МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

Гомельский государственный технический университет имени П.О. Сухого

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к дипломной работе

на тему: «Программный комплекс мониторинга текущей успеваемости студентов ГГТУ им. П. О. Сухого»

|  |  |  |  |
| --- | --- | --- | --- |
| Разработал студент гр. ИП-41 |  |  | Рогачёв Д. С. |
| (подпись) |  | (Ф.И.О.) |
| Руководитель работы |  |  | доцент, к.т.н., Трохова Т. А. |
| (подпись) |  | (ученое звание, ученая степень, Ф.И.О.) |
| Консультант по  экономической части |  |  | доцент, к.э.н., Соловьева Л. Л. |
| (подпись) |  | (ученое звание, ученая степень, Ф.И.О.) |
| Консультант по охране труда  и технике безопасности |  |  | профессор, д.т.н., Кудин В. П. |
| (подпись) | (ученое звание, ученая степень, Ф.И.О.) |
| Нормоконтроль |  |  | Романькова Т. Л. |
| (подпись) | (ученое звание, ученая степень, Ф.И.О.) |
| Рецензент |  |  |  |
| (подпись) | (ученое звание, ученая степень, должность,  организация, Ф.И.О.) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дипломная работа (\_\_\_\_\_\_\_с.) допущена к защите

в Государственной экзаменационной комиссии.

|  |  |  |  |
| --- | --- | --- | --- |
| Зав. кафедрой |  |  | доцент, к.т.н., Трохова Т. А. |
| (подпись) | (ученое звание, ученая степень, Ф.И.О.) |

Гомель 2023

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| Перечень условных обозначений и сокращений | | 7 |
| Введение | | 8 |
| 1 Аналитический обзор | | 9 |
|  | 1.1 Анализ предметной области | 9 |
|  | 1.2 Обзор существующих аналогов | 10 |
|  | 1.3 Обзор используемых технологий | 19 |
|  | 1.4 Постановка задачи | 21 |
| 2 Архитектура программного обеспечения | | 23 |
|  | 2.1 Пользовательские роли и функции | 23 |
|  | 2.2 Архитектура приложения | 25 |
|  | 2.3 Модель данных и структура базы данных | 26 |
| 3 Структура и основные алгоритмы созданного программного обеспечения. | | 39 |
|  | 3.1 Описание слоя доступа к данным | 39 |
|  | 3.2 Описание слоя бизнес-логики | 41 |
|  | 3.3 Описание слоя пользовательского интерфейса | 45 |
| 4 Тестирование и верификация | | 51 |
|  | 4.1 Верификация функционала приложения | 51 |
|  | 4.2 Тестирование приложения | 63 |
| Список использованных источников | | -- |

Тут будет продолжение содержания

**Перечень условных обозначений и сокращений**

В настоящей пояснительной записке применяются следующие термины, обозначения и сокращения.

*API* (*Application Programming Interface*)– программный интерфейс приложения, который представляет собой унифицированное описание способов взаимодействия одной компьютерной системы с другой.

*ASP.NET* (*Active Server Pages* для .*NET*) – платформа разработки веб-приложений от компании Майкрософт, в состав которой входят: веб-сервисы, программная инфраструктура, модель программирования.

*CIL* (*Common Intermediate Language*) или *IL* (*Intermediate Language*) – язык виртуальной машины платформы *.NET*.

*CLI* (*Common Language Infrastructure*) – спецификация общеязыковой инфраструктуры, которая определяет архитектуру исполнительной системы *.NET*.

*CLR* (*Common Language Runtime*) – общеязыковая среда выполнения, которая исполняет код *CIL*.

*DLL* (*Dynamic Link Library*) – динамически подключаема библиотека, позволяющая её многократное использование различными программными приложениями.

*DTO* (*Data Transfer Object*) – шаблон проектирования, который используется для передачи данных между подсистемами приложения.

*HTML* (*HyperText* *Markup* *Language*) – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

*IDE* (*Integrated Development Environment*) – комплекс программных средств, используемый программистами для разработки программного обеспечения.

*Mock*-тестирование – вид тестирования программного обеспечения, в ходе которого некоторые объекты окружения подменяются на объекты-имитации, что позволяет тестировать логику в отрыве от них в лабораторных условиях.

*MVC* (*Model*-*View*-*Controller*) – шаблон проектирования, предполагающий разделение данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер.

*ORM* (*Object-Relational Mapping*) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

*TDD* (*Test-Driven Development*) – разработка через тестирование – техника разработки программного обеспечения, при которой сначала пишутся тесты, покрывающие изменения, а потом уже код, который позволяет пройти тест.

*Unit*-тестирование – вид тестирования программного обеспечения, в ходе которого тестируются отдельные модули исходного кода программы.

*URL* (*Uniform Resource Locator*) – унифицированный указатель или адрес электронного ресурса.

**ВВЕДЕНИЕ**

Образовательный процесс является неотъемлемой и потому крайне важной частью человеческой жизни. Именно поэтому необходимо стараться обеспечить максимальную продуктивность и эффективность данного процесса. С увеличением этих характеристик, будет увеличиваться качество образования и скорость освоения новых знаний человеком. Улучшать образовательный процесс можно с разных сторон и различными способами. Одним из вариантов является полный или частичный перевод данного процесса в цифровую плоскость. Результатом этого перевода является ускорение всех рабочих процессов, сопровождающих образовательный процесс, как следствие того, что огромные объемы данных больше не приходится обрабатывать и обсчитывать вручную. Кроме этого, повышается удобство обучения студентов как за счёт различных цифровых ресурсов, которые могут облегчать донесение информации и её изучение, так и за счёт автоматизации рабочего места преподавателя, которое предполагает более удобную систематизацию данных, получаемых в ходе проведения занятий.

Данный проект ставит перед собой главную цель – автоматизировать, где это возможно, процесс доставки сведений об успеваемости и пропусках студентов, сгруппированных в виде результатов аттестаций до заведующих кафедрами и методистов деканатов, а также упростить процесс сбора и группировки этих сведений для преподавателей путём автовычисления некоторых промежуточных результатов и предоставления возможностей для программного учёта лабораторных работ и пропусков. Помимо этого, необходимо обеспечить студентам доступ к их текущей успеваемости, то есть прогрессу выполнения лабораторных работ и количеству пропусков по заданному предмету, а также к результатам текущей и всех прочих аттестаций.

Для выполнения поставленных целей необходимо решить ряд как аналитических так и практических задач, среди которых можно выделить следующие: анализ предметной области и обзор проектов, предоставляющих аналогичный или схожий функционал, для выявления функциональных особенностей проектируемого приложения, которые, в сущности, являются его конкурентными преимуществами, проектирование модели данных, отражающей суть и потребности предметной области, разработка самого программного продукта с учётом поставленных требований как к функционалу, так и к визуальному оформлению.

**1 АНАЛИТИЧЕСКИЙ ОБЗОР**

**1.1 Анализ предметной области**

В Гомельском государственном техническом университете имени П. О. Сухого основной формой организации образовательного процесса при реализации образовательных программ высшего образования является учебное занятие: лекция, семинарское, лабораторное, практическое занятие, консультация и иное занятие.

Образовательный процесс при реализации образовательных программ высшего образования осуществляется в учебных группах или индивидуально [1].

Как видно из вышеуказанных сведений об организации образовательного процесса, существуют различные виды занятий. В данном проекте особое внимание уделено лабораторным занятиям.

Лабораторным занятием называется такое занятие, на котором учащиеся самостоятельно выполняют различные лабораторные работы с целью углубления и закрепления теоретических знаний. На каждую лабораторную работу отводится определённое количество учебного времени. Учёт процесса выполнения и сдачи данных работ и является главным функционалом необходимым для реализации в приложении. Процесс выполнения и сдачи лабораторных работ может быть разделён на четыре стадии:

* лабораторная работа выполнена;
* отчёт подготовлен и сдан;
* отчёт проверен преподавателем и подписан к защите;
* защита отчёта выполнена.

Помимо процесса выполнения лабораторных работ, важной величиной для учёта является количество пропусков учащимися занятий, связанных с лабораторными работами.

Для проведения лабораторных занятий все учащиеся разбиваются на группы, большие группы также подвергаются разделению на более мелкие единицы – подгруппы. К определённому набору групп, связанных с определённой специальностью или с некоторым её ответвлением, относится множество дисциплин, изучаемых в данных группах в ходе различных типов занятий.

Самой крупной временной единицей обучения является курс, который, по сути, является учебным годом. Курс состоит из двух семестров. Изучаемые студентами дисциплины могут изменяться как каждый курс, так и каждый семестр, кроме этого, лабораторные занятия по одной и той же дисциплине могут вести разные преподаватели. В каждом семестре для получения промежуточных результатов учёбы и посещаемости проводятся две аттестации.

Во время аттестации преподавателю необходимо высчитать все оценки и усреднить их некоторым образом, а также получить общее количество пропусков для каждого студента по каждой преподаваемой дисциплине, что является достаточно трудозатратой задачей, если производить все вычисления вручную.

После вычисления преподавателем аттестационных данных, они попадают к заведующему кафедры и в деканат, у которых продолжают свое существование в виде отчётов, сгруппированных по различным параметрам.

**1.2 Обзор существующих аналогов**

***1.2.1*** Для того чтобы более детально определить минимально необходимый и при этом удобный для пользователей функционал, кроме анализа предметной области и определения основных стадий учебного процесса, необходимо также исследовать проекты аналоги, т.е. такие проекты, которые работают с той же или схожей предметной областью, что была задана в данной дипломной работе. После исследования аналогичных проектов необходимо определить тот набор функциональных особенностей, которые позже будут выдвинуты как требования к конечному продукту.

Поскольку целью данной дипломной работы является создание программного комплекса мониторинга текущей успеваемости студентов, то при поиске проектов аналогов учитывались их возможности как по учёту отдельных лабораторных работ, так и по созданию отчётов об успеваемости разного масштаба и детализации. В результате поиска были выделены три аналога: «1С: Университет ПРОФ», *Moodle* и «Галактика *ERP*. Управление учебным процессом».

***1.2.2*** «1С:Университет ПРОФ» представляет собой комплексное решение для автоматизации управленческой деятельности в образовательных организациях высшего образования.

Данное решение предоставляет множество модулей, позволяющих автоматизировать рабочие места сотрудников самых разных структурных подразделений вуза:

* приемная комиссия;
* деканаты;
* кафедры;
* учебно-методический отдел;
* бухгалтерия;
* студенческий отдел кадров;
* профсоюзный комитет.

«1С:Университет ПРОФ» позволяет автоматизировать: прием в вуз; планирование учебного процесса; расчет и распределение учебной нагрузки; управление контингентом студентов; учет успеваемости и посещаемости; воинский учет; формирование приказов, справок и отчетности; формирование документов об образовании и о квалификации; расчет стипендиального обеспечения; учет трудоустройства студентов и выпускников; формирование регламентированной отчетности; управление научно-исследовательской деятельностью и инновациями; управление аспирантурой и докторантурой; поддержку работы диссертационных советов; составление расписания занятий; управление дополнительным образованием; управление кампусом вуза [2].

Далее будут описаны основные функциональные возможности из имеющих непосредственное отношение к заданной предметной области модулей данного решения.

Модуль «Планирование учебного процесса» предоставляет следующие функциональные возможности:

* учёт базовых и рабочих учебных планов в соответствии с различными стандартами;
* поддержка уровневой системы (бакалавр, магистр и специалист);
* закрепление дисциплин учебного плана за кафедрами и подразделениями вуза;
* загрузка и выгрузка учебных планов в различных форматах;
* возможность массовой загрузки учебных планов различных форматов, настройка правил проведения загружаемых учебных планов;
* учёт профилей в учебных планах;
* учёт объемов нагрузки учебных планов в часах и зачётных единицах трудоёмкости;
* возможность проверки соответствия данных учебного плана заданным эталонным значениям и составление печатной формы протокола проверки;
* учёт закрепления дисциплин по выбору за обучающимися;
* возможность копирования данных из одного учебного плана в другой (как одиночное, так и массовое копирование);
* возможность автоматической замены реквизитов документа «Учебный план» в соответствии с заданными условиями;
* создание, хранение и обработка программ дисциплин;
* создание, хранение и обработка образовательных программ, формирование учебных планов на основании образовательных программ;
* создание учебно-методических комплексов для дисциплин;
* редактор учебных планов;
* создание и вывод на печать соответствующих печатных форм.

Модуль «Расчёт и распределение нагрузки» предоставляет следующие функциональные возможности:

* формирование правил расчета нагрузки;
* настройка формирования контингента (разделение по виду образования или подразделению, настройка параметров автоматического объединения и разделения контингента);
* настройка закрепления правил расчёта за нагрузкой в зависимости от различных условий;
* планирование и распределение основной и дополнительной нагрузки по подразделениям и преподавателям;
* хранение и обработка сведений о профессорско-преподавательском составе, анализ штатного состава подразделений;
* учёт планируемого и фактического контингента при расчете нагрузки;
* учёт закрепления дисциплин по выбору за обучающимися при расчете нагрузки;
* учёт квалификационных требований;
* возможность настройки автоматического объединения и разделения контингента;
* возможность объединения и разделения контингента вручную;
* установка и проверка норм нагрузки сотрудников;
* формирование индивидуальных планов работы преподавателей с учётом учебной и внеучебной нагрузки;
* формирование планов работы кафедры с учётом учебной и внеучебной нагрузки;
* формирование и работа с портфолио преподавателей;
* интеграция с программами кадрового учёта («1С: Зарплата и управление персоналом», «1С:Зарплата и кадры бюджетного учреждения»).

Модуль «Управление контингентом» предоставляет следующие функциональные возможности:

* хранение и обработка сведений о контингенте студентов вуза, в том числе учёт иностранных обучающихся;
* контроль движения контингента студентов;
* ведение зачётных книг студентов (предусмотрена возможность одновременного обучения по нескольким направлениям подготовки), возможность настройки формата правил создания новой зачётной книги (при зачислении в вуз, переводах);
* учёт успеваемости и посещаемости, создание соответствующих печатных форм, возможность массового формирования и вывода на печать ведомостей успеваемости;
* возможность закрепления конкретных дисциплин по выбору за студентом;
* создание, редактирование и хранение портфолио обучающихся;
* управление практиками;
* формирование, вывод на печать и учёт документов об образовании и (или) о квалификации и приложений к ним, ведение книги регистрации бланков дипломов и приложений;
* ведение сведений о воинском учете физических лиц;
* управление деятельностью студпрофкома;
* управление трудоустройством выпускников, в том числе ведение банка вакансий и учет предпочтений выпускников;
* создание и вывод на печать статистических, аналитических и списочных отчетных форм.

Модуль «Расписание учебных занятий» предоставляет следующие функциональные возможности:

* механизм составления расписания в режиме конструктора;
* возможность составления расписания на конкретный период времени с последующим тиражированием;
* возможность добавления в расписание занятий не из распределений учебных поручений;
* возможность составления проекта расписания и фактического расписания;
* подсказки пользователю и визуализация процесса составления расписания;
* контроль коллизий при составлении расписания;
* возможность учета предпочтений и ограничений проведения занятий (для дисциплин, аудиторий, преподавателей, учебных групп);
* составление расписания по преподавателю, по учебной группе и дисциплине, по аудитории;
* формирование соответствующей отчётности;
* произвольная сетка проведения занятий для разных подразделений организации;
* анализ составленного расписания на выполнение требований и критериев.

Также для «1С:Университет ПРОФ» реализован модуль «Портал вуза», объединяющий личные кабинеты абитуриента, студента и преподавателя. В личном кабинете абитуриента реализована возможность дистанционной подачи заявлений.

Личный кабинет студента предоставляет следующие функциональные возможности:

* просмотр расписания;
* просмотр собственной успеваемости;
* просмотр списка приказов, проведенных в отношении обучающегося;
* просмотр учебных планов обучающегося;
* запись на курсы по выбору;
* просмотр тем курсовых и дипломных работ;
* просмотр информации о стипендиях;
* доступ к учебно-методическим материалам (чтение, скачивание);
* добавление информации в собственное портфолио обучающегося;
* общение с другими студентами и преподавателями на форуме;
* просмотр результатов освоения образовательной программы.

Личный кабинет преподавателя предоставляет следующие функциональные возможности:

* просмотр расписания;
* просмотр учебных планов, согласно которым проводит занятия преподаватель;
* доступ к учебно-методическим материалам (чтение, скачивание);
* просмотр портфолио студентов;
* добавление информации в собственное портфолио преподавателя;
* общение со студентами и другими преподавателями на форуме [3].

***1.2.3*** *Moodle* – это свободная система управления обучением, ориентированная прежде всего на организацию взаимодействия между преподавателем и учениками, хотя подходит и для организации традиционных дистанционных курсов, а также поддержки очного обучения [4].

Система *Moodle* обеспечивает несколько уровней доступа:

* администратор системы – это основной пользователь, имеющий полные права на управление системой;
* управляющий имеет доступ к курсу или категории и могут изменять его;
* создатель курса может создавать новые курсы, но, в отличие от управляющего, они не могут управлять категориями;
* учитель может делать в курсе всё, в том числе изменять элементы курса и оценивать студентов;
* ассистент может преподавать в курсах и выставлять оценки, но не может изменять содержание курса;
* студент имеет права на просмотр материалов курса и выполнение различного рода проверочных работ, также может принимать участие в обсуждениях на форумах, в чате, отправлять персональные сообщения другим участникам курса;
* гость может просто посмотреть разделы курса, если это разрешено, но не может выполнять какие-либо виды учебной деятельности [5].

Каждый курс состоит из различных интерактивных элементов, т.е. таких элементов, при использовании которых студент может получать обратную или прямую связь с преподавателем. К таким элементам относят: лекцию, задание, семинар, тест, *wiki*, глоссарий, форум.

Лекция строится по принципу чередования страниц с теоретическим материалом и страниц с обучающими тестовыми заданиями и вопросами. Последовательность переходов со страницы на страницу заранее определяется преподавателем – автором курса, и зависит от того, как студент отвечает на вопрос. На неправильные ответы преподаватель может дать соответствующий комментарий.

Задание позволяет преподавателю ставить задачи, которые требуют от студентов ответа в электронной форме (в любом формате) и дает возможность загрузить его на сервер. Элемент «Задание» позволяет оценивать полученные ответы.

Семинар похож на элемент «Задание», основным отличием от предыдущего элемента является возможность организовать взаимную оценку студенческих работ самими студентами.

Тест позволяет создавать наборы тестовых заданий. Тестовые задания могут быть с несколькими вариантами ответов, с выбором верно (неверно), предполагающие короткий текстовый ответ, на соответствие, эссе и др. Все вопросы хранятся в базе данных и могут быть в последствии использованы снова в этом же курсе (или в других). Тесты могут быть обучающими (показывать правильные ответы) или контрольными (сообщать только оценку).

*Wiki* делает возможной совместную групповую работу обучаемых над документами. Любой участник курса может редактировать *wiki*-статьи. Все правки *wiki*-статей хранятся в базе данных, можно запрашивать любой прошлый вариант статьи или для сравнения разницу между любыми двумя прошлыми вариантами статей с помощью ссылки «Последние правки». Используя инструментарий *Wiki*, обучаемые работают вместе над редактированием одной *wiki*-статьи, обновлением и изменением ее содержания. Редактор, встроенный в *Wiki*, позволяет вставлять в текст статьи таблицы, рисунки и формулы. В зависимости от настроек групповой работы *Moodle* может включать в себя двенадцать различных *wiki*-редакторов. При коллективной работе преподаватель, используя функцию «История», может отследить вклад каждого участника в создании статьи и оценить его.

Глоссарий позволяет создавать и редактировать список определений, как в словаре. Наличие глоссария, объясняющего ключевые термины, употреблённые в учебном курсе, просто необходимо в условиях внеаудиторной самостоятельной работы. Элемент «Глоссарий» облегчает преподавателю задачу создания подобного словаря терминов. В виде глоссария можно организовать также персоналий. Глоссарий может быть открыт для создания новых записей (статей), не только для преподавателя, но и для обучающихся.

Форум используется для организации дискуссии и группируются по темам. После создания темы каждый участник дискуссии может добавить к ней свой ответ или прокомментировать уже имеющиеся ответы. Для того чтобы вступить в дискуссию, пользователь может просто просмотреть темы дискуссий и ответы, которые предлагаются другими. Это особенно удобно для новых членов группы, для быстрого освоения основных задач, над которыми работает группа. История обсуждения этих проблем сохраняется в базе данных. Пользователь также может сыграть и более активную роль в обсуждении, предлагая свои варианты ответов, комментарии и новые темы для обсуждения.

В каждом электронном курсе система *Moodle* дает возможность создания нескольких форумов.

Чат-система предназначена для организации дискуссий и деловых игр в режиме реального времени. Пользователи системы имеют возможность обмениваться текстовыми сообщениями, доступными как всем участникам дискуссии, так и отдельным участникам по выбору.

Форум-система предназначена для организации дискуссий и деловых игр в асинхронном режиме, т.е. в течении длительного времени. Пользователи системы имеют возможность обмениваться текстовыми сообщениями, доступными как всем участникам дискуссии, так и отдельным участникам по выбору.

Опрос служит для проведения быстрых опросов и голосований. Задается вопрос и определяются несколько вариантов ответов.

Анкеты используются для сбора данных, которые могут использоваться в качестве отзывов и оценок студентов об обучении [6].

***1.2.4*** Контур «Управление учебным процессом» системы «Галактика *ERP*» является универсальным решением для образовательных учреждений, осуществляющих подготовку специалистов в области высшего и среднего профессионального образования, и позволяет автоматизировать процессы планирования, учёта, контроля и анализа деятельности образовательного учреждения.

Данный контур состоит из следующих модулей:

* приемная кампания;
* платное обучение;
* учебный процесс;
* управление контингентом студентов;
* учёт успеваемости студентов.

Далее будет описан функционал тех модулей, назначение которых сопоставимо с предметной областью данной дипломной работы.

Модуль «Учебный процесс» предоставляет следующие функциональные возможности:

* формирование учебных планов по специальностям, включая индивидуальные учебные планы;
* планирование потоков учебных групп по видам (общих, специализированных, межфакультетских и проч.), формирование рабочих планов учебных занятий групп;
* формирование сводного линейного графика на год;
* планирование объемов педагогической нагрузки, анализ и учёт выполнения преподавателями педагогической нагрузки;
* планирование объемов нагрузки по руководству всеми видами практик (учебной, ознакомительной, преддипломной);
* учёт и анализ выполнения преподавателями нагрузки по руководству практиками;
* планирование штатов сотрудников по бюджетам (по кафедрам и учебному заведению), необходимых для реализации объемов педагогической нагрузки;
* планирование штатного расписания преподавателей с учетом квалификации (по кафедрам и учебному заведению), необходимых для реализации объемов педагогической нагрузки;
* формирование извещений кафедре о планируемой учебной нагрузке;
* формирование расписания учебного процесса и сетки занятий на период (семестр), в том числе индивидуального расписания, с учётом занятости преподавательского состава и ресурсов аудиторного и лабораторного фондов;
* формирование отчетности и аналитических данных, например, анализ соответствия штатной расстановки штатному расписанию (по качественному составу и показателю штатности), анализ наличия вакансий [7].

Модуль «Управление контингентом студентов» предоставляет следующие функциональные возможности:

* ведение картотеки личных дел студентов (с сохранением истории изменения сведений) с возможностью поиска информации о студенте по штрих-коду на любом из его документов (студенческого билета, зачетной книжки, личного  
  дела);
* учёт перемещений студентов в процессе обучения с сохранением истории перемещений;
* ведение информации о проживании студентов в общежитиях;
* формирование типового набора документов студента (договоров, анкет, заявлений, справок, приказов и т.д.);
* формирование отчетности о движении студентов, анализ контингента студентов (общий, по основам обучения, из числа коренных малочисленных на-  
  родов, сироты, малоимущие выделение студентов, находящихся в академическом отпуске и др.);
* привязка основы обучения;
* ведение информации о стипендии студентов;
* ведение архива студентов, закончивших образование в данном учреждении;
* учет дальнейшего трудоустройства [8].

Модуль «Учёт успеваемости студентов» предоставляет следующие функциональные возможности:

* ведение карточки академической группы;
* ведение журнала посещаемости занятий студентами;
* формирование учебной карточки студента;
* учет контрольных мероприятий (зачетов, экзаменов, курсовых, тестирования и т. д.) в соответствии с учебным планом образовательного учреждения;
* формирование и учет документов (ведомостей) к экзаменам, зачетам, защитам курсовых и дипломных проектов;
* ввод результатов контроля успеваемости студентов по его итогам (зачеты, экзамены, компьютерное тестирование, курсовые и дипломные проекты), идентификация ведомости по штрих-коду;
* ведение экзаменационных листов и повторных экзаменационных ведомостей на пересдачу;
* ведение базы данных выпускных работ, курсовых проектов и т. д.;
* оформление вкладыша для диплома, академической справки;
* анализ посещаемости в различных разрезах (студент, дисциплина, вид работы, период);
* анализ академической задолженности студентов;
* анализ успеваемости студентов в различных разрезах [9].

Помимо описанных ранее модулей, система «Галактика *ERP.* Управление учебным процессом» предоставляет функционал, связанный с личными кабинетами пользователей: абитуриента, обучающегося и сотрудника. Далее будут описаны возможности предоставляемые двумя последними видами личных кабинетов.

Личный кабинет обучающегося предоставляет следующие функциональные возможности:

* личная страница студента (личный профиль) с возможностью ведения дополнительной информации и настройкой приватности;
* просмотр организационной структуры университета с возможностью поиска сотрудников, преподавателей и обучающихся;
* доступ к учебному плану с различной степенью детализации (на весь период обучения, на текущий учебный год или семестр);
* формирование предпочтений по изучению дисциплин вариативной части основной образовательной программы с последующим просмотром персональной траектории обучения;
* доступ к электронной зачётной книжке;
* просмотр персональных приказов и движений (зачисление, переводы, допуски и т.д.);
* просмотр действующих договоров с возможностью онлайн-оплаты;
* просмотр назначенных стипендий;
* онлайн-общение с одногруппниками и преподавателями в тематических группах;
* создание и отправка формализованных заявок различных категорий в администрацию вуза;
* центр документов (шаблоны заявлений, материалы для самостоятельного изучения);
* формирование электронного портфолио достижений;

Личный кабинет сотрудника предоставляет следующие функциональные возможности:

* личная страница сотрудника (личный профиль) с возможностью ведения дополнительной информации и настройкой приватности;
* просмотр организационной структуры университета с возможностью поиска сотрудников, преподавателей и обучающихся;
* ведение реестра собственных достижений в соответствии с положением об эффективном контракте и их отправка на верификацию;
* верификация студенческого портфолио;
* создание и отправка формализованных заявок различных категорий в администрацию вуза (на получение справок, на командировку и др.);
* центр документов (шаблоны заявлений, материалы для самостоятельного изучения и т.п.);
* просмотр персонального расписания в различных разрезах, загруженности аудиторного фонда;
* управление задачами и проектами [10].

**1.3 Обзор используемых технологий**

Так как мы имеем дело с программным обеспечением, сперва необходимо определиться с тем, на каком языке оно будет написано. В данной дипломной работе предпочтение было отдано языку *C#* и архитектуре *.NET* в связи с достаточным для выполнения задачи опытом разработчика.

*C#* – современный объектно-ориентированный и типобезопасный язык программирования. Программы, написанные на языке *C#*, выполняются в *.NET*, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (*CLR*) и набор библиотек классов. Среда *CLR* – это реализация общеязыковой инфраструктуры языка (*CLI*), являющейся международным стандартом, от корпорации Майкрософт. *CLI* является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке *C#,* компилируется в промежуточный язык (*IL*), который соответствует спецификациям *CLI*. Код на языке *IL* и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .*dll*. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки [11].

Для того чтобы с помощью языка *C#* разрабатывать веб-приложения, была выбрана платформа *ASP.NET Core MVC*, которая представляет собой упрощенную, эффективно тестируемую платформу с открытым исходным кодом, оптимизированную для использования с *ASP.NET Core*. *ASP.NET Core MVC* предоставляет основанный на шаблонах способ создания динамических веб-сайтов с четким разделением задач. Она обеспечивает полный контроль разметки, поддерживает согласованную с *TDD* разработку и использует новейшие веб-стандарты.

Платформа *ASP.NET Core MVC* создана на основе маршрутизации *ASP.NET Core* – мощного компонента сопоставления *URL*-адресов, который позволяет создавать приложения с понятными и поддерживающими поиск *URL*-адресами. Существует два вида маршрутизации:

* маршрутизация на основе соглашений;
* маршрутизация на основе атрибутов [12].

Для создания представлений будет использоваться движок представлений *Razor*, который предоставляет синтаксис для интеграции выражений языка *C#* в верстку веб-страницы, написанную на языке *HTML*. *Razor* неявно учитывает контекст элементов *HTML* и правила типизованного языка *C#*, но при этом также позволяет устанавливать явные границы для обозначения того, какие части содержат разметку, а какие – код. Он позволяет создавать шаблоны на основе страниц, включать необязательные части по определенным условиям или подниматься на уровень самого представления [13].

Для аутентификации пользователей, а также распределения ролевых прав между ними, будет использоваться *ASP.NET Core Identity*. Система  
*ASP.NET Core Identity* – это *АРI*-интерфейс для управления пользователями и запоминания пользовательских данных в хранилищах, таких как реляционные базы данных, посредством *Entity Framework Core* [14]*.*

После выбора языка и платформы для создания веб-приложений необходимо определиться с интегрированной средой разработки, которая представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

Для разработки на выбранном языке будет использоваться среда разработки *Microsoft Visual Studio Community 2022*. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред *IDE*, *Visual Studio* включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки [15].

Для хранения данных приложения будет использоваться реляционная встраиваемая система управления базами данных *SQLite*. Данная СУБД обладает рядом преимуществ, среди которых можно выделить следующие: переносимость, легкость в использовании, компактность, производительность и надежность [16].

Для подключения к базе данных и её дальнейшего использования в приложении используется *Entity Framework Core*. *Entity Framework (EF) Core* – это простая, кроссплатформенная и расширяемая версия популярной технологии доступа к данным *Entity Framework* с открытым исходным кодом.

*EF Core* может использоваться в качестве объектно-реляционного модуля сопоставления (*O/RM*), который:

* позволяет разработчикам *.NET* работать с базой данных с помощью объектов *.NET*;
* устраняет необходимость в большей части кода для доступа к данным, который обычно приходится писать.

*Entity Framework Core* поддерживает доступ к множеству разных баз данных с использованием библиотек подключаемых модулей, которые называются поставщиками баз данных.

В *EF Core* доступ к данным осуществляется с помощью модели. Модель состоит из классов сущностей и объекта контекста, который представляет сеанс взаимодействия с базой данных. Объект контекста позволяет выполнять запросы и сохранять данные.

*EF* поддерживает следующие подходы к разработке моделей:

* создание модели на основе существующей базы данных;
* написание кода модели вручную в соответствии с базой данных.

Также *Entity Framework Core* поддерживает функцию миграции, которая позволяет последовательно применять изменения схемы к базе данных, чтобы синхронизировать ее с моделью данных в приложении без потери существующих данных [17].

Для отображения сущностей слоя представления в модели *DTO* слоя бизнес-логики будет использоваться *AutoMapper*.

*Unit*-тестирование будет проводиться при помощи тестового фреймворка *NUnit*, *mock*-тестирование – при помощи фреймворка *Moq*.

Для проектирования физической диаграммы базы данных *SQLite* будет использована программа *DbVisualizer Free*.

**1.4 Постановка задачи**

В результате проведённого анализа предметной области, а также после выполнения обзора основных аналогов программного продукта и описания технических средств для его реализации, должен быть сформулирован некоторый набор основных требований. Выполнение данных требований в конечном и готовом к использованию программном продукте будет свидетельствовать о том, что возложенные на дипломную работу цели были успешно достигнуты, а поставленные задачи решены.

Создаваемое программное обеспечение должно представлять собой *web*-приложение, позволяющее его пользователям выполнять следующие функции:

* администраторам осуществлять полный контроль за всеми данными и учётными записями пользователей, используемыми в приложении: редактировать, добавлять и удалять их;
* методистам деканатов просматривать статус текущей аттестации, а также отчёты о результатах аттестаций как отдельных студентов, так и всех групп, подведомственных данному деканату;
* заведующим кафедр просматривать отчёты о результатах аттестаций как отдельных студентов, так и всех групп, подведомственных данной кафедре;
* преподавателям формировать расписание и список лабораторных работ, заполнять журнал успеваемости и посещаемости для ведомых ими групп, а также вычислять результаты аттестаций и передавать их в деканат;
* студентам просматривать текущую успеваемость и пропуски по дисциплинам, а также результаты аттестаций.

Приведенный выше набор функциональных возможностей может показаться несколько скромным по сравнению с тем широким списком возможностей, которые предоставляют проекты аналоги. Однако, хотя данный программный продукт и не покрывает полностью все нужды образовательного проекта, он имеет ряд важных преимуществ. Среди основных можно выделить следующие преимущества.

Во-первых, разрабатываемый продукт является полностью *web*-приложением, что добавляет ему удобности и переносимости в сравнении с настольными вариантами приложений в «1С:Университет ПРОФ» и «Галактика *ERP*. Управление учебным процессом». Например, это позволяет преподавателю вести журнал успеваемости и посещаемости не имея доступ к компьютеру в аудитории со студентами. Помимо этого, мобильная версия *web*-сайта является оптимальным решением для студентов, которым необязательно иметь компьютер под рукой, чтобы просматривать данные об аттестациях и успеваемости. У перечисленных аналогов так же есть *web*-приложения, однако они имеют ограниченный функционал по сравнению с настольными версиями и служат скорее ознакомительными вариантами касательно данных, которыми оперирует настольная система. То есть они не предоставляют тех же возможностей по созданию и ведению документов, а просто визуализируют результаты работы с ними. К тому же стоит добавить, что *web*-приложения перечисленных аналогов не содержат описанных ранее возможностей для студентов.

Во-вторых, в сравнении с аналогом *Moodle*, который так же, как и разрабатываемое приложение является изначально *web*-решением, продукт, создаваемый в рамках дипломной работы, имеет преимущества в наличии возможности вести промежуточные результаты учёбы (аттестации), а также учитывать пропуски студентами занятий. Это обусловлено тем, что *Moodle* является скорее решением для обеспечения дистанционной коммуникации между преподавателем и студентом, нежели системой мониторинга результатов, что так же проявляется в отсутствии в нём контролирующих учебный процесс ролей, таких как заведующий кафедрой и методист деканата или их аналогов. Из всего вышеуказанного, естественным образом следует, что в *Moodle* отсутствуют различные формы отчётности, представленные в разрабатываемом приложении. Кроме этого, студент получает лишь ограниченное представление о своём участии в учебном процессе, имея доступ лишь к предоставлению отчётов о лабораторных работах, с возможностью получения за них оценки.

В-третьих, поскольку данное приложение разрабатывается с учётом особенностей проведения учебного процесса в конкретном вузе, а также некоторых пожеланий его сотрудников, просто необходимой является более глубокая проработка отдельных нюансов использования приложения, что, в свою очередь, неизбежно ведёт к тому, что в приложении в первую очередь появляется самый ожидаемый и полезный функционал, который может либо отсутствовать в продуктах-аналогах, либо быть труднонаходимым или же чрезмерно перегруженным. Так, например, в приложение должна быть добавлена возможность для преподавателя оставлять заметки для личного пользования о сданных конкретным студентом лабораторных работах. Кроме этого, каждая лабораторная работа имеет несколько стадий сдачи, называемых прогрессом выполнения, которые описаны в аналитическом обзоре предметной области, что является преимуществом, например, по отношению к системе *Moodle*, в которой из всех возможных стадий, по сути, присутствует только одна – возможность прикрепить отчёт.

В заключение стоит сделать вывод о том, что, благодаря описанным выше преимуществам и функциональным особенностям, разрабатываемый в данной дипломной работе программный продукт является вполне конкурентноспособным, а, возможно, в некоторых аспектах даже и незаменимым на данный момент, в той нише, на он которую претендует.

**2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**2.1 Пользовательские роли и функции**

В ходе анализа предметной области были выделены различные типы пользователей программного обеспечения, а также функции, которые каждый из них выполняет. Для того чтобы систематизировать, а также наглядно и кратко описать полученные сведения, уместно использовать диаграммы вариантов использования. Благодаря ним сразу же можно определить все виды пользователей и их иерархию, а также функционал, предоставляемый им программным обеспечением, в обобщенной форме.

На рисунке 2.1 изображена общая иерархия пользователей, из которой можно сделать вывод, что по умолчанию пользователь может только авторизоваться и аутентифицироваться, чтобы затем продолжить работу в приложении в одной из ролей наследников. Далее будут приведены отдельные диаграммы для каждого вида пользователя.

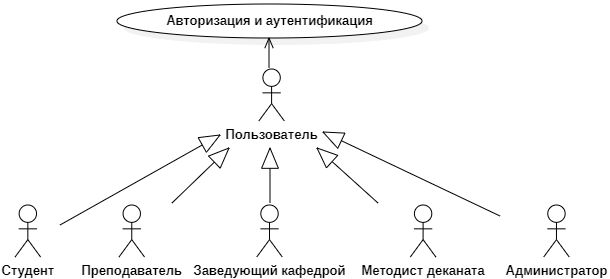


Рисунок 2.1 – Диаграмма вариантов использования общего вида

На рисунке 2.2 приведена диаграмма вариантов использования для студента.

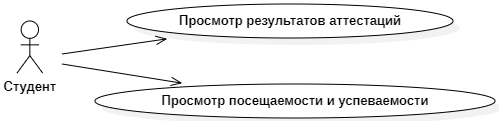


Рисунок 2.2 – Диаграмма вариантов использования для студента

На рисунке 2.3 приведена диаграмма вариантов использования для преподавателя.

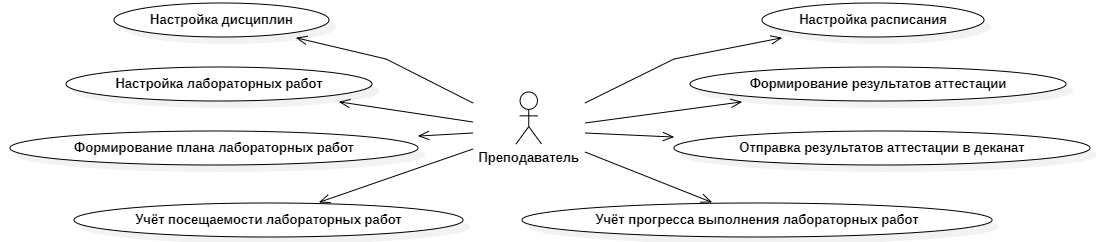


Рисунок 2.3 – Диаграмма вариантов использования для преподавателя

На рисунке 2.4 приведена диаграмма вариантов использования для заведующего кафедрой.

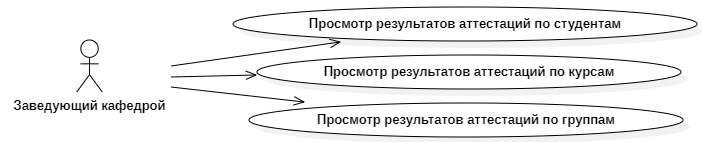


Рисунок 2.4 – Диаграмма вариантов использования для заведующего кафедрой

На рисунке 2.5 приведена диаграмма вариантов использования для методиста деканата.

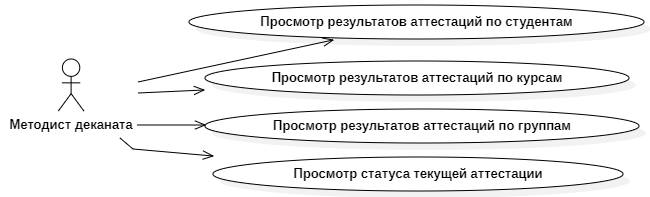


Рисунок 2.5 – Диаграмма вариантов использования для методиста деканата

На рисунке 2.6 приведена диаграмма вариантов использования для администратора. Стоит отметить, что под администрированием системы в данном случае подразумевается ведение с возможностью добавления, редактирования и удаления данных из таблиц базы данных, связанных как с учебным процессом, так и с учётными записями пользователей.



Рисунок 2.6 – Диаграмма вариантов использования для администратора

**2.2 Архитектура приложения**

Для данного проекта была выбрана трёхслойная архитектура. Это значит, что в приложении чётко выделены три слоя: слой доступа к данным, слой бизнес-логики и слой представления. Слой доступа к данным и слой бизнес-логики выделены в отдельные библиотеки. Также, помимо слоёв, присутствует библиотека, в которой хранятся различные постоянные значения, необходимые для конфигурации приложения.

Слой доступа к данным использует шаблон проектирования «Репозиторий» для предоставления независящего от конкретной базы данных интерфейса. Доступ к базе данных *SQLite* производится посредством *EntityFramework Core*. Сам же *EF Core* отображает данные из таблиц баз данных в классы, называемые сущностями. Работа с базой данных ведётся через класс контекста данных, который в данном случае при помощи механизма наследования совмещает, по сути, два контекста: контекст *ASP.NET Core Identity* и контекст, связанный непосредственно с таблицами предметной области.

Слой бизнес-логики содержит сервисы, которые инкапсулируют в себе взаимодействие с интерфейсами репозиториев слоя доступа к данным. Для представления объектов сущностей базы данных в виде объектов, непосредственно моделирующих предметную область, используется шаблон проектирования «*Data Transfer Object*»: с помощью библиотеки *AutoMapper*, для которой на данном слое создан набор специальных профилей, соответствующих преобразованию каждой сущности в *DTO*, объекты сущностей отображаются в *DTO* моделей. Также эти объекты выполняют функцию моделей для слоя представления, т.е. они содержат атрибуты для валидации данных, с помощью которых предметная область становится более выраженной.

Слой представления использует шаблон проектирования «*Model-View-Controller*» для разделения данных приложения и управляющей логики на три компонента: модель, представление и контроллер, что позволяет модифицировать каждый из них вне зависимости от другого.

Для увеличения изолированности и, как следствие, гибкости слоёв используется шаблон проектирования «Внедрение зависимостей». Для этого на каждом слое создаются конфигурационные классы, которые выполняются взаимосвязанно, что и создаёт по итогу общую среду для доступа к зависимостям для каждого нуждающегося в них класса на слое, который и инициирует цепочку зависимостей. Также с помощью внедрения зависимостей осуществляется передача строк соединения для базы данных и подключение к проекту *ASP.NET Core Identity* вместе с контекстом данных.

**2.3 Модель данных и структура базы данных**

В данном подразделе описывается модель данных проектируемого приложения в тех видах, в которых она в нём представлена. Среди данных видов можно выделить два конкретных, использование которых зависит от места хранения данных: в постоянном хранилище или же в оперативной памяти. Постоянным хранилищем в данном проекте служит реляционная СУБД *SQLite*, данные в которой хранятся в виде записей в таблицах. База данных создаётся путём использования подхода *Code First*, предоставляемого технологией *Entity Framework Core*, при котором модель сначала описывается в виде классов-сущностей, которые хранятся в оперативной памяти, на языке *C#*, а потом отображается в выбранную СУБД согласно её правилам, при этом полученная таблица СУБД эквивалентна по столбцам полям сущности языка *C#*. Использование этого подхода в сочетании с несовпадением наименований и представлений типов приводит к тому, что типизация столбцов таблиц и свойств классов-сущностей может различаться. Однако, стоит отметить, что типы, используемые в СУБД, при таком подходе всегда являют собой вариант полностью эквивалентный или же избыточный по затратам памяти по отношению к типам, используемым в *C#*. Особенно заметным это становится в СУБД *SQLite*, где, благодаря использованию механизма *Type Affinity*, диапазон исходных типов сведён к минимуму. Остальные типы приводятся к исходным путём либо изменения размера исходного типа в байтах, либо проведения обратимых преобразований над ними (например, преобразования даты в строку).

Из-за вышеуказанных различий для полного описания модели данных будут приведены наименования типов как в СУБД *SQLite*, так и в языке *C#*. То есть в каждой таблице будет описана таблица СУБД *SQLite* и сущность в языке *C#*, а каждая строка таблицы будет содержать описание столбца таблицы СУБД и свойства сущности *C#*. Помимо этого, при описании модели данных будут приведены примечания к каждому столбцу-свойству, которые, в общем случае, будут содержать просто достаточный для понимания значения перевод. В других случаях примечания будут содержать описание использования данных столбцов-свойств и тип отношений в реляционной СУБД. Таблицы с описанием столбцов-свойств расположены в алфавитном порядке согласно английским наименованиям таблиц СУБД и классов-сущностей в языке *C#*.

Схема модели базы данных в СУБД *SQLite* приведена на рисунке 2.7. Однако не все таблицы СУБД, представленные на этих рисунках, будут описаны в этом подразделе. Некоторые из этих таблиц создаются автоматически технологией *ASP.NET Core Identity*. Такие таблицы имеют в названии приставку *AspNet*, а их назначение заключается в том, чтобы поддерживать различные возможности для авторизации и аутентификации в приложении. Также на рисунке присутствует таблица «*\_EFMigrationsHistory»*, которая создаётся автоматически технологией *Entity Framework Core* и содержит историю миграций, применённых к базе данных. Описание данных автоматически создаваемых таблиц производится не будет.

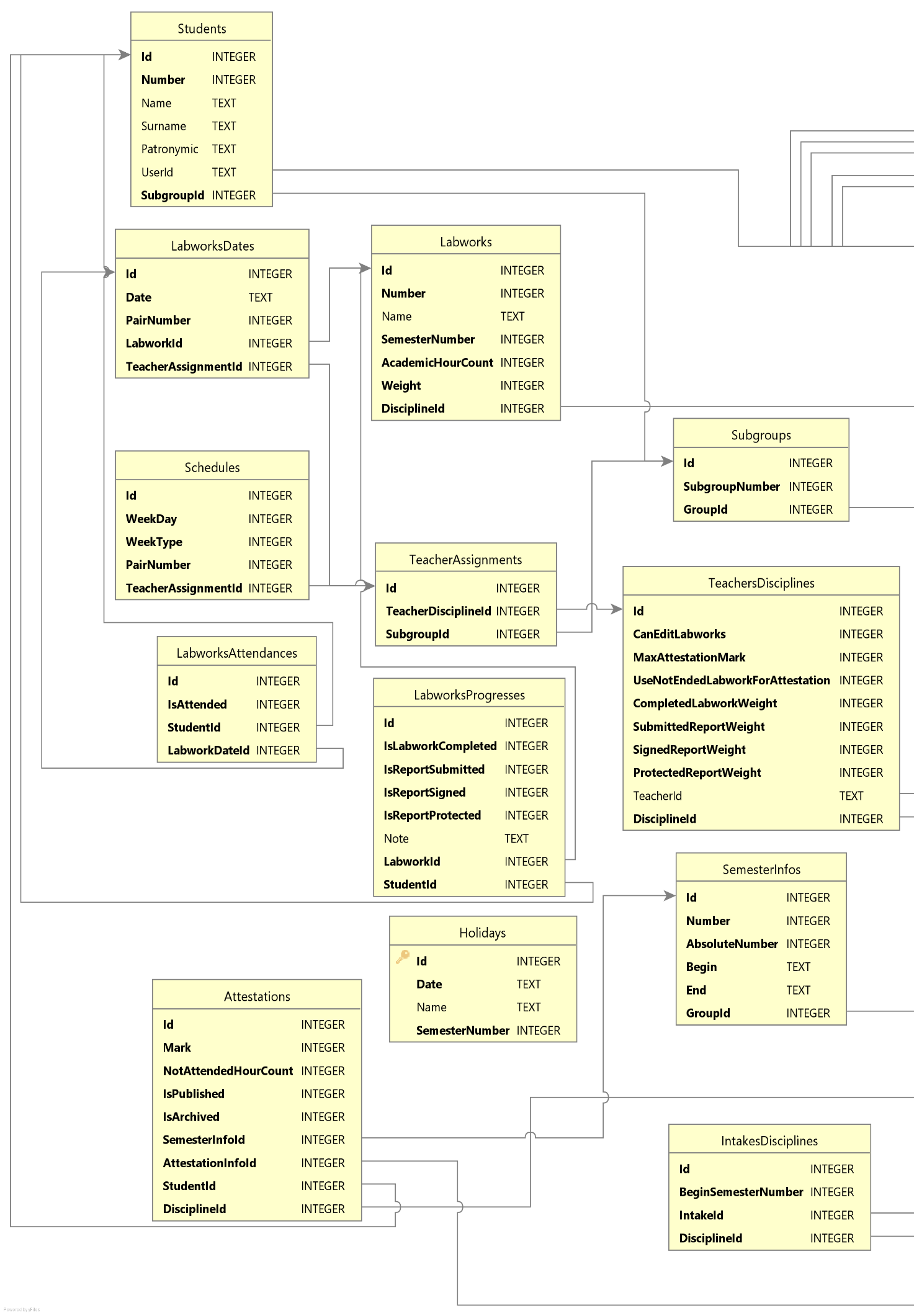


Рисунок 2.7 – Схема модели данных

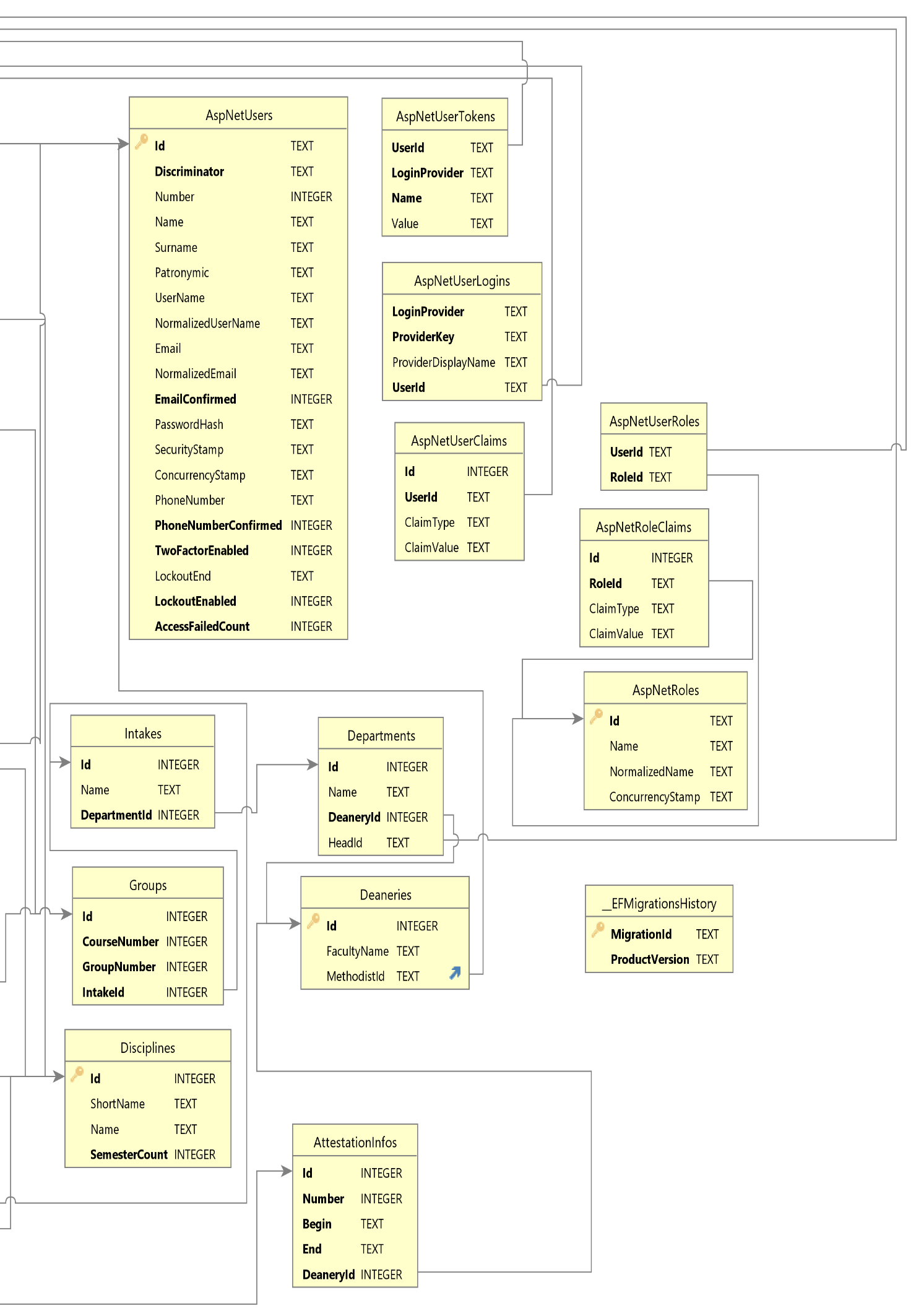


Рисунок 2.7, лист 2

В таблице 2.1 приведено описание таблицы-сущности «*AttestationInfos»*, которая отвечает за хранение информации об аттестациях относительно деканата.

Таблица 2.1 – Описание таблицы-сущности «*AttestationInfos*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Number* | *INTEGER* | *byte* | Номер аттестации в семестре |
| *Begin* | *TEXT* | *DateTime* | Дата и время начала |
| *End* | *TEXT* | *DateTime* | Дата и время окончания |
| *DeaneryId* | *INTEGER* | *int* | Идентификатор деканата, к которому относится информация об аттестации (отношение «многие-к-одному») |

В таблице 2.2 приведено описание таблицы-сущности «*Attestations*», которая отвечает за хранение информации об аттестациях студентов относительно семестра и дисциплины.

Таблица 2.2 – Описание таблицы-сущности «*Attestations*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Mark* | *INTEGER* | *byte* | Оценка по предмету |
| *NotAttendedPairCount* | *INTEGER* | *byte* | Количество непосещённых пар |
| *IsPublished* | *INTEGER* | *bool* | Отметка об отправке аттестации в деканат |
| *IsArchived* | *Integer* | *bool* | Отметка об сохранении аттестации в отчёты |
| *SemesterInfoId* | *INTEGER* | *int* | Идентификатор информации о семестре, к которому относится аттестация (отношение «многие-к-одному») |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *AttestationInfoId* | *INTEGER* | *int* | Идентификатор информации об аттестации, к которой относится аттестация (отношение «многие-к-одному») |
| *StudentId* | *INTEGER* | *int* | Идентификатор студента, к которому относится аттестация (отношение «многие-к-одному») |
| *DisciplineId* | *INTEGER* | *int* | Идентификатор предмета, к которому относится аттестация (отношение «многие-к-одному») |

В таблице 2.3 приведено описание таблицы-сущности «*Deaneries*», которая отвечает за хранение информации об деканатах.

Таблица 2.3 – Описание таблицы-сущности «*Deaneries*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *FacultyName* | *TEXT* | *string* | Наименование факультета |
| *MethodistId* | *TEXT* | *string* | Идентификатор методиста деканата (отношение «один-к-одному») |

В таблице 2.4 приведено описание таблицы-сущности «*Departments*», которая отвечает за хранение информации о кафедрах относительно деканата.

Таблица 2.4 – Описание таблицы-сущности «*Departments*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Name* | *TEXT* | *string* | Наименование |
| *DeaneryId* | *INTEGER* | *int* | Идентификатор деканата, к которому относится кафедра (отношение «многие-к-одному») |

Продолжение таблицы 2.4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *HeadId* | *TEXT* | *string* | Идентификатор заведующего кафедрой (отношение «один-к-одному») |

В таблице 2.5 приведено описание таблицы-сущности «*Disciplines*», которая отвечает за хранение информации об предметах.

Таблица 2.5 – Описание таблицы-сущности «*Disciplines*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *ShortName* | *TEXT* | *string* | Сокращённое наименование |
| *Name* | *TEXT* | *string* | Наименование |
| *SemesterCount* | *INTEGER* | *byte* | Количество семестров на изучение |

В таблице 2.6 приведено описание таблицы-сущности «*Groups*», которая отвечает за хранение информации о группах относительно потока.

Таблица 2.6 – Описание таблицы-сущности «*Groups*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *CourseNumber* | *INTEGER* | *byte* | Номер курса |
| *GroupNumber* | *INTEGER* | *byte* | Номер группы |
| *IntakeId* | *INTEGER* | *int* | Идентификатор потока, к которому относится группа (отношение «многие-к-одному») |

В таблице 2.7 приведено описание таблицы-сущности «*Holidays*», которая отвечает за хранение информации о праздничных днях.

Таблица 2.7 – Описание таблицы-сущности «*Holidays*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |

Продолжение таблицы 2.7

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *Date* | *TEXT* | *DateTime* | Дата |
| *Name* | *TEXT* | *string* | Наименование |
| *SemesterNumber* | *INTEGER* | *byte* | Номер семестра |

В таблице 2.8 приведено описание таблицы-сущности «*IntakeDisciplines*», которая отвечает за хранение информации о предметах относительно потоков с учётом номером начального семестра.

Таблица 2.8 – Описание таблицы-сущности «*IntakeDisciplines*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *BeginSemesterNumber* | *INTEGER* | *byte* | Номер начального семестра |
| *IntakeId* | *INTEGER* | *int* | Идентификатор потока, к которому относится предмет (отношение «многие-к-одному») |
| *DisciplineId* | *INTEGER* | *int* | Идентификатор предмета, к которому относится поток (отношение «многие-к-одному») |

В таблице 2.9 приведено описание таблицы-сущности «*Intakes*», которая отвечает за хранение информации о потоках относительно кафедры.

Таблица 2.9 – Описание таблицы-сущности «*Intakes*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Name* | *TEXT* | *string* | Наименование |
| *DepartmentId* | *INTEGER* | *int* | Идентификатор кафедры, к которой относится поток (отношение «многие-к-одному») |

В таблице 2.10 приведено описание таблицы-сущности «*Labworks*», которая отвечает за хранение информации об лабораторных работах относительно дисциплины.

Таблица 2.10 – Описание таблицы-сущности «*Labworks*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Number* | *INTEGER* | *byte* | Номер |
| *Name* | *TEXT* | *string* | Наименование |
| *SemesterNumber* | *INTEGER* | *byte* | Номер семестра |
| *AcademicHourCount* | *INTEGER* | *int* | Количество академических часов на выполнение |
| *Weight* | *INTEGER* | *byte* | Вес в общем списке |
| *DisciplineId* | *INTEGER* | *int* | Идентификатор предмета, к которому относится лабораторная работа (отношение «многие-к-одному») |

В таблице 2.11 приведено описание таблицы-сущности «*LabworkAttendances*», которая отвечает за хранение информации о студентах относительно дат с отметкой о посещении, т.е. о посещаемости лабораторных работ.

Таблица 2.11 – Описание таблицы-сущности «*LabworkAttendances*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *IsAttended* | *INTEGER* | *bool* | Отметка о посещаемости |
| *StudentId* | *INTEGER* | *int* | Идентификатор студента, к которому относится дата лабораторной работы (отношение «многие-к-одному») |
| *LabworkDateId* | *INTEGER* | *int* | Идентификатор даты лабораторной работы, к которой относится студент (отношение «многие-к-одному») |

В таблице 2.12 приведено описание таблицы-сущности «*LabworkDates*», которая отвечает за хранение информации о лабораторных работах относительно назначений преподавателей с привязкой к дате, т.е. о датах лабораторных работ.

Таблица 2.12 – Описание таблицы-сущности «*LabworkDates*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Date* | *TEXT* | *DateTime* | Дата |
| *PairNumber* | *INTEGER* | *byte* | Номер пары |
| *LabworkId* | *INTEGER* | *int* | Идентификатор лабораторной работы, к которому относится назначение преподавателя (отношение «многие-к-одному») |
| *TeacherAssignmentId* | *INTEGER* | *int* | Идентификатор назначения преподавателя, к которому относится лабораторная работа (отношение «многие-к-одному») |

В таблице 2.13 приведено описание таблицы-сущности «*LabworkProgresses*», которая отвечает за хранение информации о студентах относительно лабораторных работ с отметками о выполнении, т.е. об успеваемости по лабораторным работам.

Таблица 2.13 – Описание таблицы-сущности «*LabworkProgresses*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *IsLabworkCompleted* | *INTEGER* | *bool* | Отметка о выполнении лабораторной работы |
| *IsReportSubmitted* | *INTEGER* | *bool* | Отметка о предоставлении отчёта по лабораторной работе |
| *IsReportSigned* | *INTEGER* | *bool* | Отметка о проверке отчёта по лабораторной работе и подписи к защите |
| *IsReportProtected* | *INTEGER* | *bool* | Отметка о защите отчёта по лабораторной работе |
| *Note* | *TEXT* | *string* | Заметки преподавателя о выполнении студентом лабораторной работы |

Продолжение таблицы 2.13

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *LabworkId* | *INTEGER* | *int* | Идентификатор лабораторной работы, к которой относится студент (отношение «многие-к-одному») |
| *StudentId* | *INTEGER* | *int* | Идентификатор студента, к которому относится лабораторная работа (отношение «многие-к-одному») |

В таблице 2.14 приведено описание таблицы-сущности «*Schedules*», которая отвечает за хранение информации о расписаниях относительно назначения преподавателя.

Таблица 2.14 – Описание таблицы-сущности «*Schedules*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *WeekDay* | *INTEGER* | *byte* | День недели |
| *WeekType* | *INTEGER* | *byte* | Тип недели |
| *PairNumber* | *INTEGER* | *byte* | Номер пары |
| *TeacherAssignmentId* | *INTEGER* | *int* | Идентификатор назначения преподавателя, к которому относится расписание (отношение «многие-к-одному») |

В таблице 2.15 приведено описание таблицы-сущности «*SemesterInfos»*, которая отвечает за хранение информации об семестрах относительно группы.

Таблица 2.15 – Описание таблицы-сущности «*SemesterInfos*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Number* | *INTEGER* | *byte* | Номер семестра в курсе |
| *AbsoluteNumber* | *INTEGER* | *byte* | Номер семестра у группы |

Продолжение таблицы 2.15

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *Begin* | *TEXT* | *DateTime* | Дата и время начала |
| *End* | *TEXT* | *DateTime* | Дата и время окончания |
| *GroupId* | *INTEGER* | *int* | Идентификатор группы, к которой относится информация о семестре (отношение «многие-к-одному») |

В таблице 2.16 приведено описание таблицы-сущности «*Students»*, которая отвечает за хранение информации о студентах относительно подгруппы.

Таблица 2.16 – Описание таблицы-сущности «*Students*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *Number* | *INTEGER* | *uint* | Номер зачётной книжки |
| *Name* | *TEXT* | *string* | Имя |
| *Surname* | *TEXT* | *string* | Фамилия |
| *Patronymic* | *TEXT* | *string* | Отчество |
| *UserId* | *TEXT* | *string* | Идентификатор пользователя (отношение «один-к-одному») |
| *SubgroupId* | *INTEGER* | *int* | Идентификатор подгруппы, к которой относится студент (отношение «многие-к-одному») |

В таблице 2.17 приведено описание таблицы-сущности «*Subgroups»*, которая отвечает за хранение информации о подгруппах относительно группы.

Таблица 2.17 – Описание таблицы-сущности «*Subgroups*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *SubgroupNumber* | *INTEGER* | *byte* | Номер |

Продолжение таблицы 2.17

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *GroupId* | *TEXT* | *string* | Идентификатор группы, к которой относится подгруппа (отношение «многие-к-одному») |

В таблице 2.18 приведено описание таблицы-сущности «*TeacherAssignments»*, которая отвечает за хранение информации о предметах преподавателей относительно подгрупп, т.е. о назначении преподавателей.

Таблица 2.18 – Описание таблицы-сущности «*TeacherAssignments*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *TeacherDisciplineId* | *INTEGER* | *int* | Идентификатор предмета преподавателя, к которому относится подгруппа (отношение «многие-к-одному») |
| *SubgroupId* | *INTEGER* | *int* | Идентификатор подгруппы, к которой относится предмет преподавателя (отношение «многие-к-одному») |

В таблице 2.19 приведено описание таблицы-сущности «*TeacherDisciplines*», которая отвечает за хранение информации о предметах относительно преподавателей с дополнительными параметрами.

Таблица 2.19 – Описание таблицы-сущности «*TeacherDisciplines*»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование столбца-свойства | Тип данных столбца | Тип данных свойства | Примечания |
| 1 | 2 | 3 | 4 |
| *Id* | *INTEGER* | *int* | Идентификатор |
| *CanEditLabwork* | *INTEGER* | *bool* | Отметка о разрешении редактировать лабораторные работы |
| *MaxAttestationMark* | *INTEGER* | *byte* | Максимальная оценка за аттестацию |

Продолжение таблицы 1.19

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| *UseNotEndedLabwork-ForAttestation* | *INTEGER* | *bool* | Отметка об использовании незаконченных лабораторных работ в аттестации |
| *ProvidedLabworkWeight* | *INTEGER* | *byte* | Вес предоставленной лабораторной работы |
| *ProvidedReportWeight* | *INTEGER* | *byte* | Вес предоставленного отчёта по лабораторной работе |
| *AllowedDefenseWeight* | *INTEGER* | *byte* | Вес проверенного отчёта и допуска к защите |
| *ProvidedDefenseWeight* | *INTEGER* | *byte* | Вес предоставленной защиты |
| *TeacherId* | *INTEGER* | *int* | Идентификатор преподавателя, к которому относится предмет (отношение «многие-к-одному») |
| *DisciplineId* | *INTEGER* | *int* | Идентификатор предмета, к которому относится преподаватель (отношение «многие-к-одному») |

**3 СТРУКТУРА И ОСНОВНЫЕ АЛГОРИТМЫ СОЗДАВАЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**3.1 Описание слоя доступа к данным**

Слой доступа к данным представляет собой тот слой спроектированной архитектуры приложения, который предоставляет другим слоям унифицированный доступ к источникам данных. Общая структура данного слоя представлена на рисунке 3.1.

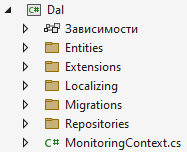


Рисунок 3.1 – Общая структура слоя доступа к данным

Прежде всего в структуре слоя стоит обратить внимание на класс «*MonitoringContext*», который представляет собой инструмент, связующий таблицы базы данных и классы-сущности приложения, расположенные в папке *Entities*. Для каждой сущности контекст содержит свой *DbSet* – коллекцию, при вызове методов которой происходит выполнение запросов к базе данных. Список классов-сущностей представлен на рисунке 3.2.

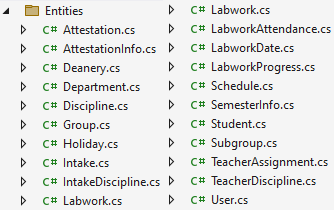


Рисунок 3.2 – Классы-сущности приложения

Среди классов-сущностей необходимо выделить класс «*User*», который наследуется от встроенного класса «*IdentityUser*», и представляет собою расширенную версию класса пользователя, которая помимо унаследованных свойств содержит также номер зачётной книжки для студента и фамилию, имя, отчество пользователя.

Для того чтобы сделать доступ к данным независимым от конкретного источника, в приложении используется шаблон проектирования «Репозиторий». Результат достигается путём вынесения необходимых для корректного и всестороннего доступа к данным методов в интерфейсы, что обязует каждый класс, реализующий их, создавать собственные версии этих методов. Классы, реализующие интерфейсы репозиториев, в свою очередь, являются зависимыми от конкретной платформы хранения данных.

Исполнение шаблона проектирования «Репозиторий» в данном приложении представлено различными интерфейсами и классами, которые сопоставляются с определёнными сущностями. Интерфейсы, а также абстрактный класс «*RepositoryBase*», приведены на рисунке 3.3.

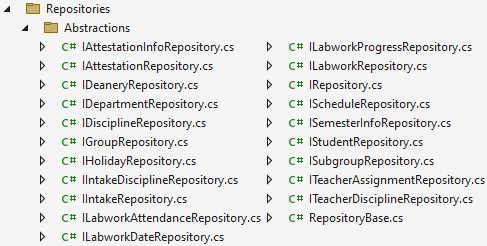


Рисунок 3.3 – Интерфейсы репозиториев

На рисунке 3.4 приведены классы, реализующие интерфейсы репозиториев.

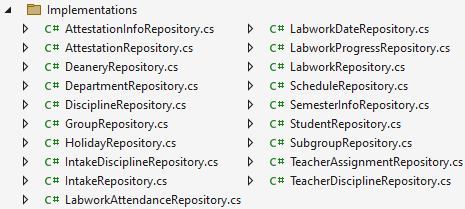


Рисунок 3.4 – Классы репозиториев

Интерфейс «*IRepository*» описывает те методы, которые должен реализовывать любой из репозиториев, относящихся к конкретной сущности. Стоит отметить, что данный интерфейс является обобщённым по типу сущности и типу идентификатора, что позволяет создавать для него реализации в различных сущностях. Методы этого интерфейса по возможности являются асинхронными, что выражается в возвращаемом типе «*Task*». Интерфейсы репозиториев, содержащие в своём названии имена сущностей, служат для того, чтобы добавлять к конкретным репозиториям свой собственный функционал. Методы интерфейса «*IRepository*» описаны в таблице 3.1.

Таблица 3.1 – Описание методов интерфейса «*IRepository*»

|  |  |  |
| --- | --- | --- |
| Название | Возвращаемое значение (параметры) | Описание |
| *Create* | *Task*<*TEntity*> (*TEntity*) | Служит для создания сущности типа «*TEntity*» |
| *GetById* | *Task*<*TEntity*> (*TId*) | Служит для поиска сущности типа «*TEntity*» по её идентификатору типа «*TId*» |
| *GetAll* | *Task*<*IEnumerable*<*TEntity*>> (*Expression*<*Func*<*TEntity*, *bool*>>) | Служит для выборки всех сущностей типа «*TEntity*» по переданной в *Expression* функции отбора |
| *Update* | *Task*<*TEntity*> (*TEntity*) | Служит для обновления сущности типа «*TEntity*» |
| *Delete* | *Task* (*TId*) | Служит для удаления сущности по её идентификатору типа «*TId*» |
| *QueryEntities* | *IQueryable*<*TEntity*> () | Служит для получения прямого доступа к базе данных |

Класс «*RepositoryBase*» содержит базовые реализации методов интерфейса с возможностью их дальнейшего переопределения. Каждый класс, реализующий интерфейс репозитория наследуется от него.

Папка «*Migrations*» содержит информацию о миграциях, созданных для базы данных, что позволяет модифицировать её без потери введённых ранее в неё данных.

Папка «*Localizing*» содержит один класс «*RussianIdentityErrorDescriber*», который предоставляет перевод на русский текстов различных ошибок для *ASP.NET Core Identity*.

Папка «*Extensions*» содержит один класс «*ServiceCollectionExtensions*», который содержит метод расширения «*ConfigureDalServices*». Данный метод отвечает за внедрение зависимостей слоя доступа к данным в контейнер зависимостей «*IServiceCollection*».

**3.2 Описание слоя бизнес-логики**

Слой бизнес-логики представляет собой слой, который содержит в себе необходимые для реализации логики приложения классы. Данный слой служит посредником между слоем доступа к данным и слоем пользовательского интерфейса, что позволяет пользовательскому интерфейсу полностью абстрагироваться от источников данных и работать с чистой логикой предметной области. Общая структура данного слоя представлена на рисунке 3.5.

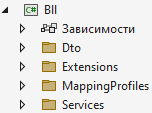


Рисунок 3.5 – Общая структура слоя бизнес-логики

Папка «*Dto*» содержит ряд классов, являющиеся реализацией шаблона проектирования «*Data Transfer Object*» (*DTO*), в которые отображаются сущности слоя доступа к данным при переходе в слой бизнес-логики. Это, как было сказано ранее, необходимо для того, чтобы отделить логику предметной области от данных в чистом виде путём преобразования их в вид удобный для выполнения над ними операций данного слоя и слоя пользовательского интерфейса. Также в *DTO*-классах присутствуют атрибуты валидации, которые используются для проверки корректности данных, вводимых в эти поля на уровне интерфейса пользователя. Список данных классов представлен на рисунке 3.6.

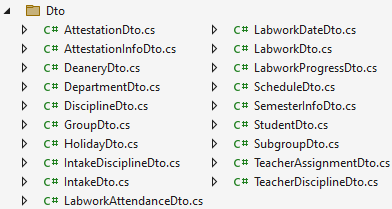


Рисунок 3.6 – Список *DTO*-классов

Отображение классов сущностей в *DTO*-классы осуществляется при помощи библиотеки «*AutoMapper*». Для использования данной библиотеки создаётся набор профилей, каждый из которых отвечает за хранение правил преобразования для одной конкретной сущности в конкретный *DTO*-класс. Данные профили содержатся в папке «*MappingProfiles*», а их список представлен на рисунке 3.7.

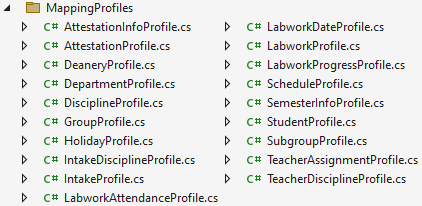


Рисунок 3.7 – Список профилей отображений

Непосредственно логика предметной области реализуется с помощью набора сервисов бизнес-логики, которые структурно организованы аналогично репозиториям. Список интерфейсов данных сервисов, включающий также абстрактный класс «*ServiceBase*», отвечающий за базовую реализацию методов интерфейса «*IService*», представлен на рисунке 3.8.

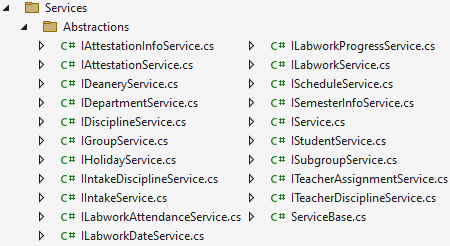


Рисунок 3.8 – Интерфейсы сервисов

Интерфейс «*IService*» описывает те методы, которые должен реализовывать любой из сервисов, относящихся к конкретному *DTO*-классу. Стоит отметить, что данный интерфейс является обобщённым по типу *DTO*-класса и типу идентификатора, что позволяет создавать для него реализации в различных *DTO*-классах. Методы этого интерфейса являются асинхронными, что выражается в возвращаемом типе «*Task*». Интерфейсы сервисов, содержащие в своём названии имена сущностей, служат для того, чтобы добавлять к конкретным сервисам свой собственный функционал.

На рисунке 3.9 приведены классы, реализующие интерфейсы сервисов.

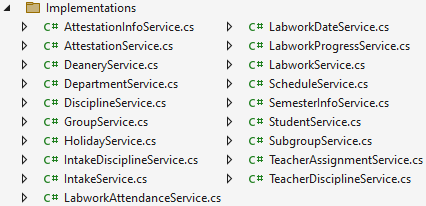


Рисунок 3.9 – Классы сервисов

Среди всех классов сервисов, которые, в общем, выполняют достаточно стандартные операции, стоит выделить два класса: «*AttestationService*» и «*ScheduleService*». Первый, помимо всего прочего, содержит метод «*Create*», который отвечает за автоматический расчёт результатов аттестации у заданной подгруппы по заданной дисциплине, ведомой заданным преподавателем. Второй содержит метод «*CreateJournal*», который отвечает за генерацию журнала, состоящего из меток об успеваемости и дат лабораторных работ с их посещаемостью, на основании предоставленного расписания преподавателя по заданной дисциплине у заданной подгруппы.

Методы интерфейса «*IService*» описаны в таблице 3.2.

Таблица 3.2 – Описание методов интерфейса «*IService*»

|  |  |  |
| --- | --- | --- |
| Название | Возвращаемое значение (параметры) | Описание |
| *Create* | *Task*<*TDto*> (*TDto*) | Служит для создания класса типа «*TDto*» |
| *GetById* | *Task*<*TDto*> (*TId*) | Служит для поиска класса типа «*TDto*» по его идентификатору типа «*TId*» |
| *GetAll* | *Task*<*IEnumerable*<*TDto*>> () | Служит для выборки всех классов типа «*TDto*» |
| *Update* | *Task*<*TDto*> (*TDto*) | Служит для обновления класса типа «*TDto*» |
| *Delete* | *Task* (*TId*) | Служит для удаления класса по его идентификатору типа «*TId*» |

В папке «*Extensions*» содержится один класс «*ServiceCollectionExtensions*», который аналогично классу из слоя доступа к данным отвечает за реализацию внедрения зависимостей в приложении. Он содержит один метод расширения «*ConfigureBllServices*» для контейнера зависимостей «*IServiceCollection*», который, в свою очередь, подключает все необходимые для успешной работы программы зависимости слоя бизнес-логики. Также данный метод по цепочке вызывает метод «*ConfigureDalServices*» для подключения зависимостей из слоя доступа к данным.

**3.3 Описание слоя пользовательского интерфейса**

Слой пользовательского интерфейса представляет собой слой, который содержит все необходимые инструменты для обеспечения взаимодействия приложения с пользователем. Для удобства он был реализован с помощью двух проектов: первый проект содержит элементы пользовательского интерфейса только для администратора, а второй – для всех пользовательских остальных ролей.

В обоих проектах архитектура данного слоя строится на основании шаблона проектирования «*Model-View-Controller*» (*MVC*) и, естественно, имеют сходную структуру. В общем виде структура слоя пользовательского интерфейса представлена на рисунке 3.10.

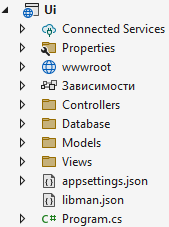


Рисунок 3.10 – Структура слоя пользовательского интерфейса

Класс «*Program*» содержит код, отвечающий за настройку и запуск приложения. В частности, там содержится вызов метода расширения «*ConfigureBll-Services*», который отвечает за внедрение зависимостей в коллекцию зависимостей «*IServiceCollection*» по цепочке, начиная с слоя бизнес-логики. Кроме этого, там содержатся вызовы методов, которые предоставляют возможности для аутентификации и авторизации пользователя в приложении.

Папка «*Controllers*» содержит классы контроллеров, которые отвечают за обработку логики уровня представлений и за связь слоя пользовательского интерфейса со слоем бизнес-логики. Список контроллеров для администратора представлен на рисунке 3.11.

Можно отметить, что в случае проекта для администратора, все контроллеры в основании выполняют однотипные операции: добавление, обновление, вывод в таблице и редактирование записей. Для того чтобы объединить общий для всех контроллеров код, был создан базовый контроллер «*BaseController*», который обращается к общему на уровне интерфейса «*IService*» функционалу сервисов. Различный же для разных сервисов функционал используется уже именными контроллерами. В проекте для всех остальных ролей такого объединения не производилось.

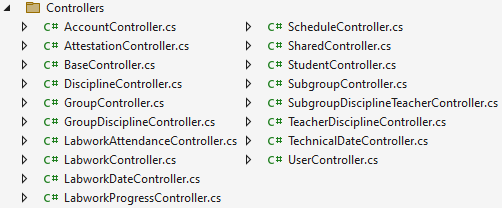


Рисунок 3.11 – Список контроллеров для администратора

Список контроллеров для всех остальных пользовательских ролей представлен на рисунке 3.12.

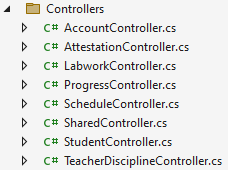


Рисунок 3.12 – Список контроллеров для остальных ролей

В папке «*Database*» хранятся файлы базы данных под управлением СУБД SQLite.

Папка «*Models*» содержит в себе модели представлений. Они нужны для того, чтобы в систематизированном порядке передавать информацию из контроллера в модель и наоборот. Список моделей представлен на рисунке 3.13. Он идентичен для обоих проектов.

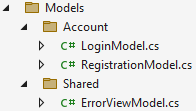


Рисунок 3.13 – Список моделей представлений

Папка «*Views*» содержит представления, которые являются, по сути, *html*-страницами, содержащими выражения на языке движка *Razor*. Перед тем как быть отправленными клиенту, все выражения вычисляются, и страница отправляется в виде чистого *html*.

Представления для администратора являются однотипными для всех *DTO*-классов, и поэтому их список приводиться не будет. Набор представлений для каждого класса включает четыре вида:

* *Create* – представление, которое выводит форму для заполнения, и, после нажатия на кнопку создания, добавляет в базу данных запись указанного типа с введёнными в форму данными;
* *Update* – представление, которое выводит форму для изменения, содержащую предыдущие данные записи, и, после нажатия на кнопку сохранения, изменяет в базе данных указанную запись на введённые в форму данные;
* *Delete* – представление, которое выводит данные об удаляемой записи с просьбой о подтверждении удаления, и, после нажатия на кнопку удаления, удаляет указанную запись из базы данных;
* *Index* – представление, которое выводит таблицу со всеми записями данного типа, содержащимися в базе данных, а также с кнопками для создания новой записи, изменения или удаления существующей.

Список представлений для остальных пользовательских ролей приведён на рисунке 3.14.

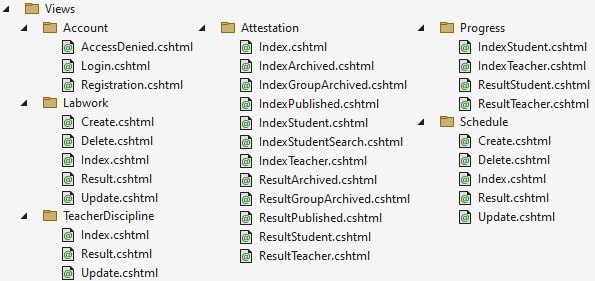


Рисунок 3.14 – Список представлений для остальных ролей

После того как структура слоя пользовательского интерфейса была подробно описана, будут приведены типовые компоненты каждой страницы, отображаемой пользователю, т.е. схематически будут описаны возможные варианты вёрстки представлений.

Любое представление начинается с навигационной панели, внешний вид которой изображён на рисунке 3.15.



Рисунок 3.15 – Внешний вид навигационной панели

Навигационная панель состоит из двух частей. Первая содержит все категории, доступные к просмотру для пользователя в данной роли. Вторая содержит описание типа аккаунта вместе с ФИО пользователя и ссылку для выхода.

После навигационной панели идёт форма для конкретизации выбора, содержащая один или нескольких выпадающих списков и кнопку для вывода информации по выбранным критериям. Вид данной формы приведён на рисунке 3.16.

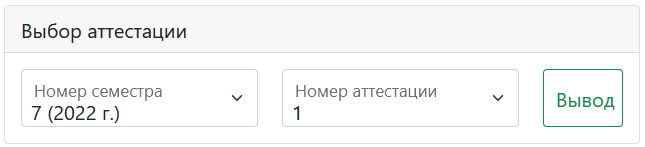


Рисунок 3.16 – Внешний вид формы для выбора

После того как выбор был осуществлён выводится карточка с результатами. Пример такой карточки изображён на рисунке 3.17.

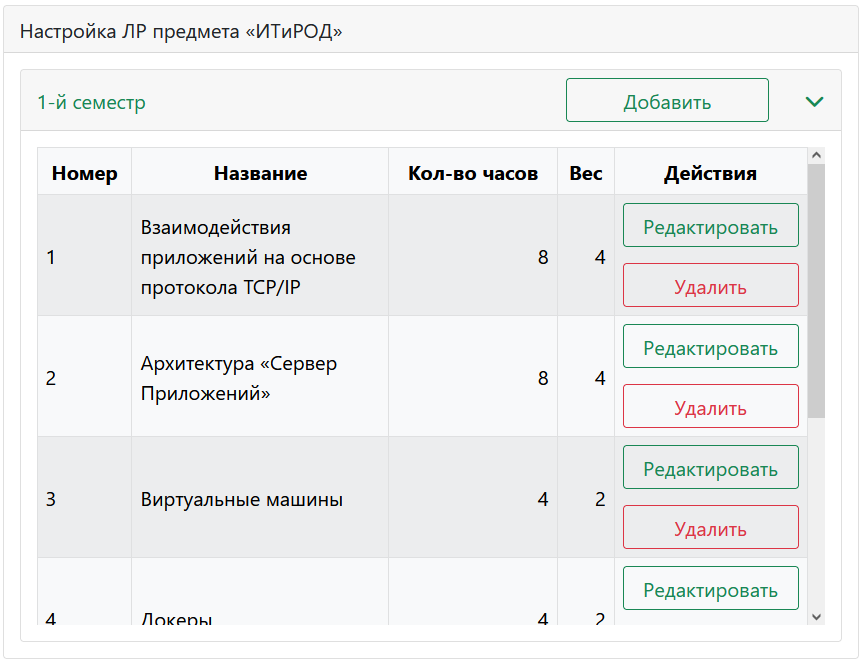


Рисунок 3.17 – Внешний вид карточки с результатами

Результаты, выведенные на карточке, указанной ранее, оформляются, как правило, в виде одной таблицы или в виде списка с несколькими сворачиваемыми таблицами. Помимо таблиц карточка может содержать кнопки для осуществления действий (например, создания записей). Таблицы также могут содержать кнопки для изменения или удаления записей. Если таблица не помещается на экране, то на неё добавляются полосы для вертикальной или горизонтальной прокрутки. Сверху карточки содержится заголовок, который позволяет понять, что именно было выбрано.

При нажатии на кнопку добавления или изменения открывается страница, содержащая форму с полями для ввода данных. Данные поля отражают тип данных, который в них вводится. Помимо этого, на форме присутствуют кнопки для сохранения или добавления, а также для возврата на предыдущую страницу. Вид данной карточки представлен на рисунке 3.18.

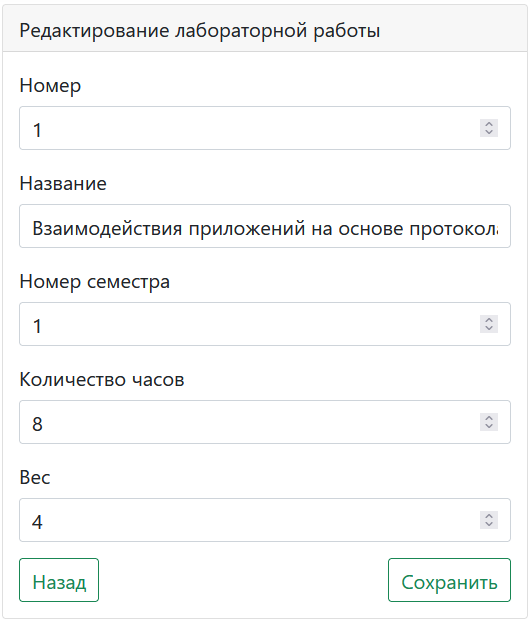


Рисунок 3.18 – Внешний вид формы для изменения данных

Поля каждой формы для редактирования или удаления подвергаются валидации, т.е. проверке на заполненность, соответствие типу данных, диапазону значений и т.п. Если же валидация не проходит, то вверху формы под заголовком карточки красным цветом выводится список ошибок. Внешний вид списка ошибок приведён на рисунке 3.19.

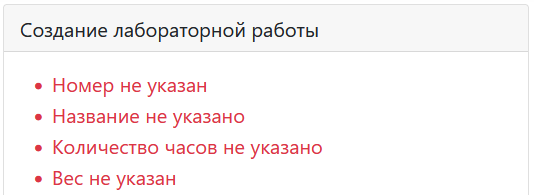


Рисунок 3.19 – Внешний вид списка ошибок валидации

Форма для удаления данных выглядит аналогично формам для создания или изменения, указанным ранее, за тем исключением, что она выводит данные об удаляемом объекте в виде таблицы, а не полей. На ней так же присутствуют две кнопки: для подтверждения удаления и для возврата на предыдущую страницу. Внешний вид данной формы представлен на рисунке 3.20.

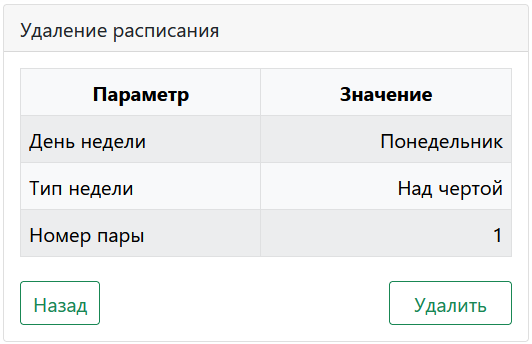


Рисунок 3.20 – Внешний вид формы для удаления записи

Также на странице могут отображаться уведомления об успешном или неуспешном совершении каких-либо действий. Данные уведомления, в случае ошибки, отображают её текст. Внешний вид уведомления представлен на рисунке 3.21.

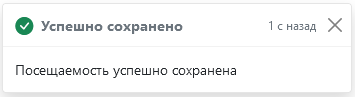


Рисунок 3.21 – Внешний вид уведомления

Код всех трёх слоёв приведён в приложении А.

**4 ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ**

**4.1 Верификация функционала приложения**

Если пользователь, открывший приложение, не авторизован в нем, то навигационная панель примет вид, указанный на рисунке 4.1. Размер навигационной панели урезан и будет урезаться для того, чтобы вместить её на странице.



Рисунок 4.1 – Вид навигационной панели без авторизации

Регистрация производится только в роль студента. Далее его идентификация будет производиться по номеру зачётной книжки. Вид страницы регистрации представлен на рисунке 4.2.

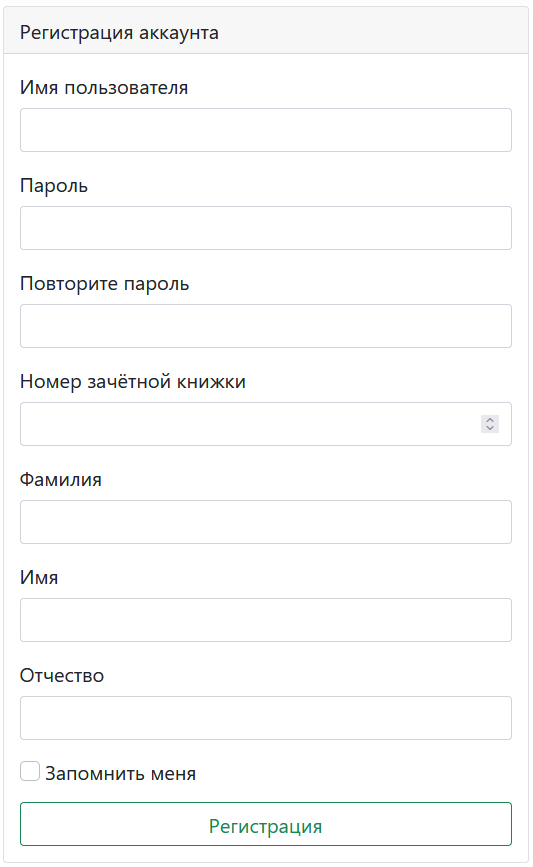


Рисунок 4.2 – Вид страницы регистрации

Кроме регистрации существует страница для входа под уже зарегистрированным пользователем в любой роли. Внешний вид страницы входа представлен на рисунке 4.3.

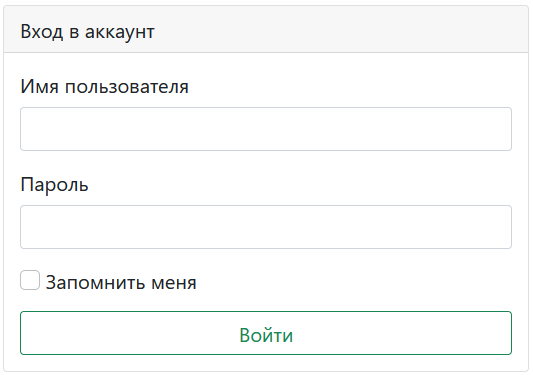


Рисунок 4.3 – Вид страницы входа в аккаунт

Навигационная панель пользователя, вошедшего в систему в роли администратора, принимает вид, указанный на рисунке 4.4.



Рисунок 4.4 – Навигационная панель администратора

Поскольку весь функционал администратора сводится к ведению записей в базе данных, что является однотипной, не требующей значимых различий в интерфейсе и обработке поведения задачей, полное описание всех страниц управления различными типами записей, доступных администратору, производиться не будет. Как пример была выбрана вкладка по ведению дисциплин. Все действия, показанные на ней, идентичны действиям в других вкладках.

На рисунке 4.5 показана страница по изменению дисциплины, которая, по сути, идентична странице добавления.

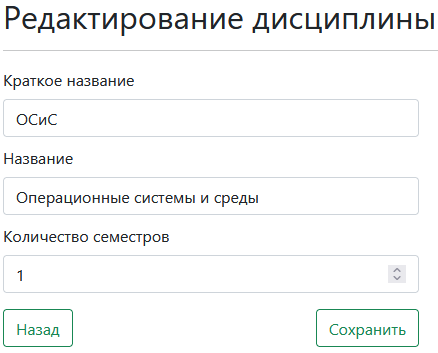


Рисунок 4.5 – Вид страницы для изменения дисциплины

На рисунке 4.6 показана страница для просмотра списка всех дисциплин, с возможностью добавления новой дисциплины, либо изменения или удаления уже существующей дисциплины.

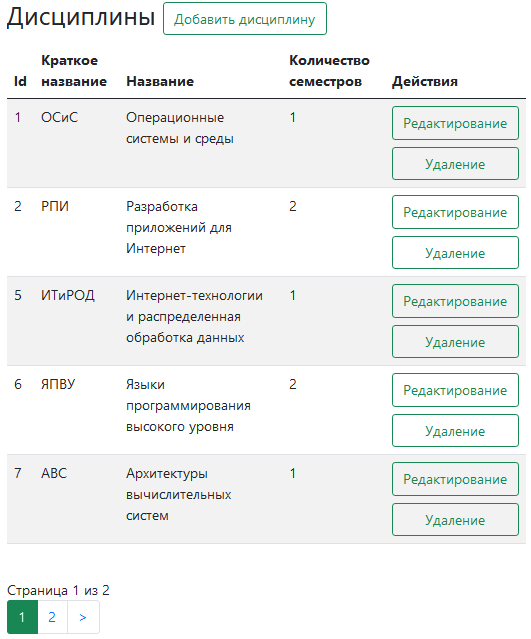


Рисунок 4.6 – Вид страницы для просмотра дисциплин

На рисунке 4.7 показана страница подтверждения, показываемая администратору при удалении дисциплины.

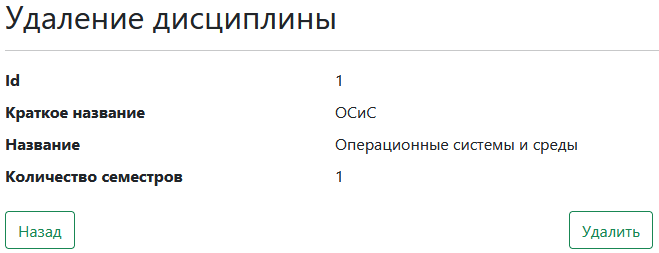


Рисунок 4.7 – Вид страницы для удаления дисциплины

Навигационная панель пользователя, вошедшего в систему в роли студента, принимает вид, указанный на рисунке 4.8.



Рисунок 4.8 – Навигационная панель студента

На первой странице у студента есть возможность просмотреть результаты по аттестации за выбранный семестр. Вид данной страницы представлен на рисунке 4.9.



Рисунок 4.9 – Вид страницы для просмотра аттестаций

На второй странице студент может просмотреть свою успеваемость и посещаемость по одному из предметов, которые назначены ему на данный момент. Вид данной страницы представлен на рисунке 4.10.

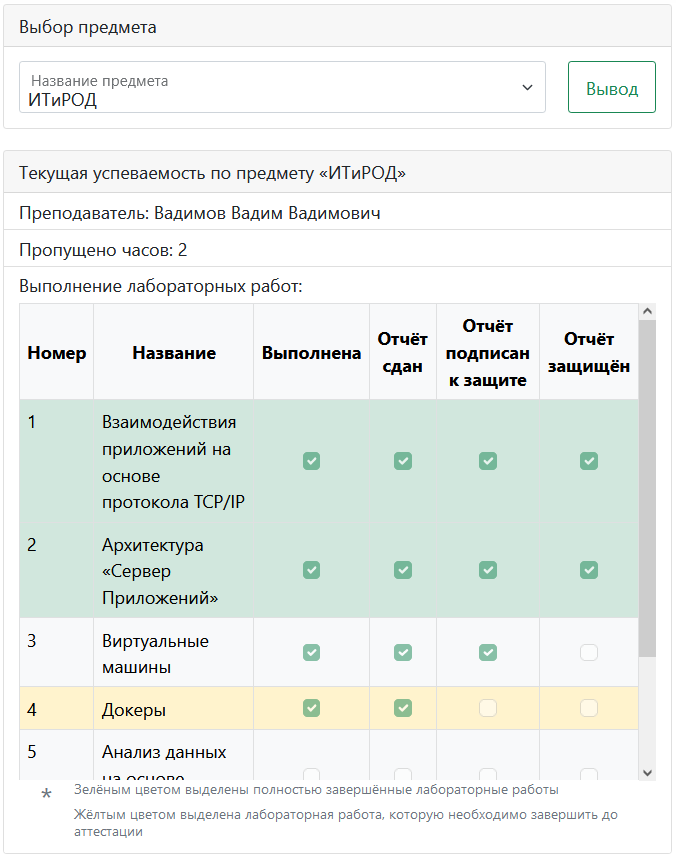


Рисунок 4.10 – Вид страницы для просмотра успеваемости и посещаемости

Навигационная панель пользователя, вошедшего в систему в роли преподавателя, принимает вид, указанный на рисунке 4.11.

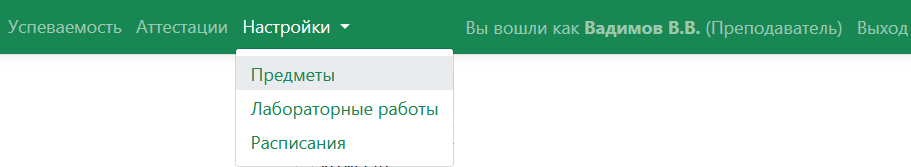


Рисунок 4.11 – Навигационная панель преподавателя

На рисунке 4.12 показана страница для настройки предмета.

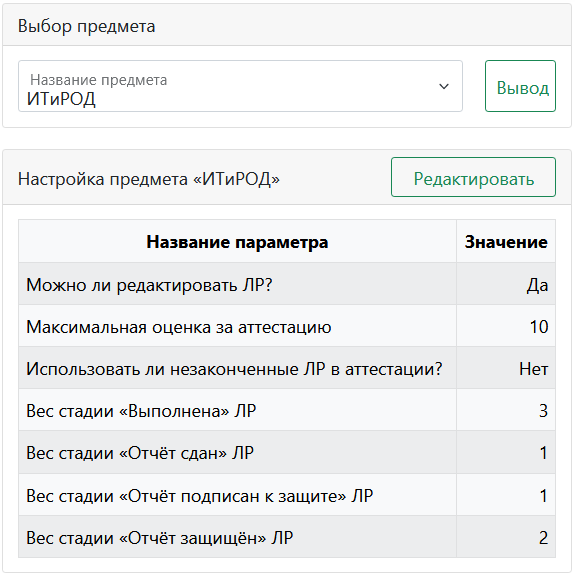


Рисунок 4.12 – Вид страницы для просмотра предметов

Сейчас и далее страницы для создания, изменения и удаления записей в различных таблицах будут опущены, ввиду их схожести с аналогичными страницами у администратора.

На рисунке 4.13 показана страница для настройки расписания.

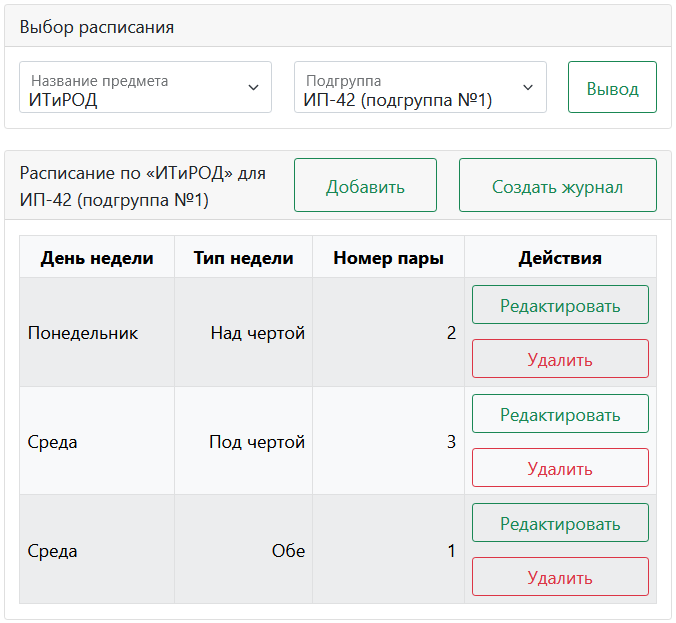


Рисунок 4.13 – Вид страницы для настройки расписания

На рисунке 4.14 показана страница для настройки лабораторных работ.

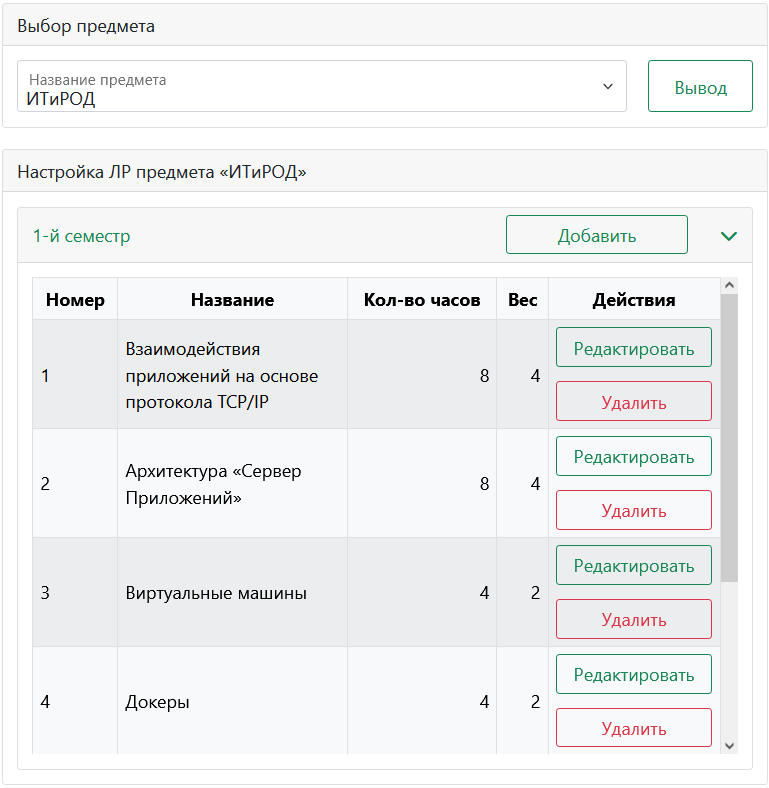


Рисунок 4.14 – Вид страницы для настройки лабораторных работ

На рисунке 4.15 показана страница для заполнения посещаемости.

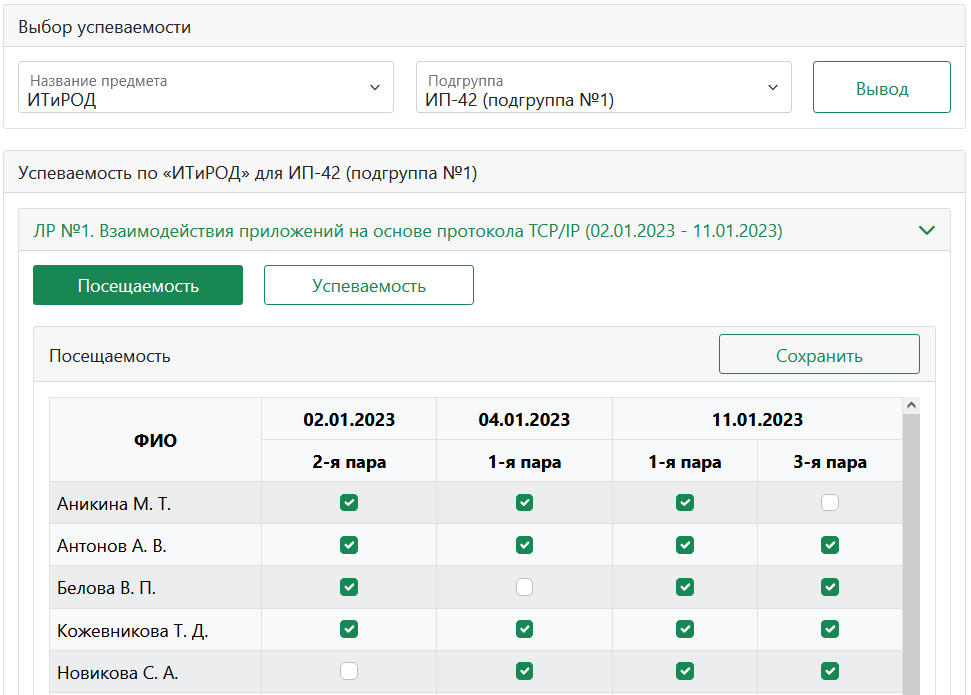


Рисунок 4.15 – Вид страницы для заполнения посещаемости

На рисунке 4.16 показана страница для заполнения успеваемости.

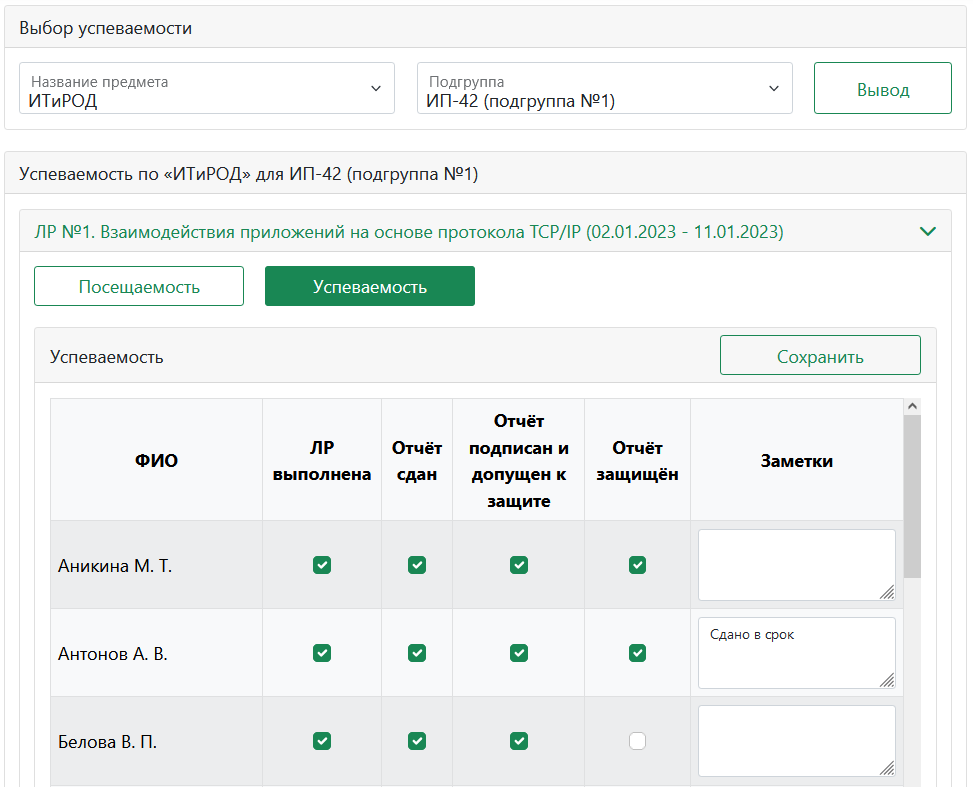


Рисунок 4.16 – Вид страницы для заполнения успеваемости

На рисунке 4.17 показана страница для формирования аттестаций.

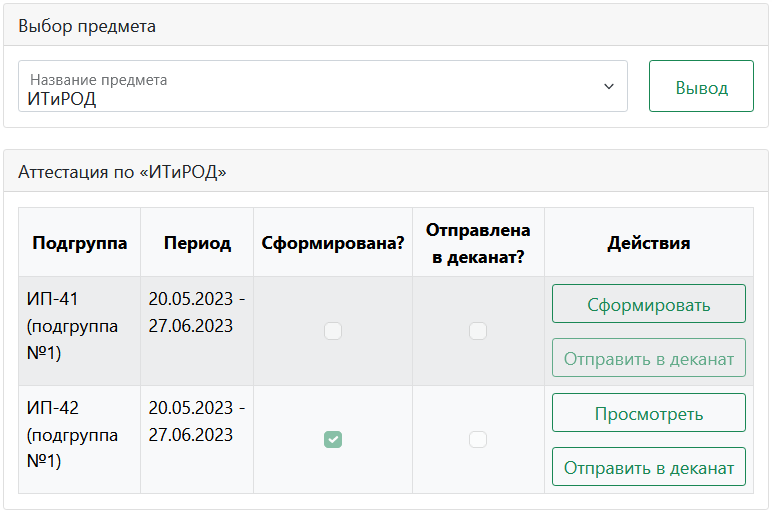


Рисунок 4.17 – Вид страницы для формирования аттестаций

При нажатии на кнопку «Сформировать» результаты аттестации вычисляются автоматически согласно посещаемости и успеваемости студентов данной подгруппы. После этого преподаватель может вручную изменить результаты аттестации, нажав на кнопку «Просмотр», или же отправить результаты в деканат, нажав на кнопку «Отправить в деканат». Вид страницы просмотра и изменения результатов аттестации представлен на рисунке 4.18.

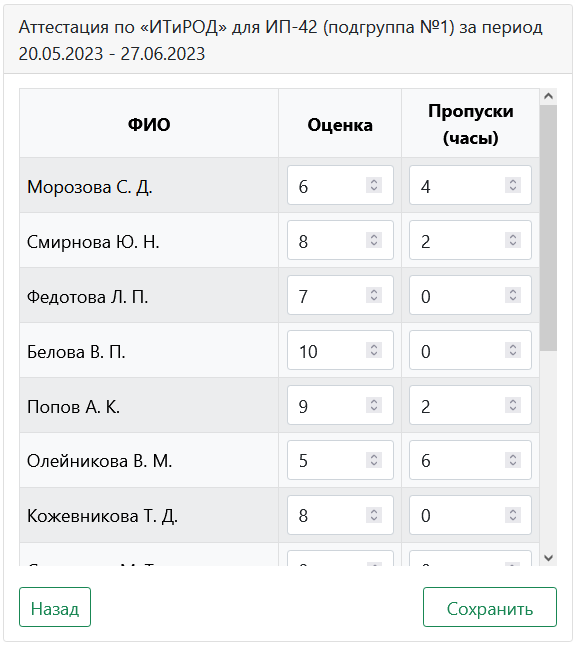


Рисунок 4.18 – Вид страницы для просмотра и изменения аттестации

Навигационная панель пользователя, вошедшего в систему в роли методиста деканата, принимает вид, указанный на рисунке 4.19.

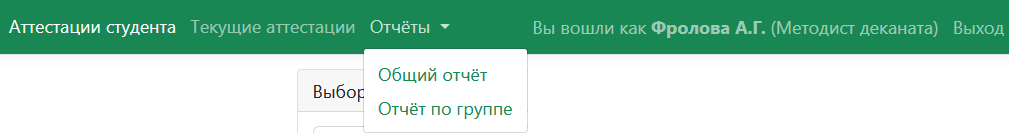


Рисунок 4.19 – Навигационная панель методиста деканата

На рисунке 4.20 показана страница для просмотра аттестации по студенту. Студент ищется в системе после ввода произвольной комбинации более трёх символов входящих в ФИО, номер зачётной книжки или группу.

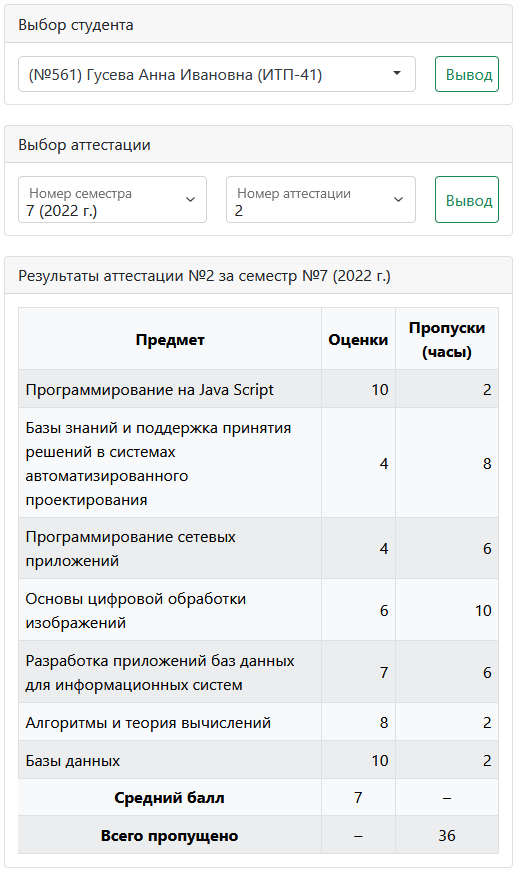


Рисунок 4.20 – Вид страницы для просмотра аттестации по студенту

На рисунке 4.21 показана страница для просмотра состояния текущей аттестации по группам. Если все предметы, которые проводятся у данной группы имеют отметку аттестации «Отправлена в деканат», то активируется кнопка «Сохранить в отчёты», которая переносит результаты в состояние отчётов, которые можно посмотреть на следующих страницах.

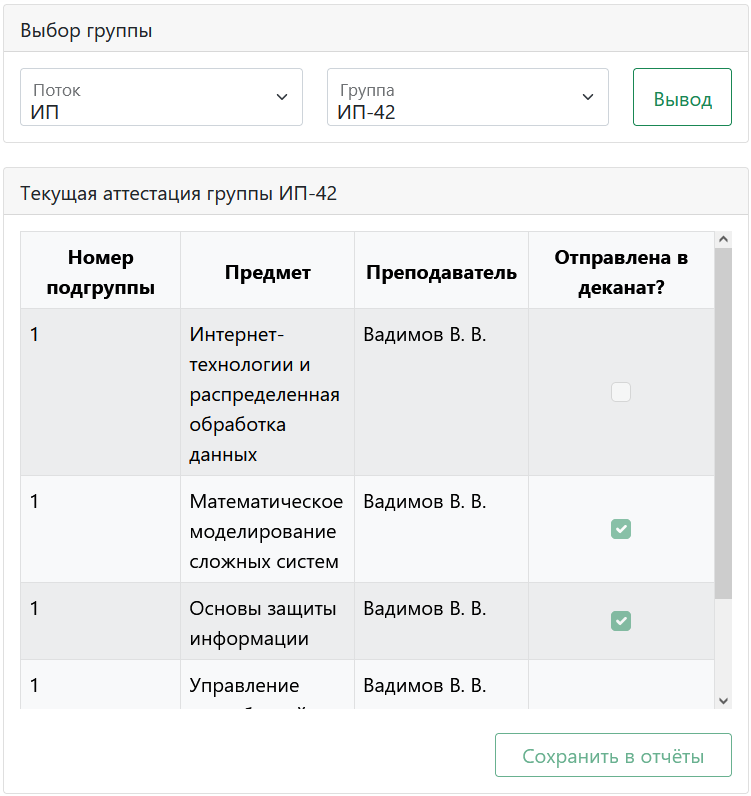


Рисунок 4.21 – Вид страницы для просмотра состояния текущей аттестации

На рисунке 4.22 показана страница для просмотра общего отчёта по аттестации за выбранный период. Данные отчёты можно также сохранить в *Excel*-документ для удобства дальнейшего использования.

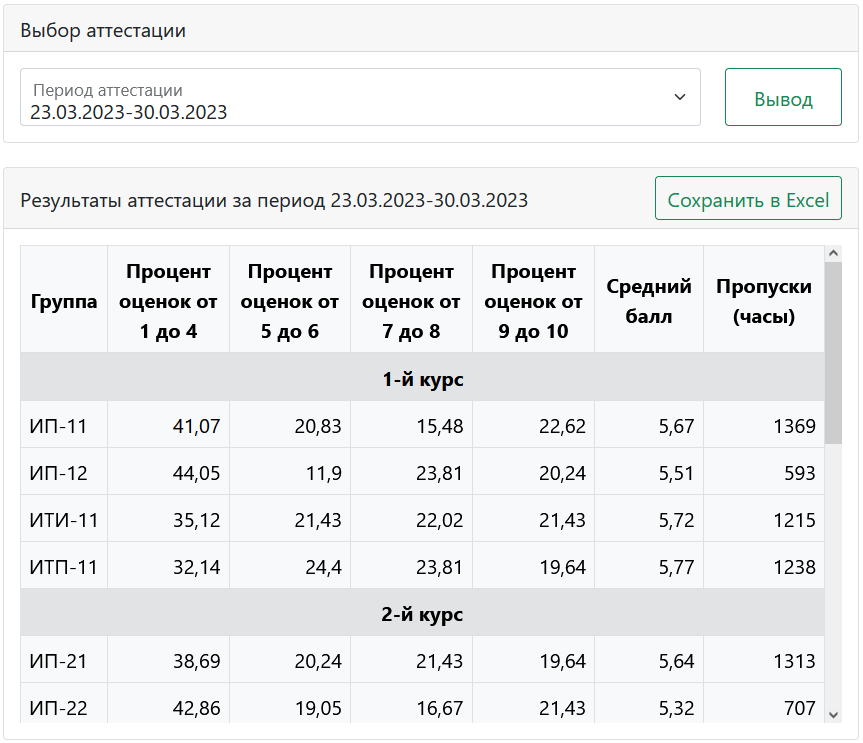


Рисунок 4.22 – Вид страницы для просмотра общего отчёта по аттестации

На рисунке 4.23 показана страница для просмотра отчёта группе по аттестации за выбранный период. Данные отчёты можно также сохранить в *Excel*-документ для удобства дальнейшего использования.

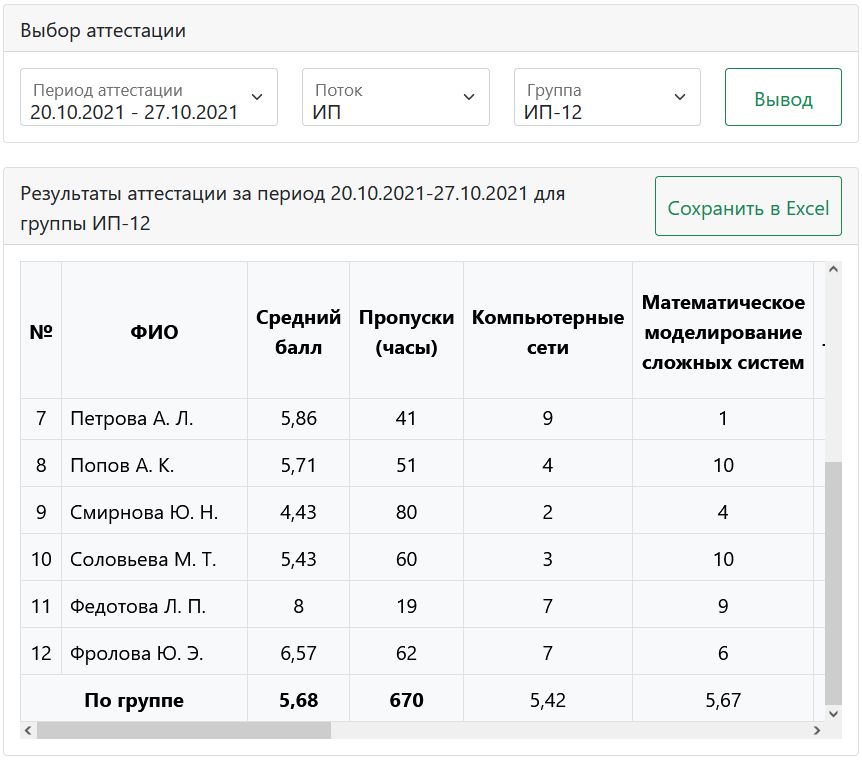


Рисунок 4.23 – Вид страницы для просмотра отчёта по группе по аттестации

Навигационная панель пользователя, вошедшего в систему в роли заведующего кафедрой, принимает вид, указанный на рисунке 4.24.

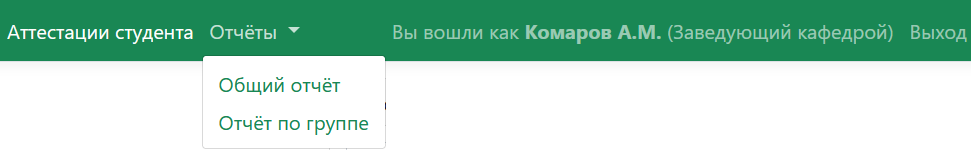


Рисунок 4.24 – Навигационная панель заведующего кафедрой

Заведующий кафедрой имеет функционал, схожий с функционалом методиста деканата, за исключением того, что у заведующего кафедрой нет доступа к текущим аттестациям. Помимо этого, заведующий кафедрой может просматривать результаты и отчёты по аттестациям только у подведомственных его кафедре групп. Из-за ранее указанной схожести функционала, описание страниц, доступных заведующему кафедрой, производиться не будет.

Как итог, можно отметить, что разработанное приложение полностью предоставляет сформулированный в постановке задачи функционал.

Руководство системного программиста приведено в приложении Б.

Руководство программиста приведено в приложении В.

Руководство пользователя приведено в приложении Г.

**4.2 Тестирование приложения**

После создания, программное обеспечение необходимо протестировать для того, чтобы выявить потенциальные ошибки и исправить их. В данном случае было выполнено три группы тестов: *unit*-тесты, *mock*-тесты и интеграционные тесты.

*Unit*-тестирование в данной дипломной работе было необходимо для того, чтобы проверить границы предметной области, т.е. проверить валидацию различных свойств у *DTO-*классов, которые указаны у них в виде атрибутов валидации. Для выполнения тестирования для начала были очерчены допустимые значения для каждого свойства. Значения были выбраны таким образом, чтобы не быть избыточными, т.е. не перекрывать друг друга. После установки допустимых значений для каждого поля, были созданы списки тестов. Для удобства использования эти списки были разделены на две группы: позитивные и негативные тесты. Для создания списков были использованы различные методики создания тестов, в их числе анализ классов эквивалентности и анализ граничных значений классов эквивалентностей, попарное и доменное тестирование.

После *unit*-тестирования было проведено *mock*-тестирование. Его целью является проверка классов бизнес-логики, в частности сервисов, изолированно от источников данных путём подмены стандартных реализаций интерфейсов на объекты-заглушки. Тестирование было разделено на два этапа: тестирование состояния и тестирование поведения. В первом этапе за счёт исходных данных было смоделировано нужное состояние тестируемого объекта и после этого было проверено, что нужное состояние достигнуто. В данном случае под нужным состоянием понимается некоторый заранее предопределённый результат возврата методов. Во втором этапе было проверено поведение тестируемого объекта, т.е. были вызваны определённые методы объекта с определёнными входными параметрами, а после этого с помощью механизма тестового фреймворка была выполнена проверка, что внутри эти методы вызывают методы *mock*-реализации интерфейса с необходимыми входными параметрами. Таким образом проверяется правильность преобразования *DTO-*классов в сущности и обратно.

После выполнения *mock*-тестирования, где в качестве источника данных передавался имитирующий реализацию интерфейса тестовый объект, необходимо провести тестирование интерфейса с реальным источником базы данных. Как и в случае с *mock*-тестированием интеграционное тестирование было разделено на два этапа: тестирование состояния и тестирование поведения. Отличие заключается лишь в том, что в данном случае данные, с которыми работали методы, зависели от тестовой базы данных и условий, связанных с нею.

**Список использованных источников**

* 1. Устав университета | Гомельский государственный технический университет имени П.О.Сухого. – Электрон. данные. – Режим доступа: https://www.gstu.by/atom/28850/. – Дата доступа: 19.04.2023.
  2. 1С. Решения для высшего образования. – Электрон. данные. – Режим доступа: https://v8.1c.ru/upload/iblock/773/7731f272ae1cef898f1c287616987b29.p-df. – Дата доступа: 19.04.2023.
  3. 1С:Университет ПРОФ - Возможности продукта. – Электрон. данные. – Режим доступа: https://solutions.1c.ru/catalog/university-prof/features/. – Дата доступа: 19.04.2023.
  4. Электронная образовательная среда Moodle. Инструкция пользователя. – Электрон. данные. – Режим доступа: http://www.nsmu.ru/workers/cit/sistema-elektronnogo-obucheniya/eos\_pps.pdf. – Дата доступа: 19.04.2023.
  5. Змеев, М. В. Дистанционное обучение в программной среде Moodle: от урока до курса (учебное пособие для учителей и преподавателей) / М. В. Змеев, Р. Р. Камалов, А. И. Макурин. – Глазов : АНО Центр НИОКР «Универсум», 2018. – 118 с.
  6. Работа в системе Moodle: руководство пользователя. – Электрон. данные. – Режим доступа: https://moodle.tma.uz/pluginfile.php/7204/mod\_forum/attachment/1/Пособие\_Moodle.pdf. – Дата доступа: 19.04.2023.
  7. Система «Галактика ERP». Учебный процесс. – Электрон. данные. – Режим доступа: https://galaktika-it.ru/wp-content/uploads/2020/08/uchebnyj-process.-rukovodstvo-polzovatelja.pdf. – Дата доступа: 19.04.2023.
  8. Система «Галактика ERP». Управление контингентом студентов. – Электрон. данные. – Режим доступа: https://spbguga.ru/files/Galaktika-Upravlenie\_kontingentom\_studentov.pdf. – Дата доступа: 19.04.2023.
  9. Система «Галактика ERP». Учёт успеваемости студентов. – Электрон. данные. – Режим доступа: https://spbguga.ru/files/Galaktika-Uchet\_uspevaemosti\_studentov.pdf. – Дата доступа: 19.04.2023.
  10. Корпорация «Галактика». Решения для высших учебных заведений. – Электрон. данные. – Режим доступа: https://galaktika.ru/docs/Galaktika\_VUZ.pdf. – Дата доступа: 19.04.2023.
  11. Документация по C#. Начало работы, руководства, справочные материалы. | Microsoft Docs. – Электрон. данные. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/. – Дата доступа: 19.04.2023.
  12. Общие сведения ASP.NET Core MVC | Microsoft Docs. – Электрон. данные. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview/. – Дата доступа: 19.04.2023.
  13. Чамберс, Д. ASP.NET Core. Разработка приложений / Д. Чамберс, Д. Пэкетт, С. Тиммс. – СПб. : Питер, 2018. – 464 с.
  14. Фримен, А. ASP.NET Core MVC 2 с примерами на C# для профессионалов / А. Фримен. – 7-е изд. – СПб. : ООО «Диалектика», 2019. – 1008 с.
  15. Документация по интегрированной среде разработки Visual Studio | Microsoft Docs. – Электрон. данные. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/ide/. – Дата доступа: 19.04.2023.
  16. Шилдс, У. SQL: быстрое погружение / У. Шилдс. – СПб. : Питер, 2022. – 224 с.
  17. Обзор Entity Framework Core — EF Core | Microsoft Docs. – Электрон. данные. – Режим доступа: https://docs.microsoft.com/ru-ru/ef/core/. – Дата доступа: 19.04.2023.
  18. Кожевников, Е. А. Расчёт экономической эффективности разработки программных продуктов: метод. указания по подготовке организационно-экономического раздела дипломных работ для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» дневной формы обучения / Е. А. Кожевников, Н. В. Ермалинская. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 68 с.
  19. Охрана труда: учебно-методическое пособие / В. Н. Босак [и др.]; под общ. ред. В. Н. Босака. – Горки : БГСХА, 2021. – 154 с.
  20. Челноков, А. А*.* Охрана труда: учеб. пособие для студентов технологических специальностей / А. А. Челноков. – Минск : БГТУ, 2006. – 294 с.