МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет «Информационных систем и технологий»
Кафедра «Интеллектуальных систем автоматизации и управления»

Направление подготовки: 09.03.02 - Информационные системы и технологии

Направленность (профиль): Прикладные информационные системы и технологии

ЛАБОРАТОРНЫЕ РАБОТЫ № 1-4

по дисциплине:

Объектно-ориентированное проектирование автоматизированных систем управления

на тему:

Проектирование и разработка автоматизированной системы управления электронным журналом

Бригада № 1

Выполнили студенты группы ИБ-11вп
Иванова Е.С., Махоткин А.П.,
Соболева К.С., Филиппова А.О.
Фамилия И. О.

Руководитель — ассистент кафедры ИСАУ
оценка — *уч. степень, уч. звание*

Шабанов А.П.
дата, подпись — *Фамилия И. О.*

Санкт-Петербург
2024

Содержание

Цель работы

Целью работы является проектирование и разработка продукта на основании предметной области «Электронный журнал в высшем учебном заведении» с учетом его использования разными типами пользователей (студенты, преподаватели, старосты).

Постановка задачи

— проанализировать и описать предметную область для электронного журнала в высшем учебном заведении;

— описать видение системы электронного журнала, в т. ч. позиционирование, описание, возможности и требования к продукту;

— разработать концептуальную модель электронного журнала;

— разработать диаграмму вариантов использования системы электронного журнала и описать прецеденты;

— разработать диаграмму классов и их спецификацию в виде таблиц (описание скалярных свойств, свойств навигации и свойств, содержащих внешние ключи);

— создать программное решение в среде разработки Visual Studio:

- в соответствии со спецификацией классов программно реализовать доменную модель предметной области;
- реализовать взаимодействие с сущностями в соответствии с шаблоном проектирования "Репозиторий";
- собрать автоматическую миграцию классов в базу данных при помощи инструмента Entity Core Framework;
- реализовать программный интерфейс приложения;
- реализовать веб-оболочку приложения при помощи инструмента Blazor Web Assembly (или Blazor .NET).

Проектирование системы

1 Анализ предметной области

На сегодняшний день автоматизация управления документами, включая электронные журналы для высших учебных заведений, находится в стадии значительного развития. Учебные учреждения все чаще переходят на электронные платформы для более эффективного учета и хранения информации о занятиях. Отмечается постоянное нарастание тенденции использования современных технологий и инструментов для создания и улучшения журналов в учебной среде.

Электронный журнал используется для получения данных об образовательном процессе в высшем учебном заведении.

В соответствии с правилами внутреннего учебного распорядка установлено расписание занятий:

Пары занятий	**Время проведения пары занятий**
Первая	9:00 – 10:35
Вторая	10:45 – 12:20
Третья	13:00 – 14:35
Четвёртая	14:45 – 16:20
Пятая	16:30 – 18:05
Шестая	18:25 – 19:50
Седьмая	20:00 – 21:35

Рисунок 1. Расписание занятий

Основные информационные поля журнала – поля для заполнения преподавателем оценок либо отметок о непосещении занятий (по уважительной причине/без уважительной причины). По умолчанию поля заполнены меткой «+», что при закрытии ведомости будет интерпретировано как присутствие студента на занятии без получения оценки.

Пользователями электронного журнала могут быть преподаватель, студент и староста:

— Студент. Про него известны личные данные: фамилия, имя, отчество, дата рождения, номер студенческого билет; принадлежит

определенной группе со своей формой обучениям конкретной специальностью. У него есть возможность просматривать свои личные данные. Студент в своем журнале может видеть отметки о посещении/непосещении занятий, либо полученные оценки.

— Преподаватель. О преподавателе известны личные данные: фамилия, имя, отчество, а также его должность. При работе с журналом преподаватель выставляет отметки студентам и отмечает их присутствие на занятии, а также указывает темы проводимых занятий.

— Староста группы имеет доступ к той же информации в электронном журнале, что и остальные студенты. Помимо этого, старосте доступен просмотр журнала, в котором можно ознакомиться с посещаемостью и оценками всей группы в разрезе предмета.

2 Видение системы

Разрабатываемый продукт – электронный журнал, включающий в себя функции, набор которых различен для каждого из трех типов пользователей: преподавателя, студента и старосты.

Функционал включает в себя внесение и просмотр информации о посещаемости и оценках студентов, тем проводимых занятий. Вносить информацию может только преподаватель, просматривать информацию о себе может студент, старосте доступна функция просмотра оценок и посещаемости всей группы в рамках семестра по конкретным предметам.

В проекте должна быть предусмотрена возможность ведения одного предмета у группы разными преподавателями в зависимости от типа занятия (например, лекционные и практические занятия). Помимо этого, необходимо добавить для старосты не только функционал студента (просмотр информации о себе), но и возможность просмотра журнала по одной конкретной группе. Указывать темы занятия, проставлять отсутствие и оценки может только преподаватель.

3 Концептуальная модель предметной области

Концептуальная модель представляет собой описании структурных элементов и концептуальных ограничений в выбранной предметной области. На Рисунке 2 представлена концептуальная модель электронного журнала, основными структурными элементами которой являются: студент, группа, дисциплина, занятие и преподаватель. Также учтены атрибуты и взаимосвязи каждого структурного элемента.

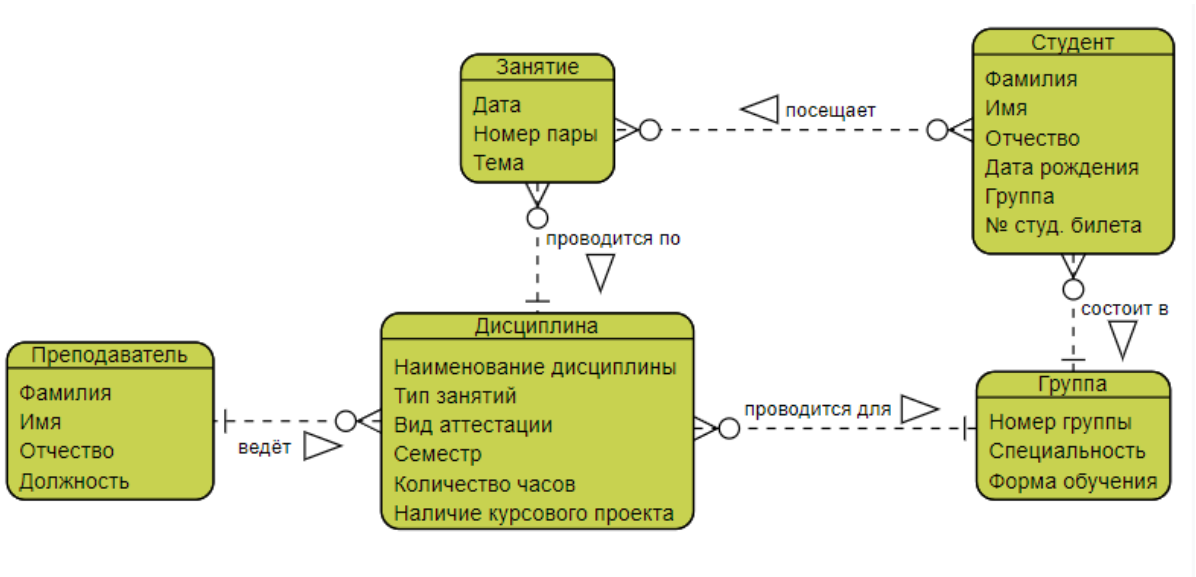

Рисунок 2. Концептуальная модель электронного журнала

Основными сущностями предметной области являются:

— преподаватель;

— студент;

— группа;

— занятие;

— дисциплина.

Сущность «Преподаватель» характеризуется атрибутами «ФИО» и «должность».

Сущность «Студент» имеет набор атрибутов:

— ФИО студента;

— дата рождения;

— номер студенческого билета.

Сущность «Группа» имеет набор атрибутов:

— номер группы;

— специальность, по которой обучаются студенты;

— форма обучения - очное, заочное, очно-заочное.

Сущность «Занятие» имеет базовый набор атрибутов:

— дата занятия;

— тема занятия;

— номер пары, на которой проводится занятие.

Сущность «Дисциплина» содержит следующую информацию:

— наименование дисциплины;

— тип занятий - лекция, практика, экзамен, зачёт и т.д.;

— вид аттестации - зачёт с оценкой, зачёт без оценки, экзамен;

— порядковый номер семестра, в котором проводится дисциплина;

— количество часов, отведенных на текущую дисциплину в текущем семестре для группы студентов;

— наличие или отсутствие курсового проекта по дисциплине.

Описание связей приведено ниже.

Связь «Студент» – «Группа»: студенты состоят в группе. Один студент состоит в одной группе, но в одной группе много студентов.

Связь «Студент» – «Занятие»: студент посещает занятие. Сущность необходима для учёта посещаемости и отметок студентов в разрезе занятия.

Связь «Преподаватель» – «Дисциплина»: преподаватель ведёт дисциплину. Один преподаватель может вести одну конкретную дисциплину, но по дисциплине может быть несколько преподавателей.

Связь «Дисциплина» – «Группа»: дисциплина ведётся для определенной группы. Одна дисциплина ведется у одной группы, но у одной группы может быть много дисциплин.

Связь «Дисциплина» – «Занятие»: занятие проводится по определённой дисциплине. Занятие может быть по одной дисциплине, но по одной дисциплине может быть много занятий.

4 Диаграмма вариантов использования системы

На основании концептуальной модели (рис. 2) спроектирована диаграмма вариантов использования системы (рис. 3), в которой рассмотрено взаимодействие системы с пользователями. Данная диаграмма используется для выявления требований по отношению к системе, а также для иллюстрации функций, доступных разным типам пользователей.

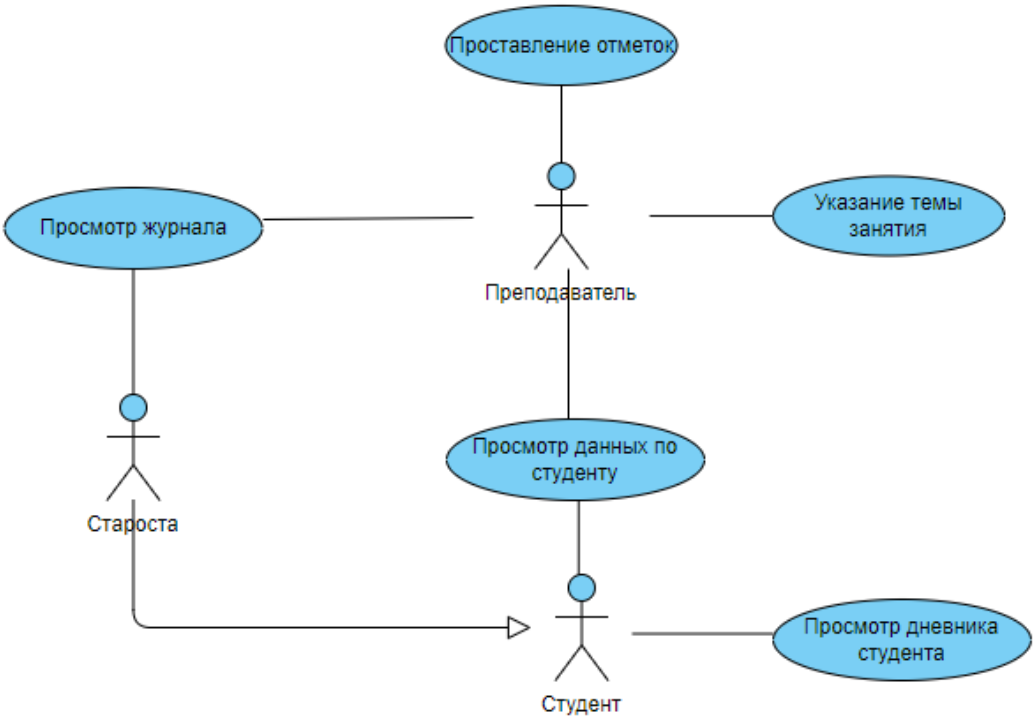

Рисунок 3. Диаграмма вариантов использования электронного журнала

5 Описание прецедентов

Ниже представлены таблицы с описанием прецедентов в рамках электронного журнала.

Таблица 1. Прецедент «Просмотр дневника студента»

Имя прецедента	Просмотр дневника студента
Действующие лица	Студент, староста
Предусловие	Пользователь нажимает на вкладку «Дневник».
Постусловие	Открыт перечень всех занятий и отметок по ним.
Основной сценарий	Студент, староста авторизируются в электронном журнале при помощи логина и пароля, переходят во вкладку «Дневник», где представлена информация по занятиям.

Таблица 2. Прецедент «Просмотр данных по студенту»

Имя прецедента	Просмотр данных по студенту
Действующие лица	Студент, староста
Предусловие	Пользователь нажимает на вкладку «Личные данные».
Постусловие	Открыт раздел с личной информацией о студенте.
Основной сценарий	Студент, староста авторизируются в электронном журнале при помощи логина и пароля, переходят во вкладку «Личные данные», где представлена информация о студенте: фамилия, имя, отчество, дата рождения, группа, специальность, номер студенческого билета.

Таблица 3. Прецедент «Просмотр журнала»

Имя прецедента	Просмотр журнала
Действующие лица	Преподаватель, староста
Предусловие	Пользователь нажимает на вкладку «Журнал занятий».
Постусловие	Открыт перечень занятий по предмету с оценками и посещениями студентов.
Основной сценарий	Преподаватель, староста авторизируются в электронном журнале при помощи логина и пароля, переходят во вкладку «Журнал занятий», где представлена информация по посещаемости и успеваемости студентов по предмету.

Таблица 4. Прецедент «Проставление отметок»

Имя прецедента	Проставление отметок
Действующие лица	Преподаватель
Предусловие	Пользователь нажимает на вкладку «Журнал занятий».
Постусловие	Студенту поставлена оценка по предмету или отмечено отсутствие на занятии.
Основной сценарий	Преподаватель авторизируется в электронном журнале при помощи логина и пароля, переходит во вкладку «Журнал занятий», выбирает предмет и группу. В появившемся списке выбирает студента и ставит оценку либо отметку о непосещении.

Таблица 5. Прецедент «Указание темы занятия»

Имя прецедента	Указание темы занятия
Действующие лица	Преподаватель
Предусловие	Пользователь нажимает на вкладку «Журнал занятий»,
Постусловие	Тема занятия указана в журнале.
Основной сценарий	Преподаватель авторизируется в электронном журнале при помощи логина и пароля, переходит во вкладку «Журнал занятий», выбирает занятие и нажимает кнопку «Указать тему». В появившемся поле указывает тему.

6 Диаграмма классов

В отличие от концептуальной модели (рис. 2) диаграмма классов представляет программные классы, на которые отображаются сущности концептуальной модели. UML-диаграмма классов представлена на рисунке 4.

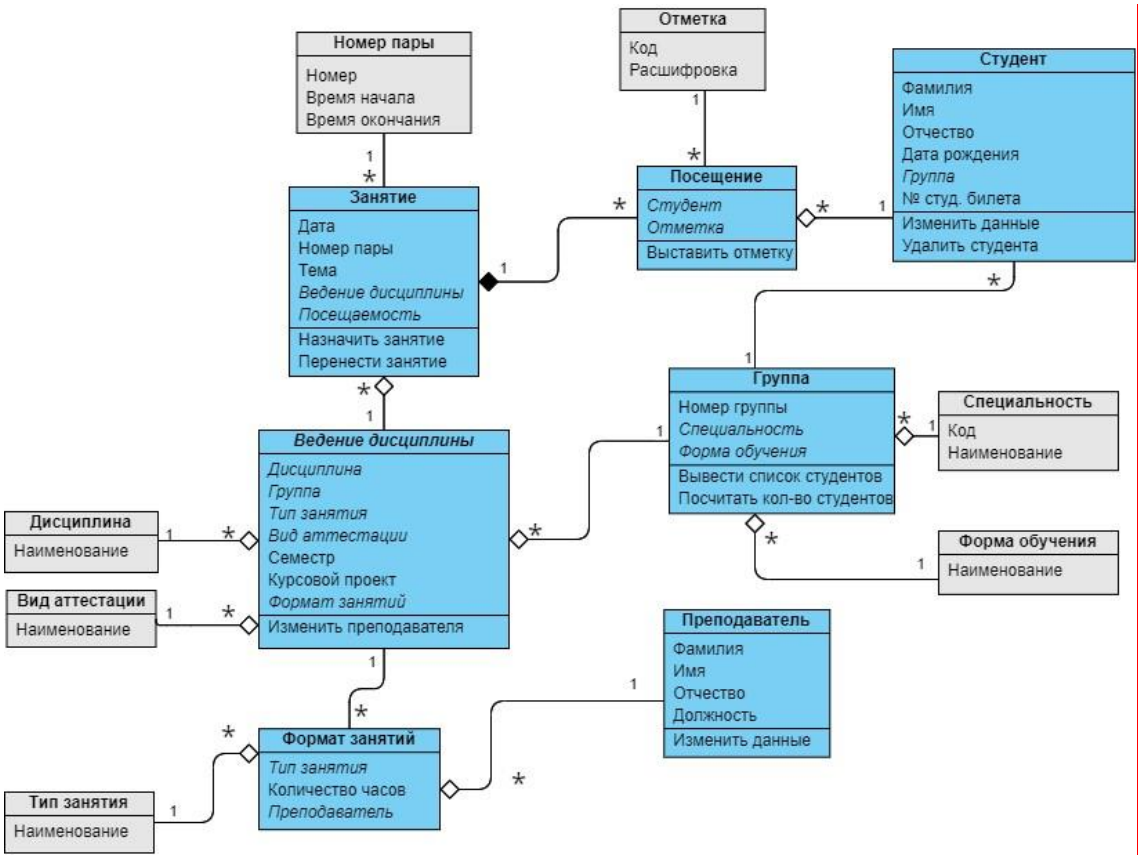

Рисунок 4. Диаграмма классов

7 Спецификация классов

Спецификация классов является основой для отображения проекта классов в исходный код на C#. В таблицах 6-19 представлены спецификации классов, используемых в процессе разработки автоматизированный системы управления электронным журналом.

Таблица 6. Класс «Специальность»

Specialization		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Code	string	Код специальности
Name	string	Наименование специальности

Таблица 7. Класс «Форма обучения»

EducationalForm		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Name	string	Наименование формы обучения

Таблица 8. Класс «Группа»

Group		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Number	string	Номер группы
Свойства навигации		
Имя	Кратность	Комментарии
Specializaton	1..1	Связь со специальностью
EducationalForm	1..1	Связь с формой обучения

Таблица 9. Класс «Отметка»

Mark		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Code	char	Отметка (Н/+/оценка)
Description	string	Пояснение к отметке

Таблица 10. Класс «Студент»

Student		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
FirstName	string	Имя
LastName	string	Фамилия
Patronymic	string	Отчество
BirthDate	DateOnly	Дата рождения
IdCard	int	Номер студенческого билета
Свойства навигации		
Имя	Кратность	Комментарии
Group	1..1	Связь с группой

Таблица 11. Класс «Посещение»

Attend		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Свойства навигации		
Имя	Кратность	Комментарии
Student	1..1	Связь со студентом
Mark	1..1	Связь с отметкой

Таблица 12. Класс «Вид аттестации»

AttestationType		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Name	string	Наименование вида аттестации

Таблица 13. Класс «Тип занятия»

LessonType		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Name	string	Наименование типа занятия

Таблица 14. Класс «Номер пары»

Schedule		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Number	int	Номер пары
StartTime	TimeOnly	Время начала
EndTime	TimeOnly	Время окончания

Таблица 15. Класс «Дисциплина»

Subject		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Name	string	Наименование дисциплины

Таблица 16. Класс «Преподаватель»

Teacher		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
FirstName	string	Имя
LastName	string	Фамилия
Patronymic	string	Отчество
Post	string	Должность преподавателя

Таблица 17. Класс «Формат занятий»

LessonForm		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Hours	int	Количество часов формата занятий
Свойства навигации		
Имя	Кратность	Комментарии
Teacher	1..1	Связь с преподавателем
LessonType	1..1	Связь с типом занятия

Таблица 18. Класс «Ведение дисциплины»

Teaching		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Term	int	Семестр
TermProject	bool	Наличие курсового проекта
Свойства навигации		
Имя	Кратность	Комментарии
Subject	1..1	Связь с дисциплиной
Group	1..1	Связь с группой
LessonType	1..1	Связь с типом занятий
AttestationType	1..1	Связь с видом аттестации
LessonForm	1..1	Связь с форматом занятия

Таблица 19. Класс «Занятие»

Lesson		
Скалярные свойства		
Имя	Тип	Комментарии
Id	Guid	Глобально уникальный идентификатор
Topic	string	Тема занятия
Date	DateOnly	Дата занятия
Свойства навигации		
Имя	Кратность	Комментарии
Schedule	1..1	Связь с расписанием (номерами пар)
Teaching	1..1	Связь с ведением дисциплины
Attendance	1..*	Связь с посещениями

Описание программной реализации

1 Доменная модель предметной области

Доменная модель в контексте DDD (Domain-Driven Design, Проектирование на основе доменной модели) представляет собой структуру и описание ключевых элементов и концепций предметной области. Она выражает основные понятия, правила, взаимосвязи и процессы, присущие этой предметной области.

Основная идея заключается в том, что разработчики и эксперты по предметной области совместно создают модель, которая отражает их понимание предметной области. Доменная модель становится общим языком, который используется как основа для создания программного обеспечения.

Основные элементы доменной модели в DDD включают в себя:

– сущности (Entities): это объекты с определенной идентичностью, которые имеют длительный жизненный цикл и изменяют свое состояние со временем.

– значения (Value Objects) – это объекты, которые не имеют своей собственной идентичности и определяются только своими характеристиками. Они часто используются для представления атрибутов объектов и могут быть неизменяемыми;

– агрегаты (Aggregates) – это группы связанных сущностей и значений, представляющие единую транзакционную границу. Один из объектов внутри агрегата является корневым объектом, который обеспечивает точку входа для доступа к остальным частям агрегата;

– репозитории (Repositories) – это интерфейсы или сервисы, которые предоставляют методы для поиска, сохранения и извлечения объектов предметной области, скрывая детали работы с хранилищем данных;

– сервисы (Services) – это объекты, представляющие операции и функциональность, которые не могут быть легко привязаны к одной сущности или значению, а скорее относятся к предметной области в целом.

Эти элементы составляют основу для построения доменной модели в DDD. Создание качественной доменной модели помогает команде разработчиков и экспертам по предметной области лучше понять бизнес-задачи, улучшает коммуникацию и обеспечивает более эффективное развитие программного продукта.

Реализация доменной модели включает несколько этапов для того, чтобы воплотить представление предметной области в программном коде:

1. Изучение предметной области.

Изучение бизнес-задач и основных концепций, которые требуется отразить в доменной модели. Это включает в себя общение с экспертами по предметной области, изучение документации и проведение анализа требований.

2. Идентификация ключевых элементов.

Определение сущностей, значений, агрегатов, сервисов и других элементов, которые составляют основу предметной области. Выявление важных атрибутов и отношений между ними.

3. Проектирование классов и структур данных.

С использованием информации о ключевых элементах предметной области, создаются классы, интерфейсы, структуры данных и другие программные конструкции, которые отражают эти элементы.

4. Использование паттернов проектирования.

Применение паттернов проектирования, чтобы эффективно отобразить отношения и функциональность между элементами предметной области.

5. Тестирование и итеративное улучшение.

Разработка доменной модели требует тестирования, чтобы убедиться, что она отражает требования предметной области. Требуется использование юнит-тестов для проверки функциональности и соответствия модели реальным сценариям использования.

6. Интеграция с другими частями системы.

Доменная модель интегрируется с другими слоями системы, такими как слой представления (например, пользовательский интерфейс) и слой доступа к данным (например, база данных).

2 Реализация доменной модели

Исходя из описанного выше, алгоритм реализации доменной модели предметной области выглядит следующим образом:

1. Изучение предметной области.
2. Идентификация ключевых элементов.
3. Проектирование классов и структур данных.
4. Использование паттернов проектирования.
5. Тестирование и итеративное улучшение.
6. Интеграция с другими частями системы.

Пункты 1-2 были описаны выше в разделах «Анализ предметной области», «Видение системы», «Концептуальная модель предметной области», «Описание прецедентов», «Спецификация классов», а также в формате диаграммы вариантов использования и диаграммы классов.

Далее будет описана программная реализация системы, включающая в себя пункты 3-6 алгоритма.

2.1 Создание программного решения .API и проекта .Domain

Работа была начата с создания программного решения в IDE “Visual Studio”. За основу был взят шаблон проекта для создания приложения ASP.NET Core с образцом контроллера для службы HTTP RESTFull, который также можно использовать для представлений MVC и контроллеров ASP.NET Core. Это заготовка под будущий API (E-Journal.API).

Далее в решение был добавлен проект типа «Библиотека классов C#» (E-Journal.Domain), в котором размещены все равнее описанные классы предметной области.

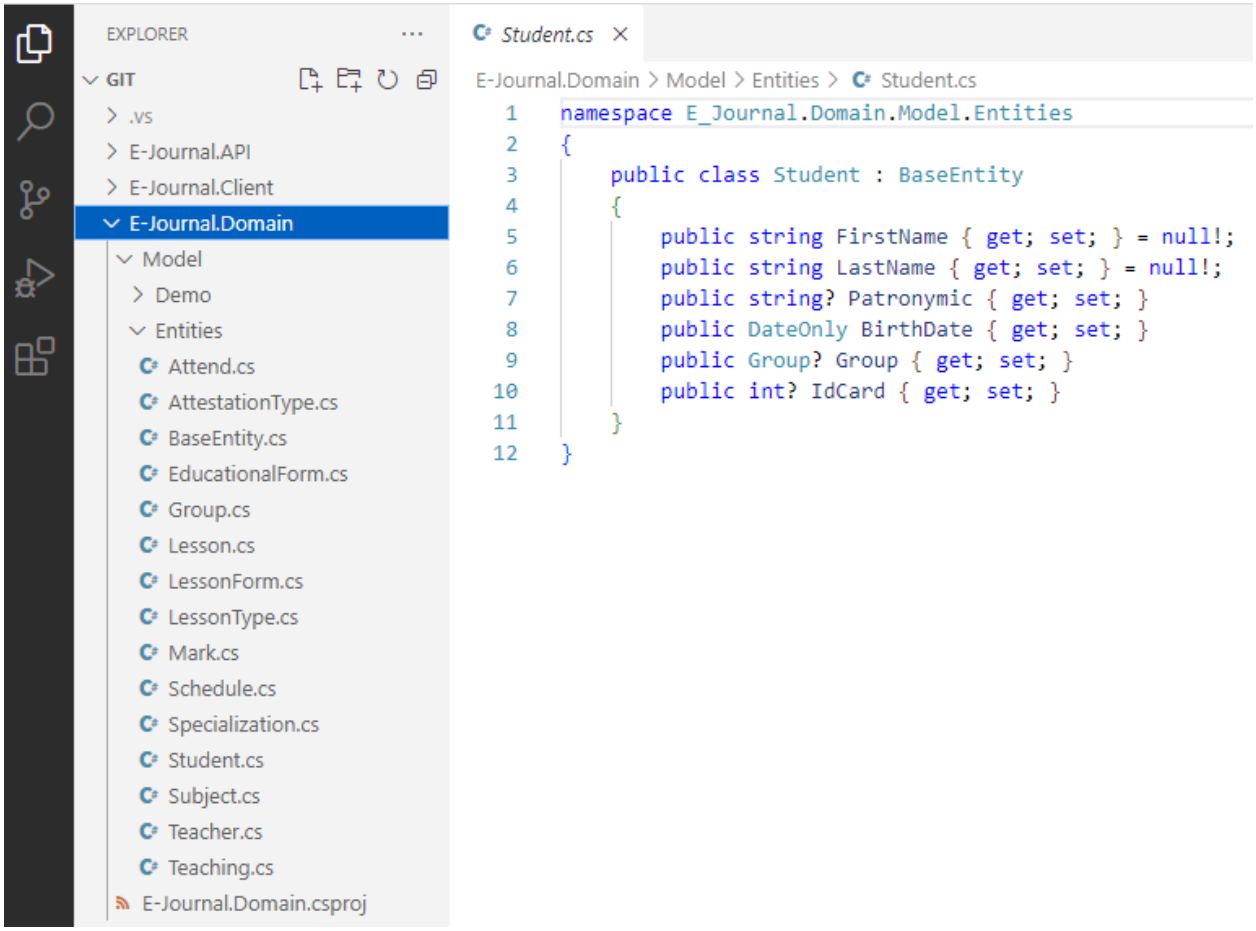

Рисунок 5. Структура проекта “E-Journal.Domain”

Каждая сущность представлена в виде класса, а атрибуты сущности – свойства этого класса. Сущность может содержать как простые поля с простыми типами данных, так и могут в своём составе иметь атрибуты, соответствующие типам данных других сущностей этой предметной области.

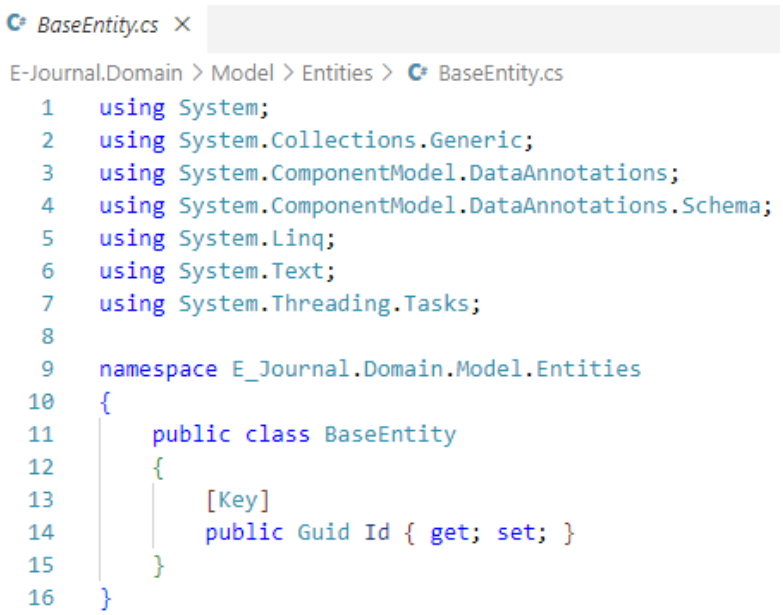

Рисунок 6. Сущность BaseEntity

Для упрощения работы с объектами создан базовый класс “BaseEntity”, который содержит ID, так как этот атрибут присущ всем сущностям предметной области. Остальные классы, соответствующие сущностям предметной области, наследуются от этого класса. Таким образом, не происходит дублирования одного и того же атрибута для каждого из классов.

2.2 Создание проекта .Infrastructure

Для управления сущностями создан проект E-Journal.Infrastructure типа «Библиотека классов C#». Название “.Infrastructure” означает, что это слой с инфраструктурой, отвечающий за логику взаимодействия с СУБД.

2.3 Настройка зависимостей

Для управления сущностями создан проект E-Journal.Infrastructure типа «Библиотека классов C#». Название “.Infrastructure” означает, что это слой с инфраструктурой, отвечающий за логику взаимодействия с СУБД.

К решениям .API и .Infrastructure присоединены необходимые для последующей разработки зависимости, а именно: Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.PostgreSQL (для преобразования классов в таблицы СУБД Postgre Sql) и Microsoft.EntityFrameworkCore.Tools (для добавления инструментов преобразования сущностей в таблицы базы данных).

Далее были связаны между собой решения .API, .Infrastructure и .Domain по схеме, представленной на рисунке 7.

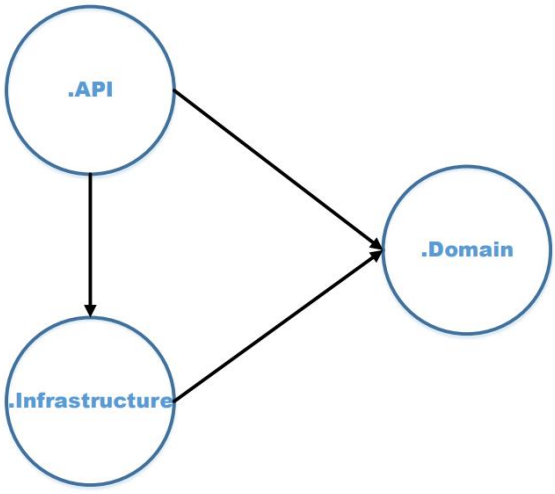

Рисунок 7. Схема связывания решений

Это сделано для того, чтобы расширить область видимости классов между решениями. Представленная схема демонстрирует связи между проектами-слоями в RESTFull решении. Такой тип связи используется для дополнительного взаимодействия при сборке всего решения. Слой с API представляет собой набор контроллеров для взаимодействия с клиентской стороной. Слой с доменом используется исключительно для классы объектной модели. Слой с инфраструктурой включает в себя взаимодействие с созданием объектной модели и миграцией данных в СУБД. Для этого имеются ранее установленные библиотеки с EFCore.Tools и EFCore.PostgreSQL.

2.4 Содержание решения .Infrastructure

Слой инфраструктуры состоит из трёх папок:

– Data – содержит класс DBContext, описывающий установку связей между таблицами в СУБД в соответствии с паттерном «Репозиторий»;

– Repository – содержит интерфейс для взаимодействия с базой данных “IRepository” и класс “BaseRepository”, реализующий этот интерфейс. Здесь определены методы асинхронного взаимодействия с базой данных через средство EntityFrameworkCore, позволяющий не прописывать конкретные SQL-команды, а использовать готовые методы на C#;

– Migrations – содержит генерируемые заранее скрипты для создания, обновления и заполнения базы данных. Скрипты создаются автоматически при помощи команды Add-Migration в диспетчере пакетов NuGet и библиотеки EntityFrameworkCore.PostgreSQL, преобразующей модель базы данных из DBContext в код на C#, создающий/обновляющий реальную базу данных PostgreSQL.

На рисунке 8 представлена структура слоя инфраструктуры и набор методов, определенных в интерфейсе взаимодействия с базой данных.

Рисунок 8. Структура решения .Infrastructure и интерфейс паттерна «Репозиторий»

2.5 Содержание решения .API

Слой программируемого интерфейса приложения (API) содержит описание HTTP-запросов для взаимодействия с интерфейсом базы данных.

Рисунок 9. Структура решения .API и класс базового контроллера

Файлы .json содержат настройки конфигурация приложения, библиотек и базы данных. Класс Program.cs – точка входа в приложение, в нём происходит построение и его запуск.

Папка Controllers содержит базовый контроллер – CRUDController.cs, содержащий общие методы для всех сущностей предметной области

(получить все экземпляры сущности, получить экземпляр по его идентификатору, создание/добавление/удаление экземпляра сущности), а также специфические методы для каждой из сущностей предметной области – [ИмяСущности]Controller.cs.

Вместе с программой запускается Swagger UI - веб-приложение, позволяющая визуализировать определения спецификаций Open API в интерактивном пользовательском интерфейсе. При помощи него можно просматривать и использовать все HTTP-функции, определённые в приложении.

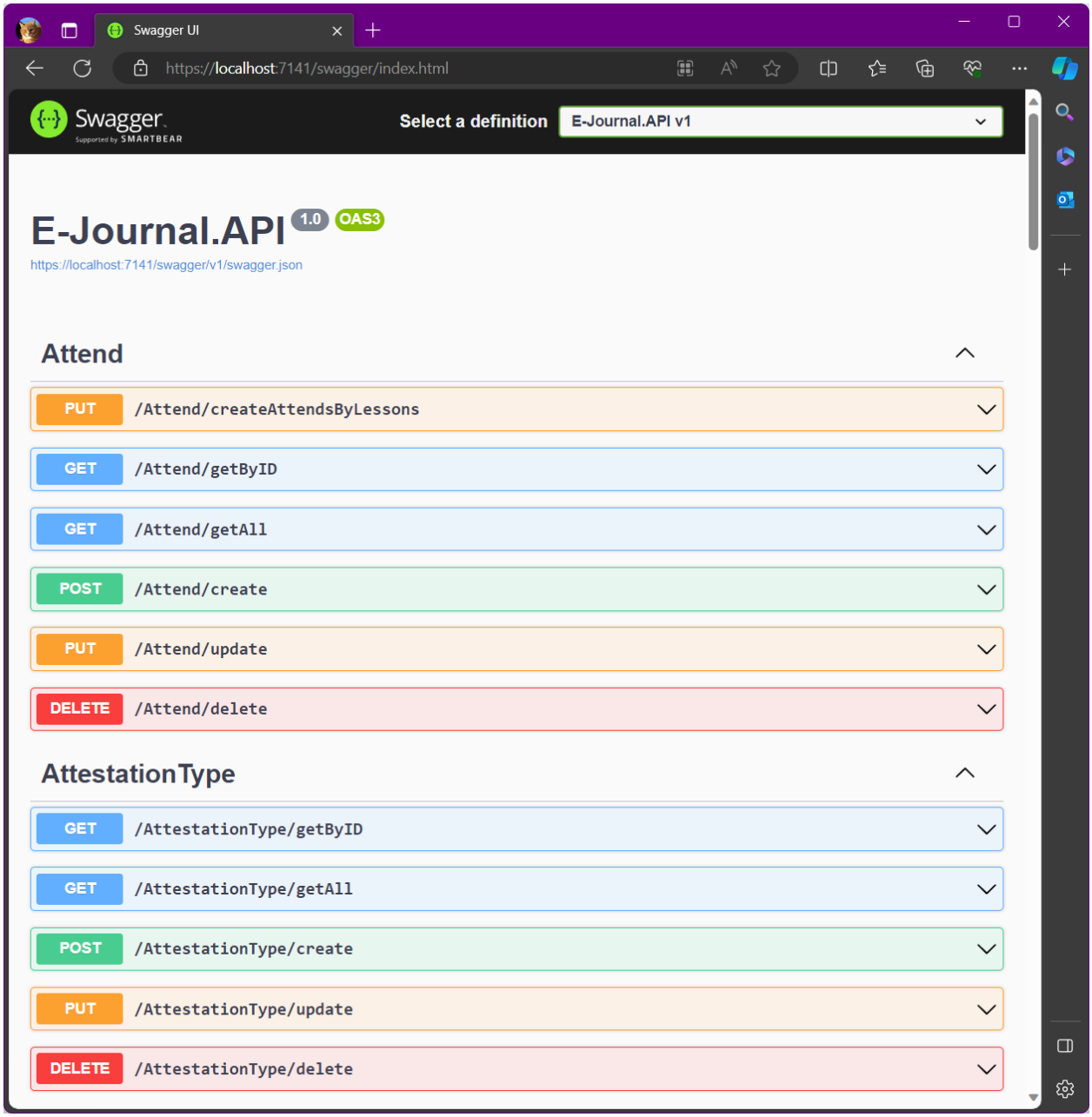

Рисунок 10. Фрагмент описания методов E-Journal.API в Swagger UI

2.6 Содержание клиентской части приложения .Client

Клиентская часть приложения написана при помощи фреймворка Blazor. Blazor представляет UI-фреймворк для создания интерактивных приложений, которые могут работать как на стороне сервера, так и на стороне клиента, на платформе .NET.

Blazor предоставляет разработчикам следующие преимущества:

– написание кода веб-приложений с помощью C# вместо JavaScript;

– использование возможностей экосистемы .NET, в частности, библиотек .NET при создании приложений, безопасности и производительности платформы .NET;

– клиентская и серверная части приложения могут использовать общую логику;

– использование Visual Studio в качестве инструмента для разработки, который имеет встроенные шаблоны для упрощения создания приложения.

Основные элементы проекта:

– папка Properties хранит файл launchSettings.json с конфигурацией, применяемой при разработке, в частности, запуске проекта;

– папка wwwroot хранит статические файлы. По умолчанию в ней находятся используемые файлы css, в частности, файлы фреймворка Bootstrap.

– папка Pages содержит компоненты Razor, которые определяют визуальную часть приложения и его логику:

- Index.razor хранит код стартовой страницы веб-приложения, на которой производися вход в учётную запись;
- Student.razor хранит код страницы с функциональностью студента и старосты;
- Teacher.razor хранит код страницы с функциональностью преподавателя;

– _Imports.razor содержит подключения пространств имен с помощью директивы using, которые будут подключаться в компоненты Razor (файлы с расширением .razor);

– App.razor содержит определение корневого компонента приложения, который представляет веб-страницу приложения;

– файл appsettings.json хранит конфигурацию приложения;

– файл Program.cs содержит класс Program, который представляет точку входа в приложение. В данном случае это стандартный для приложения ASP.NET Core класс Program, который запускает и конфигурирует хост, в рамках которого разворачивается приложение с Blazor.

3 Автоматическое тестирование взаимодействия с базой данных

Для тестирования методов контроллеров было использовано Unit-тестирование при помощи “xUnit”, наиболее распространенный фреймворк в связке с ASP.NET Core.

Тестированию подлежит функционал взаимодействия с базой данных, используемый в проекте, а именно:

– подключение, создание, удаление базы данных;

– метод добавления посещений занятия;

– метод получения занятий студента по его идентификатору;

– метод получения студента по номеру его студенческого билета;

– метод получения старосты по номеру его студенческого билета;

– метод получения занятий преподавателя по его идентификатору;

– метод получения преподавателя по его ФИО;

– метод обновления информации о занятии.

В целях тестирования создан отдельный проект E-Journal.Test, который состоит из:

– настроечных файлов;

– класса TestHelper, обеспечивающего подключение к базе данных, её удаление, создание, а также метод, разрывающий подключение;

– класса Test, содержащего методы тестирования.

На рисунке 11 представлен результат прохождения автоматизированного тестирования.

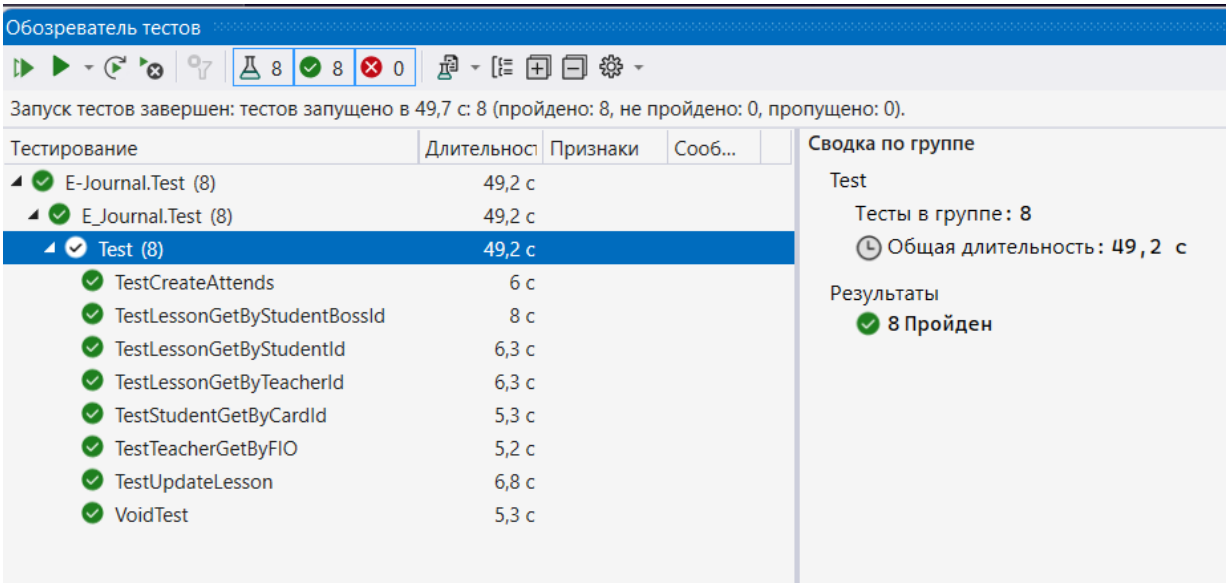

Рисунок 11. Результаты автотестирования

Пользовательский интерфейс приложения

Ниже представлены скриншоты пользовательского интерфейса приложения, отображающие графически все возможности разработанного веб-приложения.

1. Стартовая страница.

Пользователю при входе на страницу предоставляется выбор роли: войти как преподаватель или как студент.

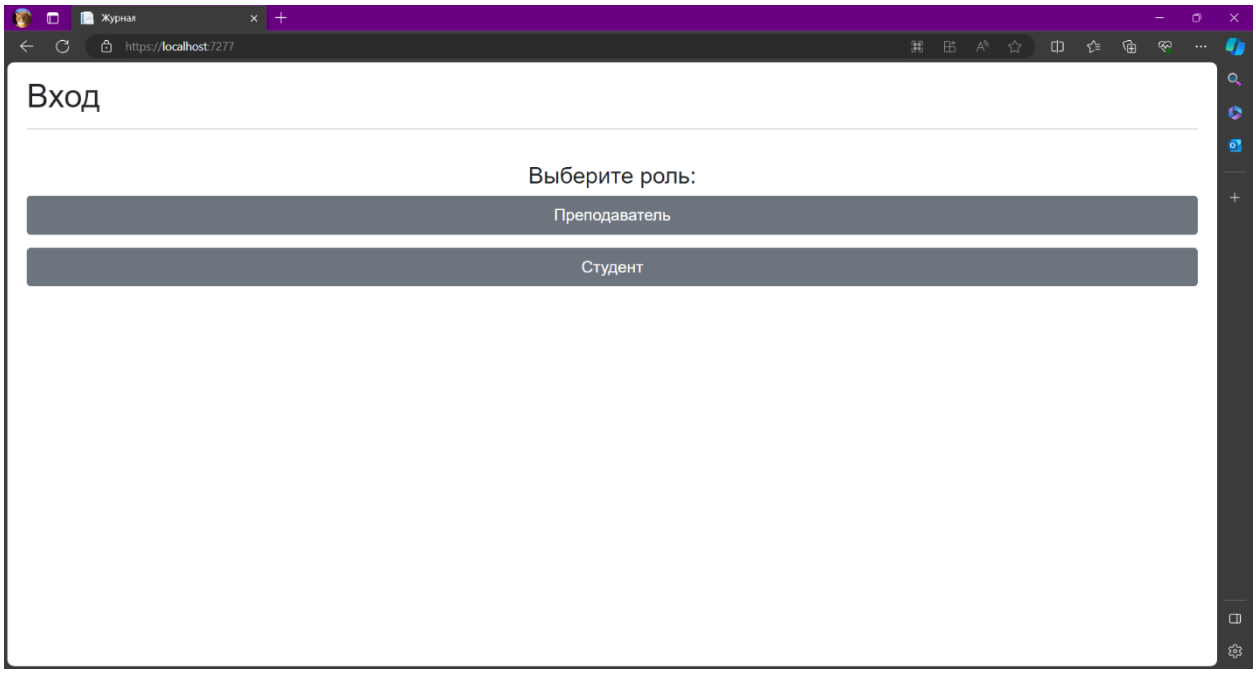

Рисунок 12. Стартовая страница

2. Авторизация под преподавателем.

При авторизации под преподавателем необходимо ввести ФИО преподавателя.

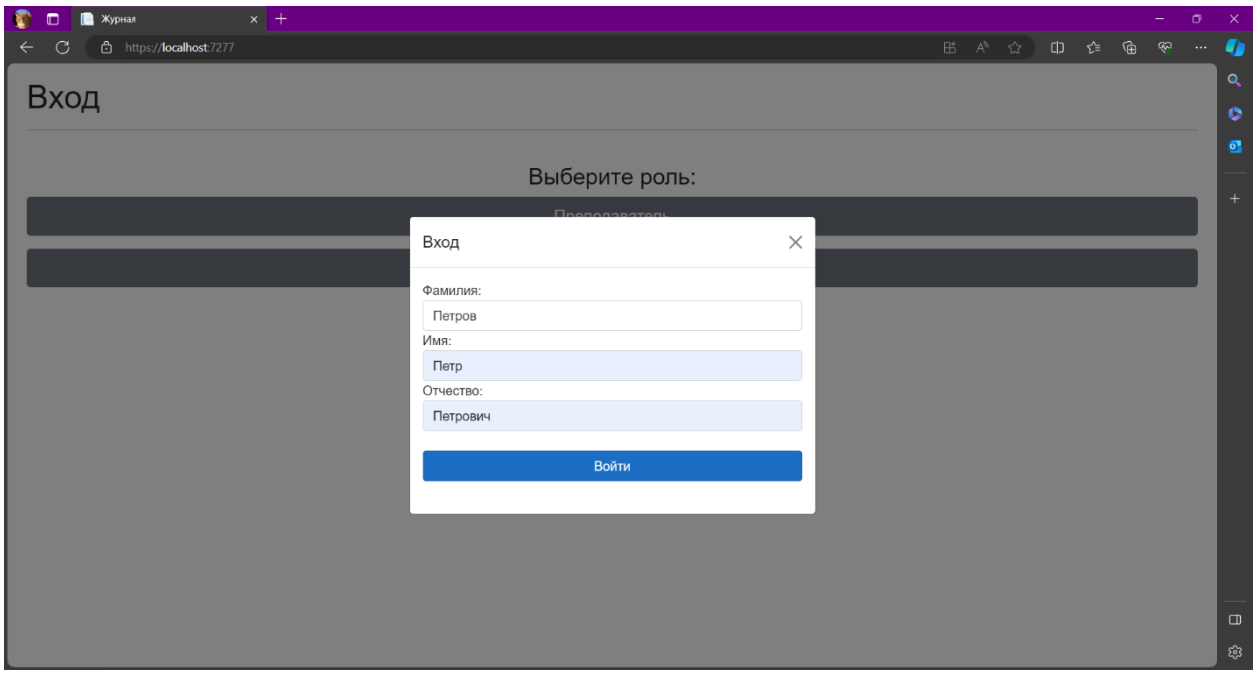

Рисунок 13. Авторизация с ролью «Преподаватель»

В случае осуществления пользователем попытки входа с незаполненными полями предусмотрено информационное сообщение (рис. 13), которое укажет пользователю на некорректность его действий.

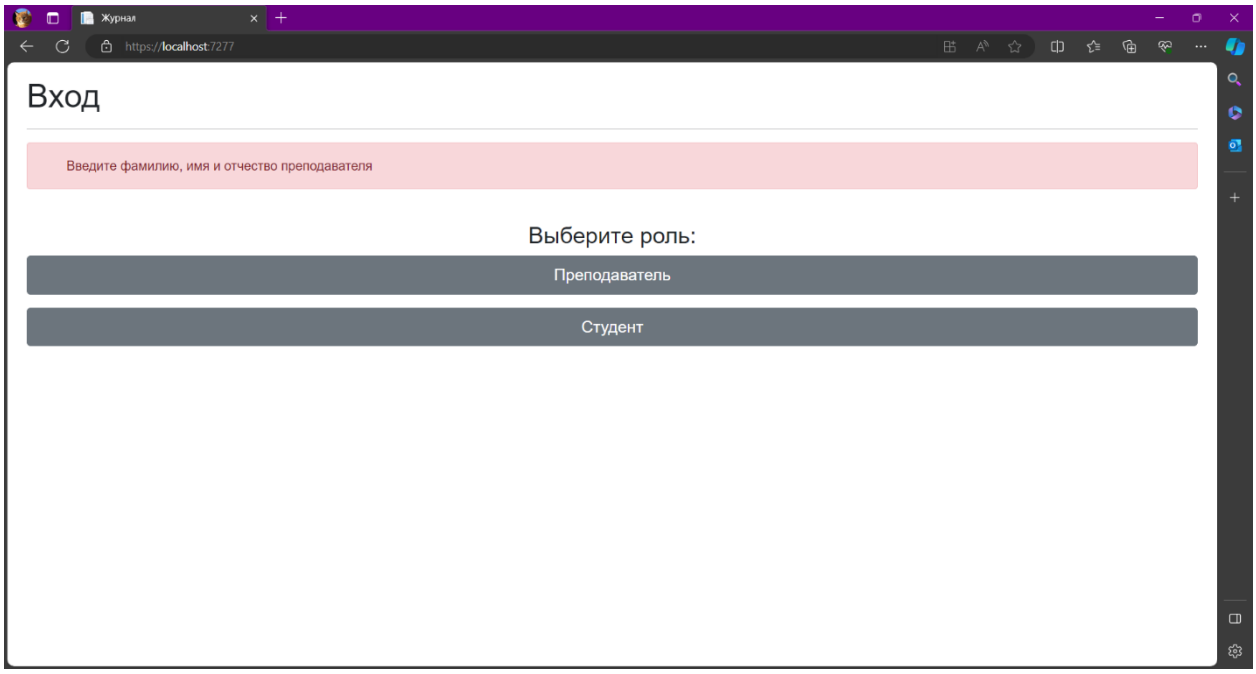

Рисунок 14. Ошибка авторизации с ролью «Преподаватель»

3. Авторизация под студентом.

При авторизации под студентом необходимо ввести номер студенческого билета студента.

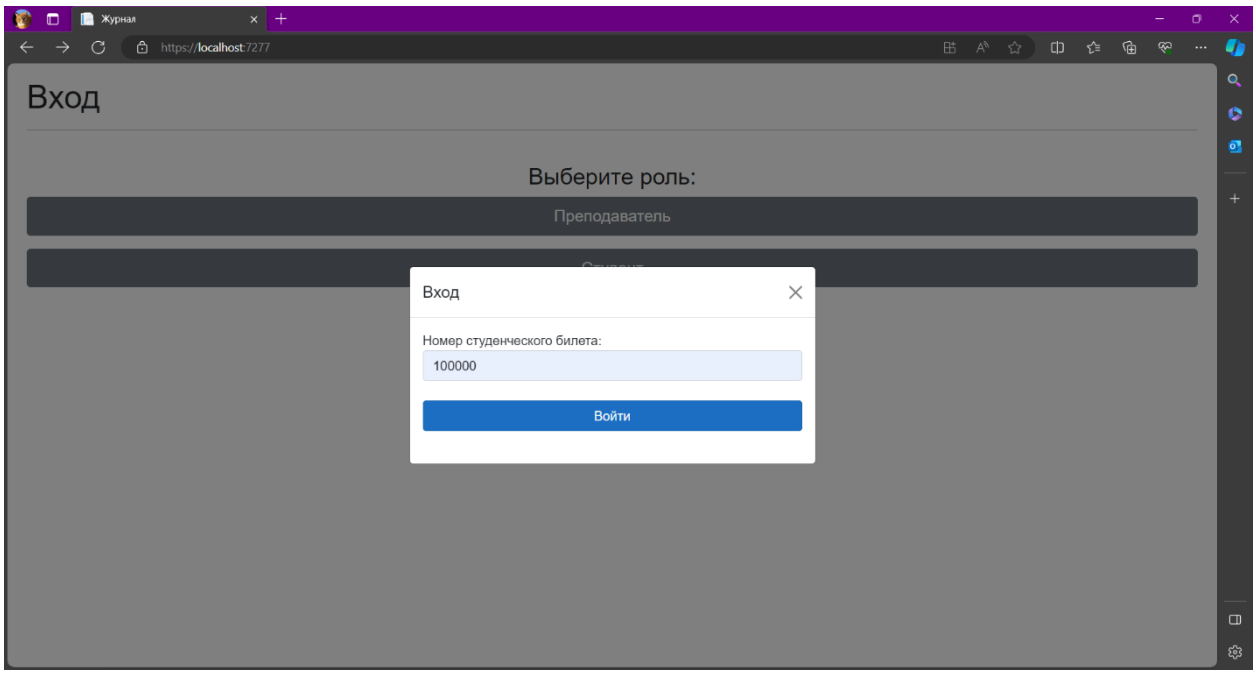

Рисунок 15. Авторизация с ролью «Студент»

В случае осуществления студентом попытки входа с незаполненными полями предусмотрено информационное сообщение (рис. 15), которое укажет пользователю на некорректность его действий.

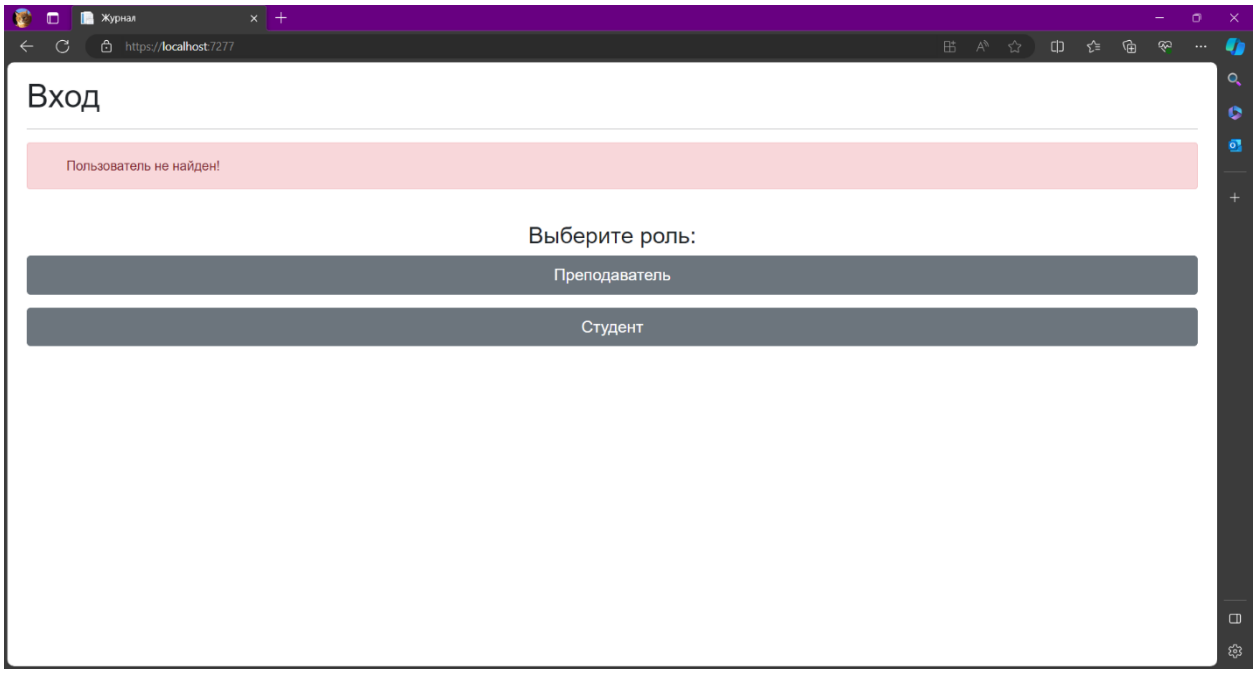

Рисунок 16. Ошибка авторизации с ролью «Студент»

4. Страница преподавателя.

Преподаватель может просматривать личную информацию, выбирать своё занятие, менять тему занятия, просматривать данные по студенту, выставлять отметки студентам по занятию.

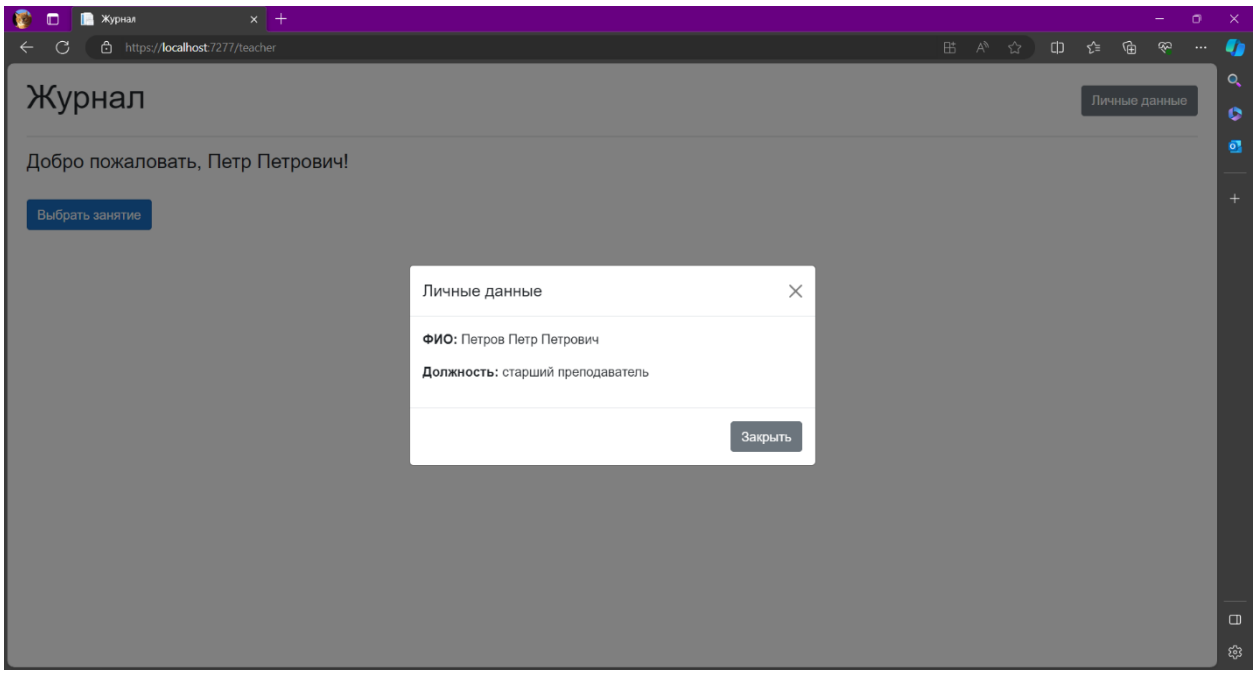

Рисунок 17. Просмотр личных данных с ролью «Преподаватель»

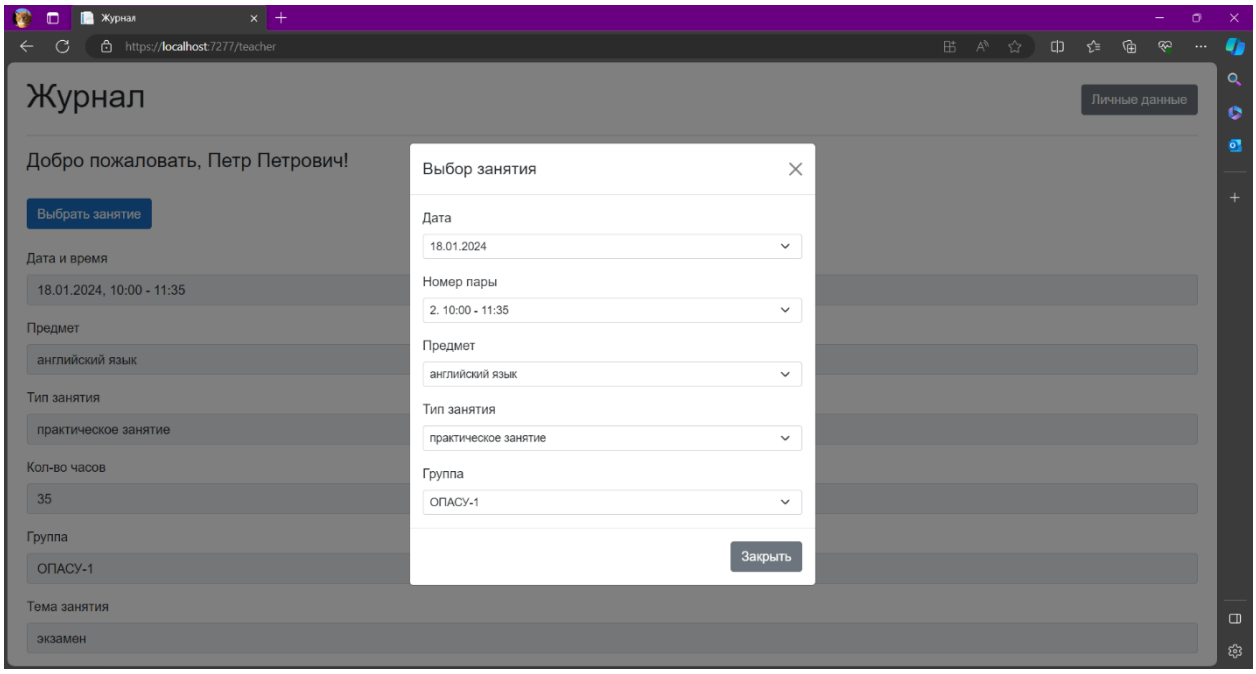

Рисунок 18. Выбор занятия с ролью «Преподаватель»

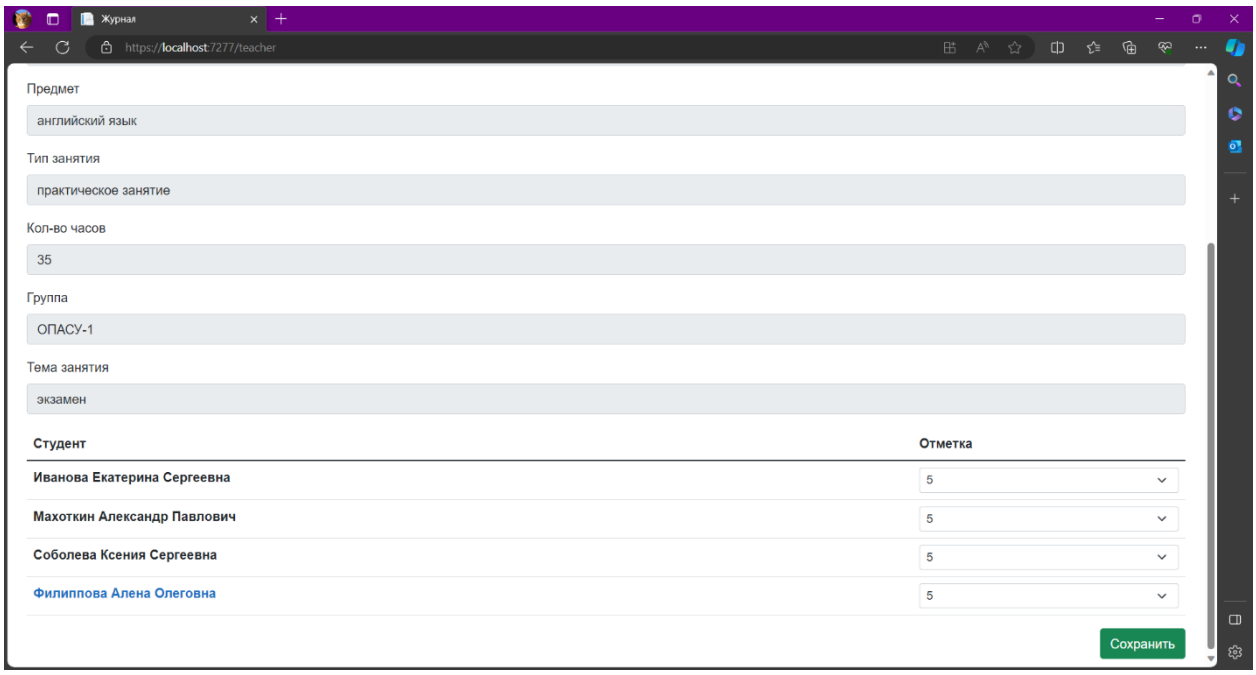

Рисунок 19. Просмотр занятия группы

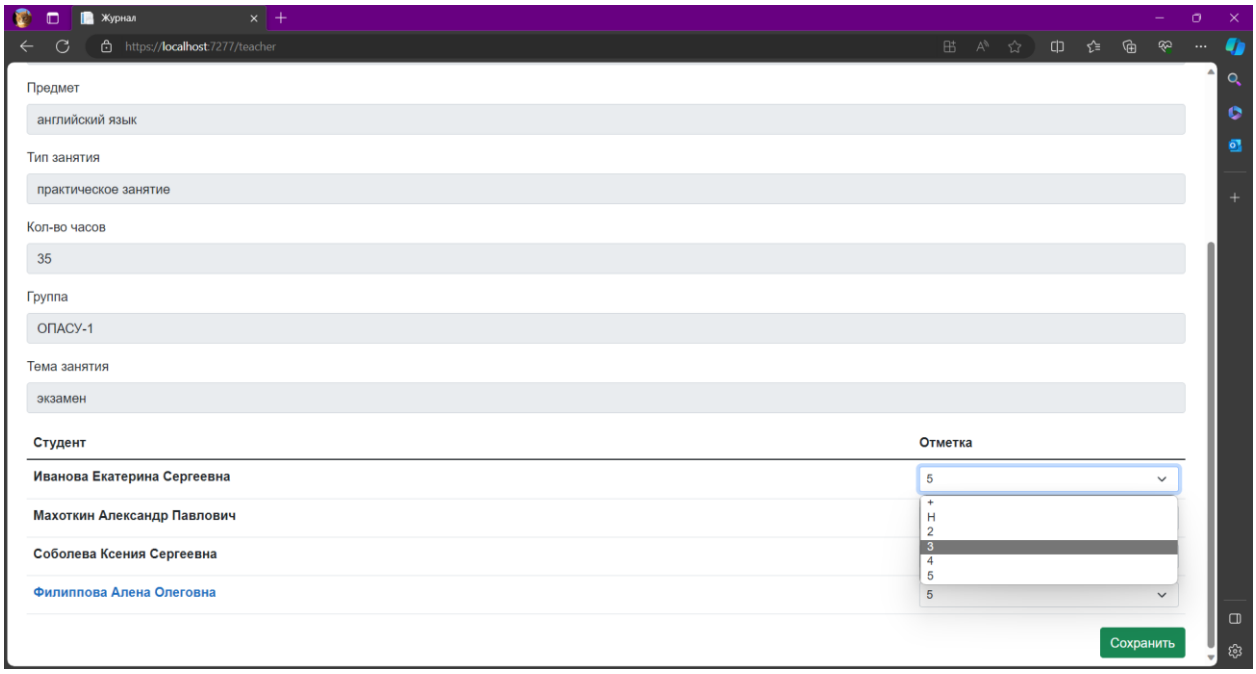

Рисунок 20. Проставление отметки за занятие

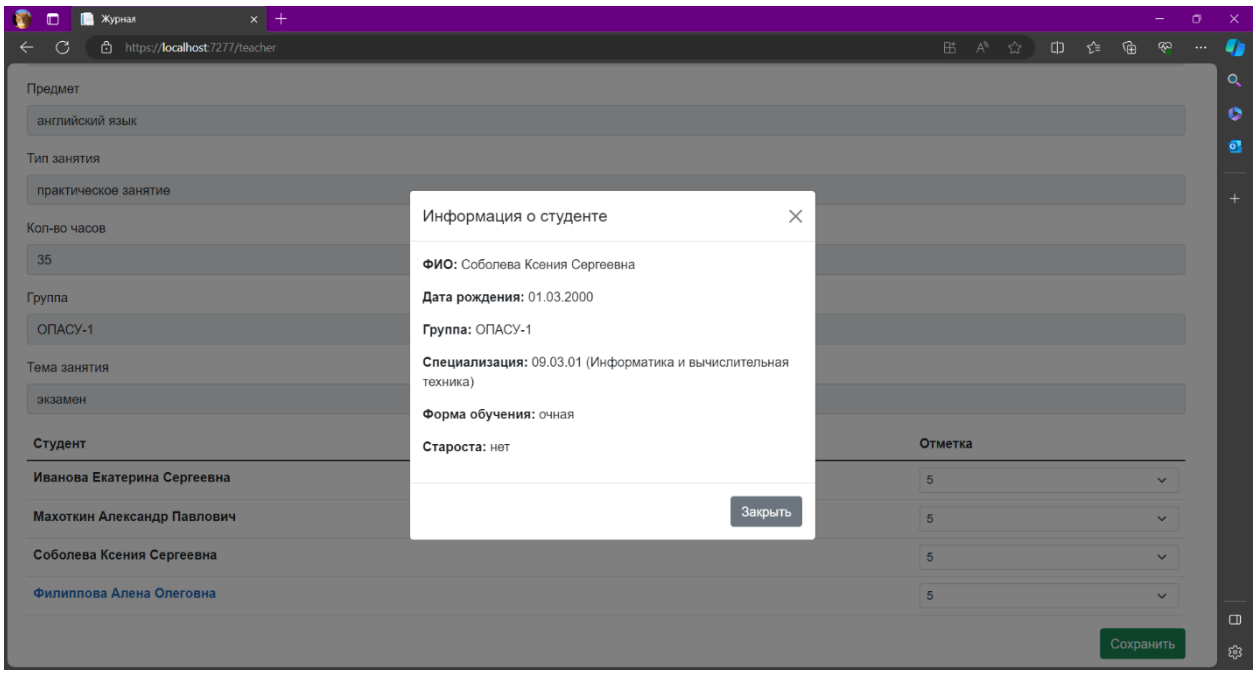

Рисунок 21. Просмотр информации по выбранному студенту

Расписание экзаменов, зачётов преподаватель не может задавать, поэтому при попытке сменить ранее выставленную тему занятия «экзамен»/«зачёт»/«к/п», отображается ошибка. Это ограничение добавлено для того, чтобы отключить преподавателю возможность самостоятельно переносить даты экзаменов/зачетов и исключить дублирования дат и отметок за эти занятия.

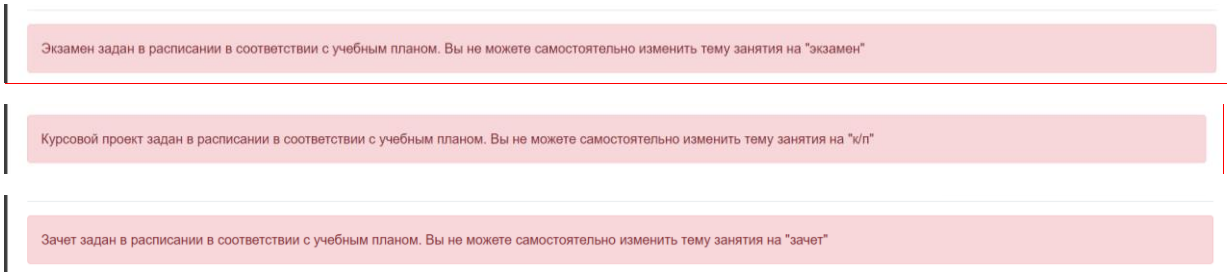

Рисунок 22. Ошибки при изменении темы

5. Страница студента.

Студенту доступен просмотр своего дневника (отметки по всем занятиям) и просмотр личных данных.

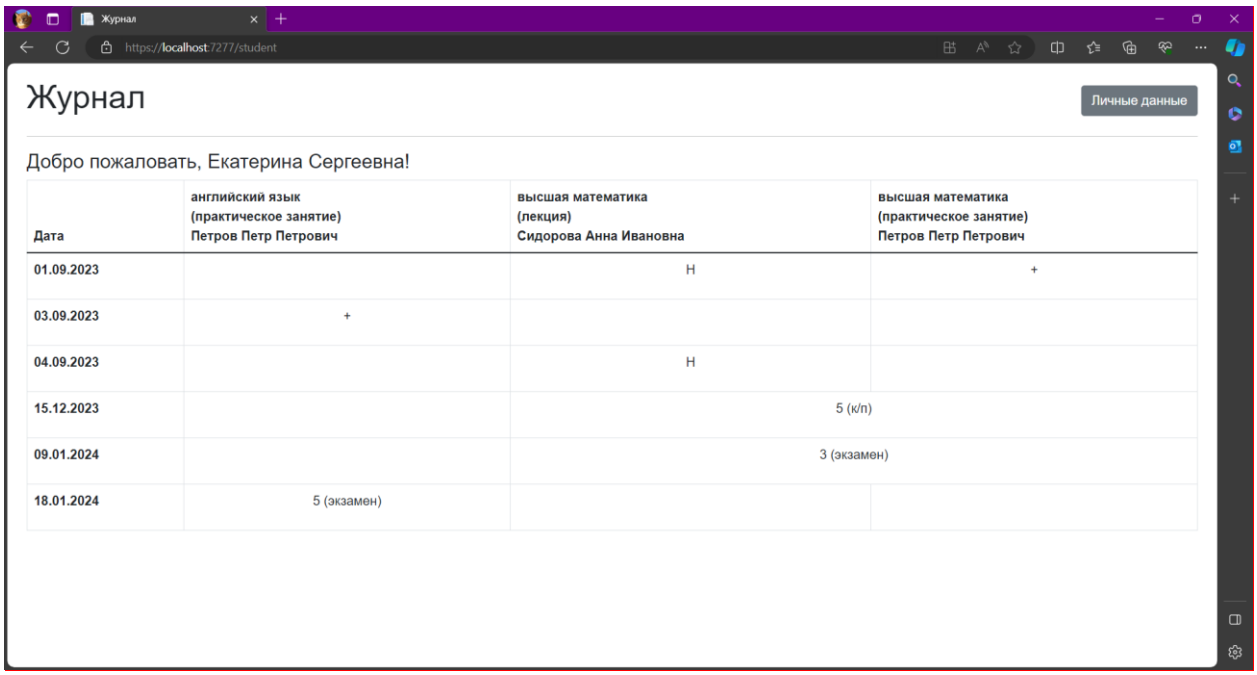

Рисунок 23. Дневник студента

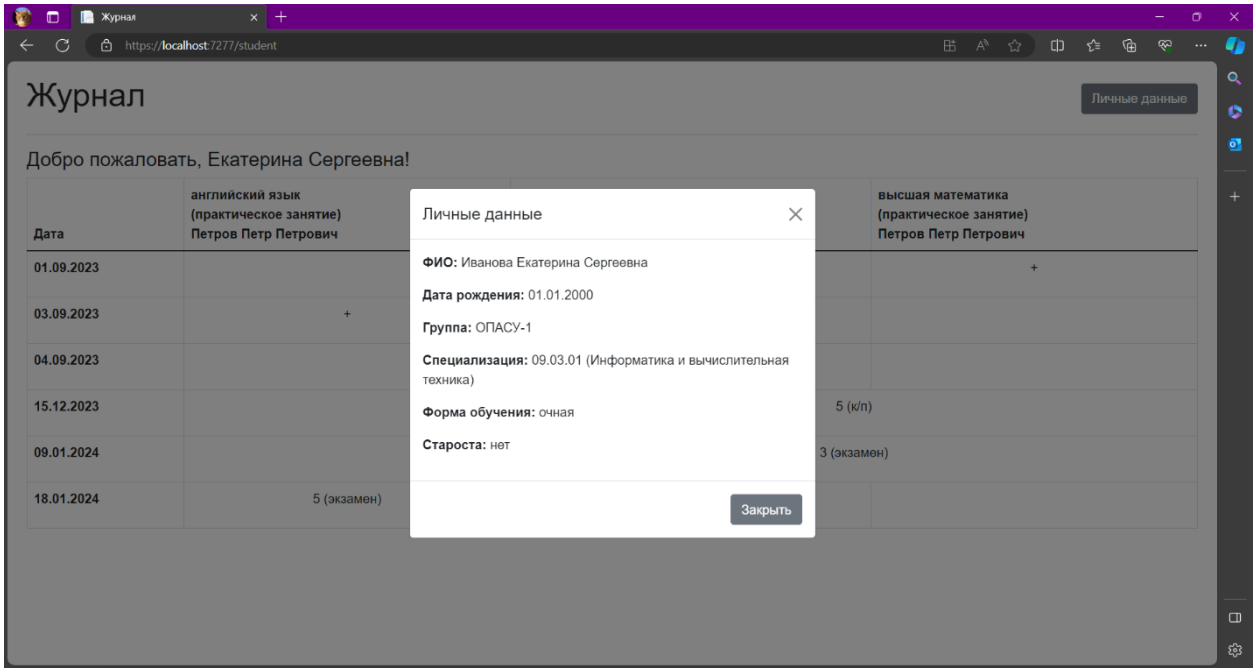

Рисунок 24. Просмотр данных студента о себе

6. Функционал старосты.

Старосте доступен функционал студента, а также возможность перехода в режим старосты, в котором можно просматривать информацию по занятиям и по студентам своей группы.

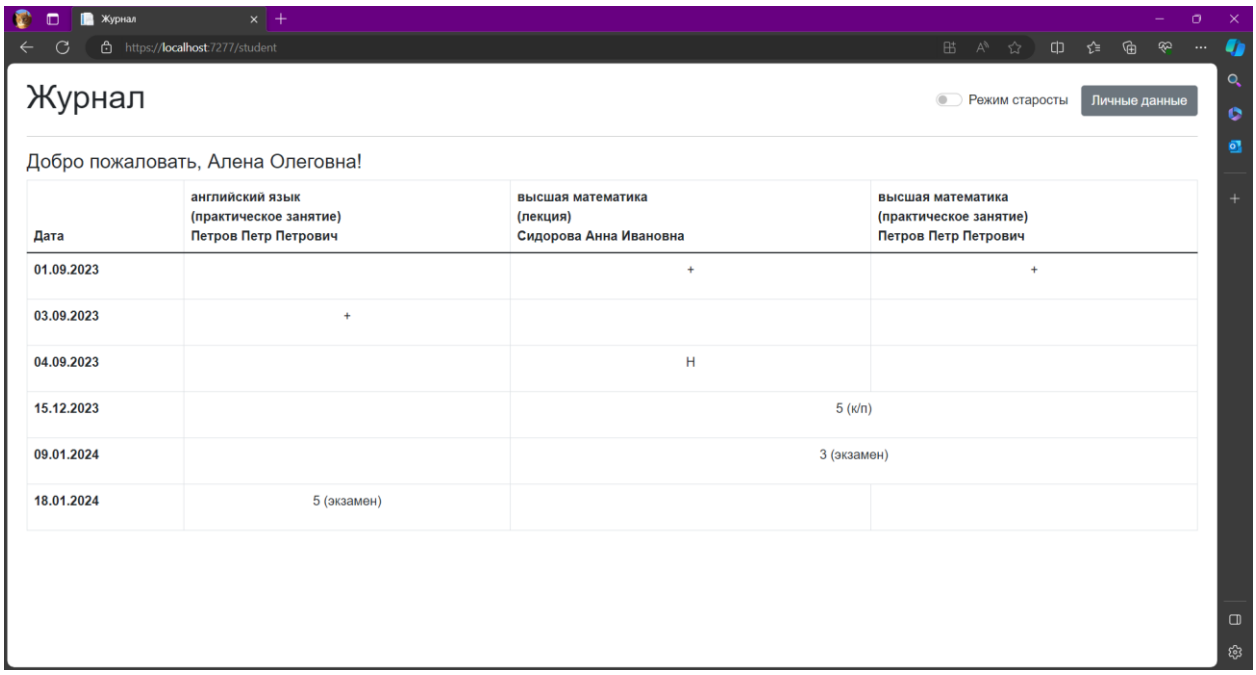

Рисунок 25. Просмотр данных старосты о себе

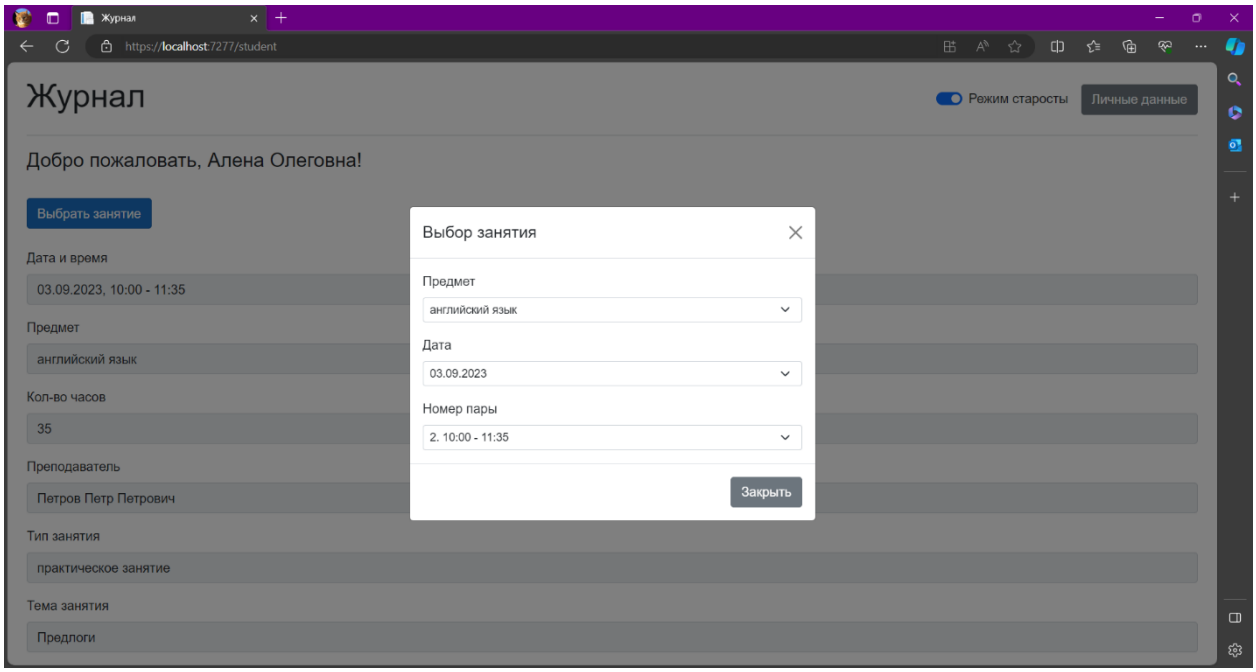

Рисунок 26. Выбор занятия

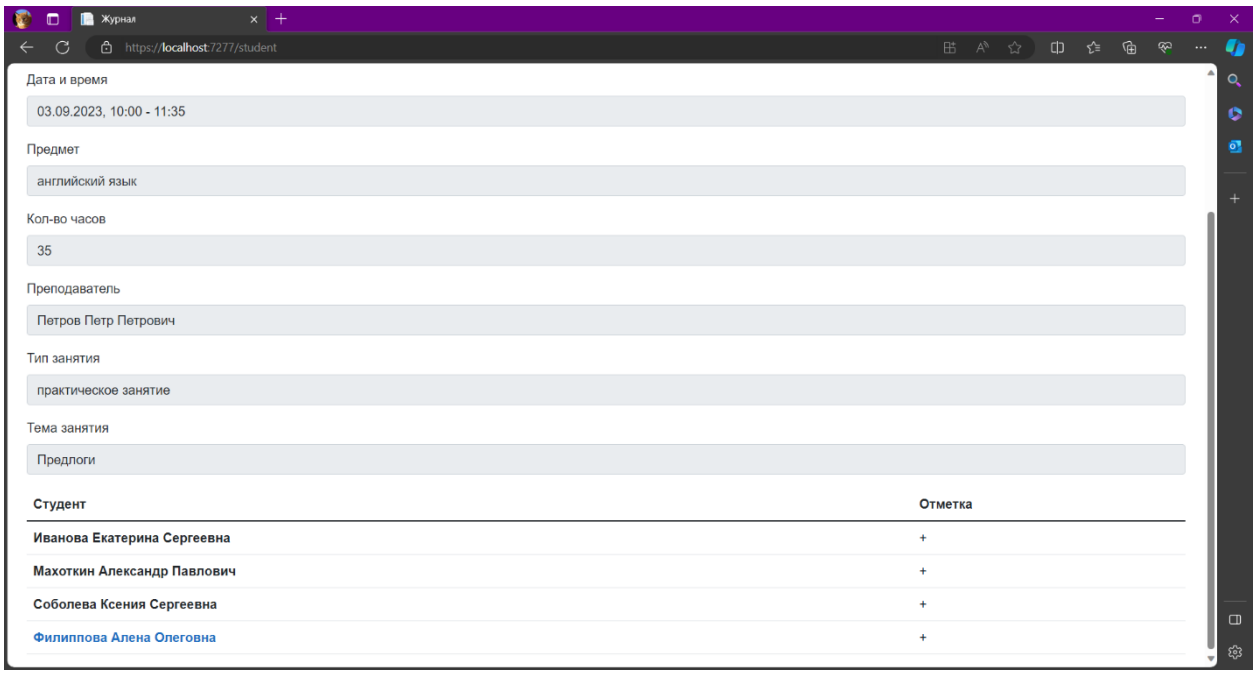

Рисунок 27. Просмотр данных по занятию

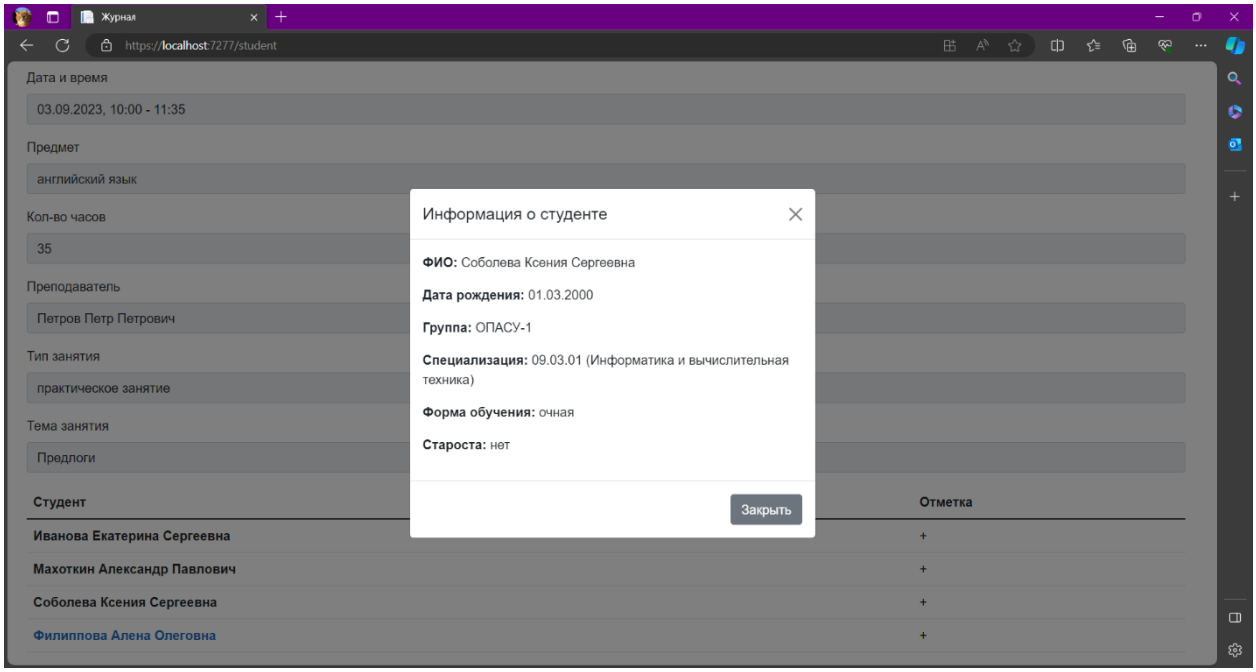

Рисунок 28. Просмотр данных по студенту

Вывод

По итогам выполнения лабораторных работ 1-4 было спроектировано и разработано веб-приложение по предметной области «Электронный журнал в высшем учебном заведении» с учетом его использования разными типами пользователей (студенты, преподаватели, старосты). Для этого были выполнены следующие этапы:

1. Проанализирована и описана предметная область для электронного журнала в высшем учебном заведении.
2. Описано видение системы электронного журнала, в т. ч. позиционирование, описание, возможности и требования к продукту.
3. Разработана концептуальная модель электронного журнала.
4. Разработана диаграмма вариантов использования системы электронного журнала и описаны прецеденты.
5. Разработана диаграмма классов и их спецификация в виде таблиц (описание скалярных свойств, свойств навигации и свойств, содержащих внешние ключи).
6. Создано программное решение в среде разработки Visual Studio:
    - в соответствии со спецификацией классов программно реализована доменную модель предметной области;
    - реализовано взаимодействие с сущностями в соответствии с шаблоном проектирования "Репозиторий";
    - собрана автоматическая миграция классов в базу данных при помощи инструмента Entity Core Framework;
    - реализован программный интерфейс приложения;
    - реализована веб-оболочка приложения при помощи инструмента Blazor Web Assembly (или Blazor .NET);
    - проведено юнит-тестирование взаимодействия приложения с базой данных.