Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Институт информационных технологий

Факультет компьютерных технологий

Кафедра программного обеспечения информационных технологий

|  |
| --- |
| *К защите допустить*: |
| Заведующий кафедрой ПОИТ |
| \_\_\_\_\_\_\_\_\_\_\_\_Н.В. Лапицкая |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**ВЕБ-Приложение для УЧЕТА МОБИЛЬНЫХ УСТРОЙСТВ ЗАО «КЬЮЛИКС СИСТЕМС» на языке программирования c#**

БГУИР ДП 1-40 01 01 01 095 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А.В. Третьякова |
| Руководитель |  | Д.В. Горбачев |
| Консультанты: |  |  |
| *от кафедры* |  | Д.В. Горбачев |
| *по экономической части* |  | С.В. Наркевич |
|  |  |  |
| Нормоконтролер |  | С.В. Болтак |
|  |  |  |
| Рецензент |  |  |

Минск 2022

**РЕФЕРАТ**

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА МОБИЛЬНЫХ УСТРОЙСТВ ЗАО «КЬЮЛИКС СИСТЕМС»: дипломный проект/ А.В. Третьякова. – Минск: БГУИР, 2022, – п.з. – 103 с., чертежей (плакатов) – 6 л. формата А1.

Объектом исследования является процесс учета мобильных устройств ЗАО «Кьюликс Системс».

Цель дипломного проекта: проектирование и разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

Разработка данного программного средства обеспечит возможность вести учет мобильных устройств ЗАО «Кьюликс Системс», что упростит организацию работы сотрудников организации.

Проанализированы существующие программные средства для учета компьютерной техники в организации, выполнена постановка задач.

Выполнено моделирование предметной области, разработаны функциональная модель в нотации IDEF0, диаграмма вариантов использования и функциональные требования к программному средству. В процессе проектирования программного средства тестирования и контроля знаний разработаны схема модели базы данных, а также ряд диаграмм в нотации UML. На основании проектной документации выполнена реализация программного средства. Проведено тестирование и разработана использования программного средства.

Результаты проектирования: функциональное назначение программного средства, описаны аналоги программного средства, разработаны алгоритмы работы, спроектирована логическая и физическая модель базы данных, описана структура программного средства с отображением диаграмм классов, уделено внимание вопросам технико-экономического обоснования, проведено тестирование работоспособности разработанного программного средства, разработана методика использования программного средства.

В разделе технико-экономического обоснования были произведены расчеты затрат, связанных с реализацией проекта, а также рентабельности разработки проекта. Проведенные расчеты показали экономическую целесообразность проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ  Институт информационных технологий | | | |
|  | | | |
| Факультет | КТ | Кафедра | ПОИТ |
| Специальность | 1–40 01 01 | Специализация | 01 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УТВЕРЖДАЮ | | | | | |
|  |  | |  | Н.В. Лапицкая | |
| « |  | » | 20 | | г. |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ЗАДАНИЕ  по дипломному проекту студента | | | | | | | | | |
|  | | | | | | | | | |
| **Третьяковой Анастасии Владимировны** | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | |
|  | | | | | | | | | |
| 1. Тема проекта: | **ВЕБ–ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА МОБИЛЬНЫХ** | | | | | | | | |
| **УСТРОЙСТРВ ЗАО «КЬЮЛИКС СИСТЕМС» НА ЯЗЫКЕ** | | | | | | | | | |
| **ПРОГРАММИРОВАНИЯ C#** | | | | | | | | | |
|  | | | | | | | | | |
| утверждена приказом по университету от | | | « | 01 | » | ноября | 2021 г. | № | 224-и |
|  | | | | | | | | | |
| 2. Срок сдачи студентом законченной работы | | | | 03 января 2022 г. | | | | | |
|  | | | | | | | | | |
| 3. Исходные данные к проекту | | Тип операционной системы: Windows 11; | | | | | | | |
| Языки программирования: C#; | | | | | | | | | |
| Назначение разработки: учет мобильных устройств ЗАО «Кьюликс Системс». | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | |
| Введение | | | | | | | | | |
| 1. Анализ литературных источников и формирование требований к проектируемому | | | | | | | | | |
| программному средству | | | | | | | | | |
| 2. Моделирование программного средства | | | | | | | | | |
| 3. Проектирование программного средства | | | | | | | | | |
| 4. Тестирование программного средства | | | | | | | | | |
| 5. Методика использования программного средства | | | | | | | | | |
| 6. Технико-экономическое обоснование разработки и использования программного | | | | | | | | | |
| средства | | | | | | | | | |
| Заключение | | | | | | | | | |
| Список использованных источников | | | | | | | | | |
| Приложение А. Текст программы | | | | | | | | | |

|  |
| --- |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала) |
| Алгоритм проверки уровня доступа на стороне сервера. Схема алгоритма – формат А1, лист 1. |
| Алгоритм добавления сотрудника. Схема алгоритма – формат А1, лист 1. |
| Алгоритм добавления устройства. Схема алгоритма – формат А1, лист 1. |
| Диаграмма вариантов использования. Плакат – формат А1, лист 1. |
| Схема базы данных. Плакат – формат А1, лист 1. |
| Диаграмма классов. Плакат – формат А1, лист 1. |
|  |
|  |
|  |
| 6. Технико-экономическое обоснование дипломного проекта: |
| Описание функций, назначения и потенциальных пользователей программного средства |
| Расчет затрат на разработку программного средства |
| Оценка экономического эффекта от разработки и применения программного средства |
| для собственных нужд организации |

|  |  |  |
| --- | --- | --- |
| Задание выдал |  | / С.В. Наркевич / |

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной |  |  |  |
| области, разработка технического задания | 10 | 27.10 – 09.11 |  |
| Моделирование предметной области | 20 | 10.11 – 18.11 |  |
| Проектирование программного средства | 20 | 19.11 – 25.11 |  |
| Тестирование программного средства | 10 | 26.11 – 02.12 |  |
| Методика использования разработанного |  |  |  |
| программного средства | 20 | 02.12 – 16.12 |  |
| Оформление пояснительной записки |  |  |  |
| и графического материала | 20 | 17.12 – 31.12 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 01 ноября 2021 г. | | Руководитель | |  | / Д.В. Горбачев / |
|  | | | | | | |
| Задание принял к исполнению | |  | | / А.В. Третьякова / | | |

СОДЕРЖАНИЕ

[Введение 7](#_Toc89812136)

[1 Анализ литературных источников и формирование требований к проектируемому программному средству 8](#_Toc89812137)

[1.1 Описание предметной области 8](#_Toc89812138)

[1.2 Анализ существующих аналогов 9](#_Toc89812139)

[1.3 Постановка цели и задач 15](#_Toc89812140)

[1.4 Входные данные 15](#_Toc89812141)

[1.5 Выходные данные 16](#_Toc89812142)

[2 Моделирование программного средства 17](#_Toc89812143)

[2.1 Функциональные требования 17](#_Toc89812144)

[2.2 Описание функциональности программного средства 17](#_Toc89812145)

[2.3 Разработка информационной модели 19](#_Toc89812146)

[2.4 Разработка модели взаимодействия пользователя с интерфейсом 21](#_Toc89812147)

[2.5 Разработка технических требований к программному средству 22](#_Toc89812148)

[3 Проектирование программного средства 23](#_Toc89812149)

[3.1 Обоснование выбора среды разработки 23](#_Toc89812150)

[3.2 Разработка структурной схемы программного средства 25](#_Toc89812151)

[3.3 Разработка структуры классов 29](#_Toc89812152)

[3.4 Разработка физической модели данных 31](#_Toc89812153)

[3.5 Проектирование алгоритмов работы программного средства 34](#_Toc89812154)

[4 Тестирование программного средства 38](#_Toc89812155)

[5 Методика использования программного средства 45](#_Toc89812156)

[5.1 Регистрация пользователя 45](#_Toc89812157)

[5.2 Авторизация пользователя 46](#_Toc89812158)

[5.3 Создание и редактирование заметки 48](#_Toc89812159)

[5.4 Поиск заметок 55](#_Toc89812160)

[5.5 Поделиться заметкой 56](#_Toc89812161)

[5.6 Администратор 57](#_Toc89812162)

[6 Технико-экономическое обоснование разработки и использования программного средства 58](#_Toc89812163)

[6.1 Описание функций, назначения и потенциальных пользователей программного средства 58](#_Toc89812164)

[6.2 Расчет затрат на разработку программного средства 58](#_Toc89812165)

[6.3 Оценка экономического эффекта от разработки и применения программного средства для собственных нужд организации 61](#_Toc89812166)

[Заключение 63](#_Toc89812167)

[Список использованых источников 64](#_Toc89812168)

[Приложение А. Текст программы 65](#_Toc89812169)

**ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В настоящей пояснительной записке применяются следующие определения и сокращения.

*Закрытое акционерное общество (ЗАО)* – акционерное общество, акции которого распределяются только среди учредителей или заранее определенного круга лиц (в противоположность открытому) [11].

*Авторизация* – предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

*Аутентификация* – проверка подлинности предъявленного пользователем идентификатора.

*БД* – база данных.

*ОС* – операционная система.

*Программа* – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.

*Программное средство* *(ПС)* – совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

API – application programming interface (интерфейс программирования приложений).

DOM – Document Object Model (Объектная модель документа).

HTTPS – HyperText Transfer Protocol Secure (Безопасный протокол передачи гипертекста).

JSON – Javascript Object Notation.

JWT – JSON Web Token (это JSON объект, который определен в открытом стандарте RFC 7519).

LINQ – Language-Integrated Query (Интегрированный язык запросов).

SPA – Single Page Application (Одностраничное приложение).

*Веб-приложение* – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера, которое выполняет функцию контроля пользования мобильными устройствами [1].

*Мобильное устройство* – переносное устройство, принадлежащее организации, с операционной системой IOS или Android.

*Администратор* – сотрудник ЗАО «Кьюликс Системс», который ведет учет мобильных устройств.

*Держатель (сотрудник)* – пользователь мобильного устройства в текущий момент времени.

Ящик – физическое общедоступное место хранения устройств, не связанных с держателем.

Короткое имя держателя – короткое имя сотрудника ЗАО «Кьюликс Системс», указанное в названии компьютера.

Статус – статус мобильного устройства в веб-приложении.

# ВВЕДЕНИЕ

Актуальность учета компьютерной и другой техники в организации возрастает при наличии большого количества компьютеров, офисной техники и другого оборудования.

Автоматизация сфер деятельности организации, будь то сбор данных или управление различными технологическими процессами, позволяет ускорить работу, сделать ее более точной и эффективной, избежать потерь нужной информации, ошибок персонала, дублирования документов. Для эффективности работы по учету техники необходимо проводить мероприятия по обновлению материально-технической и информационной базы, внедрению современных методов управления с использованием автоматизированных систем, созданию базы данных о технике организации, его своевременному пополнению, оперативному представлению необходимой информации пользователям.

Целью дипломного проекта является разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

Задачи дипломного проекта:

* изучить аналоги программного средства по учету компьютерной либо любой другой техники в организации;
* спроектировать информационную структуру разрабатываемого программного средства;
* разработать схему базы данных и пользовательский интерфейс для доступа к данным;
* провести расчет технико-экономических показателей разрабатываемого программного средства.

Объектом дипломного проекта является процесс учета мобильных устройств ЗАО «Кьюликс Системс».

Предметом дипломного проекта является автоматизация учета мобильных устройств ЗАО «Кьюликс Системс».

Практическая значимость дипломного проекта состоит в том, что использование веб-приложения сократит трудозатраты пользователей за счет снижения трудоемкости выполнения «ручных» операций и сокращения времени на устранение инцидентов (сбой в работе устройства, нежелательное обновление операционной системы).

# 1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

## 1.1 Описание предметной области

Каждая организация приходит к тому, что требуется навести полный порядок в инфраструктуре организации, получить контроль над перемещениями техники, спланировать сервисные работы, заказы и закупки, списание техники. Необходимо иметь возможность в любой момент централизованно получать полную информацию о состоянии подотчетного оборудования.

Учет – это количественное отражение событий и фактов, используемое для управления обществом на различных организационных уровнях. Учет представляет собой сбор, обработку, классификацию, систематизацию и отражение информации в специальных регистрах на каких-либо носителях. В экономике используются различные виды учета: бухгалтерский, оперативно-технический, статистический и другие. Общая цель всех видов учета – упорядочение большого количества информации для эффективного использования в управленческих решениях, формирования отчетности и сохранение информации для архива [3].

Оперативная информация используется для принятия управленческих решений, составления и анализа финансовой отчетности организации, долгосрочного и текущего планирования, исполнения налоговых обязательств. Отличительной особенностью оперативной информации является ее конфиденциальность. Оперативная информация должна быть доступна только для сотрудников организации. Управленческий (оперативный) учет – это внутреннее дело каждой организации. Для ведения этого учета могут быть использованы любые формы и регистры, разработанные самой организацией [4].

Все виды учета являются составной частью интегрированной управленческой информационной системы. При создании такой системы необходимо ответить на следующие вопросы:

* какая информация нужна (информация различается по степени детализации и стилю представления. Степень детализации оперативной информации зависит от иерархии управления. Детализация и стиль представления оперативной информации зависят от получателя. Высшему управленческому персоналу требуется обобщенная оперативная информация. Она должна быть наиболее интегрированной и с более длительным периодом представления. Обобщенная оперативная информация может представляться ежемесячно);
* кому нужна эта информация (оперативная информация предназначена для внутреннего пользования. Состав пользователей информации определяется структурой организации, которая зависит от вида деятельности. Пользователей информации можно разделить на две составляющие: управленческий персонал и непосредственно исполнители);
* почему нужна эта информация (оперативная информация необходима для принятия оперативных управленческих решений управленческим персоналом, реализации управленческих решений специалистами организации, определения отклонений достигнутых результатов от прогнозируемых, разработки мероприятий по выявленным отклонениям);
* когда она необходима (информация должна быть достоверной и достаточной для принятия управленческих решений);
* где получить информацию (информацию различают как первичную и вторичную. Источник информации, определяемый схемой документооборота, утвержденной руководством организации, можно отнести к вторичной информации. Внешними источниками информации могут быть поставщики и покупатели, конкуренты, реклама, государственные органы и др. Таким образом, наблюдение за внешней средой позволяет собрать первичную информацию, необходимую для бизнеса) [4].

Проектирование программного средства должно предусматривать использование информационных технологий и способы обмена информацией.

Требования к качеству информации. Основными принципами учета оперативной информации являются качественные характеристики этой информации и принципы ее учета. Основное требование к учетной информации – это ее полезность для принятия решений различными группами пользователей. Полезность информации определяется специальными характеристиками:

* ясность;
* уместность;
* содержательность;
* существенность;
* своевременность;
* полнота отражения;
* логичность;
* достоверность [4].

Таким образом, качественные характеристики оперативной учетной информации предопределяют полезность оперативной отчетности для пользователей. В нахождении оптимального сочетания всех характеристик проявляется профессионализм специалистов по контролю и экономистов-аналитиков. Соблюдение всех вышеперечисленных требований к качеству информации должно сочетаться с ограничением затрат и выгоды, которое состоит в том, что выгода от информации должна быть больше, чем затраты на ее получение. Выполнение требований к качеству оперативной учетной информации должно дать в результате правдивую, полную, полезную оперативную отчетность о деятельности организации.

## 1.2 Анализ существующих аналогов

Сейчас создано много различных программных средств для учета различной техники в организации: с простым и сложным интерфейсом, бесплатные и платные программные средства. Но каждая программа содержит различные функции и распространяется по-разному.

### **1.2.1** Программное средство «Hardware Inspector»

ПС «Hardware Inspector» предназначено для инвентарного учета компьютеров и другой техники в организациях. Ориентировано на выполнение работ, связанных с учетом и планированием компьютерного (и иного) аппаратного обеспечения. Оно позволяет всегда быть в курсе всей информации о компьютерном парке компании, получать разнообразные отчеты, планировать его обслуживание, ремонт и обновление [5].

Программное средство «Hardware Inspector» решает задачи автоматизации инвентарного учета компьютерной техники и комплектующих, с возможностью хранения всей истории перемещений и обслуживания. Механизм ревизий рабочих мест предохраняет компьютеры и комплектующие от хищения и подмены. Детальный контроль за параметрами конфигурации компьютера обеспечивает свободу и оперативность действий по планированию модернизации и перераспределения устройств [5].

Функционал «Hardware Inspector» не ограничивается одним лишь учетом компьютеров – система позволяет автоматизировать почти все аспекты деятельности ИТ-подразделения. Набор из более чем 30 встроенных отчетов позволит оценить эффективность его работы и предоставить нужные отчеты вышестоящему руководству.

Основные возможности программного средства «Hardware Inspector»:

* учет комплектующих;
* отслеживание истории перемещения устройств, их ремонта, профилактики и инвентаризации;
* возможность не только ручного ввода данных, но и импорта информации из отчетов программ анализа конфигурации компьютеров AIDA32, EVEREST, ASTRA И ASTRA32;
* учет лицензий на программное средство;
* мониторинг инсталляции пользователями запрещенного ПС;
* мониторинг изменений, происходящих в конфигурациях компьютеров (аудит рабочих мест);
* учет заявок от пользователей;
* инвентаризация устройств с использованием штрих-кодов;
* гибкое разграничение доступа к данным;
* мощные механизмы поиска устройств, лицензий и прочего;
* большой набор настраиваемых отчетов, экспорт в различные форматы;
* поддержка многопользовательской работы с базой данных в сети [5].

Уникальность ПС «Hardware Inspector» заключается в возможности вести учет не просто текущего состояния параметров компьютера, а всей истории жизни каждого устройства.

Механизм поиска поможет найти интересующее устройство (группу устройств) по заданному критерию. В фильтре поиска могут участвовать такие параметры устройств, как тип, модель, производитель, продавец, инвентарный номер, серийный номер, произвольная строка поиска и так далее (смотрите рисунок 1.1).

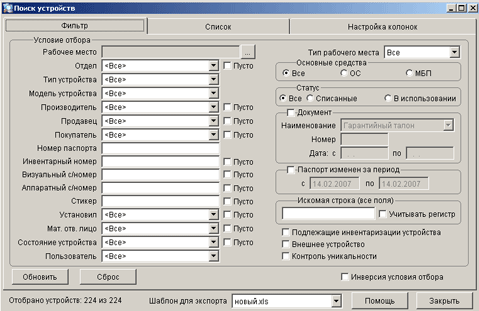


Рисунок 1.1 – Поиск устройства в «Hardware Inspector»

Достоинствами данной системы являются полный учет комплектующих, а не просто описание параметров рабочих станций; сопровождение системы после покупки и регулярное обновление системы (смотрите справочник типов и моделей устройств на рисунке 1.2).

Имеет гибкую систему разграничения прав доступа, большой набор встроенных отчетов, а также механизм создания отчетов, требуемых компании.

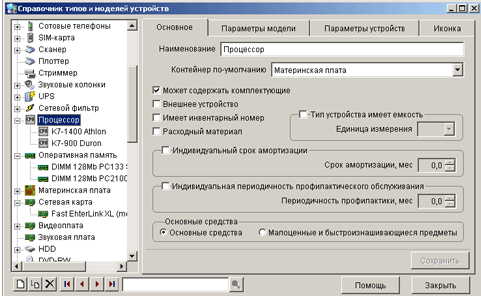


Рисунок 1.2 – Справочник типов и моделей устройств в «Hardware Inspector»

Данная система имеет закрытую архитектуру, в связи с чем отсутствует возможность вносить изменения в функционал системы. Такой недостаток не позволяет в полной мере внедрить систему в организацию, так же функционал, имеющийся в «Hardware Inspector», излишен для нужд ЗАО «Кьюликс Системс».

### **1.2.2** Программное средство «Инвентаризация сети и учета и компьютеров»

Программное средство «Инвентаризации сети и учет компьютеров» является разработкой компании Clear Apps.

Основная задача, которую выполняет система – это инвентаризация компьютерных сетей. В инвентаризацию сети входят учёт установленного программного средства, а также регистрация всего аппаратного парка сети [6].

Данная система позволяет вести учёт всей компьютерной техники и программного средства сети. Эта процедура просто необходима при планировании обновления аппаратного обеспечения, добавлении новых рабочих мест или закупки дополнительных аппаратных или программных ресурсов (смотрите главную страницу ПС на рисунке 1.3) [6].

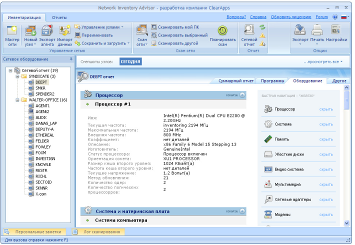


Рисунок 1.3 – Главная страница ПЛ «Инвентаризации сети и учет компьютеров»

Система также предназначена для контроля использования аппаратных средств и лицензий ПС. Инвентаризация сети необходима для контроля соответствия установленного программного и аппаратного средства с заданным стандартным набором для каждого рабочего места.

Данное программное средство имеет систему отчётности выполнения плановых мероприятий по всей ИТ-инфраструктуре компьютерной сети.

Имеется возможность добавлять в отчёты по инвентаризации компьютеры, которые не подключены к сети. Для этого в системе предусмотрен специальный модуль, обрабатывающий все данные, которые можно будет импортировать в интерфейс программы.

Для данной системы можно выделить следующие достоинства:

* автоматизация - сканирование сети осуществляется автоматически с высокой скоростью;
* наглядность – группировка собранной информации производится как для всей сети в целом, так и для каждого компьютера в отдельности;
* гибкость – имеется возможность добавлять специальные поля для каждого компьютера;
* удобство – собранные данные можно распечатать или экспортировать в нужный формат;
* отчёты – генератор отчётов и предустановленные шаблоны максимально ускорят процедуру построения отчётов по сети [6].

Недостатком системы можно выделить сложность ее внедрения: для внесения данных о компьютере, на него необходимо установить специальный модуль, сканирующий компьютер. Лишь после этого компьютер будет внесен в базу данных.

Кроме того, данный модуль проработан недостаточно хорошо, вследствие чего некоторые данные об оборудовании приходится вносить вручную. Для нужд ЗАО «Кьюликс Системс» данная система является слишком громоздкой и трудозатратой в плане переработки для собственных нужд.

### **1.2.3** Программное средство «Учет компьютеров»

Программное средство предназначено для учета компьютеров, оргтехники и других объектов в организации, разработана компанией ООО «Простой софт».

ПС позволяет фиксировать производимые ремонты, замены расходных материалов и частей, профилактические работы, установленное программное средство. Все объекты учета можно закреплять за сотрудниками с разбиением на отделы и филиалы (рабочее окно ПС «Учет компьютеров» смотрите на рисунке 1.4) [6].

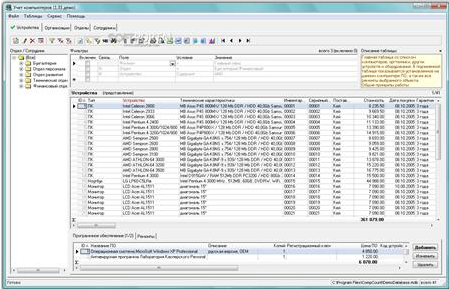


Рисунок 1.4 – Рабочее окно программы «Учет компьютеров»

ПС необходимо системным администраторам, руководителям ИТ – отделов, владельцам компьютерных магазинов, держателям домашних сетей, обладателям большого количества комплектующих и компьютерной техники.

Основные функции:

* ведение базы компьютеров и оргтехники: в базе данных содержится информация обо всех компьютерах и оргтехнике организации;
* предусмотрены удобные способы сортировки и фильтрации данных;
* поддерживается экспорт таблиц в MS Excel, MS Word или текстовый формат CSV, а также импорт из других источников данных;
* фиксация ремонтов и апгрейдов, а также контроль гарантийных сроков;
* учет установленного программного средства;
* закрепление объектов за ответственными сотрудниками: все объекты учета можно закреплять за ответственными сотрудниками с разбиением на отделы и филиалы [6].

Данная система позволяет не только вести учет оборудования, но и составлять бюджет отдела: спланировать потребность в необходимом количестве расходных материалов на определённый период, комплектующих для резерва на случай возможных отказов и тому подобное.

Многопользовательский режим работы позволяет распределить зоны ответственности: кто-то будет заносить в базу информацию о новом поступившем оборудовании, кто-то о поставщиках и прочих деловых партнерах. В результате обеспечивается согласованная работа сотрудников различных подразделений. Права на просмотр и изменение информации в системе настраиваются администратором в зависимости от функциональных обязанностей каждого пользователя.

Преимуществами системы можно выделить следующие возможности и функции:

* возможность изменения структуры данных – добавления таблиц, полей, связей, что очень важно при внедрении системы на предприятие;
* возможность создания шаблонов отчетов (счетов, актов и других документов);
* используется генерация документов Word на основе шаблонов (счетов, актов и других документов);
* поддерживается многопользовательский режим;
* настраиваемый пользовательский интерфейс [6].

Но при этом, анализируя отзывы пользователей, было выявлено, что при большом компьютерном парке работа базы данных и системы в целом начинает работать медленно, возникают различные ошибки и прерывания в работе системы. Для нужд ЗАО «Кьюликс Системс» данная система является слишком громоздкой и трудозатратой в плане переработки для собственных нужд.

**1.2.4** Вывод

Таким образом, проанализировав данные разработки, можно сделать следующие выводы.

Все системы используют автоматическое сканирование компьютеров для получения информации обо всех компьютерах сети. Для этого могут быть использованы как сторонние приложения (Everest, ASTRA и тому подобные), запускаемые на рабочих местах удаленно с помощью скриптов, так и встроенные модули системы.

Аналогичным образом происходит инвентаризация программного средства, установленного на компьютерах сети предприятия. Данная возможность также характерна для всех рассматриваемых систем и является неотъемлемой частью системы учета компьютеров.

Все системы поддерживают интеграцию с различными базами данных, а также возможность импорта отчетов во все распространенные форматы файлов. Кроме того, в базах данных рассмотренных систем предусмотрена возможность привязки каждого компьютера к конкретному сотруднику и отделу.

Обязательной характеристикой оборудования при инвентаризации является информация о гарантийном сроке, дате поступления и поставщике оборудования.

Также неотъемлемой частью всех систем учета является генерация разнообразных отчетов. При этом имеются не только встроенные, шаблонные, отчеты, но и у большинства систем имеется возможность создания собственных, индивидуальных, отчетов.

Анализируя выделенные у данных систем достоинства и недостатки, можно сказать, что нет необходимости в разработке или же в приобретении сторонних громоздких систем для учета мобильных устройств для нужд ЗАО «Кьюликс Системс», так как основная цель разрабатываемой системы – это учет местоположения девайса и сокращение времени реакции на какие-либо инциденты, связанные с девайсом, за счет того, что все отметки будут вестись в одном месте. Так же нет необходимости в предоставлении отчетности руководству о перемещениях мобильных устройств между сотрудниками.

## 1.3 Постановка цели и задач

Как показал анализ программных решений в области учета компьютерной техники, они обладают следующими недостатками:

* закрытая архитектура, в связи с чем отсутствует возможность вносить изменения в функционал системы;
* сложность внедрения и переработки имеющегося функционала ПС для нужд ЗАО «Кьюликс Системс»;
* громоздкость, то есть имеющийся функционал не нужен для учета мобильных устройств в ЗАО «Кьюликс Системс»;
* высокая стоимость.

Целью дипломного проекта является разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

ПС должно обеспечивать выполнение следующих функций:

* авторизация сотрудника;
* добавление, редактирование и удаление информации о сотрудниках (доступно только администратору);
* добавление, редактирование и удаление информации о мобильных устройствах (доступно только администратору);
* изменение статуса мобильного устройства;
* просмотр информации о мобильных устройствах;
* просмотр информации о сотрудниках (доступно только администратору);
* комментирование состояния мобильного устройства (необходимо для быстрого информирования о состоянии мобильного устройства (сбой в работе, нежелательное обновление ОС)).

## 1.4 Входные данные

К входной информации будут относиться все вводимые сотрудником и администратором данные.

Администратор при добавлении и редактировании информации о сотрудниках и мобильных устройствах будет указывать о них соответствующую информацию, также при авторизации сотрудник будет указывать свою персональную информацию (логин и пароль).

Так же сотрудник может оставлять комментарии по поводу состояния мобильного устройства.

Входной информацией будут является:

* данные для добавления, редактирования и удаления информации о сотрудниках;
* данные для добавления, редактирования и удаления информации о мобильных устройствах;
* данные для авторизации в системе;
* комментарии по статусу мобильного устройства.

## 1.5 Выходные данные

В качестве выходной информации будут выступать данные после добавления и сохранения отредактированной информации о сотрудниках и мобильных устройствах.

Система будет отдавать в качестве выходной информации список сотрудников (доступно только для администратора) и список мобильных устройств.

При изменении статуса мобильного устройства будет назначаться соответствующее мобильное устройство на сотрудника, который его взял.

К выходной информации также относятся системные данные: сессии и токены авторизации, ошибки системы.

К выходной информации относится:

* статус мобильного устройства;
* сессия и JWT;
* данные о сотрудниках;
* данные о мобильных устройствах;
* ошибки.

# 2 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Функциональные требования

Функциональным назначением разрабатываемого веб-приложения является предоставление возможности вести учет информации о мобильных устройствах ЗАО «Кьюликс Системс».

В качестве пользователя программного средства может выступать сотрудник ЗАО «Кьюликс системс», который добавлен администратором в данную систему и имеет доступ к персональному компьютеру с доступом к сети интернет. Для использования программного средства не требуется специальная подготовка или обучение пользователей.

Предполагается возможность одновременной эксплуатации разрабатываемого решения широким кругом пользователей.

Исходя из предполагаемого использования, можно заключить, что проектируемое программное средство должно реализовывать следующие группы функций:

* авторизация сотрудника;
* добавление, редактирование и удаление информации о сотрудниках (доступно только администратору);
* добавление, редактирование и удаление информации о мобильных устройствах (доступно только администратору);
* изменение статуса мобильного устройства;
* просмотр информации о мобильных устройствах;
* просмотр информации о сотрудниках (доступно только администратору);
* комментирование состояния мобильного устройства (необходимо для быстрого информирования о состоянии мобильного устройства (сбой в работе, нежелательное обновление ОС)).

## 2.2 Описание функциональности программного средства

Диаграмма вариантов использования ‒ это наиболее общее представление функционального назначения системы.

Диаграмма вариантов использования призвана ответить на главный вопрос моделирования: «что делает система во внешнем мире?». На диаграмме применяются два типа основных сущностей: варианты использования и действующие лица, между которыми устанавливаются следующие основные типы отношений:

* ассоциация между действующим лицом и вариантом использования;
* обобщение между действующими лицами;
* обобщение между вариантами использования;
* зависимости (различных типов) между вариантами использования [7].

В системе можно выделить следующие действующие лица:

* держатель (сотрудник);
* администратор.

На рисунке 2.1 представлена общая диаграмма вариантов использования.

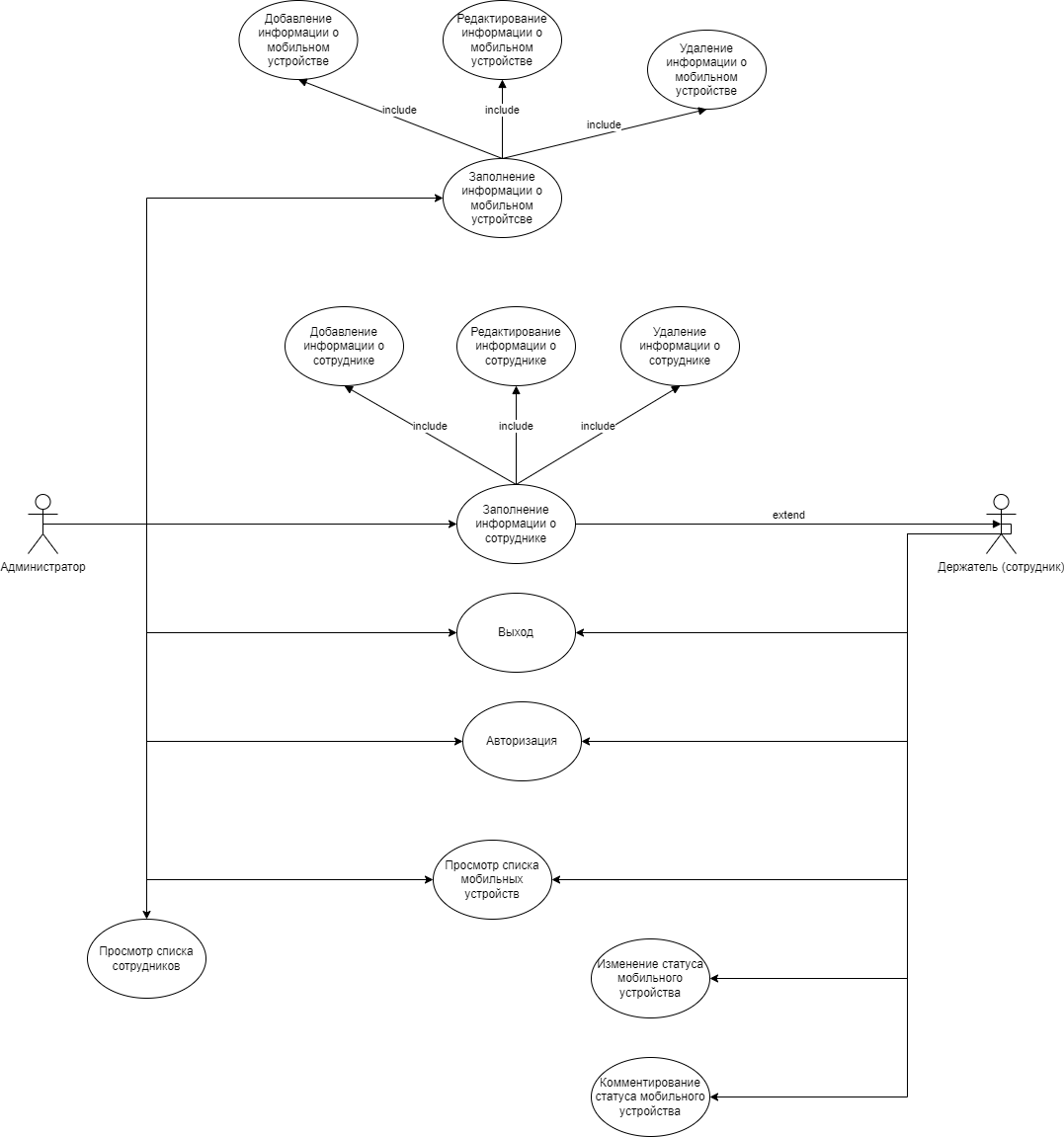


Рисунок 2.1 – Диаграмма вариантов использования

Учет мобильных устройств ЗАО «Кьюликс Системс» представляет собой процесс регистрации устройства, которое необходимо взять для работы, сотрудником в веб-приложении.

После добавления администратором сотрудника в систему, он может авторизоваться в ней.

После авторизации сотруднику доступен следующий функционал:

* просмотр списка мобильных устройств;
* изменение статуса мобильных устройств;
* комментирование статуса мобильных устройств;
* выход из системы.

В системе присутствует роль «Администратор», которая включает в себя возможности «Держателя», а также другие возможности такие как:

* добавление, редактирование и удаление информации о сотруднике;
* добавление, редактирование и удаление информации о мобильных устройствах;
* просмотр информации о сотрудниках.

## 2.3 Разработка информационной модели

Информационный объект – это описание некоторой сущности предметной области реального объекта, процесса, явления или события. Информационный объект образуется совокупностью логически взаимосвязанных реквизитов, представляющих качественные и количественные характеристики сущности.

Проанализировав возможные пользовательские действия была спроектирована информационная модель базы данных, представленная на рисунке 2.2.

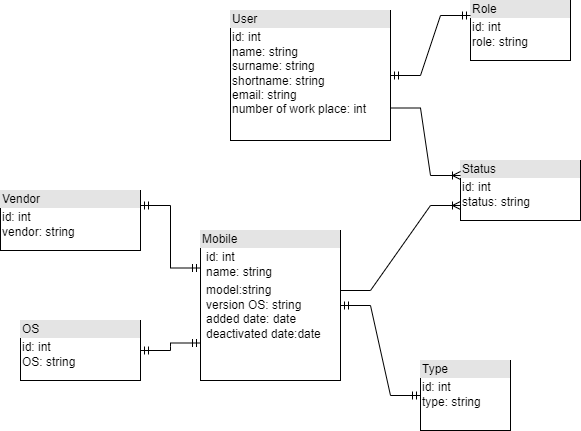


Рисунок 2.2 – Информационная модель БД

В процессе анализа предметной области были выделены следующие типы сущностей:

* User – сущность, хранящая пользовательские учетные записи с требуемой пользовательской информацией. Атрибуты: id – идентификатор сотрудника; name – имя сотрудника; surname – фамилия сотрудника; shortname – сокращенное имя сотрудника; email – почтовый адрес сотрудника; number of work place – номер рабочего места сотрудника;
* Role – сущность для хранения ролей в системе. Атрибуты: id – идентификатор роли; role – наименование роли;
* Mobile – сущность для хранения информации о больных устройствах. Атрибуты: id – идентификатор мобильного устройства; name – название мобильного устройства; model – модель мобильного устройства; version OS – версия операционной системы; added date – дата покупки мобильного устройства; deactivated date – дата деактивации мобильного устройства;
* Status – сущность для хранения статуса мобильного устройства. Атрибуты: id – идентификатор статуса мобильного устройства; status – статус мобильного устройства;
* Vendor – сущность для хранения информации о вендоре мобильного устройства. Атрибуты: id – идентификатор вендора мобильного устройства; vendor – наименование вендора мобильного устройства;
* OS – сущность для хранения информации об операционной системе мобильного устройства. Атрибуты: id – идентификатор операционной системы мобильного устройства; OS – наименование операционной системы мобильного устройства;
* Type – сущность для хранения информации о типе устройства. Атрибуты: id – идентификатор типа устройства; type – тип устройства.

## 2.4 Разработка модели взаимодействия пользователя с интерфейсом

Для архитектуры был выбран архитектурный шаблон MVC.

Шаблон архитектуры Model-View-Controller (MVC) разделяет приложение на три основных компонента: модель, представление и контроллер. Платформа ASP.NET MVC представляет собой альтернативу схеме веб-форм ASP.NET при создании веб-приложений. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например, с главными страницами и проверкой подлинности на основе членства. Платформа MVC определяется в сборке System.Web.Mvc. Схема шаблона разработки MVC представлен на рисунке 2.3.

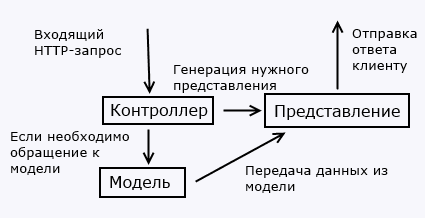


Рисунок 2.3 – Схема шаблона разработки Model View Controller

Шаблон разработки MVC позволяет создавать приложения, различные аспекты которых (логика ввода, бизнес-логика и логика интерфейса) разделены, но достаточно тесно взаимодействуют друг с другом. Эта схема указывает расположение каждого вида логики в приложении. Пользовательский интерфейс располагается в представлении. Логика ввода располагается в контроллере. Бизнес-логика находится в модели. Это разделение позволяет работать со сложными структурами при создании приложения, так как обеспечивает одновременную реализацию только одного аспекта. Например, разработчик может сконцентрироваться на создании представления отдельно от бизнес-логики [8].

Связь между основными компонентами приложения MVC также облегчает параллельную разработку. Например, один разработчик может создавать представление, другой – логику контроллера, а третий – бизнес-логику модели.

Платформа ASP.NET MVC имеет следующие преимущества:

* облегчает управление сложными структурами путем разделения приложения на модель, представление и контроллер;
* не использует состояние просмотра и серверные формы. Это делает платформу MVC идеальной для разработчиков, которым необходим полный контроль над поведением приложения;
* использует схему основного контроллера, при которой запросы веб-приложения обрабатываются через один контроллер. Это позволяет создавать приложения, поддерживающие расширенную инфраструктуру маршрутизации;
* обеспечивает расширенную поддержку разработки на основе тестирования;
* хорошо подходит для веб-приложений, поддерживаемых крупными коллективами разработчиков, а также веб-разработчикам, которым необходим высокий уровень контроля над поведением приложения [8].

## 2.5 Разработка технических требований к программному средству

Разрабатываемое программное средство должно обеспечивать корректное функционирование при развертывании программных модулей на сервере со следующими техническими характеристиками:

* Intel 2.2 Ггц или более быстродействующий процессор;
* оперативная память 4 Гбайт или более;
* доступный объем дискового пространства 100 Гбайт.

Для нормального функционирования клиентской части программного средства должны выполняться следующие технические требования:

* Pentium 2 Ггц или более быстродействующий процессор;
* оперативная память 2 Гбайт или более;
* 32- или 64-битная версия операционных систем Windows 10, 11;

браузер Google Chrome версии 95.x.

# 3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 3.1 Обоснование выбора среды разработки

В качестве среды разработки была выбрана Microsoft Visual Studio 2019 Community.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного средства и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного средства (например, клиент Team Explorer для работы с Team Foundation Server) [9].

Для написания данной системы была выбрана технология ASP.NET Core.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

* создавать веб-приложения и службы, приложения IoT и серверные части для мобильных приложений;
* использовать избранные средства разработки в Windows, macOS и Linux;
* выполнять развертывания в облаке или локальной среде;
* работать в .NET Core или .NET Framework [10].

ASP.NET Core предоставляет следующие преимущества:

* единое решение для создания пользовательского веб-интерфейса и веб-API;
* интеграция современных клиентских платформ и рабочих процессов разработки;
* облачная система конфигурации на основе среды;
* встроенное введение зависимостей;
* упрощенный высокопроизводительный модульный конвейер HTTP- запросов;
* возможность размещения в IIS, Nginx, Apache, Docker или в собственном процессе;
* параллельное управление версиями приложения, ориентированное на .NET Core;
* инструментарий, упрощающий процесс современной веб-разработки;
* возможность сборки и запуска в ОС Windows, macOS и Linux;
* открытый исходный код и ориентация на сообщество [10].

ASP.NET Core поставляется полностью в виде пакетов NuGet. Использование пакетов NuGet позволяет оптимизировать приложения для включения только необходимых зависимостей. На самом деле для приложений ASP.NET Core 2.x, ориентированных на .NET Core, требуется только один пакет NuGet. За счет небольшого размера контактной зоны приложения доступны такие преимущества, как более высокий уровень безопасности, минимальное обслуживание и улучшенная производительность [10].

Для написания данной системы был выбран язык программирования C#.

C# – это изящный объектно-ориентированный язык со строгой типизацией, позволяющий разработчикам создавать различные безопасные и надежные приложения, работающие на платформе .NET Framework. C# можно использовать для создания клиентских приложений Windows, XML-веб-служб, распределенных компонентов, приложений клиент-сервер, приложений баз данных и т. д. Visual C# предоставляет развитый редактор кода, удобные конструкторы пользовательского интерфейса, интегрированный отладчик и многие другие средства, которые упрощают разработку приложений на языке C# для платформы .NET Framework.

Помимо этого, C# предлагает ряд инновационных языковых конструкций, упрощающих разработку программных компонентов. Инкапсулированные сигнатуры методов, именуемые делегатами, которые позволяют реализовать типобезопасные уведомления о событиях. Свойства, выполняющие функцию акцессоров для закрытых переменных-членов. Атрибуты, предоставляющие декларативные метаданные о типах во время выполнения. Внутри строчные комментарии для XML-документации. LINQ для создания запросов к различным источникам данных [10].

Так же при разработке данного программного средства были задействованы: HTML, CSS, Bootstrap.

HTML – язык разметки (маркировки) гипертекста. HTML дает возможность производить переход от одной части текста к другой, и, что замечательно, эти части могут храниться на совершенно разных компьютерах. Он создан специально для разметки веб-страниц. Именно язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе.

CSS – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Bootstrаp – это инструментарий с открытым исходным кодом для разработки с помощью HTML (HyperText Mаrkup Lаnguаge), CSS (Cаscаding Style Sheets), JS (JаvаScript). Он включает в себя множество разных компонентов для веб-сайта: типографику, веб-формы, кнопки, блоки навигации и другие. Основные преимущества Bootstrаp:

* адаптивность – дизайн сайта будет корректно отображаться на экранах устройств разных размеров вне зависимости от диагонали;
* кроссбраузерность – одинаковое отображение во всех браузерах;
* легкость в использовании;
* понятный код – позволяет писать качественный и понятный код, что облегчает чтение кода для других разработчиков.

Для работы с БД использовался MS SQL Server.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

MS SQL Server характеризуется такими особенностями как:

* производительность. MS SQL Server работает очень быстро;
* надежность и безопасность. MS SQL Server предоставляет шифрование данных;
* простота. С данной СУБД относительно легко работать и вести администрирование.

При создании ПС использовалась технология Razor Pages.

Razor Pages предоставляет технологию, альтернативную системе Model-View-Controller. Razor Pages позволяют создавать страницы с кодом Razor, которые могут обрабатывать запросы. В некоторой степени эта функциональности напоминает работу веб-форм, которые представляли страницу с расширением aspx и имели файл логики на C#, связанный с данной страницей. В этом плане Razor Pages представляют альтернативу стандартной модели MVC для построения приложения.

## 3.2 Разработка структурной схемы программного средства

В ходе моделирования было определено, что для разработки структурной схемы программного средства будет использоваться архитектура MVC.

MVC представляет собой стандартный шаблон разработки, знакомый многим специалистам. Некоторые типы веб-приложений имеют преимущества при создании на платформе MVC. Для других может быть целесообразно использование традиционной схемы приложения ASP.NET, основанной на веб-формах и обратной передаче. В некоторых случаях возможно сочетание двух подходов: применение одной схемы не исключает использования другой [8].

В состав платформы MVC входят следующие компоненты.

Модели являются частями приложения, реализующими логику для домена данных приложения. Объекты моделей часто получают и сохраняют состояние модели в базе данных. Например, объект Product может получать информацию из базы данных, работать с ней, а затем записывать обновленные данные в таблицу Products базы данных SQL Server [8].

Представления служат для отображения пользовательского интерфейса приложения. Пользовательский интерфейс обычно создается на основе данных модели. Примером может служить представление для редактирования таблицы Products, которое содержит текстовые поля, раскрывающиеся списки и флажки, значения которых основаны на текущем состоянии объекта Product [8].

Контроллеры осуществляют взаимодействие с пользователем, работу с моделью, а также выбор представления, отображающего пользовательский интерфейс. В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных [8].

На рисунке 3.1 показана архитектура MVC приложения.

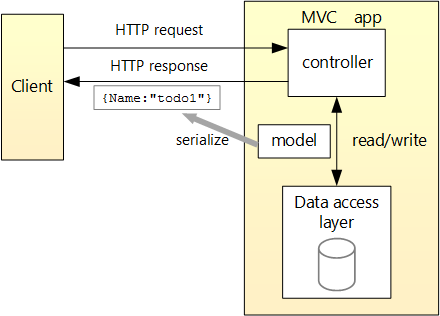


Рисунок 3.1 – архитектура MVC приложения

## 3.3 Разработка структуры классов

Диаграмма классов (class diagram) ‒ основной способ описания структуры системы. На диаграмме классов применяется один основной тип сущностей: классы (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений:

* ассоциация между классами;
* обобщение между классами;
* зависимости (различных типов) между классами и между классами и интерфейсами.

На рисунке 3.2 представлена диаграмма классов уровня контроллера.

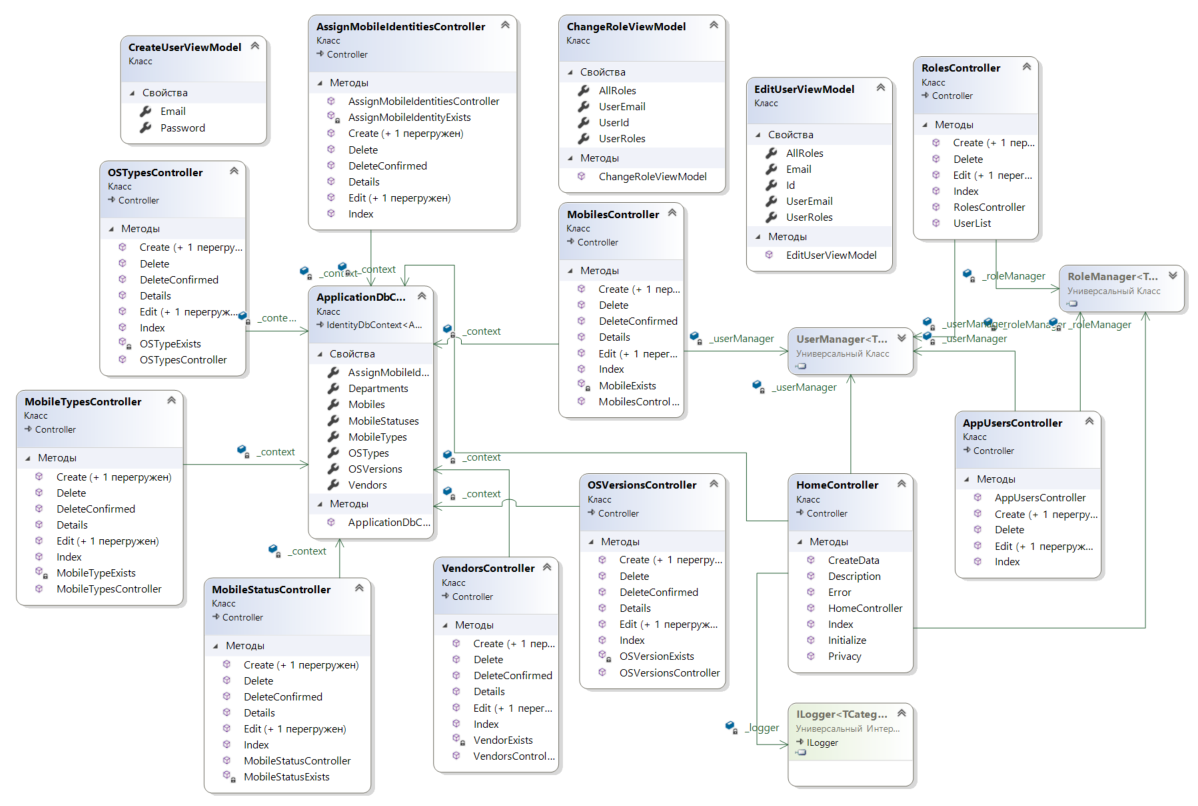


Рисунок 3.2 – Диаграмма классов уровня контроллера

Диаграмма классов уровня модели данных представлена на рисунке 3.3.

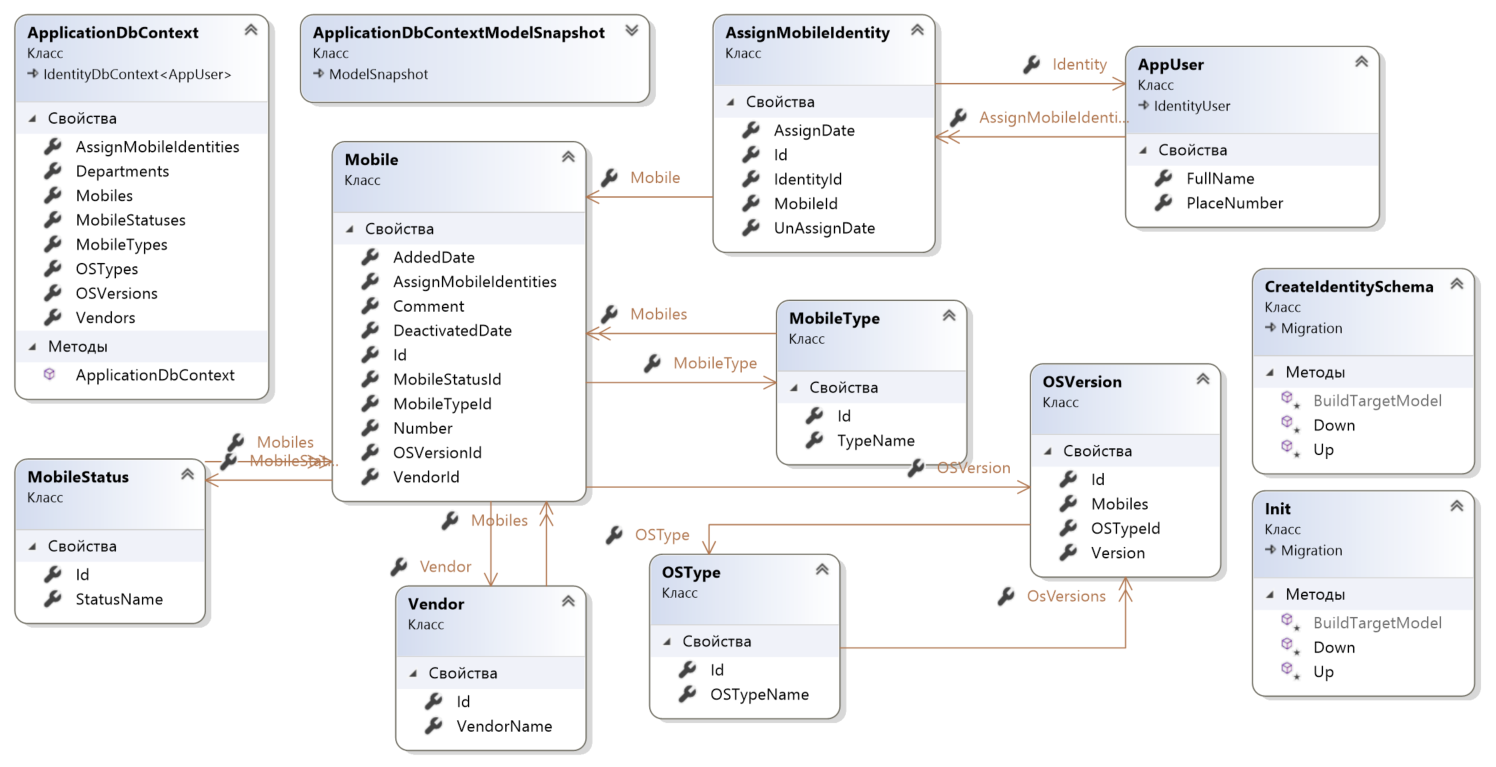


Рисунок 3.3 – Диаграмма классов уровня модели данных

## 3.4 Разработка физической модели данных

На рисунке 3.4 представлена физическая модель базы данных, построенная на основе информационной модели базы данных. Данная модель жестко связана с конкретной системой управления базами данных. В данной модели представлены реальные типы данных, ограничения, реальные названия таблиц базы данных.

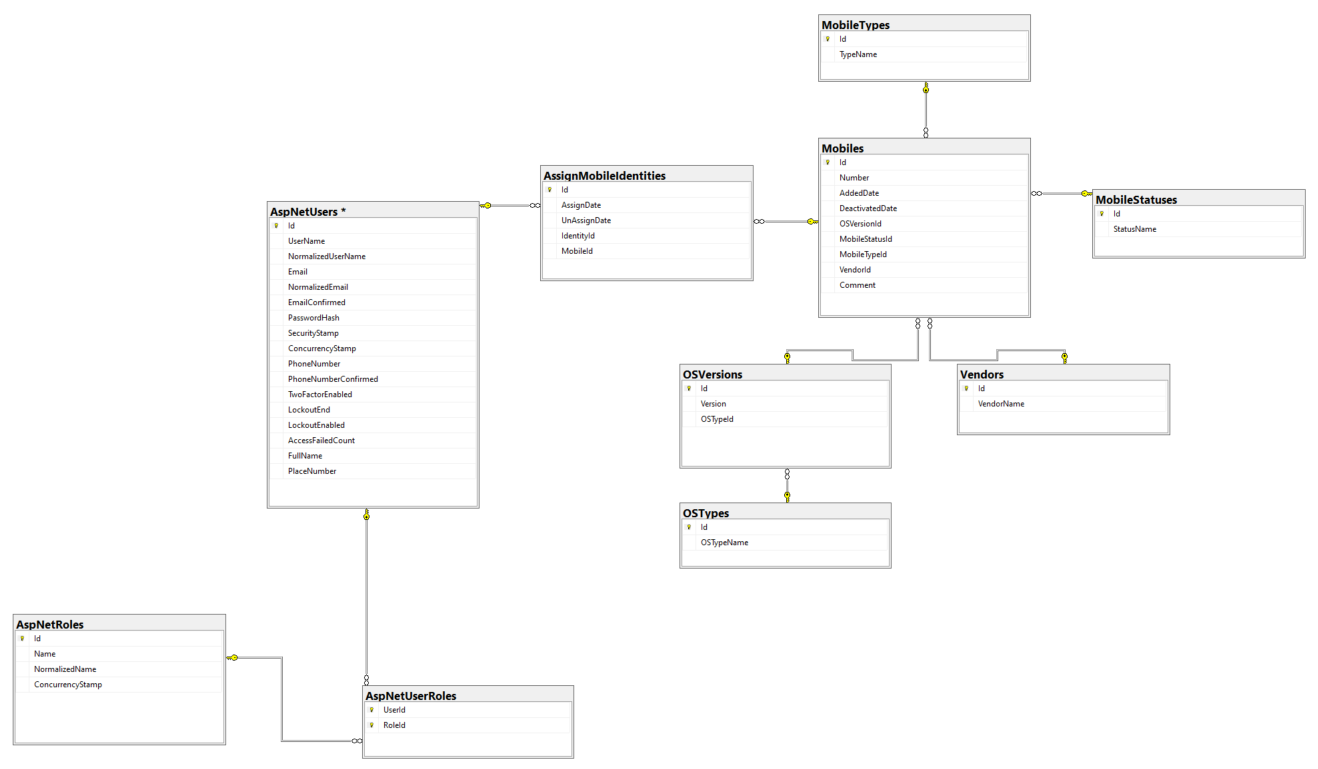


Рисунок 3.4 – Физическая модель БД

Формализованное описание объектов представлено в таблицах 3.1–3.10 Описание таблицы «AspNetUsers» представлено в таблице 3.1.

Таблица 3.1 – Описание таблицы «AspNetUsers»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| 1 | 2 | 3 |
| Id | nvarchar(450) | Идентификатор сотрудника |
| UserName | nvarchar(256) | Имя сотрудника |
| NormalizedUserName | nvarchar(256) | Нормализованное имя сотрудника |
| Email | nvarchar(256) | Почтовый адрес сотрудника |
| NormalizedEmail | nvarchar(256) | Нормализованный почтовый адрес сотрудника |
| EmailConfirmed | bit | Флаг подтверждения почтового адреса сотрудника |
| PasswordHash | nvarchar(MAX) | Хеш пароля сотрудника |
| SecurityStamp | nvarchar(MAX) | Метка безопасности |
| ConcurrencyStamp | nvarchar(MAX) | Метка параллелизма |
| PhoneNumber | nvarchar(MAX) | Мобильный номер сотрудника |
| 1 | 2 | 3 |
| PhoneNumberConfirmed | bit | Флаг подтверждения мобильного номера сотрудника |
| TwoFactorEnabled | bit | Флаг двухфакторной аутентификации |
| LockoutEnd | datetimeoffset(7) | Дата окончания запрета авторизации сотрудника |
| LockoutEnabled | bit | Флаг запрета авторизации сотрудника |
| AccessFailedCount | int | Число попыток неудачного входа в систему |
| FullName | nvarchar(MAX) | Полное имя сотрудника |
| PlaceNumber | nvarchar(MAX) | Рабочее место сотрудника |

Описание таблицы «Mobiles» представлено в таблице 3.2.

Таблица 3.2 – Описание таблицы «Mobiles»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор мобильного устройства |
| Number | nvarchar(MAX) | Краткий идентификатор мобильного устройства |
| AddedDate | datetime2(7) | Дата покупки мобильного устройства |
| DeactivatedDate | datetime2(7) | Дата деактивации мобильного устройства |
| OSVersionId | int | Внешний ключ идентификатора версии операционной системы |
| MobileStatusId | int | Внешний ключ идентификатора статуса мобильного устройства |
| MobileTypeId | int | Внешний ключ типа мобильного устройства |
| VendorId | int | Внешний ключ вендора мобильного устройства |
| Comment | nvarchar(MAX) | Комментарий |

Описание таблицы «AspNetRoles» представлено в таблице 3.3.

Таблица 3.3 – Описание таблицы «AspNetRoles»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | nvarchar(450) | Идентификатор роли сотрудника |
| Name | nvarchar(256) | Наименование роли сотрудника |
| NormalizedName | nvarchar(256) | Нормализованное наименование роли сотрудника |
| ConcurrencyStamp | nvarchar(MAX) | Метка параллелизма |

Описание таблицы «AspNetUserRoles» представлено в таблице 3.4.

Таблица 3.4 – Описание таблицы «AspNetUserRoles»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| UserId | nvarchar(450) | Идентификатор сотрудника |
| RoleId | nvarchar(450) | Идентификатор роли сотрудника |

Описание таблицы «Vendors» представлено в таблице 3.5.

Таблица 3.5 – Описание таблицы «Vendors»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор вендора |
| VendorName | nvarchar(MAX) | Наименование вендора |

Описание таблицы «MobileStatuses» представлено в таблице 3.6.

Таблица 3.6 – Описание таблицы «MobileStatuses»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор статуса мобильного устройства |
| StatusName | nvarchar(MAX) | Наименование статуса мобильного устройства |

Описание таблицы «MobileTypes» представлено в таблице 3.7.

Таблица 3.7 – Описание таблицы «MobileTypes»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор типа мобильного устройства |
| TypeName | nvarchar(MAX) | Наименование типа мобильного устройства |

Описание таблицы «OSTypes» представлено в таблице 3.8.

Таблица 3.8 – Описание таблицы «OSTypes»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор типа операционной системы |
| OSTypeName | nvarchar(MAX) | Наименование типа операционной системы |

Описание таблицы «OSVersions» представлено в таблице 3.9.

Таблица 3.9 – Описание таблицы «OSVersions»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор версии операционной системы |
| Version | nvarchar(MAX) | Версия операционной системы |
| OSTypeId | int | Внешний ключ идентификатора типа операционной системы |

Описание таблицы «AssignMobileIdentities» представлено в таблице 3.10.

Таблица 3.10 – Описание таблицы «AssignMobileIdentities»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | int | Идентификатор |
| AssignDate | datetime2(7) | Дата назначения мобильного устройства на сотрудника |
| UnAssignDate | datetime2(7) | Дата снятия мобильного устройства с сотрудника |
| IdentityId | int | Внешний ключ идентификатора сотрудника |
| MobileId | int | Внешний ключ идентификатора мобильного устройства |

## 3.5 Проектирование алгоритмов работы программного средства

Для определения логики написания программы были спроектированы следующие алгоритмы работы программного средства.

**3.5.1** Алгоритм проверки уровня доступа на стороне сервера

Схема работы алгоритма регистрации на стороне сервера представлена на рисунке 3.5. Данный алгоритм демонстрирует как происходит проверка уровня доступа пользователя при отправке запроса на сторону сервера.

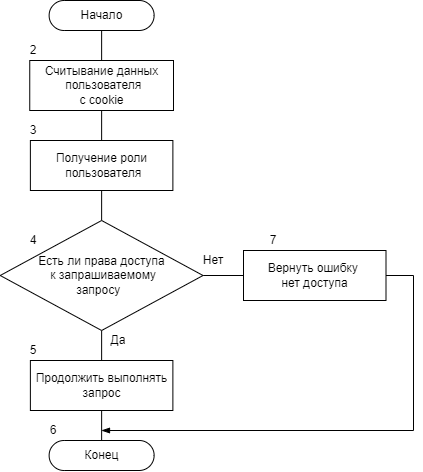


Рисунок 3.5 – Блок-схема алгоритма проверки уровня доступа на стороне сервера

**3.5.2** Алгоритм добавления сотрудника

**3.5.3** Алгоритм добавления устройства

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства – процесс исследования, испытания ПС с целью получения информации о качестве продукта.

Существующие на сегодня методы тестирования ПС не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПС.

Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода. То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла ПС.

Существуют следующие виды тестирования:

* функциональные;
* нефункциональные;
* связанные с изменениями;
* модульное.

Функциональное тестирование – это тестирование программное средства в целях проверки реализуемости функциональных требований, то есть способности программного средства в определённых условиях решать задачи, нужные пользователям.

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного средства, которые могут быть измерены различными величинами.

Модульное тестирование – процесс в программировании, позволяющий проверить на корректность единицы исходного кода, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

В рамках разработки дипломного проекта функциональное тестирование реализовано с помощью набора тест-кейсов.

Тест-кейс – это профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату.

Тест-кейс №1 – Проверка работы сайта на разных ОС. Тест кейс представлен в таблице 4.1.

Таблица 4.1 – Тест-кейс №1

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Проверка работы сайта на разных ОС | 1 Открыть веб-приложение  2 Проверить отображение приложения в браузере Opera версии 81.x  3 Проверить отображение приложения в браузере Google Chrome версии 95.x | Отображается все корректно | Отображается все корректно |

Тест-кейс №2 – Проверка отображения главной страницы. Тест кейс представлен в таблице 4.2.

Таблица 4.2 – Тест-кейс №2

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Проверка отображения главной страницы | 1 Открыть веб-приложение  2 Открыть главную страницу | Отображается главная страница с описанием приложения | Отображается главная страница с описанием приложения |

Тест-кейс №3 – Регистрация пользователя: успешное создание пользователя. Тест кейс представлен в таблице 4.3.

Таблица 4.3 – Тест-кейс №3

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Регистрация пользователя: успешное создание пользователя | 1 Открыть веб-приложение  2 Нажать на панели навигации «Регистрация»  3 Заполнить все поля корректно  4 Нажать кнопку «Регистрация» | Пользователь успешно зарегистрирован | Пользователь успешно зарегистрирован |

Тест-кейс №4 – Регистрации пользователя: ошибка при регистрации. Тест кейс представлен в таблице 4.4.

Таблица 4.4 – Тест-кейс №4

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Регистрации пользователя: ошибка при регистрации | 1 Открыть веб-приложение  2 Нажать на панели навигации «Регистрация»  3 Заполнить все поля корректно и написать существующие имя пользователя  4 Нажать кнопку «Регистрация» | Получить ошибку, что пользователь с таким именем существует | Получить ошибку, что пользователь с таким именем существует |

Тест-кейс №5 – Вход в систему. Тест кейс представлен в таблице 4.5.

Таблица 4.5 – Тест-кейс №5

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Вход в систему | 1 Открыть веб-приложение  2 Нажать на панели навигации «Вход»  3 Заполнить все поля корректно  4 Нажать кнопку «Вход» | Пользователь успешно вошел в систему | Пользователь успешно вошел в систему |

Тест-кейс №6 – Открыть форму создания заметки. Тест кейс представлен в таблице 4.6.

Таблица 4.6 – Тест-кейс №6

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Открыть форму создания заметки | 1 Открыть веб-приложение  2 Войти в систему | Откроется форма создания заметки | Откроется форма создания заметки |
| Продолжение таблицы 4.6 | | | |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
|  | 3 В навигационной панели нажать «Создать заметку» |  |  |

Тест-кейс №7 – Создание заметки. Тест кейс представлен в таблице 4.7.

Таблица 4.7 – Тест-кейс №7

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Создание заметки через форму создания | 1 Открыть веб-приложение  2 Войти в систему  3 В навигационной панели нажать «Создать заметку»  4 Ввести все корректные данные  5 Нажать на кнопку «Создать» | Заметка успешно создалась | Заметка успешно создалась |

Тест-кейс №8 – Обновить заметку через форму создания. Тест кейс представлен в таблице 4.8.

Таблица 4.8 – Тест-кейс №8

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Обновить заметку через форму создания | 1 Открыть веб-приложение  2 Войти в систему  3 В навигационной панели нажать «Создать заметку»  4 Ввести все корректные данные  5 Нажать на кнопку «Создать»  6 Обновить данные  7 Нажать на кнопку «Обновить» | Заметка успешно обновилась | Заметка успешно обновилась |

Тест-кейс №9 – Просмотр списка заметок и поиск. Тест кейс представлен в таблице 4.9.

Таблица 4.9 – Тест-кейс №9

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| Просмотр списка заметок и поиск | 1 Открыть веб-приложение  2 Войти в систему  3 В навигационной панели нажать «Заметки»  4 Ввести в поисковую строку существующее значение тега или текста заметки  5 Нажать на кнопку «Поиска»  6 Нажать на карточку заметки | Страница с результатом отображается и при нажатии на заметку заметка подгружается | Страница с результатом отображается и при нажатии на заметку заметка подгружается |

Как видно из результатов функциональное тестироваие тестирования, приложение работает исправно.

Было проведено модульное тестирование. Результаты первичного тестирования представлена на рисунке 4.1. После запуска первичного модульного тестирования были выявлены следующие ошибки:

* + строка подключения к базе данных была не корректна;
  + ошибки в передачи данных между контроллером и представлением;
  + ошибки в формировании LINQ запросов;
  + не правильная передача данных для создания тестов;
  + отсутствует валидация на модели;
  + проблемы с доступом к контроллерам;
  + синтаксические ошибки в коде.

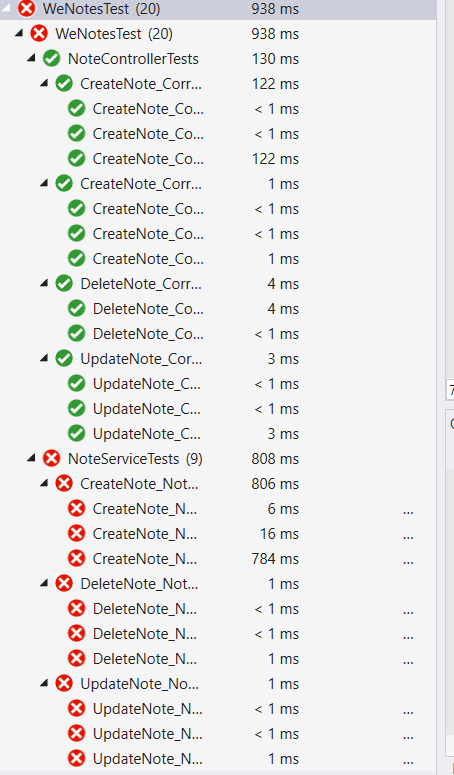


Рисунок 4.1 – Первичный результат модульного тестирования

После получения результатов, были проведены исправления кода и выполнено повторное тестирование после проведения изменений. Результат итогового выполнения модульного тестирования представлен на рисунке 4.2.

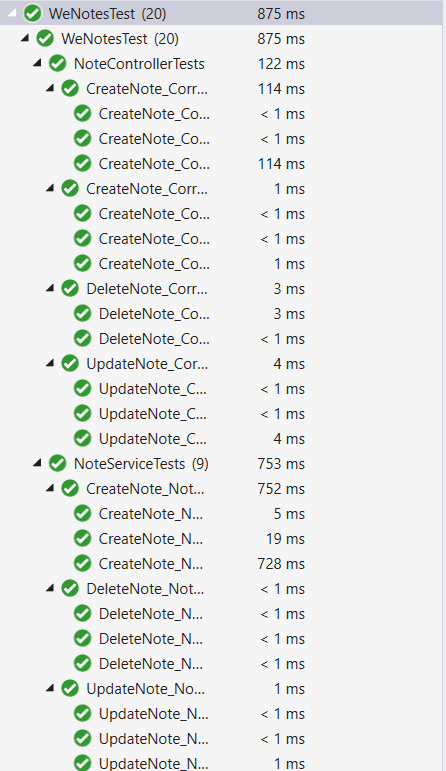


Рисунок 4.2 – Результат итогового выполнения модульного тестирования

# 5 МЕТОДИКА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

После входа в программное средство открывается главная страница с описанием основных функций приложения. Данная страница доступна для всех типов пользователей. Главная страница приложения представлена на рисунке 5.1.

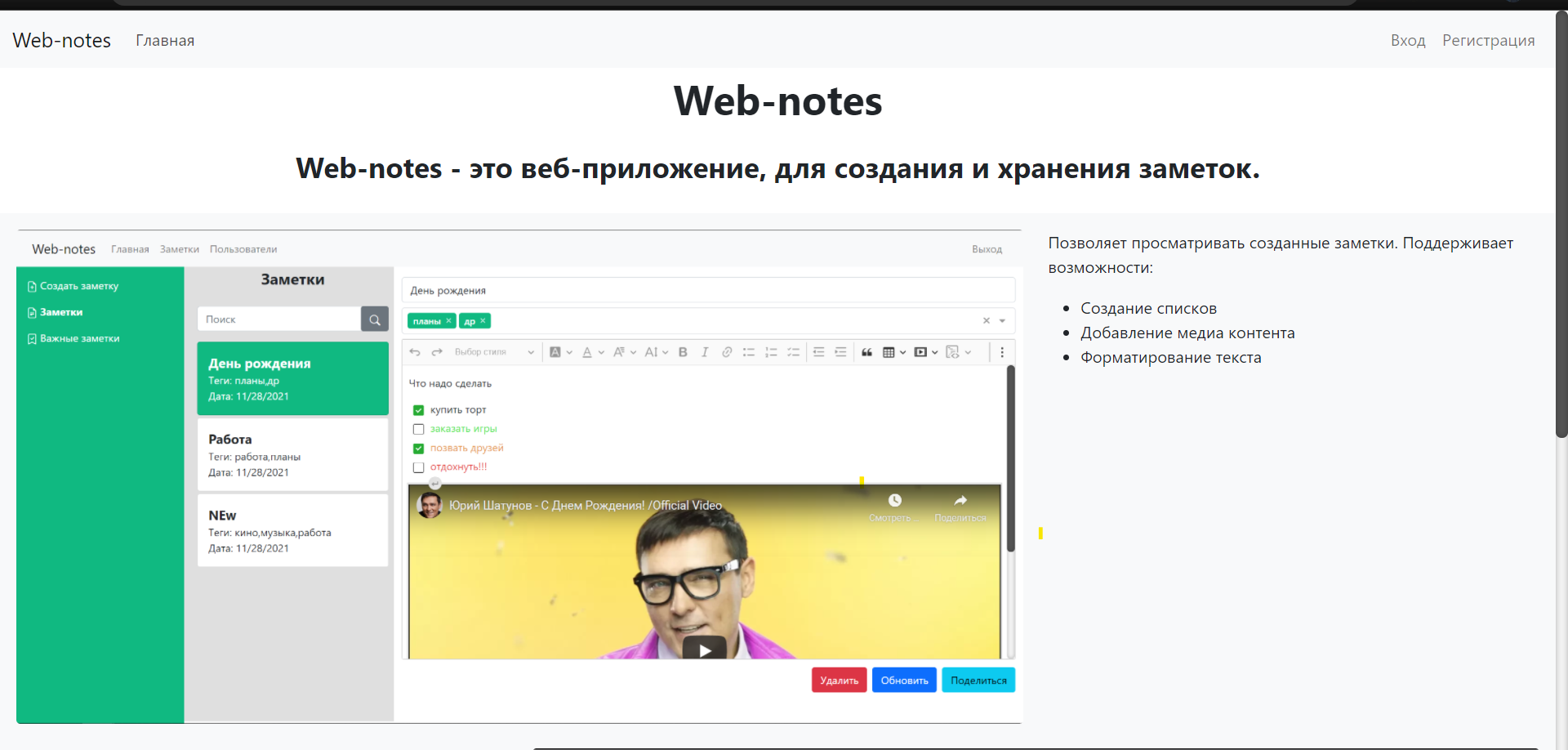


Рисунок 5.1 – Главная страница веб-приложения

## 5.1 Регистрация пользователя

Для того чтобы зарегистрироваться в системе требуется на панели навигации нажать на элемент управления «Регистрация». После нажатия открывается форма регистрации, где пользователю нужно ввести все обязательные поля. Данная функция доступна только не авторизованным пользователям. Форма регистрации представлена на рисунке 5.2.

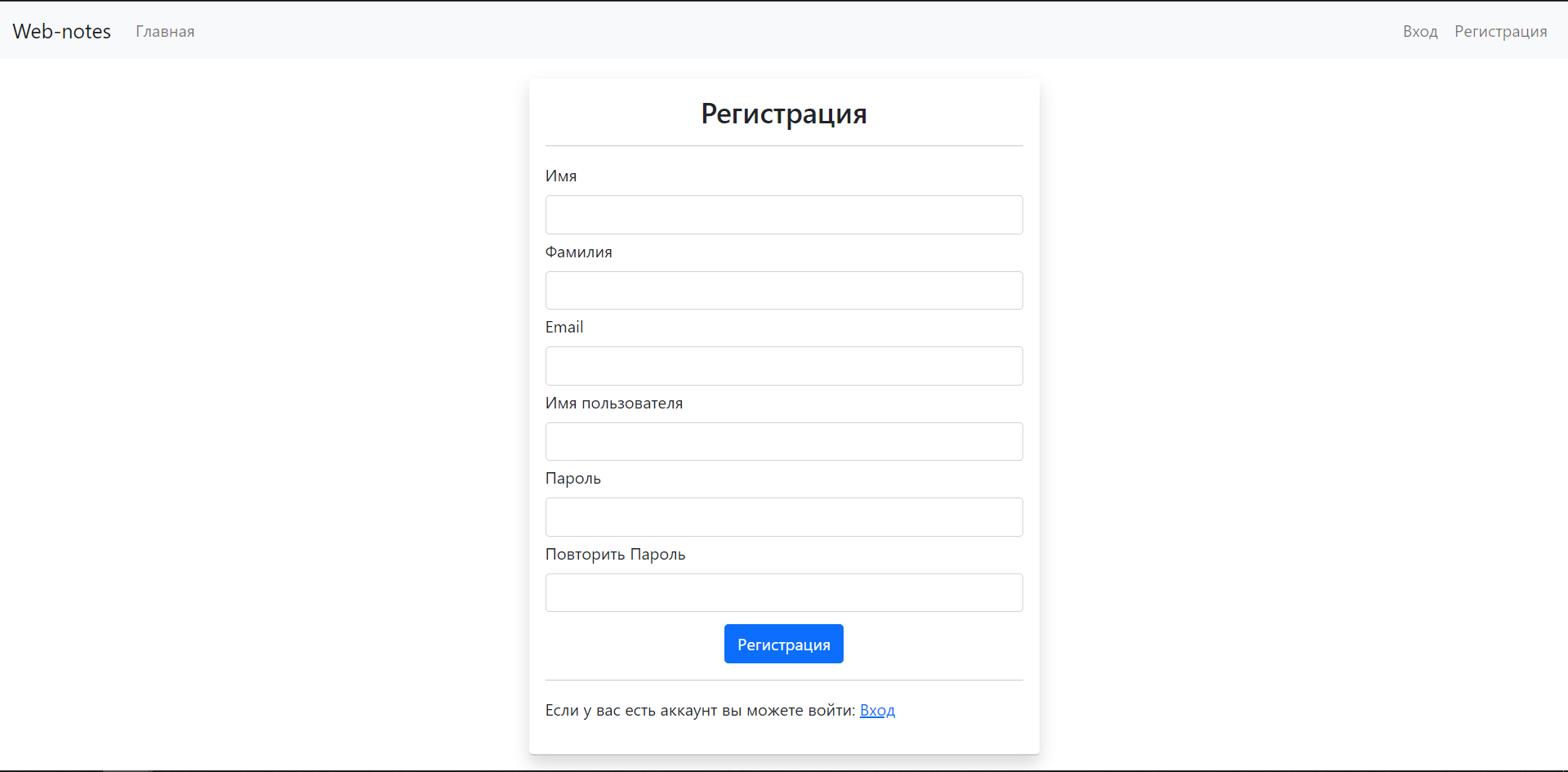


Рисунок 5.2 – Форма регистрации веб-приложения

Если пользователь заполнил не все обязательные поля в форме регистрации и нажал на кнопку «Регистрация», то увидит сообщение о том, что требуется заполнить пропущенное поле. Пример заполнения формы и с указанием пропущенного поля представлен на рисунке 5.3.

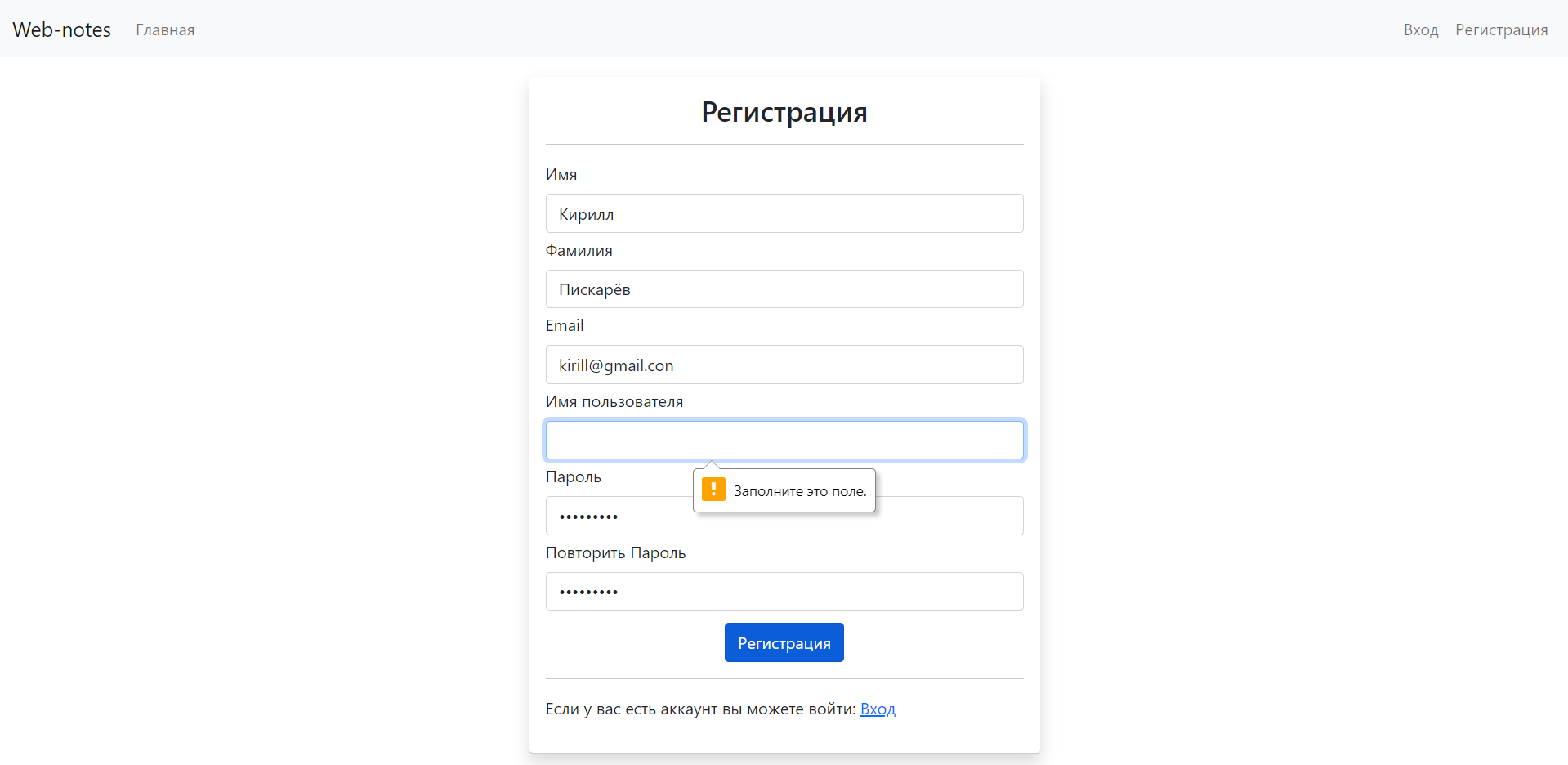


Рисунок 5.3 – Форма регистрации веб-приложения с указанием ошибки

## 5.2 Авторизация пользователя

Для того чтобы пользователь авторизовался, требуется наличие учетной записи в системе. Если пользователь имеет учетную запись требуется в панели навигации нажать на элемент управления «Вход». Форма входа представлена на рисунке 5.4.

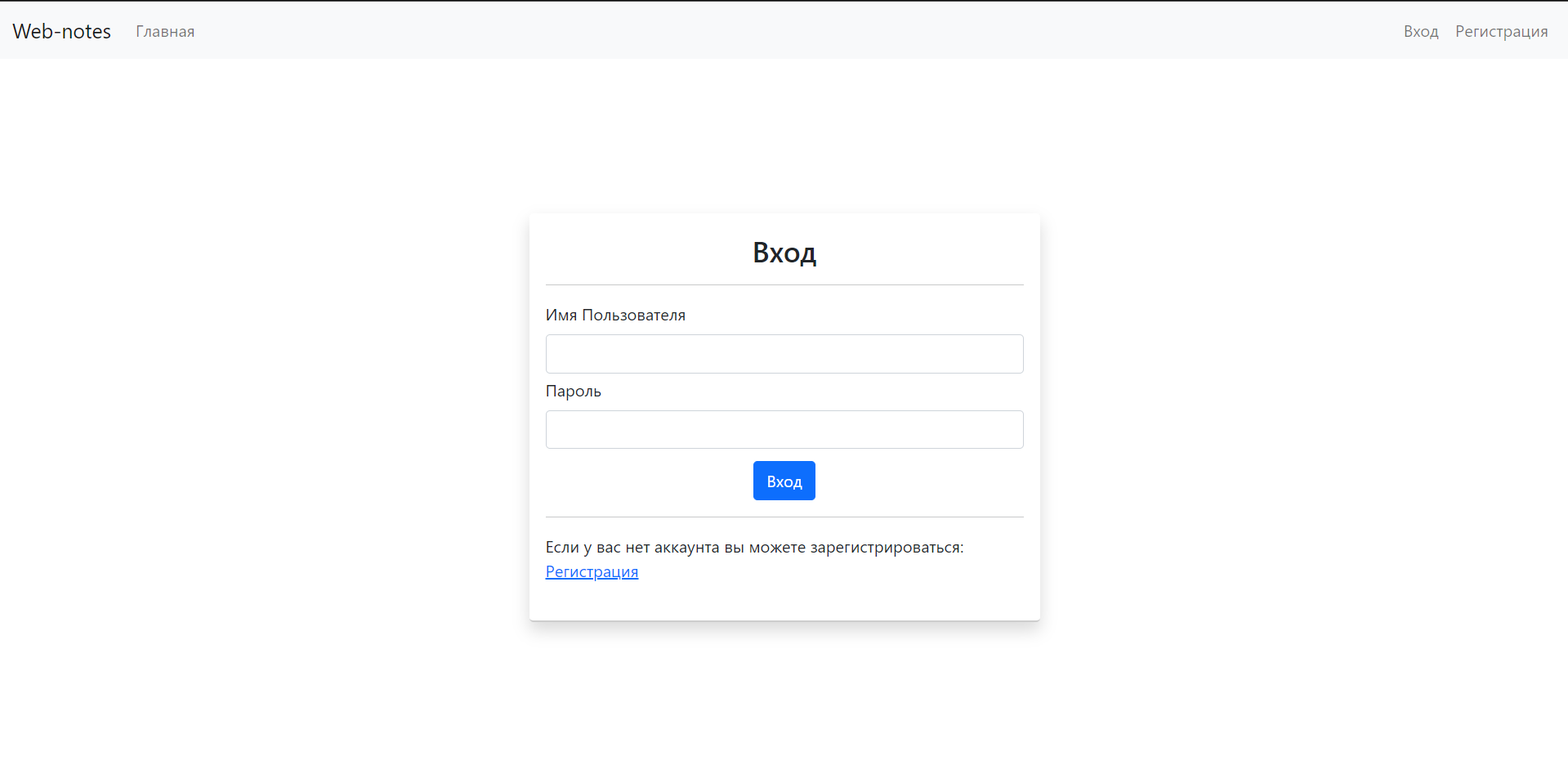


Рисунок 5.4 – Форма входа в веб-приложения

На форме присутствует проверка ввода данных. Если пользователь не введет данные в одно из полей, появится уведомление, что требуется ввести данные. Пример проверки ввода данных представлен на рисунке 5.5.

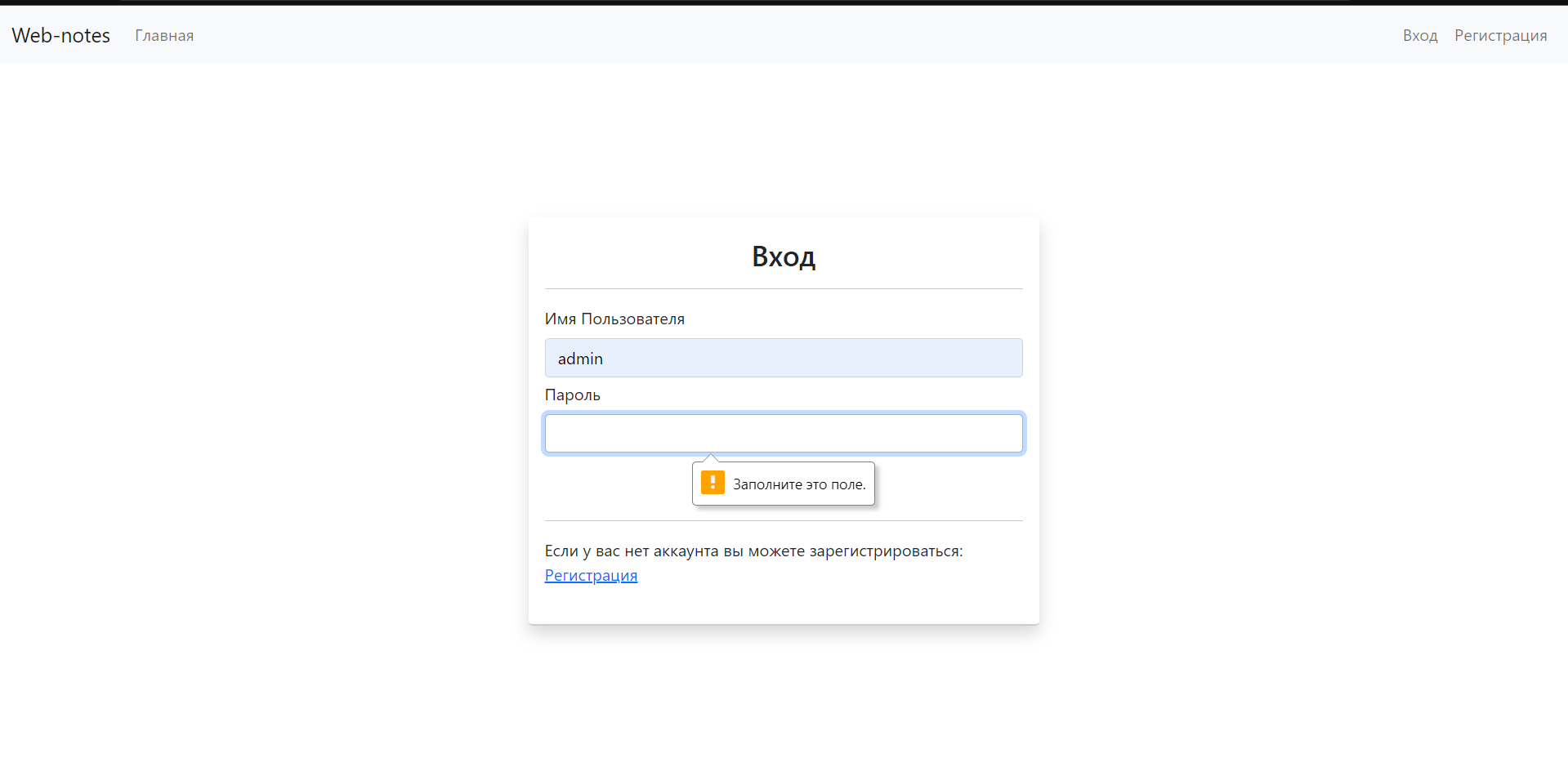


Рисунок 5.5 – Форма входа веб-приложения с указанием ошибки

На форму входа в приложении можно попасть с формы регистрации веб-приложения. Для этого при переходе на форму регистрации требуется внизу формы нажать на ссылку «Вход». Форма с ссылкой представлена на рисунке 5.2.

После успешной авторизации в веб-приложении, пользователя перенаправляет на страницу со списком заметок. После входа навигационная панель меняет свои элементы. Главная страница с измененной навигационной панелью для авторизованного пользователя представлена на рисунке 5.6.

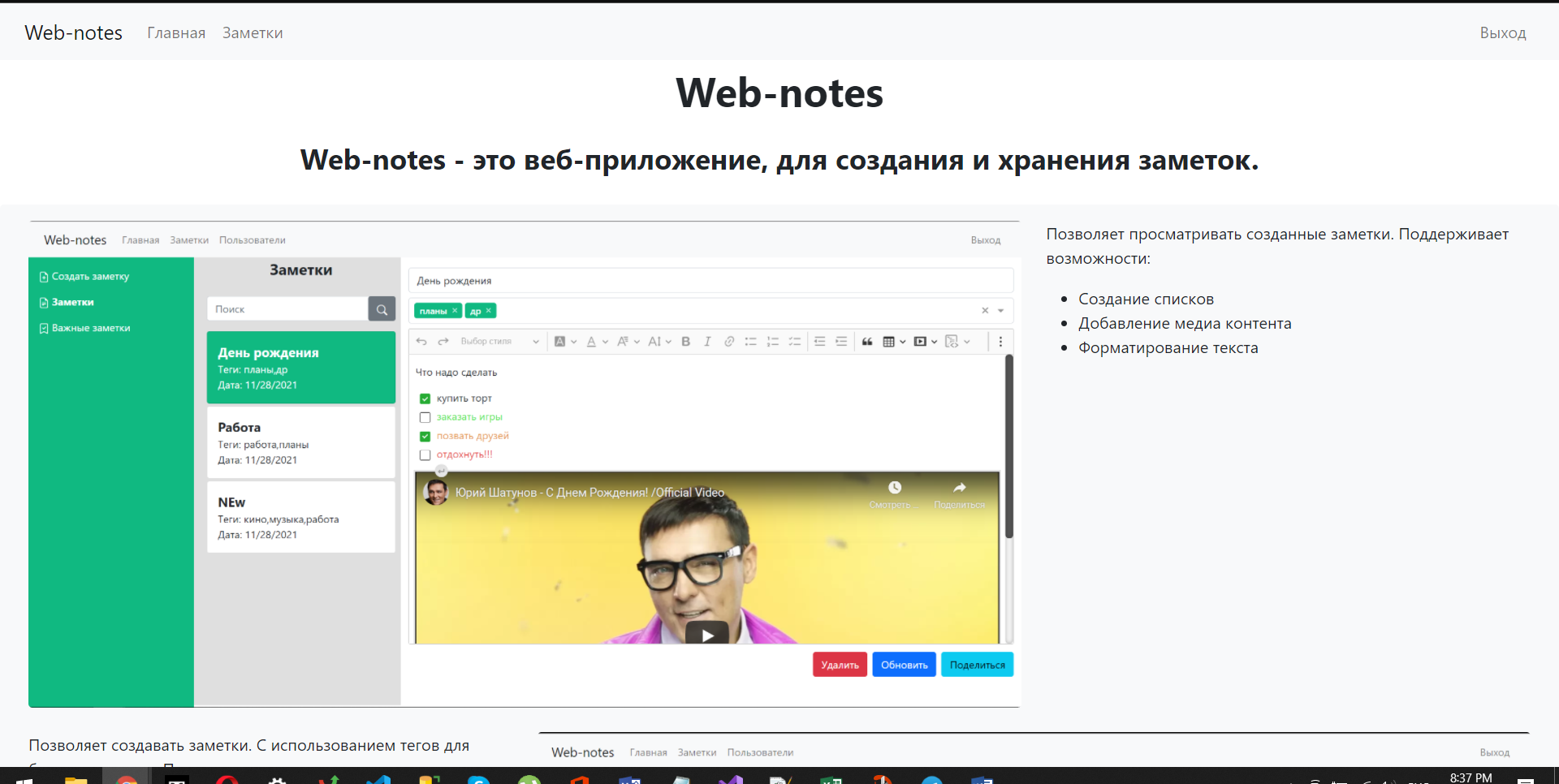


Рисунок 5.6 – Главная страница веб-приложения для авторизованного пользователя

Для выхода из учетной записи требуется нажать на элемент управления «Выход». После нажатия на элемент управления, закончится авторизационная сессия и пользователя перенаправит на главную страницу.

## 5.3 Создание и редактирование заметки

После авторизации в системе пользователю будут доступны возможности для управления заметками. Для создания заметки требуется в навигационной панели нажать на элемент управления «Заметки», после чего откроется страница для управления заметками. На странице управления будут доступны возможности: создания, редактирования, просмотра и поиска заметок.

Для создания заметки в левой части страницы в навигационной панели требуется нажать на элемент «Создать заметку». Форма создания заметки представлена на рисунке 5.7.

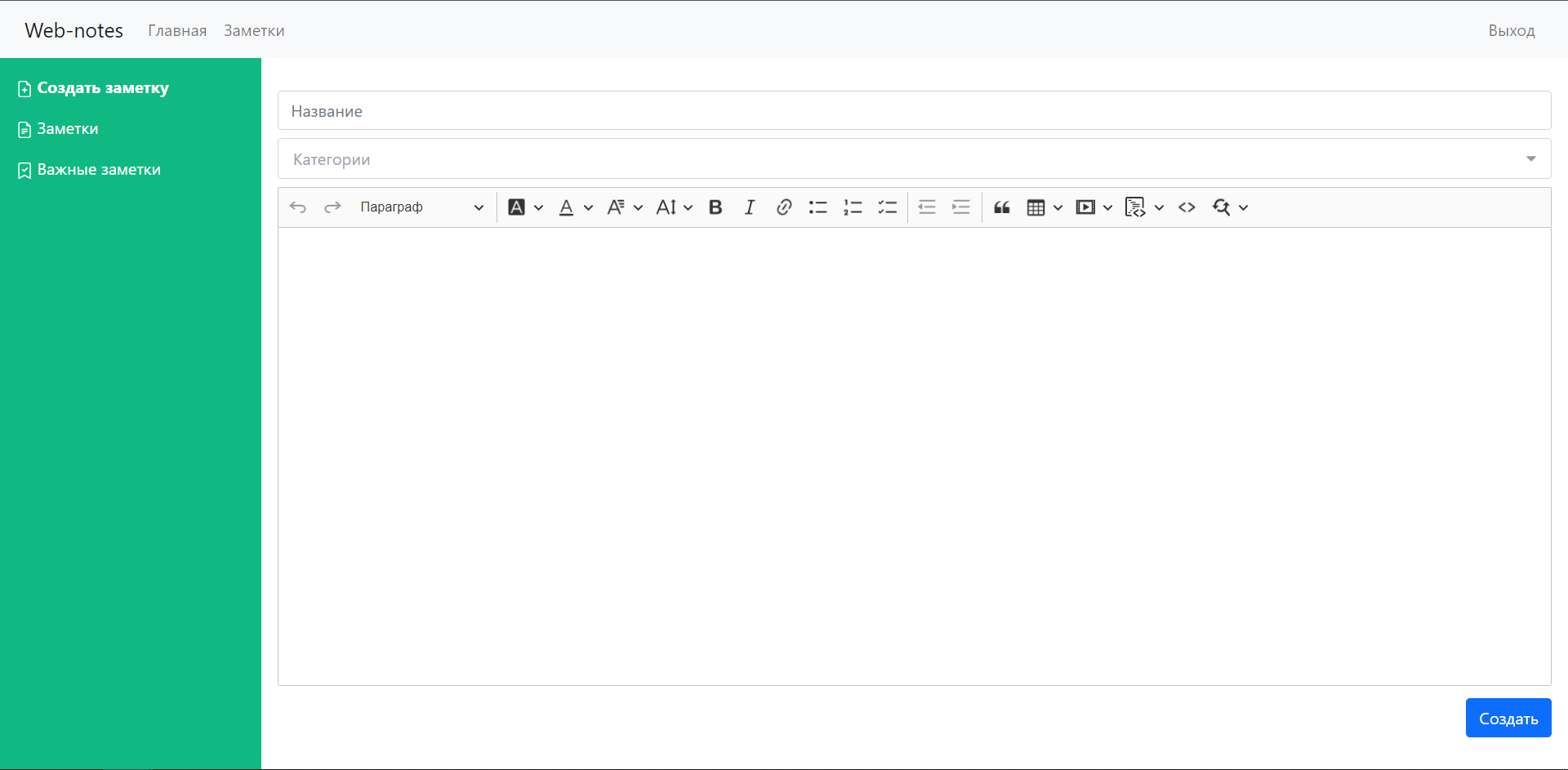


Рисунок 5.7 – Форма создания заметки

На форме расположены следующие элементы страницы: название, категории и элемент для формирования заметки. Название служит для обозначение краткого наименования заметки, для более удобного поиска. Элемент категории представляет собой поле для ввода несколько категорий заметки, используется также для поиска и краткой характеристики типа заметки. Элемент для создания заметки представляет собой многофункциональный редактор для формирования различного типа и содержания заметки.

Категории включают в себя преопределенный базовый список категорий, который можно выбрать из выпадающего списка при нажатии на элемент категории. Также можно создать новую категорию, написав ее и нажав на кнопку «Ввода». Пример заполнения и списка категорий представлен на рисунке 5.8.



Рисунок 5.8 – Указание категорий в заметке

Элемент для формирования заметки имеет следующие возможности:

* написание текста и его форматирование: изменения стиля, изменения цвета фона, цвета текста, семейство шрифта, жирность и курсив;
* создания ссылок;
* формирования списка;
* создания листа задания;
* выравнивание текста;
* цитирования;
* создание таблиц;
* добавление медиа ссылок;
* формирования блока кода;
* поиск по тексту и замена.

Пункт меню для изменения стиля текста представлен на рисунке 5.9. изменение стиля позволяет выбрать тип заголовка или использовать обычный текст.

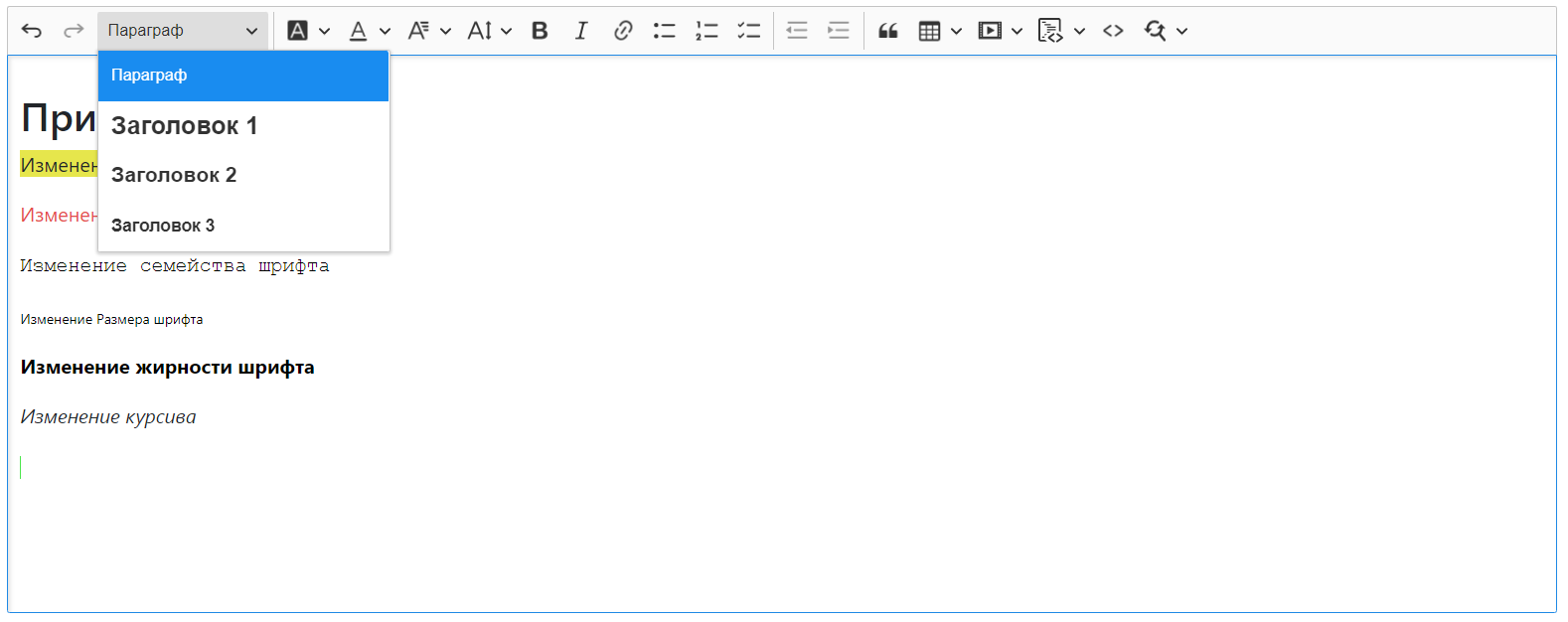


Рисунок 5.9 – Меню для изменения стиля сообщения

Пункт меню для изменения цвета фона представлен на рисунке 5.10. Позволяет задать цвет фона выделенного текста.

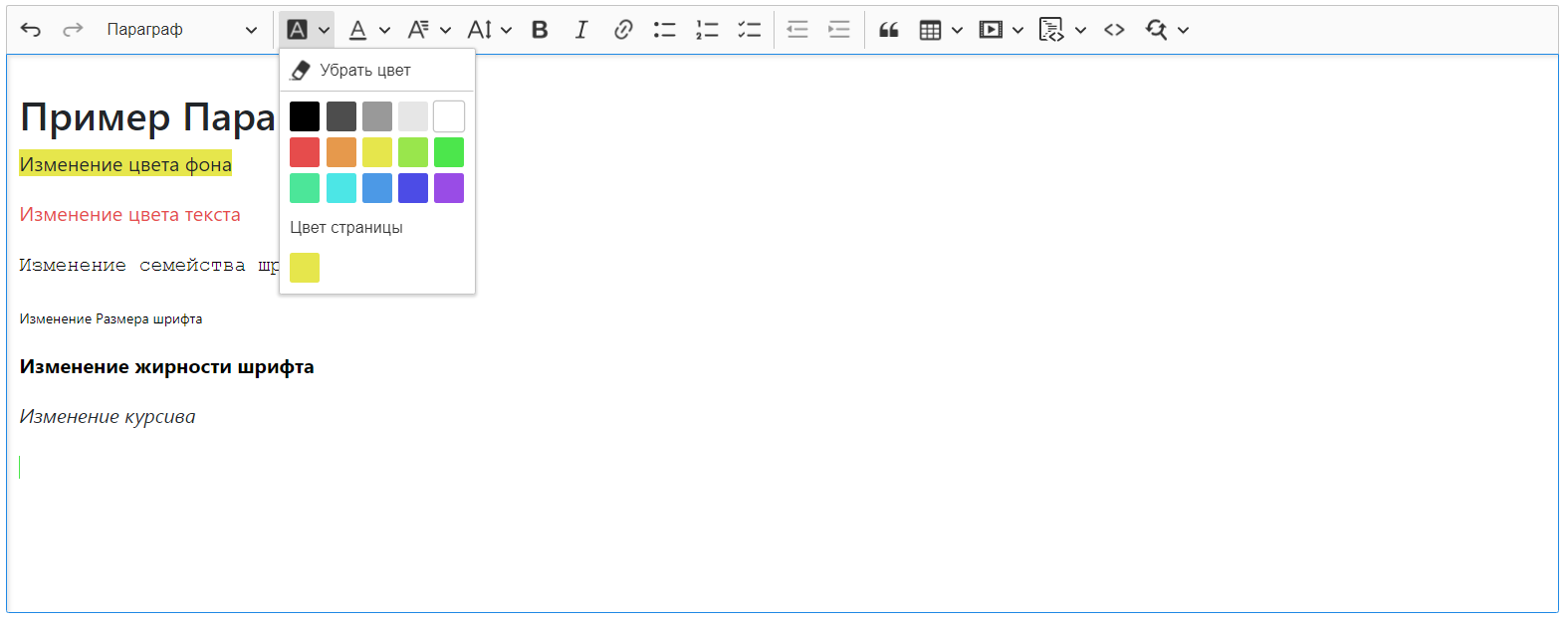


Рисунок 5.10 – Меню для изменения цвета фона

Для изменения цвета текста используется такого же типа пункт меню. Пункт меню для изменения семейства шрифта представлен на рисунке 5.11.

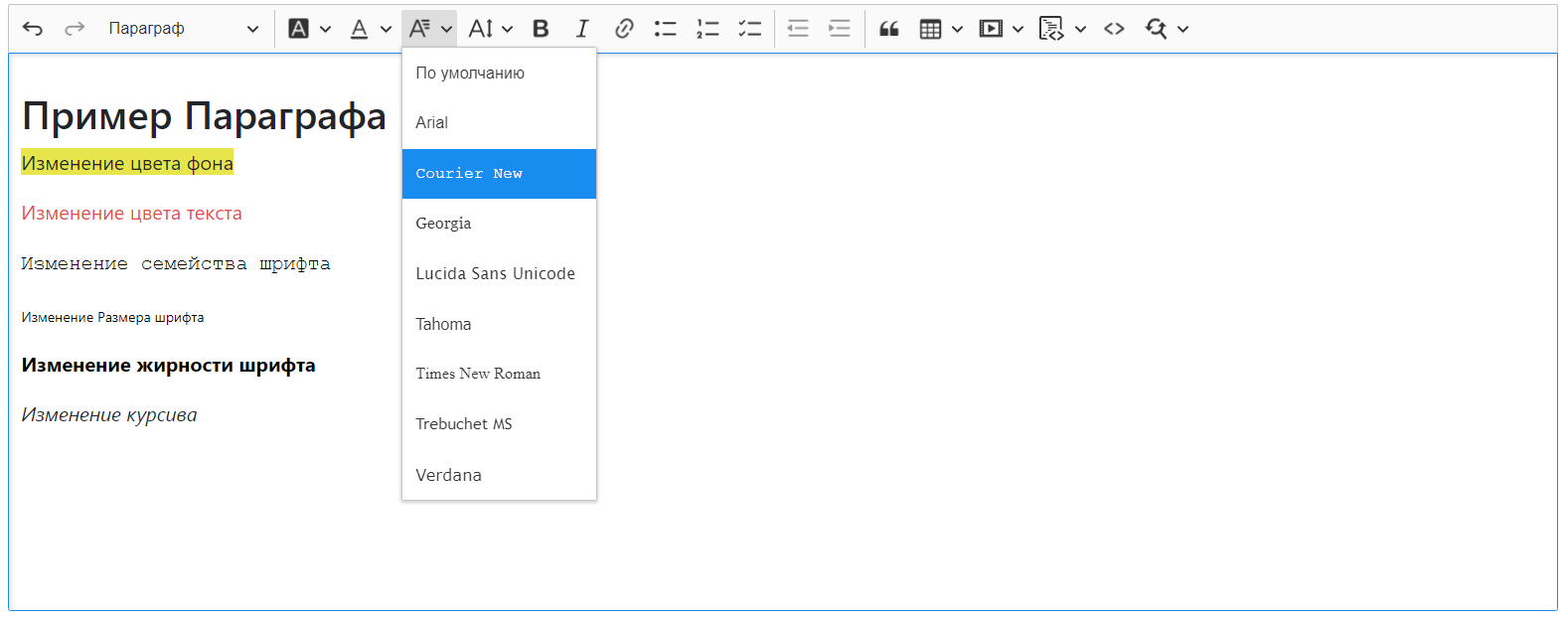


Рисунок 5.11 – Меню для изменения семейства шрифта

Пункт меню для изменения размера шрифта представлен на рисунке 5.12.

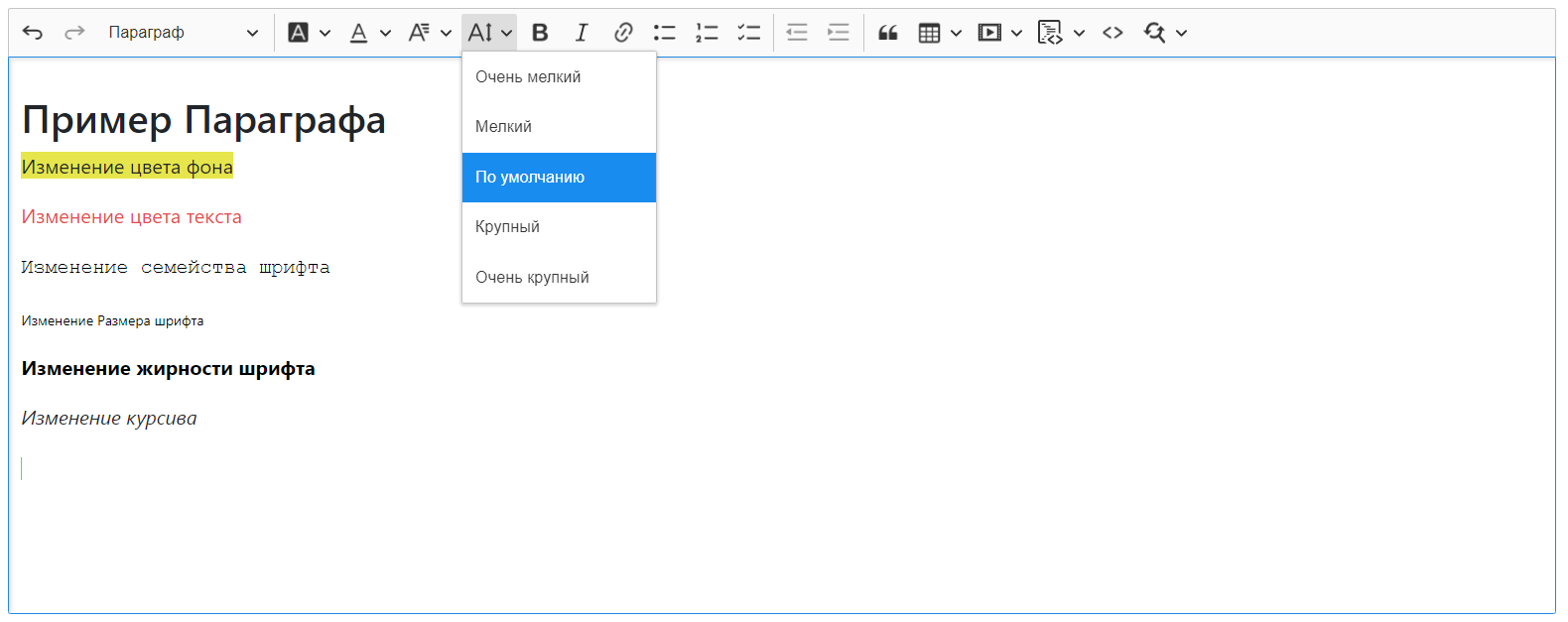


Рисунок 5.12 – Меню для изменения размера шрифта

Пример с использованием перечисленных возможностей форматирования текста представлен на рисунке 5.13.

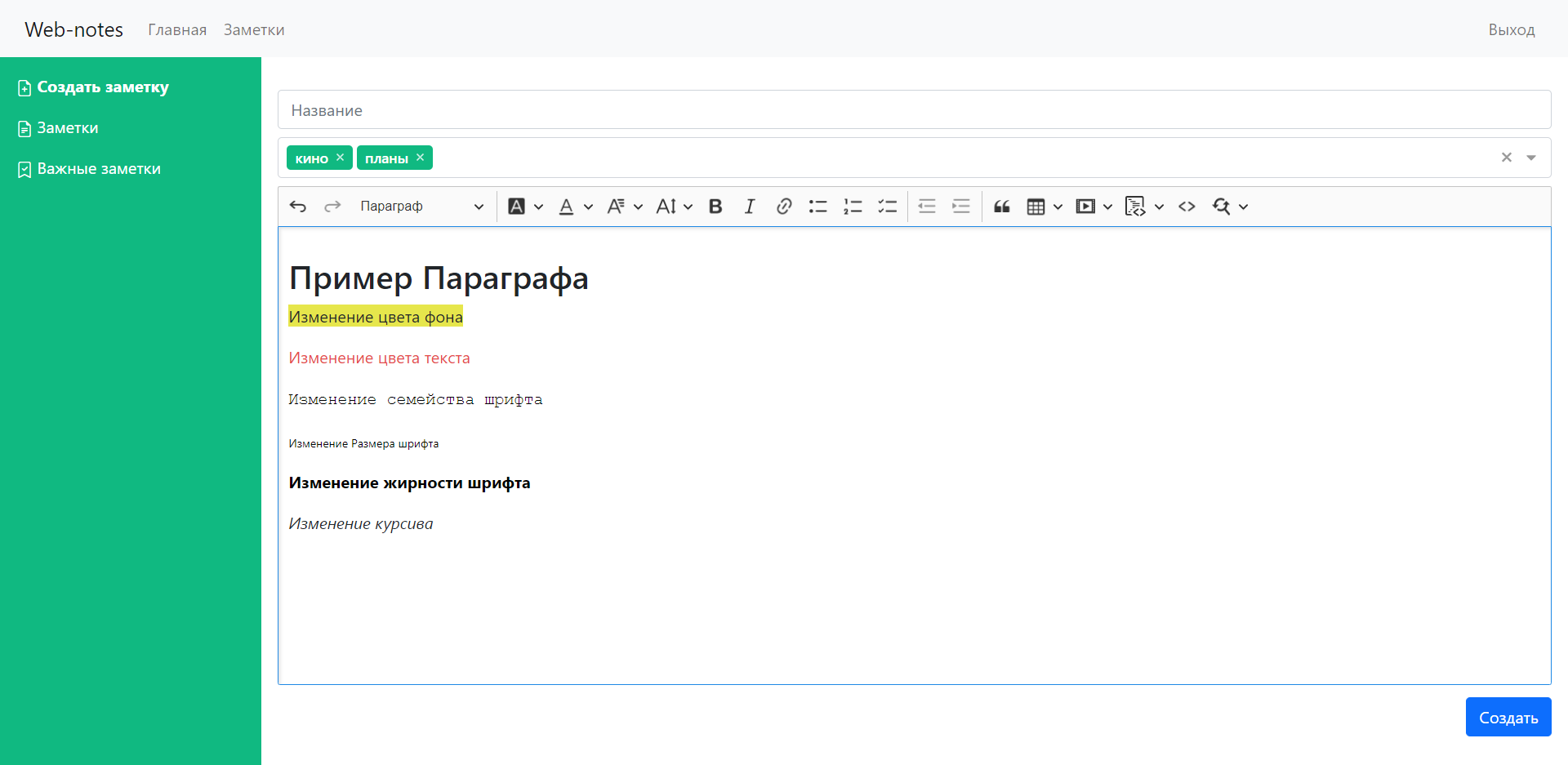


Рисунок 5.13 – Пример форматирования текста

Для создания ссылок есть пункт меню «Ссылка». Результат формирования и пример формирования ссылки представлен на рисунке 5.14.

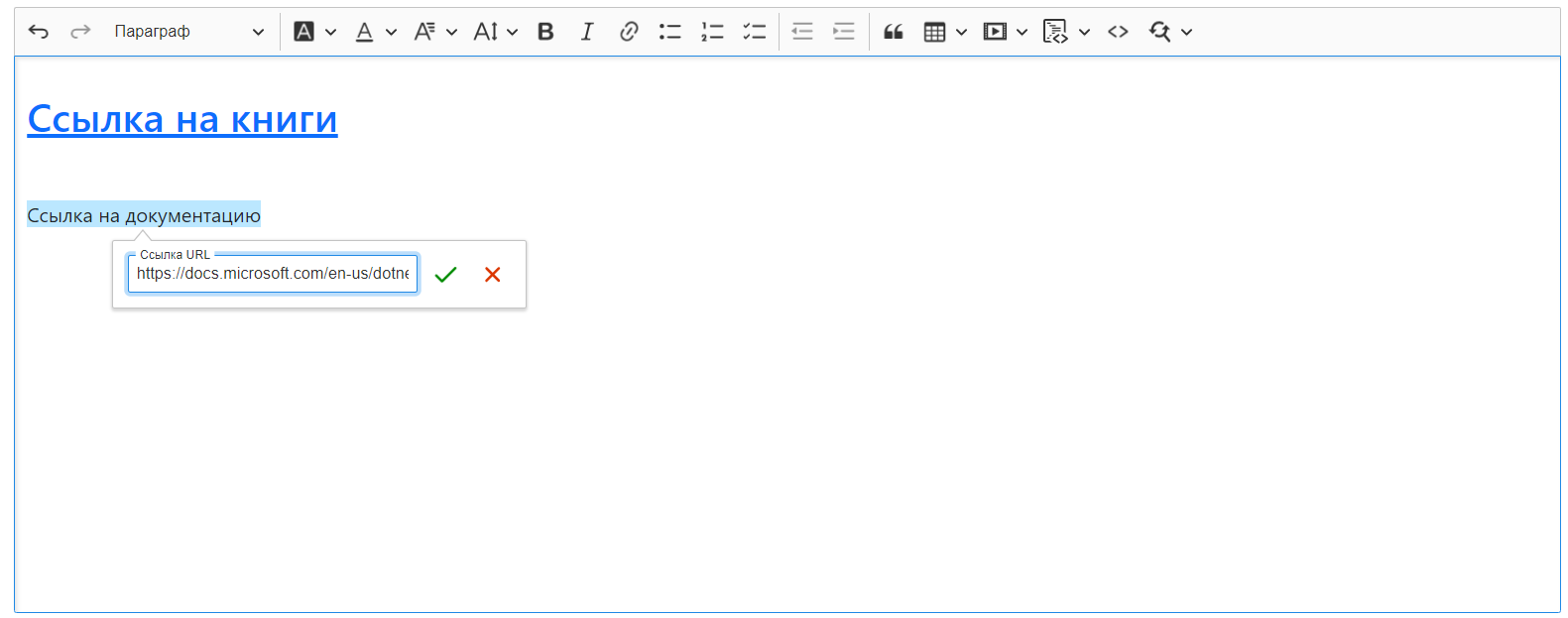


Рисунок 5.14 – Формирование ссылки

Элемент позволяет создавать списки трех видов:

* маркированный список;
* нумерованный список;
* список задач.

Пример с использованием всех типов списка представлен на рисунке 5.15.

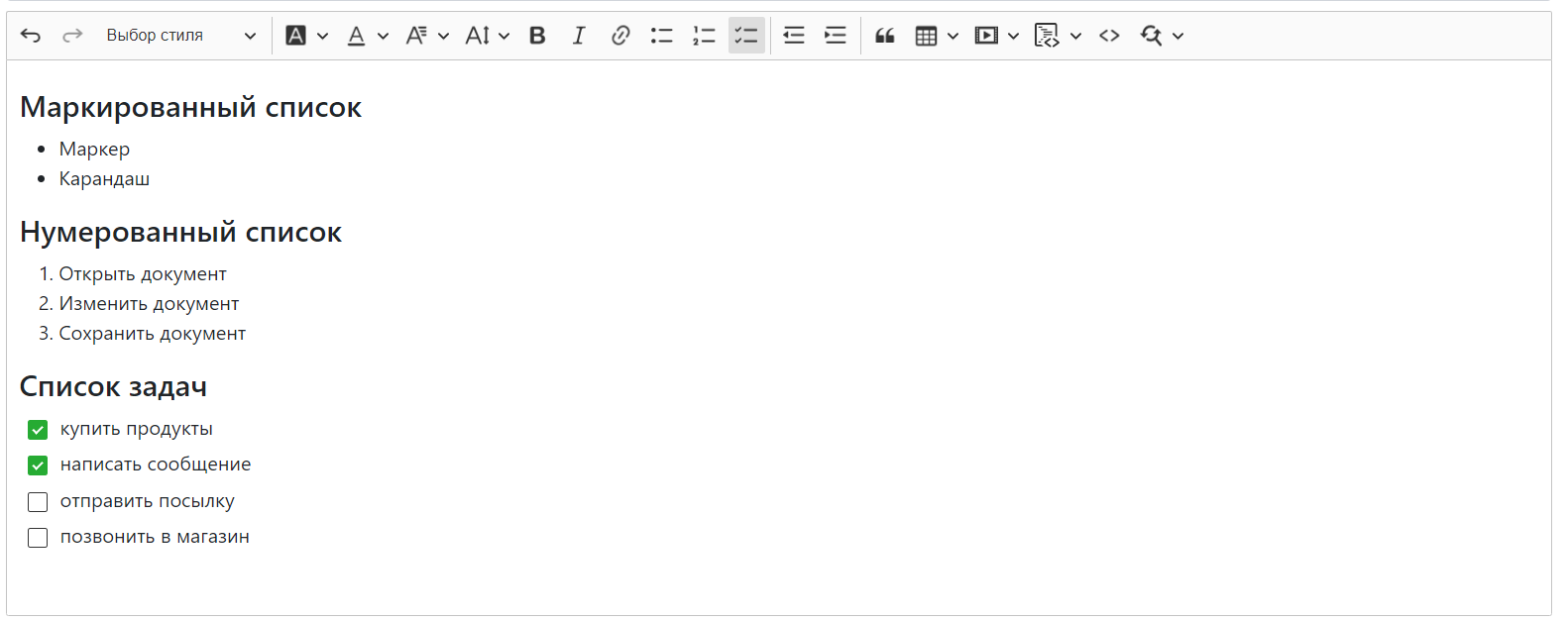


Рисунок 5.15 – Создание списков

Также элемент для формирования заметки позволяет делать цитаты. Для этого есть пункт меню «Цитаты». Пример текста с использованием цитат представлен на рисунке 5.16.

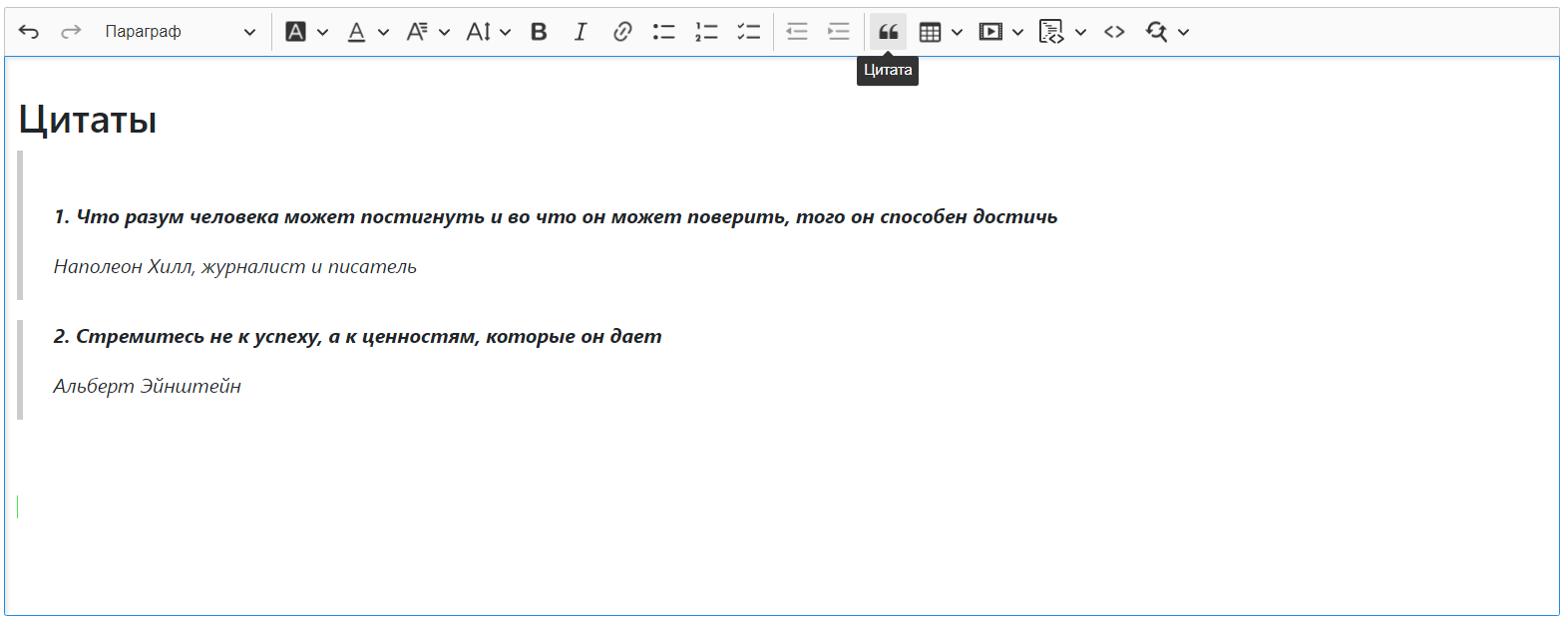


Рисунок 5.16 – Пример формирования цитат

Для добавления медиа ссылок требуется выбрать пункт меню «Вставить медиа» и вставить ссылку на видео. После чего в заметку добавиться превью к видео. Пример использования представлен на рисунке 5.17.

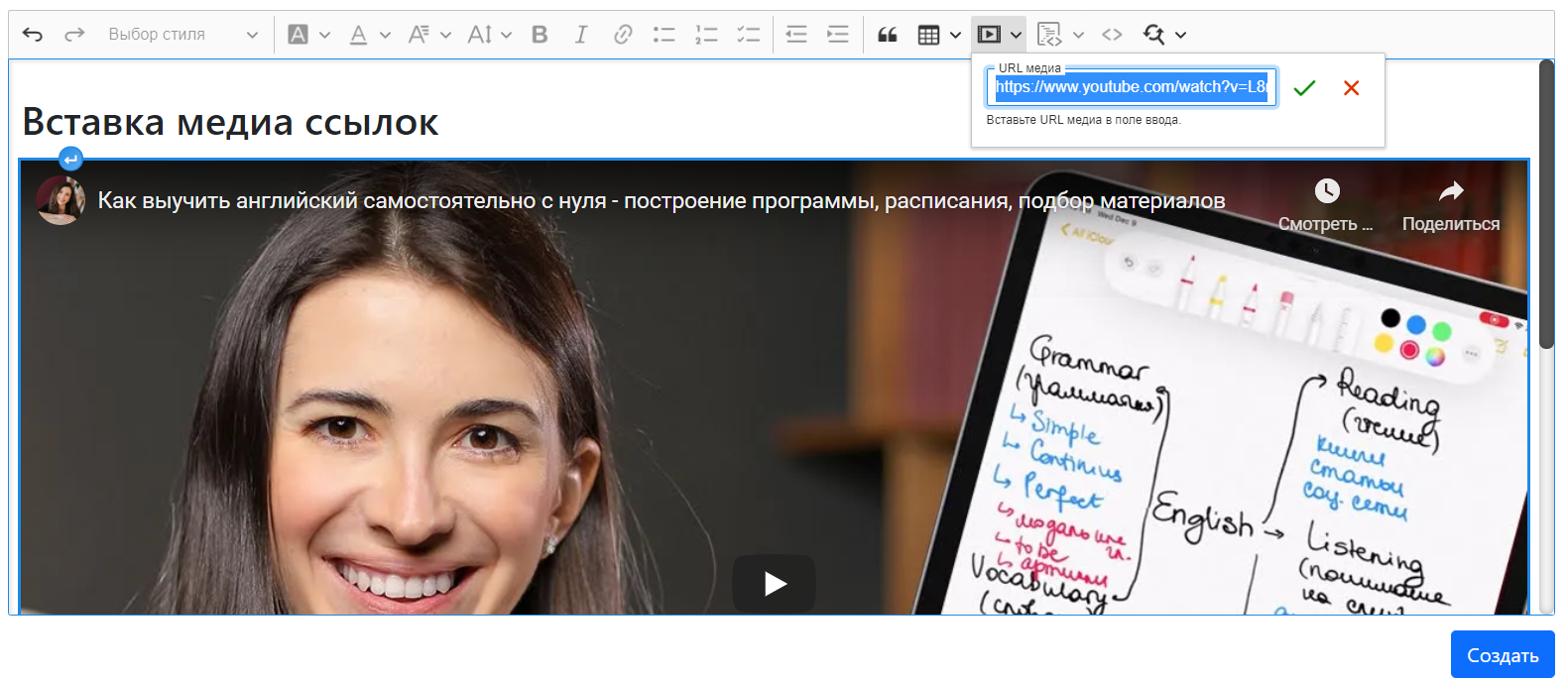


Рисунок 5.18 – Меню для вставки медиа ссылок

Для пользователей, которые хотят вставить в заметку код, есть пункт меню для вставки кода с выбором языка программирования. Пример вставки кода представлен на рисунке 5.19.

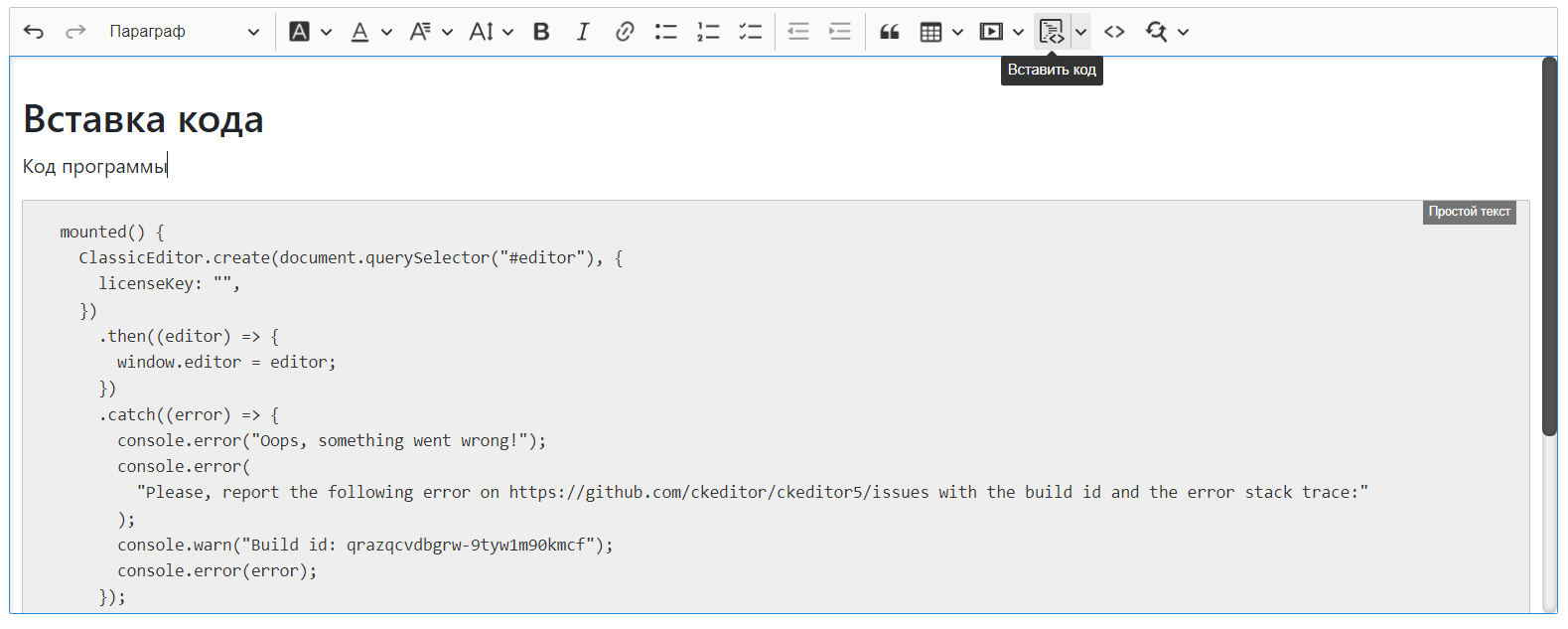


Рисунок 5.19 – Пример вставки кода

Пункт меню поиска позволяет найти фрагмент текста и заменить. Пример поиска представлен на рисунке 5.20.

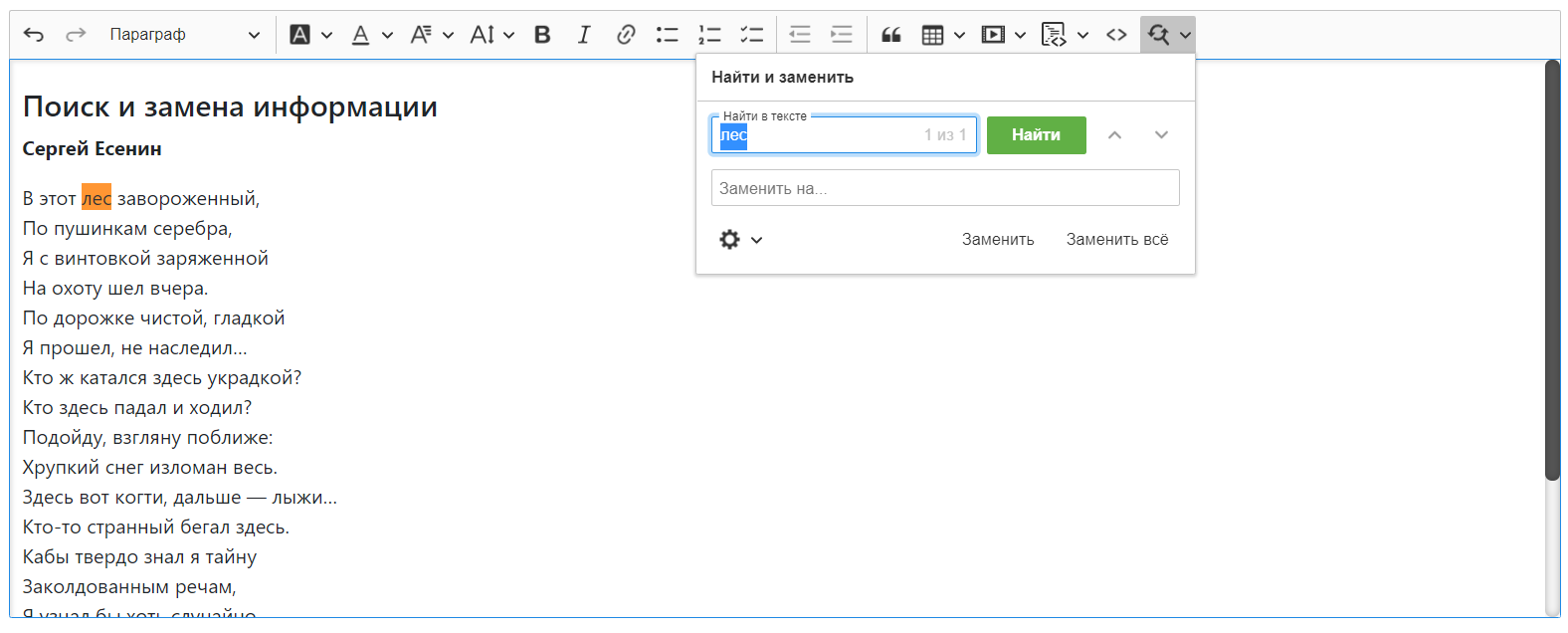


Рисунок 5.20 – Пример использования функций поиска по тексту

После заполнения заметки требуется нажать на кнопку «Создать». Появятся кнопки для обновления и удаления заметки. Результат создания заметки представлен на рисунке 5.21.

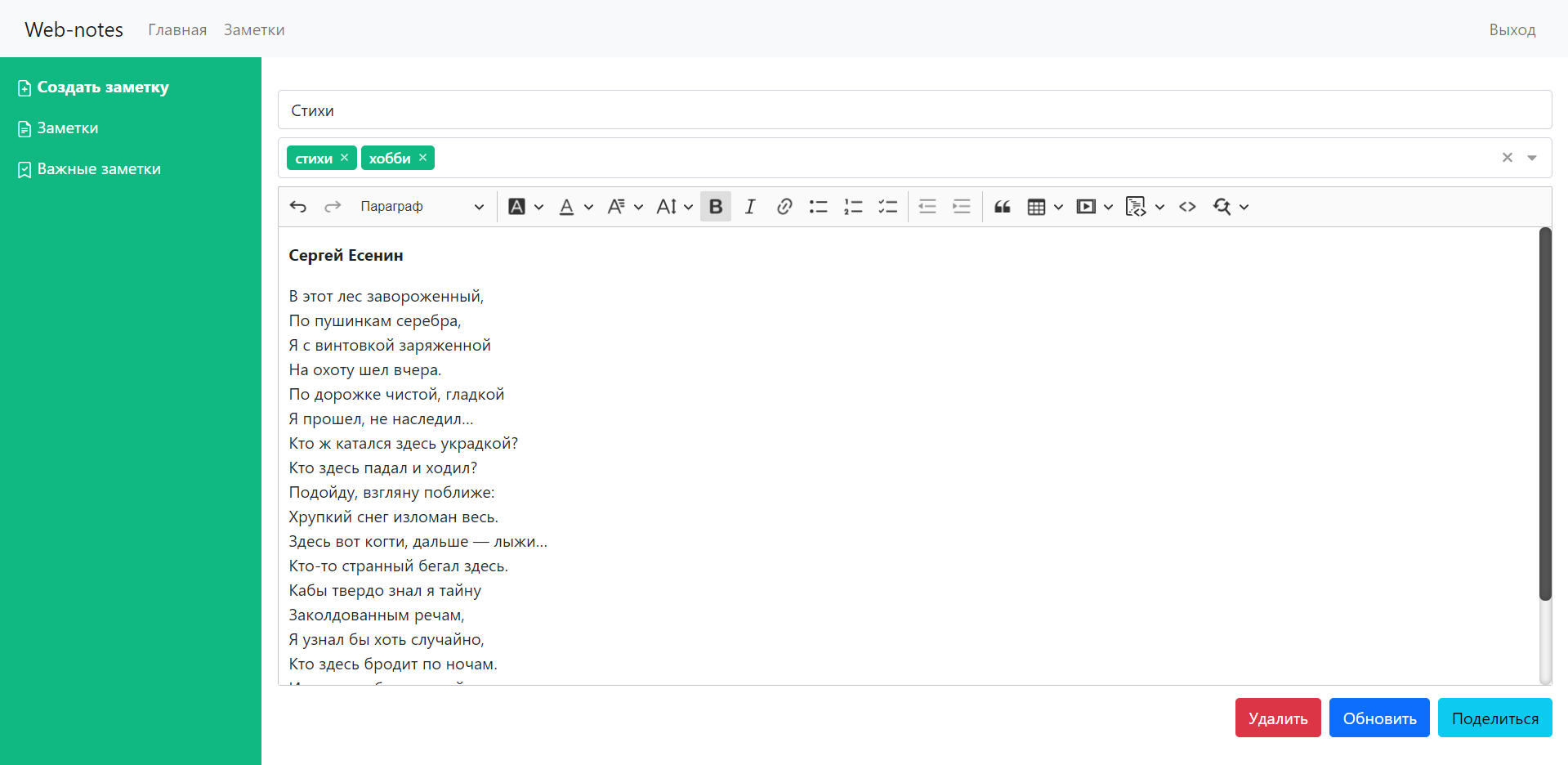


Рисунок 5.21 – Результат создания заметки

## 5.4 Поиск заметок

Для поиска заметки требуется в левой части навигации выбрать пункт «Заметки», после нажатия откроется форма со списком заметок и возможностью поиска по заметкам. На рисунке 5.22 представлена страница со списком заметок.

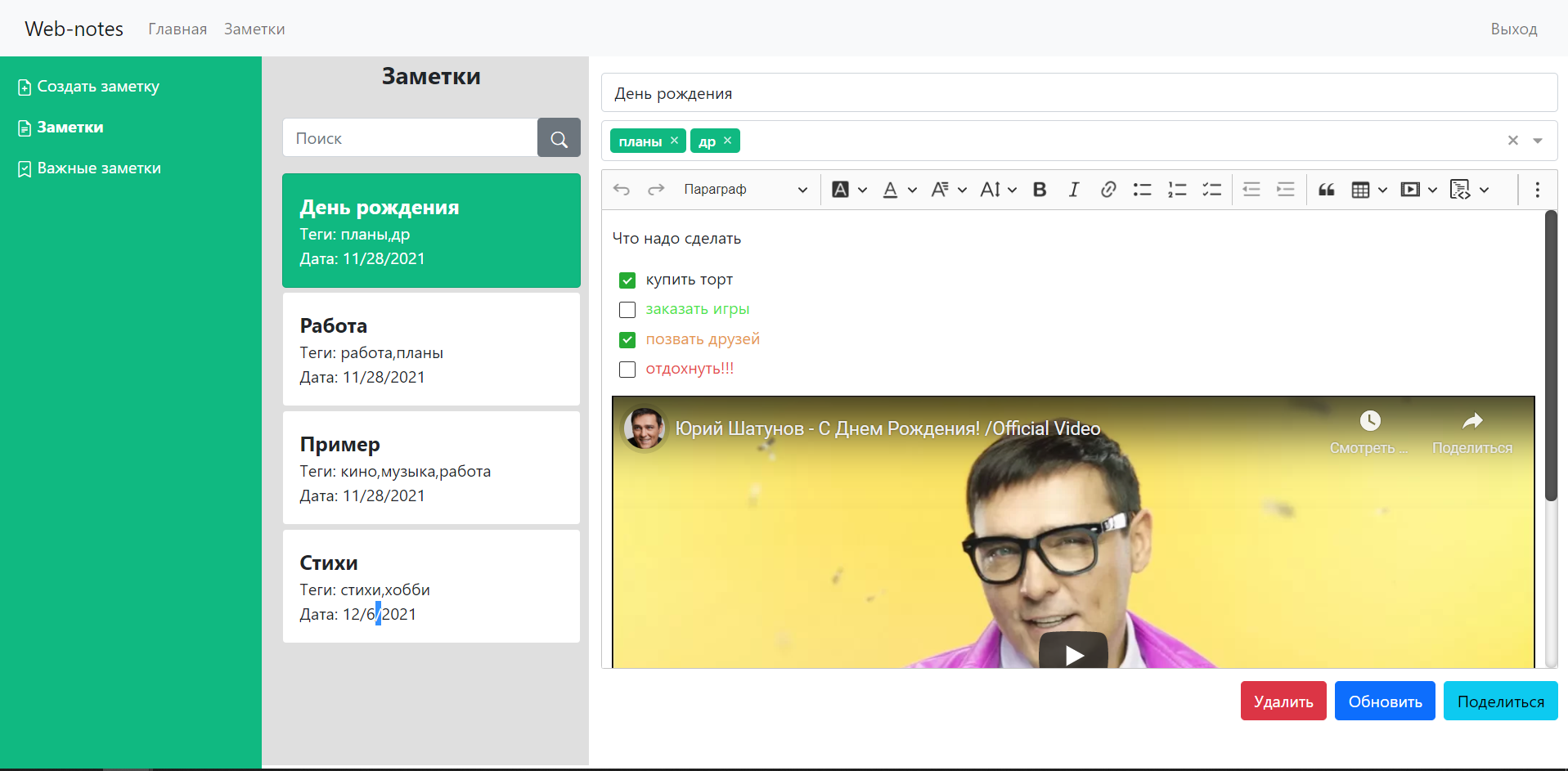


Рисунок 5.22 – Страница поиска заметок

Поиск по заметкам позволяет искать по полям: название, категория, текст заметки. Пример поиска по категории представлен на рисунке 5.23.

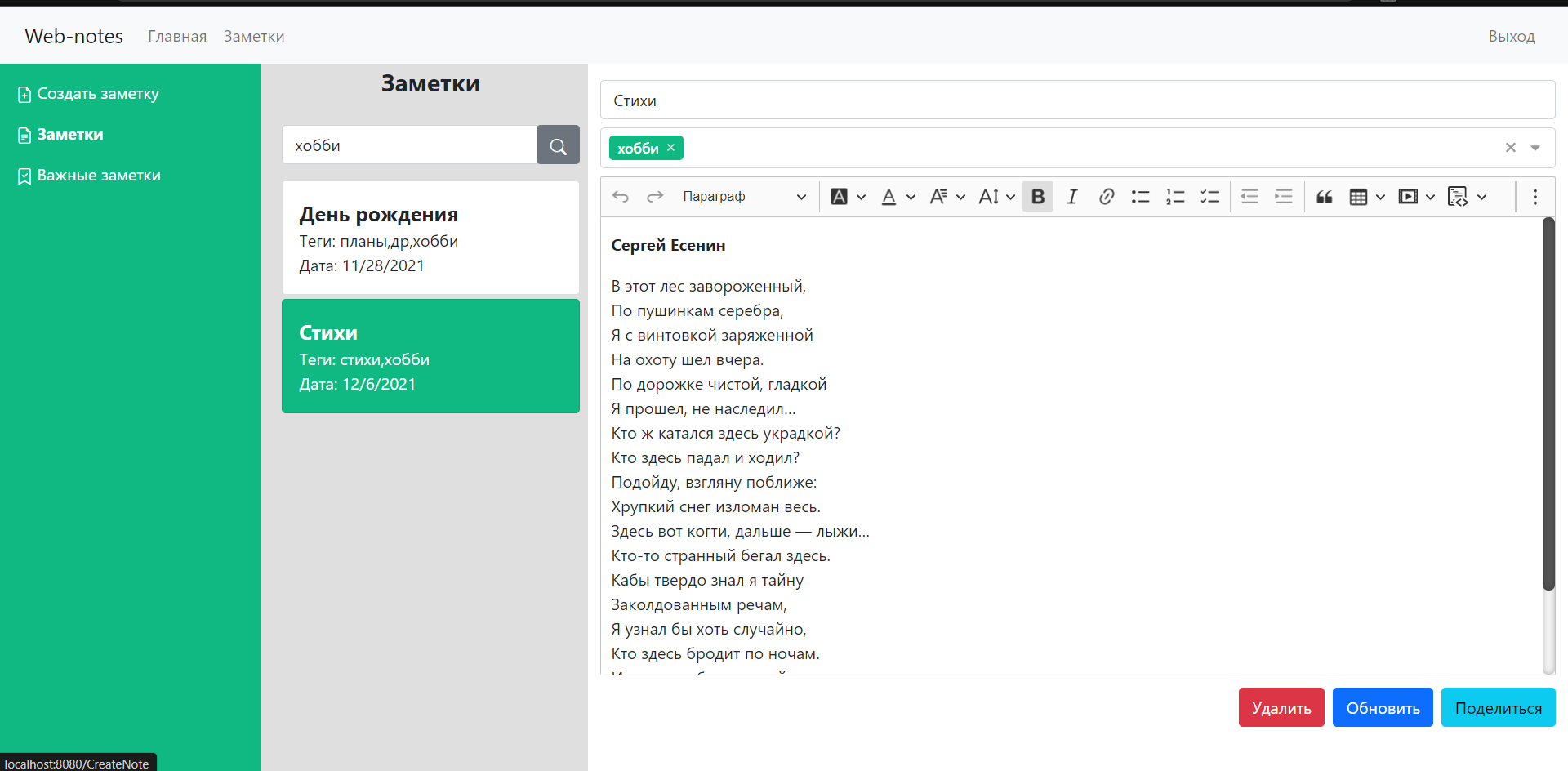


Рисунок 5.23 – Поиск по заметкам с использованием критерия категории

## 5.5 Поделиться заметкой

После создания заметки на форме появляется кнопка для возможности поделиться заметкой с неавторизованным пользователем. Для этого требуется нажать на кнопку «Поделиться» и скопировать ссылку. После того как пользователь для просмотра содержимого заметки откроет страницу по ссылке, ему будет доступна заметка в режиме чтения. Страница для просмотра заметки представлена на рисунке 5.24.

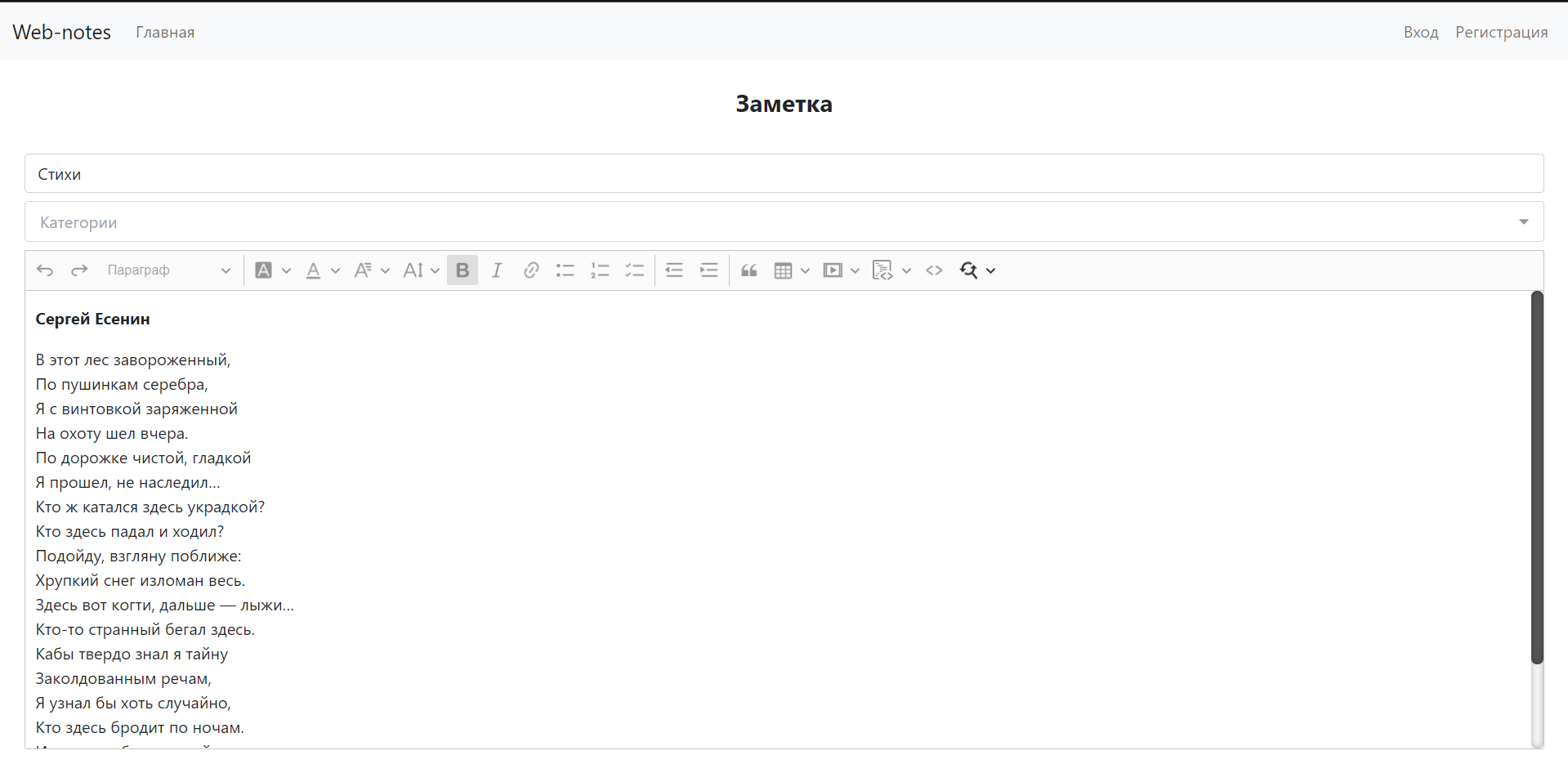


Рисунок 5.24 – Заметка в режиме чтения

## 5.6 Администратор

В веб-приложении существует пользователь с ролью администратор, ему доступны все вышеперечисленные возможности системы, а также возможность просматривать список пользователей, удалять пользователей и сбрасывать пароль по заявке пользователя. Страница со списком пользователей представлена на рисунке 5.25.

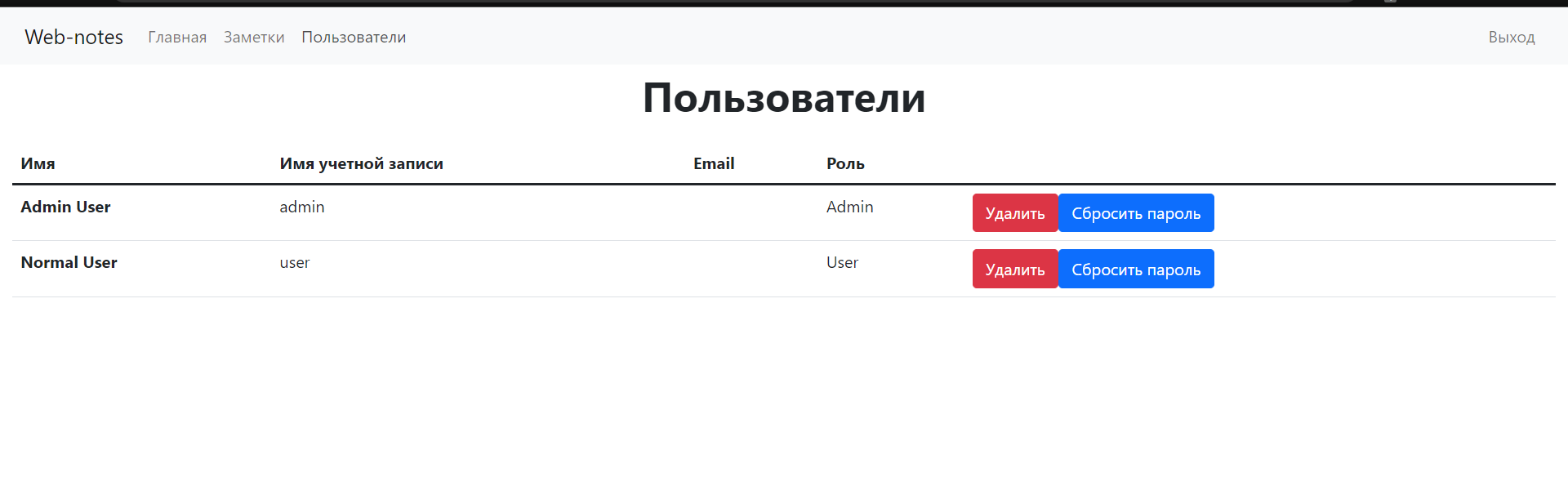


Рисунок 5.25 – Страница со списком пользователей

# 6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

## 6.1 Описание функций, назначения и потенциальных пользователей программного средства

Целью данного дипломного проекта является создание веб-приложения для ведения учета мобильных устройств департамента Комплексных решений ЗАО «Кьюликс Системс».

Разрабатываемое веб-приложение предназначено для применения командой тестирования департамента Комплексных решений ЗАО «Кьюликс Системс». Данное веб-приложение позволяет отследить, какой сотрудник использует конкретное устройство в данный момент и где устройство находится (дома или же в офисе). Так же данное веб-приложение позволяет создавать, редактировать и удалять информацию об устройствах департамента и о сотрудниках.

Программное средство разрабатывается для собственных нужд организации.

## 6.2 Расчет затрат на разработку программного средства

Затраты на разработку программного средства включают в себя следующие статьи:

* затраты на основную заработную плату разработчиков;
* затраты на дополнительную заработную плату разработчиков;
* отчисления на социальные нужды;
* прочие затраты (амортизационные отчисления, расходы на электроэнергию, командировочные расходы, арендная плата за офисные помещения и оборудование, расходы на управление и реализацию и т.п.).

**6.2.1** **Расчет затрат на основную заработную плату команды разработчиков**

Затраты на основную заработную плату определяются составом команды, которая занимается разработкой программного средств, месячным окладом специалистов и трудоемкостью процесса разработки и рассчитываются по формуле:

(1)

где n – количество исполнителей, занятых разработкой конкретного ПС;

Кпр – коэффициент премий;

– часовая заработная плата i-го исполнителя, руб.;

– трудоемкость работ, выполняемых i-м исполнителем, ч.

В разработке веб-приложения будет участвовать 3 исполнителя.

Данные по заработной плате команды разработчиков предоставлены ЗАО «Кьюликс Системс» на 29.10.2021.

Расчет затрат на основную заработную плату команды разработчиков представлено в таблице 1.

Таблица 1 – Расчет затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Участник команды | Вид выполняемой работы | Месячная заработная плата, руб. | Часовая заработная плата, руб. | Трудоемкость работ, ч. | Зарплата по тарифу, руб. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Инженер-программист | Разработка ПС | 1780 | 10,60 | 248 | 2628,80 |
| 2 | Бизнес-аналитик | Анализ требований | 1396 | 8,30 | 80 | 664 |
| 3 | Тестировщик | Тестирование ПС | 1480 | 8,80 | 88 | 774,4 |
| Премия (90% от основной заработной платы) | | | | | | 2365,92 |
| Итого затраты на основную заработную плату разработчика | | | | | | 4994,72 |

**6.2.2 Расчет затрат на дополнительную заработную плату**

Затраты на дополнительную заработную плату разработчика включает выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

(2)

где – затраты на основную заработную плату, руб.;

– норматив дополнительной заработной платы (18%).

Затраты на дополнительную заработную плату составят:

.

**6.2.3 Расчет отчислений на социальные нужды**

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

(3)

где – норматив отчислений на социальные нужды (34,6%).

Отчисления на социальные нужды составят:

**6.2.4 Расчет прочих затрат**

Прочие затраты включают затраты, связанные с разработкой конкретного программного средства напрямую, а также связанные с функционированием организации-разработчика в целом. Расчет прочих затрат выполняется в процентах от затрат на основную заработную плату команды разработчиков с учетом премии по формуле:

(4)

где – норматив прочих затрат (145%).

Рассчитаем сумму прочих затрат:

7242,34 руб.

Полная сумма затрат на разработку программного средства находится путем суммирования всех рассчитанных статей затрат. Расчет приведен в таблице 2.

Таблица 2 – Затраты на разработку программного средства

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| 1 | 2 |
| Основная заработная плата разработчика | 4994,72 |
| Дополнительная заработная плата разработчика | 899,05 |
| Отчисления на социальные нужды | 2039,24 |
| Прочие затраты | 7242,34 |
| Общая сумма затрат на разработку | 15175,35 |

Общая сумма затрат на разработку составила 15175,35 руб.

## 6.3 Оценка экономического эффекта от разработки и применения программного средства для собственных нужд организации

Расчет экономического эффекта в результате применения программного средства, разработанного для собственных нужд организации, рассчитывается по формуле:

(5)

где – затраты на разработку программного средства, руб.;

– экономия текущих затрат на ремонте мобильных устройств и уменьшении издержек на тестирование рабочих проектов сотрудниками, за счет возможности использовать мобильные устройства вне офиса, полученная в результате применения ПС, руб.;

– прирост текущих затрат на приобретение новых мобильных устройств, связанных с использованием ПС, руб.;

– ставка налога на прибыль, в соответствии с действующим законодательством (18%).

Экономия текущих затрат () по данным организации на 23.11.2021 составляет 29378 руб.

Прирост текущих затрат () по данным организации на 23.11.2021 составляет 3275 руб.

Рассчитаем экономический эффект:

29378 – 15175,35 – 3275) \*(1 – 0,18) = 8960,67 руб.

Также необходимо рассчитать рентабельность затрат на разработку ПС по формуле:

(6)

Рассчитаем рентабельность затрат на разработку ПС:

В данном случае экономический эффект при разработке программного средства «Веб-приложение для учета мобильных устройств ЗАО «Кьюликс Системс»» для собственных нужд составляет 8960,67 руб., а уровень его рентабельности составляет %.

Затраты на разработку программного средства составили 15175,35 руб.

# ЗАКЛЮЧЕНИЕ

В ходе дипломного проектирования проведен анализ литературы по теме дипломного проекта, изучены наиболее известные программные решения для создания и управления заметок. В процессе работы были исследованы основные направления по формирования заметок.

Для каждого программного средства были выделены его достоинства и недостатки использования, которые учитывались при формировании требований к программному средству.

В ходе моделирования программного средства были разработаны функциональные требования и описана функциональность к программному средству. На основе требований была разработана информационная модель и схема вариантов использования.

Исходя из полученных на этапе моделирования данных, была спроектирована архитектура программного решения и разработаны структуры классов и спроектированы алгоритмы работы программного средства. На основании информационной модели была спроектирована модель базы данных.

После разработки программного средства было проведено тестирование и исправление недочетов программы.

Таким образом, задачи, поставленные в рамках индивидуального задания, были выполнены. Знания и опыт полученные в процессе прохождения дипломного проектирования будут полезны при дальнейшей работе по специальности

# СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

[1] Интерактивная доска [Электронный ресурс]. – Электронные данные. – Режим доступа: https://hrwiki.ru/wiki/Interactive\_whiteboard – Дата доступа: 01.11.2021

[2] Интеллект карты [Электронный ресурс]. – Электронные данные. – Режим доступа: https://externat.foxford.ru/polezno-znat/kak-ispolzovat – Дата доступа: 01.11.2021

[3] Основные различия между веб-сайтом и веб-приложением [Электронный ресурс]. – Электронные данные. – Режим доступа: http://artismedia.by/blog/osnovnye-razlichiya-mezhdu-veb-sajtom – Дата доступа: 01.11.2021

[4] Google Keep – что это такое и как им пользоваться? [Электронный ресурс]. – Электронные данные. – Режим доступа: https://impression.ua/google-keep-chto-eto-takoe-i-kak-im-polzovatsya – Дата доступа: 05.11.2021

[5] Simplenote [Электронный ресурс]. – Электронные данные. – Режим доступа: https://simplenote.com – Дата доступа: 10.11.2021

[6] Notion. One workspace. Every team. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.notion.so – Дата доступа: 11.11.2021

[7] Моделирование на UML. Общие диаграммы [Электронный ресурс]. – Электронные данные. – Режим доступа: http://book.uml3.ru/sec\_1\_5 – Дата доступа: 17.11.2021

[8] Проектирование реляционной базы данных [Электронный ресурс]. – Электронные данные. – Режим доступа: https://itteach.ru/bazi-dannich/proektirovanie-relyatsionnoy-bazi-dannich – Дата доступа: 20.11.2021

[9] Одностраничные (spa) и многостраничные (pwa) веб-приложения [Электронный ресурс]. – Электронные данные. – Режим доступа: https://vc.ru/seo/odnostranichnye-spa-i-mnogostranichnye-pwa-veb-prilozheniya – Дата доступа: 21.11.2021

[10] Введение в ASP.NET Core [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0 – Дата доступа: 23.11.2021

[11] Бураков, П.В. Введение в системы баз данных: учебное пособие / П.В. Бураков, В.Ю. Петров – СПб: СПбГУ ИТМО, 2010. – 128 с.

[12] Горовой, В. Г. Экономическое обоснование проекта по разработке программного обеспечения : метод.пособие / В. Г. Горовой, А. В. Грицай, В.А. Пархименко. – Минск : БГУИР, 2018. – 12 с.

[1] Веб-приложение [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5> Дата доступа: 02.11.2021

[2] Штриховой код [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D1%80%D0%B8%D1%85%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%B4> Дата доступа: 02.11.2021

[3] Виды учета [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://ta-aspect.by/project-materials/hozyajstvennyj-uchet> Дата доступа: 05.11.2021

[4] Характеристика полной оперативной информации [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://infopedia.su/18x14bba.html> Дата доступа: 05.11.2021

[5] ПО «Hardware Inspector» [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://www.hwinspector.com/> Дата доступа: 08.11.2021

[6] ПО «Инвентаризация сети и учета и компьютеров» [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://studbooks.net/2226090/informatika/programmnyy_produkt_inventarizatsiya_seti_ucheta_kompyuterov> Дата доступа: 09.11.2021

[7] Диаграмма вариантов использования [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <http://book.uml3.ru/sec_1_5> Дата доступа: 11.11.2021

[8] Общие сведения об ASP.NET MVC [Электронный ресурс]. – Электронный ресурс. – Режим доступа:<https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview> Дата доступа: 12.11.2021

[9] Microsoft Visual Studio [Электронный ресурс]. – Электронный ресурс. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio Дата доступа: 15.11.2021

[10] Введение в язык C# и .NET Framework [Электронный ресурс]. – Электронный ресурс. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework Дата доступа: 16.11.2021

[11] <https://dic.academic.ru/dic.nsf/lower/14807> Закрытое акционерное общество

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Текст программы**

import { createApp } from "vue";

import App from "./App.vue";

import router from "./router";

import store from "./store";

import "bootstrap/dist/css/bootstrap.min.css";

import "@vueform/multiselect/themes/default.css";

import "bootstrap";

createApp(App).use(store).use(router).mount("#app");

<template>

<template v-if="!isAuthorized">

<NavBar />

<router-view />

</template>

<template v-else>

<div class="container-fluid p-0 m-0 min-vh-100 d-flex flex-column">

<div class="row p-0 m-0">

<div class="col p-0 m-0">

<NavBar />

</div>

</div>

<div class="row flex-grow-1 p-0 m-0">

<router-view />

</div>

</div>

</template>

</template>

<script>

import { mapGetters } from "vuex";

import NavBar from "@/components/NavBar";

export default {

name: "App",

components: {

NavBar,

},

created() {

this.$store.dispatch("Auth/checkAuth");

},

computed: {

...mapGetters("Auth", ["isAuthorized"]),

},

};

</script>

<style>

html,

body,

#app {

height: 100%;

overflow: hidden;

}

::-webkit-scrollbar-track {

-webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);

box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);

border-radius: 10px;

background-color: #f5f5f5;

}

::-webkit-scrollbar {

width: 12px;

background-color: #f5f5f5;

}

::-webkit-scrollbar-thumb {

border-radius: 10px;

-webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);

box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);

background-color: #555;

}

.sidebar {

background-color: #10b981;

color: white;

}

</style>

<template>

<div class="users-container container-fluid">

<p class="fs-1 fw-bold text-center">Пользователи</p>

<table class="table">

<thead>

<tr>

<th scope="col">Имя</th>

<th scope="col">Имя учетной записи</th>

<th scope="col">Email</th>

<th scope="col">Роль</th>

<th scope="col"></th>

</tr>

</thead>

<tbody>

<UserCard

v-for="user in users"

:key="user.id"

:user="user"

@delete="onDelete"

@reset="onReset"

/>

</tbody>

</table>

</div>

</template>

<script>

import UserCard from "@/components/UserCard.vue";

import { getUsers } from "../providers/userService.js";

export default {

name: "Users",

components: {

UserCard,

},

data() {

return {

users: [],

};

},

created() {

getUsers(this.$store.state.Auth.user).then(({ data }) => {

this.users = data;

});

},

methods: {

onDelete(id) {},

onReset(id) {},

},

};

</script>

<template>

<div class="p-4">

<p class="text-center fw-bold fs-4">Заметка</p>

<div class="form-group pt-3 pb-2">

<input

class="form-control"

type="text"

id="title"

v-model="note.title"

placeholder="Название"

/>

</div>

<div class="form-group pb-2">

<Multiselect

id="noteTags"

v-model="note.noteTags"

mode="tags"

placeholder="Категории"

:closeOnSelect="false"

:searchable="true"

:createTag="true"

:options="[

{ value: 'кино', label: 'кино' },

{ value: 'музыка', label: 'музыка' },

{ value: 'работа', label: 'работа' },

]"

/>

</div>

<div class="form-group pb-1">

<div id="editor">{{ note.noteDocument }}</div>

</div>

</div>

</template>

<script>

import { getNoteShared } from "../providers/noteService.js";

import Multiselect from "@vueform/multiselect";

export default {

name: "SharedNote",

components: {

Multiselect,

},

data() {

return {

isNew: true,

note: {},

};

},

mounted() {

ClassicEditor.create(document.querySelector("#editor"), {

licenseKey: "",

})

.then((editor) => {

window.editor = editor;

editor.isReadOnly = true;

debugger;

getNoteShared(this.$route.query.id, this.$store.state.Auth.user).then(

({ data }) => {

this.note = { ...data };

window.editor.setData(this.note.noteDocument);

}

);

})

.catch((error) => {

console.error("Oops, something went wrong!");

console.error(

"Please, report the following error on https://github.com/ckeditor/ckeditor5/issues with the build id and the error stack trace:"

);

console.warn("Build id: qrazqcvdbgrw-9tyw1m90kmcf");

console.error(error);

});

},

};

</script>

<template>

<div>

<div class="div-center shadow p-3 bg-body rounded">

<div class="content">

<h3 class="text-center">Регистрация</h3>

<hr />

<form @submit.prevent="register">

<div class="form-group pb-1">

<label for="userName" class="form-label">Имя</label>

<input

class="form-control"

type="text"

id="userName"

v-model="registerModel.firstName"

required

/>

</div>

<div class="form-group pb-1">

<label for="userName" class="form-label">Фамилия</label>

<input

class="form-control"

type="text"

id="userName"

v-model="registerModel.lastName"

required

/>

</div>

<div class="form-group pb-1">

<label for="userName" class="form-label">Email</label>

<input

class="form-control"

type="email"

id="userName"

v-model="registerModel.email"

required

/>

</div>

<div class="form-group pb-1">

<label for="userName" class="form-label">Имя пользователя</label>

<input

class="form-control"

type="text"

id="userName"

v-model="registerModel.userName"

required

/>

</div>

<div class="form-group pb-1">

<label for="password" class="form-label">Пароль</label>

<input

class="form-control"

type="password"

id="password"

v-model="registerModel.password"

required

/>

</div>

<div class="form-group pb-1">

<label for="comparePassword" class="form-label"

>Повторить Пароль</label

>

<input

class="form-control"

type="password"

id="comparePassword"

v-model="registerModel.comparePassword"

required

/>

</div>

<div class="text-center pt-2">

<button class="btn btn-primary" type="submit">Регистрация</button>

</div>

<hr />

<p>

Если у вас есть аккаунт вы можете войти:

<router-link to="/login">Вход</router-link>

</p>

</form>

</div>

</div>

</div>

</template>

<script>

export default {

name: "Register",

data() {

return {

registerModel: {

userName: "",

comparePassword: "",

password: "",

firstName: "",

lastName: "",

},

};

},

methods: {

register() {

this.$store.dispatch("Auth/register", this.registerModel);

},

},

};

</script>

<template>

<div class="col-md-2 p-0 m-0 sidebar">

<SideBar />

</div>

<div class="col-md-10 p-0 m-0">

<div class="row p-0 m-0 h-100">

<div class="col-md-3 list-container">

<p class="fw-bold fs-4 text-center">Заметки</p>

<SearchNotes @search="onSearch" />

<div class="overflow-auto list-notes">

<ListNotes :notes="notes" @openNote="onOpenNote" />

</div>

</div>

<div class="col-md-9">

<NoteForm

:isNew="isNew"

:note="note"

@update="onUpdate"

@delete="onDelete"

/>

</div>

</div>

</div>

</template>

<script>

import NoteForm from "@/components/NoteForm.vue";

import ListNotes from "@/components/ListNotes.vue";

import SideBar from "@/components/SideBar";

import SearchNotes from "@/components/SearchNotes.vue";

import { getNotes, deleteNote, updateNote } from "../providers/noteService.js";

export default {

name: "Notes",

components: {

NoteForm,

ListNotes,

SearchNotes,

SideBar,

},

data() {

return {

notes: [],

note: null,

isNew: false,

searchValue: "",

};

},

created() {

getNotes(this.$store.state.Auth.user, this.searchValue).then(({ data }) => {

this.notes = data;

this.note = data[0];

this.note.isCurrent = true;

});

},

methods: {

onOpenNote(id) {

this.note.isCurrent = false;

this.note = this.notes.find((x) => {

return x.id === id;

});

this.note.isCurrent = true;

},

onSearch(searchValue) {

this.searchValue = searchValue;

this.updateListNotes();

},

onUpdate(note) {

updateNote(this.$store.state.Auth.user, note).then(({ data }) => {

console.log(data);

this.updateListNotes().then(() => {

this.onOpenNote(note.id);

});

});

},

onDelete(id) {

deleteNote(this.$store.state.Auth.user, id).then(() => {

getNotes(this.$store.state.Auth.user, this.searchValue).then(

({ data }) => {

this.notes = data;

this.note.isCurrent = false;

this.note = data[0];

this.note.isCurrent = true;

}

);

});

},

updateListNotes() {

return getNotes(this.$store.state.Auth.user, this.searchValue).then(

({ data }) => {

this.notes = data;

return data;

}

);

},

},

};

</script>

<style scoped>

.list-container {

background-color: rgb(223, 223, 223);

height: 92vh;

padding-right: 0px;

}

.list-notes {

height: 77vh;

}

</style>

<template>

<div>

<div class="div-center shadow p-3 bg-body rounded">

<div class="content">

<h3 class="text-center">Вход</h3>

<hr />

<form @submit.prevent="login">

<div class="form-group pb-1">

<label for="userName" class="form-label">Имя Пользователя</label>

<input

class="form-control"

type="text"

id="userName"

v-model="loginModel.userName"

required

/>

</div>

<div class="form-group pb-1">

<label for="password" class="form-label">Пароль</label>

<input

class="form-control"

type="password"

id="password"

v-model="loginModel.password"

required

/>

</div>

<div class="pt-2 text-center">

<button type="submit" class="btn btn-primary">Вход</button>

</div>

<hr />

<p>

Если у вас нет аккаунта вы можете зарегистрироваться:

<router-link to="/register">Регистрация</router-link>

</p>

</form>

</div>

</div>

</div>

</template>

<script>

export default {

name: "Login",

data() {

return {

loginModel: {

userName: "",

password: "",

},

};

},

methods: {

login() {

this.$store.dispatch("Auth/login", this.loginModel).then(() => {

debugger;

this.$router.push("Notes");

});

},

},

};

</script>

<style>

.div-center {

width: 500px;

height: 400px;

background-color: #fff;

position: absolute;

left: 0;

right: 0;

top: 60px;

bottom: 0;

margin: auto;

max-width: 100%;

max-height: 100%;

overflow: auto;

padding: 1em 2em;

border-bottom: 2px solid #ccc;

display: table;

}

div.content {

display: table-cell;

vertical-align: middle;

}

</style>

import api from "../providers/api.js";

export const createNote = (user, note) => {

return api.post(

"/Notes/CreateNote",

{

...note,

userId: user.id,

},

headers(user)

);

};

export const updateNote = (user, note) => {

return api.post(

"/Notes/UpdateNote",

{

...note,

userId: user.id,

},

headers(user)

);

};

export const deleteNote = (user, id) => {

return api.delete(`/Notes/DeleteNote/${id}`, headers(user));

};

export const getNotes = (user, searchValue = "") => {

return api.get(`/Notes/GetNotes?SearchValue=${searchValue}`, headers(user));

};

export const getNote = (id, user) => {

return api.get(`/Notes/GetNote/${id}`, headers(user));

};

export const getNoteSharedId = (id, user) => {

return api.get(`/Notes/GetNoteSharedId/${id}`, headers(user));

};

export const getTags = ( user) => {

return api.get(`/Notes/GetTags/${user.id}`, headers(user));

};

export const getNoteShared= (id, user) => {

return api.get(`/Notes/GetNoteShared/${id}`, headers(user));

};

function headers(user) {

const authHeader = user.token

? { Authorization: "Bearer " + user.token }

: {};

return {

headers: {

...authHeader,

"Content-Type": "application/json",

},

};

}

import api from "../providers/api.js";

export const getUsers = (user) => {

return api.get("/Users/GetAll", headers(user));

};

function headers(user) {

const authHeader = user.token

? { Authorization: "Bearer " + user.token }

: {};

return {

headers: {

...authHeader,

"Content-Type": "application/json",

},

};

}

import axios from "axios";

const apiProvider = axios.create({

baseURL: "https://localhost:44325/api/",

});

apiProvider.interceptors.response.use(

(response) => response,

(error) => {

let title = "Error";

let errorMessage = error.response.data.message;

if (errorMessage) {

alert("Error message: " + errorMessage);

}

return Promise.reject(error);

}

);

export default apiProvider;

import api from "../providers/api.js";

export default {

namespaced: true,

state: {

user: {},

},

getters: {

isAuthorized: (state) => {

return !!state.user.token;

},

role: (state) => {

return state.user.role;

},

isAdmin: (state) =>{

return state.user.role === "Admin";

}

},

mutations: {

setUser(state, user) {

state.user = user;

},

},

actions: {

checkAuth(context) {

context.commit("setUser", JSON.parse(localStorage.getItem("auth")) || {});

},

login(context, loginData) {

return api.post("/Users/Authenticate", loginData).then(({ data }) => {

console.log(data);

context.commit("setUser", data);

localStorage.setItem("auth", JSON.stringify(data));

});

},

register(context, registerData) {

return api.post("/Users/Register", registerData).then(({ data }) => {

console.log(data);

context.commit("setUser", data);

localStorage.setItem("auth", JSON.stringify(data));

});

},

logout(context) {

context.commit("setUser", {});

localStorage.setItem("auth", JSON.stringify({}));

},

},

};

import { createStore } from "vuex";

import AuthStoreModule from "../store/auth.js";

export default createStore({

modules: {

Auth: AuthStoreModule

},

});

<template>

<div class="h-100 me-2 sidebar pt-2">

<ul class="nav nav-pills flex-column mb-auto">

<li class="nav-item">

<router-link class="nav-link text-white" :class="{'fw-bold': $route.name === 'CreateNote'}" to="/CreateNote"

><i class="bi bi-file-earmark-plus"></i> Создать заметку</router-link

>

</li>

<li class="nav-item">

<router-link class="nav-link text-white" :class="{'fw-bold': $route.name === 'Notes'}" to="/Notes"

><i class="bi bi-file-earmark-text"></i> Заметки</router-link

>

</li>

<li class="nav-item">

<router-link class="nav-link text-white" :class="{'fw-bold': $route.name === 'ImportantNotes'}" to="/ImportantNotes"

><i class="bi bi-bookmark-check"></i> Важные заметки</router-link

>

</li>

</ul>

</div>

</template>

<template>

<div class="form-group pt-3 pb-2">

<!-- <label for="title" class="form-label">Название</label> -->

<input

class="form-control"

type="text"

id="title"

v-model="noteModel.title"

placeholder="Название"

/>

</div>

<div class="form-group pb-2">

<Multiselect

id="noteTags"

v-model="noteModel.noteTags"

mode="tags"

placeholder="Категории"

:closeOnSelect="false"

:searchable="true"

:createTag="true"

:options="categoryOptions"

/>

</div>

<div class="form-group pb-1">

<div id="editor">{{ noteModel.noteDocument }}</div>

</div>

<div class="d-md-flex justify-content-md-end py-2">

<div v-if="isNew">

<button class="btn btn-primary" v-on:click="onCreateNote()">

Создать

</button>

</div>

<div v-else>

<button class="btn btn-danger me-md-2" v-on:click="onDeleteNote()">

Удалить

</button>

<button class="btn btn-primary me-md-2" v-on:click="onUpdateNote()">

Обновить

</button>

<button class="btn btn-info" v-on:click="onShareNote()">

Поделиться

</button>

</div>

</div>

</template>

<script>

import Multiselect from "@vueform/multiselect";

import { getNoteSharedId, getTags } from "../providers/noteService.js";

export default {

name: "NoteForm",

props: {

isNew: Boolean,

note: Object,

},

emits: ["create"],

components: {

Multiselect,

},

data() {

return {

categoryOptions: [],

editor: null,

noteModel: {

title: "",

noteTags: [],

noteDocument: "",

options: [],

},

options: ["Batman", "Robin", "Joker"],

};

},

created() {

getTags(this.$store.state.Auth.user).then(({ data }) => {

this.categoryOptions = data.map((value) => {

return { value: value, label: value };

});

});

},

watch: {

note() {

if (this.note) {

this.noteModel = { ...this.note };

window.editor.setData(this.noteModel.noteDocument);

}

},

},

mounted() {

ClassicEditor.create(document.querySelector("#editor"), {

licenseKey: "",

})

.then((editor) => {

window.editor = editor;

})

.catch((error) => {

console.error("Oops, something went wrong!");

console.error(

"Please, report the following error on https://github.com/ckeditor/ckeditor5/issues with the build id and the error stack trace:"

);

console.warn("Build id: qrazqcvdbgrw-9tyw1m90kmcf");

console.error(error);

});

},

methods: {

onCreateNote() {

this.$emit("create", {

...this.note,

title: this.noteModel.title,

noteTags: this.noteModel.noteTags,

noteDocument: window.editor.getData(),

});

},

onUpdateNote() {

this.$emit("update", {

...this.note,

title: this.noteModel.title,

noteTags: this.noteModel.noteTags,

noteDocument: window.editor.getData(),

});

},

onDeleteNote() {

if (window.confirm("Вы точно хотите удалить заметку?")) {

this.$emit("delete", this.note.id);

}

},

onShareNote() {

getNoteSharedId(this.note.id, this.$store.state.Auth.user).then(

({ data }) => {

debugger;

alert(`http://localhost:8080/Shared?id=${data}`);

}

);

},

},

};

</script>

<style>

.ck-editor\_\_editable {

height: 450px;

}

</style>

<template>

<div class="p-2">

<div v-for="note in notes" :key="note.id">

<ListCardNote :note="note" @openNote="onOpenNote"></ListCardNote>

</div>

</div>

</template>

<script>

import ListCardNote from "@/components/ListCardNote.vue";

export default {

name: "ListNotes",

components: {

ListCardNote,

},

props: {

notes: Array,

},

emits: ["openNote"],

methods:{

onOpenNote(id){

this.$emit("openNote", id);

}

}

};

</script>

<template>

<div class="card m-0 mb-1" :class="{ 'current-note': isCurrent}" v-on:click="cardClick">

<div class="card-body">

<p class="p-0 m-0 fw-bold fs-5">

{{ note.title }}

</p>

<p class="p-0 m-0 fs-6">Теги: {{ categories }}</p>

<p class="p-0 m-0 fs-6">Дата: {{ date }}</p>

</div>

</div>

</template>

<script>

export default {

name: "ListCardNote",

props: {

note: Object,

},

emits: ["openNote"],

computed: {

categories() {

return this.note.noteTags.join();

},

date() {

let date = new Date(this.note.createdDateTime);

return date.toLocaleDateString("en-US");

},

isCurrent(){

return this.note.isCurrent;

}

},

methods:{

cardClick(){

this.$emit("openNote", this.note.id);

}

}

};

</script>

<style scoped>

.current-note{

background-color: #10b981;

color: white;

}

.card:hover{

border: solid 1px green;

}

</style>

using System;

using System.Collections.Generic;

using AutoMapper;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

using WebNotesApi.Authorization;

using WebNotesApi.Models.Notes;

using WebNotesApplication.Models;

using WebNotesApplication.Services;

namespace WebNotesApi.Controllers

{

[Authorize]

[ApiController]

[Route("api/[controller]")]

public class NotesController

{

private readonly INoteService \_noteService;

private readonly IMapper \_mapper;

public NotesController(INoteService noteService, IMapper mapper)

{

\_noteService = noteService;

\_mapper = mapper;

}

[HttpGet("[action]/{id:int}")]

public async Task<ActionResult<NoteResponse>> GetNote(int id)

{

var response = await \_noteService.GetNote(new SearchNoteModel()

{

Id = id

});

return \_mapper.Map<NoteResponse>(response);

}

[HttpGet("[action]")]

public async Task<ActionResult<IEnumerable<NoteResponse>>> GetNotes([FromQuery] SearchNoteModel searchNoteModel)

{

var response = await \_noteService.GetNotes(searchNoteModel);

var lResponse = \_mapper.Map<IEnumerable<NoteResponse>>(response);

foreach (var note in lResponse)

{

note.AllAccessTags = await \_noteService.Tags( note.UserId);

}

return new OkObjectResult(lResponse);

}

[HttpPost("[action]")]

public async Task<ActionResult<NoteResponse>> CreateNote(CreateNoteRequest request)

{

var response = await \_noteService.CreateNoteAsync(\_mapper.Map<CreateNoteModel>(request));

var lResponse = \_mapper.Map<NoteResponse>(response);

lResponse.AllAccessTags = await \_noteService.Tags( lResponse.UserId);

return new OkObjectResult(lResponse);

}

[HttpPost("[action]")]

public async Task<ActionResult<NoteResponse>> UpdateNote(UpdateNoteRequest request)

{

var response = await \_noteService.UpdateNoteAsync(\_mapper.Map<UpdateNoteModel>(request));

var lResponse = \_mapper.Map<NoteResponse>(response);

lResponse.AllAccessTags = await \_noteService.Tags( lResponse.UserId);

return new OkObjectResult(lResponse);

}

[HttpDelete("[action]/{id:int}")]

public async Task<ActionResult> DeleteNote(int id)

{

await \_noteService.DeleteNoteAsync(id);

return new OkResult();

}

[AllowAnonymous]

[HttpGet("[action]/{id}")]

public async Task<ActionResult<NoteResponse>> GetNoteShared(Guid id)

{

var response = await \_noteService.GetNote(new SearchNoteModel()

{

SharedId = id

});

var lResponse = \_mapper.Map<NoteResponse>(response);

lResponse.AllAccessTags = await \_noteService.Tags( lResponse.UserId);

return new OkObjectResult(lResponse);

}

[AllowAnonymous]

[HttpGet("[action]/{id}")]

public async Task<Guid> GetNoteSharedId(int id)

{

var response = await \_noteService.CreateSharedId(id);

return response;

}

[AllowAnonymous]

[HttpGet("[action]/{id}")]

public async Task<ActionResult<IEnumerable<string>>> GetTags(int userId)

{

var response = await \_noteService.Tags(userId);

return response;

}

}

}

using System.Collections.Generic;

using AutoMapper;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

using WebNotesApi.Authorization;

using WebNotesApi.Models.Users;

using WebNotesApplication.Models;

using WebNotesApplication.Services;

using WebNotesData.Entities;

namespace WebNotesApi.Controllers

{

[Authorize]

[ApiController]

[Route("api/[controller]")]

public class UsersController : ControllerBase

{

private readonly IUserService \_userService;

private readonly IMapper \_mapper;

public UsersController(IUserService userService, IMapper mapper)

{

\_userService = userService;

\_mapper = mapper;

}

[AllowAnonymous]

[HttpPost("[action]")]

public async Task<IActionResult> Authenticate(AuthenticateRequest model)

{

var response = await \_userService.AuthenticateAsync(\_mapper.Map<LoginModel>(model));

HttpContext.Items["User"] = \_mapper.Map<User>(response);

return Ok(response);

}

[AllowAnonymous]

[HttpPost("[action]")]

public async Task<IActionResult> Register(RegisterRequest model)

{

var response = await \_userService.RegisterAsync(\_mapper.Map<User>(model));

return Ok(response);

}

[Authorize(Role.Admin)]

[HttpGet("[action]")]

public async Task<ActionResult<List<User>>> GetAll()

{

var users = await \_userService.GetAllAsync();

return users;

}

[Authorize(Role.Admin)]

[HttpDelete("[action]/{id:int}")]

public async Task<ActionResult<List<User>>> Delete(int id)

{

await \_userService.DeleteUserAsync(id);

return Ok();

}

[Authorize(Role.Admin)]

[HttpPost("[action]/{id:int}")]

public async Task<ActionResult<List<User>>> ResetPassword(int id)

{

await \_userService.ResetPassword(id);

return Ok();

}

[HttpGet("{id:int}")]

public async Task<IActionResult> GetById(int id)

{

// only admins can access other user records

var currentUser = (User)HttpContext.Items["User"];

if (id != currentUser.Id && currentUser.Role != Role.Admin)

{

return Unauthorized(new { message = "Unauthorized" });

}

var user = await \_userService.GetByIdAsync(id);

return Ok(user);

}

}

}

using System.Collections.Generic;

namespace WebNotesApi.Models.Notes

{

public class CreateNoteRequest

{

public string Title { get; set; }

public string NoteDocument { get; set; }

public int UserId { get; set; }

public List<string> NoteTags { get; set; }

}

}

using System.Linq;

using AutoMapper;

using BCrypt.Net;

using WebNotesApi.Models.Notes;

using WebNotesApi.Models.Users;

using WebNotesApplication.Models;

using WebNotesData.Entities;

using BCryptNet = BCrypt.Net.BCrypt;

namespace WebNotesApi.Profiles

{

public class MappingProfile : Profile

{

public MappingProfile()

{

CreateMap<User, AuthenticateResult>().ReverseMap();

CreateMap<AuthenticateResponse, AuthenticateResult>().ReverseMap();

CreateMap<AuthenticateRequest, LoginModel>().ReverseMap();

CreateMap<RegisterRequest, User>()

.ForMember(user => user.PasswordHash, opt => opt.MapFrom(request => BCryptNet.HashPassword(request.Password, SaltRevision.Revision2B)))

.ForMember(user => user.Role, opt => opt.MapFrom(request => Role.User));

CreateMap<CreateNoteModel, Note>()

.ForMember(x => x.NoteTags, m => m.Ignore());

CreateMap<CreateNoteRequest, CreateNoteModel>();

CreateMap<UpdateNoteRequest, UpdateNoteModel>();

CreateMap<UpdateNoteModel, Note>()

.ForMember(x => x.NoteTags, m => m.Ignore());

CreateMap<Note, NoteResponse>().ForMember(x => x.NoteTags, c => c.MapFrom(t => t.NoteTags.Select(x => x.Tag.Value)));

}

}

}

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using WebNotesApplication.Models;

using WebNotesData.Entities;

namespace WebNotesApplication.Services

{

public interface INoteService

{

Task<Note> GetNote(SearchNoteModel searchNoteModel);

Task<List<Note>> GetNotes(SearchNoteModel searchNoteModel);

Task<Note> CreateNoteAsync(CreateNoteModel createNoteModel);

Task<Note> UpdateNoteAsync(UpdateNoteModel updateNoteModel);

Task DeleteNoteAsync(int id);

Task<Guid> CreateSharedId(int id);

Task<List<string>> Tags( int userId);

}

}

using System.Collections.Generic;

using System.Threading.Tasks;

using WebNotesApplication.Models;

using WebNotesData.Entities;

namespace WebNotesApplication.Services

{

public interface IUserService

{

Task<AuthenticateResult> AuthenticateAsync(LoginModel model);

Task<AuthenticateResult> RegisterAsync(User model);

Task<List<User>> GetAllAsync();

Task<User> GetByIdAsync(int id);

Task DeleteUserAsync(int id);

Task ResetPassword(int id);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using AutoMapper;

using Microsoft.EntityFrameworkCore;

using WebNotesApplication.Exceptions;

using WebNotesApplication.Models;

using WebNotesData.Context;

using WebNotesData.Entities;

namespace WebNotesApplication.Services

{

public class NoteService : INoteService

{

private readonly ApplicationContext \_context;

private readonly IMapper \_mapper;

public NoteService(

ApplicationContext context,

IMapper mapper)

{

\_context = context;

\_mapper = mapper;

}

public async Task<Note> GetNote(SearchNoteModel searchNoteModel)

{

var note = await \_context.Notes.AsNoTracking()

.Include(x => x.NoteTags)

.ThenInclude(x => x.Tag)

.Include(x=>x.ShareNotes)

.SingleOrDefaultAsync(n => n.Id == searchNoteModel.Id || n.ShareNotes.Any(s=>s.Link == searchNoteModel.SharedId.ToString()));

if (note is null)

{

throw new AppException($"Note with id = {searchNoteModel.Id} not founded");

}

return note;

}

public async Task<List<Note>> GetNotes(SearchNoteModel searchNoteModel)

{

var notes = await \_context.Notes.AsNoTracking()

.Include(x => x.NoteTags).ThenInclude(x => x.Tag)

.Where(x =>

string.IsNullOrEmpty(searchNoteModel.SearchValue) ||

x.NoteDocument.Contains(searchNoteModel.SearchValue) ||

x.Title.Contains(searchNoteModel.SearchValue) ||

x.NoteTags.Any(n => n.Tag.Value.Contains(searchNoteModel.SearchValue)))

.ToListAsync();

if (!notes.Any())

{

throw new AppException($"Notes with search criteria not founded");

}

return notes;

}

public async Task<Note> CreateNoteAsync(CreateNoteModel createNoteModel)

{

var note = \_mapper.Map<Note>(createNoteModel);

note.CreatedDateTime = DateTime.Now;

await \_context.Notes.AddAsync(note);

await \_context.SaveChangesAsync();

await CreateOrUpdateTags(note, createNoteModel.NoteTags);

// TODO Add NoteTags create and update

return await GetNoteFull(note.Id);// include extend table

}

public async Task<Note> UpdateNoteAsync(UpdateNoteModel updateNoteModel)

{

var note = await \_context.Notes.SingleOrDefaultAsync(n => n.Id == updateNoteModel.Id);

note.NoteDocument = updateNoteModel.NoteDocument;

note.Title = updateNoteModel.Title;

// TODO Add NoteTags create and update

\_context.Notes.Update(note);

await \_context.SaveChangesAsync();

await CreateOrUpdateTags(note, updateNoteModel.NoteTags);

return await GetNoteFull(note.Id);// include extend table

}

public async Task DeleteNoteAsync(int id)

{

var note = await \_context.Notes.SingleOrDefaultAsync(n => n.Id == id);

\_context.Notes.Remove(note);

await \_context.SaveChangesAsync();

}

public async Task<Guid> CreateSharedId(int id)

{

var note = await \_context.Notes.SingleOrDefaultAsync(n => n.Id == id);

var shared = new ShareNote()

{

CreatedDateTime = DateTime.Now,

Link = Guid.NewGuid().ToString(),

NoteId = note.Id

};

await \_context.ShareNotes.AddAsync(shared);

await \_context.SaveChangesAsync();

return Guid.Parse(shared.Link);

}

public async Task<List<string>> Tags(int userId)

{

var tags = \_context.Tags.Where(x => x.UserId == userId || x.UserId == null);

return await tags.Select(t=>t.Value).Distinct().ToListAsync();

}

private async Task CreateOrUpdateTags(Note note, List<string> tags)

{

var existedNoteTags = await \_context.NoteTags

.AsNoTracking()

.Include(x => x.Tag)

.Where(t => t.NoteId == note.Id && (t.Tag.UserId == null || t.Tag.UserId == t.Tag.UserId))

.ToListAsync();

var removeNoteTags = existedNoteTags.Where(t => !tags.Contains(t.Tag.Value));

var addTags = tags.Where(t => existedNoteTags.All(nt => nt.Tag.Value != t)).ToList();

\_context.NoteTags.RemoveRange(removeNoteTags);

await \_context.SaveChangesAsync();

var tagsDb = await \_context.Tags.AsNoTracking().ToListAsync();

var addNewNoteTags = tagsDb.Where(at => addTags.Exists(x => x == at.Value)).ToList();

var addNewTags = addTags.Where(at => addNewNoteTags.All(x => x.Value != at)).ToList();

foreach (var tag in addNewTags)

{

var newTag = new Tag()

{

Value = tag,

UserId = note.UserId

};

await \_context.Tags.AddAsync(newTag);

var newNoteTag = new NoteTag()

{

NoteId = note.Id,

Tag = newTag

};

await \_context.NoteTags.AddAsync(newNoteTag);

await \_context.SaveChangesAsync();

}

foreach (var tag in addNewNoteTags)

{

var existTag = await \_context.Tags.SingleOrDefaultAsync(t => t.Value == tag.Value);

var newNoteTag = new NoteTag()

{

NoteId = note.Id,

TagId = existTag.Id

};

await \_context.NoteTags.AddAsync(newNoteTag);

await \_context.SaveChangesAsync();

}

}

private async Task<Note> GetNoteFull(int id)

{

return await \_context.Notes.AsNoTracking().Include(x => x.NoteTags).ThenInclude(x => x.Tag).SingleOrDefaultAsync(n => n.Id == id);

}

}

}

using AutoMapper;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Options;

using System.Collections.Generic;

using System.Threading.Tasks;

using WebNotesApplication.Authorization;

using WebNotesApplication.Exceptions;

using WebNotesApplication.Models;

using WebNotesData.Context;

using WebNotesData.Entities;

using BCryptNet = BCrypt.Net.BCrypt;

namespace WebNotesApplication.Services

{

public class UserService : IUserService

{

private readonly ApplicationContext \_context;

private readonly IJwtUtils \_jwtUtils;

private readonly AppSettings \_appSettings;

private readonly IMapper \_mapper;

public UserService(

ApplicationContext context,

IJwtUtils jwtUtils,

IOptions<AppSettings> appSettings,

IMapper mapper)

{

\_context = context;

\_jwtUtils = jwtUtils;

\_appSettings = appSettings.Value;

\_mapper = mapper;

}

public async Task<AuthenticateResult> AuthenticateAsync(LoginModel model)

{

var user = await \_context.Users.SingleOrDefaultAsync(x => x.UserName == model.Username);

// validate

if (user == null || !BCryptNet.Verify(model.Password, user.PasswordHash))

{

throw new AppException("Username or password is incorrect");

}

// authentication successful so generate jwt token

var jwtToken = \_jwtUtils.GenerateJwtToken(user);

var authResult = \_mapper.Map<AuthenticateResult>(user);

authResult.Token = jwtToken;

return authResult;

}

public async Task<List<User>> GetAllAsync()

{

var users = await \_context.Users.AsNoTracking().ToListAsync();

return users;

}

public async Task<User> GetByIdAsync(int id)

{

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

throw new KeyNotFoundException("User not found");

}

return user;

}

public Task DeleteUserAsync(int id)

{

throw new System.NotImplementedException();

}

public Task ResetPassword(int id)

{

throw new System.NotImplementedException();

}

public async Task<AuthenticateResult> RegisterAsync(User user)

{

var existUser = await \_context.Users.SingleOrDefaultAsync(x => x.UserName == user.UserName);

if (existUser is not null)

{

throw new AppException($"User with user name {user.UserName} exists");

}

await \_context.Users.AddAsync(user);

await \_context.SaveChangesAsync();

var jwtToken = \_jwtUtils.GenerateJwtToken(user);

var authResult = \_mapper.Map<AuthenticateResult>(user);

authResult.Token = jwtToken;

return authResult;

}

}

}

using WebNotesData.Entities;

namespace WebNotesApplication.Authorization

{

public interface IJwtUtils

{

public string GenerateJwtToken(User user);

public int? ValidateJwtToken(string token);

}

}

using Microsoft.Extensions.Options;

using Microsoft.IdentityModel.Tokens;

using System;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Text;

using WebNotesData.Entities;

namespace WebNotesApplication.Authorization

{

public class JwtUtils : IJwtUtils

{

private readonly AppSettings \_appSettings;

public JwtUtils(IOptions<AppSettings> appSettings)

{

\_appSettings = appSettings.Value;

}

public string GenerateJwtToken(User user)

{

// generate token that is valid for 7 days

var tokenHandler = new JwtSecurityTokenHandler();

var key = Encoding.ASCII.GetBytes(\_appSettings.Secret);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new Claim[]

{

new Claim("id", user.Id.ToString()),

new Claim(ClaimTypes.Role, ((int)user.Role).ToString())

}),

Expires = DateTime.UtcNow.AddDays(7),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

}

public int? ValidateJwtToken(string token)

{

if (token == null)

{

return null;

}

var tokenHandler = new JwtSecurityTokenHandler();

var key = Encoding.ASCII.GetBytes(\_appSettings.Secret);

try

{

tokenHandler.ValidateToken(token, new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false,

// set clockskew to zero so tokens expire exactly at token expiration time (instead of 5 minutes later)

ClockSkew = TimeSpan.Zero

}, out SecurityToken validatedToken);

var jwtToken = (JwtSecurityToken)validatedToken;

var userId = int.Parse(jwtToken.Claims.First(x => x.Type == "id").Value);

// return user id from JWT token if validation successful

return userId;

}

catch

{

// return null if validation fails

return null;

}

}

}

}

using Microsoft.EntityFrameworkCore;

using WebNotesData.Entities;

namespace WebNotesData.Context

{

public class ApplicationContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<Note> Notes { get; set; }

public DbSet<Tag> Tags { get; set; }

public DbSet<NoteTag> NoteTags { get; set; }

public DbSet<ShareNote> ShareNotes { get; set; }

public ApplicationContext(DbContextOptions<ApplicationContext> options)

: base(options)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WebNotesData.Entities

{

public class Note

{

public int Id { get; set; }

[Column(TypeName = "NVARCHAR(MAX)")]

public string Title { get; set; }

[Column(TypeName = "NVARCHAR(MAX)")]

[MaxLength]

public string NoteDocument { get; set; }

[Required]

public DateTime CreatedDateTime { get; set; }

public int UserId { get; set; }

public User User { get; set; }

public List<NoteTag> NoteTags { get; set; }

public bool IsBookmark { get; set; }

public List<ShareNote> ShareNotes { get; set; }

}

}

namespace WebNotesData.Entities

{

public class NoteTag

{

public int Id { get; set; }

public int TagId { get; set; }

public int NoteId { get; set; }

public Tag Tag { get; set; }

public Note Note { get; set; }

}

}

namespace WebNotesData.Entities

{

public enum Role

{

Admin,

User

}

}

using System;

using System.ComponentModel.DataAnnotations;

namespace WebNotesData.Entities

{

public class ShareNote

{

public int Id { get; set; }

[Required]

public string Link { get; set; }

[Required]

public DateTime CreatedDateTime { get; set; }

public DateTime InactivatedDateTime { get; set; }

public Note Note { get; set; }

public int NoteId { get; set; }

}

}

#nullable enable

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace WebNotesData.Entities

{

public class Tag

{

public int Id { get; set; }

[Required]

public string Value { get; set; }

public User? User { get; set; }

public int? UserId { get; set; }

public List<NoteTag> NoteTags { get; set; }

}

}

using System.Collections.Generic;

using System.Text.Json.Serialization;

namespace WebNotesData.Entities

{

public class User

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Email { get; set; }

public string UserName { get; set; }

public Role Role { get; set; }

[JsonIgnore]

public string PasswordHash { get; set; }

public List<Note> Notes { get; set; }

}

}

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.IdentityModel.Logging;

using Microsoft.OpenApi.Models;

using System.Collections.Generic;

using System.Text;

using System.Text.Json.Serialization;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using WebNotesApi.Authorization;

using WebNotesApi.Helpers;

using WebNotesApplication;

using WebNotesApplication.Authorization;

using WebNotesApplication.Services;

using WebNotesData.Context;

using WebNotesData.Entities;

using BCryptNet = BCrypt.Net.BCrypt;

namespace web\_notes\_api

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")), ServiceLifetime.Transient);

services.AddAutoMapper(typeof(Startup));

services.AddCors();

services.AddControllers().AddJsonOptions(x =>

{

// serialize enums as strings in api responses (e.g. Role)

x.JsonSerializerOptions.Converters.Add(new JsonStringEnumConverter());

});

// configure strongly typed settings object

services.Configure<AppSettings>(Configuration.GetSection("AppSettings"));

// configure DI for application services

services.AddScoped<IJwtUtils, JwtUtils>();

services.AddScoped<IUserService, UserService>();

services.AddScoped<INoteService, NoteService>();

services.AddSwaggerGen(opt =>

{

opt.SwaggerDoc("v1", new OpenApiInfo { Title = "My Api", Version = "v1" });

opt.AddSecurityDefinition("bearer", new OpenApiSecurityScheme

{

Type = SecuritySchemeType.Http,

BearerFormat = "JWT",

In = ParameterLocation.Header,

Scheme = "bearer"

});

});

var key = Encoding.ASCII.GetBytes(Configuration.GetSection("AppSettings:secret").Value);

services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.RequireHttpsMetadata = false;

x.SaveToken = true;

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false

};

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env, ApplicationContext context)

{

if (env.IsDevelopment())

{

IdentityModelEventSource.ShowPII = true;

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "WebNotesApi v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

// global cors policy

app.UseCors(x => x

.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader());

// global error handler

app.UseMiddleware<ErrorHandlerMiddleware>();

// custom jwt auth middleware

app.UseMiddleware<JwtMiddleware>();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Примечание* | |
|  | | | | | *Текстовые документы* | | | |  | |
|  | | | | |  | | | |  | |
| *БГУИР ДП 1-40 01 01 069 ПЗ* | | | | | *Пояснительная записка* | | | | *103 с.* | |
|  | | | | | *Отзыв руководителя* | | | |  | |
|  | | | | | *Рецензия* | | | |  | |
|  | | | | | *Отчёт о проверке на* | | | |  | |
|  | | | | | *заимствования* | | | |  | |
|  | | | | | *Справка о внедрении* | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | *Графические документы* | | | |  | |
|  | | | | |  | | | |  | |
| *ГУИР.881074.001 СА* | | | | | *Алгоритм программы от лица* | | | | *Формат А1* | |
|  | | | | | *пользователя на клиентской* | | | |  | |
|  | | | | | *стороне* | | | |  | |
|  | | | | | *Схема алгоритма* | | | |  | |
| *ГУИР.881074.002 СА* | | | | | *Алгоритм регистрации на* | | | | *Формат А1* | |
|  | | | | | *стороне сервера* | | | |  | |
|  | | | | | *Схема алгоритма* | | | |  | |
| *ГУИР.881074.003 СА* | | | | | *Алгоритм проверки доступа на* | | | | *Формат А1* | |
|  | | | | | *стороне сервера* | | | |  | |
|  | | | | | *Схема алгоритма* | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
| *ГУИР.881074.001 ПЛ* | | | | | *Диаграмма вариантов* | | | | *Формат А1* | |
|  | | | | | *использования* | | | |  | |
|  | | | | | *Плакат* | | | |  | |
| *ГУИР.881074.002 ПЛ* | | | | | *Схема базы данных* | | | | *Формат А1* | |
|  | | | | | *Плакат* | | | |  | |
| *ГУИР.881074.003 ПЛ* | | | | | *Диаграмма классов* | | | | *Формат А1* | |
|  | | | | | *Плакат* | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | *БГУИР ДП 1-40 01 01 069 Д1* | | | | | |
|  |  |  |  |  |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| *Разраб.* | | *К.А. Пискарёв* |  |  | *Веб-приложение для управления и хранения пользовательских заметок на языке программирования c#*  *Ведомость дипломного проекта* | Лит. | | | Лист | Листов |
| *Пров.* | | *Д.В. Горбачев* |  |  |  | Т |  | 103 | 103 |
| *Т.контр.* | | *Д.В. Горбачев* |  |  | *ПОИТ, гр. 881074* | | | | |
| *Н.контр.* | | *С.В. Болтак* |  |  |
| *Утв.* | |  |  |  |