*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***ПОЯСНИТЕЛЬНАЯ ЗАПИСКА***

***к курсовому проекту по дисциплине   
"Технологии программирования"***

***на тему***

*Проектирование и разработка программной системы   
информационной системы «Деканат университета»*

*Выполнил: ст. гр. ЗИСТд-122*

*Саванин Д.В.*

*Принял: доц. кафедры ИСПИ*

*Вершинин В.В.*

*Владимир, 2025*

**<утвержденный Лист задания нА курсовой проект>**

**Аннотация**

Курсовой проект посвящен разработке информационной системы для университета, с использованием программного языка C# и архитектуры ASP.NET MVC. Цель работы — создание веб-приложения "Деканат университета", предназначенного для автоматизации учебного процесса в вузе. В рамках проекта была разработана модель базы данных, которая включает таблицы для хранения информации о студентах, преподавателях, курсах, оценках и расписаниях. Система обеспечивает удобное взаимодействие между студентами, преподавателями и администраторами, позволяет отслеживать успеваемость студентов, управлять курсами, назначать экзамены и формировать расписание занятий.

Для реализации проекта использована архитектура Model-View-Controller (MVC), что позволяет обеспечить разделение логики приложения, интерфейса и управления данными. Применение Entity Framework для работы с базой данных упрощает управление данными, а использование AJAX-технологий позволяет повысить удобство и отзывчивость интерфейса.

Особое внимание в проекте уделено безопасности данных с помощью механизма авторизации и аутентификации пользователей. Реализация системы SimpleMembership позволяет эффективно управлять доступом пользователей в зависимости от их ролей.

Работа ориентирована на создание удобного и эффективного инструмента для администраторов и преподавателей, который позволяет сократить время на административные процессы и улучшить качество образовательного процесса.

Курсовой проект представлен на 56 страницах, рисунков – 5, таблиц – 11, использованных источников – 4, приложений – 2, иллюстрационный материал на 3 листах формата А1.

**Annotation**

The course project is dedicated to the development of an information system for a university, using C# programming language and ASP.NET MVC architecture. The goal of the work is to create a web application "University Dean's Office", designed to automate the educational process in a higher education institution. As part of the project, a database model was developed, which includes tables for storing information about students, lecturers, courses, grades, and schedules. The system ensures convenient interaction between students, lecturers, and administrators, allows tracking student performance, managing courses, assigning exams, and creating class schedules.

The project was implemented using the Model-View-Controller (MVC) architecture, which ensures the separation of application logic, user interface, and data management. Entity Framework is used for database management, simplifying data handling, while AJAX technologies are applied to enhance user interface responsiveness and convenience.

The project places particular emphasis on data security through the implementation of user authorization and authentication mechanisms. The SimpleMembership system provides effective management of user access based on their roles.

The work is focused on creating a convenient and efficient tool for administrators and lecturers, which reduces the time spent on administrative tasks and improves the quality of the educational process.

The course project is presented on 56 pages, figures – 5, tables – 11, sources used – 4, appendices – 2, illustrative material on 3 A1 sheets.

**Содержание**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc195660987)

[1 Описание предметной области 6](#_Toc195660988)

[1.1. Словарь предметной области 6](#_Toc195660989)

[1.2. Сценарий взаимодействия пользователей с системой 7](#_Toc195660996)

[2 проектирование системы 9](#_Toc195661027)

[заключение 26](#_Toc195661028)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc195661029)

[ПРИЛОЖЕНИЕ А Исходный код программной системы 29](#_Toc195661030)

[ПРИЛОЖЕНИЕ Б Структура таблиц БД 32](#_Toc195661033)

# ВВЕДЕНИЕ

Информационные системы играют важную роль в автоматизации процессов управления в образовательных учреждениях. Современные вузы сталкиваются с необходимостью обеспечения эффективного взаимодействия между студентами, преподавателями и административным персоналом. В связи с этим, разработка специализированных систем управления учебным процессом становится актуальной задачей, направленной на повышение качества образовательного процесса и улучшение взаимодействия между всеми участниками учебного процесса.

Проект "Деканат университета" направлен на создание веб-приложения, которое будет использоваться для автоматизации процессов, связанных с управлением учебным процессом в вузе. Внедрение такой системы позволяет значительно упростить и ускорить выполнение рутинных административных задач, таких как учет студентов, распределение курсов, назначение экзаменов и обработка оценок. Данная система предоставляет пользователю интерфейс для управления расписанием, мониторинга успеваемости студентов, управления группами и преподавателями, а также обеспечивает безопасность работы с данными всех участников учебного процесса.

Целью разработки системы "Деканат университета" является создание централизованной платформы, которая объединяет все важнейшие компоненты учебного процесса в единую систему. Система должна обеспечивать хранение и обработку информации о студентах, преподавателях, курсах, группах и оценках, а также предоставлять доступ к этой информации для пользователей с различными правами доступа, включая студентов, преподавателей и администраторов. В результате реализации данного проекта будет создана эффективная и удобная система, которая позволит автоматизировать многие процессы, улучшить качество и скорость обслуживания пользователей, а также снизить вероятность ошибок, возникающих при ручной обработке данных.

Одной из важнейших задач при разработке системы является обеспечение ее функциональности и удобства использования. Система должна обеспечивать легкий доступ к информации, интуитивно понятный интерфейс и высокий уровень безопасности. Важным элементом проекта является возможность авторизации пользователей с различными ролями, что позволяет контролировать доступ к информации в зависимости от полномочий. Например, студенты будут иметь доступ только к своим данным, преподаватели смогут управлять оценками и расписанием, а администраторы смогут осуществлять более широкий контроль над системой.

Для реализации системы выбрана архитектура ASP.NET MVC 4, которая является одной из наиболее популярных для разработки веб-приложений. Этот подход позволяет эффективно разделять логику приложения на три компонента: модель, представление и контроллер, что упрощает поддержку и расширение системы. Использование Entity Framework как ORM (Object-Relational Mapping) инструмента обеспечит удобную работу с базой данных, позволяя взаимодействовать с ней через объектно-ориентированные модели данных, что также улучшает производительность и облегчает миграцию данных.

Система будет построена на основе базы данных, которая будет включать несколько ключевых таблиц, таких как Пользователи, Студенты, Преподаватели, Курсы, Оценки и Расписание. Для эффективной работы с этими данными будет использоваться подход Database-First, что означает создание базы данных до начала разработки приложения. С помощью этого подхода будут сгенерированы соответствующие классы-сущности, которые будут отражать структуру базы данных и позволят легко управлять данными через объектно-ориентированные модели.

Основными пользователями системы будут студенты, преподаватели и административный персонал. Студенты смогут просматривать свое расписание, оценки, а также получать информацию о курсе и экзаменах. Преподаватели смогут управлять оценками студентов, просматривать расписание своих лекций и сдавать экзамены. Администраторы системы смогут управлять пользователями, курсами, расписанием и другими аспектами работы учебного процесса.

Безопасность данных является ключевым аспектом, на который будет обращено особое внимание. Система обеспечит авторизацию и аутентификацию пользователей с использованием стандартных механизмов безопасности. Каждый пользователь будет иметь доступ только к тем данным и функциям, которые соответствуют его роли в системе. Для этого будет использована стандартная библиотека SimpleMembership, которая уже интегрирована в ASP.NET MVC и позволяет легко управлять пользователями и их ролями.

Не менее важной частью проекта является использование технологии AJAX для динамического обновления данных на страницах без необходимости полной перезагрузки веб-страниц. AJAX позволит улучшить взаимодействие с пользователем, ускорив работу системы и повысив удобство интерфейса. Это особенно важно для образовательных систем, где требуется быстрое обновление данных, таких как оценки, расписания и информация о курсах.

Таким образом, система "Деканат университета" будет обеспечивать удобство в управлении учебным процессом, повышать эффективность взаимодействия между студентами, преподавателями и администрацией, а также снизить количество ошибок, связанных с ручным вводом и обработкой данных. Внедрение такой системы позволит вузам сосредоточиться на ключевых аспектах образовательного процесса, а также повысить качество обслуживания студентов и преподавателей. Реализация проекта создаст не только удобную и функциональную платформу для управления учебным процессом, но и гибкую систему, которая легко адаптируется под изменения и будущие потребности образовательных учреждений.

# Описание предметной области

* 1. Словарь предметной области

Информационная система деканата университета (University Dean’s Office System) представляет собой веб-приложение, предназначенное для автоматизации процессов управления учебным процессом в университете, включая регистрацию студентов, учет курсов, преподавателей, расписания и оценок.

Студент (Student) – это человек, который обучается в университете. Основными характеристиками студента являются: уникальный идентификатор, ФИО, дата рождения, группа, курс, учебные достижения (оценки) и статус (например, зачислен, дипломник). Студент может быть привязан к нескольким курсам и оценкам.

Преподаватель (Teacher) – это сотрудник университета, который ведет учебные занятия и проводит экзамены. Основными характеристиками преподавателя являются: уникальный идентификатор, ФИО, курсы, которые он преподает, и экзамены, которые он проводит. Преподаватель может вести несколько курсов и быть связан с множеством студентов.

Курс (Course) – это учебная дисциплина, которая преподается в университете. Каждый курс имеет уникальный идентификатор, название, описание, преподавателя и расписание. Студенты записываются на курсы, и по окончании получают оценки.

Оценка (Grade) – это результат аттестации студента по курсу или экзамену. Оценка фиксируется в системе и связывается с конкретным студентом и курсом. Основными характеристиками являются: оценка (например, A, B, C), дата выставления, и курс, к которому она относится.

Расписание (Schedule) – это информация о времени и месте проведения занятий по каждому курсу. Расписание может быть связано с конкретным курсом и преподавателем. В системе хранится информация о времени, месте и группе студентов, на которые проводится занятие.

* 1. Сценарий взаимодействия пользователей с системой

Студент

* Просмотр расписания:
  + Студент заходит в систему и открывает раздел "Расписание".
  + Система отображает расписание всех обязательных курсов, на которые студент записан, с указанием времени, даты и места проведения занятий.
* Просмотр и оценка успеваемости:
  + Студент заходит в личный кабинет и просматривает свои оценки.
  + Система отображает список всех курсов, по которым студент получил оценки, включая экзамены и зачеты.

Преподаватель

* Создание и ведение курса:
  + Преподаватель входит в систему и создает новый курс, указывая его название, описание, время проведения и назначенного преподавателя.
  + Система сохраняет данные о курсе и автоматически присваивает его преподавателю.
* Добавление оценок:
  + Преподаватель выбирает курс, для которого необходимо внести оценки.
  + Он вводит оценки для каждого студента, прошедшего курс, и сохраняет данные в системе.
  + Система обновляет информацию об успеваемости студентов.
* Просмотр расписания:
  + Преподаватель заходит в раздел "Расписание" и выбирает курс.
  + Система отображает время и место проведения занятий по выбранному курсу.
  + Администратор
* Управление пользователями:
  + Администратор имеет возможность добавлять новых пользователей в систему. Он создает учетные записи для студентов и преподавателей.
  + Для создания нового пользователя администратор вводит его личные данные (ФИО, роль, факультет и т.д.) и сохраняет информацию в системе.
  + Система присваивает пользователю уникальный идентификатор и сохраняет данные.
* Управление курсами:
  + Администратор может редактировать или удалять курсы, добавлять новых преподавателей.
  + Администратор управляет расписанием, добавляя новые занятия или изменяя существующие.
  + Система обновляет данные о курсах и информирует преподавателей и студентов о любых изменениях.
* Просмотр отчетов:
  + Администратор может просматривать отчеты о текущем учебном процессе, успеваемости студентов, списке курсов и преподавателей.
  + Система генерирует отчеты на основе введенных данных и отображает их в удобном для адмистратора формате.

# проектирование системы

* 1. **Создание диаграмм UML**
     1. **Диаграмма прецендентов**

Это диаграмма взаимодействия пользователей с системой ИС Деканат университета, которая описывает ключевые действия каждого типа пользователя и их взаимосвязь с системой. Эта диаграмма позволяет понять, как различные пользователи взаимодействуют с системой, какие операции они выполняют и как эти действия связаны с процессами в ИС Деканат университета.

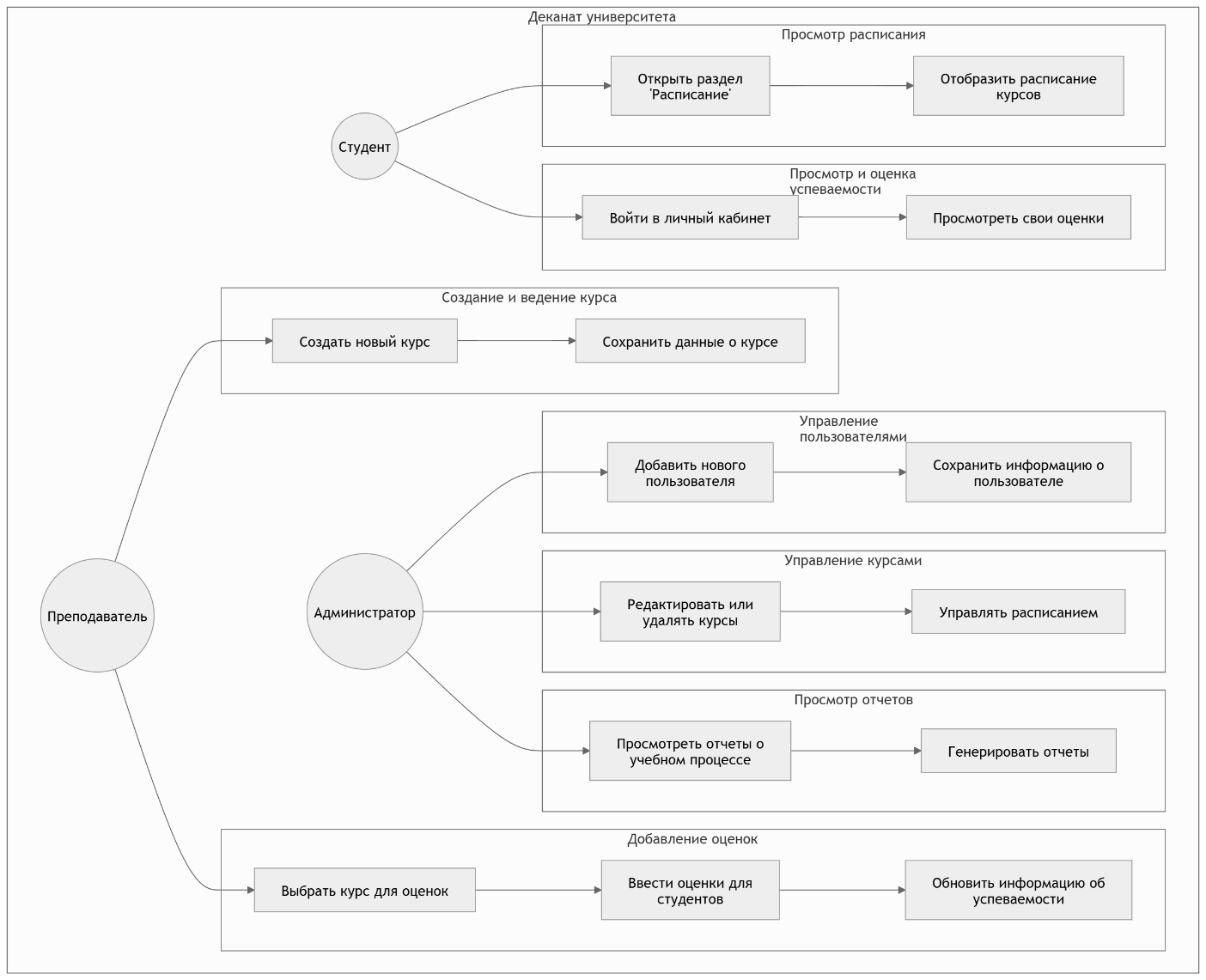


Рисунок 1. Диаграмма прецендентов.

* + 1. **Диаграмма последовательностей**

Взаимодействие пользователей с системой происходит по заранее определенным сценариям: студент просматривает расписание и оценки, преподаватель вводит оценки по курсам, а администратор управляет курсами, пользователями и может генерировать отчеты.

Каждое действие связано с системой, которая выполняет соответствующие функции (например, отображение расписания, обновление успеваемости, создание нового курса и т.д.).

Весь процесс организован таким образом, чтобы обеспечить эффективную работу всех участников учебного процесса, включая студентов, преподавателей и администраторов.

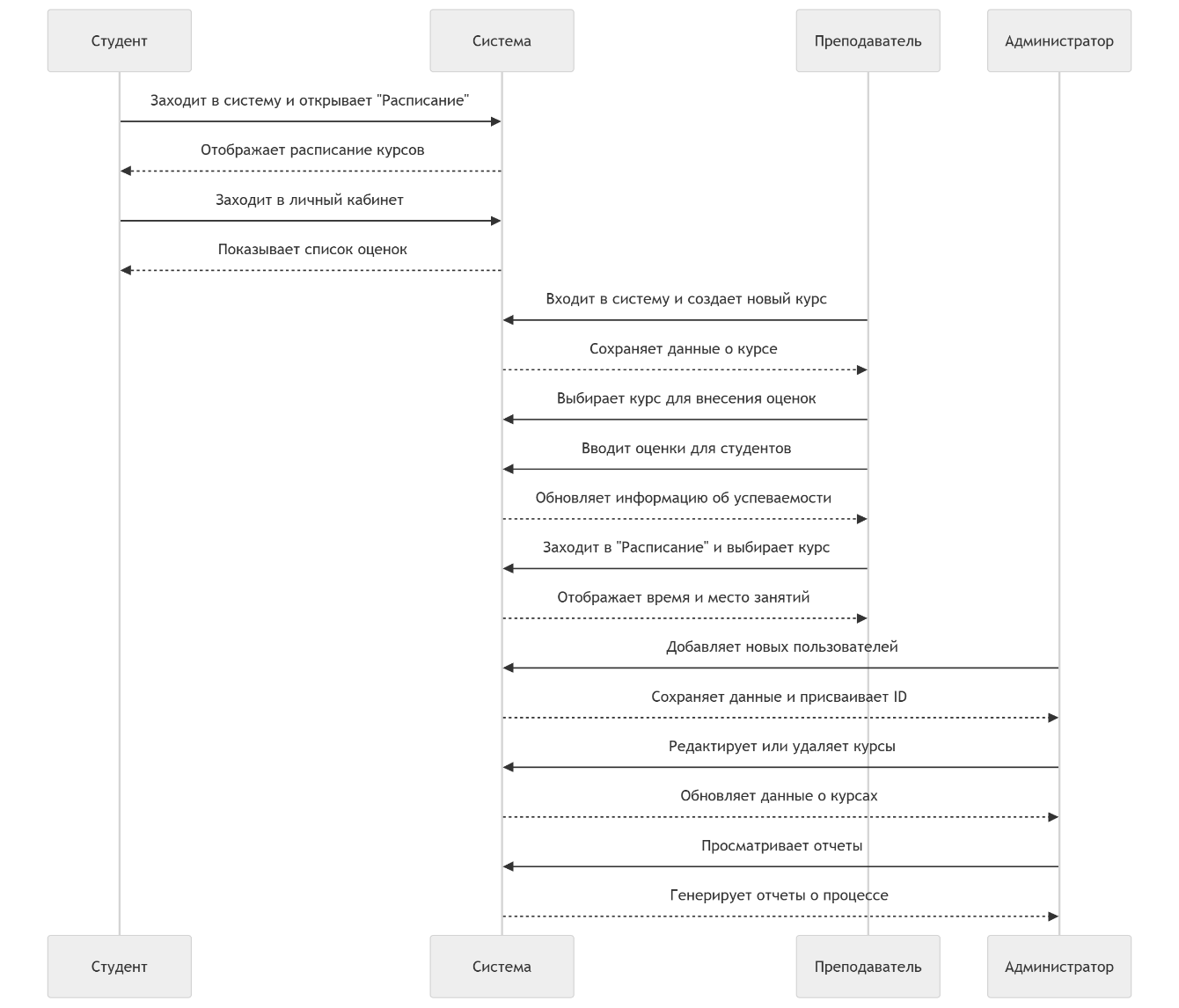


Рисунок 2. Диаграмма последовательностей.

* + 1. **Диаграмма видов деятельности**

Диаграмма видов деятельности представляет собой визуальное отображение процесса взаимодействия студента с информационной системой "Деканат университета". Она охватывает два ключевых процесса: просмотр расписания и просмотр и оценка успеваемости.

* Просмотр расписания:
  + Студент входит в систему и открывает раздел "Расписание".
  + Система отображает расписание всех обязательных курсов студента.
  + В расписании указаны время, дата и место проведения занятий, что позволяет студенту эффективно планировать свое время.
* Просмотр и оценка успеваемости:
  + Студент заходит в личный кабинет, где система отображает список всех курсов, которые студент проходил.
  + В этом разделе студент может увидеть свои оценки, включая результаты экзаменов и зачетов.

Эта диаграмма помогает наглядно представить, как студент взаимодействует с системой и какие действия он выполняет для получения информации о своем расписании и успеваемости.

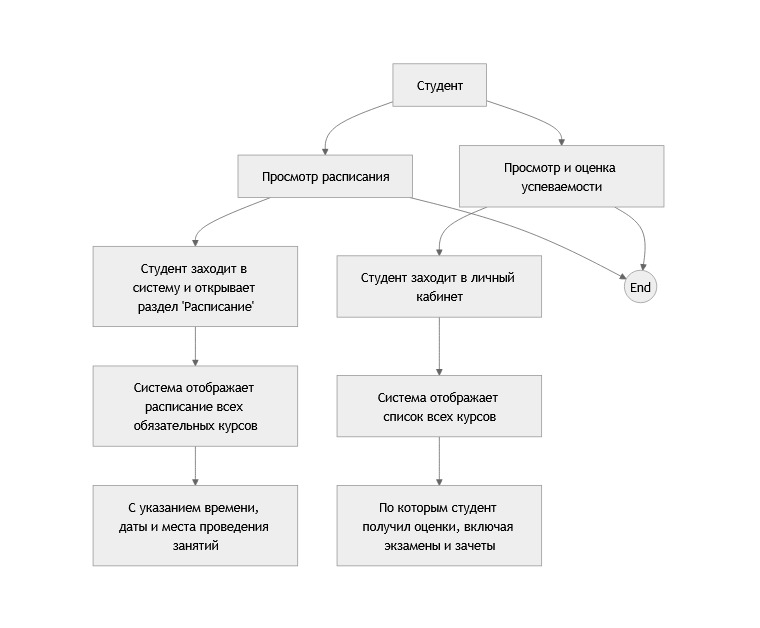


Рисунок 3. Диаграмма видов деятельности.

* + 1. **Диаграмма классов**

Диаграмма классов представляет собой схематическое отображение структуры системы и описывает основные классы, их атрибуты и методы, а также связи между ними. Эта диаграмма классов помогает четко понять структуру системы и функциональность каждого компонента.

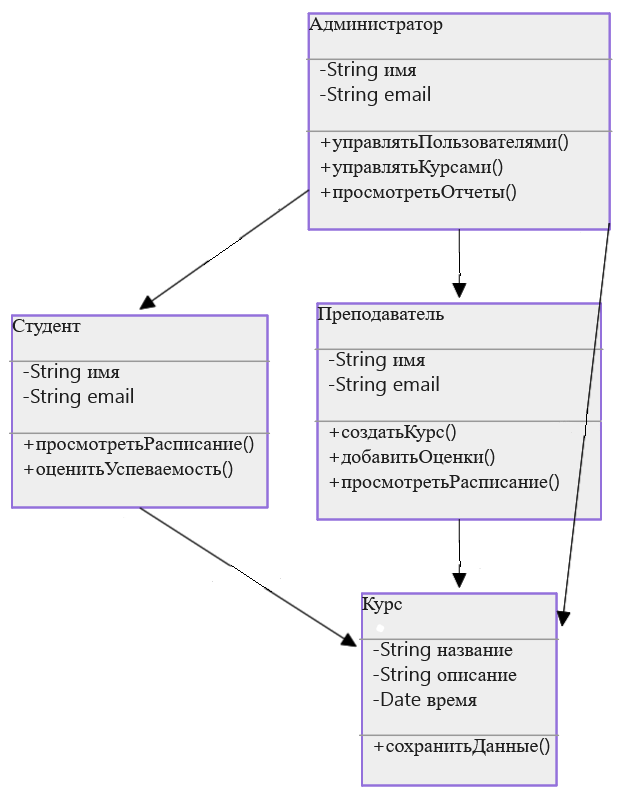


Рисунок 4. Диаграмма классов.

* + 1. **Диаграмма состояний**

Диаграмма состояний представляет собой диаграмму, которая описывает изменения состояния объектов системы в зависимости от различных событий и действий. В данном случае диаграмма отображает изменения состояний в системе "Деканат университета" для различных пользователей и их взаимодействий с системой.

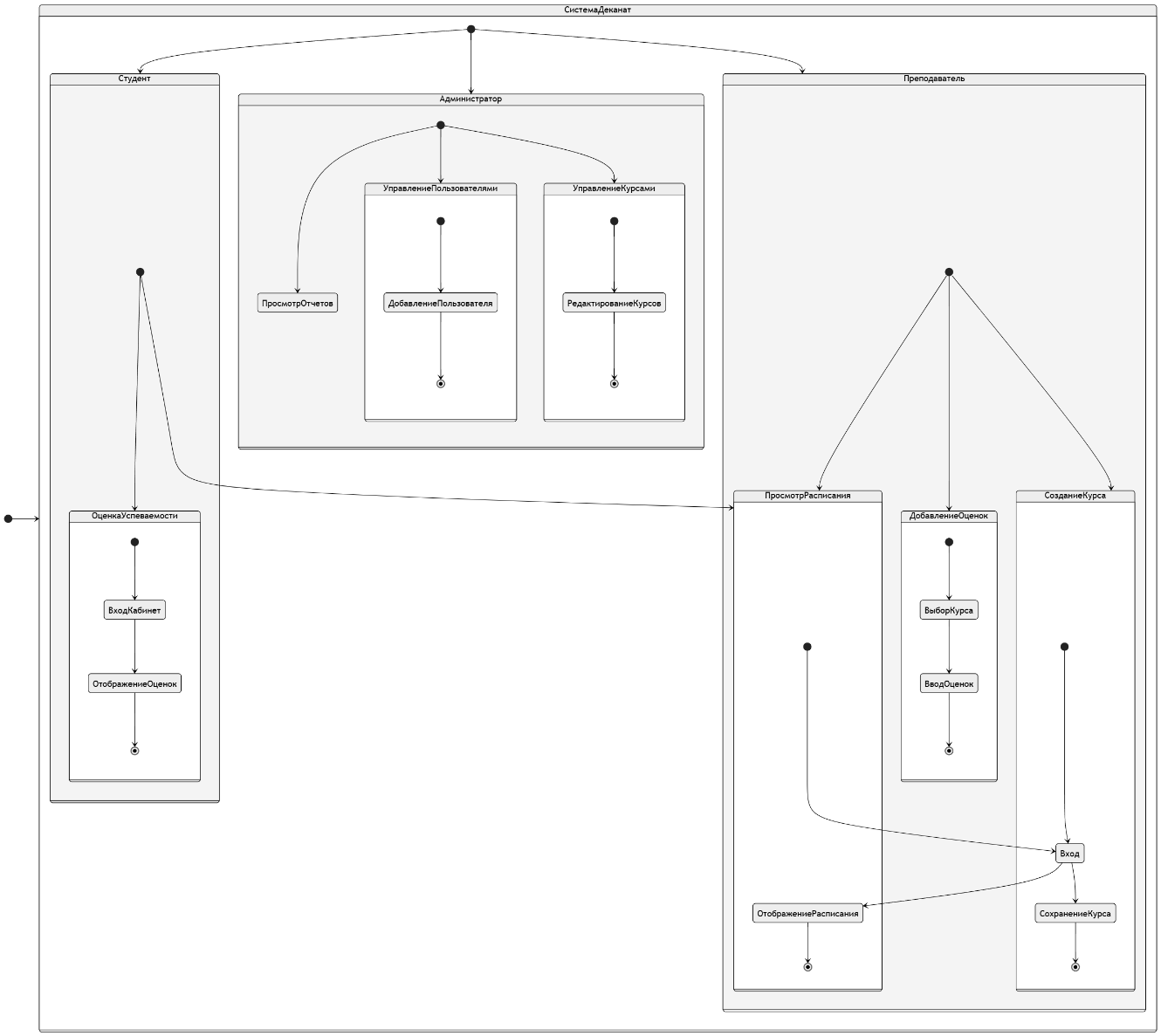


Рисунок 5. Диаграмма состояний.

* 1. **Общие принципы организации системы**

Разрабатываемая система представляет собой веб-приложение, построенное на базе архитектуры **ASP.NET MVC** 4 с использованием шаблона "Интернет-приложение".

Как и в любом приложении, реализующем шаблон **MVC** (Model-View-Controller), система содержит все компоненты: контроллеры, модели и представления, которые взаимодействуют друг с другом для реализации бизнес-логики.

Архитектура и технологии:

* Фреймворк: ASP.NET Core 7.0
* Паттерн проектирования: MVC (Model-View-Controller)
* ORM: Entity Framework Core 7.0
* База данных: Microsoft SQL Server
* Аутентификация: Cookie-based authentication
* Frontend: Bootstrap, Font Awesome для иконок

Основные компоненты системы:

* Модели (Models):
  + User - базовая сущность для всех пользователей системы
  + Course - информация о курсах
  + Schedule - расписание занятий
  + StudentCourse - связь между студентами и курсами
  + Все модели определены в пространстве имен StudentManagementSystem.Models
* Контроллеры (Controllers):
  + AccountController - управление аутентификацией
  + StudentController - функционал для студентов
  + TeacherController - функционал для преподавателей
  + AdminController - административные функции
  + Все контроллеры используют атрибуты авторизации для защиты доступа
* Представления (Views):
  + Реализованы с использованием Razor Pages (.cshtml)
  + Используют Layout для общего шаблона страниц
  + Адаптивный дизайн на основе Bootstrap
* Доступ к данным:
  + Реализован через ApplicationDbContext
  + Использует Entity Framework Core для ORM
  + Миграции для управления схемой базы данных
* Безопасность:
  + Cookie-based аутентификация
  + Ролевая авторизация (Student, Teacher, Administrator)
* Основной функционал:
  + Управление курсами (создание, редактирование)
  + Управление расписанием занятий
  + Система оценивания
  + Запись студентов на курсы
  + Просмотр расписания и оценок
* Дополнительные особенности:
  + Логирование через ILogger
  + Обработка ошибок и исключений
  + Dependency Injection для сервисов
  + Session management
  + Responsive дизайн
  1. **Реализация модели**

Для реализации модели данных в системе использован подход Code-First с Entity Framework Core 7.0. Рассмотрим основные принципы реализации модели:

* **Определение сущностей**

В системе определены основные сущности, представляющие участников и объекты учебного процесса:

* + User - базовая сущность для пользователей системы (студенты, преподаватели, администраторы)
  + Course - учебные курсы
  + Schedule - расписание занятий
  + StudentCourse - связующая сущность между студентами и курсами
* **Валидация данных**

Для обеспечения целостности данных используются атрибуты Data Annotations:

* [Required] для обязательных полей
* [StringLength] для ограничения длины строк
* [EmailAddress] для валидации email
* [Range] для числовых диапазонов
* **Связи между сущностями**

В системе реализованы следующие типы связей:

* Один-ко-многим: преподаватель и его курсы
* Многие-ко-многим:студенты и курсы (через таблицу StudentCourse)
* Один-ко-многим: курс и его расписание
* **Контекст базы данных**

Доступ к данным осуществляется через ApplicationDbContext, который:

* Определяет наборы сущностей (DbSet)
* Конфигурирует связи между моделями
* Обеспечивает доступ к базе данных
* **Управление данными**
  + Миграции используются для управления схемой базы данных
  + Начальные данные (seed data) добавляются при первом запуске
  + Изменения в базе данных отслеживаются через систему миграций
* **Особенности реализации**
  + Использование навигационных свойств для удобной работы со связанными данными
  + Каскадное удаление настроено выборочно для обеспечения целостности данных
  + Ленивая загрузка (lazy loading) отключена в пользу явной загрузки связанных данных
* **Безопасность данных**
  + Пароли хранятся в зашифрованном виде
  + Доступ к данным ограничен через систему ролей
* **Такой подход к реализации модели обеспечивает:**
  + Типобезопасность при работе с данными
  + Удобное управление схемой базы данных
  + Четкое разделение ответственности
  + Легкость поддержки и расширения системы

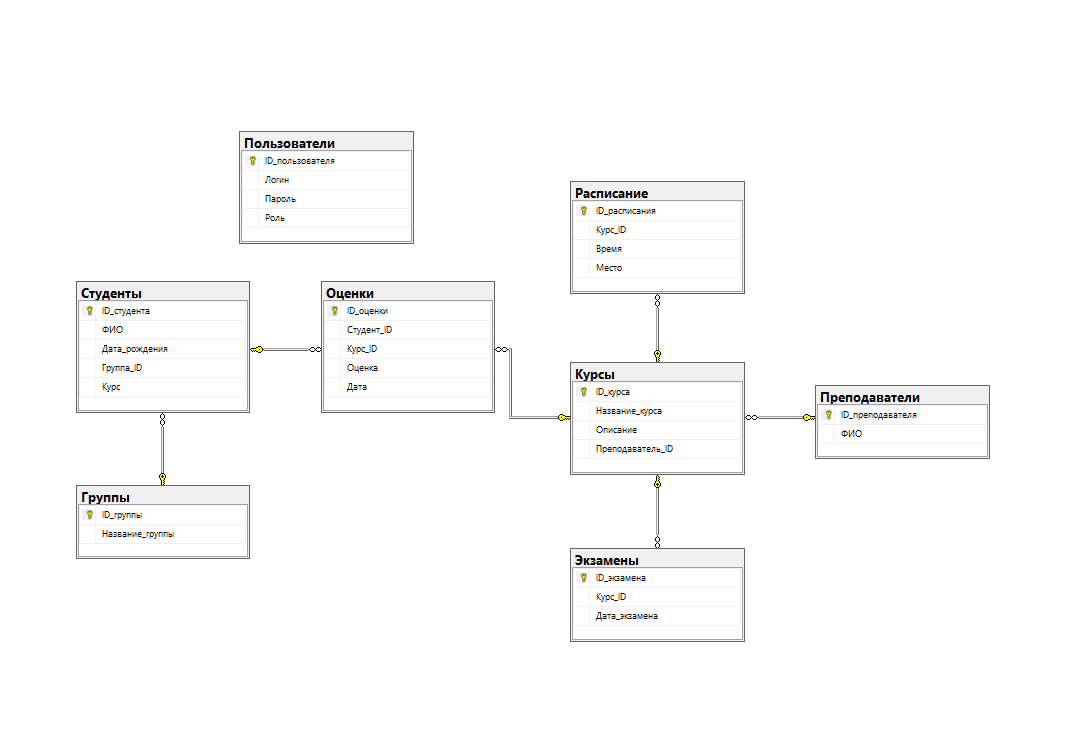


Рисунок 6. База данных.

* 1. **Реализация бизнес-логики**

Для системы "Деканат университета" важным аспектом является бизнес-логика, которая касается студентов, преподавателей, курсов и оценок. Например, при изменении курса, который уже был назначен студентам, система должна корректно обновлять все связанные данные, например, переназначать оценки.

Рассмотрим один из нюансов бизнес-логики: что происходит, если студент отменяет подачу на экзамен? В таком случае, система не удаляет данные, а меняет их состояние на "Отменено". Это позволяет сохранить всю историю действий и избежать неприятных ситуаций для студентов и преподавателей.

* 1. **Реализация механизмов регистрации и авторизации**

В данном веб-приложении реализация механизмов регистрации и авторизации построена на основе встроенной в ASP.NET Core 7.0 системы аутентификации и авторизации с использованием Cookie Authentication.

**Конфигурация системы аутентификации**

В файле Program.cs настроены основные сервисы аутентификации:

**Управление пользователями**

* Пользователи хранятся в таблице Users
* Каждый пользователь имеет определенную роль (Student, Teacher, Administrator)
* Пароли хранятся в зашифрованном виде
* Реализована базовая модель пользователя с необходимыми полями (email, имя, фамилия, роль)

**Процесс аутентификации**

При входе в систему:

* Проверяются учетные данные пользователя
* Создаются claims с информацией о пользователе
* Устанавливается аутентификационный cookie
* Пользователь перенаправляется на соответствующую его роли страницу

**Авторизация**

Реализована через:

* Атрибуты авторизации на контроллерах ([Authorize(Roles = "...")])
* Проверку ролей в представлениях
* Middleware авторизации в конвейере обработки запросов

**Основные компоненты**

* AccountController - управляет процессами входа/выхода
* Представления для входа в систему
* Обработка ошибок аутентификации

**Безопасность**

* Защита от подбора паролей
* Безопасное хранение учетных данных

**Процесс входа в систему**

* Пользователь вводит email и пароль
* Система проверяет учетные данные
* При успешной проверке:
* Создается набор claims
  + Происходит перенаправление на нужную страницу
* При неудаче показывается сообщение об ошибке

**Дополнительные функции**

* Перенаправление на запрошенную страницу после входа
* Возможность выхода из системы
* Защита маршрутов от неавторизованного доступа
* Такая реализация обеспечивает:
  + Надежную защиту данных
  + Удобное управление доступом
  + Гибкость в настройке прав
  + Простоту поддержки и расширения функционала
  + Система построена с учетом современных требований безопасности и лучших практик разработки веб-приложений на платформе ASP.NET Core.
  1. **Интеграция с внешними сервисами**

В рамках системы "Деканат университета" предусмотрена возможность интеграции с внешними сервисами для получения дополнительных данных, таких как расписание преподавателей, информация о экзаменах и другие данные. Эта интеграция будет использовать REST API или SOAP для получения актуальной информации.

* 1. **Интерфейс веб-приложения**

1. Домашняя страница

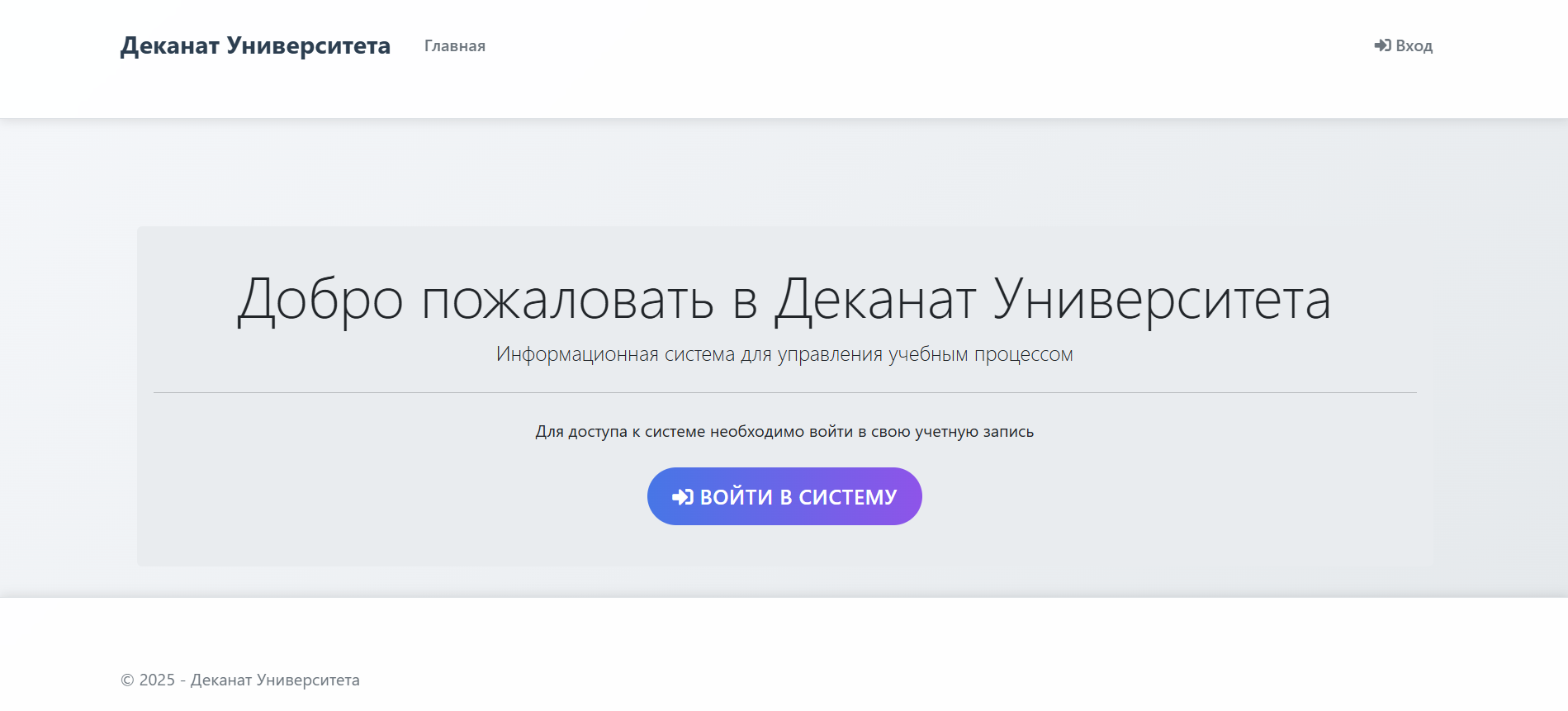


Рисунок 7. Домашняя страница.

1. Страница авторизации

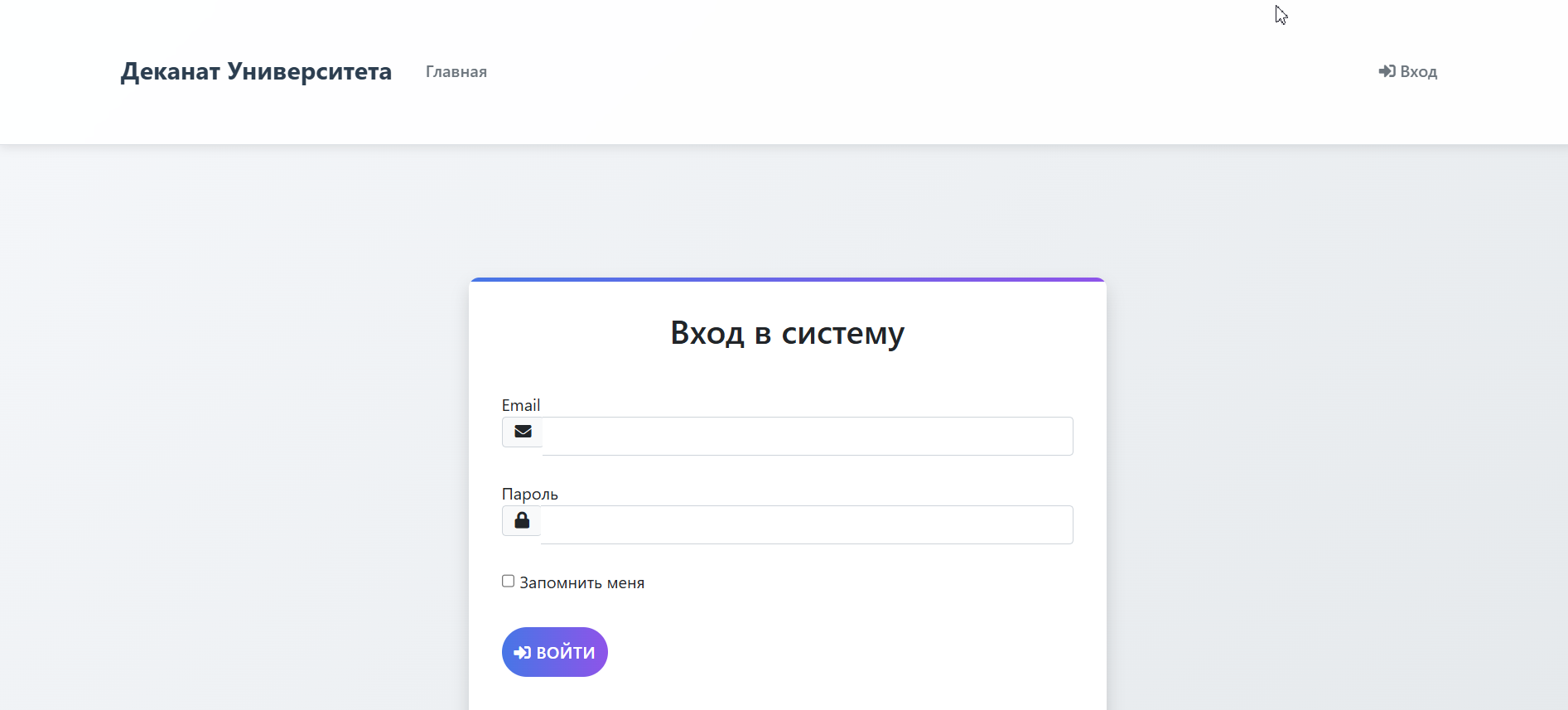


Рисунок 8. Страница авторизации

1. Интерфейс, доступный с ролью “Администратор”

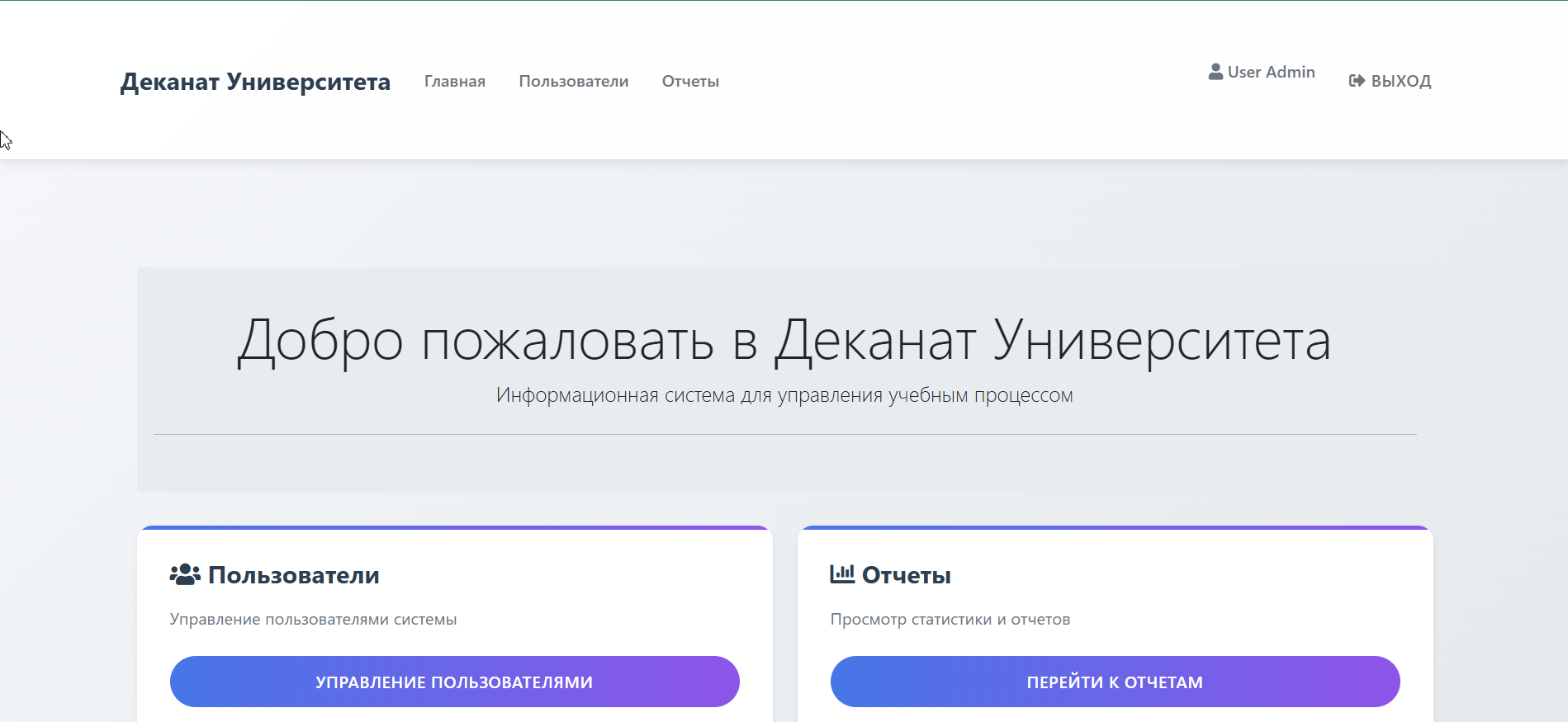


Рисунок 9. Доступный интерфейс с ролью “Администратор”

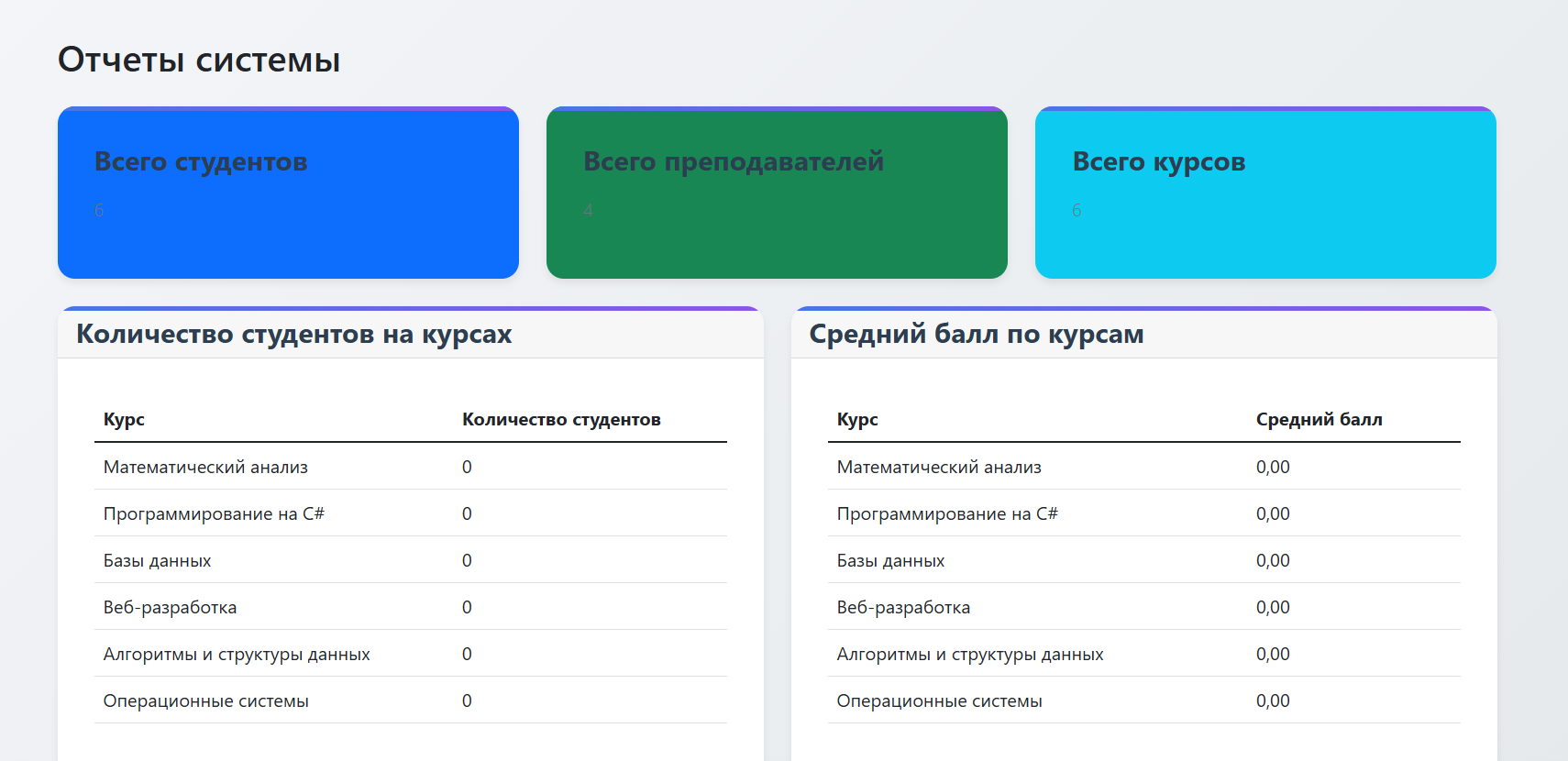


Рисунок 10. Интерфейс “Отчеты системы”, доступный для Администратора

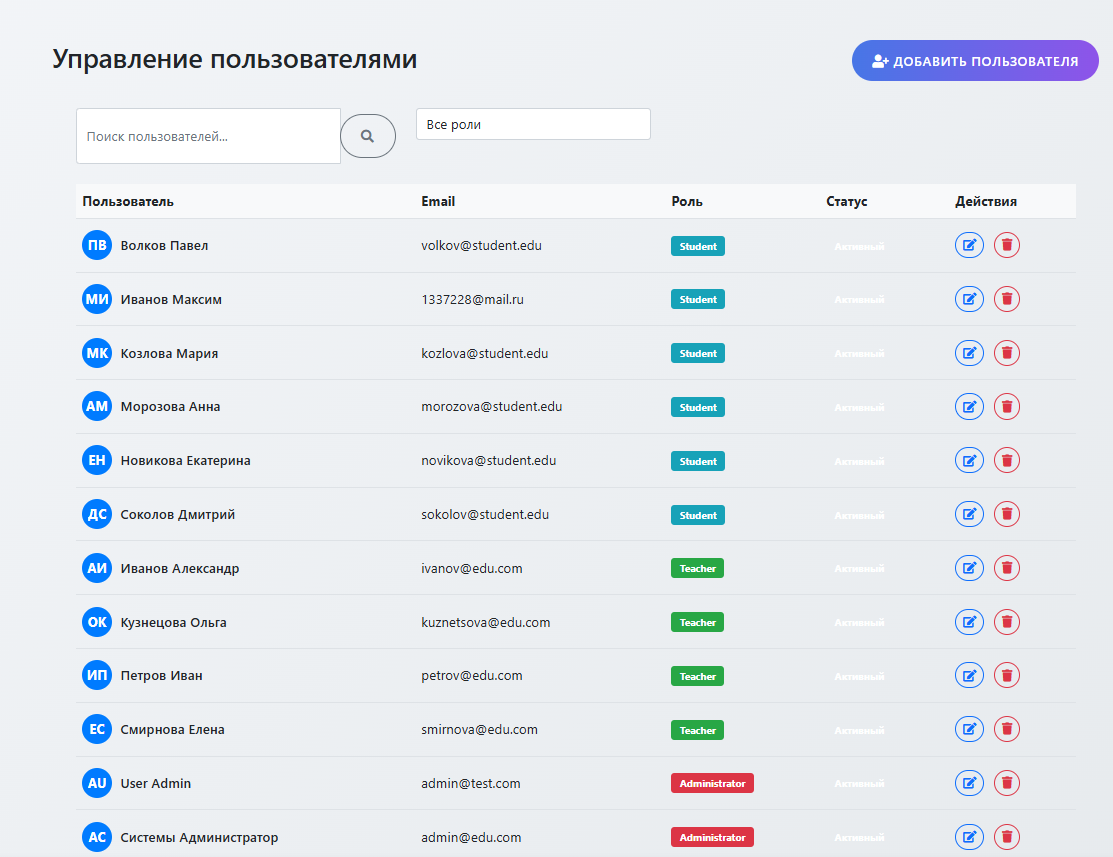


Рисунок 11. Интерфейс, доступный администратору, для создания и редактирования пользователей.

1. Интерфейс, доступный с ролью “Студент”.

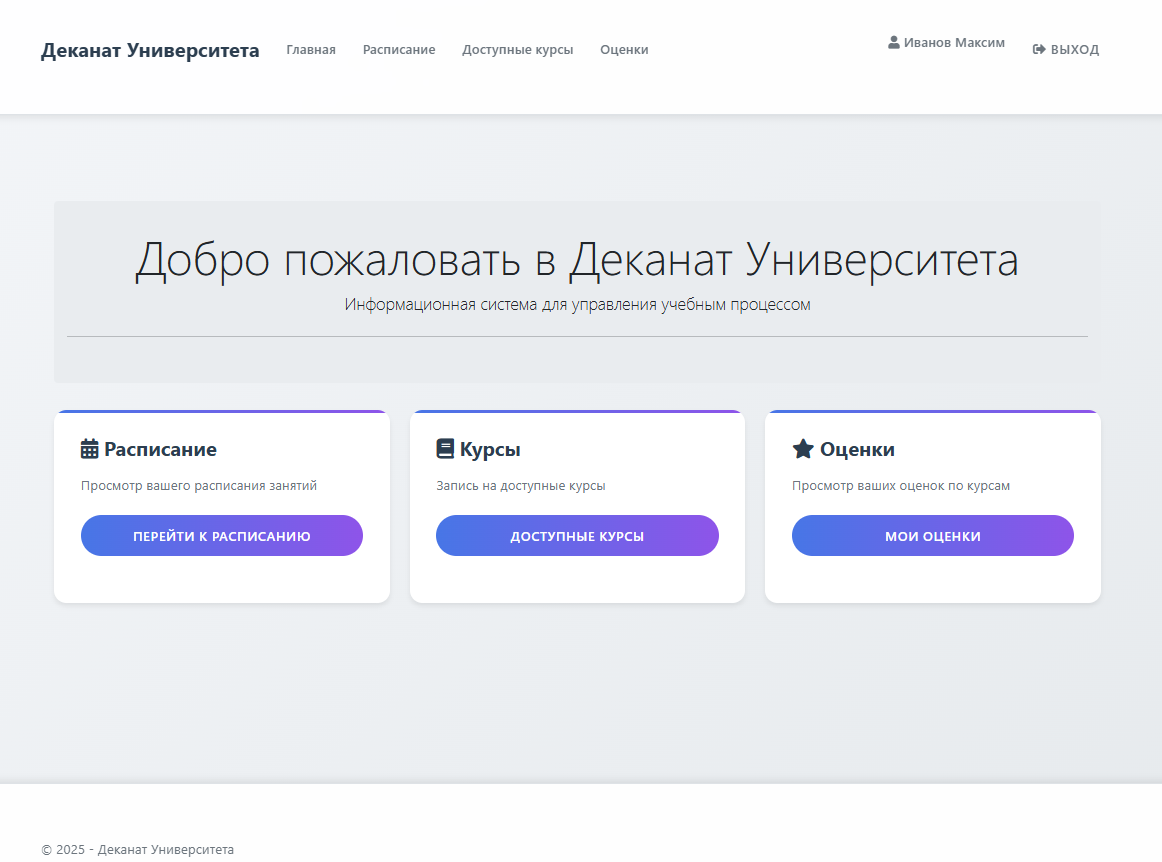


Рисунок 12. Интерфейс, доступный с ролью “Студент”.

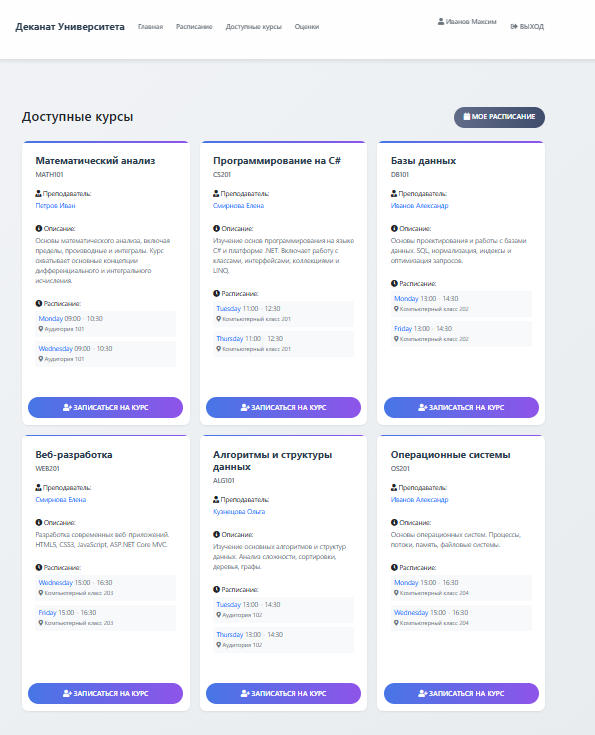


Рисунок 13. Интерфейс “Доступные курсы”, который предназначен для студентов.

1. Интерфейс, доступный для пользователей с ролью “Преподаватель”.

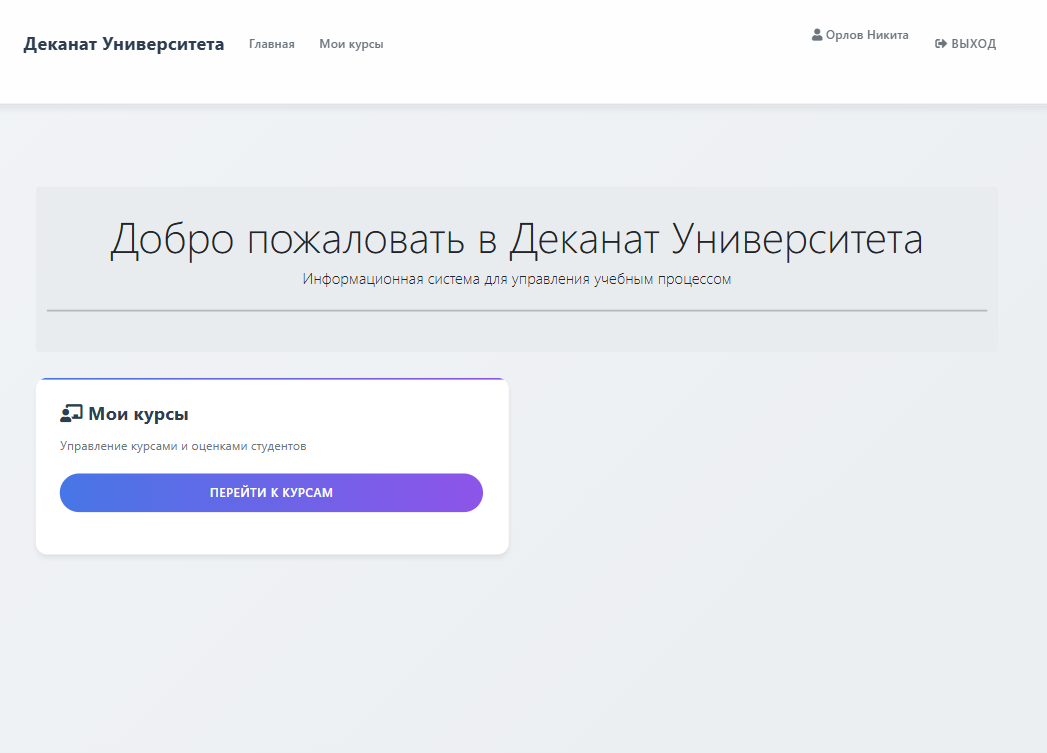


Рисунок 14. Интерфейс для преподавателей

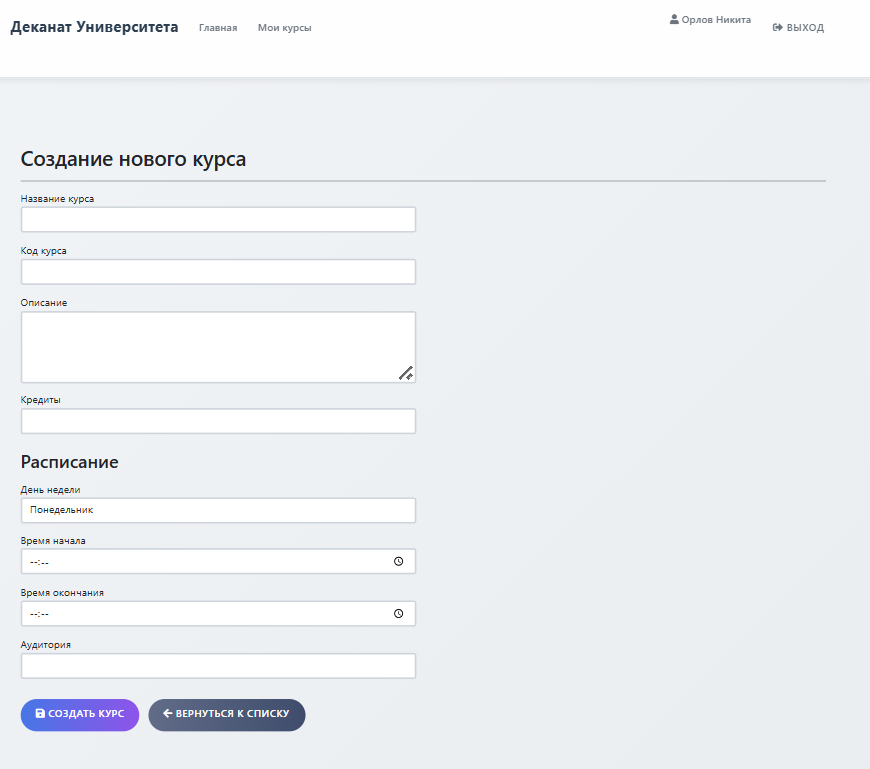


Рисунок 15. Форма создания нового курса для преподавателей

# заключение

В ходе выполнения курсового проекта была разработана и внедрена информационная система для управления учебным процессом на базе современной платформы ASP.NET Core с использованием архитектурного паттерна MVC. Веб-приложение реализует основные функции, необходимые для эффективной работы студентов, преподавателей и администрации: ведение расписания, управление курсами, учет оценок и ролевая авторизация пользователей. Благодаря использованию Entity Framework Core обеспечена надежная работа с базой данных и поддержка миграций, что позволяет легко расширять и модифицировать структуру данных.

В процессе разработки особое внимание уделялось вопросам безопасности и удобства использования. Для защиты данных реализованы механизмы аутентификации и авторизации, а также валидация пользовательского ввода на всех уровнях приложения. Интерфейс системы построен с применением современных веб-технологий, таких как Bootstrap и Razor, что обеспечивает адаптивность и интуитивную понятность для пользователей с различным уровнем подготовки.

В результате проделанной работы была создана гибкая и масштабируемая система, которую можно легко адаптировать под нужды различных образовательных учреждений. Архитектура приложения позволяет интегрировать дополнительные модули и внешние сервисы, а также реализовать новые функции без существенных изменений в существующем коде. Это делает систему перспективной для дальнейшего развития и внедрения в реальных условиях.

В заключение стоит отметить, что выполнение данного курсового проекта позволило закрепить на практике знания по современным технологиям веб-разработки, принципам построения безопасных и удобных информационных систем, а также навыки командной работы и проектирования архитектуры программных продуктов. Полученный опыт будет полезен для дальнейшего профессионального роста и решения более сложных задач в области разработки программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Козлов, В. А., Мищенко, В. И. Информационные системы и технологии в образовании. – М.: Издательский дом «КноРус», 2017. – 384 с.
2. Гильманова, Р. М., Юсупов, Р. В. Технология разработки программных систем. – СПб.: БХВ-Петербург, 2018. – 512 с.
3. Microsoft. Основы ASP.NET MVC. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/mvc/ (дата обращения: 07.04.2025).
4. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2016. – 416 с.
5. Шевченко, А. В. Разработка веб-приложений с использованием ASP.NET MVC. – Киев: Прогресс, 2019. – 360 с.
6. Entity Framework Documentation. Entity Framework Core. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/ef/core/ (дата обращения: 07.04.2025).
7. W3Schools. AJAX Tutorial. [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/xml/ajax\_intro.asp (дата обращения: 07.04.2025).
8. Информационные технологии в образовании. Современные информационные системы для управления учебным процессом. Журнал «Информатика и образование», 2020. – № 3. – С. 56-62.
9. SimpleMembership в ASP.NET MVC. Реализация системы безопасности в веб-приложениях. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/web-pages/overview/security/simple-membership (дата обращения: 07.04.2025).
10. Шилов, В. И. Методы и средства разработки программных систем. – М.: Наука, 2015. – 352 с.

ПРИЛОЖЕНИЕ А Исходный код программной системы

using Microsoft.EntityFrameworkCore;

using StudentManagementSystem.Data;

using Microsoft.AspNetCore.Authentication.Cookies;

using StudentManagementSystem.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromMinutes(30);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.LoginPath = "/Account/Login";

options.AccessDeniedPath = "/Account/AccessDenied";

});

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

using (var scope = app.Services.CreateScope())

{

var services = scope.ServiceProvider;

var context = services.GetRequiredService<ApplicationDbContext>();

context.Database.EnsureCreated();

if (!context.Users.Any(u => u.Email == "admin@test.com"))

ПРИЛОЖЕНИЕ А Исходный код программной системы

{

var adminUser = new User

{

FirstName = "Admin",

LastName = "User",

Email = "admin@test.com",

Password = "admin123",

Role = UserRole.Administrator,

Faculty = "Administration"

};

context.Users.Add(adminUser);

context.SaveChanges();

}

}

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

else

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "areas",

pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}");

app.MapControllerRoute(

ПРИЛОЖЕНИЕ А Исходный код программной системы

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

ПРИЛОЖЕНИЕ Б Структура таблиц БД

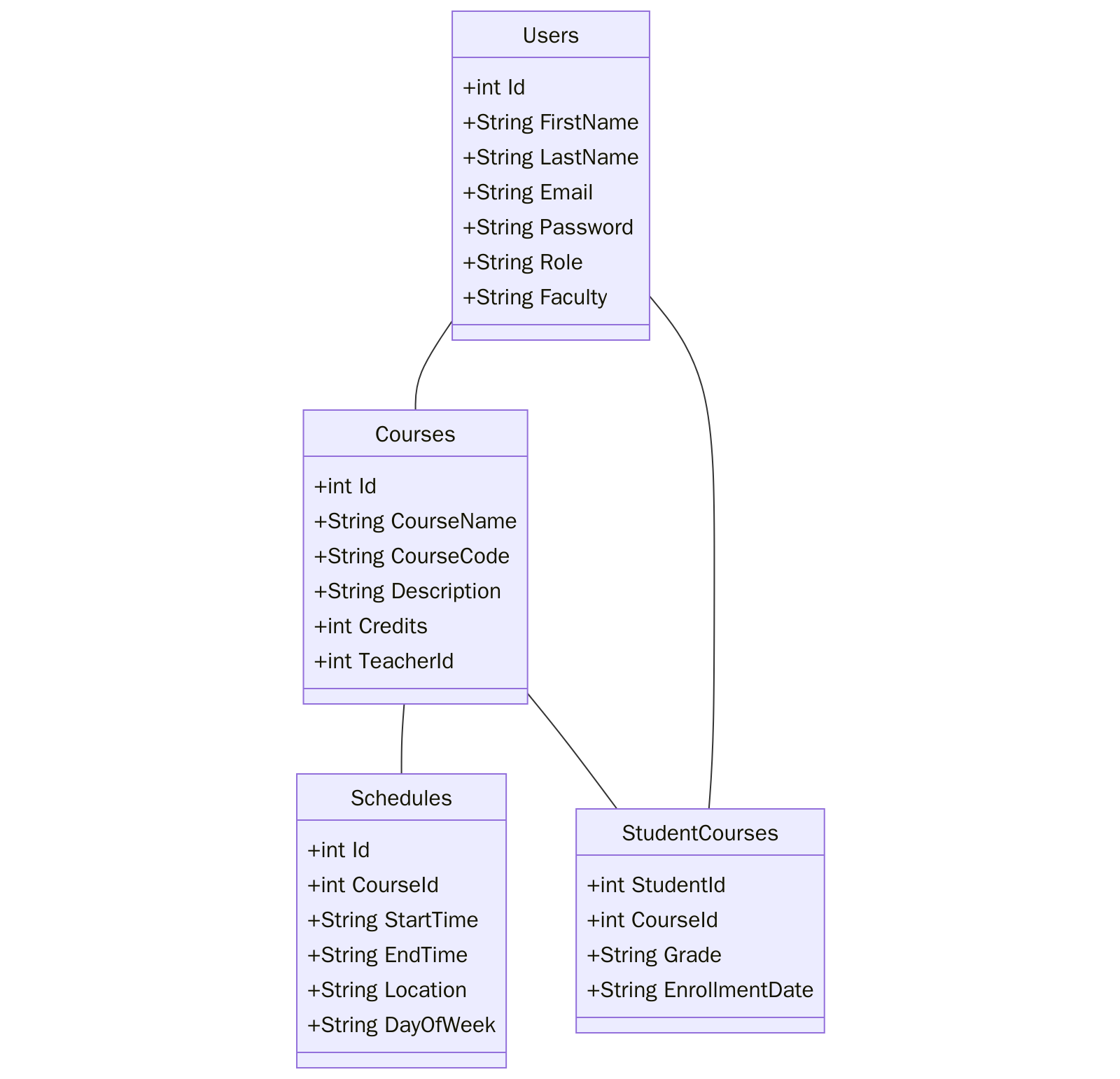


Рисунок Б.1. Структура базы данных.

**карман с диском на котором записаны материалы курсового проектирования:**

1. пояснительная записка

2. проект системы с исходными кодами

3. дополнительные компоненты системы, необходимые для ее развертывания

4. прочие вспомогательные источники

5. презентация по которой будет проходить защита работы

**Папка(файлик) для графического материала к пояснительной записке и сам графический материал в ней**