

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

На тему

**Программные средства для организации и
управления курсами по подготовке школьников к
ОГЭ и ЕГЭ**

Студент

(подпись, дата)

Левин Максим Вячеславович
(ФИО полностью)

Руководитель

(подпись, дата)

Балашова И.Ю.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Левин Максим Вячеславович гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ

Тема утверждена приказом ПГУ № _____ от "_____" _____ 20____

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Требуется разработать программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ.

2) Среда выполнения: операционная система Ubuntu 20.04, Web-браузер

3) Языковые средства разработки: TypeScript, UML

4) Инструментальные средства: VS Code, Microsoft SQL Server 2018, SQL Server Management Studio, Microsoft Project 2013, Visual Paradigm 17.2, Draw.io, NPM

5) Основные функции:

а) регистрация и авторизация;

б) просмотр списка учебных курсов;

в) приобретение учебных курсов;

г) получение расписания занятий и списка домашних работ;

д) выполнение самостоятельных работ;

е) создание самостоятельных работ;

ж) проверка самостоятельных работ;

з) получение статистики по учебе.

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи
2) Проектирование
3) Реализация
4) Контроль качества программного обеспечения
5) План разработки и оценка бюджета
6) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	План разработки бюджета	5%	23.05.2024 – 25.06.2024	
6	Оформление пояснительной записки	15%	26.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 95 листов, 54 рисунка, 7 таблиц, 12 использованных источников и 7 приложений.

ОНЛАЙН ШКОЛА, УЧЕБНЫЙ КУРС, ОНЛАЙН ЗАНЯТИЕ, ЭКЗАМЕН, ОГЭ, ЕГЭ, UML, БАЗА ДАННЫХ, MICROSOFT SQL SERVER, TYPESCRIPT, NEXTJS, NESTJS, TYPEORM, REST API

Объектом исследования являются средства автоматизации управления и организации учебными курсами.

Предметом исследования являются программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ.

Разработка базы данных проводилась в MICROSOFT SQL SERVER 2018 на языке T-SQL. Разработка приложения проводилась на языке программирования TypeScript с использованием фреймворков NextJS и NestJS. В качестве среды разработки использовалась Visual Studio Code.

Среда исполнения – Linux, web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная части приложения онлайн-школы по подготовке школьников к ОГЭ и ЕГЭ.

					ПГУ 09.03.04 – 08BP201.14 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ Пояснительная записка	Лит.	Лист	Листов
Разраб.		Левин М.В.						
Провер.		Балашова И.Ю.					4	95
Реценз.						Группа 20ВП1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Постановка задачи и анализ требований для программных средств, реализующих функционал онлайн-школы	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Обзор программных средств аналогичного назначения.....	11
1.2.1 Онлайн платформа Maximum Education.....	11
1.2.2 Онлайн-школа Фоксфорд.....	12
1.2.3 Образовательный портал Школково.....	13
1.2.4 Онлайн-школа Вебиум	15
1.2.5 Образовательный портал Сдам ГИА	16
1.2.6 Сравнение продуктов-аналогов.....	17
1.3 Анализ функциональных требований.....	18
1.4 Анализ нефункциональных требований.....	21
2 Проектирование программных средств для онлайн-школы.....	22
2.1 Архитектура приложения.....	22
2.2 Проектирование структур данных	23
2.2.1 Концептуальная модель базы данных	23
2.2.2 Логическая модель базы данных.....	24
2.3 Проектирование программных средств.....	26
2.4 Проектирование пользовательского интерфейса	29
3 Реализация программных средств для онлайн-школы.....	42
3.1 Развертывание базы данных в среде MS SQL Server	42
3.2 Наполнение базы тестовыми данными	44

3.3 Разработка программного обеспечения онлайн-школы	45
4 Контроль качества программного обеспечения	50
4.1 Выбор режима тестирования	50
4.2 Тестирование серверной части приложения	51
4.3 Тестирование клиентской части приложения	53
4.4 Метрики кода	54
5 Планирование разработки и оценка бюджета	58
Заключение	60
Список использованных источников	62
Приложение А (справочное) Глоссарий	64
Приложение Б (рекомендуемое) Графическая часть	66
Приложение В (рекомендуемое) Набор атрибутов таблиц базы данных и их характеристики	72
Приложение Г (рекомендуемое) Листинг серверной части приложения	76
Приложение Д (рекомендуемое) Листинг клиентской части приложения	82
Приложение Е (рекомендуемое) Листинг автоматизированных тестов	88
Приложение Ж (рекомендуемое) Результаты тестирования	92

Введение

Современная система образования предъявляет высокие требования к уровню подготовки школьников, особенно в период сдачи ОГЭ и ЕГЭ. Эти экзамены являются ключевыми моментами в образовательной карьере учеников, так как их результаты напрямую влияют на дальнейшее обучение в высших учебных заведениях и выбор будущей профессии. В связи с этим, организация качественной и эффективной подготовки к ОГЭ и ЕГЭ приобретает особую актуальность.

Традиционные методы подготовки, такие как дополнительные занятия в школах и частные репетиторы, не всегда способны удовлетворить все потребности учеников. Современные технологии позволяют разработать программные средства для онлайн-школы, которые могут существенно повысить качество подготовки за счет индивидуального подхода и гибкости в обучении. Создание таких программных средств может стать значительным шагом в развитии системы образования и предоставлении школьникам более широких возможностей для успешного прохождения экзаменационных испытаний.

Объект исследования – средства автоматизации управления и организации учебными курсами.

Предмет исследования – программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ.

Целью выпускной квалификационной работы является разработка приложения для онлайн-школы, позволяющее управлять и организовывать учебные курсы по подготовке школьников к ОГЭ и ЕГЭ.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести сравнительный анализ продуктов-аналогов;
- выявить и проанализировать требования к разрабатываемому приложению;

- спроектировать программные средства и структуры данных;
- провести планирование и расчет бюджета проекта;
- осуществить разработку программного продукта;
- провести тестирования разработанного приложения;
- оформить пояснительную записку в соответствии с требованиями.

Разработка программных средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ повысит эффективность учебного процесса, предоставив ученикам и преподавателям удобные инструменты для достижения высоких результатов обучения.

1 Постановка задачи и анализ требований для программных средств, реализующих функционал онлайн-школы

1.1 Анализ предметной области и постановка задачи

ОГЭ и ЕГЭ являются стандартизированными тестами, которые проверяют знания учеников по различным предметам. Они включают в себя вопросы различных уровней сложности и требуют от учеников умения анализировать информацию, применять знания и решать задачи. Вопросы бывают нескольких типов:

- вопрос с выбором ответа/ответов из предложенных вариантов;
- вопрос с кратким ответом;
- вопрос, требующий развернутый ответ с подробным ходом решения.

Подготовка к этим экзаменам требует систематического подхода и регулярной практики.

Информационная система для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ является специализированной платформой (онлайн-школа), которая обеспечивает централизованное управление образовательным процессом. Основная цель данной системы – создание эффективной среды для подготовки учащихся к государственным экзаменам, включающей в себя управление курсами, взаимодействие между учениками и преподавателями, а также мониторинг успеваемости.

Ученик будет иметь возможность записаться на интересующий его курс, просмотреть учебный план и расписание этого курса, выполнять самостоятельные работы, которые выкладывает преподаватель, отслеживать свой прогресс в личном кабинете, в котором будет отображаться основная информация: количество выполненных самостоятельных работ, количество пройденных занятий, динамика набранных баллов по каждому заданию на пробных экзаменах.

Преподаватель будет иметь возможность создавать учебные курсы, а также контрольные и домашние работы. При создании курса преподаватель будет

составлять учебный план и расписание занятий. При создании самостоятельных работ преподаватель будет использовать специальный конструктор, в котором предусмотрена возможность добавления как существующих, так и собственных задач. После того, как ученик выполнит контрольную или домашнюю работу, преподаватель сможет ее проверить. Если в этой работе были задания с развернутым ответом, то преподавателю будет необходимо сделать подробный разбор решения, поставить оценку и при необходимости написать комментарий.

В процессе анализа были выделены следующие ключевые сущности, которые понадобятся при реализации приложения:

- учебные курсы: модули и материалы для подготовки по различным предметам;
- ученики: школьники, проходящие подготовку к ОГЭ и ЕГЭ;
- преподаватели: педагоги, ведущие курсы и оценивающие знания учеников;
- администраторы: лица, отвечающие за организацию и управление процессом обучения;
- тесты и задания: контрольные работы, тренировочные материалы и пробные варианты экзаменов для оценки знаний учеников.

Целями разработки приложения для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ являются:

- автоматизация процессов обучения: обеспечение удобного и эффективного доступа к учебным материалам и тестам;
- улучшение взаимодействия между участниками: обеспечение канала коммуникации между учениками, преподавателями и администрацией;
- мониторинг и анализ успеваемости: введение инструментов для отслеживания прогресса учащихся и оценки эффективности учебных программ;
- повышение качества подготовки: обеспечение качественного образовательного контента и дополнительных (факультативных) программ подготовки.

Ключевые категории пользователей и приложения (подсистемы) информационной системы представлены в таблице 1.

Таблица 1 – Категории пользователей

Категория пользователей	Наименование приложения (подсистемы)	Автоматизируемые бизнес-процессы	Классы информационных объектов
Ученик	Платформа онлайн-курсов	Доступ к учебным материалам, выполнение тестов, получение статистики и прогресса обучения, участие в обсуждениях	Курсы, Тесты, Задания, Результаты тестов, Статистика, Комментарии
Преподаватель	Система управления курсами	Создание и обновление курсов, оценка тестов, обратная связь с учениками	Курсы, Тесты, Задания, Результаты тестов, Обратная связь

У категорий пользователей будут различаться права, для этого необходимо предусмотреть систему авторизации с соответствующими ролями.

1.2 Обзор программных средств аналогичного назначения

1.2.1 Онлайн платформа Maximum Education

Maximum Education – онлайн платформа для проведения уроков, контроля выполнения домашних заданий и аналитики прогресса учеников. Данная образовательная платформа предлагает широкий спектр материалов для подготовки к экзаменам, включая видеоуроки, тестовые задания и рекомендации. Обучение проходит в онлайн и гибридном форматах и состоит из трех основных блоков: онлайн-модуль, домашнее задание, прогресс ученика [1]. Интерфейс приложения приведен на рисунке 1.

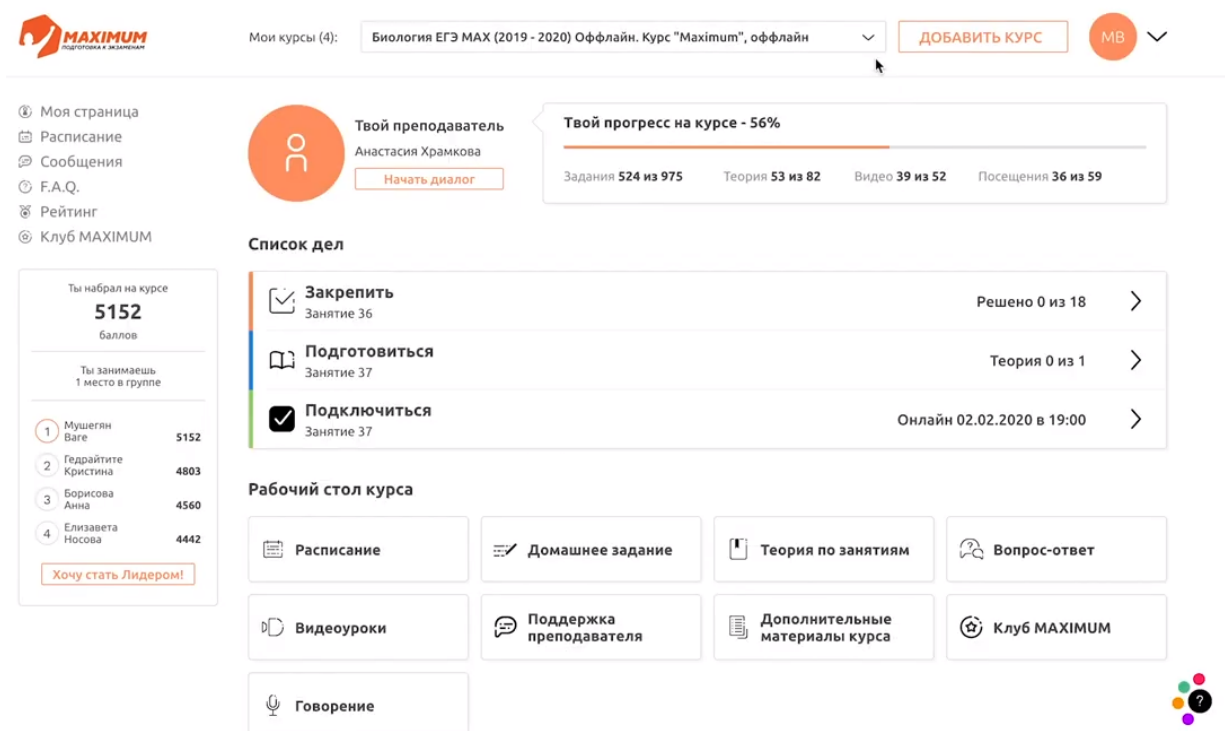


Рисунок 1 – Интерфейс онлайн платформы Maximum Education

К достоинствам можно отнести возможность заниматься с любого устройства; занятия проходят в группах до четырех человек, что позволяет преподавателю уделить внимание каждому ученику; наличие удобного личного кабинета; наличие гарантий возврата средств.

К недостаткам можно отнести невозможность преподавателю создавать собственные курсы; отсутствие загрузки авторских задач.

1.2.2 Онлайн-школа Фоксфорд

Фоксфорд – онлайн-школа для учеников 1 – 11 классов, учителей и родителей. На онлайн-курсах и индивидуальных занятиях с репетитором школьники готовятся к ЕГЭ, ОГЭ, олимпиадам, изучают школьные предметы. Занятия ведут преподаватели МГУ, МФТИ, ВШЭ и других ведущих вузов страны [2]. Интерфейс личного кабинет ученика приведен на рисунке 2.

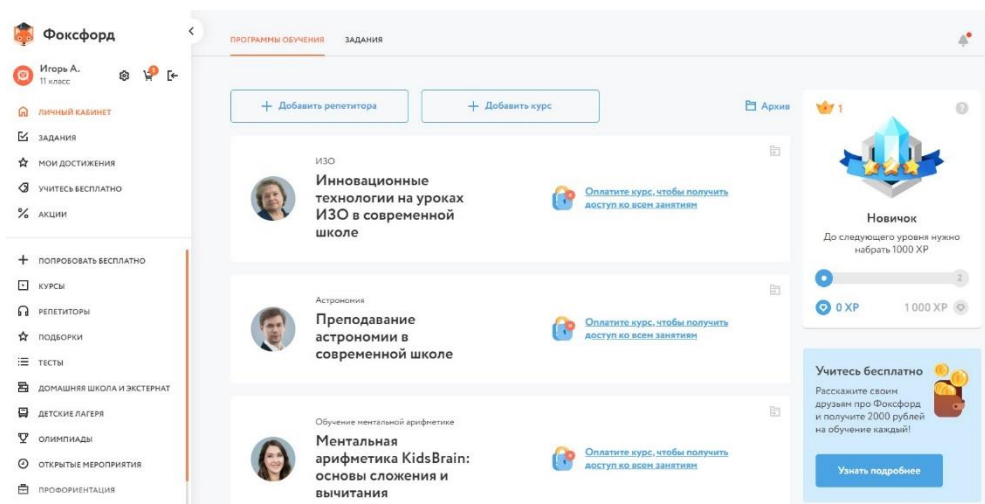


Рисунок 2 – Интерфейс личного кабинета онлайн-школы Фоксфорд

К достоинствам можно отнести удобный личный кабинет; возможность очных занятий в небольших группах; наличие программ обучения для учеников с ограниченными возможностями здоровья; наличие интерактивных домашних заданий; предусмотрены бесплатные занятия, а также есть отдельные программы для учеников, которые готовятся к олимпиадам.

К недостаткам можно отнести сложность пользовательского интерфейса; достаточно высокие цены на учебные курсы.

1.2.3 Образовательный портал Школково

Школково – образовательный портал, помогающий школьникам успешно занять первые места на олимпиадах, сдать ЕГЭ и ОГЭ с помощью уникальной методики проработки сложных задач. К особенностям данного портала относятся [3]:

- индивидуальная программа обучения (на основе текущего уровня знаний система автоматически подбирает ученику подходящий набор для ежедневной практической тренировки);
- возможность самостоятельно корректировать план обучения (пропускать легкие задачи, делать упор на нужных темах, компоновать собственные варианты заданий).

Интерфейс главной страницы приведен на рисунке 3.

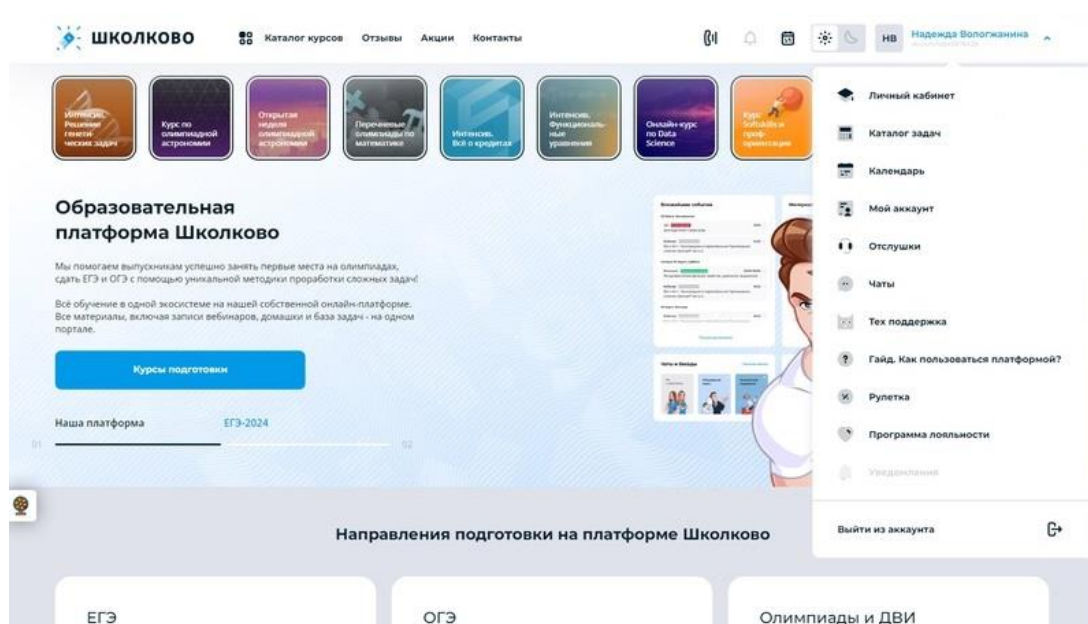


Рисунок 3 – Интерфейс главной страницы Школково

На рисунке 4 приведен интерфейс личного кабинета Школково.

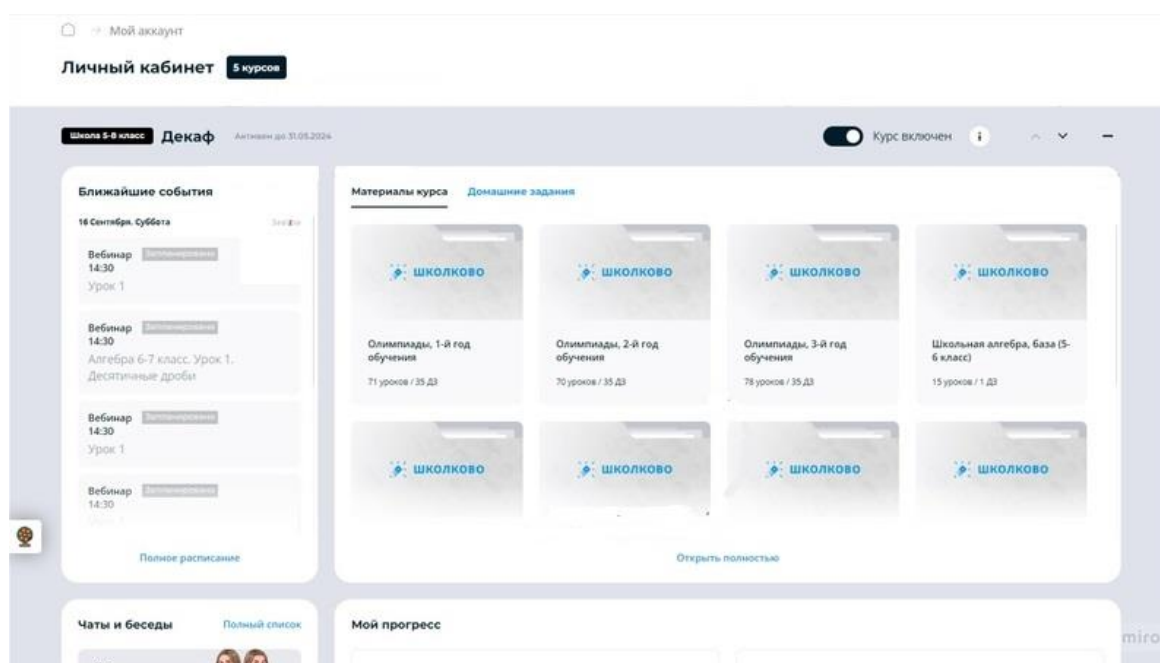


Рисунок 4 – Интерфейс личного кабинета Школково

К достоинствам можно отнести удобный личный кабинет; наличие разнообразных учебных курсов; возможность получить скидку в раз месяц, прокрутив рулетку с призами; наличие программы лояльности; наличие

специального предложения: возврат средств за курс при сдаче ЕГЭ на сто баллов.

К недостаткам можно отнести невозможность создания собственных самостоятельных заданий; отсутствие адаптивного интерфейса.

1.2.4 Онлайн-школа Вебиум

Вебиум – это онлайн-школа для подготовки к экзаменам. Основные направления обучения: подготовка к ОГЭ и ЕГЭ. Ядром платформы служит рекомендательная система, которая анализирует действия пользователей при обучении, сопоставляет модель предметной области и модель знаний обучаемого, и предлагает эффективные персонифицированные образовательные траектории [4]. Интерфейс приложения приведен на рисунке 5.

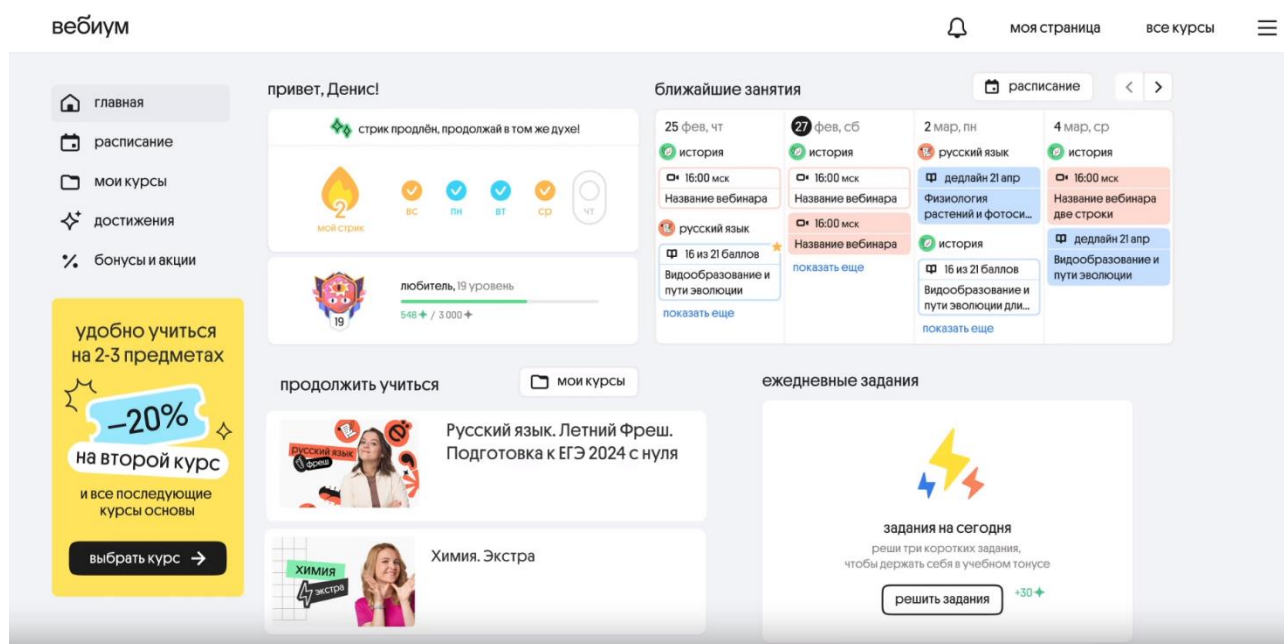


Рисунок 5 – Интерфейс онлайн-школы Вебиум

К достоинствам можно отнести удобный личный кабинет; автоматическое создание коротких версии уроков (вебинаров) для экономии времени; наличие личного наставника для каждого ученика.

К недостаткам можно отнести невозможность создания собственных самостоятельных заданий; наличие курсов только по подготовке к ЕГЭ.

1.2.5 Образовательный портал Сдам ГИА

Сдам ГИА – образовательный портал для подготовки к экзаменам. На сайте представлены тренировочные варианты тестов по всем предметам, которые сдаются в рамках Государственной итоговой аттестации (ГИА) [5]. Интерфейс данного портала представлен на рисунке 6.

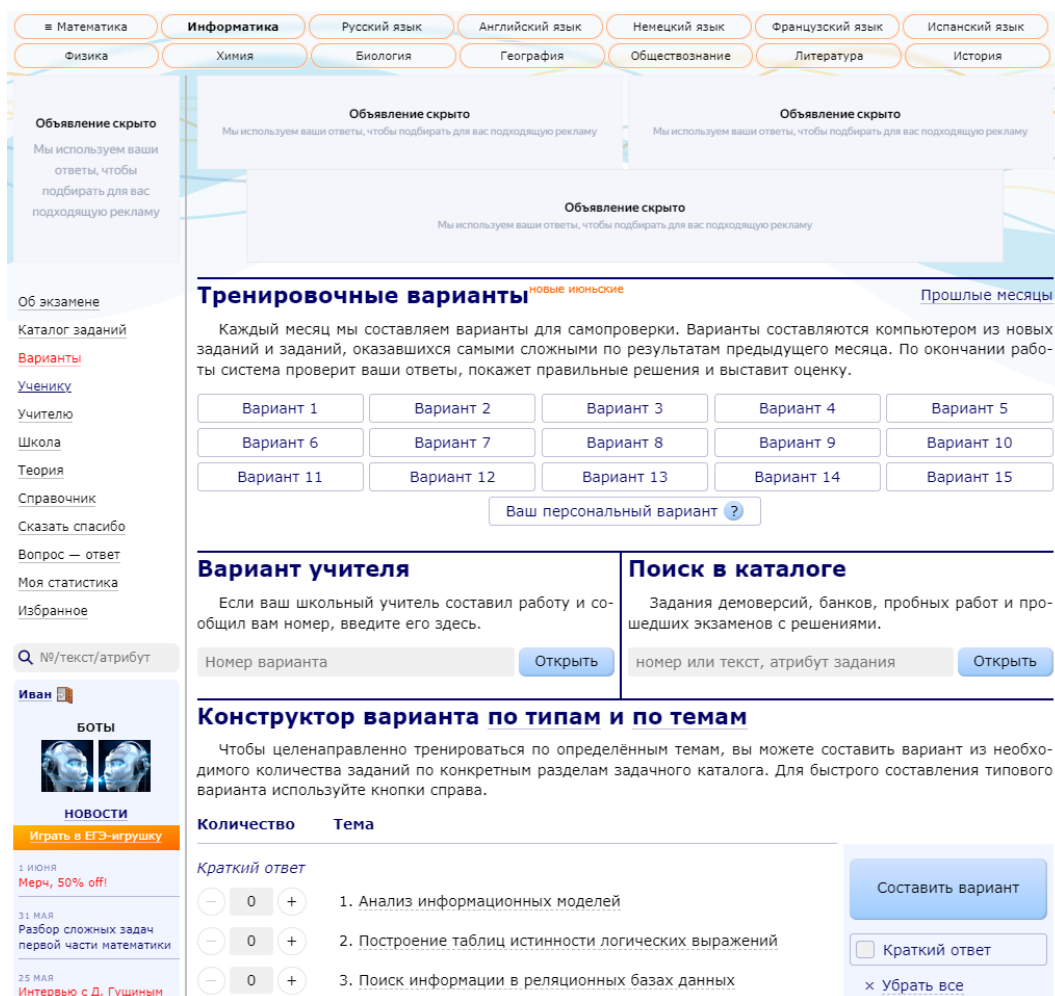


Рисунок 6 – Интерфейс онлайн-портала Сдам ГИА

К достоинствам можно отнести возможность создания учебных групп и курсов; создания собственных самостоятельных работ с загрузкой авторских заданий; большой банк готовых заданий; ежемесячный выход списка пробных вариантов, которые максимально приближены к реальным вариантам на экзаменах.

К недостаткам можно отнести интуитивно непонятный интерфейс, отсутствие удобного личного кабинета.

1.2.6 Сравнение продуктов-аналогов

Проведем сравнительный анализ рассмотренных выше программных средств.

При оценивании будет использоваться трехбалльная шкала оценки, в которой 0 означает «полное отсутствие реализации», 1 – «частичную реализацию», а 2 – «полную реализацию» рассматриваемой функции. Результаты сравнения приведены в таблице 1.

Таблица 1 – Сравнение программных средств аналогичного назначения

Критериальная оценка	Maximum Education	Фоксфорд	Школково	Вебиум	Сдам ГИА
Наличие личного кабинета	1	1	1	1	0
Возможность добавления авторских задач	0	0	0	0	2
Возможность создать учебный курс	0	0	0	1	1
Наличие дополнительных курсов по подготовке к олимпиадам	0	1	2	0	0
Наличие программы лояльности	0	1	1	0	0
Интуитивно понятный интерфейс	2	1	2	1	1
Автоматическое ежемесячное генерирование пробных экзаменов	0	0	1	0	1
ИТОГО	3	4	7	3	5

Подводя итог сравнения можно сделать вывод, что ни одно из рассмотренных приложений не соответствует объявленным критериям в полной мере. Среди всех онлайн-школ большее количество баллов набрала Школково,

однако ее программные средства не обеспечивают возможность создавать учебные курсы и добавлять авторские задачи.

В результате принято решение разработать собственное программное средство, которое будет включать в себя лучшие решения от рассмотренных приложений и будет лишено наиболее существенных недостатков.

1.3 Анализ функциональных требований

Согласно поставленным задачам, были выделены следующие функциональные требования:

- регистрация и авторизация;
- получение списка учебных курсов;
- фильтрация курсов по нескольким критериям;
- приобретение учебного курса;
- получение расписания занятий и списка домашних заданий;
- выполнение самостоятельных работ;
- получение текущего прогресса по учебному курсу;
- создание учебного курса;
- создание самостоятельных работ;
- добавление задач для самостоятельной работы;
- проверка самостоятельных работ;
- выставление баллов за решенную задачу.

В соответствии с поставленной задачей и выявленными функциональными требованиями была разработана диаграмма вариантов использования [6], которая представлена на рисунке Б.1 в приложении Б.

Спецификация прецедента «Создать самостоятельную работу» представлена на рисунке 7.

Прецедент: Создать самостоятельную работу
ID: 1
Краткое описание: Преподаватель создает самостоятельную работу для учебного курса
Главные актеры: Преподаватель
Второстепенные актеры: Нет
Предусловие: Преподаватель должен быть авторизован, а также он должен выбрать учебный курс который он введет
Основной поток: <ol style="list-style-type: none"> 1. преподаватель выбирает учебный курс; 2. преподаватель дает название самостоятельной работе и указывает вид работы (контрольная работа или домашняя работа); 3. преподаватель добавляет задания; 4. преподаватель указывает сроки выполнения; 5. преподаватель публикует работу.
Постусловие: Для выбранного учебного курса создана новая самостоятельная работа

Рисунок 7 – Спецификация прецедента «Создать самостоятельную работу»

Спецификация прецедента «Проверить самостоятельную работу» представлена на рисунке 8.

Прецедент: Проверить самостоятельную работу
ID: 2
Краткое описание: Преподаватель проверяет работу, выполненную учеником
Главные актеры: Преподаватель
Второстепенные актеры: Нет
Предусловие: Ученик должен выполнить самостоятельную работу и опубликовать решение
Основной поток: <ol style="list-style-type: none"> 1. преподаватель выбирает выполненную самостоятельную работу; 2. преподаватель оставляет комментарии к заданиям с развернутым ответом 3. преподаватель завершает проверку и размещает результат в системе
Постусловие: В системе появляется проверенная самостоятельная работа; ученик получает уведомление, что его работа проверена

Рисунок 8 – Спецификация прецедента «Проверить самостоятельную работу»

Спецификация прецедента «Создать курс» представлена на рисунке 9.

Прецедент: Создать курс
ID: 3
Краткое описание: Преподаватель создает новый учебный курс
Главные актеры: Преподаватель
Второстепенные актеры: Нет
Предусловие: Преподаватель должен быть авторизован и курса с таким названием не должно существовать
Основной поток: <ol style="list-style-type: none"> 1. преподаватель должен выбрать предмет; 2. преподаватель должен выбрать тип экзамена (ОГЭ или ЕГЭ); 3. преподаватель должен указать уникальное название учебного курса; 4. преподаватель должен составить расписание занятий; 5. преподаватель должен составить учебный план
Постусловие: Создан новый учебный курс

Рисунок 9 – Спецификация прецедента «Создать курс»

Спецификация прецедента «Выполнить самостоятельную работу» представлена на рисунке 10.

Прецедент: Выполнить самостоятельную работу
ID: 3
Краткое описание: Ученик выполняет самостоятельную работу, выложенную преподавателем
Главные актеры: Ученик
Второстепенные актеры: Нет
Предусловие: Ученик должен быть авторизован в системе, а также он должен быть участником учебного курса, самостоятельную работу, которого он собирается выполнить
Основной поток: <ol style="list-style-type: none"> 1. ученик выбирает учебный курс; 2. ученик выбирает самостоятельную работу, выложенную преподавателем; 3. ученик отвечает на вопросы с кратким ответом; 4. ученик прикрепляет документ с решением с заданию с развернутым ответом; 5. ученик завершает работу и отправляет ее на проверку
Постусловие: Ученик выполнил самостоятельную работу и выложил решеную работу на проверку

Рисунок 10 – Спецификация прецедента «Выполнить самостоятельную работу»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить

последовательность действий пользователя и системы при выполнении определенных функций.

1.4 Анализ нефункциональных требований

Анализ нефункциональных требований для разрабатываемого приложения включает в себя рассмотрение нескольких аспектов.

Производительность. Использование современных технологий, таких как TypeScript и фреймворков NextJS и NestJS, позволяет оптимизировать производительность приложения. Применение ORM TypeORM для работы с базой данных Microsoft SQL Server обеспечивает эффективный доступ к данным и уменьшает вероятность ошибок, связанных с прямым взаимодействием с базой данных.

Безопасность. Фреймворк NestJS предоставляет встроенные механизмы безопасности, такие как JWT (JSON Web Tokens) для аутентификации и авторизации пользователей. При работе с базой данных необходимо учитывать вопросы безопасности, такие как шифрование паролей, использование подготовленных выражений для предотвращения SQL-инъекций и другие меры защиты от внешних угроз [7].

Масштабируемость. Выбор технологии TypeScript и фреймворков NextJS и NestJS позволяет легко масштабировать приложение в соответствии с ростом нагрузки. Использование Microsoft SQL Server в качестве СУБД также способствует масштабируемости, так как этот продукт поддерживает горизонтальное и вертикальное масштабирование [8].

Интерфейс пользователя. Клиентская часть приложения будет разработана с использованием React-подобного интерфейса, предоставляемого NextJS, что обеспечит удобство и интуитивно понятный интерфейс для пользователей [9].

Все эти аспекты будут учтены при разработке и реализации проекта для обеспечения его успешной работы и удовлетворения потребностей пользователей.

2 Проектирование программных средств для онлайн-школы

2.1 Архитектура приложения

При разработке веб-приложения будет использоваться трехуровневая архитектура. Схема архитектуры представлена на рисунке 11.

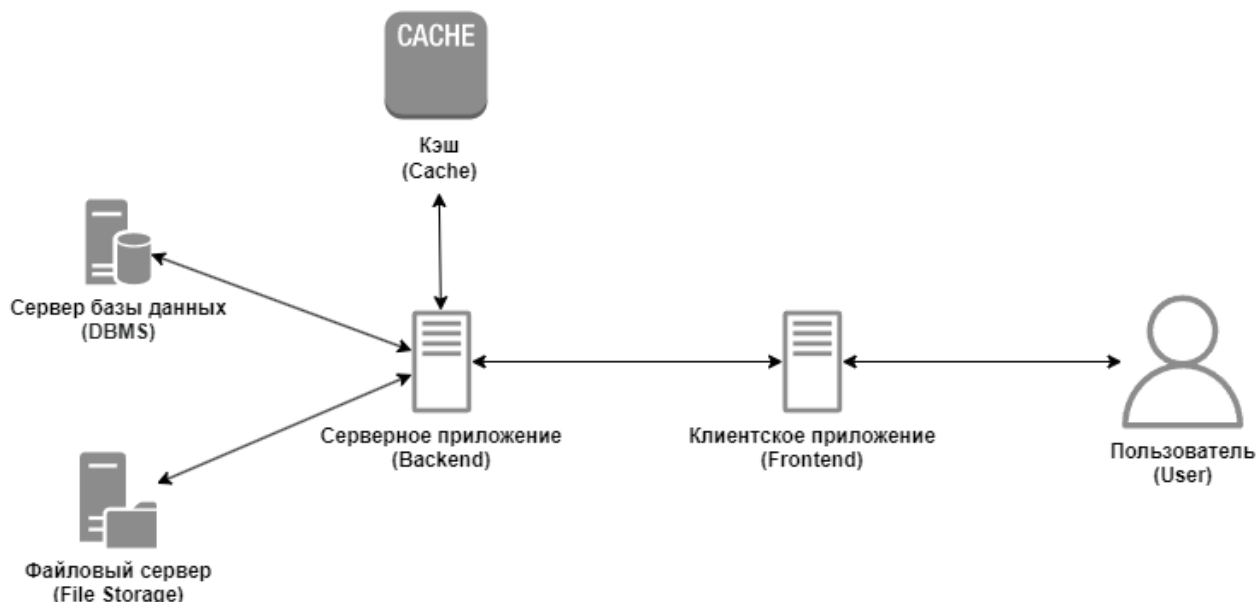


Рисунок 11 – Клиент-серверная архитектура

Трехуровневая архитектура – архитектурная модель программного комплекса, предполагающая наличие в нем трех слоев: слой представления, с которым работают пользователи; слой бизнес-логики, с которым работает клиентское приложение и слой доступа к данным, обращение к которому происходит из бизнес-логики.

Взаимодействие между слоем представления и слоем бизнес-логики осуществляется с помощью REST API по протоколу HTTP. Слой бизнес-логики взаимодействует с базой данных посредством выполнения SQL-запросов.

Различные файлы, документы и фотографии будут храниться на файловом сервере. Для повышения производительности будет использоваться механизм кеширования.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель базы данных представляет собой абстрактное представление о структуре данных и их взаимосвязях, она разрабатывается независимо от ограничений, вытекающих из модели данных, поддерживаемых той или иной СУБД [10].

На основе анализа предметной области и результатов моделирования бизнес-процессов была составлена концептуальная модель базы данных в виде диаграммы «сущность-связь», представленной на рисунке 12.

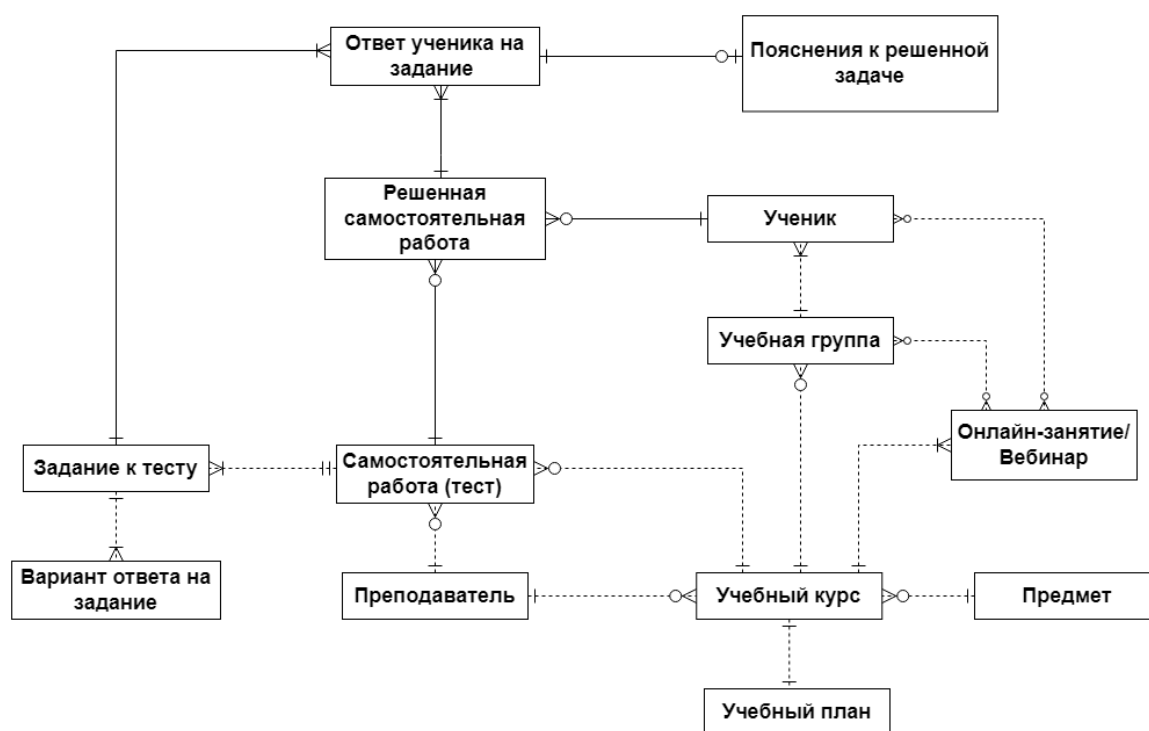


Рисунок 12 – Концептуальная модель базы данных

На основе данной диаграммы была составлена таблица сущностей, приведенная в таблице 2.

Таблица 2 – Сущности предметной области

Название сущности	Идентификатор сущности	Описание
Предмет	Subject	Содержит информацию об учебных предметах
Учебный курс	Course	Содержит сведения о преподаваемых курсах

Продолжение таблицы 2

Онлайн-занятие	Lesson	Содержит сведения об онлайн-занятии
Учебная группа	EdGroup	Содержит сведения об учебной группе
Ученик	Student	Содержит сведения об ученике
Преподаватель	Teacher	Содержит сведения о преподавателе
Самостоятельная работа	Test	Содержит сведения о самостоятельных работах
Задание к работе	TaskOfTest	Содержит вопросы для самостоятельных работ
Варианты ответа на задание	VariantAnswerForTask	Содержит ответы для вопросов в самостоятельной работе
Решенная самостоятельная работа	CompletedTest	Содержит сведения о решенных самостоятельных работах
Ответ ученика на вопрос	AnswerOfStudent	Содержит ответы ученика на вопросы в самостоятельной работе
Пояснения	Explanation	Содержит комментарий преподавателя к решенной работе

Полученные результаты являются исходными данными для создания логической и физической схем базы данных.

2.2.2 Логическая модель базы данных

Логическая модель – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи. Логическая модель представляет данные таким образом, чтобы они легко воспринимались бизнес-пользователями. Проектирование логической модели должно быть свободно от требований платформы и языка реализации или способа дальнейшего использования данных [11].

На основании концептуальной модели была сконструирована логическая модель базы данных, которая изображена на рисунке Б.2 в приложении Б.

В таблице 3 представлено описание связей между сущностями.

Таблица 3 – Описание связей

Сущность 1	Сущность 2	Описание	Мощность	Тип связи
Учебный курс	Предмет	Предмет может изучаться на нескольких учебных курсах	М:1	Не идентифицирующая
Учебный курс	Преподаватель	Преподаватель может вести несколько учебных курсов	М:1	Не идентифицирующая
Самостоятельная работа	Учебный курс	На учебном курсе может быть несколько самостоятельных работ	М:1	Не идентифицирующая
Онлайн-занятие	Учебный курс	На учебном курсе содержится несколько уроков	М:1	Не идентифицирующая
Учебная группа	Учебный курс	Учебный курс могут изучать несколько учебных групп	М:1	Не идентифицирующая
Ученик	Учебная группа	В учебной группе учатся несколько учеников	М:1	Не идентифицирующая
Самостоятельная работа	Преподаватель	Преподаватель может создавать несколько самостоятельных работ	М:1	Не идентифицирующая
Задание к тесту	Самостоятельная работа	Самостоятельная работа состоит из нескольких заданий	М:1	Не идентифицирующая
Учебная группа	Онлайн-занятие	У учебных групп есть несколько запланированных онлайн-занятий	М:N	Не идентифицирующая
Ученик	Онлайн-занятие	Ученики посещают несколько онлайн-занятий	М:N	Не идентифицирующая
Вариант ответа на задание	Задание к тесту	У задания имеется несколько вариантов ответов	М:1	Не идентифицирующая
Решенная самостоятельная работа	Самостоятельная работа	Самостоятельную работу решают несколько учеников	М:1	Идентифицирующая

Продолжение таблицы 3

Решенная самостоятельная работа	Ученик	Ученик выполняет несколько самостоятельных работ	М:1	Идентифицирующая
Ответ ученика на вопрос	Решенная самостоятельная работа	Ученик оставляет несколько ответов в самостоятельной работе	М:1	Идентифицирующая
Задание к тесту	Ответ ученика на вопрос	Ученики отвечают на задание теста	М:1	Идентифицирующая
Пояснения	Решенная самостоятельная работа	Преподаватель может оставить пояснения к решенной самостоятельной работе	1:1	Идентифицирующая

На основе спроектированной логической схемы базы данных будет осуществлено ее развертывание в СУБД – Microsoft SQL Server.

2.3 Проектирование программных средств

На основе логической модели базы данных и анализа предметной области была спроектирована диаграмма классов, представленная на рисунке Б.3 в приложении Б. На ней отображены основные классы, их поля, методы и связи между ними. А также около каждой связи присутствует надпись, поясняющая характер этой связи.

Для сценария выполнения самостоятельной работы была разработана диаграмма деятельности, изображенная на рисунке 13. На данной диаграмме присутствуют две дорожки, каждая из которых представляет отдельного участника процесса, выполняющего определенные действия. Дорожка «Ученик» представляет пользователя, вошедшего в систему с ролью ученика. Дорожка «Система» представляет серверную часть приложения, которая отвечает за логику бизнес-процессов.

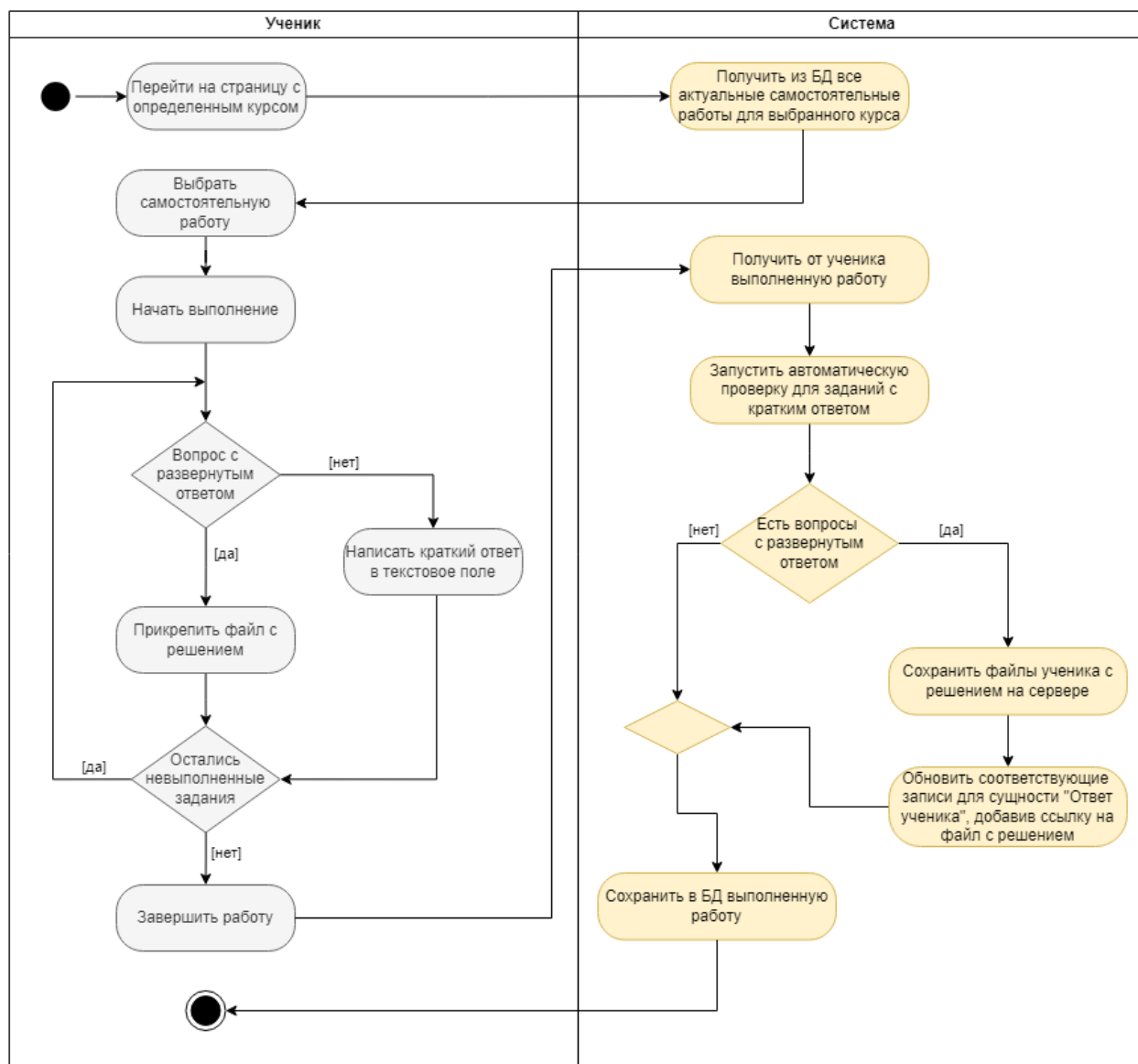


Рисунок 13 – Диаграмма деятельности

Чтобы выполнить самостоятельную работу ученик должен сначала выбрать учебный курс, а потом саму работу. На все задания с кратким ответом ответы записываются в текстовое поле. Для ответа на задания с развернутым ответом ученик должен будет загружать файл с решением, который будет сохраняться на сервере.

После завершения самостоятельной работы учеником, все задания с кратким ответом будут автоматически проверены, и результаты проверки будут сразу доступны.

Для сценария проверки самостоятельной работы была разработана диаграмма последовательности, изображенная на рисунке 14. На данной диаграмме присутствуют четыре основных участника процесса: учитель, web-приложение, серверная часть и база данных.

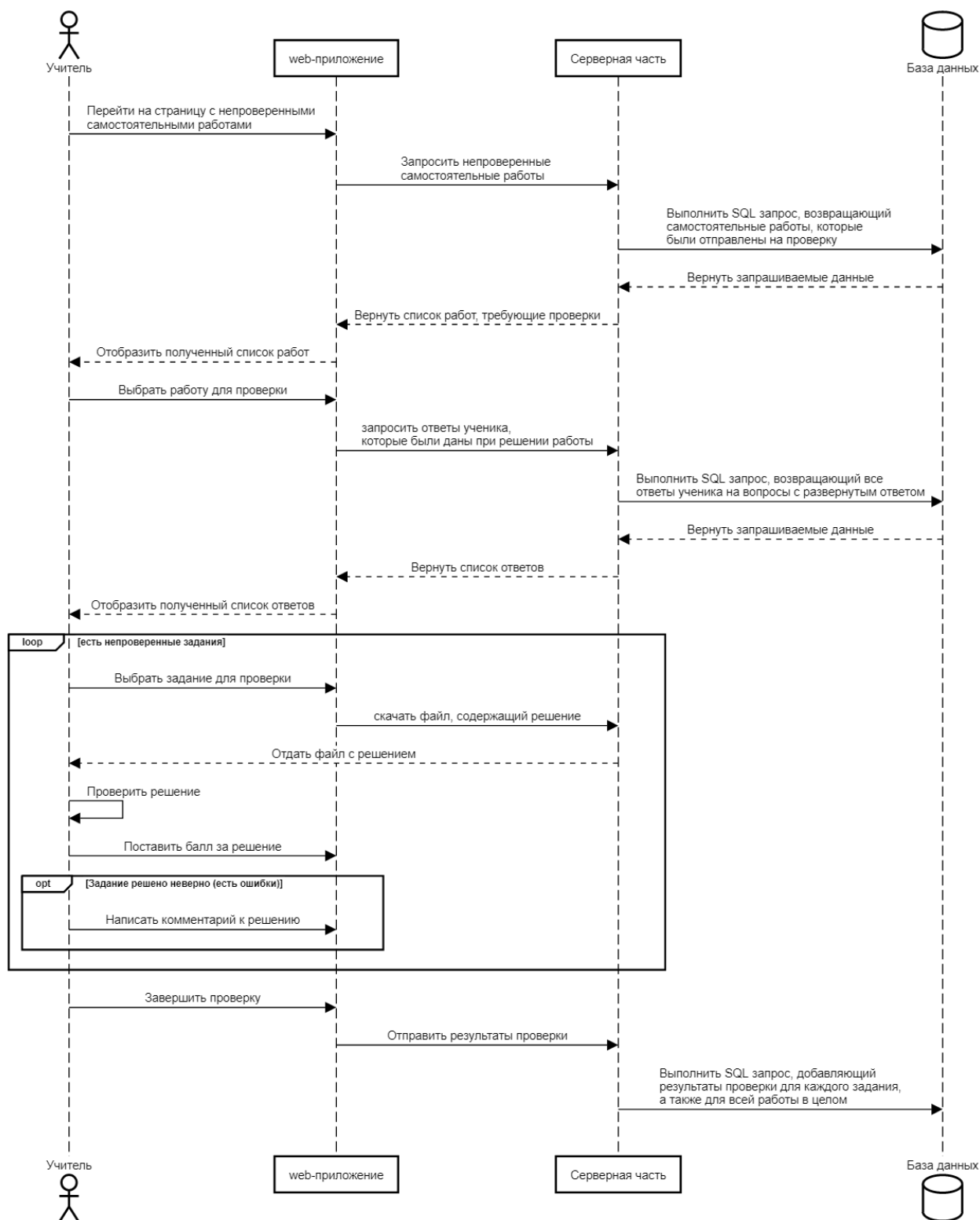


Рисунок 14 – Диаграмма последовательности

Проверке подлежат только самостоятельные работы, в которых есть хотя бы одно задание с развернутым ответом. Для каждого такого задания ученики выкладывают файлы с решением, а учителя скачивают эти файлы для проверки и выставления баллов.

2.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс разрабатываемого приложения должен обеспечивать привычное и удобное представление информации, простое и эффективное выполнение основных функций пользователей.

При запуске приложения пользователю будет предложено выбрать роль (рисунок 15).

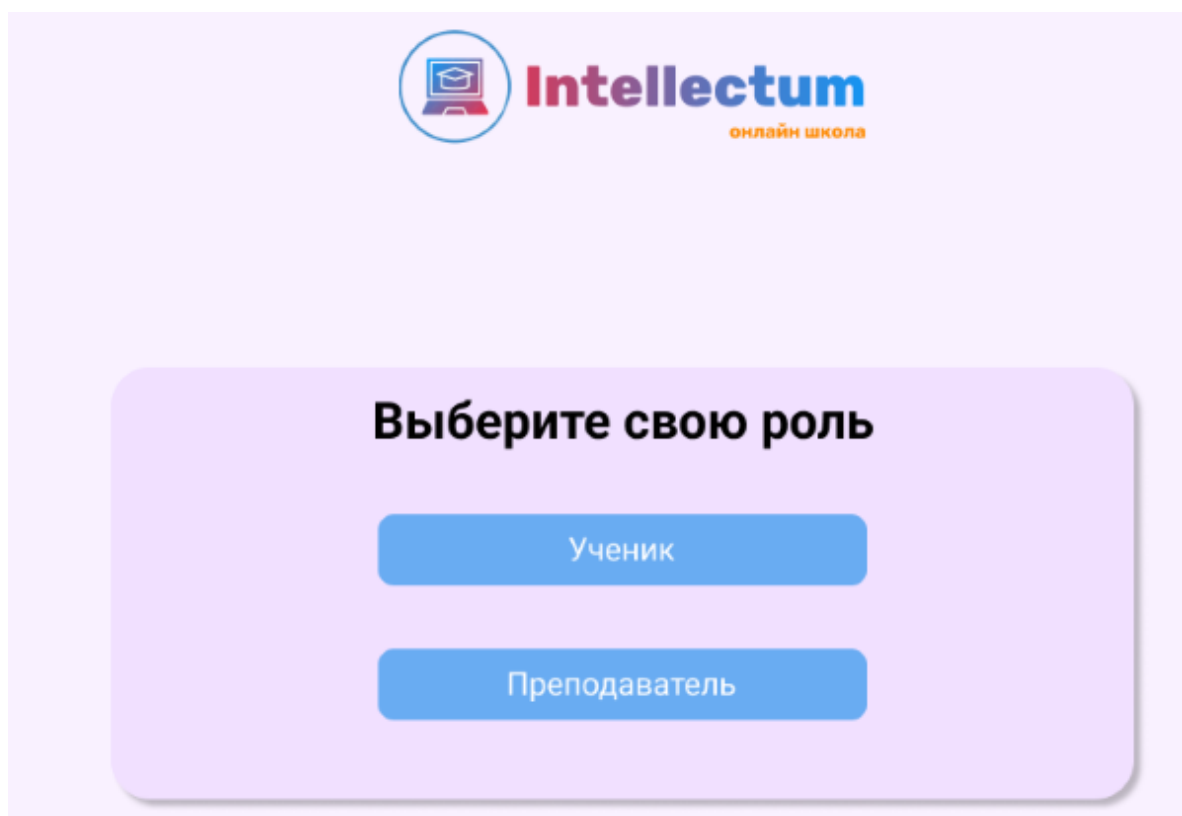
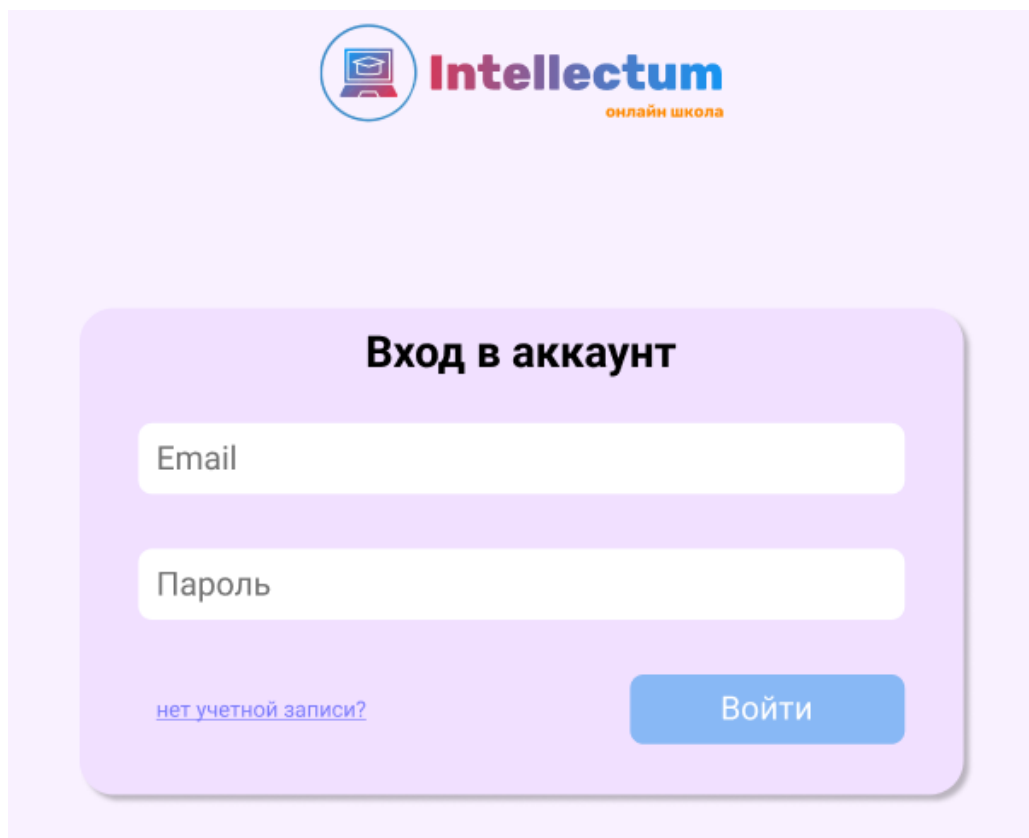


Рисунок 15 – Форма выбора роли

После выбора роли, пользователя переадресует на страницу входа, внешний вид которой представлен на рисунке 16.

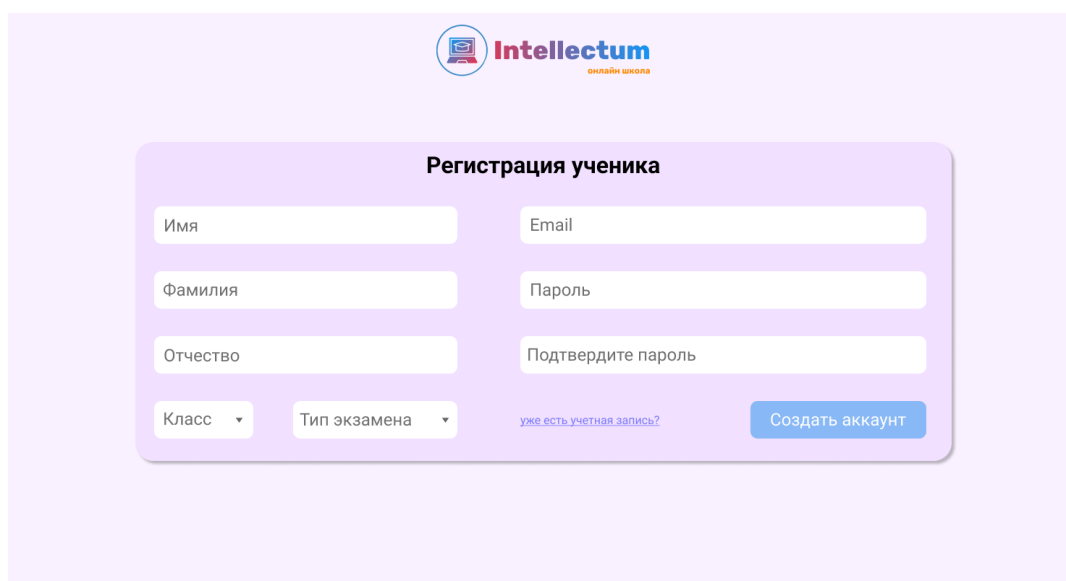


The screenshot shows the login interface for Intellectum, an online school. At the top, there is a logo consisting of a blue circle with a white envelope icon inside, followed by the word "Intellectum" in blue and "онлайн школа" in orange below it. The main heading is "Вход в аккаунт" (Login to account) in bold black text. Below this, there are two white input fields: "Email" and "Пароль" (Password). Under the password field, there is a blue button labeled "Войти" (Login). To the left of the button, there is a blue link that says "нет учетной записи?" (no account?).

Рисунок 16 – Форма входа в систему

Если у пользователя нет учетной записи, то он может нажать на ссылку «нет учетной записи?», тогда его переадресует на страницу регистрации.

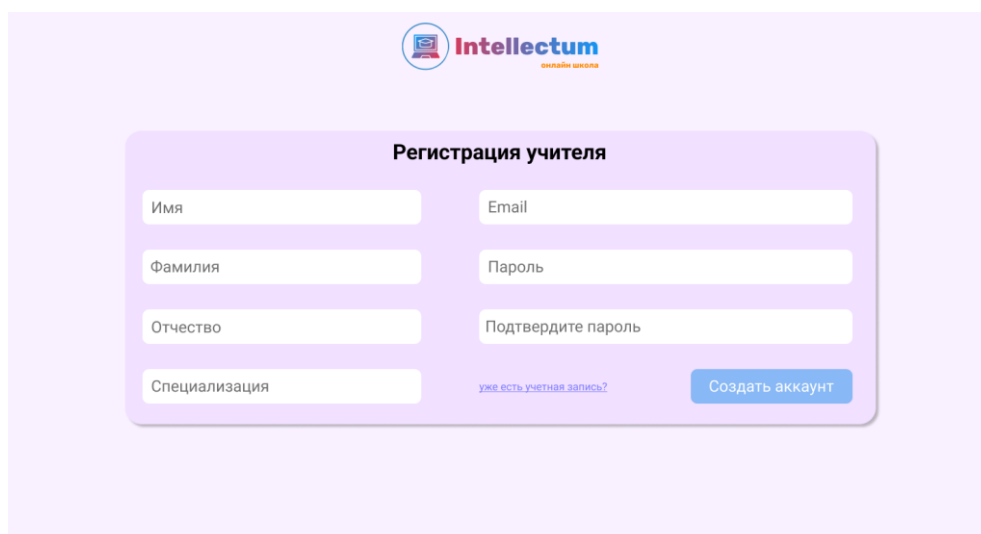
Страница регистрации для ученика будет выглядеть так, как представлено на рисунке 17.



The screenshot shows the student registration interface for Intellectum. At the top, there is the same logo as in the previous image. The main heading is "Регистрация ученика" (Student registration) in bold black text. Below this, there are several white input fields arranged in two columns. The left column contains fields for "Имя" (Name), "Фамилия" (Surname), and "Отчество" (Patronymic). The right column contains fields for "Email", "Пароль" (Password), and "Подтвердите пароль" (Confirm password). At the bottom left, there are two dropdown menus labeled "Класс" (Class) and "Тип экзамена" (Exam type). To the right of these, there is a blue link that says "уже есть учетная записи?" (already have an account?). At the bottom right, there is a blue button labeled "Создать аккаунт" (Create account).

Рисунок 17 – Форма регистрации ученика

Страница регистрации для преподавателя будет выглядеть так, как представлено на рисунке 18.



The screenshot shows a registration form titled "Регистрация учителя" (Teacher Registration) on the Intellectum platform. The form is set against a light purple background. It contains several input fields: "Имя" (Name), "Email", "Фамилия" (Surname), "Пароль" (Password), "Отчество" (Patronymic), "Подтвердите пароль" (Confirm password), and "Специализация" (Specialization). There is a link "уже есть учетная запись?" (already have an account?) and a blue button "Создать аккаунт" (Create account).

Рисунок 18 – Форма регистрации учителя

После успешно входа в систему с ролью ученика, пользователя переадресует на главную страницу приложения (рисунок 19).

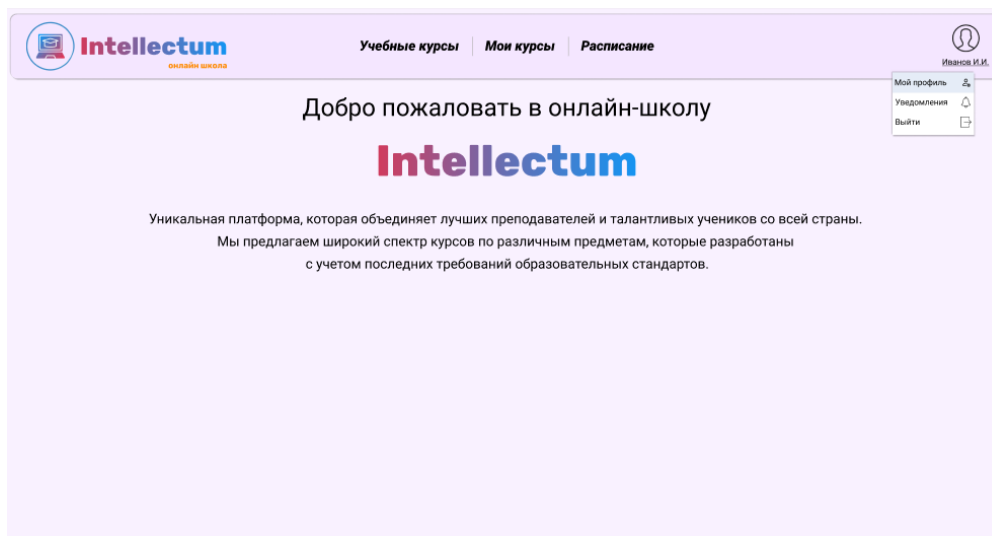


Рисунок 19 – Главная страница приложения

Все страницы приложения будут иметь общие элементы. Вверху будет располагаться логотип приложения, нажав на который произойдет переадресация на главную страницу. Расположенное по центру меню будет содержать три основных пункта: «Учебные курсы», «Мои Курсы» и «Расписание». В конце верхней части будет располагаться фото пользователя с

его фамилией и инициалами, нажав на которое откроется контекстное меню, из которого можно перейти на страницу своего профиля, на страницу с уведомлениями, либо можно будет выйти из учетной записи.

Для перехода на страницу своего профиля, ученику нужно выбрать пункт «Мой профиль» из выпадающего меню (рисунок 20).

The screenshot shows the 'Мой профиль' (My Profile) page in the Intellectum system. The page has a light purple header with the Intellectum logo and navigation links: 'Учебные курсы', 'Мои курсы', and 'Расписание'. The user's name 'Иванов И.И.' is displayed in the top right corner. The main content area is titled 'Профиль' and contains a profile card for 'Иван Иванович Иванов'. The card includes fields for 'Имя' (Name), 'Фамилия' (Surname), 'Отчество' (Patronymic), 'Email', 'Класс' (Class), and 'Тип образования' (Type of education). The 'Класс' is set to '9' and 'Тип образования' is set to 'ОГЭ'. There is a 'Загрузить фото' (Upload photo) button and a 'Сохранить изменения' (Save changes) button at the bottom.

Рисунок 20 – Профиль ученика

Для просмотра уведомлений ученик должен выбрать пункт «Уведомления» из выпадающего меню (рисунок 21).

The screenshot shows the 'Уведомления' (Notifications) page in the Intellectum system. The page has a light purple header with the Intellectum logo and navigation links: 'Учебные курсы', 'Мои курсы', and 'Расписание'. The user's name 'Иванов И.И.' is displayed in the top right corner. The main content area is titled 'Уведомления' and contains a notification card. The card has a blue header with 'ДЗ Проверено' and a timestamp '17.10.23'. The notification text includes 'Курс: Полноценный курс по подготовке к ОГЭ по русскому языку (8 мес.)' and 'Работа: ДЗ. Приставки -ПРИ, -ПРЕ'. There is a 'Посмотреть результат' (View result) button at the bottom right of the notification card.

Рисунок 21 – Уведомления

Чаще всего в уведомлениях будет информация о проверке самостоятельных работ. Нажав на кнопку «Посмотреть результаты» ученик сможет ознакомиться с результатами за самостоятельную работу.

Для просмотра список всех учебных курсов необходимо выбрать пункт меню «Учебные курсы» (рисунок 22). При отображения учебного курса

указываются основные параметры: название, количество занятий и самостоятельных работ, а также стоимость. Курсы можно фильтровать. По курсу можно получить подробную информацию, нажав на соответствующую кнопку.

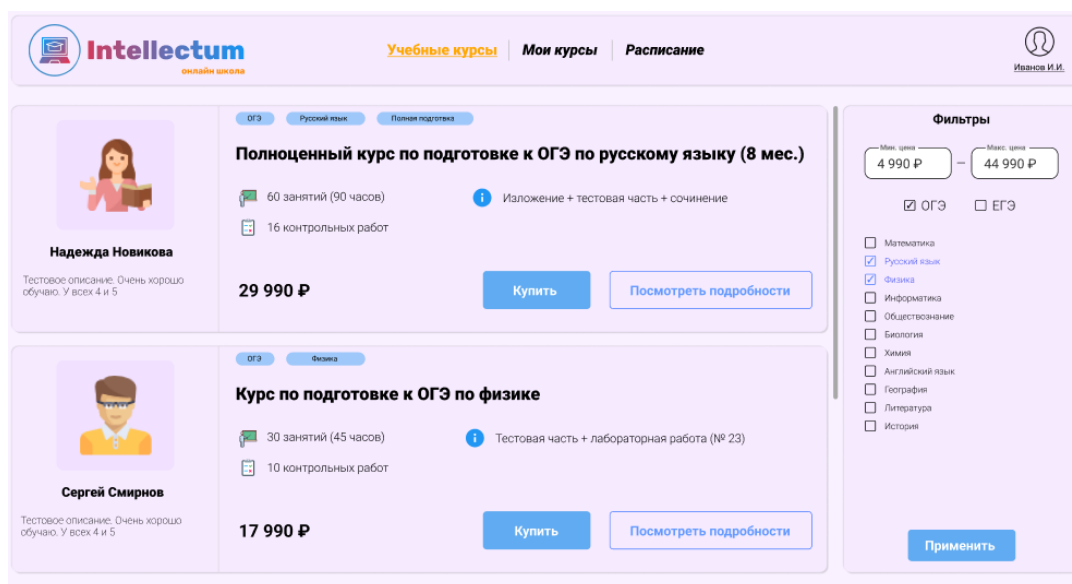


Рисунок 22 – Список учебных курсов

Для просмотра приобретенных курсов необходимо выбрать пункт меню «Мои курсы» (рисунок 23). Для каждого того курса указывается краткая информация: количество проведенных занятий, количество заданных домашних заданий, сведения о ближайшем занятии, а также список актуальных самостоятельных работ с указанием крайнего срока сдачи.

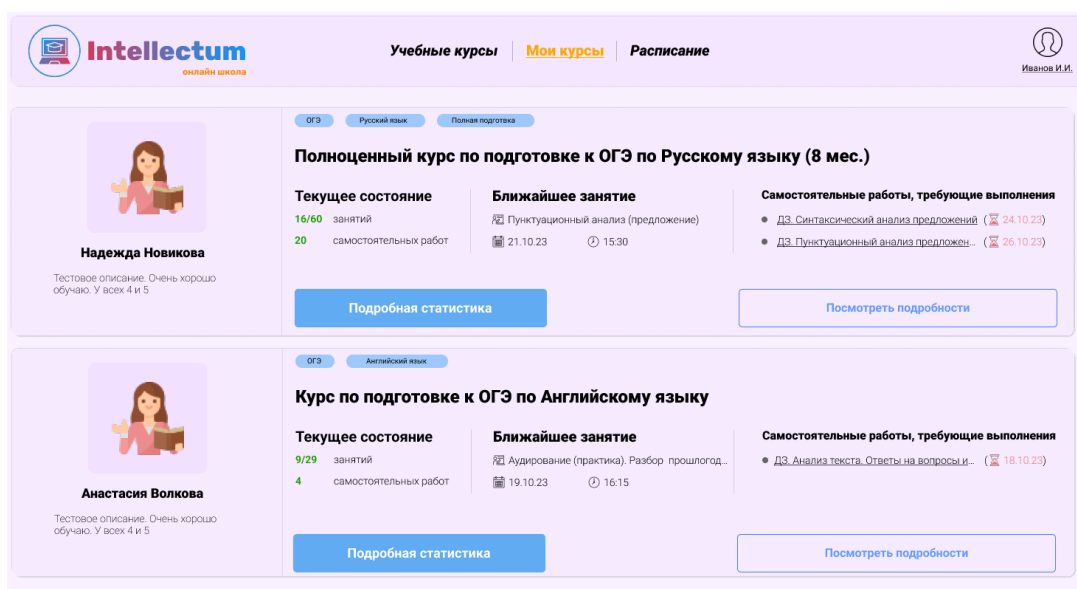


Рисунок 23 – Список приобретенных курсов

Для просмотра прогресса по выбранному курсу необходимо нажать на кнопку «Подробная статистика» (рисунок 24).

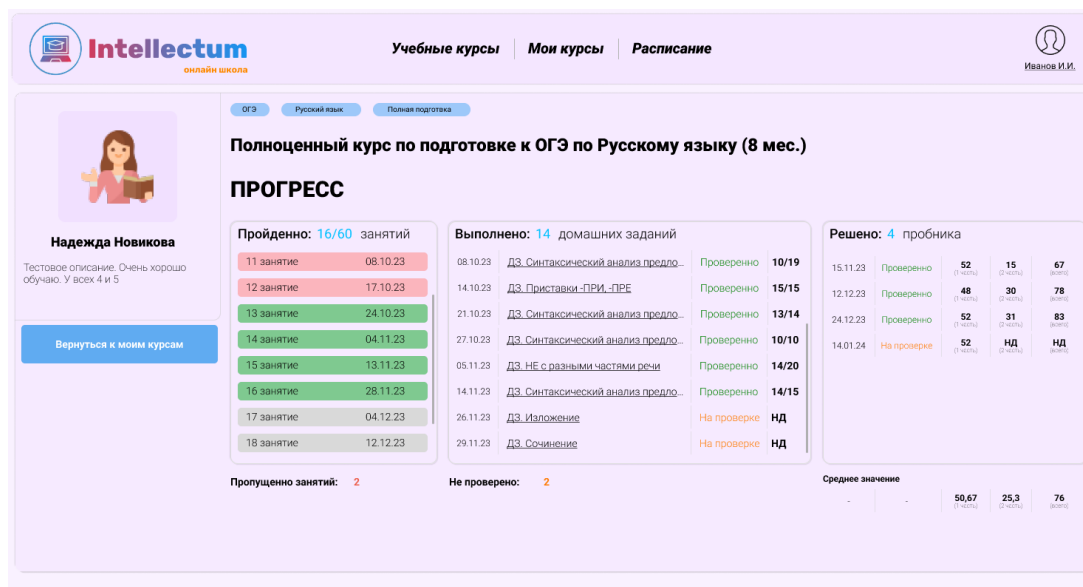


Рисунок 24 – Прогресс по курсу

На данной странице будет информация по трем основным пунктам: по пройденным занятиям, по выполненным домашним заданиям, а также по решенным пробным экзаменам.

Для выполнения самостоятельной работы необходимо ее выбрать из соответствующего списка (смотри рисунок 23). После выбора необходимой работы, ученика переадресует на страницу, изображенную на рисунке 25.

The screenshot displays the 'Intellectum' online school interface for a homework assignment. The user is logged in as 'Иванов И.И.'. The main navigation bar includes 'Учебные курсы', 'Мои курсы', and 'Расписание'. The selected course is 'Полноценный курс по подготовке к ОГЭ по Русскому языку (8 мес.)'. The assignment is titled 'Домашнее задание: синтаксический анализ предложений'. It includes a 'Срок сдачи: 24.10.23' (Deadline: 24.10.23) and a 'Прошло: 0:12:38' (Time elapsed: 0:12:38) timer. The assignment consists of two parts: 1) 'Укажите варианты ответов, в которых даны верные характеристики предложений текста. Запишите номера ответов.' (Specify the options in which the correct characteristics of the text sentences are given. Write down the numbers of the answers.) and 2) 'Напишите сочинение' (Write an essay). The first part includes a list of five sentences and a table for recording answers. The second part includes instructions for writing an essay on the topic 'В чём ценность настоящего искусства?' (What is the value of real art?). The essay should be at least 70 words and include two arguments. The interface also features a 'Загрузить файл' (Upload file) button.

Рисунок 25 – Выполнение самостоятельной работы

На задания с кратким ответом ученик вписывает ответ в текстовое поле, а для заданий с развернутым ответом загружает файл с решением. Для завершения работы ученик должен нажать на кнопку «Завершить».

Для просмотра расписания ученик должен выбрать пункт меню «Расписание» (рисунок 26). На странице будет отображаться шесть колонок, каждая из которых соответствует дню недели. Внутри каждой колонки будут располагаться элементы, представляющие собой занятия по курсам ученика.

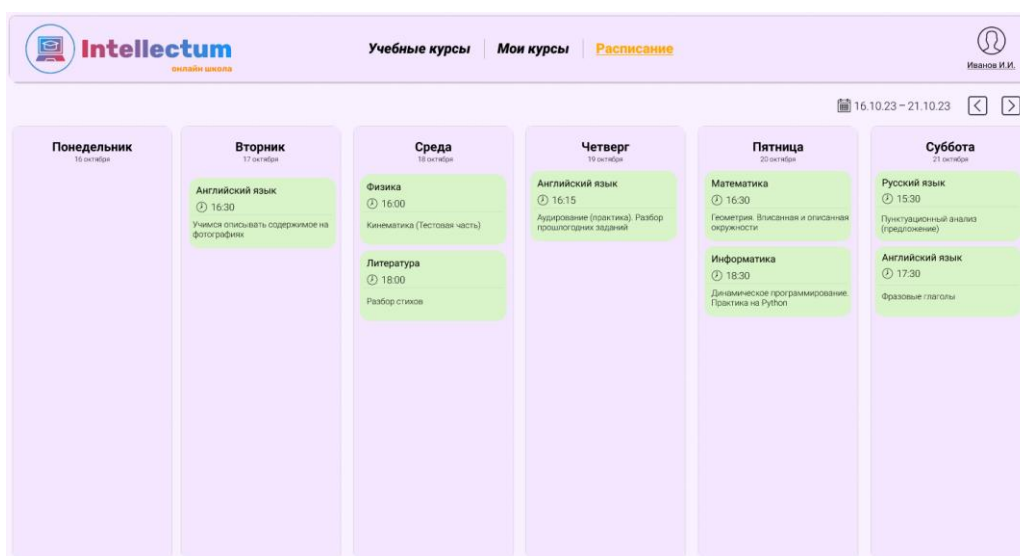


Рисунок 26 – Расписание

Пользовательский интерфейс для учителя будет отличаться.

После успешного входа в систему с ролью преподавателя, пользователя переадресует на главную страницу (рисунок 27).

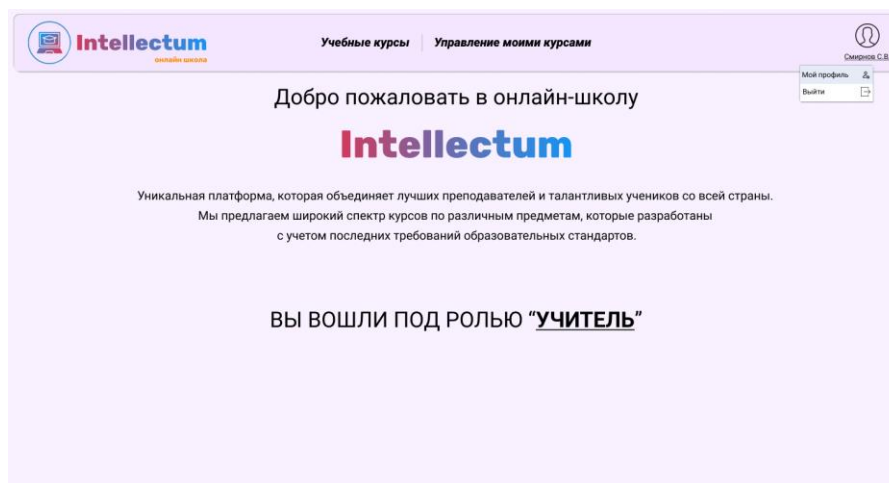


Рисунок 27 – Главная страница для преподавателя

Учитель может просмотреть существующие учебные курсы, перейдя на соответствующую страницу, выбрав пункт меню «Учебные курсы» (рисунок 28).

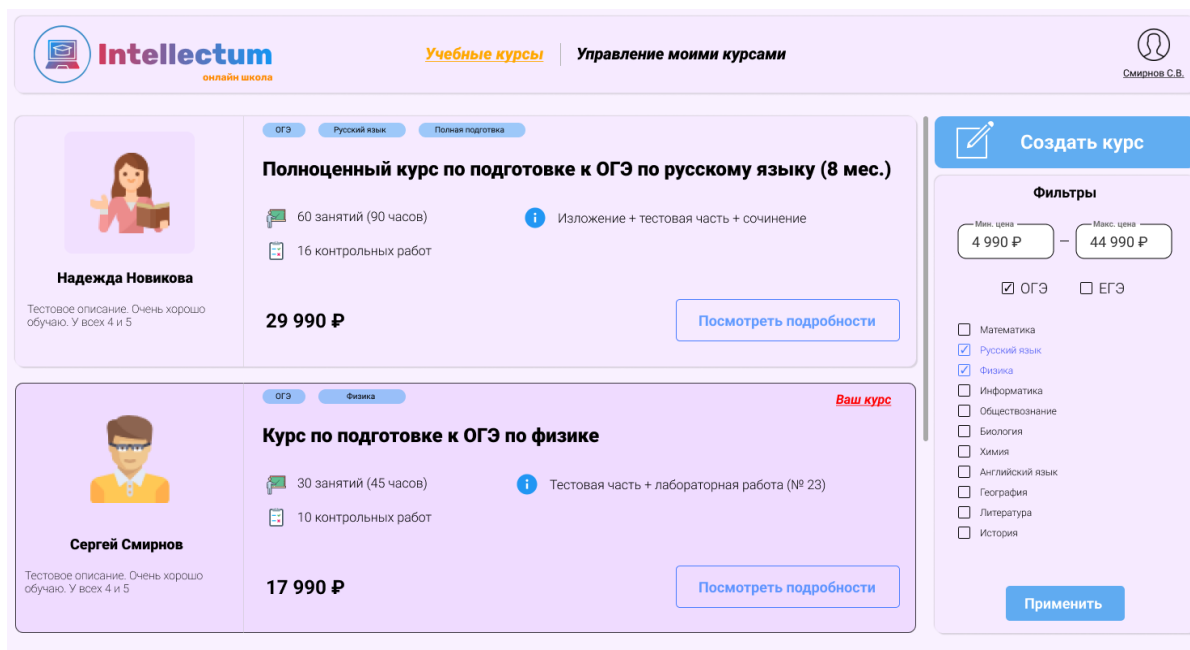


Рисунок 28 – Список курсов

Учитель может создать новый учебный курс, перейдя на соответствующую страницу, нажав на кнопку «Создать курс» (рисунок 29).

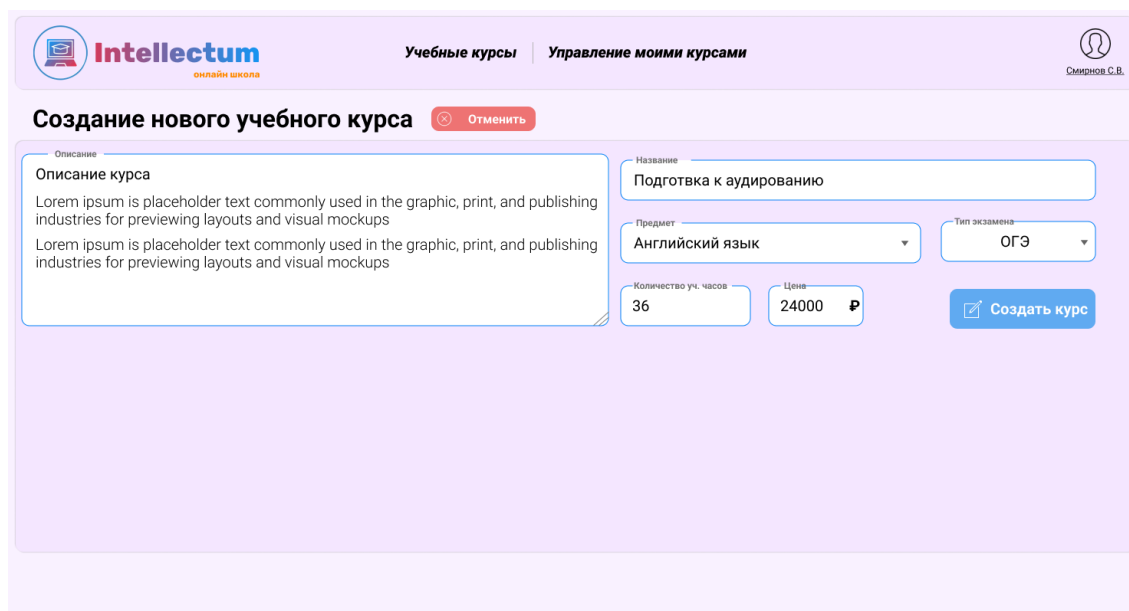


Рисунок 29 – Страница создания курса

Для управления и организации учебными курсами учителю необходимо выбрать пункт меню «Управление моими курсами» (рисунок 30).

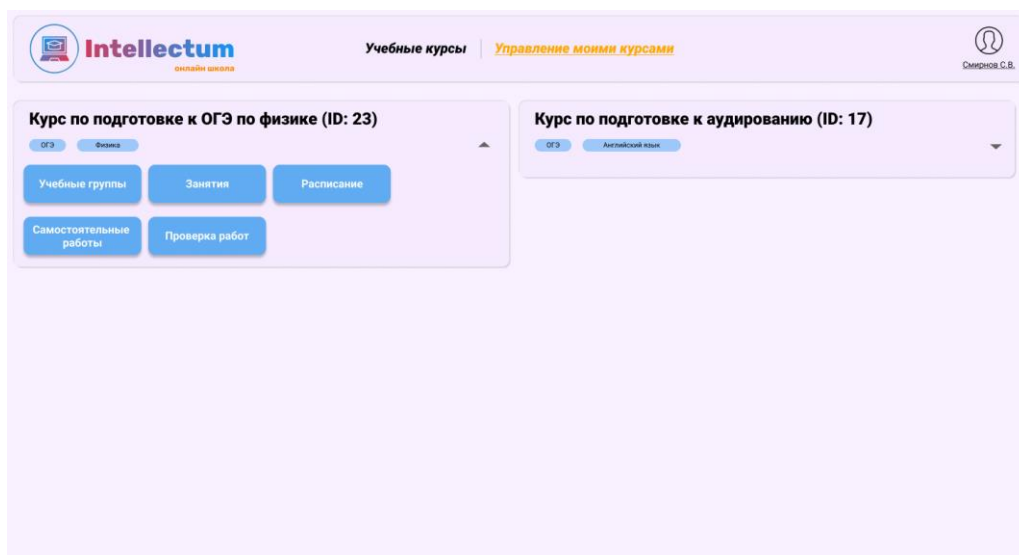


Рисунок 30 – Страница по управлению курсами

На странице будут отображены все курсы, которые ведет учитель. Выбрав и нажав на интересующий курс, внизу него отобразятся кнопки, которые соответствуют определенным действиям. Для просмотра групп учитель должен нажать на кнопку «Учебные группы» (рисунок 31).

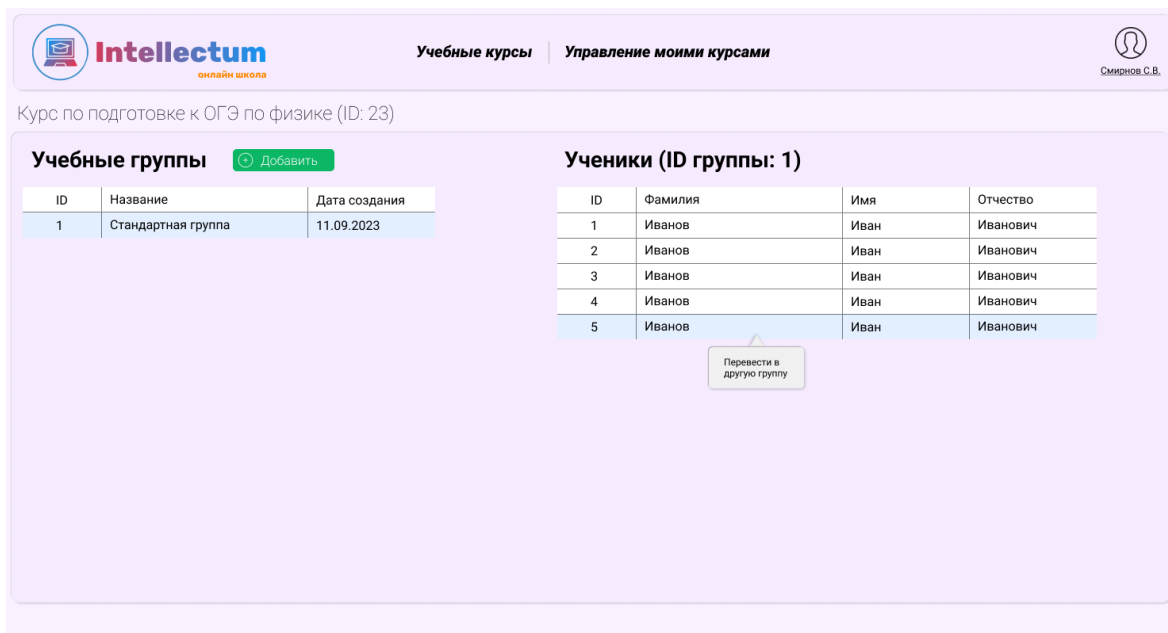


Рисунок 31 – Список групп и учеников на курсе

Учитель может создавать новые группы, нажав на кнопку «Добавить», а также перемещать учеников из одной группы в другую в рамках текущего учебного курса, нажав на соответствующий пункт контекстного меню.

Нажав на кнопку «Занятия» (смотри рисунок 30), учителя переадресует на страницу, которая содержит список занятий (рисунок 32).



Рисунок 32 – Список занятий для выбранного курса

Учитель может добавлять новые занятия, а также их редактировать, меняя название или описание.

Нажав на кнопку «Расписание» (смотри рисунок 30), учителя переадресует на страницу, которая содержит расписание занятий для групп (рисунок 33).

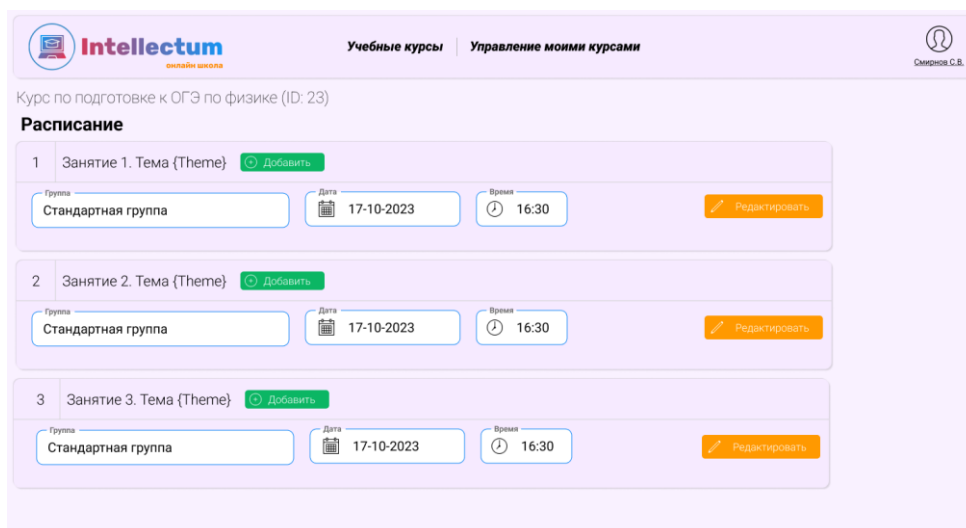


Рисунок 33 – Расписание занятий

На данной странице учитель задает дату и время проведения занятий для каждой из групп. После создания имеется возможность редактирования: изменить группу, дату или время.

Нажав на кнопку «Самостоятельные работы» (смотри рисунок 30), учителя переадресует на страницу, которая содержит список самостоятельных работ (рисунок 34).

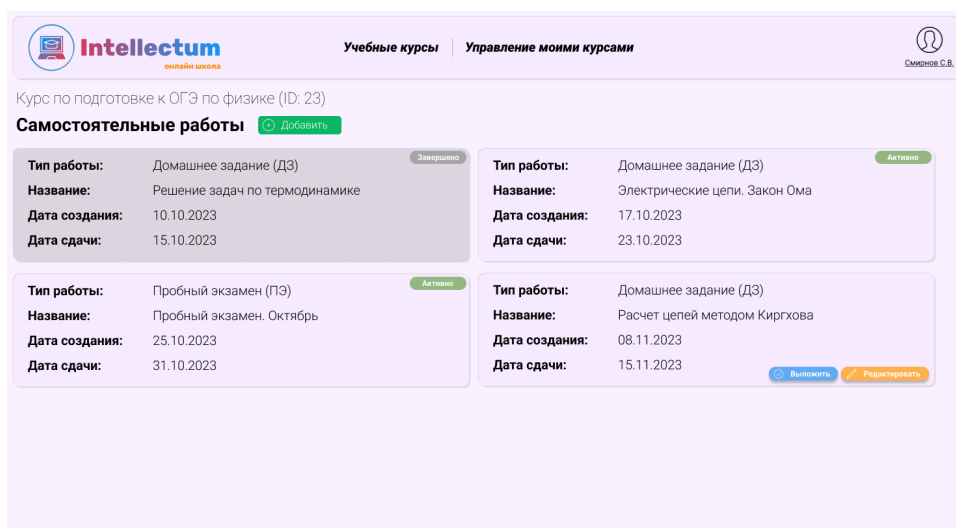


Рисунок 34 – Список самостоятельных работ на курсе

У элементов, представляющих собой самостоятельную работу, наверху имеется надпись, сообщающая о текущем состоянии работы: завершено, активно – отсутствие такой надписи означает, что работа находится в черновом варианте. Редактировать самостоятельную работу можно, нажав на кнопку «Редактировать» у соответствующей элемента (рисунок 35).

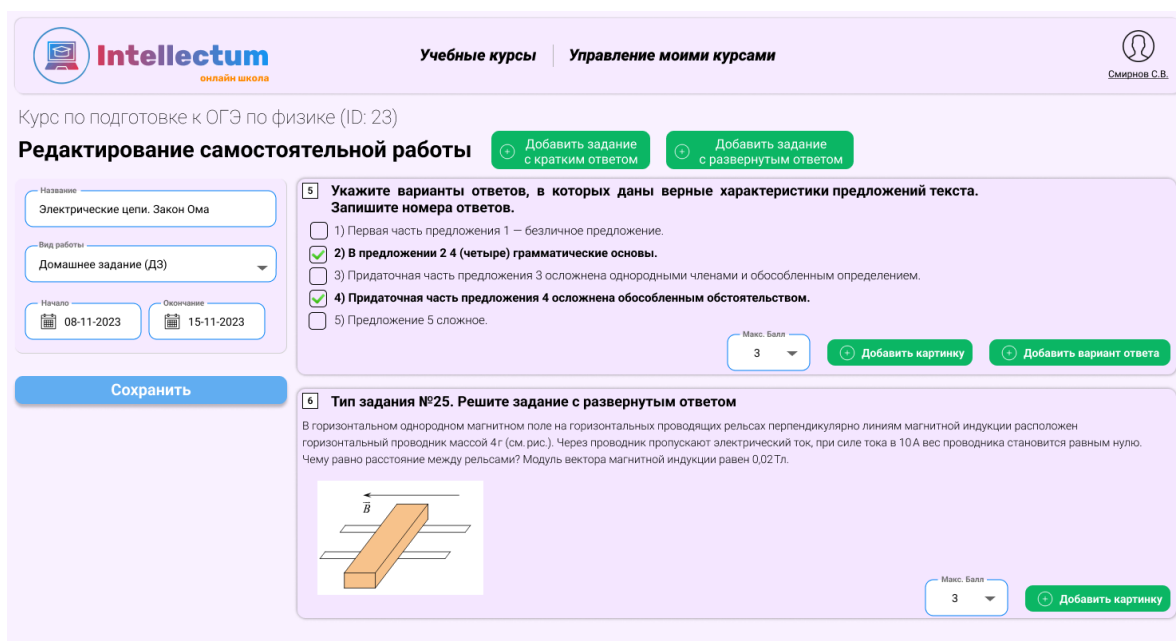


Рисунок 35 – Редактирование самостоятельной работы

Учитель может добавлять задания с кратким ответом, нажав на соответствующую кнопку, а также варианты ответов, отмечая при этом правильные из них. Аналогично учитель может добавлять задания с развернутым ответом. Для заданий необходимо указать максимальный балл за решение. Добавив необходимые задания и указав параметры для самостоятельной работы: название, тип работы, даты начала и окончания – учитель должен сохранить внесенные изменения нажав на кнопку «Сохранить».

Нажав на кнопку «Проверка работ» (смотри рисунок 30), учителя переадресует на страницу, которая содержит список выполненных самостоятельных работ, требующих проверки (рисунок 36).

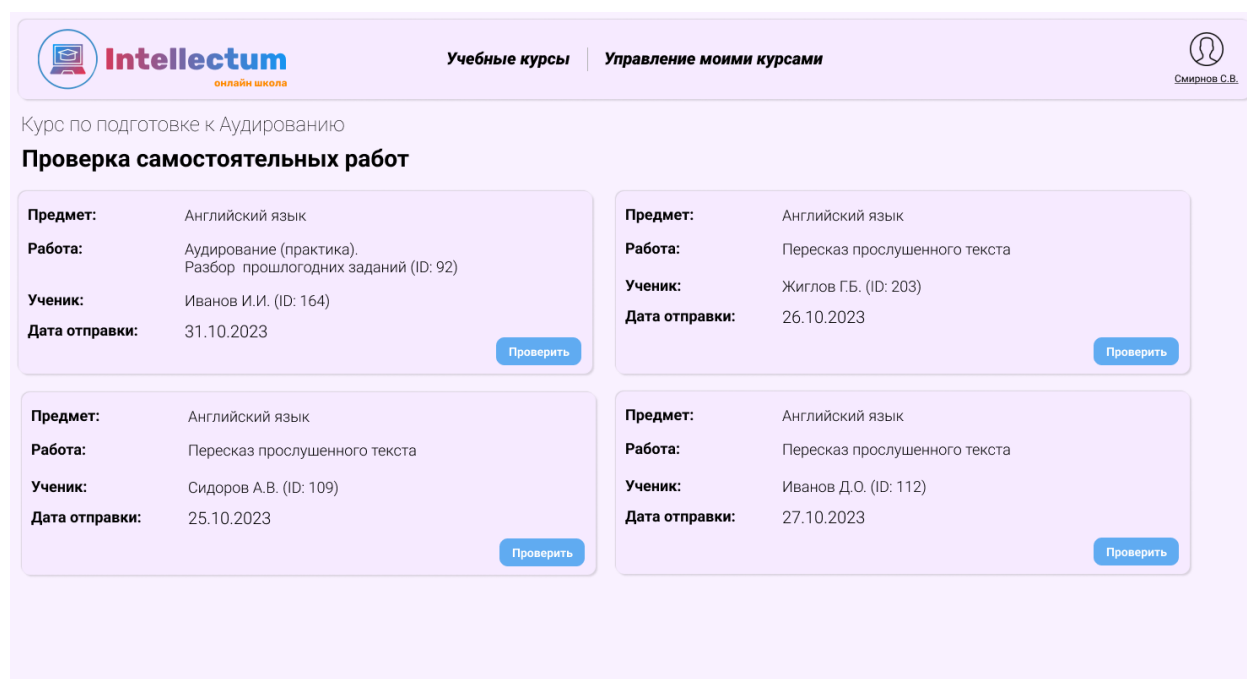


Рисунок 36 – Список самостоятельных работ на проверку

На странице отображается список самостоятельных работ, требующие проверки. Каждый элемент этого списка содержит информацию о предмете, о самостоятельной работе, об ученике, а также о дате, когда работа была выполнена. Выбрав интересующую работу на проверку, учитель должен нажать на кнопку «Проверить» (рисунок 37).


Intellectum
онлайн школа

Учебные курсы | Управление моими курсами


Смирнов С.В.

Курс по подготовке к Аудированию

Проверка самостоятельной работы

Работа

Пересказ прослушенного текста

Ученик

Сидоров А.В. (ID: 109)

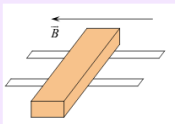
Дата отправки

25-10-2023

Сохранить

6 Тип задания №25. Решите задание с развернутым ответом

В горизонтальном однородном магнитном поле на горизонтальных проводящих рельсах перпендикулярно линиям магнитной индукции расположен горизонтальный проводник массой 4 г (см. рис.). Через проводник пропускают электрический ток, при силе тока в 10 А вес проводника становится равным нулю. Чему равно расстояние между рельсами? Модуль вектора магнитной индукции равен 0,02 Тл.



Скачать решение

Балл
3

☒ Оставить комментарий к решению

Комментарий к решению

Lorem ipsum is placeholder text commonly used in the graphic, print, and publishing industries for previewing layouts and visual mockups

Lorem ipsum is placeholder text commonly used in the graphic, print, and publishing industries for previewing layouts and visual mockups

Рисунок 37 – Проверка работы

Для оценки решения задачи, учитель должен скачать файл с решением, нажав на кнопку «Скачать решение». После проверки выставляется балл и при необходимости пишется комментарий. После проверки всех заданий, необходимо нажать на кнопку «Сохранить», чтобы закончить проверку и опубликовать результаты.

3 Реализация программных средств для онлайн-школы

3.1 Развертывание базы данных в среде MS SQL Server

Физическая модель данных – это реализация логической модели, выполняемая администраторами и разработчиками баз данных [11]. Модель разрабатывается для определенных СУБД, в данном случае для – Microsoft SQL Server. На основе логической модели была разработана физическая модель базы данных при помощи системы Visual Paradigm 17.2. Физическая модель в нотации IDEF1X представлена на рисунке Б.4 в приложении Б.

Физическая модель полностью отражает реальную структуру базы данных. Наборы атрибутов для каждой таблицы и их характеристики приведены в приложении В.

Развертывание базы данных представляет собой процесс создания и настройки новой базы данных в рамках данной системы управления базами данных (СУБД), в данном случае – Microsoft SQL Server. В контексте выпускной квалификационной работы, где требуется разработка программных средств, этот процесс может быть автоматизирован с помощью программного кода.

Для развертывания базы данных используется язык программирования TypeScript и специальная библиотека, выполняющая функции ORM, под названием TypeORM. Одним из ключевых инструментов TypeORM является механизм миграций.

Миграции позволяют управлять изменениями схемы базы данных в течение жизненного цикла приложения. Каждая миграция представляет собой набор изменений, которые необходимо внести в базу данных для обновления ее структуры. Эти изменения могут включать добавление, удаление или изменение таблиц, столбцов, индексов и других элементов схемы базы данных.

Для того, чтобы была возможность создавать таблицы базы данных, обновлять их структуру и манипулировать данными, находящимися внутри, при помощи программного кода с использованием ORM, необходимо создать классы (модели) в приложение, которые будут отвечать за объектно-реляционное

отображение с соответствующими таблицами. Пример такого класса приведен в листинге 1.

Листинг 1 – Определение класса для сущности «Самостоятельная работа»

```
@Entity("Test")
export class Test {
  @PrimaryGeneratedColumn()
  id: number;

  @ManyToOne(() => Teacher, (teacher) => teacher.tests)
  @JoinColumn({ name: "teacher_id" })
  teacher: Teacher;

  @ManyToOne(() => Course, (course) => course.tests)
  @JoinColumn({ name: "course_id" })
  course: Course;

  @OneToMany(() => TaskOfTest, (task) => task.test)
  tasks: TaskOfTest[];

  @OneToMany(() => TestStudent, (joinTable) => joinTable.test)
  testStudent: TestStudent[];

  @Column({ name: "type_of_test" })
  typeOfTest: string;

  @Column({ name: "name" })
  name: string;

  @Column({ name: "date_created" })
  dateCreated: Date;

  @Column({ name: "deadline" })
  deadline: Date;
}
```

После определения моделей в приложении необходимо описать параметры подключения к базе данных в конфигурационном файле. Пример такого файла приведен в листинге 2.

Листинг 2 – Конфигурационный файл (typeorm.config.ts)

```
export const connectionConfig = {
  type: "mysql",
  host: process.env.DATABASE_HOST,
  port: process.env.DATABASE_PORT,
  username: process.env.DATABASE_USERNAME,
  password: process.env.DATABASE_PASSWORD,
  database: process.env.DATABASE_DATABASE,
};

const appDataSource = new DataSource({
  ...connectionConfig,
  entities: ["**/*.entity.ts"],
});
```

```
migrations: ["src/migrations/*.ts"],
});

export default appDataSource;
```

Для создания, запуска и отмены миграций TypeORM предоставляет специальный интерфейс интерфейса командной строки (CLI). В листинге 3 показана команда для создания миграции.

Листинг 3 – Команда для создания миграции

```
npx typeorm-ts-node-commonjs migration:generate -- -d
src/typeorm.config.ts src/migrations/auto
```

В листинге 4 показана команда для выполнения миграции.

Листинг 4 – Команда для запуска миграций

```
npx typeorm-ts-node-commonjs migration:run -- -d src/typeorm.config.ts
```

Таким образом, использование TypeORM и механизма миграций позволяет автоматизировать процесс развертывания базы данных, делая его быстрым, надежным и легко повторяемым.

3.2 Наполнение базы тестовыми данными

Наполнение базы тестовыми данными будет происходить при помощи программного кода с использованием специальных библиотек: `faker` и `typeorm-seeding`. Первая из них отвечает за генерирование объектов со случайными значениями полей, а вторая – за автоматическое выполнение SQL запросов на добавление тестовых данных.

В листинге 5 приведена функция, представляющая собой фабрику по созданию учебных курсов со случайными значениями полей.

Листинг 5 – Фабрика для создания учебных курсов

```
define(Subject, (faker) => {
  const subject = new Subject();
  subject.name = faker.string.alpha({length: {min: 5, max: 10}});
  subject.description = faker.lorem.paragraphs(2);
  subject.typeOfExam = faker.helpers.arrayElement(["ОГЭ", "ЕГЭ"]);
  subject.numberOfStudyHours = faker.number.int({min: 10, max: 100});
  return subject;
});
```

В листинге 6 приведен класс, отвечающий за автоматическое выполнение запросов в базу данных.

Листинг 6 – Класс, заполняющий базу данных

```
export default class InitialDatabaseSeed implements Seeder {
  public async run(factory: Factory, connection: Connection):
  Promise<void> {
    const [ numOfSubjects, numOfTeachers ] = [ 5, 7 ];
    const subjects = await factory(Subject)().createMany(numOfSubjects);
    const teachers = await factory(Teacher)().createMany(numOfTeachers);

    for(let i = 0, num = 0; i < numOfTeachers; i++) {
      const teacher = teachers[i];
      await factory(Course)()
        .map(async (course) => {
          course.Subject = subjects[num];
          course.Teacher = teacher;
          return course;
        })
        .create();
      num = (num + 1) % numOfSubjects;
    }
  }
}
```

После создания фабрик для сущностей и определения классов, отвечающих за наполнение базы данных, необходимо выполнить специальную команду, приведенную в листинге 7.

Листинг 7 – Команда для заполнения базы данных

```
npx ts-node ./node_modules/typeorm-seeding/dist/cli.js seed
```

По итогу выполнения описанных шагов, база будет заполнена случайными данными.

3.3 Разработка программного обеспечения онлайн-школы

Серверная часть разрабатываемого приложения для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ реализована на языке программирования TypeScript с использованием фреймворка NestJS. Исходный код приведен в приложении Г. На рисунке 38 представлена диаграмма компонентов приложения.

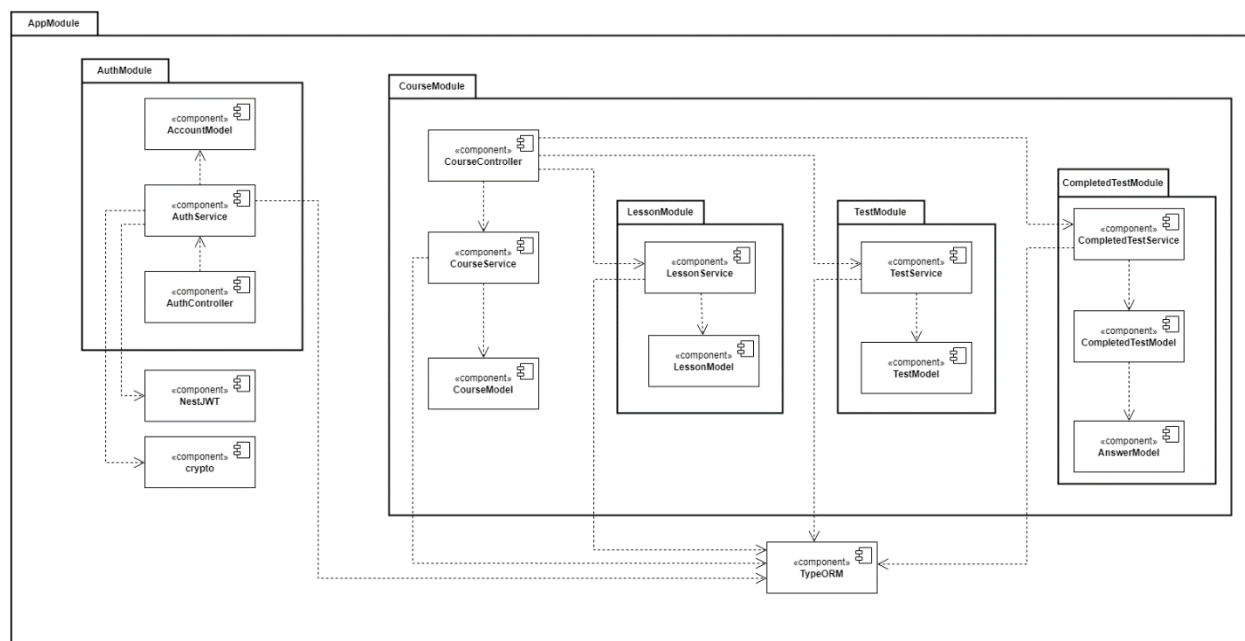


Рисунок 38 – Диаграмма компонентов

В таблице 4 приведено краткое описание для компонентов системы.

Таблица 4 – Описание компонентов системы

Название компонента	Описание компонента
TypeORM	Компонент для работы с базой данных в объектно-ориентированном стиле (ORM)
Crypto	Компонент, предоставляющий криптографическую функциональность
NestJWT	Компонент для работы с JWT: создание и проверка токенов
AccountModel	Сущность, хранящая авторизационные данные
AccountService	Компонент, отвечающий за логику авторизации и регистрации
AccountController	Компонент, содержащий логику по обработке HTTP запросов для авторизации и регистрации
CourseModel	Сущность, хранящая данные по учебному курсу
CourseService	Компонент, отвечающий за логику, связанную с учебными курсами
CourseController	Компонент, содержащий логику по обработке HTTP запросов при работе с учебными курсами
LessonModel	Сущность, хранящая данные по занятиям
LessonService	Компонент, отвечающий за логику, связанную с занятиями
TestModel	Сущность, хранящая данные по самостоятельным работам

Продолжение таблицы 4

TestService	Компонент, отвечающий за логику, связанную с самостоятельными работами
AnswerModel	Сущность, хранящая данные по ответам учеников
CompletedTestModel	Сущность, хранящая данные по выполненным самостоятельным работам
CompletedTestService	Компонент, отвечающий за логику, связанную с выполненными самостоятельными работами

Серверная часть будет представлять из себя REST API приложение. В таблице 5 описаны все конечные точки (end-points), к которым можно отправлять HTTP запросы для выполнения определенных операций. Каждая конечная точка представляет собой конкретный URI (Uniform Resource Identifier), который соответствует определенному ресурсу или функциональности системы.

Таблица 5 – Описание конечных точек серверной части приложения

URI	HTTP метод	Описание	Требуемая роль
api/auth/register	POST	регистрация нового ученика	
api/auth/login	POST	вход в систему	
api/auth/logout	POST	выход из системы	Ученик. Преподаватель
api/users/me	GET	получение информации о текущем пользователе	Ученик. Преподаватель
api/users/me	PUT	обновление информации о текущем пользователе	Ученик. Преподаватель
api/courses	GET	получение списка всех курсов	Ученик. Преподаватель
api/courses	POST	создание нового курса	
api/courses/{courseId}	GET	получение информации о конкретном курсе	
api/courses/{courseId}	PUT	обновление информации о курсе	Преподаватель
api/courses/{courseId}	DELETE	удаление курса	Преподаватель
api/courses/{courseId}/enroll	POST	запись на курс	Ученик
api/courses/{courseId}/unenroll	POST	отмена записи на курс	Ученик
api/courses/{courseId}/schedule	GET	получение расписания курса	Ученик. Преподаватель
api/courses/{courseId}/schedule	PUT	обновление расписания курса	Преподаватель

Продолжение таблицы 5

api/courses/{courseId}/lessons	GET	получение списка всех занятий курса	Ученик. Преподаватель
api/courses/{courseId}/lessons	POST	создание нового занятия	Преподаватель
api/courses/{courseId}/lessons/{lessonId}	GET	получение информации о конкретном занятии	Ученик. Преподаватель
api/courses/{courseId}/lessons/{lessonId}/report	POST	Отметить присутствующих	Преподаватель
api/courses/{courseId}/lessons/{lessonId}	PUT	обновление информации о занятии	Преподаватель
api/courses/{courseId}/lessons/{lessonId}	DELETE	удаление занятия	Преподаватель
api/courses/{courseId}/tests	GET	получение списка всех самостоятельных работ курса	Ученик. Преподаватель
api/courses/{courseId}/tests	POST	создание новой самостоятельной работы	Преподаватель
api/courses/{courseId}/tests/{testId}	GET	получение информации о конкретной самостоятельной работе	Ученик. Преподаватель
api/courses/{courseId}/tests/{testId}	PUT	обновление информации о самостоятельной работе	Преподаватель
api/courses/{courseId}/tests/{testId}/start-doing	POST	Начать выполнять самостоятельную работу	Ученик
api/courses/{courseId}/tests/{testId}/submit	POST	Отправить выполненную самостоятельную работу	Ученик
api/courses/{courseId}/completed-tests	GET	получение списка всех выполненных работ	Преподаватель
api/courses/{courseId}/completed-tests/me	GET	получение списка всех выполненных работ текущего ученика	Ученик
api/courses/{courseId}/completed-tests/me/{testId}	GET	получение конкретной выполненной работы текущего ученика	Ученик
api/courses/{courseId}/completed-tests/{studentId}	GET	получение конкретной выполненной работы	Преподаватель
api/courses/{courseId}/completed-tests/{studentId}/{taskId}/grade	POST	оценка выполненного задания	Преподаватель

На рисунке Б.5 в приложение Б представлена диаграмма развертывания приложения.

Серверная и клиентская части приложения, а также база данных будут развернуты внутри отдельных Docker контейнеров. Для каждого такого

контейнера будет определен специальный файл – Dockerfile, содержащий инструкции по его созданию.

Все компоненты разрабатываемого приложения будут развернуты внутри узла «Linux сервер», представляющего собой компьютер, на котором установлена серверная операционная система семейства Linux.

Узел «Database» представляет из себя базу данных – SQL Server. Данный узел будет отвечать за хранение данных. Доступ к этому узлу возможен только по адресу локального хоста по 5433 TCP порту. Обращение к базе данных будет происходить только от серверной части приложения.

Узел «Backend» представляет из себя серверную часть приложения. Внутри данного узла будут выполняться все бизнес процессы. Доступ к этому узлу возможен только по адресу локального хоста по 6000 TCP порту. Обращение будет происходить только от клиентской части приложения.

Узел «Frontend» представляет из себя клиентскую часть приложения. Данный узел будет отвечать за пользовательский интерфейс. Доступ к этому узлу возможен по 80 TCP порту.

Узел «Устройство клиента» представляет из себя компьютер (смартфон) пользователя, на котором установлен WEB-браузер, с помощью которого он может открыть приложение, обратившись по соответствующему URL адресу по 80 TCP порту.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует достаточно много видов тестирования программных продуктов [12]. Наиболее популярны следующие виды тестирования:

- функциональное тестирование;
- нефункциональное тестирование;
- интеграционное тестирование.

Функциональное тестирование – процесс обеспечения качества (QA) в рамках цикла разработки программного обеспечения, необходимый для проверки реализуемости функциональных требований, согласно спецификации тестируемого программного обеспечения. Оно проводится для оценки соответствия системы или компонента заданным функциональным требованиям. Функциональное тестирование проводится по принципу черного ящика, в связи с чем функциональность ПО можно протестировать, не зная принципа его внутренней работы. Это снижает требования к тестировщикам в части знания языков программирования или конкретных аспектов реализации программного обеспечения.

Нефункциональное тестирование – вид тестирования, направленный на проверку характеристик программного продукта, которые не связаны напрямую с его функциональностью. К таким характеристикам относятся производительность, надежность, безопасность, удобство использования (usability) и другие. Целью нефункционального тестирования является оценка того, насколько хорошо продукт соответствует требованиям заказчика и ожиданиям пользователей в отношении этих характеристик.

Интеграционное тестирование – это процесс проверки взаимодействия между различными компонентами системы. Оно направлено на выявление проблем совместимости и взаимодействия между отдельными модулями программы. В ходе интеграционного тестирования проверяются связи между компонентами, их взаимодействие друг с другом и с другими системами.

Было принято решение осуществить проверку приложения с помощью функционального тестирования. Данный вид тестирования был выбран потому, что он позволяет проверить соответствие программного продукта его функциональным требованиям. Это означает, что **мы** можем убедиться, что программа выполняет все необходимые функции правильно и без ошибок. Кроме того, функциональное тестирование помогает выявить проблемы в интерфейсе пользователя и логике работы программы, что важно для обеспечения качества продукта.

4.2 Тестирование серверной части приложения

Для выполнения функционального тестирования серверной части приложения, можно использовать инструменты вроде Postman для ручного тестирования или написать автоматизированные тесты. Тестирование будет проводиться вторым способом на языке программирования Python с использованием фреймворка PyTest и двух специальных библиотек: Requests и Faker. Requests позволяет выполнять HTTP запросы, Faker генерирует случайные данные.

Набор созданных тестов включает основные проверки для функциональности серверной части приложения:

- регистрация и вход пользователей;
- создание курсов;
- запись учеников на курсы;
- добавление занятий.

В таблице 6 приведено описания для каждого теста.

Таблица 6 – Описание тестов

Состав теста	Ожидаемый результат	Наблюдаемый результат	Результат теста
Регистрация нового пользователя с корректно введенными данными (test_registration)	В системе появиться запись о новом пользователе	В системе появился новый пользователь. Сервер вернул 201 HTTP код.	Тест пройден

Продолжение таблицы 6

Вход пользователя в систему (test_login)	Пользователь успешно пройдет аутентификацию и получит токен для авторизации	Пользователь успешно вошел в систему. Сервер вернул 200 HTTP код, а также токен доступа (JWT)	Тест пройден
Создание нового учебного курса (test_create_course)	Преподаватель успешно создает новый курс, который добавляется в систему	Курс успешно создан. Сервер вернул 201 HTTP код	Тест пройден
Запись на учебный курс (test_enroll_in_course)	Ученик успешно записывается на учебный курс	Ученик записался на курс. Сервер вернул 200 HTTP код	Тест пройден
Создание нового занятия (test_add_lesson)	Преподаватель успешно создает занятие для курса	Занятие добавлено для выбранного курса	Тест пройден
Регистрация пользователя с некорректными данными: отсутствие пароля, некорректная длина логина и пароля, некорректная роль пользователя (test_registration_invalid_data)	Ошибка регистрации	Ошибка регистрации. Сервер вернул 400 HTTP код	Тест пройден
Вход пользователя с некорректными данными: неправильный логин или пароль (test_login_invalid_data)	Ошибка аутентификации	Ошибка входа. Сервер вернул 401 HTTP код	Тест пройден
Создание курса без названия (test_create_course_invalid_data)	Ошибка создания курса	Ошибка создания курса. Сервер вернул 400 HTTP код	Тест пройден
Запись на несуществующий курс (test_enroll_in_course_invalid_data)	Ошибка записи на курс	Ошибка записи на курс. Сервер вернул 404 HTTP код	Тест пройден
Добавление занятия с некорректными данными: отсутствует название занятия, некорректный формат времени (test_add_lesson_invalid_data)	Ошибка добавления нового занятия	Ошибка добавления нового занятия. Сервер вернул 400 HTTP код	Тест пройден

Для каждого теста в таблице в скобках приведено название тестирующей функции. Исходный код автоматизированных тестов приведен в приложении Е. На рисунке 39 показан результат тестирования.

```

> pytest --no-header -vv .\api_tests.py 39% 9/23GB
===== test session starts =====
collected 10 items

api_tests.py::test_registration PASSED [ 10%]
api_tests.py::test_login PASSED [ 20%]
api_tests.py::test_create_course PASSED [ 30%]
api_tests.py::test_enroll_in_course PASSED [ 40%]
api_tests.py::test_add_lesson PASSED [ 50%]
api_tests.py::test_registration_invalid_data PASSED [ 60%]
api_tests.py::test_login_invalid_data PASSED [ 70%]
api_tests.py::test_create_course_invalid_data PASSED [ 80%]
api_tests.py::test_enroll_in_course_invalid_data PASSED [ 90%]
api_tests.py::test_add_lesson_invalid_data PASSED [100%]

===== 10 passed in 0.04s =====

```

Рисунок 39 – Результаты тестирования серверной части

В ходе тестирования были выявлены и исправлены ошибки в коде серверной части приложения.

4.3 Тестирование клиентской части приложения

В процессе разработки клиентской части программы было проведено функциональное тестирование. Результаты тестирования представлены в таблице 7.

Таблица 7 – Результаты тестирования клиентской части приложения

Состав теста	Ожидаемый результат	Наблюдаемый результат	Результат теста
Регистрация ученика с некорректно заполненной формой	Регистрация не успешна	Регистрация не успешна. Все некорректно заполненные поля, отмечены красным цветом, кнопка отправки не активна	Тест пройден (Рисунок Е.1)
Вход в систему с неверным паролем	Ошибка аутентификации	Ошибка аутентификации. Отображение всплывающего окна, информирующего об ошибке	Тест пройден (Рисунок Е.2)
Получить список отфильтрованных курсов	Отобразятся только подходящие под фильтры курсы	Отобразились только желаемые курсы	Тест пройден (Рисунок Е.3)

Продолжение таблицы 7

Проверка самостоятельной работы	После проверки работы ученику должно прийти уведомление	Ученику пришло уведомление	Тест пройден (Рисунок Е.4)
Время отведенное на выполнение самостоятельной работы вышло	Самостоятельная работа автоматически помечается завершенной	У работы появилась надпись «Завершено»	Тест пройден (Рисунок Е.5)

В ходе тестирования были выявлены и исправлены ошибки в коде программы.

4.4 Метрики кода

В процессе разработки программного обеспечения крайне важно иметь инструменты и методы для оценки качества кода. Метрики кода позволяют объективно измерить различные аспекты программного продукта, что способствует его улучшению и поддержке.

Метрика устойчивости кода показывает, насколько легко или сложно модулю изменяться без нарушения работы других частей системы. Для вычисления используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где I – мера неустойчивости, изменяется в диапазоне от $[0; 1]$; 0 означает максимальную устойчивость (модуль ни от кого не зависит), а 1 – минимальную устойчивость (модуль зависит от большого числа других модулей).

Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

В созданном приложении можно выделить:

- модуль регистрации и авторизации;
- модуль по взаимодействию с учебными курсами;

- модуль по взаимодействию с занятиями;
- модуль по взаимодействию с самостоятельными работами.

Для каждой компоненты была просчитана величина I – метрика неустойчивости:

$$I_1 = \frac{4}{7} = 0.57,$$

$$I_2 = \frac{0}{3} = 0,$$

$$I_3 = \frac{0}{3} = 0,$$

$$I_4 = \frac{0}{2} = 0.$$

Все модули, кроме модуля авторизации, имеют максимальную меру устойчивости.

Метрика абстрактности оценивает долю абстрактных классов и интерфейсов в модуле по сравнению с общим числом классов в модуле. Это помогает понять, насколько проект следует принципам объектно-ориентированного программирования, таким как инкапсуляция и полиморфизм.

Для вычисления меры абстрактности используется следующая формула:

$$A = \frac{N_a}{N_c},$$

где A – мера абстрактности, изменяется в диапазоне от 0 до 1, где 0 означает полное отсутствие абстрактных классов в компоненте, а 1 означает, что компонент не содержит ничего, кроме абстрактных классов;

N_c – количество классов в компоненте;

N_a – количество абстрактных классов в том же компоненте.

Для всех компонентов приложения была вычислена мера абстрактности:

$$A_1 = \frac{2}{5} = 0.4,$$

$$A_2 = \frac{1}{4} = 0.25,$$

$$A_3 = \frac{1}{4} = 0.25,$$

$$A_4 = \frac{2}{4} = 0.5.$$

Метрика главная последовательность используется для определения сбалансированности между абстрактностью и устойчивостью. Идеальным считается модуль, который находится на «главной последовательности» графика (рисунок 40), где ось X представляет устойчивость, а ось Y – абстрактность.

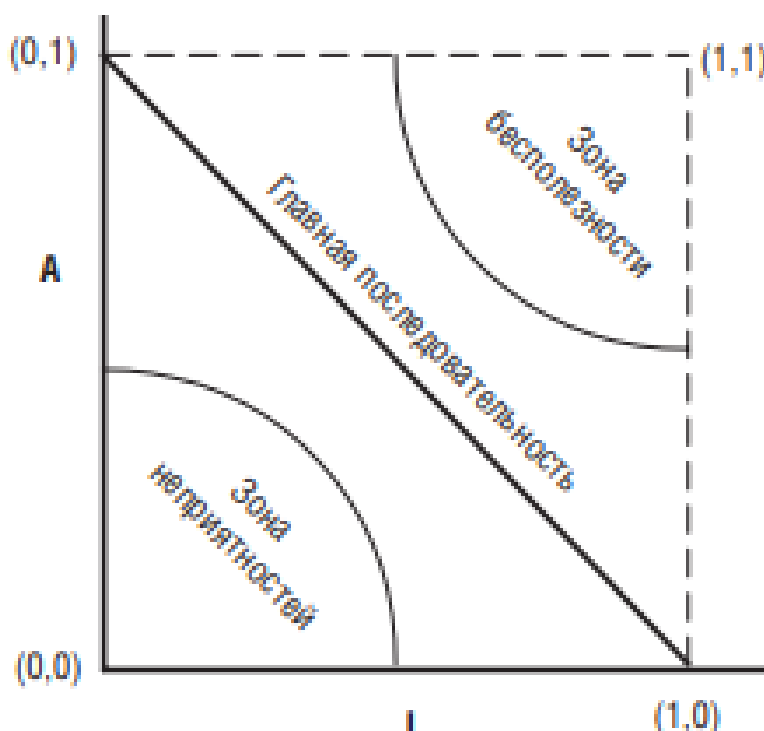


Рисунок 40 – Главная последовательность и зоны исключения

Для вычисления количественной оценки используется формула вычисления расстояние до главной последовательности:

$$D = \left| \frac{(A + I - 1)}{\sqrt{2}} \right|$$

Идеальное значение D – это 0, что означает, что модуль идеально сбалансирован в плане устойчивости и абстрактности. Чем больше отклонение от нуля, тем менее сбалансирован модуль.

Было вычислено расстояние до главной последовательности:

$$D_1 = \left| \frac{0.4 + 0.57 - 1}{\sqrt{2}} \right| = 0.021,$$

$$D_2 = \left| \frac{0.25 + 0 - 1}{\sqrt{2}} \right| = 0.53,$$

$$D_3 = \left| \frac{0.25 + 0 - 1}{\sqrt{2}} \right| = 0.53,$$

$$D_4 = \left| \frac{0.5 + 0 - 1}{\sqrt{2}} \right| = 0.353.$$

Большинство модулей имеют значения D , близкие к 0, что свидетельствует о хорошем балансе между абстрактностью и устойчивостью.

Первый модуль (модуль авторизации) показал наилучший результат с минимальным значением, что делает его наиболее сбалансированным.

5 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета проекта была выполнена при помощи Microsoft Project. Дата начала проекта – 8 апреля 2024 года. В результате планирования было выделено 36 задач. Задачи проекта, а также их длительность и данные о предшествующих задачах представлены на рисунке 41.

Режим задачи	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 08.04.24	Пн 08.04.24		
2	Подготовительная фаза	8,5 дней	Пн 08.04.24	Чт 18.04.24		
3	Исследование и анализ требований	7,5 дней	Пн 08.04.24	Ср 17.04.24		
4	Определение целевой аудитории	2 дней	Пн 08.04.24	Вт 09.04.24	1	Аналитик
5	Анализ конкурентов	3 дней	Ср 10.04.24	Пт 12.04.24	4	Аналитик
6	Сбор и анализ требований	2,5 дней	Пн 15.04.24	Ср 17.04.24	5	Аналитик
7	завершение анализа	0 дней	Ср 17.04.24	Ср 17.04.24	6	
8	Составление технического задания	1 день	Ср 17.04.24	Чт 18.04.24	7	Аналитик
9	Подготовительная фаза завершена	0 дней	Чт 18.04.24	Чт 18.04.24	8	
10	Проектирование	7 дней	Чт 18.04.24	Пн 29.04.24		
11	Проектирование архитектуры приложения	2 дней	Чт 18.04.24	Пн 22.04.24		
12	Определение технологического стека	1 день	Чт 18.04.24	Пт 19.04.24	9	Бэкендер[50%];Фронтендер[50%]
13	Создание схемы базы данных	1 день	Пт 19.04.24	Пн 22.04.24	12	Бэкендер
14	Проектирование архитектуры завершено	0 дней	Пн 22.04.24	Пн 22.04.24	13	
15	Прототипирование основных страниц	5 дней	Чт 18.04.24	Чт 25.04.24	9	
16	Подготовка проектной документации	2 дней	Чт 25.04.24	Пн 29.04.24	14;15	Бэкендер[50%];Фронтендер[25%];Дизайнер[25%]
17	Проектирование завершено	0 дней	Пн 29.04.24	Пн 29.04.24	16	
18	Разработка	8 дней	Пн 29.04.24	Чт 09.05.24		
19	Создание основных страниц и компонентов интерфейса	5 дней	Пн 29.04.24	Пн 06.05.24	17	Фронтендер
20	Разработка серверной части приложения	8 дней	Пн 29.04.24	Чт 09.05.24		
21	Реализация бизнес-логики	5 дней	Пн 29.04.24	Пн 06.05.24	17	Бэкендер
22	Интеграция с базой данных	1 день	Пн 06.05.24	Вт 07.05.24	21	Бэкендер
23	Создание API для взаимодействия с клиентской частью приложения	2 дней	Вт 07.05.24	Чт 09.05.24	21;22	Бэкендер
24	Серверная часть приложения завершена	0 дней	Чт 09.05.24	Чт 09.05.24	23	
25	Разработка завершена	0 дней	Чт 09.05.24	Чт 09.05.24	24	
26	Тестирование	5 дней	Чт 09.05.24	Чт 16.05.24		
27	Функциональное тестирование приложения	5 дней	Чт 09.05.24	Чт 16.05.24	25	Бэкендер;Фронтендер
28	Интеграционное тестирование	2 дней	Пн 13.05.24	Ср 15.05.24	27НН+2 дней	Бэкендер
29	Тестирование пользовательского интерфейса	5 дней	Чт 09.05.24	Чт 16.05.24	25	Фронтендер
30	Тестирование завершено	0 дней	Чт 16.05.24	Чт 16.05.24	27;28;29	
31	Внедрение и запуск	5 дней	Чт 16.05.24	Чт 23.05.24		
32	Покупка домена + аренда сервера	1 день	Чт 16.05.24	Вт 21.05.24	30	Покупка доменного имени[1];Ежемесячная оплата
33	Развертывание приложения на сервере	1 день	Вт 21.05.24	Ср 22.05.24	32	Бэкендер[50%];Фронтендер[50%]
34	Настройка системы мониторинга и логирования	1 день	Ср 22.05.24	Чт 23.05.24	33	Бэкендер[50%]
35	Проект запущен	0 дней	Чт 23.05.24	Чт 23.05.24	34;32;33	
36	Завершение проекта	0 дней	Чт 23.05.24	Чт 23.05.24	35	

Рисунок 41 – Список задач проекта

Список ресурсов, используемых при выполнении задач представлен на рисунке 42.

	Название ресурса	Тип	Единицы измерения	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочн	Затраты на исполыз.	Начисление
1	Аналитик	Трудовой		А		100%	130 000,00 Р/мес	1 000,00 Р/ч	0,00 Р	Пропорционал
2	Дизайнер	Трудовой		Д		100%	80 000,00 Р/мес	600,00 Р/ч	0,00 Р	Пропорционал
3	Фронтендер	Трудовой		Фронт		100%	90 000,00 Р/мес	800,00 Р/ч	0,00 Р	Пропорционал
4	Бэкендер	Трудовой		БЭК		100%	100 000,00 Р/мес	900,00 Р/ч	0,00 Р	Пропорционал
5	Покупка доменного имени	Материальн		П			10 000,00 Р		0,00 Р	Пропорционал
6	Ежемесячная оплата хостинг	Затраты		Е						Пропорционал

Рисунок 42 – Список ресурсов

В результате выделения задач и ресурсов была построена диаграмма Ганта (рисунок 43). На ней представлены продолжительность задач и взаимосвязи их друг с другом.

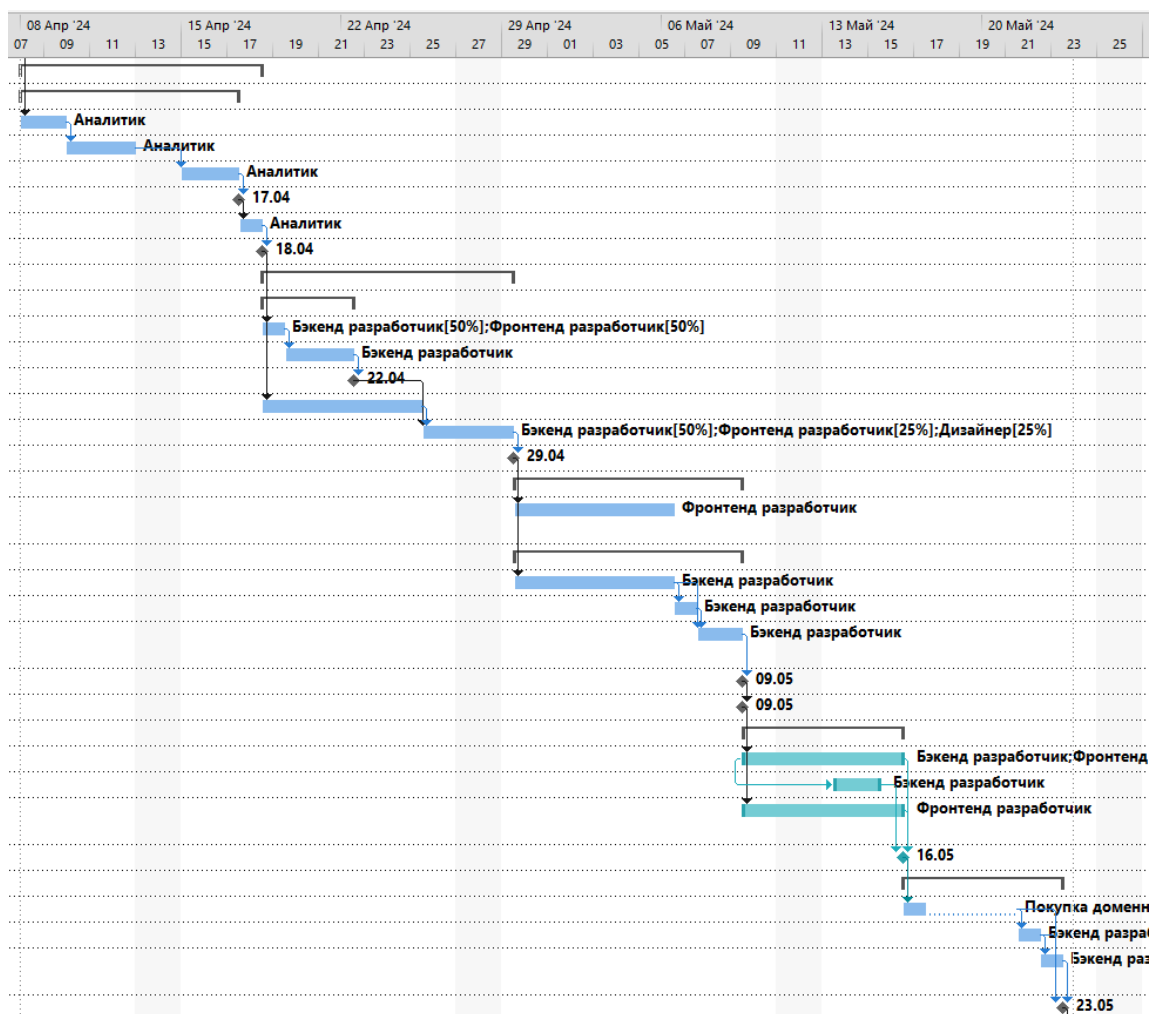


Рисунок 43 – Диаграмма Ганта

На рисунке 44 представлена сводная информация по проекту.

	Начало		Окончание	
Текущее	Пн 08.04.24		Чт 23.05.24	
Базовое	НД		НД	
Фактическое	НД		НД	
Отклонение	0д		0д	
	Длительность	Трудозатраты	Затраты	
Текущие	33,5д	352ч	249 000,00 Р	
Базовые	0д	0ч	0,00 Р	
Фактические	0д	0ч	0,00 Р	
Оставшиеся	33,5д	352ч	249 000,00 Р	
Процент завершения				
Длительность: 0%		Трудозатраты: 0%		Заккрыть

Рисунок 44 – Сводная информация по проекту

Таким образом, разработка проекта составит 34 полных рабочих дней с 08.04.24 по 23.05.24. Трудозатраты составят 352 человеко-часов. Оценка бюджета проекта составляет 249 000,00 рублей.

Заключение

В рамках выпускной квалификационной работы созданы программные инструменты для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ.

Был проведен анализ предметной области, изучены актуальные информационные материалы по данной тематике, выявлены наиболее востребованные аналогичные продукты.

Осуществлен сравнительный анализ существующих аналогов от различных онлайн-школ, выявлены их сильные и слабые стороны. На основании полученных результатов сформулированы ключевые требования к разрабатываемым программным инструментам. Разработана диаграмма использования, которая описывает основные функции онлайн-школы по подготовке учеников к экзаменам: ОГЭ и ЕГЭ.

На стадии проектирования приложения была определена его архитектура, разработана модель базы данных и спроектирован пользовательский интерфейс. Были также созданы концептуальная и логическая модели базы данных, а также диаграмма классов UML.

На этапе реализации создана физическая модель базы данных, на основе которой в MSSQL была развернута соответствующая схема базы данных, и все необходимые таблицы были заполнены тестовыми данными. Была разработана диаграмма компонентов приложения.

Разработка программного обеспечения осуществлялась в интегрированной среде разработки VS Code на языке программирования TypeScript в операционной системе Ubuntu 20.04 с использованием следующих фреймворков: NextJS для реализации клиентской части приложения, NestJS – серверной части приложения.

Тестирование программного средства проводилось в той же операционной системе. Все тесты пройдены успешно и в результате тестирования не было выявлено ошибок.

В ходе работы выполнены планирование разработки и оценка бюджета при помощи Microsoft Project. Длительность проекта составила 34 дня, а суммарные затраты равны 249 000,00 рублей.

Таким образом в результате выполнения выпускной квалификационной работы разработаны программные средства для организации и управления курсами по подготовке школьников к ОГЭ и ЕГЭ.

Список использованных источников

1. Maximum Education [Электронный ресурс]. URL: <https://maximumtest.ru> (дата обращения 07.06.2024);
2. Фоксфорд [Электронный ресурс]. URL: <https://foxford.ru> (дата обращения 07.06.2024);
3. Школково [Электронный ресурс]. URL: <https://shkolково.online> (дата обращения 07.06.2024);
4. Вебиум [Электронный ресурс]. URL: <https://webium.ru> (дата обращения 07.06.2024);
5. Сдам ГИА [Электронный ресурс]. URL: <https://sdamgia.ru> (дата обращения 07.06.2024);
6. Арлоу Д. UML 2 и Унифицированный процесс, 2-е издание. Практический объектно-ориентированный анализ и проектирование. / Арлоу Д., Нейштадт А. – СПб: Символ Плюс, 2007. – 624 с.
7. NestJS [Электронный ресурс]. – URL: <https://nestjs.com> (дата обращения 07.06.2024);
8. Microsoft SQL Server [Электронный ресурс]. URL: <https://www.microsoft.com/ru-ru/sql-server/sql-server-2019> (дата обращения 07.06.2024);
9. NextJS [Электронный ресурс]. – URL: <https://nextjs.org> (дата обращения 07.06.2024);
10. Костюкова А. Базы данных: общий курс. Часть 1: учебное пособие. – Москва.: Академия Естествознания, 2018. – ISBN 978-5-91327-538-7.
11. Моделирование данных. [Электронный ресурс]. URL: <https://powerbi.microsoft.com/ru-ru/what-is-data-modeling> (дата обращения: 31.01.2024)
12. Лайза Криспин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile

Testing: A Practical Guide for Testers and Agile Teams. – М.: «Вильямс», 2010. – 464 с. – (Addison-Wesley Signature Series).

Приложение А
(справочное)
Глоссарий

UML (Unified Modeling Language) – это унифицированный язык моделирования, который используется для графического описания и объектного моделирования в разработке программного обеспечения, бизнес-процессов, системного проектирования и отображения организационных структур. Он также позволяет разработчикам программного обеспечения достичь согласия в графических обозначениях для представления общих понятий.

ORM (Object-Relational Mapping) – это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, позволяя работать с данными в терминах классов, а не таблиц данных. Это упрощает взаимодействие с базами данных и избавляет от необходимости написания SQL-кода.

SQL (Structured Query Language) – это декларативный язык программирования, который используется для создания, модификации и управления данными в реляционных базах данных.

REST API (Representational State Transfer Application Programming Interface) – архитектурный стиль взаимодействия компонентов распределенного приложения в сети, который использует HTTP для взаимодействия между клиентом и сервером. REST API предоставляет доступ к ресурсам через URL и принимает и отправляет данные в формате JSON. Основные принципы REST API включают клиент-серверную архитектуру, отсутствие состояния сессии на сервере, использование HTTP методов (GET, POST, PUT, DELETE) и идентификаторов ресурсов для манипулирования данными.

Docker – это программное обеспечение, которое используется для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации. Оно позволяет упаковать приложение со всеми его зависимостями и окружением в контейнер, который может быть развернут на любой Linux-системе с поддержкой контрольных групп в ядре. Docker предоставляет набор команд для управления этими контейнерами.

Приложение Б
(рекомендуемое)
Графическая часть

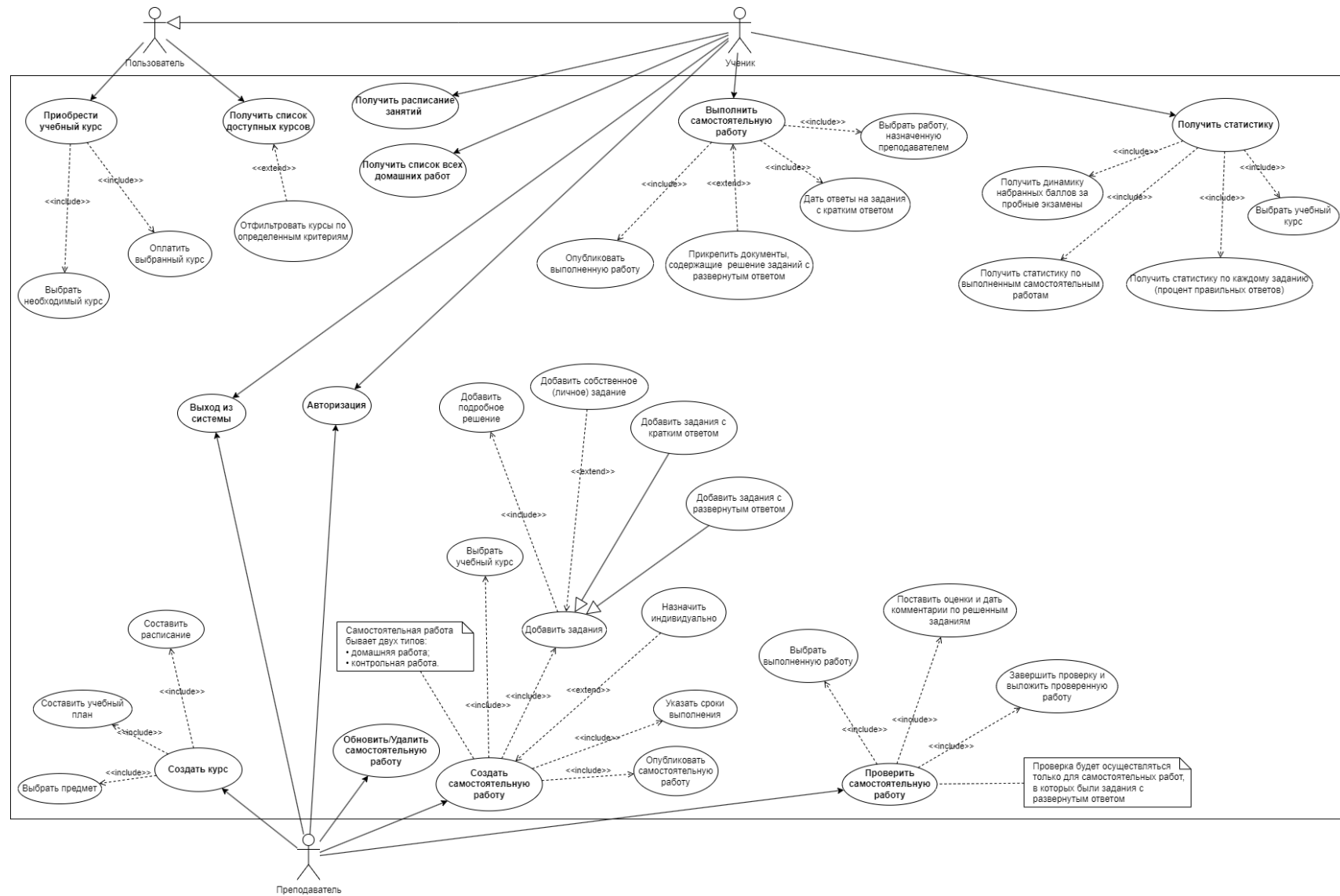


Рисунок Б.1 – Диаграмма вариантов использования

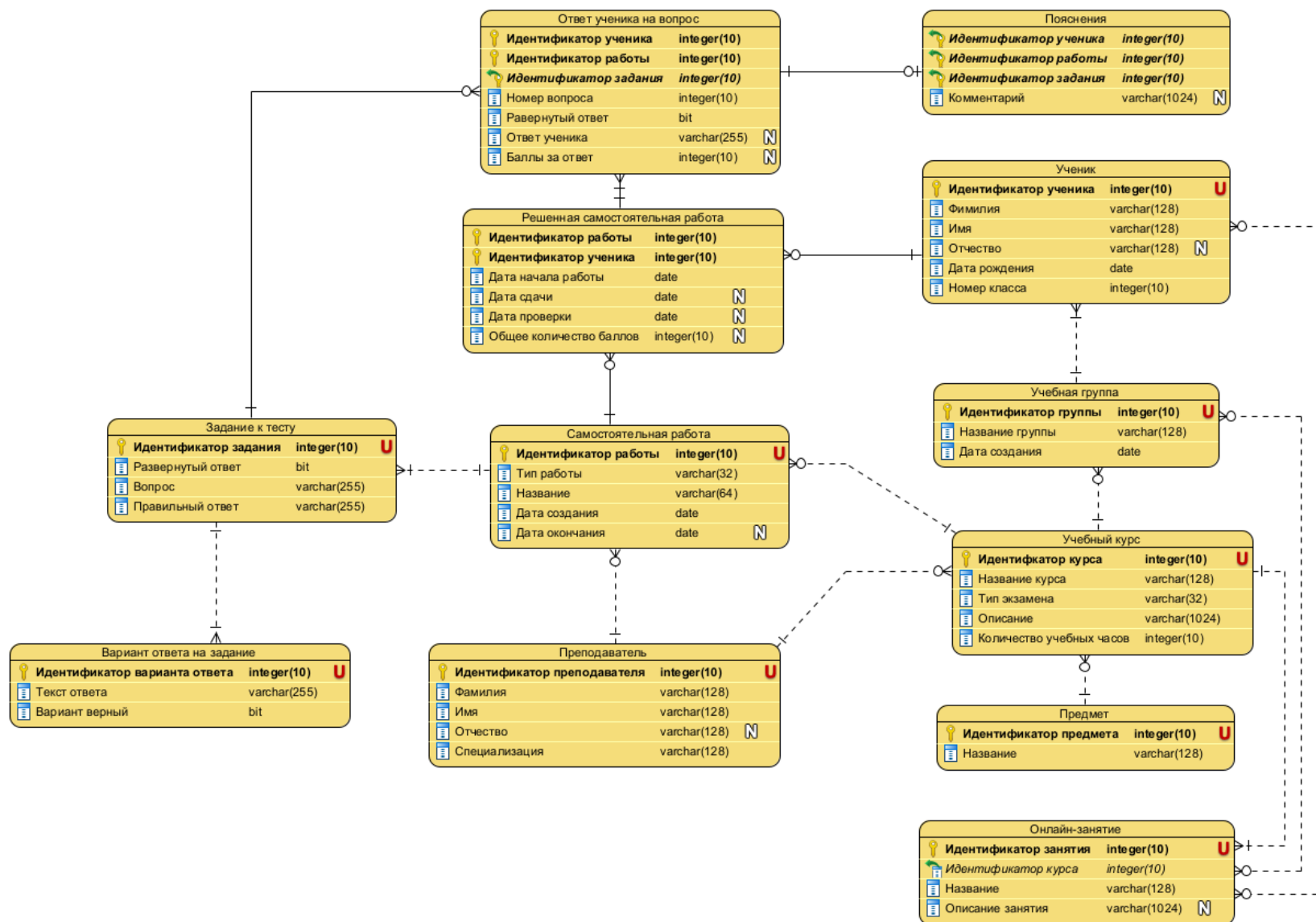


Рисунок Б.2 – Диаграмма в нотации IDEF1X, отображающая логическую модель базы данных

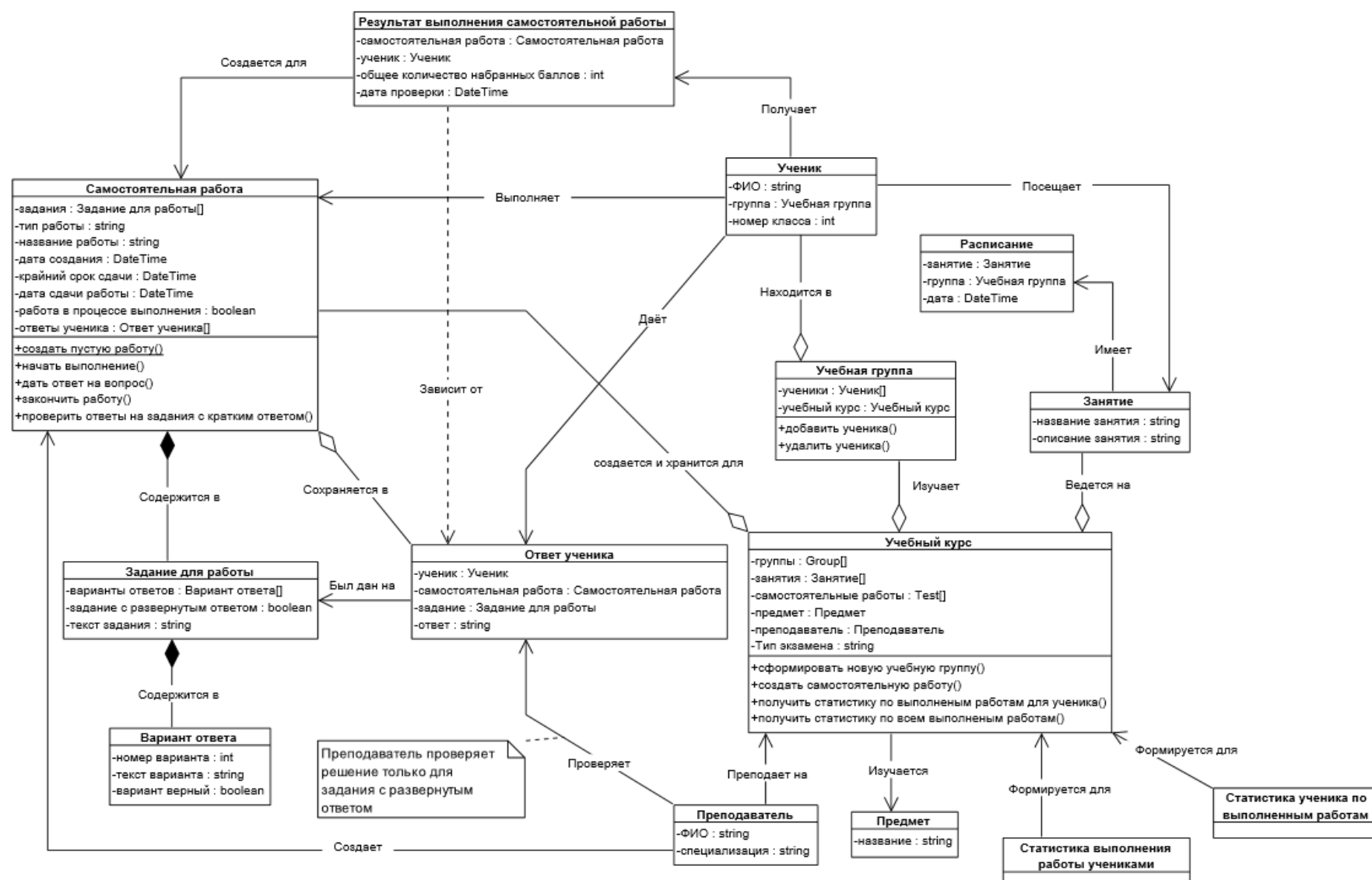


Рисунок Б.3 – Диаграмма классов предметной области

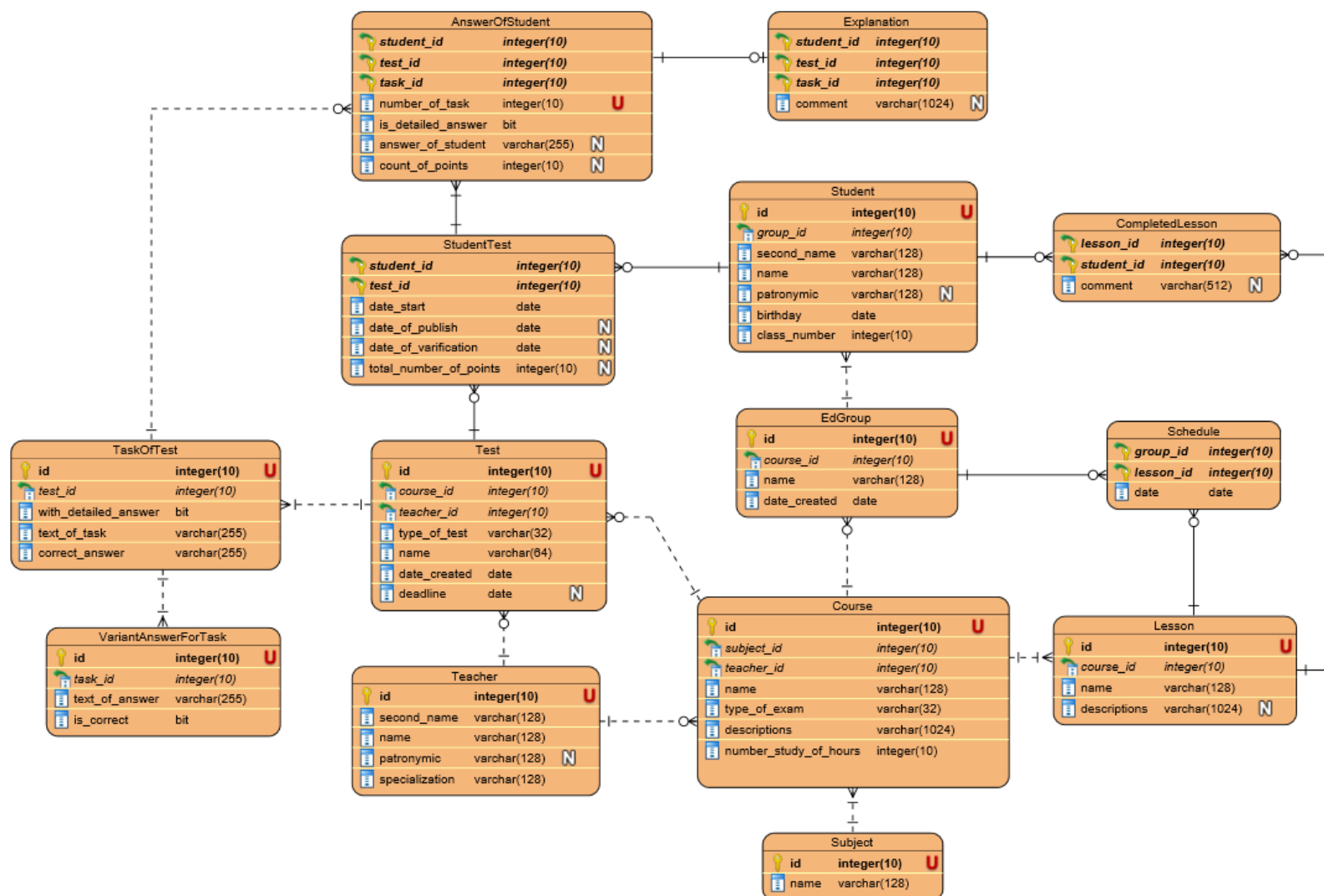


Рисунок Б.4 – Диаграмма в нотации IDEF1X, отображающая физическую модель базы данных

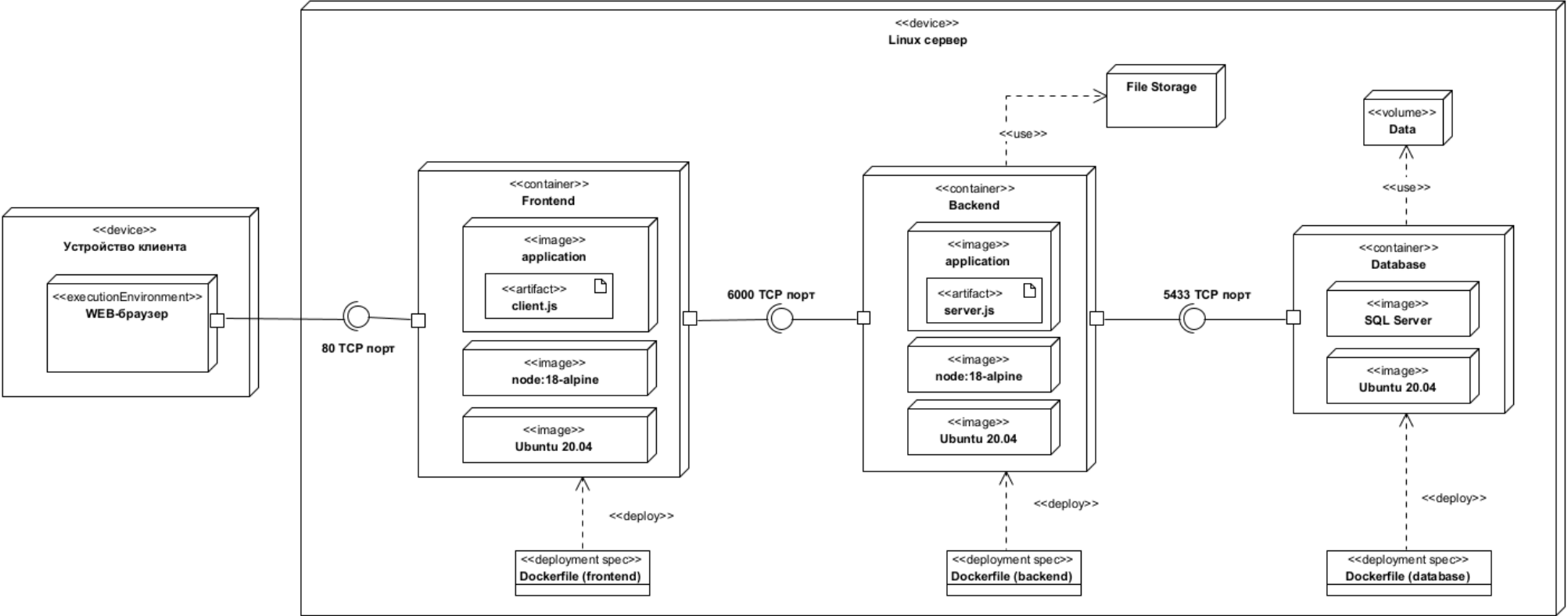


Рисунок Б.5 – Диаграмма развертывания

Приложение В
(рекомендуемое)

Набор атрибутов таблиц базы данных и их характеристики

Таблица Б.1 – Описание атрибутов для таблицы «Course»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
subject_id	Идентификатор предмета	integer	10	FK	Да		
teacher_id	Идентификатор преподавателя	integer	10	FK	Да		
curriculum_id	Идентификатор учебного плана	integer	10	FK	Да		
name	Название курса	Varchar	128		Да		
type_of_exam	Тип экзамена (ОГЭ, ЕГЭ)	Varchar	32		Да		

Таблица Б.2 – Описание атрибутов для таблицы «Subject»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
name	Название предмета	Varchar	128		Да		

Таблица Б.3 – Описание атрибутов для таблицы «Lesson»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
course_id	Идентификатор курса	integer	10	FK	Да		
name	Название занятия	Varchar	128		Да		
description	Описание занятия	Varchar	1024		Да		

Таблица Б.4 – Описание атрибутов для таблицы «EdGroup»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
course_id	Идентификатор курса	integer	10	FK	Да		
name	Название группы	Varchar	128		Да		
date_created	Дата создания	Date	24		Да		

Таблица Б.5 – Описание атрибутов для таблицы «Student»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
group_id	Идентификатор группы	integer	10	FK	Да		
second_name	Фамилия	Varchar	128		Да		
name	Имя	Varchar	128		Да		
patronymic	Отчество	Varchar	128		Нет		
birthday	Дата рождения	Date	24		Да		
class_number	Номер класса	integer	10		Да	[1; 11]	

Таблица Б.6 – Описание атрибутов для таблицы «Teacher»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
second_name	Фамилия	Varchar	128		Да		
name	Имя	Varchar	128		Да		
patronymic	Отчество	Varchar	128		Нет		
specialization	Специализация	Varchar	128		Нет		

Таблица Б.7 – Описание атрибутов для таблицы «Test»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
course_id	Идентификатор курса	integer	10	FK	Да		
teacher_id	Идентификатор преподавателя	integer	10	FK	Да		
type_of_test	Тип самостоятельной работы	Varchar	32		Да		
name	Название работы	Varchar	64		Да		
date_created	Дата создания	Date	24		Да		
deadline	Дата создания	Date	24		Нет		

Таблица Б.8 – Описание атрибутов для таблицы «TaskOfTest»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
test_id	Идентификатор работы	integer	10	FK	Да		
text_of_task	Текст вопроса	Varchar	255		Да		
correct_answer	Правильный ответ	Varchar	255		Да		

Таблица Б.9 – Описание атрибутов для таблицы «VariantAnswerForTask»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
id	Идентификатор	integer	10	PK	Да		
task_id	Идентификатор вопроса	integer	10	FK	Да		
text_of_answer	Текст ответа	Varchar	255		Да		
is_correct	Ответ верный	bit	1		Да		0

Таблица Б.10 – Описание атрибутов для таблицы «Schedule»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
group_id	Идентификатор группы	integer	10	PK, FK	Да		
lesson_id	Идентификатор занятия	integer	10	PK, FK	Да		
date	Дата занятия	Date	24		Да		

Таблица Б.11 – Описание атрибутов для таблицы «CompletedLesson»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
student_id	Идентификатор ученика	integer	10	PK, FK	Да		
lesson_id	Идентификатор занятия	integer	10	PK, FK	Да		
comment	Комментарий	Date	24		Нет		

Таблица Б.12 – Описание атрибутов для таблицы «StudentTest»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
student_id	Идентификатор ученика	integer	10	PK, FK	Да		
test_id	Идентификатор работы	integer	10	PK, FK	Да		
date_of_start	Дата начала	Date	24		Нет		
date_of_publish	Дата отправки	Date	24		Нет		
date_of_varification	Дата проверки	Date	24		Нет		
total_number_of_point	Общий балл за работу	integer	10		Нет		

Таблица Б.13 – Описание атрибутов для таблицы «AnswerOfStudent»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
student_id	Идентификатор ученика	integer	10	PK, FK	Да		
test_id	Идентификатор работы	integer	10	PK, FK	Да		
task_id	Идентификатор вопроса	integer	10	PK, FK	Да		
number_of_taks	Номер вопроса	integer	10		Да		
answer_of_student	Ответ ученика	varchar	255		Нет		
is_detailed_answer	Ответ на вопрос с развернутым ответом	bit	1		Да		0
count_of_points	Балл за ответ	integer	10		Нет		

Таблица Б.14 – Описание атрибутов для таблицы «Explanation»

Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значения по умолчанию
student_id	Идентификатор ученика	integer	10	PK, FK	Да		
test_id	Идентификатор работы	integer	10	PK, FK	Да		
task_id	Идентификатор вопроса	integer	10	PK, FK	Да		
comment	Текст пояснения к решенной самостоятельной работе	Varchar	1024		Да		

Приложение Г
(рекомендуемое)
Листинг серверной части приложения

Файл – course.entity.ts

```
import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm';

@Entity()
export class Course {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  name: string;

  @Column('text')
  description: string;

  @Column()
  subject: string;
}
```

Файл – student.entity.ts

```
import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm';

@Entity()
export class Student {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  name: string;

  @Column()
  email: string;

  @Column()
  grade: number;
}
```

Файл – teacher.entity.ts

```
import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm';

@Entity()
export class Teacher {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  name: string;

  @Column()
  email: string;

  @Column()
  subject: string;
}
```

Файл – test.entity.ts

```
@Entity("Test")
export class Test {
  @PrimaryGeneratedColumn()
  id: number;

  @ManyToOne(() => Teacher, (teacher) => teacher.tests)
  @JoinColumn({ name: "teacher_id" })
  teacher: Teacher;

  @ManyToOne(() => Course, (course) => course.tests)
  @JoinColumn({ name: "course_id" })
  course: Course;

  @OneToMany(() => TaskOfTest, (task) => task.test)
  tasks: TaskOfTest[];

  @OneToMany(() => TestStudent, (joinTable) => joinTable.test)
  testStudent: TestStudent[];

  @Column({ name: "type_of_test" })
  typeOfTest: string;

  name: string;

  @Column({ name: "date_created" })
  dateCreated: Date;

  deadline: Date;
}
```

Файл – test.entity.ts

```
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Course } from '../entities/course.entity';

@Injectable()
export class CoursesService {
  constructor(
    @InjectRepository(Course)
    private coursesRepository: Repository<Course>,
  ) {}

  findAll(): Promise<Course[]> {
    return this.coursesRepository.find();
  }

  findOne(id: number): Promise<Course> {
    return this.coursesRepository.findOneBy({ id });
  }

  create(course: Course): Promise<Course> {
    return this.coursesRepository.save(course);
  }

  async remove(id: number): Promise<void> {
```

```

        await this.coursesRepository.delete(id);
    }
}

```

Файл – test.entity.ts

```

import { Controller, Get, Post, Body, Param, Delete } from
'@nestjs/common';
import { CoursesService } from '../courses.service';
import { Course } from '../entities/course.entity';

@Controller('courses')
export class CoursesController {
  constructor(private readonly coursesService: CoursesService) {}

  @Get()
  findAll(): Promise<Course[]> {
    return this.coursesService.findAll();
  }

  @Get('/:id')
  findOne(@Param('id') id: number): Promise<Course> {
    return this.coursesService.findOne(id);
  }

  @Post()
  create(@Body() course: Course): Promise<Course> {
    return this.coursesService.create(course);
  }

  @Delete('/:id')
  remove(@Param('id') id: number): Promise<void> {
    return this.coursesService.remove(id);
  }
}

```

Файл – students.service.ts

```

import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Student } from '../entities/student.entity';

@Injectable()
export class StudentsService {
  constructor(
    @InjectRepository(Student)
    private studentsRepository: Repository<Student>,
  ) {}

  findAll(): Promise<Student[]> {
    return this.studentsRepository.find();
  }

  findOne(id: number): Promise<Student> {
    return this.studentsRepository.findOneBy({ id });
  }

  create(student: Student): Promise<Student> {

```

```

    return this.studentsRepository.save(student);
  }

  async remove(id: number): Promise<void> {
    await this.studentsRepository.delete(id);
  }
}

```

Файл – students.service.ts

```

import { Controller, Get, Post, Body, Param, Delete } from
'@nestjs/common';
import { StudentsService } from './students.service';
import { Student } from './entities/student.entity';

@Controller('students')
export class StudentsController {
  constructor(private readonly studentsService: StudentsService) {}

  @Get()
  findAll(): Promise<Student[]> {
    return this.studentsService.findAll();
  }

  @Get('/:id')
  findOne(@Param('id') id: number): Promise<Student> {
    return this.studentsService.findOne(id);
  }

  @Post()
  create(@Body() student: Student): Promise<Student> {
    return this.studentsService.create(student);
  }

  @Delete('/:id')
  remove(@Param('id') id: number): Promise<void> {
    return this.studentsService.remove(id);
  }
}

```

Файл – app.module.ts

```

import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { CoursesModule } from './courses/courses.module';
import { StudentsModule } from './students/students.module';
import { TeachersModule } from './teachers/teachers.module';
import { AdminsModule } from './admins/admins.module';
import { TestsModule } from './tests/tests.module';
import { Course } from './courses/entities/course.entity';
import { Student } from './students/entities/student.entity';
import { Teacher } from './teachers/entities/teacher.entity';
import { Admin } from './admins/entities/admin.entity';
import { Test } from './tests/entities/test.entity';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: "mssql",

```

```
        host:      process.env.DATABASE_HOST,
        port:      process.env.DATABASE_PORT,
        username: process.env.DATABASE_USERNAME,
        password: process.env.DATABASE_PASSWORD,
        database: process.env.DATABASE_DATABASE,
        entities: ["**/**/*entity.ts"],
        migrations: ["src/migrations/*.ts"],
    }
    CoursesModule,
    StudentsModule,
    TeachersModule,
    AdminsModule,
    TestsModule,
  ],
}))
export class AppModule {}
```

Приложение Д
(рекомендуемое)
Листинг клиентской части приложения

Файл ./app/layout.tsx

```

import "../globals.css";
import type { Metadata } from "next";
import { AppRouterCacheProvider } from "@mui/material-nextjs/v14-appRouter";
import { ThemeProvider } from "@mui/material/styles";
import CssBaseline from "@mui/material/CssBaseline";
import theme from "../theme";
import { Box, Container, Stack, Link, Typography } from "@mui/material";
import Header from "@components/header";
import Sidebar from "@components/sidebar";

export const metadata: Metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang='en'>
      <body>
        <AppRouterCacheProvider options={{ enableCssLayer: true }}>
          <ThemeProvider theme={theme}>
            <CssBaseline />
            <Container className='h-screen max-w-[1600px]'>
              <Header />
              <Box className='flex gap-2 h-[calc(100vh-5rem)]'>
                <Box className='w-1/5 bg-red-50 py-1 px-2'>
                  <Sidebar />
                </Box>
                <Box className='w-4/5 bg-green-100/70 py-1 px-2'>{children}</Box>
              </Box>
            </Container>
          </ThemeProvider>
        </AppRouterCacheProvider>
      </body>
    </html>
  );
}

```

Файл ./app/clients/page.tsx

```

import { getAllClients } from "@actions/db-clients";
import * as utils from "@utils";
import {
  Box,
  Paper,
  Table,
  TableBody,
  TableCell,

```

```

    TableContainer,
    TableHead,
    TableRow,
    Typography,
  } from "@mui/material";

export default async function ClientsPage() {
  const clients = await getAllClients();

  return (
    <Box className='p-1 h-full w-full overflow-auto'>
      <Box>
        <Typography variant='h6' className='mb-3'>
          Клиенты (общее количество: <b>{clients.length}</b> )
        </Typography>
        <TableContainer component={Paper} elevation={5} className='w-
max'>
          <Table>
            <TableHead>
              <TableRow>
                <TableCell align='right'>ID</TableCell>
                <TableCell align='right'>Фамилия</TableCell>
                <TableCell align='right'>Имя</TableCell>
                <TableCell align='right'>Отчество</TableCell>
                <TableCell align='right'>Дата рождения</TableCell>
                <TableCell align='right'>Почта</TableCell>
                <TableCell align='right'>Телефон</TableCell>
              </TableRow>
            </TableHead>
            <TableBody>
              {clients.map((client) => (
                <TableRow key={client.id} className='odd:bg-gray-100'>
                  <TableCell align='right'>{client.id}</TableCell>
                  <TableCell
align='right'>{client.second_name}</TableCell>
                  <TableCell align='right'>{client.name}</TableCell>
                  <TableCell
align='right'>{client.patronymic}</TableCell>
                  <TableCell
align='right'>{utils.formatDate(client.birthday)}</TableCell>
                  <TableCell align='right'>{client.email}</TableCell>
                  <TableCell align='right'>{client.phone}</TableCell>
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </TableContainer>
      </Box>
    </Box>
  );
}

```

Файл ./app/tours/page.tsx

```

import { Box } from "@mui/material";
import ToursTable from "../ToursTable";
import { getAllTours } from "@actions/db-tours";

```

```
export default async function ToursPage() {
  const tours = await getAllTours();

  return (
    <Box className='h-full w-full overflow-auto'>
      <ToursTable tours={tours} />
    </Box>
  );
}
```

Файл ./app/tours/pageTable.tsx

```
"use client";

import * as utils from "@/utils";
import { clsx } from "clsx";
import {
  Box,
  Button,
  Link,
  Paper,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
} from "@mui/material";
import { TourEntity } from "@/types/tour";
import { useState } from "react";
import TourInfoModal from "../TourInfoModal";
import TourFilter from "../TourFilter";
import NextLink from "next/link";

interface ToursTableProps {
  tours: TourEntity[];
}

export default function ToursTable({ tours }: ToursTableProps) {
  const [toursState, setTours] = useState(tours);
  const [modalIsOpen, setModalIsOpen] = useState<boolean>(false);
  const [selectedTourId, setSelectedTourId] = useState<number>();

  const handleClose = () => setModalIsOpen(false);
  const handleOpen = () => setModalIsOpen(true);

  const handleDoubleClickOnRow = (tourId: number) => {
    setSelectedTourId(tourId);
    handleOpen();
  };

  return (
    <Box className='overflow-auto mr-1'>
      <TourInfoModal tourId={selectedTourId} open={modalIsOpen}
        handleClose={handleClose} />
      <Box className='mb-4 py-2 flex gap-20'>
```

```

<TourFilter setTours={setTours} />
<Link component={NextLink} href={` /tours/create`} >
  <Button variant='contained'>Добавить новый тип</Button>
</Link>
<Link component={NextLink} href={` /tourStats`} >
  <Button variant='contained' size='small' className='text-
[0.7rem] w-[12rem]'>
    Получить информация по турам для каждой страны
  </Button>
</Link>
</Box>
<TableContainer component={Paper} elevation={5} className='w-
max'>
  <Table>
    <TableHead>
      <TableRow>
        <TableCell align='right'>ID</TableCell>
        <TableCell align='right'>Страна</TableCell>
        <TableCell align='right'>ID Отеля</TableCell>
        <TableCell align='right'>ID типа номера</TableCell>
        <TableCell align='right'>ID типа питания</TableCell>
        <TableCell align='right'>Цена покупки</TableCell>
        <TableCell align='right'>Цена продажи</TableCell>
        <TableCell align='right'>Дата начала</TableCell>
        <TableCell align='right'>Дата окончания</TableCell>
        <TableCell align='right'>Доступность</TableCell>
      </TableRow>
    </TableHead>
    <TableBody>
      {toursState.map((tour) => (
        <TableRow
          key={tour.id}
          onClick={() => handleDoubleClickOnRow(tour.id)}
          className={clsx(
            "select-none odd:bg-gray-100 hover:cursor-pointer
            hover:bg-blue-200 active:bg-blue-300",
            {
              "bg-red-100 odd:bg-red-200/85": !tour.isActive,
            }
          )}
        >
          <TableCell align='right'>{tour.id}</TableCell>
          <TableCell align='right'>{tour.country}</TableCell>
          <TableCell align='right'>{tour.hotel_id}</TableCell>
          <TableCell
            align='right'>{tour.type_room_id}</TableCell>
          <TableCell
            align='right'>{tour.type_food_id}</TableCell>
          <TableCell align='right'>{`$${tour.price_purchase}
₽`}</TableCell>
          <TableCell align='right'>{`$${tour.price_sale}
₽`}</TableCell>
          <TableCell
            align='right'>{utils.formatDate(tour.date_start)}</TableCell>
          <TableCell
            align='right'>{utils.formatDate(tour.date_end)}</TableCell>
          <TableCell align='right'>{tour.isActive ? "✔" :
"✗"}</TableCell>

```

```
        </TableRow>
      )))
    </TableBody>
  </Table>
</TableContainer>
</Box>
);
}
```

Приложение Е
(рекомендуемое)
Листинг автоматизированных тестов

```

import requests
from faker import Faker

BASE_URL = "http://localhost:6000/api"
ROLE_STUDENT = "STUDENT"
ROLE_TEACHER = "TEACHER"
fake = Faker()

def register_user(username, password, role):
    response = requests.post(f"{BASE_URL}/auth/register", json={
        "username": username,
        "password": password,
        "role": role
    })
    return response

def login_user(username, password):
    response = requests.post(f"{BASE_URL}/auth/login", json={
        "username": username,
        "password": password
    })
    return response

def test_registration():
    reg_response = register_user("student1", "password123",
ROLE_STUDENT)
    assert reg_response.status_code == 201

def test_login():
    login_response = login_user("student1", "password123")
    assert login_response.status_code == 200

    token = login_response.json().get("token")
    assert token is not None

def test_create_course():
    reg_response = register_user("teacher1", "password123",
ROLE_TEACHER)
    assert reg_response.status_code == 201

    login_response = login_user("teacher1", "password123")
    assert login_response.status_code == 200

    token = login_response.json().get("token")
    assert token is not None

    name = fake.pystr(min_chars=5, max_chars=10)
    description = "A course on advanced mathematics"

    response = requests.post(f"{BASE_URL}/courses", json={
        "name": name,
        "description": description
    }, headers={"Authorization": f"Bearer {token}"})
    assert response.status_code == 201

    course = response.json()
    assert course.get("name") == name
    assert course.get("description") == description

```

```

def test_enroll_in_course():
    login_response = login_user("teacher1", "password123")
    token = login_response.json().get("token")
    assert token is not None

    course_id = 1
    enroll_response =
requests.post(f"{BASE_URL}/courses/{course_id}/enroll", headers={
    "Authorization": f"Bearer {token}"
})
    assert enroll_response.status_code == 200

def test_add_lesson():
    # Login as teacher to create a course
    teacher_login_response = login_user("teacher1", "password123")
    teacher_token = teacher_login_response.json().get("token")
    assert teacher_token is not None

    name = faker.pystr()

    course_id = 1
    lesson_response =
requests.post(f"{BASE_URL}/courses/{course_id}/lessons", json={
    "name": name,
    "description": "Basics of organic chemistry",
    "date": "2024-06-21",
    "time": "10:00"
}, headers={"Authorization": f"Bearer {teacher_token}"})
    assert lesson_response.status_code == 201
    lesson = lesson_response.json()
    assert lesson.get("name") == name

def test_registration_invalid_data():
    # Missing password
    response = register_user("student1", "", ROLE_STUDENT)
    assert response.status_code == 400

    # Invalid role
    response = register_user("student2", "password123",
"invalid_role")
    assert response.status_code == 400

    # Username too short
    response = register_user("st", "password123", ROLE_STUDENT)
    assert response.status_code == 400

def test_login_invalid_data():
    # Incorrect password
    response = login_user("student1", "wrongpassword")
    assert response.status_code == 401

    # Non-existent user
    response = login_user("nonexistent", "password123")
    assert response.status_code == 401

def test_create_course_invalid_data():
    # Register and login a teacher

```

```

login_response = login_user("teacher1", "password123")
token = login_response.json().get("token")
assert token is not None

# Missing name
response = requests.post(f"{BASE_URL}/courses", json={
    "name": "",
    "description": "A course on advanced mathematics",
}, headers={"Authorization": f"Bearer {token}"})
assert response.status_code == 400

def test_enroll_in_course_invalid_data():
    # login a student
    login_response = login_user("student1", "password123")
    token = login_response.json().get("token")
    assert token is not None

    # Enroll in a non-existent course
    course_id = -149
    enroll_response =
requests.post(f"{BASE_URL}/courses/{course_id}/enroll", headers={
    "Authorization": f"Bearer {token}"
})
assert enroll_response.status_code == 404

def test_add_lesson_invalid_data():
    # Login as teacher to create a course
    teacher_login_response = login_user("teacher1", "password123")
    teacher_token = teacher_login_response.json().get("token")
    assert teacher_token is not None

    course_id = 1
    # Add a lesson with missing name
    lesson_response =
requests.post(f"{BASE_URL}/courses/{course_id}/lessons", json={
    "name": "",
    "description": "Basics of organic chemistry",
    "date": "2024-06-21",
    "time": "10:00"
}, headers={"Authorization": f"Bearer {teacher_token}"})
assert lesson_response.status_code == 400

    # Add a lesson with invalid time format
    lesson_response =
requests.post(f"{BASE_URL}/courses/{course_id}/lessons", json={
    "title": "Introduction to Organic Chemistry",
    "description": "Basics of organic chemistry",
    "date": "invalid_date",
    "time": "34:87"
}, headers={"Authorization": f"Bearer {teacher_token}"})
assert lesson_response.status_code == 400

```

Приложение Ж
(рекомендуемое)
Результаты тестирования

Intellectum
онлайн школа

Регистрация

Имя: иНк5
Имя должно состоять только из букв

Фамилия: иНк5
Фамилия должна состоять только из букв

Отчество: иНк5
Отчество должно состоять только из букв

Класс: 9

Тип экзамена: ОГЭ
ОГЭ
ЕГЭ

Email: ivan.ivanov2008@mail.ru

Пароль:

Подтверждение пароля:

Пароли не совпадают

[уже есть учетная запись?](#)

Создать аккаунт

Рисунок Ж.1 – Некорректно заполненная форма регистрации

Intellectum
онлайн школа

Вход в аккаунт

Email: ivan.petrov97@mail.ru

Пароль:

[нет учетной записи?](#)

Войти

Ошибка аутентификации
Неверный email и/или пароль

Рисунок Ж.2 – Ошибка аутентификации

Intellectum онлайн школа

Учебные курсы | Мои курсы | Расписание

Иванов И.И.

Полноценный курс по подготовке к ОГЭ по русскому языку (8 мес.)

60 занятий (90 часов) | Изложение + тестовая часть + сочинение

16 контрольных работ

29 990 ₽

Купить | Посмотреть подробности

Курс по подготовке к ОГЭ по физике

30 занятий (45 часов) | Тестовая часть + лабораторная работа (№ 23)

10 контрольных работ

17 990 ₽

Купить | Посмотреть подробности

Фильтры

Мин. цена: 4 990 ₽ | Макс. цена: 44 990 ₽

☒ ОГЭ ☐ ЕГЭ

☐ Математика
☒ Русский язык
☒ Физика
☐ Информатика
☐ Обществознание
☐ Биология
☐ Химия
☐ Английский язык
☐ География
☐ Литература
☐ История

Применить

Рисунок Ж.3 – Отфильтрованный список учебных курсов

Intellectum онлайн школа

Учебные курсы | Мои курсы | Расписание

Иванов И.И.

Уведомления

ДЗ Проверено 17.10.23

Курс: Полноценный курс по подготовке к ОГЭ по русскому языку (8 мес.)

Работа: ДЗ. Приставки -ПРИ, -ПРЕ

Посмотреть результат

Рисунок Ж.4 – Уведомление после проверки самостоятельной работы


Intellectum
онлайн школа

Учебные курсы | Управление моими курсами


Смирнов С.В.

Курс по подготовке к ОГЭ по физике (ID: 23)

Самостоятельные работы [Добавить](#)

Тип работы: Домашнее задание (ДЗ) Название: Решение задач по термодинамике Дата создания: 10.10.2023 Дата сдачи: 15.10.2023	Завершено	Тип работы: Домашнее задание (ДЗ) Название: Электрические цепи. Закон Ома Дата создания: 17.10.2023 Дата сдачи: 23.10.2023	Активно
Тип работы: Пробный экзамен (ПЭ) Название: Пробный экзамен. Октябрь Дата создания: 25.10.2023 Дата сдачи: 31.10.2023	Активно	Тип работы: Домашнее задание (ДЗ) Название: Расчет цепей методом Киргхова Дата создания: 08.11.2023 Дата сдачи: 15.11.2023	Выложить Редактировать

Рисунок Ж.5 – Автоматическая пометка завершенной самостоятельной работы