

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	4
ВВЕДЕНИЕ	5
ОБОСНОВАНИЕ ТЕМЫ И ПОСТАНОВКА ЗАДАЧИ	8
1.1. Обоснование темы	8
1.1.1. Обоснование актуальности работы	8
1.1.2. Анализ предметной области	9
1.1.3. Анализ существующих решений	11
1.1.4. Выбор средств разработки и языков программирования	14
1.2. Постановка задачи	16
2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	18
2.1. Модель требований к разрабатываемой системе	18
2.1.1. Функциональные требования	18
2.1.2. Требования к информативности	25
2.1.3. Архитектурные требования	41
2.1.4. Требования к поведению	42
2.2. Проектирование системы	43
2.2.1. Архитектура данных и процессов	43
2.2.2. Модель поведения	48
2.2.3. Модель реализации	52
2.2.3.1. Модель исходного кода	52
2.2.3.2. Модель исполняемого кода	53
2.2.3.3. Модель поставляемых артефактов	54
2.2.3.4. Модель размещения артефактов	55
2.2.3.5. Модель управления	56
2.2.3.6. Интерфейс пользователя	57
2.3. Описание технологии разработки клиентской части веб-приложений с использованием фреймворка «Angular»	59
2.4. Реализация программного обеспечения	61
2.4.1 Модуль «Авторизация»	65
2.4.2 Модуль «Статьи»	67
2.4.3 Модуль «Файлы»	68

2.4.4 Модуль «Тесты»	69
2.4.5 Модуль «Дневник»	72
2.4.6 Модуль «Редактирование меню»	73
2.4.7 Модуль «Редактирование пользователей»	75
2.4.8 Модуль «Редактирование статей»	77
2.4.9 Модуль «Редактирование файлов»	78
2.4.10 Модуль «Редактирование тестов»	79
2.4.11 Реализация мобильной версии	80
3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	82
4. ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ	91
4.1. Описание применения	91
4.1.1. Назначение программы	91
4.1.2. Условия применения	91
4.1.3. Описание задачи	91
4.2. Руководство оператора	91
4.2.1. Назначение программы	91
4.2.2. Условия применения программы	92
4.2.3. Выполнение программы	92
4.2.4. Сообщения оператору	96
4.3. Руководство программиста	98
4.3.1. Назначение и условия применения программы	98
4.3.2. Характеристика программы	98
4.3.3. Обращение к программе	99
4.3.4. Сообщения	99
4.4. Руководство системного программиста	99
4.4.1. Общие сведения о программе	99
4.4.2. Структура программы	100
4.4.3. Настройка программы	100
4.4.4. Проверка программы	100
ЗАКЛЮЧЕНИЕ	101
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	102

ОПРЕДЕЛЕНИЯ ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЭИОС – электронная информационно-образовательная среда,

ВУЦ – Военный учебный центр при Рязанском государственном радиотехническом университете имени В. Ф. Уткина,

БД – база данных.

ВВЕДЕНИЕ

Образование является неотъемлемой частью жизни человека. Без образования невозможно представить развитие человека в какой-либо сфере общественной жизни. Технологический прогресс не стоит на месте, поэтому каждая образовательная организация должна повышать качество образования и идти в ногу со временем.

Информационная образовательная среда (ИОС) всегда была основой любой образовательной системы. Изменения в экономической и социальной жизни общества, стремительное развитие информационных технологий, изменения на рынке труда все эти процессы существенно влияют на формирование современной информационной образовательной среды учебного заведения и ее роль в системе образования [8].

Главная цель деятельности любой образовательной организации – обеспечить максимально возможный уровень образования. ВУЦ также является образовательной организацией, поэтому придерживается той же миссии. Ручная обработка информации занимает огромное количество времени сотрудников и может привести к допущению ошибок. Поэтому разработка ЭИОС является актуальной. Для того чтобы облегчить знакомство и получить положительный опыт работы с ЭИОС ей необходим хороший интерфейс, который позволит обеспечить максимально комфортное взаимодействие пользователей с ЭИОС, а также поможет избежать ошибок при работе с ней. Таким образом, разработка веб-интерфейса для ЭИОС является не менее важной задачей чем разработка ЭИОС в целом.

Целью выпускной квалификационной работы является разработка веб-интерфейса для ЭИОС ВУЦ, а также его интеграция с серверной частью ЭИОС.

Для того, чтобы достичь цель выпускной квалификационной работы, необходимо решить следующие задачи:

- проанализировать существующие решения и выявить их недостатки;

- выявить основные сущности, а также пользователей системы и их возможности;
- составить требования к системе;
- разработать архитектуру приложения;
- реализовать отдельные модули и алгоритмы системы, а также пользовательский интерфейс;
- протестировать полученный программный продукт;
- составить документацию к системе.

В результате решения поставленных задач будет разработан простой в понимании и удобный веб-интерфейс для ЭИОС ВУЦ, который в совокупности с серверной частью ЭИОС позволит сосредоточить выполнение образовательных процессов ВУЦ в одной системе, а также автоматизировать их часть. Данное решение позволит сэкономить самый ценный ресурс сотрудников и студентов – время.

Для достижения цели выпускной квалификационной работы были выбраны следующие средства разработки и языки программирования:

- язык программирования – *TypeScript*;
- языка разметки – *HTML*;
- языка таблиц стилей – *Stylus*;
- платформа для разработки веб-приложений – *Angular*;
- среда программирования – *JetBrains WebStorm*;
- инструмент визуального моделирования систем – *StarUML 3*.

В выпускной квалификационной работе представлено 4 основных раздела.

1. Обоснование темы и постановка задачи. Обосновывается актуальность работы, проводится анализ предметной области и выявляются критерии оценки ЭИОС, проводится анализ существующих решений на основе выявленных критериев, описываются выбранные средства разработки и языки программирования, приводится пример порядка внедрения разработки, ставится решаемая задача.

2. Разработка программного обеспечения. Составляются требования к программному обеспечению, проектируется и разрабатывается архитектура приложения, описывается разработка каждого модуля программы, проектируется и реализуется пользовательский интерфейс.
3. Тестирование программного обеспечения. Составляется план тестирования, приводятся и анализируются результаты тестирования.
4. Программная документация. Описывается применения разработанной системы, приводится руководство оператора, руководство программиста и руководство системного программиста.

1. ОБОСНОВАНИЕ ТЕМЫ И ПОСТАНОВКА ЗАДАЧИ

1.1. Обоснование темы

1.1.1. Обоснование актуальности работы

Необходимость разработки ЭИОС для ВУЦ обусловлена современными тенденциями образования и развитием информационных технологий. ЭИОС позволит решать следующие задачи:

- предоставление единого авторизованного доступа к собственным информационным ресурсам (методическим материалам, рабочим программам и т.д.), а также к электронным библиотечным системам и электронным подписанным изданиям, с которыми заключен договор;
- предоставление единого авторизованного доступа к текущей информации об учебном процессе и новостям;
- проведение электронного тестирования студентов, с сохранением результатов в БД.

Автоматизация ряда процессов позволит повысить качество образования и снизить временные затраты преподавателей и сотрудников ВУЦ, а также снизить потребление важного для любой образовательной организации ресурса – бумаги. Например, автоматизация тестирования студентов позволит освободить преподавателей от ручной проверки работ, что сильно сэкономит их время и исключит вероятность ошибки при проверке, а также сведет к минимуму вероятность утери результатов работ студентов, а автоматизация размещения новостей решит проблему донесения необходимой информации до студентов, так как вся информация будет находиться в одном месте.

Ключевым фактором в ЭИОС, как и в любой другой электронной системе, является ее интерфейс, так как именно через него происходит взаимодействие пользователей с системой. Программный интерфейс не только решает проблему взаимодействия с приложением и делает это взаимодействие максимально комфортным, но и позволяет проверять входные данные и сообщать об ошибках в них, тем самым снижая вероятность ошибки при работе с системой. Очень важно разработать

интерфейс, позволяющий при меньшем количестве усилий ознакомиться с возможностями системы и понять принципы работы в ней. Таким образом, без хорошего интерфейса не получится хорошей ЭИОС.

Исходя из сказанного в этом разделе, можно сделать вывод о том, что разработка ЭИОС для ВУЦ, в частности ее веб-интерфейса, является актуальной.

1.1.2. Анализ предметной области

ЭИОС – совокупность электронных информационных ресурсов, электронных образовательных ресурсов, информационных технологий, телекоммуникационных технологий, соответствующих технологических средств, обеспечивающих освоение обучающимися образовательных программ или их частей, а также взаимодействие обучающихся с педагогическим, учебно-вспомогательным персоналом и между собой [9].

Модель AS IS отражает текущее состояние дел в предметной области, в частности, текущую организацию бизнес-процессов.

1. Функциональный аспект.

На данный момент предприятие функционирует следующим образом:

- преподаватели составляют обучающие материалы и документы и сохраняют их в сети предприятия;
- преподаватели лично доносят до студентов различную информацию;
- преподаватели составляют тесты используя старое ПО и сохраняют их в сети организации;
- студенты проходят тесты, составленные преподавателями;
- после прохождения тестов студентами преподаватели вручную собирают и анализируют их результаты.

2. Информационный аспект.

Перечислим все виды информации и носителей, используемых в процессе реализации задач предприятия (таблица 1.1).

Таблица 1.1 – Виды информации и носителей, используемые в процессе реализации задач предприятия

Название	Класс	Носитель	Место хранения
Учебный план	Управляющая	doc-файл	Сеть предприятия
Обучающие материалы и документы	Выходная	doc-файл	Сеть предприятия
Тесты	Выходная	файл	Сеть предприятия
Список студентов и учебных групп	Входная	Бумажный	Журнал
Результаты тестов	Внутренняя	Бумажный	Журнал

ЭИОС должна удовлетворять информационные потребности всех групп пользователей, которые с ней взаимодействуют. Выделим группы пользователей ВУЦ и их потребности:

- студентам необходимо получить доступ к информационным ресурсам, электронным библиотечным системам и новостям организации, также иметь возможность проходить тесты, составленные преподавателями;
- преподаватель не только использует эту среду для ведения образовательной деятельности, но и является участником создания ЭИОС (подготавливает электронные материалы и рабочие программы). Для него основными функциями являются: загрузка файлов (методических материалов, рабочих программ и т.д.), публикация статей, составление тестов и просмотр результатов их прохождения студентами;
- для администратора важно, чтобы система была гибкой и настраиваемой, поэтому от системы требуется динамически

настраиваемое меню и возможность создавать и удалять пользователей, а также управлять их ролями.

Исходя из выявленных требований к функциональности системы можно выделить основные сущности и их характеристики:

- «Контекст пользователя» – уникальный идентификатор, ФИО и роль (Студент, Преподаватель, Администратор);
- «Пункт меню» – заголовок, иконка, дочерние пункты меню (поддержка вложенности), ссылка;
- «Тест» – категория, заголовок и вопросы (сами вопросы состоят из заголовка, типа вопроса, вариантов ответа и дополнительного контента, если это необходимо, например, картинка);
- «Результат теста» – сдан ли тест, количество баллов которые набрал студент, максимально возможное количество баллов, дата прохождения;
- «Файл» – название файла, ссылка на файл, категория файла;
- «Статья» – заголовок, категория, тип (Статья, Новость), дата создания, тело статьи.

Далее сформируем критерии оценки ЭИОС для ВУЦ:

- удобный и простой в понимании интерфейс – влияет на время обучения сотрудников и студентов для возможности использования системы;
- простота настройки – влияет на время, затраченное на поддержку ЭИОС;
- реализация всех описанных ранее функций;
- цена системы.

В данном разделе была проанализирована предметная область, выделены ключевые функции и сущности разрабатываемой ЭИОС.

1.1.3. Анализ существующих решений

Исходя из специфики разрабатываемой ЭИОС, были отобраны и проанализированы несколько решений, описанных далее.

1. *Moodle* – представляет собой систему для управления курсами, предназначенную для облачного обучения. Эта виртуальная обучающая среда может быть настроена с нуля непосредственно под требования заказчика [4]. Анализ системы представлен в таблице 1.2.

Таблица 1.2 – Анализ системы *Moodle*.

Критерий	Описание	Результат
Удобный и простой в понимании интерфейс	Moodle состоит из огромного количества элементов и интерфейс местами очень сильно перегружен и сложен для восприятия. Например, для того, чтобы добавить всего лишь один вопрос к тесту, необходимо заполнить очень большую форму, наполненную, в основном, необязательными полями, которые невозможно скрыть, часть формы показана на рисунке 1.1.	+ -
Простота настройки	Включает в себя огромное количество инструментов, большинство из которых не нужны в разрабатываемой ЭИОС, что сильно усложняет настройку	-
Реализация всех необходимых функций	Реализует все необходимые функции, поддерживает расширение функционала с помощью плагинов. Реализация функций через плагины также является и минусом, так как с каждой функцией придется разбираться отдельно	+ -
Цена системы	Бесплатно	+

Добавление вопроса «В закрытой форме (множественный выбор)»

» Развернуть всё

▼ Общее

Категория

По умолчанию для Schweigepflicht 2020

Название вопроса

?

Текст вопроса

?

Балл по умолчанию

?

1

Общий отзыв к вопросу

?

ID number

?

Один или несколько ответов?

Только один ответ

Случайный порядок ответов

Нумеровать варианты ответов?

a, b, c ...

▼ Ответы

Вариант ответа 1

Оценка

Пусто

Отзыв

Вариант ответа 2

Оценка

Пусто

Отзыв

Рисунок 1.1 – Добавление вопроса к тесту в Moodle.

2. *iSpring* – предлагает комплексное решение для корпоративного онлайн-обучения. В него входит учебный портал *iSpring Learn* и конструктор курсов *iSpring Suite*. Решение ориентировано на быстрый запуск онлайн-обучения [5]. Анализ системы представлен в таблице 1.3.

Таблица 1.3 – Анализ системы *iSpring*.

Критерий	Описание	Результат
Удобный и простой в понимании интерфейс	В <i>iSpring</i> удобный, простой в понимании и красивый интерфейс	+
Простота настройки	Компоненты <i>iSpring</i> легко настраиваются	+
Реализация всех необходимых функций	Из необходимого функционала в <i>iSpring</i> присутствует только тестирование	+/-
Цена системы	Стоимость рассчитывается в зависимости от количества пользователей [2]	-

Таким образом, ни одно из рассмотренных решений не подходит для внедрения в ВУЦ.

1.1.4. Выбор средств разработки и языков программирования

Для разработки клиентской части ЭИОС стоял выбор между языками программирования *TypeScript* и *JavaScript*. Был выбран язык программирования *TypeScript*. *TypeScript* отличается от *JavaScript* возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ. *TypeScript* является обратно совместимым с *JavaScript* и компилируется в последний [14].

В качестве языка разметки был выбран язык *HTML* – стандартизованный язык разметки документов во Всемирной паутине [11].

В качестве языка таблиц стилей был выбран язык *Stylus*. *Stylus* – это препроцессор *CSS*. *Stylus* позволяет использовать унарные и бинарные математические операторы, миксины, функции, переменные, циклы и условия [13].

В качестве фреймворка для разработки клиентской части был выбран *Angular*. *Angular* позволяет создавать так называемые «Одностраничные приложения» или *SPA* (веб-приложения или веб-сайты, использующие единственный *HTML*-документ как оболочку для всех веб-страниц и организующие взаимодействие с пользователем через динамически подгружаемые *HTML*, *CSS* и *JavaScript*) [3].

Преимущества *Angular*:

- инструменты разработчика (*CLI*);
- единая структура проекта;
- поддержка *TypeScript*;
- *dependency injection*;
- шаблоны, основанные на расширении *HTML*;
- кросбраузерная поддержка *HTTP*, *WebSockets*, *Service Workers*;
- динамический роутинг;
- *material design* – библиотека компонентов пользовательского интерфейса.

Одним из основных минусов *Angular* является высоких порог вхождения из-за *Observable* (*RxJS*) и *Dependency Injection* [6].

Для реализации написания статей и составления тестов был выбран модуль для *Angular* – *Ckeditor 5*. *CKEditor* — свободный WYSIWYG-редактор, который может быть использован на веб-страницах [10].

В качестве UI библиотеки была выбрана библиотека *Angular Material*.

В качестве *IDE* была выбрана *JetBrains WebStorm*, так как она поддерживает *Angular*, имеет встроенный отладчик, с ней легко тестировать, имеет хорошую систему подсказок и подсветки кода, обладает гибкой системой настроек и интегрируется с системами контроля версий.

Для визуального моделирования систем на языке *UML* была выбрана программа *StarUML 3*.

В данном разделе были выбраны и описаны средства разработки и языки программирования, выделены их ключевые особенности.

1.2. Постановка задачи

Первоочередной задачей является анализ предметной области и выявление недостатков существующих решений. Не менее важно изучить методы проектирования пользовательских интерфейсов.

Для реализации программной системы необходимо выполнить все описанные далее задачи.

1. Предусмотреть создание, редактирование и изменение пользователей.
Разработать права доступа к различным ресурсам и инструментам системы для каждой роли пользователей (Студент, Преподаватель, Администратор).
2. Разработать динамически редактируемое меню с поддержкой вложений.
3. Продумать и реализовать подсистему управляющую текстами приложения так, чтобы изменение любого текста не требовало перекомпиляции приложения (т.е. хранить тексты в БД).
4. Предусмотреть возможность загрузки файлов в систему и просмотр этих файлов.
5. Реализовать механизм написания статей и блок новостей, в статьях предусмотреть возможность загрузки картинок.
6. Реализовать систему тестирования студентов, результаты прохождения тестов студентами должны сохраняться в БД и быть доступными для

просмотра преподавателям, должно поддерживаться отображение и создание следующих типов вопросов для тестов:

- альтернативный ответ (только 1 правильный ответ);
- множество вариантов (от 0 и до N правильных ответов, где N – количество всех ответов);
- выражение (ввод правильного ответа с клавиатуры);
- установление последовательности (например, расположить события в хронологическом порядке);
- установление соответствия (например, сопоставить даты и события).

По завершению разработки программного продукта необходимо составить план тестирования и протестировать систему, а также составить документацию к системе.

В данном разделе была рассмотрена задача разработки веб-интерфейса ЭИОС для ВУЦ, как совокупность множества подзадач.

2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Модель требований к разрабатываемой системе

2.1.1. Функциональные требования

Прежде чем составить требования к системе, введем некую терминологию.

Студент – студент образовательного учреждения, который проходит тесты, просматривает контент ЭИОС.

Преподаватель – преподаватель образовательного учреждения, который наполняет ЭИОС контентом, следит за результатами прохождения тестов.

Администратор – преподаватель или сотрудник образовательного учреждения, ответственный за работу разрабатываемой системы, имеет полный контроль над системой.

Back End – это внешняя система, серверная часть ЭИОС. Отвечает за сохранение и выдачу контента, подсчет результатов тестов, а также авторизацию. Через эту систему происходит взаимодействие с БД.

Пользователями системы являются студенты и сотрудники ВУЦ. Сгруппируем всех пользователей по потребностям от системы и правам доступа к ней, получим 3 группы пользователей: студент, преподаватель и администратор. Далее будут перечислены возможности каждой из групп пользователей.

1. Администратор.

- a. Вход в систему.
- b. Выход из системы.
- c. Создание пользователей (только преподавателей и студентов).
- d. Редактирование пользователей (только преподавателей и студентов).

- e. Удаление пользователей (только преподавателей и студентов).
 - f. Создание контента.
 - g. Редактирование контента.
 - h. Удаление контента.
 - i. Просмотр контента.
 - j. Создание пункта меню.
 - k. Удаление пункта меню.
 - l. Редактирование пунктов меню.
 - m. Просмотр результатов прохождения тестов.
2. Преподаватель.
 - a. Вход в систему.
 - b. Выход из системы.
 - c. Создание контента.
 - d. Редактирование контента (только своего).
 - e. Удаление контента (только своего).
 - f. Просмотр контента.
 - g. Просмотр результатов прохождения тестов.
 3. Студент.
 - a. Вход в систему.
 - b. Выход из системы.
 - c. Прохождение тестов.
 - d. Просмотр контента.

Исходя из составленного списка возможностей системы, построим диаграмму прецедентов, которая отразит функциональные требования к разработке (рисунок 2.1). В нотации диаграммы прецедентов всего два основных типа сущностей (действующие лица и варианты использования) и три типа отношений (зависимости, ассоциации, обобщения) [1].

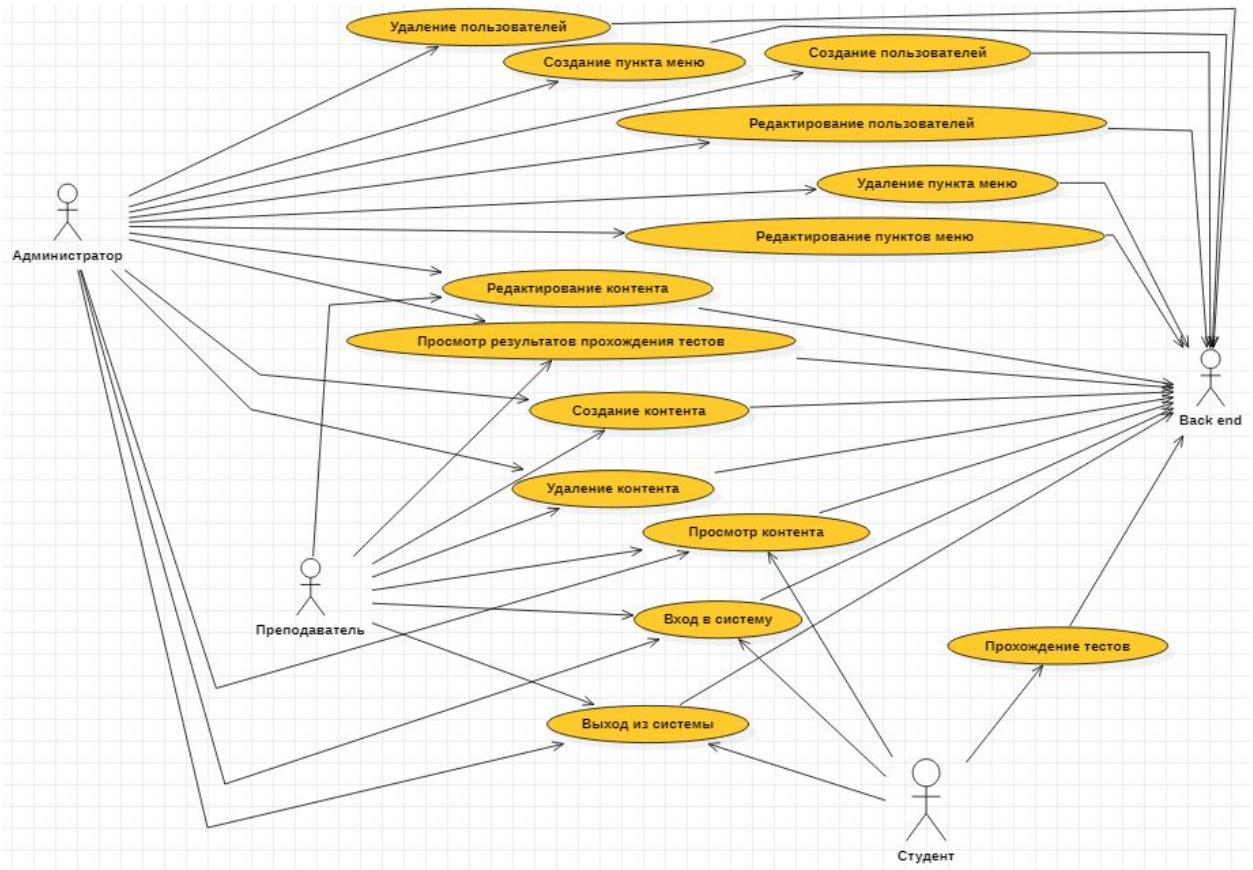


Рисунок 2.1 – Диаграмма прецедентов разрабатываемой системы

Детализируем наиболее важные для работы прецеденты: «Создание контента» (рисунок 2.2), «Вход в систему» (рисунок 2.3), «Прохождение тестов» (рисунок 2.4).

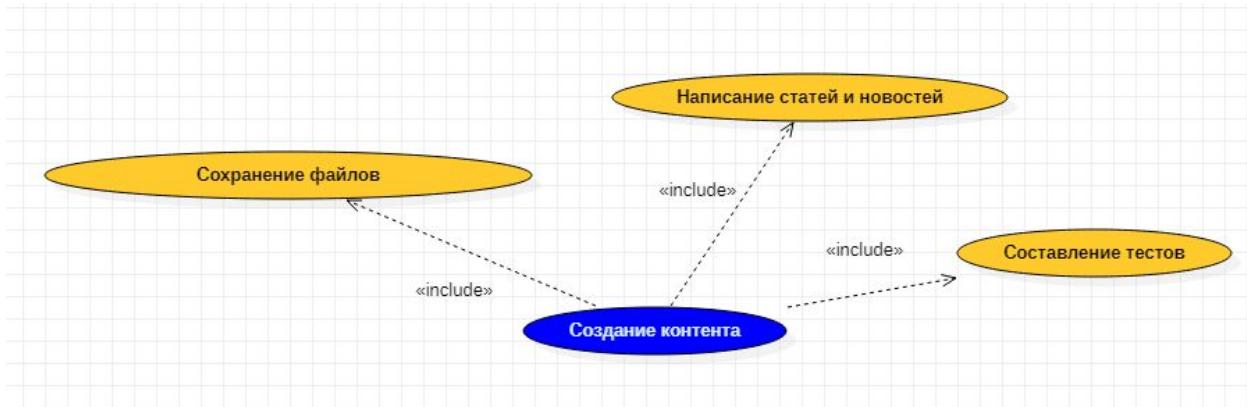


Рисунок 2.2 – Диаграмма прецедентов «Создание контента»

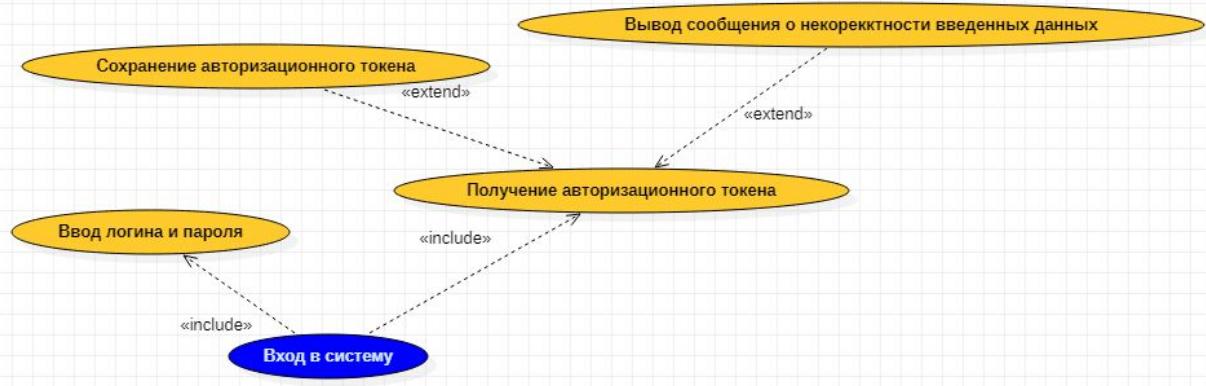


Рисунок 2.3 – Диаграмма прецедентов «Вход в систему»

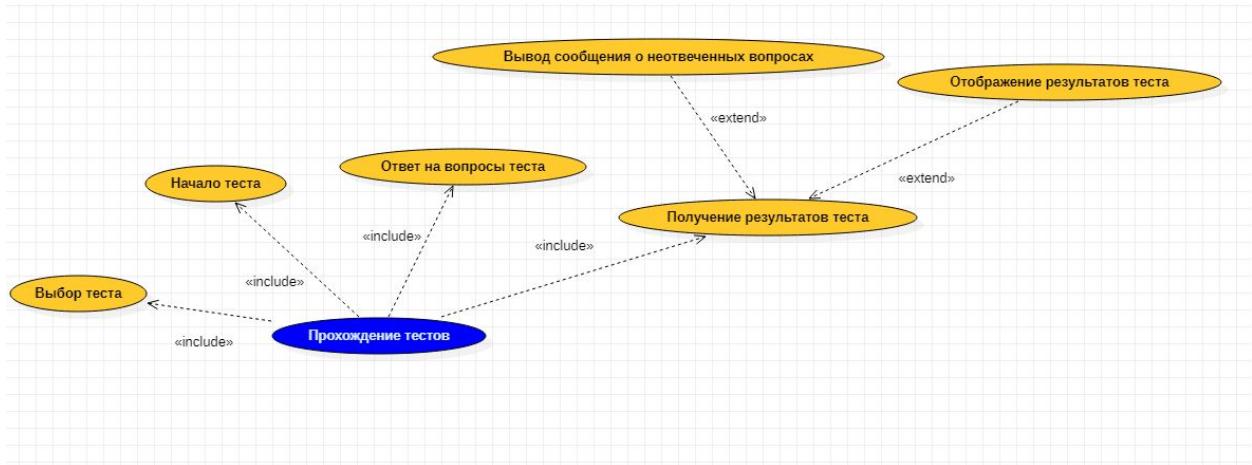


Рисунок 2.4 – Диаграмма прецедентов «Прохождение тестов»

Ниже приведена спецификация диаграммы прецедентов разрабатываемой системы:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 4 актора;
- 14 прецедентов (все базисные);
- 38 отношений.

2. Характеристика акторов.

Актор с ролью «Студент» представляет собой пользователя системы. Использует следующие доступные ему функции системы: «Вход в систему», «Выход из системы», «Прохождение тестов», «Просмотр контента».

Актор с ролью «Преподаватель» представляет собой пользователя системы. Использует следующие доступные ему функции: «Вход в систему», «Выход из системы», «Создание контента», «Редактирование контента», «Удаление контента», «Просмотр контента», «Просмотр результатов прохождения тестов».

Актор с ролью «Администратор» представляет собой пользователя системы. Использует все возможные функции системы: «Вход в систему», «Выход из системы», «Создание пользователей», «Редактирование пользователей», «Удаление пользователей», «Создание контента», «Редактирование контента», «Удаление контента», «Просмотр контента», «Создание пункта меню», «Удаление пункта меню», «Редактирование пунктов меню», «Просмотр результатов прохождения тестов».

Актор с ролью «*Back End*» представляет собой внешнюю систему, с помощью нее происходит работа с БД ЭИОС, а также подсчет результатов прохождения тестов.

3. Характеристика прецедентов.

Для некоторых, особо значимых прецедентов были составлены спецификации: «Вход в систему» (таблица 2.1), «Прохождение тестов» (таблица 2.2).

Таблица 2.1 – Спецификация прецедента «Вход в систему»

Название:	Вход в систему
Действующие лица:	Администратор, преподаватель, студент, система, <i>Back End</i> .
Краткое описание:	Система позволяет пользователю авторизоваться.

Предусловия:	Неавторизованный пользователь открыл систему.
Постусловия:	Система сохраняет авторизационный токен пользователя и предоставляет ему доступ к положенным ролью возможностям.
Основной поток (нормальное течение):	<ol style="list-style-type: none"> Пользователь вводит «Логин». Пользователь вводит «Пароль». Система отправляет в <i>Back end</i> эти данные, в ответ получает авторизационный токен. Система сохраняет полученный токен. Выполнение прецедента завершается.
Альтернативный поток (альтернативные течения):	<p>A1. [Пользователь не существует]</p> <ol style="list-style-type: none"> Система выводит сообщение о том, что пользователя с такими данными не существует. Поток возвращается к этапу 1 основного потока. <p>A2. [<i>Back end</i> не доступен]</p> <ol style="list-style-type: none"> Система выводит сообщение о том, что временно недоступна. Выполнение прецедента завершается.
Приоритет (Критично Важно Желательно):	Критично
Частота использования (Всегда Часто Иногда Редко Один раз):	Редко

Таблица 2.2 – Спецификация прецедента «Прохождение тестов»

Название:	Прохождение тестов
Действующие лица:	Студент, система, <i>Back End</i> .
Краткое описание:	Система позволяет пользователю пройти тест.
Предусловия:	Студенту необходимо пройти тест.
Постусловия:	Система отправляет ответы пользователя в <i>Back end</i> .
Основной поток (нормальное течение):	<ol style="list-style-type: none"> 1. Студент выбирает тест. 2. Студент начинает прохождение теста, система уведомляет об этом <i>Back end</i>. 3. Студент отвечает на вопросы теста. 4. Система отправляет в <i>Back end</i> ответы на вопросы теста, в ответ получает результат прохождения теста. 5. Система выводит результат прохождения теста. 6. Выполнение прецедента завершается.
Альтернативный поток (альтернативные течения):	<p>A1. [Не все вопросы отвечены]</p> <ol style="list-style-type: none"> 1. Система выводит сообщение о том, необходимо ответить на все вопросы теста для его завершения. 2. Поток возвращается к этапу 2 основного потока. <p>A2. [<i>Back end</i> не доступен]</p> <ol style="list-style-type: none"> 1. Система выводит сообщение о том, что временно недоступна.

	2. Выполнение прецедента завершается.
Приоритет (Критично Важно Желательно):	Критично
Частота использования (Всегда Часто Иногда Редко Один раз):	Часто

В данном разделе были выявлены и описаны (с помощью диаграмм прецедентов) требования к разрабатываемой системе.

2.1.2. Требования к информативности

Перед тем как описать требования к информативности, введем некую терминологию.

Фронтенд (англ. *front-end*) — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

Бэкенд (англ. *back-end*) — программно-аппаратная часть сервиса. Фронт- и бэкенд — это вариант архитектуры программного обеспечения.

Термины появились в программной инженерии вследствие развития принципа разделения ответственности между внешним представлением и внутренней реализацией. Back-end для предоставления своей функции реализует API, которое использует front-end.

Таким образом front-end разработчику не нужно знать особенностей реализации сервера, а back-end разработчику — реализацию front-end [7].

Разрабатываемая система (*Front End*) будет взаимодействовать с Back End с помощью *HTTP* запросов, ответом сервера будет являться объект, сериализованный в *JSON* (текстовый формат обмена данными, основанный на JavaScript [12]).

На основе поставленной задачи (пункт 1.2.), выделим классы-данные, объекты которых будут являться соглашениями между *Front End* и *Back End* и передаваться между ними (рисунки 2.5, 2.6, 2.7, 2.8).

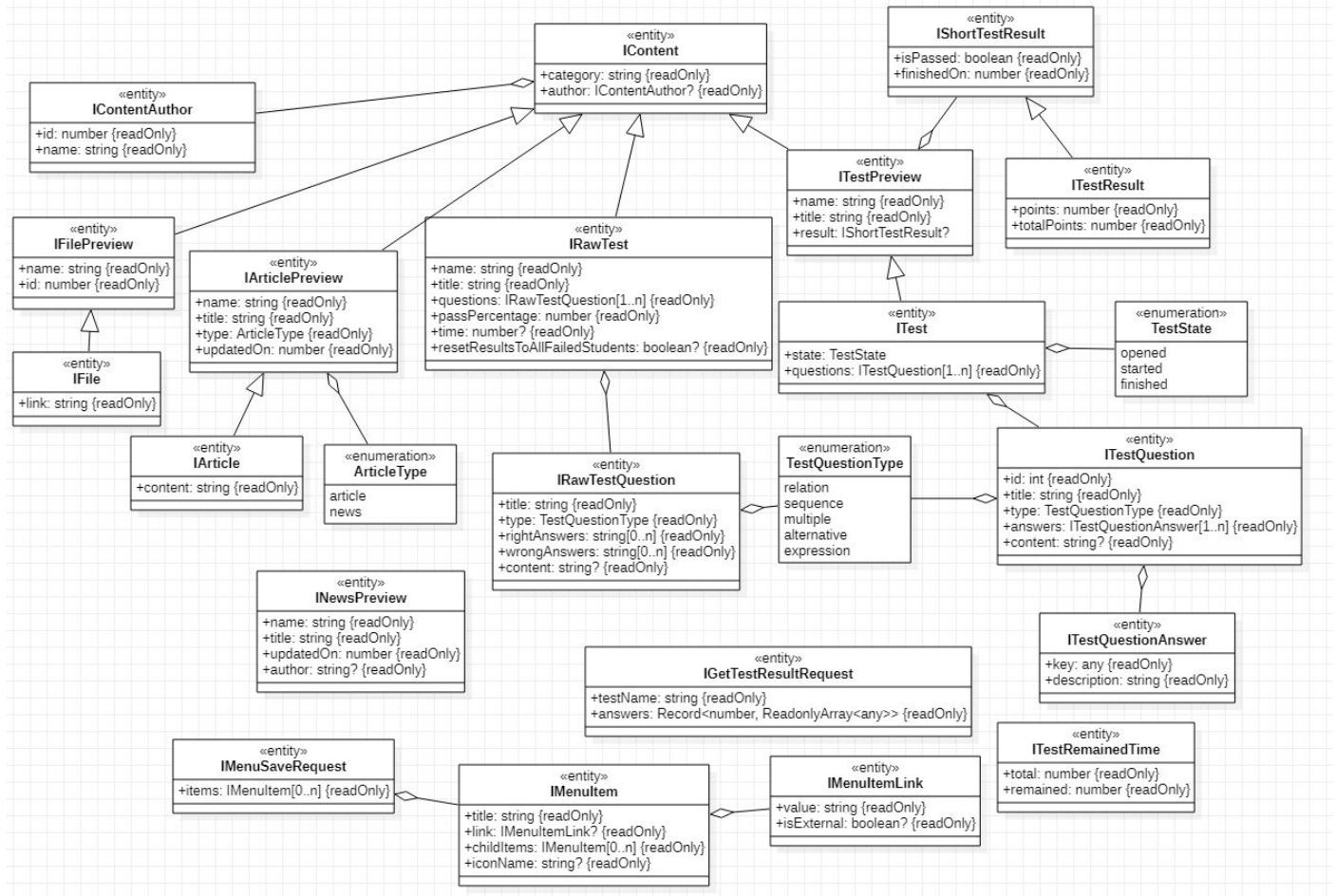


Рисунок 2.5 – Диаграмма классов контента

Спецификация диаграммы классов контента:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 20 классов;
- 3 перечисления.

2. Характеристика классов.

Класс `«IContentAuthor»` представляет собой информацию об авторе контента и содержит следующие атрибуты:

- `«id»` – уникальный идентификатор пользователя, создавшего контент;

- «*name*» – имя пользователя, создавшего контент.

Класс «*IContent*» является базовым классом для классов представляющих собой контентное наполнение ЭИОС, содержит следующие атрибуты:

- «*category*» – категория контента, предназначена для группировки контента в логические группы;
- «*author*» – информация об авторе контента.

Класс «*IFilePreview*» является наследником класса «*IContent*» и содержит следующие атрибуты:

- «*name*» – название файла;
- «*id*» – уникальный идентификатор файла.

Класс «*IFile*» является наследником класса «*IFilePreview*» и содержит следующие атрибуты:

- «*link*» – ссылка на открытие файла.

Перечисление «*ArticleType*» может принимать следующие значения:

- «*article*» – стандартный тип статьи;
- «*news*» – означает, что статья является новостью и будет отображена в блоке новостей.

Класс «*IArticlePreview*» является наследником класса «*IContent*» и содержит следующие атрибуты:

- «*name*» – транслитерированный заголовок статьи, является уникальным для каждой статьи, с помощью этого атрибута будет формироваться ссылка на статью в системе;
- «*title*» – заголовок статьи;
- «*type*» – тип статьи;
- «*updatedOn*» – дата обновления статьи (в формате *Unix Timestamp*).

Класс «*IArticle*» является наследником класса «*IArticlePreview*» и содержит следующие атрибуты:

- «*content*» – тело статьи в формате *HTML*.

Класс «*INewsPreview*» представляет собой краткую информацию о новости и содержит следующие атрибуты:

- «*name*» – имя новости (для формирования ссылки на полную новость);
- «*title*» – заголовок новости;
- «*updatedOn*» – дата обновления новости (в формате *Unix Timestamp*);
- «*author*» – имя автора новости.

Класс «*IShortTestResult*» представляет собой краткую информацию о результате прохождения теста и содержит следующие атрибуты:

- «*isPassed*» – флаг, указывающий на то сдан ли тест;
- «*finishedOn*» – дата прохождения теста (в формате *Unix Timestamp*).

Класс «*ITestResult*» является наследником класса «*IShortTestResult*», представляет собой полную информацию о результате прохождения теста и содержит следующие атрибуты:

- «*points*» – максимально возможное количество «очков», которые можно набрать в пройденном тесте;
- «*totalPoints*» – набранное количество «очков».

Класс «*ITestPreview*» является наследником класса «*IContent*» и содержит следующие атрибуты:

- «*name*» – транслитерированный заголовок теста, является уникальным для каждого теста, с помощью этого атрибута будет формироваться ссылка на тест в системе;
- «*title*» – заголовок теста;
- «*result*» – краткая информация о результате прохождения теста (если тест уже был пройден).

Перечисление «*TestState*» может принимать следующие значения:

- «*opened*» – значение по умолчанию, означает что студент еще не начинал тест;
- «*started*» – означает, что студент начал проходить тест;
- «*finished*» – означает, что студент завершил прохождение теста.

Класс «*ITest*» является наследником класса «*ITestPreview*» и содержит следующие атрибуты:

- «*state*» – текущее состояние теста.
- «*questions*» – вопросы теста.

Класс «*IRawTest*» является наследником класса «*IContent*», представляет собой модель теста, использующуюся при его создании и редактировании, и содержит следующие атрибуты:

- «*name*» – транслитерированный заголовок теста, является уникальным для каждого теста, с помощью этого атрибута будет формироваться ссылка на тест в системе;
- «*title*» – заголовок теста;
- «*questions*» – вопросы теста;
- «*passPercentage*» – количество процентов правильных ответов, при достижении которого считает, что студент сдал тест;
- «*time*» – время на прохождение теста в минутах;
- «*resetResultsToAllFailedStudents*» – флаг, указывающий на то, что необходимо дать возможность всем не сдавшим тест студентам пройти его еще раз.

Перечисление «*TestQuestionType*» может принимать следующие значения:

- «*relation*» – означает, что в вопросе необходимо установить соответствия;
- «*multiple*» – означает, что в вопросе необходимо выбрать несколько вариантов ответа;
- «*alternative*» – означает, что в вопросе необходимо выбрать только один вариант ответа;

- «*sequence*» – означает, что в вопросе необходимо установить ответы в правильной последовательности;
- «*expression*» – означает, что правильный ответ необходимо ввести с клавиатуры.

Класс «*ITestQuestion*» представляет собой вопрос теста и содержит следующие атрибуты:

- «*id*» – уникальный идентификатор вопроса в системе;
- «*title*» – заголовок вопроса;
- «*type*» – тип вопроса;
- «*answers*» – возможные ответы на вопрос;
- «*content*» – дополнительный контент вопроса в формате *HTML* (необязательный атрибут), будет использоваться, например, в случае необходимости добавления в вопрос поясняющей картинки.

Класс «*ITestQuestionAnswer*» представляет собой возможный ответ на вопрос теста и содержит следующие атрибуты:

- «*key*» – идентификатор ответа на вопрос, с помощью которого будет устанавливаться какие ответы выбрал пользователь;
- «*description*» – пояснительный текст ответа на вопрос.

Класс «*IRawTestQuestion*» представляет собой модель вопроса теста, использующуюся при его создании и редактировании, содержит следующие атрибуты:

- «*title*» – заголовок вопроса;
- «*type*» – тип вопроса;
- «*rightAnswers*» – правильные ответы на вопрос;
- «*wrongAnswers*» – неправильные ответы на вопрос;
- «*content*» – дополнительный контент вопроса в формате *HTML*.

Класс «*ITestRemainedTime*» представляет собой время на прохождение теста и содержит следующие атрибуты:

- «*total*» – полное время на прохождение теста в минутах;

- «*remained*» – оставшееся время на прохождение теста в минутах.

Класс «*IGetTestResultRequest*» представляет собой тело запроса, отправляющегося в *Back end* при завершении теста, содержит следующие атрибуты:

- «*testName*» – название теста;
- «*answers*» – ответы пользователя на вопросы теста.

Класс «*MenuItem*» представляет собой пункт меню и содержит следующие атрибуты:

- «*title*» – заголовок пункта меню;
- «*iconName*» – название иконки;
- «*link*» – ссылка на страницу или раздел, на которые указывает пункт меню;
- «*childItems*» – подпункты меню.

Класс «*MenuItemLink*» представляет собой ссылку пункта меню и содержит следующие атрибуты:

- «*value*» – значение ссылки;
- «*isExternal*» – флаг, который указывает на то, является ли ссылка внешней.

Класс «*MenuSaveRequest*» представляет тело запроса, который отправляется в *Back end* при сохранении меню и содержит следующие атрибуты:

- «*items*» – пункты меню.

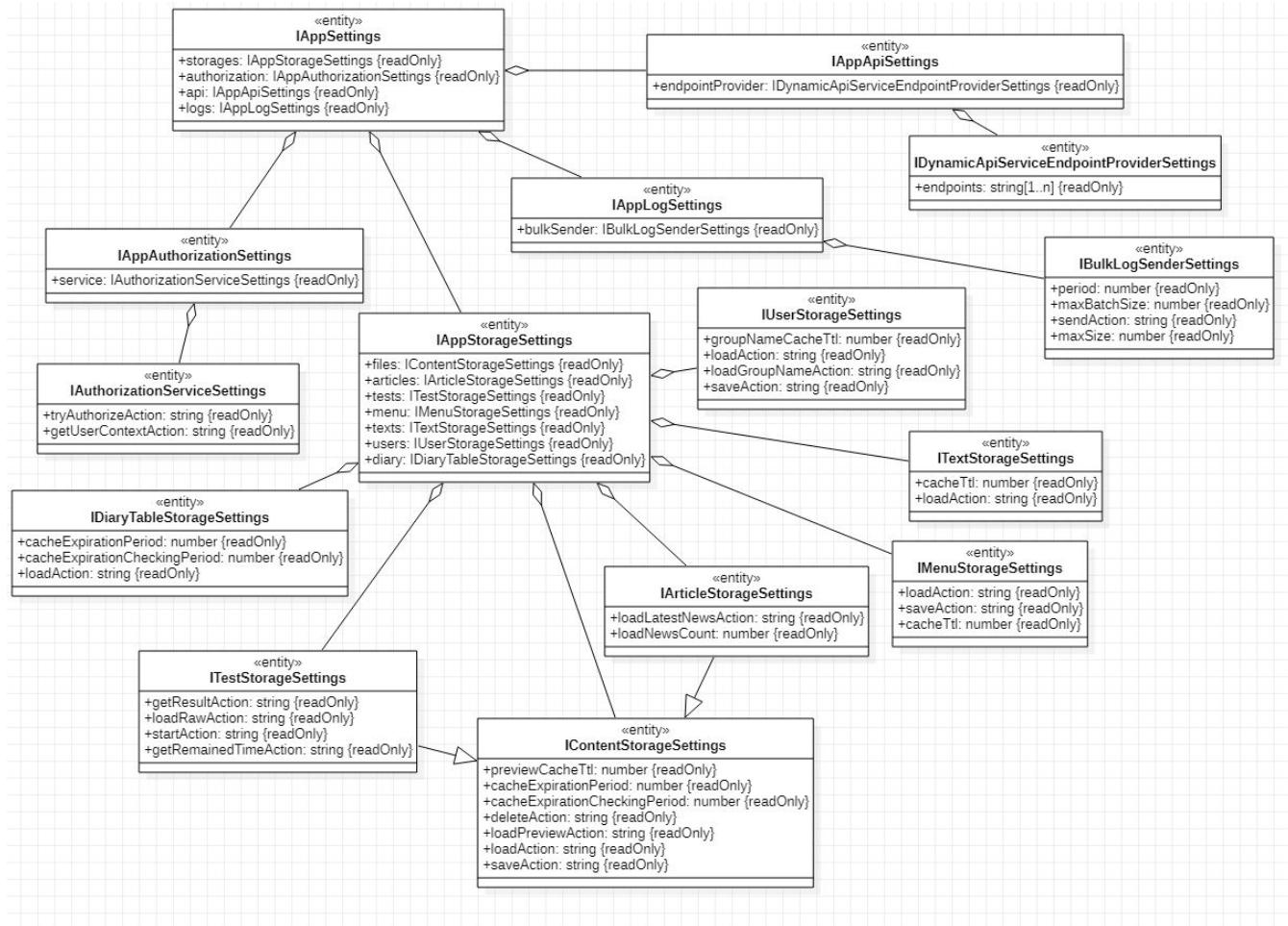


Рисунок 2.6 – Диаграмма классов настроек приложения.

Спецификация диаграммы классов настроек приложения:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 15 классов;

2. Характеристика классов.

Класс `«IAppSettings»` представляет собой все настройки приложения и содержит следующие атрибуты:

- `«storages»` – настройки для сервисов, отвечающих за хранение какой-либо информации;
- `«authorization»` – настройки авторизации;
- `«api»` – настройки взаимодействия с *Back end*;
- `«logs»` – настройки логирования.

Класс «*IAppApiSettings*» представляет собой настройки взаимодействия с *Back end* и содержит следующие атрибуты:

- «*endpointProvider*» – настройки сервиса, осуществляющего подбор хоста *Back end*.

Класс «*IDynamic ApiServiceEndpointProviderSettings*» представляет собой настройки сервиса, осуществляющего подбор домена *Back end* и содержит следующие атрибуты:

- «*endpoints*» – список хостов.

Класс «*IAppLogSettings*» представляет собой настройки логирования и содержит следующие атрибуты:

- «*bulkSender*» – настройки сервиса, который отправляет логи приложения пачками.

Класс «*IBulkLogSenderSettings*» представляет собой настройки сервиса, который отправляет логи приложения пачками и содержит следующие атрибуты:

- «*period*» – период отправки логов;
- «*maxBatchSize*» – размер пачки логов, при котором отправление происходит раньше запланированного времени;
- «*sendAction*» – путь к методу контроллера, отвечающему за прием логов;
- «*maxSize*» – максимальный размер накопления логов, если логов количество превысит этот лимит, то они будут игнорироваться.

Класс «*IAppAuthorizationSettings*» представляет собой настройки авторизации и содержит следующие атрибуты:

- «*service*» – настройки сервиса авторизации.

Класс «*IAuthServiceSettings*» представляет собой настройки сервиса авторизации и содержит следующие атрибуты:

- «*tryAuthorizeAction*» – путь к методу контроллера, отвечающему за авторизацию;
- «*getUserContextAction*» – путь к методу контроллера, отвечающему за подгрузку контекста пользователя.

Класс «*IAppStorageSettings*» представляет собой настройки для сервисов, отвечающих за хранение какой-либо информации и содержит следующие атрибуты:

- «*files*» – настройки хранилища файлов;
- «*articles*» – настройки хранилища статей;
- «*tests*» – настройки хранилища тестов;
- «*menu*» – настройки хранилища пунктов меню;
- «*texts*» – настройки хранилища текстов;
- «*users*» – настройки хранилища пользователей;
- «*diary*» – настройки хранилища электронного дневника.

Класс «*IContentStorageSettings*» представляет собой базовый класс для настроек хранилищ контента и содержит следующие атрибуты:

- «*previewCacheTtl*» – время жизни краткой информации о контенте в секундах;
- «*cacheExpirationPeriod*» – период обновления кэша полных объектов контента;
- «*cacheExpirationCheckingPeriod*» – период обновления элементов кэша полных объектов контента;
- «*deleteAction*» – путь к методу контроллера, отвечающему за удаление контента;
- «*loadPreviewAction*» – путь к методу контроллера, отвечающему за загрузку краткой информации о контенте;
- «*loadAction*» – путь к методу контроллера, отвечающему за загрузку полного объекта контента;
- «*saveAction*» – путь к методу контроллера, отвечающему за сохранение контента.

Класс «*IArticleStorageSettings*» является наследником класса «*IContentStorageSettings*», представляет собой настройки хранилища статей и содержит следующие атрибуты:

- «*loadLatestNewsAction*» – путь к методу контроллера, отвечающему за загрузку последних новостей;
- «*loadNewsCount*» – количество новостей для загрузки.

Класс «*ITestStorageSettings*» является наследником класса «*IContentStorageSettings*», представляет собой настройки хранилища тестов и содержит следующие атрибуты:

- «*getActionResult*» – путь к методу контроллера, отвечающему за завершение теста;
- «*loadRawAction*» – путь к методу контроллера, отвечающему за загрузку теста для редактирования;
- «*startAction*» – путь к методу контроллера, отвечающему за начало прохождения теста;
- «*getRemainedTimeAction*» – путь к методу контроллера, отвечающему за загрузку времени, оставшегося на прохождение теста.

Класс «*IDiaryStorageSettings*» представляет собой настройки хранилища электронного дневника и содержит следующие атрибуты:

- «*cacheExpirationPeriod*» – период обновления кэша таблиц дневника;
- «*cacheExpirationCheckingPeriod*» – период обновления элементов кэша таблиц дневника;
- «*loadAction*» – путь к методу контроллера, отвечающему за загрузку таблицы дневника.

Класс «*IUserStorageSettings*» представляет собой настройки хранилища пользователей и содержит следующие атрибуты:

- «*groupNameCacheTtl*» – период обновления элементов кэша названий студенческих групп;
- «*loadAction*» – путь к методу контроллера, отвечающему за загрузку пользователей для редактирования;

- «*loadGroupNameAction*» – путь к методу контроллера, отвечающему за загрузку списки студенческих групп;
- «*saveAction*» – путь к методу контроллера, отвечающему за сохранение пользователей.

Класс «*ITextStorageSettings*» представляет собой настройки хранилища текстов и содержит следующие атрибуты:

- «*cacheTtl*» – период обновления элементов кэша с текстами;
- «*loadAction*» – путь к методу контроллера, отвечающему за загрузку текстов.

Класс «*IMenuStorageSettings*» представляет собой настройки хранилища пунктов меню и содержит следующие атрибуты:

- «*cacheTtl*» – период обновления элементов кэша с пунктами меню;
- «*loadAction*» – путь к методу контроллера, отвечающему за загрузку пунктов меню;
- «*saveAction*» – путь к методу контроллера, отвечающему за сохранение пунктов меню.

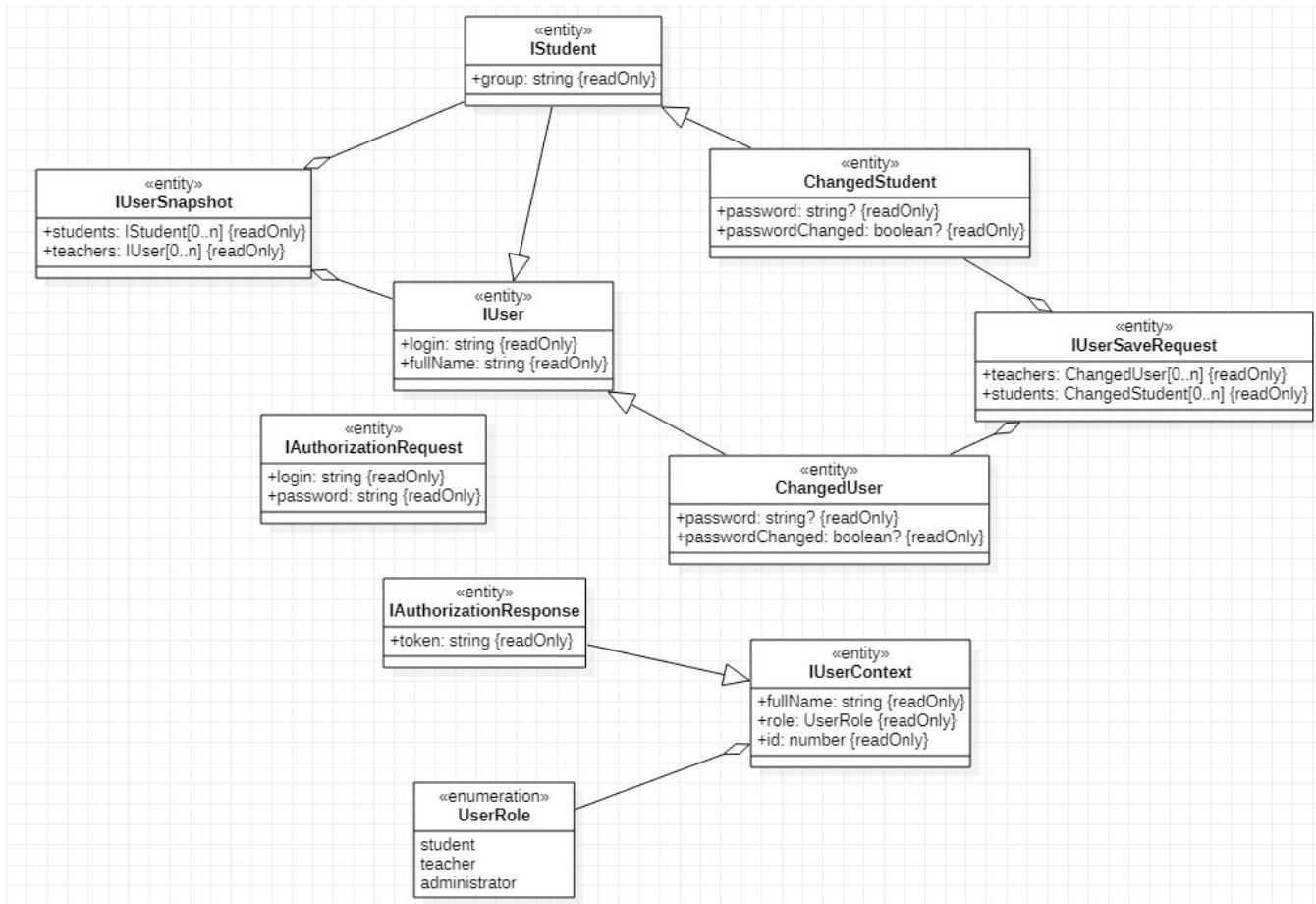


Рисунок 2.7 – Диаграмма классов, относящихся к пользователям

Спецификация диаграммы классов, относящихся к пользователям:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 9 классов;
- 1 перечисление.

2. Характеристика классов.

Перечисление «*UserRole*» может принимать следующие значения:

- «*student*» – означает, что пользователь является студентом;
- «*teacher*» – означает, что пользователь является преподавателем;
- «*administrator*» – означает, что пользователь является администратором.

Класс «*IUserContext*» представляет собой контекст пользователя и содержит следующие атрибуты:

- «*fullName*» – ФИО пользователя;
- «*role*» – роль пользователя (Студент, Преподаватель, Администратор);
- «*id*» – уникальный идентификатор пользователя.

Класс «*IAuthorizationResponse*» является наследником класса «*IUserContext*», представляет собой данные ответа *Back end* на авторизационный запрос и содержит следующие атрибуты:

- «*token*» – авторизационный токен.

Класс «*IAuthorizationRequest*» представляет авторизационный запрос и содержит следующие атрибуты:

- «*login*» – логин пользователя;
- «*password*» – пароль пользователя.

Класс «*IUser*» представляет модель пользователя для редактирования и содержит следующие атрибуты:

- «*login*» – логин пользователя;
- «*fullName*» – ФИО пользователя.

Класс «*IStudent*» является наследником класса «*IUser*», представляет модель студента для редактирования и содержит следующие атрибуты:

- «*group*» – учебная группа студента.

Класс «*IUserSnapshot*» представляет собой модель всех редактируемых пользователей и содержит следующие атрибуты:

- «*students*» – список студентов для редактирования;
- «*teachers*» – список преподавателей для редактирования.

Класс «*ChangedStudent*» является наследником класса «*IStudent*», представляет собой сохраняемую модель студента и содержит следующие атрибуты:

- «*password*» – новый пароль пользователя;
- «*passwordChanged*» – флаг, указывающий на то что пароль был изменен.

Класс «*ChangedUser*» является наследником класса «*IUser*», представляет собой сохраняемую модель пользователя и содержит следующие атрибуты:

- «*password*» – новый пароль пользователя;
- «*passwordChanged*» – флаг, указывающий на то что пароль был изменен.

Класс «*IUserSaveRequest*» представляет собой тело запроса на сохранение пользователей и содержит следующие атрибуты:

- «*students*» – список студентов;
- «*teachers*» – список преподавателей.

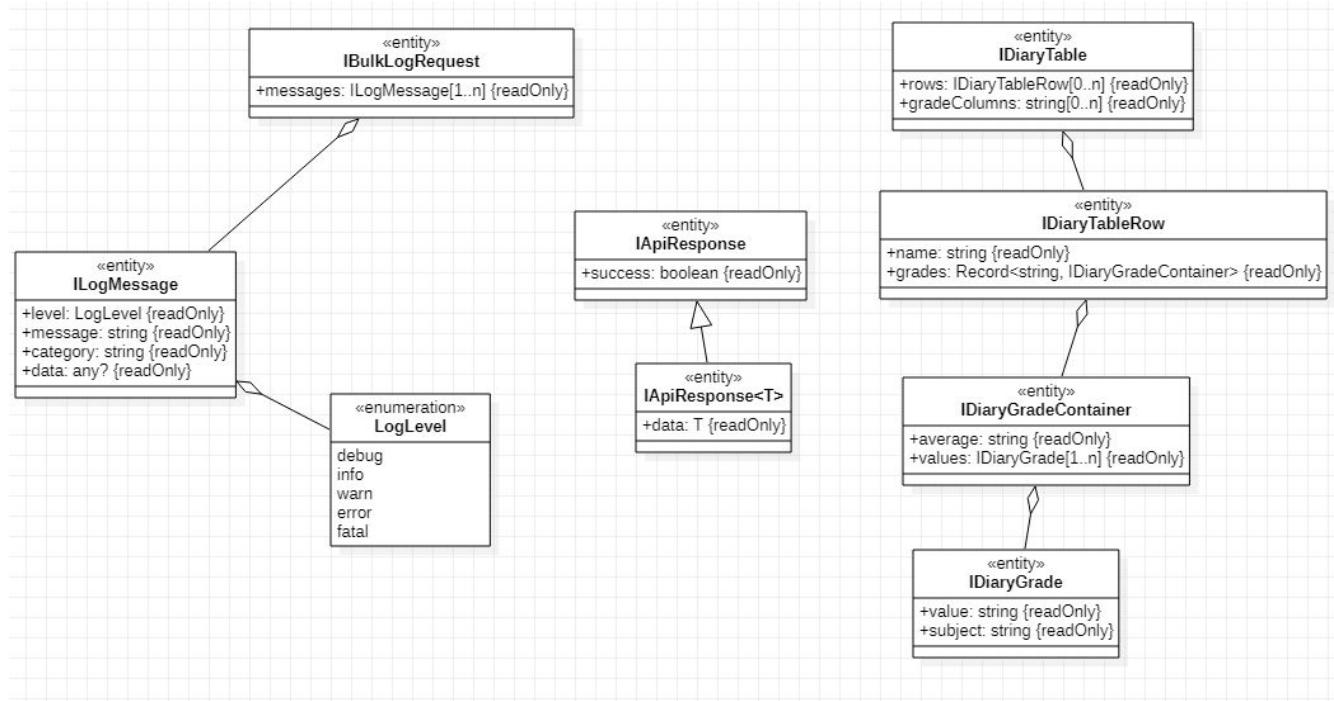


Рисунок 2.8 – Диаграмма классов логирования, электронного дневника и общих классов

Спецификация диаграммы классов, относящихся к пользователям:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 8 классов;
- 1 перечисление.

2. Характеристика классов

Класс «*IApiResponse*» представляет базовый класс для всех ответов на запросы в *Back end*, во время которых выполняется какое то действие и содержит следующие атрибуты:

- «*success*» – флаг, указывающий на успешность выполнения действия.

Класс «*IApiResponse<T>*» является наследником класса «*IApiResponse*» и используется в ответах на запросы в *Back end*, во время которых помимо выполнения действия необходимо получить какие то данные, содержит следующие атрибуты:

- «*data*» – запрашиваемые данные.

Перечисление «*LogLevel*» может принимать следующие значения:

- «*debug*» – отладочный уровень;
- «*info*» – информационный уровень;
- «*warn*» – не серьезная ошибка;
- «*error*» – ошибка;
- «*fatal*» – фатальная ошибка.

Класс «*ILogMessage*» представляет собой сообщение для журнала сообщений и содержит следующие атрибуты:

- «*level*» – уровень сообщения;
- «*message*» – сообщение;
- «*category*» – категория сообщения;
- «*data*» – дополнительные данные.

Класс «*IBulkLogRequest*» представляет тело запроса с логами и содержит следующие атрибуты:

- «*messages*» – сообщения.

Класс «*IDiaryGrade*» представляет оценку электронного дневника и содержит следующие атрибуты:

- «*value*» – значение оценки;
- «*subject*» – то, за что была получена оценка.

Класс «*IDiaryGradeContainer*» представляет агрегированную оценку электронного дневника и содержит следующие атрибуты:

- «*average*» – среднее значение из всех оценок;
- «*values*» – оценки.

Класс «*IDiaryTableRow*» представляет собой строку таблицы электронного дневника и содержит следующие атрибуты:

- «*name*» – имя строки дневника (имя студента);
- «*grades*» – оценки, относящиеся к этой строке.

Класс «*IDiaryTable*» представляет собой строку таблицы электронного дневника и содержит следующие атрибуты:

- «*rows*» – строки таблицы;
- «*gradeColumns*» – список всех колонок (числа в месяце).

В данном разделе были выявлены и описаны (с помощью диаграммы классов) требования к информативности разрабатываемой системы.

2.1.3. Архитектурные требования

В качестве базовой архитектуры разрабатываемой ЭИОС будет использоваться многозвенная сетевая клиент-серверная архитектура с «толстым» клиентом (управление отображением и часть логики перенесены на клиентскую часть).

На рисунке 2.9 представлена диаграмма развертывания уровня типов для разрабатываемой ЭИОС.

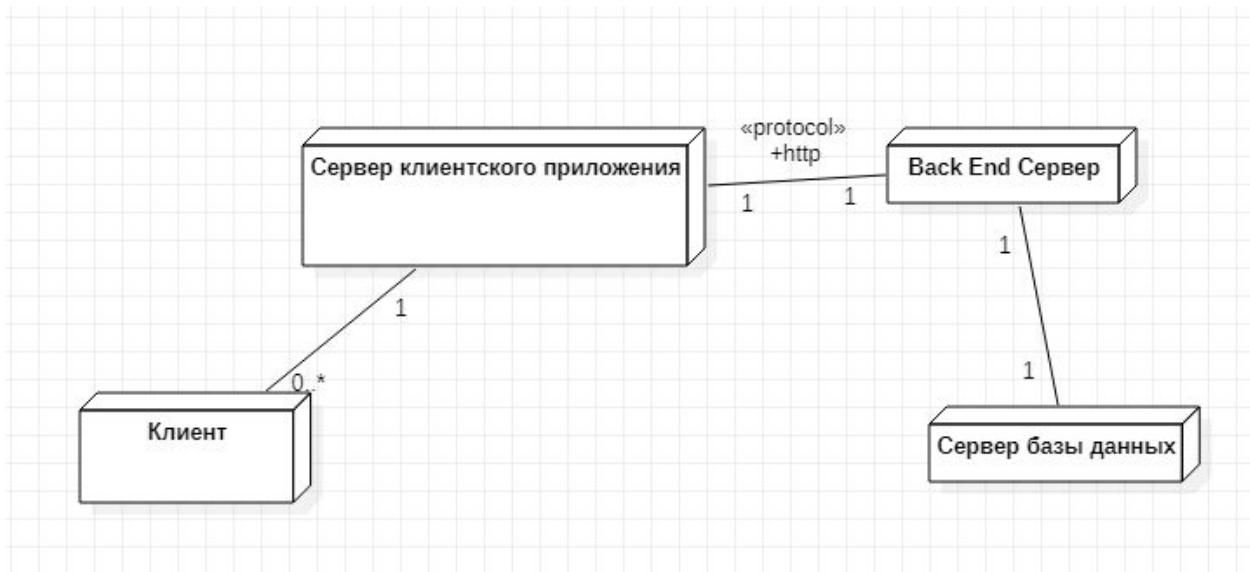


Рисунок 2.9 – Диаграмма развертывания уровня типов для разрабатываемой системы

Спецификация диаграммы развертывания уровня типов:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 4 узла;
- 3 связи.

2. Спецификация связей.

Узел «Клиент» взаимодействует с узлом «Сервер клиентского приложения».

Узел «Сервер клиентского приложения» взаимодействует с узлом «Back End сервер» используя протокол *HTTP*.

Узел «Back End Сервер» взаимодействует с узлом «Сервер базы данных».

В данном разделе были описаны требования к базовой архитектуре разрабатываемой системы.

2.1.4. Требования к поведению

Выделим основные страницы разрабатываемого веб-приложения и наиболее важные переходы между ними. На рисунке 2.10 представлен граф

наиболее важных переходов между основными веб-страницами разрабатываемой системы.

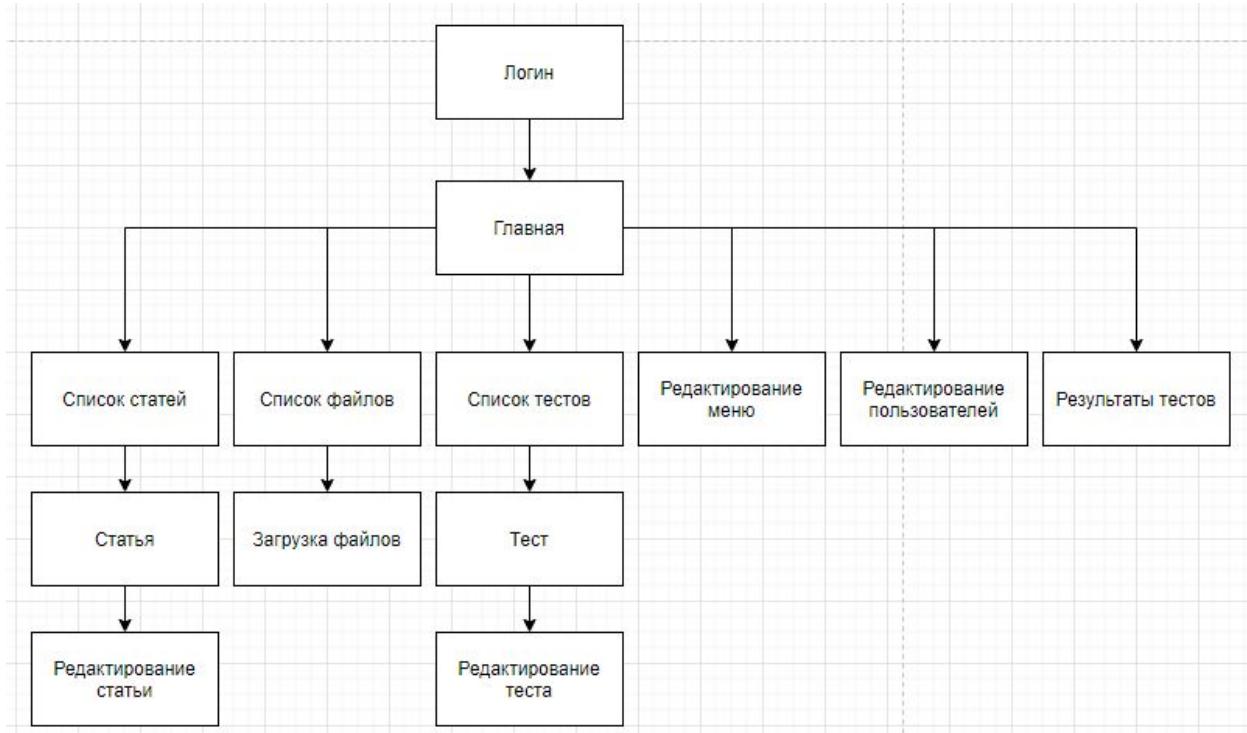


Рисунок 2.10 – Граф переходов между основными страницами разрабатываемой системы.

В данном разделе были описаны требования поведению разрабатываемой системы.

2.2. Проектирование системы

2.2.1. Архитектура данных и процессов

Основным средством графического представления архитектуры данных и процессов при использовании объектно-ориентированного подхода к моделированию систем являются диаграммы классов.

Построим диаграммы классов для наиболее важных прецедентов («Вход в систему» – рисунок 2.11, «Прохождение тестов» – рисунок 2.12).

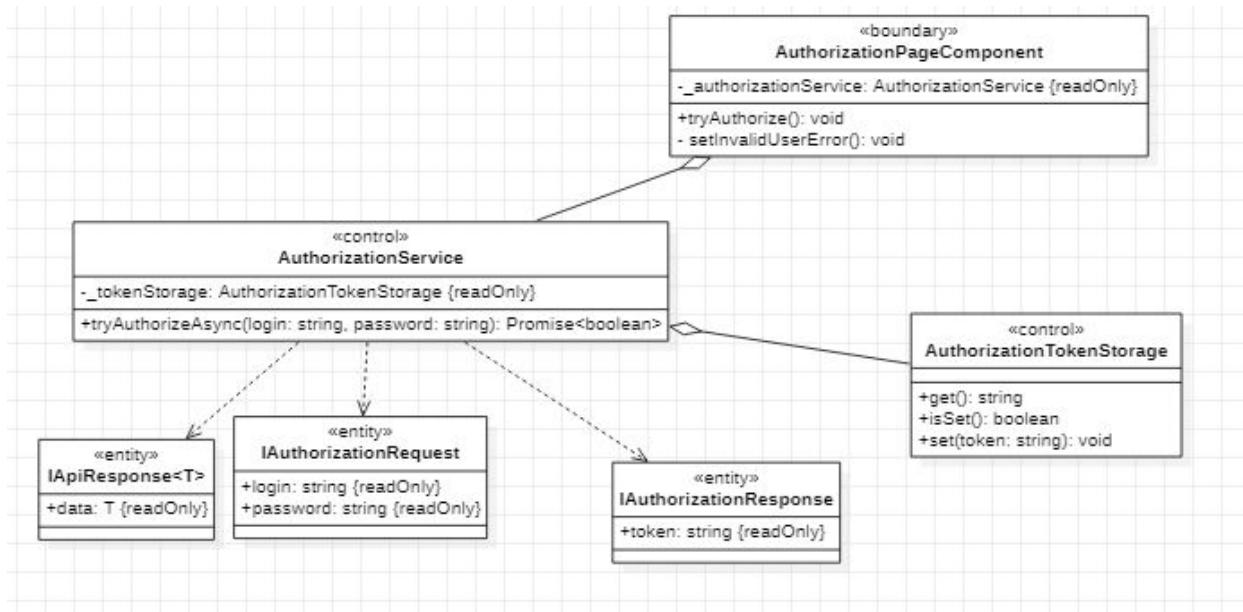


Рисунок 2.11 – Диаграмма классов прецедента «Вход в систему»

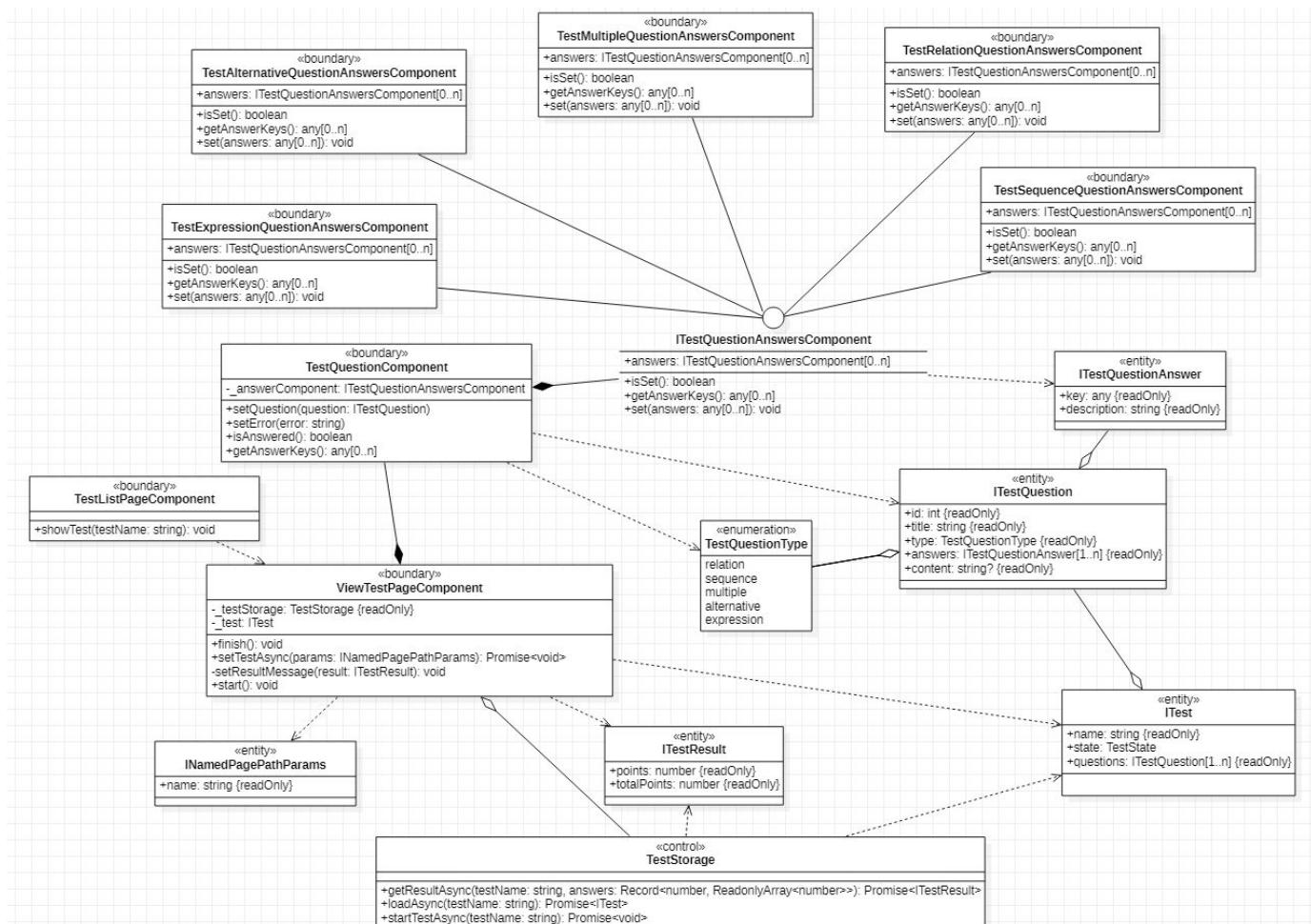


Рисунок 2.12 – Диаграмма классов прецедента «Прохождение тестов»

Спецификация диаграммы классов для прецедента «Вход в систему»:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 5 классов.

2. Характеристика классов.

Границный класс *«AuthorizationPage»* представляет собой страницу авторизации.

Содержит следующие атрибуты:

- *«_authorizationService»* – экземпляр класса *«AuthorizationService»*.

Содержит следующие операции:

- *«tryAuthorize»* – в этом методе считываются введенные логин и пароль и с их помощью осуществляется авторизация;
- *«setInvalidUserError»* – метод, который вызывается при неправильно введенном логине или пароле, выводит ошибку об этом.

Управляющий класс *«AuthorizationService»* представляет собой сервис, который получает авторизационный токен и сохраняет его.

Содержит следующие атрибуты:

- *«_tokenStorage»* – экземпляр класса *«AuthorizationTokenStorage»*.

Содержит следующие операции:

- *«tryAuthorizeAsync»* – метод, который принимает логин и пароль, формирует *«IAuthorizationRequest»* и отправляет его в *Back end*, оттуда возвращается *«IAuthorizationResponse»* на основе которого определяется успешно ли прошла авторизация и если она прошла успешно, то полученный авторизационной токен сохраняется.

Управляющий класс *«AuthorizationTokenStorage»* представляет собой сервис, который отвечает за сохранение авторизационного токена.

Содержит следующие операции:

- *«get»* – метод, который возвращает сохраненный авторизационный токен;

- «*isSet*» – метод, определяющий был ли сохранен авторизационный токен;
- «*set*» – метод, принимающий авторизационный токен и сохраняющий его.

Классы-сущности «*IApiResponse<T>*», «*IAuthorizationRequest*» и «*IAuthorizationResponse*» были описаны ранее в пункте 2.1.2.

Спецификация диаграммы классов для прецедента «Прохождение тестов»:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 14 классов;
- 1 интерфейс;
- 1 перечисление.

2. Характеристика классов.

Классы «*ITest*», «*ITestResult*», «*ITestQuestion*», «*ITestQuestionAnswer*» и перечисление «*TestQuestionType*» были описаны в пункте 2.1.2.

Интерфейс «*ITestQuestionAnswersComponent*» представляет собой соглашения для компонента, отвечающего за отображение ответов на вопрос и взаимодействие с ними.

Содержит следующие атрибуты:

- «*answers*» – массив ответов на вопрос.

Содержит следующие операции:

- «*isSet*» – метод, определяющий установлен ли ответ;
- «*set*» – метод, устанавливающий ответ;
- «*getAnswerKeys*» – метод, возвращающий выбранные пользователем ответы.

Для каждого возможного типа вопроса существует своя реализация интерфейса «*ITestQuestionAnswersComponent*»:

- «*TestExpressionQuestionAnswersComponent*» – реализация компонента для типа вопроса «*Expression*»;

- «*TestAlternativeQuestionAnswersComponent*» – реализация компонента для типа вопроса «*Alternative*»;
- «*TestMultipleQuestionAnswersComponent*» – реализация компонента для типа вопроса «*Multiple*»;
- «*TestRelationQuestionAnswersComponent*» – реализация компонента для типа вопроса «*Relation*»;
- «*TestSequenceQuestionAnswersComponent*» – реализация компонента для типа вопроса «*Sequence*».

Границный класс «*TestQuestionComponent*» представляет собой компонент, отвечающий за отображение вопроса и взаимодействие с ним.

Содержит следующие атрибуты:

- «*_answerComponent*» – экземпляр класса, реализующего интерфейс «*ITestQuestionAnswersComponent*».

Содержит следующие операции:

- «*setQuestion*» – метод, инициализирующий компонент;
- «*setError*» – метод, отображающий ошибку;
- «*isAnswered*» – метод, определяющий установлен ли ответ;
- «*getAnswerKeys*» – метод, возвращающий выбранные пользователем ответы.

Границный класс «*TestListPage*» представляет собой страницу со списком тестов, содержит следующие операции:

- «*showTest*» – метод, который переадресовывает пользователя на страницу конкретного теста.

Границный класс «*ViewTestPage*» представляет собой компонент, отвечающий за отображение вопроса и взаимодействие с ним.

Содержит следующие атрибуты:

- «*_testStorage*» – экземпляр класса «*TestStorage*»;
- «*_test*» – текущий, отображаемый тест.

Содержит следующие операции:

- «*finish*» – метод, который завершает тест, если все вопросы отвечены, то отображается результат теста, иначе отображается сообщение о том, что не все вопросы отвечены;
- «*start*» – метод, который начинает тест;
- «*setTestAsync*» – метод, устанавливающий текущий тест;
- «*setResultMessage*» – метод, отображающий результат теста.

Управляющий класс «*TestStorage*» представляет сервис, отвечающий за работу с данными тестов, содержит следующие операции:

- «*getResultAsync*» – метод, который отправляет ответы пользователя в *Back end* и возвращает результат теста – экземпляр класса «*ITestResult*»;
- «*startTestAsync*» – метод, который сообщает *Back end* о том, что тест был начат;
- «*loadAsync*» – метод, принимающий название теста и запрашивающий по этому названию тест из *Back end*.

Класс-сущность «*INamedPagePathParams*» представляет параметры страницы теста и содержит следующие атрибуты:

- «*name*» – название теста.

В данном разделе была описана архитектура данных и процессов с помощью диаграмм классов для основных прецедентов разрабатываемой системы.

2.2.2. Модель поведения

Модель поведения ПС на этапе детального проектирования представляет собой описание алгоритмов реализации прецедентов системы. Опишем алгоритмы реализации основных прецедентов проектируемой системы. Для прецедента «Вход в систему» построим диаграмму коммуникации (рисунок 2.13), а для прецедента «Прохождение тестов» – диаграмму последовательности (рисунок 2.14).

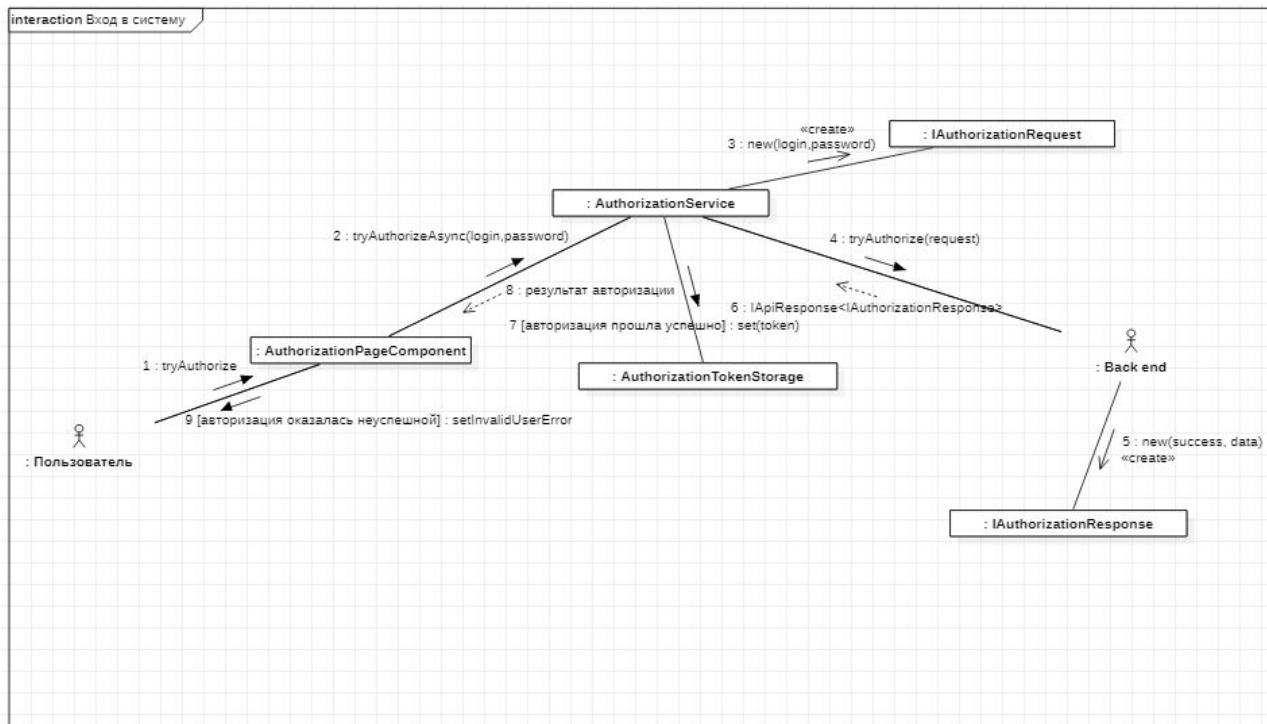


Рисунок 2.13 – Диаграмма коммуникации прецедента «Вход в систему»

Спецификация диаграммы коммуникации прецедента «Вход в систему»:

- Количественная характеристика диаграммы.

Диаграмма содержит:

- 5 экземпляров классов;
- 9 связей.

- Характеристика связей.

Пользователь взаимодействуя с системой провоцирует вызов метода «tryAuthorize» у объекта класса «AuthorizationPageComponent». Во время выполнения этого метода он вызывает метод «tryAuthorizeAsync» экземпляра класса «AuthorizationService», который, в свою очередь, создает объект класса «IAuthorizationRequest» и отправляет его в *Back end*, вызывая там метод «tryAuthorize». *Back end* создает «IAuthorizationResponse» и возвращает его обратно. Экземпляр класса «AuthorizationService» в случае успешной авторизации вызывает метод «set» экземпляра класса «AuthorizationTokenStorage». Результат авторизации возвращается в

«*AuthorizationPageComponent*» и, если авторизация была неуспешной, то вызывается метод «*setInvalidUserError*».

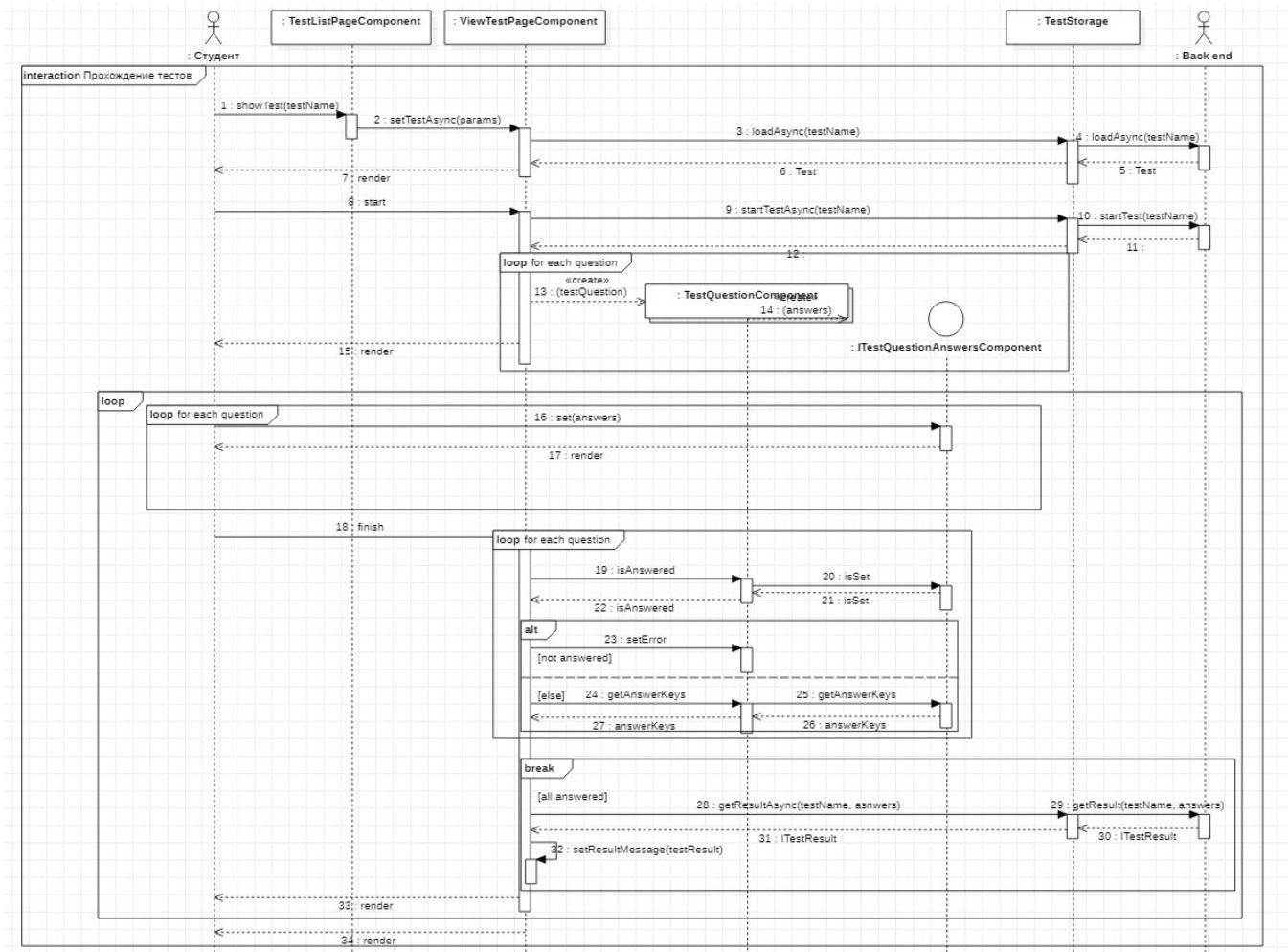


Рисунок 2.14 – Диаграмма последовательности прецедента «Прохождение тестов»

Спецификация диаграммы последовательности прецедента «Прохождение тестов»:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 5 экземпляров классов;
- 34 связи.

2. Характеристика связей.

Студент при выборе теста провоцирует вызов метода «*showTest*» экземпляра класса «*TestListPageComponent*». Далее происходит вызов метода «*setTestAsync*» объекта класса «*ViewTestPageComponent*». Затем вызывается метод «*loadAsync*» экземпляра класса «*TestStorage*». Объект класса «*TestStorage*» обращается к *Back end* и получает данные теста, представленные объектом класса «*ITest*». Объект класса «*ITest*» возвращается в объект страницы «*ViewTestPageComponent*», далее отображается заголовок теста и кнопка «начать», нажатием на кнопку «начать», провоцируется вызов метода «*start*» страницы «*ViewTestPageComponent*», где после сообщения *Back end* о начале теста для каждого вопроса теста создается экземпляр класса «*TestQuestionComponent*», который, в свою очередь, создает объект «*ITestQuestionAnswerComponent*». Затем происходит отображение страницы и ожидается следующее действие студента.

Студент отвечая на вопросы провоцирует вызов метода «*set*» соответствующего объекта «*ITestQuestionAnswersComponent*», этот объект отвечает отображением изменения (помечает ответ как выбранный).

Когда студент считает, что он готов завершить тест, то он нажатием кнопки «Завершить» провоцирует вызов метода «*finish*» объекта класса «*ViewTestPageComponent*», который проверяет, что отвенчен каждый вопрос, вызывая метод «*isAnswered*» соответствующего объекта класса «*TestQuestionComponent*». Затем, если вопрос отвенчен, то собираются ответы пользователя посредством вызова метода «*getAnswerKeys*» объекта класса «*TestQuestionComponent*», иначе происходит отображение ошибки, посредством вызова метода «*setError*» класса «*TestQuestionComponent*». Если все вопросы были отвенчены, то происходит вызов метода *getResultAsync* класса «*TestStorage*», который получает результат из *Back end*, после получения результата он отображается посредством вызова метода «*setResultMessage*» и цикл прерывается. Если хотя бы один вопрос не был отвенчен, то этот сегмент повторяется.

В данном разделе была описана модель поведения основных прецедентов проектируемой системы.

2.2.3. Модель реализации

2.2.3.1. Модель исходного кода

Модель исходного кода показывает – какие классы в каких файлах содержатся. Построим диаграмму компонентов для модели исходного кода классов, описанных в пункте 2.1.1. (рисунок 2.15).

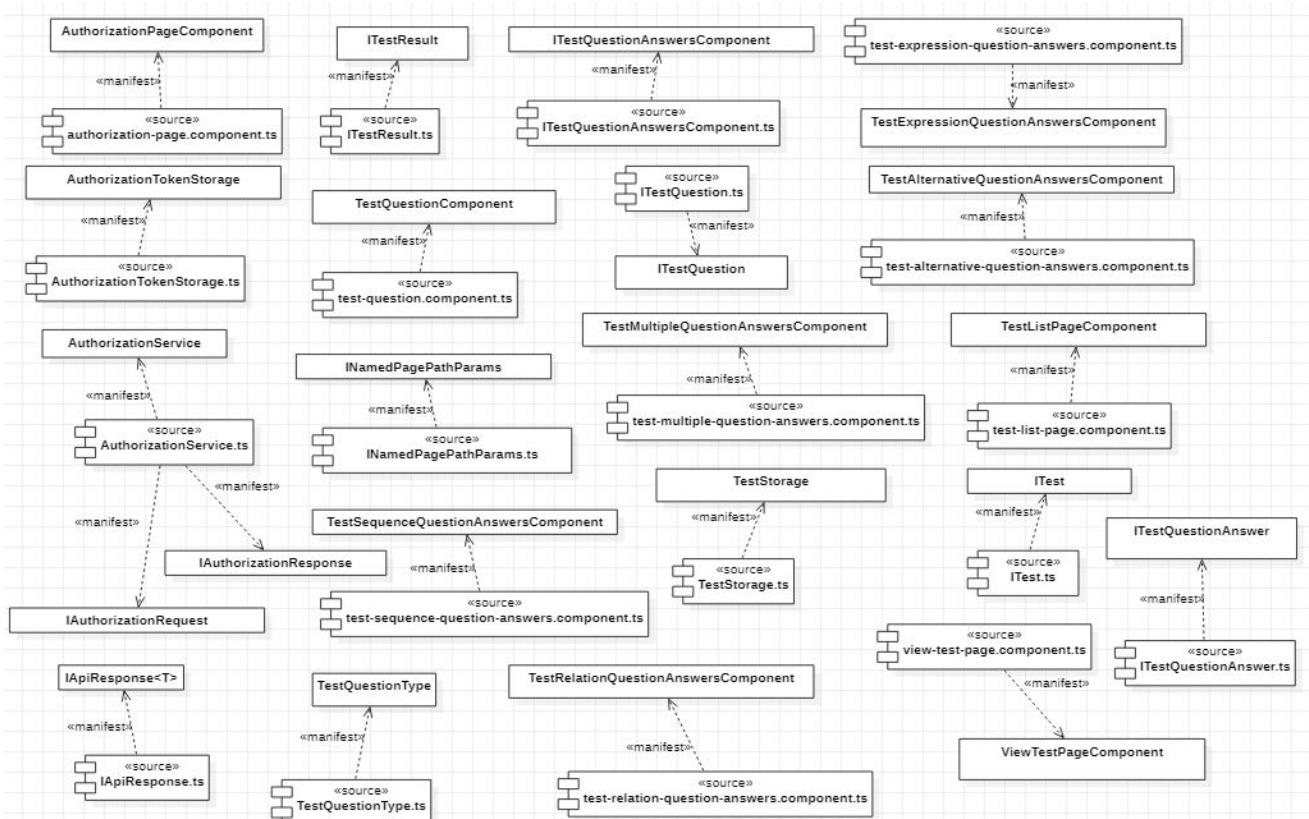


Рисунок 2.15 – Диаграмма компонентов для модели исходного кода классов, описанных в пункте 2.1.1.

На диаграмме представлены следующие файлы с исходным кодом:

- «*authorization-page.component.ts*» – содержит класс «*AuthorizationPageComponent*».
 - «*AuthorizationTokenStorage.ts*» – содержит класс «*AuthorizationTokenStorage*».
 - «*AuthorizationService.ts*» – содержит классы «*AuthorizationService*», «*IAuthorizationResponse*» и «*IAuthorizationRequest*».
 - «*IApiResponse.ts*» – содержит класс «*IApiResponse<T>*».
 - «*ITestResult.ts*» – содержит класс «*ITestResult*».

- «*test-question.component.ts*» – содержит класс «*TestQuestionComponent*».
- «*INamedPagePathParams.ts*» – содержит класс «*INamedPagePathParams*».
- «*test-sequence-question-answers.component.ts*» – содержит класс «*TestSequenceQuestionAnswersComponent*».
- «*TestQuestionType.ts*» – содержит перечисление «*TestQuestionType*».
- «*ITestQuestionAnswersComponent.ts*» – содержит интерфейс «*ITestQuestionAnswersComponent*».
- «*ITestQuestion.ts*» – содержит класс «*ITestQuestion*».
- «*test-multiple-question-answers.component.ts*» – содержит класс «*TestMultipleQuestionAnswersPageComponent*».
- «*TestStorage.ts*» – содержит класс «*TestStorage*».
- «*test-relation-question-answers.component.ts*» – содержит класс «*TestRelationQuestionAnswersPageComponent*».
- «*test-expression-question-answers.component.ts*» – содержит класс «*TestExpressionQuestionAnswersPageComponent*».
- «*test-alternative-question-answers.component.ts*» – содержит класс «*TestAlternativeQuestionAnswersPageComponent*».
- «*test-list-page.component.ts*» – содержит класс «*TestListPageComponent*».
- «*ITest.ts*» – содержит класс «*ITest*».
- «*ITestQuestionAnswer.ts*» – содержит класс «*ITestQuestionAnswer*».
- «*view-test-page.component.ts*» – содержит класс «*ViewTestPageComponent*».

2.2.3.2. Модель исполняемого кода

В модели исполняемого кода на диаграмме компонентов отображаются только артефакты – файлы с исходными, объектными и загружаемыми кодами. Построим диаграмму компонентов для модели исполняемого кода артефактов из пункта 2.2.3.1. (рисунок 2.16).

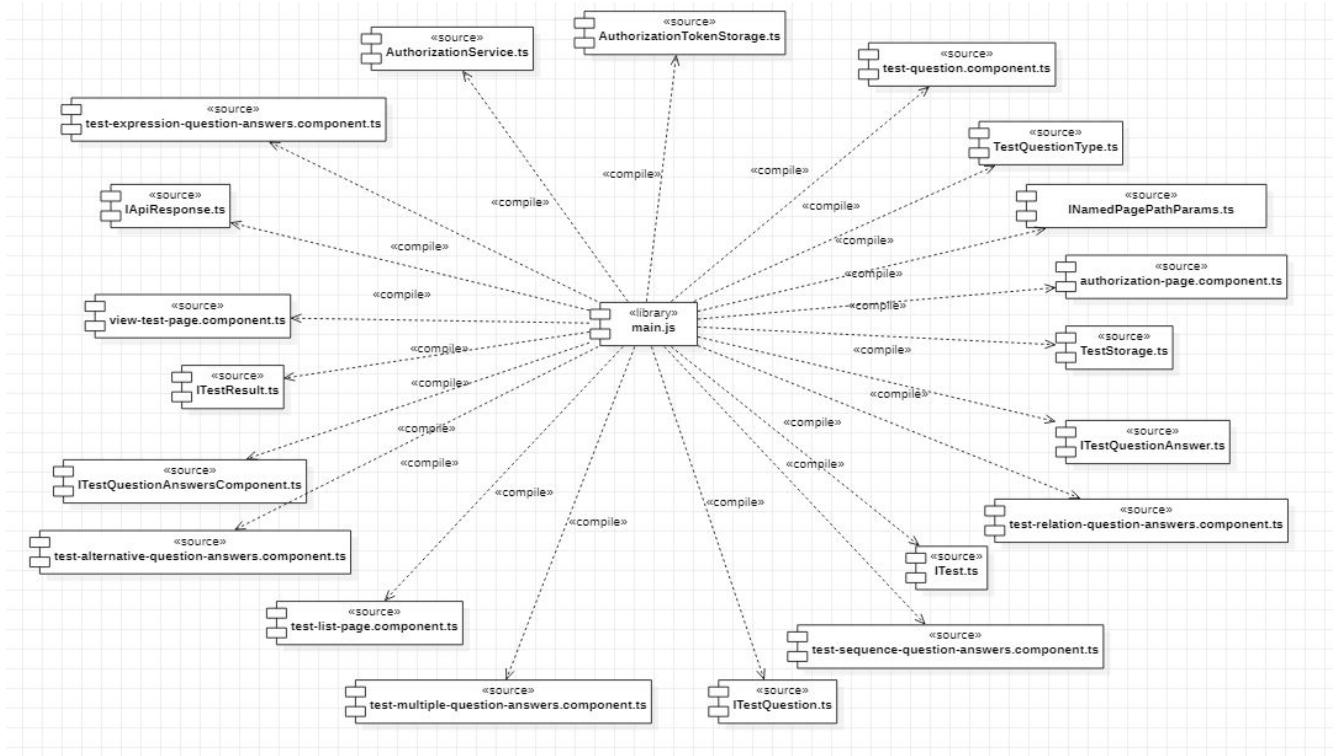


Рисунок 2.16 – Диаграмма компонентов для модели исполняемого кода артефактов из пункта 2.2.3.1

Все выявленные ранее файлы с исходным кодом компилируются в один файл с исполняемым кодом – «main.js».

2.2.3.3. Модель поставляемых артефактов

Модель поставляемых артефактов представляется диаграммой компонентов, на которой отображаются артефакты исполняемого кода и данных, отторгаемые от разработчика, и передаваемые потребителю – заказчику или покупателю. Диаграмма изображена на рисунке 2.17.

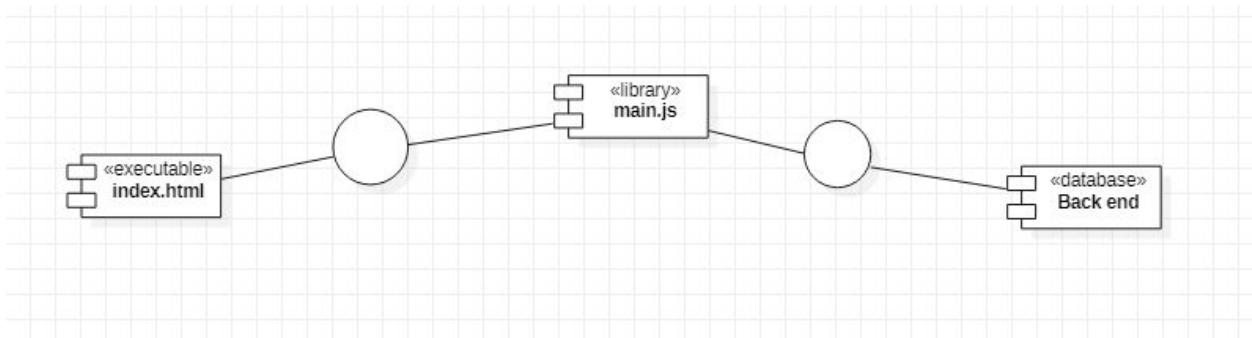


Рисунок 2.17 – Диаграмма компонентов для модели поставляемых артефактов

Выполняемый файл «index.html» подгружает библиотеку «main.js», с помощью которой клиентское приложение взаимодействует с *Back end*.

2.2.3.4. Модель размещения артефактов

Модель размещения артефактов отображает распределение артефактов, поставляемых потребителю по физическим устройствам. Диаграмма развертывания для этой модели изображена на рисунке 2.18.

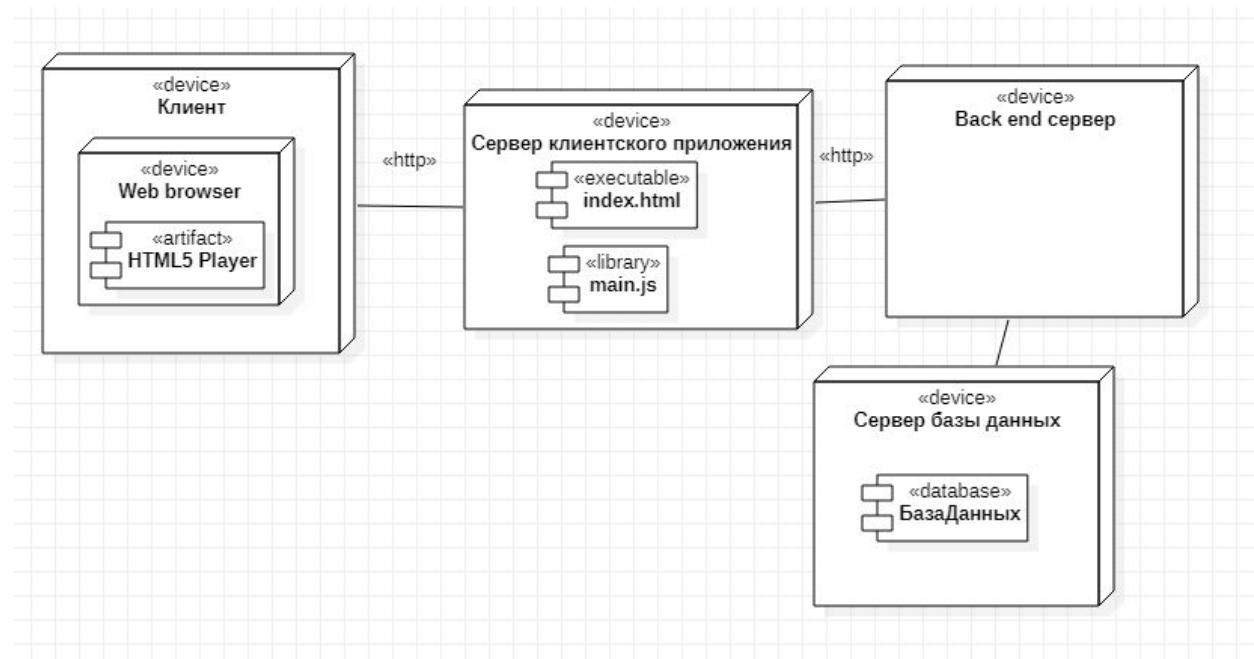


Рисунок 2.18 – Диаграмма развертывания для модели размещения артефактов

Спецификация диаграммы развертывания для модели размещения артефактов:

1. Количественная характеристика диаграммы.

Диаграмма содержит:

- 5 узлов (1 вложенный);
- 4 компонента;
- 3 связи.

2. Спецификация узлов.

Узел «Клиент» содержит в себе узел «Web browser», который содержит компонент «HTML5 Player», с помощью которого происходит взаимодействие с узлом «Сервер клиентского приложения».

Узел «Сервер клиентского приложения» содержит компоненты «index.html» (выполняемый файл клиентского приложения) и «main.js» (библиотека со скриптами клиентского приложения), с помощью которых взаимодействует с узлом «Back End сервер».

Узел «Back End Сервер» взаимодействует с узлом «Сервер базы данных».

Узел «Сервер базы данных» содержит в себе компонент «БазаДанных».

2.2.3.5. Модель управления

В разработке информационных систем модели управления изображаются схемами процедурных вызовов между структурными блоками приложения. На рисунке 2.19 изображена схема процедурных вызовов для прецедента «Вход в систему».

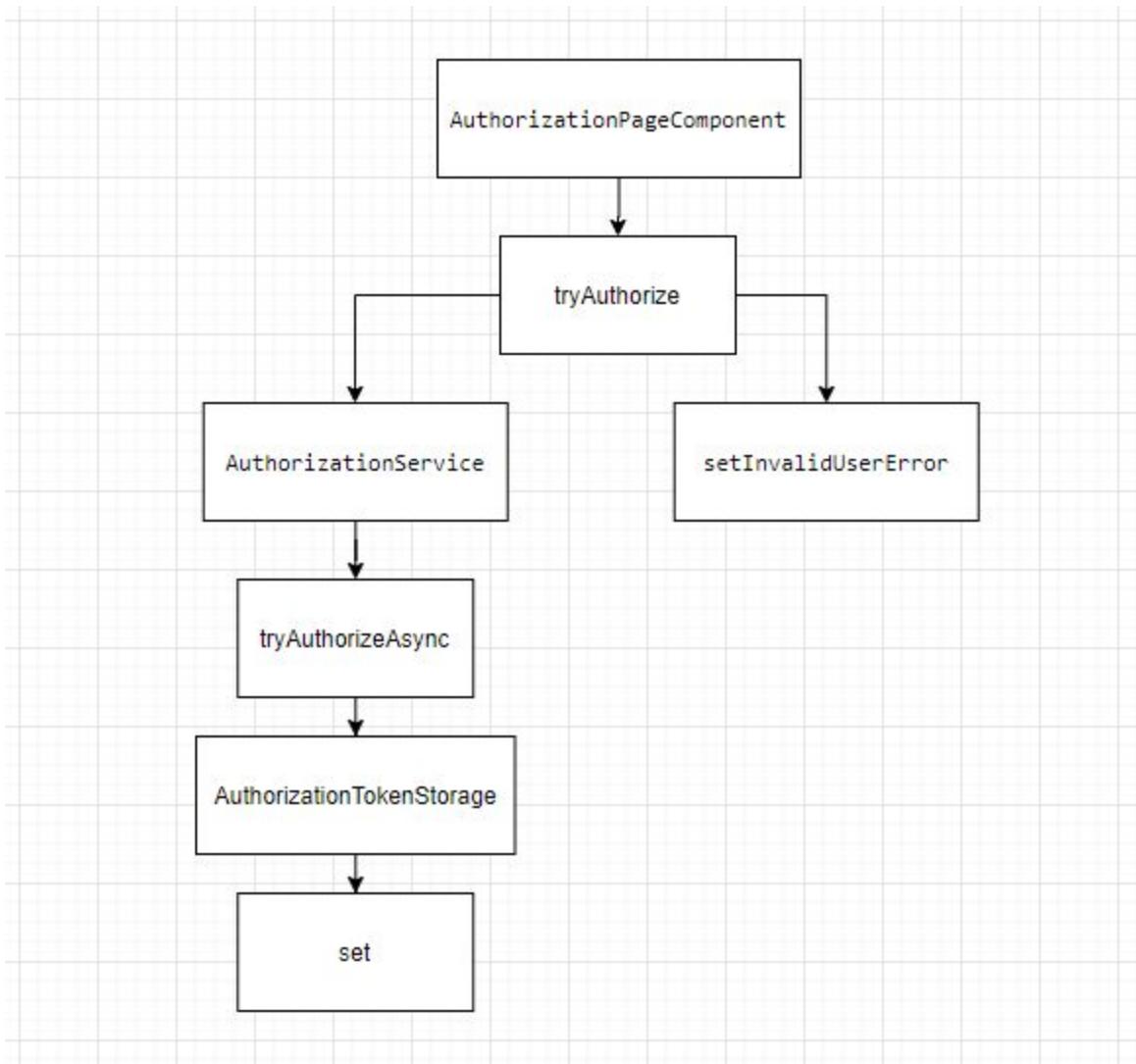


Рисунок 2.19 – Схема процедурных вызовов для прецедента «Вход в систему»

2.2.3.6. Интерфейс пользователя

Между страницами разрабатываемого приложения пользователи будут переключаться посредством навигационного меню или с помощью отдельных ссылок на конкретных страницах.

Целесообразно выделить общие для всех страниц элементы пользовательского интерфейса.

Так как разрабатываемая система не подразумевает доступ неавторизованного пользователя к какой-либо ее части, то страница авторизации должна быть отдельной от всех остальных страниц. Эскиз страницы авторизации изображен на рисунке 2.20.



Рисунок 2.20 – Эскиз страницы авторизации

На всех остальных страницах необходимо отображать наиболее информативные компоненты, а именно меню и блок новостей, макет изображен на рисунке 2.21.

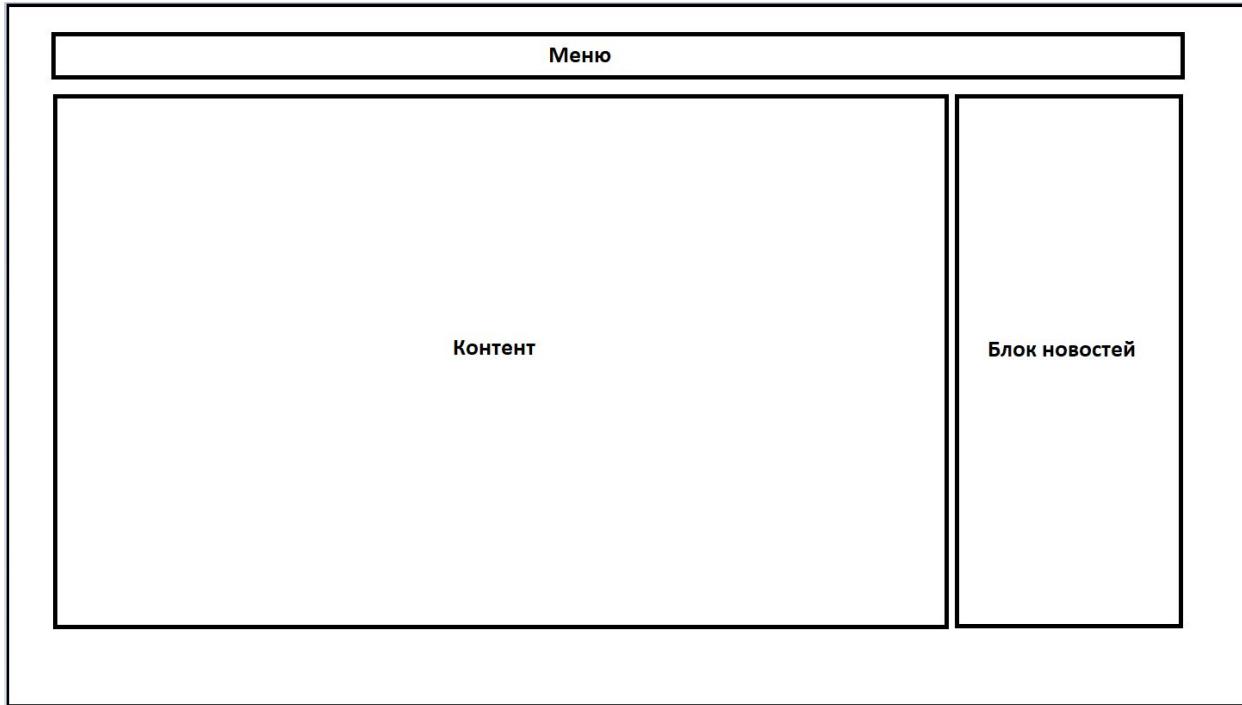


Рисунок 2.21 – Макет всех страниц приложения (кроме страницы авторизации)

В данном разделе были описаны составляющие модели реализации, а именно модель исходного кода, модель исполняемого кода, модель поставляемых артефактов, модель размещения артефактов, модель управления и интерфейс пользователя.

2.3. Описание технологии разработки клиентской части веб-приложений с использованием фреймворка «Angular»

Angular – это платформа и фреймворк для создания одностраничных клиентских приложений с использованием *HTML* и *TypeScript*. *Angular* написан на *TypeScript*. Он реализует основные и дополнительные функции в виде набора *TypeScript* библиотек.

Архитектура *Angular*-приложения опирается на определенные фундаментальные концепции. Основными строительными блоками являются *NgModules* (модули *Angular*), которые предоставляют контекст компиляции для компонентов. *NgModules* собирают связанный код в функциональные

наборы; *Angular*-приложение определяется набором *NgModules*. В приложении всегда есть по крайней мере корневой модуль [15].

NgModule определяется классом с декоратором `@NgModule()`. Декоратор `@NgModule()` – это функция, которая принимает один объект метаданных, свойства которого описывают модуль. Наиболее важные свойства:

- *declarations*: компоненты, директивы и пайпы, которые принадлежат этому *NgModule*;
- *exports*: подмножество объявлений, которые должны быть видны и использоваться в шаблонах компонентов других *NgModules*;
- *imports*: другие модули, чьи экспортированные классы необходимы шаблонам компонентов, объявленным в этом *NgModule*;
- *providers*: создатели сервисов, которые этот *NgModule* вносит в глобальный набор сервисов; они становятся доступными во всех частях приложения;
- *bootstrap*: основное представление приложения, называемое корневым компонентом, в котором размещены все остальные представления приложения. Только корневой *NgModule* должен устанавливать свойство *bootstrap* [16].

Компонент контролирует участок экрана, называемый представлением. Внутри класса компонента определяется его логика – что он делает для поддержки представления. Класс взаимодействует с представлением через *API* свойств и методов. Вид компонента определяется его сопутствующим шаблоном. Шаблон – это форма HTML, которая сообщает *Angular*, как визуализировать компонент. Иерархия представлений может включать представления от компонентов в одном и том же *NgModule*, но она также может включать представления от компонентов, определенных в разных *NgModule* [17].

Сервис – это широкая категория, охватывающая любое значение, функцию или особенность, которая нужна приложению. Сервис обычно

представляет собой класс с узкой, четко определенной целью. Он должен делать что-то конкретное. *Angular* отличает компоненты от сервисов для повышения модульности и возможности повторного использования. Работа компонента заключается в том, чтобы обеспечить взаимодействие с пользователем и ничего более. Компонент должен представлять свойства и методы для привязки данных, чтобы быть посредником между представлением (отображаемым шаблоном) и логикой приложения (которая часто включает в себя некоторое представление о модели) [18].

В данном разделе были описаны основные принципы работы с фреймворком *Angular* и его основные части.

2.4. Реализация программного обеспечения

Для удобства разработки был реализован механизм подмены классов в зависимости от среды выполнения (так как разработка клиентской части ведется отдельно от серверной). То есть было реализовано по 2 версии сервисов, которые работают с данными – те, которые работают с *Back end* и так называемые «Заглушки», которые работают с «*Local Storage*» или просто с памятью.

Для реализации доступа к *Back end* был использован класс фреймворка *Angular* – *HttpClient* [20], для удобства работы с ним была реализована обертка *IApiService*:

```
export interface IApiService {
    // GET запрос
    getAsync<T>(action: string, params?: HttpParams): Observable<T>;
    // POST запрос
    postAsync<T>(action: string, body?: any, params?: HttpParams): Observable<T>;
    // DELETE запрос
    deleteAsync<T>(action: string, params?: HttpParams): Observable<T>;
}

export class ApiService implements IApiService {
    public constructor(private readonly _httpClient: HttpClient,
        private readonly _endpointProvider: IApiServiceEndpointProvider,
        private readonly _httpHeadersProvider?: IHttpHeadersProvider) {
        Argument.isNotNullOrUndefined(this._httpClient, 'httpClient');
        Argument.isNotNullOrUndefined(this._endpointProvider, 'endpointProvider');
    }
}
```

```

}

protected get responseType(): ResponseType {
    return ResponseType.json;
}

public getAsync<T>(action: string, params?: HttpParams): Observable<T> {
    return      this._httpClient.get<T>(this.buildUrl(action),
this.getOptions(params));
}

public postAsync<T>(action: string, body?: any, params?: HttpParams): Observable<T> {
    return      this._httpClient.post<T>(this.buildUrl(action),     body,
this.getOptions(params));
}

public deleteAsync<T>(action: string, params?: HttpParams): Observable<T> {
    return      this._httpClient.delete<T>(this.buildUrl(action),
this.getOptions(params));
}

private buildUrl(action: string): string {
    return `${this._endpointProvider.getEndpoint()}/${action}`;
}

private getOptions(params?: HttpParams) {
    return {
        headers:      this._httpHeadersProvider      &&
this._httpHeadersProvider.getHeaders(),
        params,
        responseType: this.responseType as 'json'
    };
}
}

```

С помощью метода «*getAsync*» делается *GET* запрос, с помощью «*postAsync*» – *POST* запрос, с помощью «*deleteAsync*» *DELETE* запрос.

При загрузке приложение делает следующие запросы к *Back end*:

- GET запрос для получения настроек (диаграмма классов настроек показана на рисунке 2.6);
- GET запрос для получения текстов;
- если есть авторизационные *cookie*, то GET запрос для получения контекста пользователя.

Если хотя бы один из этих запросов выполняется с ошибкой, то отображается сообщение о том что приложение временно недоступно (рисунок 2.22).

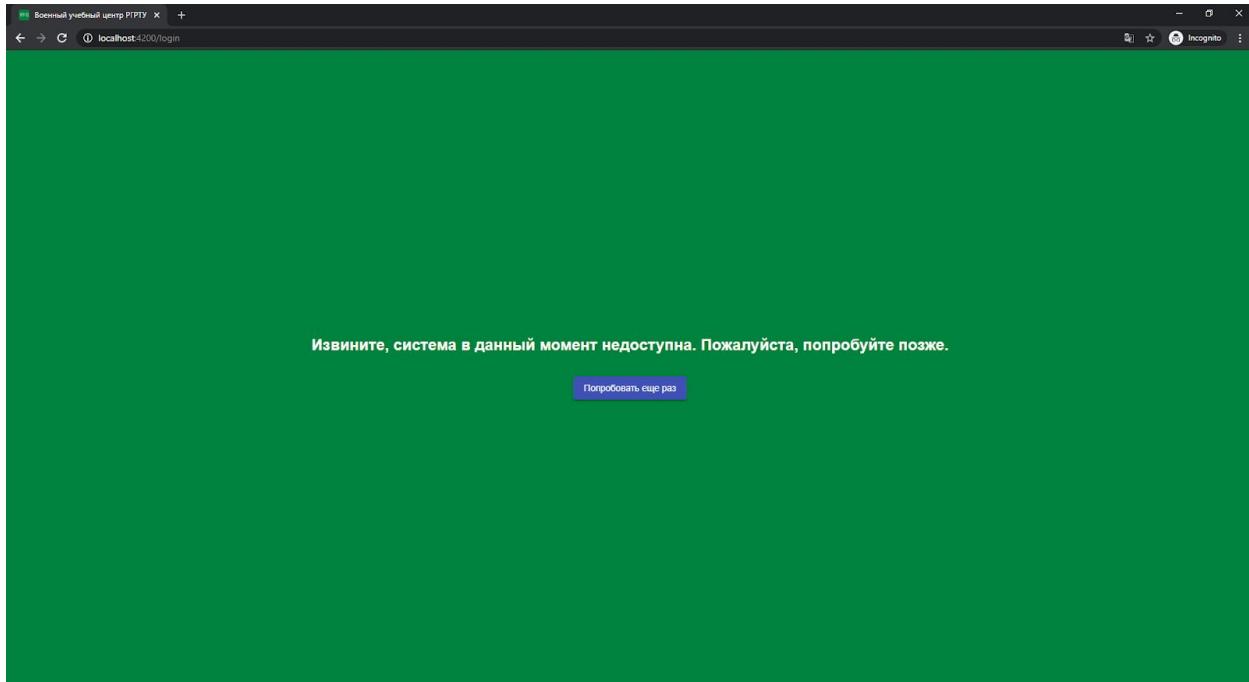


Рисунок 2.22 – Сообщение о том, что приложение недоступно

Как и описывалось в пункте 2.3 все веб-приложение было разбито на модули. Все сервисы разбиты на модули и эти модули собраны в отдельном модуле *ServiceModule*. Все компоненты также разбиты на модули и эти модули собраны в модуле *UIModule*. Модули *ServiceModule* и *UIModule* собраны в корневом модуле приложения *AppModule*.

Для уменьшения времени загрузки приложения и уменьшения размера изначально загруженных скриптов, было принято решение использовать такой функционал *Angular*, как «Ленивая загрузка модулей» – шаблон проектирования, который загружает *NgModules* по мере необходимости [19].

Далее рассмотрим модули компонентов приложения подробнее.

В основной модуль (*UIModule*) приложения помимо каркаса были также включены главная страница и страница «Страница не найдена», так

как они не используют никаких особых функций и выделять их в отдельный модуль не имеет смысла.

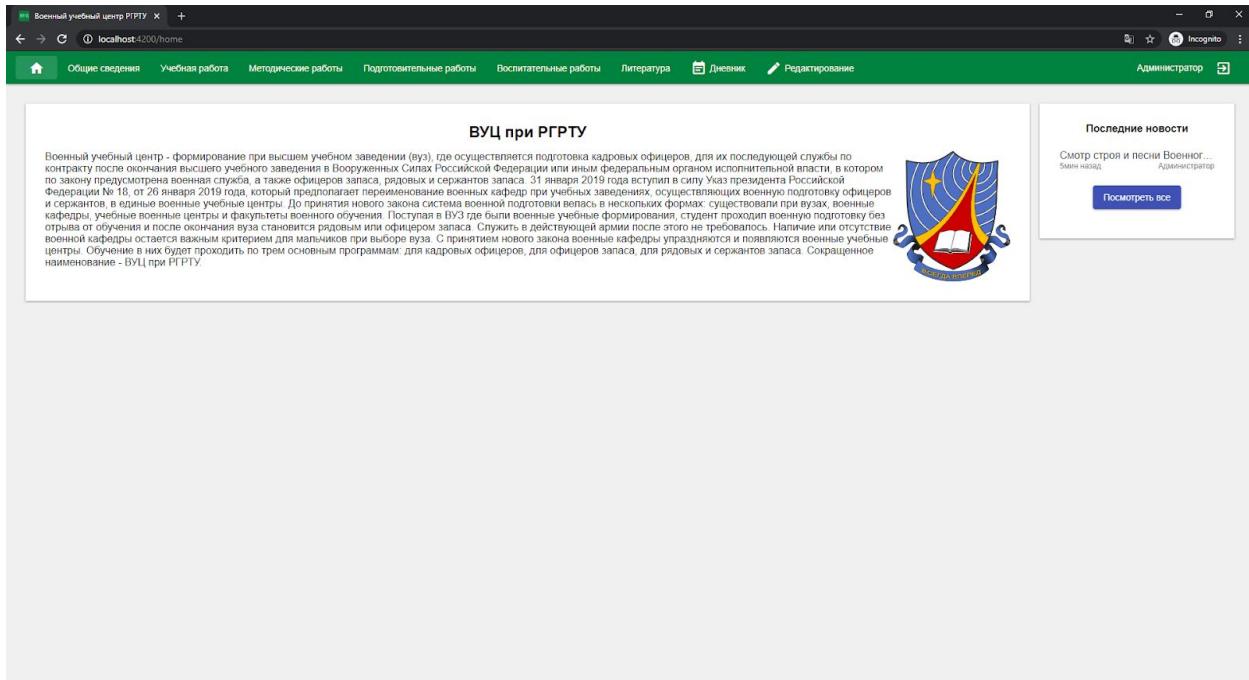


Рисунок 2.23 – Главная страница

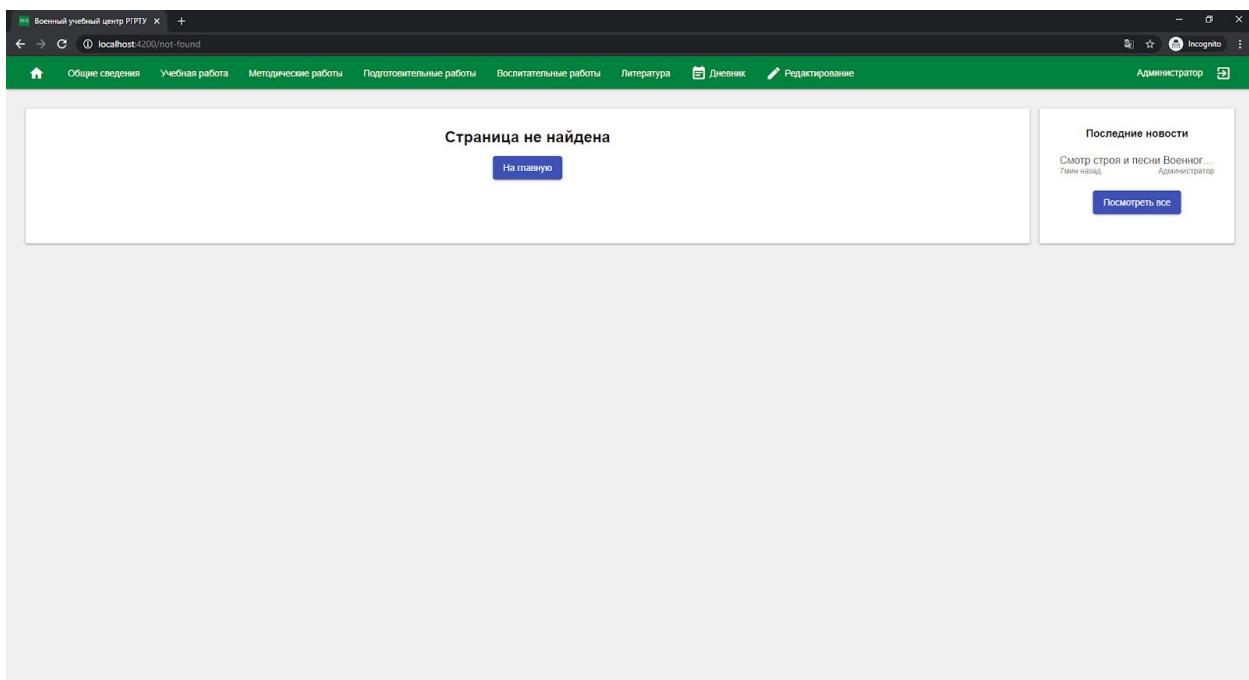


Рисунок 2.24 – Страница «Страница не найдена»

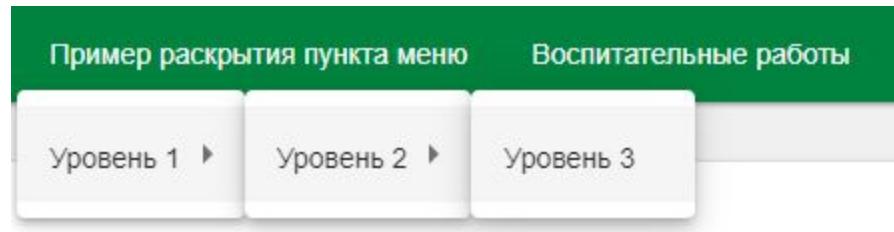


Рисунок 2.25 – Пример раскрытия пункта меню

2.4.1 Модуль «Авторизация»

Для того, чтобы воспользоваться любой функцией системы, необходимо авторизоваться. Авторизация происходит на странице авторизации, для ее осуществления пользователь должен ввести логин и пароль. Чтобы пользователю не приходилось авторизовываться каждый раз как хочет воспользоваться системой, его авторизационный токен сохраняется в *cookie*. Время действия авторизационного токена определяется *Back end*, когда оно истечет пользователь будет вынужден авторизоваться снова. При попытке ввода несуществующей пары логин-пароль система отобразит ошибку и авторизация не произойдет.

Страница авторизации изображена на рисунке 2.26, алгоритм авторизации на рисунке 2.27.

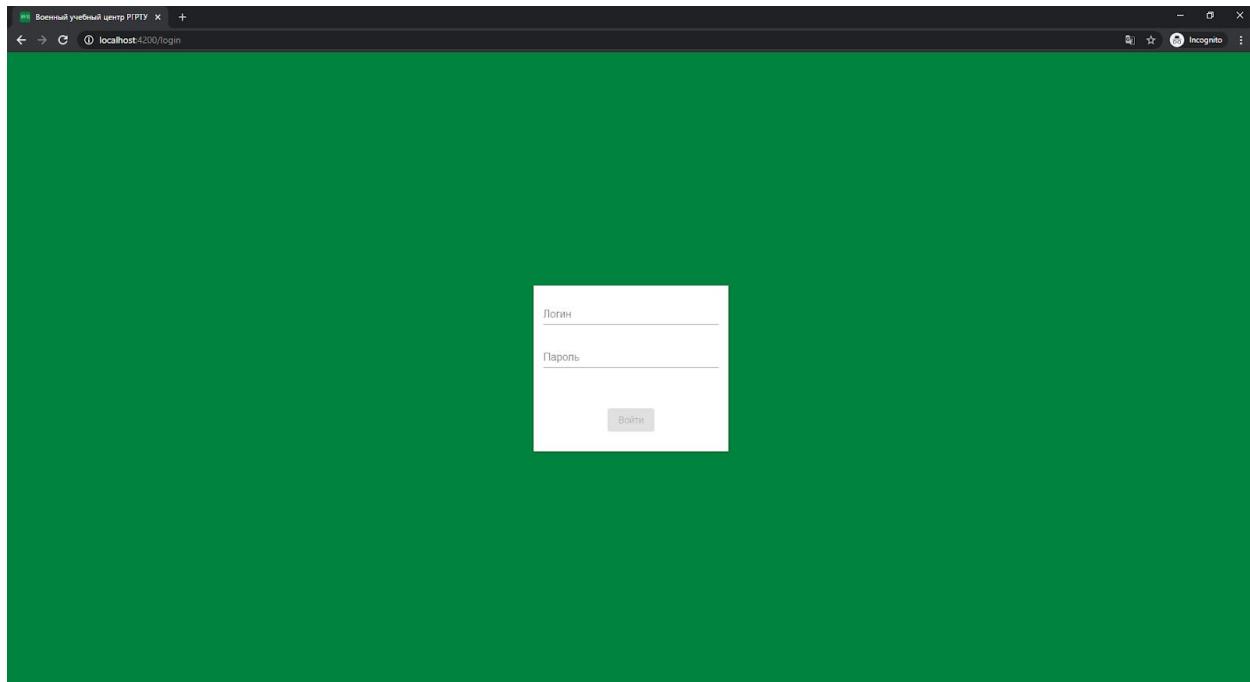


Рисунок 2.26 – Страница авторизации

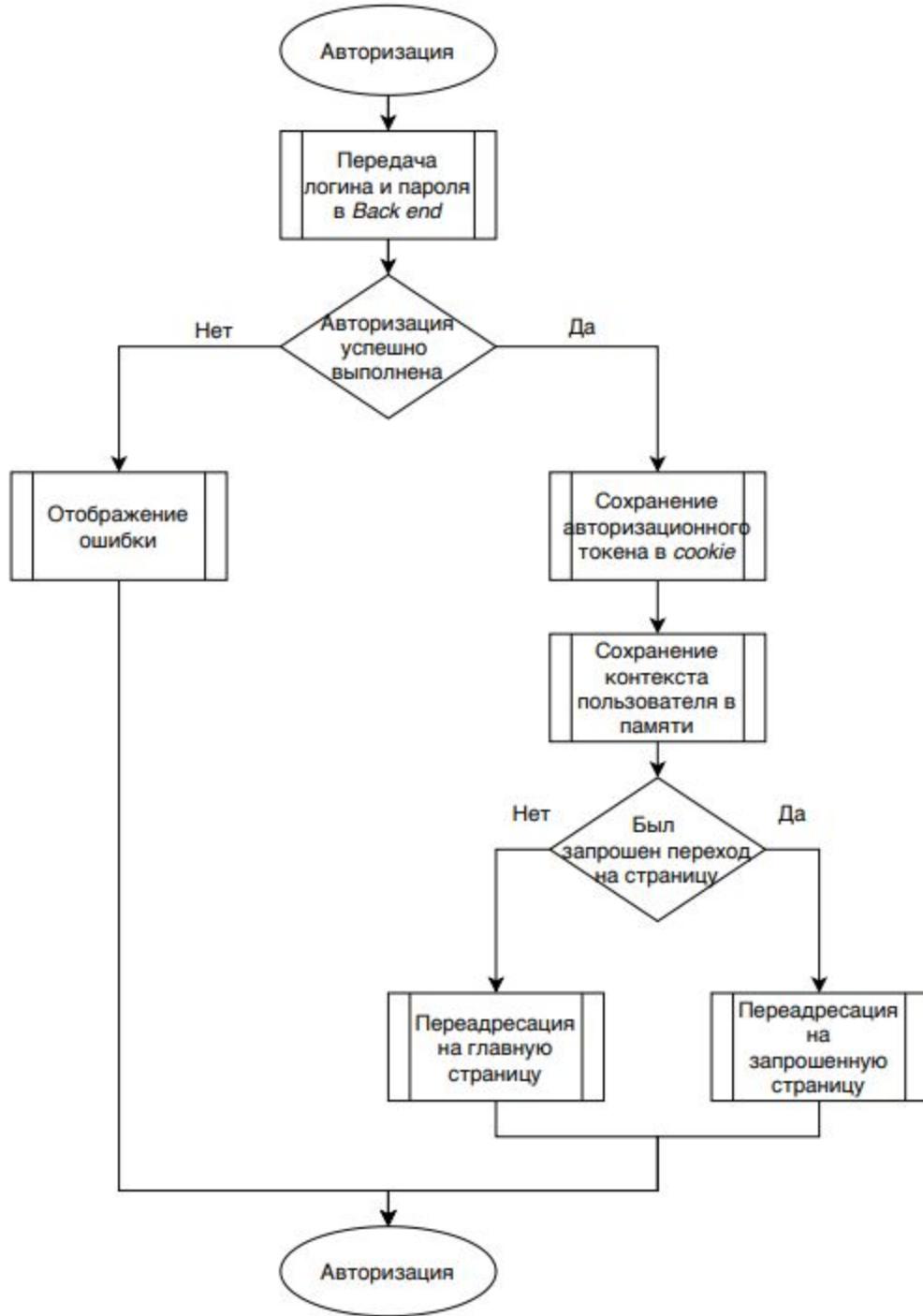


Рисунок 2.27 – Алгоритм авторизации

2.4.2 Модуль «Статьи»

Для того чтобы обеспечить динамическое наполнение системы, был разработан модуль статей. Содержание данных страниц хранятся в БД системы, пользователь может получить к ним доступ используя меню или список статей. Статьи, являющиеся новостями, также отображаются в блоке новостей. Модуль «Статьи» включает в себя 2 страницы:

1. Страница просмотра статьи.

Компонент «*ViewArticlePageComponent*» являющийся реализацией страницы, содержит логику по отображению статей. Пример статьи показан на рисунке 2.28.

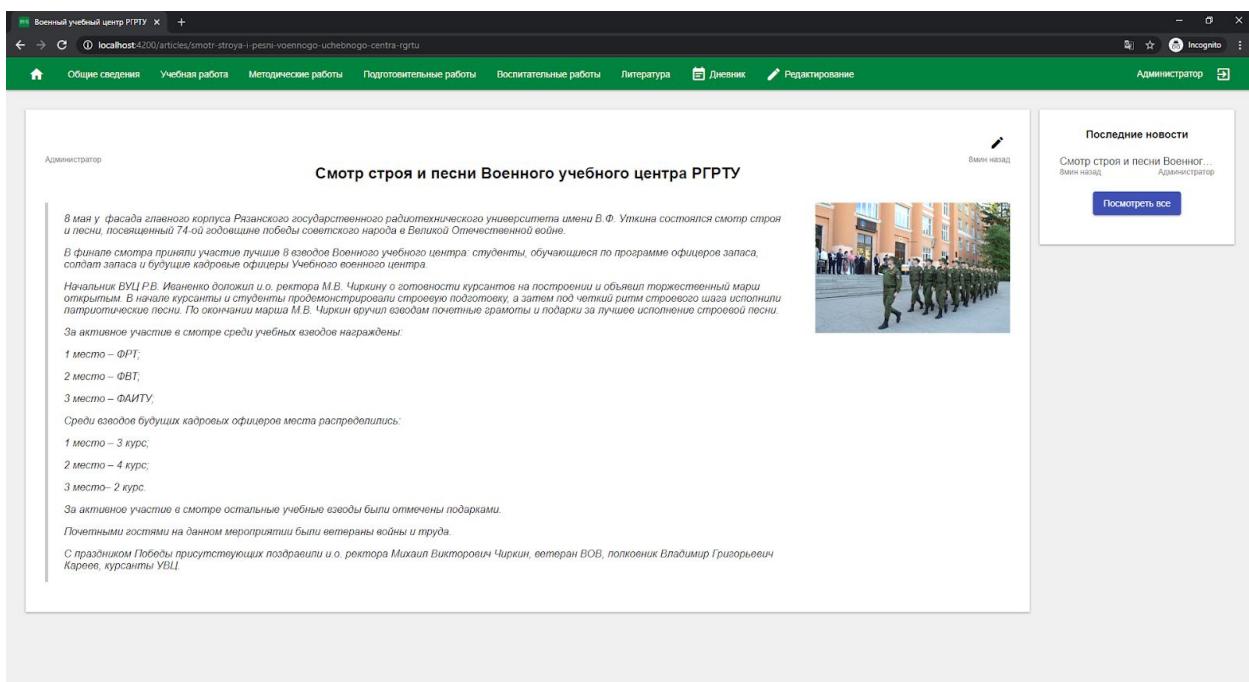


Рисунок 2.28 – Пример отображения статьи

2. Страница списка статей.

Для того, чтобы получить возможность поиска статей (не все статьи могут находиться в меню – это зависит от решения администратора) была реализована страница со списком статей. На этой странице присутствуют фильтры, с помощью которых можно сузить круг подходящих статей. Доступны следующие фильтры: фильтр по категории, фильтр по типу (Новость, Статья), фильтр по названию. Преподаватели и администраторы также имеют возможность

перейти к редактированию статьи или удалить статью прямо из списка на этой странице (Преподаватель может выполнить эти действия только со статьями, автором которых он является). Страница со списком статей изображена на рисунке 2.29.

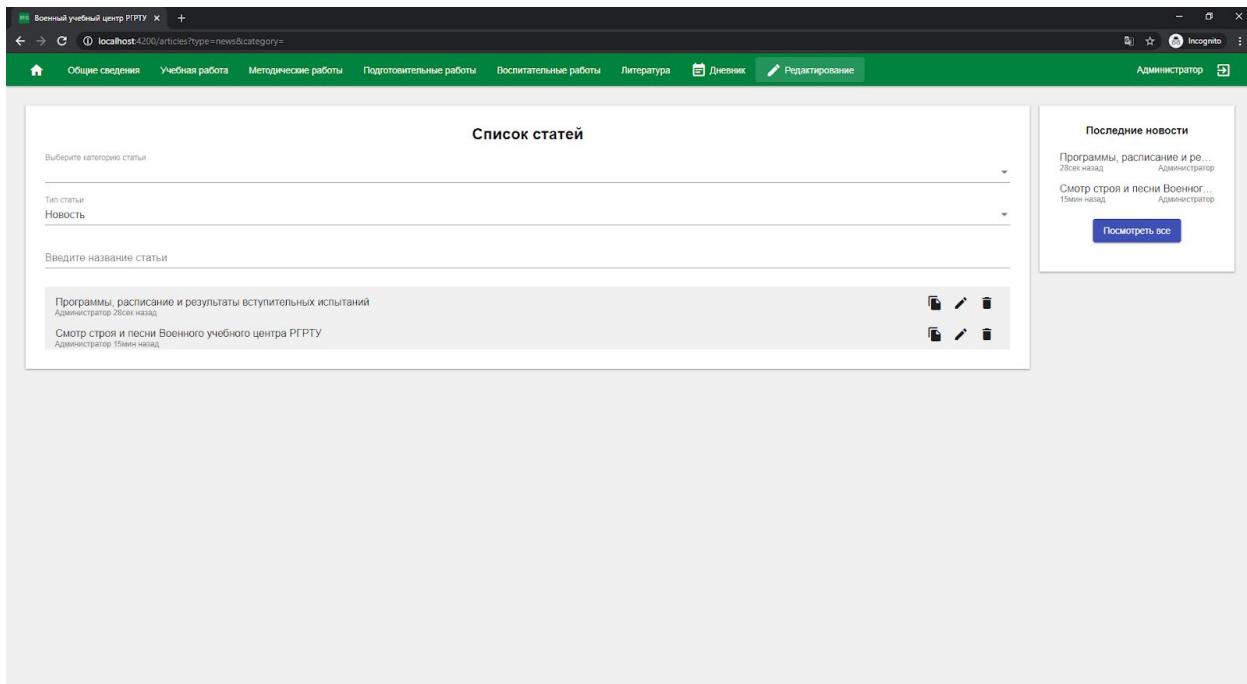


Рисунок 2.29 – Страница со списком статей

2.4.3 Модуль «Файлы»

В модуле «Файлы» находится страница со списком файлов. На этой странице присутствуют фильтры, с помощью которых можно сузить круг подходящих файлов. Доступны следующие фильтры: фильтр по категории, фильтр по названию. Также к файлам можно получить доступ посредством меню (зависит от того, как администратор настроит систему). Для того, чтобы отобразить файл используется встроенный в браузер функционал, если браузер открыть файл не может, то файл скачивается. Преподаватели и администраторы также имеют возможность удалить файл прямо из списка на этой странице (Преподаватель может удалить только им загруженные файлы). Страница со списком файлов изображена на рисунке 2.30.

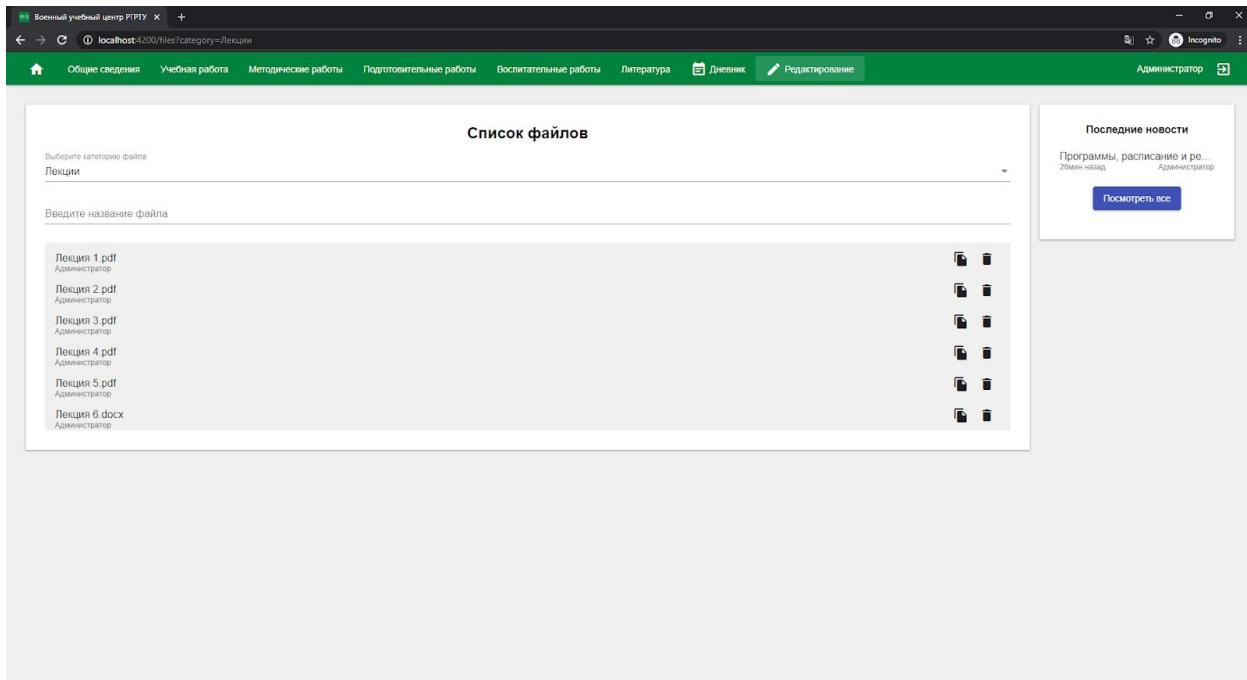


Рисунок 2.30 – Страница со списком файлов

2.4.4 Модуль «Тесты»

Одной из самых важных задач системы является проведение тестирования студентов. Модуль «Тесты» включает в себя 2 страницы:

1. Страница прохождения теста.

Эта страница предназначена для прохождения теста студентами или для проверки созданного теста преподавателями. Поведение страницы зависит от роли пользователя, который на нее зашел. Завершить тест может только студент, остальные группы пользователей могут проверить внешний вид теста, проверить тест. Была реализована возможность проведения тестов с ограниченным временем, в этих тестах по истечению заданного времени тест завершается автоматически.

Для каждого типа вопроса был разработан отдельный компонент, который отвечает за взаимодействие пользователя с ответами на этот вопрос. Также для каждого типа вопроса были созданы подсказки, для более легкого понимания механики работы системы.

Пример отображения страницы теста для студента изображен на рисунке 2.31, для администратора на рисунке 2.32.

Администратор

Тест про военное дело

1) Сколько звезд на погонах армейского капитана?



Погоны армейского капитана

- Четыре
- Две
- Три
- Пять

2) Что чаще всего военнослужащий получает вне очереди?



Ваш ответ
наряд



3) Расположите в хронологической последовательности исторические события.



- Ливонская война
- Отечественная война
- Крымская война
- Первая мировая война
- Великая Отечественная Война

4) Укажите существующие воинские звания.



- майор
- подполковник
- сержант
- гладиатор
- центурион

5) Установите соответствие между событиями и их датами.



1941-1945	→	Великая Отечественная Война
1914-1917	→	Первая мировая война

Закончить

Рисунок 2.31 – Пример отображения теста для студента

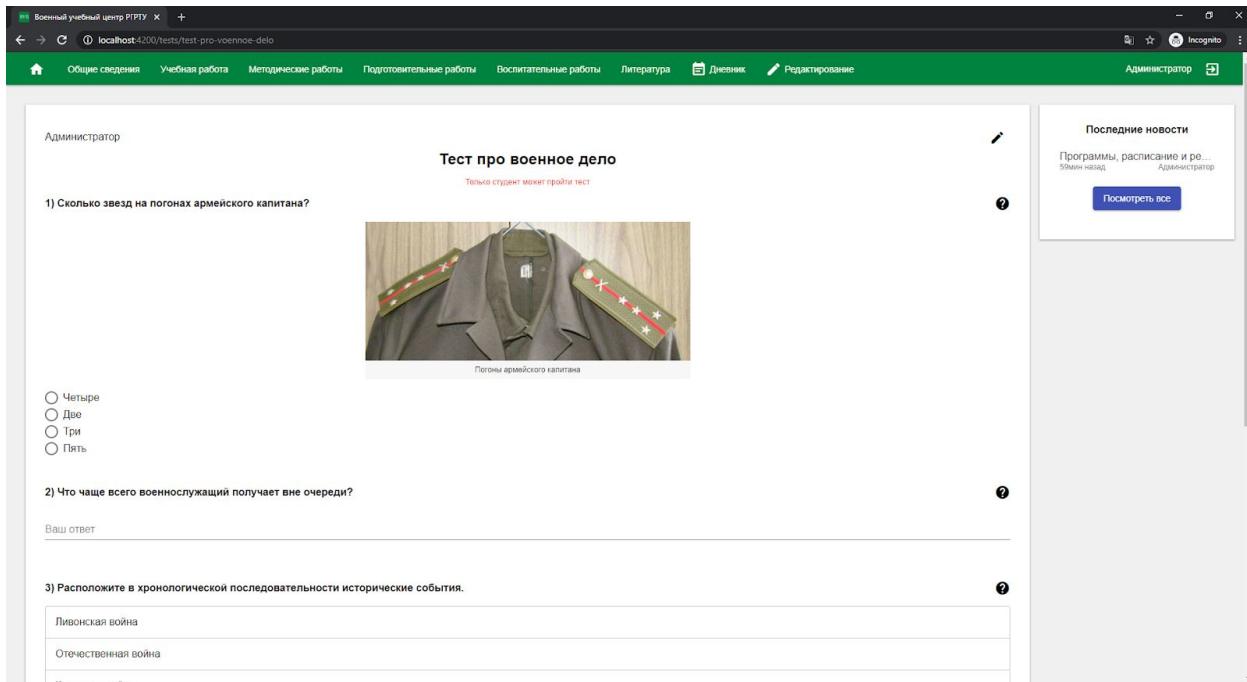


Рисунок 2.32 – Пример отображения теста для администратора

2. Страница со списком тестов.

Для того, чтобы получить возможность поиска тестов была реализована страница со списком тестов. На этой странице присутствуют фильтры, с помощью которых можно сузить круг подходящих тестов. Доступны следующие фильтры: фильтр по категории, фильтр по названию. Если тест был пройден то в нижней части соответствующего ему элемента отображается его результат (сдан он или нет) и дата завершения. Преподаватели и администраторы также имеют возможность перейти к редактированию теста или удалить его прямо из списка на этой странице (Преподаватель может выполнить эти действия только со тестами, создателем которых он является). Страница со списком тестов изображена на рисунке 2.33.

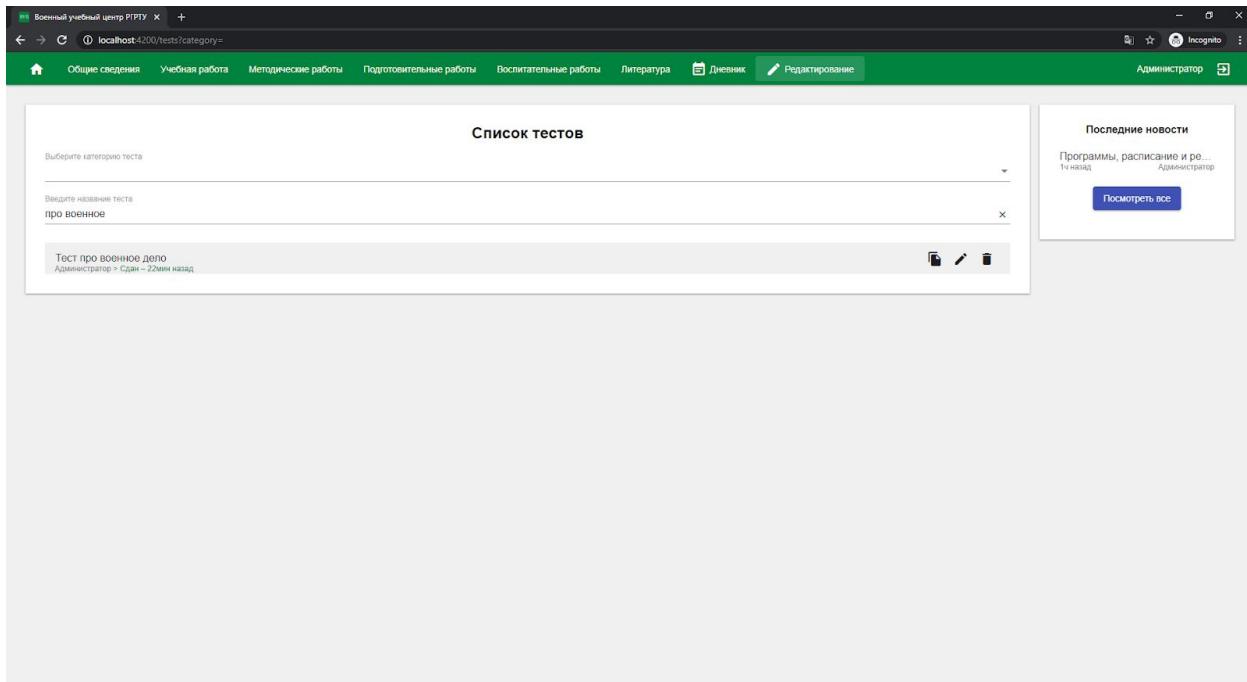


Рисунок 2.33 – Страница со списком тестов

2.4.5 Модуль «Дневник»

Преподавателям необходимо следить за успеваемостью студентов, для того чтобы облегчить эту задачу была реализована страница для просмотра результатов прохождения тестов студентами. На этой странице можно указать месяц за который требуется вывести информацию о студенческую группу. Доступ к этой странице имеют только преподаватели и администраторы.

Изначально данные в таблице представлены в агрегированном виде, при нажатии на строку в таблице, раскрываются детали. Страница с результатами тестов изображена на рисунке 2.34.

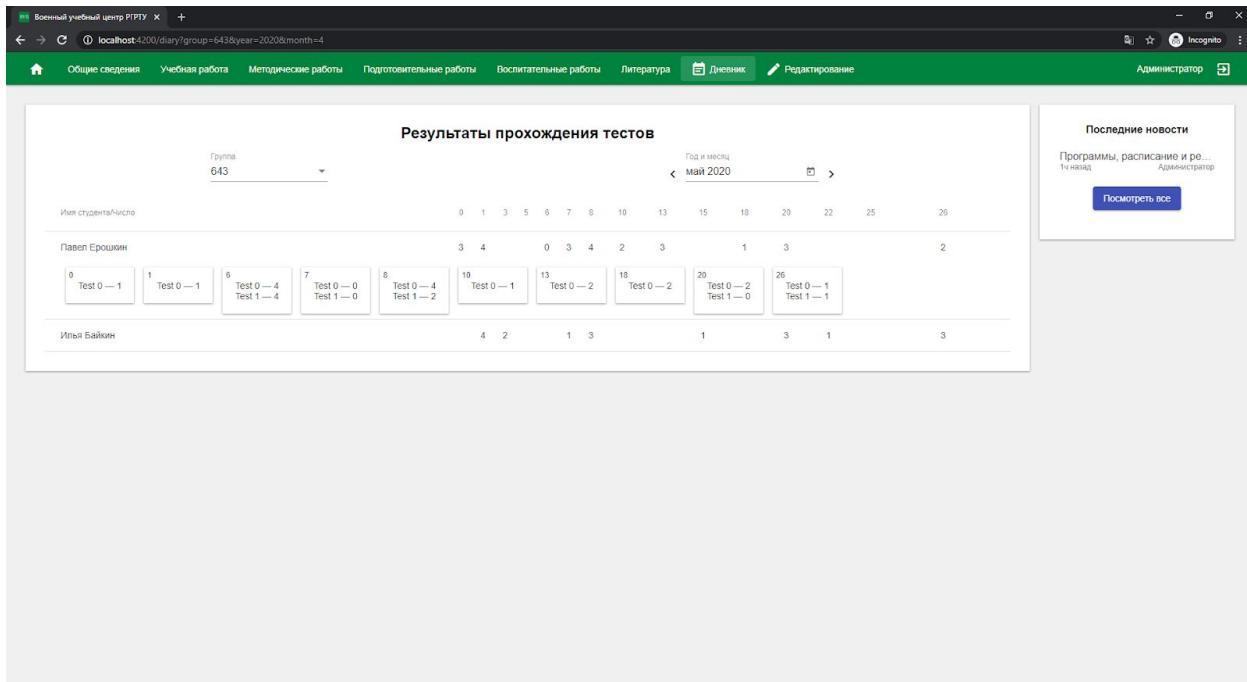


Рисунок 2.34 – Страница «Результаты прохождения тестов»

2.4.6 Модуль «Редактирование меню»

Данный модуль содержит страницу редактирования меню. Доступ к этой странице имеет только администратор. Для упрощения понимания принципа работы страницы, были сделаны подсказки. Для ускорения создания меню, были выделены типы пунктов меню:

- группа – пункт меню содержит в себе другие пункты меню;
- файл – пункт меню является ссылкой на файл;
- статья – пункт меню является ссылкой на статью;
- тест – пункт меню является ссылкой на тест;
- внешняя ссылка – пункт меню является внешней ссылкой.

Под каждый тип пункта меню был создан специальный конструктор, упрощающий и ускоряющий его создание. Например, для типа «Файл», не нужно искать ссылку на файл, а просто выбрать нужный файл из списка (рисунок 2.35).

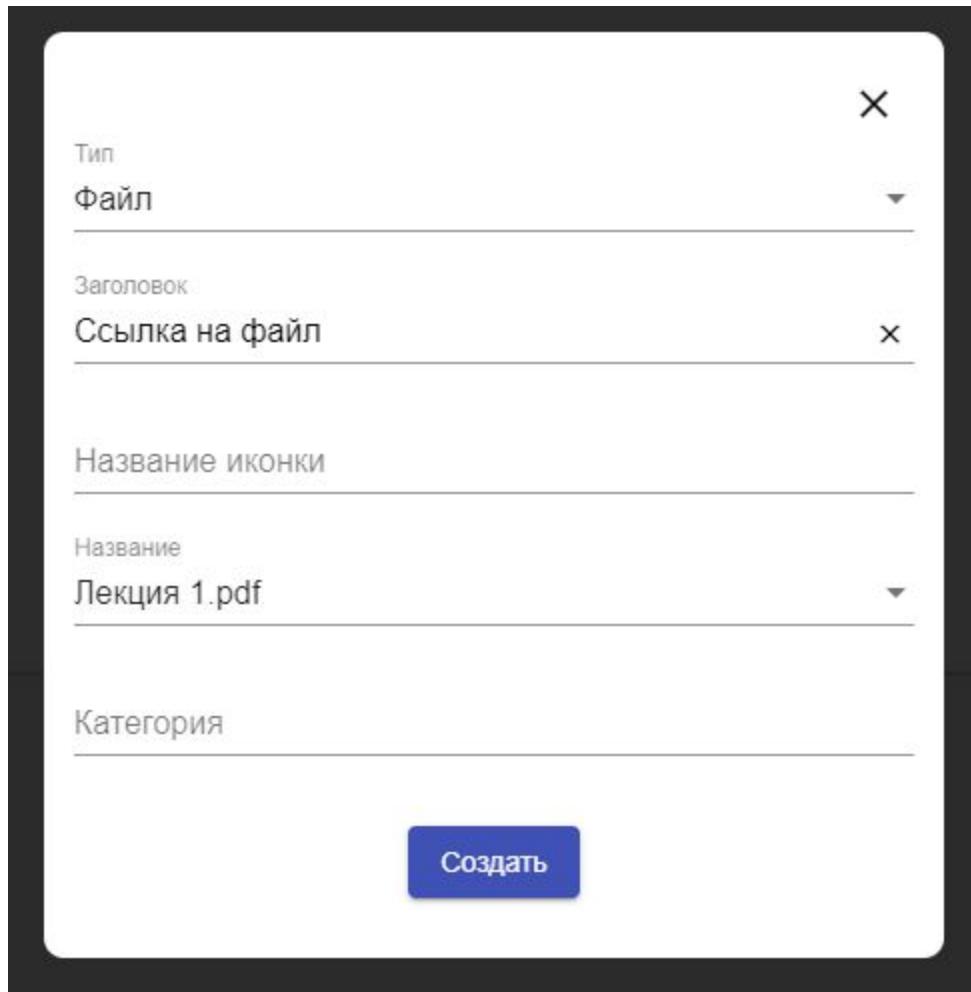


Рисунок 2.35 – Конструктор пункта меню типа «Файл»

Страница представляет собой дерево пунктов меню, элементы которого, помимо создания удаления, можно менять местами. Страница редактирования меню изображена на рисунке 2.36.

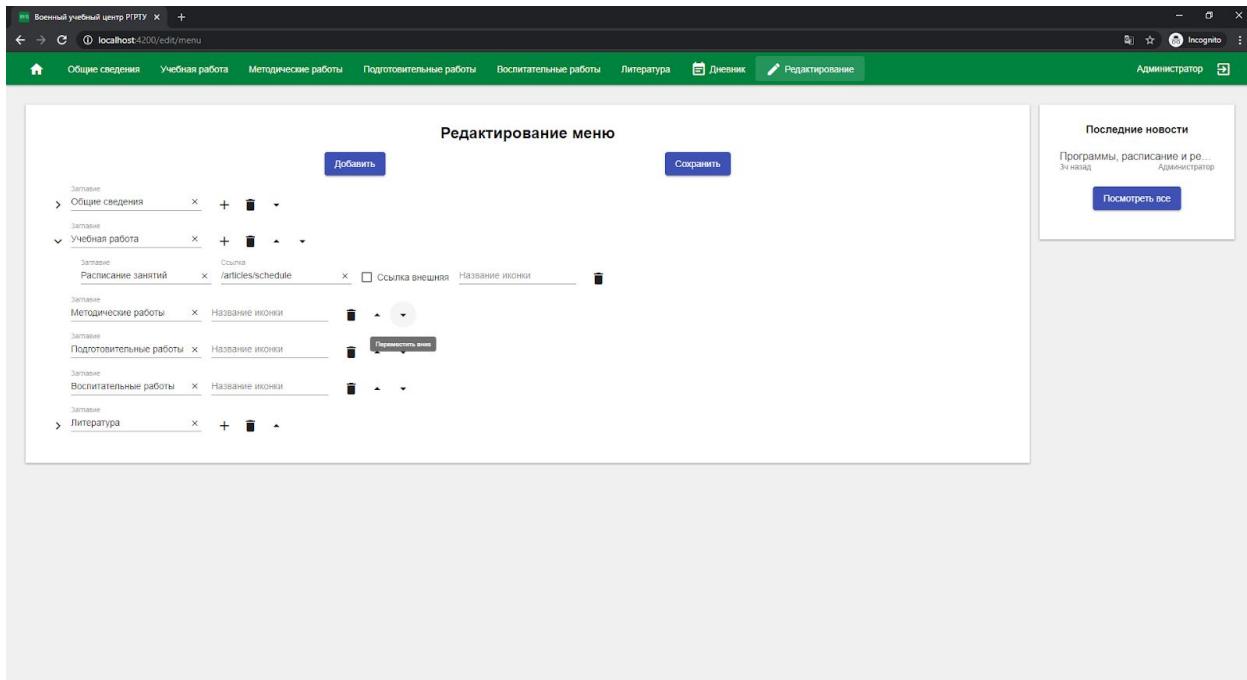


Рисунок 2.36 – Страница редактирования меню

2.4.7 Модуль «Редактирование пользователей»

Данный модуль предназначен для контроля пользователей. Содержит в себе страницу редактирования пользователей, доступ к которой имеет только администратор. Для того чтобы исключить возможность «поломки» учетной записи администратора, было решено запретить редактировать администраторов, поэтому редактировать можно только преподавателей и студентов.

Для удобства, для студентов и преподавателей были сделаны разные вкладки, студенты были объединены в группы. Механизм валидации не позволит сохранить пользователей с некорректными данными, например, двух пользователей с одинаковыми логинами (рисунок 2.37).

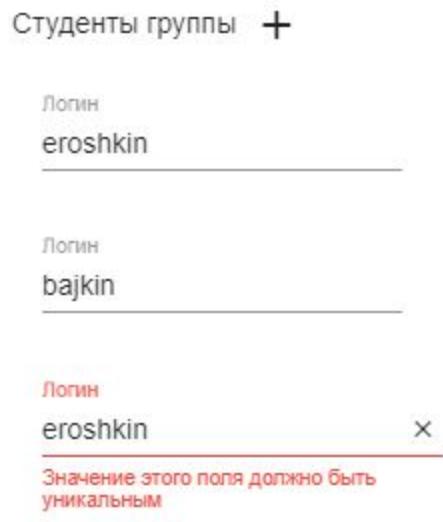


Рисунок 2.37 – Валидационное сообщение при попытке ввода повторного логина

Вкладка с редактированием студентов изображена на рисунке 2.38.

The screenshot shows the 'Редактирование пользователей' (Edit users) page. The top navigation bar includes tabs for 'Общие сведения', 'Учебная работа', 'Методические работы', 'Подготовительные работы', 'Вспомогательные работы', 'Литература', 'Дневник', 'Редактирование', and 'Администратор'. The main content area is titled 'Редактирование пользователей' and has a 'Сохранить' (Save) button. It features two tabs: 'Преподаватели' and 'Группы студентов', with 'Группы студентов' selected. A 'Добавить группу' (Add group) button is present. The 'Группы студентов' section displays a table with two rows:

Название группы	Логин	ФИО	Изменить пароль
643	eroshekin	Павел Ерошин	<input type="checkbox"/> Изменить пароль
	bajkin	Илья Байкин	<input type="checkbox"/> Изменить пароль

On the right side of the page, there is a sidebar titled 'Последние новости' (Recent news) with a link to 'Посмотреть все' (View all).

Рисунок 2.38 – Вкладка с редактированием студентов

2.4.8 Модуль «Редактирование статей»

Создание статей также выделено в отдельных модуль, так как использует относительно «тяжелую» библиотеку *CKeditor*. Данный модуль содержит страницу редактирования статей, которая используется как для редактирования уже существующий статей, так и для создания новых. Доступ к этой странице имеют только администраторы и преподаватели, преподаватели имеют ограниченный доступ – могут либо создавать новые, либо редактировать созданные ими статьи. Для создания новой статьи, в меню есть соответствующий пункт, для редактирования уже существующей статьи на эту страницу можно попасть либо из списка статей, либо со страницы конкретной статьи. При сохранении новой статьи ее заглавие посредством транслитерации преобразовывается в относительную ссылку на статью, если статья с такой ссылкой уже существует к ссылке на новую статью добавляется уникальное значение (случайно созданный *Guid* – статистически уникальный 128-битный идентификатор [22]).

Для создания статей предоставлен широкий набор функций, а именно:

1. Создание таблиц;
2. Вставка ссылкой;
3. Множество инструментов для работы с текстом (жирный шрифт, цитаты и так далее);
4. Поддержка отмены совершенного действия;
5. Вставка картинок (картинки при вставке сохраняются в формате *Base64* – стандарт кодирования двоичных данных при помощи только 64 символов *ASCII* [23]).

Для того, чтобы заранее посмотреть как примерно будет выглядеть статья в системе, был реализован механизм «предпросмотр» (рисунок 2.39).



Рисунок 2.40 – Предпросмотр статьи
Пример редактирования статьи изображен на рисунке 2.40.

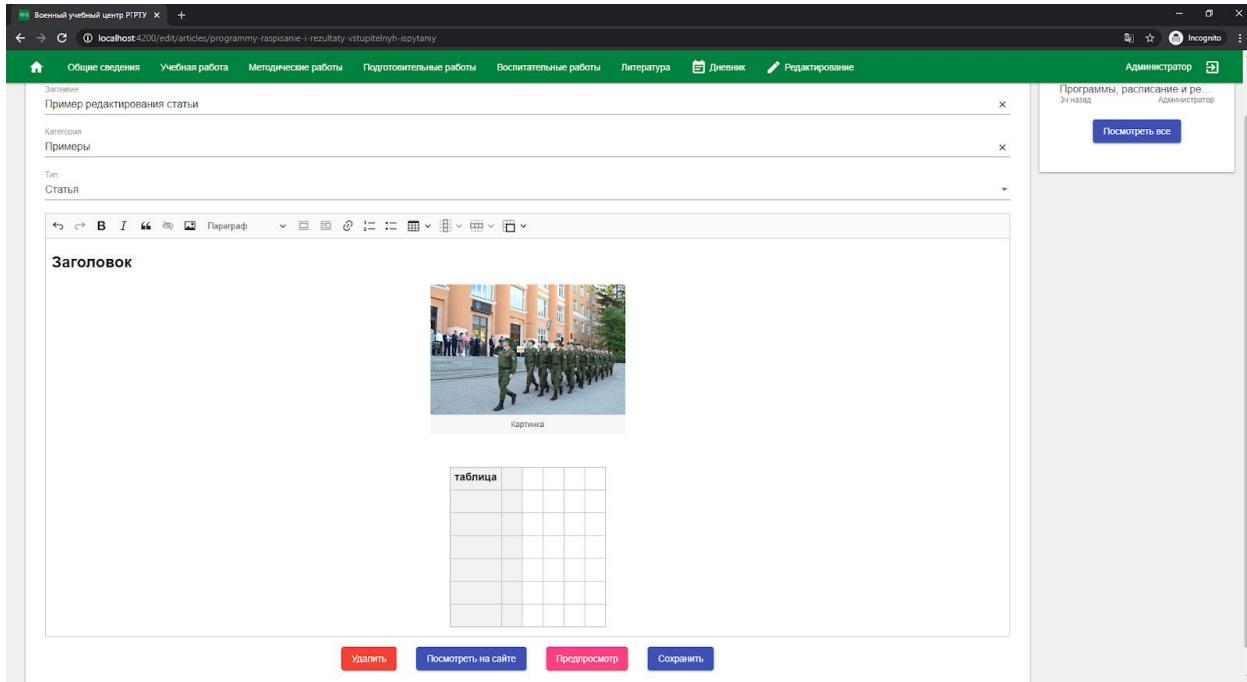


Рисунок 2.41 – Пример редактирования статьи

2.4.9 Модуль «Редактирование файлов»

Модуль «Редактирование файлов» содержит в себе страницу загрузки файлов. К данной странице имеют доступ преподаватели и администраторы. Поддерживается перетаскивание сразу нескольких файлов, или их выбор посредством стандартного файлового диалога установленной операционной системы. Страница загрузки файлов изображена на рисунке 2.42.

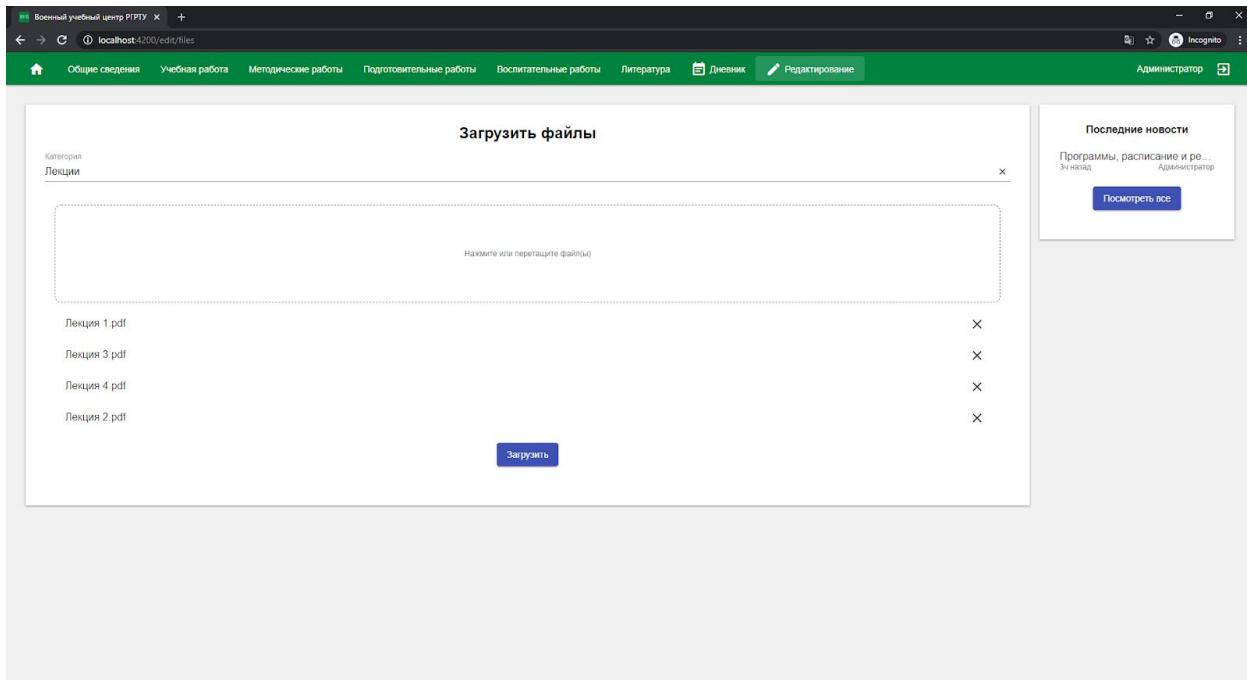


Рисунок 2.42 – Страница загрузки файлов

2.4.10 Модуль «Редактирование тестов»

Модуль «Редактирование тестов» предназначен для управления тестами в системе. Данный модуль содержит страницу редактирования тестов, к которой имеют доступ преподаватели (только к новым или созданным ими) и администраторы. Для каждого типа вопроса реализован отдельный компонент по созданию ответов на данный вопрос. Каждое поле ввода валидируется, как и на остальных страницах редактирования, и не даст сохранить некорректные данные. Помимо данных относящихся непосредственно к тесту, можно поставить флаг «Сбросить результаты для всех несдавших», что дает возможность студентам, которые провалили этот тест пройти его еще раз. В вопросы по желанию можно добавить поясняющий контент, используя тот же компонент что и при редактировании статей (пункт 2.4.8). Генерация ссылки на тест происходит также как и генерация ссылки на статью (пункт 2.4.8). Пример редактирования теста изображен на рисунке 2.43.

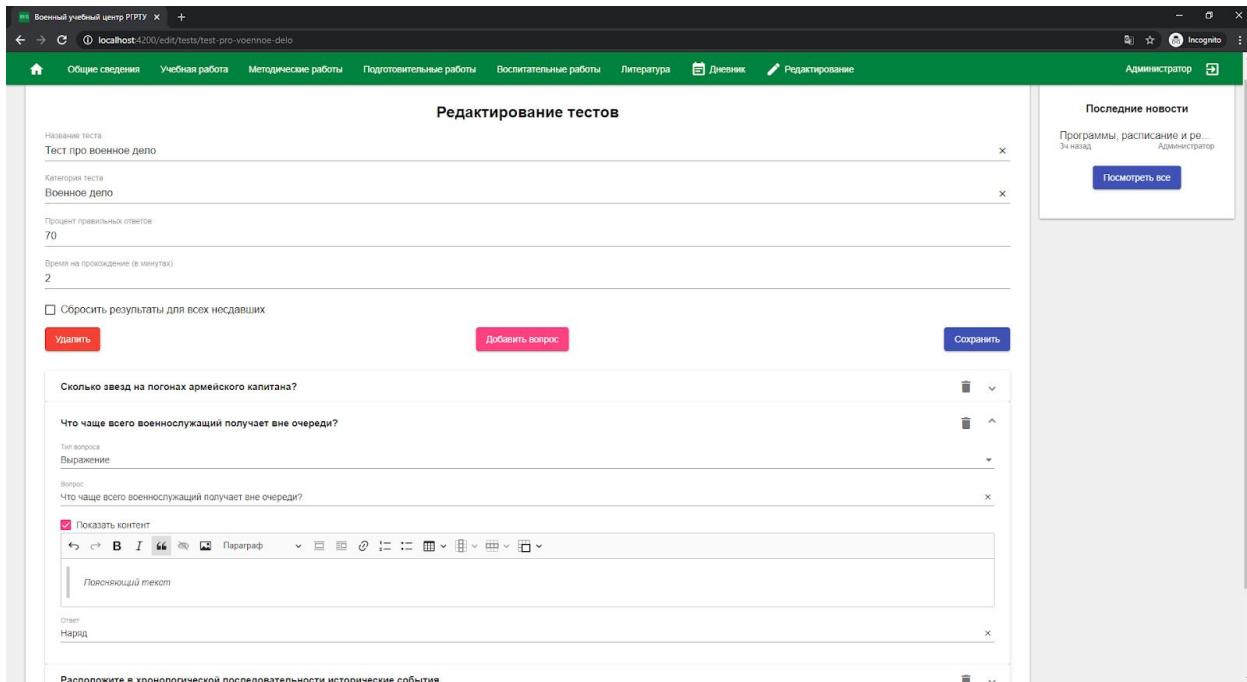


Рисунок 2.43 – Пример редактирования теста

2.4.11 Реализация мобильной версии

Для того, чтобы систему можно было использовать на мобильных устройствах, во время верстки страниц приложения применялась техника адаптивной верстки – дизайн, который подстраивается (адаптируется) под размер экрана, в том числе может происходить перестройка блоков с одного места на другое, или их замена блоками отображаемыми только при определенном разрешении. Адаптивная верстка пришла на смену идеи создания специальных мобильных версий сайта, «живущих» на отдельных поддоменах [21].

Так, например, блок новостей который в основной версии находится в правом верхнем углу, в мобильной версии находится внизу, а блок со страницами приложения растягивается на всю ширину экрана. Пример страниц мобильной версии изображен на рисунке 2.44.

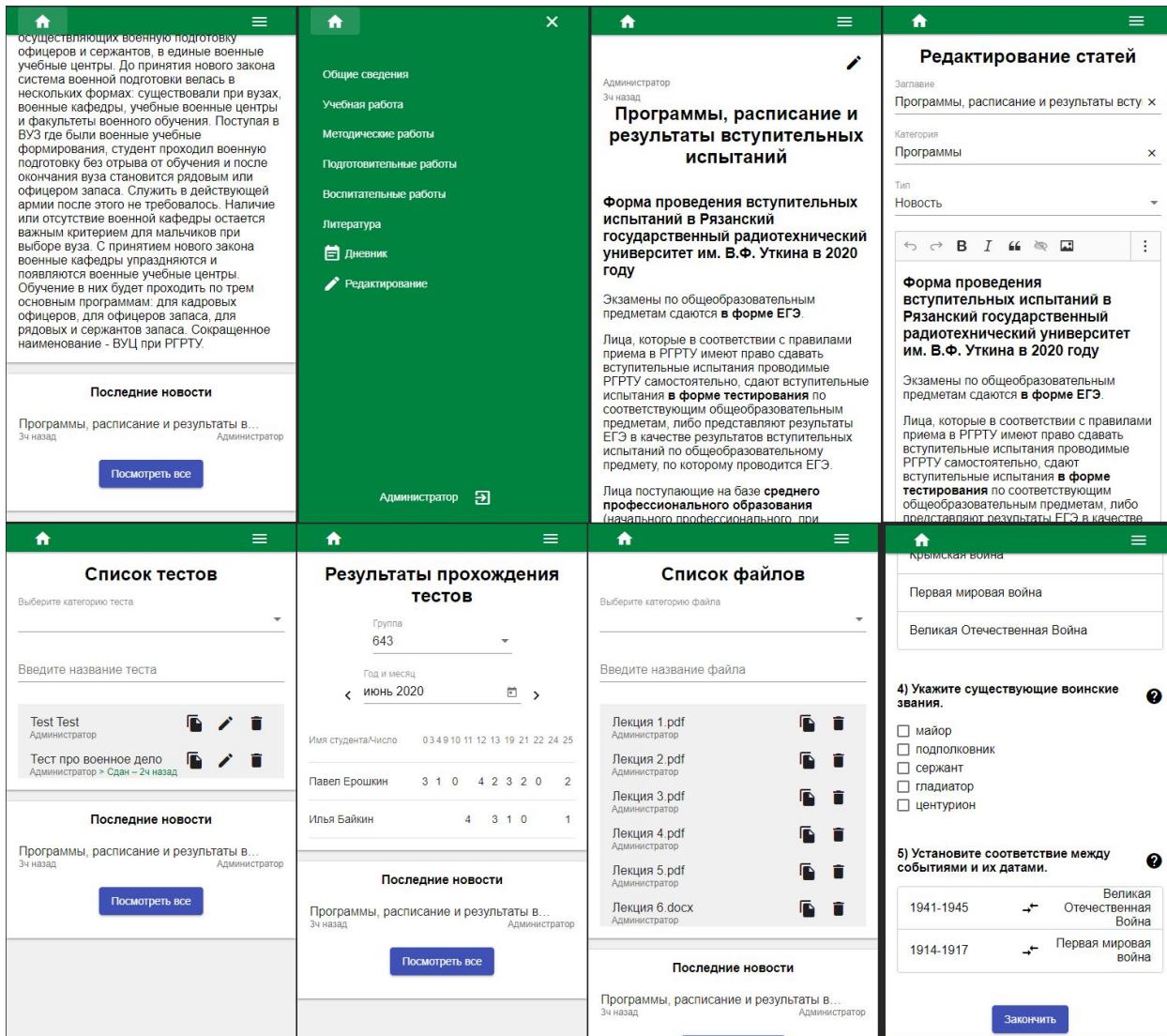


Рисунок 2.44 – Скриншоты некоторых страниц в мобильной версии системы

В данном разделе была описана реализация программной системы, приведены скриншоты пользовательского интерфейса, описаны механизмы работы с основными страницами веб-приложения.

3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Одним из ключевых этапов в разработке любого программного обеспечения является его тестирование – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом [25].

Было проведено несколько типов тестирования.

1. Юнит-тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки [26].

Для написания юнит-тестов была использована среда Jasmine – среда тестирования с открытым исходным кодом для JavaScript [28].

Тест-кейс – это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату [30].

Далее опишем тест-кейсы написанные к наиболее важным классам приложения.

Таблица 3.1 – тест-кейсы к классу *AuthorizationService*.

Номер	Описание	Ожидаемый результат	Статус
1	Проверка сохранения контекста пользователя в памяти после успешной авторизации. В teste имитируется ответ сервера на запрос авторизации, затем текущий контекст пользователя сопоставляется заданному.	Контекст пользователя после авторизации должен соответствовать заданному.	Пройден
2	Проверка того, при ответе сервера о неуспешной авторизации.	Неуспешная авторизация.	Пройден

	авторизации, система также посчитает авторизацию неуспешной.		
3	Проверка того, что при смене пользователя, система очистит <i>cookie</i> и контекст пользователя.	Удаление авторизационного токена из <i>cookie</i> , установка в <i>null</i> контекста пользователя.	Пройден
4	Проверка того, что при отсутствии авторизационного токена в <i>cookie</i> пользователя, система, при инициализации, не обратится к <i>Back end</i> для получения контекста пользователя.	Приложение не должно обратиться к <i>Back end</i> .	Пройден
5	Проверка того, что при наличии авторизационного токена в <i>cookie</i> пользователя, система, при инициализации, обратится к <i>Back end</i> для получения контекста пользователя и сохранит полученный контекст в памяти.	Обращение системы к <i>Back end</i> , сохранение контекста пользователя в памяти.	Пройден

Таблица 3.2 – тест-кейсы к классу *AuthorizationTokenStorage*.

Номер	Описание	Ожидаемый результат	Статус
1	Проверка сохранения авторизационного токена в <i>cookie</i> . В тесте происходит вызов метода <i>set</i> с параметром « <i>test_token_value</i> », затем проверяется, что такое	<i>Cookie</i> содержит значение « <i>test_token_value</i> »	Пройден

	значение содержится в <i>cookie</i> .		
2	Проверка правильности определения наличия авторизационного токена в <i>cookie</i> (метод <i>isSet</i>). В <i>cookie</i> сохраняется значение авторизационного токена, затем <i>cookie</i> очищаются.	После сохранения авторизационного токена в <i>cookie</i> , результат должен быть « <i>true</i> », после очищения <i>cookie</i> « <i>false</i> ».	Пройден
3	Проверка того, что метод <i>get</i> , всегда возвращает актуальное значение авторизационного токена.	Результат вызова метода <i>get</i> должен соответствовать действительности.	Пройден
4	Проверка удаления авторизационного токена из <i>cookie</i> (метод <i>clear</i>).	Отсутствие авторизационного токена в <i>cookie</i> .	Пройден

2. Сквозное тестирование (*E2E*) – тесты, которые тестируют систему в целом, имитируя реальную пользовательскую среду [27].

Для написания *E2E* тестов был использован фреймворк *Protractor* – фреймворк комплексного тестирования *Angular*-приложений [29].

Опишем наиболее важные сценарии.

Таблица 3.3 – наиболее важные сценарии *E2E* тестов.

Номер	Описание	Сценарий	Ожидаемый результат	Статус
1	Отображение ошибки авторизации при вводе данных пользователя, которого не существует в системе.	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести несуществующую пару логин-пароль. 4. Нажать кнопку «Войти».	Отображение ошибки о несуществующем пользователе.	Успешно

2	Переадресация на главную страницу после успешной авторизации.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 	Переадресация на главную страницу системы.	Успешно
3	Копирование ссылки на файл.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке <code>/files</code>. 6. На первом элементе списка файлов нажать на кнопку «Скопировать ссылку». 	Буфер обмена не должен быть пустым.	Успешно
4	Переход к просмотру файла из списка файлов.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке <code>/files</code>. 6. Нажать на первый элемент списка. 	Открытие в новой вкладке ссылки на файл	Успешно

5	Копирование ссылки на статью.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке <code>/articles</code>. 6. На первом элементе списка файлов нажать на кнопку «Скопировать ссылку». 	Буфер обмена не должен быть пустым.	Успешно
6	Переход к просмотру статьи из списка статей.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке <code>/articles</code>. 6. Нажать на первый элемент списка. 	Переход на страницу со статьей.	Успешно
7	Копирование ссылки на тест.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Перейти по относительной 	Буфер обмена не должен быть пустым.	Успешно

		<p>ссылке <i>/tests</i>.</p> <p>6. На первом элементе списка файлов нажать на кнопку «Скопировать ссылку».</p>		
8	Переход к тесту из списка тестов.	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p> <p>3. В форму авторизации ввести пару логин-пароль существующего пользователя.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке <i>/tests</i>.</p> <p>6. Нажать на первый элемент списка.</p>	Переход на страницу теста.	Успешно
9	Проверка наполнения страницы «Страница найдена».	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p> <p>3. В форму авторизации ввести пару логин-пароль существующего пользователя.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке <i>/not-found</i>.</p> <p>6. Нажать на кнопку «На главную».</p>	Отображение страницы «Страница не найдена», на странице должна присутствовать кнопка «На главную», при нажатии на которую должна произойти переадресация на главную страницу.	Успешно
10	Проверка работоспособности страницы «Результаты	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p>	Должна отобразиться таблица с результатами	Успешно

	прохождения тестов».	<p>3. В форму авторизации ввести пару логин-пароль существующего пользователя.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке /diary.</p> <p>6. Нажать на первую строку таблицы.</p> <p>7. Нажать повторно на ту же строку таблицы.</p>	прохождения тестов, при клике на строки которой, должны раскрываться (при повторном клике скрываться) детали результатов.	
11	Проверка корректного отображения статей.	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p> <p>3. В форму авторизации ввести пару логин-пароль существующего пользователя.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке /articles/test (путь к предопределенной статье в режиме разработки).</p>	Должен быть отображен заголовок статьи, ее тело и автор.	Успешно
12	Проверка прохождения тестов.	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p> <p>3. В форму авторизации ввести пару логин-пароль существующего студента.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке /tests/test (путь к</p>	После прохождения теста должен быть отображен результат.	Успешно

		<p>предопределенному тесту в режиме разработки).</p> <ol style="list-style-type: none"> 6. Нажать на кнопку «Начать». 7. Нажать на первый ответ первого вопроса (минимальный набор действий для прохождения этого теста). 8. Нажать на кнопку «Завершить». 		
13	Проверка работы вкладок на странице редактирования пользователей.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего администратора. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке /edit/users. 6. Нажать на заголовок вкладки «Группы студентов». 	При нажатии на заголовок неактивной вкладки, она должна становиться активной и наполнение страницы должно изменяться.	Успешно
14	Проверка возможности перехода со страницы статьи к ее редактированию.	<ol style="list-style-type: none"> 1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего администратора. 4. Нажать кнопку «Войти». 5. Перейти по относительной ссылке /articles/test (путь к предопределенной статье в 	Должна отобразиться страница редактирования статьи.	Успешно

		<p>режиме разработки).</p> <p>6. Нажать на кнопку «Редактировать» (изображена в виде «карандаша»).</p>		
15	Проверка перехода к редактированию теста из списка тестов.	<p>1. Запустить браузер в режиме «инкогнито».</p> <p>2. Перейти по адресу приложения.</p> <p>3. В форму авторизации ввести пару логин-пароль существующего пользователя.</p> <p>4. Нажать кнопку «Войти».</p> <p>5. Перейти по относительной ссылке /tests.</p> <p>6. Нажать на кнопку «Редактировать» (изображена в виде «карандаша») у первого элемента списка.</p>	<p>Должна отобразиться страница редактирования теста.</p>	Успешно

Протестировав разработанную систему, можно сделать вывод о ее работоспособности и о выполнении предъявленных требований к системе.

4. ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ

4.1. Описание применения

4.1.1. Назначение программы

Система предназначена для автоматизации части процесса обучения в ВУЦ РГРТУ и обладает следующими особенностями:

- создание, удаление и изменение пользователей;
- имеет динамически редактируемое меню с поддержкой вложенности;
- загрузка файлов в систему и просмотр этих файлов;
- написания статей, тестов и блок новостей;
- тестирование студентов с сохранением результатов прохождения в БД и возможностью просмотреть эти результаты преподавателями.

4.1.2. Условия применения

Для запуска приложения подойдет любой современный веб-браузер, например *Chrome* 83, *IE* 11 или *Edge* 44, минимальные системные требования приложения будут соответствовать минимальным системным требованиям браузера. При открытии главной страницы системы размер загруженных скриптов приблизительно равен 1 Мб. Полный размер приложения при загрузке всех модулей приблизительно составляет 3 Мб. При первой загрузке какого-либо компонента приложения он кешируется в браузере и дальнейшее обращение к этому компоненту загружает минимальное количество данных с сервера.

4.1.3. Описание задачи

Основной задачей, которую решает разработанное программное обеспечение является автоматизация части процесса обучения в ВУЦ РГРТУ.

Для решения этой задачи разработанная система предоставляет определенный набор функций, таких как проведение тестирования, написание статей, загрузка файлов.

4.2. Руководство оператора

4.2.1. Назначение программы

О назначении программы было сказано в пункте 4.1.1.

Приложение предоставляет следующие функции:

- авторизация пользователя в системе;

- просмотр, добавление, удаление и изменения пользователей системы;
- просмотр и изменение меню системы;
- просмотр результатов тестирования студентов;
- просмотр основных страниц системы;
- просмотр, добавление, удаление и изменение статей;
- просмотр, добавление, удаление и изменение тестов;
- просмотр, добавление и удаление файлов;
- просмотр новостей.

4.2.2. Условия применения программы

Условия применения описаны в пункте 4.1.2.

4.2.3. Выполнение программы

Для того чтобы начать работу с системой, необходимо открыть веб-браузер и перейти по адресу, выделенному системе. Если пользователь открывает приложение в первый раз или его авторизационный токен невалидный (отсутствует или истек срок действия), то он попадает на страницу авторизации (рисунок 2.26). Для продолжения работы с системой пользователю необходимо ввести логин и пароль в соответствующие поля ввода, затем нажать кнопку «Войти». Если авторизация прошла успешно, то пользователь будет перенаправлен на главную страницу (рисунок 2.23).

Большинство кнопок системы, изображенных в виде иконок, имеет подсказки, посмотреть которые можно наведя курсор мыши на соответствующую кнопку.

После того, как пользователь будет авторизован, на каждой странице будет отображаться блок новостей и меню. Помимо редактируемых пунктов меню, существуют также предопределенные, такие как «Домой», «Выход», «Редактирование» и «Дневник». Предопределенные пункты меню отображается по разному, в зависимости от роли пользователя.

1. Администратор. Для администратора видны предопределенные пункты меню (рисунок 4.1).
2. Преподаватель. Для преподавателя видны все предопределенные пункты меню за исключением некоторых подпунктов пункта «Редактирование» – «Меню» и «Пользователи» (рисунок 4.2).

3. Студент. Для студента из предопределенных пунктов меню видны только пункты меню «Домой» и «Выйти» (рисунок 4.3).

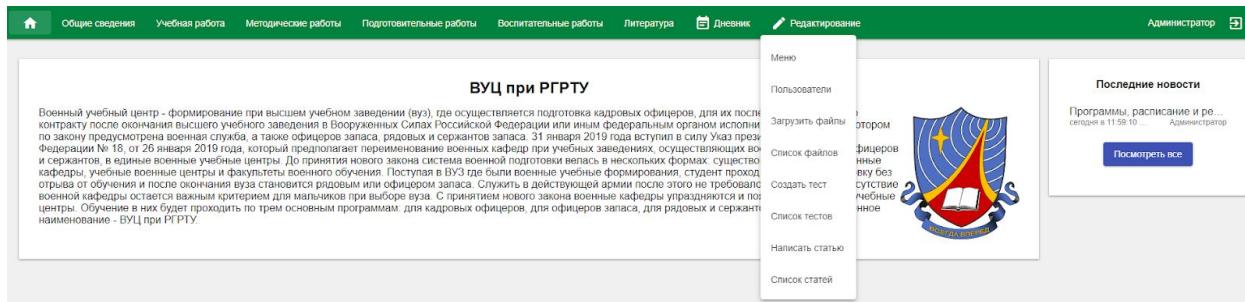


Рисунок 4.1 – Пример отображения меню для администратора



Рисунок 4.2 – Пример отображения меню для преподавателя

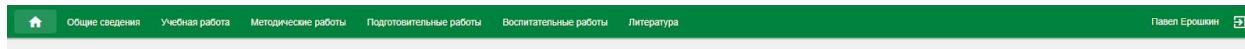


Рисунок 4.3 – Пример отображения меню для студента

Администратор может редактировать меню, для этого необходимо в пункте меню «Редактирование» выбрать подпункт «Меню», отобразится страница редактирования меню (рисунок 2.36). На этой странице администратор может добавлять, удалять или изменять пункты меню. Для того чтобы добавить корневой пункт меню необходимо нажать кнопку «Добавить», после чего отобразится конструктор пункта меню (рисунок 2.35), после заполнения всех необходимых полей ввода нужно нажать кнопку «Создать», пункт меню добавится как последний корневой элемент дерева. Создать подпункт меню можно таким же образом, как и корневой, только нажав на «плюс» у соответствующего пункта меню. Для того, чтобы удалить пункт меню, необходимо нажать на «корзину» в правой части нужного элемента дерева. Пункты меню также можно передвигать, используя стрелки в правой части. Для того, чтобы скрыть или раскрыть элемент дерева,

необходимо нажать на стрелку слева элемента. После того, как администратор внесет необходимые изменения в меню, требуется нажать кнопку «Сохранить», система запросит подтверждение сохранения.

Для редактирования пользователей администратору необходимо в пункте меню «Редактирование» выбрать подпункт «Пользователи», отобразится страница редактирования пользователей (рисунок 2.38). На этой странице администратор может добавлять, удалять или изменять пользователей. Переключаться между вкладками, можно нажав на соответствующую вкладку подпись («Преподаватели», «Группы студентов»). Чтобы добавить какой-либо элемент необходимо нажать на соответствующий «плюс», чтобы удалить – на кнопку «корзина». Чтобы сменить пароль уже существующему пользователю, необходимо поставить галочку в поле «Изменить пароль», после чего отобразится поле ввода для пароля. После того, как администратор внесет необходимые изменения в список пользователей, требуется нажать кнопку «Сохранить», система запросит подтверждение сохранения.

Для того, чтобы просмотреть результаты прохождения тестов администратору или преподавателю требуется нажать на пункт меню «Дневник», отобразится страница с результатами прохождения тестов студентами (рисунок 2.34). На этой странице возможен выбор года и месяца результатов и студенческой группы.

Для того, чтобы написать статью администратору или преподавателю необходимо в пункте меню «Редактирование» выбрать подпункт «Написать статью», отобразится страница редактирования статей (рисунок 2.41). После заполнения всех полей ввода требуется нажать кнопку «Сохранить», система запросит подтверждение сохранения. Для того, чтобы удалить статью, необходимо на странице редактирования статьи нажать кнопку «Удалить» или в пункте меню «Редактирование» выбрать подпункт «Список статей» и на полученной странице (рисунок 2.29) найти нужную статью и нажать кнопку «Удалить», в обоих случаях система запросит подтверждение удаления.

Для того, чтобы создать тест администратору или преподавателю необходимо в пункте меню «Редактирование» выбрать подпункт «Создать

тест», отобразится страница редактирования тестов (рисунок 2.43). После заполнения всех полей ввода требуется нажать кнопку «Сохранить», система запросит подтверждение сохранения. Для того, чтобы удалить тест, необходимо на странице редактирования теста нажать кнопку «Удалить» или в пункте меню «Редактирование» выбрать подпункт «Список тестов» и на полученной странице (рисунок 2.33) найти нужный тест и нажать кнопку «Удалить», в обоих случаях система запросит подтверждение удаления. Если необходимо разрешить не сдавшим тест студентам пройти его снова, то, при редактировании уже существующего теста, следует поставить флаг «Сбросить результаты для всех несдавших».

Для того, чтобы загрузить файлы администратору или преподавателю необходимо в пункте меню «Редактирование» выбрать подпункт «Загрузить файлы», отобразится страница загрузки файлов (рисунок 2.42). После выбора файлов и заполнения всех полей ввода требуется нажать кнопку «Загрузить», система запросит подтверждение загрузки. Для того, чтобы удалить файл, необходимо в пункте меню «Редактирование» выбрать подпункт «Список файлов» и на полученной странице (рисунок 2.30) найти нужный файл и нажать кнопку «Удалить», система запросит подтверждение удаления.

Чтобы пройти тест, студенту необходимо перейти на страницу соответствующего теста (это можно сделать несколькими способами, например, посредством меню, зависит от настройки системы) и нажать кнопку «Начать», после того как тест будет отображен, студенту необходимо ответить на его вопросы, взаимодействуя со специальными компонентами (пример прохождения теста изображен на рисунке 2.31). После того, как все вопросы будут отвечены, необходимо нажать на кнопку «Завершить», затем отобразится результат прохождения теста.

Для просмотра статьи, необходимо попасть на страницу с нужной статьей. Сделать это можно несколькими способами. Если статья является одной из последних новостей (количество последних зависит от настроек системы), то попасть на нее можно посредством нажатия на соответствующий статье элемент в блоке новостей. Статья также может быть внесена в меню, тогда попасть на нее можно посредством клика на

соответствующий пункт меню. Последним способом является поиск статьи в списке статей.

Для просмотра или скачивания файла, необходимо найти его в меню (если он был туда внесен) или в списке файлов.

Для того, чтобы скопировать ссылку на файл, статью или тест необходимо перейти на соответствующую страницу со списком контента, найти интересующий элемент и нажать кнопку «Скопировать ссылку».

Для того чтобы сменить пользователя необходимо нажать на пункт меню «Выйти», расположенный рядом с именем текущего пользователя в меню.

Чтобы завершить работу с системой пользователю необходимо закрыть вкладку браузера, в которой запущено приложение, или сам браузер.

4.2.4. Сообщения оператору

В процессе работы с системой оператор может получать 2 типа сообщений: уведомления и подтверждение действия.

Уведомления, в свою очередь, могут быть об успешном совершении какого-либо действия (рисунок 4.4) или о том, что действие совершить не удалось (рисунок 4.5). Тексты возможных уведомлений и ситуации при которых они отображаются приведены ниже (тексты могут быть изменены, так как хранятся в БД ЭИОС).

1. «Извините, система в данный момент недоступна. Пожалуйста, попробуйте позже.» – во время ошибки при обращении к *Back end*.
2. «Внесенные вами изменения успешно сохранены.» – после успешного сохранения чего-либо.
3. «Ваши изменения сохранить не удалось, пожалуйста попробуйте позже.» – при неуспешном сохранения чего-либо.
4. «Возникла ошибка при удалении элемента, пожалуйста попробуйте позже.» – при неуспешном удаления чего либо.
5. «Элемент был успешно удален.» – после успешного удаления чего-либо.
6. «Ссылка была успешно скопирована.» – после успешного копирования ссылки на контент.

7. «Возникла ошибка при копировании ссылки, пожалуйста, попробуйте позже.» – при неуспешном копировании ссылки на контент.

Пользователь может скрыть уведомление самостоятельно, либо оно будет скрыто автоматически через короткое время.

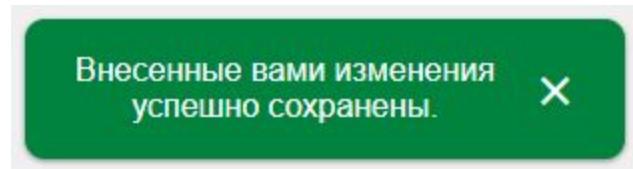


Рисунок 4.4 – Пример уведомления об успешно совершенном действии

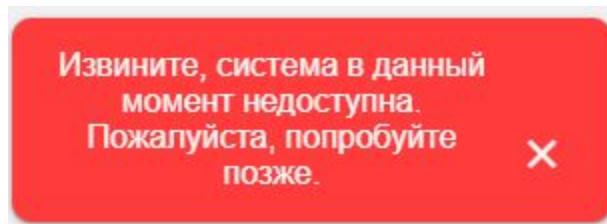


Рисунок 4.5 – Пример уведомления о действии, выполнить которое не удалось

Подтверждения действия отображаются во время сохранения или удаления элементов системы. При получении такого сообщения пользователю необходимо, подтвердить или отменить совершение действия, нажав на соответствующие кнопки. Пример сообщения изображен на рисунке 4.6.

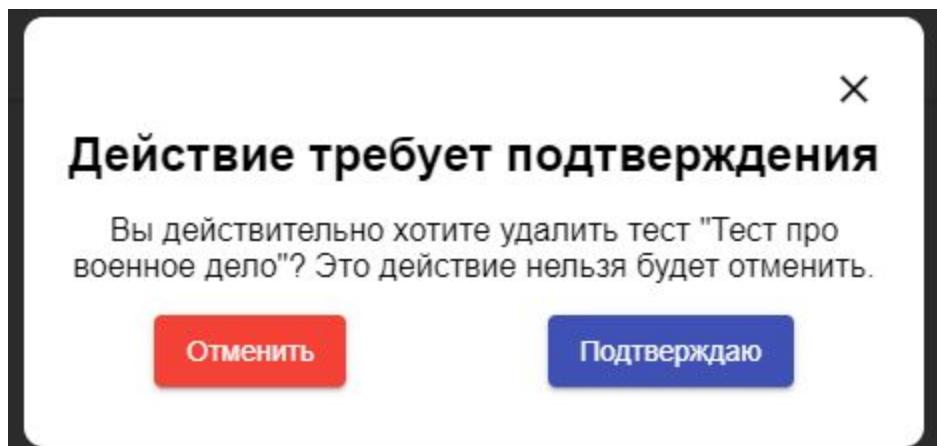


Рисунок 4.6 – Пример сообщения о подтверждении какого-либо действия

4.3. Руководство программиста

4.3.1. Назначение и условия применения программы

О назначении программы было сказано в пункте 4.2.1.

Необходимое программное обеспечение:

- *JetBrains WebStorm*;
- *Node.js*.

Минимальные системные требования соответствуют минимальным системным требованиям к необходимому программного обеспечению:

- *RAM – 8Гб;*
- Место на диске – 3.5Гб;
- Разрешение экрана – 1024x768;
- Операционная система – 64-битная версия Microsoft Windows 8 и выше или macOS 10.13 и выше, также любой дистрибутив Linux, который поддерживает Gnome, KDE или Unity DE.

Перед тем как скомпилировать проект в первый раз, необходимо выполнить консольную команду «`npm install`» для того, чтобы установить зависимости проекта.

4.3.2. Характеристика программы

В программе предопределено несколько видов компиляции:

- Компиляция в окружении разработки – консольная команда «`ng build`» – результат компиляции находится по относительному папке с проектом пути – *dist\military-department*;
- Компиляция в окружении «`production`» – консольная команда «`ng build --prod`» – результат компиляции находится по относительному папке с проектом пути – *dist\production*;
- Компиляция в окружении «`production`» для *IIS* (проприетарный набор серверов для нескольких служб Интернета от компании *Microsoft* [24]) – консольная команда «`npm run build-iis`» – результат компиляции находится по относительному папке с проектом пути – *dist\iis*.

Для того, чтобы проверить правильность выполнения программы необходимо выполнить следующие консольные команды:

- «ng test» – результатом выполнения команды будет запуск *Unit* тестов;
- «ng e2e» – результатом выполнения команды будет запуск *e2e* тестов.

4.3.3. Обращение к программе

Программу можно запустить в двух окружениях, в зависимости от цели запуска.

1. В окружении разработки.

В этом окружении удобно тестировать функции не связанные с работой с *Back end*. Для запуска в этом окружении не требуется наличие *Back end*, программа будет использовать специальные «заглушки», которые работают с данными в памяти или через «*Local storage*». Для запуска программы в этом режиме можно воспользоваться встроенным в среду разработки функционалом, выполнением команды «*ng serve*». Приложение также можно запустить используя любой другой веб-сервер, для этого необходимо скомпилировать проект в режиме разработки (описано в пункте 4.3.2) и используя результат компиляции выполнить инструкции по запуску веб-приложения для конкретного веб-сервера.

2. В окружении «production».

Для запуска в этом окружении обязательно наличие *Back end*. Для запуска с помощью *IIS*, необходимо скомпилировать проект следуя инструкциям описанным в пункте 4.3.2, и используя *IIS Manager* настроить новый сайт на папку с результатом компиляции. Следует отметить, что для корректной работы приложения в *IIS*, следует установить модуль *URL Rewrite*. Для запуска, используя другой веб-сервер, требуется скомпилировать проект в режиме «*production*» не для *IIS* (описано в пункте 4.3.2) и используя результат компиляции выполнить инструкции по запуску веб-приложения для конкретного веб-сервера.

4.3.4. Сообщения

Сообщения описаны в пункте 4.2.4.

4.4. Руководство системного программиста

4.4.1. Общие сведения о программе

О назначении и функциях программы было сказано в пункте 4.2.1.

Для того, чтобы запустить систему в общее пользование необходим веб-сервер, инструкции по его настройке описаны в пункте 4.3.3. Помимо веб-сервера требуется обеспечить связь с *Back end*.

4.4.2. Структура программы

Все файлы, находящиеся в папке с результатом компиляции, необходимы для корректной работы приложения. Приложение взаимодействует с *Back End* посредством *HTTP*-запросов.

4.4.3. Настройка программы

Настройку программы требуется делать на стороне *Back end*. Диаграмма классов настроек приложения изображена на рисунке 2.6.

4.4.4. Проверка программы

Настройку можно считать успешной, если при открытии приложения отображается какая-либо страница системы, кроме страницы изображенной на рисунке 2.22.

В данном разделе было описано применение разработанного программного обеспечения, составлены руководства оператора, программиста и системного программиста.

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы была достигнута. Созданное веб-приложение позволяет автоматизировать часть процесса обучения студентов ВУЦ РГРТУ. В процессе выполнения выпускной квалификационной работы были выполнены следующие задачи:

- проанализирована предметная область и существующие решения;
- обоснована необходимость разработки системы;
- составлены требования к системе;
- спроектирована и описана наиболее важные части системы;
- написан программный код системы, в том числе взаимодействующий в *Back end*;
- реализован пользовательский интерфейс;
- произведено тестирование приложения;
- составлена документация программному продукту.

Полученный программный продукт позволяет создавать, удалять и изменять пользователей, загружать и просматривать файлы, публиковать статьи и новости, проводить тестирование студентов с автоматическим сбором результатов.

Разработанное веб-приложение имеет удобный и многофункциональный интерфейс, который позволяет пользователям эффективно взаимодействовать с ЭИОС и корректирует их действия, указывая им на ошибки. Для использования системы необходим лишь веб-браузер, который доступен почти для любого устройства.

В дальнейшем систему можно доработать, используя следующие направления:

- сделать ее персонализированной для каждого студента, добавив, например, персональные задания;
- доработать «Электронный дневник», добавив туда, например, функцию ручного ввода оценок и расписание занятий или работу с компетенциями.
- провести языковую локализацию системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

9. Электронная информационно-образовательная среда МГЭУ [Электронный ресурс]. URL: <http://www.szfmgei.ru/elektronnaya-informatsionno-obrazovatelnaya-sreda/> (дата обращения: 29.03.2020).
10. CKeditor — Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/CKeditor> (дата обращения: 29.03.2020).
11. HTML — Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/HTML> (дата обращения: 29.03.2020).
12. JSON — Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/JSON> (дата обращения: 03.04.2020).
13. Stylus (язык таблиц стилей) — Википедия [Электронный ресурс]. URL: [https://ru.wikipedia.org/wiki/Stylus_\(%D1%8F%D0%B7%D1%8B%D0%BA_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86_%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B9\)](https://ru.wikipedia.org/wiki/Stylus_(%D1%8F%D0%B7%D1%8B%D0%BA_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86_%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B9)) (дата обращения: 29.03.2020).
14. TypeScript — Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki>TypeScript> (дата обращения: 29.03.2020).
15. Introduction to Angular concepts [Электронный ресурс]. URL: <https://angular.io/guide/architecture> (дата обращения: 30.05.2020).
16. Introduction to modules [Электронный ресурс]. URL: <https://angular.io/guide/architecture-modules> (дата обращения: 30.05.2020).
17. Introduction to components and templates [Электронный ресурс]. URL: <https://angular.io/guide/architecture-components> (дата обращения: 30.05.2020).
18. Introduction to services and dependency injection [Электронный ресурс]. URL: <https://angular.io/guide/architecture-services> (дата обращения: 30.05.2020).
19. Lazy-loading feature modules [Электронный ресурс]. URL: <https://angular.io/guide/lazy-loading-ngmodules> (дата обращения: 30.05.2020).

- 28.Jasmine (JavaScript testing framework) [Электронный ресурс]. URL: [https://en.wikipedia.org/wiki/Jasmine_\(JavaScript_testing_framework\)](https://en.wikipedia.org/wiki/Jasmine_(JavaScript_testing_framework)) (дата обращения: 31.05.2020).
- 29.Protractor - end-to-end testing for AngularJS URL: <https://www.protractortest.org/#/> (дата обращения: 31.05.2020).
- 30.Пишем максимально эффективный тест-кейс URL: <https://habr.com/ru/post/246463/> (дата обращения: 31.05.2020).