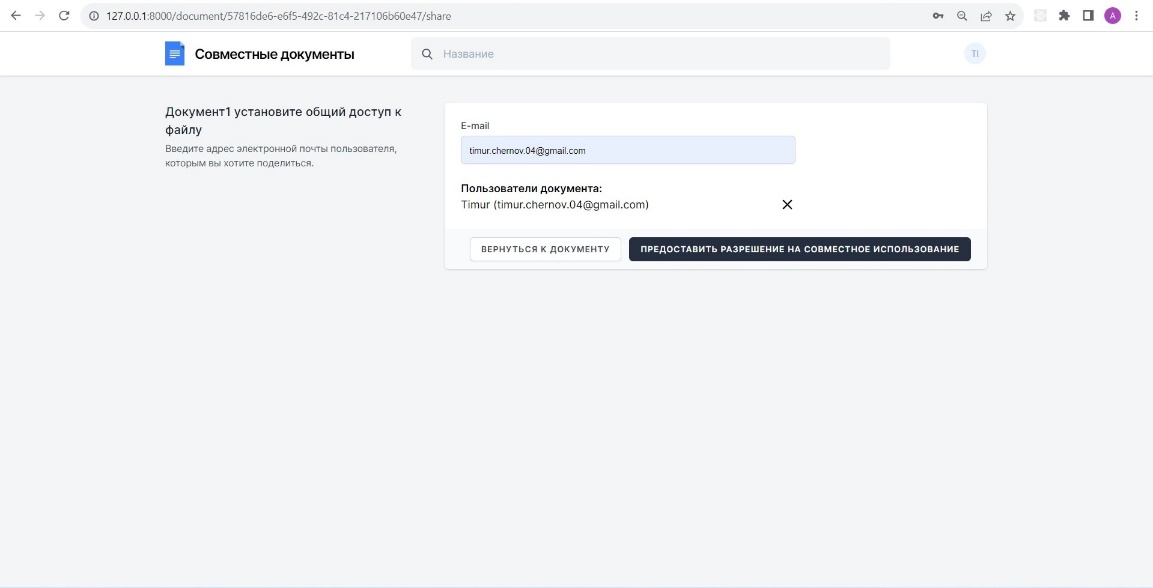


Рисунок 4.1.7 – Панель администратора

## Создание программно-аппаратной части веб-приложения

В файле artisan.php установлено соединение с системными файлами, а также подключены зависимости Composer.

Программный код представлен в листинге 4.2.1:

Листинг 4.2.1 – Код файла artisan.php

<?php

define('LARAVEL\_START', microtime(true));

require \_\_DIR\_\_.'/vendor/autoload.php';

$app = require\_once \_\_DIR\_\_.'/bootstrap/app.php';

$kernel = $app->make(Illuminate\Contracts\Console\Kernel::class);

$status = $kernel->handle(

$input = new Symfony\Component\Console\Input\ArgvInput,

new Symfony\Component\Console\Output\ConsoleOutput

);

$kernel->terminate($input, $status);

exit($status);

Если говорить о дополнительной информации по языку PHP, то она была получена из официальной документации [11], по менеджеру зависимостей Composer была получена также на официальной документации [12].

Затем, была создана база данных MySQL, необходимая для хранения данных будущего интернет-ресурса.

Сущности пользователей состоят из полей идентификационного номера, почты, имени, пароля и т.д.

Данные о пользователях после регистрации хранятся в базе данных, которая создана с помощью MySQL диалекта языка SQL. Если говорить о дополнительной информации и возможностях MySQL, то она была получена из официальной документации [13]. На рисунке 4.2.2 можно рассмотреть таблицу с данными пользователей.

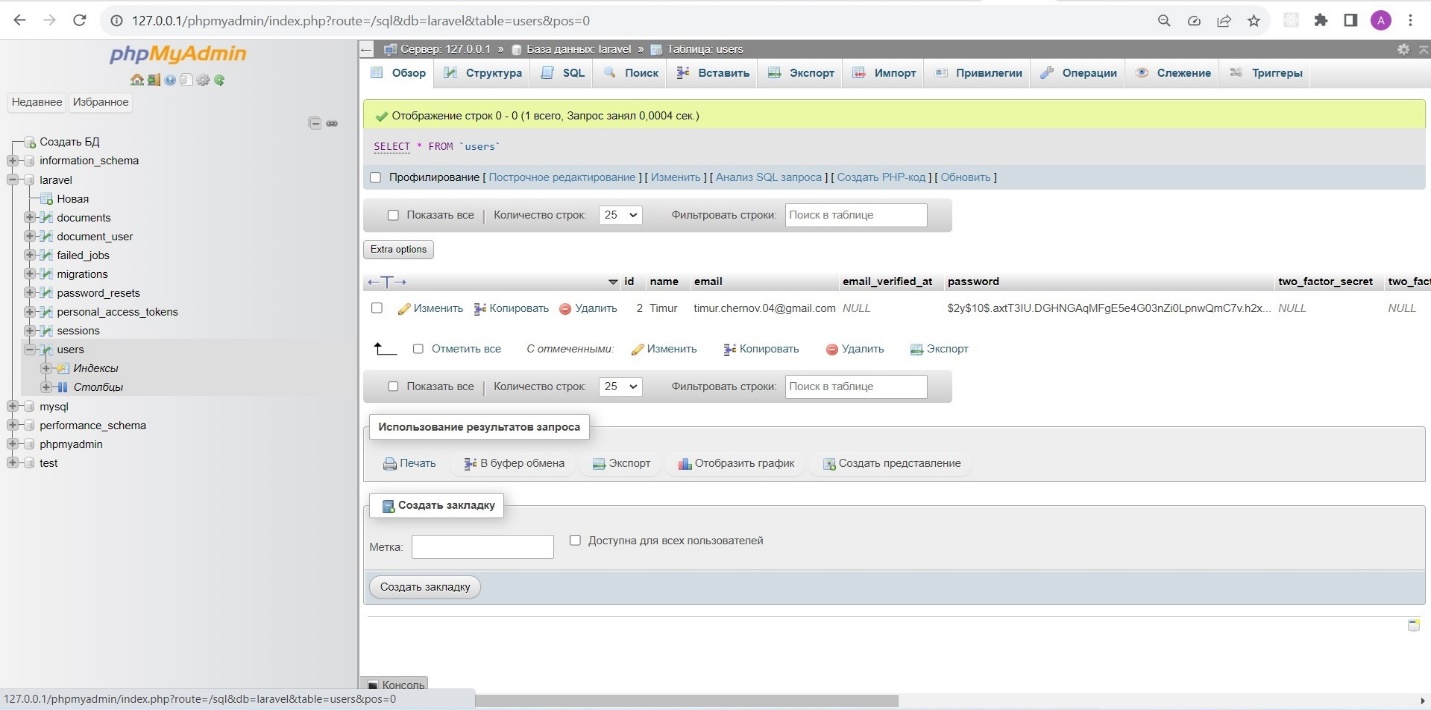


Рисунок 4.2.2 – Данные пользователей

Поскольку MySQL – реляционная СУБД, поэтому все таблицы связаны между собой и взаимодействуют друг с другом. Рассмотреть визуализацию же можно на рисунке 4.2.3.

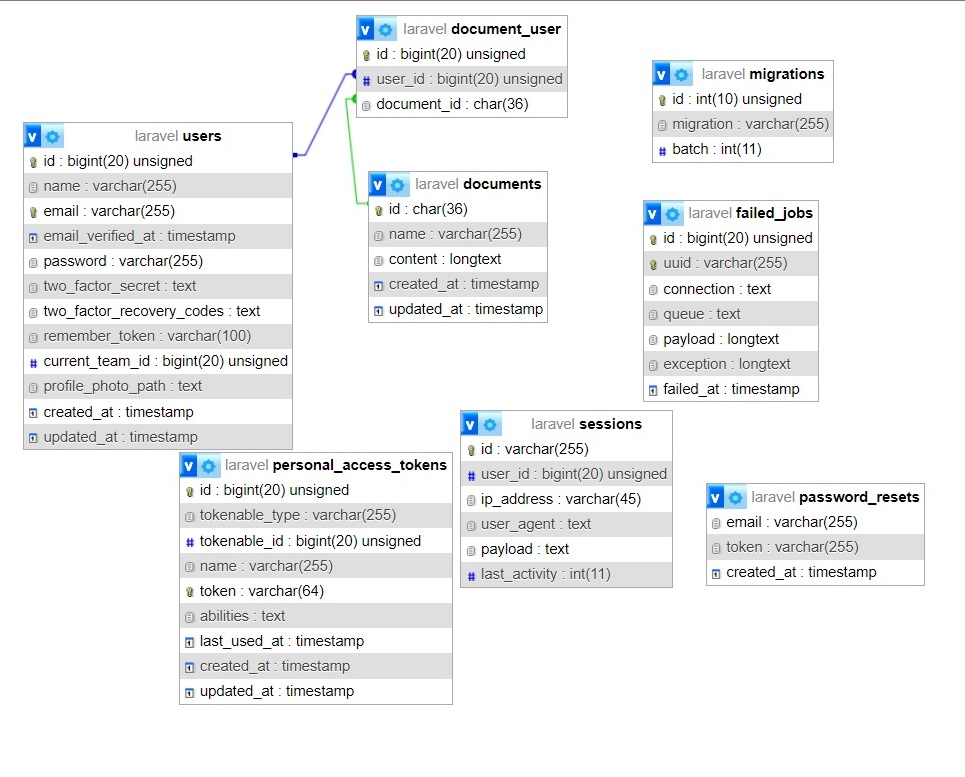


Рисунок 4.2.3 – Визуализация БД

Рассмотрим теперь подробнее, что содержат некоторые таблицы.

Таблица “documents”:

1. content: longtext – контент документа;
2. created\_at: timestamp – дата создания;
3. updated\_at: timestamp – дата обновления;
4. name: varchar(255) – название документа;
5. id: char – уникальный ключ документа.

Таблица “document\_user”:

1. document\_id: char – id документа;
2. user\_id: bigint –порядок сортировки;
3. id: bigint – уникальный ключ пользователя документа.

Таблица “failed\_jobs”:

1. connection: text – информация о соединении;
2. payload: longtext – информация о загрузке;
3. queue: text – информация об очереди;
4. exception: lontext – информация о ошибке;
5. failed\_at: timestamp – дата возникновения ошибки;
6. uuid: varchar – универсальный уникальный идентификатор;
7. id: bigint – уникальный ключ ошибки.

Таблица “migrations”:

1. id: int – уникальный номер пользователя;
2. batch: id – имя;
3. migration: varchar(255) – фамилия.

Таблица “password\_resets”:

1. token: varchar(255) – новое значение пароля;
2. email: varchar(255) – почта пользователя;
3. created\_at: timestamp – дата изменения пароля.

Саму же структуру таблицы пользователей можно рассмотреть на листинге 4.2.4.

Листинг 4.2.4 – Структура таблицы ‘users’

CREATE TABLE users (

id bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

name varchar(255) NOT NULL,

email varchar(255) NOT NULL,

email\_verified\_at timestamp NULL DEFAULT NULL,

password varchar(255) NOT NULL,

two\_factor\_secret text DEFAULT NULL,

two\_factor\_recovery\_codes text DEFAULT NULL,

remember\_token varchar(100) DEFAULT NULL,

current\_team\_id bigint(20) unsigned DEFAULT NULL,

profile\_photo\_path text DEFAULT NULL,

created\_at timestamp NULL DEFAULT NULL,

updated\_at timestamp NULL DEFAULT NULL,

PRIMARY KEY (id),

UNIQUE KEY users\_email\_unique (email),

KEY id (id,name,email)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci

Приложение написано на основе MVC, поскольку паттерны MVC разделяют аспекты приложения и обеспечивают свободную связь между ними, но для связи с базой данных необходимо написать класс.

Программный код представлен в листинге 4.2.5:

Листинг 4.2.5 – Код файла database.php

<?php

use Illuminate\Support\Str;

return [

'default' => env('DB\_CONNECTION', 'mysql'),

'connections' => [

'sqlite' => [

'driver' => 'sqlite',

'url' => env('DATABASE\_URL'),

Продолжение листинга 4.2.5

'database' => env('DB\_DATABASE', database\_path('database.sqlite')),

'prefix' => '',

'foreign\_key\_constraints' => env('DB\_FOREIGN\_KEYS', true),

],

'mysql' => [

'driver' => 'mysql',

'url' => env('DATABASE\_URL'),

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '3306'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'unix\_socket' => env('DB\_SOCKET', ''),

'charset' => 'utf8mb4',

'collation' => 'utf8mb4\_unicode\_ci',

'prefix' => '',

'prefix\_indexes' => true,

'strict' => true,

'engine' => null,

'options' => extension\_loaded('pdo\_mysql') ? array\_filter([

PDO::MYSQL\_ATTR\_SSL\_CA => env('MYSQL\_ATTR\_SSL\_CA'),

]) : [],

],

'pgsql' => [

'driver' => 'pgsql',

'url' => env('DATABASE\_URL'),

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '5432'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'charset' => 'utf8',

'prefix' => '',

'prefix\_indexes' => true,

'schema' => 'public',

'sslmode' => 'prefer',

],

Продолжение листинга 4.2.5

'sqlsrv' => [

'driver' => 'sqlsrv',

'url' => env('DATABASE\_URL'),

'host' => env('DB\_HOST', 'localhost'),

'port' => env('DB\_PORT', '1433'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'charset' => 'utf8',

'prefix' => '',

'prefix\_indexes' => true,

],

],

'migrations' => 'migrations',

'redis' => [

'client' => env('REDIS\_CLIENT', 'phpredis'),

'options' => [

'cluster' => env('REDIS\_CLUSTER', 'redis'),

'prefix' => env('REDIS\_PREFIX', Str::slug(env('APP\_NAME', 'laravel'), '\_').'\_database\_'),

],

'default' => [

'url' => env('REDIS\_URL'),

'host' => env('REDIS\_HOST', '127.0.0.1'),

'password' => env('REDIS\_PASSWORD', null),

'port' => env('REDIS\_PORT', '6379'),

'database' => env('REDIS\_DB', '0'),

],

'cache' => [

'url' => env('REDIS\_URL'),

'host' => env('REDIS\_HOST', '127.0.0.1'),

'password' => env('REDIS\_PASSWORD', null),

'port' => env('REDIS\_PORT', '6379'),

'database' => env('REDIS\_CACHE\_DB', '1'),

],

],

];

Файл server.php позволяет нам создать веб-сервер.

Программный код представлен в листинге 4.2.6:

Листинг 4.2.6 – Код класса server.php

<?php

$uri = urldecode(

parse\_url($\_SERVER['REQUEST\_URI'], PHP\_URL\_PATH)

);

if ($uri !== '/' && file\_exists(\_\_DIR\_\_.'/public'.$uri)) {

return false;

}

require\_once \_\_DIR\_\_.'/public/index.php';

Рассмотрим создание страницы регистрации. Создадим файл register.blade.php. Необходимо сначала расписать переменные для формы и их проверить. Код класса register.blade.php для регистрации представлен в листинге 4.2.7:

Листинг 4.2.7 – Код класса register.blade.php

<x-guest-layout>

<x-jet-authentication-card>

<x-slot name="logo">

<x-jet-authentication-card-logo />

</x-slot>

<x-jet-validation-errors class="mb-4" />

<form method="POST" action="{{ route('register') }}">

@csrf

<div>

<x-jet-label for="name" value="{{ \_\_('Name') }}" />

<x-jet-input id="name" class="block mt-1 w-full" type="text" name="name" :value="old('name')" required autofocus autocomplete="name" />

</div>

<div class="mt-4">

<x-jet-label for="email" value="{{ \_\_('Email') }}" />

<x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required />

</div>

Продолжение листинга 4.2.7

<div class="mt-4">

<x-jet-label for="password" value="{{ \_\_('Password') }}" />

<x-jet-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />

</div>

<div class="mt-4">

<x-jet-label for="password\_confirmation" value="{{ \_\_('Confirm Password') }}" />

<x-jet-input id="password\_confirmation" class="block mt-1 w-full" type="password" name="password\_confirmation" required autocomplete="new-password" />

</div>

<div class="flex items-center justify-end mt-4">

<a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{ route('login') }}">

{{ \_\_('Already registered?') }}

</a>

<x-jet-button class="ml-4">

{{ \_\_('Register') }}

</x-jet-button>

</div>

</form>

</x-jet-authentication-card>

</x-guest-layout>

Говоря об анимации на сайте, то она была разработана с помощью языка JavaScript. Ниже в листинге 4.2.8 представлен фрагмент файла app.js:

Листинг 4.2.8 – Код файла app.js

!function(e){

var t={};

function n(r){

if(t[r])return t[r].exports;

var o=t[r]={i:r,l:!1,exports:{}};

return e[r].call(o.exports,o,o.exports,n),o.l=!0,o.exports}n.m=e,n.c=t,n.d=function(e,t,r){n.o(e,t)||Object.defineProperty(e,t,{enumerable:!0,get:r})},n.r=function(e)

{"undefined"!=typeof Symbol&&Symbol.toStringTag&&Object.defineProperty(e,Symbol.toStringTag,{value:"Module"}),Object.defineProperty(e,"\_\_esModule",{value:!0})},n.t=function(e,t){if(1&t&&(e=n(e)),8&t)return e;

if(4&t&&"object"==typeof e&&e&&e.\_\_esModule)return e;

var r=Object.create(null);

if(n.r(r),Object.defineProperty(r,"default",{enumerable:!0,value:e}),2&t&&"string"!=typeof e)

for(var o in e)n.d(r,o,function(t){return e[t]}.bind(null,o));return r},n.n=function(e){var t=e&&e.\_\_esModule?function(){return e.default}:function(){return e};

return n.d(t,"a",t),t},n.o=function(e,t){return Object.prototype.hasOwnProperty.call(e,t)},n.p="/",n(n.s=0)}([function(e,t,n){n(1),e.exports=n(2)},function(e,t){},function(e,t){}]);

## Выводы к разделу 4

На основе требований, описанных в разделах 1, 2 и 3, были сформированы более детальные задачи и функции, благодаря чему была проведена разработка серверной части приложения «Совместный редактор документов» с использованием инструментария и технологий, описанных в разделах 1-3.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы была достигнута поставленная цель, которая заключалась в том, чтобы создать серверную часть интернет-ресурса на тему «Совместный редактор документов», согласно всем перечисленным требованиям. В процессе разработки были освоены такие технологии, как PHP, паттерн проектирования Model-View-Controller, разработка клиентской части приложения с помощью HTML5, Bootstrap, CSS3 и JavaScript. Также была освоена система управления базами данных MySQL, благодаря которой была создана реляционная база данных с таблицами, хранящих данные о документах и пользователях.

Отчет сформирован согласно учебно-методическому пособию по КР [14]. Курсовая работа выполнена в соответствие с требованиями ГОСТ 7.32-2017 [15].

Результат работы можно посмотреть на GitHub по ссылке, находящейся в Приложении А.

Для достижения поставленной цели было необходимо решить следующие задачи: провести анализ предметной области и сформировать основные требования к приложению, обосновать выбор средств ведения разработки, разработать архитектуру веб-приложения на основе выбранного паттерна проектирования, разработать веб-приложение с использованием PHP, выбранной технологии и инструментария, и реализовать слой серверной логики, базы данных и клиентской логики веб-приложения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Совместный редактор документов Google Docs [Электронный ресурс]. — URL: https://www.google.ru/intl/ru/docs/about/ (дата обращения 22.09.2023)
2. Совместный редактор документов Яндекс. Документы [Электронный ресурс]. — URL: https://docs.yandex.ru/docs?type=docx (дата обращения 22.09.2023)
3. Совместный редактор документов Zoho Writer [Электронный ресурс]. — URL:  https://writer.zoho.eu/writer/open/cda50dd3048008985440e8d7be98dc218efc4 (дата обращения 22.09.2023)
4. PHPStorm [Электронный ресурс]. — URL: https://www.jetbrains.com/ru-ru/phpstorm/features/?ysclid=lay219dlw2565714308 (дата обращения 23.09.2023)
5. Composer [Электронный ресурс]. — URL: https://htmlacademy.ru/tutorial/php/composer?ysclid=lay1jhz0rk643714057 (дата обращения 23.09.2023)
6. Базы данных. Курс лекций и материалы для практических заданий. – Учебное пособие. – М.: Питер, 2013. [Электронный ресурс]. — URL: https://publications.hse.ru/mirror/pubs/share/direct/259052819?ysclid=lb9t7abivd555038652 (дата обращения 23.09.2023)
7. MVC [Электронный ресурс]. — URL: https://habr.com/ru/post/150267/ (дата обращения 23.09.2023)
8. Localhost [Электронный ресурс]. — URL: https://habr.com/ru/post/150267/ (дата обращения 23.09.2023)
9. Иванов, К. К. Использование шаблона проектирования MVC в разработке современных информационных систем. [Электронный ресурс]. — URL: https://moluch.ru/archive/153/43308/ (дата обращения 24.09.2023)
10. MySQL [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/MySQL (дата обращения 24.09.2023)
11. PHP: Documentation [Электронный ресурс]. — URL: [https://www.php.net/docs.php](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fwww.php.net%2Fdocs.php) (дата обращения 24.09.2023)
12. Composer Documentation [Электронный ресурс]. — URL: [https://getcomposer.org/doc/01-basic-usage.md](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fgetcomposer.org%2Fdoc%2F01-basic-usage.md) (дата обращения 24.09.2023)
13. MySQL Documentation [Электронный ресурс]. — URL: [https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-31.html](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fdev.mysql.com%2Fdoc%2Frelnotes%2Fmysql%2F8.0%2Fen%2Fnews-8-0-31.html) (дата обращения 24.09.2023)
14. Метод КР РСЧИР 2021 для студентов [Электронный ресурс]. — URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F967557%2Fmod\_resource%2Fcontent%2F0%2F%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_%D0%9A%D0%A0\_%D0%A0%D0%A1%D0%A7%D0%98%D0%A0\_2021\_%D0%B4%D0%BB%D1%8F\_%D1%81%D1%82%D1%83%D0%B4%D0%B5%D0%BD%D1%82%D0%BE%D0%B2\_docx.pdf (дата обращения 25.09.2023)
15. ГОСТ 7.32-2017. Межгосударственный стандарт. [Электронный ресурс]. — URL: https://www.rea.ru/ru/org/managements/orgnirupr/Documents/gost\_7.32-2017.pdf?ysclid=lb9rmz4hzj573332082 (дата обращения 25.09.2023)

# ПРИЛОЖЕНИЕ А

# (справочное)

# Исходный код веб-приложения

Исходный код веб-приложения хранится по ссылке - https://github.com/OgGospod1n/SRD