

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 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Информационная система контроля
учебного процесса

Студент _____ Гречишкина Маргарита Сергеевна
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

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

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

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

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Гречишкина Маргарита Сергеевна гр. 20ВИ1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Информационная система контроля учебного процесса

Тема утверждена приказом ПГУ № от « » 20

3. Руководитель работы к.т.н., доцент Князев Виктор Николаевич

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

Назначение: информационная система предназначена для оптимизации управления данными об успеваемости и посещаемости студентов кафедры ВУЗа в форме web-приложения

Основные реализуемые функции:

1) просмотр данных о посещаемости и успеваемости студентов;

2) редактирование данных о посещаемости;

3) редактирование данных об успеваемости;

4) составление отчетов по заданной статистике;

5) добавление данных в систему.

Среда выполнения: Windows 10 и выше для сервера, веб-браузер для клиента

Технологии разработки: RUP, ООП, MVC, Entity Framework

Языковые средства разработки: C#, UML, SQL, Linq

Инструментальные средства разработки: Microsoft SQL Server 2020, Microsoft Visual Studio 2022, Visual Paradigm, Project Libre, dbdiagram.io, draw.io

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование информационной системы

3) Разработка программных средств

4) Тестирование программных средств

5) 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	Оформление пояснительной записки	20%	23.05.2024 – 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 126 листов, 46 рисунков, 12 таблиц, 34 использованных источника, 6 приложений.

КОНТРОЛЬ УЧЕБНОГО ПРОЦЕССА, ИНФОРМАЦИОННАЯ СИСТЕМА, УЧЕТ ПОСЕЩАЕМОСТИ И УСПЕВАЕМОСТИ, WEB-ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, MS SQL SERVER, C#, ASP.NET CORE MVC

Объектом разработки является информационная система контроля учебного процесса в ВУЗе в форме web-приложения.

Цель работы – разработка информационной системы для контроля учебного процесса ВУЗа, которая позволяет оптимизировать учетные процессы, связанные с посещаемостью и успеваемостью студентов.

Разработка серверной части web-приложения производилась на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД MS SQL Server 2020.

Разработка клиентской части производилась на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022.

					ПГУ 09.03.02 – 08ВР201.17 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Информационная система контроля учебного процесса. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Гречишкина М.С.					4	126
Пров.		Князев В.Н				гр. 20ВИ1		
Реценз.								
Н. контр.		Такташкин Д.В						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Анализ информационных систем аналогичного назначения	10
1.2.1 Анализ ИС «ТАНДЕМ.университет».....	10
1.2.2 Анализ ИС «1С: Университет»	12
1.2.3 Анализ модуля «Электронный деканат»	13
1.2.4 Сравнение аналогов	14
1.3 Анализ целевой аудитории	14
1.4 Анализ функциональных требований	17
1.5 Анализ нефункциональных требований	21
1.6 Планирование разработки и оценка бюджета	22
2 Проектирование информационной системы	30
2.1 Выбор технологий и инструментальных средств разработки.....	30
2.2 Проектирование архитектуры программного обеспечения.....	31
2.3 Проектирование базы данных.....	32
2.3.1 Разработка концептуальной модели базы данных	32
2.3.2 Логическое моделирование базы данных.....	38
2.4 Проектирование пользовательского интерфейса	41
2.5 Проектирование серверной части приложения	47
3 Реализация информационной системы	48
3.1 Реализация базы данных	48
3.2 Разработка серверной части.....	49
4 Тестирование web-приложения	57
4.1 Функциональное тестирование.....	57
4.2 Модульное тестирование	58
4.3 Анализ кода.....	60
4.3.1 Индекс удобства поддержки.....	60
4.3.2 Метрики кода.....	61

Заключение	67
Список использованных источников	69
Приложение А (справочное) Глоссарий	73
Приложение Б (рекомендуемое) Исходный код создания базы данных.....	75
Приложение В (рекомендуемое) Исходный код контроллеров	84
Приложение Г (рекомендуемое) Исходный код моделей.....	95
Приложение Д (рекомендуемое) Исходный код представлений	105

Введение

Исторически, страны, в которых развитие информационных систем (ИС) поддерживается на государственном уровне, достигают значительных прорывов в этой области за счет целостного и регулируемого процесса внедрения новых разработок. Доказательством тому служит международный опыт реализации Окинавской хартии 2000 года. Подписавшие этот документ страны Большой Восьмерки G-8 за прошедшее время только укрепили свои лидерские позиции на мировой арене. Этот важный этап современной истории показал, что формирование «цифрового общества» и наращивание капитала информационно-коммуникационных технологий (ИКТ) неотделимы от устойчивого экономического развития, роста наукоёмких исследований и улучшения качества жизни граждан [1]. Отдельное внимание в документе было уделено укреплению человеческого потенциала: предоставлению широкого доступа к базовому образованию, подготовке технических специалистов, а также внедрению инновационной технической помощи (единый образовательный портал, электронный университет и т. д.) [2].

Современные исследователи соглашаются в следующем: «Необходимо внедрение современной цифровой базы образовательного процесса информационно-образовательной среды (ИОС). Реализация эффективной ИОС является основой развития любого Высшего учебного заведения (ВУЗа)» [3].

На сегодняшний день, в Российской Федерации придерживаются намеченного курса и продолжают вносить и поддерживать законодательные инициативы для создания благоприятных условий для информационного развития. Актуальным примером послужит Государственная программа «Цифровая экономика Российской Федерации» 2017 года, принятая с целью развития информационных технологий, цифровизации экономики и общества. Программа направлена на создание условий для развития современной инфраструктуры, цифровизации отраслей экономики, развития технологий и улучшения условий жизни населения [4]. Кроме того, предусмотрены меры по

обеспечению информационной безопасности, развитию цифровой грамотности и поддержке инновационной деятельности в сфере информационных технологий.

В документе «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы» описаны подробные шаги по реализации политики Российской Федерации, направленной на повсеместное применение ИКТ [5]. Цифровизация сферы образования в ней с нескольких сторон: от обеспечения условий для научно-технического творчества до популяризации русской культуры и науки за рубежом. В рамках данной работы остановимся на пункте 24, в котором обосновывается необходимость «создания условий для удовлетворения их (граждан) потребностей в постоянном развитии, получении качественных и достоверных сведений, новых компетенций». Создание информационной системы по контролю учебного процесса способствует исполнению данной задачи, так как увеличивает прозрачность и доступность образовательного процесса для студентов, а также упрощает ряд операционных задач для сотрудников.

Еще один документ, поощряющий использование технических средств в сфере образования – Федеральный закон «Об образовании в Российской Федерации» № 273 – ФЗ от 29.12.2012 года с изменениями 2023 года [6].

Актуальность данной выпускной квалификационной работы (ВКР) обусловлена всемирными тенденциями цифровизации общества, государственными инициативами и непредвиденными изменениями общественных порядков (глобальная пандемия обусловила усиленное внедрение технологий во все сферы жизни).

Таким образом, на основании приведенных выше факторов можно сделать вывод о том, что разработка эффективной информационной системы, которая позволит контролировать учет посещаемости и успеваемости студентов в высшем учебном заведении, весьма актуальна в условиях современного общества.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Создаваемая информационная система будет использоваться студентами и сотрудниками университета в целях управления данными о посещаемости и успеваемости студентов, а также анализа этих данных. Эти цели будут достигнуты путем создания отчетов с соответствующими данными, которые будут извлечены из базы данных.

Категориями пользователей данной системы являются студенты и сотрудники. Студенты смогут войти в систему и просмотреть посещаемость и успеваемость в соответствии со своими предметами. Преподаватели будут иметь тот же функционал, но для всех студентов и с возможностью внесения новых данных.

Ещё более расширенный набор функций предоставляется пользователям с правами администратора. Администраторы будут иметь доступ к данным о каждом студенте. Кроме того, должна быть реализована возможность создавать отчеты о посещаемости и успеваемости студентов, чтобы проанализировать прогресс.

Для поддержки достаточного количества отчетов необходимо хранить дополнительную информацию, связанную с устройством учебного процесса. Например, в базе данных должна храниться информация обо всех предметах, предлагаемых кафедрой, а также соответствующие программы, частью которых являются эти предметы. Это даст пользователям с соответствующим доступом возможность легко сформировать отчетность по необходимым критериям.

На основе анализа предметной области были выделены следующие функции, подлежащие реализации:

- регистрация пользователей и вход в систему;
- просмотр данных о посещаемости и успеваемости студентов;
- редактирование данных о посещаемости;
- редактирование данных об успеваемости;

- составление отчетов по заданной статистике;
- добавление данных в систему.

1.2 Анализ информационных систем аналогичного назначения

Система контроля учебного процесса представляет собой компьютерное программное обеспечение, предназначенное для хранения и управления данными о посещаемости и успеваемости. Она включает информацию о расписании занятий, данных студентов и преподавателей, а также об образовательных программах. Для контроля посещаемости занятий студентами и преподавателями были разработаны следующие программные инструменты:

- автоматизированный комплекс учета посещаемости и успеваемости студентов «ТАНДЕМ.Университет»;
- автоматизированная система на базе «1С: Предприятие 7.7», под названием «1С: Деканат»;
- «Электронный деканат», модуль для среды дистанционного обучения Moodle.

Рассмотрим каждое решение более подробно.

1.2.1 Анализ ИС «ТАНДЕМ.университет»

Автоматизированная система «ТАНДЕМ.Университет» содержит базу данных всех зачисленных студентов, электронные журналы посещаемости и успеваемости, а также инструменты для документирования и статистического анализа. Доступ к программе предоставляется исключительно зарегистрированным сотрудникам. Студенты заносятся в базу данных и распределяются по подгруппам в иерархическом справочнике, уровнями которого являются факультеты, группы и подгруппы. Это позволяет проводить выборки по отдельным группам, специальностям, курсам, факультетам или всему учебному заведению [7].

Примеры интерфейсов представлены на рисунках 1, 2.

Расписание групп Расписание аудиторий Расписание преподавателей Расписание мероприятий Календарное расписание групп Календарное расписание аудиторий Календарное расписание преподавателей Календарное расписание мероприятий События расписания									
Версия расписания 2022/2023 (черновик) (01.09.2022 - 31.08.2023)		Дата с 05.12.2022	Дата по 18.12.2022	Дни недели ☒ Пн ☒ Вт ☒ Ср ☒ Чт ☒ Пт ☐ Сб ☐ Вс					
<input type="button" value="Добавить событие"/> <input type="button" value="Составить автоматическое"/> <input type="button" value="Оценить оптимальность"/> <input type="button" value="Проверить события"/>				☐ Группировать по дням недели ☐ Выводить незаполненные блоки ☒ Отображать окна от 30 мин					
<input type="button" value="Филары по строкам"/>									
<input type="button" value="Филары по событиям"/>									
Расписание групп									
Группа	Понедельник, 05.12.2022 15 нед. (веч.)	Вторник, 06.12.2022 15 нед. (веч.)	Среда, 07.12.2022 15 нед. (веч.)	Четверг, 08.12.2022 15 нед. (веч.)	Пятница, 09.12.2022 15 нед. (веч.)	Суббота, 10.12.2022 15 нед. (веч.)	Понедельник, 16 нед. (веч.)		
ПОСО (Вас) 115	08:00 – 11:00 (180 мин)	08:00 – 11:00 (180 мин)	08:00 – 09:30 (90 мин)	08:00 – 09:30 (11 пара) Естественные Майорова З.А. ауд. сто-059 (Корп.4)	08:00 – 12:30 (270 мин)	08:00 – 12:30 (270 мин)	08:00 – 09:30		
			09:30 – 10:50 (2 пара) Профессиональное самоопределение Практическое занятие Хоников В.А. ауд. сто-059 (Корп.4)	09:30 – 10:50 (2 пара) Естественные Практическое занятие Майорова З.А. ауд. сто-059 (Корп.4)			09:30 – 10:50 ОБЖ Практическое занятие Вуно И.О. ауд. сто-059 (Корп.4)		
	11:00 – 12:20 (3 пара) Математика Практическое занятие Филатова Н.А. ауд. сто-059 (Корп.4)	11:00 – 12:20 (3 пара) Физическая культура Практическое занятие Хоников В.А. ауд. сто-04 (Корп.4)	11:00 – 12:20 (3 пара) Математика Практическое занятие Филатова Н.А. ауд. сто-059 (Корп.4)	11:00 – 12:20 (3 пара) Инструментальный язык Практическое занятие Майорова З.А. ауд. сто-059 (Корп.4)			11:00 – 12:20 Математика Практическое занятие Филатова Н.А. ауд. сто-059 (Корп.4)		
	12:20 – 14:00 (100 мин)	12:30 – 13:50 (4 пара) Физическая культура Практическое занятие Хоников В.А. ауд. сто-04 (Корп.4)		12:30 – 13:50 (4 пара) Инструментальный язык Практическое занятие Майорова З.А. ауд. сто-059 (Корп.4)	12:30 – 13:50 (4 пара) Литература Занятия Филатова Н.А. ауд. сто-059 (Корп.4)	12:30 – 13:50 (4 пара) Математика Занятия Хоников В.А. ауд. сто-059 (Корп.4)	12:20 – 14:00		
	14:00 – 15:20 (5 пара) География				14:00 – 15:20 (5 пара) Литература	14:00 – 15:20 (5 пара) Математика	14:00 – 15:20 География Практическое занятие		

Рисунок 1 – Календарное расписание группы в ИС «ТАНДЕМ.Университет»

Версия дисциплины: Цифровая трансформация бизнеса (4 з.е., 144 ч., частей 1, ИТвЭ, Д № 482)

Версия дисциплины

УП

РУП

Шкалы оценок по КМ

Лог

Данные для расписания

Отправить в архив

Отклонить

Равнозначить

Создать копию версии дисциплины

Изменить подразделение

Удалить

Дисциплина

Номер

Состояние

Название

Сокращенное название

Полное название

Перечень направлений

Дополнительные характеристики

Комментарий

Цифровая трансформация бизнеса (Д № 482)

482

Согласовано

Цифровая трансформация бизнеса

Цифровая трансформация бизнеса

Цифровая трансформация бизнеса

Перечень направлений магистратуры 2013

Данные версии дисциплины

Тип (группа)

Подразделение

Способ деления потоков

Ограничение деления по виду потока

Дисциплины

Кафедра информационных технологий в экономике

Добавить файл

Список приложенных файлов

№

Имя файла

Дата добавления

Тип приложенного файла

Описание

Элементов в списке 5

Нагрузка

Читающее подразделение

ИТвЭ

Труд-сть

Лекции

Лаб.зан-я

Практика

Семинары

Сам. раб.

Контроль

Всего

Аудитор-

Самост.

Коллаб.

Э

Цифровая трансформация бизнеса

Цифровая трансформация бизнеса (4 з.е., 144 ч., 1/1)

Цифровая трансформация бизнеса (Иб. 482, 1/1)

4

18

36

54

36

54

36

144

54

54

54

1

4

18

36

54

36

54

36

144

54

54

54

1

4

18

36

54

36

54

36

144

54

54

54

1

- цена внедрения программного продукта довольно высока, учитывая невозможность самостоятельного конфигурирования и долгосрочной поддержки ИС – всякое изменение потребует привлечения сотрудников «ТАНДЕМА»;
- перегруженные интерфейсы с множеством излишних функций усложняют взаимодействие с пользователем и требуют дополнительных временных и финансовых затрат на обучение сотрудников.

1.2.2 Анализ ИС «1С: Университет»

Автоматизированная информационная система (АИС) «1С: Университет», созданная на базе «1С: Предприятие», спроектированная с целью решения ряда учетных задач, таких как «ведение базы данных студентов, учет экзаменов и зачетов, подготовка справок и выписок результатов, расчет платы за обучение» [8]. На рисунке 3 представлен пример интерфейса АИС.

Основное удобство этой автоматизированной системы учета для образовательных учреждений заключается в сетевой версии «1С: Бухгалтерия 7.7», которая присутствует на всех компьютерах бухгалтерии и администрации.

Сводная ведомость успеваемости

Вариант отчета: Основная

Сформировать Настройка...

Начало этого дня: Уфимский колледж рекламы и пиара

Все действия ?

Сводная ведомость по успеваемости

Параметры данных: Период = 14.12.2010 0:00:00
Организация = Уфимский колледж рекламы и пиара

Организация	1 семестр 2009/2010 г.		1 семестр 2009/2010 г.		1 семестр 2009/2010 г.		1 семестр 2009/2010 г.		1 семестр 2009/2010 г.		1 семестр 2009/2010 г.
Группа	Работа в творческой мастерской		Безопасность жизнедеятельности		Видеосъемка		Инженерная графика		Иностранный язык		Математика
Студент	Практика		Зачетная за семестр		Зачетная за семестр		Экзамен семестровый		Зачетная за семестр		Зачет
	Оценка	Зачет	Оценка	Зачет	Оценка	Зачет	Оценка	Зачет	Оценка	Зачет	Оценка
Уфимский колледж рекламы и пиара	3,83		3,72		3,52		3,68		3,60		
P-01/2008	3,83		3,72		3,52		3,68		3,60		
Алтынбаев Руслан Тагирович	4,0		3,0		5,0		4,0		3,0		Не
Арсланова Юлия Александровна	5,0		5,0		5,0		5,0		5,0		
Ахметов Антон Константинович	4,0		4,0		3,0		3,0		4,0		

Рисунок 3 – Скриншот АИС «1С:Университет»

Такие системы обладают следующими преимуществами:

- системы могут работать как локально, так и с помощью компьютеров при помощи Интернета;
- система разделения прав гарантирует, что каждый пользователь имеет доступ только к своим собственным данным.

Однако такая система обладает следующими минусами:

- многие программы для учета посещаемости имеют узкую направленность, что затрудняет их эксплуатацию и внедрение;
- программа приобретается в едином скопированном файле, поэтому изменения в ней невозможны, так как исходные материалы отсутствуют.

1.2.3 Анализ модуля «Электронный деканат»

Free Dean's Office (Электронный деканат) – это модуль для среды дистанционного обучения Moodle, который добавляет возможность управления процессом обучения, типичным для российских школ, колледжей и ВУЗов. «Free Dean's Office позволяет оперировать такими объектами, как «Специальность», «Дисциплина», «Курс», «Академическая группа», «Семестр», «Учебный план слушателя», «Нагрузка преподавателя», «Итоговые оценки по дисциплинам», «Расписание», «Текущие оценки и посещаемость», «Журнал успеваемости и посещаемости», «Зачетная книжка», «Табельный номер преподавателя» и т. д.» [9].

К преимуществам данной системы можно отнести удобный и конфигурируемый интерфейс и доступность – Free Dean's Office распространяется под лицензией GNU GPL в виде свободного программного обеспечения. Другими словами, продукт бесплатен и открыт для модификаций.

Однако то, что с одной стороны покажется преимуществом, так же может быть рассмотрено как недостаток. В данном контексте, ответственность за конфигурацию и внедрение свободного кода целиком лежит на сотрудниках учебного заведения, что требует дополнительной технической подготовки и трудно прогнозируемых затрат по времени.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше систем учета учебного процесса представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Название	Функционал	Удобство использования	Простота внедрения	Сопровождение
ТАНДЕМ.Университет	Избыточный, требует затрат на изучение	Перегруженный, контр-интуитивный интерфейс	Требует привлечения разработчиков	Требует привлечения разработчиков
1С:Университет	Избыточный, требует затрат на изучение	Избыточный, не подлежит конфигурации	Требует привлечения разработчиков	Требует привлечения разработчиков
Электронный деканат	Достаточный, но требует затрат на конфигурацию	Относительно удобный интерфейс	Усложнено необходимостью самостоятельной настройки плагинов	Отсутствует
Разрабатываемая ИС	Достаточный	Удобный и интуитивный интерфейс	Не требует дополнительных затрат	Не требует дополнительных затрат

По итогам сравнительного анализа систем учета посещаемости и успеваемости, было принято решение о создании собственной информационной системы контроля учебного процесса, оптимально соответствующей функциональным и нефункциональным требованиям, описанным в следующих разделах, а также лишенной недостатков рассмотренных аналогов.

1.3 Анализ целевой аудитории

Целевую аудиторию разрабатываемой системы составляют студенты и сотрудники университета: две категории пользователей с разными потребностями и ожиданиями от конечного продукта. Согласно актуальным публикациям, пользователи разных поколений выделяют разные требования к

программному обеспечению (ПО) [10]. Для студентов ВУЗов актуальными требованиями будут мобильность и минимальный дизайн, что объясняется тенденцией к широкому распространению смартфонов в данной возрастной группе. Статистика распределения возраста пользователей смартфона представлена на рисунке 4.

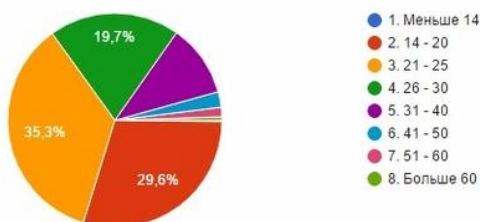


Рисунок 4 – График возраста пользователей смартфона

Для преподавателей и администраторов же предпочтительны будут высокая производительность и безопасность приложений, используемых на рабочих и персональных компьютерах.

Кроме того, при обращении к международной статистике использования интернета за 2023 – 2024 года, представленной на рисунке 5, видно, что несмотря на огромный процент пользователей, использующих мобильные телефоны для доступа к сети (96.3%), использование ноутбуков и персональных компьютеров остается достаточно востребованным средством доступа (62.2%) [11].

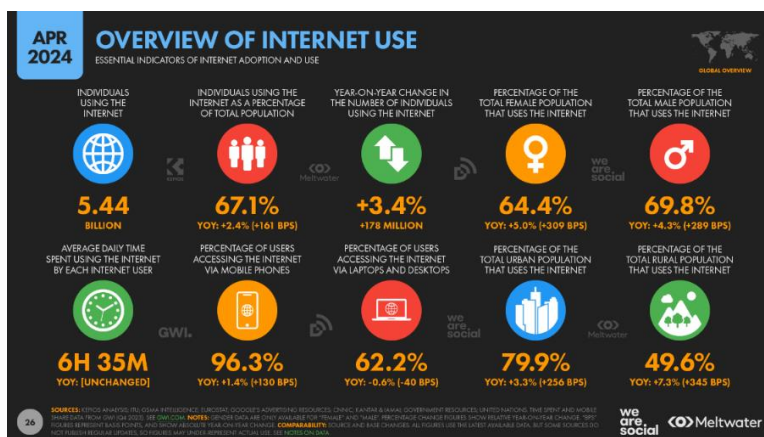


Рисунок 5 – Обзор использования интернета

В свете вышеперечисленных преимуществ, разработка web-приложения для системы учета учебного процесса является более предпочтительным вариантом по сравнению с десктопными или мобильными приложениями. Она обеспечивает широкую доступность и удобство для всех групп пользователей. Кроме того, нельзя оставить без внимания простоту управления и обновлений для разработчиков, а также экономическую эффективность, что делает web-приложение оптимальным выбором для учебных заведений.

Дополнительно следует рассмотреть распространение интернет-браузеров. Выбор наиболее популярных средств позволит значительно повысить доступность разрабатываемой системы.

На рисунке 6 представлен результат исследования распространенности браузеров по России за 2023 год.

Распространение браузеров в Российской Федерации (все устройства, первая половина 2023 года)

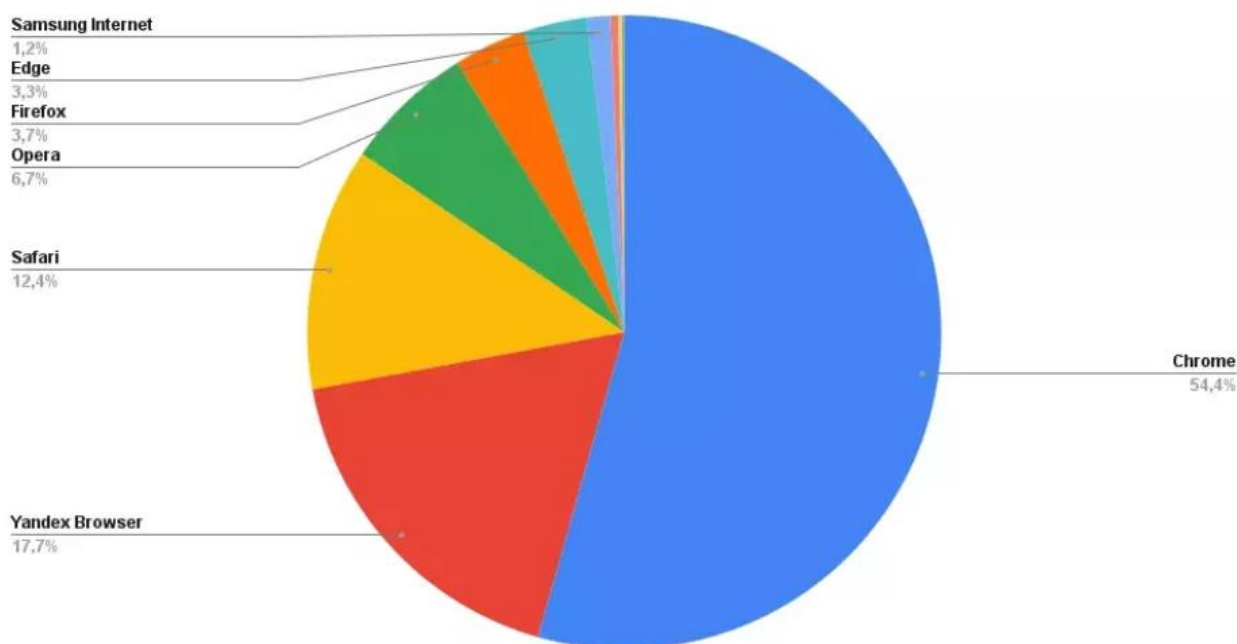


Рисунок 6 – Статистика популярности браузеров

Исследователи пришли к выводу, что «на сегодняшний день в числе лидеров находятся Google Chrome, Yandex, Safari, Opera и Mozilla Firefox» [12]. Совместимость именно с этими браузерами будет приоритетной в разработке современного веб-приложения.

1.4 Анализ функциональных требований

Разрабатываемое приложение главным образом предназначено для хранения, обработки и представления данных о посещаемости и успеваемости студентов ВУЗа, что значительно упростит процесс учёта учебного процесса.

На основе анализа предметной области и существующих решений были выделены следующие функции, подлежащие реализации:

- регистрация пользователей и вход в систему;
- просмотр данных о посещаемости и успеваемости студентов;
- редактирование данных о посещаемости;
- редактирование данных об успеваемости;
- составление отчетов по заданной статистике;
- добавление данных в систему.

Диаграмма вариантов использования (Use Case Diagram) – это графическое представление, которое отображает взаимодействие между актерами (пользователями системы, другими системами или внешними компонентами) и самой системой в рамках определенных сценариев использования. Она помогает визуализировать функциональные требования к системе и ее внешние связи.

Диаграмма вариантов использования, составленная на основе выделенных требований, изображена на рисунке 7.

Созданная диаграмма описывает создаваемую систему на концептуальном уровне. Было выделено 4 актера:

- студент, просматривающий соответствующие данные;
- СУБД, хранящая данные о системе;
- преподаватель, с частичными правами модификации данных;
- администратор, обладающий наибольшим функционалом по управлению данными.

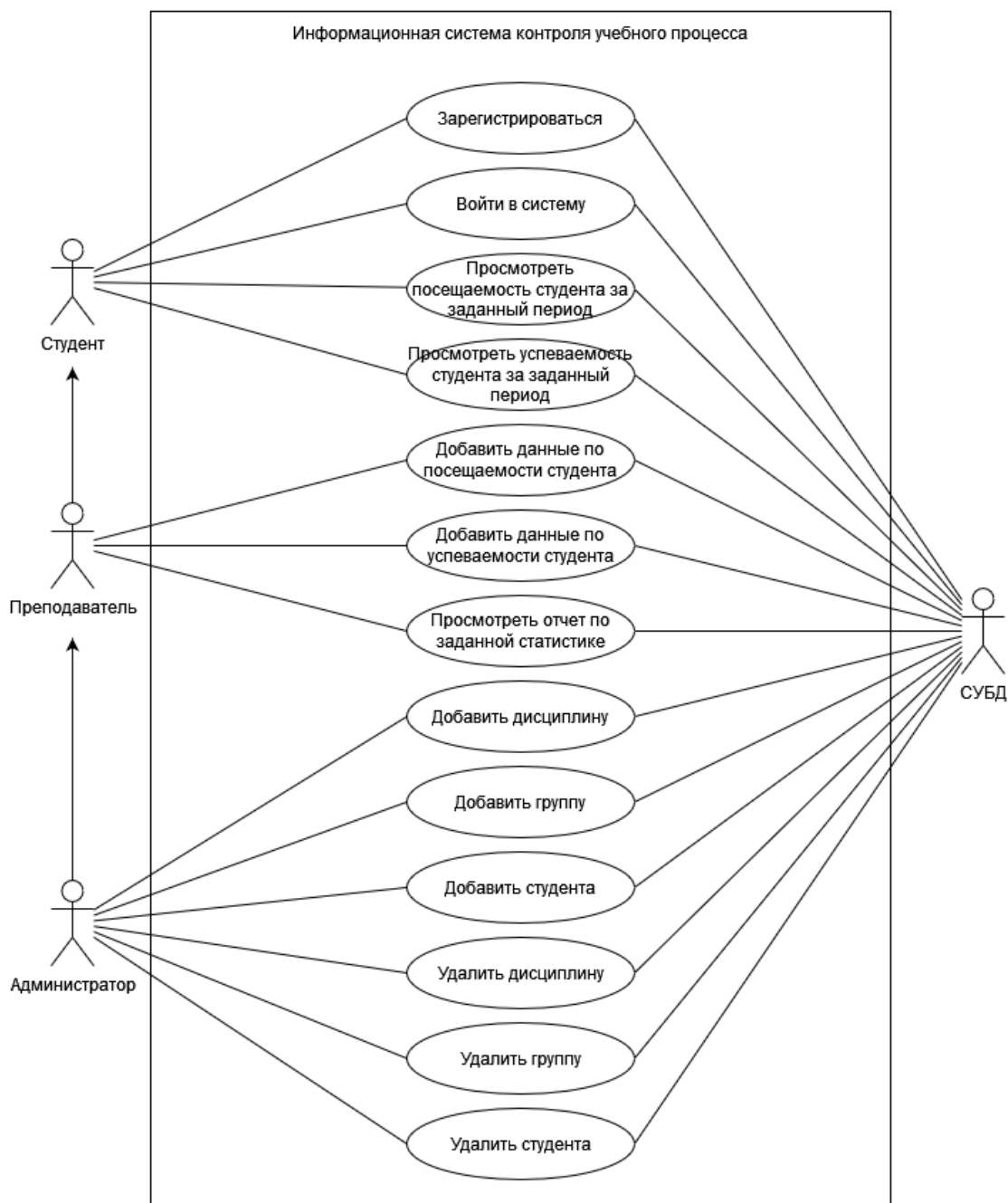


Рисунок 7 – Диаграмма вариантов использования

Приведем спецификацию вариантов использования «Войти в систему» и «Просмотреть посещаемость за заданный период», которые изображены на рисунках 8 и 9 соответственно.

Войти в систему
Id: 1
Описание: Система принимает и обрабатывает данные, введённые пользователем и авторизовывается в приложении
Действующие лица: пользователь, система
Предусловие: пользователь открыл ИС
Основной поток: <ol style="list-style-type: none"> 1. Система открывает форму для входа в приложение 2. Если пользователь регистрировался ранее: <ol style="list-style-type: none"> a. Пользователь вводит логин и пароль b. Пользователь нажимает кнопку «Войти» c. Система предоставляет доступ к основной форме 3. Если пользователь не зарегистрирован: <ol style="list-style-type: none"> a. Пользователь нажимает кнопку «Зарегистрироваться» b. Система открывает форму для регистрации c. Пользователь вводит личные данные для регистрации d. Пользователь нажимает кнопку «Завершить регистрацию» e. Выполняются пункты 1-2
Постусловие: вход выполнен

Рисунок 8 – Спецификация варианта использования «Войти в систему»

Просмотреть посещаемость за заданный период
Id: 2
Описание: Администратор должен посмотреть посещаемость учебных программ, дат, данных студентов.
Действующие лица: Администратор, система
Предусловие: Учебные программы, даты, данные студентов и данные преподавателей загружены в систему
Основной поток: <ol style="list-style-type: none"> 1. Администратор выбирает учебные программы, даты, студентов и преподавателей для просмотра. 2. Система возвращает результат на основе выбранных данных. 3. При необходимости администратор может вносить корректировки.
Постусловие: Администратор получил данные о посещаемости студентов за заданный период.

Рисунок 9 – Спецификация варианта использования «Просмотреть посещаемость за заданный период»

Для более наглядного рассмотрения требований, на рисунке 10 приведена диаграмма деятельности для первого прецедента. Пользователь открывает приложение и, если он зарегистрирован, вводит свои данные для входа. Если же пользователь не зарегистрирован, направляет его к регистрации. В случае, если пользователь зарегистрирован, приложение отправляет введенные данные на сервер через API. Сервер получает запрос, ищет пользователя в базе данных и формирует ответ. Если пользователь найден, открывается главный экран приложения. Если пользователь не найден, отображается сообщение об ошибке.

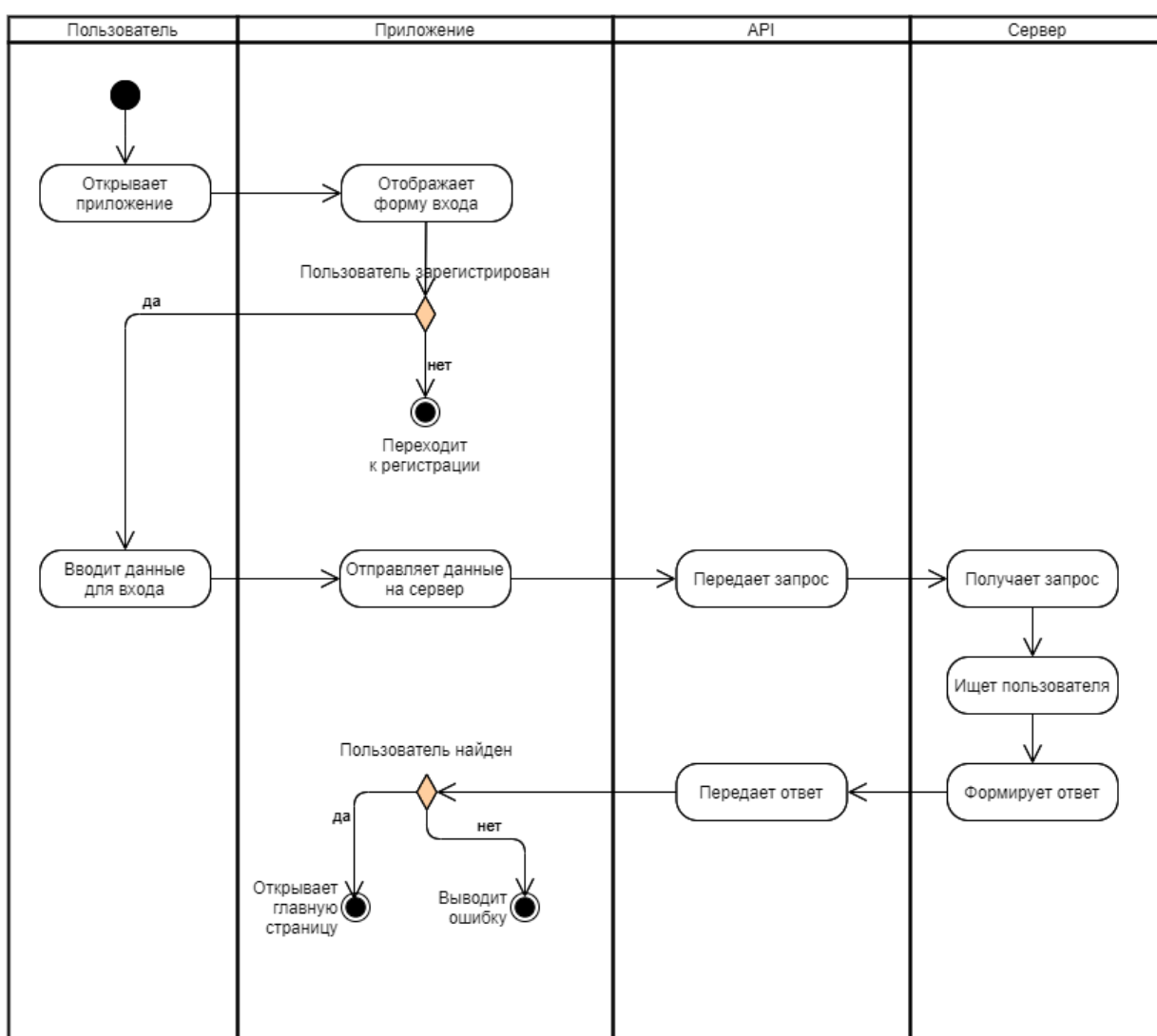


Рисунок 10 – Диаграмма деятельности для прецедента «Войти в систему»

Для прецедента «Просмотреть данные о посещаемости» была построена диаграмма последовательности, представленная на рисунке 11. Диаграмма

иллюстрирует, как данные проходят через различные уровни системы от пользователя до сервера и обратно.

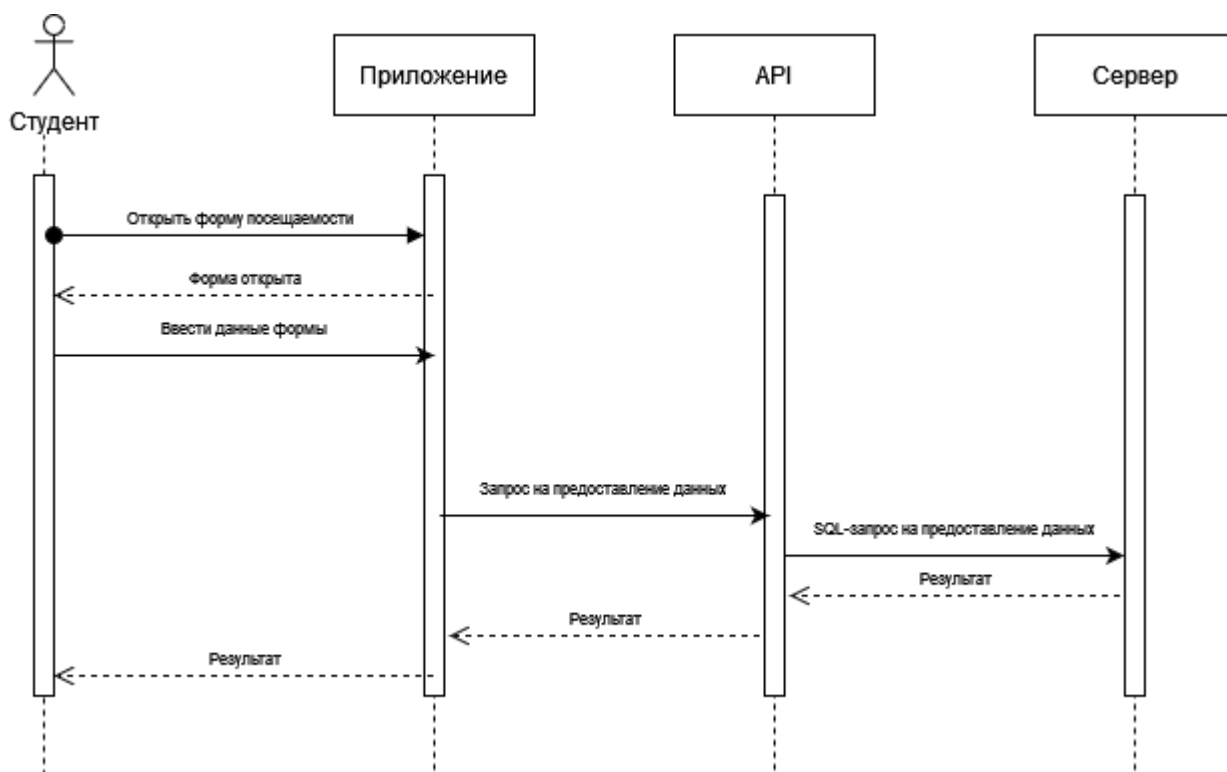


Рисунок 11 – Диаграмма последовательности для прецедента «Просмотреть данные о посещаемости»

Безусловно, здесь представлены требования не для всех возможных сценариев использования системы, но этого должно быть достаточно для общего понимания её функционала.

1.5 Анализ нефункциональных требований

В дополнение к функциональным требованиям к веб-приложению, также могут быть установлены и нефункциональные требования:

- usability (удобство использования), интерфейс пользователя должен быть интуитивно понятным и простым в использовании, с минимальным количеством кликов для выполнения основных задач, таких как регистрация или просмотр оценок [13];
- поддержка браузеров, веб-приложение должно корректно работать в последних версиях основных браузеров: Google Chrome и Mozilla Firefox;

– скорость работы, приложение должно обладать высокой производительностью и быстрой загрузкой страниц для обеспечения комфортной работы пользователя.

1.6 Планирование разработки и оценка бюджета

Качественное выполнение работ в рамках любого проекта, в том числе и в сфере информационных технологий, невозможно при отсутствии грамотного планирования. План, если не гарантирует, то кратно повышает вероятность своевременного достижения поставленных целей, помогает оптимизировать ресурсы (материальные, трудовые и временные) и повышает прозрачность процесса разработки [14 – 18].

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Поиск литературы	2 дня	1
4	Сравнительный анализ ИС контроля учебного процесса	4 дня	3
5	Выделение функциональных требований	1 день	4
6	Выделение нефункциональных требований	1 день	5
7	Консультация с представителем ВУЗа	1 день	6
8	Принятие окончательного решения	1 день	7
9	Анализ требований завершен	-	8
10	Выбор средств реализации	-	
11	Анализ возможных программных средств	1 день	1

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
12	Выбор программных средств для реализации приложения	2 дня	9
13	Выбор средств завершен	-	12
14	Проектирование приложения	-	
15	Проектирование архитектуры приложения	4 дня	9; 13
16	Создание макета пользовательского интерфейса	3 дня	15
17	Проектирование базы данных	7 дней	15
18	Проектирование приложения завершено	-	16; 17
19	Реализация проекта	-	
20	Реализация базы данных	5 дней	17
21	Реализация серверной части приложения	10 дней	18; 20
22	Реализация клиентской части приложения	10 дней	16;20
23	Реализация проекта завершена	-	21; 22
24	Контроль качества программного продукта	-	
25	Тестирование серверной части приложения	5 дней	21
26	Тестирование клиентской части приложения	10 дней	22
27	Исправление ошибок	10 дней	25; 26
28	Отладка завершена	-	27
29	Внедрение приложения	-	
30	Внесение данных в систему	1 день	28
31	Внедрение программного продукта в эксплуатацию	3 дня	30

Окончание таблицы 2

№	Название задачи	Длительность	Предшественники
32	Поиск и исправление ошибок при внедрении	7 дней	31
33	Внедрение завершено	-	32
34	Конец проекта	-	33

Таким образом, для реализации проекта нужно выполнить 34 задачи.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	450 Р/ч	550 Р/ч
Программист2	Трудовой	350 Р/ч	450 Р/ч
Программист3	Трудовой	400 Р/ч	450 Р/ч
Дизайнер	Трудовой	300 Р/ч	350 Р/ч
Интернет	Затраты		
Компьютер	Материальный		

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Поиск литературы	Программист1 Компьютер Интернет	50% 1 шт
4	Сравнительный анализ ИС контроля учебного процесса	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
5	Выделение функциональных требований	Программист1 Компьютер Интернет	50% 1 шт

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
6	Выделение нефункциональных требований	Программист1	50%
		Программист2	50%
		Программист3	50%
7	Консультация с представителем ВУЗа	Программист1 Компьютер Интернет	50% 1 шт
8	Принятие окончательного решения		
9	Анализ требований завершен		
10	Выбор средств реализации	Программист1	50%
		Программист2	50%
		Программист3	50%
		Компьютер	1 шт
		Интернет	
11	Анализ возможных программных средств	Программист1	50%
		Компьютер	1 шт
		Интернет	
12	Выбор программных средств для реализации приложения	Программист1	50%
		Программист2	25%
		Компьютер	2 шт
		Интернет	
13	Выбор средств завершен	Программист1	50%
		Программист2	25%
		Компьютер	2 шт
		Интернет	
14	Проектирование приложения	Программист1	25%
		Программист2	25%
		Программист3	100%
		Дизайнер	50%
		Компьютер	шт.
		Интернет	
15	Проектирование архитектуры приложения	Программист1	75%
		Программист2	75%
		Программист3	75%
		Компьютер	3 шт.
		Интернет	

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
16	Создание макета пользовательского интерфейса	Программист1 Программист2 Программист3 Дизайнер Компьютер Интернет	25% 25% 100% 50% шт.
17	Проектирование базы данных	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 25% 3 шт.
18	Проектирование приложения завершено		
19	Реализация проекта		
20	Реализация базы данных	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 75% 3 шт.
21	Реализация серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 75% 3 шт.
22	Реализация клиентской части приложения	Программист2 Программист3 Компьютер	50% 50% 1 шт.
23	Реализация проекта завершена		
24	Контроль качества программного продукта		
25	Тестирование серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	25% 25% 75% 3 шт.

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
26	Тестирование клиентской части приложения	Программист1	75%
		Программист2	75%
		Программист3	75%
		Компьютер	3 шт.
		Интернет	
27	Исправление ошибок	Программист1	75%
		Программист2	75%
		Программист3	25%
		Компьютер	3 шт.
		Интернет	
28	Отладка завершена		
29	Внедрение приложения	Программист1	50%
		Компьютер	1 шт
		Интернет	
30	Внесение данных в систему	Программист1	50%
		Компьютер	1 шт
		Интернет	
31	Внедрение программного продукта в эксплуатацию	Программист1	75%
		Программист2	75%
		Программист3	25%
		Компьютер	3 шт.
		Интернет	
32	Поиск и исправление ошибок при внедрении	Программист1	50%
		Компьютер	1 шт
		Интернет	
33	Внедрение завершено		
34	Конец проекта		

Таким образом, реализацией проекта будут заниматься **3** программиста и **1** дизайнер, которым для работы нужны компьютеры и доступ к интернету.

Для вычисления длительности реализации проекта и его критического пути, воспользуемся системой управления проектами Project Libre, которая позволяет автоматически произвести все необходимые вычисления [19 – 23].

Список задач из таблицы 2, внесенный в систему Project Libre, представлен на рисунке 12. В качестве времени начала проекта была принята дата 01.12.2023.

		Название	Продолжител...	Начало	Окончание	Предшествующие
1		Начало проекта	0 дней	01.12.23 8:00	01.12.23 8:00	
2		Анализ требований	10 дней	01.12.23 8:00	14.12.23 17:00	
3		Поиск литературы	2 дней	01.12.23 8:00	04.12.23 17:00	1
4		Сравнительный анализ ИС контроля учебного процесса	4 дней	05.12.23 8:00	08.12.23 17:00	3
5		Выделение функциональных требований	1 день	11.12.23 8:00	11.12.23 17:00	4
6		Выделение нефункциональных требований	1 день	12.12.23 8:00	12.12.23 17:00	5
7		Консультация с представителем ВУЗа	1 день	13.12.23 8:00	13.12.23 17:00	6
8		Принятие окончательного решения	1 день	14.12.23 8:00	14.12.23 17:00	7
9		Анализ требований завершен	0 дней	14.12.23 17:00	14.12.23 17:00	8
10		Выбор средств реализации	12 дней	01.12.23 8:00	18.12.23 17:00	
11		Анализ возможных программных средств	1 день	01.12.23 8:00	01.12.23 17:00	1
12		Выбор программных средств для реализации	2 дней	15.12.23 8:00	18.12.23 17:00	9
13		Выбор средств завершен	0 дней	18.12.23 17:00	18.12.23 17:00	12
14		Проектирование приложения	11 дней	19.12.23 8:00	02.01.24 17:00	
15		Проектирование архитектуры приложения	4 дней	19.12.23 8:00	22.12.23 17:00	9;13
16		Создание макета пользовательского интерфейса	3 дней	25.12.23 8:00	27.12.23 17:00	15
17		Проектирование базы данных	7 дней	25.12.23 8:00	02.01.24 17:00	15
18		Проектирование приложения завершено	0 дней	02.01.24 17:00	02.01.24 17:00	16;17
19		Реализация проекта	15 дней	03.01.24 8:00	23.01.24 17:00	
20		Реализация базы данных	5 дней	03.01.24 8:00	09.01.24 17:00	17
21		Реализация серверной части приложения	10 дней	10.01.24 8:00	23.01.24 17:00	18;20
22		Реализация клиентской части приложения	10 дней	10.01.24 8:00	23.01.24 17:00	16;20
23		Реализация проекта завершена	0 дней	23.01.24 17:00	23.01.24 17:00	21;22
24		Контроль качества программного продукта	20 дней	24.01.24 8:00	20.02.24 17:00	
25		Тестирование серверной части приложения	5 дней	24.01.24 8:00	30.01.24 17:00	21
26		Тестирование клиентской части приложения	10 дней	24.01.24 8:00	06.02.24 17:00	22
27		Исправление ошибок	10 дней	07.02.24 8:00	20.02.24 17:00	25;26
28		Отладка завершена	0 дней	20.02.24 17:00	20.02.24 17:00	27
29		Внедрение приложения	11 дней	21.02.24 8:00	06.03.24 17:00	
30		Внесение данных в систему	1 день	21.02.24 8:00	21.02.24 17:00	28
31		Внедрение программного продукта в эксплуатацию	3 дня	22.02.24 8:00	26.02.24 17:00	30
32		Поиск и исправление ошибок при внедрении	7 дней	27.02.24 8:00	06.03.24 17:00	31
33		Внедрение завершено	0 дней	06.03.24 17:00	06.03.24 17:00	32
34		Конец проекта	0 дней	06.03.24 17:00	06.03.24 17:00	33

Рисунок 12 – Список задач проекта

В среде Project Libre на основе введенных данных была построена диаграмма Ганта (Gantt Chart). Диаграмма Ганта представляет собой графическое изображение расписания задач проекта. Она состоит из горизонтальных столбцов (линий) времени и вертикальных столбцов (полос), отображающих задачи проекта. Каждая задача представлена на диаграмме в виде полосы, длина которой соответствует времени, необходимому для выполнения задачи. Диаграмма Ганта позволяет наглядно увидеть хронологию и продолжительность каждой задачи, а также зависимости между ними и сроки выполнения проекта в целом. На ее основании можно понять, что критический путь проходит через задачи 1, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 17, 18, 20, 22, 23, 26, 27, 28, 30, 31, 32, 33, 34. Диаграмма представлена на рисунке 13.

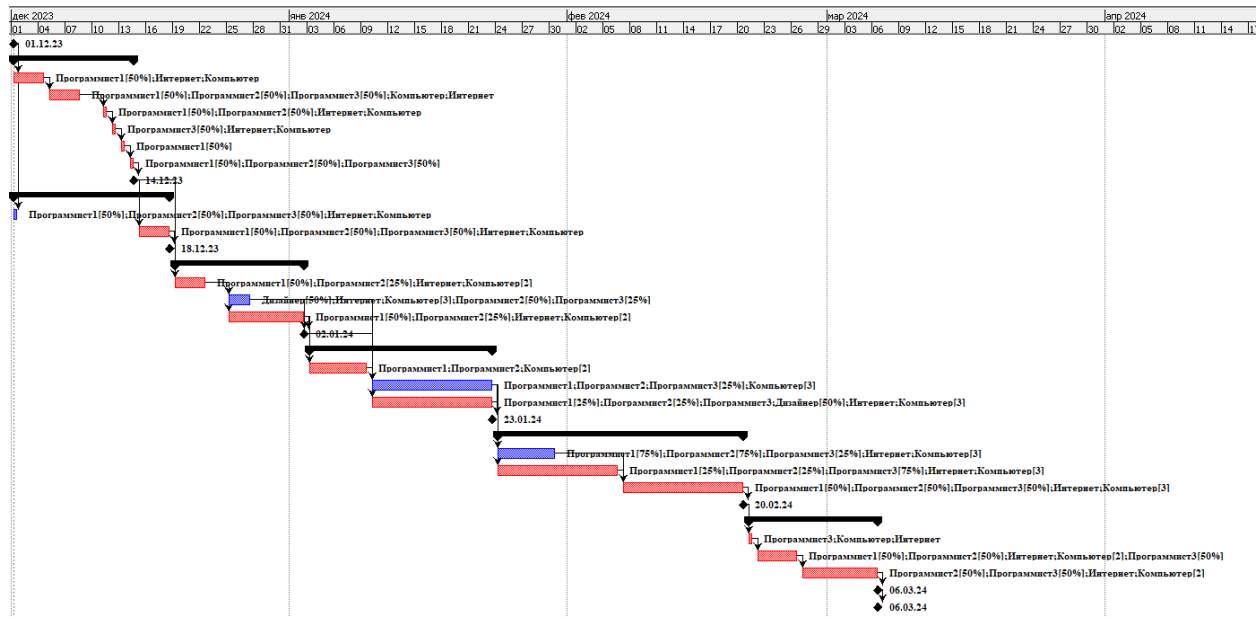


Рисунок 13 – Диаграмма Ганта

На рисунке 14 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 69 дней, а предполагаемые затраты – 392 150 руб.

Duration			
Scheduled	69 дней	Remaining	69 дней
Baseline	0 дней	Actual	0 дней
		Percent Complete	0%

Work			
Scheduled	996 часов	Remaining	996 часов
Baseline	0 часов	Actual	0 часов

Costs			
Scheduled	392150 руб.	Remaining	392150 руб.
Baseline	0 руб.	Actual	0 руб.
		Variance	0 руб.

Рисунок 14 – Обзорный отчет проекта

На основе произведенных оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование информационной системы

2.1 Выбор технологий и инструментальных средств разработки

Первым этапом проектирования информационной системы в рамках данной ВКР является обоснованный выбор методологии, технологий, языковых и инструментальных средств, отвечающих современным стандартам и требованиям разрабатываемого проекта.

В качестве методологии было решено использовать RUP. На каждой итерации перед разработчиками ставятся определенные цели, которые должны быть достигнуты к ее концу. «В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемой системы. В методологии RUP существуют 4 основных фазы, каждая из которых может включать в себя одну или несколько итераций» [24]:

- фаза начала проекта – формируется общее видение проекта и экономическое обоснование;
- уточнение – анализ предметной области и построение архитектуры продукта;
- построение – реализация основного функционала продукта;
- внедрение – получение финальной версии продукта и его передача заказчику, фаза включает в себя бета-тестирование и устранение выявленных в его процессе ошибок, а также обучение пользователей.

В конце каждой фазы разработчики проходят контрольную точку, где оценивает достигнутые цели и принимает ключевые решения о дальнейшем ходе работы над продуктом. Выбор в пользу данной методологии продиктован рядом ее достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро адаптировать проект в случае выявления ошибок или изменения требований к приложению. Кроме того, данная методология способствует раннему выявлению основных рисков. В отличие от некоторых других популярных методологий, таких как XP, RUP не

требует непрерывного взаимодействия с заказчиком, что может оказаться невозможным в рамках конкретного проекта.

Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования UML [25]. Для построения диаграмм использовалось средство UML-моделирования Visual Paradigm.

Для создания веб-приложения было решено использовать среду программирования Visual Studio, язык программирования C# с использованием фреймворка ASP.Net Core. Для хранения данных используется БД, работающая под управлением СУБД Microsoft SQL Server. Проектирование базы данных выполнялось с использованием приложения dbdiagram.io и draw.io. Реализация базы данных производилась с использованием языка программирования SQL.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Ее схема представлена на рисунке 15.



Рисунок 15 – Двухуровневая архитектура «клиент-сервер»

В данной архитектуре на сервере располагаются база данных и СУБД, а также серверная часть приложения, которая отвечает за непосредственное взаимодействие с базой данных и, как правило, реализуется в виде интерфейса прикладного программирования (API).

Клиентская же часть разрабатываемой системы обеспечивает для пользователя удобный интерфейс, позволяющий пользователю просматривать необходимые ему данные из БД и взаимодействовать с ними, а также реализует возможность удаленно обращаться к серверу. В нашем случае эту роль выполняет web-приложение.

Взаимодействие клиента и сервера осуществляется следующим образом: клиент делает HTTP-запрос серверу. Серверная часть приложения, в свою очередь, на основе полученного запроса создает SQL-запрос к базе данных. СУБД обрабатывает SQL-запрос и возвращает его результаты серверу, который на основе полученных результатов формирует ответ на запрос клиента. Получив ответ, клиент представляет его пользователю в удобной для того форме.

Архитектурным преимуществом является хранение всех данных на сервере, что помогает сократить размер клиентского приложения. Благодаря независимости серверной части от клиентской, можно создать несколько различных вариантов клиентов. Это позволяет разработать разнообразные интерфейсы без изменения серверной логики приложения.

2.3 Проектирование базы данных

2.3.1 Разработка концептуальной модели базы данных

Концептуальная модель данных (Conceptual Data Model) – это высокоуровневое описание структуры данных в предметной области, независимое от конкретных технических деталей реализации. Она используется для описания сущностей (entities) и их взаимосвязей в информационной системе.

На основе анализа предметной области и результатов моделирования бизнес-процессов построена концептуальная модель базы данных, представленная в виде диаграммы «сущность-связь» в нотации Чена на рисунке 16.

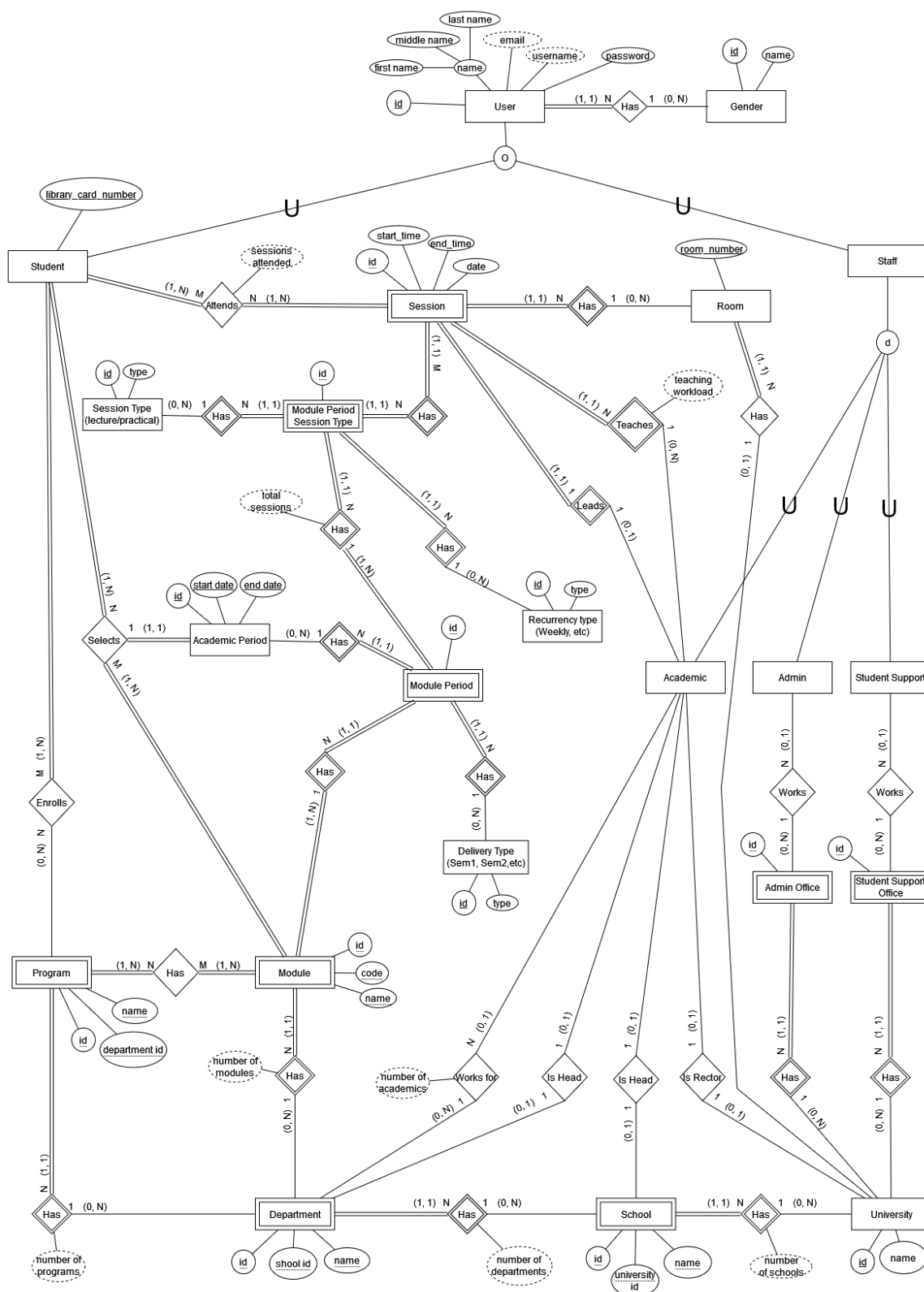


Рисунок 16 – Концептуальная модель базы данных

В соответствии с концептуальной моделью, в таблице 5 были описаны сущности.

Таблица 5 – Описание сущностей предметной области

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Преподаватель	Информация о преподавателях	Справочная	300	0.5%
2	Академический период	Учебный год, 2019/20	Справочная	30	1%
3	Администратор	Информация об администраторах	Справочная	150	1.5%
4	Кафедра	Информация о кафедрах факультета	Ассоциативная	30	0.5%
5	Пол	Информация о поле	Справочная	3	0%
6	Предмет	Предметы, преподаваемые на кафедре	Ассоциативная	300	1%
7	Период преподавания	Срок, сколько преподается предмет (Весенний/осенний семестр, год)	Справочная	3	0%
8	Длительность предмета	Информация о периоде преподавания конкретного предмета	Ассоциативная	10	1%
9	Программа	Информация об образовательных программах	Справочная	30	1%
10	Аудитория	Информация об аудиториях	Справочная	300	0,5%
11	Факультет	Информация о факультетах	Ассоциативная	15	1%
12	Пара	Информация о парах	Ассоциативная	10000	0.5%
13	Периодичность пары	Период преподавания пары (Каждую неделю, каждую вторую неделю)	Справочная	15	1%

Окончание таблицы 5

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
14	Тип пары	Тип пары (лекция, лабораторная)	Справочная	5	1.5%
15	Студент	Информация о студентах	Справочная	20000	2%
16	Посещение	Информация о посещении студентом пары	Ассоциативная	7000	2%
17	Университет	Информация об университете	Справочная	1	0.5%
18	Пользователь	Информация о пользователях	Справочная	8000	3%

В таблице 6 были описаны связи между сущностями. Связи между сущностями в таблице базы данных представляют отношения между различными сущностями, которые описываются с помощью внешних ключей.

Таблица 6 – Описание связей между сущностями

№ п/п	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
1	Пользователь	Пол	Пользователь имеет пол	1:M	Необязательная (Может не быть пользователей определенного пола)
2	Пользователь	Студент	Пользователь может являться студентом	1:1	Пользователь может быть как студентом, так и сотрудником, так и тем, и другим одновременно
3	Пользователь	Сотрудник	Пользователь может являться сотрудником	1:1	
4	Сотрудник	Преподаватель	Сотрудник может являться преподавателем	1:1	Сотрудник обязательно должен быть либо преподавателем, либо администратором, либо сотрудником поддержки студентов

Продолжение таблицы 6

№ п/п	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
5	Сотрудник	Администра тор	Сотрудник может являться администратор ом	1:1	
6	Сотрудник	Поддержка студентов	Сотрудник может являться поддержкой студентов	1:1	
7	Университет	Офис поддержки студентов	В университете есть офис поддержки студентов	1:M	Необязательная (В университете может не быть офиса поддержки студентов)
8	Офис подержки студентов	Сотрудник поддержки студентов	Сотдрудник работает в офисе	1:M	Необязательная (В офисе может не быть сотрудников, сотрудник может не принадлежать к офису)
9	Университет	Офис администра ции	В университете есть офис администрации	1:M	Необязательная (В университете может не быть офиса администрации)
10	Офис администрац ии	Администра тор	Администратор работает в офисе	1:M	Необязательная (В офисе может не быть сотрудников, сотрудник может не принадлежать к офису)
11	Университет	Аудитория	В университете есть аудитории	1:M	Обязательная
12	Университет	Преподавате ль	Преподаватель является ректором	1:1	Необязательная
13	Университет	Факультет	Факультет принадлежит университету	1:M	Необязательная

Окончание таблицы 6

№ п/п	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
14	Факультет	Преподаватель	Преподаватель руководит факультетом	1:1	Необязательная
15	Факультет	Кафедра	Кафедра принадлежит факультету	1:M	Необязательная
6	Кафедра	Преподаватель	Преподаватель руководит кафедрой	1:1	Необязательная
17	Кафедра	Преподаватель	Преподаватель преподает на кафедре	1:M	Необязательная
18	Кафедра	Программа	На кафедре преподаются учебные программы	1:M	Необязательная
19	Кафедра	Предмет	На кафедре преподаются предметы	1:M	Необязательная
20	Программа	Предмет	Программа содержит предметы	N:M	Обязательная
21	Студент	Программа	Студент зачисляется на учебную программу	N:M	Необязательная
22	Студент	Предмет	Студент выбирает предмет	N:M	Обязательная

Концептуальная модель базы данных не зависит от конкретных технических деталей реализации. В проектировании она служит основой для создания более детализированных моделей, таких как логическая и физическая модели баз данных.

2.3.2 Логическое моделирование базы данных

Логическая модель данных представляет собой описание фактов и объектов, которые будут зарегистрированы в будущей базе данных. Ключевыми элементами этой модели являются сущности, их атрибуты и взаимосвязи между ними. Обычно сущность в базе данных соответствует таблице, а атрибут сущности – полю этой таблицы.

С логической точки зрения, сущность представляет собой группу объектов или фактов одного типа, каждый из которых называется экземпляром этой сущности. Важно отметить, что физически экземпляр обычно представлен записью в таблице базы данных. Подобно записям в таблице реляционной СУБД, экземпляры сущности должны быть уникальными, то есть полный набор значений их атрибутов не должен повторяться. И так же, как и поля в таблице, атрибуты могут быть ключевыми и неключевыми.

На этапе логического проектирования для каждого атрибута обычно определяется примерный тип данных (строковый, числовой, логический и др.). «Конкретизация происходит на этапе физического проектирования, так как различные СУБД поддерживают разные типы данных и ограничения на их длину или точность» [26].

Перед построением физической модели было произведено логическое моделирование, которое позволило определить первичные ключи для различных сущностей. Исходя из концептуальной модели, была разработана логическая модель, представленная на рисунке 17.

Логическая модель данных является основой физической модели данных, которая фокусируется на фактических деталях реализации в конкретной системе **управления базами данных (СУБД).**

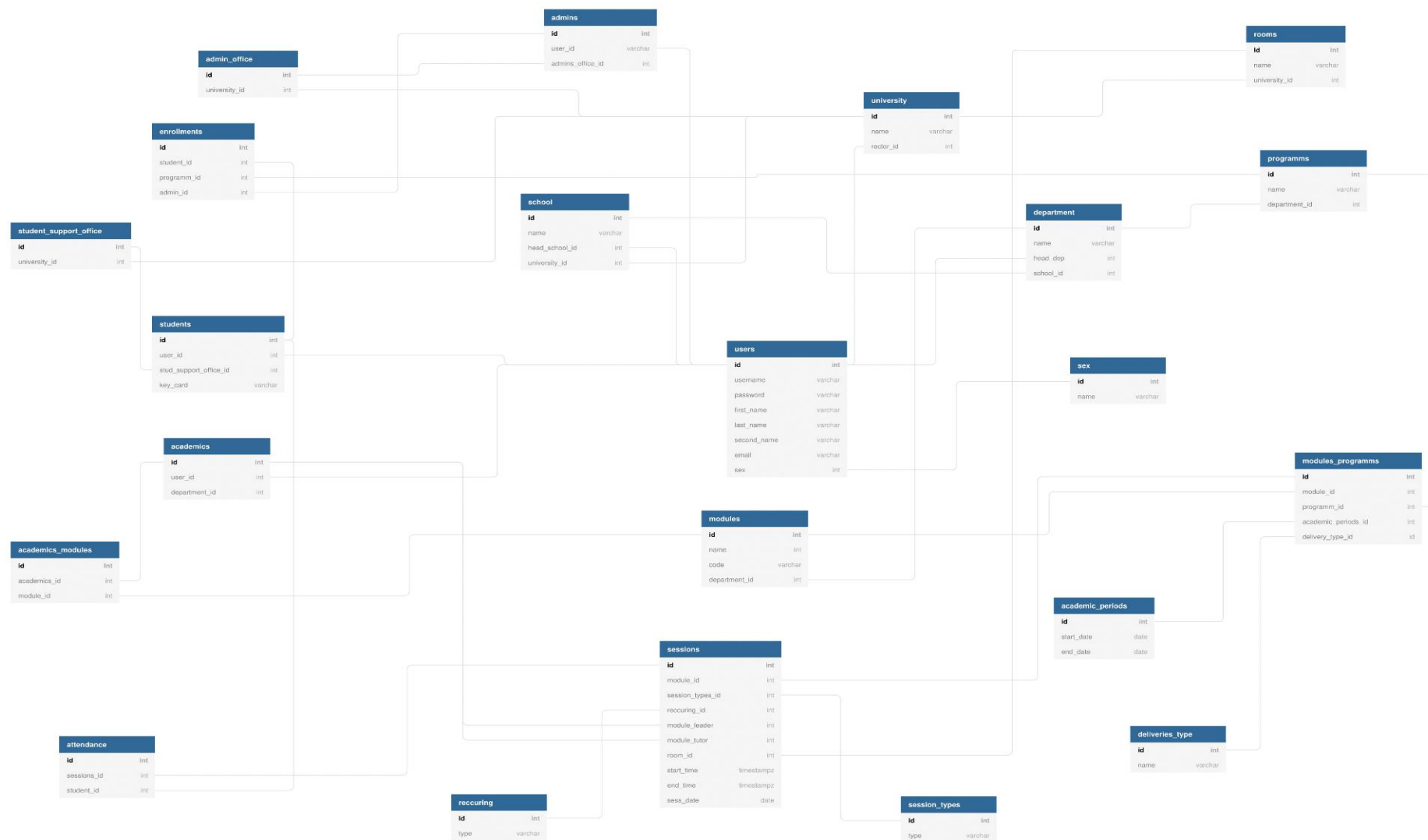


Рисунок 17 – Логическая модель базы данных

В таблице 7 приведено описание сущностей модели. Следует отметить, что с логической модели связь «многие ко многим» требует конкретизации через промежуточную таблицу. Например, для реализации связи «многие ко многим» между «Студентами» и «Курсами» создается таблица «Запись на курс», которая содержит внешние ключи на идентификаторы студентов и курсов. Эта таблица позволяет отслеживать, какие студенты записаны на какие курсы и наоборот.

Таблица 7 – Описание сущностей предметной области

№ п/п	Наименование	Описание	Категория
1	Преподаватель	Информация о преподавателях	Ассоциативная
2	Академический период	Учебный год, 2019/20	Справочная
3	Администратор	Информация об администраторах	Ассоциативная
4	Офис администрации	Информация об офисе администрации	Ассоциативная
5	Кафедра	Информация о кафедрах факультета	Справочная
6	Пол	Информация о поле	Справочная
7	Предмет	Предметы, преподаваемые на кафедре	Справочная
8	Период преподавания	Срок, сколько преподается предмет (Весенний/осенний семестр, год)	Справочная
9	Длительность предмета	Информация о периоде преподавания конкретного предмета	Ассоциативная
10	Проведение пар	Информация о парах в рамках преподавания предмета в академический период	Ассоциативная
11	Программа	Информация об образовательных программах	Ассоциативная
12	Дисциплины программы	Информация о предметах, включенных в образовательную программу	Ассоциативная
13	Аудитория	Информация об аудиториях	Справочная
14	Факультет	Информация о факультетах	Ассоциативная
15	Пара	Информация о парах	Ассоциативная

Окончание таблицы 7

№ п/п	Наименование	Описание	Категория
16	Периодичность пары	Период преподавания пары (Каждую неделю, каждую вторую неделю)	Справочная
17	Тип пары	Тип пары (лекция, лабораторная)	Справочная
18	Студент	Информация о студентах	Ассоциативная
19	Поддержка студентов	Информация о сотрудниках поддержки студентов	Ассоциативная
20	Офис поддержки студентов	Информация об офисе поддержки студентов	Ассоциативная
21	Выбранные предметы	Предметы, на которые зарегистрирован студент	Ассоциативная
22	Зачисление	Информация о зачислении студента на образовательные программы	Ассоциативная
23	Посещение	Информация о посещении студентом пары	Ассоциативная
24	Университет	Информация об университете	Ассоциативная
25	Пользователь	Информация о пользователей	Ассоциативная

В проектировании логической модели базы данных ключевым моментом является трансформация концептуальной модели в конкретные структуры данных, оптимизированные для хранения и манипулирования информацией. Логическая модель детализирует связи между сущностями, включая реализацию сложных отношений. Это обеспечивает эффективность и надежность работы баз данных в рамках конкретной системы.

2.4 Проектирование пользовательского интерфейса

При проектировании пользовательского интерфейса следует учитывать все специфики информационной системы с пользователем.

На основе анализа предметной области были выделены следующие виды пользовательских интерфейсов:

- страница входа и регистрации;

- страница просмотра данных о посещаемости и успеваемости студента;
- страницы для внесения новых данных в систему;
- страницы для удаления существующих данных.

Конкретная реализация и отображение интерфейсов будет варьироваться в зависимости от прав доступа пользователей.

На рисунке 18 показана форма для входа на учебный портал. Форма включает в себя следующие элементы:

- 1 – заголовок «Учебный портал»;
- 2 – поле для ввода логина (или электронного адреса);
- 3 – поле для ввода пароля;
- 4 – ссылка «Регистрация» для создания нового аккаунта;
- 5 – кнопка «Вход» для подтверждения ввода и входа в систему.

Рисунок 18 – Макет интерфейса формы входа в систему

На рисунке 19 показана форма для регистрации в системе. Форма включает в себя следующие элементы:

- 1 – заголовок «Учебный портал»;
- 2 – поле для ввода логина (или электронного адреса);

3 – поле для ввода пароля;

4 – поле для повторного ввода пароля;

5 – кнопка «Регистрация» для подтверждения ввода и регистрации.

Рисунок 19 – Макет интерфейса формы регистрации

На рисунке 20 представлен макет интерфейса главного меню Администратора. В верхней части находится панель с заголовком «Главная». Далее располагается содержимое с заголовком «Все программы». Под заголовком расположено меню выбора конкретной программы для просмотра.

Рисунок 20 – Макет интерфейса главного меню

Рисунок 21 представляет собой макет интерфейса страницы отображения данных о конкретной образовательной программе. В верхней части находится

меню с тремя пунктами. Нижняя часть разделена на две секции: секция «Дисциплины», в которой будут отображаться добавленные дисциплины, и секция «Группы», где будут представлены добавленные группы.

Главная Добавить дисциплину Добавить группу

Программа

Дисциплины

Группы

Рисунок 21 – Макет интерфейса образовательной программы

Рисунок 22 изображает макет отчета по выбранной статистике, в частности, по группе. Сверху расположен заголовок с кнопками «Главная» и «Добавить студента». Ниже содержится основная таблица. В первой колонке указаны студенты, во второй – их средний балл, и в третьей – данные о посещаемости.

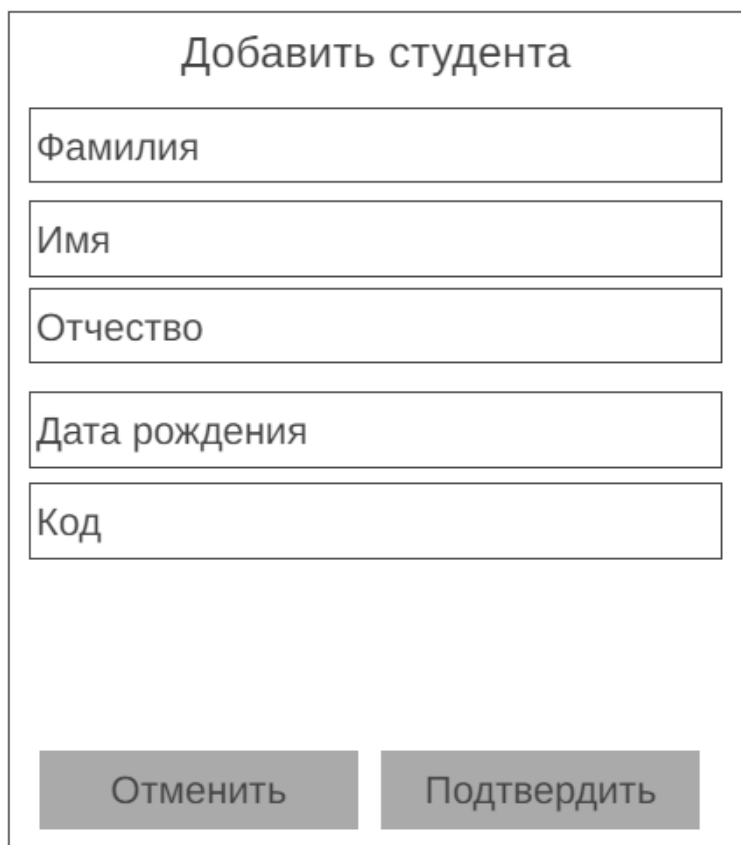
Главная Добавить студента

Группа

Студент	Средний балл	Посещаемость

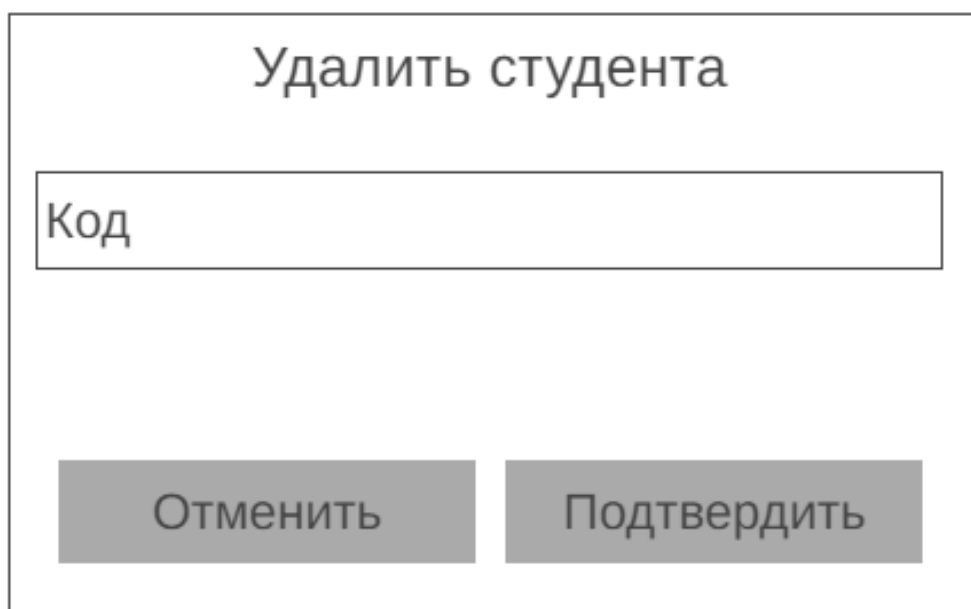
Рисунок 22 – Макет интерфейса отчета по заданной статистике

Рисунки 23 и 24 изображают простые формы для манипуляций с данными о студентах в разрабатываемой информационной системе.



The form is titled "Добавить студента" (Add student) in a large, bold, black font at the top center. Below the title are five input fields, each with a label in a smaller black font: "Фамилия" (Surname), "Имя" (Name), "Отчество" (Patronymic), "Дата рождения" (Date of birth), and "Код" (Code). The input fields are rectangular with thin black borders. At the bottom of the form are two gray buttons with white text: "Отменить" (Cancel) on the left and "Подтвердить" (Confirm) on the right.

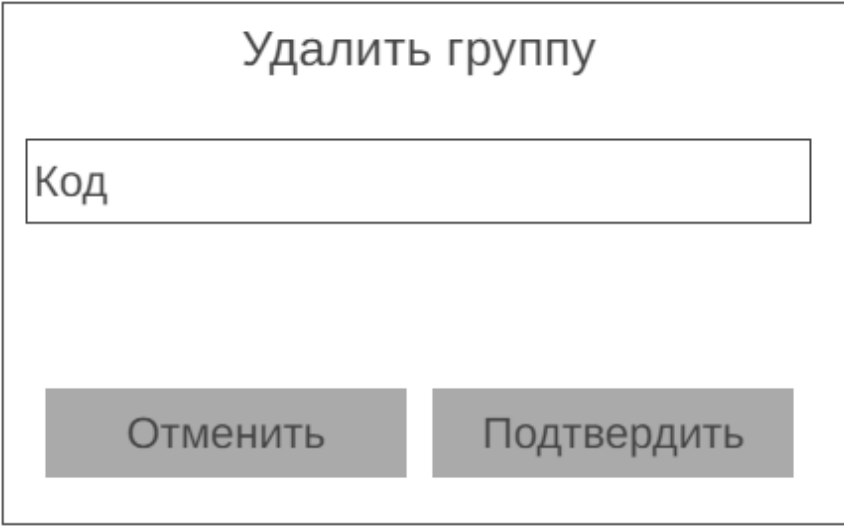
Рисунок 23 – Макет формы добавления студента



The form is titled "Удалить студента" (Delete student) in a large, bold, black font at the top center. Below the title is a single input field with the label "Код" (Code) in a smaller black font. The input field is rectangular with a thin black border. At the bottom of the form are two gray buttons with white text: "Отменить" (Cancel) on the left and "Подтвердить" (Confirm) on the right.

Рисунок 24 – Макет формы удаления студента

Рисунки 25 и 26 изображают простые формы для манипуляций с данными в разрабатываемой информационной системе.



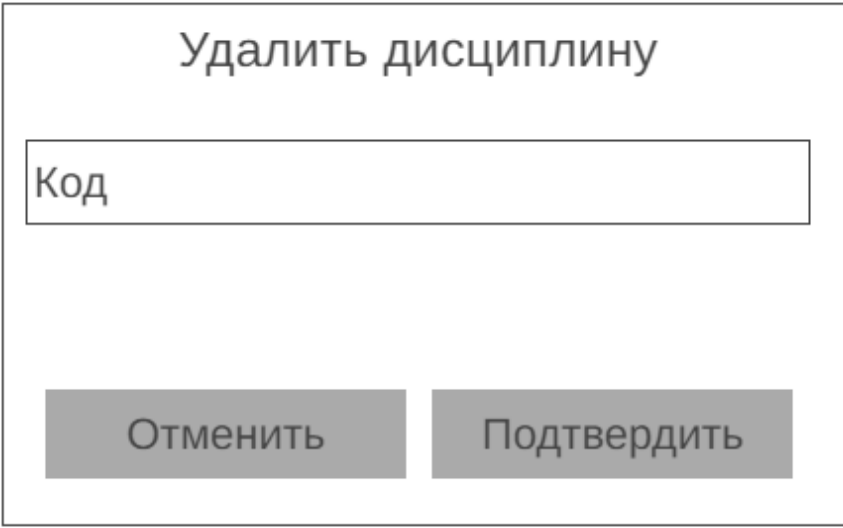
Удалить группу

Код

Отменить Подтвердить

This is a web form titled "Удалить группу" (Delete group). It features a text input field labeled "Код" (Code). Below the input field are two buttons: "Отменить" (Cancel) and "Подтвердить" (Confirm).

Рисунок 25 – Макет формы удаления группы



Удалить дисциплину

Код

Отменить Подтвердить

This is a web form titled "Удалить дисциплину" (Delete discipline). It features a text input field labeled "Код" (Code). Below the input field are two buttons: "Отменить" (Cancel) and "Подтвердить" (Confirm).

Рисунок 26 – Макет формы удаления дисциплины

В процессе проектирования пользовательского интерфейса разрабатываемого веб-приложения было обеспечено удобство и эффективность взаимодействия пользователей с системой. Интерфейс разработан с учетом интуитивно понятной навигации и минимизации сложности выполнения основных задач. Это позволяет пользователям быстро адаптироваться к новому приложению и эффективно использовать его функционал.

2.5 Проектирование серверной части приложения

Серверная часть приложения представляет собой REST API. Доступ к данным осуществляется при помощи протокола HTTP с использованием его стандартных методов (GET применяется для получения данных ресурса, POST – для создания нового ресурса, PUT – для обновления существующего, DELETE – для удаления ресурса). Обращение к ресурсам происходит при помощи уникальных идентификаторов ресурсов (URI).

В приложении используется архитектура MVC (Model-View-Controller). Ее схема представлена на рисунке 27.

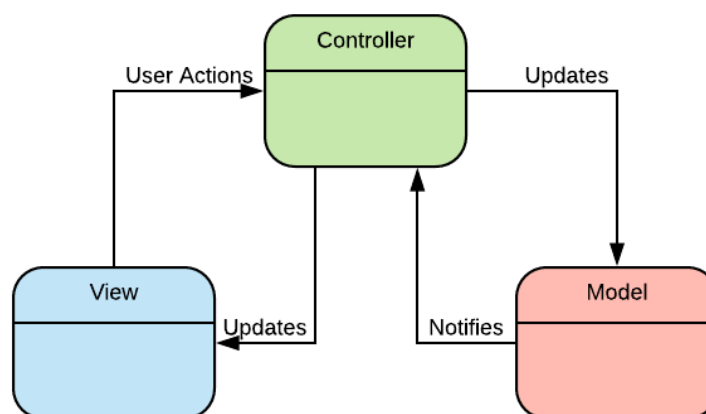


Рисунок 27 – Архитектура MVC

Серверная часть приложения включает в себя классы Модели, отвечающие за организацию доступа к данным (т. е. за непосредственное взаимодействие с БД) и классы Контроллера, отвечающие за организацию взаимодействия пользователя с клиентом. Именно они обрабатывают HTTP-запросы клиента и формируют ответы на них. К клиенту же отойдет только уровень Представления, отвечающий за взаимодействие с пользователем.

Таким образом, классическая архитектура «Модель-Контроллер-Представление» (MVC) помогает структурировать код таким образом, чтобы каждая часть системы была ответственна за свою конкретную задачу, что способствует лучшей поддерживаемости и расширяемости приложения.

3 Реализация информационной системы

3.1 Реализация базы данных

На основе составленной ранее логической модели БД с учетом специфики СУБД Microsoft SQL Server была построена физическая модель. Ее структура представлена на рисунке 28.

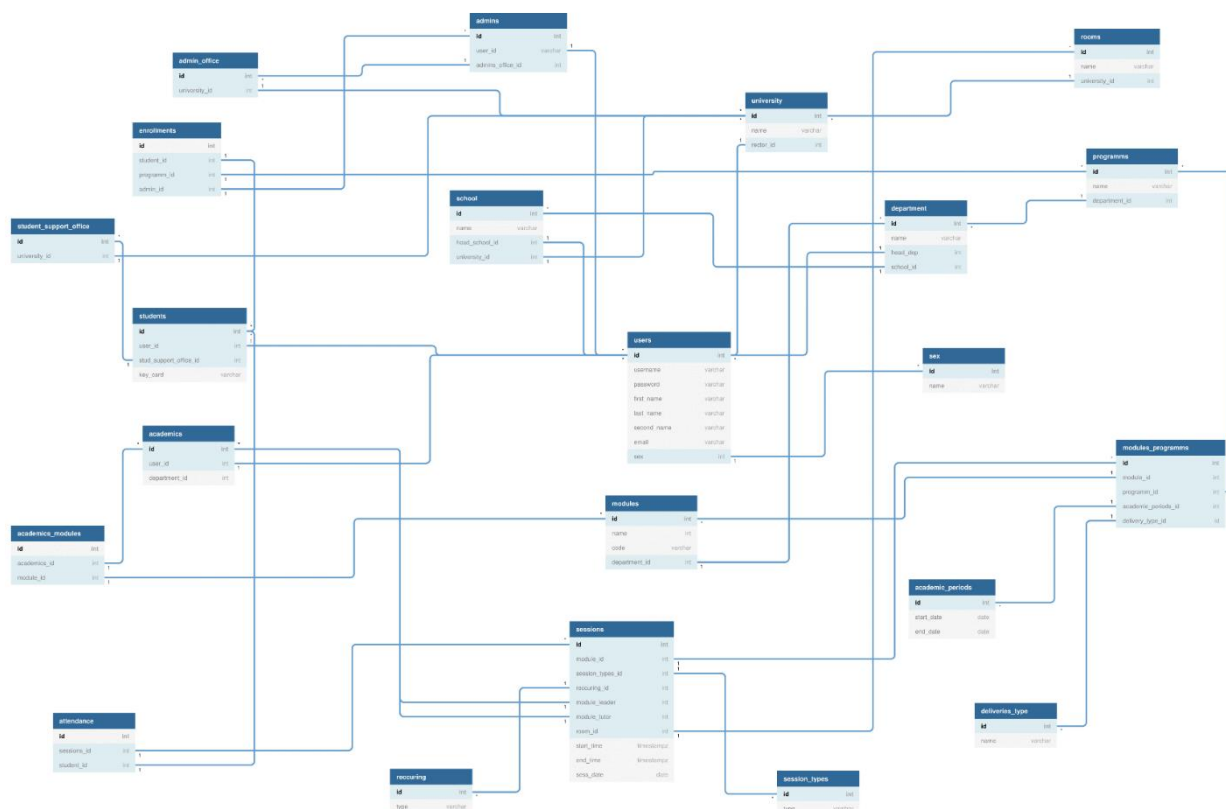


Рисунок 28 – Физическая модель базы данных

На основе составленной физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта с использованием утилиты SQL Server Management Studio, используемой для администрирования баз данных под управлением СУБД Microsoft SQL Server.

3.2 Разработка серверной части

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2022 на языке C# с применением технологии MVC [27 – 28]. Для взаимодействия с базой данных использовалась технология Entity Framework [29]. Диаграмма классов серверной части представлена на рисунке 29.

В соответствии с используемой архитектурой MVC все классы серверной части можно разделить на два слоя: слой модели, предназначенный для взаимодействия с БД, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу.

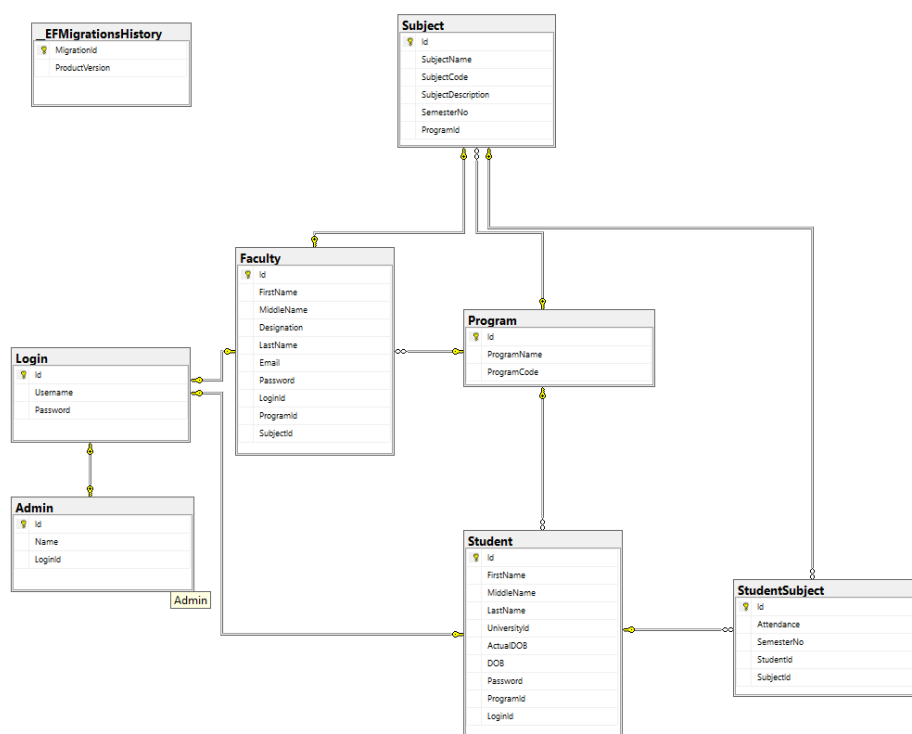


Рисунок 29 – Диаграмма классов серверной части (слой модели)

Первый класс, «EFMigrationsHistory», содержит поля «MigrationId» и «ProductVersion», необходимые для ведения истории миграций базы данных. Таблица «Subject» включает в себя поля «Id», «SubjectName», «SubjectCode», «SubjectDescription», «SemesterNo» и «ProgramId», обеспечивающие управление

предметами учебной программы. Таблица «Faculty» содержит поля «Id», «FirstName», «MiddleName», «Designation», «LastName», «Email», «Password», «ProgramId» и «SubjectId», служащие для хранения информации о преподавательском составе.

Класс «Login» состоит из полей «Id», «Username» и «Password», используемых для аутентификации пользователей. Таблица «Admin» включает поля «Id», «Name» и «LoginId», предназначенные для управления администраторами системы. Класс «Program» содержит поля «Id», «ProgramName» и «ProgramCode», обеспечивающие управление академическими программами. Класс «Student» включает в себя поля «Id», «FirstName», «MiddleName», «LastName», «UniversityId», «ActualDOB», «DOB», «Password», и «ProgramId», предназначенные для хранения данных о студентах.

Наконец, класс «StudentSubject» содержит поля «Id», «Attendance», «SemesterNo», «StudentId» и «SubjectId», которые служат для учета посещаемости и успеваемости студентов по предметам. Между таблицами установлены различные связи, обеспечивающие целостность данных и их правильное распределение по системе.

Диаграмма классов слоя контроллеров представлена на рисунке 30.

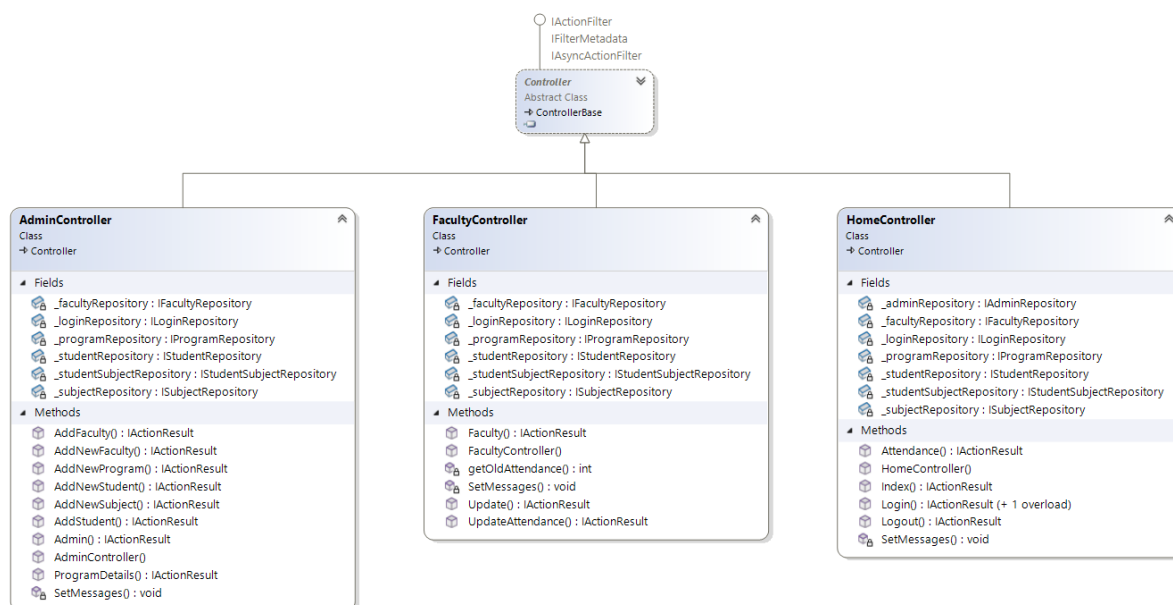


Рисунок 30 – Диаграмма классов серверной части (слой контроллеров)

Классы контроллеров организуют взаимодействие пользователя с клиентом, управляют потоком данных между клиентом и сервером, принимают HTTP-запросы от клиентов, обрабатывают их и возвращают соответствующие HTTP-ответы. Контроллеры анализируют входящие запросы, вызывают соответствующие методы моделей для выполнения необходимых операций с данными и формируют ответы на основе результатов этих операций. Они могут возвращать данные в различных форматах, таких как JSON или XML, в зависимости от требований клиента, а также обрабатывать ошибки и возвращать соответствующие HTTP-коды состояния.

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 8. Листинг контроллеров программы представлен в приложении В.

Таблица 8 – Описание классов слоя контроллеров

Класс	Назначение
AdminController	Отвечает за обработку запросов, доступных администратору: добавление новых студентов и групп на программы.
FacultyController	Отвечает за обработку запросов, доступных преподавателю: просмотр и добавление записей об успеваемости и посещаемости студентов.
HomeController	Отвечает за обработку запросов на вход и регистрацию пользователей, а также отображение сведений об успеваемости и посещаемости студентов, что доступно всем категориям пользователей.

В ASP.NET Core MVC классы модели и класс контекста данных DbContext используются вместе для управления данными и взаимосвязями между объектами и базой данных. Каждый из классов на диаграмме, представленной на рисунке 31, представляет собой сущность в базе данных. Например, Admin представляет администратора с такими свойствами, как Id, Login, LoginId и Name. И так далее для остальных сущностей.

DbContext управляет соединением с базой данных. Он используется для настройки сущностей и их отношений. Контроллеры и сервисы используют ApplicationDbContext для выполнения операций с данными.

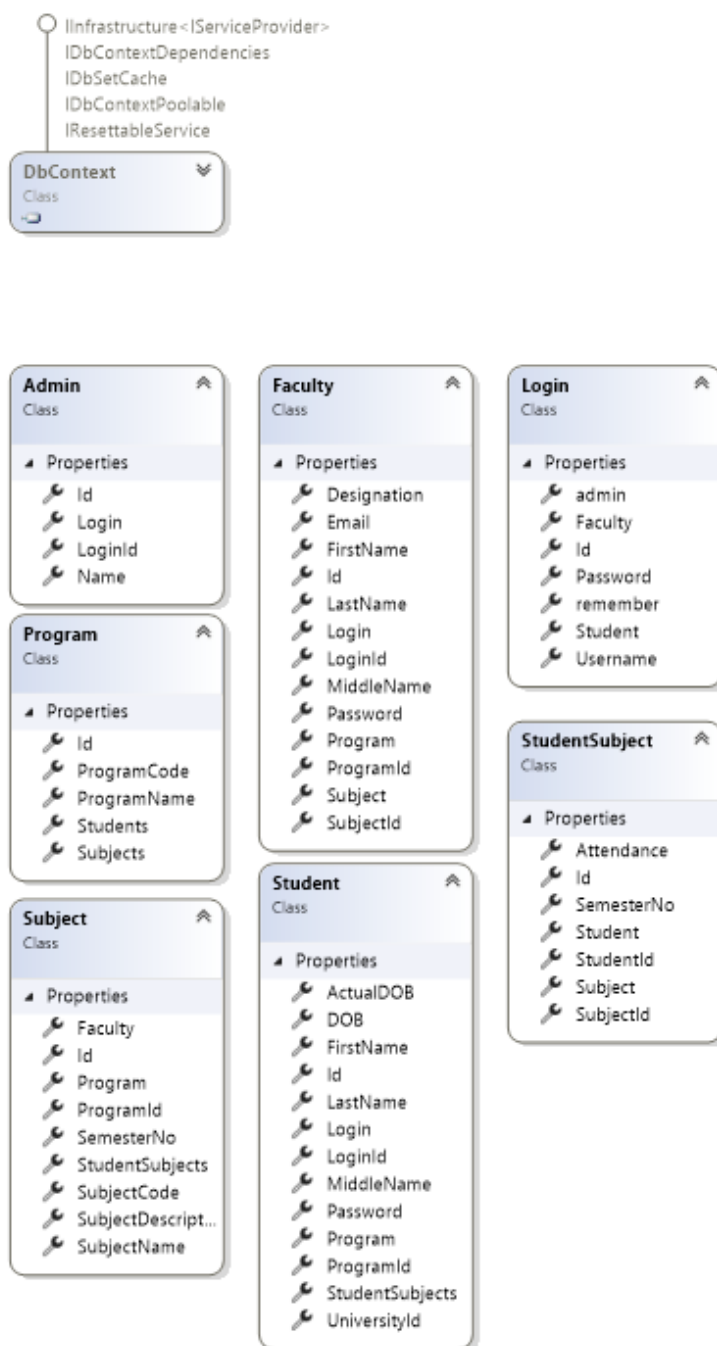


Рисунок 31 – Диаграмма классов слоя модели

Далее, на рисунке 32 представлены интерфейсы и классы, их реализующие, относящиеся к DAO (Data Access Object) классам. DAO-классы предоставляют интерфейс для выполнения операций с данными (CRUD – Create, Read, Update, Delete), скрывая детали реализации. Это позволяет работать с данными на более высоком уровне абстракции. Кроме того, если необходимо изменить источник данных (например, перейти с одной базы данных на другую),

изменения будут локализованы в DAO-классах. Это минимизирует влияние на остальную часть приложения.

Листинг контроллеров программы представлен в приложении Г.

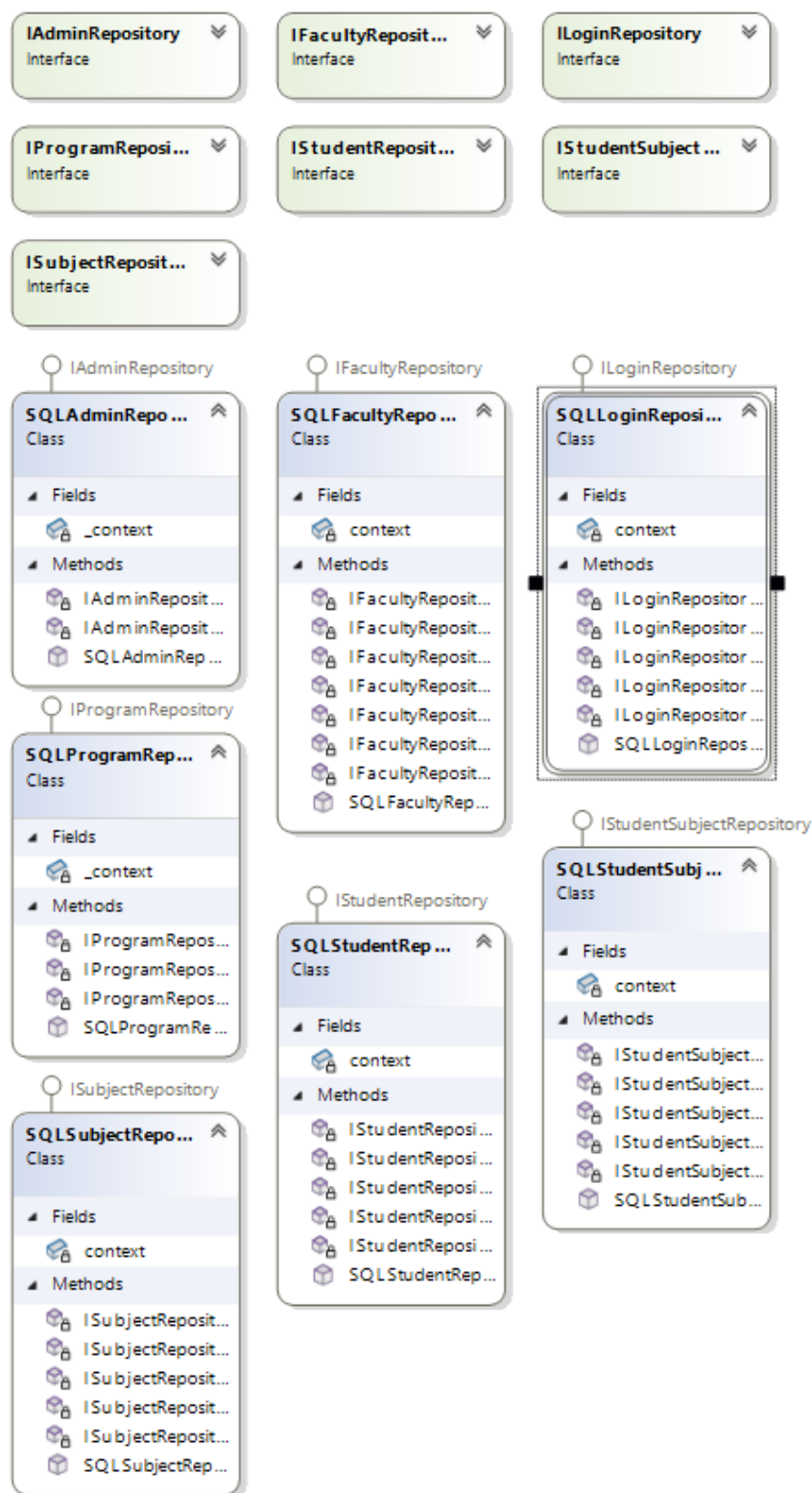


Рисунок 32 – Диаграмма классов DAO

Все эти классы были автоматически сгенерированы средой программирования Visual Studio с использованием технологии Entity Framework. Генерация производилась на основе предварительно разработанной до этого базы данных (подход Database First) [30 – 31].

3.3 Реализация клиентской части приложения

Реализация слоя представлений осуществляется с использованием технологии Razor. Razor – это синтаксис разметки для внедрения кода на основе .NET на веб-страницы. Синтаксис Razor состоит из разметки Razor, **C#** и HTML. Синтаксис аналогичен обработчикам шаблонов различных платформ одностраничных приложений JavaScript (SPA), таких как Angular, React, VueJs и Svelte.

Диаграмма компонентов, отражающая структуру слоя представлений, представлена на рисунке 33.

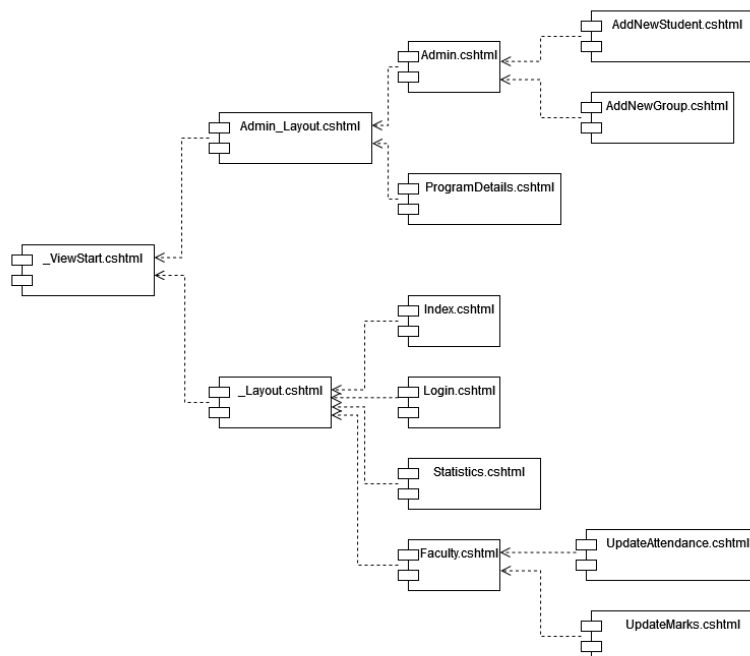


Рисунок 33 – Диаграмма компонентов представлений web-приложения

Описание компонентов представлено в таблице 9.

Таблица 9 – Компоненты клиентского приложения

Компонент	Назначение
_ViewStart.cshtml	Используется для определения общих настроек, которые будут применяться ко всем представлениям (views) в приложении. Устанавливает макет по умолчанию.
_Layout.cshtml	Шаблон по умолчанию. Содержит базовые настройки, ссылки на css-файлы и библиотеки jQuery, Popper и Bootstrap.
Admin_Layout.cshtml	Шаблон главной страницы Администратора с навигационной панелью и модальным окном добавления новой программы.
Admin.cshtml	Главная страница администратора. Отображает список образовательных программ.
ProgramDetails.cshtml	Страница, отображающая детали образовательной программы и модальное окно добавления дисциплины в программу.
AddNewStudents.cshtml	Форма добавления студента. Включает в себя поля для ввода данных, сообщения об ошибках, а также кнопку для отправки формы.
AddNewGroup.cshtml	Форма добавления группы. Включает в себя поля для ввода данных, сообщения об ошибках, а также кнопку для отправки формы.
Faculty.cshtml	Главная страница администратора. Отображает список преподаваемых дисциплин.
UpdateAttendance.cshtml	Форма добавления записи о посещаемости студента. Включает в себя поля для ввода данных, сообщения об ошибках, а также кнопку для отправки формы.
UpdateMarks.cshtml	Форма добавления записи об успеваемости студента. Включает в себя поля для ввода данных, сообщения об ошибках, а также кнопку для отправки формы.
Index.cshtml	Главная страница студента. Отображает список семестров, за которые можно посмотреть статистику.
Statistics.cshtml	Страница отображения статистики посещаемости и успеваемости студента за семестр.
Login.cshtml	Страница входа в систему и регистрации.

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 34.

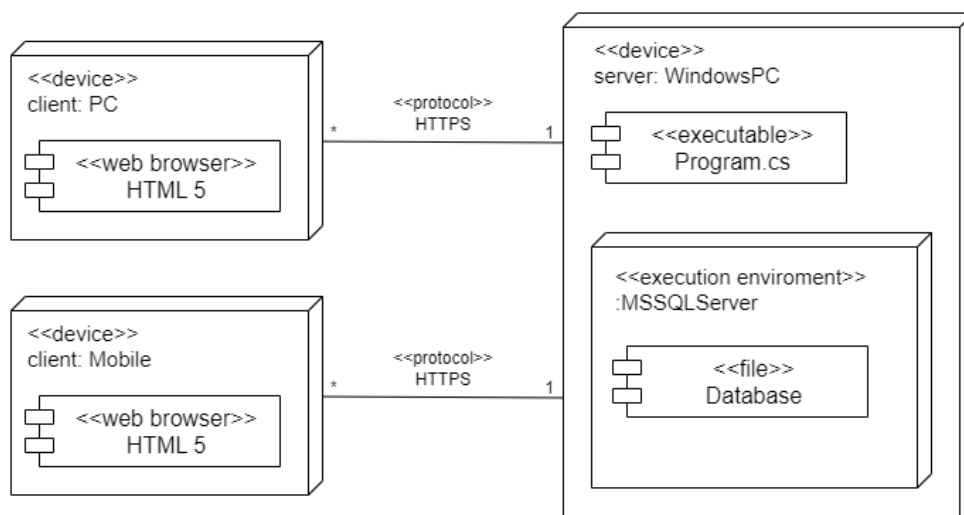


Рисунок 34 – Диаграмма развертывания web-приложения

Как можно увидеть на этой диаграмме, клиентское приложение работает в браузере пользователя. API и база данных располагаются на сервере, работающем под управлением ОС Windows. Взаимодействие между серверной и клиентской частями приложения происходит с использованием протокола HTTPS. Листинг клиентской части программы представлен в приложении Д.

4 Тестирование web-приложения

Разработка программного обеспечения всегда включает в себя этап тестирования, что помогает обеспечивать контроль качества разрабатываемого программного продукта. «Оно помогает выявить ошибки на раннем этапе и определить соответствия между реальным и ожидаемым поведением разработанного продукта» [32].

Тестирование разработанного web-приложения проводилось в следующих браузерах: Google Chrome и Mozilla Firefox.

4.1 Функциональное тестирование

Для проверки соответствия техническим требованиям web-приложения было выполнено функциональное тестирование, охватывающее функции, предварительно выделенные на стадии анализа предметной области. Описание функциональных тестов представлено в таблице 10.

Таблица 10 – Функциональные тесты web-приложения

№ п/п	Назначение теста	Описание теста	Результат теста
1	Тестирование формы входа в систему	Проверяет правильность работы интерфейса для ввода пользовательских данных и обработки ошибок при входе в систему веб-приложения. Браузер – Google Chrome.	Тест выполнен. Рисунки Е.1 – Е.3.
2	Тестирование отображения страницы образовательной программы	Проверяет корректность загрузки и отображения информации о образовательных программах, включая верный вывод данных и их форматирование. Браузер – Google Chrome.	Тест выполнен. Рисунок Е.4.
3	Тестирование формы добавления студента	Проверяет правильность работы интерфейса добавления новых записей о студентах в систему. Браузер – Google Chrome.	Тест выполнен. Рисунки Е.5 – Е.7.

Окончание таблицы 10

№ п/п	Назначение теста	Описание теста	Результат теста
4	Тестирование формы добавления группы	Проверяет правильность работы интерфейса добавления новых записей о группе в систему. Браузер – Google Chrome.	Тест выполнен. Рисунок Е.8.
5	Тестирование формы добавления дисциплины	Проверяет правильность работы интерфейса добавления новых записей о дисциплине в систему. Браузер – Google Chrome.	Тест выполнен. Рисунок Е.9.
6	Тестирование формирования отчета по заданной статистике	Браузер – Yandex.	Тест выполнен. Рисунок Е.10.
7	Тестирование отображения страницы о посещаемости студента за заданный период	Браузер – Mozilla Firefox.	Тест выполнен. Рисунки Е.11 – Е.12.

Результаты функционального тестирования показывают, что разработанное web-приложение корректно выполняет весь функционал в соответствии с техническим заданием.

4.2 Модульное тестирование

Модульное тестирование, иногда блочное тестирование или юнит-тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

Для тестирования приложения применяются средства тестирования, встроенные в интегрированную среду разработки Visual Studio.

На основе вариантов использования были составлены Unit-тесты. Пример теста для класса «дисциплина» представлен в листинге 1.

Листинг 1 – тесты модуля «Дисциплина»

```

/// <summary>
/// Обновление дисциплины без заданного названия
/// </summary>

```

```

[TestMethod]
public void UCDiscipline_1()
{
    // arrange
    MDiscipline gr = new MDiscipline(null, «КП», «1»);
    bool expected = false;

    // act
    bool actual = refData.CDiscipline.Update(gr);

    // assert
    Assert.AreEqual(expected, actual);
}

```

Подобным же образом было запущено автоматизированное модульное тестирование для всех модулей, результаты которого представлены на рисунке 35.

Test run finished: 189 Tests (189 Passed, 0 Failed, 0 Skipped) run in 3.1 sec			
Test	Duration	Traits	Error Message
✔ UnitTestOfTimetableOfClasses (189)	1.9 sec		
▶ ✔ UnitTestOfTimetableOfClasses (2)	390 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	346 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	20 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	17 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	510 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAuditor....	8 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CDirectio...	41 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CDiscipli...	6 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	13 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	110 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	10 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	25 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	103 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	72 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	14 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	45 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	5 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CTeacher...	2 ms		

Рисунок 35 – Результат модульного тестирования

Таким образом все методы класса оказались покрыты тестами. Все функциональные требования выполнены, что подтверждается отсутствием ошибок после проведения тестов.

4.3 Анализ кода

4.3.1 Индекс удобства поддержки

В качестве основной метрики используется Maintainability Index – комплексный показатель качества кода, который автоматически рассчитывается средствами Visual Studio. Данная метрика зависит от других метрик, среди которых: объем программы (Halstead Volume, HV), цикломатическая сложность (Cyclomatic Complexity, CC) и количество строк кода (Lines of Code, LOC), которые подробно будут описаны в следующей главе. Вычисляется данная метрика по формуле:

$$MI =$$

$$\max(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171),$$

где *HV* (Halstead Volume) – вычислительная сложность;

CC (Cyclomatic Complexity) – структурная сложность кода, т. е. количество различных ветвей в коде;

LoC (Lines of Code) – количество строк кода.

Данный показатель принимает значения от 0 до 100, чем выше значение, тем более простой считается поддержка кода. Хорошим показателем считается значение от 20 до 100 (зеленая метка), приемлемым – от 10 до 20 (желтая метка), плохим – менее 10 (красная метка).

После применения автоматизированного программного средства, встроенного в Visual Studio, для анализа и расчёта метрик были получены следующие показатели, представленные на рисунке 36.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
StudentAttendanceManagementSystem (Debug)	79	486	4	148	5,651	1,359
AspNetCore	73	196	4	35	2,819	557
StudentAttendanceManagementSystem	79	6	1	34	74	26
StudentAttendanceManagementSystem.Controllor	58	73	3	38	660	256
StudentAttendanceManagementSystem.Migration	40	10	2	36	1,556	411
StudentAttendanceManagementSystem.Models	96	126	2	28	216	25
StudentAttendanceManagementSystem.Models.D	91	75	1	24	326	84

Рисунок 36 – Расчет метрик кода

Результаты оценки всех модулей системы находятся в диапазоне от 40 до 96, что расценивается как хороший показатель.

4.3.2 Метрики кода

Source Lines of Code (SLOC) или количество строк кода, – это метрика оценки ПО, подсчитывающая физические и логические строки кода, а также комментарии. Проведем подсчет объема классов Student и AttendanceSystem, используя данную метрику. Классы приведены в листинге 2.

Листинг 2 – Классы Student и AttendanceSystem

```

/// <summary>
/// Represents a student with a name and attendance status.
/// </summary>
class Student
{
    public string Name { get; set; }
    public bool IsPresent { get; set; }
}

/// <summary>
/// Manages attendance records for students in different subjects.
/// </summary>
class AttendanceSystem
{
    private Dictionary<string, Dictionary<string, bool>>
attendanceRecords = new Dictionary<string, Dictionary<string, bool>>();

    /// <summary>
    /// Marks the attendance of students for a specific subject and
teacher.
    /// </summary>
    /// <param name=«subject»>The subject for which attendance is being
marked.</param>
    /// <param name=«teacher»>The teacher who is marking the
attendance.</param>
    /// <param name=«students»>The list of students to mark attendance
for.</param>
    public void MarkAttendance(string subject, string teacher,
List<Student> students)
    {
        if (!attendanceRecords.ContainsKey(subject))
        {
            attendanceRecords[subject] = new Dictionary<string, bool>();
        }

        foreach (var student in students)
        {
            attendanceRecords[subject][$»{student.Name} ({teacher})»] =
student.IsPresent;
        }
    }

    /// <summary>
    /// Displays the attendance records for a specific subject and
teacher.

```

```

    /// </summary>
    /// <param name=«subject»>The subject for which attendance is to be
displayed.</param>
    /// <param name=«teacher»>The teacher whose attendance records are to
be displayed.</param>
    public void DisplayAttendance(string subject, string teacher)
    {
        if (attendanceRecords.ContainsKey(subject))
        {
            Console.WriteLine($»Attendance for {subject} taught by
{teacher}:»);
            foreach (var studentAttendance in attendanceRecords[subject])
            {
                if (studentAttendance.Key.Contains(teacher))
                {
                    Console.WriteLine($»{studentAttendance.Key}:
{(studentAttendance.Value ? «Present» : «Absent»)»);
                }
            }
        }
        else
        {
            Console.WriteLine($»No attendance records found for {subject}
taught by {teacher}.»);
        }
    }
}

```

Физические строки кода (PLOC) – 58 строк:

- класс Student: 8 строк;
- класс AttendanceSystem: 50 строк.

Логические строки кода (LLOC) – 13 строк:

- класс Student: 2 строки;
- класс AttendanceSystem: 11 строк.

Комментарии – 17 строк:

- класс Student: 3 строки;
- класс AttendanceSystem: 14 строк.

Итого, получаем, что SLOC для данного листинга программы составляет 88. Результаты данных подсчётов как правило используются совместно с другими метриками в процессе оценки качества ИС.

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа синтаксических элементов (операторов и операндов) исходного кода программы. Операнды – переменные или константы. Операторы – комбинации символов, влияющие на значение и порядок операндов.

Для расчета необходимо просчитать следующие значения:

- n1 – число различных операторов данной реализации;
- n2 – число различных операндов модуля;
- N1 – общее число всех операторов;
- N2 – общее число всех операндов;
- число различных входных и выходных операндов;
- величина словаря;
- длина модуля.

Операторы и операнды приведены в таблицах 11 и 12 соответственно.

Таблица 11 – Операторы

Операторы	Число вхождений
=	3
!	1
.	11
if	3
else	1
new	2
foreach	2
get	2
set	2
MarkAttendance	1
DisplayAttendance	1
attendanceRecords.ContainsKey()	2
Console.WriteLine	3

Окончание таблицы 11

Операторы	Число вхождений
studentAttendance.Key.Contains()	1
;	5
,	4
{}	8
n1 = 17	N1 = 52

Таблица 12 – Операнды

Операнды	Число вхождений
Student	2
Name	2
IsPresent	2
attendanceRecords	2
Операнды	Число вхождений
subject	4
studentAttendance	1
studentAttendance.Key	1
teacher	4
studentAttendance.Value	6
«Present»	1
«Absent»	1
boolean	2

Окончание таблицы 12

Операнды	Число вхождений
string	6
List<Student>	3
Dictionary<string, boolean>	1
Dictionary<string, Dictionary<string, bool>>	2
n2 = 16	N2 = 40

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 52 + 40 = 92.$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 17 + 16 = 33.$$

Объем программы рассчитывается по формуле:

$$NV = N * \log_2(n) = 92 * 5.05 = 464.6.$$

Уровень программы рассчитывается по формуле:

$$ND = (n1 / 2) * (N2 / n2) = (33 / 2) * (40 / 16) = 41.25.$$

Следует отметить, что уровень программы оказывает двоякое влияние на ее восприятие. Для специалиста, хорошо владеющего языком программирования, программа будет легче для понимания, если она компактна, то есть имеет высокий уровень и малый объем. Однако для менее опытного программиста требуется более объемная и менее абстрактная программа для лучшего понимания.

Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда, из чего можно сделать вывод, что представленный код удобочитаем для специалистов любого уровня [33 – 34].

Рассчитаем также цикломатическую сложность кода. (Control Flow Graph, CFG) – это граф, где узлы представляют базовые блоки кода, а ребра представляют переходы между ними. В статическом анализе кода, CFG может

быть использован для обнаружения потенциальных проблем в коде. Граф потока управления программы для рассматриваемой функции приведен на рисунке 37.



Рисунок 37 – Граф потока управления программы

На основе этого графа вычисляется цикломатическая сложность программы по формуле

$$V(G) = e - n + 2p,$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит 6 вершин, 7 дуг и один компонент связности.

$$V(G) = 7 - 6 + 2 = 3.$$

Таким образом значение цикломатической сложности равно 3.

Это значение не превышает 10, следовательно, функция не является структурно сложной.

Заключение

В ходе выполнения выпускной квалификационной работы была проведена всесторонняя работа по анализу, проектированию и реализации информационной системы контроля учебного процесса.

В первой части работы был осуществлен анализ существующих систем аналогичного назначения, что позволило выявить их сильные и слабые стороны. На основе этого анализа были сформулированы функциональные и нефункциональные требования к разрабатываемому веб-приложению, описанные с помощью UML-диаграмм: диаграмма вариантов использования, диаграмма деятельности, диаграмма последовательности. Особое внимание было уделено обеспечению удобства использования, что является критически важным для успешного внедрения и эксплуатации информационной системы в образовательной среде.

Во второй части работы было выполнено проектирование самой системы. В рамках проектирования базы данных была создана концептуальная схема в нотации Чена, а также логическая модель, подробно описывающая сущности и их взаимосвязи. Это обеспечило структурированное и четкое представление данных, необходимых для функционирования системы. Дополнительно, были разработаны макеты пользовательского интерфейса, которые ориентированы на интуитивное взаимодействие пользователя с системой. Эти макеты обеспечивают наглядное представление о том, как пользователи будут работать с системой, и помогают выявить потенциальные улучшения на ранних этапах разработки.

В третьей части был реализован весь спроектированный функционал. Для серверной части было произведено физическое проектирование базы данных, развертывание базы данных в СУБД Microsoft SQL Server 2020. Клиентская часть была реализована с использованием фреймворка ASP.Net Core и архитектуры MVC.

Также было выполнено тестирование серверной и клиентской частей разрабатываемого приложения. Оно показало корректность его работы. Также рассчитаны метрики программного кода для оценки его качества.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи. Проведенная работа заложила прочную основу для дальнейшей реализации информационной системы контроля учебного процесса, способной удовлетворить потребности образовательных учреждений и повысить эффективность учебного процесса.

Список использованных источников

1. Канюков Н. А., Яхонтова И. М. Международный опыт развития информационных технологий (Окинавская хартия глобального информационного общества) // Colloquium-journal. 2019. №3 – 4(27). URL: <https://cyberleninka.ru/article/n/mezhdunarodnyy-opyt-razvitiya-informatsionnyh-tehnologiy-okinavskaya-hartiya-globalnogo-informatsionnogo-obschestva> (дата обращения: 10.03.2024).
2. Окинавская хартия глобального информационного общества от 21 июля 2000 г. / [Электронный ресурс] // Президент России : [сайт]. – URL: <http://www.kremlin.ru/supplement/3170> (дата обращения: 10.03.2024).
3. Сафуанов Р. М., Лехмус М.Ю., Колганов Е.А. Цифровизация системы образования // Вестник УГНТУ. Наука, образование, экономика. Серия: Экономика . 2019. №2 (28). URL: <https://cyberleninka.ru/article/n/tsifrovizatsiya-sistemy-obrazovaniya> (дата обращения: 10.03.2024).
4. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. – URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 13.03.2024).
5. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. – URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 13.03.2024).
6. Федеральный закон «Об образовании в Российской Федерации» № 273 – ФЗ от 29.12.2012 года с изменениями 2023 года. [Электронный ресурс]. – URL: <http://zakon-ob-obrazovanii.ru/> (дата обращения: 13.03.2024).
7. Единая информационная система управления учебным процессом / [Электронный ресурс] // ТАНДЕМ. Университет : [сайт]. – URL: <https://tandemservice.ru/products/tandem-university/> (дата обращения: 21.04.2024).

8. 1С:Университет / [Электронный ресурс] // 1С:Отраслевые реше : [сайт]. – URL: <https://solutions.1c.ru/catalog/university/> (дата обращения: 21.04.2024).

9. О проекте / [Электронный ресурс] // Электронный деканат (Free Dean's Office) : [сайт]. – URL: <https://www.deansoffice.ru/> (дата обращения: 21.04.2024).

10. Ionut Andone For how long different age groups use the phone daily / Ionut Andone [Электронный ресурс] // Research Gate: [сайт]. – URL: <https://www.researchgate.net/figure/For-how-long-different-age-groups-use-the-phone-daily> (дата обращения: 21.04.2024).

11. Digital 2024 April Global Statshot Report / [Электронный ресурс] // Datareportal: [сайт]. – URL: <https://datareportal.com/reports/digital-2024-april-global-statshot> (дата обращения: 21.04.2024).

12. Статистика использования браузеров и мобильных устройств в 2023 году / [Электронный ресурс] // L-TECH : [сайт]. – URL: <https://l-tech.ru/posts/statistika-ispolzovaniya-brauzerov-i-mobilnyh-ustroystv-v-2023-godu> (дата обращения: 21.04.2024).

13. Спицина, И. А. Применение системного анализа при разработке пользовательского интерфейса информационных систем [Учебное пособие] / И. А. Спицина –. – Екатеринбург: Издательство Уральского университета, 2018 – **1 с.**

14. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

15. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

16. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. –102 с.

17. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.

18. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
19. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.
20. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
21. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.
22. Джонсон, Т., Четфилд К. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.
23. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
24. Унифицированный процесс Rational (RUP) // Интернет-технологии.ру [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/newbie/unificirovannyu-process-rational-rup.html> (дата обращения: 25.05.2024).
25. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
26. Точилкина, Т.Е. Принципы создания информационных систем и моделирования бизнес-процессов с использованием пакета программ AllFusion Modeling Suite. Часть II. Моделирование данных и проектирование баз данных с ERwin Data Modeler. Учебное пособие./ Т.Е. Точилкина – М.: Академия бюджета и казначейства, 2009 – 167 с.
27. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 17.05.2024).
28. Андерсон Р., Ларкин К., Уоссон М. Создание веб-API с помощью ASP.NET Core и Visula Studio для Windows // Microsoft [Электронный ресурс].

URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> (дата обращения: 25.05.2024).

29. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 25.05.2024).

30. Database First // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first> (дата обращения: 25.05.2024).

31. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 25.05.2024).

32. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 25.05.2024).

33. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.

34. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
(справочное)
Глоссарий

Web-приложение – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

СУБД (Система Управления Базой Данных) – специализированное программное обеспечение, предоставляющее программный интерфейс, который

дает возможность управлять созданием и использованием баз данных

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит использование итеративной модели разработки.

UML (Unified Modeling Language) – язык графического описания, с помощью которого осуществляется объектное моделирование в области разработки программного обеспечения, проектирования систем и отображения организационных структур.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных, работающей под управлением соответствующей системой управления базами данных, а также для управления этими данными.

HTTPS (HyperText Transfer Protocol Secured) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

Модульное тестирование (Unit testing) – тестирование каждой атомарной функциональности приложения отдельно, в искусственно созданной среде.

Приложение Б
(рекомендуемое)
Исходный код создания базы данных

```

USE [master]
GO
/***** Object:  Database [Assignment Part B - G20760401]      *****/
CREATE DATABASE [Assignment Part B - G20760401]
GO
ALTER DATABASE [Assignment Part B - G20760401] SET COMPATIBILITY_LEVEL = 130
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Assignment Part B - G20760401].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ARITHABORT OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Assignment Part B - G20760401] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [Assignment Part B - G20760401] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET  DISABLE_BROKER
GO
ALTER DATABASE [Assignment Part B - G20760401] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [Assignment Part B - G20760401] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET RECOVERY SIMPLE
GO
ALTER DATABASE [Assignment Part B - G20760401] SET  MULTI_USER
GO
ALTER DATABASE [Assignment Part B - G20760401] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [Assignment Part B - G20760401] SET DB_CHAINING OFF
GO
ALTER DATABASE [Assignment Part B - G20760401] SET FILESTREAM( NON_TRANSACTED_ACCESS
= OFF )
GO

```

```

ALTER DATABASE [Assignment Part B - G20760401] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [Assignment Part B - G20760401] SET DELAYED_DURABILITY = DISABLED
GO
ALTER DATABASE [Assignment Part B - G20760401] SET QUERY_STORE = OFF
GO
USE [Assignment Part B - G20760401]
GO
ALTER DATABASE SCOPED CONFIGURATION SET LEGACY_CARDINALITY_ESTIMATION = OFF;
GO
ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 0;
GO
ALTER DATABASE SCOPED CONFIGURATION SET PARAMETER_SNIFFING = ON;
GO
ALTER DATABASE SCOPED CONFIGURATION SET QUERY_OPTIMIZER_HOTFIXES = OFF;
GO
USE [Assignment Part B - G20760401]
GO
/***** Object: Table [dbo].[Academic] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Academic](
    [user_id] [int] NOT NULL,
    [code] [nvarchar](10) NOT NULL,
    [department_id] [int] NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_ACADEMIC] PRIMARY KEY CLUSTERED
(
    [user_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Academic_Period] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Academic_Period](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [period] AS
(concat(datepart(year,[start_date]), '/', substring(CONVERT([nvarchar](4),datepart(yea
r,[end_date])),(3),(2))),
    [start_date] [date] NOT NULL,
    [end_date] [date] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_ACADEMIC_PERIOD] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Admin] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Admin](
    [user_id] [int] NOT NULL,
    [admin_office_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,

```

```

CONSTRAINT [PK_Admin] PRIMARY KEY CLUSTERED
(
    [user_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Admin_Office]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Admin_Office](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [university_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Admin_Office] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Department]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Department](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](100) NOT NULL,
    [school_id] [int] NOT NULL,
    [head_id] [int] NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_DEPARTMENT] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Gender]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Gender](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](6) NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Gender] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Module]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Module](
    [id] [int] IDENTITY(1,1) NOT NULL,

```

```

        [name] [nvarchar](100) NOT NULL,
        [code] [char](6) NOT NULL,
        [department_id] [int] NOT NULL,
        [last_update] [datetime] NOT NULL,
        [last_updated_by] [nvarchar](max) NOT NULL,
        CONSTRAINT [PK_MODULE] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Module_Delivery_Type] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Module_Delivery_Type] (
    [id] [int] IDENTITY(1,1) NOT NULL,
    [type] [nvarchar](15) NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Module_Delivery] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Module_Period] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Module_Period] (
    [id] [int] IDENTITY(1,1) NOT NULL,
    [module_id] [int] NOT NULL,
    [period_id] [int] NOT NULL,
    [module_delivery_type_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_MODULE_PERIOD] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Module_Period_Session_Type]      Script Date: 05/04/2020
04:10:39 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Module_Period_Session_Type] (
    [id] [int] IDENTITY(1,1) NOT NULL,
    [module_period_id] [int] NOT NULL,
    [session_type_id] [int] NOT NULL,
    [session_recurrency_type_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_MODULE_PERIOD_SESSION_TYPE] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

```

```

GO
/***** Object: Table [dbo].[Programme]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Programme](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](100) NOT NULL,
    [department_id] [int] NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_PROGRAMME] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Programmes_Modules]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Programmes_Modules](
    [programme_id] [int] NOT NULL,
    [module_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Room]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Room](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [code] [nvarchar](10) NOT NULL,
    [university_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_ROOM] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[School]      *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[School](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](100) NOT NULL,
    [university_id] [int] NOT NULL,
    [head_id] [int] NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_SCHOOL] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

```

```

GO
/***** Object: Table [dbo].[Session] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Session](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [module_period_session_type_id] [int] NOT NULL,
    [date] [date] NOT NULL,
    [start_time] [time](0) NOT NULL,
    [end_time] [time](0) NOT NULL,
    [tutor_id] [int] NOT NULL,
    [leader_id] [int] NULL,
    [room_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_SESSION] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Session_Recurrency_Type] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Session_Recurrency_Type](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [type] [nvarchar](15) NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Session_Recurrency_Type] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Session_Type] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Session_Type](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [type] [nvarchar](50) NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_SESSION_TYPE] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Student] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Student](
    [user_id] [int] NOT NULL,
    [code] [int] NOT NULL,
    [library_card_number] [int] NOT NULL,

```

```

        [last_update] [datetime] NOT NULL,
        [last_updated_by] [nvarchar](max) NOT NULL,
        CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
    (
        [user_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Student_Support] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Student_Support](
    [user_id] [int] NOT NULL,
    [student_support_office_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Student_Support] PRIMARY KEY CLUSTERED
    (
        [user_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Student_Support_Office] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Student_Support_Office](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [university_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_Student_Support_Office] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Students_Module_Periods_Select] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Students_Module_Periods_Select](
    [student_id] [int] NOT NULL,
    [module_period_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[Students_Programmes_Enroll] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Students_Programmes_Enroll](
    [programme_id] [int] NOT NULL,
    [student_id] [int] NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

```

```

/***** Object: Table [dbo].[Students_Sessions_Attend] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Students_Sessions_Attend](
    [session_id] [int] NOT NULL,
    [student_id] [int] NOT NULL,
    [attendance] [nchar](1) NOT NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[University] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[University](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [name] [nchar](50) NOT NULL,
    [rector_id] [int] NULL,
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_UNIVERSITY] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/***** Object: Table [dbo].[User] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[User](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [first_name] [nvarchar](50) NOT NULL,
    [middle_name] [nvarchar](50) NOT NULL,
    [last_name] [nvarchar](50) NOT NULL,
    [password] [nvarchar](50) NOT NULL,
    [gender_id] [int] NOT NULL,
    [email] AS
(concat(substring([first_name],(1),(1)),substring([middle_name],(1),(1)),[last_name]
,[id], '@email.com')),
    [last_update] [datetime] NOT NULL,
    [last_updated_by] [nvarchar](max) NOT NULL,
    CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

```

Приложение В
(рекомендуемое)
Исходный код контроллеров

Листинг В.1 – AdminController.cs

```

namespace StudentAttendanceManagementSystem.Controllers
{
    [Route («[controller]»)]
    public class AdminController : Controller
    {
        private readonly IProgramRepository _programRepository;
        private readonly ISubjectRepository _subjectRepository;
        private readonly ILoginRepository _loginRepository;
        private readonly IStudentRepository _studentRepository;
        private readonly IFacultyRepository _facultyRepository;
        private readonly IStudentSubjectRepository _studentSubjectRepository;
        public AdminController(IProgramRepository programRepository,
            ISubjectRepository subjectRepository,
            ILoginRepository loginRepository,
            IStudentRepository studentRepository,
            IFacultyRepository facultyRepository,
            IStudentSubjectRepository studentSubjectRepository)
        {
            this._programRepository = programRepository;
            this._subjectRepository = subjectRepository;
            this._loginRepository = loginRepository;
            this._studentRepository = studentRepository;
            this._facultyRepository = facultyRepository;
            this._studentSubjectRepository = studentSubjectRepository;
        }

        [Route («[action]»)]
        public IActionResult Admin()
        {
            string userType = HttpContext.Session.GetString («UserType»);
            if («Admin» != userType)
                return RedirectToAction («Login», «Home»);

            SetMessages ();

            List<Models.Program> programs =
            _programRepository.GetAllPrograms ().ToList ();
            ViewData[«programList»] = programs;
            return View ();
        }

        [HttpPost]
        [Route («[action]»)]
        public IActionResult AddNewProgram (Models.Program newProgram)
        {
            if (ModelState.IsValid)
            {
                HttpContext.Session.SetString («success», «Программа « +
                newProgram.ProgramName + « успешно добавлена»);
                _programRepository.Add (newProgram);
                return RedirectToAction («Admin», «Admin»);
            }
            HttpContext.Session.SetString («error», «Неверно заполнены поля»);
            return RedirectToAction («Admin», «Admin»);
        }

        [Route («[action]/{id}»)]
        public IActionResult ProgramDetails (int Id)
        {
            Models.Program currentProgram = _programRepository.GetById (Id);
            if (null == currentProgram)
            {

```

```

        HttpContext.Session.SetString(«error», «Программа не найдена»);
        return RedirectToAction(«Admin», «Admin»);
    }
    SetMessages();
    List<Faculty> faculties = _facultyRepository.GetAll(Id).ToList();
    List<Subject> subjects =
        _subjectRepository.GetAllByProgramId(Id).ToList();

    ViewData[«Program»] = currentProgram;
    ViewData[«Subjects»] = subjects;
    ViewData[«Faculties»] = faculties;
    return View();
}

[Route(«[action]»)]
public IActionResult AddNewSubject(Subject newSubject)
{
    Subject isExist =
        _subjectRepository.GetBySubjectCode(newSubject.SubjectCode);
    if (null != isExist)
    {
        HttpContext.Session.SetString(«error», newSubject.SubjectCode + «
уже существует»);
    }
    else
    {
        Subject subject = _subjectRepository.Add(newSubject);
        List<Student> allStudents =
            _studentRepository.GetByProgramId(newSubject.ProgramId).ToList();
        Subject newsub =
            _subjectRepository.GetBySubjectCode(newSubject.SubjectCode);
        if (null != newsub && null != allStudents)
        {
            for (int i = 0; i < allStudents.Count; i++)
            {
                StudentSubject stusub = new StudentSubject();
                stusub.Attendance = 0;
                stusub.SemesterNo = newSubject.SemesterNo;
                stusub.StudentId = allStudents[i].Id;
                stusub.SubjectId = newsub.Id;
                _studentSubjectRepository.Add(stusub);
            }
        }
        HttpContext.Session.SetString(«success», newSubject.SubjectName + «
Предмет добавлен»);
    }
    return RedirectToAction(«ProgramDetails», «Admin», new { id =
newSubject.ProgramId });
}

[Route(«[action]/{id}»)]
public IActionResult AddNewStudent(int Id)
{
    SetMessages();
    Models.Program currentProgram = _programRepository.GetById(Id);
    if (null == currentProgram)
    {
        HttpContext.Session.SetString(«error», «Программа не существует»);
        return RedirectToAction(«ProgramDetails», «Admin», new { id = Id });
    }
    ViewData[«Program»] = currentProgram;
    return View();
}

[HttpPost]
[Route(«[action]/{id}»)]
public IActionResult AddStudent(Student newStudent)
{

```

```

newStudent.ActualDOB = newStudent.DOB.ToShortDateString();
newStudent.Password = newStudent.ActualDOB.ToString();
if (ModelState.IsValid)
{
    Login isExist =
_loginRepository.GetByUserName(newStudent.UniversityId);
    if (null != isExist)
    {
        HttpContext.Session.SetString(«error», «Студент уже добавлен»);
    }
    else
    {
        Login newStudentLogin = new Login();
        newStudentLogin.Username = newStudent.UniversityId;
        newStudentLogin.Password = newStudent.Password;
        Login addNewLogin = _loginRepository.Add(newStudentLogin);
        Login newLogin =
_loginRepository.GetLogin(newStudent.UniversityId, newStudent.Password);
        if (null == newLogin)
        {
            HttpContext.Session.SetString(«error», «Что-то пошло не
так»);
        }
        else
        {
            newStudent.LoginId = newLogin.Id;
            Student newAddedStudent =
_studentRepository.Add(newStudent);
            Student newstu =
_studentRepository.GetByLoginId(newStudent.LoginId);
            Models.Program stuprogram =
_programRepository.GetById(newStudent.ProgramId);
            if (null != newstu && null != stuprogram)
            {
                for (int i = 1; i <= 8; i++)
                {
                    List<Subject> subjects =
_subjectRepository.GetAllBySemesterProgramId(i, newStudent.ProgramId).ToList();
                    for (int j = 0; j < subjects.Count; j++)
                    {
                        StudentSubject stusub = new StudentSubject();
                        stusub.Attendance = 0;
                        stusub.SemesterNo = i;
                        stusub.StudentId = newstu.Id;
                        stusub.SubjectId = subjects[j].Id;
                        _studentSubjectRepository.Add(stusub);
                    }
                }
                HttpContext.Session.SetString(«success»,
newStudent.UniversityId + « успешно добавлен»);
            }
            else
            {
                // remove login details
                HttpContext.Session.SetString(«error», «Что-то пошло не
так (SERVER ERROR).»);
            }
        }
    }
}
else
{
    HttpContext.Session.SetString(«error», «Некорректный ввод»);
}
return RedirectToAction(«AddNewStudent», «Admin», new { id =
newStudent.ProgramId });

```

```

    }

    // add faculty
    [Route («[action]/{id}»)]
    public IActionResult AddNewFaculty(int Id)
    {
        SetMessages();
        Models.Program currentProgram = _programRepository.GetById(Id);
        if (null == currentProgram)
        {
            HttpContext.Session.SetString («error», «Программа не существует»);
            return RedirectToAction («ProgramDetails», «Admin», new { id = Id });
        }
        List<Subject> subjects =
        _subjectRepository.GetAllByProgramId(Id).ToList<Subject>();
        ViewData[«Program»] = currentProgram;
        ViewData[«Subjects»] = subjects;
        return View();
    }
    [HttpPost]
    [Route («[action]/{id}»)]
    public IActionResult AddFaculty(Faculty faculty)
    {
        if (ModelState.IsValid)
        {
            if (-1 == faculty.SubjectId)
            {
                HttpContext.Session.SetString («error», «Выберите дисциплину»);
            }
            else
            {
                Faculty fsub =
                _facultyRepository.GetBySubjectId(faculty.SubjectId);
                if (null != fsub)
                {
                    HttpContext.Session.SetString («error», «Дисциплина уже
назначена»);
                }
                else
                {
                    Faculty fem = _facultyRepository.GetByEmail(faculty.Email);
                    if (null != fem)
                    {
                        HttpContext.Session.SetString («error», «Эл. почта уже
используется»);
                    }
                    else
                    {
                        faculty.Password = faculty.FirstName + «@PSU»;
                        Login nLogin = new Login();
                        nLogin.Username = faculty.Email;
                        nLogin.Password = faculty.Password;
                        nLogin = _loginRepository.Add(nLogin);
                        if (null == nLogin)
                        {
                            HttpContext.Session.SetString («error», «Что-то пошло
не так»);
                        }
                        else
                        {
                            faculty.LoginId = nLogin.Id;
                            if (null == _facultyRepository.Add(faculty))
                            {
                                HttpContext.Session.SetString («error», «Что-то
пошло не так»);
                            }
                        }
                    }
                }
            }
        }
    }

```

```

        {
            HttpContext.Session.SetString(«success»,
faculty.FirstName + « успешно добавлен»);
        }
    }
}
}
}
return RedirectToAction(«AddNewFaculty», «Admin», new { id =
faculty.ProgramId });
}

private void SetMessages()
{
    ViewData[«ErrorMessage»] = HttpContext.Session.GetString(«error»);
    ViewData[«SuccessMessage»] = HttpContext.Session.GetString(«success»);
    HttpContext.Session.Remove(«error»);
    HttpContext.Session.Remove(«success»);
}
}
}

```

Листинг В.2 – FacultyController.cs

namespace StudentAttendanceManagementSystem.Controllers

```

{
    [Route(«[controller]»)]
    public class FacultyController : Controller
    {
        private readonly IProgramRepository _programRepository;
        private readonly ISubjectRepository _subjectRepository;
        private readonly ILoginRepository _loginRepository;
        private readonly IStudentRepository _studentRepository;
        private readonly IFacultyRepository _facultyRepository;
        private readonly IStudentSubjectRepository _studentSubjectRepository;
        public FacultyController(IProgramRepository programRepository,
            ISubjectRepository subjectRepository,
            ILoginRepository loginRepository,
            IStudentRepository studentRepository,
            IFacultyRepository facultyRepository,
            IStudentSubjectRepository studentSubjectRepository)
        {
            this._programRepository = programRepository;
            this._subjectRepository = subjectRepository;
            this._loginRepository = loginRepository;
            this._studentRepository = studentRepository;
            this._facultyRepository = facultyRepository;
            this._studentSubjectRepository = studentSubjectRepository;
        }

        [Route(«[action]»)]
        public IActionResult Faculty()
        {
            string userType = HttpContext.Session.GetString(«UserType»);
            if («Faculty» != userType)
                return RedirectToAction(«Login», «Home»);

            int fcid = Convert.ToInt32(HttpContext.Session.GetInt32(«FacultyId»));
            Faculty faculty = _facultyRepository.GetById(fcid);
            if(null == faculty)
            {
                HttpContext.Session.SetString(«error», «Что-то пошло не так (SERVER
ERROR) .»);
            }
            else
            {

```

```

        List<StudentSubject> studentSubjects =
        _studentSubjectRepository.GetBySubjectId(faculty.SubjectId).ToList();
        for(int i = 0; i < studentSubjects.Count; i++)
        {
            studentSubjects[i].Student =
            _studentRepository.GetById(studentSubjects[i].StudentId);
        }
        Subject currentSubject =
        _subjectRepository.GetById(faculty.SubjectId);
        if(null == currentSubject)
        {
            HttpContext.Session.SetString(«error», «Что-то пошло не так
(SERVER ERROR).»);
        }
        else
        {
            ViewData[«StudentSubjectList»] = studentSubjects;
            ViewData[«Subject»] = currentSubject;
            ViewData[«Faculty»] = faculty;
        }
    }
    SetMessages();

    return View();
}

[HttpPost]
[Route(«[action]»)]
public IActionResult Update(StudentSubject updateSS)
{
    if (ModelState.IsValid)
    {
        _studentSubjectRepository.Update(updateSS);
        HttpContext.Session.SetString(«success», «Посещаемость добавлена»);
        return RedirectToAction(«UpdateAttendance», «Faculty», new { objId =
updateSS.Id });
    }
    HttpContext.Session.SetString(«error», «ERROR-404 (Страница не
найдена)»);
    return RedirectToAction(«Faculty», «Faculty»);
}

private int getOldAttendance(int id)
{
    StudentSubject oldss = _studentSubjectRepository.GetById(id);
    if (null == oldss)
        return -1;
    return oldss.Attendance;
}

private void SetMessages()
{
    ViewData[«ErrorMessage»] = HttpContext.Session.GetString(«error»);
    ViewData[«SuccessMessage»] = HttpContext.Session.GetString(«success»);
    HttpContext.Session.Remove(«error»);
    HttpContext.Session.Remove(«success»);
}
}
}

```

Листинг Б.3 – HomeController.cs

```

namespace StudentAttendanceManagementSystem.Controllers
{
    [Route(«[controller]»)]
    public class HomeController : Controller
    {
        private readonly ILoginRepository _loginRepository;
    }
}

```

```

private readonly IAdminRepository _adminRepository;
private readonly IFacultyRepository _facultyRepository;
private readonly IStudentRepository _studentRepository;
private readonly IProgramRepository _programRepository;
private readonly IStudentSubjectRepository _studentSubjectRepository;
private readonly ISubjectRepository _subjectRepository;
public HomeController(ILoginRepository loginRepository,
    IAdminRepository adminRepository,
    IFacultyRepository facultyRepository,
    IStudentRepository studentRepository,
    IProgramRepository programRepository,
    IStudentSubjectRepository studentSubjectRepository,
    ISubjectRepository subjectRepository)
{
    this._loginRepository = loginRepository;
    this._adminRepository = adminRepository;
    this._facultyRepository = facultyRepository;
    this._studentRepository = studentRepository;
    this._programRepository = programRepository;
    this._studentSubjectRepository = studentSubjectRepository;
    this._subjectRepository = subjectRepository;
}

[Route («[action]»)]
[Route («~/»)]
[Route («»)]
public IActionResult Index()
{
    SetMessages();
    string userType = HttpContext.Session.GetString («UserType»);
    if (null == userType)
        return RedirectToAction («Login»);

    if («Admin» == userType)
        return RedirectToAction («Admin», «Admin»);

    if («Faculty» == userType)
        return RedirectToAction («Faculty», «Faculty»);
    int stuid = Convert.ToInt32(HttpContext.Session.GetInt32 («StudentId»));
    Student student = _studentRepository.GetById(stuid);
    if (null != student)
    {
        int programid = student.ProgramId;
        Models.Program program = _programRepository.GetById(programid);
        if (null != program)
        {
            ViewData[«StudentObj»] = student;
            ViewData[«ProgramObj»] = program;
            return View();
        }
    }

    HttpContext.Session.SetString («error», «Что-то пошло не так (SERVER
ERROR)»);
    SetMessages();
    return View();
}

[HttpGet]
[Route («[action]»)]
public IActionResult Login()
{
    SetMessages();
    if (null != HttpContext.Request.Cookies[«UserType»])
    {
        string utype = HttpContext.Request.Cookies[«UserType»];
        int uid = 0;
    }
}

```

```

        if («Admin» == utype)
        {
            uid = Convert.ToInt32(HttpContext.Session.GetInt32 («AdminId»));
            HttpContext.Session.SetInt32 («AdminId», uid);
        }
        else if («Faculty» == utype)
        {
            uid =
Convert.ToInt32(HttpContext.Session.GetInt32 («FacultyId»));
            HttpContext.Session.SetInt32 («FacultyId», uid);
        }
        else
        {
            uid =
Convert.ToInt32(HttpContext.Session.GetInt32 («StudentId»));
            HttpContext.Session.SetInt32 («StudentId», uid);
        }
        HttpContext.Session.SetString («UserType», utype);
    }

    string userType = HttpContext.Session.GetString («UserType»);
    if («Admin» == userType)
        return RedirectToAction («Admin», «Admin»);

    if («Faculty» == userType)
        return RedirectToAction («Faculty», «Faculty»);

    if («Student» == userType)
        return RedirectToAction («Index»);
    return View();
}

[HttpPost]
[Route («[action]»)]
public IActionResult Login(Login _login)
{
    CookieOptions cookieOptions = new CookieOptions();
    cookieOptions.Expires = new DateTimeOffset (DateTime.Now.AddMonths(1));
    if (ModelState.IsValid)
    {
        Login login = _loginRepository.GetLogin(_login.Username,
_login.Password);
        if (null != login)
        {
            Admin _admin = _adminRepository.GetAdminByLoginId(login.Id);
            if (null != _admin)
            {
                HttpContext.Session.SetString («UserType», «Admin»);
                HttpContext.Session.SetInt32 («AdminId», _admin.Id);
                HttpContext.Session.SetString («success», _admin.Name + «,
добро пожаловать!»);
                if (_login.remember)
                {
                    HttpContext.Response.Cookies.Append («UserType», «Admin»,
cookieOptions);
                    HttpContext.Response.Cookies.Append («AdminId»,
_admin.Id.ToString(), cookieOptions);
                }
                return RedirectToAction («Admin», «Admin»);
            }
            Faculty _faculty = _facultyRepository.GetByLoginId(login.Id);
            if (null != _faculty)
            {
                HttpContext.Session.SetString («UserType», «Faculty»);
                HttpContext.Session.SetInt32 («FacultyId», _faculty.Id);
            }
        }
    }
}

```

```

        HttpContext.Session.SetString(«success», _faculty.FirstName
+ «, добро пожаловать!»);
        if (_login.remember)
        {
            HttpContext.Response.Cookies.Append(«UserType»,
«Faculty», cookieOptions);
            HttpContext.Response.Cookies.Append(«FacultyId»,
_faculty.Id.ToString(), cookieOptions);
        }
        return RedirectToAction(«Faculty», «Faculty»);
    }
    Student _student = _studentRepository.GetByLoginId(login.Id);
    if (null != _student)
    {
        HttpContext.Session.SetString(«UserType», «Student»);
        HttpContext.Session.SetInt32(«StudentId», _student.Id);
        HttpContext.Session.SetString(«success», _student.FirstName
+ «, добро пожаловать!»);
        if (_login.remember)
        {
            HttpContext.Response.Cookies.Append(«UserType»,
«Student», cookieOptions);
            HttpContext.Response.Cookies.Append(«StudentId»,
_student.Id.ToString(), cookieOptions);
        }
        return RedirectToAction(«Index», «Home»);
    }
    HttpContext.Session.SetString(«error», «Что-то пошло не так»);
}
else
{
    HttpContext.Session.SetString(«error», «Неверные данные»);
}
}
return RedirectToAction(«Login», «Home»);
}

[Route(«[action]/{program}/{semester}»)]
public IActionResult Attendance(int semester, int program)
{
    int stuid = Convert.ToInt32(HttpContext.Session.GetInt32(«StudentId»));
    Models.Program stuProgram = _programRepository.GetById(program);
    if((null != stuProgram) && (1 <= semester) && (semester <= 8))
    {
        List<StudentSubject> studentSubjects =
        _studentSubjectRepository.getBySemesterNoStudentId(semester, stuid).ToList();
        for(int i = 0; i < studentSubjects.Count; i++)
        {
            Subject sub =
            _subjectRepository.GetById(studentSubjects[i].SubjectId);
            studentSubjects[i].Subject = sub;
        }
        Student cstu = _studentRepository.GetById(stuid);
        ViewData[«Student»] = cstu;
        ViewData[«SubjectWiseAttendance»] = studentSubjects;
        return View();
    }
    HttpContext.Session.SetString(«error», «ERROR-404 (Страница не
найдена)»);
    return RedirectToAction(«Index», «Home»);
}

[Route(«[action]»)]
public IActionResult Logout()
{
    HttpContext.Session.Remove(«UserType»);
}

```

```

        HttpContext.Session.Remove («AdminId»);
        HttpContext.Session.Remove («FacultyId»);
        HttpContext.Session.Remove («StudentId»);

        Response.Cookies.Delete («UserType»);
        Response.Cookies.Delete («AdminId»);
        Response.Cookies.Delete («FacultyId»);
        Response.Cookies.Delete («StudentId»);
        return RedirectToAction («Login», «Home»);
    }

    private void SetMessages()
    {
        ViewData[«ErrorMessage»] = HttpContext.Session.GetString («error»);
        ViewData[«SuccessMessage»] = HttpContext.Session.GetString («success»);
        HttpContext.Session.Remove («error»);
        HttpContext.Session.Remove («success»);
    }
}

```

Приложение Г
(рекомендуемое)
Исходный код моделей

Листинг Г.1 – DAO/IAdminRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface IAdminRepository
    {
        Admin GetAdmin(int Id);
        Admin GetAdminByLoginId(int LoginId);
    }
}
```

Листинг Г.2 – DAO/IFacultyRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface IFacultyRepository
    {
        Faculty GetById(int Id);
        Faculty GetByLoginId(int loginId);
        IEnumerable<Faculty> GetByProgramId(int programId);
        IEnumerable<Faculty> GetAll(int programId);
        Faculty GetBySubjectId(int subjectId);
        Faculty GetByEmail(string email);
        Faculty Add(Faculty faculty);
    }
}
```

Листинг Г.3 – DAO/ILoginRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface ILoginRepository
    {
        Login GetLogin(string username, string password);
        Login GetByUserName(string username);
        Login Add(Login login);
        Login Update(Login login);
        Login Delete(int Id);
    }
}
```

Листинг Г.4 – DAO/IProgramRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface IProgramRepository
    {
        Program Add(Program program);
        Program GetById(int id);
        IEnumerable<Program> GetAllPrograms();
    }
}
```

Листинг Г.5 – DAO/IStudentRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface IStudentRepository
    {
        Student Add(Student student);
        Student GetById(int Id);
        Student GetByUniversityId(string UniversityId);
        Student GetByLoginId(int LoginId);
        IEnumerable<Student> GetByProgramId(int ProgramId);
    }
}
```

```

    }
}

```

Листинг Г.6 – DAO/IStudentSubjectRepository.cs

```
using System.Collections.Generic;
```

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface IStudentSubjectRepository
    {
        IEnumerable<StudentSubject> getBySemesterNoStudentId(int semesterNo, int
studentId);
        IEnumerable<StudentSubject> GetBySubjectId(int subjectId);
        StudentSubject Update(StudentSubject studentSubject);
        StudentSubject GetById(int id);
        StudentSubject Add(StudentSubject studentSubject);
    }
}

```

Листинг Г.7 – DAO/ISubjectRepository.cs

```
using System.Collections.Generic;
```

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public interface ISubjectRepository
    {
        Subject Add(Subject subject);
        Subject GetById(int Id);
        Subject GetBySubjectCode(string code);
        IEnumerable<Subject> GetAllByProgramId(int ProgramId);
        IEnumerable<Subject> GetAllBySemesterProgramId(int SemesterNo, int
ProgramId);
    }
}

```

Листинг Г.8 – DAO/SQLAdminRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public class SQLAdminRepository : IAdminRepository
    {
        private readonly AppDbContext _context;
        public SQLAdminRepository(AppDbContext context)
        {
            _context = context;
        }

        Admin IAdminRepository.GetAdmin(int id)
        {
            return _context.Admin.FirstOrDefault(m => m.Id == id);
        }
        Admin IAdminRepository.GetAdminByLoginId(int loginId)
        {
            return _context.Admin.FirstOrDefault(m => m.LoginId == loginId);
        }
    }
}

```

Листинг Г.9 – DAO/SQLFacultyRepository.cs

```
namespace StudentAttendanceManagementSystem.Models.DAO
{
    public class SQLFacultyRepository : IFacultyRepository
    {
        private readonly AppDbContext context;
        public SQLFacultyRepository(AppDbContext context)
        {

```

```

        this.context = context;
    }
    Faculty IFacultyRepository.GetById(int Id)
    {
        return context.Faculty.FirstOrDefault(m => m.Id == Id);
    }

    Faculty IFacultyRepository.GetByLoginId(int loginId)
    {
        return context.Faculty.FirstOrDefault(m => m.LoginId == loginId);
    }

    IEnumerable<Faculty> IFacultyRepository.GetByProgramId(int programId)
    {
        return context.Faculty.Where(m => m.ProgramId == programId);
    }

    IEnumerable<Faculty> IFacultyRepository.GetAll(int programId)
    {
        return context.Faculty.Where(m => m.ProgramId == programId);
    }

    Faculty IFacultyRepository.GetBySubjectId(int subjectId)
    {
        return context.Faculty.FirstOrDefault(m => m.SubjectId == subjectId);
    }

    Faculty IFacultyRepository.GetByEmail(string email)
    {
        return context.Faculty.FirstOrDefault(m => m.Email == email);
    }

    Faculty IFacultyRepository.Add(Faculty newFaculty)
    {
        context.Faculty.Add(newFaculty);
        context.SaveChanges();
        return newFaculty;
    }
}

```

Листинг Г.10 – DAO/SQLLoginRepository.cs

namespace StudentAttendanceManagementSystem.Models.DAO

```

{
    public class SQLLoginRepository : ILoginRepository
    {
        private readonly AppDbContext context;
        public SQLLoginRepository(AppDbContext context)
        {
            this.context = context;
        }
        Login ILoginRepository.Add(Login login)
        {
            context.Login.Add(login);
            context.SaveChanges();
            return login;
        }
        Login ILoginRepository.Update(Login loginChanges)
        {
            var login = context.Login.Attach(loginChanges);
            login.State = Microsoft.EntityFrameworkCore.EntityState.Modified;
            context.SaveChanges();
            return loginChanges;
        }
        Login ILoginRepository.Delete(int Id)
        {
            Login login = context.Login.Find(Id);
            if (null != login)

```

```

        {
            context.Login.Remove(login);
            context.SaveChanges();
        }
        return login;
    }

    Login ILoginRepository.GetLogin(string username, string password)
    {
        return context.Login.FirstOrDefault(m => m.Username == username &&
m.Password == password);
    }

    Login ILoginRepository.GetByUserName(string username)
    {
        return context.Login.FirstOrDefault(m => m.Username == username);
    }
}

```

Листинг Г.11 – DAO/SQLProgramRepository.cs

```

namespace StudentAttendanceManagementSystem.Models.DAO
{
    public class SQLProgramRepository : IProgramRepository
    {
        private readonly AppDbContext _context;
        public SQLProgramRepository(AppDbContext context)
        {
            _context = context;
        }

        Program IProgramRepository.Add(Program newProgram)
        {
            _context.Program.Add(newProgram);
            _context.SaveChanges();
            return newProgram;
        }

        IEnumerable<Program> IProgramRepository.GetAllPrograms()
        {
            return _context.Program;
        }

        Program IProgramRepository.GetById(int id)
        {
            return _context.Program.FirstOrDefault(m => m.Id == id);
        }
    }
}

```

Листинг Г.12 – DAO/SQLStudentRepository.cs

```

namespace StudentAttendanceManagementSystem.Models.DAO
{
    public class SQLStudentRepository : IStudentRepository
    {
        private readonly AppDbContext context;
        public SQLStudentRepository(AppDbContext context)
        {
            this.context = context;
        }

        Student IStudentRepository.Add(Student student)
        {
            context.Student.Add(student);
            context.SaveChanges();
            return student;
        }

        Student IStudentRepository.GetById(int Id)
        {

```

```

        return context.Student.FirstOrDefault(m => m.Id == Id);
    }

    Student IStudentRepository.GetByUniversityId(string UniversityId)
    {
        return context.Student.FirstOrDefault(m => m.UniversityId ==
UniversityId);
    }
    Student IStudentRepository.GetByLoginId(int LoginId)
    {
        return context.Student.FirstOrDefault(m => m.LoginId == LoginId);
    }
    IEnumerable<Student> IStudentRepository.GetByProgramId(int ProgramId)
    {
        return context.Student.Where(m => m.ProgramId == ProgramId);
    }
}

```

Листинг Г.13 – DAO/SQLStudentSubjectRepository.cs

```

namespace StudentAttendanceManagementSystem.Models.DAO
{
    public class SQLStudentSubjectRepository : IStudentSubjectRepository
    {
        private readonly AppDbContext context;
        public SQLStudentSubjectRepository(AppDbContext context)
        {
            this.context = context;
        }

        IEnumerable<StudentSubject>
        IStudentSubjectRepository.getBySemesterNoStudentId(int semesterNo, int studentId)
        {
            return context.StudentSubject.Where(m => m.SemesterNo==semesterNo &&
m.StudentId==studentId);
        }

        IEnumerable<StudentSubject> IStudentSubjectRepository.GetBySubjectId(int
subjectId)
        {
            return context.StudentSubject.Where(m => m.SubjectId == subjectId);
        }

        StudentSubject IStudentSubjectRepository.GetById(int id)
        {
            return context.StudentSubject.FirstOrDefault(m => m.Id == id);
        }
        StudentSubject IStudentSubjectRepository.Update(StudentSubject
studentSubject)
        {
            var StudentSubject = context.StudentSubject.Attach(studentSubject);
            StudentSubject.State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
            context.SaveChanges();
            return studentSubject;
        }
        StudentSubject IStudentSubjectRepository.Add(StudentSubject newstusub)
        {
            context.StudentSubject.Add(newstusub);
            context.SaveChanges();
            return newstusub;
        }
    }
}

```

Листинг Г.14 – DAO/SQLSubjectRepository.cs

```

namespace StudentAttendanceManagementSystem.Models.DAO
{

```

```

public class SQLSubjectRepository : ISubjectRepository
{
    private readonly AppDbContext context;
    public SQLSubjectRepository(AppDbContext context)
    {
        this.context = context;
    }

    Subject ISubjectRepository.Add(Subject newSubject)
    {
        context.Subject.Add(newSubject);
        context.SaveChanges();
        return newSubject;
    }

    Subject ISubjectRepository.GetById(int id)
    {
        return context.Subject.FirstOrDefault(m => m.Id == id);
    }

    Subject ISubjectRepository.GetBySubjectCode(string code)
    {
        return context.Subject.FirstOrDefault(m => m.SubjectCode == code);
    }

    IEnumerable<Subject> ISubjectRepository.GetAllByProgramId(int ProgramId)
    {
        return context.Subject.Where(m => m.ProgramId == ProgramId).OrderBy(m =>
m.SemesterNo);
    }

    IEnumerable<Subject> ISubjectRepository.GetAllBySemesterProgramId(int
SemesterNo, int ProgramId)
    {
        return context.Subject.Where(m => (m.SemesterNo==SemesterNo &&
m.ProgramId == ProgramId));
    }
}

```

Листинг Г.15 – Admin.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Admin
    {
        public int Id { get; }

        [Required]
        [StringLength(50)]
        public string Name { get; set; }

        public int LoginId { get; set; }
        public Login Login { get; set; }
    }
}

```

Листинг Г.16 – Faculty.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Faculty
    {
        public int Id { get; }

        [Required]
        public string FirstName { get; set; }
    }
}

```

```

        [Required]
        public string MiddleName { get; set; }

        [Required]
        public string LastName { get; set; }
        [Required]
        public string Designation { get; set; }
        [Required]
        public string Email { get; set; }

        public string Password { get; set; }

        public int LoginId { get; set; }
        public Login Login { get; set; }

        public int ProgramId { get; set; }
        public Program Program { get; set; }

        public int SubjectId { get; set; }
        public Subject Subject { get; set; }
    }
}

```

Листинг Г.17 – Login.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Login
    {
        public int Id { get; }
        [Required(ErrorMessage = «Please enter username»), StringLength(50,
MinimumLength = 2, ErrorMessage = «Name must be 2 to 50 cahracter long»)]
        public string Username { get; set; }

        [Required(ErrorMessage = «Please enter password»), StringLength(50)]
        public string Password { get; set; }

        [NotMapped]
        public bool remember { get; set; }
        public Admin admin { get; set; }
        public Student Student { get; set; }
        public Faculty Faculty { get; set; }
    }
}

```

Листинг Г.18 – Program.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Program
    {
        public int Id { get; }

        [Required]
        public string ProgramName { get; set; }

        [Required]
        [RegularExpression(@"^\d{2}\.\d{2}\.\d{2}$», ErrorMessage = «Код введен
неверно»)]
        public string ProgramCode { get; set; }

        public ICollection<Subject> Subjects { get; set; }

        public ICollection<Student> Students { get; set; }
    }
}

```

Листинг Г.19 – Student.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Student

```

```

{
    public int Id { get; }

    [Required]
    public string FirstName { get; set; }

    [Required]
    public string MiddleName { get; set; }

    [Required]
    public string LastName { get; set; }

    [Required]
    public string UniversityId { get; set; }

    public string ActualDOB { get; set; }
    public DateTime DOB { get; set; }

    public string Password { get; set; }

    public int ProgramId { get; set; }
    public Program Program { get; set; }

    public int LoginId { get; set; }
    public Login Login { get; set; }

    public ICollection<StudentSubject> StudentSubjects { get; set; }
}

```

Листинг Г.20 – StudentSubject.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class StudentSubject
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public int Attendance { get; set; }

        [Required]
        public int SemesterNo { get; set; }

        public int StudentId { get; set; }
        public Student Student { get; set; }

        public int SubjectId { get; set; }
        public Subject Subject { get; set; }
    }
}

```

Листинг Г.21 – Subject.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class Subject
    {
        public int Id { get; }

        [Required]
        public string SubjectName { get; set; }

        [Required]
        public string SubjectCode { get; set; }
        [Required]
        public string SubjectDescription { get; set; }

        [Required]

```

```

        public int SemesterNo { get; set; }

        public int ProgramId { get; set; }
        public Program Program { get; set; }

        public Faculty Faculty { get; set; }

        public ICollection<StudentSubject> StudentSubjects { get; set; }
    }
}

```

Листинг Г.22 – AppDbContext.cs

```

namespace StudentAttendanceManagementSystem.Models
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }

        public DbSet<Program> Program { get; set; }
        public DbSet<Subject> Subject { get; set; }
        public DbSet<Student> Student { get; set; }
        public DbSet<Login> Login { get; set; }
        public DbSet<Admin> Admin { get; set; }
        public DbSet<Faculty> Faculty { get; set; }
        public DbSet<StudentSubject> StudentSubject { get; set; }
        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<Subject>()
                .HasOne(s => s.Faculty)
                .WithOne(f => f.Subject)
                .HasForeignKey<Faculty>(f => f.SubjectId)
                .OnDelete(DeleteBehavior.Restrict);
            builder.Entity<StudentSubject>()
                .HasOne(ss => ss.Student)
                .WithMany(st => st.StudentSubjects)
                .HasForeignKey(ss => ss.StudentId)
                .OnDelete(DeleteBehavior.Restrict);
            builder.Entity<StudentSubject>()
                .HasOne(ss => ss.Subject)
                .WithMany(sub => sub.StudentSubjects)
                .HasForeignKey(ss => ss.SubjectId)
                .OnDelete(DeleteBehavior.Restrict);

            builder.Entity<Admin>().HasKey(x => x.Id);
            builder.Entity<Program>().HasKey(x => x.Id);
            builder.Entity<Faculty>().HasKey(x => x.Id);
            builder.Entity<Login>().HasKey(x => x.Id);
            builder.Entity<Student>().HasKey(x => x.Id);
            builder.Entity<Subject>().HasKey(x => x.Id);
            base.OnModelCreating(builder);
        }
    }
}

```

Приложение Д
(рекомендуемое)
Исходный код представлений

Листинг Д.1 – AddNewFaculty.cshtml

```
@model Faculty
@{
    ViewData[«Title»] = «Добавить группу»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Program currentProgram = ViewData[«Program»] as Program;
    List<Subject> subjects = ViewData[«Subjects»] as List<Subject>;
}
<nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-color:
#e3f2fd;»>
    <a class=«navbar-brand» asp-action=«Admin» asp-controller=«Admin»>ПГУ</a>
    <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
        <span class=«navbar-toggler-icon»></span>
    </button>
    <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
        <div class=«navbar-nav»>
            <a class=«nav-item nav-link active» asp-action=«Admin» asp-
controller=«Admin»>Главная<span class=«sr-only»>(current)</span></a>
            <a class=«nav-item nav-link» asp-action=«ProgramDetails» asp-
controller=«Admin» asp-route-id=«@currentProgram.Id»>@currentProgram.ProgramName</a>
        </div>
    </div>
    <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
</nav>

<section class=«vh-100»>
    <div class=«container py-5 h-100»>
        <div class=«row d-flex justify-content-center align-items-center h-100»>
            <div class=«col-12 col-md-8 col-lg-6 col-xl-5»>
                <div class=«card shadow-2-strong» style=«border-radius: 1rem;»>
                    <div class=«card-body p-5»>
                        <form asp-action=«AddFaculty» asp-controller=«Admin»
method=«post»>
                            <h3 class=«mb-5 text-center»>Добавить группу</h3>
                            @if (null != ErrorMessage)
                            {
                                <div class=«alert alert-warning alert-dismissible
fade show» role=«alert»>
                                    <strong>
                                        @if (null != ErrorMessage)
                                        {
                                            @ErrorMessage
                                        }
                                    </strong>
                                    <button type=«button» class=«close» data-
dismiss=«alert» aria-label=«Close»>
                                        <span aria-hidden=«true»>&times;</span>
                                </button>
                                </div>
                            }
                            @if (null != SuccessMessage)
                            {
                                <div class=«alert alert-success alert-dismissible
fade show» role=«alert»>
                                    <strong>
                                        @if (null != SuccessMessage)
                                        {
                                            @SuccessMessage
                                        }
                                    </strong>
                                </div>
                            }
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```

```

                                <button type=«button» class=«close» data-
dismiss=«alert» aria-label=«Close»>
                                <span aria-hidden=«true»>&times;</span>
                                </button>
                                </div>
                                }
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«FirstName» class=«form-control form-
control-lg» placeholder=«Enter First Name» autofocus=«autofocus» required />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«MiddleName» class=«form-control
form-control-lg» placeholder=«Enter Middle Name» required />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«LastName» class=«form-control form-
control-lg» placeholder=«Enter Last Name» required />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«Designation» class=«form-control
form-control-lg» placeholder=«Enter Designation» required />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«Email» class=«form-control form-
control-lg» placeholder=«Enter Email» required />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <select asp-for=«SubjectId» class=«form-control»
required>
                                <option value=-1 selected hidden>Выбрать
предметы</option>
                                @foreach (Subject sub in subjects)
                                {
                                    <option
value=«@sub.Id»>@sub.SubjectName</option>
                                }
                                </select>
                                </div>

                                <input asp-for=«ProgramId» value=«@currentProgram.Id»
hidden />

                                <button class=«btn btn-primary btn-lg btn-block»
type=«submit»>>Добавить</button>
                                </form>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </section>

```

Листинг Д.2 – AddNewStudent.cshtml

```

@model Student
@{
    ViewData[«Title»] = «Добавить студента»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Program currentProgram = ViewData[«Program»] as Program;
}

<nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-color:
#e3f2fd;»>
    <a class=«navbar-brand» asp-action=«Admin» asp-controller=«Admin»>ПТУ</a>

```

```

<button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
    <span class=«navbar-toggler-icon»></span>
</button>
<div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
    <div class=«navbar-nav»>
        <a class=«nav-item nav-link active» asp-action=«Admin» asp-
controller=«Admin»>Главная<span class=«sr-only»>(current)</span></a>
        <a class=«nav-item nav-link» asp-action=«ProgramDetails» asp-
controller=«Admin» asp-route-id=«@currentProgram.Id»>@currentProgram.ProgramName</a>
    </div>
    <div>
        <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
    </div>
</nav>
<section class=«vh-100»>
    <div class=«container py-5 h-100»>
        <div class=«row d-flex justify-content-center align-items-center h-100»>
            <div class=«col-12 col-md-8 col-lg-6 col-xl-5»>
                <div class=«card shadow-2-strong» style=«border-radius: 1rem;»>
                    <div class=«card-body p-5»>
                        <form asp-action=«AddStudent» asp-controller=«Admin»
method=«post»>
                            <h3 class=«mb-5 text-center»>Добавить нового
студента</h3>
                            @if (null != ErrorMessage)
                            {
                                <div class=«alert alert-warning alert-dismissible
fade show» role=«alert»>
                                    <strong>
                                        @if (null != ErrorMessage)
                                        {
                                            @ErrorMessage
                                        }
                                    </strong>
                                    <button type=«button» class=«close» data-
dismiss=«alert» aria-label=«Close»>
                                        <span aria-hidden=«true»>&times;</span>
                                    </button>
                                </div>
                                }
                            @if (null != SuccessMessage)
                            {
                                <div class=«alert alert-success alert-dismissible
fade show» role=«alert»>
                                    <strong>
                                        @if (null != SuccessMessage)
                                        {
                                            @SuccessMessage
                                        }
                                    </strong>
                                    <button type=«button» class=«close» data-
dismiss=«alert» aria-label=«Close»>
                                        <span aria-hidden=«true»>&times;</span>
                                    </button>
                                </div>
                                }
                            <div class=«form-outline mb-4»>
                                <input asp-for=«FirstName» class=«form-control form-
control-lg» placeholder=«Имя» autofocus=«autofocus» required/>
                            </div>
                            <div class=«form-outline mb-4»>
                                <input asp-for=«MiddleName» class=«form-control
form-control-lg» placeholder=«Отчество» required />
                            </div>
                            <div class=«form-outline mb-4»>

```

```

        <input asp-for=«LastName» class=«form-control form-
control-lg» placeholder=«Фамилия» required />
    </div>
    <div class=«form-outline mb-4»>
        <input asp-for=«UniversityId» class=«form-control
form-control-lg» placeholder=«Номер зачетной книжки» />
    </div>

    <div class=«form-outline mb-4»>
        <input type=«date» max=«2050-12-31» asp-for=«DOB»
class=«form-control form-control-lg» placeholder=«Дата рождения» required />
    </div>

    <input asp-for=«ProgramId» value=«@currentProgram.Id»
hidden />

    <button class=«btn btn-primary btn-lg btn-block»
type=«submit»>Сохранить</button>
</form>
</div>
</div>
</div>
</div>
</div>
</section>

```

Листинг Д.3 – Admin.cshtml

```

@{
    ViewData[«Title»] = «Администратор»;
    Layout = «~/Views/Shared/Admin_Layout.cshtml»;

    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    List<Program> programList = ViewData[«programList»] as List<Program>;
}

<div>
    @if (null != ErrorMessage)
    {
        <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != ErrorMessage)
                {
                    @ErrorMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>
            </button>
        </div>
    }
    @if (null != SuccessMessage)
    {
        <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != SuccessMessage)
                {
                    @SuccessMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>
            </button>
        </div>
    }
}

```

```

    }
</div>
<br />
<div class=«container»>
    <h1 class=«text-center mt-4 mb-4»>Все программы</h1>
    <div class=«row d-flex justify-content-start mt-4»>
        @foreach (Program p in programList)
        {
            <div class=«col-lg-6 col-md-4 col-sm-12 col-s-12 d-flex justify-content-
center py-3»>
                <a asp-action=«ProgramDetails» asp-controller=«Admin» asp-route-
Id=«@p.Id» class=«col-12 btn btn-secondary py-3»><h3>@p.ProgramName&nbsp;</h3></a>
            </div>
        }
    </div>
</div>

```

Листинг Д.4 – ProgramDetails.cshtml

@model Subject

```

@{
    ViewData[«Title»] = «Программы»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Program currentProgram = ViewData[«Program»] as Program;
    List<Subject> subjects = ViewData[«Subjects»] as List<Subject>;
    List<Faculty> faculties = ViewData[«Faculties»] as List<Faculty>;
}

<nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-color:
#e3f2fd;»>
    <a class=«navbar-brand» asp-action=«Admin» asp-controller=«Admin»>ПГУ</a>
    <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
        <span class=«navbar-toggler-icon»></span>
    </button>
    <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
        <div class=«navbar-nav»>
            <a class=«nav-item nav-link active» asp-action=«Admin» asp-
controller=«Admin»>Главная<span class=«sr-only»>(current)</span></a>
            <a class=«nav-item nav-link» data-toggle=«modal» data-
target=«#addSubjectModal» href=«#»>Добавить предмет</a>

            <div class=«modal fade» id=«addSubjectModal» tabindex=«-1» role=«dialog»
aria-labelledby=«exampleModalCenterTitle» aria-hidden=«true»>
                <div class=«modal-dialog» role=«document»>
                    <div class=«modal-content»>
                        <div class=«modal-header»>
                            <h5 class=«modal-title»
id=«exampleModalLongTitle»>Добавить новый предмет</h5>
                            <button type=«button» class=«close» data-dismiss=«modal»
aria-label=«Close»>
                                <span aria-hidden=«true»>&times;</span>
                            </button>
                        </div>
                        <form asp-action=«AddNewSubject» asp-controller=«Admin»
method=«post»>
                            <div class=«modal-body»>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«SubjectName» class=«form-control
form-control-lg» placeholder=«Название предмета» autofocus=«autofocus» required />
                                </div>
                                <div class=«form-outline mb-3»>
                                    <input asp-for=«SubjectCode» class=«form-control
form-control-lg» placeholder=«Код предмета» required />
                                </div>
                                <div class=«form-outline mb-3»>

```

```

        <input asp-for=«SubjectDescription» class=«form-
control form-control-lg» placeholder=«Описание предмета» required />
    </div>
    <div class=«form-outline mb-3»>
        <input asp-for=«SemesterNo» class=«form-control
form-control-lg» placeholder=«Семестр» required />
    </div>
    <input asp-for=«ProgramId»
value=«@currentProgram.Id» hidden />

    </div>
    <div class=«modal-footer»>
        <button type=«button» class=«btn btn-secondary»
data-dismiss=«modal»>Закрыть</button>
        <button type=«submit» class=«btn btn-
primary»>Добавить</button>
    </div>
</form>
</div>
</div>
<div>
    <a class=«nav-item nav-link» asp-action=«AddNewStudent» asp-
controller=«Admin» asp-route-Id=«@currentProgram.Id»>Добавить студента</a>
    <a class=«nav-item nav-link» asp-action=«AddNewGroup» asp-
controller=«Admin» asp-route-Id=«@currentProgram.Id»>Добавить группу</a>
</div>
</div>
    <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
</nav>

<div>
    @if (null != ErrorMessage)
    {
        <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != ErrorMessage)
                {
                    @ErrorMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>
            </button>
        </div>
    }
    @if (null != SuccessMessage)
    {
        <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != SuccessMessage)
                {
                    @SuccessMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>
            </button>
        </div>
    }
</div>

<div class=«ml-2 mr-2»>
    <h1 class=«text-center mt-2»>@currentProgram.ProgramName</h1>
</div>

```

```

<div class=«container mt-4»>

    <section class=«mt-2»>
        <h3 class=«text-info»>Все дисциплины</h3>
        <div class=«rounded»>

            @{
                <div class=«row mb-5 ml-1 mr-1»>
                    @for (int i = 0; i < subjects.Count(); i++)
                    {

                        <div class=«col-lg-4 col-md-6 col-sm-6 col-s-12 mb-4 card
rounded»>
                            <div class=«card-body»>
                                <h5 class=«card-title»>@subjects[i].SubjectName
(@subjects[i].SubjectCode)</h5>
                                <h6 class=«card-subtitle mb-2 text-muted»>Семестр :
@subjects[i].SemesterNo</h6>
                                <p class=«card-
text»>@subjects[i].SubjectDescription</p>
                                <a href=«#» class=«btn btn-secondary card-
link»>Редактировать</a>
                            </div>
                        </div>
                    }
                </div>
            }

        </div>
    </section>
    <hr />
    <section class=«mt-2»>
        <h3 class=«text-info»>Все группы</h3>
        <div class=«rounded»>

            @{
                <div class=«row mb-5 ml-1 mr-1»>
                    @for (int i = 0; i < faculties.Count(); i++)
                    {
                        <div class=«col-lg-4 col-md-6 col-sm-6 col-s-12 card mb-4»>
                            <div class=«card-body text-center»>
                                // <img src=«https://mdbcdn.b-
cdn.net/img/Photos/new-templates/bootstrap-chat/ava3.webp alt=«avatar»
                                // class=«rounded-circle img-fluid» style=«width:
150px;»>
                                <h5 class=«my-3»>@faculties[i].FirstName
@faculties[i].LastName</h5>
                                <p class=«text-muted mb-1»>@faculties[i].Email</p>
                                <p class=«text-muted mb-
4»>@faculties[i].Designation</p>
                                <p class=«text-muted mb-1»>Subject :
@faculties[i].Subject.SubjectName</p>
                                <p class=«text-muted mb-4»>Semester :
@faculties[i].Subject.SemesterNo</p>
                                <div class=«d-flex justify-content-center mb-2»>
                                    <button type=«button» class=«btn btn-
primary»>Редактировать</button>
                                </div>
                            </div>
                        </div>
                    }
                </div>
            }

        </div>
    </section>

```

```

    }
  </div>
</section>

</div>

```

Листинг Д.5 – Faculty.cshtml

```

@{
    ViewData[«Title»] = «Преподаватель»;
    Layout = «~/Views/Shared/_Layout.cshtml»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;
    List<StudentSubject> studentlist = ViewData[«StudentSubjectList»] as
List<StudentSubject>;
    Subject CurrentSubject = ViewData[«Subject»] as Subject;
    Faculty CurrentFaculty = ViewData[«Faculty»] as Faculty;
}
<div>
    <nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-
color: #e3f2fd;»>
        <a class=«navbar-brand» asp-action=«/» asp-controller=«Home»>ПГУ</a>
        <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
            <span class=«navbar-toggler-icon»></span>
        </button>
        <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
            <div class=«navbar-nav»>
                <a class=«nav-item nav-link active» asp-action=«Faculty» asp-
controller=«Faculty»>Главная</a>
                <a class=«nav-item nav-link» href=«#»>Предметы</a>
                <a class=«nav-item nav-link» href=«#»>Личный кабинет</a>
            </div>
        </div>
        <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
    </nav>
</div>
@* Error messages *@
<div>
    @if (null != ErrorMessage)
    {
        <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != ErrorMessage)
                {
                    @ErrorMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>
            </button>
        </div>
    }
    @if (null != SuccessMessage)
    {
        <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
            <strong>
                @if (null != SuccessMessage)
                {
                    @SuccessMessage
                }
            </strong>
            <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                <span aria-hidden=«true»>&times;</span>

```

```

        </button>
    </div>
}
</div>
<div class=«container»>
    <h1 class=«text-center my-4»>@CurrentSubject.SubjectName
    (@CurrentSubject.SubjectCode)</h1>
    <table class=«table table-hover mt-3»>
        <thead>
            <tr>
                <th scope=«col»>#</th>
                <th scope=«col»>Код студента</th>
                <th scope=«col»>Посещаемость</th>
                <th scope=«col»><span class=«glyphicon»>&#x270f;</span>Edit</th>
            </tr>
        </thead>
        <tbody>
            @for(int i=0;i<studentlist.Count;i++){
                <tr>
                    <th scope=«row»>@(i+1)</th>
                    <td>@studentlist[i].Student.UniversityId</td>
                    <td>@studentlist[i].Attendance</td>
                    <td><a asp-action=«UpdateAttendance» asp-controller=«Faculty»
asp-route-objId=«@studentlist[i].Id» class=«btn btn-primary»>Обновить</a></td>
                </tr>
            }
        </tbody>
    </table>
</div>

```

Листинг Д.6 – UpdateAttendance.cshtml

```

@model StudentSubject
@{
    ViewData[«Title»] = «Преподаватель»;
    Layout = «~/Views/Shared/_Layout.cshtml»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Student stu = ViewData[«StudentObj»] as Student;
    StudentSubject stusub = ViewData[«StuSubObj»] as StudentSubject;
}

<div>
    <nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-
color: #e3f2fd;»>
        <a class=«navbar-brand» asp-action=«Faculty» asp-
controller=«Faculty»>ПГТУ (@stu.UniversityId)</a>
        <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
            <span class=«navbar-toggler-icon»></span>
        </button>
        <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
            <div class=«navbar-nav»>
                <a class=«nav-item nav-link active» asp-action=«Faculty» asp-
controller=«Faculty»>Главная</a>
            </div>
            <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
        </div>
    </nav>
</div>

@* Error messages *@
<div>
    @if (null != ErrorMessage)
    {
        <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>

```

```

        <strong>
            @if (null != ErrorMessage)
            {
                @ErrorMessage
            }
        </strong>
        <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
            <span aria-hidden=«true»>&times;</span>
        </button>
    </div>
}
@if (null != SuccessMessage)
{
    <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
        <strong>
            @if (null != SuccessMessage)
            {
                @SuccessMessage
            }
        </strong>
        <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
            <span aria-hidden=«true»>&times;</span>
        </button>
    </div>
}
</div>

<div class=«container»>
    <section class=«vh-100»>
        <div class=«container py-5 h-100»>
            <div class=«row d-flex justify-content-center align-items-center h-100»>
                <div class=«col-12 col-md-8 col-lg-6 col-xl-5»>
                    <div class=«card shadow-2-strong» style=«border-radius: 1rem;»>
                        <div class=«card-body p-5»>
                            <form asp-action=«Update» asp-controller=«Faculty»
method=«post»>
                                <h3 class=«mb-5 text-center»>Добавить посещение для:
@stu.UniversityId</h3>
                                <div class=«form-outline mb-4»>
                                    <input class=«form-control form-control-lg»
value=«Код студента : @stu.UniversityId» readonly />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input class=«form-control form-control-lg»
value=«Текущая посещаемость : @stusub.Attendance» readonly />
                                </div>
                                <div class=«form-outline mb-4»>
                                    <input asp-for=«Attendance» class=«form-control
form-control-lg» placeholder=«Добавить посещение» autofocus required />
                                </div>
                                <input asp-for=«Id» value=«@stusub.Id» class=«form-
control form-control-lg» hidden />
                                <input asp-for=«SemesterNo»
value=«@stusub.SemesterNo» class=«form-control form-control-lg» hidden />
                                <input asp-for=«StudentId» value=«@stusub.StudentId»
class=«form-control form-control-lg» hidden />
                                <input asp-for=«SubjectId» value=«@stusub.SubjectId»
class=«form-control form-control-lg» hidden />
                                <button class=«btn btn-primary btn-lg btn-block»
type=«submit»>Добавить</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
</div>

```

```

        </div>
    </div>
</div>
</section>
</div>

```

Листинг Д.7 – Attendance.cshtml

```

@{
    ViewData[«Title»] = «Посещаемость»;
    Layout = «~/Views/Shared/_Layout.cshtml»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Student currentStudent = ViewData[«Student»] as Student;
    List<StudentSubject> attendanceList = ViewData[«SubjectWiseAttendance»] as
List<StudentSubject>;
}

<div>
    <nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-
color: #e3f2fd;»>
        <a class=«navbar-brand» asp-action=«Index» asp-
controller=«Home»>ПГУ (@currentStudent.UniversityId)</a>

        <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
            <span class=«navbar-toggler-icon»></span>
        </button>
        <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
            <div class=«navbar-nav»>
                <a class=«nav-item nav-link active» asp-action=«Index» asp-
controller=«Home»>Главная</a>
                <a class=«nav-item nav-link» href=«#»>Предметы</a>
            </div>
            <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
        </nav>
    </div>

    @* Error messages *@
    <div>
        @if (null != ErrorMessage)
        {
            <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>
                <strong>
                    @if (null != ErrorMessage)
                    {
                        @ErrorMessage
                    }
                </strong>
                <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
                    <span aria-hidden=«true»>&times;</span>
                </button>
            </div>
        }
        @if (null != SuccessMessage)
        {
            <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
                <strong>
                    @if (null != SuccessMessage)
                    {
                        @SuccessMessage
                    }
                </strong>
            </div>
        }
    </div>

```

```

        <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
            <span aria-hidden=«true»>&times;</span>
        </button>
    </div>
}
</div>

<div class=«container»>
    <h1 class=«text-center my-5»>Посещаемость за семестр</h1>

    <table class=«table table-striped mt-4»>
        <thead>
            <tr>
                <th scope=«col»>#</th>
                <th scope=«col»>Предмет</th>
                <th scope=«col»>Посещаемость</th>
            </tr>
        </thead>
        <tbody>
            @for(int i=0;i<attendanceList.Count;i++){
                <tr>
                    <th scope=«row»>@(i+1)</th>
                    <td>@attendanceList[i].Subject.SubjectName</td>
                    <td>@attendanceList[i].Attendance</td>
                </tr>
            }
        </tbody>
    </table>
</div>

```

Листинг Д.8 – Index.cshtml

```

@{
    ViewData[«Title»] = «Главная»;
    Layout = «~/Views/Shared/_Layout.cshtml»;
    string ErrorMessage = ViewData[«ErrorMessage»] as string;
    string SuccessMessage = ViewData[«SuccessMessage»] as string;

    Student currentStudent = ViewData[«StudentObj»] as Student;
    Program currentProgram = ViewData[«ProgramObj»] as Program;
}

<div>
    <nav class=«navbar navbar-expand-lg navbar-light bg-light» style=«background-
color: #e3f2fd;»>
        <a class=«navbar-brand» asp-action=«/» asp-
controller=«Home»>ИГУ (@currentStudent.UniversityId)</a>
        <button class=«navbar-toggler» type=«button» data-toggle=«collapse» data-
target=«#navbarNavAltMarkup» aria-controls=«navbarNavAltMarkup» aria-
expanded=«false» aria-label=«Toggle navigation»>
            <span class=«navbar-toggler-icon»></span>
        </button>
        <div class=«collapse navbar-collapse» id=«navbarNavAltMarkup»>
            <div class=«navbar-nav»>
                <a class=«nav-item nav-link active» asp-action=«/» asp-
controller=«Home»>Главная</a>
                <a class=«nav-item nav-link» href=«#»>Предметы</a>
            </div>
        </div>
        <a class=«btn btn-secondary nav-item mr-3» asp-action=«Logout» asp-
controller=«Home»>Выйти</a>
    </nav>
</div>

@* Error messages *@
<div>
    @if (null != ErrorMessage)
    {
        <div class=«alert alert-warning alert-dismissible fade show» role=«alert»>

```

```

        <strong>
            @if (null != ErrorMessage)
            {
                @ErrorMessage
            }
        </strong>
        <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
            <span aria-hidden=«true»>&times;</span>
        </button>
    </div>
}
@if (null != SuccessMessage)
{
    <div class=«alert alert-success alert-dismissible fade show» role=«alert»>
        <strong>
            @if (null != SuccessMessage)
            {
                @SuccessMessage
            }
        </strong>
        <button type=«button» class=«close» data-dismiss=«alert» aria-
label=«Close»>
            <span aria-hidden=«true»>&times;</span>
        </button>
    </div>
}
</div>

<div class=«container»>
    <h1 class=«text-center mt-4 mb-4»>@currentProgram.ProgramName</h1>

    <div class=«row d-flex justify-content-around mt-4»>
        @for (int i = 1; i <= 8; i++)
        {
            <div class=«col-lg-6 col-md-4 col-sm-12 col-s-12 d-flex justify-content-
center py-3»>
                <a asp-action=«Attendance» asp-controller=«Home» asp-route-
semester=«@i» asp-route-program=«@currentProgram.Id» class=«col-12 btn btn-info py-
3»><h2>Семестр-@i</h2></a>
            </div>
        }
    </div>
</div>

```

Листинг Д.9 – Login.cshtml

```

@model Login;
@{
    ViewData[«Title»] = «Вход»;
    Layout = «~/Views/Shared/_Layout.cshtml»;
}

<section class=«vh-100»>
    <div class=«container py-5 h-100»>
        <div class=«row d-flex justify-content-center align-items-center h-100»>
            <div class=«col-12 col-md-8 col-lg-6 col-xl-5»>
                <div class=«card shadow-2-strong» style=«border-radius: 1rem;»>
                    <div class=«card-body p-5»>
                        <form asp-action=«Login» method=«post»>
                            <h3 class=«mb-5 text-center»>Вход в систему</h3>
                            @if (null != ViewData[«ErrorMessage»])
                            {
                                <div class=«alert alert-warning alert-dismissible
fade show» role=«alert»>
                                    <strong>
                                        @ViewData[«ErrorMessage»]
                                    </strong>
                                </div>
                            }
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <button type=<button> class=<close> data-
dismiss=<alert> aria-label=<Close>>
            <span aria-hidden=<true>>&times;</span>
        </button>
    </div>
}
<div class=<form-outline mb-4>
    <input asp-for=<Username> class=<form-control form-
control-lg> placeholder=<Введите эл. почту> autofocus=<autofocus> />
    <span asp-validation-for=<Username> class=<text-
danger ml-1>></span>

</div>

<div class=<form-outline mb-4>
    <input type=<password> asp-for=<Password>
class=<form-control form-control-lg> placeholder=<Введите пароль> />
    <span asp-validation-for=<Password> class=<text-
danger ml-1>></span>

</div>

<!-- Checkbox -->
<div class=<form-check d-flex justify-content-start mb-
4>>
    <input class=<form-check-input> asp-for=<remember>
/>
    <label class=<form-check-label> asp-for=<remember>>
Запомнить меня</label>

</div>
<button class=<btn btn-primary btn-lg btn-block>
type=<submit>>Войти</button>

</form>
</div>
</div>
</div>
</div>
</div>
</section>

```

Приложение Е
(рекомендуемое)
Результаты тестирования

Вход в систему

Введите эл. почту

Введите пароль

☐ Запомнить меня

Войти

Рисунок Е.1 – Форма входа в систему

Вход в систему

Неверные данные x

Введите эл. почту

Введите пароль

☐ Запомнить меня

Войти

Рисунок Е.2 – Форма входа в систему

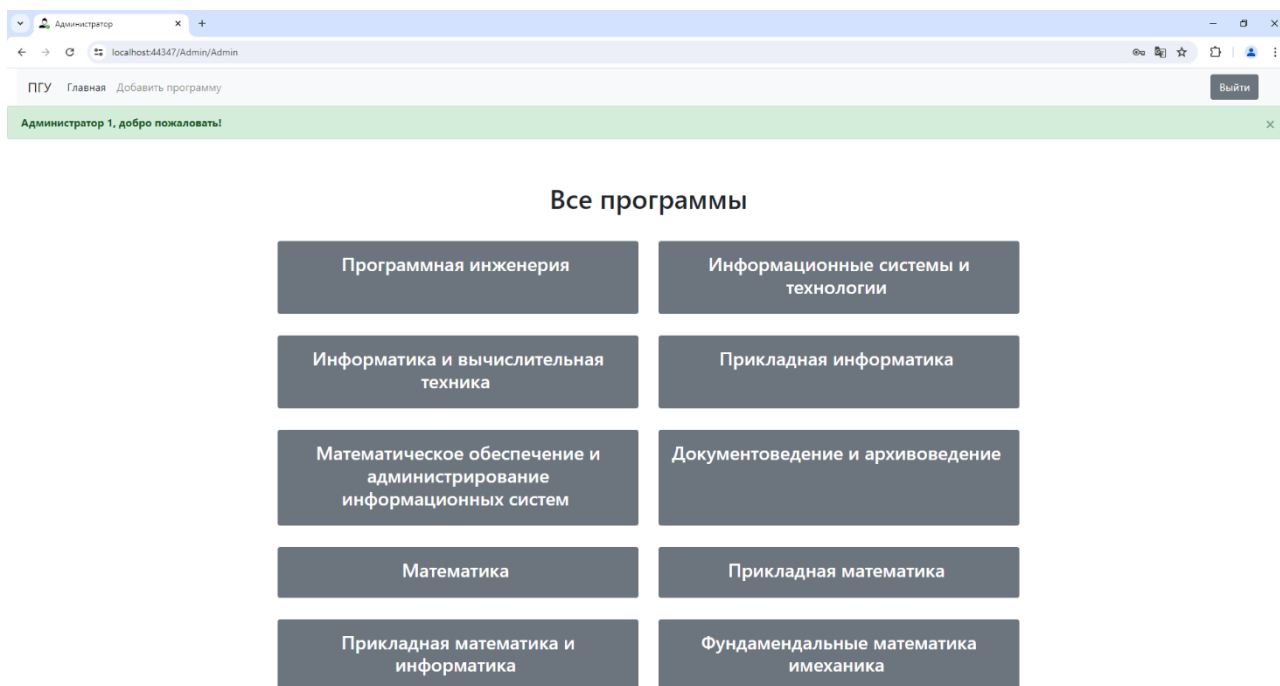


Рисунок Е.3 – Главная страница администратора

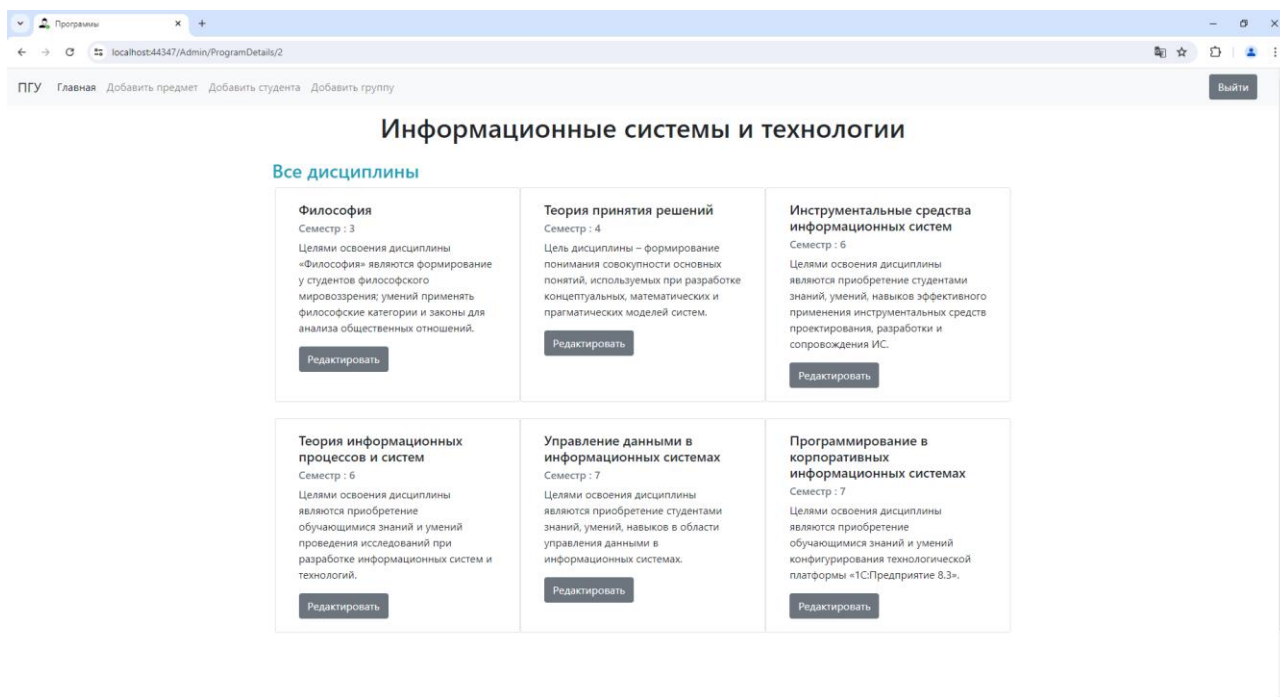


Рисунок Е.4 – Страница программы

Добавить нового студента

Иванов

Иван

Иванович

22ВП1

01/04/2004

Сохранить

Рисунок Е.5 – Форма добавления студента

Добавить нового студента

Студент уже добавлен

Имя

Отчество

Фамилия

Номер зачетной книжки

mm/dd/yyyy

Сохранить

Рисунок Е.6 – Форма добавления студента

Добавить нового студента

22ВР117 успешно добавлен

Имя

Отчество

Фамилия

Номер зачетной книжки

mm/dd/yyyy

Сохранить

Рисунок Е.7 – Форма добавления студента

Добавить группу

20ВР1

Добавить

Рисунок Е.8 – Добавление группы

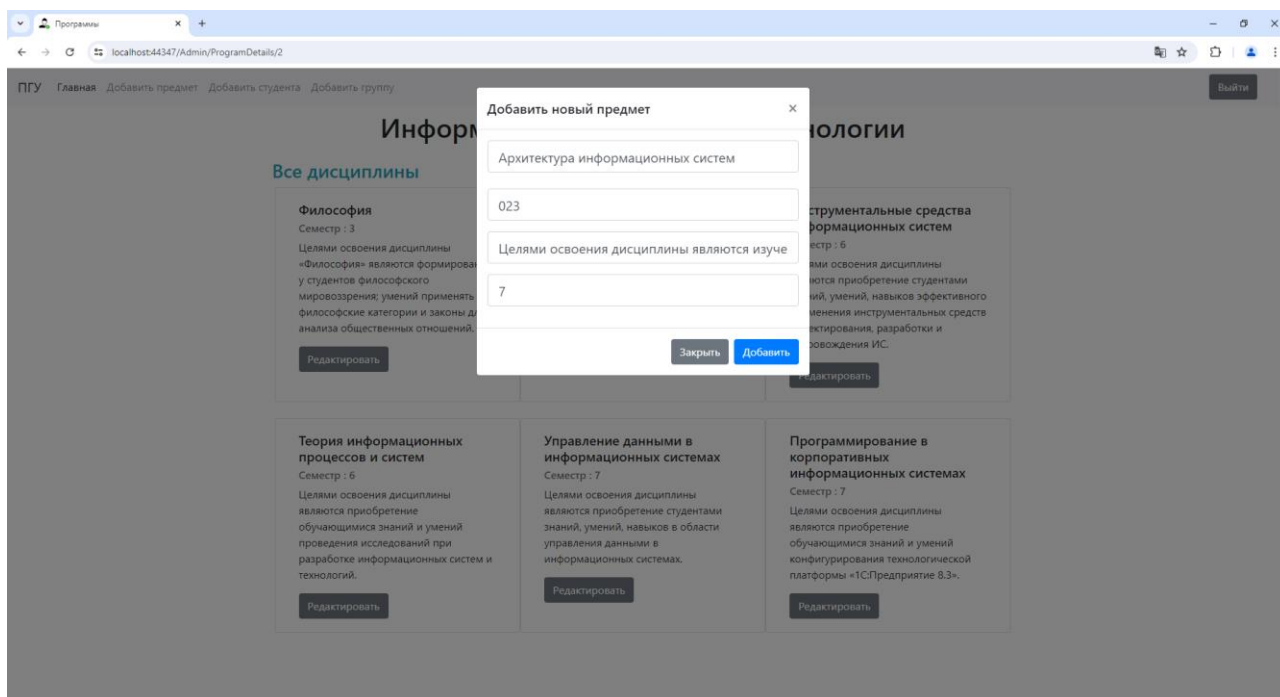


Рисунок Е.9 – Добавление дисциплины

Преподаватель

https://localhost:44347/Faculty/Faculty

ПГУ Главная Предметы Личный кабинет

Сергей, добро пожаловать!

Инструментальные средства информационных систем (001)

#	Код студента	Посещаемость	Edit
1	228П117	96	Обновить
2	228П116	100	Обновить
3	228П115	72	Обновить
4	228П114	85	Обновить
5	228П113	96	Обновить

Рисунок Е.10 – Страница отчета о посещаемости

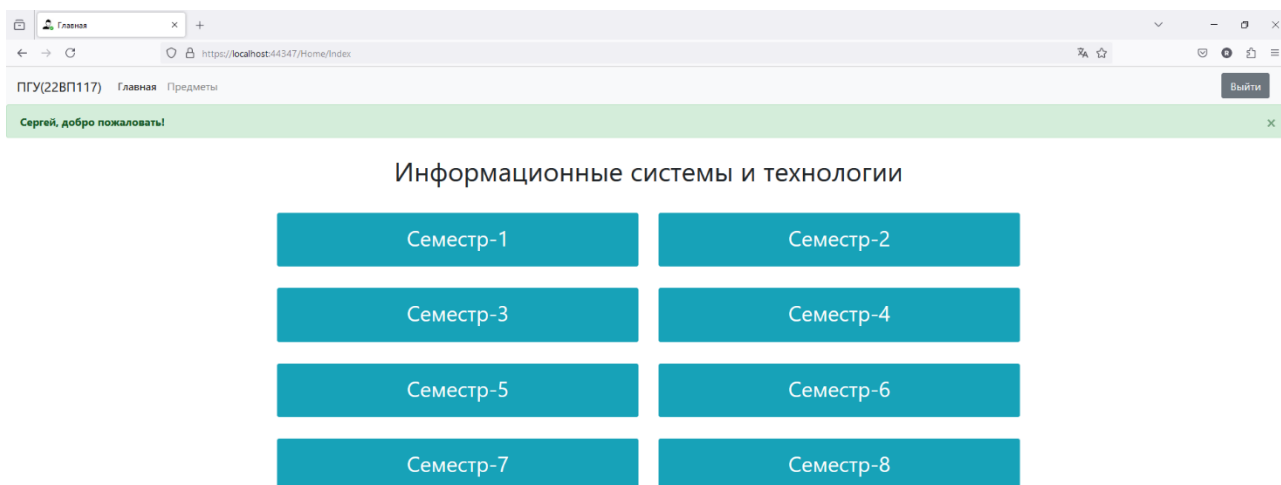


Рисунок Е.11 – Выбор семестра для отображения статистики

Посещаемость за семестр

#	Предмет	Посещаемость
1	Программирование в корпоративных информационных системах	85
2	Инструментальные средства информационных систем	96

Рисунок Е.12 – Отчет о посещаемости студента